

Appendix – CODE

Content

Default layer	4
Main.....	4
Model layer (tweakmc.Model)	4
Bearing.....	4
Company.....	5
CompanyCatalog.....	8
Cylinder.....	9
CylinderPosition.....	10
Login	10
Measurement	12
Order	12
OrderCatalog	15
Owner	17
OwnerCatalog.....	21
SuperSearch.....	24
User.....	24
UserCatalog	25
Vehicle	26
VehicleCatalog	29
Workstation	32
WorkstationCatalog.....	33
Model/Measurement layer (tweakmc.model.Measurement)	34
Calculation	34
Letter	34
Measurement	35
Number.....	36
Model/Speradsheet (tweakmc.model.spreadsheet)	36
Spreadsheet.....	36
SpreadsheetCalalog.....	38
Model/Result layer (tweakmc.model.Result).....	39
Result	39
ResultCatalog.....	45
Model/result/whiteboard layer (tweakmc.model.Result.Whiteboard).....	47
BearingToleranceTable.....	47
ConnectingRod	48
CylinderToleranceTable.....	48
Head.....	49
ValveAjustmentTable	50
Whiteboard.....	53

WhiteboardCatalog	54
Dataaccess layer (tweakmc.dataaccess)	55
AutocompleteDAO	55
CompanyDAO	58
DAO.....	60
DBHandler.....	65
ObjectHandler	67
OrderDAO	70
OwnerDAO.....	77
ResultDAO.....	84
SuperSearchDAO	88
UserDAO	94
VehicleDAO.....	96
WhiteboardDAO	106
WorkstationDAO.....	110
Control layer (tweakmc.control)	112
FileReaderController	112
LoginController	114
ManageCompanyController	115
ManageOwnerController.....	116
ManageResultController	118
ManageSperadsheetController	119
ManageUserController	120
ManageVehicleController.....	120
ManageWhiteboardController	122
ManageWorkstationController	123
OrderController	124
Utility layer (tweakmc.utility)	126
CheckInput.....	126
FileHandlerUtility.....	131
FileWriterException	131
GUIHelpUtil.....	132
Mail.....	136
OrderPDF	138
StartUpTest.....	141
View layer (tweakmc.view)	144
AttachOwnerGUI	144
AttachVehidleGUI	150
AttachVehicleToOrderGUI	156
CompanyGUI.....	162
CreateSpreadsheetView	174
GUIDesign (unused).....	180
GUIFrame.....	190

JavaHelp.....	199
MailGUI.....	203
ManageSpreadsheetView.....	209
NameColumns	216
OrderGUI	221
OwnerGUI.....	250
SuperSearch.....	287
UploadMediaView	307
ValveadjustmentTableGUI.....	318
VehicleGUI	326
WSGUI.....	356
WaitFrame	366
WhiteboardView.....	369
WorkStationGUI.....	374
View/resultViewUtility (tweakmc.view.resultviewUtility)	390
ResultTreeUtility	390
Model/ spreadsheet (tweakmc.Model.spreadsheet)	393
Spreadsheet.....	393
SpreadsheetCatalog.....	394

Default layer

Main

```
1
2 import tweakmc.view.*;
3 /**
4  *
5  * @author TeaN TALC
6  * @version 1.0
7  */
8 public class Main
9 {
10     public static void main(String[] args)
11     {
12         GUIFrame.main(args);
13     } // End main method
14
15 } // End class main
```

Model layer (tweakmc.Model)

Bearing

```
1 package tweakmc.model;
2
3 /**
4  *
5  * @author NegoZiatoR
6  */
7 public class Bearing
8 {
9     // Instance variables
10     private int bearingNumber;
11
12     /**
13      * Constructor for Bearing
14      * @param bearingNumber on bearing
15      */
16     public Bearing(int bearingNumber)
17     {
18         this.bearingNumber = bearingNumber;
19     }
20
21     /**
22      * Return bearing number
23      * @return bearing number
24      */
25     public int getBearing()
26     {
27         return bearingNumber;
28     }
29
30 }
```

Company

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model;
7
8 /**
9  *
10 * @author TeaN TALC -ROMANTY
11 */
12 public class Company {
13
14     private String cvrNo;
15     private String name;
16     private String address;
17     private String phone;
18     private String email;
19     private String homePage;
20     private String bankInfo;
21 /**
22  * constructor for the company
23  * @param cvrNo for the company
24  * @param name of the company
25  * @param address of the company
26  * @param email for the company
27  * @param homepage for the company
28  * @param bankInfo for the company
29 */
30
31
32     public Company(String curNo, String name, String address,String phone,String email, String homePage, String
bankInfo)
33     {
34         this.cvrNo = curNo;
35         this.name = name;
36         this.address = address;
37         this.phone = phone;
38         this.email = email;
39         this.homePage = homePage;
40         this.bankInfo = bankInfo;
41     }
42
43
44 }
45 /**
46  * methord returns company cvrNo
47  * @return
48  */
49
50 public String getCvrNo()
51 {
52     return cvrNo;
53 }
54 /**
55  * Methord sets cvrno for the company
```

```

56  * @param cvrNo
57  */
58  public void setCvrNo(String cvrNo)
59  {
60
61  }
62  /**
63   * Method sets the company name
64   * @param name
65   */
66
67  public void setName(String name)
68
69  {
70      this.name = name;
71  }
72  /**
73   * Method returns company name
74   * @return
75   */
76  public String getName()
77
78  {
79      return name;
80  }
81  /**
82   * Method to set company address
83   * @param address
84   */
85  public void setAddress(String address)
86
87  {
88      this.address = address;
89  }
90  /**
91   * Method returns company address
92   * @return
93   */
94  public String getAddress()
95
96  {
97      return address;
98  }
99
100  /**
101   * Method sets company phone
102   * @param phone
103   */
104
105  public void setPhone(String phone)
106  {
107      this.phone = phone;
108  }
109  /**
110   * Method returns phone number of the company
111   * @return
112   */
113  public String getPhone()

```

```

114     {
115         return phone;
116     }
117     /**
118     * methord returns email
119     * @param email
120     */
121     public void setEmail(String email)
122     {
123         this.email = email;
124     }
125     /**
126     * Methord returns email
127     * @return
128     */
129     public String getEmail()
130     {
131         return email;
132     }
133     /**
134     * Methord sets homePage
135     * @param homePage
136     */
137     public void setHomePage(String homePage)
138     {
139         this.homePage = homePage;
140     }
141     /**
142     * @return
143     */
144     public String getHomePage()
145     {
146         return homePage;
147     }
148     /**
149     * Methord set bankInfo of the company
150     * @param bankInfo
151     */
152     public void setBankInfo(String bankInfo)
153     {
154         this.bankInfo = bankInfo;
155     }
156     /**
157     * return bankInfo of the company
158     * @return
159     */
160     public String getBankInfo()
161     {
162         return bankInfo;
163     }
164     /**
165     * Return all information

```

```

172  * @return
173  */
174  @Override
175  public String toString()
176  {
177      return cvrNo + " " + name + " " + address + " " + phone + " " + email + " " + homePage + " " + bankInfo + "
";
178  }
179  //End of company Class
180 }

```

CompanyCatalog

```

1  /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6  package tweakmc.model;
7
8  import java.io.Serializable;
9  import tweakmc.dataaccess.CompanyDAO;
10
11  /**
12   *
13   * @author Romanty / TeaN TALC
14   */
15  public class CompanyCatalog implements Serializable
16  {
17
18      private static CompanyCatalog instance;
19
20
21
22
23      private CompanyCatalog()
24      {
25
26
27      }
28
29      /**
30       * single turn pattern
31       * @return instance of company catalog
32       */
33      public static CompanyCatalog getInstance()
34      {
35
36
37          if (instance==null)
38          {
39              instance=new CompanyCatalog ();
40          }
41
42
43
44          return instance;

```



```

45
46 }
47 /**
48  *
49  * @param curNo
50  * @param name
51  * @param address
52  * @param phone
53  * @param email
54  * @param homePage
55  * @param bankInfo
56  * @return
57  */
58
59 public boolean updateCompany (String curNo, String name, String address,String phone,String email, String
homePage, String bankInfo)
60
61 {
62     return CompanyDAO.getInstance().createCompany(curNo, name, address, phone, email, homePage, bankInfo);
63
64 }
65
66 /**
67  *
68  * @param curNo
69  * @param name
70  * @param Address
71  * @param phone
72  * @param email
73  * @param homePage
74  * @param bankInformation
75  * @return
76  */
77
78 public boolean editCompany(String curNo, String name, String Address, String phone, String email, String
homePage, String bankInformation)
79 {
80     return CompanyDAO.getInstance().editCompany(curNo, name, Address, phone, email, homePage,
bankInformation);
81 } // End of edit Company.
82 }

```

Cylinder

```

1
2 package tweakmc.model;
3
4 /**
5  *
6  * @author NegoZiatoR
7  */
8 public class Cylinder
9 {
10     // Instance variables
11     private int cylinderNumber;
12
13     /**

```

```

14  * Constructor for Cylinder
15  * @param cylinderNumber number of cylinders
16  */
17  public Cylinder(int cylinderNumber)
18  {
19      this.cylinderNumber = cylinderNumber;
20  }
21
22  /**
23   * Return number of cylinder
24   * @return number of cylinder
25   */
26  public int getCylinders()
27  {
28      return cylinderNumber;
29  }
30
31 } // End of cylinder Class

```

CylinderPosition

```

1  package tweakmc.model;
2
3  /**
4   *
5   * @author NegoZiatoR
6   */
7  public class CylinderPosition
8  {
9      // Instance variables
10     private String cPositionName;
11
12     /**
13      * Constructor for cylinderposition
14      * @param cylinderPosition on cylinder
15      */
16     public CylinderPosition(String cPositionName)
17     {
18         this.cPositionName = cPositionName;
19     } // End of CylinderPosition
20
21     /**
22      * Return cylinderposition
23      * @return cylinders position
24      */
25     public String getCylinderPosition()
26     {
27         return cPositionName;
28     } // End of getCylinderPositin
29
30 } // End of CylinderPosition class

```

Login

```

1  package tweakmc.model;
2

```

```

3 import java.util.HashMap;
4
5 /**
6  *
7  * @author Nyvang
8  * @Edited by NegoZiatoR
9  */
10 public class Login
11 {
12     // Instance variables
13     private HashMap<String, String> loginContainer;
14     private static Login instance;
15
16     // Private constructor for Login
17     private Login()
18     {
19         loginContainer = new HashMap<String, String>();
20     }
21
22     /**
23      * Singleton method for Login
24      * @return ONE instance of Login
25      */
26     public static Login getInstance()
27     {
28         if(instance == null)
29         {
30             instance = new Login();
31         }
32         return instance;
33     }
34
35     /**
36      * Add a new login initial
37      * @param initials initial to add
38      * @param name name on the new login
39      */
40     public void addLogin(String initials, String name)
41     {
42         loginContainer.put(initials, name);
43     }
44
45     /**
46      * Return all login initials and names in a HashMap
47      * @return HashMap with all names and initials
48      */
49     public HashMap<String, String> getLoginContainer()
50     {
51         return loginContainer;
52     }
53
54     /**
55      * Return name on given initial
56      * @param initials initial to check
57      * @return name for initial if present
58      */
59     public String getName(String initials)
60     {

```

```

61     String name = loginContainer.get(initials);
62     return name;
63 }
64
65 /**
66  * If this is the method choosen (and not the database) to hold the logins
67  * for the system, the following methods are missing:
68  * - remove
69  * - create
70  * - find (login)
71  * - find (name)
72  * etc
73  */
74
75 }

```

Measurement

```

1
2 package tweakmc.model;
3
4 /**
5  *
6  * @author NegoZiatoR
7  */
8 public class MeasurementType
9 {
10     // Instance variables
11     private String mTypeName;
12
13     /**
14      * Constructor for MeasurementType
15      * @param name on MeasurementType
16      */
17     public MeasurementType(String mTypeName)
18     {
19         this.mTypeName = mTypeName;
20     } // End of MeasurementType constructor
21
22     /**
23      * Return measurement data type
24      * @return measurement data type
25      */
26     public String getMeasurementDataType()
27     {
28         return mTypeName;
29     } // End of GetMeasurementDataType
30
31 } // End of MeasurementType

```

Order

```

1
2 package tweakmc.model;
3

```

```

4 /**
5 *
6 * @author Nyvang
7 */
8 public class Order {
9
10    //(orderId, orderType, descriptionOrder, descriptionSpareparts, startDate, expEndDate, expPrice, finishPrice)
11    private String orderId;
12    private String orderType;
13    private String descriptionOrder;
14    private String descriptionSpareparts;
15    private int startDate;
16    private int expEndDate;
17    private String expPrice;
18    private String finishPrice;
19
20    /**
21     * Constructor for the Order class in the model layer, requires all fields
22     * @param orderId
23     * @param orderType
24     * @param descriptionOrder
25     * @param descriptionSpareparts
26     * @param startDate
27     * @param expEndDate
28     * @param expPrice
29     * @param finishPrice
30     */
31    public Order(String orderId, String orderType, String descriptionOrder, String descriptionSpareparts, int
startDate, int expEndDate, String expPrice, String finishPrice) {
32        this.orderId = orderId;
33        this.orderType = orderType;
34        this.descriptionOrder = descriptionOrder;
35        this.descriptionSpareparts = descriptionSpareparts;
36        this.startDate = startDate;
37        this.expEndDate = expEndDate;
38        this.expPrice = expPrice;
39        this.finishPrice = finishPrice;
40    } // end constructor
41
42
43    /**
44     * Get-method for orderId
45     * @return orderId
46     */
47    public String getOrderId() {
48        return orderId;
49    }
50
51    /**
52     * Set-method for orderId
53     * @param orderId
54     */
55    public void setOrderId(String orderId) {
56        this.orderId = orderId;
57    }
58
59    /**
60     * Get-method for orderType

```

```

61  * @return orderType
62  */
63  public String getOrderType() {
64      return orderType;
65  }
66
67  /**
68   * Set-method for orderType
69   * @param orderType
70   */
71  public void setOrderType(String orderType) {
72      this.orderType = orderType;
73  }
74
75  /**
76   * Get-method for descriptionOrder
77   * @return descriptionOrder
78   */
79  public String getDescriptionOrder() {
80      return descriptionOrder;
81  }
82
83  /**
84   * Set-method for descriptionOrder
85   * @param descriptionOrder
86   */
87  public void setDescriptionOrder(String descriptionOrder) {
88      this.descriptionOrder = descriptionOrder;
89  }
90
91  /**
92   * Get-method for descriptionSpareparts
93   * @return descriptionSpareparts
94   */
95  public String getDescriptionSpareparts() {
96      return descriptionSpareparts;
97  }
98
99  /**
100   * Set-method for descriptionSpareparts
101   * @param descriptionSpareparts
102   */
103  public void setDescriptionSpareparts(String descriptionSpareparts) {
104      this.descriptionSpareparts = descriptionSpareparts;
105  }
106
107  /**
108   * Get-method for startDate
109   * @return startDate
110   */
111  public int getStartDate() {
112      return startDate;
113  }
114
115  /**
116   * Set-method for startDate
117   * @param startDate
118   */

```

```

119 public void setStartDate(int startDate) {
120     this.startDate = startDate;
121 }
122
123 /**
124  * Get-method for expEndDate
125  * @return expEndDate
126  */
127 public int getExpEndDate() {
128     return expEndDate;
129 }
130
131 /**
132  * Set-method for extEndDate
133  * @param expEndDate
134  */
135 public void setExpEndDate(int expEndDate) {
136     this.expEndDate = expEndDate;
137 }
138
139 /**
140  * Get-method for expPrice
141  * @return expPrice
142  */
143 public String getExpPrice() {
144     return expPrice;
145 }
146
147 /**
148  * Set-method for expPrice
149  * @param expPrice
150  */
151 public void setExpPrice(String expPrice) {
152     this.expPrice = expPrice;
153 }
154
155 /**
156  * Get-method for finishPrice
157  * @return finishPrice
158  */
159 public String getFinishPrice() {
160     return finishPrice;
161 }
162
163 /**
164  * Set-method for finishPrice
165  * @param finishPrice
166  */
167 public void setFinishPrice(String finishPrice) {
168     this.finishPrice = finishPrice;
169 }
170 } // end class Order

```

OrderCatalog

```

1 package tweakmc.model;
2

```

```

3 import java.io.Serializable;
4 import java.util.ArrayList;
5 import tweakmc.dataaccess.OrderDAO;
6
7 /**
8  *
9  * @author Nyvang
10 */
11 public class OrderCatalog implements Serializable {
12
13     private static OrderCatalog instance;
14     private ArrayList<Order> orderList;
15
16     private OrderCatalog()
17     {
18         orderList = new ArrayList<Order>();
19     }
20
21     public static OrderCatalog getInstance()
22     {
23         if(instance == null)
24         {
25             instance = new OrderCatalog();
26         }
27         return instance;
28     }
29
30     /**
31      * Add-method
32      * @param orderType
33      * @param orderDescription
34      * @param descriptionSpareparts
35      * @param startDate
36      * @param expEndDate
37      * @param expPrice
38      * @param finishPrice
39      * @return
40      */
41     public boolean addOrder(String orderType, String orderDescription, String descriptionSpareparts, int startDate, int
expEndDate, String expPrice, String finishPrice)
42     {
43         return OrderDAO.getInstance().createOrder(orderType, descriptionSpareparts, descriptionSpareparts, startDate,
expEndDate, expPrice, finishPrice );
44     }
45 }
46
47     public boolean updateOrder(String orderID, String orderType, String orderDescription, String
descriptionSpareparts, int startDate, int expEndDate, String expPrice, String finishPrice)
48     {
49         return OrderDAO.getInstance().updateOrder(orderID, orderType, descriptionSpareparts, descriptionSpareparts,
startDate, expEndDate, expPrice, finishPrice);
50     }
51
52     /**
53      * Search-method
54      * @param <T>
55      * @param searchCriteria
56      * @return

```



```

57  */
58  public <T> ArrayList<Order> searchOrder (T searchCriteria)
59  {
60      return OrderDAO.getInstance().searchOrder(searchCriteria);
61  }
62
63  public ArrayList<Order> generateTodo()
64  {
65      return OrderDAO.getInstance().generateTodo();
66  }
67
68  public boolean attachVehicle(String vehicleID, String orderID)
69  {
70      return OrderDAO.getInstance().attachVehicle(vehicleID, orderID);
71  } // end attachVehicle method
72
73 } //end class OrderCatalog

```

Owner

```

1  package tweakmc.model;
2
3  /**
4   * @author Nyvang / TeaN TALC
5   * @edited by Lars
6   * @date Oktober – December 2010
7   */
8
9  public class Owner implements Comparable<Owner>
10 {
11     private String ownerID;
12     public String firstName;
13     private String lastName;
14     public String address;
15     private String address2;
16     private String zip;
17     private String city;
18     private String country;
19     public String phone;
20     public String email;
21     // public Customer customerType;
22
23     /**
24      * Constructor of the Owner class
25      * @param String ownerID, String firstName, String lastName, String address, int id, String phone, String email
26      *
27      */
28
29     public Owner(String ownerID, String firstName, String lastName, String address, String address2, String zip,
30 String city, String country, String phone, String email)
31     {
32         this.ownerID = ownerID;
33         this.firstName = firstName;
34         this.lastName = lastName;
35         this.address = address;
36         this.address2 = address2;

```

```

36     this.zip = zip;
37     this.city = city;
38     this.country = country;
39     this.phone = phone;
40     this.email = email;
41 }// End of Owner constructor
42
43 /**
44  * Overridden method
45  * This method is a part of the implementing of comparable.
46  * Object Owner is sorted by their IDs
47  * @param o
48  * @return Integer
49  */
50 @Override
51 public int compareTo(Owner o)
52 {
53     return this.ownerID.compareTo(o.getID());
54 }// End of compareTo method
55
56 /**
57  * Return owners ID
58  * @return owners ID
59  */
60
61 public String getID()
62 {
63     return ownerID;
64 }// End of getID method
65
66 /**
67  * Set owners ID
68  * @param ownerID owners new ID
69  */
70 public void setID(String ownerID)
71 {
72     this.ownerID = ownerID;
73 }// End of setID method
74
75 /**
76  * Get owners first name
77  * @return owners first name
78  */
79 public String getFirstName()
80 {
81     return firstName;
82 }// End of getFirstName method
83
84 /**
85  * Set the owners first name
86  * @param firstName owners new first name
87  */
88 public void setFirstName(String firstName)
89 {
90     this.firstName = firstName;
91 }// End of setFirstName method
92
93 /**

```

```

94  * Return owners last name
95  * @return owners last name
96  */
97  public String getLastName()
98  {
99      return lastName;
100 }// End of getLastName method
101
102 /**
103  * Set owners last name
104  * @param lastName owners new last name
105  */
106 public void setLastName(String lastName)
107 {
108     this.lastName = lastName;
109 }// End of setLastName method
110
111 /**
112  * Return owners address1
113  * @return owners address1
114  */
115 public String getAddress()
116 {
117     return address;
118 }// End of getAddress method
119
120 /**
121  * Set owners address1
122  * @param address1 owners new address1
123  */
124 public void setAddress(String address)
125 {
126     this.address = address;
127 }// End of setAddress method
128
129 /**
130  *Return owners address2
131  * @return owners address2
132  */
133 public String getAddress2()
134 {
135     return address2;
136 }// End of getAddress2 method
137
138 /**
139  * Set owners address2
140  * @param address2 owners new address2
141  */
142 public void setAddress2(String address2)
143 {
144     this.address2 = address2;
145 }// End of setAddress2 method
146
147 /**
148  * Return owners zipcode
149  * @return owners zipcode
150  */
151 public String getZip()

```

```

152 {
153     return zip;
154 }// End of getZip method
155
156 /**
157  * Set owners zipcode
158  * @param zip owners new zip code
159  */
160 public void setZip(String zip)
161 {
162     this.zip = zip;
163 }// End of setZip method
164
165 /**
166  * Return owners city
167  * @return owners city
168  */
169 public String getCity()
170 {
171     return city;
172 }// End of getCity method
173
174 /**
175  * Set owners city
176  * @param city owners new city
177  */
178 public void setCity(String city)
179 {
180     this.city = city;
181 }// End of setCity method
182
183 /**
184  * Return owners country
185  * @return country on owner
186  */
187 public String getCountry()
188 {
189     return country;
190 }// End of getCountry method
191
192 /**
193  * Set the owners country
194  * @param country new country on owner
195  */
196 public void setCountry(String country)
197 {
198     this.country = country;
199 }// End of setCountry method
200
201
202 /**
203  * Return owners phonenummer
204  * @return owners phonenummer
205  */
206 public String getPhone()
207 {
208     return phone;
209 }// End of getPhone method

```

```

210
211 /**
212  * Set owners phone number
213  * @param phone owners new phone number
214  */
215 public void setPhone(String phone)
216 {
217     this.phone = phone;
218 }// End of setPhone method
219
220 /**
221  * Return owners email
222  * @return owners email
223  */
224 public String getEmail()
225 {
226     return email;
227 }// End of getEmail method
228
229 /**
230  * Set owners email
231  * @param email owners new email
232  */
233 public void setEmail(String email)
234 {
235     this.email = email;
236 }// End of set Email method
237
238 @Override
239 public String toString()
240 {
241     return "(" + ownerID + ") " + firstName + " " + lastName + "\n" +
242         address + "\n" + address2 + "\n" + zip + " " + city + "\n" + country +
243         "\n" + phone + "\n" + email;
244 }// End of ToString Method
245 }// End Owner-class

```

OwnerCatalog

```

1 package tweakmc.model;
2
3 import tweakmc.dataaccess.OwnerDAO;
4 import java.io.Serializable;
5 import java.util.ArrayList;
6 import java.util.List;
7
8
9 /**
10  * @author Nyvang / TeaN TALC
11  * @editedby Tvup
12  * @date October – December 2010
13  */
14
15 public class OwnerCatalog implements Serializable
16 {
17
18     private ArrayList<Owner> ownerList;

```

```

19 private static OwnerCatalog instance;
20
21 /**
22  * Constructor of the OwnerCatalog, private because of the Singleton pattern
23  */
24
25 private OwnerCatalog()
26 {
27     ownerList = new ArrayList<Owner>();
28 } // End of OwnerCatalog constructor
29
30 /**
31  * Singleton pattern
32  * @return instance of OwnerCatalog
33  */
34 public static OwnerCatalog getInstance()
35 {
36     if(instance == null)
37     {
38         instance = new OwnerCatalog();
39     }
40     return instance;
41 } // End of OwnerCatalog getInstance method
42
43 /**
44  * Get-method for ownerList
45  * @return ownerList
46  */
47
48 public ArrayList<Owner> getOwnerCatalog()
49 {
50     return ownerList;
51 } // End of ArrayList get OwnerCatalog method
52
53 /**
54  *
55  * @param ownerList
56  */
57 public void setOwnerList(ArrayList<Owner> ownerList)
58 {
59     this.ownerList = ownerList;
60 }
61
62 public void addOwner(Owner newOwner)
63 {
64     ownerList.add(newOwner);
65 } // End of AddOwner method
66
67 /**
68  * Create a new Owner
69  * @param firstName on owner
70  * @param lastName on owner
71  * @param address on owner
72  * @param address2 on owner
73  * @param zip on city
74  * @param city of owner
75  * @param phone for owner
76  * @param email for owner

```

```

77  * @return true if created else false
78  */
79  public boolean makeNewOwner(String firstName, String lastName, String address, String address2, String zip,
String city, String country, String phone, String email)
80  {
81      return OwnerDAO.getInstance().createActOwner(firstName, lastName, address, address2, zip, city, country,
phone, email);
82  } //End of MakeOwner method
83
84  /**
85   * Update data about an owner
86   * @param ownerID
87   * @param firstName
88   * @param lastName
89   * @param address
90   * @param address2
91   * @param zip
92   * @param city
93   * @param phone
94   * @param email
95   * @return boolean
96   */
97  public boolean updateOwner(String ownerID, String firstName, String lastName, String address, String address2,
String zip, String country, String city, String phone, String email)
98  {
99      return OwnerDAO.getInstance().editActOwner(ownerID, firstName, lastName, address, address2, zip, city,
country, phone, email);
100 } //End of updateOwner method
101
102 /**
103  * Search for an owner
104  * @param <T>
105  * @param searchCriteria
106  * @return List<Owner>
107  */
108 public <T> List<Owner> searchOwner (T searchCriteria)
109 {
110     return OwnerDAO.getInstance().searchActOwner(searchCriteria);
111 } //End of searchOwner method
112
113 /**
114  * Search for an owner and the vehicle he owns
115  * @param ownerID owner id for search
116  * @return List with vehicles the found owner owns
117  */
118 public List<Vehicle> searchOwnerWithVehicle(String ownerID)
119 {
120     return OwnerDAO.getInstance().searchActOwnerWithVehicles(ownerID);
121 } // End of searchOwnerWithVehicle method
122
123 /**
124  * Delete an owner
125  * @param ownerID on owner
126  * @return true if deleted else false
127  */
128 public boolean deleteOwner(String ownerID)
129 {
130     return OwnerDAO.getInstance().deleteActOwner(ownerID);

```

```

131 }//End of deleteOwner method
132
133 public boolean[] getActiveNess()
134 {
135     return OwnerDAO.getInstance().getActiveNess();
136 }//End of deleteOwner Method
137
138 }//End of OwnerCatalog class

```

SuperSearch

User

```

1 package tweakmc.model;
2
3 /**
4  *
5  * @author romanty
6  *///variables of Uers class(initials,and name)
7 public class User
8 {
9
10     private String initials;
11     private String name;
12
13     /**
14      * constructor for the User
15      * @param initials constructor
16      * @param name constructor
17      */
18     public User(String initials,String name)
19     {
20         this.initials = initials;
21         this.name = name;
22     }
23     /**
24      * method to set initials
25      * @param initials
26      */
27
28     public void setInitials(String initials)
29     {
30         this.initials = initials;
31     }
32     /**
33      * method to get initials
34      * @return
35      */
36
37     public String getInitials()
38     {
39         return initials;
40     }
41     /**
42      * method to set name

```



```

43  * @param name
44  */
45
46  public void setName( String name)
47  {
48      this.name=name;
49  }
50  /**
51   * method to get name
52   * @return
53   */
54
55  public String getName()
56  {
57      return name;
58  }
59  /**
60   * return all information
61   * @return
62   */
63
64  @Override
65  public String toString()
66  {
67      return initials + " " + name + " ";
68  }
69 }

```

UserCatalog

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package tweakmc.model;
7
8  import java.io.Serializable;
9  import tweakmc.dataaccess.UserDAO;
10
11  /**
12   * implementation of UserCatalog
13   * @author ADAMZ
14   */
15  public class UserCatalog implements Serializable
16  {
17      private static UserCatalog instance;
18      /**
19       * Constructor for the UserCatalog and it is private cause it is a single turn
20       */
21
22      private UserCatalog()
23      {
24
25      }
26      /**
27       * single turn pattern

```

```

28  * @return instance of UserContainer
29  */
30  public static UserCatalog getInstance()
31  {
32      if (instance == null)
33      {
34          instance = new UserCatalog();
35      }
36      return instance;
37  }
38  }
39  /**
40   * edit method called from the DAO
41   * @param initials
42   * @param name
43   * @return
44   */
45
46  public boolean editUser(String initials, String name)
47  {
48      return UserDao.getInstance().editUser(initials, name);
49  } //end of edit User
50
51 }

```

Vehicle

```

1  package tweakmc.model;
2
3  import java.io.Serializable;
4
5  /**
6   *
7   * @author TeaN TALC -NegoZiatoR
8   * @version 1.0
9   */
10 public class Vehicle implements Serializable
11 {
12     // Instance variables
13     private String vehicleID;
14     private String vType;
15     private String brand;
16     private String model;
17     private int vYear;
18     private String chassisNumber;
19     private String licensPlate;
20
21
22
23     /**
24      * Constructor for vehicle
25      * @param vType type of vehicle
26      * @param model model on vehicle
27      * @param vYear vYear on vehicle
28      * @param chassisNumber chassisNumber on vehicle
29      * @param licensPlate licensPlate on vehicle
30      */

```

```

31  public Vehicle(String vehicleID, String vType, String brand, String model, int vYear, String chassisNumber,
String licensPlate)
32  {
33      this.vehicleID = vehicleID;
34      this.brand = brand;
35      this.vType = vType;
36      this.model = model;
37      this.vYear = vYear;
38      this.chassisNumber = chassisNumber;
39      this.licensPlate = licensPlate;
40
41  }
42
43  /**
44   * Return vehicles id, set by system
45   * @return vehicleID
46   */
47  public String getVehicleID()
48  {
49      return vehicleID;
50  }
51
52  /**
53   * Return vehicles type
54   * @return vehicles type
55   */
56  public String getvType()
57  {
58      return vType;
59  }
60
61  /**
62   * Set vehicles type
63   * @param type vehicles new type
64   */
65  public void setvType(String vType)
66  {
67      this.vType = vType;
68  }
69
70  /**
71   * Return vehicles brand
72   * @return vehicles brand
73   */
74  public String getBrand()
75  {
76      return brand;
77  }
78
79  /**
80   * Set the vehicles brand
81   * @param brand vehicles new brand
82   */
83  public void setBrand(String brand)
84  {
85      this.brand = brand;
86  }
87

```

```
88  /**
89   * Return vehicles model
90   * @return vehicles model
91   */
92  public String getModel()
93  {
94      return model;
95  }
96
97  /**
98   * Set Vehicles model
99   * @param model vehicles new model
100   */
101  public void setModel(String model)
102  {
103      this.model = model;
104  }
105
106  /**
107   * Return vehicles year
108   * @return vehicles year
109   */
110  public int getvYear()
111  {
112      return vYear;
113  }
114
115  /**
116   * Set vehicles year
117   * @param year vehicles new year
118   */
119  public void setvYear(int vYear)
120  {
121      this.vYear = vYear;
122  }
123
124  /**
125   * Return vehicles chassis number
126   * @return vehicles chassis number
127   */
128  public String getChassisNumber()
129  {
130      return chassisNumber;
131  }
132
133  /**
134   * Set vehicles chassis number
135   * @param chassisNumber vehicles new chassisnumber
136   */
137  public void setChassisNumber(String chassisNumber)
138  {
139      this.chassisNumber = chassisNumber;
140  }
141
142  /**
143   * Return the vehicles licensplate
144   * @return vehicles licensplate
145   */
```

```

146 public String getLicensPlate()
147 {
148     return licensPlate;
149 }
150
151 /**
152  * Set vehicles licens plate
153  * @param licensPlate vehicles new licens plate
154  */
155 public void setLicensPlate(String licensPlate)
156 {
157     this.licensPlate = licensPlate;
158 }
159
160 /**
161  * Return all information
162  * @return all information
163  */
164 @Override
165 public String toString()
166 {
167     return vehicleID + " " + vType + " " + brand + " " + model + " " + vYear + " " + chassisNumber + " " +
licensPlate + " ";
168 }
169 // Class Vehicle end
170 }

```

VehicleCatalog

```

1
2
3 package tweakmc.model;
4
5 import tweakmc.dataaccess.VehicleDAO;
6
7 import java.io.Serializable;
8 import java.util.List;
9 import tweakmc.dataaccess.SuperSearchDAO;
10
11
12 /**
13  * Send and receive orders from/to DB about Vehicle
14  * @version 191010
15  * @author clausPallisgaardBeck
16  */
17 public class VehicleCatalog implements Serializable
18 {
19
20     private static VehicleCatalog instance;
21
22     /**
23      * Constructor of the VehicleCatalog, private because of the Singleton pattern
24      */
25
26     private VehicleCatalog()
27     {
28

```

```

29 }
30 /**
31  * Singleton pattern
32  * @return instance of OwnerContainer
33  */
34 public static VehicleCatalog getInstance()
35 {
36     if(instance == null)
37     {
38         instance = new VehicleCatalog();
39     }
40     return instance;
41 }
42
43 /**
44  * Search for a vehicle by the given criterias
45  *
46  * @param searchCriteriaString if its text
47  * @param searchCriteriaInt if its an int
48  * @return Array with the founds or null
49  */
50 public <T> List<Vehicle> searchVehicle(T searchCriteria)
51 {
52     return VehicleDAO.getInstance().searchActVehicle(searchCriteria);
53 } //end method searchVehicle
54
55 /**
56  * Search for a Vehicle and the owners
57  * @param vehicleID vehicles id for search
58  * @return List with owners
59  */
60 public List<Owner> superSearchVehicleWithOwner(String vehicleID)
61 {
62     return SuperSearchDAO.getInstance().searchActVehicleWithOwner(vehicleID);
63 }
64
65 public List<Owner> searchVehicleWithOwner(String vehicleID)
66 {
67     return VehicleDAO.getInstance().searchActOwnerWithVehicles(vehicleID);
68 }
69
70 /**
71  * Create a new vehicle
72  * @param vType of vehicle
73  * @param brand of vehicle
74  * @param model of vehicle
75  * @param vYear of vehicle
76  * @param chassisNumber of vehicle could be null
77  * @param licensPlate of vehicle could be null
78  * @return true if created else false
79  */
80 public boolean makeNewVehicle(String vType, String brand, String model, int vYear, String chassisNumber,
String licensPlate)
81 {
82     return VehicleDAO.getInstance().createActVehicle(vType, brand, model, vYear, chassisNumber, licensPlate);
83 } //end method createVehicle
84
85 /**

```

```

86  * Edit the non null paramters in the choosen vehicle (recognized by vehicleID)
87  * @param vehicleID of vehicle there have to be edited
88  * @param vType new vType or null
89  * @param brand new brand or null
90  * @param model new model or null
91  * @param vYear new vYear or null
92  * @param chassisNumber new chassisNumber or null
93  * @param licensPlate new licensPlate or null
94  * @return true if edited else false
95  */
96  public boolean editVehicle(String vehicleID, String vType, String brand, String model, int vYear, String
chassisNumber, String licensPlate)
97  {
98      return VehicleDAO.getInstance().editActVehicle(vehicleID, vType, brand, model, vYear, chassisNumber,
licensPlate);
99  } //end method editVehicle
100
101  /**
102   * Attach owner to a vehicle
103   * @param vehicleID on vehicle
104   * @param ownerID on owner
105   * @param active 1 for active - 0 for inactive
106   * @return true if attached else false
107   */
108  public boolean attachOwner(String vehicleID, String ownerID, int active)
109  {
110      return VehicleDAO.getInstance().attachActOwner(vehicleID, ownerID, active);
111  }
112
113  /**
114   * Delete a vehicle
115   * @param vehicleID on vehicle
116   * @return true if deleted else false
117   */
118  public boolean deleteVehicle(String vehicleID)
119  {
120      return VehicleDAO.getInstance().deleteActVehicle(vehicleID);
121  } //end deleteVehicle(String vehicleID)
122
123  /**
124   * Get then brand on a vehicle and the evt. owner
125   * @param vehicleID to get information from
126   * @return String with the requested inforamtion if found else NULL
127   */
128  public String getVehicleAndNameLine(String vehicleID)
129  {
130      return VehicleDAO.getInstance().getVehicleNameAndOwnerName(vehicleID);
131  } //end method getVehicleAndNameLine(String vehicleID)
132
133  /**
134   * Search for a vehicle and the owners attached
135   * @param vehicleID vehicle id for search
136   * @return List with owners the found vehicle belongs to
137   */
138
139  public boolean[] getActiveNess()
140  {
141      return VehicleDAO.getInstance().getActiveNess();

```

```
142 }
143 }
```

Workstation

```
1 package tweakmc.model;
2
3 /**
4  * Create a workstation for use then working on a vehicle in an order
5  * @author clausPallisgaardBeck
6  */
7 public class Workstation
8 {
9     //Instance variables
10    private String workstationID;
11    private String workstationName;
12
13    /**
14     * Create an instance of a workstation
15     * @param workstationID to identify workstation
16     * @param workstationName of the workstation
17     */
18    public Workstation(String workstationID, String workstationName)
19    {
20        this.workstationID = workstationID;
21        this.workstationName = workstationName;
22    } //end constructor
23
24    /**
25     * Get the ID of a workstation
26     * @return the ID
27     */
28    public String getWorkstationID()
29    {
30        return workstationID;
31    } //end method getWorkstationID()
32
33    /**
34     * Set the ID on a workstation
35     * @param workstationID for the station
36     */
37    public void setWorkstationID(String workstationID)
38    {
39        this.workstationID = workstationID;
40    } //end method setWorkstationID(String workstationID)
41
42    /**
43     * Get the name of a workstation
44     * @return the name of the station
45     */
46    public String getWorkstationName()
47    {
48        return workstationName;
49    } //end method getWorkstationName()
50
51    /**
52     * Set the name on a workstation
```



```

53  * @param workstationName of the station
54  */
55  public void setWorkstationName(String workstationName)
56  {
57      this.workstationName = workstationName;
58  } //end method setWorkstationName(String workstationName)
59 } //end class Workstation

```

WorkstationCatalog

```

1  package tweakmc.model;
2
3  import tweakmc.dataaccess.WorkstationDAO;
4
5  /**
6   * Create a catalog for workstations
7   * @author clausPallisgaardBeck
8   */
9  public class WorkstationCatalog
10 {
11     //Instance variables
12     private static WorkstationCatalog instance;
13
14     /**
15      * Create an instance of a WorkstationCatalog
16      */
17     private WorkstationCatalog()
18     {
19
20     } //end constructor for WorkstationCatalog
21
22     /**
23      * Provide a instance of workstationcatalog for use the altering or getting
24      * information about a workstation
25      * @return instance of workstation
26      */
27     public static WorkstationCatalog getInstance()
28     {
29         if(instance == null)
30         {
31             instance = new WorkstationCatalog();
32         } //end if
33         return instance;
34     } //end method getInstance()
35
36     /**
37      * Change the name on a workstation
38      * @param id on the station to change name on
39      * @param name new name for workstation
40      * @return true is changed else false
41      */
42     public boolean editWorkstation(String id, String name)
43     {
44         return WorkstationDAO.getInstance().changeWorkstationName(id, name);
45     } //end method editWorkstation(String id, String name)
46
47     /**

```

```

48  * Get the name for a workstation
49  * @param id workstation id to get name for
50  * @return String with name if found else NULL
51  */
52  public String getWorkstationName(String id)
53  {
54      String workstationName = null;
55      Vector<Workstation> wsv = (Vector<Workstation>) WorkstationDAO.getInstance().getAllWorkstation();
56
57      for (Workstation ws : wsv)
58      {
59          if(ws.getWorkstationID().equalsIgnoreCase(id))
60          {
61              workstationName = ws.getWorkstationName();
62          }//end if
63      }//end for each loop
64      return workstationName;
65  }//end getWorkstationName(String id)
66 }//end class WorkstationCatalog

```

Model/Measurement layer (tweakmc.model.Measurement)

Calculation

```

1  package tweakmc.model.Measurement;
2
3  import java.io.Serializable;
4
5  /**
6   *
7   * @author NegoZiatoR
8   */
9  public class Calculation extends Measurement implements Serializable
10 {
11     // Instance variables
12
13     /**
14      * Constructor for Calculation
15      */
16     public Calculation()
17     {
18
19     }
20
21 }
22
23

```

Letter

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.

```

```

4 */
5
6 package tweakmc.model.Measurement;
7
8 import java.io.Serializable;
9
10 /**
11 *
12 * @author NegoZiatoR
13 */
14 public class Letter extends Measurement implements Serializable
15 {
16     // Instance variables
17     protected String letterMeasurement;
18
19     /**
20      * Constructor for letterMeasurement
21      * @param letterMeasurement letter on measurement
22      */
23     public Letter(String letterMeasurement)
24     {
25         this.letterMeasurement = letterMeasurement;
26     }
27
28 }

```

Measurement

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model.Measurement;
7
8 /**
9 *
10 * @author NegoZiatoR
11 */
12 public class Measurement
13 {
14     // Instance variables
15
16     /**
17      * Constructor for Measurement
18      */
19     public Measurement()
20     {
21
22     }
23
24 }
25
26

```

Number

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model.Measurement;
7
8 import java.io.Serializable;
9
10 /**
11  *
12  * @author NegoZiatoR
13  */
14 public class Number extends Measurement implements Serializable
15 {
16     // Instance variables
17     protected float numberMeasurement;
18
19     /**
20      * Constructor for Number
21      * @param numberMeasurement number on measurement
22      */
23     public Number(float numberMeasurement)
24     {
25         this.numberMeasurement = numberMeasurement;
26     }
27
28
29 }
30
31
```

Model/Speradsheet (tweakmc.model.spreadsheet)

Spreadsheet

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model.spreadsheet;
7
8 import javax.swing.JTable;
9
10 /**
11  *
12  * @author clausPallisgaardBeck
13  */
14 public class Spreadsheet
15 {
16
17     //instance variables
```

```

18 private JTable jTable;
19 private JTable saveTable;
20
21 /**
22  * Constructor for spreadsheet, used to create a new instance to spreadsheet,
23  * for creation of empty or populated spreadsheet
24  */
25 public Spreadsheet()
26 {
27
28 } //end Spreadsheet constructor
29
30 /**
31  * Creates the tabel for the resultset after the users specifies
32  * rows and columns
33  * @param rows
34  * @param columns
35  * @param columnNames
36  * @return jTable the tabel for containing the
37  */
38 public JTable buildJTable(int rows, int columns, String[] columnNames)
39 {
40     jTable = new JTable(rows, columns);
41     if(columnNames.length > 0)
42     {
43         for (int i = 0; i < columnNames.length; i++)
44         {
45             jTable.getColumnModel().getColumn(i).setHeaderValue(columnNames[i]);
46         } //end for
47     } //end if
48     return jTable;
49 } //end buildJTable method
50
51 /**
52  * Buil a new populated spreadsheet for saving as a object thru Objecthandler
53  * @param tableToSave actual table from GUI with data
54  * @return new JTable with data, ready for saving
55  */
56 public JTable buildPopulatedJTable(JTable tableToSave)
57 {
58
59     Object[][] data = new Object[tableToSave.getRowCount()][tableToSave.getColumnCount()];
60     String[] columnNames = new String[tableToSave.getColumnCount()];
61
62     //Save data from table to 2 dimensionel array
63     for (int i = 0; i < tableToSave.getRowCount(); i++)
64     {
65         for (int j = 0; j < tableToSave.getColumnCount(); j++)
66         {
67
68             data[i][j] = tableToSave.getModel().getValueAt(i, j);
69         } //end for
70     } //end for
71
72     //Save columnnames to array
73     for(int i = 0; i < tableToSave.getColumnCount(); i++)
74     {
75         columnNames[i] = tableToSave.getColumnName(i);

```

```

76     }//end for
77
78     //Create a new table with the data and names for saving as bytes
79     saveTable = new JTable(data, columnNames);
80
81     System.gc();
82
83     return saveTable;
84 }//end buildPopulatedJTable method
85 }//end class Spreadsheet
86
87

```

SpreadsheetCatalog

```

1
2 package tweakmc.model.spreadsheet;
3
4 import javax.swing.JTable;
5 import tweakmc.dataaccess.ObjectHandler;
6
7 /**
8  *
9  * @author Tvup
10 */
11 public class SpreadsheetCatalog {
12     private static SpreadsheetCatalog instance;
13
14     //instance variables
15
16
17     public SpreadsheetCatalog()
18     {
19
20     }//end constructor
21
22     /**
23      * Singleton constructor
24      * @return instance
25      */
26     public static SpreadsheetCatalog getInstance()
27     {
28         if(instance==null)
29             return new SpreadsheetCatalog();
30         else
31             return instance;
32     }
33
34     /**
35      * This method creates a spreadsheet
36      * @param rows
37      * @param columns
38      * @param coulumnNames
39      * @return jTable
40      */
41     public JTable createSpreadsheet(int rows, int columns, String[] coulumnNames)
42     {

```

```

43     return new Spreadsheet().buildJTable(rows, columns, coulumnNames);
44 }//end method createSpreadsheet
45
46 /**
47  * This metgod saves the table with the data
48  * @param tableToSave
49  * @param fileName
50  * @return true if saved else false
51  */
52 public boolean savePopulatedTable(JTable tableToSave, String path)
53 {
54     JTable actTable = new Spreadsheet().buildPopulatedJTable(tableToSave);
55
56     if(actTable != null)
57     {
58         return ObjectHandler.objectWriter(actTable, path);
59     }//end if
60
61     else
62     {
63         return false;
64     }//end else
65 }//end method savePopulatedTable
66
67 /**
68  * Find and present selected JTable. It is returned ad Object and must be casted
69  * @param path there table is saved
70  * @return JTable to manage
71  */
72 public JTable findSelectedSpreadsheet(String path)
73 {
74     return (JTable) ObjectHandler.objectReader(path);
75 }//end method findSelectedSpreadsheet(String path)
76
77 }//end class
78

```

Model/Result layer (tweakmc.model.Result)

Result

```

1 package tweakmc.model.Result;
2
3 import java.io.Serializable;
4 import java.text.SimpleDateFormat;
5 import java.util.Calendar;
6 import java.util.Date;
7 import java.util.GregorianCalendar;
8
9 /**
10  * Creating a result for a Vehicle. Types of result are UserDefinedSpreadsheets, documents, image, videos,
11  * sounds and whiteboards
12  * @author NegoZiatoR
13  */

```

```

14 public class Result implements Serializable
15 {
16     //Types should be as follow:
17     // IMAGE, VIDEO, SOUND, DOCUMENT, SPREADSHEET or WHITEBOARD
18
19     // Instance variables
20     private String resultID;
21     private String mechanicComment;
22     private String customerComment;
23     private String resultName;
24     private String timeStamp;// yyyyymmddhhmm
25     private String resultType;
26     private String vehicleID;
27     private String workstationID;
28     private String orderID;
29     private String path = "";
30     private String whiteboardID = "";
31
32     //Calendar and Dateformats
33     Calendar localCalendar = GregorianCalendar.getInstance();
34     private SimpleDateFormat timeStampComment = new SimpleDateFormat("dd-MM-yyyy @ HH:mm | ");
35
36     //Final variables
37     private final String IMAGE = "Image";
38     private final String VIDEO = "Video";
39     private final String SOUND = "Sound";
40     private final String DOCUMENT = "Document";
41     private final String SPREADSHEET = "Spreadsheet";
42     private final String WHITEBOARD = "Whiteboard";
43
44     /**
45      * Build up a already existing result to edit or delete for and order and a vehicle, referring to the place in the
46      * filefolder or in the DB
47      * @param resultID the of the result
48      * @param mechanicComment info for mechanic
49      * @param customerComment info for customer and mechanic
50      * @param resultName of the result, added by user
51      * @param timeStamp of the result, added by the system
52      * @param resultType is what kind of result
53      * @param path if the result is anything else than a whiteboard the path for file shpuld be here else null
54      * @param vehicleID on the vehicle who should have this result assignt
55      * @param workstationID on the workstation where the result is done
56      * @param orderID the act order this Result is about
57      * @param whiteboardID if result is a whiteboard the whiteboardID should be here else null
58      */
59     public Result(String resultID, String mechanicComment, String customerComment, String resultName, String
60     timeStamp, String resultType, String path, String vehicleID, String workstationID, String orderID, String
61     whiteboardID)
62     {
63         this.resultID = resultID;
64         this.mechanicComment = mechanicComment;
65         this.customerComment = customerComment;
66         this.resultName = resultName;
67         this.timeStamp = timeStamp;
68         this.resultType = resultType;
69         this.path = path;
70         this.vehicleID = vehicleID;
71         this.workstationID = workstationID;

```



```

69     this.orderID = orderID;
70     this.whiteboardID = whiteboardID;
71 } //end constructor Result (existing Result)
72
73 /**
74  * Buil a new Result for an Vehicle and order
75  * @param mechanicComment info for mechanic
76  * @param customerComment info for customer and mechanic
77  * @param resultName of the result, added by user
78  * @param resultType is what kind of result
79  * @param path if the result is anything else than a whiteboard the path for file shpuld be here else null
80  * @param vehicleID on the vehicle who should have this result assignt
81  * @param workstationID on the workstation where the result is done
82  * @param orderID the act order this Result is about
83  * @param whiteboardID if result is a whiteboard the whiteboardID should be here else null
84  */
85 public Result(String mechanicComment, String customerComment, String resultName, String resultType, String
path, String vehicleID, String workstationID, String orderID, String whiteboardID)
86 {
87     this.mechanicComment = mechanicComment;
88     this.customerComment = customerComment;
89     this.resultName = resultName;
90     this.timeStamp = getFormattedDateNow();
91     this.resultType = resultType;
92     this.path = path;
93     this.vehicleID = vehicleID;
94     this.workstationID = workstationID;
95     this.orderID = orderID;
96     this.whiteboardID = whiteboardID;
97 } //end constructor Result (new Result)
98
99 /**
100  * Returns the date (String) for presenting resultname and comments
101  * @return dateNowFormatted ("dd-MM-yyyy @ HH:mm | ")
102  */
103 private String getFormattedDateNow()
104 {
105     Date dateNow = localCalendar.getTime();
106     String dateNowFormatted = timeStampComment.format(dateNow);
107     return dateNowFormatted;
108 } //end getDateNow method
109
110 /**
111  * Get resultID
112  * @return String with resultID
113  */
114 public String getResultID()
115 {
116     return resultID;
117 } //end method getResultID()
118
119 /**
120  * Set resultID
121  * @param resultID
122  */
123 public void setResultID(String resultID)
124 {
125     this.resultID = resultID;

```

```

126 }//end method setResultID(String resultID)
127
128 /**
129  * Get the customercomment
130  * @return customercomment with timestamp
131  */
132 public String getCustomerComment()
133 {
134     return customerComment;
135 }//end method getCustomerComment()
136
137 /**
138  * Set the customercomment
139  * @param customerComment
140  */
141 public void setCustomerComment(String customerComment)
142 {
143     this.customerComment = customerComment;
144 }//end method setCustomerComment
145
146 /**
147  * Get mechaniccomment
148  * @return mechaniccomment with timestamp
149  */
150 public String getMechanicComment()
151 {
152     return mechanicComment;
153 }//end method getMechanicComment
154
155 /**
156  * Set mechaniccomments
157  * @param mechanicComment
158  */
159 public void setMechanicComment(String mechanicComment)
160 {
161     this.mechanicComment = mechanicComment;
162 }//end method setMechanicComment
163
164 /**
165  * Get the name of the result, set by the user
166  * @return the name of the result with the timestamp for creation of the result
167  */
168 public String getResultName()
169 {
170     return this.resultName;
171 }//end method getResultName
172
173 /**
174  * Set resultName, can be changed by user
175  * @param resultName name of the result
176  */
177 public void setResultName(String resultName)
178 {
179     this.resultName = resultName;
180 }//end method setResultName
181
182 /**
183  * Get the type of this result

```

```

184  * @return the type
185  */
186  public String getResultType()
187  {
188      return resultType;
189  }//end method getResultType
190
191  /**
192   * Set the type of this result
193   * @param resultType actual result type
194   */
195  public void setResultType(String resultType)
196  {
197      this.resultType = resultType;
198  }//end method setResultType
199
200  /**
201   * Get the vehicleid for this result
202   * @return actual vehicleID
203   */
204  public String getVehicleID()
205  {
206      return vehicleID;
207  }//end method getVehicleID
208
209  /**
210   * Set the actual vehicleID for this result
211   * @param vehicleID to be added for this result
212   */
213  public void setVehicleID(String vehicleID)
214  {
215      this.vehicleID = vehicleID;
216  }//end method setVehicleID
217
218  /**
219   * Get the workstationID for this result
220   * @return actualworkstationID for this result
221   */
222  public String getWorkstationID()
223  {
224      return workstationID;
225  }//end method getWorkstationID
226
227  /**
228   * Set the workstationID for this result
229   * @param workstationID been used for this Result
230   */
231  public void setWorkstationID(String workstationID)
232  {
233      this.workstationID = workstationID;
234  }//end method setWorkstationID
235
236  /**
237   * Set the orderID
238   * @param orderID
239   */
240  public void setOrderID(String orderID)
241  {

```

```

242     this.orderID = orderID;
243 }//end method setOrderID(String orderID)
244
245 /**
246  * Get the orderID
247  * @return orderID
248  */
249 public String getOrderID()
250 {
251     return orderID;
252 }//end method orderID
253
254 /**
255  * Get the date and time then this result is created
256  * @return a timestamp in the format yyyyMMddHHmm
257  */
258 public String getTimeStamp()
259 {
260     return this.timeStamp;
261 }//end method getTimeStamp
262
263 /**
264  * Get the path where this file is saved
265  * @return the complete path
266  */
267 public String getPath()
268 {
269     return path;
270 }//end method getPath
271
272 /**
273  * Set the path where this file is saved
274  * @param path place to save file
275  */
276 public void setPath(String path)
277 {
278     this.path = path;
279 }//end method setPath
280
281 /**
282  * Return the ID ont his whiteboard
283  * @return id
284  */
285 public String getWhiteboardID()
286 {
287     return whiteboardID;
288 }//end method getWhiteboardID()
289
290 /**
291  * set the ID for this whiteboard
292  * @param whiteboardID to set for this whiteboard
293  */
294 public void setWhiteboardID(String whiteboardID)
295 {
296     this.whiteboardID = whiteboardID;
297 }//end method setWhiteboardID(String whiteboardID)
298
299 /**

```

```

300  * Return the names of the result as a string formatted: date ("dd-MM-yyyy @ HH:mm | ") + name
301  * @return string with name for result
302  */
303  @Override
304  public String toString()
305  {
306      return timeStamp + resultName;
307  } //end method toString()
308 } //end class Result

```

ResultCatalog

```

1  package tweakmc.model.Result;
2
3  import java.io.Serializable;
4  import java.text.SimpleDateFormat;
5  import java.util.Calendar;
6  import java.util.Date;
7  import java.util.GregorianCalendar;
8  import java.util.Vector;
9  import javax.swing.JTable;
10 import tweakmc.dataaccess.*;
11 import tweakmc.model.spreadsheet.SpreadsheetCatalog;
12 import tweakmc.utility.FileHandlerUtility;
13 /**
14  * This is the catalog for the results. It will create spreadsheets and you're able to get results from this class
15  * @author NegoZiatoR
16  */
17 public class ResultCatalog implements Serializable
18 {
19     // Instance Variables
20     private static ResultCatalog instance;
21
22
23     //Calendar and Dateformats
24     Calendar localCalendar = GregorianCalendar.getInstance();
25     private SimpleDateFormat timeStampPath = new SimpleDateFormat("ddMMyyyyHHmm");
26
27     //Final variables
28     private final String IMAGE = "Image";
29     private final String VIDEO = "Video";
30     private final String SOUND = "Sound";
31     private final String DOCUMENT = "Document";
32     private final String SPREADSHEET = "Spreadsheet";
33     private final String WHITEBOARD = "Whiteboard";
34
35     private final String FOLDERS = "C:/Tweakfiles/";
36     private final String SPREADSHEET_EXTENSION = ".47";
37     private final String EXTSEPERATOR = ".";
38
39
40     // Private constructor for ResultCatalog
41     private ResultCatalog()
42     {
43
44     } //end constructor
45

```

```

46  /**
47   * Singleton method for ResultCatalog
48   * @return ONE instance of ResultCatalog
49   */
50  public static ResultCatalog getInstance()
51  {
52      if(instance == null)
53      {
54          instance = new ResultCatalog();
55      }
56      return instance;
57  } //end getInstance
58
59  /**
60   * Create a new result in Database
61   * @param resultType wich result to create
62   * @return true if created else false
63   */
64  public boolean createNewUploadResult(String originalDestination, String mechanicComment, String
customerComment, String resultName, String resultType, String vehicleID, String workstationID, String orderID,
String whiteboardID)
65  {
66      String path = FOLDERS + vehicleID + "/" + vehicleID + getTimeStampForPath() + resultName +
FileHandlerUtility.getPathExtension(originalDestination);
67      if(ObjectHandler.copyFileToNewDestination(originalDestination, path))
68      {
69          return ResultDAO.getInstance().createNewActResult(new Result(mechanicComment, customerComment,
resultName, resultType, path, vehicleID, workstationID, orderID, whiteboardID));
70      }
71      else
72      {
73          return false;
74      }
75  }
76  } //end method makeNewResult
77
78  /**
79   * This method creates a new spreadsheet-result
80   * @param tableToSave
81   * @param mechanicComment
82   * @param customerComment
83   * @param resultName
84   * @param vehicleID
85   * @param workstationID
86   * @param orderID
87   * @return boolean
88   */
89  public boolean createNewSpreadsheetResult(JTable tableToSave, String mechanicComment, String
customerComment, String resultName, String vehicleID, String workstationID, String orderID)
90  {
91
92      String path = FOLDERS + vehicleID + "/" + vehicleID + getTimeStampForPath() + resultName +
SPREADSHEET_EXTENSION;
93      if(SpreadsheetCatalog.getInstance().savePopulatedTable(tableToSave, path))
94      {
95          return ResultDAO.getInstance().createNewActResult(new Result(mechanicComment, customerComment,
resultName, SPREADSHEET, path, vehicleID, workstationID, orderID, ""));
96      } //end if

```

```

97     else
98     {
99         return false;
100    } //end else
101 } //end method spreadsheetResult(JTable tableToSave, String mechanicComment, String customerComment,
String resultName, String resultType, String vehicleID, String workstationID, String timeStamp)
102
103 /**
104  * Return a vector with found result for given IDs
105  * @param vehicleID to find result for
106  * @param workstationID to find result for
107  * @return Vector with result or NULL
108  */
109 public Vector<Result> getResultFromIDs(String vehicleID, String workstationID)
110 {
111     return ResultDAO.getInstance().getResultsForIDs(vehicleID, workstationID);
112 } //end method getResultFromIDs(String vehicleID, String workstationID)
113
114 /**
115  * Get a timestamp for a unique path to save files with
116  * @return string with unique timeStamp
117  */
118 private String getTimeStampForPath()
119 {
120     Date dateNow = localCalendar.getTime();
121     String pathTimeStamp = timeStampPath.format(dateNow);
122     return pathTimeStamp;
123 } //end method getTimeStampForPath()
124 } //end class ResultCatalog
125

```

Model/result/whiteboard layer (tweakmc.model.Result.Whiteboard)

BearingToleranceTable

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model.Result.Whiteboard;
7
8 import java.io.Serializable;
9
10 /**
11  *
12  * @author NegoZiatoR
13  */
14 public class BearingToleranceTable extends Whiteboard implements Serializable
15 {
16     // Instance variables
17

```

```

18  /**
19   * Constructor for BearingToleranceTable
20   * @param comment on BearingToleranceTable
21   * @param resultName BearingToleranceTable name
22   * @param timeStamp on BearingToleranceTable
23   * @param tableID on BearingToleranceTable
24   */
25  public BearingToleranceTable()
26  {
27
28  }
29
30 }

```

ConnectingRod

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package tweakmc.model.Result.Whiteboard;
7
8  import java.io.Serializable;
9
10 /**
11  *
12  * @author NegoZiatoR
13  */
14 public class ConnectingRod extends BearingToleranceTable implements Serializable
15 {
16     // Instance variables
17
18     /**
19      * Constructor for ConnectingRod
20      * @param comment comment on ConnectingRod
21      * @param resultName ConnectingRod name
22      * @param timeStamp on ConnectingRod
23      * @param tableID on ConnectingRod
24      */
25  public ConnectingRod()
26  {
27
28  }
29
30 }

```

CylinderToleranceTable

```

1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5
6  package tweakmc.model.Result.Whiteboard;
7

```



```

8 import java.io.Serializable;
9
10 /**
11  *
12  * @author NegoZiatoR
13  */
14 public class CylinderToleranceTable extends Whiteboard implements Serializable
15 {
16     // Instance variables4
17     protected int noOfCylinders;
18
19     /**
20      * Constructor for CylinderToleranceTable
21      * @param comment comment on CylinderToleranceTable
22      * @param resultName CylinderToleranceTable name
23      * @param timeStamp on CylinderToleranceTable
24      * @param tableID on CylinderToleranceTable
25      * @param noOfCylinders on CylinderToleranceTable
26      */
27     public CylinderToleranceTable(int noOfCylinders)
28     {
29         this.noOfCylinders = noOfCylinders;
30     }
31
32 }

```

Head

```

1 /**
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.model.Result.Whiteboard;
7
8 import java.io.Serializable;
9
10 /**
11  *
12  * @author NegoZiatoR
13  */
14 public class Head extends BearingToleranceTable implements Serializable
15 {
16     // Instance variables
17
18     /**
19      * Constructor for Head
20      * @param comment on Head
21      * @param resultName Head name
22      * @param timeStamp on Head
23      * @param tableID on Head
24      */
25     public Head()
26     {
27
28     }
29

```

```
30 }
31
```

ValveAjustmentTable

```
1 package tweakmc.model.Result.Whiteboard;
2
3 import java.io.Serializable;
4 import java.text.SimpleDateFormat;
5 import java.util.Calendar;
6 import java.util.Date;
7 import java.util.GregorianCalendar;
8
9 /**
10  * This class is the template for the Valve-adjustment table.
11  * It's serializable since it's possible to write it to a file
12  * @author NegoZiatoR
13  */
14 public class ValveAdjustmentTable extends Whiteboard implements Serializable
15 {
16     // Instance variables
17     private String timeStamp;// yyyyymmddhhmm
18     private String resultID;
19     private String[] cylinderNo;
20     private String[] cylinderPos;
21
22     private String vehicleID;
23     private String workstationID;
24     private String orderID;
25
26     // Arrays with filled table content
27     private String[] inArray;
28     private String[] tollArray;
29     private String[] diffArray;
30     private String[] shimArray;
31     private String[] newShimArray;
32
33     //Calendar and Dateformats
34     Calendar localCalendar = GregorianCalendar.getInstance();
35     private SimpleDateFormat timeStampComment = new SimpleDateFormat("dd-MM-yyyy @ HH:mm | ");
36
37     /**
38      * Constructor for valveadjustment table
39      * @param mechanicComment mechanicComment is a comment set by the user for internal use in the company
40      * @param customerComment customerComment is a comment set by the user to be shown to the owner
41      * (customer)
42      * @param cylinderNo cylinder numbers on table
43      * @param cylinderPos cylinder positions on cylinder numbers
44      * @param inArray in value on cylinders
45      * @param tollArray tollerance on cylinders
46      * @param diffArray difference on cylinders
47      * @param shimArray shim on cylinders
48      * @param newShimArray new shim on cylinders
49      */
50     public ValveAdjustmentTable(String mechanicComment, String customerComment, String[] cylinderNo, String[]
51 cylinderPos, String[] inArray, String[] tollArray,
```

```

50         String[] diffArray, String[] shimArray, String[] newShimArray, String vehicleID,
String workstationID, String orderID)
51     {
52         super(mechanicComment, customerComment);
53
54
55         this.resultID = "";
56         this.vehicleID = vehicleID;
57         this.workstationID = workstationID;
58         this.orderID = orderID;
59         this.timeStamp = getFormattedDateNow();
60         this.cylinderNo = cylinderNo;
61         this.cylinderPos = cylinderPos;
62         this.inArray = inArray;
63         this.tollArray = tollArray;
64         this.diffArray = diffArray;
65         this.shimArray = shimArray;
66         this.newShimArray = newShimArray;
67     }
68
69
70     /**
71     * Return vehicleID on table
72     * @return vehicleId on table
73     */
74     public String getVehicleID()
75     {
76         return vehicleID;
77     }
78
79     /**
80     * Return workstationID on table
81     * @return workstationID on table
82     */
83     public String getWorkstationID()
84     {
85         return workstationID;
86     }
87
88     /**
89     * Return orderID on table
90     * @return orderID on table
91     */
92     public String getOrderID()
93     {
94         return orderID;
95     }
96
97     /**
98     * Return all cylinder numbers
99     * @return all cylinder numbers
100    */
101    public String[] getCylinderNo()
102    {
103        return cylinderNo;
104    }
105
106    /**

```

```
107  * Return all cylinder positions
108  * @return all cylinder positions
109  */
110  public String[] getCylinderPos()
111  {
112      return cylinderPos;
113  }
114
115  /**
116   * return all differences on cylinders
117   * @return all differences on cylinders
118   */
119  public String[] getDiffArray()
120  {
121      return diffArray;
122  }
123
124  /**
125   * return all in values on cylinders
126   * @return all in values on cylinders
127   */
128  public String[] getInArray()
129  {
130      return inArray;
131  }
132
133  /**
134   * return all newShims on cylinders
135   * @return all newshims on cylinders
136   */
137  public String[] getNewShimArray()
138  {
139      return newShimArray;
140  }
141
142  /**
143   * return all shims on cylinders
144   * @return all shims on cylinders
145   */
146  public String[] getShimArray()
147  {
148      return shimArray;
149  }
150
151  /**
152   * Return the table ID
153   * @return
154   */
155  public String getResultID()
156  {
157      return resultID;
158  }
159
160  public void setResultID(String resultID)
161  {
162      this.resultID = resultID;
163  }
164
```

```

165  /**
166   * Return the tollerance on cylinders
167   * @return all tollerances on cylinders
168   */
169  public String[] getTollArray()
170  {
171      return tollArray;
172  }
173
174  /**
175   * Returns the date (String) for presenting resultname and comments
176   * @return dateNowFormatted ("dd-MM-yyyy @ HH:mm | ")
177   */
178  private String getFormattedDateNow()
179  {
180      Date dateNow = localCalendar.getTime();
181      String dateNowFormatted = timeStampComment.format(dateNow);
182      return dateNowFormatted;
183  } //end getDateNow method
184
185  /**
186   * Get the date and time then this result is created
187   * @return a timestamp in the format yyyyMMddHHmm
188   */
189  public String getTimeStamp()
190  {
191      return this.timeStamp;
192  } //end method getTimeStamp
193
194
195 }

```

Whiteboard

```

1 package tweakmc.model.Result.Whiteboard;
2
3 import java.io.Serializable;
4 /**
5  * This class is the template for the whiteboards
6  * It's serializable since it's possible to store to a file
7  * @author NegoZiatoR
8  */
9 public class Whiteboard implements Serializable
10 {
11     // Instance variables
12     private String resultID;
13     private String mechanicComment;
14     private String customerComment;
15
16     /**
17      * Constructor for whiteboard
18      * @param mechanicComment
19      * @param customerComment
20      */
21     public Whiteboard(String mechanicComment, String customerComment)
22     {
23         this.mechanicComment = mechanicComment;
24         this.customerComment = customerComment;
25     } // End constructor for Whiteboard

```

```

26
27  /**
28   * return mechanic comment
29   * @return mechanic comment
30   */
31  public String getMechanicComment()
32  {
33      return mechanicComment;
34  } // End getMechanicComment method
35
36  /**
37   * Return customer comment
38   * @return customer comment
39   */
40  public String getCustomerComment()
41  {
42      return customerComment;
43  } // End getCustomerComment method
44
45 } //end class whiteboard
46
47

```

WhiteboardCatalog

```

1  package tweakmc.model.Result.Whiteboard;
2
3  import java.io.Serializable;
4  import tweakmc.dataaccess.WhiteboardDAO;
5  /**
6   * This class is the catalog for WhiteBoards. It's possible to create WhiteBoards with the class.
7   *
8   * @author NegoZiatoR
9   */
10 public class WhiteboardCatalog implements Serializable
11 {
12     // Instance variables
13     private static WhiteboardCatalog instance;
14
15     // Private constructor for whiteboardCatalog
16     private WhiteboardCatalog()
17     {
18
19     }
20
21     /**
22      * Singleton method for whiteboardCatalog
23      * @return whiteboardCatalog
24      */
25     public static WhiteboardCatalog getInstance()
26     {
27         if(instance == null)
28         {
29             instance = new WhiteboardCatalog();
30         }
31         return instance;
32     }

```

```

33
34
35 /**
36  * Create a new valveAdjustment table whiteboard
37  * @param mechanicComment mechanics comments on table
38  * @param customerComment comments on table for customer
39  * @param cylinderNo cylinder numbers on table
40  * @param cylinderPos cylinder positions on cylinder numbers
41  * @param inArray in value on cylinders
42  * @param tollArray tollerance on cylinders
43  * @param diffArray difference on cylinders
44  * @param shimArray shim on cylinders
45  * @param newShimArray new shim on cylinders
46  * @return true if created else false
47  */
48 public boolean makeNewValveAdjustmentWhiteboard(String mechanicComment, String customerComment,
String[] cylinderNo, String[] cylinderPos, String[] inArray, String[] tollArray,
49 String[] diffArray, String[] shimArray, String[] newShimArray, String vehicleID,
String workstationID, String orderID)
50 {
51     ValveAdjustmentTable whiteboard = new ValveAdjustmentTable(mechanicComment, customerComment,
cylinderNo, cylinderPos, inArray, tollArray,
52 diffArray, shimArray, newShimArray, vehicleID, workstationID,
orderID);
53
54     return WhiteboardDAO.getInstance().createActValveadjustmentTablewhiteboard(whiteboard);
55 }
56 }
57

```

Dataaccess layer (tweakmc.dataaccess)

AutocompleteDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.List;
8 import tweakmc.utility.FileWriterException;
9
10
11 /**
12  * Connection to the Database when doing autocomplete things
13  * @author Tvup
14  */
15 public class AutocompleteDAO extends DAO
16 {
17
18     private static AutocompleteDAO instance;
19
20     /**

```

```

21  * Private constructor for AutocompleteDAO
22  */
23  private AutocompleteDAO()
24  {
25
26  } //end constructor
27
28  public static AutocompleteDAO getInstance()
29  {
30      if(instance == null)
31      {
32          instance = new AutocompleteDAO();
33      } //end if
34      return instance;
35  } //end getInstance method
36
37  /**
38   * This method inserts new suggestions in the relevant database tables.
39   * @param suggestionType
40   * @param suggestion
41   * @return boolean
42   */
43  public boolean createNewSuggestion(String suggestionType, String suggestion)
44  {
45      try
46      {
47          //Get the highest identifier from the database at the moment, and add 1 to that number
48          int suggestionID = getHighestOwnerIdentifierfromDatabase(suggestionType);
49          suggestionID++;
50          PreparedStatement pstmt = getPreparedStatement("INSERT INTO " + suggestionType + " (Suggestions, " +
suggestionType + "ID) VALUES(?,?)");
51          pstmt.setString(1, suggestion);
52          pstmt.setInt(2, suggestionID);
53          pstmt.execute();
54          return true;
55      } // end try
56
57      catch (SQLException sqllex)
58      {
59          FileWriterException.writeLogFile(AutocompleteDAO.class.getName() + sqllex.getMessage());
60          return false;
61      } // end catch
62
63      finally
64      {
65          close();
66      } //end finally
67  } // end method createNewSuggestion
68
69  /**
70   * Search for owner with given searchcriteria
71   * @param suggestionType
72   * @return list with actFounds or null
73   */
74  public List<String> getSuggestions(String suggestionType)
75  {
76      //Create the list which we're going to store the suggestions in.
77      ArrayList<String> actFounds = new ArrayList<String>();

```



```

78
79     ResultSet rsr = null;
80     try
81     {
82
83         PreparedStatement pstmt = getPreparedStatement("SELECT * FROM " + suggestionType + " ORDER BY
suggestions ASC");
84         rsr = pstmt.executeQuery();
85
86         while(rsr.next())
87         {
88             String suggestion = rsr.getString("suggestions");
89             actFounds.add(suggestion);
90         }//end while
91     }
92 }//end try
93
94 catch (SQLException sqllex)
95 {
96     FileWriterException.writeLogFile(AutocompleteDAO.class.getName() + sqllex.getMessage());
97 }//end catch
98
99 finally
100 {
101     close();
102 }//end finally
103
104 return actFounds;
105 }//end method getSuggestions
106
107 /**
108  * Search for owners and generate a list of their first and last names
109  * @return list with fullNames or null
110  */
111 public List<String> getAllNames()
112 {
113     //Create the list which is going to hold the first and lastnames
114     ArrayList<String> allNames = new ArrayList<String>();
115     ResultSet rsr = null;
116
117     try
118     {
119
120         PreparedStatement pstmt = getPreparedStatement("SELECT firstName,lastName FROM OWNER");
121         rsr = pstmt.executeQuery();
122
123         while(rsr.next())
124         {
125             String fullName = rsr.getString("firstName");
126             fullName = fullName + " ";
127             fullName = fullName + rsr.getString("lastName");
128             allNames.add(fullName);
129         }//end while
130
131
132
133     }//end try
134

```

```

135     catch (SQLException sqlex)
136     {
137         FileWriterException.writeLogFile(AutocompleteDAO.class.getName() + sqlex.getMessage());
138     } //end catch
139
140     finally
141     {
142         close();
143     } //end finally
144
145     return allNames;
146 } //end method getAllNames
147
148 /**
149  * This method gets the highest identifier from the database. It could be used in the process of generating an ID.
150  * @param suggestionType
151  * @return int
152  */
153 public int getHighestOwnerIdentifierfromDatabase(String suggestionType)
154 {
155     try
156     {
157         PreparedStatement pstm = getPreparedStatement("SELECT MAX(identifier) FROM " +
158 suggestionType);
159         ResultSet rsr = null;
160         rsr = pstm.executeQuery();
161         rsr.next();
162         return rsr.getInt(1);
163     } //end try
164     catch (SQLException sqlex)
165     {
166         FileWriterException.writeLogFile(AutocompleteDAO.class.getName() + sqlex.getMessage());
167         return -1;
168     } //end catch
169
170     finally
171     {
172         close();
173     } //end finally
174 } //end getHighestOwnerIdentifierfromDatabase method
175 } //end class

```

CompanyDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.SQLException;
5 import tweakmc.utility.FileWriterException;
6
7 /**
8  * Connection between the system and the the Database
9  * Transforming java code to SQL
10 * @author Romanty
11 */
12 public class CompanyDAO extends DAO

```

```

13 {
14     // instance variable
15
16     private static CompanyDAO instance;
17
18     private CompanyDAO()
19     {
20
21     } //end constructor
22
23     /** singleturn methord for CompanyDAO
24     * @return instance of CompanyDAO
25     *
26     */
27
28     public static CompanyDAO getInstance()
29     {
30         if (instance == null)
31         {
32             instance = new CompanyDAO();
33         }
34         return instance;
35
36     } //end get instance methord
37
38     /**
39     * Setting the company information, to use for print out
40     * @param cvrNo
41     * @param name
42     * @param address
43     * @param phone
44     * @param email
45     * @param homePage
46     * @param bankInfo
47     * @return
48     */
49
50
51     public boolean createCompany(String cvrNo, String name, String address, String phone, String email, String
homePage, String bankInfo)
52     {
53         try
54         {
55
56             PreparedStatement pstmt= getPreparedStatement("INSERT INTO
COMPANY(cvrNo,name,address,phone,email,homePage,bankInfo) VALUE (?,?,,?,?,,?)");
57             pstmt.setString(1, cvrNo);
58             pstmt.setString(2, name);
59             pstmt.setString(3, address);
60             pstmt.setString(4, phone);
61             pstmt.setString(5, email);
62             pstmt.setString(6, homePage);
63             pstmt.setString(7, bankInfo);
64             pstmt.execute();
65             return true;
66         } //end try
67         catch (SQLException sqllex)
68         {

```

```

69         FileWriterException.writeLogFile(CompanyDAO.getInstance()
70             + sqllex.getMessage());
71         return false;
72     }
73 } //end catch
74 } //end method createCompany
75
76 /**
77  * Edit company method
78  * @param cvrNo
79  * @param name
80  * @param address
81  * @param phone
82  * @param email
83  * @param homePage
84  * @param bankInfo
85  * @return boolean
86  */
87 public boolean editCompany(String cvrNo, String name, String address,String phone,String email, String
homePage, String bankInfo)
88 {
89     try
90     {
91
92         PreparedStatement pstmt= getPreparedStatement("UPDATE Company SET cvrNo=?, name=?, address=?,
phone=?, email=?, homePage=?, bankInfo=?");
93         pstmt.setString(1, cvrNo);
94         pstmt.setString(2, name);
95         pstmt.setString(3, address);
96         pstmt.setString(4, phone);
97         pstmt.setString(5, email);
98         pstmt.setString(6, homePage);
99         pstmt.setString(7, bankInfo);
100        pstmt.execute();
101        return true;
102    }
103 } //end try
104 catch(SQLException sqllex)
105 {
106     FileWriterException.writeLogFile(CompanyDAO.class.getName() + sqllex.getMessage());
107     return false;
108 } // End Catch
109 finally
110 {
111
112     } // End Finally
113 } //end method editCompany
114
115 } //end class CompanyDAO

```

DAO

```

1 package tweakmc.dataaccess;
2
3 import java.io.File;
4 import java.sql.Connection;
5 import java.sql.PreparedStatement;
6 import java.sql.ResultSet;

```

```

7 import java.sql.SQLException;
8 import java.sql.Statement;
9 import tweakmc.utility.FileWriterException;
10
11 /**
12  * Connection between the system and the the Database
13  * Transforming java code to SQL
14  * This is the super there we define the most common statements for use in all DAO classes
15  * @author NegoZiatoR
16  */
17 public class DAO
18 {
19     private final String ROOT_DIRECTORY = "C:/Tweakfiles/";
20
21     /**
22      * Return the statement
23      * @return
24      * @throws SQLException
25      */
26     protected static Statement getStatement() throws SQLException
27     {
28         Connection dbconn = DBHandler.getInstance().getConnection();
29         return dbconn.createStatement();
30     } //end getStatement
31
32
33     protected static PreparedStatement getPreparedStatement(String sql) throws SQLException
34     {
35         Connection dbconn = DBHandler.getInstance().getConnection();
36         return dbconn.prepareStatement(sql);
37     } //end getPreparedStatement
38
39     protected static void close()
40     {
41         try
42         {
43             DBHandler.getInstance().getConnection().close();
44         } //end try
45
46         catch (SQLException ex)
47         {
48             FileWriterException.writeLogFile(DAO.class.getName() + " / close/ " + ex.getMessage());
49         } //end catch
50     } //end close method
51
52     /**
53      * Gets the identifier from the database for the vehicle in question
54      * @param vehicleID
55      * @return Integer
56      */
57     protected Integer getVehicleIdentifier(String vehicleID)
58     {
59         Integer integer_identifier = null;
60         try
61         {
62             PreparedStatement pstmtGetTypeBrandModel = getPreparedStatement("SELECT identifier FROM vehicle
WHERE vehicleID = ?");
63             pstmtGetTypeBrandModel.setString(1, vehicleID);

```

```

64     pstmtGetTypeBrandModel.execute();
65     ResultSet rsr = null;
66     rsr = pstmtGetTypeBrandModel.executeQuery();
67     if(rsr.next())
68         integer_identifier = rsr.getInt(1);
69     } //end try
70     catch (SQLException sqllex)
71     {
72         FileWriterException.writeLogFile(VehicleDAO.class.getName() + " / getVehicleIdentifier/ " +
sqllex.getMessage());
73         return null;
74     } //end catch
75     finally
76     {
77         close();
78     } //end finally
79     return integer_identifier;
80 } //end getVehicleIdentifier(String vehicleID)
81
82 /**
83  * This returns the identifier from the database for the owner in question
84  * @param ownerID
85  * @return Integer
86  */
87 protected Integer getOwnerIdentifier(String ownerID)
88 {
89     Integer integer_identifier = null;
90     try
91     {
92         PreparedStatement pstmtGetTypeBrandModel = getPreparedStatement("SELECT identifier FROM owner
WHERE ownerID = ?");
93         pstmtGetTypeBrandModel.setString(1, ownerID);
94         pstmtGetTypeBrandModel.execute();
95         ResultSet rsr = null;
96         rsr = pstmtGetTypeBrandModel.executeQuery();
97         if(rsr.next())
98             integer_identifier = rsr.getInt(1);
99     } //end try
100    catch (SQLException sqllex)
101    {
102        FileWriterException.writeLogFile(VehicleDAO.class.getName() + "/getIdentifier/" + sqllex.getMessage());
103        return null;
104    } //end catch
105    finally
106    {
107        close();
108    } //end finally
109    return integer_identifier;
110 } //end getIdentifier(String ownerID)
111
112 /**
113  * This returns the identifier from the database for the given objectType
114  * @param id to search identifier from
115  * @param objectType
116  * @return Integer identifier
117  */
118 protected Integer getIdentifierFromID(String id, String objectType)
119 {

```

```

120     Integer integer_identifier = null;
121     try
122     {
123         PreparedStatement pstmtGetTypeBrandModel = getPreparedStatement("SELECT identifier FROM " +
objectType + " WHERE LOWER(" + objectType + "ID) = LOWER(?)");
124         pstmtGetTypeBrandModel.setString(1, id);
125         pstmtGetTypeBrandModel.execute();
126         ResultSet rsr = null;
127         rsr = pstmtGetTypeBrandModel.executeQuery();
128         if(rsr.next())
129             integer_identifier = rsr.getInt(1);
130     }//end try
131     catch (SQLException sqllex)
132     {
133         FileWriterException.writeLogFile(VehicleDAO.class.getName() + "/getIdentiferFromID/" +
sqllex.getMessage());
134         return null;
135     }//end catch
136     finally
137     {
138         close();
139     }//end finally
140     return integer_identifier;
141 }//end getIdentiferFromID(String id, String objectType)
142
143 /**
144  * This returns the ID from the database for the given objectType
145  * @param identifier to find id for
146  * @param objectType in which table
147  * @return String ID
148  */
149 protected String getIDFromIdentifier(int identifier, String objectType)
150 {
151     String string_ID = "";
152
153     try
154     {
155         PreparedStatement pstmt = getPreparedStatement("SELECT " + objectType + "ID FROM " + objectType +
" WHERE identifier = ?");
156         pstmt.setInt(1, identifier);
157         ResultSet rsr = pstmt.executeQuery();
158
159         while(rsr.next())
160         {
161             string_ID = rsr.getString(1);
162         }//end while
163     }//end try
164     catch (SQLException sqllex)
165     {
166         FileWriterException.writeLogFile(VehicleDAO.class.getName() + "/getIDFromIdentifier/" +
sqllex.getMessage());
167         return "";
168     }//end catch
169     finally
170     {
171         close();
172     }//end finally
173     return string_ID;

```

```

174 }//end method getIDFromIdentifier(int identifier, String objectType)
175
176 /**
177  * Define a autocommitment in DAO
178  * @param state true for auto commit false for not
179  */
180 protected void defineAutoCommit(boolean state)
181 {
182     DBHandler.defineAutoCommit(state);
183 } // end defineAutoCommit method
184
185 /**
186  * Execute a commit
187  */
188 protected void exeCommit()
189 {
190     DBHandler.exeCommit();
191 } // end exeCommit method
192
193 /**
194  * Make a new directory for a vehicle
195  * @param str_vehicleID on vehile
196  * @return true if created else false
197  */
198 protected boolean makeDir(String str_vehicleID)
199 {
200     //Check if root directory is created
201     File rootDir = new File(ROOT_DIRECTORY);
202     if(!rootDir.exists())
203     {
204         try
205         {
206             rootDir.mkdir();
207         }//end try
208
209         catch (Exception e)
210         {
211             FileWriterException.writeLogFile(DAO.class.getName() + " / makedir / " + e.getMessage());
212             return false;
213         }//end catch
214     }//end if
215
216     //Create vehicle dir
217     try
218     {
219         return new File(ROOT_DIRECTORY + str_vehicleID).mkdir();
220     } // end try
221
222     catch(SecurityException se)
223     {
224         FileWriterException.writeLogFile(DAO.class.getName() + " / makedir / " + se.getMessage());
225         return false;
226     }// end catch
227 } // end makeDir
228 }//end DAO class

```


DBHandler

```
1 package tweakmc.dataaccess;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import javax.swing.JOptionPane;
7 import tweakmc.utility.FileWriterException;
8
9 /**
10  * Handel connection with DB
11  * @author NegoZiatoR
12  */
13 public class DBHandler
14 {
15     private final static DBHandler instance = new DBHandler();
16     private static Connection dbconn;
17     public String db_rev;
18     public String addressToDriver;
19     public String connectionString;
20     public String dbUser;
21     public String dbPassword;
22
23
24     /**
25      * Constructor for DB Handler
26      */
27     private DBHandler()
28     {
29         db_rev = "REV_348";
30     } //end constructor
31
32     public void setDataBase()
33     {
34         addressToDriver = "org.apache.derby.jdbc.ClientDriver";
35         connectionString = "jdbc:derby://localhost:1527/tweakMC";
36     } //end method setDataBase
37
38     /**
39      * This method sets which database we're going to use (It depends on the choice of the user)
40      * and it's only relevant if a derby-db isn't available
41      * @param Option
42      */
43     public void setDataBase(int Option)
44     {
45         if(Option==1)
46         {
47             addressToDriver = "org.apache.derby.jdbc.ClientDriver";
48             connectionString = "jdbc:derby://localhost:1527/tweakMC";
49         } //end if
50
51         if(Option==0)
52         {
53             addressToDriver = "com.mysql.jdbc.Driver";
54             connectionString = "jdbc:mysql://welovefailing.com:3306/tweakMC";
55             dbUser = "webuser";
```

```

56     dbPassword = "web";
57 }//end if
58
59 if(Option==2)
60 {
61     JOptionPane.showMessageDialog(null, "Right, quit");
62     System.exit(0);
63 }//end if
64
65 }
66
67 /**
68  * Return the instance of DBHandler
69  * @return DBHandlers only instance
70  */
71 public static DBHandler getInstance()
72 {
73     return instance;
74 }//end method getInstance
75
76 /**
77  * Get one connection for DB
78  * @return actual connection or null
79  * @throws SQLException after you...watch out
80  */
81 public Connection getConnection()
82 {
83     try
84     {
85         Class.forName(addressToDriver);
86         dbconn = DriverManager.getConnection(connectionString,dbUser,dbPassword);
87         return dbconn;
88     }//end try
89     catch(SQLException sqlex)
90     {
91         FileWriterException.writeLogFile(DBHandler.class.getName() + sqlex.getMessage());
92         return null;
93     }//end catch
94     catch (ClassNotFoundException cnfex)
95     {
96         FileWriterException.writeLogFile(DBHandler.class.getName() + cnfex.getMessage());
97         return null;
98     }//end catch
99
100
101
102 }//end method getConnection
103
104 /**
105  * Aurocommit a sql statement in the database
106  * @param state true for autocommitment false for not
107  */
108 public static void defineAutoCommit(boolean state)
109 {
110     try
111     {
112         dbconn.setAutoCommit(state);
113     } // end try

```

```

114     catch(SQLException sqle)
115     {
116         FileWriterException.writeLogFile(DBHandler.class.getName() + " / DefineAutoCommit" +
sqle.getMessage());
117     } // end catch
118 } // end defineAutoCommit method
119
120 /**
121  * Execute a commit
122  */
123 public static void exeCommit()
124 {
125     try
126     {
127         dbconn.commit();
128     } // end try
129     catch (SQLException sqle)
130     {
131         FileWriterException.writeLogFile(DBHandler.class.getName() + " /exeCommy " + sqle.getMessage() );
132     } // end catch
133 } // end exeCommit method
134 } // end DBHandler class

```

ObjectHandler

```

1 package tweakmc.dataaccess;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileNotFoundException;
6 import java.io.FileOutputStream;
7 import java.io.IOException;
8 import java.io.InputStream;
9 import java.io.ObjectInputStream;
10 import java.io.ObjectOutputStream;
11 import java.io.OutputStream;
12 import java.io.Serializable;
13 import java.util.logging.Level;
14 import java.util.logging.Logger;
15 import tweakmc.utility.FileWriterException;
16
17 /**
18  * Save and load objects to a specific place on the disc
19  * Also copies files from one place to a specific Vehicle folder for results
20  * @author nn119171
21  */
22 public class ObjectHandler implements Serializable {
23
24     //instance variables
25     private static ObjectHandler instance;
26     private static Object objectRead;
27
28     /**
29      * Singleton constructor
30      */
31     private ObjectHandler()
32     {

```

```

33
34 }//end ObjectHandler constructor
35
36
37 /**
38  * Singleton instance
39  * @return instance
40  */
41 public static ObjectHandler getInstance()
42 {
43     if(instance == null)
44     {
45         instance = new ObjectHandler();
46     }
47     return instance;
48 }//end getInstance method
49
50 /**
51  * Writes the pobject to the given position
52  * @param <T> type of object to write
53  * @param objectToSave the object to be written in file
54  * @param path
55  * @return true if saved else false
56  */
57 public static <T> boolean objectWriter(T objectToSave, String path)
58 {
59     ObjectOutputStream oos;
60     try
61     {
62         oos = new ObjectOutputStream(new FileOutputStream(path));
63         oos.writeObject(objectToSave);
64         oos.close();
65
66         return true;
67     }//end try
68     catch (IOException ex)
69     {
70         Logger.getLogger(ObjectHandler.class.getName()).log(Level.SEVERE, null, ex);
71         FileWriterException.writeLogFile(ObjectHandler.class.getName() + " / objectWriter / " + ex.getMessage());
72         return false;
73     }//end catch
74 }//end objectWriter method
75
76 /**
77  * Read and return the choosen Object. Object is returnd as object and must be cast before use in GUI
78  * @param path to read
79  * @return Object if found and read else NULL
80  */
81 public static Object objectReader(String path)
82 {
83     ObjectInputStream ois;
84     try
85     {
86         ois = new ObjectInputStream(new FileInputStream(path));
87         return objectRead = ois.readObject();
88     }//end try
89
90     catch (IOException ex)

```

```

91     {
92         FileWriterException.writeLogFile(ObjectHandler.class.getName() + " /objectReader/ " + ex.getMessage());
93         return null;
94     } //end catch
95
96     catch (ClassNotFoundException ex)
97     {
98         FileWriterException.writeLogFile(ObjectHandler.class.getName() + " /objectReader/ " + ex.getMessage());
99         return null;
100    } //end catch
101 } //end method objectReader
102
103 /**
104  * Copy the original file to the vehiclefolder there it is attached
105  * @param originalDestination to copy from
106  * @param newDestination copy to
107  * @return true if copied else false
108  */
109 public static boolean copyFileToNewDestination(String originalDestination, String newDestination)
110 {
111     InputStream in = null;
112     OutputStream out = null;
113     boolean fileMoved = false;
114
115     try
116     {
117         in = new FileInputStream(new File(originalDestination));
118         out = new FileOutputStream(new File(newDestination));
119
120         byte[] buffer = new byte[1024];
121         int len;
122
123         while((len = in.read(buffer)) > 0)
124         {
125             out.write(buffer, 0, len);
126         } //end while
127
128         in.close();
129         out.close();
130
131         fileMoved = true;
132     } //end try
133
134     catch (FileNotFoundException ex)
135     {
136         FileWriterException.writeLogFile(ObjectHandler.class.getName() +
137             " /copyFileToNewDestination/ " + ex.getMessage());
138         fileMoved = false;
139     } //end catch
140
141     catch (IOException ex)
142     {
143         FileWriterException.writeLogFile(ObjectHandler.class.getName() +
144             " /copyFileToNewDestination/ " + ex.getMessage());
145         fileMoved = false;
146     } //end catch
147     return fileMoved;
148 } //end method copyFileToNewDestination(String originalDestination, String newDestination)

```

```
149 }//end class ObjectHandler
```

OrderDAO

```
1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.text.SimpleDateFormat;
7 import java.util.ArrayList;
8 import tweakmc.model.Order;
9 import tweakmc.utility.FileWriterException;
10 import tweakmc.view.OrderGUI;
11
12 /**
13  * Connection between the system and the the Database
14  * Transforming java code to SQL
15  * @author nn119171
16  */
17 public class OrderDAO extends DAO{
18
19     //Instance variables
20     private static OrderDAO instance;
21
22     private static class NyvangIsAsqlNoobException extends Exception
23     {
24         public NyvangIsAsqlNoobException()
25         {
26             //nothing can be done about that
27         }
28     }
29     OrderGUI og = new OrderGUI();
30
31     /**
32      * Private constructor for VehicleDAO
33      */
34     private OrderDAO()
35     {
36
37     }//end constructor
38
39     /**
40      * Singleton method for OrderDAO
41      * @return instance of OrderDAO
42      */
43     public static OrderDAO getInstance()
44     {
45         if(instance == null)
46         {
47             instance = new OrderDAO();
48         }//end if
49         return instance;
50     }//end getInstance method
51
52     /**
53      * Create Order in the DB
54      * @param orderType
55      * @param descriptionOrder
```

```

56  * @param descriptionSpareparts
57  * @param startDate
58  * @param expEndDate
59  * @param expPrice
60  * @param finishPrice
61  * @return
62  */
63  public String createOrder(String orderType, String descriptionOrder,
64      String descriptionSpareparts, int startDate,int expEndDate, String expPrice, String finishPrice)
65  {
66      try
67      {
68          int orderID = getHighestOrderIDfromDatabase();
69          orderID++;
70          String str_orderID = "C" + orderID;
71          PreparedStatement pstmt = getPreparedStatement("INSERT INTO vorder (vorderID, orderType,
descriptionOrder, descriptionSpareparts, startDate, expEndDate, expPrice, finishPrice) VALUES(?,?,?,?,?,?,?,?)");
72          pstmt.setString(1, str_orderID);
73          pstmt.setString(2, orderType);
74          pstmt.setString(3, descriptionOrder);
75          pstmt.setString(4, descriptionSpareparts);
76          pstmt.setInt(5, startDate);
77          pstmt.setInt(6, expEndDate);
78          pstmt.setString(7, expPrice);
79          pstmt.setString(8, finishPrice);
80          pstmt.execute();
81
82          return str_orderID;
83
84      } //end try
85      catch (SQLException sqllex)
86      {
87          FileWriterException.writeLogFile(OrderDAO.class.getName() + " /createOrder/ " + sqllex.getMessage());
88          return "";
89      } //end catch
90      finally
91      {
92          close();
93      } //end finally
94  } //end createOrder method
95
96  /**
97   * Excecutes the search order critrias into the DB
98   * @param <T>
99   * @param searchCriteria the searchcriteria
100  * @return resultFound
101  */
102  public<T> ArrayList<Order> searchOrder(T searchCriteria)
103  {
104      SortedSet<Order> resultFoundNoDublicates = new TreeSet<Order>();
105      ArrayList<Order> resultFound = new ArrayList<Order>();
106      ResultSet rs = null;
107
108      try
109      {
110          Class c = searchCriteria.getClass();
111          if(c.toString().equals("class java.lang.String"))
112          {

```

```

113
114     PreparedStatement pstmt = getPreparedStatement("SELECT * FROM VORDER WHERE "
115         + "LOWER(vorderID) LIKE LOWER(?) "
116         + "OR LOWER(orderType) LIKE LOWER(?) "
117         + "OR LOWER(descriptionOrder) LIKE LOWER(?) "
118         + "OR LOWER(descriptionSpareparts) LIKE LOWER(?) "
119         + "OR LOWER(startDate) LIKE LOWER(?) "
120         + "OR LOWER(expEndDate) LIKE LOWER(?) "
121         + "OR LOWER(expPrice) LIKE LOWER(?) "
122         + "OR LOWER(finishprice) LIKE LOWER(?)");
123     pstmt.setString(1, searchCriteria.toString());
124     pstmt.setString(2, searchCriteria.toString());
125     pstmt.setString(3, searchCriteria.toString());
126     pstmt.setString(4, searchCriteria.toString());
127     pstmt.setString(5, searchCriteria.toString());
128     pstmt.setString(6, searchCriteria.toString());
129     pstmt.setString(7, searchCriteria.toString());
130     pstmt.setString(8, searchCriteria.toString());
131     rs = pstmt.executeQuery();
132
133     while(rs.next())
134     {
135         String vorderID = rs.getString("vorderID");
136         String orderType = rs.getString("orderType");
137         String descriptionOrder = rs.getString("descriptionOrder");
138         String descriptionSpareparts = rs.getString("descriptionSpareparts");
139         int startDate = rs.getInt("startDate");
140         int expEndDate = rs.getInt("expEndDate");
141         String expPrice = rs.getString("expPrice");
142         String finishprice = rs.getString("finishprice");
143         Order o = new Order(vorderID, orderType, descriptionOrder, descriptionSpareparts, startDate,
expEndDate, expPrice, finishprice);
144         resultFound.add(o);
145     } //end while
146 } //end if
147 else
148 {
149
150     String sCstring = (String) searchCriteria;
151     int sC = Integer.parseInt(sCstring);
152     PreparedStatement pstmt = getPreparedStatement("SELECT * FROM VORDER WHERE orderID = ?");
153     pstmt.setInt(1, sC);
154     rs = pstmt.executeQuery();
155     while(rs.next())
156     {
157         String vorderID = rs.getString("vorderID");
158         String orderType = rs.getString("orderType");
159         String descriptionOrder = rs.getString("descriptionOrder");
160         String descriptionSpareparts = rs.getString("descriptionSpareparts");
161         int startDate = rs.getInt("startDate");
162         int expEndDate = rs.getInt("expEndDate");
163         String expPrice = rs.getString("expPrice");
164         String finishPrice = rs.getString("finishprice");
165         Order o = new Order(vorderID, orderType, descriptionOrder, descriptionSpareparts, startDate,
expEndDate, expPrice, finishPrice);
166         resultFound.add(o);
167     } //end while
168 } //end else

```



```

169     } //end try
170     catch (SQLException sqlex)
171     {
172         FileWriterException.writeLogFile(OrderDAO.class.getName() + sqlex.getMessage());
173     } //end catch
174     finally
175     {
176         close();
177     } //end finally
178     return resultFound;
179 } //end searchOrder
180
181 /**
182  * Update order method
183  * @param orderID
184  * @param orderType
185  * @param descriptionOrder
186  * @param descriptionSpareparts
187  * @param startDate
188  * @param expEndDate
189  * @param expPrice
190  * @param finishPrice
191  * @return true/false
192  */
193 public boolean updateOrder(String vorderID, String orderType, String descriptionOrder, String
descriptionSpareparts, int startDate, int expEndDate, String expPrice, String finishPrice)
194 {
195     try
196     {
197         PreparedStatement pstmt = getPreparedStatement("UPDATE vorder SET "
198             + "orderType=?, "
199             + "descriptionOrder=?, "
200             + "descriptionSpareparts=?, "
201             + "startDate=?, "
202             + "expEndDate=?, "
203             + "expPrice=?, "
204             + "finishPrice=? "
205             + "WHERE orderID = ?");
206         pstmt.setString(1, orderType);
207         pstmt.setString(2, descriptionOrder);
208         pstmt.setString(3, descriptionSpareparts);
209         pstmt.setInt(4, startDate);
210         pstmt.setInt(5, expEndDate);
211         pstmt.setString(6, expPrice);
212         pstmt.setString(7, finishPrice);
213         pstmt.setString(8, vorderID);
214         pstmt.execute();
215         return true;
216     } // End try
217
218     catch(SQLException sqlex)
219     {
220         FileWriterException.writeLogFile(OrderDAO.class.getName() + sqlex.getMessage());
221         return false;
222     } // End Catch
223     finally
224     {
225         close();

```

```

226     } // End Finally
227 }
228
229 /**
230  * Generate a list over all orders starting or ending within the next seven days
231  * Dates are initially calculated in milliseconds so they are easier to handle (add/subtract and store)
232  * //Possible errors: The method are dependent of the system time of the computer.
233  * @return resultFoundStart resultFoundStart An ArrayList containing the starting orders
234  */
235 public ArrayList<Order> generateTodo()
236 {
237     //Initialize the ArrayLists "resultFoundStart" and the ResultSet "rs"
238     ArrayList<Order> resultFoundStart = new ArrayList<Order>(); //initializes the list for holding objects that
start within the timegap
239     ResultSet rs = null;
240
241
242     //Get the current time, convert it to millis and do the math for the 7 days
243     SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd"); //create a SQL freindly format
244     long timeInMillis = System.currentTimeMillis(); //get the current time in milliseconds
245     final long sevenDays = 604800000; // seven days in milliseconds
246     final long nowPlusSevenDays = timeInMillis + sevenDays; // add the above and get the date (in milliseconds)
in 7 days
247     int todayInt = Integer.parseInt(sdf.format(timeInMillis)); // parse the milisecond to the choosen date format
(yyyMMdd)
248     int sevenDaysInt = Integer.parseInt(sdf.format(nowPlusSevenDays)); // parse the milisecond to the choosen
date format (yyyMMdd)
249
250     try
251     {
252         //Searches for the orders, starting within the timegap and sorts the orders by primarily startDate and second
orderId
253         PreparedStatement pstmt = getPreparedStatement("SELECT * FROM VORDER WHERE startDate OR
expEndDate BETWEEN ? AND ? ORDER BY startDate ASC, orderId ASC");
254         pstmt.setInt(1, todayInt);
255         pstmt.setInt(2, sevenDaysInt);
256         rs = pstmt.executeQuery();
257         while(rs.next())
258         {
259             String vorderID = rs.getString("vorderID");
260             String orderType = rs.getString("orderType");
261             String descriptionOrder = rs.getString("descriptionOrder");
262             String descriptionSpareparts = rs.getString("descriptionSpareparts");
263             int startDate = rs.getInt("startDate");
264             int expEndDate = rs.getInt("expEndDate");
265             String expPrice = rs.getString("expPrice");
266             String finishPrice = rs.getString("finishprice");
267             Order o = new Order(vorderID, orderType, descriptionOrder, descriptionSpareparts, startDate,
expEndDate, expPrice, finishPrice);
268             resultFoundStart.add(o);
269         } //end while
270
271     } //end try
272     catch (SQLException sqllex)
273     {
274         FileWriterException.writeLogFile(OrderDAO.class.getName() + sqllex.getMessage());
275     } //end catch
276

```

```

277     finally
278     {
279         close();
280     } //end finally
281
282     return resultFoundStart;
283
284 }
285
286 /**
287  * Generate a list over all orders ending within the next seven days
288  * Dates are initially calculated in miliseconds so they are easier to handle (add/subtract and store)
289  * //Possible errors: The method are dependent of the system time of the computer.
290  * @return resultFoundEnd resultFoundEnd An ArrayList containing the ending orders
291  */
292 public ArrayList<Order> generateTodoEnd()
293 {
294     //Initialize the ArrayList "resultFoundEnd" and the ResultSet "rs"
295     ArrayList<Order> resultFoundEnd = new ArrayList<Order>(); //initializes the list for holding objects that
ends within the time gap
296     ResultSet rs = null;
297
298     //Get the current time, convert it to millis and do the math for the 7 days
299     SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd"); //create a SQL freindly format
300     long timeInMillis = System.currentTimeMillis(); //get the current time in milliseconds
301     final long sevenDays = 604800000; // seven days in miliseconds
302     final long nowPlusSevenDays = timeInMillis + sevenDays; // add the above and get the date (in milliseconds)
in 7 days
303     int todayInt = Integer.parseInt(sdf.format(timeInMillis)); // parse the milisecond to the choosen date format
(yyyMMdd)
304     int sevenDaysInt = Integer.parseInt(sdf.format(nowPlusSevenDays)); // parse the milisecond to the choosen
date format (yyyMMdd)
305
306
307     try
308     {
309         //Searches for the orders, starting within the time gap and sorts the orders by primarily expEndDate and
second orderID
310         PreparedStatement pstmt = getPreparedStatement("SELECT * FROM VORDER WHERE expEndDate
BETWEEN ? AND ? ORDER BY expEndDate ASC, orderID ASC");
311         pstmt.setInt(1, todayInt);
312         pstmt.setInt(2, sevenDaysInt);
313         rs = pstmt.executeQuery();
314         while(rs.next())
315         {
316             String vorderID = rs.getString("vorderID");
317             String orderType = rs.getString("orderType");
318             String descriptionOrder = rs.getString("descriptionOrder");
319             String descriptionSpareparts = rs.getString("descriptionSpareparts");
320             int startDate = rs.getInt("startDate");
321             int expEndDate = rs.getInt("expEndDate");
322             String expPrice = rs.getString("expPrice");
323             String finishPrice = rs.getString("finishprice");
324             Order o = new Order(vorderID, orderType, descriptionOrder, descriptionSpareparts, startDate,
expEndDate, expPrice, finishPrice);
325             resultFoundEnd.add(o);
326         } //end while
327

```

```

328     } //end try
329     catch (SQLException sqllex)
330     {
331         FileWriterException.writeLogFile(OrderDAO.class.getName() + sqllex.getMessage());
332     } //end catch
333
334     finally
335     {
336         close();
337     } //end finally
338     return resultFoundEnd;
339 } //end method generateToDoEnd
340
341 /**
342  * Attach vehicle to order.
343  * @param vehicleID
344  * @param orderID
345  * @return true if success, false otherwise
346  */
347 public boolean attachVehicle(String vehicleID, String vorderID)
348 {
349     boolean returnHowDidItGo = false;
350     try
351     {
352         PreparedStatement pstmt = getPreparedStatement("UPDATE vorder SET vehicleIdentifier=? WHERE
vorderID=?");
353         pstmt.setInt(0, getVehicleIdentifier(vehicleID));
354         pstmt.setString(1, vorderID);
355         pstmt.execute();
356         returnHowDidItGo = true;
357     }
358     catch (SQLException n)
359     {
360         FileWriterException.writeLogFile(OrderDAO.class.getName() + n.getMessage());
361     }
362
363     return returnHowDidItGo;
364 }
365 /**
366  * This method gets the highest number of the identifier from the ordertable
367  * @return int
368  */
369 public int getHighestOrderIDfromDatabase()
370 {
371     try
372     {
373         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM vorder");
374         ResultSet rsr = null;
375         rsr = pstmt.executeQuery();
376         rsr.next();
377         return rsr.getInt(1);
378     } //end try
379     catch (SQLException sqllex)
380     {
381         FileWriterException.writeLogFile(OrderDAO.class.getName() + sqllex.getMessage());
382         return -1;
383     } //end catch
384

```

```

385     finally
386     {
387         close();
388     } //end finally
389 } //end method getHighestOrderIDfromDatabase
390
391
392 } //end class OrderDAO

```

OwnerDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.List;
8 import java.util.SortedSet;
9 import java.util.TreeSet;
10 import tweakmc.utility.FileWriterException;
11 import tweakmc.model.Owner;
12 import tweakmc.model.Vehicle;
13 /**
14  * Connection between the system and the the Database
15  * Transforming java code to SQL
16  * @author NegoZiatoR
17  */
18 public class OwnerDAO extends DAO
19 {
20     //Instance variables
21     private static OwnerDAO instance;
22     private SortedSet<Owner> actFoundsWithoutDuplicates;
23     private ArrayList<Owner> actFounds;
24     private ArrayList<Vehicle> actFoundsVehicle;
25     protected boolean[] activness;
26     private List<Boolean> listActiveness = new ArrayList<Boolean>();
27
28     /**
29      * Private constructor for VehicleDAO
30      */
31     private OwnerDAO()
32     {
33
34     } //end constructor
35
36     /**
37      * Singleton method for OwnerDAO
38      * @return instance of OwnerDAO
39      */
40     public static OwnerDAO getInstance()
41     {
42         if(instance == null)
43         {
44             instance = new OwnerDAO();
45         } //end if
46         return instance;

```

```

47  }//end getInstance method
48
49  /**
50   * Create owner to db
51   * @param firstName
52   * @param lastName on owner
53   * @param address on owner
54   * @param address2 on owner
55   * @param zip on owner
56   * @param city on owner
57   * @param country
58   * @param phone on owner
59   * @param email on owner
60   * @return true if success else false
61   */
62  public boolean createActOwner(String firstName, String lastName, String address, String address2, String zip,
String city, String country, String phone, String email)
63  {
64      try
65      {
66          int ownerID = getHighestOwnerIDfromDatabase();
67          ownerID++;
68          String str_ownerID = "O" + ownerID;
69          PreparedStatement pstmt = getPreparedStatement("INSERT INTO OWNER (OWNERID, FIRSTNAME,
LASTNAME, ADDRESS, ADDRESS2, ZIP, CITY, COUNTRY, PHONE, EMAIL) VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?,
?);");
70          pstmt.setString(1, str_ownerID);
71          pstmt.setString(2, firstName);
72          pstmt.setString(3, lastName);
73          pstmt.setString(4, address);
74          pstmt.setString(5, address2);
75          pstmt.setString(6, zip);
76          pstmt.setString(7, city);
77          pstmt.setString(8, country);
78          pstmt.setString(9, phone);
79          pstmt.setString(10, email);
80          pstmt.execute();
81          return true;
82      }// end try
83
84      catch (SQLException sqlex)
85      {
86          FileWriterException.writeLogFile(OwnerDAO.class.getName()
87              + sqlex.getMessage());
88          return false;
89      }// end catch
90
91      finally
92      {
93          close();
94      }//end finally
95  }// end method createActOwner
96
97  /**
98   * Search for owner with given searchcriteria
99   * @param <T> generic type for searchcriteria
100  * @param searchCriteria for owner
101  * @return list with actFounds or null

```

```

102  */
103  public <T> List<Owner> searchActOwner(T searchCriteria)
104  {
105      //Make a sorted list so duplicate results isn't accepted
106      actFoundsWithoutDuplicates = new TreeSet<Owner>();
107      //Make an ArrayList which is returned
108      actFounds = new ArrayList<Owner>();
109      ResultSet rsr = null;
110
111      try
112      {
113          Class c = searchCriteria.getClass();
114          if(c.toString().equals("class java.lang.String"))
115          {
116              if(searchCriteria.toString().contains(" "))
117              {
118                  for (String str : searchCriteria.toString().split(" "))
119                  {
120                      PreparedStatement pstmt = getPreparedStatement("SELECT * FROM OWNER WHERE
LOWER(firstName) LIKE LOWER(?) OR LOWER(lastName) LIKE LOWER(?) OR LOWER(address) LIKE
LOWER(?) OR LOWER(address2) LIKE LOWER(?) OR LOWER(zip) LIKE LOWER(?) OR LOWER(city) =
LOWER(?) OR LOWER(country) LIKE LOWER(?) OR LOWER(phone) = LOWER(?) OR LOWER(email) LIKE
LOWER(?)");
121                      pstmt.setString(1, "%" + (String) str + "%");
122                      pstmt.setString(2, "%" + (String) str + "%");
123                      pstmt.setString(3, "%" + (String) str + "%");
124                      pstmt.setString(4, "%" + (String) str + "%");
125                      pstmt.setString(5, (String) str);
126                      pstmt.setString(6, "%" + (String) str + "%");
127                      pstmt.setString(7, "%" + (String) str + "%");
128                      pstmt.setString(8, (String) str);
129                      pstmt.setString(9, "%" + (String) str + "%");
130                      rsr = pstmt.executeQuery();
131
132                      while(rsr.next())
133                      {
134                          String ownerID = rsr.getString("ownerID");
135                          String firstName = rsr.getString("firstName");
136                          String lastName = rsr.getString("lastName");
137                          String address1 = rsr.getString("address");
138                          String address2 = rsr.getString("address2");
139                          String zip = rsr.getString("zip");
140                          String city = rsr.getString("city");
141                          String country = rsr.getString("country");
142                          String phone = rsr.getString("phone");
143                          String email = rsr.getString("email");
144                          Owner o = new Owner(ownerID, firstName, lastName, address1, address2, zip, city, country,
phone, email);
145                          String firstNameAndLastName = o.getFirstName() + " " + o.getLastName();
146                          if(firstNameAndLastName.equals(searchCriteria))
147                              actFoundsWithoutDuplicates.add(o);
148
149                      } //end while
150
151                  } //end for-each
152                  actFounds.addAll(actFounds);
153              } //end if
154          }

```

```

155         else
156         {
157             PreparedStatement pstmt = getPreparedStatement("SELECT * FROM OWNER WHERE
LOWER(firstName) LIKE LOWER(?) OR LOWER(lastName) LIKE LOWER(?) OR LOWER(address) LIKE
LOWER(?) OR LOWER(address2) LIKE LOWER(?) OR LOWER(zip) = LOWER(?) OR LOWER(city) LIKE
LOWER(?) OR LOWER(country) LIKE LOWER(?) OR LOWER(phone) = LOWER(?) OR LOWER(email) LIKE
LOWER(?)");
158             pstmt.setString(1, "%" + (String) searchCriteria + "%");
159             pstmt.setString(2, "%" + (String) searchCriteria + "%");
160             pstmt.setString(3, "%" + (String) searchCriteria + "%");
161             pstmt.setString(4, "%" + (String) searchCriteria + "%");
162             pstmt.setString(5, (String) searchCriteria);
163             pstmt.setString(6, "%" + (String) searchCriteria + "%");
164             pstmt.setString(7, "%" + (String) searchCriteria + "%");
165             pstmt.setString(8, (String) searchCriteria);
166             pstmt.setString(9, "%" + (String) searchCriteria + "%");
167             rsr = pstmt.executeQuery();
168
169             while(rsr.next())
170             {
171                 String ownerID = rsr.getString("ownerID");
172                 String firstName = rsr.getString("firstName");
173                 String lastName = rsr.getString("lastName");
174                 String address1 = rsr.getString("address");
175                 String address2 = rsr.getString("address2");
176                 String zip = rsr.getString("zip");
177                 String city = rsr.getString("city");
178                 String country = rsr.getString("country");
179                 String phone = rsr.getString("phone");
180                 String email = rsr.getString("email");
181                 Owner o = new Owner(ownerID, firstName, lastName, address1, address2, zip, city, country, phone,
email);
182                 actFoundsWithoutDuplicates.add(o);
183             } //end while
184         } //end else
185         actFounds.addAll(actFoundsWithoutDuplicates);
186     } //end if
187
188     else
189     {
190         PreparedStatement pstmt = getPreparedStatement("SELECT * FROM OWNER WHERE ownerID = ?");
191         pstmt.setInt(1, (Integer) searchCriteria);
192         rsr = pstmt.executeQuery();
193
194         while(rsr.next())
195         {
196             String ownerID = rsr.getString("ownerID");
197             String firstName = rsr.getString("firstName");
198             String lastName = rsr.getString("lastName");
199             String address = rsr.getString("address");
200             String address2 = rsr.getString("address2");
201             String zip = rsr.getString("zip");
202             String city = rsr.getString("city");
203             String country = rsr.getString("country");
204             String phone = rsr.getString("phone");
205             String email = rsr.getString("email");
206             Owner o = new Owner(ownerID, firstName, lastName, address, address2, zip, city, country, phone,
email);

```



```

207         actFounds.add(o);
208     } //end while
209 } //end else
210
211
212 } //end try
213
214 catch (SQLException sqlex)
215 {
216     FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());
217 } //end catch
218
219 finally
220 {
221     close();
222 } //end finally
223
224 return actFounds;
225 } //end method searchActOwner
226
227 /**
228  * Search for an owner and return vehicles he owns
229  * @param ownerID
230  * @return List with vehicles owned by owner
231  */
232 public List<Vehicle> searchActOwnerWithVehicles(String ownerID)
233 {
234     //Make an ArrayList with owner
235     actFounds = new ArrayList<Owner>(searchActOwner(ownerID));
236     // Make an ArrayList for found vehicles
237     actFoundsVehicle = new ArrayList<Vehicle>();
238     int ownerIdentifier = getOwnerIdentifier(ownerID);
239     try
240     {
241         PreparedStatement pstm = getPreparedStatement("SELECT activeOwnerShip,
o.vehicleidentifier,v.identifier, v.vehicleID, vt.SUGGESTIONS as vType, vb.SUGGESTIONS as brand,
vm.SUGGESTIONS as model, v.vYear, v.chassisNumber, v.licenseplate "
242             + "FROM ownership o, vehicle v, vehicleType vt, vehicleBrand vb, vehicleModel vm "
243             + "WHERE (o.owneridentifier = ?) AND "
244             + "(v.vtype = vt.identifier AND "
245             + "v.brand = vb.identifier AND "
246             + "v.model = vm.identifier) "
247             + "AND o.vehicleidentifier = v.identifier");
248
249         pstm.setInt(1, ownerIdentifier);
250         pstm.execute();
251
252         ResultSet rsr = null;
253         rsr = pstm.executeQuery();
254         int activecounter = 0; //This counter is used to go through the ArrayList listactiveness
255         listActiveness.clear(); //Clear the list if a previous search was performed
256         while(rsr.next())
257         {
258             boolean ownerActive = rsr.getBoolean("activeOwnerShip");
259             String vehicleID = rsr.getString("vehicleID");
260             String vType = rsr.getString("vType");
261             String brand = rsr.getString("brand");
262             String model = rsr.getString("model");

```

```

263         int vYear = rsr.getInt("vYear");
264         String chassisNumber = rsr.getString("chassisNumber");
265         String licensPlate = rsr.getString("licensPlate");
266         Vehicle v = new Vehicle(vehicleID, vType, brand, model, vYear, chassisNumber, licensPlate);
267         actFoundsVehicle.add(v); //Add a vehicle to the ArrayList which is returned
268         listActiveness.add(ownerActive); //Add the value of ActiveOwnership to an ArrayList
269         activecounter++;
270     } //end while
271
272     } //end try
273     catch(SQLException sqlex)
274     {
275         FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());
276     } //end catch
277     //Make a new array from the arraylist listactiveness
278     activness = new boolean[listActiveness.size()];
279     int arrCounter = 0;
280     for (boolean b : listActiveness)
281     {
282         activness[arrCounter]=b;
283         arrCounter++;
284     } //end for each
285
286     return actFoundsVehicle;
287 } //end method searchActOwnerWithVehicles
288
289
290 /**
291  * Edit existing owner
292  * @param ownerID owners ID
293  * @param firstName on owner
294  * @param lastName on owner
295  * @param address on owner
296  * @param address2 on owner
297  * @param zip on owner
298  * @param city on owner
299  * @param country
300  * @param phone on owner
301  * @param email on owner
302  * @return true if success else false
303  */
304 public boolean editActOwner(String ownerID, String firstName, String lastName, String address, String
address2, String zip, String city, String country, String phone, String email)
305 {
306     try
307     {
308         PreparedStatement pstm = getPreparedStatement("UPDATE owner SET firstName = ?, lastName = ?,
address = ?, address2 = ?, zip = ?, city = ?, country = ?, phone = ?, email = ? where ownerID = ?");
309         pstm.setString(1, firstName);
310         pstm.setString(2, lastName);
311         pstm.setString(3, address);
312         pstm.setString(4, address2);
313         pstm.setString(5, zip);
314         pstm.setString(6, city);
315         pstm.setString(7, country);
316         pstm.setString(8, phone);
317         pstm.setString(9, email);
318         pstm.setString(10, ownerID);

```

```

319     pstmt.execute();
320     return true;
321 } // End try
322 catch(SQLException sqlex)
323 {
324     FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());
325     return false;
326 } // End Catch
327 finally
328 {
329     close();
330 } // End Finally
331
332
333 }// End editActOwner Method
334
335 /**
336  * Delete an owner
337  * @param ownerID on owner
338  * @return true if deleted else false
339  */
340 public boolean deleteActOwner(String ownerID)
341 {
342     try
343     {
344         PreparedStatement pstmt = getPreparedStatement("DELETE FROM owner WHERE ownerID = ?");
345         pstmt.setString(1, ownerID);
346         pstmt.execute();
347         return true;
348     } // End try
349     catch(SQLException sqlex)
350     {
351         FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());
352         return false;
353     } // End Catch
354     finally
355     {
356         close();
357     } // End Finally
358 }
359
360 /**
361  * This method gets the highest number of the identifier from the ownertable
362  * @return int
363  */
364 public int getHighestOwnerIDfromDatabase()
365 {
366     try
367     {
368         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM owner");
369         ResultSet rsr = null;
370         rsr = pstmt.executeQuery();
371         rsr.next();
372         return rsr.getInt(1);
373     } //end try
374     catch (SQLException sqlex)
375     {
376         FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());

```

```

377         return -1;
378     } //end catch
379
380     finally
381     {
382         close();
383     } //end finally
384 } //end method getHighestOwnerIDfromDatabase
385
386 /**
387  * Gets an array with active-statuses on ownership
388  * @return boolean[]
389  */
390 public boolean[] getActiveNess()
391 {
392     return activness;
393 }
394
395 } // End OwnerDAO class

```

ResultDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.Vector;
7 import tweakmc.model.Result.Result;
8 import tweakmc.utility.FileWriterException;
9
10 /**
11  * Connection between the system and the the Database
12  * Transforming java code to SQL
13  * @author NegoZiatoR
14  */
15 public class ResultDAO extends DAO
16 {
17     // Instance variables
18     private static ResultDAO instance;
19
20
21     /**
22      * Private constructor for ResultDAO
23      */
24     private ResultDAO()
25     {
26
27     } //end constructor
28
29     /**
30      * Singleton mehtod for ResultDAO
31      * @return ONE instance of resultDAO
32      */
33     public static ResultDAO getInstance()
34     {
35         if(instance == null)

```

```

36     {
37         instance = new ResultDAO();
38     } //end if
39     return instance;
40 } //end method getInstance
41
42 /**
43  * Create a result
44  * @param resultType type of result
45  * @return true if created else false
46  */
47 public boolean createNewActResult(Result resultType)
48 {
49     boolean resultCreated = false;
50
51     Result result = resultType;
52     try
53     {
54         //Create a ID for this result
55         int resultID = getHighestResultIDfromDatabase();
56         resultID++;
57         String str_resultID = "R" + resultID;
58
59         //Create the SQL statement
60         PreparedStatement pstmt = getPreparedStatement("INSERT INTO result (resultID, mechanicComment,
customerComment, resultName, timeStamp, resultType, path, vehicleIdentifier, workstationIdentifier, orderIdentifier,
whiteBoardIdentifier) values (?,?,?,?,?,?,?,?,?,?,?)");
61         pstmt.setString(1, str_resultID);
62         pstmt.setString(2, result.getMechanicComment());
63         pstmt.setString(3, result.getCustomerComment());
64         pstmt.setString(4, result.getResultName());
65         pstmt.setString(5, result.getTimeStamp());
66         pstmt.setString(6, result.getResultType());
67         pstmt.setString(7, result.getPath());
68         pstmt.setInt(8, getIdentifierFromID(result.getVehicleID(), "vehicle"));
69         pstmt.setInt(9, getIdentifierFromID(result.getWorkstationID(), "workstation"));
70
71         if(result.getOrderID().equalsIgnoreCase("C-1"))
72         {
73             pstmt.setNull(10, java.sql.Types.NULL);
74         } //end if
75
76         else
77         {
78             pstmt.setInt(10, getIdentifierFromID(result.getOrderID(), "vorder"));
79
80         } //end else
81
82         if(result.getWhiteboardID().equalsIgnoreCase("W-1"))
83         {
84             pstmt.setNull(11, java.sql.Types.NULL);
85         } //end if
86
87         else
88         {
89             pstmt.setInt(11, getIdentifierFromID(result.getWhiteboardID(), "whiteboard"));
90
91         } //end else

```

```

92         pstmt.execute();
93         resultCreated = true;
94
95     } // End try
96
97     catch(SQLException sql)
98     {
99         FileWriterException.writeLogFile(ResultDAO.class.getName() + " / createActResult/ " +
sql.getMessage());
100         resultCreated = false;
101     } // End catch
102
103     finally
104     {
105         close();
106     } // end finally
107     return resultCreated;
108 } //end method createActResult(T resultType)
109
110 /**
111  * Create a new whiteboard result
112  * @param whiteboardResultType
113  * @return true if created else false
114  */
115 public boolean createActWhiteboard(Result whiteboardResultType)
116 {
117     try
118     {
119         PreparedStatement pstmt = getPreparedStatement("");
120         return true;
121     } // End try
122     catch(SQLException sql)
123     {
124         FileWriterException.writeLogFile(ResultDAO.class.getName() + " / createActResult/ " + sql.getMessage());
125         return false;
126     } // End catch
127     finally
128     {
129         close();
130     } // end finally
131 } //end method createActWhiteCoard
132
133 /**
134  * Build a vector with results found from the given IDS
135  * @param actVehicleID to find results for
136  * @param actWorkstationID to find results for
137  * @return Vector with result or null
138  */
139 public Vector<Result> getResultsForIDs(String actVehicleID, String actWorkstationID)
140 {
141     Vector<Result> foundResultsForIDs = new Vector<Result>();
142     try
143     {
144         PreparedStatement pstmt = getPreparedStatement("SELECT * FROM result WHERE "
145             + "vehicleIdentifier = ? AND workstationIdentifier = ?");
146         int vid = (getIdentifierFromID(actVehicleID, "vehicle"));
147         int wid = (getIdentifierFromID(actWorkstationID, "workstation"));
148         pstmt.setInt(1, getIdentifierFromID(actVehicleID, "vehicle"));

```

```

149     pstmt.setInt(2, getIdentifierFromID(actWorkstationID, "workstation"));
150     ResultSet rs = pstmt.executeQuery();
151
152     while(rs.next())
153     {
154         String resultID = rs.getString("resultID");
155         String mechanicComment = rs.getString("mechaniccomment");
156         String customerComment = rs.getString("customercomment");
157         String resultName = rs.getString("resultname");
158         String timeStamp = rs.getString("timeStamp");
159         String resultType = rs.getString("resulttype");
160         String path = rs.getString("path");
161         String vehicleID = getIDFromIdentifier(rs.getInt("vehicleIdentifier"), "vehicle");
162         String workstationID = getIDFromIdentifier(rs.getInt("workstationIdentifier"), "workstation");
163         String orderID = "C-1";
164         if(rs.getInt("orderIdentifier") != java.sql.Types.NULL)
165         {
166             orderID = getIDFromIdentifier(rs.getInt("orderIdentifier"), "vorder");
167         }
168         String whiteboardID = "";
169         if(rs.getInt("whiteboardIdentifier") != java.sql.Types.NULL)
170         {
171             whiteboardID = getIDFromIdentifier(rs.getInt("whiteboardIdentifier"), "whiteboard");
172         } //end if
173
174         Result res = new Result(resultID, mechanicComment, customerComment, resultName, timeStamp,
resultType, path, vehicleID, workstationID, orderID, whiteboardID);
175         foundResultsForIDs.add(res);
176     } //end while
177 } //end try
178
179 catch (SQLException ex)
180 {
181     FileWriterException.writeLogFile(ResultDAO.class.getName() + " /getResultsForIDs" + ex.getMessage());
182 } //end catch
183
184 finally
185 {
186     close();
187     return foundResultsForIDs;
188 } //end finally
189 } //end method getResultsForIDs(String actVehicleID, String actWorkstationID)
190
191 /**
192  * This method will get the highest identifier for result in the database
193  * @return int
194  */
195 public int getHighestResultIDfromDatabase()
196 {
197     try
198     {
199         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM result");
200         ResultSet rsr = null;
201         rsr = pstmt.executeQuery();
202         rsr.next();
203         return rsr.getInt(1);
204     } //end try
205     catch (SQLException sqlex)

```

```

206     {
207         FileWriterException.writeLogFile(ResultDAO.class.getName() + " /getHighestResultIDfromDatabase" +
sqllex.getMessage());
208         return -1;
209     } //end catch
210
211     finally
212     {
213         close();
214     } //end finally
215 } //end method getHighestResultIDfromDatabase
216 } //end class ResultDAO

```

SuperSearchDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.List;
8 import java.util.SortedSet;
9 import java.util.TreeSet;
10 import tweakmc.model.Vehicle;
11 import tweakmc.model.Owner;
12 import tweakmc.utility.FileWriterException;
13
14 /**
15  * Connection between the system and the the Database
16  * Transforming java code to SQL
17  * @author nn119171
18  */
19 public class SuperSearchDAO extends DAO {
20
21     //Instance variables
22     private static SuperSearchDAO instance;
23     private SortedSet<Owner> actFoundsWithoutDuplicates;
24     private ArrayList<Owner> actFounds;
25     private List<Boolean> listActiveness = new ArrayList<Boolean>();
26     protected boolean[] activness;
27     // Supersearch Lists
28     private ArrayList<Owner> actFoundOwners;
29     private ArrayList<Vehicle> actFoundVehicles;
30
31
32     /**
33      * Private constructor for restricting other classes use
34      */
35     private SuperSearchDAO()
36     {
37
38     } //end constructor
39
40     /**
41      * Method for implementing the Singleton pattern
42      * @return instance

```


[illegible]

```

97         String country = rsr.getString("country");
98         String phone = rsr.getString("phone");
99         String email = rsr.getString("email");
100        Owner o = new Owner(ownerID, firstName, lastName, address1, address2, zip, city, country,
phone, email);
101        String firstNameAndLastName = o.getFirstName() + " " + o.getLastName();
102        if(firstNameAndLastName.equals(searchCriteria))
103            actFoundsWithoutDuplicates.add(o);
104
105        } //end while
106
107        } //end for-each
108        actFounds.addAll(actFounds);
109    } //end if
110
111    else
112    {
113        PreparedStatement pstm = getPreparedStatement("SELECT * FROM OWNER WHERE
LOWER(firstName) LIKE LOWER(?) OR LOWER(lastName) LIKE LOWER(?) OR LOWER(address) LIKE
LOWER(?) OR LOWER(address2) LIKE LOWER(?) OR LOWER(zip) = LOWER(?) OR LOWER(city) LIKE
LOWER(?) OR LOWER(country) LIKE LOWER(?) OR LOWER(phone) = LOWER(?) OR LOWER(email) LIKE
LOWER(?)");
114        pstm.setString(1, "%" + (String) searchCriteria + "%");
115        pstm.setString(2, "%" + (String) searchCriteria + "%");
116        pstm.setString(3, "%" + (String) searchCriteria + "%");
117        pstm.setString(4, "%" + (String) searchCriteria + "%");
118        pstm.setString(5, (String) searchCriteria);
119        pstm.setString(6, "%" + (String) searchCriteria + "%");
120        pstm.setString(7, "%" + (String) searchCriteria + "%");
121        pstm.setString(8, (String) searchCriteria);
122        pstm.setString(9, "%" + (String) searchCriteria + "%");
123        rsr = pstm.executeQuery();
124
125        while(rsr.next())
126        {
127            String ownerID = rsr.getString("ownerID");
128            String firstName = rsr.getString("firstName");
129            String lastName = rsr.getString("lastName");
130            String address1 = rsr.getString("address");
131            String address2 = rsr.getString("address2");
132            String zip = rsr.getString("zip");
133            String city = rsr.getString("city");
134            String country = rsr.getString("country");
135            String phone = rsr.getString("phone");
136            String email = rsr.getString("email");
137            Owner o = new Owner(ownerID, firstName, lastName, address1, address2, zip, city, country, phone,
email);
138            actFoundsWithoutDuplicates.add(o);
139        } //end while
140    } //end else
141    actFounds.addAll(actFoundsWithoutDuplicates);
142 } //end if
143
144 else
145 {
146     PreparedStatement pstm = getPreparedStatement("SELECT * FROM OWNER WHERE ownerID = ?");
147     pstm.setInt(1, (Integer) searchCriteria);
148     rsr = pstm.executeQuery();

```

```

149
150     while(rsr.next())
151     {
152         String ownerID = rsr.getString("ownerID");
153         String firstName = rsr.getString("firstName");
154         String lastName = rsr.getString("lastName");
155         String address = rsr.getString("address");
156         String address2 = rsr.getString("address2");
157         String zip = rsr.getString("zip");
158         String city = rsr.getString("city");
159         String country = rsr.getString("country");
160         String phone = rsr.getString("phone");
161         String email = rsr.getString("email");
162         Owner o = new Owner(ownerID, firstName, lastName, address, address2, zip, city, country, phone,
email);
163         actFounds.add(o);
164     } //end while
165 } //end else
166
167
168 } //end try
169
170 catch (SQLException sqlex)
171 {
172     FileWriterException.writeLogFile(OwnerDAO.class.getName() + sqlex.getMessage());
173 } //end catch
174
175 finally
176 {
177     close();
178 } //end finally
179
180 return actFounds;
181 } //end method searchActOwner
182
183 /**
184  * Search for vehicle with given searchcriteria
185  * @param <T> generic type for searchcriteria
186  * @param searchCriteria for vehicle
187  * @return list with actFounds or null
188  */
189 public <T> List<Vehicle> searchActVehicle(T searchCriteria)
190 {
191     ArrayList<Vehicle> actFounds = new ArrayList<Vehicle>();
192     ResultSet rsr = null;
193
194     try
195     {
196         Class c = searchCriteria.getClass();
197         if(c.toString().equals("class java.lang.String"))
198         {
199             PreparedStatement pstmt = getPreparedStatement("SELECT v.VEHICLEID, vt.SUGGESTIONS AS
vType, vb.SUGGESTIONS as brand, vm.SUGGESTIONS as model, v.VYEAR, v.CHASSISNUMBER,
v.LICENSPLATE FROM "
200                 + "VEHICLE v, vehicleType vt, vehicleBrand vb, vehicleModel vm "
201                 + "WHERE (LOWER(vt.suggestions) LIKE LOWER(?) OR LOWER(vb.suggestions) LIKE LOWER(?)
OR LOWER(vm.suggestions) LIKE LOWER(?) OR LOWER(chassisNumber) = LOWER(?) OR LOWER(licensPlate)
= LOWER(?)) AND "

```

```

202     + "(v.vtype = vt.identifier AND "
203     + "v.brand = vb.identifier AND "
204     + "v.model = vm.identifier)");
205     pstmt.setString(1, "%" + (String) searchCriteria + "%");
206     pstmt.setString(2, "%" + (String) searchCriteria + "%");
207     pstmt.setString(3, "%" + (String) searchCriteria + "%");
208     pstmt.setString(4, (String) searchCriteria);
209     pstmt.setString(5, (String) searchCriteria);
210     rsr = pstmt.executeQuery();
211
212     while(rsr.next())
213     {
214         String vehicleID = rsr.getString("vehicleID");
215         String vType = rsr.getString("vType");
216         String brand = rsr.getString("brand");
217         String model = rsr.getString("model");
218         int vYear = rsr.getInt("vYear");
219         String chassisNumber = rsr.getString("chassisNumber");
220         String licensPlate = rsr.getString("licensPlate");
221         Vehicle v = new Vehicle(vehicleID, vType, brand, model, vYear, chassisNumber, licensPlate);
222         actFounds.add(v);
223     } //end while
224 } //end if
225
226 else
227 {
228     PreparedStatement pstmt = getPreparedStatement("SELECT v.VEHICLEID, vt.SUGGESTIONS AS
vType, vb.SUGGESTIONS as brand, vm.SUGGESTIONS as model, v.VYEAR, v.CHASSISNUMBER,
v.LICENSPLATE FROM "
229     + "VEHICLE v, vehicleType vt, vehicleBrand vb, vehicleModel vm "
230     + "WHERE (vehicleID = ? OR vYear = ?) AND "
231     + "(v.vtype = vt.identifier AND "
232     + "v.brand = vb.identifier AND "
233     + "v.model = vm.identifier)");
234     pstmt.setInt(1, (Integer) searchCriteria);
235     pstmt.setInt(2, (Integer) searchCriteria);
236     rsr = pstmt.executeQuery();
237
238     while(rsr.next())
239     {
240         String vehicleID = rsr.getString("vehicleID");
241         String vType = rsr.getString("vType");
242         String brand = rsr.getString("brand");
243         String model = rsr.getString("model");
244         int vYear = rsr.getInt("vYear");
245         String chassisNumber = rsr.getString("chassisNumber");
246         String licensPlate = rsr.getString("licensPlate");
247         Vehicle v = new Vehicle(vehicleID, vType, brand, model, vYear, chassisNumber, licensPlate);
248         actFounds.add(v);
249     } //end while
250 } //end else
251
252
253 } //end try
254
255 catch (SQLException sqllex)
256 {
257     FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqllex.getMessage());

```

```

258     } //end catch
259
260     finally
261     {
262         close();
263     } //end finally
264
265     return actFound;
266 } //end method searchActVehicle
267
268 public List<Owner> searchActVehicleWithOwner(String vehicleID)
269 {
270     //Make an ArrayList with Vehicle
271     actFoundVehicles = new ArrayList<Vehicle>(searchActVehicle(vehicleID));
272     // Make an ArrayList for found vehicles
273     actFoundOwners = new ArrayList<Owner>();
274     int vehicleIdentifier = getVehicleIdentifier(vehicleID);
275     try
276     {
277         PreparedStatement pstmt = getPreparedStatement("SELECT o.activeOwnership, o.owneridentifier, ow.*"
278             + " FROM ownership o, owner ow "
279             + " WHERE (o.vehicleidentifier = ?) "
280             + " AND o.owneridentifier = ow.identifier ");
281
282         pstmt.setInt(1, vehicleIdentifier);
283         pstmt.execute();
284
285         ResultSet rsr = null;
286         rsr = pstmt.executeQuery();
287         int activecounter = 0; //This counter is used to go through the ArrayList listactiveness
288         listActiveness.clear(); //Clear the list if a previous search was performed
289         while(rsr.next())
290         {
291             boolean ownerActive = rsr.getBoolean("activeOwnership");
292             String ownerID = rsr.getString("ownerID");
293             String firstName = rsr.getString("FIRSTNAME");
294             String lastName = rsr.getString("LASTNAME");
295             String address = rsr.getString("address");
296             String address2 = rsr.getString("address2");
297             String zip = rsr.getString("zip");
298             String city = rsr.getString("city");
299             String country = rsr.getString("country");
300             String email = rsr.getString("email");
301             String phone = rsr.getString("phone");
302
303             Owner o = new Owner(ownerID, firstName, lastName, address, address2, zip, city, country, phone,
304 email);
305             actFoundOwners.add(o); //Add a owner to the ArrayList which is returned
306             listActiveness.add(ownerActive); //Add the value of ActiveOwnership to an ArrayList
307             activecounter++;
308         } //end while
309     } //end try
310     catch(SQLException sqlex)
311     {
312         FileWriterException.writeLogFile(SuperSearchDAO.class.getName() + sqlex.getMessage());
313     } //end catch
314     //Make a new array from the arraylist listactiveness

```

```

315     activness = new boolean[listActiveness.size()];
316     int arrCounter = 0;
317     for (boolean b : listActiveness)
318     {
319         activness[arrCounter]=b;
320         arrCounter++;
321     }//end for-each
322
323     return actFoundOwners;
324 }//end method searchActVehicleWithOwner
325
326 }//end class SuperSearchDAO

```

UserDAO

```

1 package tweakmc.dataaccess;
2
3
4 import java.sql.PreparedStatement;
5 import java.sql.SQLException;
6 import java.util.ArrayList;
7 import java.util.SortedSet;
8 import tweakmc.model.User;
9 import tweakmc.utility.FileWriterException;
10
11 /**
12  * Connection between the system and the the Database
13  * Transforming java code to SQL
14  * @author ADAMZ
15  */
16 public class UserDAO extends DAO
17 {
18     private static UserDAO instance;
19     private SortedSet<User> actFoundWithoutDuplicate;
20     private ArrayList<User> actFounds;
21
22     private UserDAO()
23     {
24
25     }//end constructor
26
27 // public static UserDAO getInstance
28
29 public static UserDAO getInstance()
30 {
31     if(instance == null)
32     {
33         instance = new UserDAO();
34     }//end if
35     return instance;
36 }//end geting instance methord
37
38 /**
39  * methord to create User
40  * @param initials
41  * @param name
42  * @return

```

```

43  */
44  public boolean createActUser(String initials,String name)
45  {
46      try
47      {
48          PreparedStatement pstmt = getPreparedStatement("INSERT INTO USER(initials,name) VALUE (?,?)");
49          pstmt.setString(1, initials);
50          pstmt.setString(2, name);
51          pstmt.execute();
52          return true;
53      }// end try
54
55      catch (SQLException sqlex)
56      {
57          FileWriterException.writeLogFile(OwnerDAO.class.getName()
58              + sqlex.getMessage());
59          return false;
60      }// end catch
61
62      finally
63      {
64          close();
65      }//end finally
66  }// end method createUser
67
68
69  /**
70   * This method edits an user
71   * @param initials
72   * @param name
73   * @return boolean
74   */
75  public boolean editUser(String initials, String name)
76  {
77      try
78      {
79          PreparedStatement pstmt = getPreparedStatement("UPDATE User SET initials=?, name=?");
80          pstmt.setString(1, initials);
81          pstmt.setString(2, name);
82          pstmt.execute();
83          return true;
84
85
86      }//end try
87      catch(SQLException sqlex)
88      {
89          FileWriterException.writeLogFile(CompanyDAO.class.getName() + sqlex.getMessage());
90          return false;
91      } // End Catch
92  }//end method editUser
93
94
95
96 }//end class UserDao

```

VehicleDAO

```
1 package tweakmc.dataaccess;
2
3 import java.io.File;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8 import java.util.List;
9 import tweakmc.model.Owner;
10 import tweakmc.utility.FileWriterException;
11 import tweakmc.model.Vehicle;
12
13 /**
14  * Connection between the system and the the Database
15  * Transforming java code to SQL
16  *
17  * @author NegoZiatoR
18  * @editedby Tvup
19  */
20 public class VehicleDAO extends DAO
21 {
22     //Instance variables
23     private static VehicleDAO instance;
24     private boolean[] activeness;
25     private ArrayList<Vehicle> actFounds;
26     private ArrayList<Owner> actFoundsOwner;
27     private List<Boolean> listActiveness = new ArrayList<Boolean>();
28
29     /**
30      * Private constructor for VehicleDAO
31      */
32     private VehicleDAO()
33     {
34
35     } //end constructor
36
37     /**
38      * Singleton method for VehicleDAO
39      * @return instance of VehicleDAO
40      */
41     public static VehicleDAO getInstance()
42     {
43         if(instance == null)
44         {
45             instance = new VehicleDAO();
46         } //end if
47         return instance;
48     } //end getInstance method
49
50     /**
51      * Create vehicle with the given parameters in the DB vehicle Table
52      * @param vType of vehicle
53      * @param brand of vehicle
54      * @param model of vehicle
55      * @param vYear of vehicle
56      * @param chassisNumber of vehicle
```



```

57  * @param licensPlate of vehicle
58  * @return true if created else false
59  */
60  public boolean createActVehicle(String vType, String brand, String model, int vYear, String chassisNumber,
String licensPlate)
61  {
62      boolean success = false;
63      try
64      {
65
66          Integer integer_vType = getIdentifierOrInsertNew("VehicleType", vType);
67          Integer integer_brand = getIdentifierOrInsertNew("VehicleBrand", brand);
68          Integer integer_model = getIdentifierOrInsertNew("VehicleModel", model);
69
70          int vehicleID = getHighestVehicleIDfromDatabase();
71          vehicleID++;
72
73          String str_vehicleID = "V" + vehicleID;
74
75          if(makeDir(str_vehicleID))
76          {
77              success = true;
78          } // end if
79
80          PreparedStatement pstm = getPreparedStatement("INSERT INTO VEHICLE (vehicleID, vType, brand,
model, vYear, chassisNumber, licensPlate) VALUES(?, ?, ?, ?, ?, ?, ?)");
81          defineAutoCommit(false);
82          pstm.setString(1, str_vehicleID);
83          pstm.setInt(2, integer_vType);
84          pstm.setInt(3, integer_brand);
85          pstm.setInt(4, integer_model);
86          pstm.setInt(5, vYear);
87          pstm.setString(6, chassisNumber);
88          pstm.setString(7, licensPlate);
89          pstm.execute();
90
91
92      } // end try
93      catch (SQLException sqllex)
94      {
95          FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqllex.getMessage());
96      } // end catch
97
98      finally
99      {
100          if(success)
101          {
102              exeCommit();
103          } //end if
104          close();
105
106      } //end finally
107      return success;
108  } // end method createActVehicle
109
110  /**
111   * Search for vehicle with given searchcriteria
112   * @param <T> generic type for searchcriteria

```

```

113  * @param searchCriteria for vehicle
114  * @return list with actFounds or null
115  */
116  public <T> List<Vehicle> searchActVehicle(T searchCriteria)
117  {
118      ArrayList<Vehicle> actFounds = new ArrayList<Vehicle>();
119      ResultSet rsr = null;
120
121      try
122      {
123          Class c = searchCriteria.getClass();
124          if(c.toString().equals("class java.lang.String"))
125          {
126              PreparedStatement pstmt = getPreparedStatement("SELECT v.VEHICLEID, vt.SUGGESTIONS AS
vType, vb.SUGGESTIONS as brand, vm.SUGGESTIONS as model, v.VYEAR, v.CHASSISNUMBER,
v.LICENSPLATE FROM "
127                  + "VEHICLE v, vehicleType vt, vehicleBrand vb, vehicleModel vm "
128                  + "WHERE (LOWER(vt.suggestions) LIKE LOWER(?) OR LOWER(vb.suggestions) LIKE LOWER(?)
OR LOWER(vm.suggestions) LIKE LOWER(?) OR LOWER(chassisNumber) = LOWER(?) OR LOWER(licensPlate)
= LOWER(?)) AND "
129                  + "(v.vtype = vt.identifier AND "
130                  + "v.brand = vb.identifier AND "
131                  + "v.model = vm.identifier)");
132              pstmt.setString(1, "%" + (String) searchCriteria + "%");
133              pstmt.setString(2, "%" + (String) searchCriteria + "%");
134              pstmt.setString(3, "%" + (String) searchCriteria + "%");
135              pstmt.setString(4, (String) searchCriteria);
136              pstmt.setString(5, (String) searchCriteria);
137              rsr = pstmt.executeQuery();
138
139              while(rsr.next())
140              {
141                  String vehicleID = rsr.getString("vehicleID");
142                  String vType = rsr.getString("vType");
143                  String brand = rsr.getString("brand");
144                  String model = rsr.getString("model");
145                  int vYear = rsr.getInt("vYear");
146                  String chassisNumber = rsr.getString("chassisNumber");
147                  String licensPlate = rsr.getString("licensPlate");
148                  Vehicle v = new Vehicle(vehicleID, vType, brand, model, vYear, chassisNumber, licensPlate);
149                  actFounds.add(v);
150              }//end while
151          }//end if
152
153          else
154          {
155              PreparedStatement pstmt = getPreparedStatement("SELECT v.VEHICLEID, vt.SUGGESTIONS AS
vType, vb.SUGGESTIONS as brand, vm.SUGGESTIONS as model, v.VYEAR, v.CHASSISNUMBER,
v.LICENSPLATE FROM "
156                  + "VEHICLE v, vehicleType vt, vehicleBrand vb, vehicleModel vm "
157                  + "WHERE (vehicleID = ? OR vYear = ?) AND "
158                  + "(v.vtype = vt.identifier AND "
159                  + "v.brand = vb.identifier AND "
160                  + "v.model = vm.identifier)");
161              pstmt.setInt(1, (Integer) searchCriteria);
162              pstmt.setInt(2, (Integer) searchCriteria);
163              rsr = pstmt.executeQuery();
164

```

```

165         while(rsr.next())
166         {
167             String vehicleID = rsr.getString("vehicleID");
168             String vType = rsr.getString("vType");
169             String brand = rsr.getString("brand");
170             String model = rsr.getString("model");
171             int vYear = rsr.getInt("vYear");
172             String chassisNumber = rsr.getString("chassisNumber");
173             String licensPlate = rsr.getString("licensPlate");
174             Vehicle v = new Vehicle(vehicleID, vType, brand, model, vYear, chassisNumber, licensPlate);
175             actFounds.add(v);
176         } //end while
177     } //end else
178
179
180 } //end try
181
182 catch (SQLException sqlex)
183 {
184     FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());
185 } //end catch
186
187 finally
188 {
189     close();
190 } //end finally
191
192 return actFounds;
193 } //end method searchActVehicle
194
195
196
197
198 /**
199  * Edit existing vehicle in db
200  * @param vehicleID vehiles ID
201  * @param vType
202  * @param brand of vehicle
203  * @param model of vehicle
204  * @param vYear
205  * @param chassisNumber of vehicle
206  * @param licensPlate of vehicle
207  * @return true if success else false
208  */
209 public boolean editActVehicle(String vehicleID, String vType, String brand, String model, int vYear, String
chassisNumber, String licensPlate)
210 {
211     try
212     {
213         Integer integer_vType = getIdentifierOrInsertNew("VehicleType", vType);
214         Integer integer_brand = getIdentifierOrInsertNew("VehicleBrand", brand);
215         Integer integer_model = getIdentifierOrInsertNew("VehicleModel", model);
216
217         PreparedStatement pstmt = getPreparedStatement("UPDATE vehicle SET vType = ?, brand = ?, model = ?,
vYear = ?, chassisNumber = ?, licensPlate = ? WHERE vehicleID = ? ");
218         pstmt.setInt(1, integer_vType);
219         pstmt.setInt(2, integer_brand);
220         pstmt.setInt(3, integer_model);

```

```

221     pstmt.setInt(4, vYear);
222     pstmt.setString(5, chassisNumber);
223     pstmt.setString(6, licensPlate);
224     pstmt.setString(7, vehicleID);
225     pstmt.execute();
226     return true;
227 } //end try
228 catch(SQLException sqlex)
229 {
230     FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());
231     return false;
232 } //end catch
233 finally
234 {
235     close();
236 } //end finally
237 }
238
239 /**
240  * Delete vehicle
241  * @param vehicleID on vehicle
242  * @return true if deleted else false
243  */
244 public boolean deleteActVehicle(String vehicleID)
245 {
246     try
247     {
248         PreparedStatement pstmt = getPreparedStatement("DELETE FROM vehicle WHERE vehicleID = ?");
249         pstmt.setString(1, vehicleID);
250         pstmt.execute();
251         return true;
252     } // End try
253     catch(SQLException sqlex)
254     {
255         FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());
256         return false;
257     } // End catch
258     finally
259     {
260         close();
261     } // end Finally
262
263 } // End deleteActVehicle method
264
265 /**
266  * Attach a owner to a vehicle
267  * @param vehicleID on vehicle
268  * @param ownerID on owner
269  * @param active 1 for active , 0 for inactive
270  * @return true if attached else false
271  */
272 public boolean attachActOwner(String vehicleID, String ownerID, int active)
273 {
274     try
275     {
276         Integer integer_identifierVehicle = getVehicleIdentifier(vehicleID);
277         Integer integer_identifierOwner = getOwnerIdentifier(ownerID);

```

```

278     PreparedStatement pstmt = getPreparedStatement("INSERT INTO ownership (vehicleidentifier,
owneridentifier, activeOwnership) VALUES(?, ?, ?)");
279     pstmt.setInt(1, integer_identifierVehicle);
280     pstmt.setInt(2, integer_identifierOwner);
281     pstmt.setInt(3, active);
282     pstmt.execute();
283     return true;
284 } // End try
285 catch(SQLException sqlex)
286 {
287     FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());
288     return false;
289 } // End catch
290 finally
291 {
292     close();
293 } // end Finally
294 } //end method attachActOwner
295
296 /**
297  * This method gets the highest number of the identifier from the vehicletable
298  * @return int
299  */
300 public int getHighestVehicleIDfromDatabase()
301 {
302     try
303     {
304         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM vehicle");
305         ResultSet rsr = null;
306         rsr = pstmt.executeQuery();
307         rsr.next();
308         return rsr.getInt(1);
309     } //end try
310     catch (SQLException sqlex)
311     {
312         FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());
313         return -1;
314     } //end catch
315
316     finally
317     {
318         close();
319     } //end finally
320 } //end method getHighestVehicleIDfromDatabase
321
322 public int getHighestVehicleTypeIDfromDatabase()
323 {
324     try
325     {
326         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM vehicleType");
327         ResultSet rsr = null;
328         rsr = pstmt.executeQuery();
329         rsr.next();
330         return rsr.getInt(1);
331     } //end try
332     catch (SQLException sqlex)
333     {
334         FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqlex.getMessage());

```

```

335         return -1;
336     } //end catch
337
338     finally
339     {
340         close();
341     } //end finally
342 } //end method getHighestVehicleTypeIDfromDatabase
343
344 /**
345  * This method gets the highest vehicleBrandIdentifier from the database
346  * @return int
347  */
348 public int getHighestVehicleBrandIDfromDatabase()
349 {
350     try
351     {
352         PreparedStatement pstm = getPreparedStatement("SELECT MAX(identifier) FROM vehicleBrand");
353         ResultSet rsr = null;
354         rsr = pstm.executeQuery();
355         rsr.next();
356         return rsr.getInt(1);
357     } //end try
358     catch (SQLException sqllex)
359     {
360         FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqllex.getMessage());
361         return -1;
362     } //end catch
363
364     finally
365     {
366         close();
367     } //end finally
368 } //end method getHighestVehicleBrandIDfromDatabase
369
370 /**
371  * This method gets the highest VehicleModelIdentifier from the database
372  * @return int
373  */
374 public int getHighestVehicleModelIDfromDatabase()
375 {
376     try
377     {
378         PreparedStatement pstm = getPreparedStatement("SELECT MAX(identifier) FROM vehicleModel");
379         ResultSet rsr = null;
380         rsr = pstm.executeQuery();
381         rsr.next();
382         return rsr.getInt(1);
383     } //end try
384     catch (SQLException sqllex)
385     {
386         FileWriterException.writeLogFile(VehicleDAO.class.getName() + sqllex.getMessage());
387         return -1;
388     } //end catch
389
390     finally
391     {
392         close();

```

```

393     }//end finally
394 }//end method getHighestVehicleModelIDfromDatabase
395
396 /**
397  * This method gets an identifier or inserts a new into the database
398  * @param tableName
399  * @param tableColumn
400  * @return Integer
401  */
402 private Integer getIdentiferOrInsertNew(String tableName, String tableColumn)
403 {
404     Integer integer_identifier = null;
405     try
406     {
407         PreparedStatement pstmtGetTypeBrandModel = getPreparedStatement("SELECT identifier FROM " +
tableName + " WHERE suggestions = ?");
408         pstmtGetTypeBrandModel.setString(1, tableColumn);
409         pstmtGetTypeBrandModel.execute();
410         ResultSet rsr = null;
411         rsr = pstmtGetTypeBrandModel.executeQuery();
412         if(rsr.next())
413             integer_identifier = rsr.getInt(1);
414         else
415         {
416             int ID = getHighestVehicleTypeIDfromDatabase();
417             ID++;
418             PreparedStatement pstmt = getPreparedStatement("INSERT INTO " + tableName + " (" + tableName +
"ID, suggestions) VALUES(?, ?)");
419             pstmt.setInt(1, ID);
420             pstmt.setString(2, tableColumn);
421             pstmt.execute();
422             pstmtGetTypeBrandModel = getPreparedStatement("SELECT identifier FROM " + tableName + "
WHERE suggestions = ?");
423             pstmtGetTypeBrandModel.setString(1, tableColumn);
424             pstmtGetTypeBrandModel.execute();
425             rsr = null;
426             rsr = pstmtGetTypeBrandModel.executeQuery();
427             if(rsr.next())
428                 integer_identifier = rsr.getInt(1);
429         }//end else
430     }//end try
431     catch (SQLException sqlex)
432     {
433         FileWriterException.writeLogFile(VehicleDAO.class.getName() + "/filemeu-calc/" + sqlex.getMessage());
434         return null;
435     }//end catch
436     finally
437     {
438         close();
439     }//end finally
440     return integer_identifier;
441 }//end method getIdentiferOrInsertNew
442
443
444 /**
445  * Search for a vehicle and return owners attached
446  * @param vehicleID
447  * @return List with owners belonging to that vehicle

```

```

448  */
449  public List<Owner> searchActOwnerWithVehicles(String vehicleID)
450  {
451      //Make an ArrayList with owner
452      actFounds = new ArrayList<Vehicle>(searchActVehicle(vehicleID));
453      // Make an ArrayList for found vehicles
454      actFoundsOwner = new ArrayList<Owner>();
455      int vehicleIdentifier = getVehicleIdentifier(vehicleID);
456      try
457      {
458          PreparedStatement pstmt = getPreparedStatement("SELECT ow.activeOwnerShip, ow.owneridentifier,
459          o.identifier, o.ownerID, o.firstName, o.lastName, o.address, o.address2, o.zip, o.city, o.country, o.email, o.phone "
460          + "FROM ownership ow, owner o "
461          + "WHERE (ow.vehicleidentifier = ?) "
462          + "AND ow.owneridentifier = o.identifier");
463
464          pstmt.setInt(1, vehicleIdentifier);
465          pstmt.execute();
466
467          ResultSet rsr = null;
468          rsr = pstmt.executeQuery();
469          int activecounter = 0; //This counter is used to go through the ArrayList listactiveness
470          listActiveness.clear(); //Clear the list if a previous search was performed
471          while(rsr.next())
472          {
473              boolean ownerActive = rsr.getBoolean("activeOwnerShip");
474              String ownerID = rsr.getString("ownerID");
475              String firstName = rsr.getString("firstName");
476              String lastName = rsr.getString("lastName");
477              String address = rsr.getString("address");
478              String address2 = rsr.getString("address2");
479              String zip = rsr.getString("zip");
480              String city = rsr.getString("city");
481              String country = rsr.getString("country");
482              String email = rsr.getString("email");
483              String phone = rsr.getString("phone");
484
485              Owner o = new Owner(ownerID, firstName, lastName, address, address2, zip, city, country, email,
486              phone);
487              actFoundsOwner.add(o); //Add a owner to the ArrayList which is returned
488              listActiveness.add(ownerActive); //Add the value of ActiveOwnerShip to an ArrayList
489              activecounter++;
490          } //end while
491      } //end try
492      catch(SQLException sqlex)
493      {
494          FileWriterException.writeLogFile(VehicleDAO.class.getName() + " /searchActOwnerWithVehicles " +
495          sqlex.getMessage());
496      } //end catch
497      //Make a new array from the arraylist listactiveness
498      activness = new boolean[listActiveness.size()];
499      int arrCounter = 0;
500      for (boolean b : listActiveness)
501      {
502          activness[arrCounter]=b;
503          arrCounter++;
504      } //end for-each

```



```

503
504     return actFoundsOwner;
505 } //end method searchActOwnerWithVehicles
506
507 /**
508  * Gets an array with active-statuses on ownership
509  * @return boolean[]
510  */
511 public boolean[] getActiveNess()
512 {
513     return activness;
514 } //end method getActiveNess
515
516 /**
517  * Find information about a vehicle and evt owner
518  * @param vehicleID to find info about
519  * @return String with info if vehicleID exist else empty string
520  */
521 public String getVehicleNameAndOwnerName(String vehicleID)
522 {
523     String searchResult = "";
524     try
525     {
526         PreparedStatement pstmt = getPreparedStatement("SELECT vb.suggestions AS Brand, " +
527             "vm.suggestions AS Model, ve.vyear AS Vyear " +
528             "FROM vehicle ve, vehiclebrand vb, vehiclemodel vm " +
529             "WHERE LOWER(ve.vehicleID) = LOWER(?) AND ve.brand = vb.identifier AND " +
530             "ve.model = vm.identifier");
531         pstmt.setString(1, vehicleID);
532         ResultSet rs = pstmt.executeQuery();
533
534         while(rs.next())
535         {
536             searchResult = rs.getString("Brand") + " ";
537             searchResult += rs.getString("Model") + ", ";
538             searchResult += String.valueOf(rs.getInt("Vyear"));
539         } //end while
540
541         if(!searchResult.equalsIgnoreCase(""))
542         {
543
544             PreparedStatement pstmt1 = getPreparedStatement("SELECT ow.FIRSTNAME AS fname, "
545                 + "ow.LASTNAME AS lname " +
546                 "FROM vehicle ve, ownership os, owner ow " +
547                 "WHERE LOWER(ve.vehicleID) = LOWER(?) AND ve.IDENTIFIER = os.VEHICLEIDENTIFIER "
548                 + "AND os.OWNERIDENTIFIER = ow.IDENTIFIER AND os.ACTIVEOWNERSHIP = 1");
549             pstmt1.setString(1, vehicleID);
550             ResultSet rs1 = pstmt1.executeQuery();
551
552             boolean ownerPresent = false;
553
554             while(rs1.next())
555             {
556                 searchResult += ". Owner: ";
557                 searchResult += rs1.getString("fname") + " ";
558                 searchResult += rs1.getString("lname") + ". ";
559                 ownerPresent = true;

```

```

560         } //end while
561
562         if(!ownerPresent)
563         {
564             searchResult += ". No owner.";
565         }
566     } //end if
567
568 } //end try
569
570 catch (SQLException ex)
571 {
572     FileWriterException.writeLogFile(VehicleDAO.class.getName() + " /getVehicleNameAndOwnerName/" +
573         ex.getMessage());
574     searchResult = "";
575 } //end catch
576
577 finally
578 {
579     close();
580     return searchResult;
581 } //end finally
582 } //getVehicleNameAndOwnerName(String vehicleID)
583
584
585
586 } //end VehicleDAO class

```

WhiteboardDAO

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.dataaccess;
7
8 import java.sql.PreparedStatement;
9 import java.sql.ResultSet;
10 import java.sql.SQLException;
11 import tweakmc.model.Result.Whiteboard.ValveAdjustmentTable;
12 import tweakmc.utility.FileWriterException;
13
14
15 /**
16  * Connection between the system and the the Database
17  * Transforming java code to SQL
18  * @author NegoZiatoR
19  */
20 public class WhiteboardDAO extends DAO
21 {
22     // Instance variables
23     private static WhiteboardDAO instance;
24
25     /**
26      * Private constructor
27      */
28     private WhiteboardDAO()
29     {

```

```

30
31 }
32
33 /**
34  * Singleton constructor for WhiteboardDAO
35  * @return only ONE instance of WhiteboardDAO
36  */
37 public static WhiteboardDAO getInstance()
38 {
39     if(instance == null)
40     {
41         return new WhiteboardDAO();
42     }
43     return instance;
44 }
45
46 /**
47  * Create a new Valveadjustment table
48  * @param table table to crete
49  * @return true if created else false
50  */
51 public boolean createActValveadjustmentTablewhiteboard(ValveAdjustmentTable table)
52 {
53     ValveAdjustmentTable whiteboard = table;
54     try
55     {
56         // Get whiteboard ID
57         int whiteboardID = getHighestWhiteboardIDfromDatabase();
58         whiteboardID++;
59         String str_whiteboardID = "W" + whiteboardID;
60
61         // Get Cylinder ID
62         int cylinderID = getHighestCylinderIDfromDatabase();
63         cylinderID++;
64         String str_cylinderID = "C" + cylinderID;
65
66         int i = 0;
67         while(i < whiteboard.getCylinderNo().length)
68         {
69
70             // First insert releveant values into whiteboard table
71             PreparedStatement pstmt = getPreparedStatement("INSERT INTO whiteboard(resultIdentifier,
whiteboardID) values (?,?)");
72
73             pstmt.setInt(1, getIdentifierFromID("R1", "result"));
74             pstmt.setString(2, str_whiteboardID);
75             pstmt.execute();
76
77
78             // Insert releveant values into valveadjustment table
79             pstmt = getPreparedStatement("INSERT INTO valveadjustment(tableID, whiteboardIdentifier) values(?,
?);");
80
81             pstmt.setString(1, str_whiteboardID);
82             pstmt.setInt(2, getIdentifierFromID(str_whiteboardID, "whiteboard"));
83             pstmt.execute();
84
85             // Insert releveant values into cylinder table

```

```

86      pstmt = getPreparedStatement("INSERT INTO cylinder (valveadjustmentIdentifier, cylinderNumber)
values(?, ?)");
87      String[] cylinderNo = whiteboard.getCylinderNo();
88      pstmt.setInt(1, getIdentifierFromID(whiteboard.getResultID(), "valveadjustment"));
89      pstmt.setInt(2, Integer.parseInt(cylinderNo[i+1]));
90      pstmt.execute();
91
92      // Insert releveant values into cylinderposition table
93      pstmt = getPreparedStatement("INSERT INTO cylinderposition (cylidneridentifier, cPositionName)
values(?, ?)");
94      String[] cylinderPos = whiteboard.getCylinderPos();
95      pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
96      pstmt.setInt(2, Integer.parseInt(cylinderPos[i+1]));
97      pstmt.execute();
98
99      // Insert releveant values into measurementtype table
100     pstmt = getPreparedStatement("INSERT INTO measurementtype (mTypeName, cylinderidentifier)
values(?, ?)");
101     String[] measurementTypeIN = whiteboard.getInArray();
102     pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
103     pstmt.setInt(2, Integer.parseInt(measurementTypeIN[i]));
104     pstmt.execute();
105
106     pstmt = getPreparedStatement("INSERT INTO measurementtype (mTypeName, cylinderidentifier)
values(?, ?)");
107     String[] measurementTypeTOLL = whiteboard.getTollArray();
108     pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
109     pstmt.setInt(2, Integer.parseInt(measurementTypeTOLL[i]));
110     pstmt.execute();
111
112     pstmt = getPreparedStatement("INSERT INTO measurementtype (mTypeName, cylinderidentifier)
values(?, ?)");
113     String[] measurementTypeDIFF = whiteboard.getDiffArray();
114     pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
115     pstmt.setInt(2, Integer.parseInt(measurementTypeDIFF[i]));
116     pstmt.execute();
117
118     pstmt = getPreparedStatement("INSERT INTO measurementtype (mTypeName, cylinderidentifier)
values(?, ?)");
119     String[] measurementTypeSHIM = whiteboard.getShimArray();
120     pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
121     pstmt.setInt(2, Integer.parseInt(measurementTypeSHIM[i]));
122     pstmt.execute();
123
124     pstmt = getPreparedStatement("INSERT INTO measurementtype (mTypeName, cylinderidentifier)
values(?, ?)");
125     String[] measurementTypeNEWSHIM = whiteboard.getNewShimArray();
126     pstmt.setInt(1, getIdentifierFromID(str_cylinderID, "cylinder"));
127     pstmt.setInt(2, Integer.parseInt(measurementTypeNEWSHIM[i]));
128     pstmt.execute();
129
130     }
131
132     } // End try
133     catch(SQLException sqlex)
134     {
135         FileWriterException.writeLogFile(WhiteboardDAO.class.getName() + sqlex.getMessage());
136         return false;

```

```

137     } // End Catch
138     finally
139     {
140         close();
141     } // End Finally
142     return true;
143 }
144
145 /**
146  * This method will get the highest identifier for result in the database
147  * @return int
148  */
149 public int getHighestResultIDfromDatabase()
150 {
151     try
152     {
153         PreparedStatement pstm = getPreparedStatement("SELECT MAX(identifier) FROM result");
154         ResultSet rsr = null;
155         rsr = pstm.executeQuery();
156         rsr.next();
157         return rsr.getInt(1);
158     } //end try
159     catch (SQLException sqllex)
160     {
161         FileWriterException.writeLogFile(WhiteboardDAO.class.getName() + "
162 /getHighestResultIDfromDatabase" + sqllex.getMessage());
163         return -1;
164     } //end catch
165     finally
166     {
167         close();
168     } //end finally
169 } //end method getHighestResultIDfromDatabase
170
171
172 /**
173  * This method will get the highest identifier for whiteboard in the database
174  * @return int highest whiteboardID
175  */
176 public int getHighestWhiteboardIDfromDatabase()
177 {
178     try
179     {
180         PreparedStatement pstm = getPreparedStatement("SELECT MAX(identifier) FROM whiteboard");
181         ResultSet rsr = null;
182         rsr = pstm.executeQuery();
183         rsr.next();
184         return rsr.getInt(1);
185     } //end try
186     catch (SQLException sqllex)
187     {
188         FileWriterException.writeLogFile(WhiteboardDAO.class.getName() + "
189 /getHighestWhiteboardIDfromDatabase" + sqllex.getMessage());
190         return -1;
191     } //end catch
192     finally

```

```

193     {
194         close();
195     } //end finally
196 } //end method getHighestResultIDfromDatabase
197
198 /**
199  * This method will get the highest identifier for whiteboard in the database
200  * @return int highest whiteboardID
201  */
202 public int getHighestCylinderIDfromDatabase()
203 {
204     try
205     {
206         PreparedStatement pstmt = getPreparedStatement("SELECT MAX(identifier) FROM cylinder");
207         ResultSet rsr = null;
208         rsr = pstmt.executeQuery();
209         rsr.next();
210         return rsr.getInt(1);
211     } //end try
212     catch (SQLException sqllex)
213     {
214         FileWriterException.writeLogFile(WhiteboardDAO.class.getName() + "
/getHighestCylinderIDfromDatabase" + sqllex.getMessage());
215         return -1;
216     } //end catch
217
218     finally
219     {
220         close();
221     } //end finally
222 } //end method getHighestResultIDfromDatabase
223
224 } //end class whiteboardDAO

```

WorkstationDAO

```

1 package tweakmc.dataaccess;
2
3 import java.sql.PreparedStatement;
4 import java.sql.ResultSet;
5 import java.sql.SQLException;
6 import java.util.Vector;
7 import java.util.logging.Level;
8 import java.util.logging.Logger;
9 import tweakmc.model.Workstation;
10 import tweakmc.utility.FileWriterException;
11
12 /**
13  * Connection between the system and the the Database
14  * Transforming java code to SQL
15  * @author clausPallisgaardBeck
16  */
17 public class WorkstationDAO extends DAO
18 {
19     //Instance variables

```

```

20 private static WorkstationDAO instance;
21
22 private Vector workstationVector = new Vector();
23
24 private WorkstationDAO()
25 {
26
27 } //end constructor WorkstationDAO
28
29 public static WorkstationDAO getInstance()
30 {
31     if(instance == null)
32     {
33         instance = new WorkstationDAO();
34     } //end if
35     return instance;
36 } //end method getInstance()
37 /**
38  * Set a new name for a workstation
39  * @param id on workstation where name should be changed
40  * @param name new name on workstation
41  * @return true if changed else false
42  */
43 public boolean changeWorkstationName(String id, String name)
44 {
45     try
46     {
47         PreparedStatement pstm = getPreparedStatement("UPDATE WORKSTATION SET
WORKSTATIONNAME = ? WHERE WHERE WORKSTATIONID = ?");
48         pstm.setString(1, name);
49         pstm.setString(2, id);
50         return pstm.execute();
51     } //end try
52
53     catch (SQLException ex)
54     {
55         Logger.getLogger(WorkstationDAO.class.getName()).log(Level.SEVERE, null, ex);
56         FileWriterException.writeLogFile(WorkstationDAO.class.getName() +
57             " / setName/ " + ex.getMessage());
58         return false;
59     } //end catch
60
61     finally
62     {
63         close();
64     } //end finally
65 } //end changeWorkstationName(String id, String name)
66
67 /**
68  * Get all workstations and there information
69  * @return Vector with workstation instance
70  */
71 public Vector getAllWorkstation()
72 {
73     try
74     {
75         PreparedStatement pstm = getPreparedStatement("SELECT * FROM WORKSTATION");
76         ResultSet rs = pstm.executeQuery();

```

```

77
78     while(rs.next())
79     {
80         String id = rs.getString("workstationID");
81         String name = rs.getString("workstationName");
82         workstationVector.add(new Workstation(id, name));
83     }//end while
84
85     return workstationVector;
86 }//end try
87
88 catch (SQLException ex)
89 {
90     Logger.getLogger(WorkstationDAO.class.getName()).log(Level.SEVERE, null, ex);
91     FileWriterException.writeLogFile(WorkstationDAO.class.getName() +
92         " / getAllWorkstation/ " + ex.getMessage());
93     return null;
94 }//end catch
95
96 finally
97 {
98     close();
99 }//end finally
100 }//end method getAllWorkstation()
101 }//end class WorkstationDao

```

Control layer (tweakmc.control)

FileReaderController

```

1 package tweakmc.control;
2
3 import java.io.File;
4 import java.io.FileNotFoundException;
5 import java.io.FileReader;
6 import java.io.IOException;
7 import java.util.ArrayList;
8 import java.util.List;
9 import java.util.Scanner;
10 import tweakmc.utility.FileWriterException;
11
12 /**
13  * Control the reading of files for TweakMC system
14  * @author lasch17
15  */
16 public class FileReaderController
17 {
18     //instance variable
19
20     /**
21      * Read file for autocomplete decoration in the view layer
22      * @param fileName where to read from
23      * @return list with autocompletedecoration text
24      */

```



```

25 public static List<String> readForAutocomplete(String fileName)
26 {
27     FileReader fr = null;
28     ArrayList<String> autocompleteList = new ArrayList<String>();
29
30     try
31     {
32         fr = new FileReader(new File(fileName));
33         Scanner in = new Scanner(fr);
34
35         while (in.hasNext())
36         {
37             String autoText = in.nextLine();
38             autocompleteList.add(autoText);
39         } //end while
40     } //end try
41
42     catch (FileNotFoundException ex)
43     {
44         FileWriterException.writeLogFile(ex.getMessage() + "from fileReader - readForAutoComplete");
45     } //end catch
46
47     finally
48     {
49         try
50         {
51             fr.close();
52         } //end try
53
54         catch (IOException ex)
55         {
56             FileWriterException.writeLogFile(ex.getMessage() + "from fileReader - readForAutoComplete");
57         } //end catch
58     } //end finally
59
60     return autocompleteList;
61 } //end method readForAutocomplete
62
63 /**
64  * Read logfile to List
65  * @return String
66  */
67 public static String readLogfile()
68 {
69     FileReader fr = null;
70     String logfileText = "";
71
72     try
73     {
74         fr = new FileReader(new File("logfile.txt"));
75         Scanner in = new Scanner(fr);
76
77         while (in.hasNext())
78         {
79             String autoText = in.nextLine();
80             logfileText += autoText + "\n";
81         } //end while
82     } //end try

```

```

83
84     catch (FileNotFoundException ex)
85     {
86         FileWriterException.writeLogFile(ex.getMessage() + "from fileReader - readLogfile");
87     }//end catch
88
89     finally
90     {
91         try
92         {
93             fr.close();
94         }//end try
95
96         catch (IOException ex)
97         {
98             FileWriterException.writeLogFile(ex.getMessage() + "from fileReader - readLogfile");
99         }//end catch
100     }//end finally
101
102     return logfileText;
103
104 }//end method readLogfile
105 }//end Class FileReaderController

```

LoginController

```

1 package tweakmc.control;
2
3 import tweakmc.model.Login;
4 /**
5  * Control all methods between
6  * GUI and system
7  *
8  * @author Nyvang
9  * @edited by NegoZiatoR
10 */
11 public class LoginController {
12
13     //private Login logObject;
14
15     /**
16      * Constructor for LoginController
17      */
18     public LoginController()
19     {
20         Login logObject = Login.getInstance();
21         logObject.addLogin("ADMIN", "GOD!");
22         logObject.addLogin("NN", "Nicolaj Nyvang");
23         logObject.addLogin("CB", "Claus Beck");
24         logObject.addLogin("LS", "Lars Schou");
25         logObject.addLogin("TH", "Torben Hansen");
26         logObject.addLogin("ADAM", "Adam Adams");
27     }//end constructor
28
29     /**
30      * Return login name on initial
31      * @param initials initial for check

```

```

32  * @return name for login if present
33  */
34  public String getLoginName(String initials)
35  {
36      Login logObject = Login.getInstance();
37      return logObject.getName(initials);
38  } //end method getLoginName
39
40 } //end class LoginController

```

ManageCompanyController

```

1  package tweakmc.control;
2
3  import tweakmc.dataaccess.CompanyDAO;
4
5
6  /**
7   * Control all methods between
8   * GUI and system
9   * for Company
10  */
11  * @author Romanty
12  */
13
14  public class ManageCompanyController
15  {
16      //instance variable
17      /**
18       *
19       * @param curNo
20       * @param name
21       * @param address
22       * @param phone
23       * @param email
24       * @param homePage
25       * @param bankInfo
26       * @return
27       */
28      public boolean addCompany(String curNo, String name, String address, String phone, String email, String
homePage, String bankInfo)
29      {
30          return CompanyDAO.getInstance().createCompany(curNo, name, address, phone, email, homePage, bankInfo);
31      } //End addCompany method
32
33      /**
34       *
35       * @param curNo
36       * @param name
37       * @param address
38       * @param phone
39       * @param email
40       * @param Homepage
41       * @param bankInfo
42       * @return
43       */

```

```

44 public boolean editCompany(String curNo, String name, String address, String phone, String email, String
Homepage, String bankInfo)
45 {
46     return CompanyDAO.getInstance().editCompany(curNo, name, address, phone, email, Homepage, bankInfo);
47
48 }// End of editCompany methoerd
49
50 }//end class ManageCompanyControll

```

ManageOwnerController

```

1 package tweakmc.control;
2 import java.util.List;
3 import tweakmc.model.*;
4 /**
5  * Control all methods between
6  * GUI and system
7  * for Owner
8  * @author NegoZiatoR
9  * @editedby Tvup
10 */
11 public class ManageOwnerController
12 {
13     //Instance variables
14
15     /**
16      * Create a new owner
17      * @param firstName
18      * @param lastName
19      * @param address
20      * @param address2
21      * @param zip
22      * @param city
23      * @param country
24      * @param phone
25      * @param email
26      * @return true if created else false
27      */
28 public boolean makeNewOwner(String firstName, String lastName, String address, String address2, String zip,
String city, String country, String phone, String email)
29 {
30     return OwnerCatalog.getInstance().makeNewOwner(firstName, lastName, address, address2, zip, city, country,
phone, email);
31 }// End makeNewOwner method
32
33 /**
34  * Update data about an owner
35  * @param ownerID
36  * @param firstName
37  * @param lastName
38  * @param address
39  * @param address2
40  * @param zip
41  * @param city
42  * @param country
43  * @param phone
44  * @param email

```

```

45  * @return
46  */
47  public boolean updateOwner(String ownerID, String firstName, String lastName, String address, String address2,
String zip, String city, String country, String phone, String email)
48  {
49      return OwnerCatalog.getInstance().updateOwner(ownerID, firstName, lastName, address, address2, zip, city,
country, phone, email);
50
51  } // End of updateOwner method
52  /**
53   * Search for an owner
54   * @param <T>
55   * @param seachCriteria
56   * @return
57   */
58  public <T> List<Owner> searchOwner (T seachCriteria)
59  {
60      return OwnerCatalog.getInstance().searchOwner(seachCriteria);
61  } // End of searchOwner method in ownerList
62
63
64
65  /**
66   * Search for an owner and the vehicle he owns
67   * @param ownerID owner id for search
68   * @return List with vehicles the found owner owns
69   */
70  public List<Vehicle> searchOwnerWithVehicle(String ownerID)
71  {
72      return OwnerCatalog.getInstance().searchOwnerWithVehicle(ownerID);
73  } // End of searchOwner method in VehicleList
74
75
76  /**
77   * Delete an owner
78   * @param ownerID on owner
79   * @return true if deleted else false
80   */
81  public boolean deleteOwner(String ownerID)
82  {
83      return OwnerCatalog.getInstance().deleteOwner(ownerID);
84  } // End of deleteOwner method
85
86
87  /**
88   * Method to get if the owner owns the vehicle at the memoment
89   * @return array boolean[]
90   */
91  public boolean[] getActiveNess()
92  {
93      return OwnerCatalog.getInstance().getActiveNess();
94  } // End of getActiveness mehord
95
96  } // End ManageOwnerController

```

ManageResultController

```
1 package tweakmc.control;
2 import javax.swing.JTable;
3 import tweakmc.model.Result.ResultCatalog;
4 import tweakmc.model.Result.Result;
5 /**
6  * Control all methods between
7  * GUI and system
8  * for Result
9  * @author NegoZiatoR
10 */
11 public class ManageResultController
12 {
13     // Instance Variables
14
15     /**
16      * Create a new result
17      * @param originalDestination
18      * @param mechanicComment
19      * @param customerComment
20      * @param resultName
21      * @param resultType type of result
22      * @param vehicleID
23      * @param workstationID
24      * @param orderID
25      * @param whiteboardID
26      * @return true if created else false
27      */
28     public boolean makeNewResult(String originalDestination, String mechanicComment, String customerComment,
29 String resultName, String resultType, String vehicleID, String workstationID, String orderID, String whiteboardID)
30     {
31         return ResultCatalog.getInstance().createNewUploadResult(originalDestination, mechanicComment,
32 customerComment, resultName, resultType, vehicleID, workstationID, orderID, whiteboardID);
33     } // End makeNewResult
34
35     /**
36      * This method creates a result from the spreadsheet
37      * @param tableToSave
38      * @param mechanicComment
39      * @param customerComment
40      * @param resultName
41      * @param vehicleID
42      * @param workstationID
43      * @param orderID
44      * @return true if a result is created else fail
45      */
46     public boolean newSpreadsheetResult(JTable tableToSave, String mechanicComment, String customerComment,
47 String resultName, String vehicleID, String workstationID, String orderID)
48     {
49         return ResultCatalog.getInstance().createNewSpreadsheetResult(tableToSave, mechanicComment,
50 customerComment, resultName, vehicleID, workstationID, orderID);
51     } //end method newSpreadsheetResult(JTable tableToSave, String mechanicComment, String customerComment,
52 String resultName, String vehicleID, String workstationID, String orderID)
53 }
54 //endclass ManageResultController
```

ManageSperadsheetController

```
1 package tweakmc.control;
2
3 import javax.swing.JTable;
4 import tweakmc.model.spreadsheet.SpreadsheetCatalog;
5
6 /**
7  * Control all methods between
8  * GUI and system
9  * for Spreadsheet
10 * @author Tvup
11 */
12 public class ManageSpreadsheetController {
13
14
15     public ManageSpreadsheetController()
16     {
17
18     } //end constructor
19
20     /**
21      * This method creates the spreadsheet
22      * @param rows
23      * @param columns
24      * @param columnsNames
25      * @return JTable
26      */
27     public JTable createSpreadsheet(int rows, int columns, String [] columnsNames)
28     {
29         return SpreadsheetCatalog.getInstance().createSpreadsheet(rows, columns, columnsNames);
30
31     } //end method createSpreadsheet
32
33     /**
34      * Find the selected spreadsheet to managed
35      * @param path there spreadsheet is saved
36      * @return selected spreadsheet
37      */
38     public JTable findSelectedSpreadsheet(String path)
39     {
40         return SpreadsheetCatalog.getInstance().findSelectedSpreadsheet(path);
41     } //end method findSelectedSpreadsheet(String path)
42
43     /**
44      * This method saves the table with the data
45      * @param tableToSave
46      * @param fileName
47      * @return true if created else false
48      */
49     // public boolean savePopulatedTable(JTable tableToSave, String fileName)
50     // {
51     //     return SpreadsheetCatalog.getInstance().savePopulatedTable(tableToSave, fileName);
52     // } //end method savePopulatedTable
53
54 } //end class ManageSpreadsheetController
```

ManageUserController

```
1 package tweakmc.control;
2
3 import tweakmc.dataaccess.UserDAO;
4
5 /**
6  * Control all methods between
7  * GUI and system
8  * for User
9  * @author ADAMZ
10 */
11 public class ManageUserController
12 {
13
14     /**
15      *
16      * @param initials
17      * @param name
18      * @return
19      */
20     public boolean addUser(String initials, String name)
21     {
22         return UserDAO.getInstance().createActUser(initials, name);
23     } //end addUser
24     /**
25      *
26      * @param initials
27      * @param name
28      * @return
29      */
30
31     public boolean editUse(String initials, String name)
32     {
33         return UserDAO.getInstance().editUser(initials, name);
34     } // end edit user
35
36 } //end class ManageUserController
```

ManageVehicleController

```
1 package tweakmc.control;
2
3 import java.util.List;
4 import tweakmc.model.Owner;
5 import tweakmc.model.Vehicle;
6 import tweakmc.model.VehicleCatalog;
7
8 /**
9  * Control all methods between
10 * GUI and system
11 * for Vehicle
12 * @author NegoZiatoR
13 */
14 public class ManageVehicleController
15 {
16
17     /**
```



```

18  * Pass information about vehicle to vehicle catalog
19  * @param vType of vehicle
20  * @param brand of vehicle
21  * @param model of vehicle
22  * @param vYear of vehicle
23  * @param chassisNumber of vehicle
24  * @param licensPlate of vehicle
25  * @return true if created false if not
26  */
27  public boolean makeNewVehicle(String vType, String brand, String model, int vYear, String chassisNumber,
String licensPlate)
28  {
29      return VehicleCatalog.getInstance().makeNewVehicle(vType, brand, model, vYear, chassisNumber,
licensPlate);
30  } // End makeNewVehicle method
31
32  /**
33   * Search for a vehicle
34   * @param <T>
35   * @param seachCriteria
36   * @return
37   */
38  public <T> List<Vehicle> searchVehicle (T seachCriteria)
39  {
40      return VehicleCatalog.getInstance().searchVehicle(seachCriteria);
41  } //end method searchVehicle
42
43  /**
44   * Search for a Vehicle and the owners
45   * @param vehicleID vehicles id for search
46   * @return List with owners
47   */
48  public List<Owner> searchVehicleWithOwner(String vehicleID)
49  {
50      return VehicleCatalog.getInstance().searchVehicleWithOwner(vehicleID);
51  } //end method searchVehicleWithOwner
52
53  /**
54   * This method is for the supersearcher. It searches for owners attached to vehicles
55   * @param vehicleID
56   * @return List<Owner>
57   */
58  public List<Owner> superSearchVehicleWithOwner(String vehicleID)
59  {
60      return VehicleCatalog.getInstance().superSearchVehicleWithOwner(vehicleID);
61  } //end method superSearchVehicleWithOwner
62
63  /**
64   * Method for editing a vehicle
65   * @param vehicleID
66   * @param vType
67   * @param brand
68   * @param model
69   * @param vYear
70   * @param chassisNumber
71   * @param licensPlate
72   * @return boolean
73   */

```

```

74  public boolean editVehicle(String vehicleID, String vType, String brand, String model, int vYear, String
chassisNumber, String licensPlate)
75  {
76      return VehicleCatalog.getInstance().editVehicle(vehicleID, vType, brand, model, vYear, chassisNumber,
licensPlate);
77  } //end method createVehicle
78
79  /**
80   * Delete a vehicle
81   * @param vehicleID on vehicle
82   * @return true if deleted else false
83   */
84  public boolean deleteVehicle(String vehicleID)
85  {
86      return VehicleCatalog.getInstance().deleteVehicle(vehicleID);
87  } //end method deleteVehicle
88
89  /**
90   * Attach a owner to a vehicle
91   * @param vehicleID in vehicle
92   * @param ownerID on owner
93   * @param active 1 for active 0 for not active
94   * @return true if attached else false
95   */
96  public boolean attachOwner(String vehicleID, String ownerID, int active)
97  {
98      return VehicleCatalog.getInstance().attachOwner(vehicleID, ownerID, active);
99  } //end method attachOwner
100
101  /**
102   * Search for a vehicle and the owners which belongs
103   * @return List with owners the found vehicles belongs to
104   */
105  public boolean[] getActiveNess()
106  {
107      return VehicleCatalog.getInstance().getActiveNess();
108  } //end method getActiveNess
109
110 } // End ManageVehicleController class

```

ManageWhiteboardController

```

1
2
3  package tweakmc.control;
4
5  import tweakmc.model.Result.Whiteboard.WhiteboardCatalog;
6  /**
7   * Control all methods between
8   * GUI and system
9   * for Whiteboard
10  * @author NegoZiatoR
11  */
12  public class ManageWhiteboardController
13  {
14
15  /**
16   * @param mechanicComment mechanics comments on table

```

```

17  * @param customerComment comments for customer
18  * @param cylinderNo cylinder numbers on table
19  * @param cylinderPos cylinder positions on cylinder numbers
20  * @param inArray in value on cylinders
21  * @param tollArray tollerance on cylinders
22  * @param diffArray difference on cylinders
23  * @param shimArray shim on cylinders
24  * @param newShimArray new shim on cylinders
25  * @return true if created else false
26  */
27  public boolean makeNewValveAdjustmentWhiteboard(String mechanicComment, String customerComment,
String[] cylinderNo, String[] cylinderPos, String[] inArray, String[] tollArray,
28          String[] diffArray, String[] shimArray, String[] newShimArray, String vehicleID,
String workstationID, String orderID)
29  {
30      return WhiteboardCatalog.getInstance().makeNewValveAdjustmentWhiteboard(mechanicComment,
customerComment, cylinderNo, cylinderPos, inArray, tollArray, diffArray, shimArray, newShimArray, vehicleID,
workstationID, orderID);
31  } // End makeNewWhiteBoard method
32
33 }

```

ManageWorkstationController

```

1 package tweakmc.control;
2
3 import java.util.Vector;
4 import tweakmc.model.Result.Result;
5 import tweakmc.model.Result.ResultCatalog;
6 import tweakmc.model.VehicleCatalog;
7 import tweakmc.model.WorkstationCatalog;
8
9 /**
10  * Controller class for Workstations
11  * Between GUI and system
12  * @author clausPallisgaardBeck
13  */
14 public class ManageWorkstationController
15 {
16     /**
17      * Change the name on a workstation
18      * @param id for workstation to change name on
19      * @param newName for workstation
20      * @return true if changed else false
21      */
22     public boolean editNameOnWorkstation(String id, String newName)
23     {
24         return WorkstationCatalog.getInstance().editWorkstation(id, newName);
25     } //end method editNameOnWorkstation(String id, String newName)
26
27     /**
28      * Return the name on a given workstation for building GUI class or other
29      * @param id for workstation, name is wanted on
30      * @return the name for the given workstationID
31      */
32     public String getNameForWorkstation(String id)
33     {
34         return WorkstationCatalog.getInstance().getWorkstationName(id);
35     } //end getNameForWorkstation(String id)

```

```

36
37 /**
38  * Get a String with information on the requested vehicle
39  * @param vehicleID to get information from
40  * @return String with info if vehicleID exist else NULL
41  */
42 public String getInfoFromVehicleID(String vehicleID)
43 {
44     return VehicleCatalog.getInstance().getVehicleAndNameLine(vehicleID);
45 } //end method getInfoFromVehicleID(String vehicleID)
46
47 /**
48  * This method gets the results for the actual vehicle and workstaion
49  * @param vehicleID
50  * @param workstationID
51  * @return Vector<Result>
52  */
53 public Vector<Result> getResultForActVehicleAndWorkstation(String vehicleID, String workstationID)
54 {
55     return ResultCatalog.getInstance().getResultFromIDs(vehicleID, workstationID);
56 }
57 } //end class ManageWorkstationController()

```

OrderController

```

1
2
3 package tweakmc.control;
4
5 import java.util.ArrayList;
6 import tweakmc.dataaccess.OrderDAO;
7 import tweakmc.model.Order;
8 import tweakmc.model.OrderCatalog;
9
10
11 /**
12  * Control all methods between
13  * GUI and system
14  * for Order
15  * @author Nyvang
16  */
17 public class OrderController {
18
19
20     /**
21      * Creates an order
22      * @param orderType
23      * @param orderDescription
24      * @param descriptionSpareparts
25      * @param startDate
26      * @param expEndDate
27      * @param expPrice
28      * @param finishPrice
29      * @return true/false
30      */

```

```

31 public String addOrder(String orderType, String orderDescription, String descriptionSpareparts, int startDate, int
expEndDate,String expPrice, String finishPrice)
32 {
33     return OrderDAO.getInstance().createOrder(orderType, descriptionSpareparts, descriptionSpareparts, startDate,
expEndDate, expPrice, finishPrice);
34 }// End of addOrder method
35
36 /**
37  * Updates the selected order
38  * @param orderType
39  * @param orderDescription
40  * @param descriptionSpareparts
41  * @param startDate
42  * @param expEndDate
43  * @param expPrice
44  * @param finishPrice
45  * @return true/false
46  */
47 public boolean updateOrder(String orderID, String orderType, String orderDescription, String
descriptionSpareparts, int startDate, int expEndDate,String expPrice, String finishPrice)
48 {
49     return OrderDAO.getInstance().updateOrder(orderID, orderType, descriptionSpareparts, descriptionSpareparts,
startDate, expEndDate, expPrice, finishPrice);
50 }// End of upDateOrder method
51
52 /**
53  * Generic search method
54  * @param <T>
55  * @param seachCriteria
56  * @return objectType
57  */
58 public <T> ArrayList<Order> searchOrder (T seachCriteria)
59 {
60     return OrderCatalog.getInstance().searchOrder(seachCriteria);
61 }// End of searchOrder method
62
63 /**
64  * Passes the "go" from the GUI on downwards (creating a todolist)
65  */
66 public ArrayList<Order> generateTodo()
67 {
68     return OrderCatalog.getInstance().generateTodo();
69 }// End generatorTodo method
70
71
72 public boolean attachVehicle(String vehicleID, String orderID)
73 {
74     return OrderCatalog.getInstance().attachVehicle(vehicleID, orderID);
75 }// End of attachVehicle method
76
77 }// End of attachVehicle method

```

Utility layer (tweakmc.utility)

CheckInput

```
1 package tweakmc.utility;
2
3 /**
4  *
5  * @author Tvup
6  * @author Nyvang
7  * @author Claus
8  * @author Lars
9  */
10 public class CheckInput
11 {
12
13     /**
14      * Checks field for a valid integer
15      * @param isThisAnInt String to check
16      * @return true if integer
17      */
18     public static boolean isInt(String isThisAnInt)
19     {
20         try
21         {
22             Integer.parseInt(isThisAnInt);
23             return true;
24         } //end try
25
26         catch (NumberFormatException exp)
27         {
28             return false;
29         } //end catch
30     } //end isInt method
31
32     /**
33      * checks for double
34      * @param isThisAdouble
35      * @return
36      */
37     public static boolean isDouble(String isThisAdouble)
38     {
39         try
40         {
41             Double.parseDouble(isThisAdouble);
42             return true;
43         } //end try
44
45         catch (NumberFormatException exp)
46         {
47             return false;
48         } //end catch
49     } //end isDouble-method
50
51     /**
52      * Check if login initial is valid
```

```

53  * Can contain only letters
54  * Can not contain special signs or numbers
55  * @param initials string to check
56  * @return true if valid false if not
57  */
58  public static boolean checkInitials(String initials)
59  {
60      boolean correctExpression = initials.matches
61          ("[a-zæøåA-ZÆØÅ]" + "{" + initials.length() + "}");
62      return correctExpression;
63  } //end checkInitials method
64
65  /**
66   * Check if first name is valid
67   * Can contain letters and numbers
68   * Can not contain special signs
69   * @param isThisAName string to check
70   * @return true if valid false if not
71   */
72  public static boolean checkFirstName(String isThisAName)
73  {
74      boolean correctExpression = isThisAName.matches
75          ("[a-zæøåA-ZÆØÅ 0-9-]" + "{" + isThisAName.length() + "}");
76      return correctExpression;
77  } //end checkFirstName method
78
79  /**
80   * Check if last name is valid
81   * Can contain letters and numbers
82   * Can not contain special signs
83   * @param isThisALastName string to check
84   * @return true if valid false if not
85   */
86  public static boolean checkLastName(String isThisALastName)
87  {
88      boolean correctExpression = isThisALastName.matches
89          ("[a-zæøåA-ZÆØÅ 0-9-]" + "{" + isThisALastName.length() + "}");
90      return correctExpression;
91  } //end checkLastName method
92
93  /**
94   * Check if an address is valid
95   * Can contain letters and numbers
96   * Can not contain special signs
97   * @param isThisAnAddress string to check
98   * @return true if valid false if not
99   */
100  public static boolean checkAddress(String isThisAnAddress)
101  {
102      boolean correctExpression = isThisAnAddress.matches
103          ("[a-zæøåA-ZÆØÅ 0-9-.,]" + "{" + isThisAnAddress.length() + "}");
104      return correctExpression;
105  } //end checkAddress method
106
107  /**
108   * Check if an address2 is valid
109   * Can contain letters and numbers
110   * Can not contain special signs

```

```

111  * @param isThisAnAddress2 string to check
112  * @return true if valid false if not
113  */
114  public static boolean checkAddress2(String isThisAnAddress2)
115  {
116      boolean correctExpression = isThisAnAddress2.matches
117          ("[a-zæøåA-ZÆØÅ 0-9-.,]" + "{" + isThisAnAddress2.length() + "}");
118      return correctExpression;
119  } //end checkAddress2 method
120
121  /**
122   * Check if a zip is valid
123   * Can contain numbers between 3 and 4 characters
124   * Can not contain special signs and letters
125   * @param isThisAZip string to check
126   * @return true if valid false if not
127   */
128
129  public static boolean checkZip(String isThisAZip)
130  {
131      boolean correctExpression = isThisAZip.matches
132          ("[0-9]{3,4}");
133      return correctExpression;
134  } //end checkZip method
135
136  /**
137   * Check if an city or country is valid
138   * Can only contain letters
139   * Can not contain special signs and numbers
140   * @param isThisACityOrCountry string to check
141   * @return true if valid false if not
142   */
143  public static boolean checkCityOrCountry(String isThisACityOrCountry)
144  {
145      boolean correctExpression = isThisACityOrCountry.matches
146          ("[A-ZÆØÅa-zæøå]" + "{" + isThisACityOrCountry.length() + "}");
147      return correctExpression;
148  } //end checkCityOrCountry method
149
150  /**
151   * Check if an email or country is valid
152   * Can contain letters, numbers and special signs
153   * @param isThisAnEmail string to check
154   * @return true if valid false if not
155   */
156
157  public static boolean checkEmail(String isThisAnEmail)
158  {
159      boolean correctExpression = isThisAnEmail.matches
160          ("^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\\.[A-Za-z0-9-]+)*(\\.[A-Za-z]{2,4})$" + "{" +
isThisAnEmail.length() + "}");
161
162      return correctExpression;
163  } //end checkEmail method
164
165  /**
166   * check if company name is valid
167   * can contain only latters

```



```

168 * @param isThisAName String to check
169 * @return true if valid false if not
170 */
171
172 public static boolean checkName(String isThisAName)
173 {
174     boolean correctExpression = isThisAName.matches
175         ("[a-zæøåA-ZÆØÅ 0-9-]" + "{" + isThisAName.length() + "}");
176     return correctExpression;
177 }// end of checkName method
178
179 /**
180 * can contain anything
181 * @param isThisACurNo string to Check
182 * @return true if valid false if not
183 */
184 public static boolean checkCurNo(String isThisACurNo)
185 {
186     boolean correctExpression = isThisACurNo.matches
187         ("[a-zæøåA-ZÆØÅ 0-9-.,]" + "{" + isThisACurNo.length() + "}");
188     return correctExpression;
189 }
190 /**
191 * Can contain numbers, letters and special signs
192 * @param isThisABankInfo String to check
193 * @return true if valid false
194 */
195 public static boolean CheckBankInfo(String isThisABankInfo)
196 {
197     boolean correctExpression = isThisABankInfo.matches
198         ("[0-9]{10}" + "{" + isThisABankInfo.length() + "}");
199     return correctExpression;
200
201 }//end method CheckBankInfo
202
203
204
205 /**
206 * Check if a phone number is valid
207 * Can contain only numbers of 8 digits
208 * Can not contain letters and special signs
209 * @param isThisAPhone string to check
210 * @return true if valid false if not
211 */
212 public static boolean checkPhone(String isThisAPhone)
213 {
214     boolean correctExpression = isThisAPhone.matches
215         ("[0-9]{8}" + "{" + isThisAPhone.length() + "}");
216     return correctExpression;
217 }//end checkPhone method
218
219 /**
220 * Check if a HomePage is valid
221 * Can contain anything
222 * Can numers,contain letters and special signs
223 * @param isThisAHomePage string to check
224 * @return true if valid false if not
225 */

```

```

226
227 public static boolean checkHomePage(String isThisAHomePage)
228 {
229     boolean correctExpression = isThisAHomePage.matches
230         ("[a-zæøå.]" + "{" + isThisAHomePage.length() + "}");
231     return correctExpression;
232 }//end checkHomePage method:
233
234 /**
235  * Check if year is correct YYYY (1or2 0or9 0-9 0-9)
236  * @param yearToCheck if correct
237  * @return true if correct else false
238  */
239 public static boolean correctYear(String yearToCheck)
240 {
241     if(yearToCheck.length() <= 0)
242     {
243         return false;
244     }//end if
245     else
246     {
247         if (yearToCheck.substring(0, 1).matches("1"))
248         {
249             return yearToCheck.matches("[1][9][0-9][0-9]");
250         }//end if
251
252         if (yearToCheck.substring(0,1).matches("2"))
253         {
254             return yearToCheck.matches("[2][0-9][0-9][0-9]");
255         }//end if
256
257         else
258         {
259             return false;
260         }//end else
261     }//end else
262 }//end method correctYear
263
264 /**
265  * Checks the dates to make sure that endDate cannot be before startDate
266  * @param startDate
267  * @param endDate
268  * @return true if startdate > endDate. Else false
269  */
270 public static boolean compareDates(String startDate, String endDate)
271 {
272     if(endDate.compareTo(startDate)<0)
273     {
274         return true;
275     }//end if
276     else
277     {
278         return false;
279     }//end else
280 }//end compareDates method
281
282
283 }//end class

```

FileHandlerUtility

```
1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.utility;
7
8 /**
9  *
10 * @author clausPallisgaardBeck
11 */
12 public class FileHandlerUtility
13 {
14     //Instance variables
15
16     private static final String EXTSEPERATOR = ".";
17     /**
18      * Get the extension on a path
19      * @param originalDestination to take the extension from
20      * @return the extension to the filename
21      */
22     public static String getPathExtension(String originalDestination)
23     {
24         int startAt = originalDestination.lastIndexOf(EXTSEPERATOR) + 1;
25         String fileExtension = originalDestination.substring(startAt);
26         return "." + fileExtension;
27     } //end getPathExtension(String originalDestination)
28 } //end class FileHandlerUtility
```

FileWriterException

```
1 package tweakmc.utility;
2
3 import java.io.FileNotFoundException;
4 import java.io.FileWriter;
5 import java.io.IOException;
6 import java.io.PrintWriter;
7 import java.text.SimpleDateFormat;
8 import java.util.Calendar;
9 import java.util.GregorianCalendar;
10
11
12 /**
13  *
14  * @author NegoZiatoR
15  */
16 public class FileWriterException
17 {
18     private static SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yy - HH:mm:ss");
19     private static Calendar myC = new GregorianCalendar();
20
21     /**
22      * Write exceptions to file for error searching
23      * @param logMessage the exception
24      */
25     public static void writeLogFile(String logMessage)
```

```

26 {
27     PrintWriter pw;
28
29     try
30     {
31         pw = new PrintWriter(new FileWriter("logfile.txt", true));
32         pw.println(sdf.format(myC.getTime()) + " / " + logMessage + "\n");
33         pw.close();
34
35     } // end try
36
37     catch (FileNotFoundException ex)
38     {
39
40     } //end catch
41
42     catch (IOException ioex)
43     {
44
45     } // end catch
46 } // end method writeLogFile
47 } // end FileWriterController class

```

GUIHelpUtil

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 package tweakmc.utility;
7
8 // iText imports
9 import com.itextpdf.text.Document;
10 import com.itextpdf.text.DocumentException;
11 import com.itextpdf.text.Element;
12 import com.itextpdf.text.Image;
13 import com.itextpdf.text.PageSize;
14 import com.itextpdf.text.Paragraph;
15 import com.itextpdf.text.pdf.BaseFont;
16 import com.itextpdf.text.pdf.PdfContentByte;
17 import com.itextpdf.text.pdf.PdfReader;
18 import com.itextpdf.text.pdf.PdfStamper;
19 import com.itextpdf.text.pdf.PdfWriter;
20
21 // std imports
22 import java.io.File;
23 import java.io.FileOutputStream;
24 import java.io.IOException;
25 import java.util.HashMap;
26 import javax.swing.ImageIcon;
27 import javax.swing.JButton;
28 import javax.swing.JComponent;
29 import javax.swing.JOptionPane;
30 import tweakmc.view.GUIFrame;
31
32 /**
33  * Provide GUI with help pop ups for different windows
34  * Help GUI to get correct returns until connection is establish thure system

```

```

35 *
36 * @author clausPallisgaardBeck
37 */
38 public class GUIHelpUtil
39 {
40
41     //Help and about menus for GUIFrame
42     /**
43      * Help pop up window for Login
44      */
45     public static void helpLogin()
46     {
47         JOptionPane.showMessageDialog(null, "Enter your initials\n"
48             + "Characters allowed: a - å\n"
49             + "No numbers, spaces or special signs", "Help", JOptionPane.PLAIN_MESSAGE);
50     } //end method help
51
52     /**
53      * The help menu item
54      * @throws Exception
55      */
56     public static void helpItem() throws Exception
57     {
58         String imageLocation = "helpItemKJMC.jpg";
59         JOptionPane.showMessageDialog(null, "HELP!\n\n"
60             + "As the image to your left shows,\n"
61             + "there is only one way of helping yourself.\n"
62             + "\n"
63             + "Need more help?\n"
64             + "Read a book?", "MC Help",
65             JOptionPane.PLAIN_MESSAGE, new ImageIcon(imageLocation));
66     } //end helpItem()
67
68     /**
69      * About, version contact and so on
70      */
71     public static void aboutTweakMC() throws Exception
72     {
73         String imageLocation = "pictures/guipic/aboutWLF.jpg";
74         JOptionPane.showMessageDialog(null, "", "About TweakMC",
75             JOptionPane.PLAIN_MESSAGE, new ImageIcon(imageLocation));
76     } //end aboutTweakMC
77
78     ///////////////////////////////////////////////////////////////////
79     ///////////////////////////////////////////////////////////////////Utilities////////////////////////////////////
80
81     /**
82      * Open OS calculator
83      * @return true if open else false
84      */
85     public static boolean openCalculator()
86     {
87         try
88         {
89             String osName = System.getProperty("os.name");
90
91             if(osName.contains("Windows"))
92             {

```

```

93         Runtime.getRuntime().exec("calc");
94     } //end if
95
96     if(osName.contains("Lin"))
97     {
98         Runtime.getRuntime().exec("gcalctool");
99     } //end if
100
101     return true;
102 } //end try
103
104 catch(IOException ex)
105 {
106     FileWriterException.writeLogFile(GUIFrame.class.getName() + "/filemeu-calc/" + ex.getMessage());
107     return false;
108 } //end catch
109 } //end method openCalculator
110
111 ///////////////////////////////////////////////////////////////////
112 //Set components in GUIFrame////////////////////////////////////
113
114 /**
115  * Set the components to focus and button to default button
116  * @param focusComp act component
117  * @param defaultButton act button
118  */
119 public static void setFocusAndDefaultComponents(JComponent focusComp, JButton defaultButton)
120 {
121     GUIFrame.setFocusComponent(focusComp);
122     GUIFrame.setDefaultButton(defaultButton);
123 }
124
125 ///////////////////////////////////////////////////////////////////
126 //Delete methods under here when connection is establish thure system////
127
128 /**
129  * Try if initials is ok for login
130  * @param initials to check
131  * @return true if ok else false
132  */
133 public static boolean trySystemLogin(String initials)
134 {
135     return true;
136 } //end method trySystemLogin
137
138 /**
139  * Try if initials is ok for logout
140  * @param initials to check
141  * @return true if ok else false
142  */
143 public static boolean trySystemLogout(String initials)
144 {
145     return true;
146 } //end method trySystemLogout
147
148
149 /**
150  * Create, save and open a pdf

```

```

151  */
152  public static void doPdfThings()
153  {
154
155      Document document = new Document();
156      try
157      {
158          //Create a new directory for the pdfs
159          new File("c:/tweakmc/").mkdir();
160
161          //Create simple pdf
162          PdfWriter.getInstance(document,
163              new FileOutputStream("c:/tweakmc/TweakMC.pdf"));
164          document.open();
165          document.add(new Paragraph("Congratulations, you have created, saved and open a pdf"));
166          document.close();
167
168          //Create many sides pdf
169          PdfReader reader = new PdfReader("pdf/chaptersection.pdf");
170          int n = reader.getNumberOfPages();
171          // create a stamper that will copy the document to a new file
172          PdfStamper stamp = new PdfStamper(reader,
173              new FileOutputStream("c:/tweakmc/TweakMCmanysides.pdf"));
174          // adding some metadata
175          HashMap moreInfo = new HashMap();
176          moreInfo.put("Author", "TweakMC");
177          stamp.setMoreInfo(moreInfo);
178          // adding content to each page
179          int i = 0;
180          PdfContentByte under;
181          PdfContentByte over;
182          Image img = Image.getInstance("aboutWLF.jpg");
183          BaseFont bf =
184              BaseFont.createFont
185                  (BaseFont.HELVETICA, BaseFont.WINANSI, BaseFont.EMBEDDED);
186          img.setAbsolutePosition(200, 400);
187          while (i < n) {
188              i++;
189              // watermark under the existing page
190              under = stamp.getUnderContent(i);
191              under.addImage(img);
192              // text over the existing page
193              over = stamp.getOverContent(i);
194              over.beginText();
195              over.setFontAndSize(bf, 18);
196              over.setTextMatrix(30, 30);
197              over.showText("page " + i);
198              over.setFontAndSize(bf, 32);
199              over.showTextAligned
200                  (Element.ALIGN_LEFT, "DUPLICATE", 230, 430, 45);
201              over.endText();
202          }
203          // adding an extra page
204          stamp.insertPage(1, PageSize.A4);
205          over = stamp.getOverContent(1);
206          over.beginText();
207          over.setFontAndSize(bf, 18);
208          over.showTextAligned(Element.ALIGN_LEFT,

```

```

209         "Look for watermarks on following sides", 30, 600, 0);
210     over.endText();
211     // closing PdfStamper will generate the new PDF file
212     stamp.close();
213
214     //Open and present pdf
215     Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " +
216         "c:/tweakmc/TweakMC.pdf");
217     Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " +
218         "c:/tweakmc/TweakMCmanysides.pdf");
219 }//end try
220
221 catch (DocumentException de)
222 {
223     System.err.println(de.getMessage());
224 }//end catch
225
226 catch (IOException ioe)
227 {
228     System.err.println(ioe.getMessage());
229 }//end catch
230 }//end method doPdfThings()
231
232 /**
233  * Tells operating system to open e given PDF file
234  * @param path
235  * @return true / false
236  */
237 public static boolean previewPDFFiles(String path)
238 {
239     try
240     {
241         Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + path);
242         return true;
243     }//end try
244
245     catch (IOException ex)
246     {
247         FileWriterException.writeLogFile(GUIHelpUtil.class.getName() + " /previesPDFFiles/ " +
248             ex.getMessage());
249         return false;
250     }//end catch
251 }//end previewPDFFiles method
252 }// end GUIHelpUtil Class

```

Mail

```

1 package tweakmc.utility;
2
3 /**
4  *
5  * @author Nyvang
6  */
7

```



```

8 // JavaMail imports
9 import javax.mail.*;
10 import javax.mail.internet.*;
11 // Standard Java import
12 import java.util.Properties;
13 // TweakMC class import
14 import tweakmc.view.MailGUI;
15
16 /**
17  * Define Gmail properties, and has the responsibility to send email
18  * @author Nyvang
19  */
20
21 public class Mail {
22
23     private static final String SMTP_HOST_NAME = "smtp.gmail.com"; // SMTP hostname of the email account to
    send from, here gmail (until a WLF mail has been created)
24     private static final int SMTP_HOST_PORT = 465; // gmail port
25     private static final String SMTP_AUTH_USER = "welovefailing@gmail.com"; // userlogin to gmail
26     private static final String SMTP_AUTH_PWD = "fail123456"; // password
27     public static String MESSAGE_SEND; //variable for the send message text from the GUI
28     public static String MAIL_TO; // variable for specifying an email address - just because i can :)
29     public static String MAIL_CC; // User can now add a CC recipient for the mail.
30     private MailGUI mainObject; //mailGUI object for calling the showOKdialog method
31
32
33     /**
34      * This method sends emails (phew!)
35      * @throws Exception
36      */
37     public void mail() throws Exception{
38         Properties props = new Properties();
39         props.put("mail.transport.protocol", "smtps");
40         props.put("mail.smtps.host", SMTP_HOST_NAME);
41         props.put("mail.smtps.auth", "true");
42         MAIL_TO = "errorlog@welovefailing.com";
43
44         Session mailSession = Session.getDefaultInstance(props);
45         mailSession.setDebug(true);
46         Transport transport = mailSession.getTransport();
47
48         MimeMessage message = new MimeMessage(mailSession);
49         message.setSubject("Technical error report for TweakMC"); // subject of the email
50         message.setContent(MESSAGE_SEND, "text/plain"); // message to send
51
52         message.addRecipient(Message.RecipientType.TO,
53             new InternetAddress(MAIL_TO)); // who should receive the mail?
54         message.addRecipient(Message.RecipientType.CC,
55             new InternetAddress(MAIL_CC));
56
57         transport.connect
58             (SMTP_HOST_NAME, SMTP_HOST_PORT, SMTP_AUTH_USER, SMTP_AUTH_PWD); // it all in the
    blender and send it to the cloud
59
60         transport.sendMessage(message,
61             message.getRecipients(Message.RecipientType.TO));
62         transport.close();
63         mainObject.showOKdialog();

```

```

64
65 }//end mail metohd
66 }//end class Mail

```

OrderPDF

```

1 package tweakmc.utility;
2
3 /**
4  * Creates a *.pdf file from a user selected order from the todo-list
5  * Nyvang
6  */
7
8
9 //std java imports
10 import java.io.FileOutputStream;
11 import java.util.Date;
12
13 //iText specific imports
14 import com.itextpdf.text.Document;
15 import com.itextpdf.text.DocumentException;
16 import com.itextpdf.text.Font;
17 import com.itextpdf.text.Paragraph;
18 import com.itextpdf.text.Phrase;
19 import com.itextpdf.text.pdf.PdfPTable;
20 import com.itextpdf.text.pdf.PdfWriter;
21 import com.itextpdf.text.Jpeg;
22 import java.net.URL;
23
24 /**
25  * Creates and saves the pdf file from the variables recieved from the OrderGUI.
26  * The design of the PDF are defined below.
27  * @author Nyvang
28  */
29
30 public class OrderPDF
31 {
32     //instance variables
33     private static final String FILE = "KJMC_OrderPreview" + ".pdf"; //specify filename
34     private static Font catFont = new Font(Font.FontFamily.TIMES_ROMAN, 18, Font.BOLD); //specify fonts
35     private static Font subFont = new Font(Font.FontFamily.TIMES_ROMAN, 16, Font.BOLD);
36     private static Font smallBold = new Font(Font.FontFamily.TIMES_ROMAN, 12, Font.BOLD);
37     private static Font realSmallBold = new Font(Font.FontFamily.TIMES_ROMAN, 9, Font.BOLD);
38     private static Font logoFont = new Font(Font.FontFamily.TIMES_ROMAN, 5, Font.BOLD);
39     private static Font logoFontNormal = new Font(Font.FontFamily.TIMES_ROMAN, 5, Font.NORMAL);
40
41
42
43     //name the specific order attributes for the GUI to set
44     //these are made public for the OrderGUI to set the values (we should have used set/get methods)...
45     public static String orderID = "";
46     public static String orderType = "";
47     public static String startDate = "";
48     public static String endDate = "";
49     public static String orderDesc = "";
50     public static String orderSpareDesc = "";
51     public static String priceEstamate = "";

```

```

52 public static String vehicleID = "";
53
54 /**
55  * Creates the pdf document with the given values
56  * @return
57  */
58 public static boolean run()
59 {
60     try
61     {
62         Document document = new Document();
63         PdfWriter.getInstance(document, new FileOutputStream(FILE));
64         document.open();
65         addMetaData(document);
66         addTitlePage(document);
67         document.close();
68         return true;
69     } //end try
70     catch (Exception e) {
71         e.printStackTrace();
72         return false;
73     } //end catch
74
75 } //endm main
76
77 /**
78  * Adds metadata for the PDF file
79  * @param document object
80  */
81 private static void addMetaData(Document document)
82 {
83     document.addTitle("KJMC ToDo list");
84     document.addSubject("7 days todo");
85     document.addKeywords("Orders");
86     document.addAuthor("KJMC");
87     document.addCreator("Workshop");
88
89 } //end addMetaData method
90
91
92 /**
93  * Constructs the first page of the file
94  * @param document object
95  * @throws DocumentException
96  */
97 private static void addTitlePage(Document document)
98     throws DocumentException
99 {
100     Paragraph preface = new Paragraph();
101     addEmptyLine(preface, 1);
102     preface.add(new Paragraph("TweakMc Order preview", catFont));
103     addEmptyLine(preface, 1);
104     preface.add(new Paragraph("Report generated by: " + System.getProperty("user.name") + ", " + new
Date(), realSmallBold));
105     preface.add(new Paragraph( "
", smallBold));
106     addEmptyLine(preface, 3);
107     preface.add(new Paragraph("Prewieving order: " + orderID, subFont));

```

```

108     preface.add(new Paragraph("Order type:\t " + orderType,smallBold));
109     preface.add(new Paragraph("Start date:\t " + startDate,smallBold));
110     preface.add(new Paragraph("End date: \t" + endDate, smallBold));
111     preface.add(new Paragraph("Description: \t" + orderDesc, smallBold));
112     preface.add(new Paragraph("Spareparts required: \t" + orderSpareDesc, smallBold));
113     preface.add(new Paragraph("Price estimate:\t " + priceEstamate , smallBold));
114     preface.add(new Paragraph("Vehicle ID:\t " + vehicleID, smallBold));
115     addEmptyLine(preface, 3);
116     addEmptyLine(preface, 3);
117     addEmptyLine(preface, 3);
118     addEmptyLine(preface, 3);
119     addEmptyLine(preface, 3);
120     addEmptyLine(preface, 3);
121     addEmptyLine(preface, 3);
122
123
124
125     // create the tabel with logo and contact info
126     PdfPTable table = new PdfPTable(3);
127     table.setWidthPercentage(43);
128     PdfPCell c1 = new PdfPCell(new Phrase());
129     try
130     {
131         URL imageURL = new URL("http://dl.dropbox.com/u/4784543/kj_logo_small.jpg");
132         table.addCell(new Jpeg(imageURL));
133         table.addCell(new Phrase("KJ Motorcykler\n"
134             + "Marielundvej 46 C\n"
135             + "2730 Herlev\n"
136             , logoFont));
137         table.addCell(new Phrase("Telefon: 4494 1935\n"
138             + "E-mail: info@kjmotorcykler.dk\n"
139             + "WWW: www.kjmotorcykler.dk", logoFontNormal));
140     }//end try
141     catch (Exception e)
142     {
143     }//end catch
144     preface.add(table);
145     document.add(preface);
146     document.newPage();
147 }// end addTitlePage
148
149 /**
150  * Adds "n" empty lines
151  * @param paragraph
152  * @param number
153  */
154 private static void addEmptyLine(Paragraph paragraph, int number)
155 {
156     for (int i = 0; i < number; i++)
157     {
158         paragraph.add(new Paragraph(" "));
159     }//end for
160 }//end addEmptyLine
161 }//end Class OrderPDF

```

StartUpTest

```
1 /*
2  * This class is for testing of the database-connection
3  * It's not documented and isn't planned to be a part of the final distribution-
4  */
5 package tweakmc.utility;
6
7 import java.sql.Connection;
8 import java.sql.Driver;
9 import java.sql.DriverManager;
10 import java.sql.ResultSet;
11 import java.sql.ResultSetMetaData;
12 import java.sql.SQLException;
13 import java.sql.Statement;
14 import java.util.Enumuration;
15 import tweakmc.dataaccess.DBHandler;
16 import tweakmc.utility.FileWriterException;
17
18 /**
19  *
20  * @author Tvup
21  */
22 public class StartUpTest {
23     private Driver driverClass;
24     private Connection dbconn;
25     private ResultSetMetaData rsmd;
26     private String db_rev;
27     private Statement stmt;
28     private String addressToDriver;
29     private String connectionString;
30     private String dbUser;
31     private String dbPassword;
32
33
34     /**
35      * Constructor of StartUpTest. In this case everything is just received from the DBHandler
36      */
37     public StartUpTest()
38     {
39         db_rev = DBHandler.getInstance().db_rev;
40         addressToDriver = DBHandler.getInstance().addressToDriver;
41         connectionString = DBHandler.getInstance().connectionString;
42         dbUser = DBHandler.getInstance().dbUser;
43         dbPassword = DBHandler.getInstance().dbPassword;
44     } //end constructor
45
46     /**
47      * This method consists of try-catches which will go through procedures which
48      * we have found out could return in an exception
49      * Each exception will have further details printed to the console using the outputErrors method.
50      * Returns true if everything went well.
51      * @return boolean
52      */
53     public boolean startUpTests()
54     {
55         //Uncomment the following line if you want to show a list of the drivers.
56         //listDrivers();
```

```

57     try { setDriverClass(); }
58     catch (SQLException e)
59     {
60         outputErrors(e.getMessage(), "Have you installed your libraries?");
61         return false;
62     } //end catch
63
64     try { chooseDriver(); }
65     catch (ClassNotFoundException e)
66     {
67         outputErrors(e.getMessage(), "Library not installed");
68         return false;
69     } //end catch
70
71     System.out.println("\nDriver for jdbc:someDB://somehost/somedb");
72     System.out.println(" " + driverClass.getClass().getName());
73
74     try { setDbConnection(); }
75     catch (SQLException e)
76     {
77         if(addressToDriver.equals("com.mysql.jdbc.Driver"))
78             outputErrors(e.getMessage(), "Authentication failure. Have you entered username and password for
MySQL Server?");
79         else
80             outputErrors(e.getMessage(), "JAVA DB Server is not started");
81         return false;
82     } //end catch
83
84
85     try { createAndExecuteSQLStatement(); }
86     catch (SQLException e)
87     {
88         outputErrors(e.getMessage(), "Wrong DB revision");
89         return false;
90     } //end catch
91
92     return true;
93 } //end method StartUpTests
94
95 /**
96  * This method lists the drivers.
97  */
98 private static void listDrivers()
99 {
100     Enumeration driverList = DriverManager.getDrivers();
101     System.out.println("\nList of drivers:");
102     while (driverList.hasMoreElements())
103     {
104         Driver driverClass = (Driver) driverList.nextElement();
105         System.out.println(" " + driverClass.getClass().getName());
106     } //end while
107 } //end method listDrivers
108
109 /**
110  * This method sets the driver-class or throws an exception
111  * @return Driver
112  * @throws SQLException

```

```

114  */
115  public Driver setDriverClass() throws SQLException
116  {
117      driverClass = (Driver) DriverManager.getDriver(connectionString);
118      return driverClass;
119  } //end method setDriverClass
120
121  /**
122   * This method will choose a driver or throw an exception
123   * @return Class
124   * @throws ClassNotFoundException
125   */ public Class chooseDriver() throws ClassNotFoundException
126  {
127      return Class.forName(addressToDriver);
128  } //end method chooseDriver
129
130  /**
131   * This method will establish an connection to the database or throw an exception
132   * @return Connection
133   * @throws SQLException
134   */
135  public Connection setDbConnection() throws SQLException
136  {
137      dbconn = DriverManager.getConnection(connectionString,dbUser,dbPassword);
138      return dbconn;
139  } //end method setDbConnection
140
141  /**
142   * This method will check if the database is updated to the latest revision or throw an exception
143   * @return boolean
144   * @throws SQLException
145   */
146  public boolean createAndExecuteSQLStatement() throws SQLException
147  {
148      stmt = dbconn.createStatement();
149      ResultSet results = stmt.executeQuery("select * from db_rev");
150      rsmd = results.getMetaData();
151      if(rsmd.getColumnLabel(1).toString().equals(db_rev))
152      {
153          System.out.println("Database is up-to-date");
154          stmt.close();
155          return true;
156      } //end if
157
158      else
159      {
160          System.out.println("Database revision doesn't comply with this revision\nExpected: " + db_rev + "\nFound:
" + rsmd.getColumnLabel(1));
161          stmt.close();
162          return false;
163      } //end else
164  } //end method createAndExceuteSQLStatement
165
166  /**
167   * This method will print the errors to the console and the log-file
168   * @param exceptionMessage
169   * @param moreDetailsOnError
170   */

```

```

171 public void outputErrors(String exceptionMessage, String moreDetailsOnError)
172 {
173     System.err.println(moreDetailsOnError + "\nException: " + exceptionMessage);
174     FileWriterException.writeLogFile(StartupTest.class.getName() + exceptionMessage + "\n" +
moreDetailsOnError);
175 } //end method outputErrors
176
177
178 } //end class

```

View layer (tweakmc.view)

AttachOwnerGUI

```

1 /*
2  * This class is to attach an owner to a Vehicle
3  * It takes the vehicle ID in parameters to know witch vehicle the owner
4  * should be attached to.
5  *
6  */
7
8 /*
9  * AttachOwnerGUI.java
10 *
11 * Created on 22-11-2010, 13:33:48
12 */
13
14 package tweakmc.view;
15
16 import java.text.SimpleDateFormat;
17 import java.util.ArrayList;
18 import java.util.Calendar;
19 import java.util.GregorianCalendar;
20 import java.util.List;
21 import java.util.Timer;
22 import java.util.TimerTask;
23 import java.util.Vector;
24 import javax.swing.DefaultComboBoxModel;
25 import javax.swing.JOptionPane;
26 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
27 import tweakmc.control.ManageOwnerController;
28 import tweakmc.control.ManageVehicleController;
29 import tweakmc.model.Owner;
30
31 /**
32  *
33  * @author NegoZiatoR
34  */
35 public class AttachOwnerGUI extends javax.swing.JFrame
36 {
37     // Instance Variables
38     private ManageOwnerController moc = new ManageOwnerController();
39     private ManageVehicleController mvc = new ManageVehicleController();
40     private static AttachOwnerGUI instance;

```



```

41 private Vector namesVector = OwnerGUI.getNamesVector();
42 private String vehicleID;
43 private String valueOfSearchField;
44 private String[] arrActFoundsFirstnameLastNamePhoneNumber;
45 private List<String> actFoundsFirstnameLastNamePhoneNumber;
46 private List<String> actFoundsOwnerID;
47 private List<Owner> actFoundsObjects;
48 private String[] arrActFoundsOwnerID;
49
50 private Owner selectedOwner;
51
52 private static SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
53 private static Calendar myC = new GregorianCalendar();
54
55
56 /** Creates new form AttachOwnerGUI */
57 private AttachOwnerGUI(String vehicleID)
58 {
59     initComponents();
60     this.vehicleID = vehicleID;
61
62     // Show time
63     Timer timer = new Timer ();
64     timer.schedule (new ShowTime (), 0, 1000);
65 }
66
67 /**
68  * Singleton method for AttachOwnerGUI
69  * @param vehicleID on vehicle
70  * @return ONE instance of AttachOwnerGUI
71  */
72 public static AttachOwnerGUI getInstance(String vehicleID)
73 {
74     if(instance == null)
75     {
76         instance = new AttachOwnerGUI(vehicleID);
77     }
78     instance.setVisible(true);
79     return instance;
80 }
81
82 /** This method is called from within the constructor to
83  * initialize the form.
84  * WARNING: Do NOT modify this code. The content of this method is
85  * always regenerated by the Form Editor.
86  */
87 @SuppressWarnings("unchecked")
88 // <editor-fold defaultstate="collapsed" desc="Generated Code">
89 private void initComponents() {
90
91     jLabel1 = new javax.swing.JLabel();
92     searchOwnerComboBox = new javax.swing.JComboBox();
93     searchOwnerComboBox.setEditable(true);
94     AutoCompleteDecorator.decorate(searchOwnerComboBox);
95     searchOwnerComboBox.setModel(new DefaultComboBoxModel(namesVector));
96     jButton1 = new javax.swing.JButton();
97     jScrollPane1 = new javax.swing.JScrollPane();
98     foundOwnersList = new javax.swing.JList();

```

```

99     searchOwnerButton = new javax.swing.JButton();
100     timeLabel = new javax.swing.JLabel();
101
102     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
103
104     jLabel1.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
105     jLabel1.setText("Search Owner");
106
107     searchOwnerComboBox.addActionListener(new java.awt.event.ActionListener() {
108         public void actionPerformed(java.awt.event.ActionEvent evt) {
109             searchOwnerComboBoxActionPerformed(evt);
110         }
111     });
112
113     jButton1.setText("Attach");
114     jButton1.addActionListener(new java.awt.event.ActionListener() {
115         public void actionPerformed(java.awt.event.ActionEvent evt) {
116             jButton1ActionPerformed(evt);
117         }
118     });
119
120     jScrollPane1.setViewportView(foundOwnersList);
121
122     searchOwnerButton.setText("Search");
123     searchOwnerButton.addActionListener(new java.awt.event.ActionListener() {
124         public void actionPerformed(java.awt.event.ActionEvent evt) {
125             searchOwnerButtonActionPerformed(evt);
126         }
127     });
128
129     timeLabel.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
130     timeLabel.setText("jLabel2");
131
132     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
133     getContentPane().setLayout(layout);
134     layout.setHorizontalGroup(
135         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
136             .addGroup(layout.createSequentialGroup()
137                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
138                     .addComponent(searchOwnerComboBox, javax.swing.GroupLayout.DEFAULT_SIZE, 244,
Short.MAX_VALUE)
139                     .addComponent(searchOwnerButton)
140                     .addComponent(jButton1)
141                     .addGroup(layout.createSequentialGroup()
142                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
143                             .addComponent(jLabel1)
144                             .addGap(18, 18, 18)
145                             .addComponent(timeLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 117,
Short.MAX_VALUE)))
146                     .addContainerGap())
147                 .addContainerGap())
148     );
149     layout.setVerticalGroup(
150         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
151             .addGroup(layout.createSequentialGroup()
152                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
153                     .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

154         .addComponent(timeLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 22, Short.MAX_VALUE)
155         .addComponent(jLabel1))
156     .addGap(18, 18, 18)
157     .addComponent(searchOwnerComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
158     .addGap(18, 18, 18)
159     .addComponent(searchOwnerButton)
160     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
161     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE)
162     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
163     .addComponent(jButton1)
164     .addContainerGap()
165 );
166
167 pack();
168 }// </editor-fold>
169
170 private void searchOwnerComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
171     // TODO add your handling code here:
172 }
173
174 private void searchOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
175     // TODO add your handling code here:
176     searchOwner();
177 }
178
179 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
180     // TODO add your handling code here:
181
182     if(foundOwnersList.getSelectedIndex() >=0)
183     {
184         int indexPositionOfSelectedOwner= foundOwnersList.getSelectedIndex();
185
186         String ownerID = arrActFoundsOwnerID[indexPositionOfSelectedOwner];
187
188         int i = 0;
189         boolean found = false;
190         while (actFoundsObjects.size() > i && !found )
191         {
192             if(actFoundsObjects.get(i).getID().equals(ownerID))
193             {
194                 selectedOwner = actFoundsObjects.get(i);
195                 found = true;
196             }//end if
197             else
198             {
199                 i++;
200             }//end else
201         }//end while
202         if(mvc.attachOwner(this.vehicleID, ownerID,1))
203         {
204             JOptionPane.showMessageDialog(null, selectedOwner.getFirstName() + " " +
selectedOwner.getLastName() + " successfully attached to selected vehicle");
205             this.dispose();
206         }
207
208     }
209 }

```

```

210
211 /**
212  * Search for a Owner
213  */
214 public void searchOwner() {
215     //Get the text from the searchfield
216     setValueOfSearchField((String) searchOwnerComboBox.getSelectedItem());
217     String searchCriteria = valueOfSearchField;
218     //Set the arrays with the searchResults
219     enterSearchDetails(searchCriteria);
220     //If nothing is found, display a label to the user
221     if (arrActFoundsFirstnameLastNamePhoneNumber.length == 0) {
222         foundOwnersList.setModel(new javax.swing.AbstractListModel() {
223
224             public int getSize() {
225                 return arrActFoundsFirstnameLastNamePhoneNumber.length;
226             }
227
228             public Object getElementAt(int i) {
229                 return arrActFoundsFirstnameLastNamePhoneNumber[i];
230             }
231         });
232     } //end if
233     //If there's results present them.
234     else {
235         //Clear the actual JList and make a new one
236
237         foundOwnersList.setModel(new javax.swing.AbstractListModel() {
238
239             public int getSize() {
240                 return arrActFoundsFirstnameLastNamePhoneNumber.length;
241             }
242
243             public Object getElementAt(int i) {
244                 return arrActFoundsFirstnameLastNamePhoneNumber[i];
245             }
246         });
247     } //end else
248 }
249
250 private String setValueOfSearchField(String valueOfSearchField)
251 {
252     return this.valueOfSearchField = valueOfSearchField;
253 } //end setValueOfSearchField
254
255 /**
256  * This method ask the controller-layer to perform a search for an owner
257  * Search results are set from the results of the controller-layer as multiple
258  * lists of Owner-objects, strings and ints.
259  *
260  * The variables firstName, lastName, and phoneNumber in every object in the
261  * actFoundsObjects is put together as a string and inserted to an
262  * arraylist(actFoundsFirstnameLastNamePhoneNumber) which will be converted
263  * to an array.
264  *
265  * The same applies to the list of OwnerIDs
266  * @param searchCriteria
267  * @return void

```

```

268  */
269  public void enterSearchDetails(String searchCriteria)
270  {
271      //This ArrayList will hold the Firstname, lastname and phonenumber of the search results
272      actFoundsFirstnameLastNamePhoneNumber = new ArrayList<String>();
273      //This ArrayList will hold the OwnerID of the search results
274      actFoundsOwnerID = new ArrayList<String>();
275      //This ArrayList will hold the Owner-objects of the search results
276      actFoundsObjects = new ArrayList<Owner>();
277
278      actFoundsObjects = moc.searchOwner(searchCriteria);
279
280      int i = 0;
281      for (Owner o : actFoundsObjects)
282      {
283          //ID, Firstname, lastname, phonenumber and ownerID goes to the respective ArrayLists
284          actFoundsFirstnameLastNamePhoneNumber.add("(" + o.getID() + ") " + o.getFirstname() + " " +
o.getLastname() + " (" + o.getPhone() + ") ");
285          actFoundsOwnerID.add(o.getID());
286          i++;
287      } //end for each
288
289      //Here the ArrayList is converted to arrays (1-dim) (for presentation purpose)
290      arrActFoundsFirstnameLastNamePhoneNumber = actFoundsFirstnameLastNamePhoneNumber.toArray(new
String[0]);
291      arrActFoundsOwnerID = actFoundsOwnerID.toArray(new String[0]);
292
293  }
294
295  //////////// PRIVATE INNER CLASS ////////////
296  class ShowTime extends TimerTask
297  {
298      public void run ()
299      {
300          timeLabel.setText(sdf.format(myC.getTime()));
301          myC.add (Calendar.SECOND, 1);
302          sdf.setCalendar (myC);
303      }
304  }
305
306  /**
307   * @param args the command line arguments
308   */
309  public static void main(String args[]) {
310      java.awt.EventQueue.invokeLater(new Runnable() {
311          private String vehicleID;
312          public void run() {
313              new AttachOwnerGUI(this.vehicleID).setVisible(true);
314          }
315      });
316  }
317
318  // Variables declaration - do not modify
319  private javax.swing.JList foundOwnersList;
320  private javax.swing.JButton jButton1;
321  private javax.swing.JLabel jLabel1;
322  private javax.swing.JScrollPane jScrollPane1;
323  private javax.swing.JButton searchOwnerButton;

```

```

324 private javax.swing.JComboBox searchOwnerComboBox;
325 private javax.swing.JLabel timeLabel;
326 // End of variables declaration
327
328 }

```

AttachVehidleGUI

```

1 /*
2  * This class is to attach an owner to a Vehicle
3  * It takes the owner ID in parameters to know witch owner
4  * should be attached to a vehicle.
5  *
6  */
7
8 /*
9  * AttachVehicleGUI.java
10 *
11 * Created on 22-11-2010, 13:33:48
12 */
13
14 package tweakmc.view;
15
16 import java.text.SimpleDateFormat;
17 import java.util.ArrayList;
18 import java.util.Calendar;
19 import java.util.GregorianCalendar;
20 import java.util.List;
21 import java.util.Timer;
22 import java.util.TimerTask;
23 import java.util.Vector;
24 import javax.swing.DefaultComboBoxModel;
25 import javax.swing.JOptionPane;
26 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
27 import tweakmc.control.ManageVehicleController;
28 import tweakmc.model.Vehicle;
29
30 /**
31  *
32  * @author NegoZiatoR
33  */
34 public class AttachVehicleGUI extends javax.swing.JFrame
35 {
36     private ManageVehicleController mvc = new ManageVehicleController();
37     private Vector brandVector = VehicleGUI.getBrandVector();
38     private String ownerID;
39     private String orderID;
40     private String valueOfSearchField;
41     private static AttachVehicleGUI instance;
42     private List<Vehicle> actFoundsVehicleObjects;
43     private List<String> actFoundsBrandModel;
44     private List<String> actFoundVehicle;
45     private String[] arrActFoundsVehicleID;
46     private String[] arrActFoundsBrandModel;
47     private Vehicle selectedVehicle;
48
49     private static SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

```

```

50 private static Calendar myC = new GregorianCalendar();
51
52 /** Creates new form AttachVehicleGUI */
53 private AttachVehicleGUI(String ownerID)
54 {
55     initComponents();
56     this.ownerID = ownerID;
57
58     // Show time
59     Timer timer = new Timer ();
60     timer.schedule (new ShowTime (), 0, 1000);
61 }
62
63 /**
64  * Singleton method for AttachVehicleGUI
65  * @param ownerID on owner
66  * @return ONE instance of AttachVehicleGUI
67  */
68 public static AttachVehicleGUI getInstance(String ownerID)
69 {
70     if(instance == null)
71     {
72         instance = new AttachVehicleGUI(ownerID);
73     }
74     instance.setVisible(true);
75     return instance;
76 }
77
78 /** This method is called from within the constructor to
79  * initialize the form.
80  * WARNING: Do NOT modify this code. The content of this method is
81  * always regenerated by the Form Editor.
82  */
83 @SuppressWarnings("unchecked")
84 // <editor-fold defaultstate="collapsed" desc="Generated Code">
85 private void initComponents() {
86
87     labelSearchVehicle = new javax.swing.JLabel();
88     searchVehicleComboBox = new javax.swing.JComboBox();
89     searchVehicleComboBox.setEditable(true);
90     autoCompleteDecorator.decorate(searchVehicleComboBox);
91     searchVehicleComboBox.setModel(new DefaultComboBoxModel(brandVector));
92     attachVehicleButton = new javax.swing.JButton();
93     jScrollPane1 = new javax.swing.JScrollPane();
94     foundVehiclesList = new javax.swing.JList();
95     searchVehicleButton = new javax.swing.JButton();
96     timeLabel = new javax.swing.JLabel();
97
98     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
99
100    labelSearchVehicle.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
101    labelSearchVehicle.setText("Search Vehicle");
102
103    searchVehicleComboBox.addItemListener(new java.awt.event.ItemListener() {
104        public void itemStateChanged(java.awt.event.ItemEvent evt) {
105            searchVehicleComboBoxItemStateChanged(evt);
106        }
107    });

```

```

108 searchVehicleComboBox.addActionListener(new java.awt.event.ActionListener() {
109     public void actionPerformed(java.awt.event.ActionEvent evt) {
110         searchVehicleComboBoxActionPerformed(evt);
111     }
112 });
113
114 attachVehicleButton.setText("Attach");
115 attachVehicleButton.addActionListener(new java.awt.event.ActionListener() {
116     public void actionPerformed(java.awt.event.ActionEvent evt) {
117         attachVehicleButtonActionPerformed(evt);
118     }
119 });
120
121 jScrollPane1.setViewportView(foundVehiclesList);
122
123 searchVehicleButton.setText("Search");
124 searchVehicleButton.addActionListener(new java.awt.event.ActionListener() {
125     public void actionPerformed(java.awt.event.ActionEvent evt) {
126         searchVehicleButtonActionPerformed(evt);
127     }
128 });
129
130 timeLabel.setHorizontalAlignment(javax.swing.SwingConstants.RIGHT);
131 timeLabel.setText("jLabel1");
132
133 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
134 getContentPane().setLayout(layout);
135 layout.setHorizontalGroup(
136     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
137     .addGroup(layout.createSequentialGroup()
138         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
139             .add(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 244,
140 Short.MAX_VALUE)
141             .addComponent(searchVehicleComboBox, javax.swing.GroupLayout.Alignment.TRAILING, 0, 244,
142 Short.MAX_VALUE)
143             .addComponent(searchVehicleButton)
144             .addComponent(attachVehicleButton)
145             .addGroup(layout.createSequentialGroup()
146                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
147                     .add(layout.createSequentialGroup()
148                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
149                             .add(layout.createSequentialGroup()
150                                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
151                                     .add(layout.createSequentialGroup()
152                                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
153                                             .add(layout.createSequentialGroup()
154                                                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
155                                                     .add(layout.createSequentialGroup()
156                                                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
157                                                             .add(layout.createSequentialGroup()
158                                                                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
159                                                                     .add(layout.createSequentialGroup()
160                                                                         .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
161                                                                 .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```



```

162         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE)
163         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
164         .addComponent(attachVehicleButton)
165         .addContainerGap()
166     );
167
168     pack();
169 }// </editor-fold>
170
171 private void searchVehicleComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
172     // TODO add your handling code here:
173 }
174
175 private void searchVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
176     // TODO add your handling code here:
177     searchVehicle();
178 }
179
180 private void attachVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
181     // TODO add your handling code here:
182     if(foundVehiclesList.getSelectedIndex() >= 0)
183     {
184         int indexPositionOfSelectedVehicle = foundVehiclesList.getSelectedIndex();
185         String vehicleID = arrActFoundsVehicleID[indexPositionOfSelectedVehicle];
186
187         boolean found = false;
188         int i = 0;
189         Vehicle v = null;
190
191         // Search for the right selected vehilce in the found vehicles
192         while(i < actFoundsVehicleObjects.size() && !found)
193         {
194             v = actFoundsVehicleObjects.get(i);
195             if(v.getVehicleID().equals(vehicleID))
196             {
197                 found = true;
198                 selectedVehicle = actFoundsVehicleObjects.get(i);
199             }// End inner if
200             else
201             {
202                 i++;
203             }// End inner else
204         }// End while
205         if(mvc.attachOwner(vehicleID, this.ownerID, 1 ))
206         {
207             JOptionPane.showMessageDialog(null, selectedVehicle.getBrand() + " " + selectedVehicle.getModel() + "
successfully attached to selected owner");
208             this.dispose();
209         }
210     }//End If
211
212 }
213
214 private void searchVehicleComboBoxItemStateChanged(java.awt.event.ItemEvent evt) {
215     // TODO add your handling code here:
216     //searchVehicle();
217 }
218

```

```

219  /**
220   * Search for a Owner
221   */
222  public void searchVehicle()
223  {
224      //Get the text from the searchfield
225      setValueOfSearchField((String) searchVehicleComboBox.getSelectedItem());
226      String searchCriteria = valueOfSearchField;
227      //Set the arrays with the searchResults
228      enterSearchDetails(searchCriteria);
229      //If nothing is found, display a label to the user
230      if (arrActFoundsBrandModel.length == 0) {
231          foundVehiclesList.setModel(new javax.swing.AbstractListModel() {
232
233              public int getSize() {
234                  return arrActFoundsBrandModel.length;
235              }
236
237              public Object getElementAt(int i) {
238                  return arrActFoundsBrandModel[i];
239              }
240          });
241      } //end if
242      //If there's results present them.
243      else {
244          //Clear the actual JList and make a new one
245
246          foundVehiclesList.setModel(new javax.swing.AbstractListModel() {
247
248              public int getSize() {
249                  return arrActFoundsBrandModel.length;
250              }
251
252              public Object getElementAt(int i) {
253                  return arrActFoundsBrandModel[i];
254              }
255          });
256      } //end else
257  }
258
259  private String setValueOfSearchField(String valueOfSearchField)
260  {
261      return this.valueOfSearchField = valueOfSearchField;
262  } //end setValueOfSearchField
263
264
265  /**
266   * This method ask the controller-layer to perform a search for a vehicle
267   * Search results are set from the controller-layer as multiply list of Vehicle-objects, String and ints
268   *
269   * The variables model and vType in every object in the actFoundsVehicleObjects list
270   * is put together as a string and inserted to an
271   * arraylist( actFoundsBrandModel ) which will be converted to an array
272   *
273   * The same applies to the list of vehilce id
274   * @param searchCriteria
275   * @return void
276   */

```

```

277 public void enterSearchDetails(String searchCriteria)
278 {
279     // This arrayLsit will hold the vehicles Brand and Model for presentation
280     actFoundsBrandModel = new ArrayList<String>();
281
282     // This arrayList will hold the vehicles ID
283     actFoundVehicle = new ArrayList<String>();
284
285     // This arrayList will hold the vehicle object
286     actFoundsVehicleObjects = mvc.searchVehicle(searchCriteria);
287     int i = 0;
288     for (Vehicle v : actFoundsVehicleObjects)
289     {
290         // Brand, model and id in the respective lists
291         String licensPlate = v.getLicensPlate();
292         String chassisNumber = v.getChassisNumber();
293
294         // If both licensplate and chassis number is empty
295         if(licensPlate.equals("") && chassisNumber.equals(""))
296         {
297             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel());
298         }
299         // If only chassisnumber is empty
300         else if(!licensPlate.equals("") && chassisNumber.equals(""))
301         {
302             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
303         }
304         // If only licensplate is empty
305         else if(licensPlate.equals("") && !chassisNumber.equals(""))
306         {
307             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getChassisNumber() + " )");
308         }
309         // Neither licensplate or chassisnumber is empty
310         else
311         {
312             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
313         }
314         actFoundVehicle.add(v.getVehicleID());
315         i++;
316     } //end for each
317
318
319     arrActFoundsBrandModel = actFoundsBrandModel.toArray(new String[0]);
320     arrActFoundsVehicleID = actFoundVehicle.toArray(new String[0]);
321
322 }
323
324
325 /////////////// PRIVATE INNER CLASS //////////////////////
326 class ShowTime extends TimerTask
327 {
328     public void run ()
329     {
330         timeLabel.setText(sdf.format(myC.getTime()));
331         myC.add (Calendar.SECOND, 1);

```

```

332     sdf.setCalendar (myC);
333 }
334 }
335
336 /**
337  * @param args the command line arguments
338  */
339 public static void main(String args[]) {
340     java.awt.EventQueue.invokeLater(new Runnable() {
341         private String vehicleID;
342         public void run() {
343             new AttachVehicleGUI(this.vehicleID).setVisible(true);
344         }
345     });
346 }
347
348 // Variables declaration - do not modify
349 private javax.swing.JButton attachVehicleButton;
350 private javax.swing.JList foundVehiclesList;
351 private javax.swing.JScrollPane jScrollPane1;
352 private javax.swing.JLabel labelSearchVehicle;
353 private javax.swing.JButton searchVehicleButton;
354 private javax.swing.JComboBox searchVehicleComboBox;
355 private javax.swing.JLabel timeLabel;
356 // End of variables declaration
357
358 }

```

AttachVehicleToOrderGUI

```

1 /*
2  * This class is to attach a vehicle to an order
3  * It takes the order ID in parameters to know witch order
4  * the vehicle should be attached to.
5  *
6  */
7 /*
8  * AttachVehicleToOrderGUI.java
9  *
10 * Created on 03-12-2010, 11:33:48
11 */
12
13 package tweakmc.view;
14
15 import java.util.ArrayList;
16 import java.util.List;
17 import java.util.Vector;
18 import javax.swing.DefaultComboBoxModel;
19 import javax.swing.JOptionPane;
20 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
21 import tweakmc.control.ManageVehicleController;
22 import tweakmc.model.Vehicle;
23
24 /**
25  *
26  * @author Nyvang
27  */

```

```

28 public class AttachVehicleToOrderGUI extends javax.swing.JFrame {
29
30     //instance variables
31     private ManageVehicleController oc = new ManageVehicleController();
32     private Vector brandVector = VehicleGUI.getBrandVector();
33     private String orderID;
34     private String valueOfSearchField;
35     private static AttachVehicleToOrderGUI instance;
36     private List<Vehicle> actFoundsVehicleObjects;
37     private List<String> actFoundsBrandModel;
38     private List<String> actFoundVehicle;
39     private String[] arrActFoundsVehicleID;
40     private String[] arrActFoundsBrandModel;
41     private Vehicle selectedVehicle;
42
43
44     /** Creates new form AttachVehicleToOrderGUI */
45     private AttachVehicleToOrderGUI(String orderID)
46     {
47         initComponents();
48         this.orderID = orderID;
49     }
50
51     /**
52      * Singleton method for AttachVehicleToOrderGUI
53      * @param orderID on order
54      * @return ONE instance of AttachVehicleToOrderGUI
55      */
56     public static AttachVehicleToOrderGUI getInstance(String orderID)
57     {
58         if(instance == null)
59         {
60             instance = new AttachVehicleToOrderGUI(orderID);
61         }
62         instance.setVisible(true);
63         return instance;
64     }
65
66     /** This method is called from within the constructor to
67      * initialize the form.
68      * WARNING: Do NOT modify this code. The content of this method is
69      * always regenerated by the Form Editor.
70      */
71     @SuppressWarnings("unchecked")
72     // <editor-fold defaultstate="collapsed" desc="Generated Code">
73     private void initComponents() {
74
75         labelSearchVehicle = new javax.swing.JLabel();
76         searchVehicleComboBox = new javax.swing.JComboBox();
77         searchVehicleComboBox.setEditable(true);
78         AutoCompleteDecorator.decorate(searchVehicleComboBox);
79         searchVehicleComboBox.setModel(new DefaultComboBoxModel(brandVector));
80         attachVehicleButton = new javax.swing.JButton();
81         jScrollPane1 = new javax.swing.JScrollPane();
82         foundVehiclesList = new javax.swing.JList();
83         searchVehicleButton = new javax.swing.JButton();
84
85         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

```

```

86
87     labelSearchVehicle.setFont(new java.awt.Font("Tahoma", 0, 18));
88     labelSearchVehicle.setText("Search Vehicle");
89
90     searchVehicleComboBox.addItemListener(new java.awt.event.ItemListener() {
91         public void itemStateChanged(java.awt.event.ItemEvent evt) {
92             searchVehicleComboBoxItemStateChanged(evt);
93         }
94     });
95     searchVehicleComboBox.addActionListener(new java.awt.event.ActionListener() {
96         public void actionPerformed(java.awt.event.ActionEvent evt) {
97             searchVehicleComboBoxActionPerformed(evt);
98         }
99     });
100
101     attachVehicleButton.setText("Attach");
102     attachVehicleButton.addActionListener(new java.awt.event.ActionListener() {
103         public void actionPerformed(java.awt.event.ActionEvent evt) {
104             attachVehicleButtonActionPerformed(evt);
105         }
106     });
107
108     jScrollPane1.setViewportView(foundVehiclesList);
109
110     searchVehicleButton.setText("Search");
111     searchVehicleButton.addActionListener(new java.awt.event.ActionListener() {
112         public void actionPerformed(java.awt.event.ActionEvent evt) {
113             searchVehicleButtonActionPerformed(evt);
114         }
115     });
116
117     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
118     getContentPane().setLayout(layout);
119     layout.setHorizontalGroup(
120         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
121             .addGroup(layout.createSequentialGroup()
122                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
123                     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 244,
Short.MAX_VALUE)
124                     .addComponent(labelSearchVehicle, javax.swing.GroupLayout.DEFAULT_SIZE, 244,
Short.MAX_VALUE)
125                     .addComponent(searchVehicleComboBox, javax.swing.GroupLayout.Alignment.TRAILING, 0, 244,
Short.MAX_VALUE)
126                     .addComponent(searchVehicleButton)
127                     .addComponent(attachVehicleButton))
128                 .addContainerGap())
129     );
130
131     layout.setVerticalGroup(
132         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
133             .addGroup(layout.createSequentialGroup()
134                 .addGap(19, 19, 19)
135                 .addComponent(labelSearchVehicle)
136                 .addGap(18, 18, 18)
137                 .addComponent(searchVehicleComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
138                 .addGap(18, 18, 18)
139                 .addComponent(searchVehicleButton)

```

```

140     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
141     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 147, Short.MAX_VALUE)
142     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
143     .addComponent(attachVehicleButton)
144     .addContainerGap()
145 );
146
147 pack();
148 }// </editor-fold>
149
150 private void searchVehicleComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
151     // TODO add your handling code here:
152 }
153
154 private void searchVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
155     // TODO add your handling code here:
156     searchVehicle();
157 }
158
159 private void attachVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
160     // TODO add your handling code here:
161     if(foundVehiclesList.getSelectedIndex() >= 0)
162     {
163         int indexPositionOfSelectedVehicle = foundVehiclesList.getSelectedIndex();
164         String vehicleID = arrActFoundsVehicleID[indexPositionOfSelectedVehicle];
165
166         boolean found = false;
167         int i = 0;
168         Vehicle v = null;
169
170         // Search for the right selected vehilce in the found vehicles
171         while(i < actFoundsVehicleObjects.size() && !found)
172         {
173             v = actFoundsVehicleObjects.get(i);
174             if(v.getVehicleID().equals(vehicleID))
175             {
176                 found = true;
177                 selectedVehicle = actFoundsVehicleObjects.get(i);
178             }// End inner if
179             else
180             {
181                 i++;
182             }// End inner else
183         }// End while
184         if(oc.attachOwner(vehicleID, this.orderID, 1 ))
185         {
186             JOptionPane.showMessageDialog(null, selectedVehicle.getBrand() + " " + selectedVehicle.getModel() + "
successfully attached to selected owner");
187             this.dispose();
188         }
189     }//End If
190
191 }
192
193 private void searchVehicleComboBoxItemStateChanged(java.awt.event.ItemEvent evt) {
194     // TODO add your handling code here:
195     //searchVehicle();
196 }

```

```

197
198 /**
199  * Search for a Vehicle
200  */
201 public void searchVehicle()
202 {
203     //Get the text from the searchfield
204     setValueOfSearchField((String) searchVehicleComboBox.getSelectedItem());
205     String searchCriteria = valueOfSearchField;
206     //Set the arrays with the searchResults
207     enterSearchDetails(searchCriteria);
208     //If nothing is found, display a label to the user
209     if (arrActFoundsBrandModel.length == 0) {
210         foundVehiclesList.setModel(new javax.swing.AbstractListModel() {
211
212             public int getSize() {
213                 return arrActFoundsBrandModel.length;
214             }
215
216             public Object getElementAt(int i) {
217                 return arrActFoundsBrandModel[i];
218             }
219         });
220     } //end if
221
222     //If there's results present them.
223     else {
224         //Clear the actual JList and make a new one
225
226         foundVehiclesList.setModel(new javax.swing.AbstractListModel() {
227
228             public int getSize() {
229                 return arrActFoundsBrandModel.length;
230             }
231
232             public Object getElementAt(int i) {
233                 return arrActFoundsBrandModel[i];
234             }
235         });
236     } //end else
237 }
238
239 private String setValueOfSearchField(String valueOfSearchField)
240 {
241     return this.valueOfSearchField = valueOfSearchField;
242 } //end setValueOfSearchField
243
244
245 /**
246  * This method ask the controller-layer to perform a search for a vehicle
247  * Search results are set from the controller-layer as multiply list of Vehicle-objects, String and ints
248  *
249  * The variables model and vType in every object in the actFoundsVehicleObjects list
250  * is put together as a string and inserted to an
251  * arraylist( actFoundsBrandModel ) which will be converted to an array
252  *
253  * The same applies to the list of vehilce id
254  * @param searchCriteria

```



```

255  * @return void
256  */
257  public void enterSearchDetails(String searchCriteria)
258  {
259      // This arrayLsit will hold the vehicles Brand and Model for presentation
260      actFoundsBrandModel = new ArrayList<String>();
261
262      // This arrayList will hold the vehicles ID
263      actFoundVehicle = new ArrayList<String>();
264
265      // This arrayList will hold the vehicle object
266      actFoundsVehicleObjects = oc.searchVehicle(searchCriteria);
267      int i = 0;
268      for (Vehicle v : actFoundsVehicleObjects)
269      {
270          // Brand, model and id in the respective lists
271          String licensPlate = v.getLicensPlate();
272          String chassisNumber = v.getChassisNumber();
273
274          // If both licensplate and chassis number is empty
275          if(licensPlate.equals("") && chassisNumber.equals(""))
276          {
277              actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel());
278          }
279          // If only chassisnumber is empty
280          else if(!licensPlate.equals("") && chassisNumber.equals(""))
281          {
282              actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
283          }
284          // If only licensplate is empty
285          else if(licensPlate.equals("") && !chassisNumber.equals(""))
286          {
287              actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getChassisNumber() + " )");
288          }
289          // Neither licensplate or chassisnumber is empty
290          else
291          {
292              actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
293          }
294          actFoundVehicle.add(v.getVehicleID());
295          i++;
296      } //end for each
297
298
299      arrActFoundsBrandModel = actFoundsBrandModel.toArray(new String[0]);
300      arrActFoundsVehicleID = actFoundVehicle.toArray(new String[0]);
301
302  }
303
304  /**
305   * @param args the command line arguments
306   */
307  public static void main(String args[]) {
308      java.awt.EventQueue.invokeLater(new Runnable() {
309          private String vehicleID;

```

```

310     public void run() {
311         new AttachVehicleToOrderGUI(this.vehicleID).setVisible(true);
312     }
313 });
314 }
315
316 // Variables declaration - do not modify
317 private javax.swing.JButton attachVehicleButton;
318 private javax.swing.JList foundVehiclesList;
319 private javax.swing.JScrollPane jScrollPane1;
320 private javax.swing.JLabel labelSearchVehicle;
321 private javax.swing.JButton searchVehicleButton;
322 private javax.swing.JComboBox searchVehicleComboBox;
323 // End of variables declaration
324
325 }

```

CompanyGUI

```

1 /*
2  * This class is to edit the companys information
3  *
4  */
5
6 /*
7  * CompanyGUI.java
8  *
9  * Created on 30-11-2010, 10:23:01
10 */
11
12 package tweakmc.view;
13
14 import java.awt.Color;
15 import javax.swing.JOptionPane;
16 import javax.swing.JPanel;
17 import tweakmc.control.ManageCompanyController;
18 import tweakmc.utility.CheckInput;
19
20 /**
21  *
22  * @author ADAMZ
23  */
24 public class CompanyGui extends javax.swing.JFrame {
25
26     private ManageCompanyController mcc = new ManageCompanyController();
27     private final String Ok_Company_saved = "Company saved";
28
29     private String errorMessage;
30
31     private final String CurNo_Empty = "curNo is Empty";
32     private final String Name_Empty = "name is Empty";
33     private final String Address_Invalid = "Address id invalid";
34     private final String BankInfo_Invalid = "BankInfo is invalid";
35     private final String Email_Invalid = "Email is invalid";
36     private final String HomePage_Invalid = "HomePage is Invalid";
37     private final String Phone_Invalid = "Phone is Invalid";
38

```

```

39
40
41
42
43  /** Creates new form CompanyGUI */
44  public CompanyGui() {
45      initComponents();
46      cPanel.setVisible(true);
47  } //end the constructor of Company Gui.
48
49
50
51
52
53
54  /** This method is called from within the constructor to
55   * initialize the form.
56   * WARNING: Do NOT modify this code. The content of this method is
57   * always regenerated by the Form Editor.
58   */
59  @SuppressWarnings("unchecked")
60  // <editor-fold defaultstate="collapsed" desc="Generated Code">
61  private void initComponents() {
62
63      jLabel2 = new javax.swing.JLabel();
64      cPanel = new javax.swing.JPanel();
65      jLabel1 = new javax.swing.JLabel();
66      jLabel3 = new javax.swing.JLabel();
67      jLabel4 = new javax.swing.JLabel();
68      jLabel5 = new javax.swing.JLabel();
69      jLabel6 = new javax.swing.JLabel();
70      jLabel7 = new javax.swing.JLabel();
71      jLabel8 = new javax.swing.JLabel();
72      jLabel9 = new javax.swing.JLabel();
73      curNoTextField = new javax.swing.JTextField();
74      nameTextField = new javax.swing.JTextField();
75      addressTextField = new javax.swing.JTextField();
76      phoneTextField = new javax.swing.JTextField();
77      emailTextField = new javax.swing.JTextField();
78      homePageTextField = new javax.swing.JTextField();
79      bankInfoTextField = new javax.swing.JTextField();
80      saveJButton = new javax.swing.JButton();
81      EditCompanyButton = new javax.swing.JButton();
82      resetFields = new javax.swing.JToggleButton();
83
84      jLabel2.setText("jLabel2");
85
86      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
87
88      jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18));
89      jLabel1.setText("Edit Company Info");
90
91      jLabel3.setText("CvrNo");
92
93      jLabel4.setText("Name");
94
95      jLabel5.setText("Address");
96

```

```
97     jLabel6.setText("Phone");
98
99     jLabel7.setText("Email");
100
101     jLabel8.setText("HomePage");
102
103     jLabel9.setText("BankInfo");
104
105     curNoTextField.addActionListener(new java.awt.event.ActionListener() {
106         public void actionPerformed(java.awt.event.ActionEvent evt) {
107             curNoTextFieldActionPerformed(evt);
108         }
109     });
110     curNoTextField.addFocusListener(new java.awt.event.FocusAdapter() {
111         public void focusLost(java.awt.event.FocusEvent evt) {
112             curNoTextFieldFocusLost(evt);
113         }
114     });
115
116     nameTextField.addActionListener(new java.awt.event.ActionListener() {
117         public void actionPerformed(java.awt.event.ActionEvent evt) {
118             nameTextFieldActionPerformed(evt);
119         }
120     });
121     nameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
122         public void focusLost(java.awt.event.FocusEvent evt) {
123             nameTextFieldFocusLost(evt);
124         }
125     });
126
127     addressTextField.addFocusListener(new java.awt.event.FocusAdapter() {
128         public void focusLost(java.awt.event.FocusEvent evt) {
129             addressTextFieldFocusLost(evt);
130         }
131     });
132
133     phoneTextField.addFocusListener(new java.awt.event.FocusAdapter() {
134         public void focusLost(java.awt.event.FocusEvent evt) {
135             phoneTextFieldFocusLost(evt);
136         }
137     });
138
139     emailTextField.addFocusListener(new java.awt.event.FocusAdapter() {
140         public void focusLost(java.awt.event.FocusEvent evt) {
141             emailTextFieldFocusLost(evt);
142         }
143     });
144
145     homePageTextField.addFocusListener(new java.awt.event.FocusAdapter() {
146         public void focusLost(java.awt.event.FocusEvent evt) {
147             homePageTextFieldFocusLost(evt);
148         }
149     });
150
151     bankInfoTextField.addFocusListener(new java.awt.event.FocusAdapter() {
152         public void focusLost(java.awt.event.FocusEvent evt) {
153             bankInfoTextFieldFocusLost(evt);
154         }
155     });
```

```

155     });
156
157     savejButton.setText("save");
158     savejButton.addActionListener(new java.awt.event.ActionListener() {
159         public void actionPerformed(java.awt.event.ActionEvent evt) {
160             savejButtonActionPerformed(evt);
161         }
162     });
163
164     EditCompanyButton.setText("Edit");
165     EditCompanyButton.addActionListener(new java.awt.event.ActionListener() {
166         public void actionPerformed(java.awt.event.ActionEvent evt) {
167             EditCompanyButtonActionPerformed(evt);
168         }
169     });
170
171     resetFields.setText("resetFields");
172     resetFields.addActionListener(new java.awt.event.ActionListener() {
173         public void actionPerformed(java.awt.event.ActionEvent evt) {
174             resetFieldsActionPerformed(evt);
175         }
176     });
177     resetFields.addFocusListener(new java.awt.event.FocusAdapter() {
178         public void focusLost(java.awt.event.FocusEvent evt) {
179             resetFieldsFocusLost(evt);
180         }
181     });
182
183     javax.swing.GroupLayout cPanelLayout = new javax.swing.GroupLayout(cPanel);
184     cPanel.setLayout(cPanelLayout);
185     cPanelLayout.setHorizontalGroup(
186         cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
187             .addGroup(cPanelLayout.createSequentialGroup()
188                 .addGap(30, 30, 30)
189                 .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190                     .addGroup(cPanelLayout.createSequentialGroup()
191                         .addComponent(jLabel3)
192                         .addComponent(jLabel4)
193                         .addComponent(jLabel5)
194                         .addComponent(jLabel6)
195                         .addComponent(jLabel7)
196                         .addComponent(jLabel8)
197                         .addComponent(jLabel9))
198                     .addGroup(cPanelLayout.createSequentialGroup()
199                         .addGap(40, 40, 40)
200                         .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
201                             false)
202                             .addComponent(bankInfoTextField)
203                             .addComponent(homePageTextField)
204                             .addComponent(emailTextField)
205                             .addComponent(phoneTextField)
206                             .addComponent(addressTextField)
207                             .addComponent(nameTextField)
208                             .addComponent(curNoTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 234,
209                                 Short.MAX_VALUE)))
210                     .addComponent(jLabel1))
211             .addGap(53, 53, 53)
212     );

```

```

211         .addGap(40, 40, 40)
212         .addComponent(savejButton)
213         .addGap(71, 71, 71)
214         .addComponent(resetFields)
215         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 80, Short.MAX_VALUE)
216         .addComponent(EditCompanyButton)
217         .addGap(26, 26, 26))
218     );
219     cPanelLayout.setVerticalGroup(
220         cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
221         .addGroup(cPanelLayout.createSequentialGroup()
222             .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
223             .addGap(27, 27, 27)
224             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
225                 .addComponent(jLabel3)
226                 .addComponent(curNoTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
227             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
228             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
229                 .addComponent(jLabel4)
230                 .addComponent(nameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
231             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
232             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
233                 .addComponent(jLabel5)
234                 .addComponent(addressTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
235             .addGap(18, 18, 18)
236             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
237                 .addComponent(jLabel6)
238                 .addComponent(phoneTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
239             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
240             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
241                 .addComponent(jLabel7)
242                 .addComponent(emailTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
243             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
244             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
245                 .addComponent(jLabel8)
246                 .addComponent(homePageTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
247             .addGap(18, 18, 18)
248             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
249                 .addComponent(jLabel9)
250                 .addComponent(bankInfoTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
251             .addGap(52, 52, 52)
252             .addGroup(cPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
253                 .addComponent(savejButton)
254                 .addComponent(EditCompanyButton)
255                 .addComponent(resetFields))
256             .addContainerGap())
257     );
258
259     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
260     getContentPane().setLayout(layout);

```

```

261     layout.setHorizontalGroup(
262         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
263         .addComponent(cPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
264     );
265     layout.setVerticalGroup(
266         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
267         .addComponent(cPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
268     );
269
270     pack();
271 }// </editor-fold>
272
273 private void saveJButtonActionPerformed(java.awt.event.ActionEvent evt) {
274     // TODO add your handling code here: it saves the company information
275     saveCompanyInfo();
276
277
278 }
279
280 private void EditCompanyButtonActionPerformed(java.awt.event.ActionEvent evt) {
281     // TODO add your handling code here:
282     EditCompanyButton.setEnabled(true);
283
284     curNoTextField.setEditable(true);
285     curNoTextField.setBackground(Color.white);
286
287     nameTextField.setEditable(true);
288     nameTextField.setBackground(Color.white);
289
290     addressTextField.setEditable(true);
291     addressTextField.setBackground(Color.white);
292
293     emailTextField.setEditable(true);
294     emailTextField.setBackground(Color.white);
295
296     homePageTextField.setEditable(true);
297     homePageTextField.setBackground(Color.white);
298
299     phoneTextField.setEditable(true);
300     phoneTextField.setBackground(Color.white);
301
302     bankInfoTextField.setEditable(true);
303     bankInfoTextField.setBackground(Color.white);
304
305
306
307
308 }
309
310 private void nameTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
311     // TODO add your handling code here:
312 }
313
314 private void curNoTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
315     // TODO add your handling code here:
316

```

```

317 }
318
319 private void curNoTextFieldFocusLost(java.awt.event.FocusEvent evt) {
320     // TODO add your handling code here:
321
322     if(!curNoTextField.getText().equals("") && !CheckInput.checkCurNo(curNoTextField.getText()) ||
    CheckInput.isInt(curNoTextField.getText()))
323     {
324         errorMessage = CurNo_Empty;
325         curNoTextField.setBackground(Color.PINK);
326     }
327     else
328         curNoTextField.setBackground(Color.white);
329 }
330 }
331
332 private void nameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
333     // TODO add your handling code here:
334
335     if(!nameTextField.getText().equals("") && !CheckInput.checkName(nameTextField.getText()) ||
    CheckInput.isInt(nameTextField.getText()))
336     {
337         errorMessage = Name_Empty;
338         nameTextField.setBackground(Color.PINK);
339     }
340     else
341         nameTextField.setBackground(Color.white);
342 }
343
344 private void addressTextFieldFocusLost(java.awt.event.FocusEvent evt) {
345     // TODO add your handling code here:
346     if(!addressTextField.getText().equals("") && !CheckInput.checkAddress(addressTextField.getText()) ||
    CheckInput.isInt(addressTextField.getText()))
347     {
348         errorMessage = Address_Invalid;
349         addressTextField.setBackground(Color.PINK);
350     }
351     else addressTextField.setBackground(Color.white);
352 }
353
354 private void bankInfoTextFieldFocusLost(java.awt.event.FocusEvent evt) {
355     // TODO add your handling code here:
356     if(bankInfoTextField.getText().equals(""))
357     {
358         errorMessage = BankInfo_Invalid;
359
360         bankInfoTextField.setBackground(Color.PINK);
361     }
362     else if(bankInfoTextField.getText().equals("") &&
    !CheckInput.CheckBankInfo(bankInfoTextField.getText()))
363     {
364         errorMessage = BankInfo_Invalid;
365
366         bankInfoTextField.setBackground(Color.PINK);
367     }
368     else
369     {
370         bankInfoTextField.setBackground(Color.white);

```



```

371     }
372 }
373
374 private void phoneTextFieldFocusLost(java.awt.event.FocusEvent evt) {
375     // TODO add your handling code here:
376
377     if(!phoneTextField.getText().equals("") && !CheckInput.checkPhone(phoneTextField.getText()))
378     {
379         errorMessage = Phone_Invalid;
380         phoneTextField.setBackground(Color.PINK);
381     }
382     else
383     {
384         phoneTextField.setBackground(Color.white);
385     }
386 }
387
388
389 private void emailTextFieldFocusLost(java.awt.event.FocusEvent evt) {
390     // TODO add your handling code here:
391     if(!emailTextField.getText().equals("") && !CheckInput.checkEmail(emailTextField.getText()))
392     {
393         errorMessage = Email_Invalid;
394         emailTextField.setBackground(Color.PINK);
395     }
396     else
397     {
398         emailTextField.setBackground(Color.white);
399     }
400 }
401
402 private void homePageTextFieldFocusLost(java.awt.event.FocusEvent evt) {
403     // TODO add your handling code here:
404     if(!homePageTextField.getText().equals("") &&
405     !CheckInput.checkHomePage(homePageTextField.getText()))
406     {
407         errorMessage = Email_Invalid;
408         homePageTextField.setBackground(Color.PINK);
409     }
410     else
411     {
412         homePageTextField.setBackground(Color.white);
413     }
414 }
415
416 private void resetFieldsActionPerformed(java.awt.event.ActionEvent evt) {
417     // TODO add your handling code here:
418 }
419
420 private void resetFieldsFocusLost(java.awt.event.FocusEvent evt) {
421     // TODO add your handling code here:
422     //rests fields
423     restFields();
424 }
425
426
427 private void saveCompanyInfo()

```

```

428
429     //checks if all text fields are filled up
430
431     {
432         if(curNoTextField.getText().equals(""))
433         {
434             errorMessage = errorMessage + " " + CurNo_Empty;
435             curNoTextField.setBackground(Color.PINK);
436         }
437         else if(!curNoTextField.getText().equals("") && !CheckInput.checkCurNo(curNoTextField.getText()) ||
CheckInput.isInt(curNoTextField.getText()))
438         {
439             errorMessage = errorMessage + " " + CurNo_Empty;
440             curNoTextField.setBackground(Color.PINK);
441         }
442         if(nameTextField.getText().equals(""))
443         {
444             errorMessage = errorMessage + " " + Name_Empty;
445             nameTextField.setBackground(Color.PINK);
446         }
447         else if(!nameTextField.getText().equals("") && !CheckInput.checkName(nameTextField.getText()) ||
CheckInput.isInt(nameTextField.getText()))
448         {
449             errorMessage = errorMessage + " " + Name_Empty;
450             nameTextField.setBackground(Color.PINK);
451         }
452
453         if(addressTextField.getText().equals(""))
454         {
455             errorMessage = errorMessage + " " + Address_Invalid;
456             addressTextField.setBackground(Color.PINK);
457         }
458         else if(!addressTextField.getText().equals("") && !CheckInput.checkAddress(addressTextField.getText()) ||
CheckInput.isInt(addressTextField.getText()))
459         {
460             errorMessage = errorMessage + " " + Address_Invalid;
461             curNoTextField.setBackground(Color.PINK);
462         }
463         else if(!phoneTextField.getText().equals("") && !CheckInput.checkPhone(phoneTextField.getText()) ||
CheckInput.isInt(phoneTextField.getText()))
464         {
465             errorMessage = errorMessage + " " + Phone_Invalid;
466             phoneTextField.setBackground(Color.PINK);
467         }
468         if(bankInfoTextField.getText().equals(""))
469         {
470             errorMessage = errorMessage + " " + BankInfo_Invalid;
471             bankInfoTextField.setBackground(Color.PINK);
472         }
473         else if(!bankInfoTextField.getText().equals("") &&
!CheckInput.CheckBankInfo(bankInfoTextField.getText()) || CheckInput.isInt(bankInfoTextField.getText()))
474         {
475             errorMessage = errorMessage + " " + BankInfo_Invalid;
476             bankInfoTextField.setBackground(Color.PINK);
477         }
478         if(emailTextField.getText().equals(""))
479         {
480             errorMessage = errorMessage + " " + Email_Invalid;

```

```

481     emailTextField.setBackground(Color.PINK);
482 }
483 else if(!emailTextField.getText().equals("") && !CheckInput.checkEmail(emailTextField.getText()) ||
CheckInput.isInt(emailTextField.getText()))
484 {
485     errorMessage = errorMessage + " " + Email_Invalid;
486     emailTextField.setBackground(Color.PINK);
487 }
488 if(homePageTextField.getText().equals(""))
489 {
490     errorMessage = errorMessage + " " + HomePage_Invalid;
491     homePageTextField.setBackground(Color.PINK);
492 }
493 else if(!homePageTextField.getText().equals("")) &&
!CheckInput.checkHomePage(homePageTextField.getText()) || CheckInput.isInt(homePageTextField.getText())
494 {
495     errorMessage = errorMessage + " " + HomePage_Invalid;
496     homePageTextField.setBackground(Color.PINK);
497 }
498
499 if(checkCopanyFields())
500 {
501     mcc.editCompany(curNoTextField.getText(), nameTextField.getText(), addressTextField.getText(),
502         bankInfoTextField.getText(), phoneTextField.getText(),
503         emailTextField.getText(),homePageTextField.getText());
504     JOptionPane.showMessageDialog(null, "company Successfully updated !", "Successfully updated",
JOptionPane.INFORMATION_MESSAGE);
505
506     restFields();
507 } //end if
508
509 }
510
511
512 /**
513  * Resets all fields of the Company
514  */
515
516 private void restFields()
517 {
518     curNoTextField.setBackground(Color.white);
519     curNoTextField.setText("");
520
521     nameTextField.setBackground(Color.white);
522     nameTextField.setText("");
523
524     addressTextField.setBackground(Color.white);
525     addressTextField.setText("");
526
527     phoneTextField.setBackground(Color.white);
528     phoneTextField.setText("");
529
530     emailTextField.setBackground(Color.white);
531     emailTextField.setText("");
532
533     bankInfoTextField.setBackground(Color.white);
534     bankInfoTextField.setText("");
535

```

```

536     homePageTextField.setBackground(Color.white);
537     homePageTextField.setText("");
538
539     /**
540      * rest all fields
541      */
542     curNoTextField.setText("");
543     nameTextField.setText("");
544     addressTextField.setText("");
545     emailTextField.setText("");
546     bankInfoTextField.setText("");
547     homePageTextField.setText("");
548
549     }// End of methord resetFields
550
551
552     /**
553      * Methord that checks Company firlds
554      * @return
555      */
556
557     public boolean checkCopanyFields()
558     {
559         errorMessage = "";
560         if(curNoTextField.getText().equals(""))
561         {
562             curNoTextField.setBackground(Color.PINK);
563             return false;
564         }
565         else if(!curNoTextField.getText().equals("")) && !CheckInput.checkCurNo(curNoTextField.getText()))
566         {
567             curNoTextField.setBackground(Color.PINK);
568             return false;
569         }
570
571         if(nameTextField.getText().equals(""))
572         {
573             errorMessage = errorMessage + " " + Name_Empty;
574             nameTextField.setBackground(Color.PINK);
575             return false;
576         }
577
578         else if(!nameTextField.getText().equals("")) && !CheckInput.checkName(nameTextField.getText()))
579         {
580             nameTextField.setBackground(Color.PINK);
581             return false;
582         }
583
584         if(!addressTextField.getText().equals("")) && !CheckInput.checkAddress(addressTextField.getText()))
585         {
586             errorMessage = Address_Invalid;
587             addressTextField.setBackground(Color.PINK);
588             return false;
589         }
590
591         if(!emailTextField.getText().equals("")) && !CheckInput.checkEmail(emailTextField.getText()))
592         {
593             errorMessage = Email_Invalid;

```

```

594     emailTextField.setBackground(Color.PINK);
595     return false;
596 }
597
598 if(!bankInfoTextField.getText().equals("") && !CheckInput.CheckBankInfo(bankInfoTextField.getText()))
599 {
600     errorMessage = BankInfo_Invalid;
601     bankInfoTextField.setBackground(Color.PINK);
602     return false;
603 }
604 if(!bankInfoTextField.getText().equals("") && !CheckInput.CheckBankInfo(bankInfoTextField.getText()))
605 {
606     errorMessage = BankInfo_Invalid;
607     bankInfoTextField.setBackground(Color.PINK);
608     return false;
609 }
610
611 if(!phoneTextField.getText().equals("") && !CheckInput.checkPhone(phoneTextField.getText()))
612 {
613     errorMessage = Phone_Invalid;
614     phoneTextField.setBackground(Color.PINK);
615     return false;
616 }
617
618
619
620 if(!homePageTextField.getText().equals("") &&
!CheckInput.checkHomePage(homePageTextField.getText()))
621 {
622     errorMessage = HomePage_Invalid;
623     homePageTextField.setBackground(Color.PINK);
624     return false;
625 }
626 return true;
627
628 }// End of reseltFields methord in the CompanyGui Class
629 //and it returns True
630
631
632
633
634
635
636
637 /**
638 * @param args the command line arguments
639 */
640 public static void main(String args[]) {
641     java.awt.EventQueue.invokeLater(new Runnable() {
642         public void run() {
643             new CompanyGui().setVisible(true);
644         }
645     });
646 }
647
648 }
649
650 // Variables declaration - do not modify

```

```

651 private javax.swing.JButton EditCompanyButton;
652 private javax.swing.JTextField addressTextField;
653 private javax.swing.JTextField bankInfoTextField;
654 private javax.swing.JPanel cPanel;
655 private javax.swing.JTextField curNoTextField;
656 private javax.swing.JTextField emailTextField;
657 private javax.swing.JTextField homePageTextField;
658 private javax.swing.JLabel jLabel1;
659 private javax.swing.JLabel jLabel2;
660 private javax.swing.JLabel jLabel3;
661 private javax.swing.JLabel jLabel4;
662 private javax.swing.JLabel jLabel5;
663 private javax.swing.JLabel jLabel6;
664 private javax.swing.JLabel jLabel7;
665 private javax.swing.JLabel jLabel8;
666 private javax.swing.JLabel jLabel9;
667 private javax.swing.JTextField nameTextField;
668 private javax.swing.JTextField phoneTextField;
669 private javax.swing.JToggleButton resetFields;
670 private javax.swing.JButton saveJButton;
671 // End of variables declaration
672
673 }

```

CreateSpreadsheetView

```

1 /*
2  * This class is to create a new userdefined spreadsheet, with rows and columns
3  * It takes a Point parameter to know where on the screen it should be presented
4  * It also takes a vehicle ID in parameters to know witch vehicle it should belong to
5  * At last it takes workstation name to know witch workstation it belongs to
6  *
7  */
8
9 /*
10 * CreateSpreadsheetView.java
11 *
12 * Created on 06-12-2010, 12:00:05
13 */
14
15 package tweakmc.view;
16
17 import java.awt.Point;
18 import java.util.Collections;
19 import java.util.Vector;
20 import javax.swing.DefaultComboBoxModel;
21 import javax.swing.JOptionPane;
22 import javax.swing.JTable;
23 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
24 import tweakmc.control.ManageSpreadsheetController;
25
26 /**
27  *
28  * @author clausPallisgaardBeck
29  */
30 public class CreateSpreadsheetView extends javax.swing.JFrame
31 {
32     private final int NOOF_LENGTH = 1001;
33     private Vector noOf;

```

```

34
35 private String vehicleID, workStationName;
36 private Point location;
37
38 /** Creates new form CreateSpreadsheetView */
39 public CreateSpreadsheetView()
40 {
41     buildNoOf();
42     setUndecorated(true);
43     initComponents();
44     setLocation(new Point(10,10));
45     setVisible(true);
46
47 } //end construtor
48
49 /**
50  * Set the location where the frame shoul open
51  * @param p point on screen where mouse is
52  */
53 public CreateSpreadsheetView(Point p, String vehicleID, String workStationName)
54 {
55     this.location = p;
56     this.vehicleID = vehicleID;
57     this.workStationName = workStationName;
58     buildNoOf();
59     setUndecorated(true);
60     initComponents();
61     setLocation(p);
62     setVisible(true);
63 } //end CreateSpreadsheetView(Point p)
64
65 /** This method is called from within the constructor to
66  * initialize the form.
67  * WARNING: Do NOT modify this code. The content of this method is
68  * always regenerated by the Form Editor.
69  */
70 @SuppressWarnings("unchecked")
71 // <editor-fold defaultstate="collapsed" desc="Generated Code">
72 private void initComponents() {
73
74     buttonGroup1 = new javax.swing.ButtonGroup();
75     northJPanel = new javax.swing.JPanel();
76     headerNorthLabel = new javax.swing.JLabel();
77     jSeparator1 = new javax.swing.JSeparator();
78     southJPanel = new javax.swing.JPanel();
79     createButton = new javax.swing.JButton();
80     cancelButton = new javax.swing.JButton();
81     centerJPanel = new javax.swing.JPanel();
82     rowHeaderLabel = new javax.swing.JLabel();
83     rowJCombo = new javax.swing.JComboBox();
84     AutoCompleteDecorator.decorate(rowJCombo);
85     rowJCombo.setModel(new DefaultComboBoxModel(noOf));
86     columnHeaderLabel = new javax.swing.JLabel();
87     columnJCombo = new javax.swing.JComboBox();
88     AutoCompleteDecorator.decorate(columnJCombo);
89     columnJCombo.setModel(new DefaultComboBoxModel(noOf));
90     jSeparator2 = new javax.swing.JSeparator();
91     columnNameHeaderLabel = new javax.swing.JLabel();

```

```

92     yesJRadioButton = new javax.swing.JRadioButton();
93     noJRadioButton = new javax.swing.JRadioButton();
94     infoColumnNameLabel = new javax.swing.JLabel();
95     jScrollPane1 = new javax.swing.JScrollPane();
96     columnNameTextArea = new javax.swing.JTextArea();
97     columnNameTextArea.setEnabled(false);
98
99     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
100
101     headerNorthLabel.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
102     headerNorthLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
103     headerNorthLabel.setText("Create user defined spreadsheet");
104
105     javax.swing.GroupLayout northJPanelLayout = new javax.swing.GroupLayout(northJPanel);
106     northJPanel.setLayout(northJPanelLayout);
107     northJPanelLayout.setHorizontalGroup(
108         northJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
109             .addGroup(northJPanelLayout.createSequentialGroup()
110                 .addContainerGap()
111                 .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 380, Short.MAX_VALUE)
112                 .addContainerGap()
113                 .addGroup(northJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
114                     .addGroup(northJPanelLayout.createSequentialGroup()
115                         .addContainerGap()
116                         .addComponent(headerNorthLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 380,
Short.MAX_VALUE)
117                     .addContainerGap()))
118             );
119     northJPanelLayout.setVerticalGroup(
120         northJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
121             .addGroup(northJPanelLayout.createSequentialGroup()
122                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, northJPanelLayout.createSequentialGroup()
123                     .addContainerGap(27, Short.MAX_VALUE)
124                     .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
125                 .addGroup(northJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
126                     .addGroup(northJPanelLayout.createSequentialGroup()
127                         .addContainerGap()
128                         .addComponent(headerNorthLabel)
129                         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
130             );
131     getContentPane().add(northJPanel, java.awt.BorderLayout.PAGE_START);
132
133     southJPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
134
135     createButton.setText("Create");
136     createButton.addActionListener(new java.awt.event.ActionListener() {
137         public void actionPerformed(java.awt.event.ActionEvent evt) {
138             createButtonActionPerformed(evt);
139         }
140     });
141     southJPanel.add(createButton);
142
143     cancelButton.setText("Cancel");
144     southJPanel.add(cancelButton);
145
146     getContentPane().add(southJPanel, java.awt.BorderLayout.PAGE_END);

```



```

147
148
centerJPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
149
150     rowHeaderLabel.setText("No. of row(s): ");
151
152     columnHeaderLabel.setText("No. of column(s): ");
153
154     columnNameHeaderLabel.setText("Should the column(s) have name(s): ");
155
156     buttonGroup1.add(yesJRadioButton);
157     yesJRadioButton.setText("Yes");
158     yesJRadioButton.addActionListener(new java.awt.event.ActionListener() {
159         public void actionPerformed(java.awt.event.ActionEvent evt) {
160             yesJRadioButtonActionPerformed(evt);
161         }
162     });
163
164     buttonGroup1.add(noJRadioButton);
165     noJRadioButton.setSelected(true);
166     noJRadioButton.setText("No");
167
168     infoColumnNameLabel.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
169     infoColumnNameLabel.setText("One name pr. line");
170
171     columnNameTextArea.setColumns(20);
172     columnNameTextArea.setRows(5);
173     jScrollPane1.setViewportView(columnNameTextArea);
174
175     javax.swing.GroupLayout centerJPanelLayout = new javax.swing.GroupLayout(centerJPanel);
176     centerJPanel.setLayout(centerJPanelLayout);
177     centerJPanelLayout.setHorizontalGroup(
178         centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
179             .addGroup(centerJPanelLayout.createSequentialGroup()
180                 .addGap(18, 18, 18)
181                 .addGroup(centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
182                     .addComponent(jSeparator2, javax.swing.GroupLayout.DEFAULT_SIZE, 376, Short.MAX_VALUE)
183                     .addGroup(centerJPanelLayout.createSequentialGroup()
184                         .addComponent(rowHeaderLabel)
185                         .addGap(18, 18, 18)
186                         .addComponent(rowJCombo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
187                     .addGroup(centerJPanelLayout.createSequentialGroup()
188                         .addComponent(columnHeaderLabel)
189                         .addGap(18, 18, 18)
190                         .addComponent(columnJCombo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
191                     .addGroup(centerJPanelLayout.createSequentialGroup()
192                         .addComponent(columnNameHeaderLabel)
193                         .addGap(18, 18, 18)
194                         .addComponent(yesJRadioButton)
195                         .addGap(18, 18, 18)
196                         .addComponent(noJRadioButton)
197                         .addGap(18, 18, 18)
198                         .addComponent(infoColumnNameLabel)
199                         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
200                     .addGap(18, 18, 18)

```

```

201     centerJPanelLayout.setVerticalGroup(
202         centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
203         .addGroup(centerJPanelLayout.createSequentialGroup())
204         .addContainerGap()
205         .addGroup(centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
206             .addComponent(rowHeaderLabel)
207             .addComponent(rowJCombo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
208         .addGap(18, 18, 18)
209         .addGroup(centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
210             .addComponent(columnHeaderLabel)
211             .addComponent(columnJCombo, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
212         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
213         .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
214         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
215         .addGroup(centerJPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
216             .addComponent(columnNameHeaderLabel)
217             .addComponent(yesJRadioButton)
218             .addComponent(noJRadioButton))
219         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
220         .addComponent(infoColumnNameLabel)
221         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
222         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
223         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
224     );
225
226     getContentPane().add(centerJPanel, java.awt.BorderLayout.CENTER);
227
228     pack();
229 } // </editor-fold>
230
231 private void yesJRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
232     columnNameTextArea.setEnabled(true);
233 }
234
235 private void createButtonActionPerformed(java.awt.event.ActionEvent evt) {
236     defineSpreadsheet();
237 }
238
239 /**
240  * Create spreadsheet with given parameters
241  */
242 private void defineSpreadsheet()
243 {
244
245
246     if (columnNameTextArea.isEnabled() && ((Integer) columnJCombo.getSelectedItem() !=
getColumnNames().length ))
247     {
248         int diff = ((Integer) columnJCombo.getSelectedItem() - getColumnNames().length );
249
250         if(diff > 0)
251         {
252             JOptionPane.showMessageDialog(columnNameTextArea, "There is missing\n" + diff
+ " name(s)", "Missing name!", JOptionPane.WARNING_MESSAGE);
253

```

```

254     }//end if
255
256     if(diff < 0)
257     {
258         int diff1 = (getColumnNames().length - (Integer) columnJCombo.getSelectedItem());
259         JOptionPane.showMessageDialog(columnNameTextArea, "There is\n" + diff1
260             + " name(s) too much", "Too many names!", JOptionPane.WARNING_MESSAGE);
261     }//end if
262
263     }//end if
264
265     if ((Integer)columnJCombo.getSelectedItem() == getColumnNames().length )
266     {
267         JTable actTable = (new
ManageSpreadsheetController().createSpreadsheet((Integer)rowJCombo.getSelectedItem(), (Integer)
columnJCombo.getSelectedItem(), getColumnNames()));
268         new ManageSpreadsheetView(actTable, vehicleID, workStationName);
269         dispose();
270     }//end if
271 }//end method defineSpreadSheet
272
273 /**
274  * Read the columnnames from textarea and put them in an Array
275  * @return Array this names for columns
276  */
277 private String[] getColumnNames()
278 {
279     String[] names = columnNameTextArea.getText().trim().split("\n");
280     return names;
281 }
282
283 /**
284  * Build numbers for column and rows combobox
285  */
286 private void buildNoOf()
287 {
288     noOf = new Vector();
289
290     for (int i = 1; i < NOOF_LENGTH; i++)
291     {
292         noOf.add(i);
293     }//end for
294     Collections.sort(noOf);
295 }//end method buildNoOf
296
297 /**
298  * @param args the command line arguments
299  */
300 public static void main(String args[]) {
301     java.awt.EventQueue.invokeLater(new Runnable() {
302         public void run() {
303             new CreateSpreadsheetView(new Point(10,10), "V15", "EngineStation");
304         }
305     });
306 }
307
308 // Variables declaration - do not modify
309 private javax.swing.ButtonGroup buttonGroup1;

```

```

310 private javax.swing.JButton cancelButton;
311 private javax.swing.JPanel centerJPanel;
312 private javax.swing.JLabel columnHeaderLabel;
313 private javax.swing.JComboBox columnJCombo;
314 private javax.swing.JLabel columnNameHeaderLabel;
315 private javax.swing.JTextArea columnNameTextArea;
316 private javax.swing.JButton createButton;
317 private javax.swing.JLabel headerNorthLabel;
318 private javax.swing.JLabel infoColumnNameLabel;
319 private javax.swing.JScrollPane jScrollPane1;
320 private javax.swing.JSeparator jSeparator1;
321 private javax.swing.JSeparator jSeparator2;
322 private javax.swing.JRadioButton noJRadioButton;
323 private javax.swing.JPanel northJPanel;
324 private javax.swing.JLabel rowHeaderLabel;
325 private javax.swing.JComboBox rowJCombo;
326 private javax.swing.JPanel southJPanel;
327 private javax.swing.JRadioButton yesJRadioButton;
328 // End of variables declaration
329
330 }

```

GUIDesign (unused)

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5
6 /*
7  * GUIDesign.java
8  *
9  * Created on 18-10-2010, 09:41:48
10 */
11
12 package tweakmc.view;
13
14 import java.awt.Graphics2D;
15 import java.awt.Image;
16 import java.awt.Toolkit;
17 import java.awt.image.BufferedImage;
18 import java.net.URL;
19 import javax.swing.ImageIcon;
20 import javax.swing.JFrame;
21 import javax.swing.JOptionPane;
22 import javax.swing.JPanel;
23 import javax.swing.UIManager;
24 import javax.swing.UIManager.LookAndFeelInfo;
25 import tweakmc.utility.StartupTest;
26 import tweakmc.utility.FileWriterException;
27
28 /**
29  *
30  * @author tohan79
31  */
32 public class GUIDesign extends javax.swing.JFrame
33 {

```

```

34 // Instance variables
35 private Image kjLogo;
36 private Toolkit toolK = Toolkit.getDefaultToolkit();
37 //public Login loginObj;
38 private OwnerGUI ownerGUI;
39
40 /** Creates new form GUIDesign */
41 public GUIDesign()
42 {
43     /* The following two lines are testing of the database-connections and
44      * will provide detailed error-messages if the connection fails.
45      * Should be removed before distribution.
46      */
47     StartUpTest stut = new StartUpTest();
48     if(!stut.startUpTests()) { JOptionPane.showMessageDialog(null, "Database error!"); }
49     try
50     {
51         for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
52         {
53             if ("Nimbus".equals(info.getName()))
54             {
55                 UIManager.setLookAndFeel(info.getClassName());
56                 break;
57             } //end if
58         } //end for
59     } // end try
60
61     catch (Exception e)
62     {
63         FileWriterException.writeLogFile("Look and feel settings: " + e.getMessage());
64     } //end catch
65
66     finally
67     {
68         OwnerGUI og = new OwnerGUI();
69         initComponents();
70         // createLoginObject();
71
72         //vehiclePane.setEnabledAt(1, false);
73         vehiclePane.setEnabledAt(2, false);
74
75
76     } //end finally
77 } //end GUIDesign constructor
78
79 /**
80  * Enables the panels of the GUI, after successfully login
81  */
82 protected void enablePanels()
83 {
84     JFrame frame = new JFrame("Enable tabbed panes");
85     OwnerGUI og = new OwnerGUI();
86     vehiclePane.insertTab("Owner test", null, og.getPanel(), null, 1);
87     frame.setContentPane(vehiclePane);
88     frame.pack();
89     frame.setVisible(true);
90
91     // createVehicleObject();

```

```

92 //     createOwnerObject();
93
94     vehiclePane.setEnabledAt(2, true);
95     vehiclePane.setVisible(true);
96     vehiclePane.revalidate();
97 //         initComponents();
98 //         createLoginObject();
99 //         createVehicleObject();
100 //         vTabPanel.setEnabled(false);
101 //         createOwnerObject();
102 //         ownerTabPanel.setEnabled(false);
103
104
105 }
106 /**
107  * Creates a new object of VehicleGUI and calls the insertVehiclePanel method
108  */
109 private void createVehicleObject()
110 {
111     VehicleGUI veh = new VehicleGUI();
112     insertVehiclePanel(veh.getPanel());
113 } //end insertVehiclePanel
114
115 /**
116  * Creates a new object of OwnerGUI and calls the insertOwnerPanel method
117  */
118 private void createOwnerObject()
119 {
120     OwnerGUI own = new OwnerGUI();
121     insertOwnerPanel(own.getPanel());
122 } //end insertLoginPanel
123
124 /**
125  * Creates a new object of LoginGUI and calls the insertLoginPanel method
126  */
127 // private void createLoginObject()
128 // {
129 //     Login log = new Login();
130 //     insertLoginPanel(log.getPanel());
131 // } //end insertLoginPanel
132
133 /**
134  * Inserts the VehicleObject into GUIDesign
135  * @param panel
136  */
137 private void insertVehiclePanel(JPanel panel)
138 {
139     javax.swing.GroupLayout vTabPanelLayout = new javax.swing.GroupLayout(vTabPanel);
140     vTabPanel.setLayout(vTabPanelLayout);
141     vTabPanelLayout.setHorizontalGroup(
142         vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
143             .addComponent(panel, javax.swing.GroupLayout.DEFAULT_SIZE, 800, Short.MAX_VALUE)
144     );
145     vTabPanelLayout.setVerticalGroup(
146         vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
147             .addComponent(panel, javax.swing.GroupLayout.Alignment.TRAILING,
148                 javax.swing.GroupLayout.DEFAULT_SIZE, 600, Short.MAX_VALUE)
149     );

```

```

149 }//end insertVehiclePanel
150
151 /**
152  * Inserts the VehicleObject into GUIDesign
153  * @param panel
154  */
155 private void insertOwnerPanel(JPanel panel)
156 {
157     javax.swing.GroupLayout ownerTabPanelLayout = new javax.swing.GroupLayout(ownerTabPanel);
158     ownerTabPanel.setLayout(ownerTabPanelLayout);
159     ownerTabPanelLayout.setHorizontalGroup(
160         ownerTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
161             .addComponent(panel, javax.swing.GroupLayout.DEFAULT_SIZE, 800, Short.MAX_VALUE)
162     );
163     ownerTabPanelLayout.setVerticalGroup(
164         ownerTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
165             .addComponent(panel, javax.swing.GroupLayout.Alignment.TRAILING,
166                 javax.swing.GroupLayout.DEFAULT_SIZE, 600, Short.MAX_VALUE)
167     );
168 }//end insertOwnerPanel
169
170
171 /**
172  * Inserts the LoginObject into GUIDesign
173  * @param panel
174  */
175 private void insertLoginPanel(JPanel panel)
176 {
177     javax.swing.GroupLayout loginTestPanelLayout = new javax.swing.GroupLayout(loginTestPanel);
178     loginTestPanel.setLayout(loginTestPanelLayout);
179     loginTestPanelLayout.setHorizontalGroup(
180         loginTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
181             .addComponent(panel, javax.swing.GroupLayout.DEFAULT_SIZE, 800, Short.MAX_VALUE)
182     );
183     loginTestPanelLayout.setVerticalGroup(
184         loginTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
185             .addComponent(panel, javax.swing.GroupLayout.Alignment.TRAILING,
186                 javax.swing.GroupLayout.DEFAULT_SIZE, 600, Short.MAX_VALUE)
187     );
188 }//end insertLoginPanel
189
190 //<<<<<<<<<< .mine
191 // public void setLoggedInStatusTxt()
192 // {
193 //     Login loginObj = new Login();
194 //     loggedInStatus.setText(loginObj.loginTextField.getText());
195 // }
196 // =====
197 // public void setLoggedInStatusTxt(String loginText)
198 // {
199 //     jLabel2.setText(loginText);
200 //     statusPanel.validate();
201 //     statusPanel.repaint();
202 //     System.out.println(loginText);
203 // }
204 //>>>>>>>>> .r118

```

```

205
206
207
208 /**
209  *
210  * @return
211  */
212 @SuppressWarnings("unchecked")
213 // <editor-fold defaultstate="collapsed" desc="Generated Code">
214 private void initComponents() {
215
216     vehiclePane = new javax.swing.JTabbedPane();
217     loginTabPanel = new javax.swing.JPanel();
218     loginTestPanel = new javax.swing.JPanel();
219     vTabPanel = new javax.swing.JPanel();
220     scrollbar1 = new java.awt.Scrollbar();
221     ownerTabPanel = new javax.swing.JPanel();
222     jLabel1 = new javax.swing.JLabel();
223     statusPanel = new javax.swing.JPanel();
224     loggedInStatus = new javax.swing.JLabel();
225     jLabel2 = new javax.swing.JLabel();
226     jMenuBar1 = new javax.swing.JMenuBar();
227     jMenu1 = new javax.swing.JMenu();
228     jMenuItem1 = new javax.swing.JMenuItem();
229     jMenu3 = new javax.swing.JMenu();
230     jMenuItem3 = new javax.swing.JMenuItem();
231     jMenuItem4 = new javax.swing.JMenuItem();
232     helpMenuItem = new javax.swing.JMenuItem();
233     jSeparator1 = new javax.swing.JPopupMenu.Separator();
234     helpAboutMenuItem = new javax.swing.JMenuItem();
235
236     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
237     setTitle("TweakMC");
238     setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
239     setFocusable(false);
240     setMinimumSize(new java.awt.Dimension(1024, 768));
241
242     vehiclePane.setName("vehiclePane"); // NOI18N
243     vehiclePane.setPreferredSize(new java.awt.Dimension(500, 300));
244     vehiclePane.addChangeListener(new javax.swing.event.ChangeListener() {
245         public void stateChanged(javax.swing.event.ChangeEvent evt) {
246             vehiclePaneStateChanged(evt);
247         }
248     });
249
250     loginTabPanel.setName("loginTabPanel"); // NOI18N
251
252     loginTestPanel.setName("loginTestPanel"); // NOI18N
253
254     javax.swing.GroupLayout loginTestPanelLayout = new javax.swing.GroupLayout(loginTestPanel);
255     loginTestPanel.setLayout(loginTestPanelLayout);
256     loginTestPanelLayout.setHorizontalGroup(
257         loginTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
258             .addGap(0, 1239, Short.MAX_VALUE)
259     );
260     loginTestPanelLayout.setVerticalGroup(
261         loginTestPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
262             .addGap(0, 974, Short.MAX_VALUE)

```



```

263 );
264
265 javax.swing.GroupLayout loginTabPanelLayout = new javax.swing.GroupLayout(loginTabPanel);
266 loginTabPanel.setLayout(loginTabPanelLayout);
267 loginTabPanelLayout.setHorizontalGroup(
268     loginTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
269         .addComponent(loginTestPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
270 );
271 loginTabPanelLayout.setVerticalGroup(
272     loginTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
273         .addComponent(loginTestPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
274 );
275
276 vehiclePane.addTab("Login", loginTabPanel);
277
278 vTabPanel.setName("vTabPanel"); // NOI18N
279
280 scrollbar1.setName("scrollbar1"); // NOI18N
281
282 javax.swing.GroupLayout vTabPanelLayout = new javax.swing.GroupLayout(vTabPanel);
283 vTabPanel.setLayout(vTabPanelLayout);
284 vTabPanelLayout.setHorizontalGroup(
285     vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
286         .addGap(0, 1239, Short.MAX_VALUE)
287         .addGroup(vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
288             .addGroup(vTabPanelLayout.createSequentialGroup()
289                 .addGap(0, 0, Short.MAX_VALUE)
290                 .addComponent(scrollbar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
291                 .addGap(0, 1223, Short.MAX_VALUE)))
292 );
293 vTabPanelLayout.setVerticalGroup(
294     vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
295         .addGap(0, 974, Short.MAX_VALUE)
296         .addGroup(vTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
297             .addGroup(vTabPanelLayout.createSequentialGroup()
298                 .addGap(0, 0, Short.MAX_VALUE)
299                 .addComponent(scrollbar1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
300                 .addGap(0, 926, Short.MAX_VALUE)))
301 );
302
303 vehiclePane.addTab("Vehicle", vTabPanel);
304
305 ownerTabPanel.setName("ownerTabPanel"); // NOI18N
306
307 javax.swing.GroupLayout ownerTabPanelLayout = new javax.swing.GroupLayout(ownerTabPanel);
308 ownerTabPanel.setLayout(ownerTabPanelLayout);
309 ownerTabPanelLayout.setHorizontalGroup(
310     ownerTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
311         .addGap(0, 1239, Short.MAX_VALUE)
312 );
313 ownerTabPanelLayout.setVerticalGroup(
314     ownerTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
315         .addGap(0, 974, Short.MAX_VALUE)
316 );

```

```

317
318     vehiclePane.addTab("Owner", ownerTabPanel);
319
320     jLabel1.setText("User logged in:");
321     jLabel1.setName("jLabel1"); // NOI18N
322
323     statusPanel.setName("statusPanel"); // NOI18N
324
325     loggedInStatus.setText("Not logged in!");
326     loggedInStatus.setName("loggedInStatus"); // NOI18N
327
328     javax.swing.GroupLayout statusPanelLayout = new javax.swing.GroupLayout(statusPanel);
329     statusPanel.setLayout(statusPanelLayout);
330     statusPanelLayout.setHorizontalGroup(
331         statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
332             .addGroup(statusPanelLayout.createSequentialGroup()
333                 .addComponent(loggedInStatus, javax.swing.GroupLayout.PREFERRED_SIZE, 98,
334                     javax.swing.GroupLayout.PREFERRED_SIZE)
335                 .addContainerGap(27, Short.MAX_VALUE))
336             .addGroup(statusPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
337                 .addGroup(statusPanelLayout.createSequentialGroup()
338                     .addComponent(loggedInStatus, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
339                         javax.swing.GroupLayout.PREFERRED_SIZE)
340                     .addContainerGap())
341             );
342
343     jLabel2.setText("jLabel2");
344     jLabel2.setName("jLabel2"); // NOI18N
345
346     jMenuBar1.setName("jMenuBar1"); // NOI18N
347
348     jMenu1.setText("File");
349     jMenu1.setName("jMenu1"); // NOI18N
350
351     jMenuItem1.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_Q,
352         java.awt.event.InputEvent.CTRL_MASK));
353     jMenuItem1.setText("Quit");
354     jMenuItem1.setName("jMenuItem1"); // NOI18N
355     jMenuItem1.addActionListener(new java.awt.event.ActionListener() {
356         public void actionPerformed(java.awt.event.ActionEvent evt) {
357             jMenuItem1ActionPerformed(evt);
358         }
359     });
360     jMenu1.add(jMenuItem1);
361
362     jMenuBar1.add(jMenu1);
363
364     jMenu3.setText("Help");
365     jMenu3.setName("jMenu3"); // NOI18N
366     jMenu3.addActionListener(new java.awt.event.ActionListener() {
367         public void actionPerformed(java.awt.event.ActionEvent evt) {
368             jMenu3ActionPerformed(evt);
369         }
370     });

```

```

371     jMenuItem3.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.VK_H,
java.awt.event.InputEvent.CTRL_MASK));
372     jMenuItem3.setText("Report technical issue");
373     jMenuItem3.setName("jMenuItem3"); // NOI18N
374     jMenuItem3.addActionListener(new java.awt.event.ActionListener() {
375         public void actionPerformed(java.awt.event.ActionEvent evt) {
376             jMenuItem3ActionPerformed(evt);
377         }
378     });
379     jMenu3.add(jMenuItem3);
380
381     jMenuItem4.setText("View logfile..");
382     jMenuItem4.setName("jMenuItem4"); // NOI18N
383     jMenu3.add(jMenuItem4);
384
385     helpMenuItem.setText("Please... I need help :(");
386     helpMenuItem.setName("helpMenuItem"); // NOI18N
387     helpMenuItem.addActionListener(new java.awt.event.ActionListener() {
388         public void actionPerformed(java.awt.event.ActionEvent evt) {
389             helpMenuItemActionPerformed(evt);
390         }
391     });
392     jMenu3.add(helpMenuItem);
393
394     jSeparator1.setName("jSeparator1"); // NOI18N
395     jMenu3.add(jSeparator1);
396
397     helpAboutMenuItem.setText("About");
398     helpAboutMenuItem.setName("helpAboutMenuItem"); // NOI18N
399     helpAboutMenuItem.addActionListener(new java.awt.event.ActionListener() {
400         public void actionPerformed(java.awt.event.ActionEvent evt) {
401             helpAboutMenuItemActionPerformed(evt);
402         }
403     });
404     jMenu3.add(helpAboutMenuItem);
405
406     jMenuBar1.add(jMenu3);
407
408     setJMenuBar(jMenuBar1);
409
410     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
411     getContentPane().setLayout(layout);
412     layout.setHorizontalGroup(
413         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
414             .addGroup(layout.createSequentialGroup()
415                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
416                     .addGroup(layout.createSequentialGroup()
417                         .addComponent(vehiclePane, javax.swing.GroupLayout.DEFAULT_SIZE, 1244,
Short.MAX_VALUE)
418                         .addGroup(layout.createSequentialGroup()
419                             .addComponent(jLabel1)
420                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
421                             .addComponent(statusPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
422                             .addGap(350, 350, 350)
423                             .addComponent(jLabel2)))
424                     .addContainerGap())
425             );

```

```

426     layout.setVerticalGroup(
427         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
428         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
429             .addContainerGap()
430             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
431                 .addComponent(jLabel1)
432                 .addComponent(statusPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
433                 .addComponent(jLabel2))
434             .addGap(18, 18, 18)
435             .addComponent(vehiclePane, javax.swing.GroupLayout.PREFERRED_SIZE, 1002,
javax.swing.GroupLayout.PREFERRED_SIZE)
436             .addGap(6822, 6822, 6822))
437     );
438
439     java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
440     setBounds((screenSize.width-1280)/2, (screenSize.height-1115)/2, 1280, 1115);
441 }// </editor-fold>
442
443 private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
444     quit();
445 }
446
447 private void helpAboutMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
448     /**
449      * Displays the about-box, called from the menu bar
450      * Needs a try, catch as the image is loaded from an URL.
451      */
452     try
453     {
454         aboutBox();
455     }// End try
456     catch (Exception e)
457     {
458
459     }// End catch
460
461
462 }
463
464 private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
465     MailGUI helpGUI = new MailGUI();
466     new MailGUI().setVisible(true);
467 }
468
469 private void vehiclePaneStateChanged(javax.swing.event.ChangeEvent evt) {
470     // TODO add your handling code here:
471 }
472
473 private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
474
475 }
476
477 private void helpMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
478     // Calls the helpItem method.
479     try
480     {
481         helpItem();

```

```

482     }//
483     catch (Exception e)
484     {
485     }
486     }
487 }
488
489 /**
490  * Main method for the GUI
491  * @param args the command line arguments
492  */
493 public static void main(String args[])
494 {
495     java.awt.EventQueue.invokeLater(new Runnable()
496     {
497         public void run()
498         {
499             new GUIDesign().setVisible(true);
500
501             }// end run method (inner class)
502     });
503 }// end main method
504
505 /**
506  * Method for quitting the system
507  */
508 private void quit()
509 {
510     System.exit(0);
511 }// End quit method
512
513 /**
514  * The help menu item
515  * @throws Exception
516  */
517 private void helpItem() throws Exception
518 {
519     URL imageLocation = new URL("http://db.tt/eylUfgX");
520     JOptionPane.showMessageDialog(null, "HELP!\n\n"
521         + "As the image to your left shows,\n"
522         + "there is only one way of helping yourself.\n"
523         + "\n"
524         + "Need more help?\n"
525         + "Read a book?", "MC Help",
526     JOptionPane.PLAIN_MESSAGE, new ImageIcon(imageLocation));
527 }
528
529 /**
530  * Creates the about box with an image
531  * @throws Exception
532  */
533 private void aboutBox() throws Exception
534 {
535     URL imageLocation = new URL("http://db.tt/l69alCU");
536     JOptionPane.showMessageDialog(null, "TweakMC for KJ Motorcykler\n\n"
537         + "Version 1.0.3\n"
538         + "By TeaN TALC\n"
539         + "welovefailing.com\n"

```

```

540         + "All rights reserved 2010 © ", "About TweakMC",
541         JOptionPane.PLAIN_MESSAGE, new ImageIcon(imageLocation));
542     } // end aboutBox method
543
544     // Variables declaration - do not modify
545     private javax.swing.JMenuItem helpAboutMenuItem;
546     private javax.swing.JMenuItem helpMenuItem;
547     private javax.swing.JLabel jLabel1;
548     private javax.swing.JLabel jLabel2;
549     private javax.swing.JMenu jMenu1;
550     private javax.swing.JMenu jMenu3;
551     private javax.swing.JMenuBar jMenuBar1;
552     private javax.swing.JMenuItem jMenuItem1;
553     private javax.swing.JMenuItem jMenuItem3;
554     private javax.swing.JMenuItem jMenuItem4;
555     private javax.swing.JPopupMenu.Separator jSeparator1;
556     public javax.swing.JLabel loggedInStatus;
557     private javax.swing.JPanel loginTabPanel;
558     private javax.swing.JPanel loginTestPanel;
559     private javax.swing.JPanel ownerTabPanel;
560     private java.awt.Scrollbar scrollbar1;
561     public javax.swing.JPanel statusPanel;
562     private javax.swing.JPanel vTabPanel;
563     private javax.swing.JTabbedPane vehiclePane;
564     // End of variables declaration
565
566     /**
567     * Scale an image to 200 * 190 for thumbnail
568     * @param src image to scale
569     * @return new scaled ImageIcon
570     */
571     private ImageIcon scale(Image src)
572     {
573         int w = 207;
574         int h = 190;
575         int type = BufferedImage.TYPE_INT_RGB;
576         BufferedImage dst = new BufferedImage(w, h, type);
577         Graphics2D g2 = dst.createGraphics();
578         g2.drawImage(src, 0, 0, w, h, this);
579         g2.dispose();
580         return new ImageIcon(dst);
581     }
582
583
584 }

```

GUIFrame

```

1  /*
2  * This is the main GUI.
3  * It builds up the engine of the system.
4  */
5
6  package tweakmc.view;
7
8  import tweakmc.utility.GUIHelpUtil;
9  import java.awt.BorderLayout;

```

```

10 import java.awt.Container;
11 import java.awt.Dimension;
12 import java.awt.Toolkit;
13 import java.awt.event.ActionEvent;
14 import java.awt.event.ActionListener;
15 import java.awt.event.KeyEvent;
16 import java.awt.event.KeyListener;
17 import java.awt.event.WindowAdapter;
18 import java.awt.event.WindowEvent;
19 import java.util.Properties;
20 import javax.swing.JButton;
21 import javax.swing.JComponent;
22 import javax.swing.JFrame;
23 import javax.swing.JLabel;
24 import javax.swing.JMenu;
25 import javax.swing.JMenuBar;
26 import javax.swing.JMenuItem;
27 import javax.swing.JOptionPane;
28 import javax.swing.JTabbedPane;
29 import javax.swing.JTextField;
30 import javax.swing.KeyStroke;
31 import javax.swing.UIManager;
32 import javax.swing.UIManager.LookAndFeelInfo;
33 import tweakmc.control.FileReaderController;
34 import tweakmc.utility.StartupTest;
35 import tweakmc.dataaccess.DBHandler;
36 import tweakmc.utility.FileWriterException;
37 import tweakmc.utility.Mail;
38
39 /**
40  * GUI main to hold all tabpanes menus
41  * @author clausPallisgaardBeck
42  */
43 public class GUIFrame extends JFrame
44 {
45     private WaitFrame wait;
46     private Toolkit tKit = Toolkit.getDefaultToolkit();
47     private Dimension actScreenSize;
48     private Properties comProp = System.getProperties();
49
50     //Primitive datatypes
51     private int actScreenResolution;
52
53     //Swing and AWT
54     private static JFrame frame;
55     private Container contentPane;
56     private JTabbedPane mainTabPane;
57     private JLabel statusLoginLabel;
58     private JTextField initialsTField;
59     private static JComponent focusComponent = null;
60     private static JButton defaultButton = null;
61     /**
62      * Construtor for building GUIFrame for TweakMC
63      */
64     public GUIFrame()
65     {
66         DBHandler.getInstance().setDataBase();
67         StartupTest stut = new StartupTest();

```

```

68     if(!stut.startUpTests())
69     {
70         //Custom button text
71         Object[] options = { "Yes, please, and a MySQL to go",
72             "No, thanks, i'll have the Derby to stay",
73             "No eggs, no ham!" };
74         int n = JOptionPane.showOptionDialog(frame,
75             "Would you like some green eggs to go "
76             + "with that ham?",
77             "A Silly Question",
78             JOptionPane.YES_NO_CANCEL_OPTION,
79             JOptionPane.QUESTION_MESSAGE,
80             null,
81             options,
82             options[2]);
83         System.out.println(n);
84         DBHandler.getInstance().setDataBase(n);
85         StartUpTest stut2 = new StartUpTest();
86         if(!stut2.startUpTests())
87         {
88             JOptionPane.showMessageDialog(null, "Database error!");
89         }
90     } //End if
91
92     try
93     {
94         for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
95         {
96             if ("Nimbus".equals(info.getName()))
97             {
98                 UIManager.setLookAndFeel(info.getClassName());
99                 break;
100             } //End if
101         } //End for
102     } // End try
103
104     catch (Exception e)
105     {
106         FileWriterException.writeLogFile("Look and feel settings: " + e.getMessage());
107     } //End catch
108
109     finally
110     {
111         actScreenSize = tKit.getScreenSize();
112         actScreenResolution = tKit.getScreenResolution();
113         initComponents();
114         //SendLogfile(); //Regards from Nyvang :)
115     } //End finally
116 } //End constructor
117
118 /**
119  * Send logfile via email to errorlog@welovefailing.com
120  */
121 private void sendLogfile()
122 {
123     Mail sslObj = new Mail();
124     Mail.MESSAGE_SEND = FileReaderController.readLogfile();
125     Mail.MAIL_TO = "errorlog@welovefailing.com";

```



```

126     try
127     {
128         String[] args = null;
129         WaitFrame.main(args, actScreenSize);
130         sslObj.mail();
131     }// End try
132
133     catch (Exception e)
134     {
135         if(e.getMessage() == null)
136         {
137             FileWriterException.writeLogFile(GUIFrame.class.getName() + "/sendLogfile-IF/ " +
138                 e.getMessage());
139         }//End if
140
141         else
142         {
143             FileWriterException.writeLogFile(GUIFrame.class.getName() + "/send logfile-ELSE/ " +
144                 e.getMessage());
145             JOptionPane.showMessageDialog(frame, "Could not send logfile, please send a technical issue\n" +
146                 "Close this window and press CTRL+H");
147         }//End else
148     }// End catch
149 }//End method sendLogfile
150
151 /**
152  * Backup system to a given position
153  */
154 private void backupSystem(String position)
155 {
156
157 }//End method backupSystem
158
159 /**
160  * Disable tabs in maintabmenu
161  */
162 private void disableTabs()
163 {
164     for (int i = 1; i < mainTabPane.getTabCount(); i++)
165     {
166         mainTabPane.setEnabledAt(i, false);
167     }//End for loop
168 }//End method disableTabs
169
170 /**
171  * Logout user out of system, disable tabs and go to login tab
172  * @return true if logput else false
173  */
174 private boolean systemLogout()
175 {
176     return GUIHelpUtil.trySystemLogout(null);
177 }//End systemLogout
178
179 ///////////////////////////////////////////////////
180 ///////////////////////////////////////////////////GUI STARTS HERE//////////////////////////////////////
181 /**
182  * Building up frame and add panes and menus
183  */

```

```

184 private void initComponents()
185 {
186     frame = new JFrame("TweakMC");
187     frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
188     frame.setPreferredSize(new Dimension(actScreenSize.width-4, actScreenSize.height-54));
189     frame.setMinimumSize(new Dimension(800, 600));
190     frame.setMaximumSize(new Dimension(actScreenSize.width-4, actScreenSize.height-54));
191     frame.addWindowFocusListener(new WindowAdapter()
192     {
193         @Override
194         public void windowGainedFocus(WindowEvent e)
195         {
196             focusComponent.requestFocusInWindow();
197         } //End windowGainedFocus
198     }); //End windowFocusListner
199     contentPane = frame.getContentPane();
200
201     addFileMenu();
202     defineContentPane();
203     addTabPaneMeu();
204
205     updateUI();
206
207 } //End initComponents()
208
209 /**
210  * Updates theUI JFrame
211  */
212 private void updateUI()
213 {
214     frame.validate();
215     frame.pack();
216     frame.setVisible(true);
217 } //End method updateUI
218
219 /**
220  * Set a JComponent in focus then requested
221  * @param actComp to set in focus
222  */
223 public static void setFocusComponent(JComponent actComp)
224 {
225     focusComponent = actComp;
226 } //End method setFocusComponent
227
228 /**
229  * Set default button in a given window
230  * @param actButton to set as default
231  */
232 public static void setDefaultButton(JButton actButton)
233 {
234     frame.getRootPane().setDefaultButton(actButton);
235 } //End method setDefaultButton
236
237 /**
238  * Add JMenuBar and menus for this
239  */
240 private void addFileMenu()
241 {

```

```

242 JMenu menu;
243 JMenuItem menuItem;
244
245 JMenuBar jmb = new JMenuBar();
246
247 //Add menu to bar
248 menu = new JMenu("ManageSystem");
249 menu.setMnemonic(KeyEvent.VK_M);
250 jmb.add(menu);
251
252 //Add menuItems to manageSystem menu
253 menuItem = new JMenuItem("Logout user");
254 menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_L, ActionEvent.CTRL_MASK));
255 menuItem.addActionListener(new ActionListener()
256 {
257     public void actionPerformed(ActionEvent e)
258     {
259         if(systemLogout())
260         {
261             try
262             {
263                 mainTabPane.setEnabledAt(0, true);
264                 mainTabPane.setSelectedIndex(0);
265                 disableTabs();
266             }//End try
267
268             catch (IndexOutOfBoundsException eobe)
269             {
270                 FileWriterException.writeLogFile(GUIFrame.class.getName() +
271                     "/logoutUser/" + eobe.getMessage());
272             }//End catch
273         }//End if
274     }//End method actionPerformed
275 });//End ActionListener
276 menu.add(menuItem);
277
278 menu.addSeparator();
279
280 menuItem = new JMenuItem("Register user");
281 menu.add(menuItem);
282
283 menuItem = new JMenuItem("Edit user");
284 menu.add(menuItem);
285
286 menuItem = new JMenuItem("Delete user");
287 menu.add(menuItem);
288
289 menu.addSeparator();
290
291 menuItem = new JMenuItem("Manage company information");
292 menuItem.addActionListener(new ActionListener()
293 {
294     public void actionPerformed(ActionEvent e)
295     {
296         CompanyGui helpGui = new CompanyGui();
297         new CompanyGui().setVisible(true);
298     }//End method actionPerformed
299 });//End ActionListener

```

```

300 menu.add(menuItem);
301
302
303 //Add menu to bar
304 menu = new JMenu("Help");
305 menu.setMnemonic(KeyEvent.VK_H);
306 jmb.add(menu);
307
308 //Add menuItems to Help menu
309 menuItem = new JMenuItem("Report technical issue");
310 menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_H, ActionEvent.CTRL_MASK));
311 menuItem.addActionListener(new ActionListener()
312 {
313     public void actionPerformed(ActionEvent e)
314     {
315         MailGUI helpGUI = new MailGUI();
316         new MailGUI().setVisible(true);
317     } //End method actionPerformed
318 }); //End new ActionListener
319 menu.add(menuItem);
320
321 menuItem = new JMenuItem("Please...I need help!:(");
322 menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F3, 0));
323 menuItem.addActionListener(new ActionListener()
324 {
325     public void actionPerformed(ActionEvent e)
326     {
327         try
328         {
329             GUIHelpUtil.helpItem();
330         } //End try
331
332         catch (Exception ex)
333         {
334             FileWriterException.writeLogFile(GUIFrame.class.getName() + "/helpItem/ " + ex.getMessage());
335         } //End catch
336     } //End method actionPerformed
337 }); //End new ActionListener
338 menu.add(menuItem);
339
340
341 menuItem = new JMenuItem("Index");
342 menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F1, 0));
343 menuItem.addActionListener(new ActionListener()
344 {
345     public void actionPerformed(ActionEvent e)
346     {
347         try
348         {
349             JavaHelp.main();
350         } //End try
351
352         catch (Exception ex)
353         {
354             FileWriterException.writeLogFile(JavaHelp.class.getName() + "/helpIndex/ " + ex.getMessage());
355         } //End catch
356     } //End method actionPerformed
357 }); //End new ActionListener

```

```

358 menu.add(menuItem);
359 menu.addSeparator();
360
361 menuItem = new JMenuItem("Send logfile");
362 menuItem.addActionListener(new ActionListener()
363 {
364     public void actionPerformed(ActionEvent e)
365     {
366         sendLogfile();
367     } //End method actionPerformed
368 }); //End method actionlistener
369 menu.add(menuItem);
370
371 //Add utility menu to bar
372 menu = new JMenu("Utility");
373 menu.setMnemonic(KeyEvent.VK_U);
374 jmb.add(menu);
375
376 //Add menuitem to utility menu
377 menuItem = new JMenuItem("Calculator");
378 menuItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F2, 0));
379 menuItem.addActionListener(new ActionListener()
380 {
381     @Override
382     public void actionPerformed(ActionEvent e)
383     {
384         if(!GUIHelpUtil.openCalculator())
385         {
386             JOptionPane.showMessageDialog(null, "Couldn't open extern Calculator!\n" +
387                 "Try to open local calculator from computers Start menu",
388                 "TweakMC - Information", JOptionPane.ERROR_MESSAGE);
389         } //End if
390     } //End method actionPerformed
391 }); //End actionListener
392 menu.add(menuItem);
393
394 menu.addSeparator();
395
396 menuItem = new JMenuItem("Open pdf");
397 menuItem.addActionListener(new ActionListener()
398 {
399
400     public void actionPerformed(ActionEvent e)
401     {
402         GUIHelpUtil.doPdfThings();
403     } //End method actionPerformed
404 }); //End actionListener
405 menu.add(menuItem);
406
407 //Add menu to bar
408 menu = new JMenu("About");
409 menu.setMnemonic(KeyEvent.VK_A);
410 jmb.add(menu);
411
412 //Add menuItems to About menu
413 menuItem = new JMenuItem("About - TweakMC");
414 menuItem.addActionListener(new ActionListener()
415 {

```

```

416     public void actionPerformed(ActionEvent e)
417     {
418         try
419         {
420             GUIHelpUtil.aboutTweakMC();
421         }
422
423         catch (Exception ex)
424         {
425             FileWriterException.writeLogFile(GUIFrame.class.getName() + "/ aboutButton/ " + ex.getMessage());
426         }
427     } //End method actionPerformed
428 }; //End method ActionListener
429 menu.add(menuItem);
430
431 //Add menu to bar
432 menu = new JMenu("Close");
433 menu.setMnemonic(KeyEvent.VK_C);
434 jmb.add(menu);
435
436 //Add menuItems to Close menu
437 menuItem = new JMenuItem("Close");
438 menuItem.addActionListener(new ActionListener()
439 {
440     public void actionPerformed(ActionEvent e)
441     {
442         int answer = JOptionPane.showConfirmDialog(null,
443             "Do you want to close TweakMC?",
444             "Closing TweakMC!", JOptionPane.YES_NO_OPTION,
445             JOptionPane.QUESTION_MESSAGE);
446
447         if(answer == JOptionPane.YES_OPTION)
448         {
449             System.exit(0);
450         } //end if
451
452         else
453         {
454             return;
455         } //end else
456     } //End actionPerformed
457 }; //End method ActionListener
458 menu.add(menuItem);
459
460 menu.addSeparator();
461
462 menuItem = new JMenuItem("BackUp System");
463 menuItem.addActionListener(new ActionListener()
464 {
465     public void actionPerformed(ActionEvent e)
466     {
467         backupSystem("position");
468     } //End method actionPerformed
469 }; //End method ActionListener
470 menu.add(menuItem);
471
472 //Add JMenuBar to GUIFrame
473 frame.setJMenuBar(jmb);

```

```

474 }//End method addFileMenu
475
476 /**
477  * Setup the contentpane
478  */
479 private void defineContentPane()
480 {
481     contentPane.setLayout(new BorderLayout());
482 }//End method defineContentPane
483
484
485 /**
486  * Add tabpane menu to contentpane
487  */
488 private void addTabPaneMenu()
489 {
490     //Build and populate tabPane
491     mainTabPane = new JTabbedPane(JTabbedPane.LEFT);
492     mainTabPane.addTab("Login", new Login1(mainTabPane).getPanel());
493     mainTabPane.addTab("Owner", new OwnerGUI().getPanel());
494     mainTabPane.addTab("Vehicle", new VehicleGUI().getPanel());
495     mainTabPane.addTab("Order", new OrderGUI().getPanel());
496     mainTabPane.addTab("WorkStation", new WSGUI().getWorkstationPanel());
497
498     disableTabs();
499
500     contentPane.add(mainTabPane, BorderLayout.CENTER);
501 }//End method addTabPaneMenu
502
503
504 /**
505  * @param args the command line arguments
506  */
507 public static void main(String[] args) {
508     java.awt.EventQueue.invokeLater(new Runnable() {
509         public void run() {
510             new GUIFrame();
511         }
512     });
513 }
514
515 }

```

JavaHelp

```

1 /*
2  * This class is a "Help" class
3  * It calls an XML file with all help content.
4  * Everything is put inside a new JFrame
5  */
6
7
8 /*
9  * JavaHelp.java
10 *
11 * Created on 07-12-2010, 00:53:08
12 */

```

```

13
14 package tweakmc.view;
15
16 import javax.help.*;
17 import java.net.URL;
18 import javax.swing.*;
19
20 /**
21  *
22  * @author Nyvang
23  */
24
25 public class JavaHelp {
26
27     /**
28      * Creates the JavaHelp frame by calling the helpset file.
29      * When done, the frame is created and the helpViewer are added to the content pane
30      */
31     public static void main()
32     {
33         JHelp helpViewer = null;
34         try
35         {
36             // Get the classloader of this class.
37             ClassLoader cl = JavaHelp.class.getClassLoader();
38             // Use the findHelpSet method of HelpSet to create a URL referencing the helpset file.
39             URL url = HelpSet.findHelpSet(cl, "data\\jhelpset.hs");
40             // Create a new JHelp object with a new HelpSet.
41             helpViewer = new JHelp(new HelpSet(cl, url));
42             // Set the initial entry point in the table of contents.
43             helpViewer.setCurrentID("Root");
44         } //end try
45         catch (Exception e)
46         {
47             //left emty on purpose
48         } //end catch
49
50         // Create a new frame.
51         JFrame frame = new JFrame("TweakMC Help System");
52         // Set it's size.
53         frame.setSize(800,600);
54         // Add the created helpViewer to it.
55         frame.getContentPane().add(helpViewer);
56         // Set a default close operation.
57         frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
58         // Place the frame just a little from the top left corner
59         frame.setBounds(200, 200, 800, 600);
60         // Make the frame visible.
61         frame.setVisible(true);
62     } //end main method
63 } //end Class JavaHelp

```

Helptoc

```

1
2 <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
3 <!DOCTYPE toc PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp TOC Version 1.0//EN"
"http://java.sun.com/products/javahelp/toc_1_0.dtd">
4 <toc version="1.0">

```



```

5 <tocitem target="Root" text="Contents">
6   <tocitem target="Root.Introduction" text="Shortcuts"></tocitem>
7   <tocitem target="Root.Explanation" text="TweakMC">
8     <tocitem target="Root.Explanation.Files" text="Howto"></tocitem>
9     <tocitem target="Root.Explanation.Login" text="Login"></tocitem>
10    <tocitem target="Root.Explanation.Vehicle" text="Vehicle/Order"></tocitem>
11    <tocitem target="Root.Explanation.Order" text="Order"></tocitem>
12    <tocitem target="Root.Explanation.Spreadsheets" text="Spreadsheets">
13      <tocitem target="Root.Explanation.Spreadsheets.Repairstation" text="Repairstation"></tocitem>
14      <tocitem target="Root.Explanation.Spreadsheets.Dynostation" text="Dynostation"></tocitem>
15      <tocitem target="Root.Explanation.Spreadsheets.Enginestation" text="Enginestation"></tocitem>
16      <tocitem target="Root.Explanation.Spreadsheets.Suspensionstation"
text="Suspensionstation"></tocitem>
17    </tocitem>
18    <tocitem target="Root.Explanation.Email" text="E-mail"></tocitem>
19  </tocitem>
20 </tocitem>
21 </toc>
22

```

Jhindexer (c++)

```

1 #! /bin/sh
2 # This builds a search database
3
4 # Cygwin support. $cygwin _must_ be set to either true or false.
5 case "`uname`" in
6   CYGWIN*) cygwin=true ;;
7   *) cygwin=false ;;
8 esac
9
10 # For Cygwin, ensure paths are in UNIX format before anything is touched
11 if $cygwin; then
12   [ -n "$JAVAHELP_HOME" ] &&
13   JAVAHELP_HOME=`cygpath --unix "$JAVAHELP_HOME"`
14 fi
15
16 if [ "$JAVAHELP_HOME" = "" ] ; then
17   # try to find jhindexer
18   if [ -d /opt/javahelp ] ; then
19     JAVAHELP_HOME=/opt/javahelp
20   fi
21
22   if [ -d ${HOME}/opt/jhindexer ] ; then
23     JAVAHELP_HOME=${HOME}/opt/javahelp
24   fi
25
26   ## resolve links - $0 may be a link to javahelp's home
27   PRG=$0
28   progname=`basename $0`
29
30   while [ -h "$PRG" ] ; do
31     ls=`ls -ld "$PRG"`
32     link=`expr "$ls" : '.*-> \(.*\)${`'`
33     if expr "$link" : '.*/.*' > /dev/null; then
34       PRG="$link"
35     else

```

```

36 PRG="`dirname $PRG`/$link"
37 fi
38 done
39
40 JAVAHELP_HOME=`dirname "$PRG"/../..
41
42 fi
43
44 # For Cygwin, switch paths to Windows format before running java
45 if $cygwin; then
46 JAVAHELP_HOME=`cygpath --path --windows "$JAVAHELP_HOME"`
47 fi
48
49 java -jar $JAVAHELP_HOME/javahelp/bin/jhindexer.jar "$@"
50
51

```

Helpindex

```

1 <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
2 <!DOCTYPE index PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp Index Version 1.0//EN"
"http://java.sun.com/products/javahelp/index_1_0.dtd">
3
4 <!--
5 Document : jhelpidx.xml
6 Created on : 6. december 2010, 21:55
7 Author : Nyvang
8 Description:
9 Purpose of the document follows.
10 -->
11
12
13 <index version="1.0">
14 <indexitem text="Vehicle" target="id47"/>
15 </index>
16

```

jhhelp

```

1 <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
2 <!DOCTYPE toc PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp TOC Version 1.0//EN"
"http://java.sun.com/products/javahelp/toc_1_0.dtd">
3
4 <!--
5 Document : jhelptoc.xml.xml
6 Created on : 6. december 2010, 21:54
7 Author : Nyvang
8 Description:
9 Purpose of the document follows.
10 -->
11
12 <toc version="1.0">
13 <tocitem target="Root" text="Title">
14 <tocitem target="Root.Introduction" text="Shortcuts"></tocitem>
15 <tocitem target="Root.Explanation" text="Index">
16 <tocitem target="Root.Explanation.Files" text="About"></tocitem>
17 <tocitem target="Root.Explanation.Login" text="Login"></tocitem>

```

```

18         <tocitem target="Root.Explanation.Vehicle" text="Vehicle & Owner"></tocitem>
19         <tocitem target="Root.Explanation.Order" text="Order"></tocitem>
20         <tocitem target="Root.Explanation.Spreadsheets" text="Spreadsheets">
21             <tocitem target="Root.Explanation.Spreadsheets.Repairstation" text="Repairstation"></tocitem>
22             <tocitem target="Root.Explanation.Spreadsheets.Dynostation" text="Dynostation"></tocitem>
23             <tocitem target="Root.Explanation.Spreadsheets.Enginestation" text="Enginestation"></tocitem>
24             <tocitem target="Root.Explanation.Spreadsheets.Suspensionstation"
text="Suspensionstation"></tocitem>
25         </tocitem>
26         <tocitem target="Root.Explanation.Email" text="E-mail"></tocitem>
27     </tocitem>
28
29 </tocitem>
30 </toc>
31

```

MailGUI

```

1 /*
2  * This Class is a HelpDesk System, that gives the user the orpotunity to send
3  * an email to the developers.
4  * It will automaticly also send a logfile of the system with
5  */
6
7 /*
8  * MailGUI.java
9  *
10 * Created on 08-11-2010, 00:02:27
11 */
12
13 package tweakmc.view;
14
15 import java.awt.Graphics2D;
16 import java.awt.Image;
17 import java.awt.image.BufferedImage;
18 import javax.swing.ImageIcon;
19 import tweakmc.utility.Mail;
20 import javax.swing.JOptionPane;
21 import javax.swing.UIManager;
22 import javax.swing.UIManager.LookAndFeelInfo;
23 import tweakmc.utility.FileWriterException;
24
25 /**
26  *
27  * @author Nyvang
28  */
29 public class MailGUI extends javax.swing.JFrame
30 {
31     // Instance Variables
32     private ImageIcon bufferedIcon = new ImageIcon("tweakMC_Helpdesk_System.jpg");
33     private ImageIcon bufferedIcon2 = new ImageIcon("teanTALC_2010.jpg");
34     private ImageIcon tweakMC = scale(bufferedIcon.getImage());
35     private ImageIcon teanTalc = scale2(bufferedIcon2.getImage());
36     private boolean mailSent = false;
37
38
39     /** Creates new form MailGUI */

```

```

40 public MailGUI()
41 {
42     try
43     {
44         for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
45         {
46             if ("Nimbus".equals(info.getName()))
47             {
48                 UIManager.setLookAndFeel(info.getClassName());
49                 break;
50             } //end if
51         } //end for
52     } // end try
53
54     catch (Exception e)
55     {
56         FileWriterException.writeLogFile("Look and feel settings: " + e.getMessage());
57     } //end catch
58
59     finally
60     {
61         initComponents();
62         labelHeader.setIcon(tweakMC);
63         labelButtom.setIcon(teanTalc);
64     } //end finally
65 } //end MailGUI constructor
66
67 /**
68  * This method is called from within the constructor to
69  * initialize the form.
70  * WARNING: Do NOT modify this code. The content of this method is
71  * always regenerated by the Form Editor.
72  * @return
73  */
74 @SuppressWarnings("unchecked")
75 // <editor-fold defaultstate="collapsed" desc="Generated Code">
76 private void initComponents() {
77
78     jScrollPane1 = new javax.swing.JScrollPane();
79     messageSend = new javax.swing.JTextPane();
80     jLabel3 = new javax.swing.JLabel();
81     jButton1 = new javax.swing.JButton();
82     jScrollPane2 = new javax.swing.JScrollPane();
83     jTextArea1 = new javax.swing.JTextArea();
84     jLabel4 = new javax.swing.JLabel();
85     jTextField1 = new javax.swing.JTextField();
86     buttonCancel = new javax.swing.JButton();
87     jPanel2 = new javax.swing.JPanel();
88     labelHeader = new javax.swing.JLabel();
89     panelButtom = new javax.swing.JPanel();
90     labelButtom = new javax.swing.JLabel();
91
92     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
93     setTitle("TweakMC error reporting system");
94
95     jScrollPane1.setViewportView(messageSend);
96
97     jLabel3.setText("Describe the problem in the text field below");

```

```

98
99     jButton1.setText("Send e-mail");
100    jButton1.addActionListener(new java.awt.event.ActionListener() {
101        public void actionPerformed(java.awt.event.ActionEvent evt) {
102            jButton1ActionPerformed(evt);
103        }
104    });
105
106    jTextArea1.setBackground(new java.awt.Color(240, 240, 240));
107    jTextArea1.setColumns(20);
108    jTextArea1.setEditable(false);
109    jTextArea1.setFont(new java.awt.Font("Tahoma", 0, 11));
110    jTextArea1.setRows(5);
111    jTextArea1.setText("Please describe the issues you\nare experiencing\nas precise as \npossible.\n\nThanks in
advance.");
112
113    jTextArea1.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
114    jTextArea1.setCaret(jTextArea1.getCaret());
115    jTextArea1.setCaretColor(new java.awt.Color(240, 240, 240));
116    jTextArea1.setDisabledTextColor(new java.awt.Color(240, 240, 240));
117    jScrollPane2.setViewportView(jTextArea1);
118
119    jLabel4.setText("Enter Your address if You wish a copy of the e-mail (CC)");
120
121    jTextField1.addActionListener(new java.awt.event.ActionListener() {
122        public void actionPerformed(java.awt.event.ActionEvent evt) {
123            jTextField1ActionPerformed(evt);
124        }
125    });
126
127    buttonCancel.setText("Cancel");
128    buttonCancel.addActionListener(new java.awt.event.ActionListener() {
129        public void actionPerformed(java.awt.event.ActionEvent evt) {
130            buttonCancelActionPerformed(evt);
131        }
132    });
133
134    jPanel2.setPreferredSize(new java.awt.Dimension(215, 80));
135
136    javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
137    jPanel2.setLayout(jPanel2Layout);
138    jPanel2Layout.setHorizontalGroup(
139        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
140            .addComponent(labelHeader, javax.swing.GroupLayout.DEFAULT_SIZE, 215, Short.MAX_VALUE)
141    );
142    jPanel2Layout.setVerticalGroup(
143        jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
144            .addComponent(labelHeader, javax.swing.GroupLayout.DEFAULT_SIZE, 80, Short.MAX_VALUE)
145    );
146
147    javax.swing.GroupLayout panelButtonLayout = new javax.swing.GroupLayout(panelButton);
148    panelButton.setLayout(panelButtonLayout);
149    panelButtonLayout.setHorizontalGroup(
150        panelButtonLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
151            .addComponent(labelButton, javax.swing.GroupLayout.DEFAULT_SIZE, 215, Short.MAX_VALUE)
152    );
153    panelButtonLayout.setVerticalGroup(
154        panelButtonLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

154     .addComponent(labelButton, javax.swing.GroupLayout.DEFAULT_SIZE, 40, Short.MAX_VALUE)
155 );
156
157 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
158 getContentPane().setLayout(layout);
159 layout.setHorizontalGroup(
160     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
161     .addGroup(layout.createSequentialGroup()
162         .addGap(30, 30, 30)
163         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
164             .addComponent(panelButton, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
165             .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 215,
Short.MAX_VALUE)
166             .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
167         .addGap(51, 51, 51)
168         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
169             .addComponent(jLabel3)
170             .addComponent(jLabel4)
171             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
172                 .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 375, Short.MAX_VALUE)
173                 .addGroup(layout.createSequentialGroup()
174                     .addComponent(jButton1)
175                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
176                     .addComponent(buttonCancel))
177                 .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING)))
178         .addGap(157, 157, 157))
179 );
180 layout.setVerticalGroup(
181     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
182     .addGroup(layout.createSequentialGroup()
183         .addGap(10, 10, 10)
184         .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
185         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
186         .addComponent(jLabel3)
187         .addGap(12, 12, 12)
188         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
189             .addComponent(jScrollPane2)
190             .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 182,
Short.MAX_VALUE))
191         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
192         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
193             .addGroup(layout.createSequentialGroup()
194                 .addComponent(jLabel4)
195                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
196                 .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
197             .addComponent(panelButton, 0, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
198         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
199         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
200             .addComponent(buttonCancel)
201             .addComponent(jButton1))
202         .addGap(37, 37, 37))
203 );
204

```

```

205     pack();
206 }// </editor-fold>
207
208 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
209     sendMail();
210 }
211
212 private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
213
214     if(mailSent)
215     {
216         resetFields();
217         this.dispose();
218     }// End if
219     else
220     {
221         int options = JOptionPane.showConfirmDialog(null,"Email Not Sent.. try again ?", "Email NOT sent" ,
JOptionPane.YES_NO_OPTION);
222         if(options==JOptionPane.YES_OPTION)
223         {
224             sendMail();
225
226         } // End inner if
227         else
228         {
229             this.dispose();
230         } // End inner else
231     }// End else
232
233 }
234
235 private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
236     // TODO add your handling code here:
237 }
238
239 /**
240  * Resets the text field(s)
241  */
242 private void resetFields()
243 {
244     messageSend.setText("");
245     jTextField1.setText("");
246 }//end resetFields method
247
248 /**
249  * Method for passing on the user input (email message text) to the Mail.java
250  * in the utility layer. The message has predefined information such as
251  * Subject, to-address and login information to the mail account @ gmail.
252  *
253  * For testing purpose, it is possible to enter another to-address - will be
254  * removed when the system is beeing delivered.
255  */
256 private void sendMail()
257 {
258     Mail sslObj = new Mail();
259     Mail.MESSAGE_SEND = messageSend.getText();
260     if(jTextField1.getText() == null)
261     {

```

```

262     Mail.MAIL_CC = "";
263 }// End if
264 else
265 {
266     Mail.MAIL_CC = jTextField1.getText();
267 }// End else
268 try
269 {
270     sslObj.mail();
271     JOptionPane.showMessageDialog(null, "Your email has been sendt");
272     mailSent = true;
273
274
275 }// End try
276 catch (Exception e)
277 {
278     mailSent = false;
279     // JOptionPane.showMessageDialog(null, "The following error has occoured: " + e.getMessage());
280 }// end catch
281 }// end sendMail method
282
283 /**
284  * Displays a message dialog box if mail was send succesfully
285  */
286 public void showOKdialog()
287 {
288     JOptionPane.showMessageDialog(null, "Your email has been sendt");
289 }// End showOKdialog
290
291 /**
292  * Scale an image to 215 * 80 for presentation
293  * @param src image to scale
294  * @return new scaled ImageIcon
295  */
296 private ImageIcon scale(Image src)
297 {
298     int w = 215;
299     int h = 80;
300     int type = BufferedImage.TYPE_INT_RGB;
301     BufferedImage dst = new BufferedImage(w, h, type);
302     Graphics2D g2 = dst.createGraphics();
303     g2.drawImage(src, 0, 0, w, h, this);
304     g2.dispose();
305     return new ImageIcon(dst);
306 }
307 /**
308  * Scale an image to 215 * 40 for presentation
309  * @param src image to scale
310  * @return new scaled ImageIcon
311  */
312 private ImageIcon scale2(Image src)
313 {
314     int w = 215;
315     int h = 40;
316     int type = BufferedImage.TYPE_INT_RGB;
317     BufferedImage dst = new BufferedImage(w, h, type);
318     Graphics2D g2 = dst.createGraphics();
319     g2.drawImage(src, 0, 0, w, h, this);

```



```

320     g2.dispose();
321     return new ImageIcon(dst);
322 }
323
324
325 // Variables declaration - do not modify
326 private javax.swing.JButton buttonCancel;
327 private javax.swing.JButton jButton1;
328 private javax.swing.JLabel jLabel3;
329 private javax.swing.JLabel jLabel4;
330 private javax.swing.JPanel jPanel2;
331 private javax.swing.JScrollPane jScrollPane1;
332 private javax.swing.JScrollPane jScrollPane2;
333 private javax.swing.JTextArea jTextArea1;
334 private javax.swing.JTextField jTextField1;
335 private javax.swing.JLabel labelButton;
336 private javax.swing.JLabel labelHeader;
337 private javax.swing.JTextPane messageSend;
338 private javax.swing.JPanel panelButton;
339 // End of variables declaration
340
341 }

```

ManageSpreadsheetView

```

1 /*
2  * This class is to show a spreadsheet to the user
3  * it takes a JTable in the parameters to built up the userdefined spreadsheet
4  * it also takes vehicleID - to know witch vehicle the spreadsheet belongs to
5  * it also takes workStationName - to know witch workstation the spreadsheet belongs to
6  * String resultName - Is the resultname set by the user
7  * String mechanicComment - Comments from the user on the spreadsheet
8  * String customerComment - Comments from the user on the spreadsheet for the customer
9  */
10
11 package tweakmc.view;
12
13 import java.awt.BorderLayout;
14 import java.awt.Color;
15 import java.awt.Component;
16 import java.awt.Container;
17 import java.awt.Dimension;
18 import java.awt.FlowLayout;
19 import java.awt.Font;
20 import java.awt.GridLayout;
21 import java.awt.Point;
22 import java.awt.Toolkit;
23 import java.awt.event.ActionEvent;
24 import java.awt.event.ActionListener;
25 import java.awt.event.FocusAdapter;
26 import java.awt.event.FocusEvent;
27 import java.awt.event.FocusListener;
28 import java.text.SimpleDateFormat;
29 import java.util.Calendar;
30 import java.util.Date;
31 import java.util.GregorianCalendar;
32 import javax.swing.BorderFactory;

```

```

33 import javax.swing.JButton;
34 import javax.swing.JComboBox;
35 import javax.swing.JFrame;
36 import javax.swing.JLabel;
37 import javax.swing.JOptionPane;
38 import javax.swing.JPanel;
39 import javax.swing.JScrollPane;
40 import javax.swing.JTable;
41 import javax.swing.JTextField;
42 import javax.swing.JTextPane;
43 import javax.swing.UIManager;
44 import javax.swing.UIManager.LookAndFeelInfo;
45 import javax.swing.border.BevelBorder;
46 import javax.swing.table.TableCellEditor;
47 import tweakmc.control.ManageResultController;
48 import tweakmc.utility.FileWriterException;
49
50 /**
51  *
52  * @author clausPallisgaardBeck
53  */
54 public class ManageSpreadsheetView
55 {
56     //Instance variables
57     private JFrame tableFrame;
58     private JTable actTable;
59     private JTextPane mechanicCommentTextArea, customerCommentTextArea;
60     private JTextField spreadsheetNameTField;
61     private Toolkit tKit = Toolkit.getDefaultToolkit();
62     private Dimension actScreenSize;
63     private int actScreenResolution;
64     private Component co;
65
66     private String vehicleID, workStationName, mechanicComment, customerComment;
67
68     //!!!!Remember to correct when order working!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
69     private String orderID = "C1";
70
71
72     //Final variables
73     private final Font HEADER = new Font("Tahoma", Font.BOLD, 14);
74     private final String SPREADSHEET = "Spreadsheet";
75
76     /**
77      * Present actual table
78      * @param actTable to present
79      * @param p the point from the cursor there the frame shoul build
80      * @param screenSize
81      */
82
83     public ManageSpreadsheetView(JTable actTable, String vehicleID, String workStationName)
84     {
85         this.actTable = actTable;
86         this.vehicleID = vehicleID;
87         this.workStationName = workStationName;
88
89         try
90         {

```

```

91     for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
92     {
93         if ("Nimbus".equals(info.getName()))
94         {
95             UIManager.setLookAndFeel(info.getClassName());
96             break;
97         } //end if
98     } //end for
99 } // end try
100
101 catch (Exception e)
102 {
103     FileWriterException.writeLogFile(ManageSpreadsheetView1.class.getName() + " / constructor/ " +
e.getMessage());
104 } //end catch
105
106 finally
107 {
108     actScreenSize = tKit.getScreenSize();
109     actScreenResolution = tKit.getScreenResolution();
110     initComponents();
111 } //end finally
112 } //end constructor
113
114 public ManageSpreadsheetView(JTable actTable, String vehicleID, String workStationName, String resultName,
String mechanicComment, String customerComment)
115 {
116     this.actTable = actTable;
117     this.vehicleID = vehicleID;
118     this.workStationName = workStationName;
119
120     try
121     {
122         for (LookAndFeelInfo info : UIManager.getInstalledLookAndFeels())
123         {
124             if ("Nimbus".equals(info.getName()))
125             {
126                 UIManager.setLookAndFeel(info.getClassName());
127                 break;
128             } //end if
129         } //end for
130     } // end try
131
132     catch (Exception e)
133     {
134         FileWriterException.writeLogFile(ManageSpreadsheetView1.class.getName() + " / constructor/ " +
e.getMessage());
135     } //end catch
136
137     finally
138     {
139         actScreenSize = tKit.getScreenSize();
140         actScreenResolution = tKit.getScreenResolution();
141         initComponents();
142         mechanicCommentTextArea.setText(mechanicComment);
143         customerCommentTextArea.setText(customerComment);
144         spreadsheetNameTField.setText(resultName);
145

```

```

146     } //end finally
147 } //end constructor
148
149 /**
150  * Saves the actual table in a sambo File
151  */
152 private void saveTable()
153 {
154     if(checkFields())
155     {
156         boolean saved = new ManageResultController().newSpreadsheetResult(actTable,
157             mechanicCommentTextArea.getText(),
158             customerCommentTextArea.getText(),
159             spreadsheetNameTField.getText(), vehicleID,
160             workStationName, orderID);
161
162         if(saved)
163         {
164             JOptionPane.showMessageDialog(null, "Spreadsheet for vehicleID " +
165                 vehicleID + " on " + workStationName + " is saved!",
166                 "Spreadsheet saved", JOptionPane.INFORMATION_MESSAGE);
167             closeFrame();
168         } //end if
169
170         else
171         {
172             JOptionPane.showMessageDialog(null, "Spreadsheet for vehicleID " +
173                 vehicleID + " on " + workStationName + " is NOT saved! Try again",
174                 "Spreadsheet not saved", JOptionPane.WARNING_MESSAGE);
175             return;
176         } //end else
177     } //end if
178
179     else
180     {
181         spreadsheetNameTField.setText("Name must be given!");
182         spreadsheetNameTField.setForeground(Color.PINK);
183     } //end else
184
185 } //end method saveTable()
186
187 /**
188  * Close the frame with table there are saved
189  */
190 private void closeFrame()
191 {
192     tableFrame.dispose();
193 } //end method closeFrame()
194
195 /**
196  * Check if the name is set
197  * @return true if name is given else false
198  */
199 private boolean checkFields()
200 {
201     //headerSpreadsheetName.requestFocusInWindow();
202     if(spreadsheetNameTField.getText().equalsIgnoreCase("")) ||
203         spreadsheetNameTField.getText() == null ||

```

```

204         spreadsheetNameTField.getText().equalsIgnoreCase("Name must be given!"))
205     {
206         return false;
207     } //end if
208
209     else
210     {
211         return true;
212     } //end else
213 } //end method checkFields()
214
215
216 ///////////////////////////////////////////////////
217 ///////////////////////////////////////////////////GUI CODE STARTS HERE//////////////////////////////////////
218
219 /**
220  * Build frame with choosen JTable
221  */
222 private void initComponents()
223 {
224
225     tableFrame = new JFrame("ActTable");
226     tableFrame.setPreferredSize(new Dimension(actScreenSize.width/2,actScreenSize.height/2));
227
228
229     tableFrame.setUndecorated(true);
230     tableFrame.setLocation(new Point(20,10));
231
232     Container contentPane = tableFrame.getContentPane();
233     contentPane.setLayout(new BorderLayout());
234
235     //Create and fill north
236     JPanel northJPanel = new JPanel();
237     northJPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 2, 2));
238     JLabel northHeaderLabel = new JLabel("Spreadsheet for vehicleID: " + vehicleID + " on " +
workStationName);
239     northHeaderLabel.setFont(HEADER);
240     northJPanel.add(northHeaderLabel);
241
242     contentPane.add(northJPanel, BorderLayout.NORTH);
243
244     //Create and fill center
245     JPanel mainJPanel = new JPanel();
246     mainJPanel.setLayout(new BorderLayout());
247     mainJPanel.setBorder(BorderFactory.createBevelBorder(BevelBorder.RAISED));
248     mainJPanel.add(new JScrollPane(actTable), BorderLayout.CENTER);
249     actTable.addFocusListener(new FocusAdapter()
250     {
251         public void focusGained(FocusEvent evt)
252         {
253             setCellEditorComponent();
254         } //end method focusGained()
255     }); //end focusListener
256
257     co = actTable.getEditorComponent();
258     setListenerOnCellEditorComponent();
259
260     contentPane.add(mainJPanel);

```

```

261
262 //Create and fill east
263 JPanel eastJPanel = new JPanel();
264 eastJPanel.setLayout(new GridLayout(4, 1));
265 eastJPanel.setBorder(BorderFactory.createBevelBorder(BevelBorder.RAISED));
266
267 JLabel headerMechanicComment = new JLabel("Mechanic Comments:");
268 JLabel headerCustomerComment = new JLabel("Customer Comments:");
269
270 mechanicCommentTextArea = new JTextPane();
271
272 customerCommentTextArea = new JTextPane();
273
274 JScrollPane mechanicCommentScrollPane = new JScrollPane(mechanicCommentTextArea);
275 mechanicCommentScrollPane.setPreferredSize(new Dimension((tableFrame.getWidth()/3),
(eastJPanel.getHeight()/3)));
276 JScrollPane customerCommentScrollPane = new JScrollPane(customerCommentTextArea);
277 customerCommentScrollPane.setPreferredSize(new Dimension((tableFrame.getWidth()/3),
(eastJPanel.getHeight()/3)));
278
279 eastJPanel.add(headerMechanicComment);
280 eastJPanel.add(mechanicCommentScrollPane);
281
282 eastJPanel.add(headerCustomerComment);
283 eastJPanel.add(customerCommentScrollPane);
284
285 contentPane.add(eastJPanel, BorderLayout.EAST);
286
287 //Create and fill south
288 JPanel southPanel = new JPanel();
289 southPanel.setBorder(BorderFactory.createBevelBorder(BevelBorder.RAISED));
290 southPanel.setLayout(new GridLayout(1, 4));
291
292 JLabel headerSpreadsheetName = new JLabel("Spreadsheet name: ");
293 spreadsheetNameTField = new JTextField();
294 spreadsheetNameTField.setPreferredSize(new Dimension(40, 25));
295 spreadsheetNameTField.addFocusListener(new FocusListener()
296 {
297
298     public void focusGained(FocusEvent e)
299     {
300 //        spreadsheetNameTField.setForeground(Color.WHITE);
301 //        spreadsheetNameTField.setText("");
302     } //end focus gained
303
304     public void focusLost(FocusEvent e) {} //end focus lost
305 }); //end focuslistener
306
307 JButton saveButton = new JButton("Save Table");
308 saveButton.addActionListener(new ActionListener()
309 {
310     @Override
311     public void actionPerformed(ActionEvent e)
312     {
313         saveTable();
314     } //end method actionPerformed
315 }); //end actionListener
316

```

```

317 JButton cancelButton = new JButton("Cancel");
318 cancelButton.addActionListener(new ActionListener()
319 {
320     @Override
321     public void actionPerformed(ActionEvent e)
322     {
323         int answer = JOptionPane.showConfirmDialog(null,
324             "Do you want't cancel and close this window?",
325             "Cancel and close window", JOptionPane.WARNING_MESSAGE);
326
327         if(answer == JOptionPane.YES_OPTION)
328         {
329             closeFrame();
330         }//end if
331         else
332         {
333             return;
334         }//end else
335
336     }//end method actionPerformed
337 });//end ActionListener
338
339 southPanel.add(headerSpreadsheetName);
340 southPanel.add(spreadsheetNameTField);
341 southPanel.add(saveButton);
342 southPanel.add(cancelButton);
343
344 contentPane.add(southPanel, BorderLayout.SOUTH);
345
346 //Build frame and set size
347 tableFrame.setVisible(true);
348 tableFrame.pack();
349
350 }//end method initComponents
351
352 /**
353  * Set the cell editor componenet for the focuslistener
354  */
355 private void setCellEditorComponent()
356 {
357     co = actTable.getEditorComponent();
358 }//end method setCellEditorComponent
359
360 /**
361  * Set listener on the component cell editor, to loose focus from field
362  * when, spreadsheet should be saved
363  */
364 private void setListenerOnCellEditorComponent()
365 {
366     if (co != null && !(co instanceof JComboBox))
367     {
368         co.addFocusListener(new java.awt.event.FocusAdapter()
369         {
370             public void focusLost(java.awt.event.FocusEvent evt)
371             {
372                 TableCellEditor tce = actTable.getCellEditor();
373                 if (tce != null)
374                 {

```

```

375         tce.stopCellEditing();
376     } //end if
377 } //end focusLost
378 }); //end focusAdapter
379 } //end if
380 } //end method setListenerOnCellEditorComponent()
381 } //end class ManageSpreadsheetView()

```

NameColumns

```

1 /*
2  * This class is to give columnnames to different kind of whiteboards
3  * It takes noOfColumns in parameters - To know how many columns the user must give names
4  */
5
6 /*
7  * NameColumns.java
8  *
9  * Created on 09-12-2010, 10:36:28
10 */
11
12 package tweakmc.view;
13
14 import java.awt.Point;
15 import java.util.Vector;
16 import javax.swing.JComboBox;
17 import javax.swing.JTable;
18 import javax.swing.table.TableModel;
19
20
21 /**
22  *
23  * @author NegoZiatoR
24  */
25 public class NameColumns extends javax.swing.JFrame
26 {
27     // Instance variables
28     private JComboBox comboBoxCylinderNo = new JComboBox(cylinderNo);
29     private Vector columnNumber;
30
31     // Static variables
32     private static WhiteBoardView whiteboardView;
33     private static TableModel chooseColumnnamesModel;
34     private static String[] userDefinedColumnNames;
35     private static String[] userDefinedCylinderNo;
36     private static NameColumns instance;
37     private static int noOfColumns;
38
39     // Constants
40     private final static String[] cylinderNo = {"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14",
41 "15", "16"};
42     private final static String[] nameColumn = {"Cylinder No."};
43     private final static String[] cylinderPos = {"Cylinder Pos."};
44
45     /** Creates new form NameColumns */
46     private NameColumns(int noOfColumns)
47     {

```



```

47     this.noOfColumns = noOfColumns;
48     chooseColumnnames = new JTable(2, noOfColumns);
49
50     initializeNoOfColumns(noOfColumns);
51     setUndecorated(true);
52     initComponents();
53     setVisible(true);
54     setDefaultCloseOperation(DISPOSE_ON_CLOSE);
55     setLocation(new Point(0,340));
56
57     comboBoxCylinderNo.setMaximumRowCount(17);
58
59
60 }
61
62 public static NameColumns getInstance(int noOfColumns)
63 {
64     if(instance == null)
65     {
66         instance = new NameColumns(noOfColumns);
67     }
68     return instance;
69 }
70
71 /** This method is called from within the constructor to
72  * initialize the form.
73  * WARNING: Do NOT modify this code. The content of this method is
74  * always regenerated by the Form Editor.
75  */
76 @SuppressWarnings("unchecked")
77 // <editor-fold defaultstate="collapsed" desc="Generated Code">
78 private void initComponents() {
79
80     mainPanel = new javax.swing.JPanel();
81     mainPanelScrollPane = new javax.swing.JScrollPane();
82     panelNameColumns = new javax.swing.JPanel();
83     labelNameColumns = new javax.swing.JLabel();
84     jSeparator1 = new javax.swing.JSeparator();
85     panelColumnsnumber = new javax.swing.JPanel();
86     jScrollPane1 = new javax.swing.JScrollPane();
87     chooseColumnnames = new javax.swing.JTable();
88     buttonOk = new javax.swing.JButton();
89     buttonCancel = new javax.swing.JButton();
90
91     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
92
93     labelNameColumns.setFont(new java.awt.Font("Tahoma", 0, 18)); // NOI18N
94     labelNameColumns.setText("Name Columns");
95
96     chooseColumnnames.setModel(chooseColumnnamesModel);
97     chooseColumnnames.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_ALL_COLUMNS);
98     chooseColumnnames.setColumnSelectionAllowed(true);
99     jScrollPane1.setViewportView(chooseColumnnames);
100
101     chooseColumnnames.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);

```

```

102     javax.swing.GroupLayout panelColumnsnumberLayout = new
javax.swing.GroupLayout(panelColumnsnumber);
103     panelColumnsnumber.setLayout(panelColumnsnumberLayout);
104     panelColumnsnumberLayout.setHorizontalGroup(
105         panelColumnsnumberLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
106         .addGroup(panelColumnsnumberLayout.createSequentialGroup())
107         .addContainerGap()
108         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 441, Short.MAX_VALUE)
109         .addContainerGap()
110     );
111     panelColumnsnumberLayout.setVerticalGroup(
112         panelColumnsnumberLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
113         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 176,
javax.swing.GroupLayout.PREFERRED_SIZE)
114     );
115
116     buttonOk.setText("OK");
117     buttonOk.addActionListener(new java.awt.event.ActionListener() {
118         public void actionPerformed(java.awt.event.ActionEvent evt) {
119             buttonOkActionPerformed(evt);
120         }
121     });
122
123     buttonCancel.setText("Cancel");
124     buttonCancel.addActionListener(new java.awt.event.ActionListener() {
125         public void actionPerformed(java.awt.event.ActionEvent evt) {
126             buttonCancelActionPerformed(evt);
127         }
128     });
129
130     javax.swing.GroupLayout panelNameColumnsLayout = new javax.swing.GroupLayout(panelNameColumns);
131     panelNameColumns.setLayout(panelNameColumnsLayout);
132     panelNameColumnsLayout.setHorizontalGroup(
133         panelNameColumnsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
134         .addGroup(panelNameColumnsLayout.createSequentialGroup())
135         .addContainerGap()
136         .addGroup(panelNameColumnsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
137             .addComponent(panelColumnsnumber, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
138             .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 461, Short.MAX_VALUE)
139             .addComponent(labelNameColumns, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 461, Short.MAX_VALUE))
140         .addContainerGap()
141         .addGroup(panelNameColumnsLayout.createSequentialGroup()
142             .addGap(20, 20, 20)
143             .addComponent(buttonOk, javax.swing.GroupLayout.PREFERRED_SIZE, 96,
javax.swing.GroupLayout.PREFERRED_SIZE)
144             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 242,
Short.MAX_VALUE)
145             .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 100,
javax.swing.GroupLayout.PREFERRED_SIZE)
146             .addGap(23, 23, 23))
147     );
148     panelNameColumnsLayout.setVerticalGroup(
149         panelNameColumnsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
150         .addGroup(panelNameColumnsLayout.createSequentialGroup()

```

```

151         .addContainerGap()
152         .addComponent(labelNameColumns, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE)
153         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
154         .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
155         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
156         .addComponent(panelColumnsnumber, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
157         .addGap(18, 18, 18)
158
.addGroup(panelNameColumnsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
159         .addComponent(buttonOk)
160         .addComponent(buttonCancel))
161         .addContainerGap(131, Short.MAX_VALUE))
162     );
163
164     mainPanelScrollPane.setViewportViewView(panelNameColumns);
165
166     javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
167     mainPanel.setLayout(mainPanelLayout);
168     mainPanelLayout.setHorizontalGroup(
169         mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
170         .addComponent(mainPanelScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 500,
Short.MAX_VALUE)
171     );
172     mainPanelLayout.setVerticalGroup(
173         mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
174         .addComponent(mainPanelScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 325,
javax.swing.GroupLayout.PREFERRED_SIZE)
175     );
176
177     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
178     getContentPane().setLayout(layout);
179     layout.setHorizontalGroup(
180         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
181         .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
182     );
183     layout.setVerticalGroup(
184         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
185         .addComponent(mainPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
186     );
187
188     pack();
189 } // </editor-fold>
190
191 private void buttonCancelActionPerformed(java.awt.event.ActionEvent evt) {
192     // TODO add your handling code here:
193     this.dispose();
194     instance = null;
195 }
196
197 private void buttonOkActionPerformed(java.awt.event.ActionEvent evt) {
198     // TODO add your handling code here:
199
200     userDefinedCylinderNo = getUserDefinedCylinderNo();

```

```

201     userDefinedColumnNames = getUserDefinedColumnNames();
202
203     // Set both column names and cylinder numbers
204     whiteboardView.setColumnNames(userDefinedColumnNames);
205     whiteboardView.setCylinderNo(userDefinedCylinderNo);
206
207     this.dispose();
208
209 }
210
211 /**
212  * @param args the command line arguments
213  */
214 public static void main(String args[])
215 {
216     java.awt.EventQueue.invokeLater(new Runnable()
217     {
218         private int noOfColumns;
219         public void run()
220         {
221             new NameColumns(this.noOfColumns).setVisible(true);
222         }
223     });
224 }
225
226 /**
227  * Initialize table for choosing column names
228  * @param noOfColumns number of column to present
229  */
230 private void initializeNoOfColumns(int noOfColumns)
231 {
232
233     chooseColumnnamesModel = chooseColumnnames.getModel();
234
235     for(int columnIndex = 1; columnIndex < noOfColumns; columnIndex++)
236     {
237         // TableColumn cylinderColumn = chooseColumnnames.getColumnModel().getColumn(columnIndex);
238         // cylinderColumn.setCellEditor(new DefaultCellEditor(comboBoxCylinderNo));
239         chooseColumnnamesModel.setValueAt("", 0, columnIndex);
240     }
241     for(int rowIndex = 0; rowIndex < 1; rowIndex++)
242     {
243         chooseColumnnamesModel.setValueAt(nameColumn[0], rowIndex, 0);
244     }
245     for(int rowIndex = 1; rowIndex == 1; rowIndex++)
246     {
247         chooseColumnnamesModel.setValueAt(cylinderPos[0], rowIndex, 0);
248     }
249     chooseColumnnames.setModel(chooseColumnnamesModel);
250
251
252 }
253
254 /**
255  * Return all the userdefined column names
256  * @return userdefined column names
257  */
258 public static String[] getUserDefinedColumnNames()

```

```

259 {
260     userDefinedColumnNames = new String[noOfColumns];
261     userDefinedColumnNames[0] = "CYLINDER NO.";
262     for(int columnIndex = 1; columnIndex < noOfColumns; columnIndex++)
263     {
264         String columnValue = String.valueOf(chooseColumnnamesModel.getValueAt(0, columnIndex));
265         userDefinedColumnNames[columnIndex] = columnValue;
266     }
267     return userDefinedColumnNames;
268 }
269
270 /**
271  * Return the userdefined cylinder number
272  * @return Array with cylinder numbers
273  */
274 public static String[] getUserDefinedCylinderNo()
275 {
276     userDefinedCylinderNo = new String[noOfColumns];
277     userDefinedCylinderNo[0] = "CYLINDER POS";
278     for(int columnIndex = 1; columnIndex < noOfColumns; columnIndex++)
279     {
280         String cylinderNoValue = String.valueOf(chooseColumnnamesModel.getValueAt(1, columnIndex));
281         userDefinedCylinderNo[columnIndex] = cylinderNoValue;
282     }
283     return userDefinedCylinderNo;
284 }
285
286 // Variables declaration - do not modify
287 private javax.swing.JButton buttonCancel;
288 private javax.swing.JButton buttonOk;
289 private javax.swing.JTable chooseColumnnames;
290 private javax.swing.JScrollPane jScrollPane1;
291 private javax.swing.JSeparator jSeparator1;
292 private javax.swing.JLabel labelNameColumns;
293 private javax.swing.JPanel mainPanel;
294 private javax.swing.JScrollPane mainPanelScrollPane;
295 private javax.swing.JPanel panelColumnsnumber;
296 private javax.swing.JPanel panelNameColumns;
297 // End of variables declaration
298
299 }

```

OrderGUI

```

1 /**
2  * This class is to create new orders and search, edit existing orders
3  *
4  * OrderGUI.java
5  *
6  * Created on 20-11-2010, 20:53:08
7  */
8
9 package tweakmc.view;
10
11 import java.awt.Color;
12 import java.io.IOException;
13 import java.util.ArrayList;

```

```

14 import javax.swing.JPanel;
15 import java.awt.event.ActionEvent;
16 import java.awt.event.ActionListener;
17 import java.text.SimpleDateFormat;
18 import java.util.Calendar;
19 import java.util.Date;
20 import javax.swing.JFrame;
21 import javax.swing.JOptionPane;
22 import javax.swing.JTextField;
23 import org.jdesktop.swing.JXDatePicker;
24 import org.jdesktop.swing.JXMonthView;
25 import org.jdesktop.swing.calendar.DateSelectionMode.SelectionMode;
26
27
28 import tweakmc.control.OrderController;
29 import tweakmc.model.Order;
30 import tweakmc.utility.CheckInput;
31 import tweakmc.utility.FileWriterException;
32 import tweakmc.utility.OrderPDF;
33
34 /**
35  * GUI for the order section in the mainGUI.
36  * @author Nyvang
37  */
38 public class OrderGUI extends javax.swing.JFrame {
39
40     //instance variables
41     private String errorMessage = "";
42     private OrderController oc = new OrderController();
43     private ArrayList<String> resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice;
44     private ArrayList<String> resultFoundOrderID;
45     private ArrayList<Order> resultFoundObjects;
46     private ArrayList<Order> resultStart;
47     private Order selectedOrder;
48     private String[] arrResultFoundOrderID;
49     private String[] arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice;
50     private String valueOfSearchField;
51     private String orderIDEditable;
52     private String onr;
53
54
55     //constant variables
56     private final String ORDERTYPE_EMPTY = "No ordertype has been selected";
57     private final String DESCRIPTION_EMPTY = "No order description has been entered";
58     private final String STARTDATE_EMPTY = "No startdate has been entered";
59     private final String DATE_MISMATCH = "End date must be greater than start date";
60
61
62
63     /** Creates new form OrderGUI */
64     public OrderGUI() {
65         initComponents();
66     }
67
68     /** This method is called from within the constructor to
69      * initialize the form.
70      * WARNING: Do NOT modify this code. The content of this method is
71      * always regenerated by the Form Editor.

```

```

72  */
73  @SuppressWarnings("unchecked")
74  // <editor-fold defaultstate="collapsed" desc="Generated Code">
75  private void initComponents() {
76
77      orderContentPanel = new javax.swing.JPanel();
78      orderTabbedPane = new javax.swing.JTabbedPane();
79      registerOrderScrollPane = new javax.swing.JScrollPane();
80      registerOrderPanel = new javax.swing.JPanel();
81      requiredFieldsField = new javax.swing.JLabel();
82      orderTypeLabel = new javax.swing.JLabel();
83      orderDescLabel = new javax.swing.JLabel();
84      orderDescScrollPane = new javax.swing.JScrollPane();
85      orderDescPane = new javax.swing.JTextPane();
86      sparepartsDescLabel = new javax.swing.JLabel();
87      sparepartsDescScrollPane = new javax.swing.JScrollPane();
88      sparepartsDescPane = new javax.swing.JTextPane();
89      startDateLabel = new javax.swing.JLabel();
90      expEndDateLabel = new javax.swing.JLabel();
91      expPriceLabel = new javax.swing.JLabel();
92      saveOrderButton = new javax.swing.JButton();
93      resetFieldsButton = new javax.swing.JButton();
94      expPriceField = new javax.swing.JTextField();
95      orderTypeComboBox = new javax.swing.JComboBox();
96      dkrField = new javax.swing.JLabel();
97      orderStatusLabel = new javax.swing.JLabel();
98      jSeparator1 = new javax.swing.JSeparator();
99      startDateChooserButton = new javax.swing.JButton();
100     endDateChooserButton = new javax.swing.JButton();
101     startDateTxtField = new javax.swing.JTextField();
102     endDateTxtField = new javax.swing.JTextField();
103     searchOrderScrollPane = new javax.swing.JScrollPane();
104     searchContentPanel = new javax.swing.JPanel();
105     searchEditPanel = new javax.swing.JPanel();
106     selectButton = new javax.swing.JButton();
107     editButton = new javax.swing.JButton();
108     jScrollPane1 = new javax.swing.JScrollPane();
109     foundOrdersJList = new javax.swing.JList();
110     criteriaPanel = new javax.swing.JPanel();
111     searchComboBox = new javax.swing.JComboBox();
112     noSearchResultsLabel = new javax.swing.JLabel();
113     jLabel2 = new javax.swing.JLabel();
114     searchButton = new javax.swing.JButton();
115     jButton1 = new javax.swing.JButton();
116     jPanel1 = new javax.swing.JPanel();
117     viewEditLabel = new javax.swing.JLabel();
118     sparepartsDescLabel1 = new javax.swing.JLabel();
119     jSeparator2 = new javax.swing.JSeparator();
120     orderStatusLabel1 = new javax.swing.JLabel();
121     expPriceLabel1 = new javax.swing.JLabel();
122     updateOrderButton = new javax.swing.JButton();
123     startDateLabel1 = new javax.swing.JLabel();
124     expEndDateLabel1 = new javax.swing.JLabel();
125     orderTypeLabel1 = new javax.swing.JLabel();
126     startDateField1 = new javax.swing.JTextField();
127     orderDescLabel1 = new javax.swing.JLabel();
128     expEndDateField1 = new javax.swing.JTextField();
129     resetFieldsButton1 = new javax.swing.JButton();

```



```

130 orderDescScrollPane1 = new javax.swing.JScrollPane();
131 orderDescPane1 = new javax.swing.JTextPane();
132 dkrField1 = new javax.swing.JLabel();
133 expPriceField1 = new javax.swing.JTextField();
134 subheaderOwnerLabel = new javax.swing.JLabel();
135 vehicleLabel = new javax.swing.JLabel();
136 vehicleTextField = new javax.swing.JTextField();
137 finalPriceLabel = new javax.swing.JLabel();
138 finalPriceField1 = new javax.swing.JTextField();
139 dkrField2 = new javax.swing.JLabel();
140 orderIdLabel = new javax.swing.JLabel();
141 jScrollPane4 = new javax.swing.JScrollPane();
142 sparepartsDescPane1 = new javax.swing.JTextPane();
143 orderTypeComboBox1 = new javax.swing.JComboBox();
144 chooseDateEditButton = new javax.swing.JButton();
145 chooseDateEditButton2 = new javax.swing.JButton();
146 weeklyTodoScrollPane = new javax.swing.JScrollPane();
147 jScrollPane2 = new javax.swing.JScrollPane();
148 todoContentPanel = new javax.swing.JPanel();
149 todoPanel = new javax.swing.JPanel();
150 generateButton = new javax.swing.JButton();
151 jLabel3 = new javax.swing.JLabel();
152 sevenDaysLabel = new javax.swing.JLabel();
153 statusLabel = new javax.swing.JLabel();
154 jScrollPane5 = new javax.swing.JScrollPane();
155 orderList = new javax.swing.JList();
156 jButton2 = new javax.swing.JButton();
157 printSelectedOrderButton = new javax.swing.JButton();
158
159 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
160
161 registerOrderPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter order information"));
162 registerOrderPanel.setPreferredSize(new java.awt.Dimension(800, 606));
163
164 requiredFieldsField.setFont(new java.awt.Font("Tahoma", 2, 10));
165 requiredFieldsField.setText("The fields marked with * is required to register");
166
167 orderTypeLabel.setText("Order type*");
168
169 orderDescLabel.setText("Order description*");
170
171 orderDescPane.setToolTipText("Place comments about the order description here");
172 orderDescPane.addFocusListener(new java.awt.event.FocusAdapter() {
173     public void focusGained(java.awt.event.FocusEvent evt) {
174         orderDescPaneFocusGained(evt);
175     }
176 });
177 orderDescScrollPane.setViewportView(orderDescPane);
178
179 sparepartsDescLabel.setText("Spareparts description");
180
181 sparepartsDescPane.setToolTipText("Place comments on possible spareparts here");
182 sparepartsDescScrollPane.setViewportView(sparepartsDescPane);
183
184 startDateLabel.setText("Start date*");
185
186 expEndDateLabel.setText("Expected end date");
187

```



```

188 expPriceLabel.setText("Price estimate:");
189
190 saveOrderButton.setText("Register");
191 saveOrderButton.addActionListener(new java.awt.event.ActionListener() {
192     public void actionPerformed(java.awt.event.ActionEvent evt) {
193         saveOrderButtonActionPerformed(evt);
194     }
195 });
196 saveOrderButton.addFocusListener(new java.awt.event.FocusAdapter() {
197     public void focusLost(java.awt.event.FocusEvent evt) {
198         saveOrderButtonFocusLost(evt);
199     }
200 });
201 saveOrderButton.addKeyListener(new java.awt.event.KeyAdapter() {
202     public void keyPressed(java.awt.event.KeyEvent evt) {
203         saveOrderButtonKeyPressed(evt);
204     }
205 });
206
207 resetFieldsButton.setText("Reset fields");
208 resetFieldsButton.addActionListener(new java.awt.event.ActionListener() {
209     public void actionPerformed(java.awt.event.ActionEvent evt) {
210         resetFieldsButtonActionPerformed(evt);
211     }
212 });
213
214 expPriceField.setToolTipText("Press F2 to open calculator");
215
216 orderTypeComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Please select",
"Service", "Service Contract (Leasing)", "Winter Storage", "Engine Optimizing", "Tire Change" }));
217
218 dkrField.setText("Dkr.");
219
220 startDateChooserButton.setText("Choose Date");
221 startDateChooserButton.addActionListener(new java.awt.event.ActionListener() {
222     public void actionPerformed(java.awt.event.ActionEvent evt) {
223         startDateChooserButtonActionPerformed(evt);
224     }
225 });
226
227 endDateChooserButton.setText("Choose Date");
228 endDateChooserButton.addActionListener(new java.awt.event.ActionListener() {
229     public void actionPerformed(java.awt.event.ActionEvent evt) {
230         endDateChooserButtonActionPerformed(evt);
231     }
232 });
233
234 startDateTxtField.setEditable(false);
235
236 endDateTxtField.setEditable(false);
237
238 javax.swing.GroupLayout registerOrderPanelLayout = new javax.swing.GroupLayout(registerOrderPanel);
239 registerOrderPanel.setLayout(registerOrderPanelLayout);
240 registerOrderPanelLayout.setHorizontalGroup(
241     registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
242     .addGroup(registerOrderPanelLayout.createSequentialGroup()
243         .addContainerGap()

```

```

244
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
245     .addGroup(registerOrderPanelLayout.createSequentialGroup()
246
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
247     .addComponent(requiredFieldsField, javax.swing.GroupLayout.Alignment.LEADING)
248     .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
registerOrderPanelLayout.createSequentialGroup()
249         .addComponent(orderTypeLabel)
250         .addGap(59, 59, 59)
251         .addComponent(orderTypeComboBox, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
252     .addContainerGap(512, Short.MAX_VALUE))
253     .addGroup(registerOrderPanelLayout.createSequentialGroup()
254
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
255     .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
256     .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
registerOrderPanelLayout.createSequentialGroup()
257
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
258     .addComponent(sparepartsDescLabel)
259     .addComponent(orderDescLabel)
260     .addComponent(expPriceLabel)
261     .addComponent(startDateLabel)
262     .addComponent(expEndDateLabel))
263     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
264
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
265     .addComponent(orderDescScrollPane)
266     .addComponent(sparepartsDescScrollPane)
267     .addGroup(registerOrderPanelLayout.createSequentialGroup()
268         .addComponent(expPriceField, javax.swing.GroupLayout.PREFERRED_SIZE, 181,
javax.swing.GroupLayout.PREFERRED_SIZE)
269         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
270         .addComponent(dkrField))
271     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
registerOrderPanelLayout.createSequentialGroup()
272
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
273     .addComponent(endDateTxtField, javax.swing.GroupLayout.DEFAULT_SIZE, 111,
Short.MAX_VALUE)
274     .addComponent(startDateTxtField))
275     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
276
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
277     .addComponent(startDateChooserButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
278     .addComponent(endDateChooserButton))))))
279     .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 336,
Short.MAX_VALUE))
280     .addGroup(registerOrderPanelLayout.createSequentialGroup()
281
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
282     .addComponent(orderStatusLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 334, Short.MAX_VALUE)

```

```

283         .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
registerOrderPanelLayout.createSequentialGroup()
284         .addComponent(saveOrderButton)
285         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
286         .addComponent(resetFieldsButton)))
287         .addGap(92, 92, 92)))
288         .addGap(360, 360, 360)))
289     );
290     registerOrderPanelLayout.setVerticalGroup(
291         registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
292         .addGroup(registerOrderPanelLayout.createSequentialGroup()
293         .addContainerGap()
294         .addComponent(requiredFieldsField)
295         .addGap(18, 18, 18)
296         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
297         .addComponent(orderTypeLabel)
298         .addComponent(orderTypeComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
299         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
300         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
301         .addComponent(startDateLabel)
302         .addComponent(startDateTxtField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
303         .addComponent(startDateChooserButton))
304         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
305         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
306         .addComponent(expEndDateLabel)
307         .addComponent(endDateTxtField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
308         .addComponent(endDateChooserButton))
309         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
310         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
311         .addComponent(orderDescLabel)
312         .addComponent(orderDescScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE))
313         .addGap(18, 18, 18)
314         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
315         .addComponent(sparepartsDescLabel)
316         .addComponent(sparepartsDescScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 83,
javax.swing.GroupLayout.PREFERRED_SIZE))
317         .addGap(18, 18, 18)
318         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
319         .addComponent(expPriceLabel)
320         .addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
321         .addComponent(expPriceField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
322         .addComponent(dkrField)))
323         .addGap(20, 20, 20)
324         .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
325         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

326         .addComponent(orderStatusLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 15,
javax.swing.GroupLayout.PREFERRED_SIZE)
327         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
328
.addGroup(registerOrderPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
329         .addComponent(saveOrderButton)
330         .addComponent(resetFieldsButton))
331         .addContainerGap(341, Short.MAX_VALUE))
332     );
333
334     registerOrderScrollPane.setViewportView(registerOrderPanel);
335
336     orderTabbedPane.addTab("Register Order", registerOrderScrollPane);
337
338     searchEditPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter search criterias"));
339
340     selectButton.setText("Select Order");
341     selectButton.setEnabled(false);
342     selectButton.addActionListener(new java.awt.event.ActionListener() {
343         public void actionPerformed(java.awt.event.ActionEvent evt) {
344             selectButtonActionPerformed(evt);
345         }
346     });
347
348     editButton.setText("Edit Order");
349     editButton.setEnabled(false);
350     editButton.addActionListener(new java.awt.event.ActionListener() {
351         public void actionPerformed(java.awt.event.ActionEvent evt) {
352             editButtonActionPerformed(evt);
353         }
354     });
355
356     foundOrdersJList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_INTERVAL_SELECTION);
357     jScrollPane1.setViewportView(foundOrdersJList);
358
359     criteriaPanel.setBorder(new
javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
360
361     searchComboBox.setEditable(true);
362     searchComboBox.addActionListener(new java.awt.event.ActionListener() {
363         public void actionPerformed(java.awt.event.ActionEvent evt) {
364             searchComboBoxActionPerformed(evt);
365         }
366     });
367
368     jLabel2.setFont(new java.awt.Font("Tahoma", 0, 9));
369     jLabel2.setForeground(new java.awt.Color(102, 102, 102));
370     jLabel2.setText("Searchable criterias: ID and vehicle ");
371
372     javax.swing.GroupLayout criteriaPanelLayout = new javax.swing.GroupLayout(criteriaPanel);
373     criteriaPanel.setLayout(criteriaPanelLayout);
374     criteriaPanelLayout.setHorizontalGroup(
375         criteriaPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
376         .addGroup(criteriaPanelLayout.createSequentialGroup()
377             .addContainerGap()
378             .addGroup(criteriaPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
379                 .addGroup(criteriaPanelLayout.createSequentialGroup()

```

```

380         .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
381         .addGap(50, 50, 50))
382
.addGroup(criteriaPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
383         .addComponent(noSearchResultsLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
384         .addComponent(searchComboBox, javax.swing.GroupLayout.Alignment.LEADING, 0, 139,
Short.MAX_VALUE)))
385     .addContainerGap()
386 );
387 criteriaPanelLayout.setVerticalGroup(
388     criteriaPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
389     .addGroup(criteriaPanelLayout.createSequentialGroup()
390         .addComponent(jLabel2)
391         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
392         .addComponent(searchComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
393         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 11,
Short.MAX_VALUE)
394         .addComponent(noSearchResultsLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
395         .addContainerGap()
396     );
397
398 searchButton.setText("Search");
399 searchButton.addActionListener(new java.awt.event.ActionListener() {
400     public void actionPerformed(java.awt.event.ActionEvent evt) {
401         searchButtonActionPerformed(evt);
402     }
403 });
404 searchButton.addKeyListener(new java.awt.event.KeyAdapter() {
405     public void keyPressed(java.awt.event.KeyEvent evt) {
406         enter(evt);
407     }
408 });
409
410 jButton1.setText("jButton1");
411 jButton1.addActionListener(new java.awt.event.ActionListener() {
412     public void actionPerformed(java.awt.event.ActionEvent evt) {
413         jButton1ActionPerformed(evt);
414     }
415 });
416
417 javax.swing.GroupLayout searchEditPanelLayout = new javax.swing.GroupLayout(searchEditPanel);
418 searchEditPanel.setLayout(searchEditPanelLayout);
419 searchEditPanelLayout.setHorizontalGroup(
420     searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
421     .addGroup(searchEditPanelLayout.createSequentialGroup()
422         .addContainerGap()
423
.addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
424         .addGroup(searchEditPanelLayout.createSequentialGroup()
425             .addComponent(searchButton)
426             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
427             .addComponent(jButton1))
428         .addComponent(criteriaPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 176,
javax.swing.GroupLayout.PREFERRED_SIZE))

```

```

429         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
430
431     .addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
432         .addGroup(searchEditPanelLayout.createSequentialGroup()
433             .addComponent(selectButton)
434             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
435             .addComponent(editButton))
436         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 542,
Short.MAX_VALUE))
437     .addContainerGap()
438 );
439 searchEditPanelLayout.setVerticalGroup(
440     searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
441     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
searchEditPanelLayout.createSequentialGroup()
442     .addContainerGap()
443     .addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
444         .addComponent(criteriaPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
445         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 103,
javax.swing.GroupLayout.PREFERRED_SIZE))
446     .addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
447         .addGroup(searchEditPanelLayout.createSequentialGroup()
448             .addGap(11, 11, 11)
449             .addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
450                 .addComponent(searchButton)
451                 .addComponent(jButton1)))
452         .addGroup(searchEditPanelLayout.createSequentialGroup()
453             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
454             .addGroup(searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
455                 .addComponent(selectButton)
456                 .addComponent(editButton))))
457     .addGap(114, 114, 114))
458 );
459 jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Order information"));
460
461 viewEditLabel.setFont(new java.awt.Font("Tahoma", 1, 12));
462
463 sparepartsDescLabel1.setText("Spareparts desc.");
464
465 expPriceLabel1.setText("Price estimate:");
466
467 updateOrderButton.setText("Update Order");
468 updateOrderButton.setEnabled(false);
469 updateOrderButton.addActionListener(new java.awt.event.ActionListener() {
470     public void actionPerformed(java.awt.event.ActionEvent evt) {
471         updateOrderButtonActionPerformed(evt);
472     }
473 });
474 updateOrderButton.addFocusListener(new java.awt.event.FocusAdapter() {
475     public void focusLost(java.awt.event.FocusEvent evt) {
476         updateOrderButtonFocusLost(evt);
477     }

```

```

478     });
479     updateOrderButton.addKeyListener(new java.awt.event.KeyAdapter() {
480         public void keyPressed(java.awt.event.KeyEvent evt) {
481             updateOrderButtonKeyPressed(evt);
482         }
483     });
484
485     startDateLabel1.setText("Start date*");
486
487     expEndDateLabel1.setText("Expected end date");
488
489     orderTypeLabel1.setText("Order type*");
490
491     startDateField1.setEditable(false);
492     startDateField1.addActionListener(new java.awt.event.ActionListener() {
493         public void actionPerformed(java.awt.event.ActionEvent evt) {
494             startDateField1ActionPerformed(evt);
495         }
496     });
497     startDateField1.addFocusListener(new java.awt.event.FocusAdapter() {
498         public void focusLost(java.awt.event.FocusEvent evt) {
499             startDateField1FocusLost(evt);
500         }
501     });
502
503     orderDescLabel1.setText("Order description*");
504
505     expEndDateField1.setEditable(false);
506
507     resetFieldsButton1.setText("Reset fields");
508     resetFieldsButton1.setEnabled(false);
509     resetFieldsButton1.addActionListener(new java.awt.event.ActionListener() {
510         public void actionPerformed(java.awt.event.ActionEvent evt) {
511             resetFieldsButton1ActionPerformed(evt);
512         }
513     });
514
515     orderDescPane1.setEditable(false);
516     orderDescPane1.setToolTipText("Place comments about the order description here");
517     orderDescScrollPane1.setViewportView(orderDescPane1);
518
519     dkrField1.setText("Dkr.");
520
521     expPriceField1.setEditable(false);
522     expPriceField1.setToolTipText("Press F2 to open calculator");
523
524     subheaderOwnerLabel.setFont(new java.awt.Font("Tahoma", 2, 9));
525     subheaderOwnerLabel.setForeground(new java.awt.Color(102, 102, 102));
526     subheaderOwnerLabel.setText("Change the specific value in the cooresponding text field and select save.
Values marked with * must allways be filled out.");
527
528     vehicleLabel.setText("Vehicle");
529
530     vehicleTextField.setEditable(false);
531
532     finalPriceLabel.setText("Final price");
533
534     finalPriceField1.setEditable(false);

```



```

535 dkrField2.setText("Dkr.");
536
537
538 orderIdLabel.setFont(new java.awt.Font("Tahoma", 1, 12));
539
540 jScrollPane4.setViewportViewView(sparepartsDescPane1);
541
542 orderTypeComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Please select",
"Service", "Service Contract (Leasing)", "Winter Storage", "Engine Optimizing", "Tire Change" }));
543 orderTypeComboBox1.addActionListener(new java.awt.event.ActionListener() {
544     public void actionPerformed(java.awt.event.ActionEvent evt) {
545         orderTypeComboBox1ActionPerformed(evt);
546     }
547 });
548
549 chooseDateEditButton.setText("Choose Date");
550 chooseDateEditButton.addActionListener(new java.awt.event.ActionListener() {
551     public void actionPerformed(java.awt.event.ActionEvent evt) {
552         chooseDateEditButtonActionPerformed(evt);
553     }
554 });
555
556 chooseDateEditButton2.setText("Choose Date");
557 chooseDateEditButton2.addActionListener(new java.awt.event.ActionListener() {
558     public void actionPerformed(java.awt.event.ActionEvent evt) {
559         chooseDateEditButton2ActionPerformed(evt);
560     }
561 });
562
563 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
564 jPanel1.setLayout(jPanel1Layout);
565 jPanel1Layout.setHorizontalGroup(
566     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
567     .addGroup(jPanel1Layout.createSequentialGroup()
568         .addContainerGap()
569         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
570             .addComponent(subheaderOwnerLabel)
571             .addGroup(jPanel1Layout.createSequentialGroup()
572                 .addComponent(viewEditLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 119,
javax.swing.GroupLayout.PREFERRED_SIZE)
573                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
574                 .addComponent(orderIdLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 191,
javax.swing.GroupLayout.PREFERRED_SIZE))
575             .addGroup(jPanel1Layout.createSequentialGroup()
576                 .addComponent(updateOrderButton)
577                 .addGap(27, 27, 27)
578                 .addComponent(orderStatusLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 297,
javax.swing.GroupLayout.PREFERRED_SIZE)
579                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
580                 .addComponent(resetFieldsButton1))
581             .addGroup(jPanel1Layout.createSequentialGroup()
582                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
583                     .addComponent(orderDescLabel1)
584                     .addComponent(expEndDateLabel1)
585                     .addComponent(startDateLabel1)
586                     .addComponent(orderTypeLabel1)
587                     .addComponent(sparepartsDescLabel1))
588                 .addGap(27, 27, 27)

```



```

589         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
590         .addComponent(jScrollPane4)
591         .addComponent(orderDescScrollPane1)
592         .addComponent(orderTypeComboBox1, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
593         .addGroup(jPanel1Layout.createSequentialGroup())
594
595     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
596     .addComponent(startDateField1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 279, Short.MAX_VALUE)
597     .addComponent(expEndDateField1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 289, javax.swing.GroupLayout.PREFERRED_SIZE))
598     .addGap(18, 18, 18)
599
600     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
601     .addComponent(chooseDateEditButton)
602     .addComponent(chooseDateEditButton2))))))
603     .addGroup(jPanel1Layout.createSequentialGroup())
604     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
605     .addComponent(jSeparator2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 500, Short.MAX_VALUE)
606     .addGroup(jPanel1Layout.createSequentialGroup())
607
608     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
609     .addComponent(vehicleLabel)
610     .addComponent(finalPriceLabel)
611     .addComponent(expPriceLabel1))
612     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 27,
Short.MAX_VALUE)
613
614     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
615     .addGroup(jPanel1Layout.createSequentialGroup())
616
617     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
618     .addComponent(finalPriceField1)
619     .addComponent(expPriceField1, javax.swing.GroupLayout.PREFERRED_SIZE, 361,
javax.swing.GroupLayout.PREFERRED_SIZE))
620     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
621
622     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
623     .addComponent(dkrField2)
624     .addComponent(dkrField1)))
625     .addComponent(vehicleTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 402,
javax.swing.GroupLayout.PREFERRED_SIZE)))
626     .addGap(51, 51, 51)))
627     .addContainerGap()
628 );
629
630     jPanel1Layout.setVerticalGroup(
631     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
632     .addGroup(jPanel1Layout.createSequentialGroup())
633     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
634     .addComponent(viewEditLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE)
635     .addComponent(orderIdLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 28,
javax.swing.GroupLayout.PREFERRED_SIZE))
636     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
637     .addComponent(subheaderOwnerLabel)

```

```

631         .addGap(18, 18, 18)
632         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
633             .addComponent(orderTypeLabel1)
634             .addComponent(orderTypeComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
635         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
636         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
637             .addComponent(startDateLabel1)
638             .addComponent(startDateField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
639             .addComponent(chooseDateEditButton))
640         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
641         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
642             .addComponent(expEndDateLabel1)
643             .addComponent(expEndDateField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
644             .addComponent(chooseDateEditButton2))
645         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
646         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
647             .addComponent(orderDescLabel1)
648             .addComponent(orderDescScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE))
649         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
650             .addGroup(jPanel1Layout.createSequentialGroup()
651                 .addGap(26, 26, 26)
652                 .addComponent(sparepartsDescLabel1))
653             .addGroup(jPanel1Layout.createSequentialGroup()
654                 .addGap(6, 6, 6)
655                 .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)))
656         .addGap(18, 18, 18)
657         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
658             .addGroup(jPanel1Layout.createSequentialGroup()
659                 .addComponent(expPriceLabel1)
660                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED, 23,
Short.MAX_VALUE)
661                 .addComponent(finalPriceLabel1))
662             .addGroup(jPanel1Layout.createSequentialGroup()
663                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
664                     .addComponent(expPriceField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
665                     .addComponent(dkrField1))
666                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
667                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
668                     .addComponent(finalPriceField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
669                     .addComponent(dkrField2))))
670             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
671             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
672                 .addComponent(vehicleLabel)
673                 .addComponent(vehicleTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
674             .addGap(14, 14, 14)
675             .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
676             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
677             .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

```

```

678         .addComponent(updateOrderButton)
679         .addComponent(orderStatusLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
680         .addComponent(resetFieldsButton1))
681     .addContainerGap()
682 );
683
684 javax.swing.GroupLayout searchContentPanelLayout = new javax.swing.GroupLayout(searchContentPanel);
685 searchContentPanel.setLayout(searchContentPanelLayout);
686 searchContentPanelLayout.setHorizontalGroup(
687     searchContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
688     .addGroup(searchContentPanelLayout.createSequentialGroup()
689         .addContainerGap()
690         .addGroup(searchContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
691             .addComponent(searchEditPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
692             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
693         .addContainerGap(130, Short.MAX_VALUE))
694     );
695 searchContentPanelLayout.setVerticalGroup(
696     searchContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
697     .addGroup(searchContentPanelLayout.createSequentialGroup()
698         .addContainerGap()
699         .addComponent(searchEditPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 183,
javax.swing.GroupLayout.PREFERRED_SIZE)
700         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
701         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
702         .addContainerGap(175, Short.MAX_VALUE))
703     );
704
705 searchOrderScrollPane.setViewportView(searchContentPanel);
706
707 orderTabbedPane.addTab("Search/Edit Order", searchOrderScrollPane);
708
709 todoPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Display todo"));
710
711 generateButton.setText("Generate list");
712 generateButton.addActionListener(new java.awt.event.ActionListener() {
713     public void actionPerformed(java.awt.event.ActionEvent evt) {
714         generateButtonActionPerformed(evt);
715     }
716 });
717
718 jLabel3.setFont(new java.awt.Font("Tahoma", 0, 10));
719 jLabel3.setForeground(new java.awt.Color(102, 102, 102));
720 jLabel3.setText("Showing orders starting and/or ending from now untill: ");
721
722 sevenDaysLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
723
724 orderList.setBorder(javax.swing.BorderFactory.createTitledBorder("Order list"));
725 jScrollPane5.setViewportView(orderList);
726
727 jButton2.setText("Generate PDF");
728 jButton2.addActionListener(new java.awt.event.ActionListener() {
729     public void actionPerformed(java.awt.event.ActionEvent evt) {

```

```

730         jButton1ActionPerformed(evt);
731     }
732 });
733
734 printSelectedOrderButton.setText("Print selected order");
735 printSelectedOrderButton.addActionListener(new java.awt.event.ActionListener() {
736     public void actionPerformed(java.awt.event.ActionEvent evt) {
737         printSelectedOrderButtonActionPerformed(evt);
738     }
739 });
740
741 javax.swing.GroupLayout todoPanelLayout = new javax.swing.GroupLayout(todoPanel);
742 todoPanel.setLayout(todoPanelLayout);
743 todoPanelLayout.setHorizontalGroup(
744     todoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
745         .addGroup(todoPanelLayout.createSequentialGroup()
746             .addContainerGap()
747             .addGroup(todoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
748                 .addComponent(jScrollPane5, javax.swing.GroupLayout.DEFAULT_SIZE, 678,
Short.MAX_VALUE)
749                 .addGroup(todoPanelLayout.createSequentialGroup()
750                     .addContainerGap()
751                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
752                     .addComponent(statusLabel))
753                 .addComponent(jLabel3)
754                 .addComponent(sevenDaysLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 188,
javax.swing.GroupLayout.PREFERRED_SIZE)
755                 .addGroup(todoPanelLayout.createSequentialGroup()
756                     .addContainerGap()
757                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
758                     .addComponent(printSelectedOrderButton)))
759             .addContainerGap()
760         );
761 todoPanelLayout.setVerticalGroup(
762     todoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
763         .addGroup(todoPanelLayout.createSequentialGroup()
764             .addContainerGap()
765             .addGroup(todoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
766                 .addContainerGap()
767                 .addComponent(statusLabel))
768             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
769             .addComponent(jLabel3)
770             .addGap(4, 4, 4)
771             .addComponent(sevenDaysLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)
772             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
773             .addComponent(jScrollPane5, javax.swing.GroupLayout.PREFERRED_SIZE, 201,
javax.swing.GroupLayout.PREFERRED_SIZE)
774             .addGap(18, 18, 18)
775             .addGroup(todoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
776                 .addContainerGap()
777                 .addComponent(printSelectedOrderButton))
778             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
779         );
780
781 javax.swing.GroupLayout todoContentPanelLayout = new javax.swing.GroupLayout(todoContentPanel);
782 todoContentPanel.setLayout(todoContentPanelLayout);
783 todoContentPanelLayout.setHorizontalGroup(

```

```

784     todoContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
785     .addGroup(todoContentPanelLayout.createSequentialGroup())
786     .addContainerGap()
787     .addComponent(todoPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
788     .addContainerGap(184, Short.MAX_VALUE))
789 );
790 todoContentPanelLayout.setVerticalGroup(
791     todoContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
792     .addGroup(todoContentPanelLayout.createSequentialGroup())
793     .addContainerGap()
794     .addComponent(todoPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
795     .addContainerGap(422, Short.MAX_VALUE))
796 );
797
798 jScrollPane2.setViewportViewView(todoContentPanel);
799
800 weeklyTodoScrollPane.setViewportViewView(jScrollPane2);
801
802 orderTabbedPane.addTab("Weekly todo-list", weeklyTodoScrollPane);
803
804 javax.swing.GroupLayout orderContentPanelLayout = new javax.swing.GroupLayout(orderContentPanel);
805 orderContentPanel.setLayout(orderContentPanelLayout);
806 orderContentPanelLayout.setHorizontalGroup(
807     orderContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
808     .addGroup(orderContentPanelLayout.createSequentialGroup())
809     .addContainerGap()
810     .addComponent(orderTabbedPane, javax.swing.GroupLayout.DEFAULT_SIZE, 815,
Short.MAX_VALUE)
811     .addContainerGap())
812 );
813 orderContentPanelLayout.setVerticalGroup(
814     orderContentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
815     .addGroup(orderContentPanelLayout.createSequentialGroup())
816     .addContainerGap()
817     .addComponent(orderTabbedPane, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
818     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
819 );
820
821 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
822 getContentPane().setLayout(layout);
823 layout.setHorizontalGroup(
824     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
825     .addGroup(layout.createSequentialGroup())
826     .addComponent(orderContentPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
827     .addContainerGap())
828 );
829 layout.setVerticalGroup(
830     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
831     .addGroup(layout.createSequentialGroup())
832     .addComponent(orderContentPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
833     .addContainerGap(33, Short.MAX_VALUE))
834 );
835

```

```

836     java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
837     setBounds((screenSize.width-861)/2, (screenSize.height-935)/2, 861, 935);
838 }// </editor-fold>
839
840 private void saveOrderButtonKeyPressed(java.awt.event.KeyEvent evt) {
841     // TODO add your handling code here:
842 }
843
844 private void saveOrderButtonFocusLost(java.awt.event.FocusEvent evt) {
845
846 }
847
848 private void saveOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
849     /**
850      * Calls the register order method, who sends the data through the layers, ending in the DB
851      */
852     registerOrder();
853     //after registering an order, a vehicle must be attached
854     attachVehicleToOrder();
855 }
856
857 private void resetFieldsButtonActionPerformed(java.awt.event.ActionEvent evt) {
858     resetTextFields();
859 }
860
861 private void updateOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
862     updateOrder();
863     resetTxtFieldsUpdateOrder();
864 }
865
866 private void updateOrderButtonFocusLost(java.awt.event.FocusEvent evt) {
867     // TODO add your handling code here:
868 }
869
870 private void updateOrderButtonKeyPressed(java.awt.event.KeyEvent evt) {
871     // TODO add your handling code here:
872 }
873
874 private void startDateField1ActionPerformed(java.awt.event.ActionEvent evt) {
875     // TODO add your handling code here:
876 }
877
878 private void startDateField1FocusLost(java.awt.event.FocusEvent evt) {
879     // TODO add your handling code here:
880 }
881
882 private void resetFieldsButton1ActionPerformed(java.awt.event.ActionEvent evt) {
883     resetTxtFieldsUpdateOrder();
884 }
885
886 private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
887     searchNow();
888
889 }
890
891
892 private void searchComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
893     setValueOfSearchField((String) searchComboBox.getSelectedItem());

```

```

894 }
895
896 private void startDateChooserButtonActionPerformed(java.awt.event.ActionEvent evt) {
897     startDateTxtField.setBackground(Color.WHITE);
898     pickDate(startDateTxtField);
899 }
900 }
901
902 private void endDateChooserButtonActionPerformed(java.awt.event.ActionEvent evt) {
903     endDateTxtField.setBackground(Color.WHITE);
904     pickDate(endDateTxtField);
905 }
906 }
907 }
908
909 private void orderDescPaneFocusGained(java.awt.event.FocusEvent evt) {
910     orderDescPane.setBackground(Color.WHITE);
911 }
912
913 private void selectButtonActionPerformed(java.awt.event.ActionEvent evt) {
914     selectOrder();
915 }
916 }
917 }
918
919 private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
920     enableEditing();
921 }
922
923 private void generateButtonActionPerformed(java.awt.event.ActionEvent evt) {
924     generateButton();
925 }
926
927 private void orderTypeComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
928     // TODO add your handling code here:
929 }
930
931 private void chooseDateEditButtonActionPerformed(java.awt.event.ActionEvent evt) {
932     startDateField1.setBackground(Color.WHITE);
933     pickDate(startDateField1);
934 }
935
936 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
937     SuperSearch.getInstance().run();
938 }
939
940 private void printSelectedOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
941     try
942     {
943         doPDF();
944     }//end try
945     catch (IOException ioe)
946     {
947         FileWriterException.writeLogFile("/doPDF/" + ioe);
948     }//end catch
949 }
950 }
951 }

```



```

952
953 private void chooseDateEditButton2ActionPerformed(java.awt.event.ActionEvent evt) {
954     expEndDateField1.setBackground(Color.WHITE);
955     pickDate(expPriceField1);
956 }
957
958 private void enter(java.awt.event.KeyEvent evt) {
959     // TODO add your handling code here:
960 }
961
962 private void selectOrder()
963 {
964     if(foundOrdersJList.getSelectedIndex() >=0)
965     {
966         int indexPositionOfSelectedOrder= foundOrdersJList.getSelectedIndex();
967
968         String orderID = arrResultFoundOrderID[indexPositionOfSelectedOrder];
969
970         int i = 0;
971         boolean found = false;
972         while (resultFoundObjects.size() > i && !found )
973         {
974             if(resultFoundObjects.get(i).getOrderID().equals(orderID))
975             {
976                 selectedOrder = resultFoundObjects.get(i);
977                 orderIDEditable = selectedOrder.getOrderID();
978                 found = true;
979             } //end if
980             else
981             {
982                 i++;
983             } //end else
984         } //end while
985         viewEditLabel.setText("View order:");
986         orderIdLabel.setText("" + orderIDEditable);
987         orderTypeComboBox1.setSelectedItem(selectedOrder.getOrderType());
988         orderDescPane1.setText(selectedOrder.getDescriptionOrder());
989         sparepartsDescPane1.setText(selectedOrder.getDescriptionSpareparts());
990         startDateField1.setText("" + selectedOrder.getStartDate());
991         expEndDateField1.setText("" + selectedOrder.getExpEndDate());
992         expPriceField1.setText(selectedOrder.getExpPrice());
993         finalPriceField1.setText(selectedOrder.getFinishPrice());
994         editButton.setEnabled(true);
995     } //end if
996 } //end selectOrder method
997
998 /**
999  * Acrtivated by the generateTodoLidt-button.
1000  * Excecutues the generate methods thru the layers
1001  */
1002 private void generateButton()
1003 {
1004     resetFields();
1005     //Set the arrays with the searchResults
1006     generateList();
1007     //If nothing is found, display a label to the user
1008     if(arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length==0)
1009     {

```



```

1010     statusLabel.setText("No searchresults");
1011     statusLabel.setForeground(Color.red);
1012     orderList.setModel(new javax.swing.AbstractListModel()
1013     {
1014         public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
1015         public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
1016     });
1017 } //end if
1018
1019 //If there's results present them.
1020 else
1021 {
1022     //Clear the actual JList and make a new one
1023     statusLabel.setText("");
1024     orderList.setModel(new javax.swing.AbstractListModel()
1025     {
1026         public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
1027         public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
1028     });
1029     //searchComboBox.setSelectedItem("");
1030     //selectButton.setEnabled(true);
1031 } //end else
1032
1033 } //end generateButton method
1034
1035 /**
1036  * Searches for an order according to the search criteria in the text field
1037  */
1038 private void searchNow()
1039 {
1040     //Get the text from the searchfield
1041     setValueOfSearchField((String) searchComboBox.getSelectedItem());
1042     String searchCriteria = valueOfSearchField;
1043     resetFields();
1044     //Set the arrays with the searchResults
1045     enterSearchDetails(searchCriteria);
1046     //If nothing is found, display a label to the user
1047     if(arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length==0)
1048     {
1049         noSearchResultsLabel.setText("No searchresults");
1050         noSearchResultsLabel.setForeground(Color.red);
1051         foundOrdersJList.setModel(new javax.swing.AbstractListModel()
1052         {
1053             public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
1054             public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
1055         });
1056     } //end if
1057
1058     //If there's results present them.
1059     else
1060     {
1061         //Clear the actual JList and make a new one

```

```

1062     noSearchResultsLabel.setText("");
1063     foundOrdersJList.setModel(new javax.swing.AbstractListModel()
1064     {
1065         public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
1066         public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
1067     });
1068     searchComboBox.setSelectedItem("");
1069     selectButton.setEnabled(true);
1070 } //end else
1071 } // end searchNow method
1072
1073 /**
1074  * Sets the textfields editable and informs the user that the selected
1075  * order is editable
1076  */
1077 private void enableEditing()
1078 {
1079     viewEditLabel.setText("Edit order:");
1080     orderIdLabel.setText("" + orderIDeditable);
1081     //orderTypeTextField1.setEditable(false);
1082     orderDescPane1.setEditable(true);
1083     sparepartsDescLabel1.setEnabled(true);
1084     startDateField1.setEditable(true);
1085     expEndDateField1.setEditable(true);
1086     expPriceField1.setEditable(true);
1087     finalPriceField1.setEditable(true);
1088     resetFieldsButton1.setEnabled(true);
1089     updateOrderButton.setEnabled(true);
1090 } //end enableEditing method
1091 /**
1092  * Generate the daily bigger picture (todo list), which is a list of orders expected
1093  * to be closed this week and orders starting this week.
1094  */
1095
1096 private void generateList()
1097 {
1098     resultFoundObjects = new ArrayList<Order>();
1099     resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice = new ArrayList<String>();
1100     resultFoundOrderID = new ArrayList<String>();
1101     resultFoundObjects = oc.generateTodo();
1102     SimpleDateFormat sdf = new SimpleDateFormat("EEE, d.' MMM yyyy");
1103     final long timeInMillis = System.currentTimeMillis();
1104     final long sevenDays = 604800000;
1105     final long nowPlusSevenDays = timeInMillis + sevenDays;
1106     String sevenDaysString = sdf.format(nowPlusSevenDays);
1107     sevenDaysLabel.setText(sevenDaysString);
1108
1109     int i = 0;
1110
1111     for(Order o : resultFoundObjects)
1112     {
1113         resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.add("<html>" + "<b>" + "Start
Date: " + "</b>" + o.getStartDate() + "<b>" + " End date: " + "</b>"
1114             + o.getExpEndDate() + "<b>" + " Order ID: " + "</b>" + o.getOrderID() + "</html>"); // implement
HTML tags in the String == nice!
1115         resultFoundOrderID.add(o.getOrderID());

```

```

1116     i++;
1117 }//end for-each
1118
1119 arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice =
resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.toArray(new String[0]);
1120 arrResultFoundOrderID = resultFoundOrderID.toArray(new String[0]);
1121 }//end generateList method
1122
1123 /**
1124  * Generates a pdf file with the info of the selected item in "orderList"
1125  * @throws IOException
1126  */
1127 private void doPDF() throws IOException
1128 {
1129
1130     if(orderList.getSelectedIndex() >=0)
1131     {
1132         int indexPositionOfSelectedOrder= orderList.getSelectedIndex();
1133
1134         String orderID = arrResultFoundOrderID[indexPositionOfSelectedOrder];
1135
1136         int i = 0;
1137         boolean found = false;
1138         while (resultFoundObjects.size() > i && !found )
1139         {
1140             if(resultFoundObjects.get(i).getOrderID().equals(orderID))
1141             {
1142                 selectedOrder = resultFoundObjects.get(i);
1143                 orderIDDeditable = selectedOrder.getOrderID();
1144                 found = true;
1145             }//end if
1146             else
1147             {
1148                 i++;
1149             }//end else
1150         }//end while
1151         //Send the variables to the OrderPDF class
1152         OrderPDF.orderID = orderID;
1153         OrderPDF.orderType = "" + orderTypeComboBox1.getSelectedIndex();
1154         OrderPDF.startDate = "" + selectedOrder.getStartDate();
1155         OrderPDF.endDate = "" + selectedOrder.getExpEndDate();
1156         OrderPDF.orderDesc = selectedOrder.getDescriptionSpareparts();
1157         OrderPDF.orderSpareDesc = selectedOrder.getDescriptionSpareparts();
1158         OrderPDF.priceEstamate = selectedOrder.getExpPrice();
1159         OrderPDF.vehicleID = "V20";
1160         if(OrderPDF.run())
1161         {
1162             // Confirm creation and prompt user to display the file
1163             Object[] options = { "Yes, please show me",
1164                 "Naa, just leave it", };
1165             int n = JOptionPane.showOptionDialog(null,
1166                 "Your PDF of order id: \"\" + orderID + "\" Has been generated.\n"
1167                 + "Do you wich to view it?",
1168                 "Open PDF file?",
1169                 JOptionPane.YES_NO_CANCEL_OPTION,
1170                 JOptionPane.QUESTION_MESSAGE,
1171                 null,
1172                 options,

```

```

1173         options[1]);
1174
1175         if(n == 0)
1176         { // execute the pdf file
1177             Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + "KJMC_OrderPreview.pdf");
1178         } //end if
1179         else
1180         {
1181
1182             } //end else
1183         } //end if
1184     } //end if
1185 } //end doPDF method
1186
1187 /**
1188  * method for resetting the fields after an order has been updated.
1189  */
1190 private void resetFields()
1191 {
1192
1193 }
1194
1195 /**
1196  * Gathers information from text fields, and checks input for errors, and updates the order information
1197  */
1198 private void updateOrder()
1199 {
1200     //Check if the startDate has a value
1201     if(startDateField1.getText().equals(""))
1202     {
1203         orderStatusLabel1.setBackground(Color.PINK);
1204         orderStatusLabel1.setText(STARTDATE_EMPTY);
1205     } //end if
1206     //Check weather the endDate is greater than the startDate
1207     else if(expEndDateField1 != null && CheckInput.compareDates(startDateField1.getText(),
expEndDateField1.getText()))
1208     {
1209         expEndDateField1.setBackground(Color.PINK);
1210         orderStatusLabel1.setText(DATE_MISMATCH);
1211     } //end else-if
1212     //Check if the orderDescriptionPane has a value
1213     else if(orderDescPane1.getText().equals(""))
1214     {
1215         orderDescPane1.setBackground(Color.PINK);
1216         orderStatusLabel1.setText(DESCRIPTION_EMPTY);
1217     } //end else-if
1218     // If the above checks out, save the order and confirm
1219     else
1220     {
1221         int startDate = Integer.parseInt(startDateField1.getText());
1222         int expEndDate = Integer.parseInt(expEndDateField1.getText());
1223         String orderID = orderIdLabel.getText();
1224         oc.updateOrder(orderID, setValueOfSearchField((String) orderTypeComboBox1.getSelectedItem()),
orderDescPane1.getText(), sparepartsDescPane1.getText(), startDate, expEndDate, expPriceField.getText(), null);
1225         orderStatusLabel1.setText("Order updated succesfully");
1226     } //end else
1227 } //end registerOrder method
1228

```

```

1229 /**
1230  * Lets the user open a calendar in a new frame, for easier to choose either start or finish date
1231  * @param textField textField the textfield where to set the date
1232  */
1233 private void pickDate(JTextField textField)
1234 {
1235     DatePickerClass dp = new DatePickerClass();
1236     dp.getInstance().pickDate(textField);
1237 } //end pickDate method
1238
1239 /**
1240  * Reset the textfields in the Register order tab and sets the bg colour to white
1241  */
1242 private void resetTextFields()
1243 {
1244
1245     orderDescPane.setBackground(Color.white);
1246     orderDescPane.setText("");
1247     sparepartsDescPane.setBackground(Color.white);
1248     sparepartsDescPane.setText("");
1249     startDateTxtField.setBackground(Color.white);
1250     startDateTxtField.setText("");
1251     endDateTxtField.setBackground(Color.white);
1252     endDateTxtField.setText("");
1253     expPriceField.setBackground(Color.white);
1254     expPriceField.setText("");
1255     orderStatusLabel.setText("");
1256     orderTypeComboBox1.setSelectedItem("Please select");
1257
1258 } //end resetTextFields method
1259
1260 /**
1261  * Reset the textfields in the View/Update order tab
1262  * Also "deselects" the current order
1263  */
1264 private void resetTxtFieldsUpdateOrder()
1265 {
1266     viewEditLabel.setText("Select/Edit order:");
1267     orderIdLabel.setText("\no order selected");
1268     //orderTypeTextField1.setEditable(false);
1269     orderDescPane1.setText("");
1270     sparepartsDescPane1.setText("");
1271     startDateField1.setText("");
1272     expEndDateField1.setText("");
1273     expPriceField1.setText("");
1274     finalPriceField1.setText("");
1275 } // end resetTxtFieldsUpdateOrder
1276
1277 /**
1278  * Helper method for adding the panel into the main GUI class
1279  * @return
1280  */
1281 public JPanel getPanel()
1282 {
1283     return orderContentPanel;
1284 } //end getPanel method
1285
1286 /**

```

```

1287  * Gathers information from text fields, and checks input for errors
1288  */
1289  private void registerOrder()
1290  {
1291      //check that the combobox's default value has been changed
1292      if(setValueOfSearchField((String) orderTypeComboBox.getSelectedItem()).equals("Please select"))
1293      {
1294          searchComboBox.setBackground(Color.PINK);
1295          orderStatusLabel.setText(ORDERTYPE_EMPTY);
1296      } //end if
1297      //Check if the startDate has a value
1298      if(startDateTxtField.getText().equals(""))
1299      {
1300          orderStatusLabel.setBackground(Color.PINK);
1301          startDateTxtField.setText(STARTDATE_EMPTY);
1302      } //end if
1303      //Check whether the endDate is greater than the startDate
1304      else if(endDateTxtField != null && CheckInput.compareDates(startDateTxtField.getText(),
endDateTxtField.getText()))
1305      {
1306          endDateTxtField.setBackground(Color.PINK);
1307          orderStatusLabel.setText(DATE_MISMATCH);
1308      } //end else-if
1309      //Check if the orderDescriptionPane has a value
1310      else if(orderDescPane.getText().equals(""))
1311      {
1312          orderDescPane.setBackground(Color.PINK);
1313          orderStatusLabel.setText(DESCRIPTION_EMPTY);
1314      } //end else-if
1315      // If the above checks out, save the order and confirm
1316      else
1317      {
1318          int startDate = Integer.parseInt(startDateTxtField.getText());
1319          int expEndDate = Integer.parseInt(endDateTxtField.getText());
1320          onr = oc.addOrder(setValueOfSearchField((String) orderTypeComboBox.getSelectedItem()),
orderDescPane.getText(), sparepartsDescPane.getText(), startDate, expEndDate, expPriceField.getText(), null);
1321          orderStatusLabel.setText("Order saved successfully");
1322          resetTextFields();
1323      } //end else
1324  } //end registerOrder method
1325
1326  /**
1327   * Attach a Vehicle to an order
1328   */
1329  private boolean attachVehicleToOrder()
1330  {
1331
1332      String vid = "";
1333      //vid = SuperSearch.getInstance().run();
1334      System.out.println(onr + " " + vid); // test statement for testing if the orderID has been returned.
1335      //oc.attachVehicle(onr, vid);
1336      return true;
1337  } // end attachVehicleToOrder method
1338
1339  /**
1340   * This method ask the controller-layer to perform a search for an owner
1341   * Search results are set from the results of the controller-layer as multiple
1342   * lists of Owner-objects, strings and ints.

```

```

1343  *
1344  * All variables except startDate, expEndDate, and orderID in every object in the
1345  * resultFoundObjects are put together as a string and inserted to an
1346  * arrayList(resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice) which will be converted
1347  * to an array.
1348  *
1349  * The same applies to the list of Integer (startDate, expEndDate, and orderID)
1350  * @param searchCriteria
1351  * @return void
1352  */
1353  public void enterSearchDetails(String searchCriteria)
1354  {
1355      //This ArrayList will hold the Strings of the search results
1356      resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice = new ArrayList<String>();
1357      //This ArrayList will hold the OrderID of the search results
1358      resultFoundOrderID = new ArrayList<String>();
1359      //This ArrayList will hold the Order-objects of the search results
1360      resultFoundObjects = new ArrayList<Order>();
1361
1362      resultFoundObjects = oc.searchOrder(searchCriteria);
1363
1364      int i = 0;
1365      for (Order o : resultFoundObjects)
1366      {
1367          //Strings and orderID goes to the respective ArrayLists
1368          resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.add("Order Number: \"" +
o.getOrderID()+ "\" " + "Order Type: \"" + o.getOrderType() + "\" Description: \"" + o.getDescriptionOrder());
1369          resultFoundOrderID.add(o.getOrderID());
1370          i++;
1371      } //end for each
1372
1373      //Here the ArrayList is converted to arrays for easier presentation
1374      arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice =
resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.toArray(new String[0]);
1375      arrResultFoundOrderID = resultFoundOrderID.toArray(new String[0]);
1376
1377  } //end enterSearchDetails method
1378
1379  private String setValueOfSearchField(String valueOfSearchField)
1380  {
1381      return this.valueOfSearchField = valueOfSearchField;
1382  } //end setValueOfSearchField
1383
1384
1385  // Variables declaration - do not modify
1386  private javax.swing.JButton chooseDateEditButton;
1387  private javax.swing.JButton chooseDateEditButton2;
1388  private javax.swing.JPanel criteriaPanel;
1389  private javax.swing.JLabel dkrField;
1390  private javax.swing.JLabel dkrField1;
1391  private javax.swing.JLabel dkrField2;
1392  private javax.swing.JButton editButton;
1393  private javax.swing.JButton endDateChooserButton;
1394  private javax.swing.JTextField endDateTxtField;
1395  private javax.swing.JTextField expEndDateField1;
1396  private javax.swing.JLabel expEndDateLabel;
1397  private javax.swing.JLabel expEndDateLabel1;
1398  private javax.swing.JTextField expPriceField;

```



```
1399 private javax.swing.JTextField expPriceField1;
1400 private javax.swing.JLabel expPriceLabel;
1401 private javax.swing.JLabel expPriceLabel1;
1402 private javax.swing.JTextField finalPriceField1;
1403 private javax.swing.JLabel finalPriceLabel;
1404 private javax.swing.JList foundOrdersJList;
1405 private javax.swing.JButton generateButton;
1406 private javax.swing.JButton jButton1;
1407 private javax.swing.JButton jButton2;
1408 private javax.swing.JLabel jLabel2;
1409 private javax.swing.JLabel jLabel3;
1410 private javax.swing.JPanel jPanel1;
1411 private javax.swing.JScrollPane jScrollPane1;
1412 private javax.swing.JScrollPane jScrollPane2;
1413 private javax.swing.JScrollPane jScrollPane4;
1414 private javax.swing.JScrollPane jScrollPane5;
1415 private javax.swing.JSeparator jSeparator1;
1416 private javax.swing.JSeparator jSeparator2;
1417 private javax.swing.JLabel noSearchResultsLabel;
1418 private javax.swing.JPanel orderContentPanel;
1419 private javax.swing.JLabel orderDescLabel;
1420 private javax.swing.JLabel orderDescLabel1;
1421 private javax.swing.JTextPane orderDescPane;
1422 private javax.swing.JTextPane orderDescPane1;
1423 private javax.swing.JScrollPane orderDescScrollPane;
1424 private javax.swing.JScrollPane orderDescScrollPane1;
1425 private javax.swing.JLabel orderIdLabel;
1426 private javax.swing.JList orderList;
1427 private javax.swing.JLabel orderStatusLabel;
1428 private javax.swing.JLabel orderStatusLabel1;
1429 private javax.swing.JTabbedPane orderTabbedPane;
1430 private javax.swing.JComboBox orderTypeComboBox;
1431 private javax.swing.JComboBox orderTypeComboBox1;
1432 private javax.swing.JLabel orderTypeLabel;
1433 private javax.swing.JLabel orderTypeLabel1;
1434 private javax.swing.JButton printSelectedOrderButton;
1435 private javax.swing.JPanel registerOrderPanel;
1436 private javax.swing.JScrollPane registerOrderScrollPane;
1437 private javax.swing.JLabel requiredFieldsField;
1438 private javax.swing.JButton resetFieldsButton;
1439 private javax.swing.JButton resetFieldsButton1;
1440 private javax.swing.JButton saveOrderButton;
1441 private javax.swing.JButton searchButton;
1442 private javax.swing.JComboBox searchComboBox;
1443 private javax.swing.JPanel searchContentPanel;
1444 private javax.swing.JPanel searchEditPanel;
1445 private javax.swing.JScrollPane searchOrderScrollPane;
1446 private javax.swing.JButton selectButton;
1447 private javax.swing.JLabel sevenDaysLabel;
1448 private javax.swing.JLabel sparepartsDescLabel;
1449 private javax.swing.JLabel sparepartsDescLabel1;
1450 private javax.swing.JTextPane sparepartsDescPane;
1451 private javax.swing.JTextPane sparepartsDescPane1;
1452 private javax.swing.JScrollPane sparepartsDescScrollPane;
1453 private javax.swing.JButton startDateChooserButton;
1454 private javax.swing.JTextField startDateField1;
1455 private javax.swing.JLabel startDateLabel;
1456 private javax.swing.JLabel startDateLabel1;
```



```

1457 private javax.swing.JTextField startDateTxtField;
1458 private javax.swing.JLabel statusLabel;
1459 private javax.swing.JLabel subheaderOwnerLabel;
1460 private javax.swing.JPanel todoContentPanel;
1461 private javax.swing.JPanel todoPanel;
1462 private javax.swing.JButton updateOrderButton;
1463 private javax.swing.JLabel vehicleLabel;
1464 private javax.swing.JTextField vehicleTextField;
1465 private javax.swing.JLabel viewEditLabel;
1466 private javax.swing.JScrollPane weeklyTodoScrollPane;
1467 // End of variables declaration
1468
1469 /**
1470  * An inner class are constructed for implementing of singleton
1471  * for the date picker frame.
1472  */
1473 class DatePickerClass
1474 {
1475     //instance variables
1476     private DatePickerClass instance;
1477     private JTextField textField;
1478
1479     /**
1480      * Empty private constructor for the class
1481      */
1482     private DatePickerClass()
1483     {
1484     } //end constructor
1485
1486     /**
1487      * Singleton method for the DatePickerClass
1488      * @return
1489      */
1489     public DatePickerClass getInstance()
1490     {
1491         if(instance == null)
1492         {
1493             instance = new DatePickerClass();
1494         } //end if
1495         return instance;
1496     } //end singleton method
1497
1498     /**
1499      * Lets the user open a calendar in a new frame, for easier to choose either start or finish date
1500      * @param textField textField the textfield where to set the date
1501      */
1502     private void pickDate(final JTextField textField)
1503     {
1504         final JFrame frame = new JFrame();
1505         final JXDatePicker picker = new JXDatePicker(new Date()); //initializes the datepicker function
1506         Calendar calendar = picker.getMonthView().getCalendar();
1507         picker.getMonthView().setLowerBound(calendar.getTime()); // sets todays date as default date when the
frame is set visible
1508         JXMonthView monthView = picker.getMonthView(); // initializes the month-view in the frame when set
visible
1509         monthView.setPreferredCols(1); // specifies the rows and columns of the date frame (e.g. number of months
to show in the frame)
1510         monthView.setPreferredRows(1); // same same
1511

```

```

1512     monthView.setSelectionMode(SelectionMode.SINGLE_INTERVAL_SELECTION); // specifies weather the
user can select a date interval (e.g. a full week at a time)
1513     //or just a single date
1514
1515     picker.addActionListener(new ActionListener()
1516     {
1517
1518         public void actionPerformed(ActionEvent e)
1519         {
1520             SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd"); // formats the date to a SQL-freindly
format
1521             textField.setText(sdf.format(picker.getDate())); // formats the date to a SQL-freindly format
1522             frame.dispose();
1523         } //end actionPerformed method
1524     } //end addActionlistener
1525 );
1526 frame.getContentPane().add(monthView); //as we are using a Swing-JFrame (and not a AWT-JFrame), we
need to call the method .add from the getContentPane
1527 frame.pack();
1528 frame.setVisible(true);
1529 } //end pickDate method
1530 } //end inner class DatePickerClass
1531 } //end Class OrderGUI

```

OwnerGUI

```

1 /**
2  * This class is to create new owners and search, edit existing owners
3  *
4  */
5
6 /*
7  * OwnerGUI.java
8  *
9  * Created on 28-10-2010, 14:51:31
10 */
11
12 package tweakmc.view;
13
14 import tweakmc.utility.GUIHelpUtil;
15 import java.awt.Color;
16 import java.awt.Component;
17 import java.awt.event.KeyEvent;
18 import java.io.File;
19 import java.io.FileNotFoundException;
20 import java.io.FileReader;
21 import java.util.ArrayList;
22 import java.util.Collections;
23 import java.util.HashMap;
24 import java.util.List;
25 import java.util.Scanner;
26 import java.util.Vector;
27 import javax.swing.DefaultComboBoxModel;
28 import javax.swing.JOptionPane;
29 import tweakmc.control.ManageOwnerController;
30 import tweakmc.utility.*;
31 import javax.swing.JPanel;

```

```

32 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
33 import tweakmc.dataaccess.AutoCompleteDAO;
34 import tweakmc.model.Owner;
35 import tweakmc.model.Vehicle;
36
37
38 /**
39 *
40 * @author NegoZiatoR
41 */
42 public class OwnerGUI extends javax.swing.JFrame {
43
44     // Instance variables
45     private ManageOwnerController moc = new ManageOwnerController();
46     private GUIDesign guiDesign;
47     private String errorMessage = "";
48     private File fileZip = new File("Zip.csv");
49     private File fileCity = new File("City.csv");
50     private HashMap zipCity;
51     private static Vector namesVector = new Vector();
52     private Integer options = null;
53     private boolean ownerSaved = false;
54     private List<String> actFoundsFirstnameLastNamePhoneNumber;
55     private List<String> actFoundsOwnerID;
56     private List<Owner> actFoundsObjects;
57     private Owner selectedOwner;
58     private String[] arrActFoundsOwnerID;
59     private String[] arrActFoundsFirstnameLastNamePhoneNumber;
60     private String valueOfSearchField;
61     private List<Vehicle> listWithVehiclesAttachedToOwner;
62     private boolean[] arrayWithVehiclesAttachedToOwner;
63     //end instance variables
64
65     // Constant Variables
66     private final String OK_OWNER_SAVED = "Owner saved";
67     private final String NO_OWNER_SELECTED = "No Owner is Selected";
68
69     private final String DENMARK = "Denmark";
70     private final String NAME_EMPTY = "Name is empty";
71     private final String NAME_INVALID = "Name is invalid";
72     private final String LASTNAME_EMPTY = "Lastname is empty";
73     private final String LASTNAME_INVALID = "Lastname is invalid";
74     private final String ADDRESS_INVALID = "Address is invalid";
75     private final String ADDRESS2_INVALID = "Address2 is invalid";
76     private final String ZIP_INVALID = "Zipcode is invalid";
77     private final String CITY_EMPTY = "City is empty";
78     private final String COUNTRY_INVALID = "Country is invalid";
79     private final String EMAIL_INVALID = "Email is invalid";
80     private final String PHONE_EMPTY = "Phone is empty";
81     private final String PHONE_INVALID = "Phone is invalid";
82     // end final instance variables
83
84
85     /** Creates new form OwnerGUI */
86     public OwnerGUI() {
87         buildAutoCompleteVectorList();
88         initComponents();
89         resetFields();

```

```

90     oPanel.setVisible(false);
91
92
93     zipCity = new HashMap();
94
95 }//end OwnerGUI constructor
96
97 @Override
98 public void enable()
99 {
100     oPanel.setVisible(true);
101 }
102
103 /**
104  * This method is called from within the constructor to
105  * initialize the form.
106  * WARNING: Do NOT modify this code. The content of this method is
107  * always regenerated by the Form Editor.
108  * @return
109  */
110 @SuppressWarnings("unchecked")
111 // <editor-fold defaultstate="collapsed" desc="Generated Code">
112 private void initComponents() {
113
114     oPanel = new javax.swing.JPanel();
115     searchOwnerTabPanel = new javax.swing.JTabbedPane();
116     registerOwnerPanel = new javax.swing.JPanel();
117     ownerInformationPanel = new javax.swing.JPanel();
118     nameLabel = new javax.swing.JLabel();
119     nameTextField = new javax.swing.JTextField();
120     lastNameLabel = new javax.swing.JLabel();
121     lastNameTextField = new javax.swing.JTextField();
122     addressLabel = new javax.swing.JLabel();
123     addressTextField = new javax.swing.JTextField();
124     addressLabel1 = new javax.swing.JLabel();
125     address2TextField = new javax.swing.JTextField();
126     zipAndCityLabel = new javax.swing.JLabel();
127     cityTextField = new javax.swing.JTextField();
128     zipTextField = new javax.swing.JTextField();
129     countryLabel = new javax.swing.JLabel();
130     countryTextField = new javax.swing.JTextField();
131     emailLabel = new javax.swing.JLabel();
132     emailTextField = new javax.swing.JTextField();
133     phoneLabel = new javax.swing.JLabel();
134     phoneTextField = new javax.swing.JTextField();
135     jSeparator1 = new javax.swing.JSeparator();
136     jButton1 = new javax.swing.JButton();
137     jButton2 = new javax.swing.JButton();
138     jResultLabelOwner = new javax.swing.JLabel();
139     jLabel1 = new javax.swing.JLabel();
140     jButton4 = new javax.swing.JButton();
141     searchEditOwnerPanel = new javax.swing.JPanel();
142     jScrollPane3 = new javax.swing.JScrollPane();
143     jPanel1 = new javax.swing.JPanel();
144     searchOwnerPanel = new javax.swing.JPanel();
145     searchOwnerButton = new javax.swing.JButton();
146     searchOwnerComboBox = new javax.swing.JComboBox();
147     searchOwnerComboBox.setEditable(true);

```

```

148     AutoCompleteDecorator.decorate(searchOwnerComboBox);
149     searchOwnerComboBox.setModel(new DefaultComboBoxModel(namesVector));
150     noSearchResultsLabel = new javax.swing.JLabel();
151     jButton3 = new javax.swing.JButton();
152     ownerListPanel = new javax.swing.JPanel();
153     selectOwnerButton = new javax.swing.JButton();
154     editOwnerButton = new javax.swing.JButton();
155     jScrollPane1 = new javax.swing.JScrollPane();
156     foundOwnersJList = new javax.swing.JList();
157     attachToVehicleButton = new javax.swing.JButton();
158     labelSelectOwner = new javax.swing.JLabel();
159     editOwnerPanel = new javax.swing.JPanel();
160     resetEditedOwnerFields = new javax.swing.JButton();
161     jResultLabelOwner1 = new javax.swing.JLabel();
162     saveEditedOwnerButton = new javax.swing.JButton();
163     phoneTextField1 = new javax.swing.JTextField();
164     emailTextField1 = new javax.swing.JTextField();
165     countryTextField1 = new javax.swing.JTextField();
166     cityTextField1 = new javax.swing.JTextField();
167     zipTextField1 = new javax.swing.JTextField();
168     address2TextField1 = new javax.swing.JTextField();
169     addressTextField1 = new javax.swing.JTextField();
170     lastNameTextField1 = new javax.swing.JTextField();
171     nameTextField1 = new javax.swing.JTextField();
172     nameLabel1 = new javax.swing.JLabel();
173     lastNameLabel1 = new javax.swing.JLabel();
174     addressLabel2 = new javax.swing.JLabel();
175     addressLabel3 = new javax.swing.JLabel();
176     zipAndCityLabel1 = new javax.swing.JLabel();
177     countryLabel1 = new javax.swing.JLabel();
178     emailLabel1 = new javax.swing.JLabel();
179     phoneLabel1 = new javax.swing.JLabel();
180     subheaderOwnerLabel = new javax.swing.JLabel();
181     editSelectLabel = new javax.swing.JLabel();
182     deleteOwnerButton = new javax.swing.JButton();
183     vehiclesPanel = new javax.swing.JPanel();
184     jScrollPane2 = new javax.swing.JScrollPane();
185     jTable1 = new javax.swing.JTable();
186     viewVehicleDataButton = new javax.swing.JButton();
187
188     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
189
190     oPanel.setEnabled(false);
191
192     searchOwnerTabPanel.addChangeListener(new javax.swing.event.ChangeListener() {
193         public void stateChanged(javax.swing.event.ChangeEvent evt) {
194             searchOwnerTabPanelStateChanged(evt);
195         }
196     });
197
198     registerOwnerPanel.addFocusListener(new java.awt.event.FocusAdapter() {
199         public void focusGained(java.awt.event.FocusEvent evt) {
200             registerOwnerPanelFocusGained(evt);
201         }
202     });
203     registerOwnerPanel.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
204         public void propertyChange(java.beans.PropertyChangeEvent evt) {
205             registerOwnerPanelPropertyChange(evt);

```

```

206     }
207 });
208
209 ownerInformationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter owner
information"));
210
211 nameLabel.setText("Firstname*");
212
213 nameTextField.addActionListener(new java.awt.event.ActionListener() {
214     public void actionPerformed(java.awt.event.ActionEvent evt) {
215         nameTextFieldActionPerformed(evt);
216     }
217 });
218 nameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
219     public void focusGained(java.awt.event.FocusEvent evt) {
220         nameTextFieldFocusGained(evt);
221     }
222     public void focusLost(java.awt.event.FocusEvent evt) {
223         nameTextFieldFocusLost(evt);
224     }
225 });
226
227 lastNameLabel.setText("Lastname*");
228
229 lastNameTextField.addFocusListener(new java.awt.event.FocusAdapter() {
230     public void focusGained(java.awt.event.FocusEvent evt) {
231         lastNameTextFieldFocusGained(evt);
232     }
233     public void focusLost(java.awt.event.FocusEvent evt) {
234         lastNameTextFieldFocusLost(evt);
235     }
236 });
237
238 addressLabel.setText("Address");
239
240 addressTextField.addFocusListener(new java.awt.event.FocusAdapter() {
241     public void focusGained(java.awt.event.FocusEvent evt) {
242         addressTextFieldFocusGained(evt);
243     }
244     public void focusLost(java.awt.event.FocusEvent evt) {
245         addressTextFieldFocusLost(evt);
246     }
247 });
248
249 addressLabel1.setText("Address2");
250
251 address2TextField.addActionListener(new java.awt.event.ActionListener() {
252     public void actionPerformed(java.awt.event.ActionEvent evt) {
253         address2TextFieldActionPerformed(evt);
254     }
255 });
256 address2TextField.addFocusListener(new java.awt.event.FocusAdapter() {
257     public void focusGained(java.awt.event.FocusEvent evt) {
258         address2TextFieldFocusGained(evt);
259     }
260     public void focusLost(java.awt.event.FocusEvent evt) {
261         address2TextFieldFocusLost(evt);
262     }

```

```
263     });
264
265     zipAndCityLabel.setText("Zip & city");
266
267     cityTextField.setEditable(false);
268     cityTextField.addActionListener(new java.awt.event.ActionListener() {
269         public void actionPerformed(java.awt.event.ActionEvent evt) {
270             cityTextFieldActionPerformed(evt);
271         }
272     });
273     cityTextField.addFocusListener(new java.awt.event.FocusAdapter() {
274         public void focusGained(java.awt.event.FocusEvent evt) {
275             cityTextFieldFocusGained(evt);
276         }
277         public void focusLost(java.awt.event.FocusEvent evt) {
278             cityTextFieldFocusLost(evt);
279         }
280     });
281
282     zipTextField.addActionListener(new java.awt.event.ActionListener() {
283         public void actionPerformed(java.awt.event.ActionEvent evt) {
284             zipTextFieldActionPerformed(evt);
285         }
286     });
287     zipTextField.addFocusListener(new java.awt.event.FocusAdapter() {
288         public void focusGained(java.awt.event.FocusEvent evt) {
289             zipTextFieldFocusGained(evt);
290         }
291         public void focusLost(java.awt.event.FocusEvent evt) {
292             zipTextFieldFocusLost(evt);
293         }
294     });
295
296     countryLabel.setText("Country");
297
298     countryTextField.addFocusListener(new java.awt.event.FocusAdapter() {
299         public void focusGained(java.awt.event.FocusEvent evt) {
300             countryTextFieldFocusGained(evt);
301         }
302         public void focusLost(java.awt.event.FocusEvent evt) {
303             countryTextFieldFocusLost(evt);
304         }
305     });
306
307     emailLabel.setText("E-mail");
308
309     emailTextField.addFocusListener(new java.awt.event.FocusAdapter() {
310         public void focusGained(java.awt.event.FocusEvent evt) {
311             emailTextFieldFocusGained(evt);
312         }
313         public void focusLost(java.awt.event.FocusEvent evt) {
314             emailTextFieldFocusLost(evt);
315         }
316     });
317
318     phoneLabel.setText("Phone*");
319
320     phoneTextField.addActionListener(new java.awt.event.ActionListener() {
```

```

321     public void actionPerformed(java.awt.event.ActionEvent evt) {
322         phoneTextFieldActionPerformed(evt);
323     }
324 });
325 phoneTextField.addFocusListener(new java.awt.event.FocusAdapter() {
326     public void focusGained(java.awt.event.FocusEvent evt) {
327         phoneTextFieldFocusGained(evt);
328     }
329     public void focusLost(java.awt.event.FocusEvent evt) {
330         phoneTextFieldFocusLost(evt);
331     }
332 });
333
334 jButton1.setText("Save");
335 jButton1.addActionListener(new java.awt.event.ActionListener() {
336     public void actionPerformed(java.awt.event.ActionEvent evt) {
337         jButton1ActionPerformed(evt);
338     }
339 });
340 jButton1.addKeyListener(new java.awt.event.KeyAdapter() {
341     public void keyPressed(java.awt.event.KeyEvent evt) {
342         jButton1KeyPressed(evt);
343     }
344 });
345
346 jButton2.setText("Reset fields");
347 jButton2.addActionListener(new java.awt.event.ActionListener() {
348     public void actionPerformed(java.awt.event.ActionEvent evt) {
349         jButton2ActionPerformed(evt);
350     }
351 });
352
353 jResultLabelOwner.setText(" ");
354
355 jLabel1.setFont(new java.awt.Font("Tahoma", 2, 10));
356 jLabel1.setText("The fields marked with * is required to register");
357
358 jButton4.setText("jButton4");
359 jButton4.addActionListener(new java.awt.event.ActionListener() {
360     public void actionPerformed(java.awt.event.ActionEvent evt) {
361         jButton4ActionPerformed(evt);
362     }
363 });
364
365 javax.swing.GroupLayout ownerInformationPanelLayout = new
javax.swing.GroupLayout(ownerInformationPanel);
366 ownerInformationPanel.setLayout(ownerInformationPanelLayout);
367 ownerInformationPanelLayout.setHorizontalGroup(
368     ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
369     .addGroup(ownerInformationPanelLayout.createSequentialGroup()
370         .addGap(40, 40, 40)
371         .addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
372             .addComponent(jResultLabelOwner, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
373             .addComponent(countryLabel, javax.swing.GroupLayout.Alignment.LEADING)
374             .addComponent(emailLabel, javax.swing.GroupLayout.Alignment.LEADING)
375             .addComponent(phoneLabel, javax.swing.GroupLayout.Alignment.LEADING)

```



```

376         .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING)
377         .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.LEADING)
378         .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
ownerInformationPanelLayout.createSequentialGroup())
379         .addComponent(jButton1)
380         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
381         .addComponent(jButton4)
382         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
383         .addComponent(jButton2))
384         .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
ownerInformationPanelLayout.createSequentialGroup())
385     .addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
386         .addComponent(nameLabel)
387         .addComponent(lastNameLabel)
388         .addComponent(addressLabel)
389         .addComponent(addressLabel1)
390         .addComponent(zipAndCityLabel))
391     .addGap(54, 54, 54)
392
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
393     .addComponent(address2TextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
394     .addComponent(addressTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
395     .addComponent(lastNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
396     .addComponent(nameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
397     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
ownerInformationPanelLayout.createSequentialGroup())
398     .addComponent(zipTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
399     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
400     .addComponent(cityTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 153,
javax.swing.GroupLayout.PREFERRED_SIZE))
401     .addComponent(countryTextField, javax.swing.GroupLayout.Alignment.TRAILING)
402     .addComponent(emailTextField, javax.swing.GroupLayout.Alignment.TRAILING)
403     .addComponent(phoneTextField, javax.swing.GroupLayout.Alignment.TRAILING))))
404     .addContainerGap()
405 );
406 ownerInformationPanelLayout.setVerticalGroup(
407     ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
408     .addGroup(ownerInformationPanelLayout.createSequentialGroup()
409         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
410         .addComponent(jLabel1)
411         .addGap(28, 28, 28)
412     .addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
413         .addComponent(nameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
414         .addComponent(nameLabel))
415         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
416     .addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
417         .addComponent(lastNameLabel)

```

```

418         .addComponent(lastNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
419         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
420
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
421         .addComponent(addressLabel)
422         .addComponent(addressTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
423         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
424
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
425         .addComponent(addressLabel1)
426         .addComponent(address2TextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
427         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
428
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
429         .addComponent(zipAndCityLabel)
430         .addComponent(cityTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
431         .addComponent(zipTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
432         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
433
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
434         .addComponent(countryLabel)
435         .addComponent(countryTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
436         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
437
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
438         .addComponent(emailLabel)
439         .addComponent(emailTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
440         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
441
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
442         .addComponent(phoneLabel)
443         .addComponent(phoneTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
444         .addGap(18, 18, 18)
445         .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
446         .addGap(19, 19, 19)
447
.addGroup(ownerInformationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
448         .addComponent(jButton1)
449         .addComponent(jButton2)
450         .addComponent(jButton4))
451         .addGap(37, 37, 37)
452         .addComponent(jResultLabelOwner))
453     );
454
455     javax.swing.GroupLayout registerOwnerPanelLayout = new javax.swing.GroupLayout(registerOwnerPanel);
456     registerOwnerPanel.setLayout(registerOwnerPanelLayout);
457     registerOwnerPanelLayout.setHorizontalGroup(
458         registerOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
459         .addGroup(registerOwnerPanelLayout.createSequentialGroup()

```

```

460         .addGap(18, 18, 18)
461         .addComponent(ownerInformationPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
462         .addContainerGap(805, Short.MAX_VALUE))
463     );
464     registerOwnerPanelLayout.setVerticalGroup(
465         registerOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
466         .addGroup(registerOwnerPanelLayout.createSequentialGroup()
467             .addGap(26, 26, 26)
468             .addComponent(ownerInformationPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
469             .addContainerGap(287, Short.MAX_VALUE))
470     );
471
472     searchOwnerTabPanel.addTab("Register Owner", registerOwnerPanel);
473
474     searchOwnerPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter search criterias"));
475
476     searchOwnerButton.setText("Search");
477     searchOwnerButton.addActionListener(new java.awt.event.ActionListener() {
478         public void actionPerformed(java.awt.event.ActionEvent evt) {
479             searchOwnerButtonActionPerformed(evt);
480         }
481     });
482
483     searchOwnerComboBox.setEditable(true);
484     searchOwnerComboBox.addItemListener(new java.awt.event.ItemListener() {
485         public void itemStateChanged(java.awt.event.ItemEvent evt) {
486             searchOwnerComboBoxItemStateChanged(evt);
487         }
488     });
489     searchOwnerComboBox.addActionListener(new java.awt.event.ActionListener() {
490         public void actionPerformed(java.awt.event.ActionEvent evt) {
491             searchOwnerComboBoxActionPerformed(evt);
492         }
493     });
494
495     jButton3.setText("jButton3");
496     jButton3.addActionListener(new java.awt.event.ActionListener() {
497         public void actionPerformed(java.awt.event.ActionEvent evt) {
498             jButton3ActionPerformed(evt);
499         }
500     });
501
502     javax.swing.GroupLayout searchOwnerPanelLayout = new javax.swing.GroupLayout(searchOwnerPanel);
503     searchOwnerPanel.setLayout(searchOwnerPanelLayout);
504     searchOwnerPanelLayout.setHorizontalGroup(
505         searchOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
506         .addGroup(searchOwnerPanelLayout.createSequentialGroup()
507             .addContainerGap()
508             .addGroup(searchOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
509                 .addComponent(searchOwnerComboBox, 0, 579, Short.MAX_VALUE)
510                 .addGroup(searchOwnerPanelLayout.createSequentialGroup()
511                     .addComponent(searchOwnerButton)
512                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
513                     .addComponent(jButton3)
514                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

515         .addComponent(noSearchResultsLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 364,
javax.swing.GroupLayout.PREFERRED_SIZE)))
516     .addContainerGap()
517 );
518 searchOwnerPanelLayout.setVerticalGroup(
519     searchOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
520     .addGroup(searchOwnerPanelLayout.createSequentialGroup()
521         .addGap(7, 7, 7)
522         .addComponent(searchOwnerComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
523         .addGap(18, 18, 18)
524     .addGroup(searchOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
525     .addGroup(searchOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
526         .addComponent(searchOwnerButton)
527         .addComponent(noSearchResultsLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
528         .addComponent(jButton3))
529     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
530 );
531
532 ownerListPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Owner List"));
533
534 selectOwnerButton.setText("Select Owner");
535 selectOwnerButton.addActionListener(new java.awt.event.ActionListener() {
536     public void actionPerformed(java.awt.event.ActionEvent evt) {
537         selectOwnerButtonActionPerformed(evt);
538     }
539 });
540
541 editOwnerButton.setText("Edit Owner");
542 editOwnerButton.setEnabled(false);
543 editOwnerButton.addActionListener(new java.awt.event.ActionListener() {
544     public void actionPerformed(java.awt.event.ActionEvent evt) {
545         editOwnerButtonActionPerformed(evt);
546     }
547 });
548
549 foundOwnersJList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_INTERVAL_SELECTION);
550 jScrollPane1.setViewportView(foundOwnersJList);
551
552 attachToVehicleButton.setText("Attach to Vehicle");
553 attachToVehicleButton.setEnabled(false);
554 attachToVehicleButton.addActionListener(new java.awt.event.ActionListener() {
555     public void actionPerformed(java.awt.event.ActionEvent evt) {
556         attachToVehicleButtonActionPerformed(evt);
557     }
558 });
559
560 javax.swing.GroupLayout ownerListPanelLayout = new javax.swing.GroupLayout(ownerListPanel);
561 ownerListPanel.setLayout(ownerListPanelLayout);
562 ownerListPanelLayout.setHorizontalGroup(
563     ownerListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
564     .addGroup(ownerListPanelLayout.createSequentialGroup()
565         .addContainerGap()
566     .addGroup(ownerListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

567         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 579,
Short.MAX_VALUE)
568         .addGroup(ownerLayoutPanel.createSequentialGroup())
569         .addComponent(selectOwnerButton)
570         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 278,
Short.MAX_VALUE)
571         .addComponent(attachToVehicleButton)
572         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
573         .addComponent(editOwnerButton))
574         .addComponent(labelSelectOwner, javax.swing.GroupLayout.PREFERRED_SIZE, 236,
javax.swing.GroupLayout.PREFERRED_SIZE))
575         .addContainerGap()
576     );
577     ownerLayoutPanel.setVerticalGroup(
578         ownerLayoutPanel.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
579         .addGroup(ownerLayoutPanel.createSequentialGroup())
580         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 269,
javax.swing.GroupLayout.PREFERRED_SIZE)
581         .addGap(18, 18, 18)
582     ).addGroup(ownerLayoutPanel.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
583         .addComponent(selectOwnerButton)
584         .addComponent(editOwnerButton)
585         .addComponent(attachToVehicleButton))
586         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
587         .addComponent(labelSelectOwner)
588         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
589     );
590
591     editOwnerPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Owner information"));
592
593     resetEditedOwnerFields.setText("Reset fields");
594     resetEditedOwnerFields.setEnabled(false);
595     resetEditedOwnerFields.addActionListener(new java.awt.event.ActionListener() {
596         public void actionPerformed(java.awt.event.ActionEvent evt) {
597             resetEditedOwnerFieldsActionPerformed(evt);
598         }
599     });
600
601     jResultLabelOwner1.setText(" ");
602
603     saveEditedOwnerButton.setText("Save");
604     saveEditedOwnerButton.setEnabled(false);
605     saveEditedOwnerButton.addActionListener(new java.awt.event.ActionListener() {
606         public void actionPerformed(java.awt.event.ActionEvent evt) {
607             saveEditedOwnerButtonActionPerformed(evt);
608         }
609     });
610
611     phoneTextField1.setBackground(new java.awt.Color(220, 220, 220));
612     phoneTextField1.setEditable(false);
613
614     emailTextField1.setBackground(new java.awt.Color(220, 220, 220));
615     emailTextField1.setEditable(false);
616
617     countryTextField1.setBackground(new java.awt.Color(220, 220, 220));
618     countryTextField1.setEditable(false);
619

```

```

620 cityTextField1.setBackground(new java.awt.Color(220, 220, 220));
621 cityTextField1.setEditable(false);
622
623 zipTextField1.setBackground(new java.awt.Color(220, 220, 220));
624 zipTextField1.setEditable(false);
625
626 address2TextField1.setBackground(new java.awt.Color(220, 220, 220));
627 address2TextField1.setEditable(false);
628
629 addressTextField1.setBackground(new java.awt.Color(220, 220, 220));
630 addressTextField1.setEditable(false);
631
632 lastNameTextField1.setBackground(new java.awt.Color(220, 220, 220));
633 lastNameTextField1.setEditable(false);
634
635 nameTextField1.setBackground(new java.awt.Color(220, 220, 220));
636 nameTextField1.setEditable(false);
637 nameTextField1.addActionListener(new java.awt.event.ActionListener() {
638     public void actionPerformed(java.awt.event.ActionEvent evt) {
639         nameTextField1ActionPerformed(evt);
640     }
641 });
642
643 nameLabel1.setText("Firstname*");
644
645 lastNameLabel1.setText("Lastname*");
646
647 addressLabel2.setText("Address");
648
649 addressLabel3.setText("Address2");
650
651 zipAndCityLabel1.setText("Zip & city");
652
653 countryLabel1.setText("Country");
654
655 emailLabel1.setText("E-mail");
656
657 phoneLabel1.setText("Phone*");
658
659 subheaderOwnerLabel.setFont(new java.awt.Font("Tahoma", 2, 10));
660 subheaderOwnerLabel.setText("Change the specific value in the cooresponding text field and select save");
661
662 editSelectLabel.setFont(new java.awt.Font("Tahoma", 1, 11));
663 editSelectLabel.setText("Edit / Select Owner");
664
665 deleteOwnerButton.setText("Delete Owner");
666 deleteOwnerButton.setEnabled(false);
667 deleteOwnerButton.addActionListener(new java.awt.event.ActionListener() {
668     public void actionPerformed(java.awt.event.ActionEvent evt) {
669         deleteOwnerButtonActionPerformed(evt);
670     }
671 });
672
673 javax.swing.GroupLayout editOwnerPanelLayout = new javax.swing.GroupLayout(editOwnerPanel);
674 editOwnerPanel.setLayout(editOwnerPanelLayout);
675 editOwnerPanelLayout.setHorizontalGroup(
676     editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
677     .addGroup(editOwnerPanelLayout.createSequentialGroup()

```



```

678         .addContainerGap()
679
680 .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
681         .addComponent(editSelectLabel)
682         .addComponent(countryLabel1)
683         .addComponent(emailLabel1)
684         .addComponent(phoneLabel1)
685         .addComponent(subheaderOwnerLabel)
686
687 .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
688         .addComponent(jResultLabelOwner1, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
689         .addGroup(editOwnerPanelLayout.createSequentialGroup()
690         .addComponent(saveEditedOwnerButton)
691         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 123,
Short.MAX_VALUE)
692         .addComponent(deleteOwnerButton, javax.swing.GroupLayout.PREFERRED_SIZE, 107,
javax.swing.GroupLayout.PREFERRED_SIZE)
693         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
694         .addComponent(resetEditedOwnerFields))
695         .addGroup(editOwnerPanelLayout.createSequentialGroup()
696         .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
697         .addComponent(nameLabel1)
698         .addComponent(lastNameLabel1)
699         .addComponent(addressLabel2)
700         .addComponent(addressLabel3)
701         .addComponent(zipAndCityLabel1))
702         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 130,
Short.MAX_VALUE)
703         .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
704         .addComponent(address2TextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
705         .addComponent(addressTextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
706         .addComponent(lastNameTextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
707         .addComponent(nameTextField1, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
708         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
editOwnerPanelLayout.createSequentialGroup()
709         .addComponent(zipTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
710         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
711         .addComponent(cityTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 153,
javax.swing.GroupLayout.PREFERRED_SIZE))
712         .addComponent(countryTextField1, javax.swing.GroupLayout.Alignment.TRAILING)
713         .addComponent(emailTextField1, javax.swing.GroupLayout.Alignment.TRAILING)
714         .addComponent(phoneTextField1, javax.swing.GroupLayout.Alignment.TRAILING))))))
715         .addContainerGap(75, Short.MAX_VALUE))
716     );
717     editOwnerPanelLayout.setVerticalGroup(
718     editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
719     .addGroup(editOwnerPanelLayout.createSequentialGroup()
720     .addGap(31, 31, 31)
721     .addComponent(editSelectLabel)
722     .addGap(71, 71, 71)

```

```

721         .addComponent(subheaderOwnerLabel)
722         .addGap(28, 28, 28)
723
724     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
725         .addComponent(nameTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
726             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
727         .addComponent(nameLabel1))
728     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
729
730     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
731         .addComponent(lastNameTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
732             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
733         .addComponent(lastNameLabel1))
734     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
735
736     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
737         .addComponent(addressTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
738             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
739         .addComponent(addressLabel2))
740     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
741
742     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
743         .addComponent(address2TextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
744             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
745         .addComponent(addressLabel3))
746     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
747
748     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
749         .addComponent(cityTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
750             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
751         .addComponent(zipTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
752             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
753         .addComponent(zipAndCityLabel1))
754     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
755
756     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
757         .addComponent(countryLabel1)
758         .addComponent(countryTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
759             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
760     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
761
762     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
763         .addComponent(emailLabel1)
764         .addComponent(emailTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
765             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
766     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
767
768     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
769         .addComponent(phoneLabel1)
770         .addComponent(phoneTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
771             javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
772
773     .addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
774         .addGroup(editOwnerPanelLayout.createSequentialGroup()
775             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
776             .addComponent(jResultLabelOwner1)
777             .addGap(18, 18, 18))

```



```

760         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
editOwnerPanelLayout.createSequentialGroup()
761         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 22,
Short.MAX_VALUE)
762
.addGroup(editOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
763         .addComponent(resetEditedOwnerFields)
764         .addComponent(saveEditedOwnerButton)
765         .addComponent(deleteOwnerButton))))
766     .addGap(27, 27, 27))
767 );
768
769 vehiclesPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Vehicles"));
770
771 jTable1.setModel(new javax.swing.table.DefaultTableModel(
772     new Object [][] {
773         {null, null, null, null},
774         {null, null, null, null},
775         {null, null, null, null},
776         {null, null, null, null}
777     },
778     new String [] {
779         "Brand", "Model", "Year", "Active"
780     }
781 ) {
782     Class[] types = new Class [] {
783         java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Boolean.class
784     };
785     boolean[] canEdit = new boolean [] {
786         false, false, false, true
787     };
788
789     public Class getColumnClass(int columnIndex) {
790         return types [columnIndex];
791     }
792
793     public boolean isCellEditable(int rowIndex, int columnIndex) {
794         return canEdit [columnIndex];
795     }
796 });
797 jScrollPane2.setViewportView(jTable1);
798
799 viewVehicleDataButton.setText("View vehicle data");
800
801 javax.swing.GroupLayout vehiclesPanelLayout = new javax.swing.GroupLayout(vehiclesPanel);
802 vehiclesPanel.setLayout(vehiclesPanelLayout);
803 vehiclesPanelLayout.setHorizontalGroup(
804     vehiclesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
805     .addGroup(vehiclesPanelLayout.createSequentialGroup()
806         .addContainerGap(41, Short.MAX_VALUE)
807         .addGroup(vehiclesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
808             .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 426,
javax.swing.GroupLayout.PREFERRED_SIZE)
809             .addComponent(viewVehicleDataButton)))
810 );
811 vehiclesPanelLayout.setVerticalGroup(
812     vehiclesPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
813     .addGroup(vehiclesPanelLayout.createSequentialGroup()

```

```

814         .addContainerGap()
815         .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
816         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
817         .addComponent(viewVehicleDataButton)
818         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
819     );
820
821     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
822     jPanel1.setLayout(jPanel1Layout);
823     jPanel1Layout.setHorizontalGroup(
824         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
825             .addGroup(jPanel1Layout.createSequentialGroup()
826                 .addContainerGap()
827                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
828                     .addComponent(searchOwnerPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
829                     .addComponent(ownerListPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
830                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
831                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
832                     .addComponent(editOwnerPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
833                     .addComponent(vehiclesPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
834                 .addGap(142, 142, 142))
835             );
836     jPanel1Layout.setVerticalGroup(
837         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
838             .addGroup(jPanel1Layout.createSequentialGroup()
839                 .addContainerGap()
840                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
841                     .addGroup(jPanel1Layout.createSequentialGroup()
842                         .addComponent(searchOwnerPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
843                         .addGap(18, 18, 18)
844                         .addComponent(ownerListPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
845                     .addComponent(editOwnerPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
846                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
847                 .addComponent(vehiclesPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
848                 .addGap(61, 61, 61))
849             );
850
851     jScrollPane3.setViewportView(jPanel1);
852
853     javax.swing.GroupLayout searchEditOwnerPanelLayout = new
javax.swing.GroupLayout(searchEditOwnerPanel);
854     searchEditOwnerPanel.setLayout(searchEditOwnerPanelLayout);
855     searchEditOwnerPanelLayout.setHorizontalGroup(
856         searchEditOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
857             .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE, 1192, Short.MAX_VALUE)
858         );
859     searchEditOwnerPanelLayout.setVerticalGroup(
860         searchEditOwnerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
861             .addComponent(jScrollPane3, javax.swing.GroupLayout.DEFAULT_SIZE, 750, Short.MAX_VALUE)

```

```

862     );
863
864     searchOwnerTabPanel.addTab("Search/Edit Owner", searchEditOwnerPanel);
865
866     javax.swing.GroupLayout oPanelLayout = new javax.swing.GroupLayout(oPanel);
867     oPanel.setLayout(oPanelLayout);
868     oPanelLayout.setHorizontalGroup(
869         oPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
870             .addComponent(searchOwnerTabPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 1197,
javax.swing.GroupLayout.PREFERRED_SIZE)
871     );
872     oPanelLayout.setVerticalGroup(
873         oPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
874             .addGroup(oPanelLayout.createSequentialGroup()
875                 .addComponent(searchOwnerTabPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 778,
javax.swing.GroupLayout.PREFERRED_SIZE)
876                 .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
877     );
878
879     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
880     getContentPane().setLayout(layout);
881     layout.setHorizontalGroup(
882         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
883             .addGroup(layout.createSequentialGroup()
884                 .addComponent(oPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
885                 .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
886     );
887     layout.setVerticalGroup(
888         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
889             .addGroup(layout.createSequentialGroup()
890                 .addComponent(oPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 779,
javax.swing.GroupLayout.PREFERRED_SIZE)
891                 .addGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
892     );
893
894     java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
895     setBounds((screenSize.width-1211)/2, (screenSize.height-818)/2, 1211, 818);
896 } // </editor-fold>
897
898 private void nameTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
899     // OMG !!!
900 }
901
902 private void saveEditedOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
903
904     updateOwner();
905
906 }
907
908 private void editOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
909
910     deleteOwnerButton.setEnabled(true);
911     saveEditedOwnerButton.setEnabled(true);
912     resetEditedOwnerFields.setEnabled(true);
913
914     editOwnerPanel.setVisible(true);
915     nameTextField1.setEditable(true);

```

```

916     nameTextField1.setBackground(Color.white);
917
918     lastNameTextField1.setEditable(true);
919     lastNameTextField1.setBackground(Color.white);
920
921     addressTextField1.setEditable(true);
922     addressTextField1.setBackground(Color.white);
923
924     address2TextField1.setEditable(true);
925     address2TextField1.setBackground(Color.white);
926
927     zipTextField1.setEditable(true);
928     zipTextField1.setBackground(Color.white);
929
930     cityTextField1.setEditable(true);
931     cityTextField1.setBackground(Color.white);
932
933     countryTextField1.setEditable(true);
934     countryTextField1.setBackground(Color.white);
935
936     phoneTextField1.setEditable(true);
937     phoneTextField1.setBackground(Color.white);
938
939     emailTextField1.setEditable(true);
940     emailTextField1.setBackground(Color.white);
941
942
943 }
944
945 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
946     // TODO add your handling code here:
947     resetFields();
948 }
949
950 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
951
952     //////////// REMEBER A "DO YOU WANT TO SAVE" DIAGLOG ////////////
953     registerOwner();
954
955 }
956
957 private void phoneTextFieldFocusLost(java.awt.event.FocusEvent evt) {
958     // TODO add your handling code here:
959     if(phoneTextField.getText().equals(""))
960     {
961         errorMessage = PHONE_EMPTY;
962
963         phoneTextField.setBackground(Color.PINK);
964     }
965     else if(countryTextField.getText().equals(DENMARK) &&
!CheckInput.checkPhone(phoneTextField.getText()))
966     {
967         errorMessage = PHONE_INVALID;
968
969         phoneTextField.setBackground(Color.PINK);
970     }
971     else
972     {

```

```

973     phoneTextField.setBackground(Color.white);
974 }
975 }
976
977 private void emailTextFieldFocusLost(java.awt.event.FocusEvent evt) {
978     // TODO add your handling code here:
979     if(!emailTextField.getText().equals("") && !CheckInput.checkEmail(emailTextField.getText()))
980     {
981         errorMessage = EMAIL_INVALID;
982         emailTextField.setBackground(Color.PINK);
983     }
984     else
985     {
986         emailTextField.setBackground(Color.white);
987     }
988 }
989
990 private void countryTextFieldFocusLost(java.awt.event.FocusEvent evt) {
991     // TODO add your handling code here:
992     if(!countryTextField.getText().equals("") && !CheckInput.checkCityOrCountry(countryTextField.getText()))
993     {
994         errorMessage = COUNTRY_INVALID;
995         countryTextField.setBackground(Color.PINK);
996     }
997     else
998     {
999         countryTextField.setBackground(Color.white);
1000     }
1001 }
1002
1003 private void zipTextFieldFocusLost(java.awt.event.FocusEvent evt) {
1004     // TODO add your handling code here:
1005     if(!zipTextField.getText().equals("") && countryTextField.getText().equals(DENMARK) &&
1006         !CheckInput.checkZip(zipTextField.getText()))
1007     {
1008         errorMessage = ZIP_INVALID;
1009         zipTextField.setBackground(Color.PINK);
1010     }
1011     else
1012     {
1013         try
1014         {
1015             String tempCity = zipCity.get(zipTextField.getText()).toString().toUpperCase();
1016             cityTextField.setText(tempCity);
1017             zipTextField.setBackground(Color.white);
1018         }
1019         catch(NullPointerException npe)
1020         {
1021             cityTextField.setText("");
1022         }
1023     }
1024 }
1025 }
1026
1027 private void cityTextFieldFocusLost(java.awt.event.FocusEvent evt) {
1028     // TODO add your handling code here:
1029 }
1030

```

```

1031 private void cityTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1032     // TODO add your handling code here:
1033
1034 }
1035
1036 private void cityTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1037     // TODO add your handling code here:
1038 }
1039
1040 private void address2TextFieldFocusLost(java.awt.event.FocusEvent evt) {
1041     // TODO add your handling code here:
1042     if(!address2TextField.getText().equals("") && !CheckInput.checkAddress2(address2TextField.getText()) ||
CheckInput.isInt(address2TextField.getText()))
1043     {
1044         errorMessage = ADDRESS2_INVALID;
1045         address2TextField.setBackground(Color.PINK);
1046     }
1047     else
1048         address2TextField.setBackground(Color.white);
1049 }
1050
1051 private void addressTextFieldFocusLost(java.awt.event.FocusEvent evt) {
1052     // TODO add your handling code here:
1053     if(!addressTextField.getText().equals("") && !CheckInput.checkAddress(addressTextField.getText()) ||
CheckInput.isInt(addressTextField.getText()))
1054     {
1055         errorMessage = ADDRESS_INVALID;
1056         addressTextField.setBackground(Color.PINK);
1057     }
1058     else
1059         addressTextField.setBackground(Color.white);
1060 }
1061
1062 private void lastNameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
1063     // TODO add your handling code here:
1064     if(lastNameTextField.getText().equals(""))
1065     {
1066         errorMessage = LASTNAME_EMPTY;
1067         lastNameTextField.setBackground(Color.PINK);
1068     }
1069     else if(!lastNameTextField.getText().equals("") &&
!CheckInput.checkLastName(lastNameTextField.getText()) || CheckInput.isInt(lastNameTextField.getText()))
1070     {
1071         errorMessage = LASTNAME_INVALID;
1072         lastNameTextField.setBackground(Color.PINK);
1073     }
1074     else
1075         lastNameTextField.setBackground(Color.white);
1076 }
1077
1078 private void nameTextFieldFocusLost(java.awt.event.FocusEvent evt) {
1079     // TODO add your handling code here:
1080     if(nameTextField.getText().equals(""))
1081     {
1082         errorMessage = NAME_EMPTY;
1083         nameTextField.setBackground(Color.PINK);
1084     }

```

```

1085     else if(!nameTextField.getText().equals("") && !CheckInput.checkFirstName(nameTextField.getText()) ||
CheckInput.isInt(nameTextField.getText()))
1086     {
1087         errorMessage = NAME_INVALID;
1088         nameTextField.setBackground(Color.PINK);
1089     }
1090     else
1091         nameTextField.setBackground(Color.white);
1092 }
1093
1094 private void nameTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1095     // TODO add your handling code here:
1096 }
1097
1098 private void nameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1099     // TODO add your handling code here:
1100     if(nameTextField.getBackground().equals(Color.PINK))
1101     {
1102         nameTextField.setBackground(Color.white);
1103     }
1104 }
1105
1106 private void lastNameTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1107     // TODO add your handling code here:
1108     if(lastNameTextField.getBackground().equals(Color.PINK))
1109     {
1110         lastNameTextField.setBackground(Color.white);
1111     }
1112 }
1113
1114 private void addressTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1115     // TODO add your handling code here:
1116     if(addressTextField.getBackground().equals(Color.PINK))
1117     {
1118         addressTextField.setBackground(Color.white);
1119     }
1120 }
1121
1122 private void address2TextFieldFocusGained(java.awt.event.FocusEvent evt) {
1123     // TODO add your handling code here:
1124     if(address2TextField.getBackground().equals(Color.PINK))
1125     {
1126         address2TextField.setBackground(Color.white);
1127     }
1128 }
1129
1130 private void zipTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1131     // TODO add your handling code here:
1132     if(cityTextField.getBackground().equals(Color.PINK))
1133     {
1134         zipTextField.setBackground(Color.white);
1135     }
1136 }
1137
1138 private void countryTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1139     // TODO add your handling code here:
1140     if(countryTextField.getBackground().equals(Color.PINK))
1141     {

```



```

1142     countryTextField.setBackground(Color.white);
1143 }
1144 }
1145
1146 private void emailTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1147     // TODO add your handling code here:
1148     if(emailTextField.getBackground().equals(Color.PINK))
1149     {
1150         emailTextField.setBackground(Color.white);
1151     }
1152 }
1153
1154 private void phoneTextFieldFocusGained(java.awt.event.FocusEvent evt) {
1155     // TODO add your handling code here:
1156     if(phoneTextField.getBackground().equals(Color.PINK))
1157     {
1158         phoneTextField.setBackground(Color.white);
1159     }
1160 }
1161
1162 private void zipTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1163     // TODO add your handling code here:
1164 }
1165
1166 private void searchOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
1167     //Get the text from the searchfield
1168     setValueOfSearchField((String) searchOwnerComboBox.getSelectedItem());
1169     String searchCriteria = valueOfSearchField;
1170     resetFields();
1171     disableEditOwnerPanel();
1172     //Set the arrays with the searchResults
1173     enterSearchDetails(searchCriteria);
1174     //If nothing is found, display a label to the user
1175     if(arrActFoundsFirstnameLastNamePhoneNumber.length==0)
1176     {
1177         noSearchResultsLabel.setText("No searchresults");
1178         noSearchResultsLabel.setForeground(Color.red);
1179         foundOwnersJList.setModel(new javax.swing.AbstractListModel()
1180         {
1181             public int getSize() { return arrActFoundsFirstnameLastNamePhoneNumber.length; }
1182             public Object getElementAt(int i) { return arrActFoundsFirstnameLastNamePhoneNumber[i]; }
1183         });
1184     } //end if
1185
1186     //If there's results present them.
1187     else
1188     {
1189         //Clear the actual JList and make a new one
1190         noSearchResultsLabel.setText("");
1191         foundOwnersJList.setModel(new javax.swing.AbstractListModel()
1192         {
1193             public int getSize() { return arrActFoundsFirstnameLastNamePhoneNumber.length; }
1194             public Object getElementAt(int i) { return arrActFoundsFirstnameLastNamePhoneNumber[i]; }
1195         });
1196     } //end else
1197
1198 }
1199

```



```

1200 private void registerOwnerPanelPropertyChange(java.beans.PropertyChangeEvent evt) {
1201     // TODO add your handling code here:
1202     // First check if name, lastname and phone are filled
1203
1204
1205 }
1206
1207 private void searchOwnerTabPanelStateChanged(javax.swing.event.ChangeEvent evt) {
1208     // Check if panel changed to is not registerOwnerPanel
1209     if(!searchOwnerTabPanel.getSelectedComponent().equals(registerOwnerPanel) && ownerSaved==false)
1210
1211         // First check if firstName, lastName and phone is ok
1212         if(!nameTextField.getText().equals("") && CheckInput.checkFirstName(nameTextField.getText())
1213             && !lastNameTextField.getText().equals("") &&
1214             CheckInput.checkLastName(lastNameTextField.getText())
1215             && countryTextField.getText().equals(DENMARK) &&
1216             CheckInput.checkPhone(phoneTextField.getText())
1217             || CheckInput.isInt(nameTextField.getText())
1218             || CheckInput.isInt(lastNameTextField.getText()))
1219         {
1220             // Saved the selected panel temp
1221             Component selected = searchOwnerTabPanel.getSelectedComponent();
1222             searchOwnerTabPanel.setSelectedComponent(registerOwnerPanel);
1223             options = JOptionPane.showConfirmDialog(null, "Do you want to save?", "Do you Want to save the
owner?", JOptionPane.YES_NO_OPTION);
1224             if(options==0)
1225             {
1226                 registerOwner();
1227                 resetFields();
1228                 setOwnerSavedStatus(true);
1229             }
1230             else if(options==1)
1231             {
1232                 resetFields();
1233                 searchOwnerTabPanel.setSelectedComponent(selected);
1234                 setOwnerSavedStatus(false);
1235             }
1236             resetFields();
1237         }
1238     }
1239
1240 private void selectOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
1241
1242     disableEditOwnerPanel();
1243     if(foundOwnersJList.getSelectedIndex() >=0)
1244     {
1245         labelSelectOwner.setText(" ");
1246         attachToVehicleButton.setEnabled(true);
1247         editOwnerButton.setEnabled(true);
1248         int indexPositionOfSelectedOwner= foundOwnersJList.getSelectedIndex();
1249
1250         String ownerID = arrActFoundsOwnerID[indexPositionOfSelectedOwner];
1251
1252         int i = 0;
1253         boolean found = false;
1254         while (actFoundsObjects.size() > i && !found )

```

```

1255     {
1256         if(actFoundsObjects.get(i).getID().equals(ownerID))
1257         {
1258             selectedOwner = actFoundsObjects.get(i);
1259             found = true;
1260         } //end if
1261         else
1262         {
1263             i++;
1264         } //end else
1265     } //end while
1266
1267     nameTextField1.setText(selectedOwner.getFirstName());
1268     lastNameTextField1.setText(selectedOwner.getLastName());
1269     addressTextField1.setText(selectedOwner.getAddress());
1270     address2TextField1.setText(selectedOwner.getAddress2());
1271     zipTextField1.setText(selectedOwner.getZip());
1272     cityTextField1.setText(selectedOwner.getCity());
1273     countryTextField1.setText(selectedOwner.getCountry());
1274     phoneTextField1.setText(selectedOwner.getPhone());
1275     emailTextField1.setText(selectedOwner.getEmail());
1276
1277     //Also present vehicles attached to that owner
1278     getVehiclesAttachedToOwner(moc.searchOwnerWithVehicle(ownerID), moc.getActiveNess());
1279     setTableWithVehicles(listWithVehiclesAttachedToOwner, arrayWithVehiclesAttachedToOwner);
1280
1281 }
1282 else
1283 {
1284     labelSelectOwner.setText(NO_OWNER_SELECTED);
1285     selectedOwner = null;
1286 }
1287
1288
1289 }
1290
1291 private void resetEditedOwnerFieldsActionPerformed(java.awt.event.ActionEvent evt) {
1292     // TODO add your handling code here:
1293     options = JOptionPane.showConfirmDialog(null, "Are you sure you want to reset all " +
1294     selectedOwner.getFirstName() + " " + selectedOwner.getLastName() + " Details ??", "Want to reset ?",
1295     JOptionPane.YES_NO_OPTION);
1296     if(options==JOptionPane.YES_OPTION)
1297     {
1298         resetFields();
1299     }
1300     else
1301     {
1302         return;
1303     }
1304 }
1305
1306 private void jButton1KeyPressed(java.awt.event.KeyEvent evt) {
1307     if (evt.getKeyCode() == KeyEvent.VK_ENTER)
1308     {
1309         registerOwner();
1310     }
1311 }

```

```

1311 }
1312
1313 private void searchOwnerComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
1314     setValueOfSearchField((String) searchOwnerComboBox.getSelectedItem());
1315 }
1316
1317 private void searchOwnerComboBoxItemStateChanged(java.awt.event.ItemEvent evt) {
1318     /** if(searchOwnerTextField.getSelectedIndex()==0)
1319     {
1320         searchOwnerTextField.setBackground(Color.PINK);
1321     }
1322     else
1323     {
1324         searchOwnerTextField.setBackground(Color.white);
1325     }**/
1326 }
1327
1328 private void registerOwnerPanelFocusGained(java.awt.event.FocusEvent evt) {
1329     GUIHelpUtil.setFocusAndDefaultComponents(nameTextField, jButton1);
1330 }
1331
1332 private void deleteOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
1333     // TODO add your handling code here:
1334     deleteOwner();
1335     resetFields();
1336     disableEditOwnerPanel();
1337     labelSelectOwner.setText("Owner Deleted!");
1338 }
1339
1340 private void attachToVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
1341
1342     AttachVehicleGUI.getInstance(selectedOwner.getID());
1343 }
1344
1345 private void phoneTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1346     // TODO add your handling code here:
1347 }
1348
1349 private void address2TextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1350     // TODO add your handling code here:
1351 }
1352
1353 private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
1354     SuperSearch.getInstance().run();
1355 }
1356
1357 private void viewVehicleDataButtonActionPerformed(java.awt.event.ActionEvent evt) {
1358     if (jTable1.getSelectedRow() >= 0)
1359         JOptionPane.showMessageDialog(null, listWithVehiclesAttachedToOwner.get(jTable1.getSelectedRow()));
1360     else
1361         JOptionPane.showMessageDialog(null, "No vehice selected");
1362 }
1363
1364 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
1365     // TODO add your handling code here:
1366 }
1367 }
1368

```

```

1369 /**
1370  * Returns the owner GUI object, to insert in mainGUI
1371  * @return return oPanel
1372  */
1373 public JPanel getPanel()
1374 {
1375     fillHashMap();
1376     return oPanel;
1377 }//end getPanel
1378
1379 /**
1380  * Register a new owner
1381  */
1382 private void registerOwner()
1383 {
1384     // First check if name, lastname and phone are filled
1385     if(nameTextField.getText().equals(""))
1386     {
1387         errorMessage = errorMessage + " " + NAME_EMPTY;
1388         nameTextField.setBackground(Color.PINK);
1389     }
1390     else if(!nameTextField.getText().equals("") && !CheckInput.checkFirstName(nameTextField.getText()) ||
CheckInput.isInt(nameTextField.getText()))
1391     {
1392         errorMessage = errorMessage + " " + NAME_INVALID;
1393         nameTextField.setBackground(Color.PINK);
1394     }
1395     if(lastNameTextField.getText().equals(""))
1396     {
1397         errorMessage = errorMessage + " " + LASTNAME_EMPTY;
1398         lastNameTextField.setBackground(Color.PINK);
1399     }
1400     else if(!lastNameTextField.getText().equals("") &&
!CheckInput.checkLastName(lastNameTextField.getText()) || CheckInput.isInt(lastNameTextField.getText()))
1401     {
1402         errorMessage = errorMessage + " " + LASTNAME_INVALID;
1403         lastNameTextField.setBackground(Color.PINK);
1404     }
1405     if(phoneTextField.getText().equals(""))
1406     {
1407         errorMessage = errorMessage + " " + PHONE_EMPTY;
1408         phoneTextField.setBackground(Color.PINK);
1409     }
1410     else if(countryTextField.getText().equals(DENMARK) &&
!CheckInput.checkPhone(phoneTextField.getText()))
1411     {
1412         errorMessage = errorMessage + " " + PHONE_INVALID;
1413         phoneTextField.setBackground(Color.PINK);
1414     }
1415
1416     // Check if all field are okay
1417     if(checkOwnerFields())
1418     {
1419         moc.makeNewOwner(nameTextField.getText(), lastNameTextField.getText(),
1420             addressTextField.getText(), address2TextField.getText(),
1421             zipTextField.getText(),cityTextField.getText(),
1422             countryTextField.getText(),phoneTextField.getText(),
1423             emailTextField.getText());

```

```

1424     jResultLabelOwner.setText(OK_OWNER_SAVED);
1425     resetFields();
1426
1427 }//end if
1428 else
1429 {
1430     jResultLabelOwner.setText(errorMessage);
1431 }//end else
1432
1433 }
1434
1435 public void updateOwner()
1436 {
1437     // First check if name, lastname and phone are filled
1438     if(nameTextField1.getText().equals(""))
1439     {
1440         errorMessage = errorMessage + " " + NAME_EMPTY;
1441         nameTextField1.setBackground(Color.PINK);
1442     }
1443     else if(!nameTextField1.getText().equals("") && !CheckInput.checkFirstName(nameTextField1.getText()) ||
CheckInput.isInt(nameTextField1.getText()))
1444     {
1445         errorMessage = errorMessage + " " + NAME_INVALID;
1446         nameTextField1.setBackground(Color.PINK);
1447     }
1448     if(lastNameTextField1.getText().equals(""))
1449     {
1450         errorMessage = errorMessage + " " + LASTNAME_EMPTY;
1451         lastNameTextField1.setBackground(Color.PINK);
1452     }
1453     else if(!lastNameTextField1.getText().equals("") &&
!CheckInput.checkLastName(lastNameTextField1.getText()) || CheckInput.isInt(lastNameTextField1.getText()))
1454     {
1455         errorMessage = errorMessage + " " + LASTNAME_INVALID;
1456         lastNameTextField1.setBackground(Color.PINK);
1457     }
1458     if(phoneTextField1.getText().equals(""))
1459     {
1460         errorMessage = errorMessage + " " + PHONE_EMPTY;
1461         phoneTextField1.setBackground(Color.PINK);
1462     }
1463     else if(countryTextField1.getText().equals(DENMARK) &&
!CheckInput.checkPhone(phoneTextField1.getText()))
1464     {
1465         errorMessage = errorMessage + " " + PHONE_INVALID;
1466         phoneTextField1.setBackground(Color.PINK);
1467     }
1468
1469     // Check if all field are okay
1470     if(checkOwnerEditFields())
1471     {
1472         moc.updateOwner(selectedOwner.getID(), nameTextField1.getText(), lastNameTextField1.getText(),
1473             addressTextField1.getText(), address2TextField1.getText(),
1474             zipTextField1.getText(),cityTextField1.getText(),
1475             countryTextField1.getText(), phoneTextField1.getText(),
1476             emailTextField1.getText());
1477         JOptionPane.showMessageDialog(null, "Owner Successfully updated !", "Successfully updated",
JOptionPane.INFORMATION_MESSAGE);

```

```

1478     resetFields();
1479     disableEditOwnerPanel();
1480
1481 } //end if
1482 else
1483 {
1484     jResultLabelOwner.setText(errorMessage);
1485 } //end else
1486 }
1487
1488 /**
1489  * Delete an owner if selected
1490  */
1491 public void deleteOwner()
1492 {
1493     options = JOptionPane.showConfirmDialog(null, "Are you sure you want to delete this owner??", "Do You
1494 want to delete?", JOptionPane.YES_NO_OPTION);
1495     if (options == JOptionPane.YES_OPTION)
1496     {
1497         moc.deleteOwner(selectedOwner.getID());
1498     } // End if
1499     else
1500     {
1501         return;
1502     } // End else
1503 } // end delete owner method
1504
1505 private String setValueOfSearchField(String valueOfSearchField)
1506 {
1507     return this.valueOfSearchField = valueOfSearchField;
1508 } //end setValueOfSearchField
1509
1510 /**
1511  * Checks input for fields in Owner for errors
1512  * @return return true/false
1513  */
1514 public boolean checkOwnerFields()
1515 {
1516     errorMessage = "";
1517     if (nameTextField.getText().equals(""))
1518     {
1519         nameTextField.setBackground(Color.PINK);
1520         return false;
1521     }
1522     else if (!nameTextField.getText().equals("") && !CheckInput.checkFirstName(nameTextField.getText()) ||
1523 CheckInput.isInt(nameTextField.getText()))
1524     {
1525         nameTextField.setBackground(Color.PINK);
1526         return false;
1527     }
1528     if (lastNameTextField.getText().equals(""))
1529     {
1530         lastNameTextField.setBackground(Color.PINK);
1531         return false;
1532     }

```

```

1533     else if(!lastNameTextField.getText().equals("")) &&
!CheckInput.checkLastName(lastNameTextField.getText()) || CheckInput.isInt(lastNameTextField.getText()) ||
1534     {
1535         lastNameTextField.setBackground(Color.PINK);
1536         return false;
1537     }
1538
1539     if(!addressTextField.getText().equals("")) && !CheckInput.checkAddress(addressTextField.getText()) ||
CheckInput.isInt(addressTextField.getText())
1540     {
1541         errorMessage = ADDRESS_INVALID;
1542         addressTextField.setBackground(Color.PINK);
1543         return false;
1544     }
1545
1546     if(!address2TextField.getText().equals("")) && !CheckInput.checkAddress2(address2TextField.getText()) ||
CheckInput.isInt(address2TextField.getText())
1547     {
1548         errorMessage = ADDRESS2_INVALID;
1549         address2TextField.setBackground(Color.PINK);
1550         return false;
1551     }
1552
1553     if(!zipTextField.getText().equals("")) && countryTextField.getText().equals(DENMARK) &&
!CheckInput.checkZip(zipTextField.getText())
1554         !CheckInput.checkZip(zipTextField.getText())
1555     {
1556         errorMessage = ZIP_INVALID;
1557         zipTextField.setBackground(Color.PINK);
1558         return false;
1559     }
1560
1561     if(cityTextField.getText().equals("")) && !CheckInput.checkCityOrCountry(cityTextField.getText())
1562     {
1563         errorMessage = CITY_EMPTY;
1564         cityTextField.setBackground(Color.PINK);
1565         return false;
1566     }
1567
1568     if(!countryTextField.getText().equals("")) &&
!CheckInput.checkCityOrCountry(countryTextField.getText())
1569     {
1570         errorMessage = COUNTRY_INVALID;
1571         countryTextField.setBackground(Color.PINK);
1572         return false;
1573     }
1574
1575     if(!emailTextField.getText().equals("")) && !CheckInput.checkEmail(emailTextField.getText())
1576     {
1577         errorMessage = EMAIL_INVALID;
1578         emailTextField.setBackground(Color.PINK);
1579         return false;
1580     }
1581
1582     if(phoneTextField.getText().equals(""))
1583     {
1584         phoneTextField.setBackground(Color.PINK);
1585         return false;
1586     }

```

```

1587     else if(countryTextField.getText().equals(DENMARK) &&
!CheckInput.checkPhone(phoneTextField.getText()))
1588     {
1589         phoneTextField.setBackground(Color.PINK);
1590         return false;
1591     }
1592
1593     return true;
1594 }//end checkOwnerFields
1595
1596 /**
1597  * Checks input for fields in Owner for edit
1598  * @return return true/false
1599  */
1600 public boolean checkOwnerEditFields()
1601 {
1602     errorMessage = "";
1603     if(nameTextField1.getText().equals(""))
1604     {
1605         nameTextField1.setBackground(Color.PINK);
1606         return false;
1607     }
1608     else if(!nameTextField1.getText().equals("") && !CheckInput.checkFirstName(nameTextField1.getText()))
1609     {
1610         nameTextField1.setBackground(Color.PINK);
1611         return false;
1612     }
1613
1614     if(lastNameTextField1.getText().equals(""))
1615     {
1616         lastNameTextField1.setBackground(Color.PINK);
1617         return false;
1618     }
1619     else if(!lastNameTextField1.getText().equals("") &&
!CheckInput.checkLastName(lastNameTextField1.getText()))
1620     {
1621         lastNameTextField1.setBackground(Color.PINK);
1622         return false;
1623     }
1624
1625     if(!addressTextField1.getText().equals("") && !CheckInput.checkAddress(addressTextField1.getText()))
1626     {
1627         errorMessage = ADDRESS_INVALID;
1628         addressTextField1.setBackground(Color.PINK);
1629         return false;
1630     }
1631
1632     if(!address2TextField1.getText().equals("") && !CheckInput.checkAddress2(address2TextField1.getText()))
1633     {
1634         errorMessage = ADDRESS2_INVALID;
1635         address2TextField1.setBackground(Color.PINK);
1636         return false;
1637     }
1638
1639     if(!zipTextField1.getText().equals("") && countryTextField1.getText().equals(DENMARK) &&
!CheckInput.checkZip(zipTextField1.getText()))
1640     {
1641
1642         errorMessage = ZIP_INVALID;

```



```

1643     zipTextField1.setBackground(Color.PINK);
1644     return false;
1645 }
1646
1647 if(cityTextField1.getText().equals("") && !CheckInput.checkCityOrCountry(cityTextField1.getText()))
1648 {
1649     errorMessage = CITY_EMPTY;
1650     cityTextField1.setBackground(Color.PINK);
1651     return false;
1652 }
1653
1654 if(!countryTextField1.getText().equals("") &&
!CheckInput.checkCityOrCountry(countryTextField1.getText()))
1655 {
1656     errorMessage = COUNTRY_INVALID;
1657     countryTextField1.setBackground(Color.PINK);
1658     return false;
1659 }
1660
1661 if(!emailTextField1.getText().equals("") && !CheckInput.checkEmail(emailTextField1.getText()))
1662 {
1663     errorMessage = EMAIL_INVALID;
1664     emailTextField1.setBackground(Color.PINK);
1665     return false;
1666 }
1667
1668 if(phoneTextField1.getText().equals(""))
1669 {
1670     phoneTextField1.setBackground(Color.PINK);
1671     return false;
1672 }
1673 else if(countryTextField1.getText().equals(DENMARK) &&
!CheckInput.checkPhone(phoneTextField1.getText()))
1674 {
1675     phoneTextField1.setBackground(Color.PINK);
1676     return false;
1677 }
1678
1679 return true;
1680 }//end checkOwnerFields
1681
1682 /**
1683  * Reset all fields in Owner GUI
1684  */
1685 private void resetFields()
1686 {
1687     ////////////////////////////////// REGISTER OWNER TAB PANE RESET //////////////////////////////////
1688
1689     // Reset All Register Owner fields
1690     nameTextField.setBackground(Color.white);
1691     nameTextField.setText("");
1692
1693     lastNameTextField.setBackground(Color.white);
1694     lastNameTextField.setText("");
1695
1696     addressTextField.setBackground(Color.white);
1697     addressTextField.setText("");
1698

```

```

1699     address2TextField.setBackground(Color.white);
1700     address2TextField.setText("");
1701
1702     zipTextField.setBackground(Color.white);
1703     zipTextField.setText("");
1704
1705     cityTextField.setText("");
1706
1707     countryTextField.setBackground(Color.white);
1708     countryTextField.setText(DENMARK);
1709
1710     emailTextField.setBackground(Color.white);
1711     emailTextField.setText("");
1712
1713     phoneTextField.setBackground(Color.white);
1714     phoneTextField.setText("");
1715
1716     // Set owner saved status to false
1717     ownerSaved = false;
1718
1719     //////////// SEARCH/EDIT OWNER TAB PANE RESET ////////////
1720
1721     // Reset the status label for selected owner
1722     labelSelectOwner.setText(" ");
1723
1724     // Reset attach, edit and search Owner buttons and combobox and JList
1725     attachToVehicleButton.setEnabled(false);
1726     editOwnerButton.setEnabled(false);
1727     searchOwnerComboBox.setSelectedIndex(0);
1728     foundOwnersJList.setListData(new String[0]);
1729
1730     // Reset all edit owner Fields
1731     nameTextField1.setText("");
1732     lastNameTextField1.setText("");
1733     addressTextField1.setText("");
1734     address2TextField1.setText("");
1735     zipTextField1.setText("");
1736     cityTextField1.setText("");
1737     countryTextField1.setText("");
1738     phoneTextField1.setText("");
1739     emailTextField1.setText("");
1740
1741 }
1742
1743 /**
1744  * Disable all buttons and fields in edit owner panel
1745  */
1746 public void disableEditOwnerPanel()
1747 {
1748     saveEditedOwnerButton.setEnabled(false);
1749     deleteOwnerButton.setEnabled(false);
1750     resetEditedOwnerFields.setEnabled(false);
1751
1752     // Reset name textfield in search/edit owner
1753     nameTextField1.setBackground(new java.awt.Color(220, 220, 220));
1754     nameTextField1.setText("");
1755     nameTextField1.setEditable(false);
1756

```

```

1757 // Reset lastname textfield in searc/edit owner
1758 lastNameTextField1.setBackground(new java.awt.Color(220, 220, 220));
1759 lastNameTextField1.setText("");
1760 lastNameTextField1.setEditable(false);
1761
1762 // Reset address textfield in searc/edit owner
1763 addressTextField1.setBackground(new java.awt.Color(220, 220, 220));
1764 addressTextField1.setText("");
1765 addressTextField1.setEditable(false);
1766
1767 // Reset address2 textfield in searc/edit owner
1768 address2TextField1.setBackground(new java.awt.Color(220, 220, 220));
1769 address2TextField1.setText("");
1770 address2TextField1.setEditable(false);
1771
1772 // Reset zip textfield in searc/edit owner
1773 zipTextField1.setBackground(new java.awt.Color(220, 220, 220));
1774 zipTextField1.setText("");
1775 zipTextField1.setEditable(false);
1776
1777 //Reset city textfield in searc/edit owner
1778 cityTextField1.setBackground(new java.awt.Color(220, 220, 220));
1779 cityTextField1.setText("");
1780 cityTextField1.setEditable(false);
1781
1782 //Reset country textfield in searc/edit owner
1783 countryTextField1.setBackground(new java.awt.Color(220, 220, 220));
1784 countryTextField1.setText("");
1785 countryTextField1.setEditable(false);
1786
1787 //Reset Phone textfield in searc/edit owner
1788 phoneTextField1.setBackground(new java.awt.Color(220, 220, 220));
1789 phoneTextField1.setText("");
1790 phoneTextField1.setEditable(false);
1791
1792 // Reset Email textfield in searc/edit owner
1793 emailTextField1.setBackground(new java.awt.Color(220, 220, 220));
1794 emailTextField1.setText("");
1795 emailTextField1.setEditable(false);
1796 }
1797
1798 /**
1799  * Return owners saved status
1800  * @return owners saved status
1801  */
1802 public boolean getOwnerSaved()
1803 {
1804     return ownerSaved;
1805 }
1806
1807 /**
1808  * Set the owner saved status
1809  * @param ownerSaved boolean owner saved status
1810  */
1811 public void setOwnerSavedStatus(boolean ownerSaved)
1812 {
1813     this.ownerSaved = ownerSaved;
1814 }

```

```

1815
1816 /**
1817  * Fill hashmap with cities and zipcodes
1818  */
1819 private void fillHashMap()
1820 {
1821     try
1822     {
1823         FileReader readerZip = new FileReader(fileZip);
1824         FileReader readerCity = new FileReader(fileCity);
1825         Scanner inZip = new Scanner(readerZip);
1826         Scanner inCity = new Scanner(readerCity);
1827         while(inZip.hasNext() && inCity.hasNext())
1828         {
1829             String zip = inZip.nextLine();
1830             String city = inCity.nextLine().toLowerCase();
1831             zipCity.put(zip, city);
1832         } //end while
1833     } //end try
1834     catch(FileNotFoundException fne)
1835     {
1836         FileWriterException.writeLogFile(OwnerGUI.class.getName() + fne.getMessage());
1837     } //end catch
1838
1839 } //end fillHashMap
1840
1841 /**
1842  * This method ask the controller-layer to perform a search for an owner
1843  * Search results are set from the results of the controller-layer as multiple
1844  * lists of Owner-objects, strings and ints.
1845  *
1846  * The variables firstName, lastName, and phoneNumber in every object in the
1847  * actFoundsObjects is put together as a string and inserted to an
1848  * arrayList(actFoundsFirstNameLastNamePhoneNumber) which will be converted
1849  * to an array.
1850  *
1851  * The same applies to the list of OwnerIDs
1852  * @param searchCriteria
1853  * @return void
1854  */
1855 public void enterSearchDetails(String searchCriteria)
1856 {
1857     //This ArrayList will hold the Firstname, lastname and phonenumber of the search results
1858     actFoundsFirstNameLastNamePhoneNumber = new ArrayList<String>();
1859     //This ArrayList will hold the OwnerID of the search results
1860     actFoundsOwnerID = new ArrayList<String>();
1861     //This ArrayList will hold the Owner-objects of the search results
1862     actFoundsObjects = new ArrayList<Owner>();
1863
1864     actFoundsObjects = moc.searchOwner(searchCriteria);
1865
1866     int i = 0;
1867     for (Owner o : actFoundsObjects)
1868     {
1869         //ID, Firstname, lastname, phonenumber and ownerID goes to the respective ArrayLists
1870
1871         actFoundsFirstNameLastNamePhoneNumber.add("(" + o.getID() + ") " + o.getFirstName() + " " +
o.getLastName() + " (" + o.getPhone() + ")");

```

```

1872     actFoundsOwnerID.add(o.getID());
1873     i++;
1874 } //end for each
1875
1876 //Here the ArrayList is converted to arrays (1-dim) (for presentation purpose)
1877 arrActFoundsFirstnameLastNamePhoneNumber = actFoundsFirstnameLastNamePhoneNumber.toArray(new
String[0]);
1878 arrActFoundsOwnerID = actFoundsOwnerID.toArray(new String[0]);
1879
1880 }
1881
1882 /*
1883  * Method for building a list of all the names in database.
1884  * Used for autocompletion
1885  */
1886 private void bildAutoCompleteVectorList()
1887 {
1888     namesVector.addAll(AutocompleteDAO.getInstance().getAllNames());
1889     Collections.sort(namesVector);
1890     addHeadersInJComboBox();
1891 }
1892
1893 /**
1894  * Return vectorList with names
1895  * @return vector List
1896  */
1897 public static Vector getNamesVector()
1898 {
1899     return namesVector;
1900 }
1901
1902 /**
1903  * Add headers to Vector for JComboBox
1904  */
1905 private void addHeadersInJComboBox()
1906 {
1907     namesVector.add(0, "Enter search criteria");
1908 } //end setHeadersInJComboBox
1909
1910 /**
1911  * This method is going to present the vehicles attached to the selected owner
1912  * @param vehicles
1913  */
1914 private void setTableWithVehicles(List<Vehicle> vehicles, boolean[] activeNess)
1915 {
1916     //Build an array with the column names
1917     String[] columnNames = {"Brand", "Model", "Year", "Active"};
1918
1919     //Build a two-dim array to be used in the jTable (gulp!)
1920     //We are going to present 4 columns and as many rows as there are vehicles to that owner
1921     Object[][] data = new Object[vehicles.size()][4];
1922     int counter = 0;
1923     for (Vehicle v : vehicles)
1924     {
1925         for(int i = 0; i < 4; i++)
1926         {
1927             data[counter][0] = v.getBrand();
1928             data[counter][1] = v.getModel();

```

```

1929         data[counter][2] = Integer.toString(v.getvYear());
1930         //My tool says that "Creating new Boolean is inefficient" - however, i don't care :D
1931         data[counter][3] = new Boolean(activeNess[counter]);
1932     } //end for-each
1933     counter++;
1934 } //end for
1935
1936 //Now set set the jTable - yeah :o)
1937 jTable1.setModel(new javax.swing.table.DefaultTableModel(data, columnNames)
1938 {
1939     //In order to get this table to show a checkbox for a boolean, we'll define a class array
1940     Class[] types = new Class []
1941     {
1942         //The types are: Object, Object, Object, Boolean
1943         java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Boolean.class
1944     };
1945     //Then let's override som' shit.
1946     @Override
1947     public Class getColumnClass(int columnIndex)
1948     {
1949         return types [columnIndex];
1950     }
1951 });
1952 jTable1.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
1953
1954 }
1955
1956 private void getVehiclesAttachedToOwner(List<Vehicle> searchOwnerWithVehicle, boolean[] activeNess) {
1957     listWithVehiclesAttachedToOwner = searchOwnerWithVehicle;
1958     arrayWithVehiclesAttachedToOwner = activeNess;
1959 }
1960
1961 // Variables declaration - do not modify
1962 private javax.swing.JTextField address2TextField;
1963 private javax.swing.JTextField address2TextField1;
1964 private javax.swing.JLabel addressLabel;
1965 private javax.swing.JLabel addressLabel1;
1966 private javax.swing.JLabel addressLabel2;
1967 private javax.swing.JLabel addressLabel3;
1968 private javax.swing.JTextField addressTextField;
1969 private javax.swing.JTextField addressTextField1;
1970 private javax.swing.JButton attachToVehicleButton;
1971 private javax.swing.JTextField cityTextField;
1972 private javax.swing.JTextField cityTextField1;
1973 private javax.swing.JLabel countryLabel;
1974 private javax.swing.JLabel countryLabel1;
1975 private javax.swing.JTextField countryTextField;
1976 private javax.swing.JTextField countryTextField1;
1977 private javax.swing.JButton deleteOwnerButton;
1978 private javax.swing.JButton editOwnerButton;
1979 private javax.swing.JPanel editOwnerPanel;
1980 private javax.swing.JLabel editSelectLabel;
1981 private javax.swing.JLabel emailLabel;
1982 private javax.swing.JLabel emailLabel1;
1983 private javax.swing.JTextField emailTextField;
1984 private javax.swing.JTextField emailTextField1;
1985 private javax.swing.JList foundOwnersJList;
1986 private javax.swing.JButton jButton1;

```

```

1987 private javax.swing.JButton jButton2;
1988 private javax.swing.JButton jButton3;
1989 private javax.swing.JButton jButton4;
1990 private javax.swing.JLabel jLabel1;
1991 private javax.swing.JPanel jPanel1;
1992 private javax.swing.JLabel jLabelResultLabelOwner;
1993 private javax.swing.JLabel jLabelResultLabelOwner1;
1994 private javax.swing.JScrollPane jScrollPane1;
1995 private javax.swing.JScrollPane jScrollPane2;
1996 private javax.swing.JScrollPane jScrollPane3;
1997 private javax.swing.JSeparator jSeparator1;
1998 private javax.swing.JTable jTable1;
1999 private javax.swing.JLabel labelSelectOwner;
2000 private javax.swing.JLabel lastNameLabel;
2001 private javax.swing.JLabel lastNameLabel1;
2002 private javax.swing.JTextField lastNameTextField;
2003 private javax.swing.JTextField lastNameTextField1;
2004 private javax.swing.JLabel nameLabel;
2005 private javax.swing.JLabel nameLabel1;
2006 private javax.swing.JTextField nameTextField;
2007 private javax.swing.JTextField nameTextField1;
2008 private javax.swing.JLabel noSearchResultsLabel;
2009 public javax.swing.JPanel oPanel;
2010 private javax.swing.JPanel ownerInformationPanel;
2011 private javax.swing.JPanel ownerListPanel;
2012 private javax.swing.JLabel phoneLabel;
2013 private javax.swing.JLabel phoneLabel1;
2014 private javax.swing.JTextField phoneTextField;
2015 private javax.swing.JTextField phoneTextField1;
2016 private javax.swing.JPanel registerOwnerPanel;
2017 private javax.swing.JButton resetEditedOwnerFields;
2018 private javax.swing.JButton saveEditedOwnerButton;
2019 private javax.swing.JPanel searchEditOwnerPanel;
2020 private javax.swing.JButton searchOwnerButton;
2021 private javax.swing.JComboBox searchOwnerComboBox;
2022 private javax.swing.JPanel searchOwnerPanel;
2023 private javax.swing.JTabbedPane searchOwnerTabPanel;
2024 private javax.swing.JButton selectOwnerButton;
2025 private javax.swing.JLabel subheaderOwnerLabel;
2026 private javax.swing.JPanel vehiclesPanel;
2027 private javax.swing.JButton viewVehicleDataButton;
2028 private javax.swing.JLabel zipAndCityLabel;
2029 private javax.swing.JLabel zipAndCityLabel1;
2030 private javax.swing.JTextField zipTextField;
2031 private javax.swing.JTextField zipTextField1;
2032 // End of variables declaration
2033

```

SuperSearch

```

1 /*
2  * This class is a super search feature, that gives the user the orpotunity
3  * to search for anything anywhere.
4  * It contains static methods so it can be called from anywhere.
5  */
6
7 /*

```

```

8 * SuperSearch.java
9 *
10 * Created on 03-12-2010, 13:35:37
11 */
12
13 package tweakmc.view;
14
15 import java.awt.Color;
16 import java.util.ArrayList;
17 import java.util.List;
18 import java.util.Vector;
19 import tweakmc.control.ManageOwnerController;
20 import tweakmc.control.ManageVehicleController;
21 import tweakmc.control.OrderController;
22 import tweakmc.model.Order;
23 import tweakmc.model.Vehicle;
24 import tweakmc.model.Owner;
25
26 /**
27 *
28 * @author nn119171
29 */
30 public class SuperSearch extends javax.swing.JFrame
31 {
32     // Common instance variables
33     private String searchCriteria;
34     public static SuperSearch instance;
35
36     //Instance variables for vehicle
37     private ManageVehicleController mvc = new ManageVehicleController();
38     private Vector brandVector = VehicleGUI.getBrandVector();
39     private String ownerID;
40     private String orderID;
41     private String vehicleID;
42     private String valueOfSearchField;
43     private String foundID;
44     private List<Vehicle> actFoundsVehicleObjects;
45     private List<String> actFoundsBrandModel;
46     private List<String> actFoundVehicle;
47     private String[] arrActFoundsVehicleID;
48     private String[] arrActFoundsBrandModel;
49     private Vehicle selectedVehicle;
50     private static final String VEHICLE = "Search for Vehicle";
51     private static final String OWNER = "Search for Owner";
52     private static final String ORDER = "Search for Order";
53
54     // Instance variables for owner
55     private ManageOwnerController moc = new ManageOwnerController();
56     private String[] arrActFoundsFirstnameLastNamePhoneNumber;
57     private List<String> actFoundsFirstnameLastNamePhoneNumber;
58     private List<String> actFoundsOwnerID;
59     private List<Owner> actFoundsOwnerObjects;
60     private String[] arrActFoundsOwnerID;
61     private Owner selectedOwner;
62
63     //Instance variables for Order
64     //instance variables
65     private String errorMessage = "";

```



```

66 private OrderController oc = new OrderController();
67 private ArrayList<String> resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice;
68 private ArrayList<String> resultFoundOrderID;
69 private ArrayList<Order> resultFoundOrderObjects;
70 private ArrayList<Order> resultStart;
71 private Order selectedOrder;
72 private String[] arrResultFoundOrderID;
73 private String[] arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice;
74 private String orderIDEditable;
75
76
77 /** Creates new form SuperSearch */
78 private SuperSearch() {
79     setUndecorated(false);
80     initComponents();
81     actFoundsVehicleObjects = new ArrayList<Vehicle>();
82     actFoundsOwnerObjects = new ArrayList<Owner>();
83 }
84
85 public static SuperSearch getInstance()
86 {
87     if(instance == null)
88     {
89         instance = new SuperSearch();
90     }
91     return instance;
92 }
93
94 /** This method is called from within the constructor to
95  * initialize the form.
96  * WARNING: Do NOT modify this code. The content of this method is
97  * always regenerated by the Form Editor.
98  */
99 @SuppressWarnings("unchecked")
100 // <editor-fold defaultstate="collapsed" desc="Generated Code">
101 private void initComponents() {
102
103     buttonGroupMaster = new javax.swing.ButtonGroup();
104     buttonGroupSlave = new javax.swing.ButtonGroup();
105     infoPanel = new javax.swing.JPanel();
106     orderRadioButton = new javax.swing.JRadioButton();
107     ownerRadioButton = new javax.swing.JRadioButton();
108     vehicleRadioButton = new javax.swing.JRadioButton();
109     jSeparator1 = new javax.swing.JSeparator();
110     byLabel = new javax.swing.JLabel();
111     byVehicleRadioButton = new javax.swing.JRadioButton();
112     byOwnerRadioButton = new javax.swing.JRadioButton();
113     byOrderRadioButton = new javax.swing.JRadioButton();
114     selectedLabel = new javax.swing.JLabel();
115     searchHeaderLabel = new javax.swing.JLabel();
116     searchPanel = new javax.swing.JPanel();
117     searchTextField = new javax.swing.JTextField();
118     searchButton = new javax.swing.JButton();
119     cancelButton = new javax.swing.JButton();
120     jLabelSearch = new javax.swing.JLabel();
121     searchResultPanel = new javax.swing.JPanel();
122     jScrollPane1 = new javax.swing.JScrollPane();
123     foundSearchResults = new javax.swing.JList();

```

```

124 selectButton = new javax.swing.JButton();
125 cancelButton1 = new javax.swing.JButton();
126 statusLabel = new javax.swing.JLabel();
127 tempLABELbyNYVANG = new javax.swing.JLabel();
128
129 setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
130
131 infoPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Search for"));
132 infoPanel.addFocusListener(new java.awt.event.FocusAdapter() {
133     public void focusLost(java.awt.event.FocusEvent evt) {
134         infoPanelFocusLost(evt);
135     }
136 });
137
138 buttonGroupMaster.add(orderRadioButton);
139 orderRadioButton.setText("Order");
140 orderRadioButton.addActionListener(new java.awt.event.ActionListener() {
141     public void actionPerformed(java.awt.event.ActionEvent evt) {
142         orderRadioButtonActionPerformed(evt);
143     }
144 });
145
146 buttonGroupMaster.add(ownerRadioButton);
147 ownerRadioButton.setText("Owner");
148 ownerRadioButton.addActionListener(new java.awt.event.ActionListener() {
149     public void actionPerformed(java.awt.event.ActionEvent evt) {
150         ownerRadioButtonActionPerformed(evt);
151     }
152 });
153
154 buttonGroupMaster.add(vehicleRadioButton);
155 vehicleRadioButton.setText("Vehicle");
156 vehicleRadioButton.addActionListener(new java.awt.event.ActionListener() {
157     public void actionPerformed(java.awt.event.ActionEvent evt) {
158         vehicleRadioButtonActionPerformed(evt);
159     }
160 });
161
162 buttonGroupSlave.add(byVehicleRadioButton);
163 byVehicleRadioButton.setText("Vehicle");
164 byVehicleRadioButton.setEnabled(false);
165 byVehicleRadioButton.addActionListener(new java.awt.event.ActionListener() {
166     public void actionPerformed(java.awt.event.ActionEvent evt) {
167         byVehicleRadioButtonActionPerformed(evt);
168     }
169 });
170
171 buttonGroupSlave.add(byOwnerRadioButton);
172 byOwnerRadioButton.setText("Owner");
173 byOwnerRadioButton.setEnabled(false);
174 byOwnerRadioButton.addActionListener(new java.awt.event.ActionListener() {
175     public void actionPerformed(java.awt.event.ActionEvent evt) {
176         byOwnerRadioButtonActionPerformed(evt);
177     }
178 });
179
180 buttonGroupSlave.add(byOrderRadioButton);
181 byOrderRadioButton.setText("Order");

```

```

182 byOrderRadioButton.setEnabled(false);
183 byOrderRadioButton.addActionListener(new java.awt.event.ActionListener() {
184     public void actionPerformed(java.awt.event.ActionEvent evt) {
185         byOrderRadioButtonActionPerformed(evt);
186     }
187 });
188
189 selectedLabel.setText("None selected");
190
191 javax.swing.GroupLayout infoPanelLayout = new javax.swing.GroupLayout(infoPanel);
192 infoPanel.setLayout(infoPanelLayout);
193 infoPanelLayout.setHorizontalGroup(
194     infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
195     .addGroup(infoPanelLayout.createSequentialGroup()
196         .add(infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
197             .add(infoPanelLayout.createSequentialGroup()
198                 .addGroup(infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
199                     .addGap(4, 4, 4)
200                     .addGroup(infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
201 false)
202                 .addGroup(infoPanelLayout.createSequentialGroup()
203                     .addComponent(byVehicleRadioButton)
204                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
205                     .addComponent(byOwnerRadioButton)
206                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
207                     .addComponent(byOrderRadioButton)
208                 .addGroup(infoPanelLayout.createSequentialGroup()
209                     .addComponent(vehicleRadioButton)
210                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
211                     .addComponent(ownerRadioButton)
212                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
213                     .addComponent(orderRadioButton)
214                     .addComponent(jSeparator1))
215                 .addGap(89, 89, 89))
216             .addGroup(infoPanelLayout.createSequentialGroup()
217                 .addComponent(selectedLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 242,
Short.MAX_VALUE)
218                 .addGap(24, 24, 24))
219             .addGroup(infoPanelLayout.createSequentialGroup()
220                 .addComponent(byLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 72,
javax.swing.GroupLayout.PREFERRED_SIZE)
221                 .addContainerGap(194, Short.MAX_VALUE))))
222 );
223 infoPanelLayout.setVerticalGroup(
224     infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
225     .addGroup(infoPanelLayout.createSequentialGroup()
226         .addGroup(infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
227             .addComponent(vehicleRadioButton)
228             .addComponent(ownerRadioButton)
229             .addComponent(orderRadioButton))
230         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
231         .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
232         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
233         .addComponent(byLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 12,
javax.swing.GroupLayout.PREFERRED_SIZE)
234         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
235         .addGroup(infoPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

```

```

235         .addComponent(byVehicleRadioButton)
236         .addComponent(byOwnerRadioButton)
237         .addComponent(byOrderRadioButton))
238         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 24,
Short.MAX_VALUE)
239         .addComponent(selectedLabel))
240     );
241
242     searchHeaderLabel.setFont(new java.awt.Font("Tahoma", 1, 14));
243     searchHeaderLabel.setText("Search");
244
245     searchPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter search criteria"));
246     searchPanel.setMaximumSize(new java.awt.Dimension(263, 119));
247
248     searchTextField.addFocusListener(new java.awt.event.FocusAdapter() {
249         public void focusGained(java.awt.event.FocusEvent evt) {
250             searchTextFieldFocusGained(evt);
251         }
252     });
253
254     searchButton.setText("Search");
255     searchButton.addActionListener(new java.awt.event.ActionListener() {
256         public void actionPerformed(java.awt.event.ActionEvent evt) {
257             searchButtonActionPerformed(evt);
258         }
259     });
260
261     cancelButton.setText("Cancel");
262
263     jLabelSearch.setText("Search for ");
264
265     javax.swing.GroupLayout searchPanelLayout = new javax.swing.GroupLayout(searchPanel);
266     searchPanel.setLayout(searchPanelLayout);
267     searchPanelLayout.setHorizontalGroup(
268         searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
269             .addGroup(searchPanelLayout.createSequentialGroup()
270                 .addContainerGap()
271                 .addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
272                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
searchPanelLayout.createSequentialGroup()
273                         .addComponent(searchButton)
274                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 136,
Short.MAX_VALUE)
275                         .addComponent(cancelButton))
276                     .addGroup(searchPanelLayout.createSequentialGroup()
277                         .addComponent(jLabelSearch)
278                         .addContainerGap())
279                 .addContainerGap()
280                 .addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
281                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, searchPanelLayout.createSequentialGroup()
282                         .addContainerGap()
283                         .addComponent(jLabelSearch)
284                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 40,
Short.MAX_VALUE)
285                         .addComponent(searchTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

287         .addGap(18, 18, 18)
288         .addGroup(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
289             .addComponent(searchButton)
290             .addComponent(cancelButton)))
291     );
292
293     searchResultPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Search results"));
294
295     foundSearchResults.setModel(new javax.swing.AbstractListModel() {
296         String[] strings = { "Item 1", "Item 2", "Item 3", "Item 4", "Item 5" };
297         public int getSize() { return strings.length; }
298         public Object getElementAt(int i) { return strings[i]; }
299     });
300     foundSearchResults.setCursor(new java.awt.Cursor(java.awt.Cursor.CROSSHAIR_CURSOR));
301     jScrollPane1.setViewportView(foundSearchResults);
302
303     javax.swing.GroupLayout searchResultPanelLayout = new javax.swing.GroupLayout(searchResultPanel);
304     searchResultPanel.setLayout(searchResultPanelLayout);
305     searchResultPanelLayout.setHorizontalGroup(
306         searchResultPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
307             .addGroup(searchResultPanelLayout.createSequentialGroup()
308                 .addContainerGap()
309                 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 560, Short.MAX_VALUE)
310                 .addContainerGap())
311             );
312     searchResultPanelLayout.setVerticalGroup(
313         searchResultPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
314             .addGroup(searchResultPanelLayout.createSequentialGroup()
315                 .addContainerGap(26, Short.MAX_VALUE)
316                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 130,
javax.swing.GroupLayout.PREFERRED_SIZE)
317                 .addContainerGap())
318             );
319
320     selectButton.setText("Select");
321     selectButton.addActionListener(new java.awt.event.ActionListener() {
322         public void actionPerformed(java.awt.event.ActionEvent evt) {
323             selectButtonActionPerformed(evt);
324         }
325     });
326
327     cancelButton1.setText("Cancel");
328
329     tempLABELbyNYVANG.setText(" ");
330
331     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
332     getContentPane().setLayout(layout);
333     layout.setHorizontalGroup(
334         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
335             .addGroup(layout.createSequentialGroup()
336                 .addContainerGap()
337                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
338                     .addComponent(searchHeaderLabel)
339                     .addGroup(layout.createSequentialGroup()
340                         .addComponent(selectButton)
341                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
342                             .addGroup(layout.createSequentialGroup()

```

```

343         .addGap(18, 18, 18)
344         .addComponent(statusLabel))
345     .addGroup(layout.createSequentialGroup())
346     .addGap(45, 45, 45)
347     .addComponent(tempLABELbyNYVANG)))
348     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 418,
Short.MAX_VALUE)
349     .addComponent(cancelButton1))
350     .addComponent(searchResultPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
351     .addGroup(layout.createSequentialGroup())
352     .addComponent(infoPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
353     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
354     .addComponent(searchPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
355     .addContainerGap())
356 );
357 layout.setVerticalGroup(
358     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
359     .addGroup(layout.createSequentialGroup())
360     .addContainerGap()
361     .addComponent(searchHeaderLabel)
362     .addGap(22, 22, 22)
363     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
364     .addComponent(searchPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
365     .addComponent(infoPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
366     .addGap(15, 15, 15)
367     .addComponent(searchResultPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
368     .addGap(18, 18, 18)
369     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
370     .addComponent(selectButton)
371     .addComponent(statusLabel)
372     .addComponent(cancelButton1)
373     .addComponent(tempLABELbyNYVANG))
374     .addContainerGap(13, Short.MAX_VALUE))
375 );
376
377 pack();
378 }// </editor-fold>
379
380 private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
381
382     // First reset all objects and lists
383     resetAllObjectsAndLists();
384
385     // check what is selected
386     if(vehicleRadioButton.isSelected() == true)
387     {
388         if(byVehicleRadioButton.isSelected() == true)
389         {
390
391             /**
392              * Searchable attributes: LicensePlate, chassisnumber, model, year, & manufacture
393              * Must return vehicleID

```

```

394     *
395     */
396     searchVehicleByVehicle(searchTextField.getText());
397 }
398
399 if(byOrderRadioButton.isSelected() == true)
400 {
401     /**
402     * Searchable attributes; orderID, orderType
403     * Must return: VehicleID (whos attached to the specific order)
404     */
405 }
406
407 if(byOwnerRadioButton.isSelected() == true)
408 {
409     /**
410     * Searchable attributes: ID, Name, Lastname, phone, address
411     * Must return: VehicleID (whos attached to the ownerID)
412     */
413     searchVehicleByOwner(searchTextField.getText());
414 }
415 }
416
417 if(orderRadioButton.isSelected() == true)
418 {
419     if(byVehicleRadioButton.isSelected() == true)
420     {
421         /**
422         * Searchable attributes: LicensePlate, chassisnumber, model, year, & manufacture
423         * Must return: OrderID
424         */
425     }
426
427 if(byOwnerRadioButton.isSelected() == true)
428 {
429     /**
430     * Searchable attributes: ID, Name, Lastname, phone, address
431     * Must return: OrderID
432     */
433 }
434
435 if(byOrderRadioButton.isSelected() == true)
436 {
437     /**
438     * Searchable attributes: ID, Name, Lastname, phone, address
439     * Must return: OrderID
440     */
441     searchOrderByOrder(searchTextField.getText());
442 }
443 }
444
445 if(ownerRadioButton.isSelected() == true)
446 {
447     if(byVehicleRadioButton.isSelected() == true)
448     {
449         /**
450         * Searchable attributes: ID, Name, Lastname, phone, address

```



```

452         * Must return: OwnerID from ownership-table
453         */
454         searchOwnerByVehicle(searchTextField.getText());
455     }
456     if(byOrderRadioButton.isSelected() == true)
457     {
458
459         /**
460         * Searchable attributes: ID, Name, Lastname, phone, address
461         * Must return: OwnerID
462         */
463     }
464     if(byOwnerRadioButton.isSelected() == true)
465     {
466
467         /**
468         * Searchable attributes: ID, Name, Lastname, phone, address
469         * Must return: OwnerID
470         */
471         searchOwnerByOwner(searchTextField.getText());
472     }
473 }
474
475
476 }
477
478 private void vehicleRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
479     selectedLabel.setText(VEHICLE);
480     enableSlaveGroup();
481 }
482
483 private void ownerRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
484     selectedLabel.setText(OWNER);
485     enableSlaveGroup();
486 }
487
488 private void orderRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
489     selectedLabel.setText(ORDER);
490     enableSlaveGroup();
491 }
492
493 private void byVehicleRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
494
495     if(selectedLabel.getText().contains("Vehicle"))
496     {
497         selectedLabel.setText(VEHICLE + " by Vehicle info");
498     }
499     if(selectedLabel.getText().contains("Owner"))
500     {
501         selectedLabel.setText(OWNER + " by Owner info");
502     }
503     if(selectedLabel.getText().contains("Order"))
504     {
505         selectedLabel.setText(ORDER + " by Order info");
506     }
507 }
508
509 private void byOwnerRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {

```



```

510     if(selectedLabel.getText().contains("Vehicle"))
511     {
512         selectedLabel.setText(VEHICLE + " by Vehicle info");
513     }
514     if(selectedLabel.getText().contains("Owner"))
515     {
516         selectedLabel.setText(OWNER + " by Owner info");
517     }
518     if(selectedLabel.getText().contains("Order"))
519     {
520         selectedLabel.setText(ORDER + " by Order info");
521     }
522 }
523
524 private void byOrderRadioButtonActionPerformed(java.awt.event.ActionEvent evt) {
525     if(selectedLabel.getText().contains("Vehicle"))
526     {
527         selectedLabel.setText(VEHICLE + " by Order info");
528     }
529     if(selectedLabel.getText().contains("Owner"))
530     {
531         selectedLabel.setText(OWNER + " by Order info");
532     }
533     if(selectedLabel.getText().contains("Order"))
534     {
535         selectedLabel.setText(ORDER + " by Order info");
536     }
537 }
538
539 private void infoPanelFocusLost(java.awt.event.FocusEvent evt) {
540     jLabelSearch.setText(selectedLabel.getText());
541 }
542
543 private void searchTextFieldFocusGained(java.awt.event.FocusEvent evt) {
544     // TODO add your handling code here:
545 }
546
547 private void selectButtonActionPerformed(java.awt.event.ActionEvent evt) {
548
549     // If search for a vehicle
550     if(selectedVehicle!= null)
551     {
552         if(foundSearchResults.getSelectedIndex() >= 0)
553         {
554             int indexPositionOfSelectedVehicle = foundSearchResults.getSelectedIndex();
555             vehicleID = arrActFoundsVehicleID[indexPositionOfSelectedVehicle];
556
557             boolean found = false;
558             int i = 0;
559             Vehicle v = null;
560
561             // Search for the right selected vehilce in the found vehicles
562             while(i < actFoundsVehicleObjects.size() && !found)
563             {
564                 v = actFoundsVehicleObjects.get(i);
565                 if(v.getVehicleID().equals(vehicleID))
566                 {
567                     found = true;

```

```

568         selectedVehicle = actFoundsVehicleObjects.get(i);
569         foundID = selectedVehicle.getVehicleID();
570     } // End inner if
571     else
572     {
573         i++;
574     } // End inner else
575 } // End while
576 }
577 vehicleID = selectedVehicle.getVehicleID();
578 tempLABELbyNYVANG.setText(vehicleID);
579 }
580 else if(selectedOwner!=null)
581 {
582
583 }
584
585 //!!!!!!! MUST CHANGE RETURN TYPE TO STRING AS WE NEED A ID RETURNED
!!!!!!!/
586 }
587 }
588
589 /**
590  * Enables the secondary buttonGroup
591  */
592 private void enableSlaveGroup()
593 {
594     byLabel.setText("Search by");
595     byOrderRadioButton.setEnabled(true);
596     byOwnerRadioButton.setEnabled(true);
597     byVehicleRadioButton.setEnabled(true);
598 }
599 /**
600  * @param args the command line arguments
601  */
602 public static void main(String args[])
603 {
604     java.awt.EventQueue.invokeLater(new Runnable() {
605         public void run() {
606             new SuperSearch().setVisible(true);
607         }
608     });
609 }
610
611 public void run()
612 {
613     new SuperSearch().setVisible(true);
614 }
615
616 /**
617  * Reset all selected objects and all lists
618  */
619 private void resetAllObjectsAndLists()
620 {
621
622     selectedOwner = null;
623     selectedVehicle = null;
624     selectedOrder = null;

```

```

625 }
626
627 ////////////////////////////////////////////////// ALL SEARCH COMBINATIONS START HERE //////////////////////////////////////////
628
629 ////////////////////////////////// SEARCH FOR AN ORDER BY ORDER////////////////////////////////
630 private void searchOrderByOrder(String searchCriteria)
631 {
632     searchCriteria = valueOfSearchField;
633
634     //Set the arrays with the searchResults
635     enterOrderSearchDetails(searchCriteria);
636     //If nothing is found, display a label to the user
637     if(arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length==0)
638     {
639         statusLabel.setText("No searchresults");
640         statusLabel.setForeground(Color.red);
641         foundSearchResults.setModel(new javax.swing.AbstractListModel()
642         {
643             public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
644             public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
645         });
646     } //end if
647
648     //If there's results present them.
649     else
650     {
651         //Clear the actual JList and make a new one
652         statusLabel.setText("");
653         foundSearchResults.setModel(new javax.swing.AbstractListModel()
654         {
655             public int getSize() { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.length; }
656             public Object getElementAt(int i) { return
arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice[i]; }
657         });
658         searchTextField.setText("");
659         selectButton.setEnabled(true);
660     } //end else
661
662 }
663
664
665
666
667 /**
668  * This method ask the controller-layer to perform a search for an owner
669  * Search results are set from the results of the controller-layer as multiple
670  * lists of Owner-objects, strings and ints.
671  *
672  * All variables except startDate, expEndDate, and orderID in every object in the
673  * resultFoundObjects are put together as a string and inserted to an
674  * arrayList(resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice) which will be converted
675  * to an array.
676  *
677  * The same applies to the list of Integer (startDate, expEndDate, and orderID)
678  * @param searchCriteria

```

```

679  * @return void
680  */
681  public void enterOrderSearchDetails(String searchCriteria)
682  {
683      //This ArrayList will hold the Strings of the search results
684      resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice = new ArrayList<String>();
685      //This ArrayList will hold the OrderID of the search results
686      resultFoundOrderID = new ArrayList<String>();
687      //This ArrayList will hold the Order-objects of the search results
688      resultFoundOrderObjects = new ArrayList<Order>();
689
690      resultFoundOrderObjects = oc.searchOrder(searchCriteria);
691
692      int i = 0;
693      for (Order o : resultFoundOrderObjects)
694      {
695          //Strings and orderID goes to the respective ArrayLists
696          resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.add("Order Number: \"" +
o.getOrderID()+ "\" " + "Order Type: \"" + o.getOrderType() + "\" Description: \"" + o.getDescriptionOrder());
697          resultFoundOrderID.add(o.getOrderID());
698          i++;
699      } //end for each
700
701      //Here the ArrayList is converted to arrays for easier presentation
702      arrResultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice =
resultFoundOrderTypeDescOrderDescSpareStartDateEndDateExpPrice.toArray(new String[0]);
703      arrResultFoundOrderID = resultFoundOrderID.toArray(new String[0]);
704
705  }
706
707  ////////////////////////////////// SEARCH FOR A VEHICLE BY VEHICLE //////////////////////////////////
708  private void searchVehicleByVehicle(String searchCriteria)
709  {
710      enterVehicleSearchDetails(searchCriteria);
711      //If nothing is found, display a label to the user
712      if (arrActFoundsBrandModel.length == 0) {
713          foundSearchResults.setModel(new javax.swing.AbstractListModel() {
714
715              public int getSize() {
716                  return arrActFoundsBrandModel.length;
717              }
718
719              public Object getElementAt(int i) {
720                  return arrActFoundsBrandModel[i];
721              }
722          });
723      } //end if
724      //If there's results present them.
725      else {
726          //Clear the actual JList and make a new one
727
728          foundSearchResults.setModel(new javax.swing.AbstractListModel() {
729
730              public int getSize() {
731                  return arrActFoundsBrandModel.length;
732              }
733
734              public Object getElementAt(int i) {

```

```

735         return arrActFoundsBrandModel[i];
736     }
737 });
738 } //end else
739 }
740
741 ////////////////////////////////////////////////// SEARCH FOR A OWNER BY VEHICLE //////////////////////////////////////
742
743 public void searchOwnerByVehicle(String searchCriteria)
744 {
745     enterVehicleSearchDetails(searchCriteria);
746     if(actFoundsVehicleObjects.size()==1)
747     {
748
749         vehicleID = actFoundsVehicleObjects.get(0).getVehicleID();
750
751         enterOwnerByVehicleSearchDetails(vehicleID);
752         //If nothing is found, display a label to the user
753         if (arrActFoundsFirstnameLastNamePhoneNumber.length == 0) {
754             foundSearchResults.setModel(new javax.swing.AbstractListModel() {
755
756                 public int getSize() {
757                     return arrActFoundsFirstnameLastNamePhoneNumber.length;
758                 }
759
760                 public Object getElementAt(int i) {
761                     return arrActFoundsFirstnameLastNamePhoneNumber[i];
762                 }
763             });
764         } //end if
765         //If there's results present them.
766         else
767         {
768             //Clear the actual JList and make a new one
769
770             foundSearchResults.setModel(new javax.swing.AbstractListModel() {
771
772                 public int getSize() {
773                     return arrActFoundsFirstnameLastNamePhoneNumber.length;
774                 }
775
776                 public Object getElementAt(int i) {
777                     return arrActFoundsFirstnameLastNamePhoneNumber[i];
778                 }
779             });
780         } //end else
781     }
782 }
783 }
784
785 /**
786  * This method ask the controller-layer to perform a search for an owner
787  * Search results are set from the results of the controller-layer as multiple
788  * lists of Owner-objects, strings and ints.
789  *
790  * The variables firstName, lastName, and phoneNumber in every object in the
791  * actFoundsObjects is put together as a string and inserted to an
792  * arraylist(actFoundsFirstnameLastNamePhoneNumber) which will be converted

```

```

793  * to an array.
794  *
795  * The same applies to the list of OwnerIDs
796  * @param searchCriteria
797  * @return void
798  */
799  public void enterOwnerByVehicleSearchDetails(String vehicleID)
800  {
801      //This ArrayList will hold the Firstname, lastname and phonenumber of the search results
802      actFoundsFirstnameLastNamePhoneNumber = new ArrayList<String>();
803      //This ArrayList will hold the OwnerID of the search results
804      actFoundsOwnerID = new ArrayList<String>();
805      //This ArrayList will hold the Owner-objects of the search results
806      actFoundsOwnerObjects = new ArrayList<Owner>();
807
808      actFoundsOwnerObjects = mvc.superSearchVehicleWithOwner(vehicleID);
809
810      int i = 0;
811      for (Owner o : actFoundsOwnerObjects)
812      {
813          //ID, Firstname, lastname, phonenumber and ownerID goes to the respective ArrayLists
814          actFoundsFirstnameLastNamePhoneNumber.add("(" + o.getID() + ") " + o.getFirstName() + " " +
o.getLastName() + " (" + o.getPhone() + ") ");
815          actFoundsOwnerID.add(o.getID());
816          i++;
817      } //end for each
818
819      //Here the ArrayList is converted to arrays (1-dim) (for presentation purpose)
820      arrActFoundsFirstnameLastNamePhoneNumber = actFoundsFirstnameLastNamePhoneNumber.toArray(new
String[0]);
821      arrActFoundsOwnerID = actFoundsOwnerID.toArray(new String[0]);
822
823  }
824
825
826
827  /**
828   * This method ask the controller-layer to perform a search for a vehicle
829   * Search results are set from the controller-layer as multiply list of Vehicle-objects, String and ints
830   *
831   * The variables model and vType in every object in the actFoundsVehicleObjects list
832   * is put together as a string and inserted to an
833   * arraylist( actFoundsBrandModel ) which will be converted to an array
834   *
835   * The same applies to the list of vehilce id
836   * @param searchCriteria
837   * @return void
838   */
839  public void enterVehicleSearchDetails(String searchCriteria)
840  {
841      // This arrayLsit will hold the vehicles Brand and Model for presentation
842      actFoundsBrandModel = new ArrayList<String>();
843
844      // This arrayList will hold the vehicles ID
845      actFoundVehicle = new ArrayList<String>();
846
847      // This arrayList will hold the vehicle object
848      actFoundsVehicleObjects = mvc.searchVehicle(searchCriteria);

```

```

849     int i = 0;
850     for (Vehicle v : actFoundsVehicleObjects)
851     {
852         // Brand, model and id in the respective lists
853         String licensPlate = v.getLicensPlate();
854         String chassisNumber = v.getChassisNumber();
855
856         // If both licensplate and chassis number is empty
857         if(licensPlate.equals("") && chassisNumber.equals(""))
858         {
859             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel());
860         }
861         // If only chassisnumber is empty
862         else if(!licensPlate.equals("") && chassisNumber.equals(""))
863         {
864             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
865         }
866         // If only licensplate is empty
867         else if(licensPlate.equals("") && !chassisNumber.equals(""))
868         {
869             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getChassisNumber() + " )");
870         }
871         // Neither licensplate or chassisnumber is empty
872         else
873         {
874             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
875         }
876         actFoundVehicle.add(v.getVehicleID());
877         i++;
878     } //end for each
879
880
881     arrActFoundsBrandModel = actFoundsBrandModel.toArray(new String[0]);
882     arrActFoundsVehicleID = actFoundVehicle.toArray(new String[0]);
883
884 }
885
886 ////////////////////////////////////////////////// SEARCH VEHICLE BY OWNER //////////////////////////////////////
887
888 private void searchVehicleByOwner(String searchCriteria)
889 {
890     enterOwnerSearchDetails(searchCriteria);
891     if(actFoundsOwnerObjects.size()==1)
892     {
893
894         ownerID = actFoundsOwnerObjects.get(0).getID();
895
896         enterVehicleByOwnerSearchDetails(ownerID);
897         //If nothing is found, display a label to the user
898         if (arrActFoundsBrandModel.length == 0) {
899             foundSearchResults.setModel(new javax.swing.AbstractListModel() {
900
901                 public int getSize() {
902                     return arrActFoundsBrandModel.length;
903                 }

```

```

904
905     public Object getElementAt(int i) {
906         return arrActFoundsBrandModel[i];
907     }
908 });
909 }//end if
910 //If there's results present them.
911 else {
912     //Clear the actual JList and make a new one
913
914     foundSearchResults.setModel(new javax.swing.AbstractListModel() {
915
916         public int getSize() {
917             return arrActFoundsBrandModel.length;
918         }
919
920         public Object getElementAt(int i) {
921             return arrActFoundsBrandModel[i];
922         }
923     });
924 }//end else
925 }
926 else
927 {
928
929
930 }
931 }
932
933
934 /**
935  * This method ask the controller-layer to perform a search for a vehicle
936  * Search results are set from the controller-layer as multiply list of Vehicle-objects, String and ints
937  *
938  * The variables model and vType in every object in the actFoundsVehicleObjects list
939  * is put together as a string and inserted to an
940  * arraylist( actFoundsBrandModel ) which will be converted to an array
941  *
942  * The same applies to the list of vehilce id
943  * @param searchCriteria
944  * @return void
945  */
946 public void enterVehicleByOwnerSearchDetails(String ownerID)
947 {
948     // This arrayLsit will hold the vehicles Brand and Model for presentation
949     actFoundsBrandModel = new ArrayList<String>();
950
951     // This arrayList will hold the vehicles ID
952     actFoundVehicle = new ArrayList<String>();
953
954     // This arrayList will hold the vehicle object
955     actFoundsVehicleObjects = moc.searchOwnerWithVehicle(ownerID);
956     int i = 0;
957     for (Vehicle v : actFoundsVehicleObjects)
958     {
959         // Brand, model and id in the respective lists
960         String licensPlate = v.getLicensPlate();
961         String chassisNumber = v.getChassisNumber();

```



```

962
963 // If both licensplate and chassis number is empty
964 if(licensPlate.equals("") && chassisNumber.equals(""))
965 {
966     actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel());
967 }
968 // If only chassisnumber is empty
969 else if(!licensPlate.equals("") && chassisNumber.equals(""))
970 {
971     actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
972 }
973 // If only licensplate is empty
974 else if(licensPlate.equals("") && !chassisNumber.equals(""))
975 {
976     actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getChassisNumber() + " )");
977 }
978 // Neither licensplate or chassisnumber is empty
979 else
980 {
981     actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
982 }
983 actFoundVehicle.add(v.getVehicleID());
984 i++;
985 }//end for each
986
987
988 arrActFoundsBrandModel = actFoundsBrandModel.toArray(new String[0]);
989 arrActFoundsVehicleID = actFoundVehicle.toArray(new String[0]);
990
991 }
992
993
994
995 ////////////////////////////////////////////////// SEARCH FOR AN OWNER BY OWNER////////////////////////////////////
996
997 /**
998  * Search for an owner by owner
999  * @param searchCriteria criteria to search with
1000  */
1001 private void searchOwnerByOwner(String searchCriteria)
1002 {
1003     enterOwnerSearchDetails(searchCriteria);
1004     //If nothing is found, display a label to the user
1005     if (arrActFoundsFirstnameLastNamePhoneNumber.length == 0) {
1006         foundSearchResults.setModel(new javax.swing.AbstractListModel() {
1007
1008             public int getSize() {
1009                 return arrActFoundsFirstnameLastNamePhoneNumber.length;
1010             }
1011
1012             public Object getElementAt(int i) {
1013                 return arrActFoundsFirstnameLastNamePhoneNumber[i];
1014             }
1015         });
1016     }//end if

```

```

1017 //If there's results present them.
1018 else {
1019     //Clear the actual JList and make a new one
1020
1021     foundSearchResults.setModel(new javax.swing.AbstractListModel() {
1022
1023         public int getSize() {
1024             return arrActFoundsFirstnameLastNamePhoneNumber.length;
1025         }
1026
1027         public Object getElementAt(int i) {
1028             return arrActFoundsFirstnameLastNamePhoneNumber[i];
1029         }
1030     });
1031 } //end else
1032 }
1033
1034 /**
1035  * This method ask the controller-layer to perform a search for an owner
1036  * Search results are set from the results of the controller-layer as multiple
1037  * lists of Owner-objects, strings and ints.
1038  *
1039  * The variables firstName, lastName, and phoneNumber in every object in the
1040  * actFoundsObjects is put together as a string and inserted to an
1041  * arraylist(actFoundsFirstnameLastNamePhoneNumber) which will be converted
1042  * to an array.
1043  *
1044  * The same applies to the list of OwnerIDs
1045  * @param searchCriteria
1046  * @return void
1047  */
1048 public void enterOwnerSearchDetails(String searchCriteria)
1049 {
1050     //This ArrayList will hold the Firstname, lastname and phonenumber of the search results
1051     actFoundsFirstnameLastNamePhoneNumber = new ArrayList<String>();
1052     //This ArrayList will hold the OwnerID of the search results
1053     actFoundsOwnerID = new ArrayList<String>();
1054     //This ArrayList will hold the Owner-objects of the search results
1055     actFoundsOwnerObjects = new ArrayList<Owner>();
1056
1057     actFoundsOwnerObjects = moc.searchOwner(searchCriteria);
1058
1059     int i = 0;
1060     for (Owner o : actFoundsOwnerObjects)
1061     {
1062         //ID, Firstname, lastname, phonenumber and ownerID goes to the respective ArrayLists
1063         actFoundsFirstnameLastNamePhoneNumber.add("(" + o.getID() + ") " + o.getFirstName() + " " +
1064         o.getLastName() + " (" + o.getPhone() + ") ");
1065         actFoundsOwnerID.add(o.getID());
1066         i++;
1067     } //end for each
1068
1069     //Here the ArrayList is converted to arrays (1-dim) (for presentation purpose)
1070     arrActFoundsFirstnameLastNamePhoneNumber = actFoundsFirstnameLastNamePhoneNumber.toArray(new
1071     String[0]);
1072     arrActFoundsOwnerID = actFoundsOwnerID.toArray(new String[0]);
1073 }

```

```

1073
1074
1075
1076
1077
1078 // Variables declaration - do not modify
1079 private static javax.swing.ButtonGroup buttonGroupMaster;
1080 private static javax.swing.ButtonGroup buttonGroupSlave;
1081 private static javax.swing.JLabel byLabel;
1082 private static javax.swing.JRadioButton byOrderRadioButton;
1083 private static javax.swing.JRadioButton byOwnerRadioButton;
1084 private static javax.swing.JRadioButton byVehicleRadioButton;
1085 private static javax.swing.JButton cancelButton;
1086 private static javax.swing.JButton cancelButton1;
1087 private static javax.swing.JList foundSearchResults;
1088 private static javax.swing.JPanel infoPanel;
1089 private static javax.swing.JLabel jLabelSearch;
1090 private static javax.swing.JScrollPane jScrollPane1;
1091 private static javax.swing.JSeparator jSeparator1;
1092 private static javax.swing.JRadioButton orderRadioButton;
1093 private static javax.swing.JRadioButton ownerRadioButton;
1094 private static javax.swing.JButton searchButton;
1095 private static javax.swing.JLabel searchHeaderLabel;
1096 private static javax.swing.JPanel searchPanel;
1097 private static javax.swing.JPanel searchResultPanel;
1098 private static javax.swing.JTextField searchTextField;
1099 private static javax.swing.JButton selectButton;
1100 private static javax.swing.JLabel selectedLabel;
1101 private static javax.swing.JLabel statusLabel;
1102 protected static javax.swing.JLabel tempLABELbyNYVANG;
1103 private static javax.swing.JRadioButton vehicleRadioButton;
1104 // End of variables declaration

```

UploadMediaView

```

1 /*
2  * This class is to upload media to a vehicle
3  * It takes in the parameters WSGUI - to implement to workstation gui belong to the vehicle
4  * it also takes vehicleID - to know which vehicle all the medias belongs to
5  * it also takes workstationID - to know which workstation the vehicle belongs to
6  * it also takes orderID - to know which order the medias belongs to
7  * At last it takes resultType - to know what kind of result it is
8  */
9
10 /*
11  * UploadMediaView.java
12  *
13  * Created on 14-12-2010, 11:17:24
14  */
15
16 package tweakmc.view;
17
18 import tweakmc.utility.GUIHelpUtil;
19 import java.awt.event.KeyEvent;
20 import javax.swing.JFileChooser;
21 import javax.swing.JOptionPane;
22 import tweakmc.control.ManageResultController;
23 import tweakmc.model.Result.Result;
24 import tweakmc.utility.CheckInput;

```

```

25 import tweakmc.utility.FileHandlerUtility;
26
27 /**
28  *
29  * @author clausPallisgaardBeck
30  */
31 public class UploadMediaView extends javax.swing.JFrame
32 {
33     //Instance variables
34     private String vehicleID;
35     private String workstationID;
36     private String orderID;
37     private String resultType;
38     private WSGUI wsgui;
39     private String originalDestination;
40
41
42
43     private final String IMAGE = "Image";
44     private final String VIDEO = "Video";
45     private final String SOUND = "Sound";
46     private final String DOCUMENT = "Document";
47
48
49     /** Creates new form UploadMediaView */
50     public UploadMediaView(WSGUI wsgui, String vehicleID, String workstationID, String orderID, String
resultType)
51     {
52         this.wsgui = wsgui;
53         this.vehicleID = vehicleID;
54         this.workstationID = workstationID;
55         this.orderID = orderID;
56         this.resultType = resultType;
57
58         setUndecorated(true);
59         initComponents();
60         setVisible(true);
61     } //end construtor UploadMediaView()
62
63     /**
64      * Build and uploadview for already existing medias, possible to enter new comments
65      * @param wsgui the actual WS gui window
66      * @param r the result to get information from
67      */
68     public UploadMediaView(WSGUI wsgui, Result r)
69     {
70         this.wsgui = wsgui;
71         this.vehicleID = r.getVehicleID();
72         this.workstationID = r.getWorkstationID();
73         this.orderID = r.getOrderID();
74         this.resultType = r.getResultType();
75
76         setUndecorated(true);
77         initComponents();
78         findUploadFileButton.setEnabled(false);
79         previewUploadButton.setEnabled(true);
80         originalDestination = r.getPath();
81         mediaNameTextField.setText(r.getResultName());

```

```

82     mediaNameTField.setEditable(false);
83     mechanicCommentTArea.setText(r.getMechanicComment());
84     customerCommentTArea.setText(r.getCustomerComment());
85     setVisible(true);
86 } //end edit constructor
87
88 /** This method is called from within the constructor to
89  * initialize the form.
90  * WARNING: Do NOT modify this code. The content of this method is
91  * always regenerated by the Form Editor.
92  */
93 @SuppressWarnings("unchecked")
94 // <editor-fold defaultstate="collapsed" desc="Generated Code">
95 private void initComponents() {
96
97     mainUploadPanel = new javax.swing.JPanel();
98     mainScrollPane = new javax.swing.JScrollPane();
99     contentPanel = new javax.swing.JPanel();
100    headerPanel = new javax.swing.JPanel();
101    headerLabel = new javax.swing.JLabel();
102    commentPanel = new javax.swing.JPanel();
103    mechanicCommentScrollPane = new javax.swing.JScrollPane();
104    mechanicCommentTArea = new javax.swing.JTextArea();
105    customerCommentScrollPane = new javax.swing.JScrollPane();
106    customerCommentTArea = new javax.swing.JTextArea();
107    mechanicCommentLabel = new javax.swing.JLabel();
108    customerCommentLabel = new javax.swing.JLabel();
109    buttonPanel = new javax.swing.JPanel();
110    saveResultButton = new javax.swing.JButton();
111    cancelResultButton = new javax.swing.JButton();
112    saveAsLabel = new javax.swing.JLabel();
113    mediaNameTField = new javax.swing.JTextField();
114    manageUploadPanel = new javax.swing.JPanel();
115    findUploadFileButton = new javax.swing.JButton();
116    uploadTypeLabel = new javax.swing.JLabel();
117    uploadTypeResultLabel = new javax.swing.JLabel();
118    previewUploadButton = new javax.swing.JButton();
119    workstationNameLabel = new javax.swing.JLabel();
120    workstationNameResultLabel = new javax.swing.JLabel();
121    vehicleIDLabel = new javax.swing.JLabel();
122    orderIDLabel = new javax.swing.JLabel();
123    vehicleIDResultLabel = new javax.swing.JLabel();
124    orderIDResultLabel = new javax.swing.JLabel();
125    if(orderID.equalsIgnoreCase("C-1"))
126    {
127        orderIDResultLabel.setText("No orderID for this upload!");
128    }
129    else
130    {
131        orderIDResultLabel.setText(orderID);
132    }
133
134    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
135
136    headerPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
137
138    headerLabel.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

```

```

139     headerLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
140     headerLabel.setText("Upload media to vehicleID: " + vehicleID + " on " + wsgui.getStationName() );
141
142     javax.swing.GroupLayout headerPanelLayout = new javax.swing.GroupLayout(headerPanel);
143     headerPanel.setLayout(headerPanelLayout);
144     headerPanelLayout.setHorizontalGroup(
145         headerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
146             .addGroup(headerPanelLayout.createSequentialGroup()
147                 .addContainerGap()
148                 .addComponent(headerLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 554,
javax.swing.GroupLayout.PREFERRED_SIZE)
149                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
150             );
151     headerPanelLayout.setVerticalGroup(
152         headerPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
153             .addGroup(headerPanelLayout.createSequentialGroup()
154                 .addContainerGap()
155                 .addComponent(headerLabel)
156                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
157             );
158
159     commentPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
160
161     mechanicCommentTextArea.setColumns(20);
162     mechanicCommentTextArea.setRows(5);
163     mechanicCommentScrollPane.setViewportView(mechanicCommentTextArea);
164
165     customerCommentTextArea.setColumns(20);
166     customerCommentTextArea.setRows(5);
167     customerCommentScrollPane.setViewportView(customerCommentTextArea);
168
169     mechanicCommentLabel.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
170     mechanicCommentLabel.setText("Mechanic Comment:");
171
172     customerCommentLabel.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
173     customerCommentLabel.setText("Customer Comment:");
174
175     javax.swing.GroupLayout commentPanelLayout = new javax.swing.GroupLayout(commentPanel);
176     commentPanel.setLayout(commentPanelLayout);
177     commentPanelLayout.setHorizontalGroup(
178         commentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
179             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
commentPanelLayout.createSequentialGroup()
180                 .addContainerGap()
181                 .addGroup(commentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
182                     .addComponent(mechanicCommentScrollPane, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 246, Short.MAX_VALUE)
183                     .addComponent(customerCommentScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 246,
Short.MAX_VALUE)
184                     .addComponent(mechanicCommentLabel, javax.swing.GroupLayout.Alignment.LEADING)
185                     .addComponent(customerCommentLabel, javax.swing.GroupLayout.Alignment.LEADING))
186                 .addContainerGap())
187             );
188     commentPanelLayout.setVerticalGroup(
189         commentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
190             .addGroup(commentPanelLayout.createSequentialGroup()
191                 .addContainerGap()

```

```

192     .addComponent(mechanicCommentLabel)
193     .addGap(7, 7, 7)
194     .addComponent(mechanicCommentScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)
195     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
196     .addComponent(customerCommentLabel)
197     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
198     .addComponent(customerCommentScrollPane, javax.swing.GroupLayout.PREFERRED_SIZE, 123,
javax.swing.GroupLayout.PREFERRED_SIZE)
199     .addContainerGap(29, Short.MAX_VALUE))
200 );
201
202
buttonPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));
203
204 saveResultButton.setText("Save result");
205 saveResultButton.addActionListener(new java.awt.event.ActionListener() {
206     public void actionPerformed(java.awt.event.ActionEvent evt) {
207         saveResultButtonActionPerformed(evt);
208     }
209 });
210 saveResultButton.addKeyListener(new java.awt.event.KeyAdapter() {
211     public void keyPressed(java.awt.event.KeyEvent evt) {
212         saveResultButtonKeyPressed(evt);
213     }
214 });
215
216 cancelResultButton.setText("Cancel");
217 cancelResultButton.addActionListener(new java.awt.event.ActionListener() {
218     public void actionPerformed(java.awt.event.ActionEvent evt) {
219         cancelResultButtonActionPerformed(evt);
220     }
221 });
222 cancelResultButton.addKeyListener(new java.awt.event.KeyAdapter() {
223     public void keyPressed(java.awt.event.KeyEvent evt) {
224         cancelResultButtonKeyPressed(evt);
225     }
226 });
227
228 saveAsLabel.setText("Save as (name):");
229
230 javax.swing.GroupLayout buttonPanelLayout = new javax.swing.GroupLayout(buttonPanel);
231 buttonPanel.setLayout(buttonPanelLayout);
232 buttonPanelLayout.setHorizontalGroup(
233     buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
234     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, buttonPanelLayout.createSequentialGroup()
235         .addContainerGap()
236         .addComponent(saveAsLabel)
237         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
238         .addComponent(mediaNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 285,
Short.MAX_VALUE)
239         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
240         .addComponent(saveResultButton)
241         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
242         .addComponent(cancelResultButton)
243         .addContainerGap()
244     );
245

```



```

246     buttonPanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[]
{ cancelResultButton, saveResultButton});
247
248     buttonPanelLayout.setVerticalGroup(
249         buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
250         .addGroup(buttonPanelLayout.createSequentialGroup()
251             .addContainerGap()
252             .addGroup(buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
253                 .addComponent(cancelResultButton)
254                 .addComponent(saveResultButton)
255                 .addComponent(saveAsLabel)
256                 .addComponent(mediaNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
257             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
258         );
259
260 manageUploadPanel.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISE
D));
261
262     findUploadFileButton.setText("Find file to upload");
263     findUploadFileButton.addActionListener(new java.awt.event.ActionListener() {
264         public void actionPerformed(java.awt.event.ActionEvent evt) {
265             findUploadFileButtonActionPerformed(evt);
266         }
267     });
268     findUploadFileButton.addKeyListener(new java.awt.event.KeyAdapter() {
269         public void keyPressed(java.awt.event.KeyEvent evt) {
270             findUploadFileButtonKeyPressed(evt);
271         }
272     });
273
274     uploadTypeLabel.setText("Upload media type:");
275
276     uploadTypeResultLabel.setText(" " + resultType);
277
278     previewUploadButton.setText("Preview upload");
279     previewUploadButton.setEnabled(false);
280     previewUploadButton.addActionListener(new java.awt.event.ActionListener() {
281         public void actionPerformed(java.awt.event.ActionEvent evt) {
282             previewUploadButtonActionPerformed(evt);
283         }
284     });
285     previewUploadButton.addKeyListener(new java.awt.event.KeyAdapter() {
286         public void keyPressed(java.awt.event.KeyEvent evt) {
287             previewUploadButtonKeyPressed(evt);
288         }
289     });
290
291     workstationNameLabel.setText("Workstation:");
292
293     workstationNameResultLabel.setText(wsgui.getStationName());
294
295     vehicleIDLabel.setText("VehicleID:");
296
297     orderIDLabel.setText("OrderID:");
298
299     vehicleIDResultLabel.setText(" " + vehicleID);

```



```

300
301     javax.swing.GroupLayout manageUploadPanelLayout = new
javax.swing.GroupLayout(manageUploadPanel);
302     manageUploadPanel.setLayout(manageUploadPanelLayout);
303     manageUploadPanelLayout.setHorizontalGroup(
304         manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
305         .addGroup(manageUploadPanelLayout.createSequentialGroup()
306             .addContainerGap()
307             .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
308                 .addGroup(manageUploadPanelLayout.createSequentialGroup()
309                     .addComponent(findUploadFileButton)
310                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
311                     .addComponent(previewUploadButton))
312                 .addGroup(manageUploadPanelLayout.createSequentialGroup()
313                     .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
314                         .addComponent(uploadTypeLabel, javax.swing.GroupLayout.Alignment.TRAILING)
315                         .addComponent(workStationNameLabel, javax.swing.GroupLayout.Alignment.TRAILING)
316                         .addComponent(vehicleIDLabel, javax.swing.GroupLayout.Alignment.TRAILING)
317                         .addComponent(orderIDLabel, javax.swing.GroupLayout.Alignment.TRAILING))
318                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
319                         .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
320                             .addComponent(orderIDResultLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 179,
Short.MAX_VALUE)
321                             .addComponent(workstationNameResultLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 179,
Short.MAX_VALUE)
322                             .addComponent(uploadTypeResultLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 179,
Short.MAX_VALUE)
323                             .addComponent(vehicleIDResultLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 179,
Short.MAX_VALUE))))))
324                     .addContainerGap()
325                 );
326
327     manageUploadPanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] { findUploadFileButton, previewUploadButton });
328
329     manageUploadPanelLayout.setVerticalGroup(
330         manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
331         .addGroup(manageUploadPanelLayout.createSequentialGroup()
332             .addContainerGap()
333             .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
334                 .addComponent(findUploadFileButton)
335                 .addComponent(previewUploadButton))
336             .addGap(32, 32, 32)
337             .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
338                 .addComponent(uploadTypeLabel)
339                 .addComponent(uploadTypeResultLabel))
340             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
341             .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
342                 .addComponent(workStationNameLabel)
343                 .addComponent(workstationNameResultLabel))
344             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

345 .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
346     .addComponent(vehicleIDLabel)
347     .addComponent(vehicleIDResultLabel))
348     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
349
350 .addGroup(manageUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
351     .addComponent(orderIDLabel)
352     .addComponent(orderIDResultLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE))
353     .addContainerGap(203, Short.MAX_VALUE))
354 );
355 javax.swing.GroupLayout contentPanelLayout = new javax.swing.GroupLayout(contentPanel);
356 contentPanel.setLayout(contentPanelLayout);
357 contentPanelLayout.setHorizontalGroup(
358     contentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
359     .addGroup(contentPanelLayout.createSequentialGroup()
360         .addContainerGap()
361         .addGroup(contentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.CENTER)
362             .addComponent(headerPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
363             .addGroup(contentPanelLayout.createSequentialGroup()
364                 .addComponent(manageUploadPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
365                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
366                 .addComponent(commentPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
367                 .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
368             .addGap(122, 122, 122))
369     );
370 contentPanelLayout.setVerticalGroup(
371     contentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
372     .addGroup(contentPanelLayout.createSequentialGroup()
373         .addContainerGap()
374         .addComponent(headerPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
375         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
376         .addGroup(contentPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
377             .addComponent(commentPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
378             .addComponent(manageUploadPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
379         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
380         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
381         .addGap(63, 63, 63))
382     );
383
384 mainScrollPane.setViewportViewView(contentPanel);
385
386 javax.swing.GroupLayout mainUploadPanelLayout = new javax.swing.GroupLayout(mainUploadPanel);
387 mainUploadPanel.setLayout(mainUploadPanelLayout);
388 mainUploadPanelLayout.setHorizontalGroup(
389     mainUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
390     .addComponent(mainScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 712, Short.MAX_VALUE)
391 );

```

```

392     mainUploadPanelLayout.setVerticalGroup(
393         mainUploadPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
394         .addComponent(mainScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 527, Short.MAX_VALUE)
395     );
396
397     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
398     getContentPane().setLayout(layout);
399     layout.setHorizontalGroup(
400         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
401         .addComponent(mainUploadPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
402     );
403     layout.setVerticalGroup(
404         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
405         .addComponent(mainUploadPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
406     );
407
408     pack();
409 } // </editor-fold>
410
411 private void cancelResultButtonActionPerformed(java.awt.event.ActionEvent evt) {
412     cancelAndClose();
413 }
414
415 private void cancelResultButtonKeyPressed(java.awt.event.KeyEvent evt) {
416     if(evt.getKeyCode() == KeyEvent.VK_ENTER)
417     {
418         cancelAndClose();
419     } //end if
420 }
421
422 private void saveResultButtonActionPerformed(java.awt.event.ActionEvent evt) {
423     saveResult();
424 }
425
426 private void saveResultButtonKeyPressed(java.awt.event.KeyEvent evt) {
427     if(evt.getKeyCode() == KeyEvent.VK_ENTER)
428     {
429         saveResult();
430     } //end if
431 }
432
433 private void findUploadFileButtonActionPerformed(java.awt.event.ActionEvent evt) {
434     findUploadFile();
435 }
436
437 private void findUploadFileButtonKeyPressed(java.awt.event.KeyEvent evt) {
438     if(evt.getKeyCode() == KeyEvent.VK_ENTER)
439     {
440         findUploadFile();
441     } //end if
442 }
443
444 private void previewUploadButtonActionPerformed(java.awt.event.ActionEvent evt) {
445     performPreviewOfMedia();
446 }
447

```

```

448 private void previewUploadButtonKeyPressed(java.awt.event.KeyEvent evt) {
449     if(evt.getKeyCode() == KeyEvent.VK_ENTER)
450     {
451         performPreviewOfMedia();
452     } //end if
453 }
454
455 /**
456  * Save the result in the database
457  *
458  */
459 private void saveResult()
460 {
461     boolean resultSaved = new ManageResultController().makeNewResult(originalDestination,
462         mechanicCommentTArea.getText(), customerCommentTArea.getText(),
463         mediaNameTField.getText(), resultType, vehicleID, workstationID,
464         orderID, "W-1");
465
466     if(resultSaved)
467     {
468         JOptionPane.showMessageDialog(null, "Upload is done to vehicleID: "
469             + vehicleID, "Upload is done",
470             JOptionPane.INFORMATION_MESSAGE);
471         this.dispose();
472     } //end if
473
474     else
475     {
476         JOptionPane.showMessageDialog(null, "Upload is failed for vehicleID: "
477             + vehicleID + "\n Try again", "Upload is failed",
478             JOptionPane.WARNING_MESSAGE);
479     } //end else
480 } //end method saveResult()
481
482 /**
483  * Cancel and close the uploadMediaView frame without saving
484  */
485 private void cancelAndClose()
486 {
487     int answer = JOptionPane.showConfirmDialog(null,
488         "Are you sure you want to cancel without saving?",
489         "Cancel and close upload media",
490         JOptionPane.YES_NO_OPTION
491         , JOptionPane.QUESTION_MESSAGE);
492
493     if(answer == JOptionPane.YES_OPTION)
494     {
495         this.dispose();
496     } //end if
497
498     else
499     {
500         return;
501     } //end else
502 } //end method cancelAndClose()
503
504 /**
505  * Open JFileChooser for selection of media to upload to the vehicle

```

```

506  */
507  private void findUploadFile()
508  {
509      JFileChooser jfc = new JFileChooser();
510      int returnval = jfc.showOpenDialog(findUploadFileButton);
511
512      if (returnval == JFileChooser.APPROVE_OPTION)
513      {
514          originalDestination = jfc.getSelectedFile().getPath();
515          previewUploadButton.setEnabled(true);
516      } //end if
517  } //end method findUploadFile()
518
519  /**
520   * Preview the upload files before saving
521   */
522  private void performPreviewOfMedia()
523  {
524      boolean canOpen = GUIHelpUtil.previewPDFFiles(originalDestination);
525
526      if (!canOpen)
527      {
528          JOptionPane.showMessageDialog(null, "Can't open file"
529              + vehicleID + "\n Try again", "Can't open file!",
530              JOptionPane.WARNING_MESSAGE);
531      } //end if
532
533  } //end method performPreviewOfMedia()
534
535
536  /**
537   * @param args the command line arguments
538   */
539  public static void main(String args[]) {
540      java.awt.EventQueue.invokeLater(new Runnable() {
541          public void run() {
542              //new UploadMediaView("V15", "RepairStation", "C1").setVisible(true);
543          }
544      });
545  }
546
547  // Variables declaration - do not modify
548  private javax.swing.JPanel buttonPanel;
549  private javax.swing.JButton cancelResultButton;
550  private javax.swing.JPanel commentPanel;
551  private javax.swing.JPanel contentPanel;
552  private javax.swing.JLabel customerCommentLabel;
553  private javax.swing.JScrollPane customerCommentScrollPane;
554  private javax.swing.JTextArea customerCommentTArea;
555  private javax.swing.JButton findUploadFileButton;
556  private javax.swing.JLabel headerLabel;
557  private javax.swing.JPanel headerPanel;
558  private javax.swing.JScrollPane mainScrollPane;
559  private javax.swing.JPanel mainUploadPanel;
560  private javax.swing.JPanel manageUploadPanel;
561  private javax.swing.JLabel mechanicCommentLabel;
562  private javax.swing.JScrollPane mechanicCommentScrollPane;
563  private javax.swing.JTextArea mechanicCommentTArea;

```

```

564 private javax.swing.JTextField mediaNameTField;
565 private javax.swing.JLabel orderIDLabel;
566 private javax.swing.JLabel orderIDResultLabel;
567 private javax.swing.JButton previewUploadButton;
568 private javax.swing.JLabel saveAsLabel;
569 private javax.swing.JButton saveResultButton;
570 private javax.swing.JLabel uploadTypeLabel;
571 private javax.swing.JLabel uploadTypeResultLabel;
572 private javax.swing.JLabel vehicleIDLabel;
573 private javax.swing.JLabel vehicleIDResultLabel;
574 private javax.swing.JLabel workstationNameLabel;
575 private javax.swing.JLabel workstationNameResultLabel;
576 // End of variables declaration
577
578 }

```

ValveadjustmentTableGUI

```

1 /*
2  * This class is a valveadjustment table for the mechanic
3  * It takes in the parametes numberOfColumns - to know how manu columns the table should contain
4  * It also takes cyl_no as an array - contains all the userdefined cylinder numbers
5  * it also takes cyl_pos as an array - contains all the userdefined cylinder positions
6  * it also takes vehicleID - To know witch vehicle the talbe belongs to
7  * it also takes workstationID - to know witch workstation to table belongs to
8  * At last it takes orderID to know witch order the table belongs to
9  */
10
11 /*
12  * ValveAdjustmentTableGUI.java
13  *
14  * Created on 10-12-2010, 09:06:52
15  */
16
17 package tweakmc.view;
18
19 import java.text.SimpleDateFormat;
20 import java.util.Calendar;
21 import java.util.GregorianCalendar;
22 import java.util.Timer;
23 import java.util.TimerTask;
24 import javax.swing.JTable;
25 import javax.swing.table.TableModel;
26 import tweakmc.control.ManageWhiteboardController;
27 import tweakmc.model.Result.Whiteboard.ValveAdjustmentTable;
28
29
30 /**
31  *
32  * @author NegoZiatoR
33  */
34 public class ValveAdjustmentTableGUI extends javax.swing.JFrame
35 {
36
37     // Static variables
38     private static ValveAdjustmentTableGUI instance;
39     private static SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
40     private static Calendar myC = new GregorianCalendar();
41     private final static String[] types = {"In", "Toll", "Diff", "Shim", "New shim"};

```

```

42
43 // Instance variables
44 private ManageWhiteboardController mwc = new ManageWhiteboardController();
45 private TableModel valveAdjustmentTableModel;
46 private int numberOfColumns;
47 private String mechanicComment;
48 private String customerComment;
49 private String[] cyl_no;
50 private String[] cyl_pos;
51 private String vehicleID;
52 private String workstationID;
53 private String orderID;
54
55 // Arrays with filled table content
56 private String[] inArray;
57 private String[] tollArray;
58 private String[] diffArray;
59 private String[] shimArray;
60 private String[] newShimArray;
61
62 /**
63  * Private constructor for ValveAdjustmentTableGUI
64  * @param numberOfColumns
65  * @param cyl_no array with all cylinder numbers
66  * @param cyl_pos array with all cylinder positions
67  */
68 private ValveAdjustmentTableGUI(int numberOfColumns, String[] cyl_no, String[] cyl_pos, String vehicleID,
String workstationID, String orderID)
69 {
70     // Initialize instance variables
71     this.cyl_no = cyl_no;
72     this.cyl_pos = cyl_pos;
73     this.numberOfColumns = numberOfColumns;
74     this.vehicleID = vehicleID;
75     this.workstationID = workstationID;
76     this.orderID = orderID;
77
78
79     // Create a new JTable and initialize components
80     valveAdjustmentTable = new JTable(6, numberOfColumns+1);
81     createTable();
82     initComponents();
83     changeName(valveAdjustmentTable, cyl_no);
84     setVisible(true);
85
86     // Show time
87     Timer timer = new Timer ();
88     timer.schedule (new ShowTime (), 0, 1000);
89
90 }
91
92 /**
93  * Singleton constructor for ValveAdjustmentTableGUI
94  * @param numberOfColumns number of columns to make set by user
95  * @param cyl_no array with all cylinder numbers
96  * @param cyl_pos array with all cylinder positions
97  * @return only ONE instance of valveAdjustment table
98  */

```



```

99  public static ValveAdjustmentTableGUI getInstance(int numberOfColumns, String[] cyl_no, String[] cyl_pos,
String vehicleID, String workstationID, String orderID)
100  {
101      if(instance == null)
102      {
103          instance = new ValveAdjustmentTableGUI(numberOfColumns, cyl_no, cyl_pos, vehicleID, workstationID,
orderID);
104      }
105      return instance;
106  }
107
108  /** This method is called from within the constructor to
109   * initialize the form.
110   * WARNING: Do NOT modify this code. The content of this method is
111   * always regenerated by the Form Editor.
112   */
113  @SuppressWarnings("unchecked")
114  // <editor-fold defaultstate="collapsed" desc="Generated Code">
115  private void initComponents() {
116
117      panelHeader = new javax.swing.JPanel();
118      labelHeader = new javax.swing.JLabel();
119      jSeparator1 = new javax.swing.JSeparator();
120      timeLabel = new javax.swing.JLabel();
121      panelTable = new javax.swing.JPanel();
122      jScrollPane1 = new javax.swing.JScrollPane();
123      valveAdjustmentTable = new javax.swing.JTable();
124      buttonPanel = new javax.swing.JPanel();
125      buttonSave = new javax.swing.JButton();
126      buttonCancel = new javax.swing.JButton();
127      panelComments = new javax.swing.JPanel();
128      textFieldMechanicComment = new javax.swing.JTextField();
129      textFieldCustomerComment = new javax.swing.JTextField();
130      labelMechanicComment = new javax.swing.JLabel();
131      labelCustomerComments = new javax.swing.JLabel();
132
133      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
134
135      labelHeader.setFont(new java.awt.Font("Tahoma", 0, 18));
136      labelHeader.setText("Valveadjustment Table");
137
138      timeLabel.setText("TimeLabel");
139
140      javax.swing.GroupLayout panelHeaderLayout = new javax.swing.GroupLayout(panelHeader);
141      panelHeader.setLayout(panelHeaderLayout);
142      panelHeaderLayout.setHorizontalGroup(
143          panelHeaderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
144              .addGroup(panelHeaderLayout.createSequentialGroup()
145                  .addContainerGap()
146                  .addGroup(panelHeaderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
147                      .addGroup(panelHeaderLayout.createSequentialGroup()
148                          .addComponent(labelHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 209,
javax.swing.GroupLayout.PREFERRED_SIZE)
149                          .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 483,
Short.MAX_VALUE)
150                          .addComponent(timeLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))

```



```

151         .addComponent(jSeparator1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 763, Short.MAX_VALUE))
152     .addContainerGap()
153 );
154 panelHeaderLayout.setVerticalGroup(
155     panelHeaderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
156     .addGroup(panelHeaderLayout.createSequentialGroup()
157         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
158         .addGroup(panelHeaderLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
159             .addComponent(labeblHeader, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
160             .addComponent(timeLabel))
161         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
162         .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE))
163 );
164
165 valveAdjustmentTable.setModel(valveAdjustmentTableModel);
166 valveAdjustmentTable.setAutoResizeMode(javax.swing.JTable.AUTO_RESIZE_ALL_COLUMNS);
167 valveAdjustmentTable.setColumnSelectionAllowed(true);
168 jScrollPane1.setViewportView(valveAdjustmentTable);
169
valveAdjustmentTable.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
170
171 javax.swing.GroupLayout panelTableLayout = new javax.swing.GroupLayout(panelTable);
172 panelTable.setLayout(panelTableLayout);
173 panelTableLayout.setHorizontalGroup(
174     panelTableLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
175     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 783, Short.MAX_VALUE)
176 );
177 panelTableLayout.setVerticalGroup(
178     panelTableLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
179     .addGroup(panelTableLayout.createSequentialGroup()
180         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 161,
javax.swing.GroupLayout.PREFERRED_SIZE)
181         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
182 );
183
184 buttonSave.setText("Save");
185 buttonSave.addActionListener(new java.awt.event.ActionListener() {
186     public void actionPerformed(java.awt.event.ActionEvent evt) {
187         buttonSaveActionPerformed(evt);
188     }
189 });
190
191 buttonCancel.setText("Cancel");
192
193 javax.swing.GroupLayout buttonPanelLayout = new javax.swing.GroupLayout(buttonPanel);
194 buttonPanel.setLayout(buttonPanelLayout);
195 buttonPanelLayout.setHorizontalGroup(
196     buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
197     .addGroup(buttonPanelLayout.createSequentialGroup()
198         .addComponent(buttonSave, javax.swing.GroupLayout.PREFERRED_SIZE, 75,
javax.swing.GroupLayout.PREFERRED_SIZE)
199         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
200         .addComponent(buttonCancel))
201 );

```

```

202     buttonPanelLayout.setVerticalGroup(
203         buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
204         .addGroup(buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
205             .addComponent(buttonSave)
206             .addComponent(buttonCancel))
207     );
208
209     labelMechanicComment.setText("Mechanic Comments");
210
211     labelCustomerComments.setText("Customer Comments");
212
213     javax.swing.GroupLayout panelCommentsLayout = new javax.swing.GroupLayout(panelComments);
214     panelComments.setLayout(panelCommentsLayout);
215     panelCommentsLayout.setHorizontalGroup(
216         panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
217         .addGroup(panelCommentsLayout.createSequentialGroup()
218             .addContainerGap()
219             .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
220                 .addComponent(labelMechanicComment, javax.swing.GroupLayout.Alignment.LEADING,
221                     javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
222                 .addComponent(textFieldMechanicComment, javax.swing.GroupLayout.Alignment.LEADING,
223                     javax.swing.GroupLayout.DEFAULT_SIZE, 167, Short.MAX_VALUE))
224                 .addGap(18, 18, 18)
225                 .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
226                     false)
227                     .addComponent(labelCustomerComments, javax.swing.GroupLayout.DEFAULT_SIZE,
228                         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
229                     .addComponent(textFieldCustomerComment, javax.swing.GroupLayout.DEFAULT_SIZE, 184,
230                         Short.MAX_VALUE))
231                 .addContainerGap(404, Short.MAX_VALUE))
232         );
233     panelCommentsLayout.setVerticalGroup(
234         panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
235         .addGroup(panelCommentsLayout.createSequentialGroup()
236             .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
237                 .addGroup(panelCommentsLayout.createSequentialGroup()
238                     .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
239                         panelCommentsLayout.createSequentialGroup()
240                             .createSequentialGroup()
241                             .addContainerGap()
242                             .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
243                                 .addComponent(labelMechanicComment)
244                                 .addComponent(labelCustomerComments))
245                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
246                                 .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
247                                     .addGroup(panelCommentsLayout.createSequentialGroup()
248                                         .addGroup(panelCommentsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

```

```

249         .addGroup(layout.createSequentialGroup())
250         .addGap(10, 10, 10)
251         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
252         .addComponent(panelTable, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
253         .addComponent(panelHeader, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
254         .addComponent(panelComments, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
255         .addContainerGap()
256     );
257     layout.setVerticalGroup(
258         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
259         .addGroup(layout.createSequentialGroup()
260             .addContainerGap()
261             .addComponent(panelHeader, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
262             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
263             .addComponent(panelTable, javax.swing.GroupLayout.PREFERRED_SIZE, 165,
javax.swing.GroupLayout.PREFERRED_SIZE)
264             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
265             .addComponent(panelComments, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
266             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
267             .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
268             .addContainerGap()
269         );
270
271     pack();
272 }// </editor-fold>
273
274 private void buttonSaveActionPerformed(java.awt.event.ActionEvent evt) {
275     // TODO add your handling code here:
276     saveTable();
277 }
278
279
280 /**
281  * Create a new JTable with no of columns set by user
282  */
283 private void createTable()
284 {
285     int counter = 0;
286
287     valveAdjustmentTableModel = valveAdjustmentTable.getModel();
288
289     // Set Cylinder positions - set by user
290     for(int columnIndex = 0; columnIndex <= numberOfColumns; columnIndex++)
291     {
292         valveAdjustmentTableModel.setValueAt(cyl_pos[columnIndex], 0, columnIndex);
293     } // End for loop
294     // Set default "In", "Toll", "Diff", "Shim", "New shim"
295     for(int rowIndex = 1; rowIndex < 6; rowIndex++)
296     {
297         valveAdjustmentTableModel.setValueAt(types[counter], rowIndex, 0);
298         counter++;

```

```

299     } // End for loop
300     // Set calculation on "In" - "Toll" = "Diff"
301     for(int calc = 1; calc <= numberOfColumns; calc++)
302     {
303         valveAdjustmentTableModel.setValueAt(10f, 1, calc);
304         valveAdjustmentTableModel.setValueAt(5f, 2, calc);
305
306         Float calcResult = ((Float)valveAdjustmentTableModel.getValueAt(1, calc)) -
307         (Float)valveAdjustmentTableModel.getValueAt(2, calc);
308         valveAdjustmentTableModel.setValueAt(calcResult, 3, calc);
309     } // End for loop
310     valveAdjustmentTable.setModel(valveAdjustmentTableModel);
311 } // End create table method
312
313 /**
314  * Change the column names with the users input
315  * @param table table to change
316  * @param col_index column index
317  * @param col_name column name
318  */
319 private void changeName(JTable valveAdjustmentTable, String[] cylinder_no)
320 {
321     String col_name = "";
322     int i = 0;
323     while(i < cylinder_no.length)
324     {
325         col_name = cylinder_no[i];
326         valveAdjustmentTable.getColumnModel().getColumn(i).setHeaderValue(col_name);
327         i++;
328     } // End While
329 } // End cahngeName method
330
331 /**
332  * First fill all the arrays with the user defined rows
333  * Then get the mechanic Comments and the customer comments
334  * Then Create the new valveadjustment table with all the arrays
335  */
336 public void saveTable()
337 {
338     fillAllArraysWithContent();
339     mechanicComment = textFieldMechanicComment.getText();
340     customerComment = textFieldCustomerComment.getText();
341     mwc.makeNewValveAdjustmentWhiteboard(mechanicComment, customerComment, cyl_no, cyl_pos,
342     inArray, tollArray,
343     diffArray, shimArray, newShimArray, vehicleID, workstationID,
344     orderID);
345 } // End Save Table method
346
347 /**
348  * Take all the user defined rows from the valve adjustment table and save them into arrays
349  */
350 public void fillAllArraysWithContent()
351 {
352     // built all the arrays with size of number of columns
353     inArray = new String[numberOfColumns];
354     tollArray = new String[numberOfColumns];
355     diffArray = new String[numberOfColumns];

```

```

354     shimArray = new String[numberOfColumns];
355     newShimArray = new String[numberOfColumns];
356
357
358     // Fill the inArray
359     for(int columnIndex = 1; columnIndex < numberOfColumns; columnIndex++)
360     {
361         String inValue = String.valueOf(valveAdjustmentTableModel.getValueAt(1, columnIndex));
362         inArray[columnIndex] = inValue;
363     } // End for loop
364
365     // Fill the tollArray
366     for(int columnIndex = 1; columnIndex < numberOfColumns; columnIndex++)
367     {
368         String tollValue = String.valueOf(valveAdjustmentTableModel.getValueAt(2, columnIndex));
369         tollArray[columnIndex] = tollValue;
370     } // End for loop
371
372     // Fill the diff Array
373     for(int columnIndex = 1; columnIndex < numberOfColumns; columnIndex++)
374     {
375         String diffValue = String.valueOf(valveAdjustmentTableModel.getValueAt(3, columnIndex));
376         diffArray[columnIndex] = diffValue;
377     } // End for loop
378
379     // Fill the Shim Array
380     for(int columnIndex = 1; columnIndex < numberOfColumns; columnIndex++)
381     {
382         String shimValue = String.valueOf(valveAdjustmentTableModel.getValueAt(4, columnIndex));
383         shimArray[columnIndex] = shimValue;
384     } // End for loop
385
386     // Fill the newShim Array
387     for(int columnIndex = 1; columnIndex < numberOfColumns; columnIndex++)
388     {
389         String newShimValue = String.valueOf(valveAdjustmentTableModel.getValueAt(5, columnIndex));
390         newShimArray[columnIndex] = newShimValue;
391     } // End for loop
392 }
393
394 //////////////////////////////////////////////////// PRIVATE INNER CLASS ///////////////////////////////////
395 class ShowTime extends TimerTask
396 {
397     public void run ()
398     {
399         timeLabel.setText(sdf.format(myC.getTime()));
400         myC.add (Calendar.SECOND, 1);
401         sdf.setCalendar (myC);
402     } // End inner run method
403 } // End private inner class showtime
404
405 // Variables declaration - do not modify
406 private javax.swing.JButton buttonCancel;
407 private javax.swing.JPanel buttonPanel;
408 private javax.swing.JButton buttonSave;
409 private javax.swing.JScrollPane jScrollPane1;
410 private javax.swing.JSeparator jSeparator1;
411 private javax.swing.JLabel labelHeader;

```

```

412 private javax.swing.JLabel labelCustomerComments;
413 private javax.swing.JLabel labelMechanicComment;
414 private javax.swing.JPanel panelComments;
415 private javax.swing.JPanel panelHeader;
416 private javax.swing.JPanel panelTable;
417 private javax.swing.JTextField textFieldCustomerComment;
418 private javax.swing.JTextField textFieldMechanicComment;
419 private javax.swing.JLabel timeLabel;
420 private javax.swing.JTable valveAdjustmentTable;
421 // End of variables declaration

```

VehicleGUI

```

1 /**
2  * This class is to create new Vehicles and search, edit existing vehicles
3  *
4  */
5
6 /**
7  * VehicleGUI.java
8  *
9  * Created on 28-10-2010, 14:51:21
10 */
11
12 package tweakmc.view;
13
14 import java.awt.Color;
15 import java.awt.Component;
16 import java.util.ArrayList;
17 import java.util.Collection;
18 import java.util.Collections;
19 import java.util.Iterator;
20 import java.util.List;
21 import java.util.Vector;
22 import javax.swing.DefaultComboBoxModel;
23 import javax.swing.JOptionPane;
24 import tweakmc.control.ManageVehicleController;
25 import tweakmc.utility.*;
26 import javax.swing.JPanel;
27 // removed untill all have imported the lib. Nyvang
28 import javax.swing.JScrollPane;
29 import org.jdesktop.swingx.autocomplete.AutoCompleteDecorator;
30 import tweakmc.dataaccess.AutoCompleteDAO;
31 import tweakmc.model.Owner;
32 import tweakmc.model.Vehicle;
33
34 /**
35  *
36  * @author NegoZiatoR
37  */
38 public class VehicleGUI extends javax.swing.JFrame {
39
40     private ManageVehicleController mvc = new ManageVehicleController();
41     private String valueOfVehicleType = "", valueOfVehicleBrand = "", errorMessage;
42     private String valueOfEditedVehicleType;

```

```

43 private String valueOfEditedVehicleModel;
44 private static Vector brandNameVector = new Vector();
45 private Vector vehicleTypeVector = new Vector();
46 private Integer options = null;
47 private boolean vehicleSaved = false;
48 private String[] arrActFoundsVehicle;
49 private List<Vehicle> actFoundsVehicleObjects;
50 private List<String> actFoundsBrandModel;
51 private List<String> actFoundVehicle;
52 private Vehicle selectedVehicle;
53 private List<Owner> listWithOwnersAttachedToVehicle;
54 private boolean[] arrayWithOwnersAttachedToVehicle;
55
56
57 // Constant Variables
58 private final String NO_VEHICLE_SELECTED = "No Vehicle is Selected!";
59
60 private final String VEHICLE_BRAND_CSV = "vehicleBrand.csv";
61 private final String VEHICLE_TYPE_CSV = "vehicleType.csv";
62 private final String TYPE_NOT_SELECTED = "Type is not selected";
63 private final String PLEASE_SELECT = "Please select";
64 private final String BRAND_EMPTY = "Brand is invalid";
65 private final String MODEL_EMPTY = "Model is empty";
66 private final String YEAR_EMPTY = "Year is empty";
67 private final String YEAR_INVALID = "Year is invalid";
68 private final String YEAR_NOT_NUMBER = "Year is not a number";
69 private final String PLEASE_ENTER = "Please enter brand";
70 /** Creates new form VehicleGUI */
71 public VehicleGUI()
72 {
73     bildAutoCompleteVectors();
74     initComponents();
75     resetFields();
76     disableEditVehiclePanel();
77
78 } //end VehicleGUI
79
80 /**
81  * This method is called from within the constructor to
82  * initialize the form.
83  * WARNING: Do NOT modify this code. The content of this method is
84  * always regenerated by the Form Editor.
85  * @return
86  */
87 @SuppressWarnings("unchecked")
88 // <editor-fold defaultstate="collapsed" desc="Generated Code">
89 private void initComponents() {
90
91     vehicleTabPanel = new javax.swing.JPanel();
92     jManageVehiclePane = new javax.swing.JTabbedPane();
93     registerVehiclePanel = new javax.swing.JPanel();
94     ownerInformationPanel1 = new javax.swing.JPanel();
95     nameLabel2 = new javax.swing.JLabel();
96     lastNameLabel2 = new javax.swing.JLabel();
97     addressLabel4 = new javax.swing.JLabel();
98     vehicleModelTextField = new javax.swing.JTextField();
99     addressLabel5 = new javax.swing.JLabel();
100    vehicleYearTextField = new javax.swing.JTextField();

```



```

101 zipAndCityLabel2 = new javax.swing.JLabel();
102 vehicleChassisNumberTextField = new javax.swing.JTextField();
103 countryLabel2 = new javax.swing.JLabel();
104 vehicleLicensePlateTextField = new javax.swing.JTextField();
105 jSeparator2 = new javax.swing.JSeparator();
106 saveVehicleButton = new javax.swing.JButton();
107 jButton9 = new javax.swing.JButton();
108 resultLabelVehicle = new javax.swing.JLabel();
109 jLabel4 = new javax.swing.JLabel();
110 vehicleTypeComboBox = new javax.swing.JComboBox();
111 AutoCompleteDecorator.decorate(vehicleTypeComboBox);
112 vehicleTypeComboBox.setEditable(true);
113 vehicleBrandComboBox = new javax.swing.JComboBox();
114 vehicleBrandComboBox.setEditable(true);
115 AutoCompleteDecorator.decorate(vehicleBrandComboBox);
116 searchEditPanel = new javax.swing.JPanel();
117 jScrollPane2 = new javax.swing.JScrollPane();
118 jPanel1 = new javax.swing.JPanel();
119 searchPanel = new javax.swing.JPanel();
120 searchVehicleTextField = new javax.swing.JTextField();
121 searchVehicleButton = new javax.swing.JButton();
122 noSearchResultsLabel = new javax.swing.JLabel();
123 jButton1 = new javax.swing.JButton();
124 vehicleListPanel = new javax.swing.JPanel();
125 selectVehicleButton = new javax.swing.JButton();
126 editVehicleButton = new javax.swing.JButton();
127 jScrollPane1 = new javax.swing.JScrollPane();
128 foundVehiclesJList = new javax.swing.JList();
129 attachOwnerButton = new javax.swing.JButton();
130 labelSelectedVehicle = new javax.swing.JLabel();
131 editVehiclePanel1 = new javax.swing.JPanel();
132 resetEditVehicleButton = new javax.swing.JButton();
133 jResultLabelOwner3 = new javax.swing.JLabel();
134 saveEditedVehicleButton = new javax.swing.JButton();
135 vehicleLicensePlateEditSelect = new javax.swing.JTextField();
136 vehicleChassisNumberEditSelect = new javax.swing.JTextField();
137 vehicleYearEditSelect = new javax.swing.JTextField();
138 vehicleModelEditSelect = new javax.swing.JTextField();
139 vehicleBrandEditSelect = new javax.swing.JTextField();
140 nameLabel3 = new javax.swing.JLabel();
141 lastNameLabel3 = new javax.swing.JLabel();
142 addressLabel6 = new javax.swing.JLabel();
143 addressLabel7 = new javax.swing.JLabel();
144 zipAndCityLabel3 = new javax.swing.JLabel();
145 countryLabel3 = new javax.swing.JLabel();
146 jLabel5 = new javax.swing.JLabel();
147 jLabel6 = new javax.swing.JLabel();
148 typeVehicleEditSelectComboBox = new javax.swing.JComboBox();
149 deleteVehicleButton = new javax.swing.JButton();
150 ownersPanel = new javax.swing.JPanel();
151 jScrollPane4 = new javax.swing.JScrollPane();
152 jTable2 = new javax.swing.JTable();
153 viewOwnerButton = new javax.swing.JButton();
154
155 setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
156 setTitle("KJMC Tweak MC");
157 setMinimumSize(new java.awt.Dimension(640, 480));
158

```



```

159 jManageVehiclePane.addChangeListener(new javax.swing.event.ChangeListener() {
160     public void stateChanged(javax.swing.event.ChangeEvent evt) {
161         jManageVehiclePaneStateChanged(evt);
162     }
163 });
164
165 ownerInformationPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter Vehicle
information"));
166
167 nameLabel2.setText("Type*");
168
169 lastNameLabel2.setText("Brand*");
170
171 addressLabel4.setText("Model*");
172
173 vehicleModelTextField.addActionListener(new java.awt.event.ActionListener() {
174     public void actionPerformed(java.awt.event.ActionEvent evt) {
175         vehicleModelTextFieldActionPerformed(evt);
176     }
177 });
178 vehicleModelTextField.addFocusListener(new java.awt.event.FocusAdapter() {
179     public void focusGained(java.awt.event.FocusEvent evt) {
180         vehicleModelTextFieldFocusGained(evt);
181     }
182     public void focusLost(java.awt.event.FocusEvent evt) {
183         vehicleModelTextFieldFocusLost(evt);
184     }
185 });
186
187 addressLabel5.setText("Year (YYYY)*");
188
189 vehicleYearTextField.addFocusListener(new java.awt.event.FocusAdapter() {
190     public void focusGained(java.awt.event.FocusEvent evt) {
191         vehicleYearTextFieldFocusGained(evt);
192     }
193     public void focusLost(java.awt.event.FocusEvent evt) {
194         vehicleYearTextFieldFocusLost(evt);
195     }
196 });
197
198 zipAndCityLabel2.setText("Chassis number");
199
200 vehicleChassisNumberTextField.addFocusListener(new java.awt.event.FocusAdapter() {
201     public void focusGained(java.awt.event.FocusEvent evt) {
202         vehicleChassisNumberTextFieldFocusGained(evt);
203     }
204 });
205
206 countryLabel2.setText("License plate");
207
208 vehicleLicensePlateTextField.addFocusListener(new java.awt.event.FocusAdapter() {
209     public void focusGained(java.awt.event.FocusEvent evt) {
210         vehicleLicensePlateTextFieldFocusGained(evt);
211     }
212 });
213
214 saveVehicleButton.setText("Save");
215 saveVehicleButton.addActionListener(new java.awt.event.ActionListener() {

```

```

216     public void actionPerformed(java.awt.event.ActionEvent evt) {
217         saveVehicleButtonActionPerformed(evt);
218     }
219 });
220
221 jButton9.setText("Reset fields");
222 jButton9.addActionListener(new java.awt.event.ActionListener() {
223     public void actionPerformed(java.awt.event.ActionEvent evt) {
224         jButton9ActionPerformed(evt);
225     }
226 });
227
228 resultLabelVehicle.setText(" ");
229
230 jLabel4.setFont(new java.awt.Font("Tahoma", 2, 10));
231 jLabel4.setText("The fields marked with * is required to register");
232
233 vehicleTypeComboBox.setModel(new DefaultComboBoxModel(vehicleTypeVector));
234 vehicleTypeComboBox.setSelectedIndex(0);
235 vehicleTypeComboBox.setToolTipText("Please select vehicle type");
236 vehicleTypeComboBox.addItemListener(new java.awt.event.ItemListener() {
237     public void itemStateChanged(java.awt.event.ItemEvent evt) {
238         vehicleTypeComboBoxItemStateChanged(evt);
239     }
240 });
241 vehicleTypeComboBox.addActionListener(new java.awt.event.ActionListener() {
242     public void actionPerformed(java.awt.event.ActionEvent evt) {
243         vehicleTypeComboBoxActionPerformed(evt);
244     }
245 });
246 vehicleTypeComboBox.addFocusListener(new java.awt.event.FocusAdapter() {
247     public void focusGained(java.awt.event.FocusEvent evt) {
248         vehicleTypeComboBoxFocusGained(evt);
249     }
250     public void focusLost(java.awt.event.FocusEvent evt) {
251         vehicleTypeComboBoxFocusLost(evt);
252     }
253 });
254
255 vehicleBrandComboBox.setModel(new DefaultComboBoxModel(brandNameVector));
256 vehicleBrandComboBox.setSelectedIndex(0);
257 vehicleBrandComboBox.setToolTipText("Please enter brand");
258 vehicleBrandComboBox.addItemListener(new java.awt.event.ItemListener() {
259     public void itemStateChanged(java.awt.event.ItemEvent evt) {
260         vehicleBrandComboBoxItemStateChanged(evt);
261     }
262 });
263 vehicleBrandComboBox.addActionListener(new java.awt.event.ActionListener() {
264     public void actionPerformed(java.awt.event.ActionEvent evt) {
265         vehicleBrandComboBoxActionPerformed(evt);
266     }
267 });
268
269 javax.swing.GroupLayout ownerInformationPanel1Layout = new
javax.swing.GroupLayout(ownerInformationPanel1);
270 ownerInformationPanel1.setLayout(ownerInformationPanel1Layout);
271 ownerInformationPanel1Layout.setHorizontalGroup(
272     ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

273         .addGroup(ownerInformationPanel1Layout.createSequentialGroup())
274         .addGap(40, 40, 40)
275
276     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
277         .addComponent(countryLabel2)
278         .addComponent(jLabel4)
279         .addComponent(jSeparator2)
280         .addGroup(ownerInformationPanel1Layout.createSequentialGroup())
281         .addComponent(saveVehicleButton)
282         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
283         .addComponent(jButton9))
284     .addGroup(ownerInformationPanel1Layout.createSequentialGroup())
285     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
286         .addComponent(nameLabel2)
287         .addComponent(lastNameLabel2)
288         .addComponent(addressLabel4)
289         .addComponent(addressLabel5)
290         .addComponent(zipAndCityLabel2))
291     .addGap(54, 54, 54)
292
293     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
294         .addComponent(vehicleTypeComboBox, 0, 196, Short.MAX_VALUE)
295         .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
296             .addComponent(vehicleYearTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
297             .addComponent(vehicleModelTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
Short.MAX_VALUE)
298             .addComponent(vehicleLicensePlateTextField,
javax.swing.GroupLayout.Alignment.TRAILING)
299             .addComponent(vehicleChassisNumberTextField,
javax.swing.GroupLayout.Alignment.TRAILING))
300             .addComponent(vehicleBrandComboBox, 0, 196, Short.MAX_VALUE)))
301     .addComponent(resultLabelVehicle, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
302     .addContainerGap()
303 );
304
305 ownerInformationPanel1Layout.setVerticalGroup(
306     ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
307     .addGroup(ownerInformationPanel1Layout.createSequentialGroup())
308     .addComponent(jLabel4)
309     .addGap(28, 28, 28)
310
311     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
312         .addComponent(nameLabel2)
313         .addComponent(vehicleTypeComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
314     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
315
316     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
317         .addComponent(lastNameLabel2)
318         .addComponent(vehicleBrandComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

```

```

315         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
316
317     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
318         .addComponent(addressLabel4)
319         .addComponent(vehicleModelTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
320     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
321
322     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
323         .addComponent(addressLabel5)
324         .addComponent(vehicleYearTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
325     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
326
327     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
328         .addComponent(zipAndCityLabel2)
329         .addComponent(vehicleChassisNumberTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
330     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
331
332     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
333         .addComponent(countryLabel2)
334         .addComponent(vehicleLicensePlateTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
335     .addGap(80, 80, 80)
336     .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
337     .addGap(19, 19, 19)
338
339     .addGroup(ownerInformationPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
340         .addComponent(saveVehicleButton)
341         .addComponent(jButton9))
342     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
343     .addComponent(resultLabelVehicle)
344     .addContainerGap(17, Short.MAX_VALUE))
345 );
346
347     javax.swing.GroupLayout registerVehiclePanelLayout = new
javax.swing.GroupLayout(registerVehiclePanel);
348     registerVehiclePanel.setLayout(registerVehiclePanelLayout);
349     registerVehiclePanelLayout.setHorizontalGroup(
350         registerVehiclePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
351         .addGroup(registerVehiclePanelLayout.createSequentialGroup())
352         .addContainerGap()
353         .addComponent(ownerInformationPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 421,
javax.swing.GroupLayout.PREFERRED_SIZE)
354         .addContainerGap(749, Short.MAX_VALUE))
355     );
356     registerVehiclePanelLayout.setVerticalGroup(
357         registerVehiclePanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
358         .addGroup(registerVehiclePanelLayout.createSequentialGroup())
359         .addGap(19, 19, 19)
360         .addComponent(ownerInformationPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
361         .addContainerGap(311, Short.MAX_VALUE))
362     );
363
364     jManageVehiclePane.addTab("Register Vehicle", registerVehiclePanel);

```

```

360
361 searchPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter search criterias"));
362
363 searchVehicleButton.setText("Search");
364 searchVehicleButton.addActionListener(new java.awt.event.ActionListener() {
365     public void actionPerformed(java.awt.event.ActionEvent evt) {
366         searchVehicleButtonActionPerformed(evt);
367     }
368 });
369
370 jButton1.setText("jButton1");
371 jButton1.addActionListener(new java.awt.event.ActionListener() {
372     public void actionPerformed(java.awt.event.ActionEvent evt) {
373         jButton1ActionPerformed(evt);
374     }
375 });
376
377 javax.swing.GroupLayout searchPanelLayout = new javax.swing.GroupLayout(searchPanel);
378 searchPanel.setLayout(searchPanelLayout);
379 searchPanelLayout.setHorizontalGroup(
380     searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
381         .addGroup(searchPanelLayout.createSequentialGroup()
382             .add(searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
383                 .add(searchVehicleTextField)
384                 .add(searchVehicleButton)
385                 .add(noSearchResultsLabel)
386                 .add(jButton1)
387             )
388             .addContainerGap(95, true)
389         )
390         .add(new javax.swing.GroupLayout(searchPanelLayout) {
391             public void init() {
392                 add(searchVehicleButton);
393                 add(jButton1);
394             }
395         })
396 );
397 searchPanelLayout.setVerticalGroup(
398     searchPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
399         .addGroup(searchPanelLayout.createSequentialGroup()
400             .add(searchVehicleTextField)
401             .add(searchVehicleButton)
402             .add(noSearchResultsLabel)
403             .add(jButton1)
404             .addContainerGap(18, true)
405         )
406         .add(new javax.swing.GroupLayout(searchPanelLayout) {
407             public void init() {
408                 add(searchVehicleButton);
409                 add(jButton1);
410             }
411         })
412 );
413 vehicleListPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Vehicle List"));

```

```

414 selectVehicleButton.setText("Select Vehicle");
415 selectVehicleButton.addActionListener(new java.awt.event.ActionListener() {
416     public void actionPerformed(java.awt.event.ActionEvent evt) {
417         selectVehicleButtonActionPerformed(evt);
418     }
419 });
420
421 editVehicleButton.setText("Edit Vehicle");
422 editVehicleButton.setEnabled(false);
423 editVehicleButton.addActionListener(new java.awt.event.ActionListener() {
424     public void actionPerformed(java.awt.event.ActionEvent evt) {
425         editVehicleButtonActionPerformed(evt);
426     }
427 });
428
429 jScrollPane1.setViewportView(foundVehiclesJList);
430
431 attachOwnerButton.setText("Attach Owner");
432 attachOwnerButton.setEnabled(false);
433 attachOwnerButton.addActionListener(new java.awt.event.ActionListener() {
434     public void actionPerformed(java.awt.event.ActionEvent evt) {
435         attachOwnerButtonActionPerformed(evt);
436     }
437 });
438
439 javax.swing.GroupLayout vehicleListPanelLayout = new javax.swing.GroupLayout(vehicleListPanel);
440 vehicleListPanel.setLayout(vehicleListPanelLayout);
441 vehicleListPanelLayout.setHorizontalGroup(
442     vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
443     .addGroup(vehicleListPanelLayout.createSequentialGroup()
444         .addContainerGap()
445         .addGroup(vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
446             .addComponent(labelSelectedVehicle, javax.swing.GroupLayout.DEFAULT_SIZE, 507,
Short.MAX_VALUE)
447             .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 507,
Short.MAX_VALUE)
448             .addGroup(vehicleListPanelLayout.createSequentialGroup()
449                 .addGroup(vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
450                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 218,
Short.MAX_VALUE)
451                     .addComponent(attachOwnerButton)
452                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
453                     .addComponent(editVehicleButton))
454                 .addContainerGap())
455             .addGroup(vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
456                 .addGroup(vehicleListPanelLayout.createSequentialGroup()
457                     .addGroup(vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
458                         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 276,
javax.swing.GroupLayout.PREFERRED_SIZE)
459                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
460                         .addGroup(vehicleListPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
461                             .addComponent(selectVehicleButton)
462                             .addComponent(editVehicleButton)
463                             .addComponent(attachOwnerButton))
464                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```



```

466         .addComponent(labelSelectedVehicle)
467         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
468     );
469
470     editVehiclePanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Vehicle information"));
471
472     resetEditVehicleButton.setText("Reset fields");
473     resetEditVehicleButton.setEnabled(false);
474     resetEditVehicleButton.addActionListener(new java.awt.event.ActionListener() {
475         public void actionPerformed(java.awt.event.ActionEvent evt) {
476             resetEditVehicleButtonActionPerformed(evt);
477         }
478     });
479
480     jResultLabelOwner3.setText(" ");
481
482     saveEditedVehicleButton.setText("Save");
483     saveEditedVehicleButton.setEnabled(false);
484     saveEditedVehicleButton.addActionListener(new java.awt.event.ActionListener() {
485         public void actionPerformed(java.awt.event.ActionEvent evt) {
486             saveEditedVehicleButtonActionPerformed(evt);
487         }
488     });
489
490     vehicleLicensePlateEditSelect.setEditable(false);
491
492     vehicleChassisNumberEditSelect.setEditable(false);
493
494     vehicleYearEditSelect.setEditable(false);
495
496     vehicleModelEditSelect.setEditable(false);
497
498     vehicleBrandEditSelect.setEditable(false);
499
500     nameLabel3.setText("Type*");
501
502     lastNameLabel3.setText("Brand*");
503
504     addressLabel6.setText("Model*");
505
506     addressLabel7.setText("Year*");
507
508     zipAndCityLabel3.setText("Chassis number");
509
510     countryLabel3.setText("License plate");
511
512     jLabel5.setFont(new java.awt.Font("Tahoma", 2, 10));
513     jLabel5.setText("Change the specific value in the corresponding text field and select save");
514
515     jLabel6.setFont(new java.awt.Font("Tahoma", 1, 11));
516     jLabel6.setText("Edit / Select Vehicle");
517
518     typeVehicleEditSelectComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Please
select", "Motorcycle", "ATV", "Other" }));
519     typeVehicleEditSelectComboBox.addActionListener(new java.awt.event.ActionListener() {
520         public void actionPerformed(java.awt.event.ActionEvent evt) {
521             typeVehicleEditSelectComboBoxActionPerformed(evt);
522         }
523     });

```

```

523     });
524
525     deleteVehicleButton.setText("Delete Vehicle");
526     deleteVehicleButton.setEnabled(false);
527     deleteVehicleButton.addActionListener(new java.awt.event.ActionListener() {
528         public void actionPerformed(java.awt.event.ActionEvent evt) {
529             deleteVehicleButtonActionPerformed(evt);
530         }
531     });
532
533     javax.swing.GroupLayout editVehiclePanel1Layout = new javax.swing.GroupLayout(editVehiclePanel1);
534     editVehiclePanel1.setLayout(editVehiclePanel1Layout);
535     editVehiclePanel1Layout.setHorizontalGroup(
536         editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
537             .addGroup(editVehiclePanel1Layout.createSequentialGroup()
538                 .addGap(31, 31, 31)
539                 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
540                     .addComponent(jLabel6)
541                     .addComponent(jResultLabelOwner3, javax.swing.GroupLayout.Alignment.TRAILING,
542                         javax.swing.GroupLayout.PREFERRED_SIZE, 38, javax.swing.GroupLayout.PREFERRED_SIZE)
543                     .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
544                         .addGroup(editVehiclePanel1Layout.createSequentialGroup()
545                             .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
546                                 .addGroup(editVehiclePanel1Layout.createSequentialGroup()
547                                     .addComponent(nameLabel3)
548                                     .addComponent(lastNameLabel3)
549                                     .addComponent(addressLabel6)
550                                     .addComponent(addressLabel7)
551                                     .addComponent(zipAndCityLabel3))
552                                 .addGap(54, 54, 54)
553                                 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
554                                     .addComponent(typeVehileEditSelectComboBox, 0,
555                                         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
556                                     .addComponent(vehicleYearEditSelect, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
557                                         Short.MAX_VALUE)
558                                     .addComponent(vehicleModelEditSelect, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
559                                         Short.MAX_VALUE)
560                                     .addComponent(vehicleBrandEditSelect, javax.swing.GroupLayout.DEFAULT_SIZE, 196,
561                                         Short.MAX_VALUE)
562                                     .addComponent(vehicleLicensPlateEditSelect,
563                                         javax.swing.GroupLayout.Alignment.TRAILING)
564                                     .addComponent(vehicleChassisNumberEditSelect,
565                                         javax.swing.GroupLayout.Alignment.TRAILING)))
566                                 .addGroup(editVehiclePanel1Layout.createSequentialGroup()
567                                     .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
568                                         .addComponent(saveEditedVehicleButton)
569                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 104,
570                                             Short.MAX_VALUE)
571                                         .addComponent(deleteVehicleButton)
572                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
573                                         .addComponent(resetEditVehicleButton)))
574                                 .addComponent(jLabel5)))
575                 .addContainerGap(62, Short.MAX_VALUE))

```



```

568 );
569 editVehiclePanel1Layout.setVerticalGroup(
570     editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
571     .addGroup(editVehiclePanel1Layout.createSequentialGroup())
572     .addContainerGap()
573     .addComponent(jLabel6)
574     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
575     .addComponent(jLabel5)
576     .addGap(18, 18, 18)
577
578 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
579     .addComponent(nameLabel3)
580     .addComponent(typeVehicleEditSelectComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
581     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
582
583 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
584     .addComponent(lastNameLabel3)
585     .addComponent(vehicleBrandEditSelect, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
586     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
587
588 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
589     .addComponent(addressLabel6)
590     .addComponent(vehicleModelEditSelect, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
591     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
592
593 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
594     .addComponent(addressLabel7)
595     .addComponent(vehicleYearEditSelect, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
596     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
597
598 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
599     .addComponent(countryLabel3)
600     .addComponent(vehicleLicensePlateEditSelect, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
601     .addGap(31, 31, 31)
602
603 .addGroup(editVehiclePanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
604     .addComponent(resetEditVehicleButton)
605     .addComponent(saveEditedVehicleButton)
606     .addComponent(deleteVehicleButton))
607     .addGap(177, 177, 177)
608     .addComponent(jResultLabelOwner3)
609     .addContainerGap()
610 );
611 ownersPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Owners"));
612 jTable2.setModel(new javax.swing.table.DefaultTableModel(

```

```

613     new Object [][] {
614         {null, null, null, null},
615         {null, null, null, null},
616         {null, null, null, null},
617         {null, null, null, null}
618     },
619     new String [] {
620         "Name", "Phone", "E-mail", "Active"
621     }
622 ) {
623     Class[] types = new Class [] {
624         java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Boolean.class
625     };
626     boolean[] canEdit = new boolean [] {
627         false, false, false, true
628     };
629
630     public Class getColumnClass(int columnIndex) {
631         return types [columnIndex];
632     }
633
634     public boolean isCellEditable(int rowIndex, int columnIndex) {
635         return canEdit [columnIndex];
636     }
637 });
638 jScrollPane4.setViewportViewView(jTable2);
639
640 viewOwnerButton.setText("View Owner data");
641
642 javax.swing.GroupLayout ownersPanelLayout = new javax.swing.GroupLayout(ownersPanel);
643 ownersPanel.setLayout(ownersPanelLayout);
644 ownersPanelLayout.setHorizontalGroup(
645     ownersPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
646     .addGroup(ownersPanelLayout.createSequentialGroup()
647         .addContainerGap()
648         .addGroup(ownersPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
649             .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 391,
javax.swing.GroupLayout.PREFERRED_SIZE)
650             .addComponent(viewOwnerButton))
651         .addContainerGap(47, Short.MAX_VALUE))
652 );
653 ownersPanelLayout.setVerticalGroup(
654     ownersPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
655     .addGroup(ownersPanelLayout.createSequentialGroup()
656         .addContainerGap()
657         .addComponent(jScrollPane4, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
658         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
659         .addComponent(viewOwnerButton)
660         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
661 );
662
663 javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
664 jPanel1.setLayout(jPanel1Layout);
665 jPanel1Layout.setHorizontalGroup(
666     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
667     .addGroup(jPanel1Layout.createSequentialGroup()
668         .addContainerGap()

```

```

669         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
670         .addComponent(searchPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
671         .addComponent(vehicleListPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
672         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
673         .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
674         .addComponent(ownersPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
675         .addComponent(editVehiclePanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
676         .addGap(216, 216, 216))
677     );
678     jPanel1Layout.setVerticalGroup(
679     jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
680     .addGroup(jPanel1Layout.createSequentialGroup())
681     .addContainerGap()
682     .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
683     .addGroup(jPanel1Layout.createSequentialGroup())
684     .addComponent(searchPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
685     .addGap(12, 12, 12)
686     .addComponent(vehicleListPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
687     .addGroup(jPanel1Layout.createSequentialGroup())
688     .addComponent(editVehiclePanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 355,
javax.swing.GroupLayout.PREFERRED_SIZE)
689     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
690     .addComponent(ownersPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 170,
javax.swing.GroupLayout.PREFERRED_SIZE)))
691     .addContainerGap(204, Short.MAX_VALUE))
692 );
693
694 jScrollPane2.setViewportViewView(jPanel1);
695
696 javax.swing.GroupLayout searchEditPanelLayout = new javax.swing.GroupLayout(searchEditPanel);
697 searchEditPanel.setLayout(searchEditPanelLayout);
698 searchEditPanelLayout.setHorizontalGroup(
699     searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
700     .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 1180, Short.MAX_VALUE)
701 );
702 searchEditPanelLayout.setVerticalGroup(
703     searchEditPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
704     .addComponent(jScrollPane2)
705 );
706
707 jManageVehiclePane.addTab("Search / Edit Vehicle", searchEditPanel);
708
709 javax.swing.GroupLayout vehicleTabPanelLayout = new javax.swing.GroupLayout(vehicleTabPanel);
710 vehicleTabPanel.setLayout(vehicleTabPanelLayout);
711 vehicleTabPanelLayout.setHorizontalGroup(
712     vehicleTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
713     .addGroup(vehicleTabPanelLayout.createSequentialGroup())
714     .addComponent(jManageVehiclePane, javax.swing.GroupLayout.PREFERRED_SIZE, 1185,
javax.swing.GroupLayout.PREFERRED_SIZE)
715     .addContainerGap(20, Short.MAX_VALUE))
716 );
717 vehicleTabPanelLayout.setVerticalGroup(

```

```

718     vehicleTabPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
719     .addGroup(vehicleTabPanelLayout.createSequentialGroup()
720     .addComponent(jManageVehiclePane, javax.swing.GroupLayout.PREFERRED_SIZE, 781,
javax.swing.GroupLayout.PREFERRED_SIZE)
721     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
722     );
723
724     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
725     getContentPane().setLayout(layout);
726     layout.setHorizontalGroup(
727         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
728         .addGroup(layout.createSequentialGroup()
729             .addComponent(vehicleTabPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
730             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
731         );
732     layout.setVerticalGroup(
733         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
734         .addGroup(layout.createSequentialGroup()
735             .addComponent(vehicleTabPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
736             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
737         );
738
739     pack();
740 } // </editor-fold>
741
742 private void saveVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
743
744     //////////// REMEBER A "DO YOU WANT TO SAVE" DIAGLOG ////////////
745     registerVehicle();
746
747 }
748
749 private void vehicleTypeComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
750
751     setVehicleType((String) vehicleTypeComboBox.getSelectedItem());
752 }
753
754 private void editVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
755
756     typeVehileEditSelectComboBox.setEnabled(true);
757     typeVehileEditSelectComboBox.setBackground(Color.white);
758     // Make brand field editable
759     vehicleBrandEditSelect.setEnabled(true);
760     vehicleBrandEditSelect.setEditable(true);
761     vehicleBrandEditSelect.setBackground(Color.white);
762
763     // Make model field editable
764     vehicleModelEditSelect.setEnabled(true);
765     vehicleModelEditSelect.setEditable(true);
766     vehicleModelEditSelect.setBackground(Color.white);
767
768     // Make year field editable
769     vehicleYearEditSelect.setEnabled(true);
770     vehicleYearEditSelect.setEditable(true);
771     vehicleYearEditSelect.setBackground(Color.white);
772

```

```

773 // Make the chassis number field editable
774 vehicleChassisNumberEditSelect.setEnabled(true);
775 vehicleChassisNumberEditSelect.setEditable(true);
776 vehicleChassisNumberEditSelect.setBackground(Color.white);
777
778 // Make the licens plate field editable
779 vehicleLicensPlateEditSelect.setEnabled(true);
780 vehicleLicensPlateEditSelect.setEditable(true);
781 vehicleLicensPlateEditSelect.setBackground(Color.white);
782
783 // Make Save edit button editable
784 saveEditedVehicleButton.setEnabled(true);
785
786 // Make Delete vehilce button enabled
787 deleteVehicleButton.setEnabled(true);
788
789 // Make ResetEdit button editable
790 resetEditVehicleButton.setEnabled(true);
791 }
792
793 private void saveEditedVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
794     updateVehicle();
795 }
796
797 private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
798     // TODO add your handling code here:
799     resetFields();
800 }
801
802 private void vehicleTypeComboBoxFocusLost(java.awt.event.FocusEvent evt) {
803     // TODO add your handling code here:
804     if(vehicleTypeComboBox.getSelectedIndex()==0)
805     {
806         vehicleTypeComboBox.setBackground(Color.PINK);
807     }
808     else
809     {
810         vehicleTypeComboBox.setBackground(Color.white);
811     }
812 }
813
814 private void vehicleTypeComboBoxItemStateChanged(java.awt.event.ItemEvent evt) {
815     // TODO add your handling code here:
816     if(vehicleTypeComboBox.getSelectedIndex()==0)
817     {
818         vehicleTypeComboBox.setBackground(Color.PINK);
819     }
820     else
821     {
822         vehicleTypeComboBox.setBackground(Color.white);
823     }
824 }
825
826 private void vehicleModelTextFieldFocusGained(java.awt.event.FocusEvent evt) {
827     // TODO add your handling code here:
828     if(vehicleModelTextField.getBackground().equals(Color.PINK))
829     {
830         vehicleModelTextField.setBackground(Color.white);

```

```

831     }
832
833 }
834
835 private void vehicleYearTextFieldFocusGained(java.awt.event.FocusEvent evt) {
836     // TODO add your handling code here:
837     if(vehicleYearTextField.getBackground().equals(Color.PINK))
838     {
839         vehicleYearTextField.setBackground(Color.white);
840     }
841 }
842
843 private void vehicleChassisNumberTextFieldFocusGained(java.awt.event.FocusEvent evt) {
844     // TODO add your handling code here:
845     if(vehicleChassisNumberTextField.getBackground().equals(Color.PINK))
846     {
847         vehicleChassisNumberTextField.setBackground(Color.white);
848     }
849 }
850
851 private void vehicleLicensePlateTextFieldFocusGained(java.awt.event.FocusEvent evt) {
852     // TODO add your handling code here:
853     if(vehicleLicensePlateTextField.getBackground().equals(Color.PINK))
854     {
855         vehicleLicensePlateTextField.setBackground(Color.white);
856     }
857 }
858
859 private void vehicleTypeComboBoxFocusGained(java.awt.event.FocusEvent evt) {
860     // TODO add your handling code here:
861     if(vehicleTypeComboBox.getBackground().equals(Color.PINK))
862     {
863         vehicleTypeComboBox.setBackground(Color.white);
864     }
865 }
866
867 private void vehicleModelTextFieldFocusLost(java.awt.event.FocusEvent evt) {
868     // TODO add your handling code here:
869     if(vehicleModelTextField.getText().equals(""))
870     {
871         errorMessage = MODEL_EMPTY;
872         vehicleModelTextField.setBackground(Color.PINK);
873     }
874 }
875
876 private void vehicleYearTextFieldFocusLost(java.awt.event.FocusEvent evt) {
877     // TODO add your handling code here:
878     if(vehicleYearTextField.getText().equals(""))
879     {
880         errorMessage = YEAR_EMPTY;
881         vehicleYearTextField.setBackground(Color.PINK);
882     }
883     if(!CheckInput.correctYear(vehicleYearTextField.getText()))
884     {
885         errorMessage = YEAR_INVALID;
886         vehicleYearTextField.setBackground(Color.PINK);
887     }
888 }

```

```

889     if(!CheckInput.isInt(vehicleYearTextField.getText()))
890     {
891         errorMessage = YEAR_NOT_NUMBER;
892         vehicleYearTextField.setBackground(Color.PINK);
893     }
894 }
895
896 private void vehicleBrandComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
897     setVehicleBrand((String) vehicleBrandComboBox.getSelectedItem());
898 }
899
900 private void vehicleBrandComboBoxItemStateChanged(java.awt.event.ItemEvent evt) {
901     if (vehicleBrandComboBox.getSelectedIndex() == 0)
902     {
903         vehicleBrandComboBox.setBackground(Color.pink);
904     }
905
906     else
907     {
908         valueOfVehicleBrand = (String) vehicleBrandComboBox.getSelectedItem();
909     }
910 }
911
912 private void searchVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
913
914     disableEditVehiclePanel();
915     //Get the text from the searchfield
916     final String[] arrActfounds;
917     boolean testForInt;
918     Integer integer_searchCriteria = null;
919     String str_searchCriteria = null;
920     try
921     {
922         integer_searchCriteria = Integer.parseInt(searchVehicleTextField.getText());
923         testForInt = true;
924
925     }
926     catch (Exception e)
927     {
928         str_searchCriteria = searchVehicleTextField.getText();
929         testForInt = false;
930     }
931     if (testForInt)
932     {
933         arrActfounds = enterSearchDetails(integer_searchCriteria);
934     }
935
936     else
937     {
938         arrActfounds = enterSearchDetails(str_searchCriteria);
939     }
940     //Get an array with the searchResults
941
942
943     if (arrActfounds.length==0)
944     {
945         noSearchResultsLabel.setText("No searchresults");
946         noSearchResultsLabel.setForeground(Color.red);

```



```

947     foundVehiclesJList.setModel(new javax.swing.AbstractListModel() {
948         public int getSize() { return arrActfounds.length; }
949         public Object getElementAt(int i) { return arrActfounds[i]; }
950     });
951 } //end if
952 else
953 {
954     noSearchResultsLabel.setText("");
955     //Clear the actual JList and make a new one
956     foundVehiclesJList.setModel(new javax.swing.AbstractListModel() {
957         public int getSize() { return arrActfounds.length; }
958         public Object getElementAt(int i) { return arrActfounds[i]; }
959     });
960
961
962 } //end else
963 }
964
965 private void jManageVehiclePaneStateChanged(javax.swing.event.ChangeEvent evt) {
966     // Check if panel changed to is not registervehiclePanel
967     if(!jManageVehiclePane.getSelectedComponent().equals(registerVehiclePanel))
968
969         // First check if type, brand, model and year is ok
970         if(!valueOfVehicleType.equals(PLEASE_SELECT) || !valueOfVehicleType.equals("")
971             && !valueOfVehicleBrand.equals(PLEASE_ENTER) || !valueOfVehicleBrand.equals("")
972             && !vehicleModelTextField.getText().equals("")
973             && !vehicleYearTextField.getText().equals("")
974             && CheckInput.correctYear(vehicleYearTextField.getText()))
975         {
976             // Saved the selected panel temp
977             Component selected = jManageVehiclePane.getSelectedComponent();
978             jManageVehiclePane.setSelectedComponent(registerVehiclePanel);
979             options = JOptionPane.showConfirmDialog(null, "Do you want to save?", "Do you Want to save the
owner?", JOptionPane.YES_NO_OPTION);
980             if(options==0)
981             {
982                 registerVehicle();
983                 resetFields();
984                 setVehicleSavedStatus(true);
985
986             }
987             else if(options==1)
988             {
989                 resetFields();
990                 jManageVehiclePane.setSelectedComponent(selected);
991                 setVehicleSavedStatus(false);
992             }
993             resetFields();
994         }
995
996     }
997
998 private void selectVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
999     // TODO add your handling code here:
1000     disableEditVehiclePanel();
1001     if(foundVehiclesJList.getSelectedIndex() >= 0)
1002     {
1003         attachOwnerButton.setEnabled(true);

```



```

1004     editVehicleButton.setEnabled(true);
1005     int indexPositionOfSelectedVehicle = foundVehiclesJList.getSelectedIndex();
1006     String vehicleID = arrActFoundsVehicle[indexPositionOfSelectedVehicle] + "";
1007
1008     boolean found = false;
1009     int i = 0;
1010     Vehicle v = null;
1011
1012     // Search for the right selected vehilce in the found vehicles
1013     while(i < actFoundsVehicleObjects.size() && !found)
1014     {
1015         v = actFoundsVehicleObjects.get(i);
1016         if(v.getVehicleID().equals(vehicleID))
1017         {
1018             found = true;
1019             selectedVehicle = actFoundsVehicleObjects.get(i);
1020         } // End inner if
1021         else
1022         {
1023             i++;
1024         } // End inner else
1025     } // End while
1026 } // End If
1027 else
1028 {
1029     labelSelectedVehicle.setText(NO_VEHICLE_SELECTED);
1030     selectedVehicle = null;
1031 }
1032
1033 if(selectedVehicle!=null)
1034 {
1035     if(selectedVehicle.getvType().equals("Motorcycle"))
1036     {
1037         typeVehileEditSelectComboBox.setSelectedIndex(1);
1038     }
1039     else if(selectedVehicle.getvType().equals("ATV"))
1040     {
1041         typeVehileEditSelectComboBox.setSelectedIndex(2);
1042     }
1043     else
1044     {
1045         typeVehileEditSelectComboBox.setSelectedIndex(3);
1046     }
1047     vehicleBrandEditSelect.setText(selectedVehicle.getBrand());
1048     vehicleModelEditSelect.setText(selectedVehicle.getModel());
1049     vehicleYearEditSelect.setText(Integer.toString(selectedVehicle.getvYear()));
1050     vehicleChassisNumberEditSelect.setText(selectedVehicle.getChassisNumber());
1051     vehicleLicensPlateEditSelect.setText(selectedVehicle.getLicensPlate());
1052
1053     getOwnersAttachedToVehicle(mvc.searchVehicleWithOwner(selectedVehicle.getVehicleID()),
1054     mvc.getActiveNess());
1055     setTableWithOwners(listWithOwnersAttachedToVehicle, arrayWithOwnersAttachedToVehicle);
1056 }
1057
1058
1059
1060 }

```

```

1061
1062 private void resetEditVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
1063     // TODO add your handling code here:
1064     options = JOptionPane.showConfirmDialog(null, "Are you sure you want to reset all " +
selectedVehicle.getBrand() + " " + selectedVehicle.getModel() + " " + selectedVehicle.getLicensPlate() + " Details ??",
"Want to reset ?", JOptionPane.YES_NO_OPTION);
1065     if(options==JOptionPane.YES_OPTION)
1066     {
1067         resetFields();
1068     }
1069     else
1070     {
1071         return;
1072     }
1073 }
1074
1075 private void typeVehileEditSelectComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
1076     editVehicleType((String) typeVehileEditSelectComboBox.getSelectedItem());
1077 }
1078
1079 private void deleteVehicleButtonActionPerformed(java.awt.event.ActionEvent evt) {
1080     // TODO add your handling code here:
1081     deleteVehicle();
1082 }
1083
1084 private void attachOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
1085     // TODO add your handling code here:
1086     AttachOwnerGUI.getInstance(selectedVehicle.getVehicleID());
1087 }
1088
1089 private void vehicleModelTextFieldActionPerformed(java.awt.event.ActionEvent evt) {
1090     // TODO add your handling code here:
1091 }
1092
1093 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
1094     SuperSearch.getInstance().run();
1095 }
1096
1097 private void viewOwnerButtonActionPerformed(java.awt.event.ActionEvent evt) {
1098     if(jTable2.getSelectedRow() >= 0)
1099         JOptionPane.showMessageDialog(null,
listWithOwnersAttachedToVehicle.get(jTable2.getSelectedRow()));
1100     else
1101         JOptionPane.showMessageDialog(null, "No owner selected");
1102 }
1103
1104 public JPanel getPanel()
1105 {
1106     return vehicleTabPanel;
1107 } //end getPanel
1108
1109 /**
1110  * Check all fields in vehicle gui is correct
1111  * @return true if correct else false
1112  */
1113 private boolean checkVehicleFields()
1114 {
1115     errorMessage = "";

```

```

1116     if(valueOfVehicleType.equals(PLEASE_SELECT) || valueOfVehicleType.equals(""))
1117     {
1118         errorMessage = TYPE_NOT_SELECTED;
1119         vehicleTypeComboBox.setBackground(Color.PINK);
1120         return false;
1121     } //end if
1122
1123     if(valueOfVehicleBrand.equals("") || valueOfVehicleBrand.equals(PLEASE_ENTER))
1124     {
1125         errorMessage = BRAND_EMPTY;
1126         vehicleBrandComboBox.setBackground(Color.PINK);
1127         vehicleBrandComboBox.updateUI();
1128         return false;
1129     } //end if
1130
1131     if(vehicleModelTextField.getText().equals(""))
1132     {
1133         errorMessage = MODEL_EMPTY;
1134         vehicleModelTextField.setBackground(Color.PINK);
1135         return false;
1136     } //end if
1137
1138     if(vehicleYearTextField.getText().equals(""))
1139     {
1140         errorMessage = YEAR_EMPTY;
1141         vehicleYearTextField.setBackground(Color.PINK);
1142         return false;
1143     } //end if
1144
1145     if(!CheckInput.correctYear(vehicleYearTextField.getText()))
1146     {
1147         errorMessage = YEAR_INVALID;
1148         vehicleYearTextField.setBackground(Color.PINK);
1149         return false;
1150     } //end if
1151
1152     if(!CheckInput.isInt(vehicleYearTextField.getText()))
1153     {
1154         errorMessage = YEAR_NOT_NUMBER;
1155         vehicleYearTextField.setBackground(Color.PINK);
1156         return false;
1157     } //end if
1158
1159     return true;
1160 }
1161
1162 //end checkVehicleFields
1163 /**
1164  * Return vehicles saved status
1165  * @return vehicles saved status
1166  */
1167 public boolean getVehicleSaved()
1168 {
1169     return vehicleSaved;
1170 }
1171
1172 /**
1173  * Set the vehicles saved status

```

```

1174 * @param vehicleSaved boolean vehicle saved status
1175 */
1176 public void setVehicleSavedStatus(boolean vehicleSaved)
1177 {
1178     this.vehicleSaved = vehicleSaved;
1179 }
1180
1181 /**
1182  * Reset all fields in vehicle GUI
1183  */
1184 private void resetFields()
1185 {
1186     //////////////// RESET ALL FIELDS IN REGISTER OWNER PANE ////////////////
1187
1188     // Reset type combobox
1189     vehicleTypeComboBox.setSelectedIndex(0);
1190     vehicleTypeComboBox.setBackground(Color.white);
1191
1192     // Reset Brand text field
1193     vehicleBrandComboBox.setBackground(Color.white);
1194     vehicleBrandComboBox.setSelectedIndex(0);
1195
1196     // Reset Model text field
1197     vehicleModelTextField.setBackground(Color.white);
1198     vehicleModelTextField.setText("");
1199
1200     // Reset Year text Field
1201     vehicleYearTextField.setBackground(Color.white);
1202     vehicleYearTextField.setText("");
1203
1204     // Reset Vehile ChassisNumber text field
1205     vehicleChassisNumberTextField.setBackground(Color.white);
1206     vehicleChassisNumberTextField.setText("");
1207
1208     // Reset Licens Plate text field
1209     vehicleLicensePlateTextField.setBackground(Color.white);
1210     vehicleLicensePlateTextField.setText("");
1211
1212     // Set vehicle saved status to false
1213     vehicleSaved = false;
1214
1215     // Reset label selected
1216     labelSelectedVehicle.setText(" ");
1217
1218     //////////////// RESET ALL FIELDS IN SEARCH EDIT VEHICLE PANE ////////////////
1219
1220     attachOwnerButton.setEnabled(false);
1221     editVehicleButton.setEnabled(false);
1222     searchVehicleTextField.setText("");
1223     foundVehiclesJList.setListData(new String[0]);
1224
1225     typeVehileEditSelectComboBox.setSelectedIndex(0);
1226     vehicleBrandEditSelect.setText("");
1227     vehicleModelEditSelect.setText("");
1228     vehicleYearEditSelect.setText("");
1229     vehicleChassisNumberEditSelect.setText("");
1230     vehicleLicensPlateEditSelect.setText("");
1231

```

```

1232 }//End resetFields method
1233
1234 /**
1235  * Disable all buttons and fields in edit vehicle panel
1236  */
1237 private void disableEditVehiclePanel()
1238 {
1239     // Reset type combobox
1240     typeVehileEditSelectComboBox.setSelectedIndex(0);
1241     typeVehileEditSelectComboBox.setEnabled(false);
1242
1243     // Reset brand field
1244     vehicleBrandEditSelect.setBackground(new java.awt.Color(220, 220, 220));
1245     vehicleBrandEditSelect.setText("");
1246     vehicleBrandEditSelect.setEditable(false);
1247
1248     // Reset model field
1249     vehicleModelEditSelect.setBackground(new java.awt.Color(220, 220, 220));
1250     vehicleModelEditSelect.setText("");
1251     vehicleModelEditSelect.setEditable(false);
1252
1253     // Reset year field
1254     vehicleYearEditSelect.setBackground(new java.awt.Color(220, 220, 220));
1255     vehicleYearEditSelect.setText("");
1256     vehicleYearEditSelect.setEditable(false);
1257
1258     // Reset the chassis number field
1259     vehicleChassisNumberEditSelect.setEditable(false);
1260     vehicleChassisNumberEditSelect.setText("");
1261     vehicleChassisNumberEditSelect.setBackground(new java.awt.Color(220, 220, 220));
1262
1263     // Reset the licens plate field
1264     vehicleLicensPlateEditSelect.setEditable(false);
1265     vehicleLicensPlateEditSelect.setText("");
1266     vehicleLicensPlateEditSelect.setBackground(new java.awt.Color(220, 220, 220));
1267
1268     // Reset Save edit button
1269     saveEditedVehicleButton.setEnabled(false);
1270
1271     // Reset delete edit button
1272     deleteVehicleButton.setEnabled(false);
1273
1274     // Reset ResetEdit buttob
1275     resetEditVehicleButton.setEnabled(false);
1276 }
1277 /**
1278  * Set type of vehicle for creation
1279  * @param vehicleType
1280  * @return String with name for creation
1281  */
1282 private String setVehicleType(String vehicleType)
1283 {
1284     return this.valueOfVehicleType = vehicleType;
1285 }//end setVehicleType
1286
1287 private String editVehicleType(String vehicleType)
1288 {
1289     return this.valueOfEditedVehicleType = vehicleType;

```

```

1290 }
1291
1292 private String editVehicleModel(String vehicleModel)
1293 {
1294     return this.valueOfEditedVehicleModel = vehicleModel;
1295 }
1296
1297 /**
1298  * Set brand of vehicle for creation
1299  * @param vehicleBrand
1300  * @return String with name for creation
1301  */
1302 private String setVehicleBrand(String vehicleBrand)
1303 {
1304     return this.valueOfVehicleBrand = vehicleBrand;
1305 } //end setVehicleBrand
1306
1307 /**
1308  * Fill name in autocomplete vectors for VehicleGUI
1309  */
1310 public void bildAutoCompleteVectors()
1311 {
1312     //brandNameVector.addAll(FileReaderController.readForAutocomplete(VEHICLE_BRAND_CSV));
1313     brandNameVector.addAll(AutocompleteDAO.getInstance().getSuggestions("VehicleBrand"));
1314     //vehicleTypeVector.addAll(FileReaderController.readForAutocomplete(VEHICLE_TYPE_CSV));
1315     vehicleTypeVector.addAll(AutocompleteDAO.getInstance().getSuggestions("VehicleType"));
1316     Collections.sort(brandNameVector);
1317     Collections.sort(vehicleTypeVector);
1318     addHeadersInJComboBox();
1319 } //end method bildAutoCompleteVectors
1320
1321 /**
1322  * Return the vector with brands
1323  * @return vector with brands
1324  */
1325 public static Vector getBrandVector()
1326 {
1327     return brandNameVector;
1328 }
1329
1330 /**
1331  * Add headers to Vector for JComboBox
1332  */
1333 private void addHeadersInJComboBox()
1334 {
1335     brandNameVector.add(0, PLEASE_ENTER);
1336     vehicleTypeVector.add(0, PLEASE_SELECT);
1337 } //end setHeadersInJComboBox
1338
1339 /**
1340  * Remove the headers in the Vectors before updating the file/database
1341  */
1342 private void removeHeadersInJComboBox()
1343 {
1344     brandNameVector.remove(0);
1345     vehicleTypeVector.remove(0);
1346 } //end removeHeadersInJComboBox
1347

```

```

1348 /**
1349  * If there is typed a new brand it will ask for adding it to vector
1350  */
1351 private void addNewBrandToVector()
1352 {
1353     Iterator iter = brandNameVector.iterator();
1354     System.out.println(valueOfVehicleBrand);
1355     boolean isInVector = false;
1356     while(iter.hasNext())
1357     {
1358         String actB = (String) iter.next();
1359         if(actB.equalsIgnoreCase(valueOfVehicleBrand))
1360         {
1361             isInVector = true;
1362         } //end if
1363     } //end while
1364
1365     if(!isInVector)
1366     {
1367         int answer = JOptionPane.showConfirmDialog(this, valueOfVehicleBrand +
1368             " is a new brand.\nDo you want to save it?\n(Check the spelling!);
1369         if (answer == JOptionPane.YES_OPTION)
1370         {
1371             removeHeadersInJComboBox();
1372             brandNameVector.add(valueOfVehicleBrand);
1373             AutocompleteDAO.getInstance().createNewSuggestion("VehicleBrand", valueOfVehicleBrand);
1374             Collections.sort(brandNameVector);
1375             addHeadersInJComboBox();
1376             vehicleBrandComboBox.updateUI();
1377         } //end if
1378
1379         if (answer == JOptionPane.NO_OPTION || answer == JOptionPane.CANCEL_OPTION || answer ==
1380             JOptionPane.CLOSED_OPTION)
1381         {
1382             return;
1383         } //end if
1384     } //end if
1385 } //end method addnewbrandtovector
1386
1387 /**
1388  * This method ask the controller-layer to perform a search for a vehicle
1389  * Search results are set from the controller-layer as multiply list of Vehicle-objects, String and ints
1390  *
1391  * The variables model and vType in every object in the actFoundsVehicleObjects list
1392  * is put together as a string and inserted to an
1393  * arraylist( actFoundsBrandModel ) which will be converted to an array
1394  *
1395  * The same applies to the list of vehilce id
1396  * @param <T>
1397  * @param searchCriteria
1398  * @return void
1399  */
1400 public <T> String[] enterSearchDetails(T searchCriteria)
1401 {
1402     // This arrayLsit will hold the vehicles Brand and Model for presentation
1403     actFoundsBrandModel = new ArrayList<String>();
1404
1405     // This arrayList will hold the vehicles ID

```

```

1405     actFoundVehicle = new ArrayList<String>();
1406
1407     // This arrayList will hold the vehicle object
1408     actFoundsVehicleObjects = mvc.searchVehicle(searchCriteria);
1409     int i = 0;
1410     for (Vehicle v : actFoundsVehicleObjects)
1411     {
1412         // Brand, model and id in the respective lists
1413         String licensPlate = v.getLicensPlate();
1414         String chassisNumber = v.getChassisNumber();
1415
1416         // If both licensplate and chassis number is empty
1417         if(licensPlate.equals("") && chassisNumber.equals(""))
1418         {
1419             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel());
1420         }
1421         // If only chassisnumber is empty
1422         else if(!licensPlate.equals("") && chassisNumber.equals(""))
1423         {
1424             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
1425         }
1426         // If only licensplate is empty
1427         else if(licensPlate.equals("") && !chassisNumber.equals(""))
1428         {
1429             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getChassisNumber() + " )");
1430         }
1431         // Neither licensplate or chassisnumber is empty
1432         else
1433         {
1434             actFoundsBrandModel.add("(" + v.getVehicleID() + ") " + v.getBrand() + " " + v.getModel() + " ( " +
v.getLicensPlate() + " )");
1435         }
1436         actFoundVehicle.add(v.getVehicleID());
1437         i++;
1438     } //end for each
1439
1440     String[] arrActFounds = actFoundsBrandModel.toArray(new String[0]);
1441     arrActFoundsVehicle = actFoundVehicle.toArray(new String[0]);
1442
1443     return arrActFounds;
1444 }
1445
1446 /**
1447  * Register a new Vehicle
1448  */
1449 private void registerVehicle()
1450 {
1451     // First check if type, brand, model and year is ok
1452     if(valueOfVehicleType.equals(PLEASE_SELECT) || valueOfVehicleType.equals(""))
1453     {
1454         errorMessage = errorMessage + " " + TYPE_NOT_SELECTED;
1455         vehicleTypeComboBox.setBackground(Color.PINK);
1456     }
1457
1458     if(valueOfVehicleBrand.equals(PLEASE_ENTER) || valueOfVehicleBrand.equals(""))
1459     {

```



```

1460     errorMessage = errorMessage + " " + BRAND_EMPTY;
1461     vehicleBrandComboBox.setBackground(Color.PINK);
1462 }
1463
1464 if(vehicleModelTextField.getText().equals(""))
1465 {
1466     errorMessage = errorMessage + " " + MODEL_EMPTY;
1467     vehicleModelTextField.setBackground(Color.PINK);
1468 }
1469
1470 if(vehicleYearTextField.getText().equals(""))
1471 {
1472     errorMessage = errorMessage + " " + YEAR_EMPTY;
1473     vehicleYearTextField.setBackground(Color.PINK);
1474 }
1475
1476
1477
1478 // Check if all fields are ok
1479 if(checkVehicleFields())
1480 {
1481     addNewBrandToVector();
1482
1483     mvc.makeNewVehicle(valueOfVehicleType,
1484         valueOfVehicleBrand,
1485         vehicleModelTextField.getText(),
1486         Integer.parseInt(vehicleYearTextField.getText()),
1487         vehicleChassisNumberTextField.getText(),
1488         vehicleLicensePlateTextField.getText());
1489     resultLabelVehicle.setText("Vehicle was saved");
1490     resetFields();
1491 } //end if
1492 else
1493     resultLabelVehicle.setText(errorMessage);
1494 }
1495
1496 private void updateVehicle()
1497 {
1498     // First check if type, brand, model and year is ok
1499     if(valueOfEditedVehicleType.equals(PLEASE_SELECT) || valueOfEditedVehicleType.equals(""))
1500     {
1501         errorMessage = errorMessage + " " + TYPE_NOT_SELECTED;
1502         typeVehileEditSelectComboBox.setBackground(Color.PINK);
1503     }
1504
1505     if(vehicleBrandEditSelect.getText().equals(""))
1506     {
1507         errorMessage = errorMessage + " " + BRAND_EMPTY;
1508         vehicleBrandComboBox.setBackground(Color.PINK);
1509     }
1510
1511     if(vehicleModelEditSelect.getText().equals(""))
1512     {
1513         errorMessage = errorMessage + " " + MODEL_EMPTY;
1514         vehicleModelEditSelect.setBackground(Color.PINK);
1515     }
1516
1517     if(vehicleYearEditSelect.getText().equals(""))

```

```

1518     {
1519         errorMessage = errorMessage + " " + YEAR_EMPTY;
1520         vehicleYearEditSelect.setBackground(Color.PINK);
1521     }
1522     addNewBrandToVector();
1523     mvc.editVehicle(selectedVehicle.getVehicleID(), valueOfEditedVehicleType,
1524         vehicleBrandEditSelect.getText(),
1525         vehicleModelEditSelect.getText(),
1526         Integer.parseInt(vehicleYearEditSelect.getText()),
1527         vehicleChassisNumberEditSelect.getText(),
1528         vehicleLicensePlateEditSelect.getText());
1529     //resultLabelVehicle.setText("Vehicle was saved");
1530     JOptionPane.showMessageDialog(null, "Vehicle Successfully updated !", "Successfully updated",
JOptionPane.INFORMATION_MESSAGE);
1531     resetFields();
1532     disableEditVehiclePanel();
1533
1534 }
1535
1536 /**
1537  * Delete a vehicle if selected
1538  */
1539 public void deleteVehicle()
1540 {
1541     options = JOptionPane.showConfirmDialog(null, "Are you sure you want to delete this vehicle??", "Do You
want to delete", JOptionPane.YES_NO_OPTION);
1542     if(options==JOptionPane.YES_OPTION)
1543     {
1544         mvc.deleteVehicle(selectedVehicle.getVehicleID());
1545         resetFields();
1546         disableEditVehiclePanel();
1547         labelSelectedVehicle.setText("Vehicle Deleted !");
1548     } // End id
1549     else
1550     {
1551         return;
1552     } // End else
1553 } // end delete vehicle method
1554
1555 private void setTableWithOwners(List<Owner> owners, boolean[] activeNess)
1556 {
1557     //Build an array with the column names
1558     String[] columnNames = {"Name", "Phone", "E-mail", "Active"};
1559
1560     //Build a two-dim array to be used in the jTable (that's right - we know that thing too)
1561     //We are going to present 4 columns and as many rows as there are owners to that vehicle
1562     Object[][] data = new Object[owners.size()][4];
1563     int counter = 0;
1564     for (Owner o : owners)
1565     {
1566         for(int i = 0; i < 4; i++)
1567         {
1568             data[counter][0] = o.getFirstName();
1569             data[counter][1] = o.getPhone();
1570             data[counter][2] = o.getEmail();
1571             //My tool says that "Creating new Boolean is inefficient" ... but it outputs a lot of shit anyway :p
1572             data[counter][3] = new Boolean(activeNess[counter]);
1573         }

```

```

1574     counter++;
1575 } //end for
1576
1577 //Now set set the jTable - yeah :o)
1578 jTable2.setModel(new javax.swing.table.DefaultTableModel(data, columnNames)
1579 {
1580     //In order to get this table to show a checkbox for a boolean, we'll define a class array
1581     Class[] types = new Class []
1582     {
1583         //The types are: Object, Object, Object, Boolean
1584         java.lang.Object.class, java.lang.Object.class, java.lang.Object.class, java.lang.Boolean.class
1585     };
1586     //Override - me like <3
1587     @Override
1588     public Class getColumnClass(int columnIndex)
1589     {
1590         return types [columnIndex];
1591     }
1592 });
1593 jTable2.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
1594 }
1595
1596 private void getOwnersAttachedToVehicle(List<Owner> searchVehicleWithOwner, boolean[] activeNess) {
1597     listWithOwnersAttachedToVehicle = searchVehicleWithOwner;
1598     arrayWithOwnersAttachedToVehicle = activeNess;
1599 }
1600
1601 // Variables declaration - do not modify
1602 private javax.swing.JLabel addressLabel4;
1603 private javax.swing.JLabel addressLabel5;
1604 private javax.swing.JLabel addressLabel6;
1605 private javax.swing.JLabel addressLabel7;
1606 private javax.swing.JButton attachOwnerButton;
1607 private javax.swing.JLabel countryLabel2;
1608 private javax.swing.JLabel countryLabel3;
1609 private javax.swing.JButton deleteVehicleButton;
1610 private javax.swing.JButton editVehicleButton;
1611 private javax.swing.JPanel editVehiclePanel1;
1612 private javax.swing.JList foundVehiclesJList;
1613 private javax.swing.JButton jButton1;
1614 private javax.swing.JButton jButton9;
1615 private javax.swing.JLabel jLabel4;
1616 private javax.swing.JLabel jLabel5;
1617 private javax.swing.JLabel jLabel6;
1618 private javax.swing.JTabbedPane jManageVehiclePane;
1619 private javax.swing.JPanel jPanel1;
1620 private javax.swing.JLabel jResultLabelOwner3;
1621 private javax.swing.JScrollPane jScrollPane1;
1622 private javax.swing.JScrollPane jScrollPane2;
1623 private javax.swing.JScrollPane jScrollPane4;
1624 private javax.swing.JSeparator jSeparator2;
1625 private javax.swing.JTable jTable2;
1626 private javax.swing.JLabel labelSelectedVehicle;
1627 private javax.swing.JLabel lastNameLabel2;
1628 private javax.swing.JLabel lastNameLabel3;
1629 private javax.swing.JLabel nameLabel2;
1630 private javax.swing.JLabel nameLabel3;
1631 private javax.swing.JLabel noSearchResultsLabel;

```

```

1632 private javax.swing.JPanel ownerInformationPanel;
1633 private javax.swing.JPanel ownersPanel;
1634 private javax.swing.JPanel registerVehiclePanel;
1635 private javax.swing.JButton resetEditVehicleButton;
1636 private javax.swing.JLabel resultLabelVehicle;
1637 private javax.swing.JButton saveEditedVehicleButton;
1638 private javax.swing.JButton saveVehicleButton;
1639 private javax.swing.JPanel searchEditPanel;
1640 private javax.swing.JPanel searchPanel;
1641 private javax.swing.JButton searchVehicleButton;
1642 private javax.swing.JTextField searchVehicleTextField;
1643 private javax.swing.JButton selectVehicleButton;
1644 private javax.swing.JComboBox typeVehicleEditSelectComboBox;
1645 private javax.swing.JComboBox vehicleBrandComboBox;
1646 private javax.swing.JTextField vehicleBrandEditSelect;
1647 private javax.swing.JTextField vehicleChassisNumberEditSelect;
1648 private javax.swing.JTextField vehicleChassisNumberTextField;
1649 private javax.swing.JTextField vehicleLicensePlateEditSelect;
1650 private javax.swing.JTextField vehicleLicensePlateTextField;
1651 private javax.swing.JPanel vehicleListPanel;
1652 private javax.swing.JTextField vehicleModelEditSelect;
1653 private javax.swing.JTextField vehicleModelTextField;
1654 private javax.swing.JPanel vehicleTabPanel;
1655 private javax.swing.JComboBox vehicleTypeComboBox;
1656 private javax.swing.JTextField vehicleYearEditSelect;
1657 private javax.swing.JTextField vehicleYearTextField;
1658 private javax.swing.JButton viewOwnerButton;
1659 private javax.swing.JLabel zipAndCityLabel2;
1660 private javax.swing.JLabel zipAndCityLabel3;
1661 // End of variables declaration

```

WSGUI

```

1
2 /*
3  * This is the "working" version
4  * All GUI code for
5  * Workstation
6  * WSGUI.java
7  *
8  * Created on 08-12-2010, 10:30:33
9  */
10
11 package tweakmc.view;
12
13 import java.awt.Color;
14 import java.awt.event.KeyEvent;
15 import javax.swing.JOptionPane;
16 import javax.swing.JPanel;
17 import javax.swing.border.TitledBorder;
18 import javax.swing.tree.DefaultMutableTreeNode;
19 import javax.swing.tree.DefaultTreeModel;
20 import tweakmc.control.ManageSpreadsheetController;
21 import tweakmc.control.ManageWorkstationController;
22 import tweakmc.model.Result.Result;
23 import tweakmc.utility.GUIHelpUtil;
24 import tweakmc.view.ManageSpreadsheetView;

```

```

25 import tweakmc.view.resultviewUtility.ResultTreeUtility;
26
27 /**
28  *
29  * @author clausPallisgaardBeck
30  */
31 public class WSGUI extends javax.swing.JFrame
32 {
33     //Instance variables
34     private ManageWorkstationController mwc = new ManageWorkstationController();
35
36     private String vehicleID;
37     private String workstationID;
38     private String result, orderID = "";
39
40
41     //Final variables
42     private final String IMAGE = "Image";
43     private final String VIDEO = "Video";
44     private final String SOUND = "Sound";
45     private final String DOCUMENT = "Document";
46     private final String SPREADSHEET = "Spreadsheet";
47     private final String WHITEBOARD = "Whiteboard";
48
49     private final String EMPTY_ORDERID = "C-1";
50
51     /** Creates new form WSGUI */
52     public WSGUI()
53     {
54         initComponents();
55     }
56
57     /** This method is called from within the constructor to
58      * initialize the form.
59      * WARNING: Do NOT modify this code. The content of this method is
60      * always regenerated by the Form Editor.
61      */
62     @SuppressWarnings("unchecked")
63     // <editor-fold defaultstate="collapsed" desc="Generated Code">
64     private void initComponents() {
65
66         mainWorkstationPanel = new javax.swing.JPanel();
67         tabPaneMainWorkstation = new javax.swing.JTabbedPane();
68         stationID0Tab = new javax.swing.JScrollPane();
69         stationID0Panel = new javax.swing.JPanel();
70         vehicleIDLabel = new javax.swing.JLabel();
71         vehicleIDTextField = new javax.swing.JTextField();
72         workOnButtonStationID0 = new javax.swing.JButton();
73         searchButtonStationID0 = new javax.swing.JButton();
74         uploadPanelStationID0 = new javax.swing.JPanel();
75         uploadImageButtonStationID0 = new javax.swing.JButton();
76         uploadVideoButtonStationID0 = new javax.swing.JButton();
77         uploadSoundButtonStationID0 = new javax.swing.JButton();
78         uploadDocumentButtonStationID0 = new javax.swing.JButton();
79         createPanelStationID0 = new javax.swing.JPanel();
80         createSpreadsheetButtonStationID0 = new javax.swing.JButton();
81         resultTreePanelStationID0 = new javax.swing.JPanel();
82         scrollPaneResultTreeStationID0 = new javax.swing.JScrollPane();

```

```

83     resultTreeStationID0 = new javax.swing.JTree();
84     stationID1Tab = new javax.swing.JScrollPane();
85     stationID1Panel = new javax.swing.JPanel();
86     stationID2Tab = new javax.swing.JScrollPane();
87     stationID2Panel = new javax.swing.JPanel();
88     stationID3Tab = new javax.swing.JScrollPane();
89     stationID3Panel = new javax.swing.JPanel();
90
91     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
92
93     vehicleIDLabel.setText("VehicleID:");
94
95     vehicleIDTextField.addFocusListener(new java.awt.event.FocusAdapter() {
96         public void focusGained(java.awt.event.FocusEvent evt) {
97             vehicleIDTextFieldFocusGained(evt);
98         }
99     });
100
101     workOnButtonStationID0.setText("Work on vehicle");
102     workOnButtonStationID0.addActionListener(new java.awt.event.ActionListener() {
103         public void actionPerformed(java.awt.event.ActionEvent evt) {
104             workOnButtonStationID0ActionPerformed(evt);
105         }
106     });
107     workOnButtonStationID0.addKeyListener(new java.awt.event.KeyAdapter() {
108         public void keyPressed(java.awt.event.KeyEvent evt) {
109             workOnButtonStationID0KeyPressed(evt);
110         }
111     });
112
113     searchButtonStationID0.setText("Search");
114
115     uploadPanelStationID0.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Upload",
javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP, new java.awt.Font("Tahoma", 1, 12)));
// NOI18N
116
117     uploadImageButtonStationID0.setText("Image");
118     uploadImageButtonStationID0.setEnabled(false);
119
120     uploadVideoButtonStationID0.setText("Video");
121     uploadVideoButtonStationID0.setEnabled(false);
122
123     uploadSoundButtonStationID0.setText("Sound");
124     uploadSoundButtonStationID0.setEnabled(false);
125
126     uploadDocumentButtonStationID0.setText("Document");
127     uploadDocumentButtonStationID0.setEnabled(false);
128     uploadDocumentButtonStationID0.addActionListener(new java.awt.event.ActionListener() {
129         public void actionPerformed(java.awt.event.ActionEvent evt) {
130             uploadDocumentButtonStationID0ActionPerformed(evt);
131         }
132     });
133
134     javax.swing.GroupLayout uploadPanelStationID0Layout = new
javax.swing.GroupLayout(uploadPanelStationID0);
135     uploadPanelStationID0.setLayout(uploadPanelStationID0Layout);
136     uploadPanelStationID0Layout.setHorizontalGroup(
137         uploadPanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

138     .addGroup(uploadPanelStationID0Layout.createSequentialGroup())
139     .addContainerGap()
140
141     .addGroup(uploadPanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
142         .addComponent(uploadImageButtonStationID0)
143         .addComponent(uploadVideoButtonStationID0)
144         .addComponent(uploadSoundButtonStationID0)
145         .addComponent(uploadDocumentButtonStationID0))
146     );
147
148     uploadPanelStationID0Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new
java.awt.Component[] {uploadDocumentButtonStationID0, uploadImageButtonStationID0,
uploadSoundButtonStationID0, uploadVideoButtonStationID0});
149
150     uploadPanelStationID0Layout.setVerticalGroup(
151         uploadPanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
152         .addGroup(uploadPanelStationID0Layout.createSequentialGroup()
153             .addContainerGap()
154             .addComponent(uploadImageButtonStationID0)
155             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
156             .addComponent(uploadVideoButtonStationID0)
157             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
158             .addComponent(uploadSoundButtonStationID0)
159             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
160             .addComponent(uploadDocumentButtonStationID0)
161             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
162     );
163
164     createPanelStationID0.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Create",
javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP, new java.awt.Font("Tahoma", 1, 12)));
// NOI18N
165
166     createSpreadsheetButtonStationID0.setText("Spreadsheet");
167     createSpreadsheetButtonStationID0.setEnabled(false);
168
169     javax.swing.GroupLayout createPanelStationID0Layout = new
javax.swing.GroupLayout(createPanelStationID0);
170     createPanelStationID0.setLayout(createPanelStationID0Layout);
171     createPanelStationID0Layout.setHorizontalGroup(
172         createPanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
173         .addGroup(createPanelStationID0Layout.createSequentialGroup()
174             .addContainerGap()
175             .addComponent(createSpreadsheetButtonStationID0)
176             .addContainerGap(17, Short.MAX_VALUE))
177     );
178     createPanelStationID0Layout.setVerticalGroup(
179         createPanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
180         .addGroup(createPanelStationID0Layout.createSequentialGroup()
181             .addContainerGap()
182             .addComponent(createSpreadsheetButtonStationID0)
183             .addContainerGap(66, Short.MAX_VALUE))
184     );
185
186     resultTreePanelStationID0.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Result",
javax.swing.border.TitledBorder.LEFT, javax.swing.border.TitledBorder.TOP, new java.awt.Font("Tahoma", 1, 12)));
// NOI18N
187

```



```

188     resultTreeStationID0.setModel(new
DefaultTreeModel(ResultTreeUtility.getResultTreeModelWithoutWhiteboard("Results")));
189     resultTreeStationID0.addTreeSelectionListener(new javax.swing.event.TreeSelectionListener() {
190         public void valueChanged(javax.swing.event.TreeSelectionEvent evt) {
191             resultTreeStationID0ValueChanged(evt);
192         }
193     });
194     scrollPaneResultTreeStationID0.setViewportView(resultTreeStationID0);
195
196     javax.swing.GroupLayout resultTreePanelStationID0Layout = new
javax.swing.GroupLayout(resultTreePanelStationID0);
197     resultTreePanelStationID0.setLayout(resultTreePanelStationID0Layout);
198     resultTreePanelStationID0Layout.setHorizontalGroup(
199         resultTreePanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
200         .addGroup(resultTreePanelStationID0Layout.createSequentialGroup())
201         .addContainerGap()
202         .addComponent(scrollPaneResultTreeStationID0, javax.swing.GroupLayout.DEFAULT_SIZE, 341,
Short.MAX_VALUE)
203         .addContainerGap()
204     );
205     resultTreePanelStationID0Layout.setVerticalGroup(
206         resultTreePanelStationID0Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
207         .addGroup(resultTreePanelStationID0Layout.createSequentialGroup())
208         .addContainerGap()
209         .addComponent(scrollPaneResultTreeStationID0, javax.swing.GroupLayout.DEFAULT_SIZE, 244,
Short.MAX_VALUE)
210         .addContainerGap()
211     );
212
213     javax.swing.GroupLayout stationID0PanelLayout = new javax.swing.GroupLayout(stationID0Panel);
214     stationID0Panel.setLayout(stationID0PanelLayout);
215     stationID0PanelLayout.setHorizontalGroup(
216         stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
217         .addGroup(stationID0PanelLayout.createSequentialGroup())
218         .addContainerGap()
219         .addGroup(stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
220             .addGroup(stationID0PanelLayout.createSequentialGroup())
221             .addComponent(vehicleIDLabel)
222             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
223             .addComponent(vehicleIDTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 68,
javax.swing.GroupLayout.PREFERRED_SIZE)
224             .addGap(18, 18, 18)
225             .addComponent(workOnButtonStationID0)
226             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
227             .addComponent(searchButtonStationID0))
228         .addGroup(stationID0PanelLayout.createSequentialGroup())
229         .addGroup(stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
230             .addComponent(uploadPanelStationID0, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
231             .addComponent(createPanelStationID0, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
232         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
233         .addComponent(resultTreePanelStationID0, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
234         .addContainerGap(229, Short.MAX_VALUE))

```



```

235     );
236
237     stationID0PanelLayout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[]
{searchButtonStationID0, workOnButtonStationID0});
238
239     stationID0PanelLayout.setVerticalGroup(
240         stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
241         .addGroup(stationID0PanelLayout.createSequentialGroup())
242         .addContainerGap()
243
244         .addGroup(stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
245             .addComponent(vehicleIDLabel)
246             .addComponent(vehicleIDTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
247             .addComponent(workOnButtonStationID0)
248             .addComponent(searchButtonStationID0))
249             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
250             .addGroup(stationID0PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
251                 .addGroup(stationID0PanelLayout.createSequentialGroup()
252                     .addComponent(uploadPanelStationID0, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
253                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
254                     .addComponent(createPanelStationID0, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
255                     .addComponent(resultTreePanelStationID0, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
256                     .addContainerGap(235, Short.MAX_VALUE))
257         );
258
259     stationID0Tab.setViewportView(stationID0Panel);
260
261     tabPaneMainWorkstation.addTab(mwc.getNameForWorkstation("S0"), stationID0Tab);
262
263     javax.swing.GroupLayout stationID1PanelLayout = new javax.swing.GroupLayout(stationID1Panel);
264     stationID1Panel.setLayout(stationID1PanelLayout);
265     stationID1PanelLayout.setHorizontalGroup(
266         stationID1PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
267         .addGap(0, 655, Short.MAX_VALUE)
268     );
269     stationID1PanelLayout.setVerticalGroup(
270         stationID1PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
271         .addGap(0, 397, Short.MAX_VALUE)
272     );
273
274     stationID1Tab.setViewportView(stationID1Panel);
275
276     tabPaneMainWorkstation.addTab("tab2", stationID1Tab);
277
278     javax.swing.GroupLayout stationID2PanelLayout = new javax.swing.GroupLayout(stationID2Panel);
279     stationID2Panel.setLayout(stationID2PanelLayout);
280     stationID2PanelLayout.setHorizontalGroup(
281         stationID2PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
282         .addGap(0, 655, Short.MAX_VALUE)
283     );
284     stationID2PanelLayout.setVerticalGroup(
285         stationID2PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
286         .addGap(0, 397, Short.MAX_VALUE)

```

```

286     );
287
288     stationID2Tab.setViewportView(stationID2Panel);
289
290     tabPaneMainWorkstation.addTab("tab3", stationID2Tab);
291
292     javax.swing.GroupLayout stationID3PanelLayout = new javax.swing.GroupLayout(stationID3Panel);
293     stationID3Panel.setLayout(stationID3PanelLayout);
294     stationID3PanelLayout.setHorizontalGroup(
295         stationID3PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
296             .addGap(0, 655, Short.MAX_VALUE)
297     );
298     stationID3PanelLayout.setVerticalGroup(
299         stationID3PanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
300             .addGap(0, 397, Short.MAX_VALUE)
301     );
302
303     stationID3Tab.setViewportView(stationID3Panel);
304
305     tabPaneMainWorkstation.addTab("tab4", stationID3Tab);
306
307     javax.swing.GroupLayout mainWorkstationPanelLayout = new
javax.swing.GroupLayout(mainWorkstationPanel);
308     mainWorkstationPanel.setLayout(mainWorkstationPanelLayout);
309     mainWorkstationPanelLayout.setHorizontalGroup(
310         mainWorkstationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
311             .addComponent(tabPaneMainWorkstation, javax.swing.GroupLayout.DEFAULT_SIZE, 662,
Short.MAX_VALUE)
312     );
313     mainWorkstationPanelLayout.setVerticalGroup(
314         mainWorkstationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
315             .addComponent(tabPaneMainWorkstation, javax.swing.GroupLayout.DEFAULT_SIZE, 427,
Short.MAX_VALUE)
316     );
317
318     for(int i = 0; i < tabPaneMainWorkstation.getTabCount(); i++)
319     { tabPaneMainWorkstation.setTitleAt(i, mwc.getNameForWorkstation("S" + (i))); }
320
321     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
322     getContentPane().setLayout(layout);
323     layout.setHorizontalGroup(
324         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
325             .addGroup(layout.createSequentialGroup()
326                 .addContainerGap()
327                 .addComponent(mainWorkstationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
328                 .addContainerGap()
329             );
330     layout.setVerticalGroup(
331         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
332             .addGroup(layout.createSequentialGroup()
333                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
334                     .addContainerGap()
335                     .addComponent(mainWorkstationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
336                     .addContainerGap()
337                 )
338     );
339     pack();

```

```

339 }// </editor-fold>
340
341 private void workOnButtonStationID0ActionPerformed(java.awt.event.ActionEvent evt) {
342     workOnVehicleID();
343 }
344
345 private void vehicleIDTextFieldFocusGained(java.awt.event.FocusEvent evt) {
346     vehicleIDTextField.setBackground(Color.WHITE);
347     vehicleIDTextField.setText("");
348 }
349
350 private void workOnButtonStationID0KeyPressed(java.awt.event.KeyEvent evt) {
351     if(evt.getKeyCode() == KeyEvent.VK_ENTER)
352     {
353         workOnVehicleID();
354     }//end if
355 }
356
357 private void resultTreeStationID0ValueChanged(javax.swing.event.TreeSelectionEvent evt) {
358     DefaultMutableTreeNode node = (DefaultMutableTreeNode)
359         resultTreeStationID0.getLastSelectedPathComponent();
360
361     if (node == null)
362     {
363         return;
364     }//end if
365
366     Object nodeInfo = node.getUserObject();
367     if (node.isLeaf() && node != null)
368     {
369         if (nodeInfo instanceof Result)
370         {
371             Result actResult = (Result) nodeInfo;
372
373             if(actResult.getResultType().equalsIgnoreCase(SPREADSHEET))
374             {
375
376                 new ManageSpreadsheetView(new
377 ManageSpreadsheetController().findSelectedSpreadsheet(actResult.getPath()),
378                 vehicleID, mwc.getNameForWorkstation(workstationID),
379                 actResult.getResultName().trim(), actResult.getMechanicComment(),
380                 actResult.getCustomerComment());
381             }//end if
382
383             else
384             {
385                 new UploadMediaView(this, actResult);
386             }
387         }//end if
388     }//end if
389
390 private void uploadDocumentButtonStationID0ActionPerformed(java.awt.event.ActionEvent evt) {
391     uploadDocumentToVehicle();
392 }
393 /**
394  * Open the result frame for document upload

```

```

395  */
396  private void uploadDocumentToVehicle()
397  {
398      if(orderID.equalsIgnoreCase(""))
399      {
400          orderID = EMPTY_ORDERID;
401      }
402      new UploadMediaView(this, vehicleID, workstationID, orderID, DOCUMENT);
403  } //end method uploadDocumentToVehicle()
404
405  /**
406   * Gives the panel for GUI
407   * @return JPanel with all design
408   */
409  public JPanel getWorkstationPanel()
410  {
411      return mainWorkstationPanel;
412  } //end getWorkstationPanel()
413
414  /**
415   * Confirm if the selected vehicleID exist, if it does, user have to
416   * confirm that it is the right vehicle
417   */
418  private void workOnVehicleID()
419  {
420      result = mwc.getInfoFromVehicleID(vehicleIDTextField.getText());
421      if(result.equalsIgnoreCase(""))
422      {
423          vehicleIDTextField.setBackground(Color.PINK);
424          JOptionPane.showMessageDialog(null, "Vehicle does not exist",
425              getStationName(),
426              JOptionPane.WARNING_MESSAGE);
427      } //end if
428
429      else
430      {
431          int answer = JOptionPane.showConfirmDialog(null, "Is this correct?\n"
432              + result, getStationName(),
433              JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE);
434
435          if(answer != JOptionPane.YES_OPTION)
436          {
437              vehicleIDTextField.setText("");
438              return;
439          } //end if
440          else
441          {
442              activateButtons();
443              vehicleID = vehicleIDTextField.getText();
444              workstationID = "S" + tabPaneMainWorkstation.getSelectedIndex();
445              createResultTree();
446          } //end else
447      } //end else
448  } //end method workOnVehicleID()
449
450  private void createResultTree()
451  {
452      ResultTreeUtility.setNode(mwc.getResultForActVehicleAndWorkstation(vehicleID, workstationID));

```

```

453     ResultTreeUtility.getResultTreeModelWithoutWhiteboard(result);
454 }
455 /**
456  * Activate all buttons if vehicleID is selected
457  */
458 private void activateButtons()
459 {
460     uploadDocumentButtonStationID0.setEnabled(true);
461     uploadImageButtonStationID0.setEnabled(true);
462     uploadSoundButtonStationID0.setEnabled(true);
463     uploadVideoButtonStationID0.setEnabled(true);
464     createSpreadsheetButtonStationID0.setEnabled(true);
465 } //end method activateButtons()
466
467 /**
468  * Deactivate all buttons if vehicleID is selected
469  */
470 private void deActivateButtons()
471 {
472     uploadDocumentButtonStationID0.setEnabled(false);
473     uploadImageButtonStationID0.setEnabled(false);
474     uploadSoundButtonStationID0.setEnabled(false);
475     uploadVideoButtonStationID0.setEnabled(false);
476     createSpreadsheetButtonStationID0.setEnabled(false);
477 } //end method deActivateButtons()
478
479 /**
480  * Return the name for the tab where user is
481  * @return name on active tab
482  */
483 public String getStationName()
484 {
485     return tabPaneMainWorkstation.getTitleAt(tabPaneMainWorkstation.getSelectedIndex());
486 } //end method getStationName()
487
488 /**
489  * @param args the command line arguments
490  */
491 public static void main(String args[]) {
492     java.awt.EventQueue.invokeLater(new Runnable() {
493         public void run() {
494             new WSGUI().setVisible(true);
495         }
496     });
497 }
498
499 // Variables declaration - do not modify
500 private javax.swing.JPanel createPanelStationID0;
501 private javax.swing.JButton createSpreadsheetButtonStationID0;
502 private javax.swing.JPanel mainWorkstationPanel;
503 private javax.swing.JPanel resultTreePanelStationID0;
504 private javax.swing.JTree resultTreeStationID0;
505 private javax.swing.JScrollPane scrollPaneResultTreeStationID0;
506 private javax.swing.JButton searchButtonStationID0;
507 private javax.swing.JPanel stationID0Panel;
508 private javax.swing.JScrollPane stationID0Tab;
509 private javax.swing.JPanel stationID1Panel;
510 private javax.swing.JScrollPane stationID1Tab;

```

```

511 private javax.swing.JPanel stationID2Panel;
512 private javax.swing.JScrollPane stationID2Tab;
513 private javax.swing.JPanel stationID3Panel;
514 private javax.swing.JScrollPane stationID3Tab;
515 private javax.swing.JTabbedPane tabPaneMainWorkstation;
516 private javax.swing.JButton uploadDocumentButtonStationID0;
517 private javax.swing.JButton uploadImageButtonStationID0;
518 private javax.swing.JPanel uploadPanelStationID0;
519 private javax.swing.JButton uploadSoundButtonStationID0;
520 private javax.swing.JButton uploadVideoButtonStationID0;
521 private javax.swing.JLabel vehicleIDLabel;
522 private javax.swing.JTextField vehicleIDTextField;
523 private javax.swing.JButton workOnButtonStationID0;
524 // End of variables declaration

```

WaitFrame

```

1
2 /*
3  * All GUI code for
4  * Waitframe
5  *
6  * WaitFrame.java
7  *
8  * Created on 17-11-2010, 08:21:55
9  */
10
11 package tweakmc.view;
12
13 import java.awt.Dimension;
14 import java.awt.Toolkit;
15 import java.awt.event.ActionEvent;
16 import java.awt.event.ActionListener;
17 import javax.swing.Timer;
18
19 /**
20  *
21  * @author clausPallisgaardBeck
22  */
23 public class WaitFrame extends javax.swing.JFrame {
24
25     /** Creates new form WaitFrame */
26     public WaitFrame(Dimension size)
27     {
28         Dimension waitFrameSize = getSize();
29
30         initComponents();
31         setLocation(((size.width/2)-waitFrameSize.width), ((size.height/2)-waitFrameSize.height));
32
33         pb.setValue(0);
34         pb.setStringPainted(true);
35         headerLabel.setText("TweakMC - working");
36
37         setVisible(true);
38         startWait();
39     }
40

```

```

41  /**
42   * Start waiting time
43   */
44  public void startWait()
45  {
46      //Create a timer.
47      timer = new Timer(interval, new ActionListener()
48      {
49          public void actionPerformed(ActionEvent evt)
50          {
51              if (i == 5)
52              {
53                  Toolkit.getDefaultToolkit().beep();
54                  String str = "<html>" + "<font color=\"#008000\">" + "<b>" +
55                      "File sendt." + "</b>" + "</font>" + "</html>";
56                  statusLabel.setText(str);
57              } //end if
58
59              if (i == 6)
60              {
61                  timer.stop();
62                  pb.setValue(0);
63                  killWait();
64              } //end if
65              i = i + 1;
66              pb.setValue(i);
67          } //end actionPerformed
68      }); //end method actionlistener
69
70      i = 0;
71      String str = "<html>" + "<font color=\"#FF0000\">" + "<b>" +
72          "Please wait - sending file.." + "</b>" + "</font>" + "</html>";
73      statusLabel.setText(str);
74      timer.start();
75  } //end method starWait
76
77  /**
78   * Kill the waiting time
79   */
80  public void killWait()
81  {
82      dispose();
83  } //end method killWait
84
85  /** This method is called from within the constructor to
86   * initialize the form.
87   * WARNING: Do NOT modify this code. The content of this method is
88   * always regenerated by the Form Editor.
89   */
90  @SuppressWarnings("unchecked")
91  // <editor-fold defaultstate="collapsed" desc="Generated Code">
92  private void initComponents() {
93
94      headerLabel = new javax.swing.JLabel();
95      pb = new javax.swing.JProgressBar();
96      statusLabel = new javax.swing.JLabel();
97
98      setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

```

```

99     setAlwaysOnTop(true);
100    setResizable(false);
101    setUndecorated(true);
102
103    headerLabel.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N
104    headerLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
105
106    pb.setMaximum(5);
107    pb.setStringPainted(true);
108
109    statusLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
110
111    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
112    getContentPane().setLayout(layout);
113    layout.setHorizontalGroup(
114        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
115            .addGroup(layout.createSequentialGroup()
116                .addContainerGap()
117                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
118                    .addComponent(headerLabel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE)
119                    .addComponent(pb, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE)
120                    .addComponent(statusLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE))
121                .addContainerGap())
122        );
123    layout.setVerticalGroup(
124        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
125            .addGroup(layout.createSequentialGroup()
126                .addContainerGap()
127                .addComponent(headerLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 14,
javax.swing.GroupLayout.PREFERRED_SIZE)
128                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
129                .addComponent(pb, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
130                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
131                .addComponent(statusLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
132                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
133        );
134
135    pack();
136 } // </editor-fold>
137
138 /**
139  * @param args the command line arguments
140  */
141 public static void main(String args[], final Dimension size) {
142     java.awt.EventQueue.invokeLater(new Runnable() {
143         public void run() {
144             new WaitFrame(size).setVisible(true);
145         }
146     });
147 }
148 // Variables declaration - do not modify
149 private javax.swing.JLabel headerLabel;
150 private javax.swing.JProgressBar pb;
151 private javax.swing.JLabel statusLabel;

```



```

152 // End of variables declaration
153 private Timer timer;
154 private final static int interval = 1000;
155 private static int i;
156 }

```

WhiteboardView

```

1
2 /*
3  * All GUI code for
4  * Whiteboard
5  *
6  * WhiteBoardView.java
7  *
8  * Created on 09-12-2010, 09:44:42
9  */
10
11 package tweakmc.view;
12
13 /**
14  *
15  * @author NegoZiatoR
16  */
17 public class WhiteBoardView extends javax.swing.JFrame
18 {
19     // Instance variables
20     private String selectedWhiteboard = "";
21     private int selectedNoOfCylinders = 0;
22
23     // Static variables
24     private static String[] columnNames;
25     private static String[] cylinderNo;
26     private static String vehicleID;
27     private static String workstationID;
28     private static String orderID;
29
30
31     /** Creates new form WhiteBoardView */
32     public WhiteBoardView(String vehicleID, String workstationID, String orderID)
33     {
34         WhiteBoardView.vehicleID = vehicleID;
35         WhiteBoardView.workstationID = workstationID;
36         WhiteBoardView.orderID = orderID;
37         initComponents();
38     } // End of WhiteBoardView constructor
39
40     /** This method is called from within the constructor to
41      * initialize the form.
42      * WARNING: Do NOT modify this code. The content of this method is
43      * always regenerated by the Form Editor.
44      */
45     @SuppressWarnings("unchecked")
46     // <editor-fold defaultstate="collapsed" desc="Generated Code">
47     private void initComponents() {
48
49         mainPanel = new javax.swing.JPanel();

```

```

50  mainPanelScrollPane = new javax.swing.JScrollPane();
51  WhiteBoardPanel = new javax.swing.JPanel();
52  labelWhiteboardFor = new javax.swing.JLabel();
53  labelVehicleID = new javax.swing.JLabel();
54  jSeparator1 = new javax.swing.JSeparator();
55  labelwhiteboardType = new javax.swing.JLabel();
56  comboBoxWhiteboardTypes = new javax.swing.JComboBox();
57  labelNoOfColumns = new javax.swing.JLabel();
58  comboBoxColumns = new javax.swing.JComboBox();
59  jSeparator2 = new javax.swing.JSeparator();
60  buttonOk = new javax.swing.JButton();
61  buttonCancel = new javax.swing.JButton();
62
63  setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
64
65  labelWhiteboardFor.setFont(new java.awt.Font("Tahoma", 0, 18));
66  labelWhiteboardFor.setText("Whiteboard for: ");
67
68  labelVehicleID.setFont(new java.awt.Font("Tahoma", 0, 18));
69  labelVehicleID.setText("vehicleID");
70
71  labelwhiteboardType.setText("Whiteboard Type:");
72
73  comboBoxWhiteboardTypes.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Select
Whiteboard", "Valve Adjustment", "Cylinder Tollerance", "Head Bearing", "Connecting Rod" }));
74
75  labelNoOfColumns.setText("No. of columns:");
76
77  comboBoxColumns.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Select Columns",
"1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16" }));
78  comboBoxColumns.addItemListener(new java.awt.event.ItemListener() {
79      public void itemStateChanged(java.awt.event.ItemEvent evt) {
80          comboBoxColumnsItemStateChanged(evt);
81      }
82  });
83
84  buttonOk.setText("OK");
85  buttonOk.addActionListener(new java.awt.event.ActionListener() {
86      public void actionPerformed(java.awt.event.ActionEvent evt) {
87          buttonOkActionPerformed(evt);
88      }
89  });
90
91  buttonCancel.setText("Cancel");
92
93  javax.swing.GroupLayout WhiteBoardPanelLayout = new javax.swing.GroupLayout(WhiteBoardPanel);
94  WhiteBoardPanel.setLayout(WhiteBoardPanelLayout);
95  WhiteBoardPanelLayout.setHorizontalGroup(
96      WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
97      .addGroup(WhiteBoardPanelLayout.createSequentialGroup()
98          .addContainerGap()
99
.addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
100          .addGroup(WhiteBoardPanelLayout.createSequentialGroup()
101              .addComponent(buttonOk, javax.swing.GroupLayout.PREFERRED_SIZE, 77,
javax.swing.GroupLayout.PREFERRED_SIZE)
102              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 147,
Short.MAX_VALUE)

```

```

103         .addComponent(buttonCancel, javax.swing.GroupLayout.PREFERRED_SIZE, 76,
javax.swing.GroupLayout.PREFERRED_SIZE))
104         .addGroup(WhiteBoardPanelLayout.createSequentialGroup())
105         .addComponent(labelWhiteboardFor, javax.swing.GroupLayout.PREFERRED_SIZE, 140,
javax.swing.GroupLayout.PREFERRED_SIZE)
106         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
107         .addComponent(labelVehicleID, javax.swing.GroupLayout.DEFAULT_SIZE, 150,
Short.MAX_VALUE))
108         .addGroup(WhiteBoardPanelLayout.createSequentialGroup())
109
110     .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
111         .addComponent(labelNoOfColumns, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
112         .addComponent(labelwhiteboardType, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 106, Short.MAX_VALUE))
113         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
114     .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
115         .addComponent(comboBoxColumns, 0, 190, Short.MAX_VALUE)
116         .addComponent(comboBoxWhiteboardTypes, 0, 190, Short.MAX_VALUE)))
117     .addComponent(jSeparator1, javax.swing.GroupLayout.DEFAULT_SIZE, 300, Short.MAX_VALUE)
118     .addComponent(jSeparator2, javax.swing.GroupLayout.DEFAULT_SIZE, 300,
Short.MAX_VALUE))
119     .addContainerGap(115, Short.MAX_VALUE))
120 );
121 WhiteBoardPanelLayout.setVerticalGroup(
122     WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
123     .addGroup(WhiteBoardPanelLayout.createSequentialGroup()
124         .addContainerGap()
125         .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
126             .addComponent(labelVehicleID, javax.swing.GroupLayout.DEFAULT_SIZE, 38,
Short.MAX_VALUE)
127             .addComponent(labelWhiteboardFor, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE))
128             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
129             .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
130             .addGap(18, 18, 18)
131         .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
132             .addComponent(labelwhiteboardType, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE)
133             .addComponent(comboBoxWhiteboardTypes, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
134             .addGap(27, 27, 27)
135         .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
136             .addComponent(labelNoOfColumns)
137             .addComponent(comboBoxColumns, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
138             .addGap(18, 18, 18)
139             .addComponent(jSeparator2, javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
140             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
141         .addGroup(WhiteBoardPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
142             .addComponent(buttonOk)

```

```

142         .addComponent(buttonCancel))
143     .addGap(103, 103, 103))
144 );
145
146 mainPanelScrollPane.setViewportView(WhiteBoardPanel);
147
148 javax.swing.GroupLayout mainPanelLayout = new javax.swing.GroupLayout(mainPanel);
149 mainPanel.setLayout(mainPanelLayout);
150 mainPanelLayout.setHorizontalGroup(
151     mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
152     .add(mainPanelScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 382,
Short.MAX_VALUE)
153 );
154 mainPanelLayout.setVerticalGroup(
155     mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
156     .add(mainPanelScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 300,
Short.MAX_VALUE)
157 );
158
159 javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
160 getContentPane().setLayout(layout);
161 layout.setHorizontalGroup(
162     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
163     .add(mainPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
164 );
165 layout.setVerticalGroup(
166     layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
167     .add(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
168 );
169
170 pack();
171 }// </editor-fold>
172
173 private void buttonOkActionPerformed(java.awt.event.ActionEvent evt) {
174     // TODO add your handling code here:
175     if(comboBoxWhiteboardTypes.getSelectedIndex() > 0)
176     {
177         selectedWhiteboard = (String) comboBoxWhiteboardTypes.getSelectedItem();
178         if(selectedWhiteboard.equals("Valve Adjustment"))
179         {
180             if(comboBoxColumns.getSelectedIndex() > 0)
181             {
182                 selectedNoOfCylinders = Integer.parseInt((String)comboBoxColumns.getSelectedItem());
183                 ValveAdjustmentTableGUI.getInstance(selectedNoOfCylinders, columnNames, cylinderNo,
vehicleID, workstationID, orderID);
184             }
185         }
186         else if(selectedWhiteboard.equals("Cylinder Tollerance"))
187         {
188
189         }
190         else if(selectedWhiteboard.equals("Head Bearing"))
191         {
192
193         }
194         // Selected is Connecting Rod

```

```

195     else
196     {
197
198     } // End of else
199 } // End of if
200 }
201
202 private void comboBoxColumnsItemStateChanged(java.awt.event.ItemEvent evt) {
203     // TODO add your handling code here:
204     if(comboBoxWhiteboardTypes.getSelectedIndex() > 0)
205     {
206         if(comboBoxColumns.getSelectedIndex() > 0)
207         {
208             int noOfCyl = Integer.parseInt((String)comboBoxColumns.getSelectedItem());
209             NameColumns.getInstance(noOfCyl+1);
210         }
211     } // End of if
212 }
213
214 /**
215  * Set the column names
216  * @param columnNames column names to the table
217  */
218 public static void setColumnNames(String[] columnNames)
219 {
220     WhiteBoardView.columnNames = columnNames;
221 } // End of setColumnNames method
222
223 /**
224  * Set the cylinder numbers
225  * @param cylinderNo Array with cylinder numbers
226  */
227 public static void setCylinderNo(String[] cylinderNo)
228 {
229     WhiteBoardView.cylinderNo = cylinderNo;
230 } // End of setCylinderNo method
231
232 /**
233  * @param args the command line arguments
234  */
235 public static void main(String args[]) {
236     java.awt.EventQueue.invokeLater(new Runnable() {
237         public void run() {
238             new WhiteBoardView(vehicleID, workstationID, orderID).setVisible(true);
239         }
240     });
241 }
242
243 // Variables declaration - do not modify
244 private javax.swing.JPanel WhiteBoardPanel;
245 private javax.swing.JButton buttonCancel;
246 private javax.swing.JButton buttonOk;
247 private javax.swing.JComboBox comboBoxColumns;
248 private javax.swing.JComboBox comboBoxWhiteboardTypes;
249 private javax.swing.JSeparator jSeparator1;
250 private javax.swing.JSeparator jSeparator2;
251 private javax.swing.JLabel labelNoOfColumns;

```

```

253 private javax.swing.JLabel labelVehicleID;
254 private javax.swing.JLabel labelWhiteboardFor;
255 private javax.swing.JLabel labelwhiteboardType;
256 private javax.swing.JPanel mainPanel;
257 private javax.swing.JScrollPane mainPanelScrollPane;
258 // End of variables declaration
259
260 }// End of WhiteBoardView.

```

WorkStationGUI

```

1
2
3 /*
4  * This is the template for WSGUI
5  *
6  * All GUI code for
7  * Workstation
8  *
9  * Created on 07-12-2010, 10:19:13
10 */
11
12 package tweakmc.view;
13
14 import javax.swing.JPanel;
15 import javax.swing.tree.DefaultMutableTreeNode;
16 import tweakmc.control.ManageWorkstationController;
17
18 /**
19  *
20  * @author clausPallisgaardBeck
21  */
22 public class WorkStationGUI extends javax.swing.JPanel
23 {
24     private ManageWorkstationController mwc = new ManageWorkstationController();
25
26
27     /** Creates new form WorkStationGUI */
28     public WorkStationGUI()
29     {
30         initComponents();
31     }
32
33     /**
34      * The actual workstationPanel
35      * @return the panel with all layout
36      */
37     public JPanel getPanel()
38     {
39         return this;
40     } //end getPanel
41
42     public DefaultMutableTreeNode getTreeModel()
43     {
44         javax.swing.tree.DefaultMutableTreeNode treeNode1 = new
javax.swing.tree.DefaultMutableTreeNode("Result for V15 on DynoStation");

```

```

45     javax.swing.tree.DefaultMutableTreeNode treeNode2 = new
javax.swing.tree.DefaultMutableTreeNode("Image");
46     javax.swing.tree.DefaultMutableTreeNode treeNode3 = new
javax.swing.tree.DefaultMutableTreeNode("V1520101207hans.jpg");
47     treeNode2.add(treeNode3);
48     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101206jens.jpg");
49     treeNode2.add(treeNode3);
50     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101205peter.jpg");
51     treeNode2.add(treeNode3);
52     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101204IRS.jpg");
53     treeNode2.add(treeNode3);
54     treeNode1.add(treeNode2);
55     treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Video");
56     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207hans.avi");
57     treeNode2.add(treeNode3);
58     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207jens.avi");
59     treeNode2.add(treeNode3);
60     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207lars.avi");
61     treeNode2.add(treeNode3);
62     treeNode1.add(treeNode2);
63     treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Sound");
64     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207hans.mp3");
65     treeNode2.add(treeNode3);
66     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207lars.mp3");
67     treeNode2.add(treeNode3);
68     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207per.mp3");
69     treeNode2.add(treeNode3);
70     treeNode1.add(treeNode2);
71     treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Document");
72     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207per.docx");
73     treeNode2.add(treeNode3);
74     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207lars.pdf");
75     treeNode2.add(treeNode3);
76     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207hans.xlsx");
77     treeNode2.add(treeNode3);
78     treeNode1.add(treeNode2);
79     treeNode2 = new javax.swing.tree.DefaultMutableTreeNode("Spreadsheet");
80     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207hansi.47");
81     treeNode2.add(treeNode3);
82     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207lars.47");
83     treeNode2.add(treeNode3);
84     treeNode3 = new javax.swing.tree.DefaultMutableTreeNode("V1520101207kaj.47");
85     treeNode2.add(treeNode3);
86     treeNode1.add(treeNode2);
87     return treeNode1;
88 }
89
90 /** This method is called from within the constructor to
91  * initialize the form.
92  * WARNING: Do NOT modify this code. The content of this method is
93  * always regenerated by the Form Editor.
94  */
95 @SuppressWarnings("unchecked")
96 // <editor-fold defaultstate="collapsed" desc="Generated Code">
97 private void initComponents() {
98     java.awt.GridBagConstraints gridBagConstraints;
99
100     workstationsMainPanel = new javax.swing.JPanel();

```



```
101     tabPanesMain = new javax.swing.JTabbedPane();
102     repairStationScrollPane = new javax.swing.JScrollPane();
103     mainPanelRepairstation = new javax.swing.JPanel();
104     uploadSoundButton = new javax.swing.JButton();
105     uploadDocumentButton = new javax.swing.JButton();
106     createLabel = new javax.swing.JLabel();
107     createSpreadsheetButton = new javax.swing.JButton();
108     jSeparator1 = new javax.swing.JSeparator();
109     labelVehicleIDNorthPanel = new javax.swing.JLabel();
110     vehicleIDTextFieldNorthPanel = new javax.swing.JTextField();
111     workOnButtonRepairstation = new javax.swing.JButton();
112     findVehicleButtonRepairstation = new javax.swing.JButton();
113     uploadVideoButton = new javax.swing.JButton();
114     headerUploads = new javax.swing.JLabel();
115     uploadImageButton = new javax.swing.JButton();
116     jSeparator2 = new javax.swing.JSeparator();
117     jScrollPane1 = new javax.swing.JScrollPane();
118     jTextPane1 = new javax.swing.JTextPane();
119     dynoStationScrollPane = new javax.swing.JScrollPane();
120     dynoStationMainPanel = new javax.swing.JPanel();
121     jPanel1 = new javax.swing.JPanel();
122     vehicleIDLabelDynaStation = new javax.swing.JLabel();
123     jTextField1 = new javax.swing.JTextField();
124     workOnButtonDynaStation = new javax.swing.JButton();
125     jButton1 = new javax.swing.JButton();
126     createPanelDynaStation = new javax.swing.JPanel();
127     uploadImageButtonDynaStation = new javax.swing.JButton();
128     uploadVideoButtonDynaStation = new javax.swing.JButton();
129     uploadSoundButtonDynaStation = new javax.swing.JButton();
130     uploadDocumentButtonDynaStation = new javax.swing.JButton();
131     jPanel2 = new javax.swing.JPanel();
132     createSpreadsheetButtonDynaStation = new javax.swing.JButton();
133     jPanel3 = new javax.swing.JPanel();
134     jScrollPane2 = new javax.swing.JScrollPane();
135     resultTreeDynaStation = new javax.swing.JTree();
136     engineStationScrollPane = new javax.swing.JScrollPane();
137     mainPanelRepairstation2 = new javax.swing.JPanel();
138     uploadSoundButton2 = new javax.swing.JButton();
139     uploadDocumentButton2 = new javax.swing.JButton();
140     createLabel2 = new javax.swing.JLabel();
141     createSpreadsheetButton2 = new javax.swing.JButton();
142     jSeparator5 = new javax.swing.JSeparator();
143     labelVehicleIDNorthPanel2 = new javax.swing.JLabel();
144     vehicleIDTextFieldNorthPanel2 = new javax.swing.JTextField();
145     workOnButtonRepairstation2 = new javax.swing.JButton();
146     findVehicleButtonRepairstation2 = new javax.swing.JButton();
147     uploadVideoButton2 = new javax.swing.JButton();
148     headerUploads2 = new javax.swing.JLabel();
149     uploadImageButton2 = new javax.swing.JButton();
150     jSeparator6 = new javax.swing.JSeparator();
151     jScrollPane3 = new javax.swing.JScrollPane();
152     jTextPane3 = new javax.swing.JTextPane();
153     suspenStationScrollPane = new javax.swing.JScrollPane();
154     mainPanelRepairstation3 = new javax.swing.JPanel();
155     uploadSoundButton3 = new javax.swing.JButton();
156     uploadDocumentButton3 = new javax.swing.JButton();
157     createLabel3 = new javax.swing.JLabel();
158     createSpreadsheetButton3 = new javax.swing.JButton();
```



```

159 jSeparator7 = new javax.swing.JSeparator();
160 labelVehicleIDNorthPanel3 = new javax.swing.JLabel();
161 vehicleIDTextFieldNorthPanel3 = new javax.swing.JTextField();
162 workOnButtonRepairstation3 = new javax.swing.JButton();
163 findVehicleButtonRepairstation3 = new javax.swing.JButton();
164 uploadVideoButton3 = new javax.swing.JButton();
165 headerUploads3 = new javax.swing.JLabel();
166 uploadImageButton3 = new javax.swing.JButton();
167 jSeparator8 = new javax.swing.JSeparator();
168 jScrollPane4 = new javax.swing.JScrollPane();
169 jTextPane4 = new javax.swing.JTextPane();
170
171 setName("Form"); // NOI18N
172
173 workstationsMainPanel.setName("workstationsMainPanel"); // NOI18N
174
175 tabPanesMain.setName("tabPanesMain"); // NOI18N
176
177 repairStationScrollPane.setName("repairStationScrollPane"); // NOI18N
178
179 mainPanelRepairstation.setName("mainPanelRepairstation"); // NOI18N
180 mainPanelRepairstation.setLayout(new java.awt.GridBagLayout());
181
182 uploadSoundButton.setText("Sound");
183 uploadSoundButton.setName("uploadSoundButton"); // NOI18N
184 gridBagConstraints = new java.awt.GridBagConstraints();
185 gridBagConstraints.gridx = 0;
186 gridBagConstraints.gridy = 5;
187 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
188 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
189 mainPanelRepairstation.add(uploadSoundButton, gridBagConstraints);
190
191 uploadDocumentButton.setText("Document");
192 uploadDocumentButton.setName("uploadDocumentButton"); // NOI18N
193 gridBagConstraints = new java.awt.GridBagConstraints();
194 gridBagConstraints.gridx = 0;
195 gridBagConstraints.gridy = 6;
196 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
197 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
198 mainPanelRepairstation.add(uploadDocumentButton, gridBagConstraints);
199
200 createLabel.setFont(new java.awt.Font("Tahoma", 1, 12));
201 createLabel.setText("Create");
202 createLabel.setName("createLabel"); // NOI18N
203 gridBagConstraints = new java.awt.GridBagConstraints();
204 gridBagConstraints.gridx = 0;
205 gridBagConstraints.gridy = 8;
206 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
207 gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
208 mainPanelRepairstation.add(createLabel, gridBagConstraints);
209
210 createSpreadsheetButton.setText("Spreadsheet");
211 createSpreadsheetButton.setName("createSpreadsheetButton"); // NOI18N
212 gridBagConstraints = new java.awt.GridBagConstraints();
213 gridBagConstraints.gridx = 0;
214 gridBagConstraints.gridy = 9;
215 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
216 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;

```

```

217 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
218 mainPanelRepairstation.add(createSpreadsheetButton, gridBagConstraintss);
219
220 jSeparator1.setName("jSeparator1"); // NOI18N
221 gridBagConstraintss = new java.awt.GridBagConstraints();
222 gridBagConstraintss.gridx = 0;
223 gridBagConstraintss.gridy = 7;
224 gridBagConstraintss.fill = java.awt.GridBagConstraints.HORIZONTAL;
225 gridBagConstraintss.insets = new java.awt.Insets(2, 0, 2, 0);
226 mainPanelRepairstation.add(jSeparator1, gridBagConstraintss);
227
228 labelVehicleIDNorthPanel.setText("VehicleID:");
229 labelVehicleIDNorthPanel.setName("labelVehicleIDNorthPanel"); // NOI18N
230 gridBagConstraintss = new java.awt.GridBagConstraints();
231 gridBagConstraintss.gridx = 1;
232 gridBagConstraintss.gridy = 0;
233 gridBagConstraintss.fill = java.awt.GridBagConstraints.BOTH;
234 gridBagConstraintss.anchor = java.awt.GridBagConstraints.WEST;
235 gridBagConstraintss.insets = new java.awt.Insets(0, 0, 0, 2);
236 mainPanelRepairstation.add(labelVehicleIDNorthPanel, gridBagConstraintss);
237
238 vehicleIDTextFieldNorthPanel.setMaximumSize(new java.awt.Dimension(40, 23));
239 vehicleIDTextFieldNorthPanel.setMinimumSize(new java.awt.Dimension(30, 23));
240 vehicleIDTextFieldNorthPanel.setName("vehicleIDTextFieldNorthPanel"); // NOI18N
241 vehicleIDTextFieldNorthPanel.setPreferredSize(new java.awt.Dimension(40, 23));
242 vehicleIDTextFieldNorthPanel.setRequestFocusEnabled(false);
243 gridBagConstraintss = new java.awt.GridBagConstraints();
244 gridBagConstraintss.gridx = 2;
245 gridBagConstraintss.gridy = 0;
246 gridBagConstraintss.fill = java.awt.GridBagConstraints.BOTH;
247 gridBagConstraintss.anchor = java.awt.GridBagConstraints.NORTH;
248 gridBagConstraintss.insets = new java.awt.Insets(0, 0, 0, 2);
249 mainPanelRepairstation.add(vehicleIDTextFieldNorthPanel, gridBagConstraintss);
250
251 workOnButtonRepairstation.setText("Work on Vehicle");
252 workOnButtonRepairstation.setName("workOnButtonRepairstation"); // NOI18N
253 gridBagConstraintss = new java.awt.GridBagConstraints();
254 gridBagConstraintss.gridx = 3;
255 gridBagConstraintss.gridy = 0;
256 gridBagConstraintss.fill = java.awt.GridBagConstraints.BOTH;
257 gridBagConstraintss.anchor = java.awt.GridBagConstraints.NORTH;
258 mainPanelRepairstation.add(workOnButtonRepairstation, gridBagConstraintss);
259
260 findVehicleButtonRepairstation.setText("Search");
261 findVehicleButtonRepairstation.setName("findVehicleButtonRepairstation"); // NOI18N
262 gridBagConstraintss = new java.awt.GridBagConstraints();
263 gridBagConstraintss.gridx = 4;
264 gridBagConstraintss.gridy = 0;
265 gridBagConstraintss.anchor = java.awt.GridBagConstraints.NORTHEAST;
266 mainPanelRepairstation.add(findVehicleButtonRepairstation, gridBagConstraintss);
267
268 uploadVideoButton.setText("Video");
269 uploadVideoButton.setName("uploadVideoButton"); // NOI18N
270 gridBagConstraintss = new java.awt.GridBagConstraints();
271 gridBagConstraintss.gridx = 0;
272 gridBagConstraintss.gridy = 4;
273 gridBagConstraintss.fill = java.awt.GridBagConstraints.HORIZONTAL;
274 gridBagConstraintss.insets = new java.awt.Insets(2, 2, 2, 2);

```

```

275 mainPanelRepairstation.add(uploadVideoButton, gridBagConstraints);
276
277 headerUploads.setFont(new java.awt.Font("Tahoma", 1, 12));
278 headerUploads.setText("Uploads");
279 headerUploads.setName("headerUploads"); // NOI18N
280 gridBagConstraints = new java.awt.GridBagConstraints();
281 gridBagConstraints.gridx = 0;
282 gridBagConstraints.gridy = 2;
283 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
284 mainPanelRepairstation.add(headerUploads, gridBagConstraints);
285
286 uploadImageButton.setText("Image");
287 uploadImageButton.setName("uploadImageButton"); // NOI18N
288 gridBagConstraints = new java.awt.GridBagConstraints();
289 gridBagConstraints.gridx = 0;
290 gridBagConstraints.gridy = 3;
291 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
292 gridBagConstraints.anchor = java.awt.GridBagConstraints.SOUTHWEST;
293 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
294 mainPanelRepairstation.add(uploadImageButton, gridBagConstraints);
295
296 jSeparator2.setName("jSeparator2"); // NOI18N
297 gridBagConstraints = new java.awt.GridBagConstraints();
298 gridBagConstraints.gridx = 0;
299 gridBagConstraints.gridy = 1;
300 gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
301 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
302 gridBagConstraints.insets = new java.awt.Insets(5, 0, 5, 0);
303 mainPanelRepairstation.add(jSeparator2, gridBagConstraints);
304
305 jScrollPane1.setName("jScrollPane1"); // NOI18N
306
307 jTextPane1.setName("jTextPane1"); // NOI18N
308 jTextPane1.setPreferredSize(mainPanelRepairstation.getPreferredSize());
309 jScrollPane1.setViewportView(jTextPane1);
310
311 gridBagConstraints = new java.awt.GridBagConstraints();
312 gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
313 gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
314 mainPanelRepairstation.add(jScrollPane1, gridBagConstraints);
315
316 repairStationScrollPane.setViewportView(mainPanelRepairstation);
317
318 tabPanesMain.addTab("RepairStation", repairStationScrollPane);
319
320 dynoStationScrollPane.setName("dynoStationScrollPane"); // NOI18N
321 dynoStationScrollPane.setPreferredSize(new java.awt.Dimension(640, 540));
322
323 dynoStationMainPanel.setName("dynoStationMainPanel"); // NOI18N
324 dynoStationMainPanel.setPreferredSize(new java.awt.Dimension(640, 540));
325
326 jPanel1.setName("jPanel1"); // NOI18N
327
328 vehicleIDLabelDynoStation.setText("VehicleID:");
329 vehicleIDLabelDynoStation.setName("vehicleIDLabelDynoStation"); // NOI18N
330
331 jTextField1.setName("jTextField1"); // NOI18N
332

```

```

333     workOnButtonDynoStation.setText("Work on vehicle");
334     workOnButtonDynoStation.setName("workOnButtonDynoStation"); // NOI18N
335
336     jButton1.setText("Search");
337     jButton1.setName("jButton1"); // NOI18N
338
339     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
340     jPanel1.setLayout(jPanel1Layout);
341     jPanel1Layout.setHorizontalGroup(
342         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
343             .addGroup(jPanel1Layout.createSequentialGroup()
344                 .addGap(10, 10, 10)
345                 .addComponent(vehicleIDLabelDynoStation)
346                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
347                 .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
348                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
349                 .addComponent(workOnButtonDynoStation)
350                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 142,
Short.MAX_VALUE)
351                 .addComponent(jButton1)
352                 .addGap(10, 10, 10)
353             );
354
355     jPanel1Layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[] {jButton1,
workOnButtonDynoStation});
356
357     jPanel1Layout.setVerticalGroup(
358         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
359             .addGroup(jPanel1Layout.createSequentialGroup()
360                 .addGap(10, 10, 10)
361                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
362                     .addComponent(vehicleIDLabelDynoStation)
363                     .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
364                     .addComponent(workOnButtonDynoStation)
365                     .addComponent(jButton1))
366                 .addGap(10, 10, 10)
367             );
368
369     createPanelDynoStation.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Upload",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma", 1, 12), new java.awt.Color(0,
0, 255))); // NOI18N
370     createPanelDynoStation.setName("createPanelDynoStation"); // NOI18N
371
372     uploadImageButtonDynoStation.setText("Image");
373     uploadImageButtonDynoStation.setName("uploadImageButtonDynoStation"); // NOI18N
374
375     uploadVideoButtonDynoStation.setText("Video");
376     uploadVideoButtonDynoStation.setName("uploadVideoButtonDynoStation"); // NOI18N
377
378     uploadSoundButtonDynoStation.setText("Sound");
379     uploadSoundButtonDynoStation.setName("uploadSoundButtonDynoStation"); // NOI18N
380
381     uploadDocumentButtonDynoStation.setText("Document");
382     uploadDocumentButtonDynoStation.setName("uploadDocumentButtonDynoStation"); // NOI18N
383

```

```

384     javax.swing.GroupLayout createPanelDynoStationLayout = new
javax.swing.GroupLayout(createPanelDynoStation);
385     createPanelDynoStation.setLayout(createPanelDynoStationLayout);
386     createPanelDynoStationLayout.setHorizontalGroup(
387         createPanelDynoStationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
388         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
createPanelDynoStationLayout.createSequentialGroup())
389         .addContainerGap()
390     ).addGroup(createPanelDynoStationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
391         .addComponent(uploadImageButtonDynoStation, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
392         .addComponent(uploadVideoButtonDynoStation, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
393         .addComponent(uploadSoundButtonDynoStation, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 89, Short.MAX_VALUE)
394         .addComponent(uploadDocumentButtonDynoStation,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 89,
Short.MAX_VALUE))
395         .addGap(21, 21, 21))
396 );
397 createPanelDynoStationLayout.setVerticalGroup(
398     createPanelDynoStationLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
399     .addGroup(createPanelDynoStationLayout.createSequentialGroup())
400     .addComponent(uploadImageButtonDynoStation)
401     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
402     .addComponent(uploadVideoButtonDynoStation)
403     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
404     .addComponent(uploadSoundButtonDynoStation)
405     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
406     .addComponent(uploadDocumentButtonDynoStation)
407     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
408 );
409
410 jPanel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Create",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma", 1, 12), new java.awt.Color(0,
0, 255))); // NOI18N
411 jPanel2.setName("jPanel2"); // NOI18N
412 jPanel2.setPreferredSize(new java.awt.Dimension(132, 62));
413
414 createSpreadsheetButtonDynoStation.setText("Spreadsheet");
415 createSpreadsheetButtonDynoStation.setName("createSpreadsheetButtonDynoStation"); // NOI18N
416
417 javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
418 jPanel2.setLayout(jPanel2Layout);
419 jPanel2Layout.setHorizontalGroup(
420     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
421     .addGroup(jPanel2Layout.createSequentialGroup())
422     .addContainerGap()
423     .addComponent(createSpreadsheetButtonDynoStation)
424     .addContainerGap(17, Short.MAX_VALUE))
425 );
426 jPanel2Layout.setVerticalGroup(
427     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
428     .addGroup(jPanel2Layout.createSequentialGroup())
429     .addComponent(createSpreadsheetButtonDynoStation)
430     .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

```



```

431 );
432
433 jPanel3.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Result for VehicleID: " +
TEST_V_ID + " on " + mwc.getNameForWorkstation("S" + tabPanelsMain.getSelectedIndex()),
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma", 1, 12), new java.awt.Color(0,
0, 255))); // NOI18N
434 jPanel3.setName("jPanel3"); // NOI18N
435
436 jScrollPane2.setName("jScrollPane2"); // NOI18N
437
438 resultTreeDynoStation.setModel(new javax.swing.tree.DefaultTreeModel(getTreeModel()));
439 resultTreeDynoStation.setName("resultTreeDynoStation"); // NOI18N
440 jScrollPane2.setViewportView(resultTreeDynoStation);
441
442 javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
443 jPanel3.setLayout(jPanel3Layout);
444 jPanel3Layout.setHorizontalGroup(
445     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
446     .addGroup(jPanel3Layout.createSequentialGroup()
447         .addGap(10, 10, 10)
448         .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 345, Short.MAX_VALUE)
449         .addGap(10, 10, 10)
450     );
451 jPanel3Layout.setVerticalGroup(
452     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
453     .addGroup(jPanel3Layout.createSequentialGroup()
454         .addGap(10, 10, 10)
455         .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 221,
javax.swing.GroupLayout.PREFERRED_SIZE)
456         .addGap(10, 10, 10)
457     );
458
459 javax.swing.GroupLayout dynoStationMainPanelLayout = new
javax.swing.GroupLayout(dynoStationMainPanel);
460 dynoStationMainPanel.setLayout(dynoStationMainPanelLayout);
461 dynoStationMainPanelLayout.setHorizontalGroup(
462     dynoStationMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
463     .addGroup(dynoStationMainPanelLayout.createSequentialGroup()
464         .addGap(10, 10, 10)
465         .addGroup(dynoStationMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
466             .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
467             .addGroup(dynoStationMainPanelLayout.createSequentialGroup()
468                 .addGroup(dynoStationMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
469                     .addComponent(jPanel2, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
470                     .addComponent(createPanelDynoStation, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
471                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
472                 .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
473         .addGap(10, 10, 10)
474     );
475 dynoStationMainPanelLayout.setVerticalGroup(
476     dynoStationMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

477         .addGroup(dynoStationMainPanelLayout.createSequentialGroup())
478         .addContainerGap()
479         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
480         .addGap(18, 18, 18)
481
.addGroup(dynoStationMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
482         .addGroup(dynoStationMainPanelLayout.createSequentialGroup())
483         .addComponent(createPanelDynoStation, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
484         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
485         .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
486         .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE, 270,
javax.swing.GroupLayout.PREFERRED_SIZE))
487         .addContainerGap(196, Short.MAX_VALUE))
488     );
489
490     dynoStationScrollPane.setViewportViewView(dynoStationMainPanel);
491
492     tabPanesMain.addTab("DynoStation", dynoStationScrollPane);
493
494     engineStationScrollPane.setName("engineStationScrollPane"); // NOI18N
495
496     mainPanelRepairstation2.setName("mainPanelRepairstation2"); // NOI18N
497     mainPanelRepairstation2.setLayout(new java.awt.GridBagLayout());
498
499     uploadSoundButton2.setText("Sound");
500     uploadSoundButton2.setName("uploadSoundButton2"); // NOI18N
501     gridBagConstraints = new java.awt.GridBagConstraints();
502     gridBagConstraints.gridx = 0;
503     gridBagConstraints.gridy = 5;
504     gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
505     gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
506     mainPanelRepairstation2.add(uploadSoundButton2, gridBagConstraints);
507
508     uploadDocumentButton2.setText("Document");
509     uploadDocumentButton2.setName("uploadDocumentButton2"); // NOI18N
510     gridBagConstraints = new java.awt.GridBagConstraints();
511     gridBagConstraints.gridx = 0;
512     gridBagConstraints.gridy = 6;
513     gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
514     gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
515     mainPanelRepairstation2.add(uploadDocumentButton2, gridBagConstraints);
516
517     createLabel2.setFont(new java.awt.Font("Tahoma", 1, 12));
518     createLabel2.setText("Create");
519     createLabel2.setName("createLabel2"); // NOI18N
520     gridBagConstraints = new java.awt.GridBagConstraints();
521     gridBagConstraints.gridx = 0;
522     gridBagConstraints.gridy = 8;
523     gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
524     gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
525     mainPanelRepairstation2.add(createLabel2, gridBagConstraints);
526
527     createSpreadsheetButton2.setText("Spreadsheet");
528     createSpreadsheetButton2.setName("createSpreadsheetButton2"); // NOI18N
529     gridBagConstraints = new java.awt.GridBagConstraints();

```

```

530 gridBagConstraints.gridx = 0;
531 gridBagConstraints.gridy = 9;
532 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
533 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
534 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
535 mainPanelRepairstation2.add(createSpreadsheetButton2, gridBagConstraints);
536
537 jSeparator5.setName("jSeparator5"); // NOI18N
538 gridBagConstraints = new java.awt.GridBagConstraints();
539 gridBagConstraints.gridx = 0;
540 gridBagConstraints.gridy = 7;
541 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
542 gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
543 mainPanelRepairstation2.add(jSeparator5, gridBagConstraints);
544
545 labelVehicleIDNorthPanel2.setText("VehicleID:");
546 labelVehicleIDNorthPanel2.setName("labelVehicleIDNorthPanel2"); // NOI18N
547 gridBagConstraints = new java.awt.GridBagConstraints();
548 gridBagConstraints.gridx = 1;
549 gridBagConstraints.gridy = 0;
550 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
551 gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
552 gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 2);
553 mainPanelRepairstation2.add(labelVehicleIDNorthPanel2, gridBagConstraints);
554
555 vehicleIDTextFieldNorthPanel2.setMaximumSize(new java.awt.Dimension(40, 23));
556 vehicleIDTextFieldNorthPanel2.setMinimumSize(new java.awt.Dimension(30, 23));
557 vehicleIDTextFieldNorthPanel2.setName("vehicleIDTextFieldNorthPanel2"); // NOI18N
558 vehicleIDTextFieldNorthPanel2.setPreferredSize(new java.awt.Dimension(40, 23));
559 vehicleIDTextFieldNorthPanel2.setRequestFocusEnabled(false);
560 gridBagConstraints = new java.awt.GridBagConstraints();
561 gridBagConstraints.gridx = 2;
562 gridBagConstraints.gridy = 0;
563 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
564 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
565 gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 2);
566 mainPanelRepairstation2.add(vehicleIDTextFieldNorthPanel2, gridBagConstraints);
567
568 workOnButtonRepairstation2.setText("Work on Vehicle");
569 workOnButtonRepairstation2.setName("workOnButtonRepairstation2"); // NOI18N
570 gridBagConstraints = new java.awt.GridBagConstraints();
571 gridBagConstraints.gridx = 3;
572 gridBagConstraints.gridy = 0;
573 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
574 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
575 mainPanelRepairstation2.add(workOnButtonRepairstation2, gridBagConstraints);
576
577 findVehicleButtonRepairstation2.setText("Search");
578 findVehicleButtonRepairstation2.setName("findVehicleButtonRepairstation2"); // NOI18N
579 gridBagConstraints = new java.awt.GridBagConstraints();
580 gridBagConstraints.gridx = 4;
581 gridBagConstraints.gridy = 0;
582 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
583 mainPanelRepairstation2.add(findVehicleButtonRepairstation2, gridBagConstraints);
584
585 uploadVideoButton2.setText("Video");
586 uploadVideoButton2.setName("uploadVideoButton2"); // NOI18N
587 gridBagConstraints = new java.awt.GridBagConstraints();

```



```

588     gridBagConstraints.gridx = 0;
589     gridBagConstraints.gridy = 4;
590     gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
591     gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
592     mainPanelRepairstation2.add(uploadVideoButton2, gridBagConstraintss);
593
594     headerUploads2.setFont(new java.awt.Font("Tahoma", 1, 12));
595     headerUploads2.setText("Uploads");
596     headerUploads2.setName("headerUploads2"); // NOI18N
597     gridBagConstraintss = new java.awt.GridBagConstraints();
598     gridBagConstraintss.gridx = 0;
599     gridBagConstraintss.gridy = 2;
600     gridBagConstraintss.fill = java.awt.GridBagConstraints.HORIZONTAL;
601     mainPanelRepairstation2.add(headerUploads2, gridBagConstraintss);
602
603     uploadImageButton2.setText("Image");
604     uploadImageButton2.setName("uploadImageButton2"); // NOI18N
605     gridBagConstraintss = new java.awt.GridBagConstraints();
606     gridBagConstraintss.gridx = 0;
607     gridBagConstraintss.gridy = 3;
608     gridBagConstraintss.fill = java.awt.GridBagConstraints.HORIZONTAL;
609     gridBagConstraintss.anchor = java.awt.GridBagConstraints.SOUTHWEST;
610     gridBagConstraintss.insets = new java.awt.Insets(2, 2, 2, 2);
611     mainPanelRepairstation2.add(uploadImageButton2, gridBagConstraintss);
612
613     jSeparator6.setName("jSeparator6"); // NOI18N
614     gridBagConstraintss = new java.awt.GridBagConstraints();
615     gridBagConstraintss.gridx = 0;
616     gridBagConstraintss.gridy = 1;
617     gridBagConstraintss.gridwidth = java.awt.GridBagConstraints.REMAINDER;
618     gridBagConstraintss.fill = java.awt.GridBagConstraints.BOTH;
619     gridBagConstraintss.insets = new java.awt.Insets(5, 0, 5, 0);
620     mainPanelRepairstation2.add(jSeparator6, gridBagConstraintss);
621
622     jScrollPane3.setName("jScrollPane3"); // NOI18N
623
624     jTextPane3.setName("jTextPane3"); // NOI18N
625     jTextPane3.setPreferredSize(mainPanelRepairstation.getPreferredSize());
626     jScrollPane3.setViewportView(jTextPane3);
627
628     gridBagConstraintss = new java.awt.GridBagConstraints();
629     gridBagConstraintss.gridwidth = java.awt.GridBagConstraints.REMAINDER;
630     gridBagConstraintss.gridheight = java.awt.GridBagConstraints.REMAINDER;
631     mainPanelRepairstation2.add(jScrollPane3, gridBagConstraintss);
632
633     engineStationScrollPane.setViewportView(mainPanelRepairstation2);
634
635     tabPanesMain.addTab("EngineStation", engineStationScrollPane);
636
637     suspenStationScrollPane.setName("suspenStationScrollPane"); // NOI18N
638
639     mainPanelRepairstation3.setName("mainPanelRepairstation3"); // NOI18N
640     mainPanelRepairstation3.setLayout(new java.awt.GridBagLayout());
641
642     uploadSoundButton3.setText("Sound");
643     uploadSoundButton3.setName("uploadSoundButton3"); // NOI18N
644     gridBagConstraintss = new java.awt.GridBagConstraints();
645     gridBagConstraintss.gridx = 0;

```

```

646 gridBagConstraints.gridy = 5;
647 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
648 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
649 mainPanelRepairstation3.add(uploadSoundButton3, gridBagConstraints);
650
651 uploadDocumentButton3.setText("Document");
652 uploadDocumentButton3.setName("uploadDocumentButton3"); // NOI18N
653 gridBagConstraints = new java.awt.GridBagConstraints();
654 gridBagConstraints.gridx = 0;
655 gridBagConstraints.gridy = 6;
656 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
657 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
658 mainPanelRepairstation3.add(uploadDocumentButton3, gridBagConstraints);
659
660 createLabel3.setFont(new java.awt.Font("Tahoma", 1, 12));
661 createLabel3.setText("Create");
662 createLabel3.setName("createLabel3"); // NOI18N
663 gridBagConstraints = new java.awt.GridBagConstraints();
664 gridBagConstraints.gridx = 0;
665 gridBagConstraints.gridy = 8;
666 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
667 gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
668 mainPanelRepairstation3.add(createLabel3, gridBagConstraints);
669
670 createSpreadsheetButton3.setText("Spreadsheet");
671 createSpreadsheetButton3.setName("createSpreadsheetButton3"); // NOI18N
672 gridBagConstraints = new java.awt.GridBagConstraints();
673 gridBagConstraints.gridx = 0;
674 gridBagConstraints.gridy = 9;
675 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
676 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
677 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
678 mainPanelRepairstation3.add(createSpreadsheetButton3, gridBagConstraints);
679
680 jSeparator7.setName("jSeparator7"); // NOI18N
681 gridBagConstraints = new java.awt.GridBagConstraints();
682 gridBagConstraints.gridx = 0;
683 gridBagConstraints.gridy = 7;
684 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
685 gridBagConstraints.insets = new java.awt.Insets(2, 0, 2, 0);
686 mainPanelRepairstation3.add(jSeparator7, gridBagConstraints);
687
688 labelVehicleIDNorthPanel3.setText("VehicleID:");
689 labelVehicleIDNorthPanel3.setName("labelVehicleIDNorthPanel3"); // NOI18N
690 gridBagConstraints = new java.awt.GridBagConstraints();
691 gridBagConstraints.gridx = 1;
692 gridBagConstraints.gridy = 0;
693 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
694 gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
695 gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 2);
696 mainPanelRepairstation3.add(labelVehicleIDNorthPanel3, gridBagConstraints);
697
698 vehicleIDTextFieldNorthPanel3.setMaximumSize(new java.awt.Dimension(40, 23));
699 vehicleIDTextFieldNorthPanel3.setMinimumSize(new java.awt.Dimension(30, 23));
700 vehicleIDTextFieldNorthPanel3.setName("vehicleIDTextFieldNorthPanel3"); // NOI18N
701 vehicleIDTextFieldNorthPanel3.setPreferredSize(new java.awt.Dimension(40, 23));
702 vehicleIDTextFieldNorthPanel3.setRequestFocusEnabled(false);
703 vehicleIDTextFieldNorthPanel3.setVerifyInputWhenFocusTarget(false);

```

```

704 gridBagConstraints = new java.awt.GridBagConstraints();
705 gridBagConstraints.gridx = 2;
706 gridBagConstraints.gridy = 0;
707 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
708 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
709 gridBagConstraints.insets = new java.awt.Insets(0, 0, 0, 2);
710 mainPanelRepairstation3.add(vehicleIDTextFieldNorthPanel3, gridBagConstraints);
711
712 workOnButtonRepairstation3.setText("Work on Vehicle");
713 workOnButtonRepairstation3.setName("workOnButtonRepairstation3"); // NOI18N
714 gridBagConstraints = new java.awt.GridBagConstraints();
715 gridBagConstraints.gridx = 3;
716 gridBagConstraints.gridy = 0;
717 gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
718 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTH;
719 mainPanelRepairstation3.add(workOnButtonRepairstation3, gridBagConstraints);
720
721 findVehicleButtonRepairstation3.setText("Search");
722 findVehicleButtonRepairstation3.setName("findVehicleButtonRepairstation3"); // NOI18N
723 gridBagConstraints = new java.awt.GridBagConstraints();
724 gridBagConstraints.gridx = 4;
725 gridBagConstraints.gridy = 0;
726 gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
727 mainPanelRepairstation3.add(findVehicleButtonRepairstation3, gridBagConstraints);
728
729 uploadVideoButton3.setText("Video");
730 uploadVideoButton3.setName("uploadVideoButton3"); // NOI18N
731 gridBagConstraints = new java.awt.GridBagConstraints();
732 gridBagConstraints.gridx = 0;
733 gridBagConstraints.gridy = 4;
734 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
735 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
736 mainPanelRepairstation3.add(uploadVideoButton3, gridBagConstraints);
737
738 headerUploads3.setFont(new java.awt.Font("Tahoma", 1, 12));
739 headerUploads3.setText("Uploads");
740 headerUploads3.setName("headerUploads3"); // NOI18N
741 gridBagConstraints = new java.awt.GridBagConstraints();
742 gridBagConstraints.gridx = 0;
743 gridBagConstraints.gridy = 2;
744 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
745 mainPanelRepairstation3.add(headerUploads3, gridBagConstraints);
746
747 uploadImageButton3.setText("Image");
748 uploadImageButton3.setName("uploadImageButton3"); // NOI18N
749 gridBagConstraints = new java.awt.GridBagConstraints();
750 gridBagConstraints.gridx = 0;
751 gridBagConstraints.gridy = 3;
752 gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
753 gridBagConstraints.anchor = java.awt.GridBagConstraints.SOUTHWEST;
754 gridBagConstraints.insets = new java.awt.Insets(2, 2, 2, 2);
755 mainPanelRepairstation3.add(uploadImageButton3, gridBagConstraints);
756
757 jSeparator8.setName("jSeparator8"); // NOI18N
758 gridBagConstraints = new java.awt.GridBagConstraints();
759 gridBagConstraints.gridx = 0;
760 gridBagConstraints.gridy = 1;
761 gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;

```

```

762     gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
763     gridBagConstraints.insets = new java.awt.Insets(5, 0, 5, 0);
764     mainPanelRepairstation3.add(jSeparator8, gridBagConstraints);
765
766     jScrollPane4.setName("jScrollPane4"); // NOI18N
767
768     jTextPane4.setName("jTextPane4"); // NOI18N
769     jTextPane4.setPreferredSize(mainPanelRepairstation.getPreferredSize());
770     jScrollPane4.setViewportView(jTextPane4);
771
772     gridBagConstraints = new java.awt.GridBagConstraints();
773     gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
774     gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
775     mainPanelRepairstation3.add(jScrollPane4, gridBagConstraints);
776
777     suspenStationScrollPane.setViewportView(mainPanelRepairstation3);
778
779     tabPanesMain.addTab("SuspensionStation", suspenStationScrollPane);
780
781     javax.swing.GroupLayout workstationsMainPanelLayout = new
javax.swing.GroupLayout(workstationsMainPanel);
782     workstationsMainPanel.setLayout(workstationsMainPanelLayout);
783     workstationsMainPanelLayout.setHorizontalGroup(
784         workstationsMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
785             .addGap(0, 531, Short.MAX_VALUE)
786
.addGroup(workstationsMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
787             .addComponent(tabPanesMain, javax.swing.GroupLayout.DEFAULT_SIZE, 531,
Short.MAX_VALUE))
788     );
789     workstationsMainPanelLayout.setVerticalGroup(
790         workstationsMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
791             .addGap(0, 368, Short.MAX_VALUE)
792
.addGroup(workstationsMainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
793             .addComponent(tabPanesMain, javax.swing.GroupLayout.DEFAULT_SIZE, 368,
Short.MAX_VALUE))
794     );
795
796     int tabCount = tabPanesMain.getTabCount();
797     for(int i = 0; i < tabCount; i++)
798     { tabPanesMain.setTitleAt(i, mwc.getNameForWorkstation("S"+(i)));}
799
800     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
801     this.setLayout(layout);
802     layout.setHorizontalGroup(
803         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
804             .addGroup(layout.createSequentialGroup()
805                 .addGap(0, 0, Short.MAX_VALUE)
806                 .addComponent(workstationsMainPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
807                 .addGap(0, 0, Short.MAX_VALUE))
808     );
809     layout.setVerticalGroup(
810         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
811             .addGroup(layout.createSequentialGroup()
812                 .addGap(0, 0, Short.MAX_VALUE)

```

```

813         .addComponent(workstationsMainPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
814         .addGap(0, 0, Short.MAX_VALUE))
815     );
816 }// </editor-fold>
817
818
819 // Variables declaration - do not modify
820 private javax.swing.JLabel createLabel;
821 private javax.swing.JLabel createLabel2;
822 private javax.swing.JLabel createLabel3;
823 private javax.swing.JPanel createPanelDynoStation;
824 private javax.swing.JButton createSpreadsheetButton;
825 private javax.swing.JButton createSpreadsheetButton2;
826 private javax.swing.JButton createSpreadsheetButton3;
827 private javax.swing.JButton createSpreadsheetButtonDynoStation;
828 private javax.swing.JPanel dynoStationMainPanel;
829 private javax.swing.JScrollPane dynoStationScrollPane;
830 private javax.swing.JScrollPane engineStationScrollPane;
831 private javax.swing.JButton findVehicleButtonRepairstation;
832 private javax.swing.JButton findVehicleButtonRepairstation2;
833 private javax.swing.JButton findVehicleButtonRepairstation3;
834 private javax.swing.JLabel headerUploads;
835 private javax.swing.JLabel headerUploads2;
836 private javax.swing.JLabel headerUploads3;
837 private javax.swing.JButton jButton1;
838 private javax.swing.JPanel jPanel1;
839 private javax.swing.JPanel jPanel2;
840 private javax.swing.JPanel jPanel3;
841 private javax.swing.JScrollPane jScrollPane1;
842 private javax.swing.JScrollPane jScrollPane2;
843 private javax.swing.JScrollPane jScrollPane3;
844 private javax.swing.JScrollPane jScrollPane4;
845 private javax.swing.JSeparator jSeparator1;
846 private javax.swing.JSeparator jSeparator2;
847 private javax.swing.JSeparator jSeparator5;
848 private javax.swing.JSeparator jSeparator6;
849 private javax.swing.JSeparator jSeparator7;
850 private javax.swing.JSeparator jSeparator8;
851 private javax.swing.JTextField jTextField1;
852 private javax.swing.JTextPane jTextPane1;
853 private javax.swing.JTextPane jTextPane3;
854 private javax.swing.JTextPane jTextPane4;
855 private javax.swing.JLabel labelVehicleIDNorthPanel;
856 private javax.swing.JLabel labelVehicleIDNorthPanel2;
857 private javax.swing.JLabel labelVehicleIDNorthPanel3;
858 private javax.swing.JPanel mainPanelRepairstation;
859 private javax.swing.JPanel mainPanelRepairstation2;
860 private javax.swing.JPanel mainPanelRepairstation3;
861 private javax.swing.JScrollPane repairStationScrollPane;
862 private javax.swing.JTree resultTreeDynoStation;
863 private javax.swing.JScrollPane suspenStationScrollPane;
864 private javax.swing.JTabbedPane tabPanesMain;
865 private javax.swing.JButton uploadDocumentButton;
866 private javax.swing.JButton uploadDocumentButton2;
867 private javax.swing.JButton uploadDocumentButton3;
868 private javax.swing.JButton uploadDocumentButtonDynoStation;
869 private javax.swing.JButton uploadImageButton;

```

```

870 private javax.swing.JButton uploadImageButton2;
871 private javax.swing.JButton uploadImageButton3;
872 private javax.swing.JButton uploadImageButtonDynoStation;
873 private javax.swing.JButton uploadSoundButton;
874 private javax.swing.JButton uploadSoundButton2;
875 private javax.swing.JButton uploadSoundButton3;
876 private javax.swing.JButton uploadSoundButtonDynoStation;
877 private javax.swing.JButton uploadVideoButton;
878 private javax.swing.JButton uploadVideoButton2;
879 private javax.swing.JButton uploadVideoButton3;
880 private javax.swing.JButton uploadVideoButtonDynoStation;
881 private javax.swing.JLabel vehicleIDLabelDynoStation;
882 private javax.swing.JTextField vehicleIDTextFieldNorthPanel;
883 private javax.swing.JTextField vehicleIDTextFieldNorthPanel2;
884 private javax.swing.JTextField vehicleIDTextFieldNorthPanel3;
885 private javax.swing.JButton workOnButtonDynoStation;
886 private javax.swing.JButton workOnButtonRepairstation;
887 private javax.swing.JButton workOnButtonRepairstation2;
888 private javax.swing.JButton workOnButtonRepairstation3;
889 private javax.swing.JPanel workstationsMainPanel;
890 // End of variables declaration
891
892 private final String TEST_V_ID = "V15";

```

View/resultViewUtility (tweakmc.view.resultviewUtility)

ResultTreeUtility

```

1 package tweakmc.view.resultviewUtility;
2
3 import java.util.Vector;
4 import javax.swing.tree.DefaultMutableTreeNode;
5 import tweakmc.model.Result.Result;
6
7 /**
8  * Produce Tree models , to present result for Vehicles in the GUI
9  * @author clausPallisgaardBeck
10 */
11 public class ResultTreeUtility
12 {
13     //Instance variables
14     private static DefaultMutableTreeNode root;
15     private static DefaultMutableTreeNode image;
16     private static DefaultMutableTreeNode video;
17     private static DefaultMutableTreeNode sound;
18     private static DefaultMutableTreeNode document;
19     private static DefaultMutableTreeNode spreadsheet;
20     private static DefaultMutableTreeNode whiteboard;
21
22     private static final String IMAGE = "Image";
23     private static final String VIDEO = "Video";
24     private static final String SOUND = "Sound";

```



```

25 private static final String DOCUMENT = "Document";
26 private static final String SPREADSHEET = "Spreadsheet";
27 private static final String WHITEBOARD = "Whiteboard";
28
29 private static String resultName;
30
31
32 ///////////////////////////////////////////////////
33
34 /**
35  * Return the actual model with all relevant informations
36  * @param resultName for the tree
37  * @return treemodel with info
38  */
39 public static DefaultMutableTreeNode getResultTreeModelWithoutWhiteboard(String resultName)
40 {
41     buildResultTreeModel(resultName);
42     return root;
43 } //end method getResultTreeModelWithoutWhiteboard(String resultName)
44
45 /**
46  * Return the actual model with all relevant informations
47  * @param resultName for the tree
48  * @return treemodel with info
49  */
50 public static DefaultMutableTreeNode getResultTreeModelWithWhiteboard(String resultName)
51 {
52     buildResultTreeModel(resultName);
53     addWhiteboardOnTreeModel();
54     return root;
55 } //end method getResultTreeModelWithWhiteboard(String resultName)
56
57 /**
58  * Build the tree with the nodes as predefined
59  * @param resultName for the tree
60  */
61 private static void buildResultTreeModel(String resultName)
62 {
63     root = new DefaultMutableTreeNode(resultName);
64     image = new DefaultMutableTreeNode(IMAGE);
65     video = new DefaultMutableTreeNode(VIDEO);
66     sound = new DefaultMutableTreeNode(SOUND);
67     document = new DefaultMutableTreeNode(DOCUMENT);
68     spreadsheet = new DefaultMutableTreeNode(SPREADSHEET);
69     root.add(image);
70     root.add(video);
71     root.add(sound);
72     root.add(document);
73     root.add(spreadsheet);
74 } //end method buildResultTreeModel()
75
76 private static void addWhiteboardOnTreeModel()
77 {
78     whiteboard = new DefaultMutableTreeNode("Whiteboard");
79     root.add(whiteboard);
80 } //end method addWhiteboardOnTreeModel()
81
82 /**

```

```

83  * Fill names on the results
84  * @param results for this vehicled
85  */
86  public static void setNode(Vector<Result> results)
87  {
88      if(results.size() == 0)
89      {
90          return;
91      } //end if
92
93      for (Result r : results)
94      {
95          String resultType = r.getResultType();
96
97          if(resultType.equalsIgnoreCase(IMAGE))
98          {
99              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
100              image.add(node);
101          } //end if
102
103          if(resultType.equalsIgnoreCase(VIDEO))
104          {
105              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
106              video.add(node);
107          } //end if
108
109          if(resultType.equalsIgnoreCase(SOUND))
110          {
111              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
112              sound.add(node);
113          } //end if
114
115          if(resultType.equalsIgnoreCase(DOCUMENT))
116          {
117              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
118              document.add(node);
119          } //end if
120
121          if(resultType.equalsIgnoreCase(SPREADSHEET))
122          {
123              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
124              spreadsheet.add(node);
125          } //end if
126
127          if(resultType.equalsIgnoreCase(WHITEBOARD))
128          {
129              DefaultMutableTreeNode node = new DefaultMutableTreeNode(r);
130              whiteboard.add(node);
131          } //end if
132      } //end for each loop
133  } //end method setNode(Vector<Result> setNoderesults)
134 } //end class ResultTreeUtility

```


Model/ spreadsheet (tweakmc.Model.spreadsheet)

Spreadsheet

```
1 package tweakmc.model.spreadsheet;
2
3 import javax.swing.JTable;
4
5 /**
6  *
7  * @author clausPallisgaardBeck
8  */
9 public class Spreadsheet
10 {
11
12     //instance variables
13     private JTable jTable;
14     private JTable saveTable;
15
16     /**
17      * Constructor for spreadsheet, used to create a new instance to spreadsheet,
18      * for creation of empty or populated spreadsheet
19      */
20     public Spreadsheet()
21     {
22
23     } //end Spreadsheet constructor
24
25     /**
26      * Creates the tabel for the resultset after the users specifies
27      * rows and columns
28      * @param rows
29      * @param columns
30      * @param columnNames
31      * @return jTable the tabel for containing the
32      */
33     public JTable buildJTable(int rows, int columns, String[] columnNames)
34     {
35         jTable = new JTable(rows, columns);
36         if(columnNames.length > 0)
37         {
38             for (int i = 0; i < columnNames.length; i++)
39             {
40                 jTable.getColumnModel().getColumn(i).setHeaderValue(columnNames[i]);
41             } //end for
42         } //end if
43         return jTable;
44     } //end buildJTable method
45
46     /**
47      * Buil a new populated spreadsheet for saving as a object thru Objecthandler
48      * @param tableToSave actual table from GUI with data
49      * @return new JTable with data, ready for saving
50      */
51     public JTable buildPopulatedJTable(JTable tableToSave)
52     {
```

```

53
54     Object[][] data = new Object[tableToSave.getRowCount()][tableToSave.getColumnCount()];
55     String[] columnNames = new String[tableToSave.getColumnCount()];
56
57     //Save data from table to 2 dimensionel array
58     for (int i = 0; i < tableToSave.getRowCount(); i++)
59     {
60         for (int j = 0; j < tableToSave.getColumnCount(); j++)
61         {
62             data[i][j] = tableToSave.getModel().getValueAt(i, j);
63         } //end for
64     } //end for
65
66     //Save columnnames to array
67     for (int i = 0; i < tableToSave.getColumnCount(); i++)
68     {
69         columnNames[i] = tableToSave洗getColumnName(i);
70     } //end for
71
72     //Create a new table with the data and names for saving as bytes
73     saveTable = new JTable(data, columnNames);
74
75     System.gc();
76
77     return saveTable;
78 } //end buildPopulatedJTable method
79 } //end class Spreadsheet
80

```

SpreadsheetCatalog

```

1 package tweakmc.model.spreadsheet;
2
3 import javax.swing.JTable;
4 import tweakmc.dataaccess.ObjectHandler;
5
6 /**
7  *
8  * @author Tvup
9  */
10 public class SpreadsheetCatalog {
11     private static SpreadsheetCatalog instance;
12
13     //instance variables
14
15
16     public SpreadsheetCatalog()
17     {
18
19     } //end constructor
20
21     /**
22      * Singleton constructor
23      * @return instance
24      */
25     public static SpreadsheetCatalog getInstance()
26     {

```

```

27     if(instance==null)
28         return new SpreadsheetCatalog();
29     else
30         return instance;
31 }//end method getInstance
32
33 /**
34  * This method creates a spreadsheet
35  * @param rows
36  * @param columns
37  * @param coulumnNames
38  * @return jTable
39  */
40 public jTable createSpreadsheet(int rows, int columns, String[] coulumnNames)
41 {
42     return new Spreadsheet().buildJTable(rows, columns, coulumnNames);
43 }//end method createSpreadsheet
44
45 /**
46  * This metgod saves the table with the data
47  * @param tableToSave
48  * @param path
49  * @return true if saved else false
50  */
51 public boolean savePopulatedTable(JTable tableToSave, String path)
52 {
53     jTable actTable = new Spreadsheet().buildPopulatedJTable(tableToSave);
54
55     if(actTable != null)
56     {
57         return ObjectHandler.objectWriter(actTable, path);
58     }//end if
59
60     else
61     {
62         return false;
63     }//end else
64 }//end method savePopulatedTable
65
66 /**
67  * Find and present selected jTable. It is returned ad Object and must be casted
68  * @param path there table is saved
69  * @return jTable to manage
70  */
71 public jTable findSelectedSpreadsheet(String path)
72 {
73     return (JTable) ObjectHandler.objectReader(path);
74 }//end method findSelectedSpreadsheet(String path)
75
76 }//end class

```