5th Semester Report
Roskilde Business College
School of Computer Science
6th of August 2002 – 11th of November 2002

CAR TRACKING SYSTEM

Special Appendix

by
The Jacks
Dario Pacino
Hjörtur Sheving

Car Tracking System

Special Appendix

"Programavimas šiandien tai lenktynės tarp programų sistemų inžinierių ir gamtos:
vieni stengiasi kurti programinę įranga visiškiems neišmanėliams,
o gandras vis atneša naujų idiotų.

Kol kas gamta pirmauja.."

Rich Cook

A 5th Semester Report

By:
The Jacks,
Dario Pacino
Hjörtur Scheving

Students at:
Roskilde Business College,
School of Computer Science
Bakkesvinget 67
4000 Roskilde
Denmark

Project Period:
6th of August – 11th of November 2002

Supervised by:
Michael Claudius

Synopsis:
The knowledge and experience we have acquired through the last four semesters at RBC are to be brought together here in this report.

The report is build upon the development of a demo version of the Car Tracking System, which was developed by the Jacks for Sidabrinis Tinklas.

It is our hope that this report will proof to be of great help for future developers of the system.

Table of Contents

1 Class file Documentation

1.1 Class Action

java.lang.Object

|

+--**Action**

public abstract class **Action**
extends java.lang.Object

The Action class sets the constants for the communication between the Client and the Server

Field Summary	
static int	**RECEIVE_MESSAGE** The contant for RECEIVE_MESSAGE
static int	**RECEIVE_POSITION** The contant for RECEIVE_POSITION
static int	**SEND_DBSTRUCTURE** The contant for SEND_DBSTRUCTURE
static int	**SEND_INFO** The contant for SEND_INFO
static int	**SEND_MESSAGE** The contant for SEND_MESSAGE
static int	**SEND_POSITION** The contant for SEND_POSITION
static int	**SEND_QUERY** The constant for SEND_QUERY
static int	**SEND_RESULT** The constant for SEND_RESULT
static int	**SEND_USERLOGIN** The contant for SEND_USERLOGIN
static int	**SEND_USERLOGOUT** The contant for SEND_USERLOGOUT
static int	**SHUTDOWN** The contant for SHUTDOWN

Constructor Summary	
Action()	

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SEND_QUERY

public static final int **SEND_QUERY**

The constant for SEND_QUERY

See Also:

Constant Field Values

SEND_RESULT

public static final int **SEND_RESULT**

The constant for SEND_RESULT

See Also:

Constant Field Values

SHUTDOWN

public static final int **SHUTDOWN**

The contant for SHUTDOWN

See Also:

Constant Field Values

SEND_DBSTRUCTURE

public static final int **SEND_DBSTRUCTURE**

The contant for SEND_DBSTRUCTURE

See Also:

Constant Field Values

SEND_USERLOGIN

public static final int **SEND_USERLOGIN**

The contant for SEND_USERLOGIN

See Also:

Constant Field Values

SEND_INFO

public static final int **SEND_INFO**

The contant for SEND_INFO

See Also:

Constant Field Values

SEND_USERLOGOUT

public static final int **SEND_USERLOGOUT**

The contant for SEND_USERLOGOUT

See Also:

Constant Field Values

SEND_MESSAGE

public static final int **SEND_MESSAGE**

The contant for SEND_MESSAGE

See Also:

Constant Field Values

RECEIVE_MESSAGE

public static final int **RECEIVE_MESSAGE**

The contant for RECEIVE_MESSAGE

See Also:

Constant Field Values

SEND_POSITION

public static final int **SEND_POSITION**

The contant for SEND_POSITION

See Also:

Constant Field Values

RECEIVE_POSITION

public static final int **RECEIVE_POSITION**

The contant for RECEIVE_POSITION

See Also:

Constant Field Values

Constructor Detail

Action

public **Action**()

1.2 Class CentralClient

java.lang.Object
|
+--Client
|
+--**CentralClient**

---

public class **CentralClient**
extends Client

The CentralClient class is the the main class of the Central Client Component of the Car Tracking System. It holds informations about the Central Client and is the actual Client application for the Central Operator

---

Field Summary

Fields inherited from class Client
gisCom

Constructor Summary

CentralClient(java.lang.String Name, java.lang.String IP) Creates an instance of the CentralClient with a specific Name and IP address

Method Summary

void	**getPosition**(GISObject GISO) Updates the Position of a Mobile Client in the GIS
Users	**getUsers**() Returns the users on line
java.util.Vector	**getUsersVector**() Return the Vector used by the Users class to hold instaces of the User class
static void	**main**(java.lang.String[] args) The main function used to run the application

void	**sendMessage**(java.lang.String text, int receiverIndex) Sends a message to a Mobile Client through the Server.
void	**sendResult**(Result rs, int ClientID) Sends a result to a specific MobileClient through the Server
void	**setDBStructure**(Database db) Sets the Database object used by the Central Client
void	**showMessage**(Message msg) Shows a message in the main GUI.
void	**showResult**(Result res) Shows the results in a frame
void	**userLoggedIn**(User user) Notifies the CentralClient that a user has logged in the system
void	**userLoggedOut**(User user) Notifies the CentralClient that a user has logged out of the system

Methods inherited from class Client
addResult, closeClient, getName, getResultLog, run, sendQuery, setClientType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CentralClient

public **CentralClient**(java.lang.String Name,
java.lang.String IP)

Creates an instance of the CentralClient with a specific Name and IP address

Parameters:

Name - The name of the CentralClient

IP - The IP address where the Server is running (127.0.0.1 can be used for local connections)

Method Detail

getUsersVector

public java.util.Vector **getUsersVector**()

Return the Vector used by the Users class to hold instaces of the User class
Returns:
The Users's vector

---

getUsers

public Users **getUsers**()

Returns the users on line
Returns:
the online users

---

showResult

public void **showResult**(Result res)

Shows the results in a frame
Overrides:
showResult in class Client
Parameters:
res - The Result object to be showed

---

sendResult

public void **sendResult**(Result rs,
int ClientID)

Sends a result to a specific MobileClient through the Server
Parameters:
rs - The result to be sent
ClientID - The ID of the receiving Mobile Client

---

setDBStructure

public void **setDBStructure**(Database db)

Sets the Database object used by the Central Client
Overrides:
setDBStructure in class Client
Parameters:
db - The database object

---

userLoggedIn

public void **userLoggedIn**(User user)

Notifies the CentralClient that a user has logged in the system
Parameters:

user - The logging user

---

userLoggedOut

public void **userLoggedOut**(User user)

Notifies the CentralClient that a user has logged out of the system

Parameters:

user - The logging-out user

---

getPosition

public void **getPosition**(GISObject GISO)

Updates the Position of a Mobile Client in the GIS

Parameters:

GISO - The GISObject that holds the position

---

sendMessage

public void **sendMessage**(java.lang.String text,
int receiverIndex)

Sends a message to a Mobile Client through the Server. If a message is to be sent as a broad cast the value of receiverIndex must be set to SysConst.BROADCAST.

Parameters:

text - The message in the form of text

receiverIndex - The ID of the MobileClient receiving the message

---

main

public static void **main**(java.lang.String[] args)

The main function used to run the application

---

showMessage

public void **showMessage**(Message msg)

Shows a message in the main GUI. It basically calls the addMessage(Message msg) of the CMainGUI class.

Overrides:

showMessage in class Client

Parameters:

msg - The Message object to show

1.3 Class Client

java.lang.Object

|

+--Client

Direct Known Subclasses:

CentralClient, MobileClient

---

public class **Client**
extends java.lang.Object

The Client is the superclass used to implement all the common features between the CentralClient and the MobileClient

---

Field Summary

protected GISCommunication	**gisCom**

Constructor Summary

Client(java.lang.String Name, java.lang.String IP) Creates a Client object and attempts the connection to the Server

Method Summary

void	**addResult**(Result res) Adds a Ruesult to the Client's Result Cache
void	**closeClient**() Closes the Client and the connection with the Server
java.lang.String	**getName**() Returns the Name of the Client
ResultCache	**getResultLog**() Return the Result cache used by the Client
void	**run**() Runs the Client
void	**sendQuery**(Query q) Sends a Query to the Server

protected void	**setClientType**(int type) Sets the type of Client : SysConst.MOBILE_CLIENT SysConst.CENTRAL_CLIENT
void	**setDBStructure**(Database db) Sets the Database of the Client
void	**showMessage**(Message msg)
void	**showResult**(Result res)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

gisCom

protected GISCommunication **gisCom**

Constructor Detail

Client

public **Client**(java.lang.String Name,
java.lang.String IP)

Creates a Client object and attempts the connection to the Server

Parameters:

Name - The name of the Client

IP - The Ip address where the Server is located

Method Detail

setClientType

protected void **setClientType**(int type)

Sets the type of Client : SysConst.MOBILE_CLIENT SysConst.CENTRAL_CLIENT

Parameters:

type - The Client Type

---

getResultLog

public ResultCache **getResultLog**()

Return the Result cache used by the Client

Returns:

The result cache

---

getName

public java.lang.String **getName**()

Returns the Name of the Client

Returns:

The CLient name

---

run

public void **run**()

Runs the Client

---

showResult

public void **showResult**(Result res)

---

showMessage

public void **showMessage**(Message msg)

---

addResult

public void **addResult**(Result res)

Adds a Ruesult to the Client's Result Cache

Parameters:

res - The result to be added

---

closeClient

public void **closeClient**()

Closes the Client and the connection with the Server

---

sendQuery

public void **sendQuery**(Query q)

Sends a Query to the Server

---

setDBStructure

public void **setDBStructure**(Database db)

Sets the Database of the Client

Parameters:

db - The Database object

1.4 Interface ClientServerCommunication

All Known Implementing Classes:

EBCSCommunication

---

public interface **ClientServerCommunication**

The ClientServerCommunication interface describes the communication from the Client to the Server

---

Method Summary	
java.lang.Object	**readAction**(java.lang.Object input_connection) Receives the Action that is received by the Server
java.lang.Object	**receive_message**(java.lang.Object input_connection) Receives a message from the Server
java.lang.Object	**receive_Position**(java.lang.Object input_connection) Receives a GIS Position from the Server
java.lang.Object	**receiveDBStructure**(java.lang.Object input_connection) Receives a Database Structure
java.lang.Object	**receiveResult**(java.lang.Object input_connection) Returns a recived Result
java.lang.Object	**receiveUserInfo**(java.lang.Object input_connection) Receives User's Info
void	**send_message**(java.lang.Object message, java.lang.Object output_connection) Sends a Message to the Server
void	**send_Position**(java.lang.Object Position, java.lang.Object output_connection) Sends a GIS Position to the Server
void	**sendInfo**(java.lang.Object info, java.lang.Object output_connection) Sends the User Info to the Server
void	**sendQuery**(java.lang.Object query, java.lang.Object output_connection) Sends a Query
void	**sendResult**(java.lang.Object result, java.lang.Object output_connection,

	java.lang.Object clientID) Sends a Result
void	**terminateConnection**(java.lang.Object output_connection) Sends a request of temination of the connection to the server

Method Detail

sendQuery

public void **sendQuery**(java.lang.Object query,
java.lang.Object output_connection)
throws java.lang.Exception

Sends a Query

Parameters:

query - The Query object

output_connection - The object describing the output connection to the Server

java.lang.Exception

---

sendResult

public void **sendResult**(java.lang.Object result,
java.lang.Object output_connection,
java.lang.Object clientID)
throws java.lang.Exception

Sends a Result

Parameters:

result - The Result object

output_connection - The object describing the output connection to the Server

clientID - The ID of the receiving Client

java.lang.Exception

---

receiveResult

public java.lang.Object **receiveResult**(java.lang.Object input_connection)
throws java.lang.Exception

Returns a recived Result

Parameters:

input_connection - The object describing the input connection from the Server

Returns:

The received Result

java.lang.Exception

---

receiveDBStructure

public java.lang.Object **receiveDBStructure**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a Database Structure

Parameters:

input_connection - The object describing the input connection from the Server

Returns:

The Database Structure

java.lang.Exception

---

receiveUserInfo

public java.lang.Object **receiveUserInfo**(java.lang.Object input_connection)
throws java.lang.Exception

Receives User's Info

Parameters:

input_connection - The object describing the input connection from the Server

Returns:

The User's info

java.lang.Exception

---

readAction

public java.lang.Object **readAction**(java.lang.Object input_connection)
throws java.lang.Exception

Receives the Action that is received by the Server

Parameters:

input_connection - The object describing the input connection from the Server

Returns:

The Action

java.lang.Exception

---

terminateConnection

public void **terminateConnection**(java.lang.Object output_connection)
throws java.lang.Exception

Sends a request of temination of the connection to the server

Parameters:

output_connection - The object describing the output connection to the Server

java.lang.Exception

---

sendInfo

public void **sendInfo**(java.lang.Object info,
java.lang.Object output_connection)
throws java.lang.Exception

Sends the User Info to the Server

Parameters:

info - The User's Info

output_connection - The object describing the output connection to the Server

java.lang.Exception

---

receive_message

public java.lang.Object **receive_message**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a message from the Server

Parameters:

input_connection - The object describing the input connection from the Server

Returns:

The received Message

java.lang.Exception

---

send_message

public void **send_message**(java.lang.Object message,
java.lang.Object output_connection)
throws java.lang.Exception

Sends a Message to the Server

Parameters:

output_connection - The object describing the output connection to the Server

java.lang.Exception

---

receive_Position

public java.lang.Object **receive_Position**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a GIS Position from the Server

Parameters:

input_connection - The object describing the input connection from the Server

Returns:
The GIS Position
java.lang.Exception

---

send_Position

public void **send_Position**(java.lang.Object Position,
java.lang.Object output_connection)
throws java.lang.Exception

Sends a GIS Position to the Server
Parameters:
Position - The GIS Position
output_connection - The object describing the output connection to the Server
java.lang.Exception

1.5 Class CMainGUI

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--java.awt.Window
|
+--java.awt.Frame
|
+--javax.swing.JFrame
|
+--**CMainGUI**

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **CMainGUI**
extends javax.swing.JFrame

The CMainGUI represents the main window of the Central Client

See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

protected CentralClient	**localClient**
protected OnLineUsersGUI	**usersGUI**

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR,

SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

CMainGUI(Database db, OnLineUsersGUI usersGUI, CentralClient localClient) Creates an instance of CMainGUI

Method Summary

void	**addMessage**(Message msg) Appends a Message in the Message board
void	**close**() Closes the window and the application
void	**showResult**(Result res) Shows the Results in a separate window

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners,

getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
getFont, postEvent

Field Detail

usersGUI

protected OnLineUsersGUI **usersGUI**

---

localClient

protected CentralClient **localClient**

Constructor Detail

CMainGUI

public **CMainGUI**(Database db,
OnLineUsersGUI usersGUI,
CentralClient localClient)

Creates an instance of CMainGUI

Parameters:

db - The Database structure

usersGUI - The online users GUI

localClient - The CentralClient

Method Detail

close

public void **close**()

Closes the window and the application

---

showResult

public void **showResult**(Result res)

Shows the Results in a separate window

Parameters:

res - The result to show

---

addMessage

public void **addMessage**(Message msg)

Appends a Message in the Message board

Parameters:

msg - The Message

1.6 Class CMessageGUI

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|

```
+--javax.swing.JPanel
    |
    +--CMessageGUI
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.util.Observer, java.io.Serializable

---

public class **CMessageGUI**
extends javax.swing.JPanel
implements java.util.Observer, java.awt.event.ActionListener

The CMessageGUI class represents the GUI for the Messaging System of the CentralClient

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

protected javax.swing.DefaultComboBoxModel	**listModel**
protected CentralClient	**localClient**
protected javax.swing.JScrollPane	**mainPane**

protected javax.swing.JTextArea	**mainText**
protected javax.swing.JPanel	**msgList**
protected javax.swing.JScrollPane	**msgPane**
protected javax.swing.JPanel	**msgSend**
protected javax.swing.JTextArea	**msgText**
protected javax.swing.JButton	**sendBt**
protected javax.swing.JComboBox	**userCb**

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

CMessageGUI(CentralClient localClient) Creates the GUI

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener interface
void	**addLocalMsg**(java.lang.String txt) Adds a local message to the message board

void	**addMessage**(Message msg) Adds a message to the message board
void	**update**(java.util.Observable o, java.lang.Object arg) Implementation of the Observer interface

Methods inherited from class javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

mainText

protected javax.swing.JTextArea **mainText**

---

msgText

protected javax.swing.JTextArea **msgText**

---

sendBt

protected javax.swing.JButton **sendBt**

---

msgSend

protected javax.swing.JPanel **msgSend**

---

msgList

protected javax.swing.JPanel **msgList**

---

mainPane

protected javax.swing.JScrollPane **mainPane**

---

msgPane

protected javax.swing.JScrollPane **msgPane**

---

userCb

protected javax.swing.JComboBox **userCb**

---

listModel

protected javax.swing.DefaultComboBoxModel **listModel**

---

localClient

protected CentralClient **localClient**

Constructor Detail

CMessageGUI

public **CMessageGUI**(CentralClient localClient)

Creates the GUI

Parameters:

localClient - The CentralClient

Method Detail

update

public void **update**(java.util.Observable o,
java.lang.Object arg)

Implementation of the Observer interface

Specified by:

update in interface java.util.Observer

---

addMessage

public void **addMessage**(Message msg)

Adds a message to the message board

Parameters:

msg - The message to add

---

addLocalMsg

public void **addLocalMsg**(java.lang.String txt)

Adds a local message to the message board

Parameters:

txt - The message in text form

---

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener interface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

1.7 Class Config

java.lang.Object

|

+--Config

All Implemented Interfaces:

java.io.Serializable

---

public class **Config**
extends java.lang.Object
implements java.io.Serializable

The Config class holds all the configuration data of the server

See Also:

Serialized Form

Constructor Summary

Config()	

Method Summary

java.lang.String	**getDBFilePath**() Returns the filename and path where the file holding the database structure is placed
int	**getPort**() Returns the port number where the server is running
void	**setDBFilePath**(java.lang.String filepath) Sets the filename and path where the file holding the database structure is placed
void	**setPort**(int port) Sets the port number where the server is going to run

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Config

public **Config**()

Method Detail

getDBFilePath

public java.lang.String **getDBFilePath**()

Returns the filename and path where the file holding the database structure is placed

Returns:

The filaname and path

---

setDBFilePath

public void **setDBFilePath**(java.lang.String filepath)

Sets the filename and path where the file holding the database structure is placed

---

getPort

public int **getPort**()

Returns the port number where the server is running

Returns:

The port number

---

setPort

public void **setPort**(int port)

Sets the port number where the server is going to run

Contents

Unnamed Package.*

Action		
public static final int	RECEIVE_MESSAGE	9
public static final int	RECEIVE_POSITION	11
public static final int	SEND_DBSTRUCTURE	4
public static final int	SEND_INFO	6
public static final int	SEND_MESSAGE	8
public static final int	SEND_POSITION	10

public static final int	SEND_QUERY	1
public static final int	SEND_RESULT	2
public static final int	SEND_USERLOGIN	5
public static final int	SEND_USERLOGOUT	7
public static final int	SHUTDOWN	3

Field		
public static final java.lang.String	BOOLEAN	"BOOLEAN"
public static final java.lang.String	FLOAT	"FLOAT"
public static final java.lang.String	GRAPHIC	"GRAPHIC"
public static final java.lang.String	INT	"INT"
public static final java.lang.String	TEXT	"TEXT"

Result		
public static final int	OK	1
public static final int	TOO_BROAD	2

Server		
public static final int	NO_QUERY_SERVICES	1
public static final int	READY	0

SysConst		
public static final int	BROADCAST	-2
public static final int	CENTRAL_CLIENT	2
public static final int	END_ROW	4
public static final int	END_TRANSACTION	3
public static final int	IMAGE_ON_STREAM	5
public static final int	MOBILE_CLIENT	1
public static final int	NO_RECEIVER	-1

1.8 Class Coordinate

java.lang.Object

|

+--**Coordinate**

All Implemented Interfaces:
java.io.Serializable

---

public class **Coordinate**
extends java.lang.Object
implements java.io.Serializable

The Coordinate represents the geographical position of an object in terms of coordinates

See Also:
Serialized Form

Field Summary

protected float	**lat**
protected float	**lon**

Constructor Summary

Coordinate(float lon, float lat) Creates an instance of Coordinate

Method Summary

float	**getLat**() Returns the latitudinal coordinate
float	**getLon**() Returns the longitudinal coordinate

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

lon

protected float **lon**

---

lat

protected float **lat**

Constructor Detail

Coordinate

public **Coordinate**(float lon,
float lat)

Creates an instance of Coordinate

Parameters:

lon - Longitudinal coordinate

lat - Latiditinal coordinate

Method Detail

getLon

public float **getLon**()

Returns the longitudinal coordinate

Returns:

The longitudinal value

---

getLat

public float **getLat**()

Returns the latitudinal coordinate

Returns:

The latitudinal value

1.9 Class CResultGUI

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--java.awt.Window
|
+--java.awt.Frame
|
+--javax.swing.JFrame
|

+--CResultGUI

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.util.Observer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **CResultGUI**
extends javax.swing.JFrame
implements java.util.Observer, java.awt.event.ActionListener

The CResultGUI is the window displayin the Result objects

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JFrame

accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

CResultGUI(Result rs, CentralClient localClient) Creates and display the CResultGUI

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener inteface
void	**createGUI**(int row) Starts the process of automatic creation of the GUI based on one of the Results
void	**update**(java.util.Observable o, java.lang.Object arg) Implementation of the Observer interface

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation,

getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener,

addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer

getFont, postEvent

Constructor Detail

CResultGUI

public **CResultGUI**(Result rs,
CentralClient localClient)

Creates and display the CResultGUI

Parameters:

rs - The Result to show

localClient - The Central Client

Method Detail

update

public void **update**(java.util.Observable o,
java.lang.Object arg)

Implementation of the Observer interface

Specified by:

update in interface java.util.Observer

---

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener inteface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

---

createGUI

public void **createGUI**(int row)

Starts the process of automatic creation of the GUI based on one of the Results

Parameters:

row - The row index of the result to be showed from the Result object

---

1.10 Class CSearchPanel

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|

+--javax.swing.JPanel

|

+--**CSearchPanel**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **CSearchPanel**
extends javax.swing.JPanel
implements java.awt.event.ActionListener

The CSearchPanel class represents the Querying part of the Main window of the Central

Client application

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION,

WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

CSearchPanel(Database db, CentralClient localClient, CMainGUI main) Creates and istance of CSearchPanel

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener interface

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent,

isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName,

getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CSearchPanel

public **CSearchPanel**(Database db,
CentralClient localClient,
CMainGUI main)

Creates and istance of CSearchPanel

Parameters:

db - The Database structure

localClient - The CentralClient

main - The main window

Method Detail

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener interface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

1.11 Class CTS_Server

java.lang.Object
|
+--**CTS_Server**

---

public class **CTS_Server**
extends java.lang.Object

The CTS_Server is the class representing the server application

---

Constructor Summary

CTS_Server()	

Method Summary

static void	**main**(java.lang.String[] args) The main method

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CTS_Server

public **CTS_Server**()

Method Detail

main

public static void **main**(java.lang.String[] args)
The main method

1.12 Class Database

java.lang.Object
|
+--**Database**

All Implemented Interfaces:
java.io.Serializable
Direct Known Subclasses:
SDatabase

---

public class **Database**
extends java.lang.Object
implements java.io.Serializable

The Database class represents the database connected to the system

See Also:
Serialized Form

Field Summary

protected Table	**DBTable** The Object describing the table used in the database.
protected java.lang.String	**name** The Name of the Database - used as Data Source Name for the ODBC driver

Constructor Summary

protected	**Database**()
	Database(java.lang.String name) Creates a Database Object with the specific path and no username or password

Method Summary

boolean	**equals**(java.lang.Object O) Compares two Database Objects
java.lang.String	**getName**() Returns the Database name
Table	**getTable**() This function returns the Table used in the database
void	**setName**(java.lang.String name) Set the Database name
void	**setTable**(Table DBTable) Sets the Database table

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

name

protected java.lang.String **name**

The Name of the Database - used as Data Source Name for the ODBC driver

---

DBTable

protected Table **DBTable**

The Object describing the table used in the database. For this demo version only one table is accepted from the system

Constructor Detail

Database

protected **Database**()

Database

public **Database**(java.lang.String name)

Creates a Database Object with the specific path and no username or password

Method Detail

getTable

public Table **getTable**()

This function returns the Table used in the database

Returns:

The Table

---

equals

public boolean **equals**(java.lang.Object O)

Compares two Database Objects

Overrides:

equals in class java.lang.Object

Returns:

true false

setTable

public void **setTable**(Table DBTable)

Sets the Database table

Parameters:

DBTable - The table object

setName

public void **setName**(java.lang.String name)

Set the Database name

Parameters:

name - The Database name

getName

public java.lang.String **getName**()

Returns the Database name

Returns:

The Database name

1.13 Interface DBServerCommunication

All Known Implementing Classes:

EBConnection

public interface **DBServerCommunication**

The DBServerCommunication is an interface that defines the operations that can be performed in relation with the DB Server

Method Summary	
void	**connectToDB**() Connects to the Database using the Database path, username and password stored in the internal variables
void	**disconnectFromDB**() Disconnects from the database.
java.lang.Object	**getDBPath**() Returns the Database path
java.lang.Object	**getPassword**()

	Returns the Database password
java.lang.Object	**getUsername**() Returns the Database username
java.lang.Object	**sendQuery**(java.lang.Object query) Sends a query to the database.
void	**setDBPath**(java.lang.Object path) Stores the Database path in an internal variable
void	**setPassword**(java.lang.Object password) Stores the Database password in an internal variable
void	**setUsername**(java.lang.Object username) Stores the Database username in an internal variable
java.lang.Object	**translateQuery**(java.lang.Object query) Translates the Query object in an object usable for the DBMS

Method Detail

setDBPath

public void **setDBPath**(java.lang.Object path)

Stores the Database path in an internal variable

---

setUsername

public void **setUsername**(java.lang.Object username)

Stores the Database username in an internal variable

---

setPassword

public void **setPassword**(java.lang.Object password)

Stores the Database password in an internal variable

---

connectToDB

public void **connectToDB**()

throws java.lang.Exception

Connects to the Database using the Database path, username and password stored in the internal variables

Returns:

The Database Connection

java.lang.Exception

See Also:
setDBPath(Object path), setUsername(Object username), setPassword(Object password)

---

disconnectFromDB

public void **disconnectFromDB**()
throws java.lang.Exception
Disconnects from the database.
java.lang.Exception

---

sendQuery

public java.lang.Object **sendQuery**(java.lang.Object query)
throws java.lang.Exception
Sends a query to the database.
Parameters:
query - The Query Object
Returns:
The result Object
java.lang.Exception

---

getDBPath

public java.lang.Object **getDBPath**()
Returns the Database path

---

getUsername

public java.lang.Object **getUsername**()
Returns the Database username

---

getPassword

public java.lang.Object **getPassword**()
Returns the Database password

---

translateQuery

public java.lang.Object **translateQuery**(java.lang.Object query)
Translates the Query object in an object usable for the DBMS
Returns:
The translated query

1.14 Class EBConnection

java.lang.Object
|
+--EBConnection

All Implemented Interfaces:
DBServerCommunication, java.io.Serializable

public class **EBConnection**
extends java.lang.Object
implements DBServerCommunication, java.io.Serializable

This class implements the DBServerComminication interface and is specially designed for ODBC drivers compatible Databases. It Implies direct connection to the database through LAN.

See Also:
Serialized Form

Constructor Summary

EBConnection(java.lang.String driver) Creates a database connection object with a specific jdbc driver

Method Summary

void	**connectToDB**() Connects to the Database using the Database path, username and password stored in the internal variables
void	**disconnectFromDB**() Disconnects from the database.
java.lang.Object	**getDBPath**() Returns the Database path
java.lang.Object	**getPassword**() Returns the Database Password
java.lang.Object	**getUsername**() Returnd the Database Username
java.lang.Object	**sendQuery**(java.lang.Object query) Sends a query to the database.

void	**setDBPath**(java.lang.Object path) Stores the Database path in an internal variable
void	**setDriver**(java.lang.String driver) Sets the driver class
void	**setPassword**(java.lang.Object password) Stores the Database password in an internal variable
void	**setUsername**(java.lang.Object username) Stores the Database username in an internal variable
java.lang.Object	**translateQuery**(java.lang.Object query) Translates the Query object in an object usable for the DBMS

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EBConnection

public **EBConnection**(java.lang.String driver)

Creates a database connection object with a specific jdbc driver

Parameters:

driver - The jdbc driver

Method Detail

setDBPath

public void **setDBPath**(java.lang.Object path)

Stores the Database path in an internal variable

Specified by:

setDBPath in interface DBServerCommunication

---

getDBPath

public java.lang.Object **getDBPath**()

Returns the Database path

Specified by:

getDBPath in interface DBServerCommunication

Returns:

The Database path

---

getUsername

public java.lang.Object **getUsername**()

Returnd the Database Username

Specified by:

getUsername in interface DBServerCommunication

Returns:

The Database Username

---

getPassword

public java.lang.Object **getPassword**()

Returns the Database Password

Specified by:

getPassword in interface DBServerCommunication

Returns:

The Database password

---

setDriver

public void **setDriver**(java.lang.String driver)

Sets the driver class

---

translateQuery

public java.lang.Object **translateQuery**(java.lang.Object query)

Translates the Query object in an object usable for the DBMS

Specified by:

translateQuery in interface DBServerCommunication

Returns:

The SQL equivalent of the query, as a String

---

setUsername

public void **setUsername**(java.lang.Object username)

Stores the Database username in an internal variable

Specified by:

setUsername in interface DBServerCommunication

---

setPassword

public void **setPassword**(java.lang.Object password)

Stores the Database password in an internal variable

Specified by:

setPassword in interface DBServerCommunication

---

connectToDB

public void **connectToDB**()

throws java.lang.Exception

Connects to the Database using the Database path, username and password stored in the internal variables

Specified by:

connectToDB in interface DBServerCommunication

Returns:

The Database Connection

java.lang.Exception

See Also:

setDBPath(Object path), setUsername(Object username), setPassword(Object password)

---

disconnectFromDB

public void **disconnectFromDB**()

throws java.lang.Exception

Disconnects from the database.

Specified by:

disconnectFromDB in interface DBServerCommunication

java.lang.Exception

---

sendQuery

public java.lang.Object **sendQuery**(java.lang.Object query)

throws java.lang.Exception

Sends a query to the database.

Specified by:

sendQuery in interface DBServerCommunication

Parameters:

query - The Query Object.

Returns:

The result Object

java.lang.Exception

1.15 Class Field

java.lang.Object

|

+--**Field**

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

ResultField

---

public class **Field**
extends java.lang.Object
implements java.io.Serializable

The Field class holds the information of the fields of a Table.

See Also:

Serialized Form

Field Summary

static java.lang.String	**BOOLEAN** Constant indicating a BOOLEAN field type
static java.lang.String	**FLOAT** Constant indicating a FLOAT field type
static java.lang.String	**GRAPHIC** Constant indicating a GRAPHIC field type
static java.lang.String	**INT** Constant indicating a INT field type
static java.lang.String	**TEXT** Constant indicating a TEXT field type

Constructor Summary

Field(java.lang.String name, java.lang.String type, boolean searcheable, int size) Creates a Field object with a specific name, type and seracheable option

Method Summary

boolean	**equals**(java.lang.Object obj) Compares two Field Objects
java.lang.String	**getName**() Returns the field name
boolean	**getSearcheable**() Returns if the field can be used as a search value or not
int	**getSize**()

java.lang.String	**getType**() Returns the type of the field.

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

TEXT

public static final java.lang.String **TEXT**

Constant indicating a TEXT field type

See Also:

Constant Field Values

GRAPHIC

public static final java.lang.String **GRAPHIC**

Constant indicating a GRAPHIC field type

See Also:

Constant Field Values

INT

public static final java.lang.String **INT**

Constant indicating a INT field type

See Also:

Constant Field Values

FLOAT

public static final java.lang.String **FLOAT**

Constant indicating a FLOAT field type

See Also:

Constant Field Values

BOOLEAN

public static final java.lang.String **BOOLEAN**

Constant indicating a BOOLEAN field type

See Also:

Constant Field Values

Constructor Detail

Field

public **Field**(java.lang.String name,
java.lang.String type,
boolean searcheable,
int size)

Creates a Field object with a specific name, type and seracheable option

Method Detail

equals

public boolean **equals**(java.lang.Object obj)

Compares two Field Objects

Overrides:

equals in class java.lang.Object

Parameters:

obj - The Object to compare

Returns:

True False

---

getName

public java.lang.String **getName**()

Returns the field name

Returns:

the Field name

---

getType

public java.lang.String **getType**()

Returns the type of the field. A list of valid type is described in the class constants

Returns:

The Field type

---

getSearcheable

public boolean **getSearcheable**()

Returns if the field can be used as a search value or not

Returns:

Field searcheability

---

getSize

public int **getSize**()

1.16 Class GIS

java.lang.Object

|

+--**GIS**

All Implemented Interfaces:

GISCommunication

---

public class **GIS**
extends java.lang.Object
implements GISCommunication

The GIS class represents the GIS Software and describes the functionalities connected to it

---

Field Summary

protected java.sql.Connection	**dbConnection**
protected java.sql.Statement	**stmt**

Constructor Summary

GIS()	

Method Summary

void	**connect**() Start of the communication
void	**createObject**(java.lang.Object obj) Create and Object in the GIS
void	**deleteObject**(java.lang.Object obj) Delete and Object form the GIS

void	**disconnect**() End of the communication
void	**updateObject**(java.lang.Object obj) Updating the value of an Object in the GIS

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

dbConnection

protected java.sql.Connection **dbConnection**

---

stmt

protected java.sql.Statement **stmt**

Constructor Detail

GIS

public **GIS**()

Method Detail

connect

public void **connect**()

throws java.lang.Exception

Start of the communication

Specified by:

connect in interface GISCommunication

java.lang.Exception

---

disconnect

public void **disconnect**()

throws java.lang.Exception

End of the communication

Specified by:

disconnect in interface GISCommunication

java.lang.Exception

---

updateObject

public void **updateObject**(java.lang.Object obj)

throws java.lang.Exception

Updating the value of an Object in the GIS

Specified by:

updateObject in interface GISCommunication

Parameters:

obj - The Object in the GIS

java.lang.Exception

---

deleteObject

public void **deleteObject**(java.lang.Object obj)

throws java.lang.Exception

Delete and Object form the GIS

Specified by:

deleteObject in interface GISCommunication

Parameters:

obj - The Object in the GIS

java.lang.Exception

---

createObject

public void **createObject**(java.lang.Object obj)

throws java.lang.Exception

Create and Object in the GIS

Specified by:

createObject in interface GISCommunication

Parameters:

obj - The Object in the GIS

java.lang.Exception

1.17 Interface GISCommunication

All Known Implementing Classes:

GIS

---

public interface **GISCommunication**

The GISCommunication interface represents the GIS Software and describes the functionalities connected to it

---

Method Summary	
void	**connect**() Start of the communication
void	**createObject**(java.lang.Object obj) Create and Object in the GIS
void	**deleteObject**(java.lang.Object obj) Delete and Object form the GIS
void	**disconnect**() End of the communication
void	**updateObject**(java.lang.Object obj) Updating the value of an Object in the GIS

Method Detail

connect

public void **connect**()

throws java.lang.Exception

Start of the communication

java.lang.Exception

---

disconnect

public void **disconnect**()

throws java.lang.Exception

End of the communication

java.lang.Exception

---

updateObject

public void **updateObject**(java.lang.Object obj)

throws java.lang.Exception

Updating the value of an Object in the GIS

Parameters:

obj - The Object in the GIS

java.lang.Exception

---

deleteObject

public void **deleteObject**(java.lang.Object obj)
throws java.lang.Exception

Delete and Object form the GIS

Parameters:

obj - The Object in the GIS

java.lang.Exception

---

createObject

public void **createObject**(java.lang.Object obj)
throws java.lang.Exception

Create and Object in the GIS

Parameters:

obj - The Object in the GIS

java.lang.Exception

1.18 Class GISObject

java.lang.Object
|
+--**GISObject**

All Implemented Interfaces:

java.io.Serializable

---

public class **GISObject**
extends java.lang.Object
implements java.io.Serializable

The GISObject class represents and object in the GIS system

See Also:

Serialized Form

Field Summary	
Coordinate	**coordinate** The Object coordinate
java.lang.String	**name** The Object name

Constructor Summary

GISObject(java.lang.String name, Coordinate c) Creates an instanec of the GISObject

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

name

public java.lang.String **name**

The Object name

---

coordinate

public Coordinate **coordinate**

The Object coordinate

Constructor Detail

GISObject

public **GISObject**(java.lang.String name,
Coordinate c)

Creates an instanec of the GISObject

Parameters:

name - The object's name

c - The object's coordinate

1.19 Interface GPSCommunication

All Known Implementing Classes:

GPSDevice

---

public interface **GPSCommunication**

The GPSCommunication interface represents the communication with the GPS devide

---

Method Summary

Coordinate	**getCoordinate**() Returns the coordinate read in the GPS device

Method Detail

getCoordinate

public Coordinate **getCoordinate**()

Returns the coordinate read in the GPS device

Returns:

The coordinate

1.20 Class GPSDevice

java.lang.Object

|

+--java.lang.Thread

|

+--**GPSDevice**

All Implemented Interfaces:

GPSCommunication, java.lang.Runnable

---

public class **GPSDevice**
extends java.lang.Thread
implements GPSCommunication

The GPSDevice class represents the GPS device Note: Used as a simulation device in the demo version

---

Field Summary

protected boolean	**exit**
protected float	**lat**
protected MobileClient	**localClient**
protected float	**lon**

protected float	**refLat**
protected float	**refLon**
protected int	**state**

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

GPSDevice(float refLon, float refLat, MobileClient localClient) Creates an instance of the GPSDevice with starting coordinates

Method Summary

Coordinate	**getCoordinate**() Returns the coordinate read in the GPS device
void	**halt**() Stops the running thread
void	**run**() Starts the running thread

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

refLon

protected float **refLon**

---

lon

protected float **lon**

---

refLat

protected float **refLat**

---

lat

protected float **lat**

---

state

protected int **state**

---

localClient

protected MobileClient **localClient**

---

exit

protected boolean **exit**

Constructor Detail

GPSDevice

public **GPSDevice**(float refLon,
float refLat,
MobileClient localClient)

Creates an instance of the GPSDevice with starting coordinates

Parameters:

refLon - The longitudinal coordinate

refLat - The latitudinal coordinate

localClient - The Mobile Client

Method Detail

getCoordinate

public Coordinate **getCoordinate**()

Returns the coordinate read in the GPS device

Specified by:

getCoordinate in interface GPSCommunication

Returns:

The coordinate

---

halt

public void **halt**()

Stops the running thread

---

run

public void **run**()

Starts the running thread

Specified by:

run in interface java.lang.Runnable

Overrides:

run in class java.lang.Thread

1.21 Class Info

java.lang.Object

|

+--**Info**

---

public class **Info**
extends java.lang.Object

The Info class holds information of the client

Field Summary

java.lang.String	**clientName** The client name
int	**clientType** The typr of client

Constructor Summary

Info()	

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

clientType

public int **clientType**

The typr of client

See Also:

SysConst

---

clientName

public java.lang.String **clientName**

The client name

Constructor Detail

Info

public **Info**()

1.22 Class InvalidStructureException

java.lang.Object
 |
 +--java.lang.Throwable
 |
 +--java.lang.Exception
 |
 +--**InvalidStructureException**

All Implemented Interfaces:

java.io.Serializable

---

public class **InvalidStructureException**
extends java.lang.Exception

The InvalidStructureException is thrown when the file holding the database structure describes an invalid structure

See Also:

Serialized Form

Constructor Summary

InvalidStructureException(java.lang.String file)

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

InvalidStructureException

public **InvalidStructureException**(java.lang.String file)

1.23 Class MainGUI

java.lang.Object
  |
  +--java.awt.Component
      |
      +--java.awt.Container
          |
          +--java.awt.Window
              |
              +--java.awt.Frame
                  |
                  +--javax.swing.JFrame

|

+--**MainGUI**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **MainGUI**
extends javax.swing.JFrame

The MainGUI represents the main window of the Mobile Client

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

MainGUI(Database db, MobileClient localClient) Creates an instance of MainGUI	

Method Summary

void	**addMessage**(Message msg) Appends a Message in the Message board
void	**close**() Closes the window and the application
void	**showResult**(Result res) Shows the Results in a separate window

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane,

setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains,

createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer

getFont, postEvent

Constructor Detail

MainGUI

public **MainGUI**(Database db,
MobileClient localClient)

Creates an instance of MainGUI

Parameters:

db - The Database structure

localClient - The MobileClient

Method Detail

close

public void **close**()

Closes the window and the application

---

showResult

public void **showResult**(Result res)

Shows the Results in a separate window

Parameters:

res - The result to show

---

addMessage

public void **addMessage**(Message msg)

Appends a Message in the Message board

Parameters:

msg - The Message

1.24 Class Message

java.lang.Object

|

+--**Message**

All Implemented Interfaces:

java.io.Serializable

---

public class **Message**
extends java.lang.Object
implements java.io.Serializable

The Message class represents a message in the messaging system

See Also:

Serialized Form

Field Summary

protected int	**receiver**
protected int	**sender**
protected java.lang.String	**text**

Constructor Summary

Message(java.lang.String Text) Creates a Message with a specific text and with no receiver and sender
Message(java.lang.String text, int sender) Creates a Message with a specific text and sender , and with no receiver
Message(java.lang.String text, int sender, int receiver) Creates a Message with a specific text, sender and receiver

Method Summary

boolean	**equals**(java.lang.Object obj) Check whether two instances of Message are equal
int	**getReceiver**() Returns the receiver of the message
int	**getSender**() Returns the sender of the message
java.lang.String	**getText**() Returns the text of the message
void	**setReceiver**(int receiver) Sets the receiver of the message
void	**setSender**(int sender) Sets the sender of the message

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

text

protected java.lang.String **text**

---

sender

protected int **sender**

---

receiver

protected int **receiver**

Constructor Detail

Message

public **Message**(java.lang.String Text)

Creates a Message with a specific text and with no receiver and sender

Parameters:

Text - The text message

Message

public **Message**(java.lang.String text,
int sender)

Creates a Message with a specific text and sender , and with no receiver

Parameters:

sender - The sender Client ID

Message

public **Message**(java.lang.String text,
int sender,
int receiver)

Creates a Message with a specific text, sender and receiver

Parameters:

sender - The sender Client ID

receiver - The receiver Clint ID

Method Detail

setReceiver

public void **setReceiver**(int receiver)

Sets the receiver of the message

Parameters:

receiver - The receiver Client ID

---

setSender

public void **setSender**(int sender)

Sets the sender of the message

Parameters:

sender - The sender Client ID

---

getText

public java.lang.String **getText**()

Returns the text of the message

Returns:

The text

---

getSender

public int **getSender**()

Returns the sender of the message

Returns:

The sender Client ID

---

getReceiver

public int **getReceiver**()

Returns the receiver of the message

Returns:

The receiver Client ID

---

equals

public boolean **equals**(java.lang.Object obj)

Check whether two instances of Message are equal

Overrides:

equals in class java.lang.Object

Returns:

true/false

1.25 Class MessageGUI

java.lang.Object

```
|
+--java.awt.Component
      |
      +--java.awt.Container
            |
            +--javax.swing.JComponent
                  |
                  +--javax.swing.JPanel
                        |
                        +--MessageGUI
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **MessageGUI**
extends javax.swing.JPanel
implements java.awt.event.ActionListener

The MessageGUI class represents the GUI for the Messaging System of the Mobile Client

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary	
protected MobileClient	**localClient**
protected javax.swing.JScrollPane	**mainPane**
protected javax.swing.JTextArea	**mainText**
protected javax.swing.JPanel	**msgList**
protected javax.swing.JScrollPane	**msgPane**
protected javax.swing.JPanel	**msgSend**
protected javax.swing.JTextArea	**msgText**
protected javax.swing.JButton	**sendBt**

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary
MessageGUI(MobileClient localClient) Creates the GUI

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener interface
void	**addLocalMsg**(java.lang.String txt) Adds a local message to the message board
void	**addMessage**(Message msg) Adds a message to the message board

Methods inherited from class javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText,

setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size,

toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

mainText

protected javax.swing.JTextArea **mainText**

---

msgText

protected javax.swing.JTextArea **msgText**

---

sendBt

protected javax.swing.JButton **sendBt**

---

msgSend

protected javax.swing.JPanel **msgSend**

---

msgList

protected javax.swing.JPanel **msgList**

---

mainPane

protected javax.swing.JScrollPane **mainPane**

---

msgPane

protected javax.swing.JScrollPane **msgPane**

---

localClient

protected MobileClient **localClient**

Constructor Detail

MessageGUI

public **MessageGUI**(MobileClient localClient)

Creates the GUI

Parameters:

localClient - The MobileClient

Method Detail

addMessage

public void **addMessage**(Message msg)

Adds a message to the message board

Parameters:

msg - The message to add

---

addLocalMsg

public void **addLocalMsg**(java.lang.String txt)

Adds a local message to the message board

Parameters:

txt - The message in text form

---

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener interface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

1.26 Class MobileClient

java.lang.Object

|

+--Client

|

+--**MobileClient**

---

public class **MobileClient**
extends Client

The MobileClient class is the the main class of the Mobile Client Component of the Car Tracking System. It holds informations about the Mobile Client and is the actual Client application for the Mobile User

---

Field Summary

protected boolean	**GISStarted**
protected GPSDevice	**GPS**

Fields inherited from class Client
gisCom

Constructor Summary

MobileClient(java.lang.String Name, java.lang.String IP) Creates an instance of the MobileClient with a specific Name and IP address

Method Summary

void	**closeClient**() Closes the application and the connection to the Server
static void	**main**(java.lang.String[] args) The main function used to run the application
void	**sendMessage**(java.lang.String text) Sends a message to the Central Client through the Server.
void	**setCoordinate**(Coordinate c) Sets the coordinates of the Mobile Client
void	**setDBStructure**(Database db) Sets the Database object used by the Mobile Client
void	**showMessage**(Message msg) Shows a message in the main GUI.
void	**showResult**(Result res) Shows the results in a frame
void	**startGPS**() Starts the GPS drevice reading

Methods inherited from class Client
addResult, getName, getResultLog, run, sendQuery, setClientType

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

GPS

protected GPSDevice **GPS**

GISStarted

protected boolean **GISStarted**

Constructor Detail

MobileClient

public **MobileClient**(java.lang.String Name,
java.lang.String IP)

Creates an instance of the MobileClient with a specific Name and IP address

Parameters:

Name - The name of the MobileClient

IP - The IP address where the Server is running (127.0.0.1 can be used for local connections)

Method Detail

showResult

public void **showResult**(Result res)

Shows the results in a frame

Overrides:

showResult in class Client

Parameters:

res - The Result object to be showed

---

startGPS

public void **startGPS**()

Starts the GPS drevice reading

---

setDBStructure

public void **setDBStructure**(Database db)

Sets the Database object used by the Mobile Client

Overrides:

setDBStructure in class Client

Parameters:

db - The database object

---

setCoordinate

public void **setCoordinate**(Coordinate c)

Sets the coordinates of the Mobile Client

Parameters:

c - The coordinate of the Mobile Client

---

sendMessage

public void **sendMessage**(java.lang.String text)

Sends a message to the Central Client through the Server.

Parameters:

text - The message in the form of text

---

showMessage

public void **showMessage**(Message msg)

Shows a message in the main GUI. It basically calls the addMessage(Message msg) of the MainGUI class.

Overrides:

showMessage in class Client

Parameters:

msg - The Message object to show

---

closeClient

public void **closeClient**()

Closes the application and the connection to the Server

Overrides:

closeClient in class Client

---

main

public static void **main**(java.lang.String[] args)

The main function used to run the application

1.27 Class MobileClientThread

java.lang.Object

|

+--java.lang.Thread

|

+--**MobileClientThread**

All Implemented Interfaces:

java.lang.Runnable

---

public class **MobileClientThread**
extends java.lang.Thread

The MobileClientThread class is a thread run by a client and takes care of receiving data from the Server.

Field Summary	
protected ClientServerCommunication	**CSCom**
protected java.awt.Component	**GUI**
protected java.io.ObjectInputStream	**input**
protected Client	**localClient**
protected java.net.Socket	**server**
protected User	**user**

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

MobileClientThread(java.net.Socket server, Client localClient) Creates an instance of the MobileClientThread class

Method Summary

ClientServerCommunication	**getCommunication**() Returns the communication componet
protected void	**getDBStructure**()
protected void	**getMessage**()
protected void	**getPosition**()
protected void	**getResult**()
User	**getUser**() Returns the User
protected void	**getUserLOGIN**()
protected void	**getUserLOGOUT**()
void	**run**() Starts the Thread
void	**setGUI**(java.awt.Component c)
void	**terminateConnection**(java.io.ObjectOutputStream output) Terminates the connection with the Server

Methods inherited from class java.lang.Thread

activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

server

protected java.net.Socket **server**

---

localClient

protected Client **localClient**

---

input

protected java.io.ObjectInputStream **input**

---

user

protected User **user**

CSCom

protected ClientServerCommunication **CSCom**

GUI

protected java.awt.Component **GUI**

Constructor Detail

MobileClientThread

public **MobileClientThread**(java.net.Socket server,
Client localClient)

Creates an instance of the MobileClientThread class

Parameters:

localClient - Reference to the running client

Method Detail

setGUI

public void **setGUI**(java.awt.Component c)

---

getCommunication

public ClientServerCommunication **getCommunication**()

Returns the communication componet

Returns:

The communication component

---

run

public void **run**()

Starts the Thread

Specified by:

run in interface java.lang.Runnable

Overrides:

run in class java.lang.Thread

---

getUser

public User **getUser**()

Returns the User

Returns:

The User

---

getUserLOGIN

protected void **getUserLOGIN**()

---

getUserLOGOUT

protected void **getUserLOGOUT**()

---

getPosition

protected void **getPosition**()

---

getResult

protected void **getResult**()

---

getDBStructure

protected void **getDBStructure**()

---

terminateConnection

public void **terminateConnection**(java.io.ObjectOutputStream output)

Terminates the connection with the Server

Parameters:

output - The output stream to the Sever

---

getMessage

protected void **getMessage**()

1.28 Class NotEnoughParameterException

java.lang.Object
 |
 +--java.lang.Throwable
  |
  +--java.lang.Exception
   |
   +--**NotEnoughParameterException**

All Implemented Interfaces:

java.io.Serializable

---

public class **NotEnoughParameterException**
extends java.lang.Exception

The NotEnoughParameterException is thrown when a connection is attempted to the database server, but the parameter used are not enough

See Also:

Serialized Form

Constructor Summary

NotEnoughParameterException()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace,

printStackTrace, printStackTrace, setStackTrace, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NotEnoughParameterException

public **NotEnoughParameterException**()

1.29 Class OnLineUsersGUI

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|
+--javax.swing.JPanel
|
+--**OnLineUsersGUI**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.util.Observer, java.io.Serializable

---

public class **OnLineUsersGUI**
extends javax.swing.JPanel
implements java.util.Observer

The OnLineUsersGUI class is the GUI showing the on line users of the system

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JPanel

javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

OnLineUsersGUI(CentralClient localClient) Create and instance of the OnLineUsersGUI class

Method Summary

void	**update**(java.util.Observable o, java.lang.Object arg)

	Implementation of the Observer interface

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount,

getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

OnLineUsersGUI

public **OnLineUsersGUI**(CentralClient localClient)

Create and instance of the OnLineUsersGUI class

Parameters:

localClient - The Central Client

Method Detail

update

public void **update**(java.util.Observable o,
java.lang.Object arg)

Implementation of the Observer interface

Specified by:

update in interface java.util.Observer

1.30 Class Query

java.lang.Object
|
+--**Query**

All Implemented Interfaces:

java.io.Serializable

---

public class **Query**
extends java.lang.Object
implements java.io.Serializable

The Query class describes a query made by the user

See Also:

Serialized Form

Constructor Summary

Query(java.lang.String DBName, java.lang.String tableName) Creates a Query object for a specific table in a specific database

Method Summary

void	**addField**(QueryField RF) Adds a new field to the query
java.lang.String	**getDBName**()

	Return the database name
QueryField	**getField**(int index) Returns a specific field in the query
int	**getFieldNo**() Returns the number of fields in the query
java.lang.String	**getTableName**() Returns the table where the query has been made
boolean	**isEmpty**() Returns wheter the Query is empty or not

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Query

public **Query**(java.lang.String DBName,
java.lang.String tableName)
Creates a Query object for a specific table in a specific database

Parameters:

DBName - The database name

tableName - The table name

Method Detail

getDBName

public java.lang.String **getDBName**()
Return the database name

Returns:

The database name

---

getTableName

public java.lang.String **getTableName**()
Returns the table where the query has been made

Returns:

The table name

---

getFieldNo

public int **getFieldNo**()

Returns the number of fields in the query

Returns:

The fields number

---

addField

public void **addField**(QueryField RF)

Adds a new field to the query

Parameters:

RF - The field to be added

---

getField

public QueryField **getField**(int index)

Returns a specific field in the query

Parameters:

index - The index of the field

---

isEmpty

public boolean **isEmpty**()

Returns wheter the Query is empty or not

Returns:

true/false

1.31 Class QueryField

java.lang.Object

|

+--Field

|

+--ResultField

|

+--**QueryField**

All Implemented Interfaces:

java.io.Serializable

---

public class **QueryField**
extends ResultField

The QueryField class represents a field in the Query object

See Also:

Serialized Form

Field Summary

protected java.lang.String	**criteria**

Fields inherited from class Field
BOOLEAN, FLOAT, GRAPHIC, INT, TEXT

Constructor Summary

QueryField(java.lang.String name, java.lang.String type, java.lang.String criteria, java.lang.Object value) Creates a QueryField object

Method Summary

boolean	**equals**(java.lang.Object obj) Compares two QueryField Objects
java.lang.String	**getCriteria**() Returns the field search criteria

Methods inherited from class ResultField
getValue, toString

Methods inherited from class Field
getName, getSearcheable, getSize, getType

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

criteria

protected java.lang.String **criteria**

Constructor Detail

QueryField

public **QueryField**(java.lang.String name,
java.lang.String type,
java.lang.String criteria,
java.lang.Object value)

Creates a QueryField object

Parameters:

name - The field name

type - The field type

criteria - The field search criteria

value - The field value

Method Detail

getCriteria

public java.lang.String **getCriteria**()

Returns the field search criteria

Returns:

The search criteria

---

equals

public boolean **equals**(java.lang.Object obj)

Compares two QueryField Objects

Overrides:

equals in class ResultField

Parameters:

obj - The QueryField to compare

Returns:

true false

1.32 Class QueryGUI

java.lang.Object
|
+--java.awt.Component
|

```
java.awt.Container
    |
    +--java.awt.Window
        |
        +--java.awt.Frame
            |
            +--javax.swing.JFrame
                |
                +--QueryGUI
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **QueryGUI**
extends javax.swing.JFrame
implements java.awt.event.ActionListener

The QueryGUI class is the window used by clients to send queries to the Server

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component

java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

QueryGUI(Database DB, Client localClient) Creates a QueryGUI object

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener interface
Query	**createQuery**()

	automatically generates the look of the GUI

Methods inherited from class javax.swing.JFrame

addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys,

setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer

getFont, postEvent

Constructor Detail

QueryGUI

public **QueryGUI**(Database DB,
Client localClient)

Creates a QueryGUI object

Parameters:

DB - The database

localClient - The Client

Method Detail

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener interface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

---

createQuery

public Query **createQuery**()

automatically generates the look of the GUI

1.33 Class Result

java.lang.Object
|
+--**Result**

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

ResultCache

---

public class **Result**
extends java.lang.Object
implements java.io.Serializable

The Result holds a result from the database, a row with dynamic number of fields

See Also:

Serialized Form

Field Summary

static int	**OK** Result Status OK
protected java.util.Vector	**resultRow**
protected int	**status**
static int	**TOO_BROAD** Result Status Too broad

Constructor Summary

Result() Creates a new empty Result Object

Method Summary

void	**addResultField**(int row, ResultField field) Adds a new ResultField in a specific row
int	**getColumns**() Returns the number of columns in the Result
ResultField	**getResultField**(int row, int col) Returns one Field from the results
java.util.Vector	**getResultRow**(int row) Returns a row of results
int	**getRows**() Returns the number of rows in the Result
int	**getStatus**() Returns the status of the Result
boolean	**isEmpty**() Returns true if the Result is empty
void	**setStatus**(int status) Sets the status of the Result
java.lang.String	**toString**() Returns a String representation of the Result object

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

OK

public static final int **OK**

Result Status OK

See Also:

Constant Field Values

TOO_BROAD

public static final int **TOO_BROAD**

Result Status Too broad

See Also:

Constant Field Values

---

resultRow

protected java.util.Vector **resultRow**

---

status

protected int **status**

Constructor Detail

Result

public **Result**()

Creates a new empty Result Object

Method Detail

isEmpty

public boolean **isEmpty**()

Returns true if the Result is empty

---

setStatus

public void **setStatus**(int status)

Sets the status of the Result
Parameters:
status - The result status

---

getStatus

public int **getStatus**()

Returns the status of the Result

---

getResultRow

public java.util.Vector **getResultRow**(int row)

Returns a row of results
Parameters:
row - The row index

---

getResultField

public ResultField **getResultField**(int row,
int col)

Returns one Field from the results
Parameters:
row - - The row number in the result grid
col - - the field number in the row of results
Returns:
The Field

---

addResultField

public void **addResultField**(int row,
ResultField field)

Adds a new ResultField in a specific row
Parameters:
row - The row of results
field - The field to be added

---

getRows

public int **getRows**()

Returns the number of rows in the Result

---

getColumns

public int **getColumns**()

Returns the number of columns in the Result

---

toString

public java.lang.String **toString**()

Returns a String representation of the Result object

Overrides:

toString in class java.lang.Object

Returns:

String Representation

1.34 Class ResultCache

java.lang.Object

|

+--Result

|

+--**ResultCache**

All Implemented Interfaces:

java.io.Serializable

---

public class **ResultCache**
extends Result

The ResultCache class is a cache for Result objects

See Also:

Serialized Form

Field Summary

static int	**MAXLOG**

Fields inherited from class Result
OK, resultRow, status, TOO_BROAD

Constructor Summary

ResultCache() Creates a ResultCache object

Method Summary

void	**addResult**(Result rs) Adds a Result object to the cache

Methods inherited from class Result
addResultField, getColumns, getResultField, getResultRow, getRows, getStatus, isEmpty, setStatus, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

MAXLOG

public static int **MAXLOG**

Constructor Detail

ResultCache

public **ResultCache**()

Creates a ResultCache object

Method Detail

addResult

public void **addResult**(Result rs)

Adds a Result object to the cache

Parameters:

rs - The result to be added

1.35 Class ResultField

java.lang.Object

|

+--Field

|

+--**ResultField**

All Implemented Interfaces:

java.io.Serializable

Direct Known Subclasses:

QueryField

---

public class **ResultField**
extends Field

The ResultField class extends the Field class and describes one of the fields in the Result Class

See Also:

Serialized Form

Field Summary

Fields inherited from class Field
BOOLEAN, FLOAT, GRAPHIC, INT, TEXT

Constructor Summary

ResultField(java.lang.String name, java.lang.String type, java.lang.Object value) Creates a ResultField object

Method Summary

boolean	**equals**(java.lang.Object obj) Compares two ResultField Objects
java.lang.Object	**getValue**() Returns the value of the Field
java.lang.String	**toString**() Returns a String representation of the object

Methods inherited from class Field
getName, getSearcheable, getSize, getType

Methods inherited from class java.lang.Object

clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ResultField

public **ResultField**(java.lang.String name,
java.lang.String type,
java.lang.Object value)

Creates a ResultField object

Parameters:

name - The name of the Field

type - The Type of the Field

value - The value of the Field

Method Detail

getValue

public java.lang.Object **getValue**()

Returns the value of the Field

Returns:

The value

---

equals

public boolean **equals**(java.lang.Object obj)

Compares two ResultField Objects

Overrides:

equals in class Field

Parameters:

obj - The ResultField to compare

Returns:

true false

---

toString

public java.lang.String **toString**()

Returns a String representation of the object

Overrides:

toString in class java.lang.Object

Returns:

String Representation

1.36 Class ResultGUI

java.lang.Object
 |
 +--java.awt.Component
  |
  +--java.awt.Container
  |
  +--java.awt.Window
  |
  +--java.awt.Frame
  |
  +--javax.swing.JFrame
  |
  +--**ResultGUI**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **ResultGUI**
extends javax.swing.JFrame
implements java.awt.event.ActionListener

The ResultGUI class is the window displayin the Result objects

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ResultGUI(Result rs, MobileClient localClient) Creates and display the ResultGUI

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener inteface
void	**createGUI**(int row) Starts the process of automatic creation of the GUI based on one of the Results

Methods inherited from class javax.swing.JFrame

addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame

addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation,

setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
getFont, postEvent

Constructor Detail

ResultGUI

public **ResultGUI**(Result rs,
MobileClient localClient)

Creates and display the ResultGUI

Parameters:

rs - The Result to show

localClient - The Mobile Client

Method Detail

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener inteface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

---

createGUI

public void **createGUI**(int row)

Starts the process of automatic creation of the GUI based on one of the Results

Parameters:

row - The row index of the result to be showed from the Result object

---

1.37 Class ResultInfo

java.lang.Object

|

+--**ResultInfo**

---

public class **ResultInfo**
extends java.lang.Object

The ResultInfo class is used as a transport packet for the Result Object in the network

---

Field Summary

java.lang.Integer	**CCID** The Central Client ID
Result	**result** The result Object

Constructor Summary

ResultInfo()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

result

public Result **result**

The result Object

CCID

public java.lang.Integer **CCID**

The Central Client ID

Constructor Detail

ResultInfo

public **ResultInfo**()

1.38 Class ScrollablePicture

java.lang.Object
|
+--java.awt.Component
|
+--java.awt.Container
|
+--javax.swing.JComponent
|
+--javax.swing.JLabel
|
+--**ScrollablePicture**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.Scrollable, java.io.Serializable, javax.swing.SwingConstants

---

public class **ScrollablePicture**
extends javax.swing.JLabel
implements javax.swing.Scrollable

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JLabel
javax.swing.JLabel.AccessibleJLabel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JLabel
labelFor

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.SwingConstants
BOTTOM, CENTER, EAST, HORIZONTAL, LEADING, LEFT, NEXT, NORTH, NORTH_EAST, NORTH_WEST, PREVIOUS, RIGHT, SOUTH, SOUTH_EAST, SOUTH_WEST, TOP, TRAILING, VERTICAL, WEST

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ScrollablePicture(javax.swing.ImageIcon i, int m)	

Method Summary

java.awt.Dimension	**getPreferredScrollableViewportSize**()
int	**getScrollableBlockIncrement**(java.awt.Rectangle visibleRect, int orientation, int direction)
boolean	**getScrollableTracksViewportHeight**()

boolean	**getScrollableTracksViewportWidth**()
int	**getScrollableUnitIncrement**(java.awt.Rectangle visibleRect, int orientation, int direction)
void	**setMaxUnitIncrement**(int pixels)

Methods inherited from class javax.swing.JLabel

checkHorizontalKey, checkVerticalKey, getAccessibleContext, getDisabledIcon, getDisplayedMnemonic, getDisplayedMnemonicIndex, getHorizontalAlignment, getHorizontalTextPosition, getIcon, getIconTextGap, getLabelFor, getText, getUI, getUIClassID, getVerticalAlignment, getVerticalTextPosition, imageUpdate, paramString, setDisabledIcon, setDisplayedMnemonic, setDisplayedMnemonic, setDisplayedMnemonicIndex, setHorizontalAlignment, setHorizontalTextPosition, setIcon, setIconTextGap, setLabelFor, setText, setUI, setVerticalAlignment, setVerticalTextPosition, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener,

removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent,

processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ScrollablePicture

public **ScrollablePicture**(javax.swing.ImageIcon i,
int m)

Method Detail

getPreferredScrollableViewportSize

public java.awt.Dimension **getPreferredScrollableViewportSize**()

Specified by:

getPreferredScrollableViewportSize in interface javax.swing.Scrollable

---

getScrollableUnitIncrement

public int **getScrollableUnitIncrement**(java.awt.Rectangle visibleRect,
int orientation,
int direction)

Specified by:

getScrollableUnitIncrement in interface javax.swing.Scrollable

---

getScrollableBlockIncrement

public int **getScrollableBlockIncrement**(java.awt.Rectangle visibleRect,
int orientation,
int direction)

Specified by:

getScrollableBlockIncrement in interface javax.swing.Scrollable

---

getScrollableTracksViewportWidth

public boolean **getScrollableTracksViewportWidth**()

Specified by:

getScrollableTracksViewportWidth in interface javax.swing.Scrollable

---

getScrollableTracksViewportHeight

public boolean **getScrollableTracksViewportHeight**()

Specified by:

getScrollableTracksViewportHeight in interface javax.swing.Scrollable

---

setMaxUnitIncrement

public void **setMaxUnitIncrement**(int pixels)

1.39 Class SDatabase

java.lang.Object
 |
 +--Database
  |
  +--**SDatabase**

All Implemented Interfaces:

java.io.Serializable

---

public class **SDatabase**
extends Database
implements java.io.Serializable

The SDatabase class holds the informations about specific databases that the server can access.

See Also:

Serialized Form

Field Summary

Fields inherited from class Database

DBTable, name

Constructor Summary

SDatabase(java.lang.String name, java.lang.String path, java.lang.String driver) Creates a SDatabase Object with the specific path and no username or password
SDatabase(java.lang.String name, java.lang.String path, java.lang.String username, java.lang.String password, java.lang.String driver) Creates a SDatabase Object with the specific path, username and password

Method Summary

void	**connectToDB**() Connects to the Database using the Database path, username and password stored in the internal variables.
void	**disconnectFromDB**() Disconnects from the database.
boolean	**equals**(java.lang.Object O) Compares two SDatabase Objects
java.lang.String	**getDBPath**() Returns the Database Path
java.lang.String	**getPassword**() Returns the Database Password
java.lang.String	**getUsername**() Returns the Database Username
java.lang.Object	**sendQuery**(java.lang.Object query) Sends a query to the database.
Database	**toDatabase**() Translates the SDatabase object into a Database object
java.lang.Object	**translateQuery**(java.lang.Object query) Translates a Query to an SQL sentence

Methods inherited from class Database
getName, getTable, setName, setTable

Methods inherited from class java.lang.Object

clone, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SDatabase

public **SDatabase**(java.lang.String name,
java.lang.String path,
java.lang.String driver)

Creates a SDatabase Object with the specific path and no username or password

---

SDatabase

public **SDatabase**(java.lang.String name,
java.lang.String path,
java.lang.String username,
java.lang.String password,
java.lang.String driver)

Creates a SDatabase Object with the specific path, username and password

Method Detail

connectToDB

public void **connectToDB**()
throws java.lang.Exception

Connects to the Database using the Database path, username and password stored in the internal variables.

Throws:

java.lang.Exception

---

disconnectFromDB

public void **disconnectFromDB**()
throws java.lang.Exception

Disconnects from the database.

java.lang.Exception

---

sendQuery

public java.lang.Object **sendQuery**(java.lang.Object query)
throws java.lang.Exception

Sends a query to the database.
Parameters:
query - The Query Object
Returns:
The result Object
java.lang.Exception

---

getDBPath

public java.lang.String **getDBPath**()

Returns the Database Path
Returns:
The Path

---

getUsername

public java.lang.String **getUsername**()

Returns the Database Username
Returns:
the Username

---

getPassword

public java.lang.String **getPassword**()

Returns the Database Password
Returns:
The Password

---

equals

public boolean **equals**(java.lang.Object O)

Compares two SDatabase Objects
Overrides:
equals in class Database
Returns:
true false

---

toDatabase

public Database **toDatabase**()

Translates the SDatabase object into a Database object
Returns:
The Database object

---

translateQuery

public java.lang.Object **translateQuery**(java.lang.Object query)

Translates a Query to an SQL sentence

Parameters:

query - The Query object

Returns:

The SQL statement

1.40 Class SearchPanel

java.lang.Object
 |
 +--java.awt.Component
  |
  +--java.awt.Container
   |
   +--javax.swing.JComponent
    |
    +--javax.swing.JPanel
     |
     +--**SearchPanel**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

---

public class **SearchPanel**
extends javax.swing.JPanel
implements java.awt.event.ActionListener

The SearchPanel class represents the Querying part of the Main window of the Mobile Client application

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Nested classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

SearchPanel(Database db, MobileClient localClient, MainGUI main) Creates and istance of SearchPanel	

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) Implementation of the ActionListener interface

Methods inherited from class javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout,

insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SearchPanel

public **SearchPanel**(Database db,
MobileClient localClient,
MainGUI main)

Creates and istance of SearchPanel

Parameters:

db - The Database structure

localClient - The MobileClient

main - The main window

Method Detail

actionPerformed

public void **actionPerformed**(java.awt.event.ActionEvent e)

Implementation of the ActionListener interface

Specified by:

actionPerformed in interface java.awt.event.ActionListener

1.41 Class Server

java.lang.Object
|
+--**Server**

---

public class **Server**
extends java.lang.Object

The Server class is the main class of the Server Component It holds all the informations about the database connections and handles all the service request form the clients

---

Field Summary

static int	**NO_QUERY_SERVICES** Describes the NO_QUERY_SERVICES status of the server
static int	**READY** Describes the READY status of the Server

Constructor Summary

Server() Creates and instance of the Server, setting up the socket to performe

communication

Method Summary

void	**addCentralClient**(ServerThread ST) Adds a Central Client to the list of running ServerThread
void	**addMobileClient**(ServerThread ST) Adds a Mobile Client to the list of running ServerThread
void	**broadcastMsgToMC**(Message msg) Broadcast a message to all Mobile Client
void	**broadcastUsersLOGIN**(User user) Broad casts to all the Central Clients the user that has logged in
void	**broadcastUsersLOGOUT**(User user) Broad casts to all the Central Clients the user that has logged out
Result	**fillResult**(java.sql.ResultSet rs, int dbIndex) Reads the Structure of the Database and uses it to transform the RecordSet object received from the specific Database into a Result Object.
SDatabase	**getDatabase**(int index) Returns a specific database connected to the server
int	**getDBID**(java.lang.String name)
void	**init**(java.lang.String filename, java.lang.String name, java.lang.String path, java.lang.String username, java.lang.String password, java.lang.String driver) Creates a new database file from a text file WARRING : Used only on demo version
void	**initiate**() Initiate the Server and check the sets the server status
void	**loadConfiguration**() Loads the configuration file if the configuration file is not found it will create a new file with default settings.
void	**mobileClientExit**(java.lang.String IP, ServerThread ST) Notifys to the Server that a client has closed the connection
void	**run**(boolean newTxtfile, java.lang.String filename, java.lang.String name, java.lang.String path, java.lang.String username, java.lang.String password, java.lang.String driver) Runs the Server Application
void	**sendMsgToCC**(Message msg) Sends a Message to all Central Clients

void	**sendMsgToMC**(Message msg) Sends a message to Mobile Client
void	**sendPosition**(GISObject GISO) Sends a GIS position to all Central Clients
void	**sendResult**(Result tmpRes, java.lang.Integer CID) Sends a Result object to a Mobile Client
void	**sendUsers**(int CCID) Sends a list of all the online Mobile CLients to a specific Central Client
void	**shutServer**() Shutdown the Server and free all the threads
void	**storeDBStructure**(java.lang.String DBname, java.lang.String file, java.lang.String path, java.lang.String username, java.lang.String password, java.lang.String driver) Stores the Database structure read from a file to the Database class

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

READY

public static final int **READY**

Describes the READY status of the Server

See Also:

Constant Field Values

NO_QUERY_SERVICES

public static final int **NO_QUERY_SERVICES**

Describes the NO_QUERY_SERVICES status of the server

See Also:

Constant Field Values

Constructor Detail

Server

public **Server**()

Creates and instance of the Server, setting up the socket to performe communication

Method Detail

loadConfiguration

public void **loadConfiguration**()

Loads the configuration file if the configuration file is not found it will create a new file with default settings. Present Default Settings: - Server Port number = 8888 - DB structure file name = database.dat

---

broadcastUsersLOGIN

public void **broadcastUsersLOGIN**(User user)

Broad casts to all the Central Clients the user that has logged in

---

broadcastUsersLOGOUT

public void **broadcastUsersLOGOUT**(User user)

Broad casts to all the Central Clients the user that has logged out

---

sendUsers

public void **sendUsers**(int CCID)

Sends a list of all the online Mobile CLients to a specific Central Client

Parameters:

CCID - The Central Client ID

---

sendResult

public void **sendResult**(Result tmpRes,
java.lang.Integer CID)

Sends a Result object to a Mobile Client

Parameters:

tmpRes - The Result object

CID - The Mobile Client ID

---

storeDBStructure

public void **storeDBStructure**(java.lang.String DBname,
java.lang.String file,
java.lang.String path,
java.lang.String username,
java.lang.String password,

java.lang.String driver)
throws InvalidStructureException

Stores the Database structure read from a file to the Database class

Throws:

InvalidStructureException - - the Database structure in the file is not valid

---

getDatabase

public SDatabase **getDatabase**(int index)

Returns a specific database connected to the server

Parameters:

index - The index ID of the database

Returns:

The Database

---

getDBID

public int **getDBID**(java.lang.String name)

---

fillResult

public Result **fillResult**(java.sql.ResultSet rs,
int dbIndex)

Reads the Structure of the Database and uses it to transform the RecordSet object received from the specific Database into a Result Object.

Parameters:

rs - The ResultSet containig the result from the database

dbIndex - The index ID of the database where the result come from

Returns:

A Result object

---

init

public void **init**(java.lang.String filename,
java.lang.String name,
java.lang.String path,
java.lang.String username,
java.lang.String password,
java.lang.String driver)

Creates a new database file from a text file WARRING : Used only on demo version

Parameters:

filename - The path and name of the texet file

path - The Database URL
username - The Username
password - The Password
driver - The jdbc driver

---

initiate

public void **initiate**()

Initiate the Server and check the sets the server status

---

run

public void **run**(boolean newTxtfile,
java.lang.String filename,
java.lang.String name,
java.lang.String path,
java.lang.String username,
java.lang.String password,
java.lang.String driver)

Runs the Server Application

---

addMobileClient

public void **addMobileClient**(ServerThread ST)

Adds a Mobile Client to the list of running ServerThread
Parameters:
ST - The ServerThread handling the client communication

---

addCentralClient

public void **addCentralClient**(ServerThread ST)

Adds a Central Client to the list of running ServerThread
Parameters:
ST - The ServerThread handling the client communication

---

mobileClientExit

public void **mobileClientExit**(java.lang.String IP,
ServerThread ST)

Notifys to the Server that a client has closed the connection
Parameters:

IP - The IP address of the client

ST - The server thread that served the client

---

sendMsgToCC

public void **sendMsgToCC**(Message msg)

Sends a Message to all Central Clients

Parameters:

msg - The message to be sent

---

sendPosition

public void **sendPosition**(GISObject GISO)

Sends a GIS position to all Central Clients

Parameters:

GISO - The GIS Object

---

sendMsgToMC

public void **sendMsgToMC**(Message msg)

Sends a message to Mobile Client

Parameters:

msg - The message to be sent (Receiver information are in the Message object)

---

broadcastMsgToMC

public void **broadcastMsgToMC**(Message msg)

Broadcast a message to all Mobile Client

Parameters:

msg - The message to be sent

---

shutServer

public void **shutServer**()

Shutdown the Server and free all the threads

1.42 Interface ServerClientCommunication

All Known Implementing Classes:

EBSCCommunication

---

public interface **ServerClientCommunication**

The ServerClientCommunication interface describes the communication from the Server to the Client

---

Method Summary

java.lang.Object	**readAction**(java.lang.Object input_connection) Receives the Action performad by the Client
java.lang.Object	**receive_message**(java.lang.Object input_connection) Receives a Message from one of the Clients
java.lang.Object	**receive_Position**(java.lang.Object input_connection) Receives a GIS Position from one of the Clients
java.lang.Object	**receiveInfo**(java.lang.Object input_connection) Receives an Info object from one of the Clients
java.lang.Object	**receiveQuery**(java.lang.Object input_connection) Receives a Query from one of the Clients
java.lang.Object	**receiveResult**(java.lang.Object input_connection) Receives a Result from one of the Central Clients
void	**send_message**(java.lang.Object message, java.lang.Object output_connection) Sends a Message to one of the Clients
void	**send_Position**(java.lang.Object Position, java.lang.Object output_connection) Sends a GIS Position to one of the Clients
void	**sendDBStructure**(java.lang.Object DBStructure, java.lang.Object output_connection) Sends the database structure to a Client
void	**sendResult**(java.lang.Object result, java.lang.Object DBID, java.lang.Object output_connection) Sends a result to a Client
void	**sendUserLOGIN**(java.lang.Object user, java.lang.Object output_connection) Notifys to a CentralClient that a Mobile Client has logged in
void	**sendUserLOGOUT**(java.lang.Object user, java.lang.Object output_connection) Notifys to a CentralClient that a Mobile Client has logged out

Method Detail

sendResult

public void **sendResult**(java.lang.Object result,
java.lang.Object DBID,
java.lang.Object output_connection)

throws java.lang.Exception

Sends a result to a Client

Parameters:

result - The Result to be sent

DBID - The used database ID (included for future development of the demo)

output_connection - The object describing the output connection to the client

java.lang.Exception

---

sendDBStructure

public void **sendDBStructure**(java.lang.Object DBStructure,
java.lang.Object output_connection)
throws java.lang.Exception

Sends the database structure to a Client

Parameters:

DBStructure - The database structure to be sent

output_connection - The object describing the output connection to the client

java.lang.Exception

---

receiveQuery

public java.lang.Object **receiveQuery**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a Query from one of the Clients

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

sendUserLOGIN

public void **sendUserLOGIN**(java.lang.Object user,
java.lang.Object output_connection)
throws java.lang.Exception

Notifys to a CentralClient that a Mobile Client has logged in

Parameters:

user - The User that has logged in

output_connection - The object describing the output connection to the client

java.lang.Exception

---

sendUserLOGOUT

public void **sendUserLOGOUT**(java.lang.Object user,
java.lang.Object output_connection)
throws java.lang.Exception

Notifys to a CentralClient that a Mobile Client has logged out

Parameters:

user - The User that has logged out

output_connection - The object describing the output connection to the client

java.lang.Exception

---

readAction

public java.lang.Object **readAction**(java.lang.Object input_connection)
throws java.lang.Exception

Receives the Action performad by the Client

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

receiveResult

public java.lang.Object **receiveResult**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a Result from one of the Central Clients

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

receiveInfo

public java.lang.Object **receiveInfo**(java.lang.Object input_connection)
throws java.lang.Exception

Receives an Info object from one of the Clients

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

receive_message

public java.lang.Object **receive_message**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a Message from one of the Clients

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

send_message

public void **send_message**(java.lang.Object message,
java.lang.Object output_connection)
throws java.lang.Exception

Sends a Message to one of the Clients

Parameters:

message - The Message to deliver

output_connection - The object describing the output connection to the client

java.lang.Exception

---

receive_Position

public java.lang.Object **receive_Position**(java.lang.Object input_connection)
throws java.lang.Exception

Receives a GIS Position from one of the Clients

Parameters:

input_connection - The object describing the input connection from the client

java.lang.Exception

---

send_Position

public void **send_Position**(java.lang.Object Position,
java.lang.Object output_connection)
throws java.lang.Exception

Sends a GIS Position to one of the Clients

Parameters:

Position - The GIS Position

output_connection - The object describing the output connection to the client

java.lang.Exception

1.43 Class ServerThread

java.lang.Object
|
+--java.lang.Thread

|

+--ServerThread

All Implemented Interfaces:

java.lang.Runnable

---

public class **ServerThread**
extends java.lang.Thread

The ServerTread class is a thread run by the server in order to be able to handle multiple concurrent request from the clients

---

Field Summary

Fields inherited from class java.lang.Thread
MAX_PRIORITY, MIN_PRIORITY, NORM_PRIORITY

Constructor Summary

ServerThread(Server server, java.net.Socket socket) Creates an instance of the ServerThread Object	

Method Summary

void	**getMessage**() Reads a message from the communication Stream and performes the correlated action
protected void	**getPosition**() Receives a GIS Obejct
User	**getUser**() Return the information about the user connected to the Thread
void	**performQuery**() Reads the query from the commuication stream, sends it to the server.
protected void	**prepareResultToMobile**() Prepares a result to be sent to the Mobile Client, calls the Server.sendResult(...) function
void	**run**() Starts the Thread
void	**sendMessage**(Message msg) Sends a message to the connected client

void	**sendPosition**(GISObject GISO) Sends the position of a GIS Obejct to the connected client
void	**sendResultToMobile**(Result res) Sends a result to the Connected client
void	**sendUserLOGIN**(User user) Notifys the Client that a user has logged in
void	**sendUserLOGOUT**(User user) Notifys the Client that a user has logged out
void	**setThreadID**(int ID) Sets the ID of the Thread
void	**shutdown**() Runs the shutdown procedure

Methods inherited from class java.lang.Thread
activeCount, checkAccess, countStackFrames, currentThread, destroy, dumpStack, enumerate, getContextClassLoader, getName, getPriority, getThreadGroup, holdsLock, interrupt, interrupted, isAlive, isDaemon, isInterrupted, join, join, join, resume, setContextClassLoader, setDaemon, setName, setPriority, sleep, sleep, start, stop, stop, suspend, toString, yield

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ServerThread

public **ServerThread**(Server server,
java.net.Socket socket)

Creates an instance of the ServerThread Object

Parameters:

server - Reference to the creating object

socket - The communication socket assosiated to the client

Method Detail

setThreadID

public void **setThreadID**(int ID)

Sets the ID of the Thread

Parameters:
ID - The thread ID

---

getUser

public User **getUser**()

Return the information about the user connected to the Thread
Returns:
The User

---

sendUserLOGIN

public void **sendUserLOGIN**(User user)

Notifys the Client that a user has logged in
Parameters:
user - The user that has logged in

---

sendUserLOGOUT

public void **sendUserLOGOUT**(User user)

Notifys the Client that a user has logged out
Parameters:
user - The user that has logged out

---

run

public void **run**()

Starts the Thread
Specified by:
run in interface java.lang.Runnable
Overrides:
run in class java.lang.Thread

---

prepareResultToMobile

protected void **prepareResultToMobile**()

Prepares a result to be sent to the Mobile Client, calls the Server.sendResult(...) function

---

sendResultToMobile

public void **sendResultToMobile**(Result res)

Sends a result to the Connected client

Parameters:
res - The Result

---

getPosition

protected void **getPosition**()

Receives a GIS Obejct

---

sendPosition

public void **sendPosition**(GISObject GISO)

Sends the position of a GIS Obejct to the connected client
Parameters:
GISO - The GIS object

---

shutdown

public void **shutdown**()

Runs the shutdown procedure

---

performQuery

public void **performQuery**()

Reads the query from the commuication stream, sends it to the server. The result of the query is then sent back to the connected client

---

getMessage

public void **getMessage**()

Reads a message from the communication Stream and performes the correlated action

---

sendMessage

public void **sendMessage**(Message msg)

Sends a message to the connected client
Parameters:
msg - The Message to be sent

1.44 Class SImage

java.lang.Object

|

+--**SImage**

All Implemented Interfaces:

java.io.Serializable

---

public class **SImage**
extends java.lang.Object
implements java.io.Serializable

The SImage class is the rapresentation of an image in terms of bytes

See Also:

Serialized Form

Constructor Summary

SImage()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SImage

public **SImage**()

1.45 Class SysConst

java.lang.Object

|

+--**SysConst**

---

public abstract class **SysConst**
extends java.lang.Object

The SysConst class holds all the System constants

Field Summary

static int	**BROADCAST**

static int	**CENTRAL_CLIENT**
static int	**END_ROW**
static int	**END_TRANSACTION**
static int	**IMAGE_ON_STREAM**
static int	**MOBILE_CLIENT**
static int	**NO_RECEIVER**

Constructor Summary

SysConst()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

MOBILE_CLIENT

public static final int **MOBILE_CLIENT**

See Also:

Constant Field Values

CENTRAL_CLIENT

public static final int **CENTRAL_CLIENT**

See Also:

Constant Field Values

END_TRANSACTION

public static final int **END_TRANSACTION**

See Also:

Constant Field Values

END_ROW

public static final int **END_ROW**

See Also:

Constant Field Values

IMAGE_ON_STREAM

public static final int **IMAGE_ON_STREAM**

See Also:

Constant Field Values

NO_RECEIVER

public static final int **NO_RECEIVER**

See Also:

Constant Field Values

BROADCAST

public static final int **BROADCAST**

See Also:

Constant Field Values

Constructor Detail

SysConst

public **SysConst**()

1.46 Class Table

java.lang.Object

|

+--**Table**

All Implemented Interfaces:

java.io.Serializable

---

public class **Table**
extends java.lang.Object
implements java.io.Serializable

The Table class describes the table used in the database

See Also:

Serialized Form

Constructor Summary

Table(java.lang.String name) Creates a Table object with the spacific name

Method Summary

boolean	**equals**(java.lang.Object obj) Compares two Table objects
Field	**getField**(int index) Return the one of the fields of the table stored in the position described by the index
java.util.Vector	**getFields**() Returns the Vector holding the fields of the table
java.lang.String	**getName**() Return the name of the table

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Table

public **Table**(java.lang.String name)

Creates a Table object with the spacific name

Method Detail

getName

public java.lang.String **getName**()

Return the name of the table

Returns:

The table name

---

equals

public boolean **equals**(java.lang.Object obj)

Compares two Table objects

Overrides:
equals in class java.lang.Object
Parameters:
obj - The Table to compare
Returns:
true false

---

getFields

public java.util.Vector **getFields**()
Returns the Vector holding the fields of the table

---

getField

public Field **getField**(int index)
Return the one of the fields of the table stored in the position described by the index
Returns:
A table's Field

1.47 Class User

java.lang.Object
|
+--**User**

All Implemented Interfaces:
java.io.Serializable

---

public class **User**
extends java.lang.Object
implements java.io.Serializable

The User class represents a User of the System

See Also:
Serialized Form

Field Summary	
protected java.lang.String	**Name**
protected int	**serverThreadID**

Constructor Summary

User() Creates an empty User
User(java.lang.String Name, int serverThreadID) Creates a User

Method Summary

boolean	**equals**(User u) Compares two User object and returns true is they are equal
java.lang.String	**getName**() Returns the Name of the User
int	**getServerThreadID**() Returns the Server Thread ID of Server Thread connected to the client the user is using

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

serverThreadID

protected int **serverThreadID**

Name

protected java.lang.String **Name**

Constructor Detail

User

public **User**()
Creates an empty User

User

public **User**(java.lang.String Name,

int serverThreadID)

Creates a User

Parameters:

Name - The user name

serverThreadID - The ID of the Server Thread hadling the communication with the client the user is using

Method Detail

getName

public java.lang.String **getName**()

Returns the Name of the User

Returns:

The name

---

getServerThreadID

public int **getServerThreadID**()

Returns the Server Thread ID of Server Thread connected to the client the user is using

Returns:

The Server Thread ID

---

equals

public boolean **equals**(User u)

Compares two User object and returns true is they are equal

1.48 Class Users

java.lang.Object

|

+--java.util.Observable

|

+--**Users**

---

public class **Users**
extends java.util.Observable

The Users class is used as a container for the user's classes

---

Constructor Summary

Users() Creates an object of this class

Method Summary

void	**add**(User user) Adds a User
User	**get**(int index) Returns a user in a specific position
java.util.Vector	**getVector**() Returns the Vector holding the users
boolean	**hasUser**(java.lang.String name) Returns true if the the User searched is found
void	**remove**(User user) Removes a user

Methods inherited from class java.util.Observable
addObserver, clearChanged, countObservers, deleteObserver, deleteObservers, hasChanged, notifyObservers, notifyObservers, setChanged

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Users

public **Users**()

Creates an object of this class

Method Detail

add

public void **add**(User user)

Adds a User

Parameters:

user - The user

---

getVector

public java.util.Vector **getVector**()

Returns the Vector holding the users

Returns:

The vector

---

get

public User **get**(int index)

Returns a user in a specific position

Parameters:

index - The position

Returns:

The user

---

remove

public void **remove**(User user)

Removes a user

Parameters:

user - The user to remove

---

hasUser

public boolean **hasUser**(java.lang.String name)

Returns true if the the User searched is found

Parameters:

name - The User name

2 Source Code

2.1 Action Class

```
/** The <code>Action</code> class sets the constants for the communication
between the Client and the Server*/
public abstract class Action{

           /** The constant for SEND_QUERY*/
           public static final int SEND_QUERY = 1;

           /** The constant for SEND_RESULT*/
           public static final int SEND_RESULT = 2;

           /** The contant for SHUTDOWN*/
           public static final int SHUTDOWN = 3;

           /** The contant for SEND_DBSTRUCTURE*/
           public static final int SEND_DBSTRUCTURE = 4;

           /** The contant for SEND_USERLOGIN*/
           public static final int SEND_USERLOGIN = 5;

           /** The contant for SEND_INFO*/
           public static final int SEND_INFO = 6;

           /** The contant for SEND_USERLOGOUT*/
           public static final int SEND_USERLOGOUT = 7;

           /** The contant for SEND_MESSAGE*/
           public static final int SEND_MESSAGE = 8;

           /** The contant for RECEIVE_MESSAGE*/
           public static final int RECEIVE_MESSAGE = 9;

           /** The contant for SEND_POSITION*/
           public static final int SEND_POSITION = 10;

           /** The contant for RECEIVE_POSITION*/
           public static final int RECEIVE_POSITION = 11;

}
```

2.2 CentralClient Class

```
import java.net.*;
import java.io.*;
import java.util.*;

/** The <code>CentralClient</code> class is the the main class of the Central
Client Component
    of the Car Tracking System. It holds informations about the Central Client
and is the actual
    Client application for the Central Operator*/
public class CentralClient extends Client{
           CMainGUI main;
```

```
	OnLineUsersGUI onlineUsers;
	Users users;

	/** Creates an instance of the CentralClient with a specific Name and
IP address
	  * @param Name The name of the CentralClient
	  * @param IP The IP address where the Server is running (127.0.0.1
can be used for local connections)
	  */
	public CentralClient(String Name,String IP){

			super(Name,IP);
			users= new Users();
			super.setClientType(SysConst.CENTRAL_CLIENT);
			onlineUsers= new OnLineUsersGUI(this);
			users.addObserver(onlineUsers);

	}//CentralClient

	/** Return the <code>Vector</code> used by the <code>Users</code>
class to hold instaces
	  * of the <code>User</code> class
	  * @return The Users's vector
	  */
	public Vector getUsersVector(){
			return users.getVector();
	}

	/** Returns the users on line
	  * @return the online users
	  */
	public Users getUsers(){
			return users;
	}

	/** Shows the results in a frame
	  * @param res The <code>Result</code> object to be showed
	  */
	public void showResult(Result res){
			main.showResult(res);
	}

	/** Sends a result to a specific MobileClient through the Server
	  * @param rs The result to be sent
	  * @param ClientID The ID of the receiving Mobile Client
	  */
	public void sendResult(Result rs, int ClientID){
			try{
				MCT.getCommunication().sendResult(rs,output,new
Integer(ClientID));
				}
			catch(Exception e){System.out.println("Exception at
Client"+e);}
	}

	/** Sets the <code>Database</code> object used by the Central Client
	  * @param db The database object
	  */
	public void setDBStructure(Database db){
```

```
                    this.db=db;
                    main = new CMainGUI(db,onlineUsers,this);
                    MCT.setGUI(main);
          }

          /** Notifies the CentralClient that a user has logged in the system
            * @param user The logging user
            */
          public void userLoggedIn(User user){
                    users.add(user);
                    try{
                              gisCom.createObject(new
GISObject(user.getName(),new Coordinate(0,0)));
                              }
                    catch(Exception e){System.out.println("GIS Exception at
CentralClient : "+e);}
          }


          /** Notifies the CentralClient that a user has logged out of the
system
            * @param user The logging-out user
            */
          public void userLoggedOut(User user){
                    users.remove(user);
                    try{
                              gisCom.deleteObject(new
GISObject(user.getName(),new Coordinate(0,0)));
                              }
                    catch(Exception e){System.out.println("GIS Exception at
CentralClient : "+e);}
          }

          /** Updates the Position of a Mobile Client in the GIS
            * @param GISO The <code>GISObject</code> that holds the position
            */
          public void getPosition(GISObject GISO){
                    try{
                              gisCom.updateObject(GISO);
                              }
                    catch(Exception e){System.out.println("GIS Exception at
CentralClient : "+e);}
          }


          /** Sends a message to a Mobile Client through the Server.
            * If a message is to be sent as a broad cast the value of
<code>receiverIndex</code>
            * must be set to <code>SysConst.BROADCAST</code>.
            * @param text The message in the form of text
            * @param receiverIndex The ID of the MobileClient receiving the
message
            */
          public void sendMessage(String text,int receiverIndex){
                    Message msg = new Message(text);
                    msg.setReceiver(receiverIndex);
                    try{

          MCT.getCommunication().send_message(msg,output);
```

```
                              }
                    catch(Exception e){System.out.println("Msg Exception at
centralClient "+e);}
          }


          /** The main function used to run the application*/
          public static void main(String[] args){
                    CentralClient client= new CentralClient("Central
Client",args[0]);
                    client.run();
          }

          /** Shows a message in the main GUI. It basically calls the
<code>addMessage(Message msg)</code>
            * of the <code>CMainGUI</code> class.
            * @param msg The <code>Message</code> object to show
            */
          public void showMessage(Message msg){
                    main.addMessage(msg);
          }

}
```

2.3 Client Class

```
import java.net.*;
import java.io.*;

/** The <code>Client</code> is the superclass used to implement all the common
features
  * between the CentralClient and the MobileClient
  */
public class Client{
          Socket server;
          ObjectOutputStream output;
          Database db;
          MobileClientThread MCT;
          ResultCache resultLog;
          Info info;
          protected GISCommunication gisCom;

          /** Creates a Client object and attempts the connection to the Server
            * @param Name The name of the Client
            * @param IP The Ip address where the Server is located
            */
          public Client(String Name, String IP){
                    info= new Info();
                    info.clientName=Name;
                    resultLog= new ResultCache();
                    db=new Database("temp");

                    try{
                              gisCom=new GIS();
                              gisCom.connect();
                              System.out.println("Attempt Connection...");
                              server=new Socket(IP,8888);
                              System.out.println("Connection Established
!!!!");
```

```
                                output=new
ObjectOutputStream(server.getOutputStream());
                                }
                        catch(Exception e){
                                System.out.println("Connection failed !!!");
                                System.out.println("Exception at Client "+e);
                                }
            }//Client



            /** Sets the type of Client :
              *                          <code>SysConst.MOBILE_CLIENT</code>

             <code>SysConst.CENTRAL_CLIENT</code>
              * @param type The Client Type
              */
            protected void setClientType(int type){
                        info.clientType=type;
            }


            /** Return the Result cache used by the Client
              * @return The result cache
              */
            public ResultCache getResultLog(){
                        return resultLog;
            }

            /** Returns the Name of the Client
              * @return The CLient name
              */
            public String getName(){
                        return info.clientName;
            }

            /** Runs the Client
              */
            public void run(){
                        MCT=new MobileClientThread(server,this);
                        MCT.start();
                        try{
                                MCT.getCommunication().sendInfo(info,output);
                                }
                        catch(Exception e){System.out.println("Exception at Client
"+e);}


            }//run

            public void showResult(Result res){}


            public void showMessage(Message msg){}



            /** Adds a Ruesult to the Client's Result Cache
              * @param res The result to be added
```

```
            */
           public void addResult(Result res){
                      resultLog.addResult(res);
           }


           /** Closes the Client and the connection with the Server
             */
           public void closeClient(){
                      try{

           MCT.getCommunication().terminateConnection(output);
                                  }
                      catch(Exception e){System.out.println("Exception at Client
"+e);}

           }


           /** Sends a <code>Query</code> to the Server
             * @param The Query
             */
           public void sendQuery(Query q){
                      try{
                                 MCT.getCommunication().sendQuery(q,output);
                                 }
                      catch(Exception e){System.out.println("Exception at Client
"+e);}
           }


           /** Sets the Database of the Client
             * @param db The Database object
             */
           public void setDBStructure(Database db){
                      this.db=db;
           }

}
```

2.4 ClientServerCommunication Interface

```
import java.io.*;


/** The <code>ClientServerCommunication</code> interface describes the
communication
  * from the Client to the Server
  */
public interface ClientServerCommunication{
           //*******Query System*********

           /** Sends a Query
             * @param query The Query object
             * @param output_connection The object describing the output
connection to the Server
             */
           public void sendQuery(Object query, Object output_connection) throws
Exception;
```

```
            /** Sends a Result
              * @param result The Result object
              * @param output_connection The object describing the output
connection to the Server
              * @param clientID The ID of the receiving Client
              */
            public void sendResult(Object result,Object output_connection,Object
clientID) throws Exception;

            /** Returns a recived Result
              * @param input_connection The object describing the input
connection from the Server
              * @return The received Result
              */
            public Object receiveResult(Object input_connection) throws
Exception;

            /** Receives a Database Structure
              * @param input_connection The object describing the input
connection from the Server
              * @return The Database Structure
              */
            public Object receiveDBStructure(Object input_connection) throws
Exception;

            /** Receives User's Info
              * @param input_connection The object describing the input
connection from the Server
              * @return The User's info
              */
            public Object receiveUserInfo(Object input_connection) throws
Exception;

            /** Receives the Action that is received by the Server
              * @param input_connection The object describing the input
connection from the Server
              * @return The Action
              */
            public Object readAction(Object input_connection) throws Exception;

            /** Sends a request of temination of the connection to the server
              * @param output_connection The object describing the output
connection to the Server
              */
            public void terminateConnection(Object output_connection) throws
Exception;

            /** Sends the User Info to the Server
              * @param info The User's Info
              * @param output_connection The object describing the output
connection to the Server
              */
            public void sendInfo(Object info,Object output_connection) throws
Exception;

            //*******Messaging System******

            /** Receives a message from the Server
```

```
             * @param input_connection The object describing the input
connection from the Server
             * @return The received Message
             */
           public Object receive_message(Object input_connection) throws
Exception;

           /** Sends a Message to the Server
             * @param output_connection The object describing the output
connection to the Server
             */
           public void send_message(Object message,Object output_connection)
throws Exception;

           //*******Tracking System*******

           /** Receives a GIS Position from the Server
             * @param input_connection The object describing the input
connection from the Server
             * @return The GIS Position
             */
           public Object receive_Position(Object input_connection) throws
Exception;

           /** Sends a GIS Position to the Server
             * @param Position The GIS Position
             * @param output_connection The object describing the output
connection to the Server
             */
           public void send_Position(Object Position,Object output_connection)
throws Exception;

}
```

2.5 CMainGUI Class

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

/** The <code>CMainGUI</code> represents the main window of the Central Client
  */
public class CMainGUI extends JFrame {
           Database db;
           CSearchPanel searchPanel;
           JToolBar statusBar;
           JLabel statusLb;
           CMessageGUI msgPanel;
           protected OnLineUsersGUI usersGUI;
           protected CentralClient localClient;


           /** Creates an instance of CMainGUI
             * @param db The Database structure
             * @param usersGUI The online users GUI
             * @param localClient The CentralClient
             */
```

```
	public CMainGUI(Database db,OnLineUsersGUI usersGUI,CentralClient
localClient){
			super("Car Tracking System");
			this.localClient=localClient;
			this.db=db;
			this.localClient=localClient;
			this.usersGUI=usersGUI;

			searchPanel= new CSearchPanel(db,localClient,this);
			statusBar= new JToolBar();
			statusLb= new JLabel("Connected");
			statusLb.setBorder(new
SoftBevelBorder(BevelBorder.LOWERED));
			statusBar.setFloatable(false);
			statusBar.setBorder(new
SoftBevelBorder(BevelBorder.RAISED));
			statusBar.add(statusLb);

			msgPanel= new CMessageGUI(localClient);

			JScrollPane scrollPane= new JScrollPane(usersGUI);
			scrollPane.setPreferredSize(new Dimension(200,200));



			this.getContentPane().setLayout(new BorderLayout());
			this.getContentPane().add(searchPanel,BorderLayout.NORTH);
			this.getContentPane().add(statusBar,BorderLayout.SOUTH);
			this.getContentPane().add(msgPanel,BorderLayout.CENTER);
			this.getContentPane().add(scrollPane,BorderLayout.WEST);
			this.addWindowListener(new WindowAdapter(){
				public void windowClosing(WindowEvent e){
					CMainGUI
frame=(CMainGUI)e.getWindow();
					frame.close();
				}
			});

			pack();
			setVisible(true);

	}

	/** Closes the window and the application*/
	public void close(){
			localClient.closeClient();
			this.dispose();

	}

	/** Shows the Results in a separate window
	  * @param res The result to show
	  */
	public void showResult(Result res){
				new CResultGUI(res,localClient);
				if (!(res.isEmpty()))
localClient.addResult(res);
	}
```

```
            /** Appends a Message in the Message board
              * @param msg The Message
              */
            public void addMessage(Message msg){
                        msgPanel.addMessage(msg);
            }

}
```

CMessageGUI Class

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

/** The <code>CMessageGUI</code> class represents the GUI for the Messaging
System of the CentralClient*/
public class CMessageGUI extends JPanel implements Observer, ActionListener{

            protected JTextArea mainText,msgText;
            protected JButton sendBt;
            protected JPanel msgSend,msgList;
            protected JScrollPane mainPane,msgPane;
            protected JComboBox userCb;
            protected DefaultComboBoxModel listModel;
            protected CentralClient localClient;

            /** Creates the GUI
              * @param localClient The CentralClient
              */
            public CMessageGUI(CentralClient localClient){

                        this.localClient=localClient;
                        listModel= new DefaultComboBoxModel();
                        userCb = new JComboBox(listModel);
                        (localClient.getUsers()).addObserver(this);
                        updateListModel(localClient.getUsersVector());


                        mainPane = new JScrollPane();
                        msgSend= new JPanel();
                        msgList= new JPanel();
                        msgList.setLayout(new FlowLayout());
                        msgSend.setBorder(new TitledBorder("Send Message"));
                        msgList.setBorder(new TitledBorder("Message History"));
                        // msgList
                        mainText= new JTextArea();
                        mainText.setLineWrap(true);
                        mainText.setWrapStyleWord(true);
                        mainText.setEditable(false);
                        mainPane.setPreferredSize(new Dimension (200,300));
                        mainPane.setViewportView(mainText);
                        msgList.add(mainPane);


                        // msgSend

                        GridBagLayout sendGridBag = new GridBagLayout();
```

```
            GridBagConstraints sendCons = new GridBagConstraints();
            msgSend.setLayout(sendGridBag);

            userCb.setPreferredSize(new Dimension(200,25));
            sendCons.gridx=0;
            sendCons.gridy=0;
            sendGridBag.setConstraints(userCb,sendCons);
            msgSend.add(userCb);

            sendBt = new JButton("Send");
            sendBt.addActionListener(this);

            sendCons.gridx=1;
            sendCons.gridy=0;
            sendCons.anchor=GridBagConstraints.EAST;
            sendGridBag.setConstraints(sendBt,sendCons);
            msgSend.add(sendBt);


            msgText = new JTextArea();
            msgText.setLineWrap(true);
            msgText.setWrapStyleWord(true);
            msgPane = new JScrollPane(msgText);
            msgPane.setPreferredSize(new Dimension (210,50));

            sendCons.gridx=0;
            sendCons.gridy=1;
            sendCons.gridwidth=2;
            sendGridBag.setConstraints(msgPane,sendCons);
            msgSend.add(msgPane);



            //this.setPreferredSize(new Dimension(250,450));
            this.setBorder(new EmptyBorder(10,10,10,10));

            GridBagLayout gridbag = new GridBagLayout();
            GridBagConstraints c = new GridBagConstraints();
            this.setLayout(gridbag);

            c.gridx=0;
            c.gridy=0;
            gridbag.setConstraints(msgList,c);
            this.add(msgList);

            c.gridx=0;
            c.gridy=1;
            gridbag.setConstraints(msgSend,c);
            this.add(msgSend);

}

/** Implementation of the Observer interface*/
public void update(Observable o, Object arg){
            updateListModel((Vector)arg);
            this.repaint();
            //this.setVisible(true);
}
```

```
            private void updateListModel(Vector v){
                        listModel.removeAllElements();
                        listModel.addElement("Broadcast");
                        for (int i =1; i<v.size()+1;i++){
                                    listModel.addElement(((User)v.get(i-
1)).getName());
                        }
            }

            /** Adds a message to the message board
              * @param msg The message to add
              */
            public void addMessage(Message msg){
                        mainText.append(
((localClient.getUsers()).get(msg.getSender())).getName()+" :
"+'\n'+msg.getText()+'\n'+"____________________"+'\n');
                        this.repaint();

            mainPane.getVerticalScrollBar().setValue(mainPane.getVerticalScrollBa
r().getMaximum());
            }

            /** Adds a local message to the message board
              * @param txt The message in text form
              */
            public void addLocalMsg(String txt){
                        mainText.append(localClient.getName()+" :
"+'\n'+txt+'\n'+"____________________"+'\n');
                        this.repaint();

            mainPane.getVerticalScrollBar().setValue(mainPane.getVerticalScrollBa
r().getMaximum());
            }

            /** Implementation of the ActionListener interface*/
            public void actionPerformed(ActionEvent e){
                        if (e.getSource() == sendBt){
                                    if (userCb.getSelectedIndex()!= 0){

            localClient.sendMessage(msgText.getText(),(userCb.getSelectedIndex()-
1));
                                                addLocalMsg(msgText.getText());
                                                msgText.setText("");
                                    }
                                    else{

            localClient.sendMessage(msgText.getText(),SysConst.BROADCAST);

            addLocalMsg(msgText.getText());
                                                            msgText.setText("");
                                                }

                        }
            }

}
```

2.6 Config Class

```
import java.io.*;


/**The <code>Config</code> class holds all the configuration data of the
server*/
public class Config implements Serializable {
           private int port;
           private String DBFilePath;

           /** Returns the filename and path where the file holding the database
structure is placed
             * @return The filaname and path*/
           public String getDBFilePath(){
                      return DBFilePath;
           }

           /** Sets the filename and path where the file holding the database
structure is placed
             */
           public void setDBFilePath(String filepath){
                      DBFilePath=filepath;
           }
           /** Returns the port number where the server is running
             * @return The port number*/
           public int getPort(){
                      return port;
           }
    /** Sets the port number where the server is going to run*/
           public void setPort(int port){
                      this.port=port;
           }
}
```

2.7 Coordinate Class

```
import java.io.*;

/** The <code>Coordinate</code> represents the geographical position of
  * an object in terms of coordinates*/
public class Coordinate implements Serializable{

           protected float lon,lat;

           /** Creates an instance of Coordinate
             * @param lon Longitudinal coordinate
             * @param lat Latiditinal coordinate
             */
           public Coordinate(float lon,float lat){
                      this.lon=lon;
                      this.lat=lat;
           }

           /** Returns the longitudinal coordinate
             * @return The longitudinal value
             */
           public float getLon(){
                      return lon;
           }
```

```
            /** Returns the latitudinal coordinate
              * @return The latitudinal value
              */
            public float getLat(){
                        return lat;
            }
}
```

2.8 CResultGUI Class

```
import javax.swing.*;
import javax.swing.AbstractListModel.*;
import javax.swing.JToolBar.Separator;
import java.awt.*;
import java.util.*;
import java.awt.event.*;
import javax.swing.border.*;

/** The <code>CResultGUI</code> is the window displayin the <code>Result</code>
objects
  */
public class CResultGUI extends JFrame implements Observer,ActionListener {
            private DefaultComboBoxModel listModel;
            private JComboBox usersCB;
            private JButton sendBt;
            private CentralClient localClient;

            JPanel pane;
            JScrollPane centerPane;
            JToolBar toolBar;
            JButton backBt;
            JButton forwardBt;
            JTextField countLb;
            Result rs;
            JPanel buttonPanel;
            JButton closeBt;

            int rowNum = 0;

            /** Creates and display the CResultGUI
              * @param rs The Result to show
              * @param localClient The Central Client
              */
            public CResultGUI(Result rs,CentralClient localClient){

                        super("Result Window");
                        this.localClient=localClient;

                        this.rs=rs;
                        pane = new JPanel();
                        this.getContentPane().setLayout(new BorderLayout());
                        toolBar= new JToolBar();
                        backBt = new JButton(new ImageIcon("image/Back24.gif"));
                        forwardBt= new JButton(new
ImageIcon("image/Forward24.gif"));
                        backBt.addActionListener(this);
                        forwardBt.addActionListener(this);
                        countLb= new JTextField(10);
                        countLb.setEditable(false);
```

```
                    toolBar.add(backBt);
                    toolBar.add(countLb);
                    toolBar.add(forwardBt);
                    (localClient.getUsers()).addObserver(this);
                    listModel= new DefaultComboBoxModel();
                    usersCB = new JComboBox(listModel);
                    updateListModel(localClient.getUsersVector());
                    sendBt = new JButton("Send");
                    sendBt.addActionListener(this);
                    toolBar.add(new JToolBar.Separator());
                    toolBar.add(usersCB);
                    toolBar.add(sendBt);



                    this.getContentPane().add(toolBar,BorderLayout.NORTH);
                    createGUI(0);
                    centerPane =new JScrollPane(pane);
                    this.getContentPane().add(centerPane,BorderLayout.CENTER);

                    buttonPanel=new JPanel();
                    buttonPanel.setLayout(new FlowLayout());
                    buttonPanel.setBorder(new EmptyBorder(10,0,0,0));
                    closeBt= new JButton("Close");
                    closeBt.addActionListener(this);
                    buttonPanel.add(closeBt);
                    this.getContentPane().add(buttonPanel,BorderLayout.SOUTH);


          setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                    this.pack();
                    this.setVisible(true);
          }


          private void updateListModel(Vector v){
                    listModel.removeAllElements();
                    listModel.addElement("Broadcast");
                    for (int i =1; i<v.size()+1;i++){
                              listModel.addElement(((User)v.get(i-
1)).getName());
                    }
          }

    /** Implementation of the Observer interface
      */
          public void update(Observable o, Object arg){
                    updateListModel((Vector)arg);
                    this.repaint();
                    //this.setVisible(true);
          }

          /** Implementation of the ActionListener inteface
            */
          public void actionPerformed(ActionEvent e){

                    if (e.getSource() == backBt){
                              if (rowNum != 0){
                                        rowNum--;
```

```
                                        pane.removeAll();
                                        createGUI(rowNum);
                                        pane.repaint();
                                        repaint();

                              }
                    }//if
                    else
                              if (e.getSource() == forwardBt){
                                        if (rowNum < (rs.getRows()-1)){
                                                  rowNum++;
                                                  pane.removeAll();
                                                  createGUI(rowNum);
                                                  pane.repaint();
                                                  repaint();
                                        }
                              }
                              else
                                        if (e.getSource() == closeBt){
                                                  this.dispose();
                                        }
                                        else
                                                  if(e.getSource() ==
sendBt){
                                                            Result tmpRes=
new Result();
                                                            if
((usersCB.getSelectedItem()!= null) && (rs.isEmpty()== false)){

          Vector resultRow=rs.getResultRow(rowNum);
                                                                      for
(int i=0;i<resultRow.size();i++){

          tmpRes.addResultField(0,(ResultField)resultRow.get(i));
                                                                      }
                                                                      if
(usersCB.getSelectedIndex()==0){

          localClient.sendResult(tmpRes,SysConst.BROADCAST);
                                                                      }
                                                                      else
localClient.sendResult(tmpRes,(usersCB.getSelectedIndex()-1));
                                                            }
                                                  }
          }

          /** Starts the process of automatic creation of the GUI based
            * on one of the Results
            * @param row The row index of the result to be showed from the
Result object
            */
          public void createGUI(int row){
                    JTextField text;
                    countLb.setText(" "+(row+1)+" of "+rs.getRows());
                    countLb.repaint();
                    int k=0;
                    GridBagLayout gridbag = new GridBagLayout();
                    GridBagConstraints c = new GridBagConstraints();
                    pane.setLayout(gridbag);
```

```
                    pane.setBorder(new EmptyBorder(10,10,10,10));

          pane.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);

                    if (!(rs.isEmpty())){
                              for (int i=0;i<rs.getColumns();i++){
                                        ResultField rf =
rs.getResultField(row,i);
                                        c.anchor= GridBagConstraints.WEST;
                                        c.gridx=0;
                                        c.gridy=k;
                                        JLabel label=new
JLabel(rf.getName());
                                        gridbag.setConstraints(label,c);
                                        pane.add(label);
                                        k++;
                                                  if
(rf.getType().equals(Field.BOOLEAN)){
                                                            c.anchor=
GridBagConstraints.WEST;
                                                            c.gridx=0;
                                                            c.gridy=k;
                                                            Boolean b
=(Boolean)rf.getValue();
                                                            if
(b.equals(new Boolean("true"))) text = new JTextField("Yes");
                                                            else text = new
JTextField("No");

          text.setEditable(false);

          gridbag.setConstraints(text,c);
                                                            pane.add(text);
                                                            k++;
                                                            }
                                                  else
                                                            if
(rf.getType().equals(Field.GRAPHIC)){
                                                                      int
Width=400;
                                                                      int
Height=400;

          SImage simage=(SImage)rf.getValue();

          ImageIcon image= new ImageIcon(simage.bytes);

          ScrollablePicture sp=new ScrollablePicture(image,5);

          JScrollPane jsp =new JScrollPane(sp);

          JPanel pa=new JPanel();

          pa.add(jsp);

          c.anchor= GridBagConstraints.WEST;

          c.gridx=0;
```

```
            c.gridy=k;
            gridbag.setConstraints(pa,c);
                                                                        if
(image.getIconWidth() < Width) Width=image.getIconWidth();
                                                                        if
(image.getIconHeight() < Height) Height=image.getIconHeight();
            jsp.setPreferredSize(new Dimension(Width+3,Height+3));
            pane.add(pa);
                                                                        k++;
                                                                }
                                                                else
                                                                        {
            String value;
                                                                        if
(rf.getType().equals(Field.INT)) {
            Integer integer=(Integer)rf.getValue();
            value=integer.toString();
            }
            else
                        if (rf.getType().equals(Field.FLOAT)){
                                    Float f=(Float)rf.getValue();
                                    value=f.toString();
                        }
                        else value=(String)rf.getValue();
                                                                        if
(value.length() > 30){
            JTextArea textArea = new JTextArea(value);
            textArea.setEditable(false);
            textArea.setLineWrap(true);
            JScrollPane spane = new JScrollPane(textArea);
            spane.setPreferredSize(new Dimension(250, 50));
            JPanel p=new JPanel();
            p.add(spane);
            c.anchor= GridBagConstraints.WEST;
```

```
            c.gridx=0;

            c.gridy=k;

            gridbag.setConstraints(p,c);

            pane.add(p);

            k++;


            }//if
                                                                        else
{

            text=new JTextField(value);

            text.setEditable(false);

            c.anchor= GridBagConstraints.WEST;

            c.gridx=0;

            c.gridy=k;

            gridbag.setConstraints(text,c);

            pane.add(text);

            k++;

            }//else
                                                                  }//else

                                 }//for

                                 pane.repaint();
                                 this.setVisible(true);
                        }//if
                        else{
                                  if (rs.getStatus()== Result.OK )pane.add(new
JLabel("No Mach for the query found"));
                                  else
                                                  if (rs.getStatus()==
Result.TOO_BROAD )pane.add(new JLabel("Too many results, please narrow your
search"));
                                 }
            }

}
```

2.9 CsearchPanel Class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

```
/** The <code>CSearchPanel</code> class represents the Querying part of the Main
window
  * of the Central Client application
  */
public class CSearchPanel extends JPanel implements ActionListener{

                        JButton queryBt;
                        JButton logBt,closeBt;
                        Database db;
                        CentralClient localClient;
                        CMainGUI main;

                        /** Creates and istance of CSearchPanel
                          * @param db The Database structure
                          * @param localClient The CentralClient
                          * @param main The main window
                          */
                        public CSearchPanel(Database db,CentralClient localClient,
CMainGUI main){
                                    this.main=main;
                                    this.localClient=localClient;
                                    this.db=db;
                                    this.setLayout(new FlowLayout());
                                    queryBt = new JButton("Send Query");
                                    queryBt.addActionListener(this);
                                    this.add(queryBt);

                                    logBt = new JButton("Show Result Log");
                                    logBt.addActionListener(this);
                                    this.add(logBt);

                                    closeBt = new JButton("Exit");
                                    closeBt.addActionListener(this);
                                    this.add(closeBt);

                        }

                        /** Implementation of the ActionListener interface
                          */
                        public void actionPerformed( ActionEvent e){
                                    if (e.getSource() == queryBt){
                                                if (!(db.equals(null)))
                                                            new
QueryGUI(db,localClient);
                                                else
JOptionPane.showMessageDialog(this, "The Database Structure is not loaded !");
                                    }
                                    else
                                                if (e.getSource() == logBt){
                                                            if
(localClient.getResultLog()== null){System.out.println("is null");}
                                                            else new
CResultGUI(localClient.getResultLog(),localClient);
                                                }
                                                else
                                                            if (e.getSource() ==
closeBt){
                                                                        main.close();
                                                }
```

```
                    }

}
```

2.10 CTS_Server Class

```
import java.io.*;
import java.util.*;


/** The <code>CTS_Server</code> is the class representing the server application
  */
public class CTS_Server{
           /** The main method
             */
           public static void main (String[] args){
                     Server myServer = new Server();
                     try{
                               String param= args[0];
                               if (param.equals("-lc")){
                                         try{
                                                   boolean exit=false;
                                                   int i=0;
                                                   String token;
                                                   String filename=null;
                                                   String driver=null;
                                                   String url=null;
                                                   String name=null;
                                                   String username=null;
                                                   String password=null;
                                                   BufferedReader Bfile =
new BufferedReader(new FileReader("ServConf.ini"));
                                                   do{
                                                             String line =
Bfile.readLine();
                                                             StringTokenizer
ST = new StringTokenizer(line);
                                                             if
(ST.hasMoreTokens()){
                                                                       i++;

           token= ST.nextToken();
                                                                       if
(token.equals("filename:")){ filename=ST.nextToken();}
                                                                       else

            if (token.equals("dbDriver:")){ driver=ST.nextToken();}

            else

                      if (token.equals("dbURL:")){ url=ST.nextToken();}

                      else

                                if (token.equals("dbName:")){
name=ST.nextToken();}

                                else
```

```
                                                if (token.equals("username:")){

                                                                        if
(ST.hasMoreTokens()) username=ST.nextToken();

                                                                        else
username="";

                                                                }

                                                else

                                                                if
(token.equals("password:")){

                                                                                                if
(ST.hasMoreTokens()) password=ST.nextToken();


else password="";

                                                                        }

                                                                                        if
(i==6) exit = true;

                                                                    }
                                                                    else {

System.out.println("The ServConf.ini contains errors ...");

exit = true;
                                                                                                }
                                                                  }//do
                                                                  while (exit==false);

                                                                  if (i==6)
myServer.run(true,filename,name,url,username,password,driver);

                                                              }//try
                                                catch(FileNotFoundException
e){System.out.println("Exception at Server "+e.toString());}
                                                catch(IOException
e){System.out.println("Exception at Server "+e.toString());}
                                                catch(NoSuchElementException
e){System.out.println("The ServConf.ini contains errors ...");}

                                    }
                                    else{System.out.println(" use '-lc' to load a
new configuration ...");}
                                    }
                            catch(IndexOutOfBoundsException
e){myServer.run(false,null,null,null,null,null,null);}

                }


}
```

2.11 Database Class

```
import java.util.*;
import java.io.*;


/** The <code>Database</code> class represents the database connected to the
system
  */
public class Database implements Serializable{
                      /** The Name of the Database - used as Data Source Name
for the ODBC driver*/
                      protected String name;
                      /** The Object describing the table used in the database.
For this demo
                        * version only one table is accepted from the system*/
                      protected Table DBTable;


                      protected Database(){
                      }

                      /** Creates a Database Object with the specific path and
no username or password*/
                      public Database(String name){
                                this.name=name;
                      }

                      /** This function returns the Table used in the database
                        * @return The Table
                        */
                      public Table getTable(){
                                return DBTable;
                      }


        /** Compares two Database Objects
          * @return true false
          */
           public boolean equals(Object O){
                                if (O instanceof Database){
                                          Database d=(Database)O;
                                                    if
(DBTable.equals(d.getTable()))
                                                              if
(name.equals(d.getName())) return true;
                                          return false;
                                }
                                else return false;
                      }


                      /** Sets the Database table
                        * @param DBTable The table object*/
                      public void setTable(Table DBTable){
                                this.DBTable=DBTable;
```

```
                }
        /** Set the Database name
          * @param name The Database name*/
                public void setName(String name){
                        this.name=name;
                }

                /** Returns the Database name
                  * @return The Database name*/
                public String getName(){
                        return name;
                }


}
```

2.12 DBServerCommunication Interface

```
/**
 * The <code>DBServerCommunication</code> is an interface that defines the
operations
 * that can be performed in relation with the DB Server
 * @version 0.1
 */

public interface DBServerCommunication{
           /** Stores the Database path in an internal variable*/
           public void setDBPath(Object path);
           /** Stores the Database username in an internal variable*/
           public void setUsername(Object username);
           /** Stores the Database password in an internal variable*/
           public void setPassword(Object password);
           /** Connects to the Database using the Database path, username and
password stored
             * in the internal variables
             * @see #setDBPath(Object path)
             * @see #setUsername(Object username)
             * @see #setPassword(Object password)
             * @return The Database Connection
             */
           public void connectToDB() throws Exception;
           /** Disconnects from the database.*/
           public void disconnectFromDB() throws Exception;
           /** Sends a query to the database.
            * @param query The Query Object
            * @return The result Object
            */
           public Object sendQuery(Object query) throws Exception;

           /** Returns the Database path */
           public Object getDBPath();
           /** Returns the Database username*/
           public Object getUsername();
           /** Returns the Database password*/
           public Object getPassword();
           /** Translates the Query object in an object usable for the DBMS
```

```
             * @return The translated query
             */
            public Object translateQuery(Object query);

}
```

2.13 EBConnection Class

```
import java.sql.*;
import java.io.*;
/**
 * This class implements the DBServerComminication interface and is specially
 * designed for ODBC drivers compatible Databases. It Implies direct connection
 * to the database through LAN.
*/

public class EBConnection implements DBServerCommunication, Serializable{

                        /** The database path variable*/
                        private String DBPath;
                        /** The database Username variable*/
                        private String Username;
                        /** The database Password variable*/
                        private String Password;
                        /** The Object holding the connection to the database*/
                        private Connection DBConnection;

                        private String driverClass;

                        /* The Object used to run queries*/
                        //private Statement Query;*/

                        /** Creates a database connection object with a specific
jdbc driver
                          * @param driver The jdbc driver
                          */
                        public EBConnection(String driver){
                                  driverClass=driver;
                        }

                        /** Stores the Database path in an internal variable*/
                        public void setDBPath(Object path){
                                  if (path instanceof String){
                                            this.DBPath = (String)path;
                                  }
                        };

                        /** Returns the Database path
                          * @return The Database path
                          */
                        public Object getDBPath(){
                                  return DBPath;
                        }
                        /** Returnd the Database Username
                          * @return The Database Username
                          */
                        public Object getUsername(){
```

```
                        return Username;
                }
                /** Returns the Database Password
                  * @return The Database password
                  */
                public Object getPassword(){
                        return Password;
                }
                /** Sets the driver class
                  * @param The driver class
                  */
                public void setDriver(String driver){
                        driverClass=driver;
                }


                /** Translates the Query object in an object usable for
the DBMS
                  * @return The SQL equivalent of the query, as a String
                  */
                public Object translateQuery(Object query){
                        Query q=(Query)query;
                        StringBuffer SQL= new StringBuffer("SELECT *
FROM ");
                        SQL.append(q.getTableName()+" ");
                        SQL.append("WHERE ");
                        for (int i=0;i<q.getFieldNo();i++){
                                if (i>0) SQL.append(" AND ");

                                QueryField f=q.getField(i);
                                SQL.append(f.getName()+"
"+f.getCriteria()+" ");

                                if (f.getType().equals(Field.INT)){
                                        Integer integer
=(Integer)f.getValue();
                                        if
(f.getCriteria().equals("LIKE")){

          SQL.append("'%"+integer.toString()+"%' ");
                                        }
                                        else
SQL.append(integer.toString()+" ");

                                        }
                                        else
                                                if
(f.getType().equals(Field.FLOAT)) {

          Float floatnum =(Float)f.getValue();
                                                                        if
(f.getCriteria().equals("LIKE")){

          SQL.append("'%"+floatnum.toString()+"%' ");
                                                                        }
                                                                        else
SQL.append(floatnum.toString()+" ");

                                                                        }
```

```
                                                                    else
            if (f.getType().equals(Field.BOOLEAN)){

                        SQL.append((Boolean)f.getValue());

                        SQL.append(" ");

                        }

                        else{

                                    if (f.getCriteria().equals("LIKE")){

                                                SQL.append("'%"+f.getValue()+"%' ");

                                    }

                                    else SQL.append("'"+f.getValue()+"' ");


                                    }


                                    }

                                    return new String(SQL);
                        }

                        /** Stores the Database username in an internal variable*/
                        public void setUsername(Object username){

                                    if (username instanceof String){
                                                this.Username = (String)username;
                                    }

                        };

                        /** Stores the Database password in an internal variable*/
                        public void setPassword(Object password){

                                    if (password instanceof String){
                                                this.Password = (String)password;
                                    }

                        };

                        /** Connects to the Database using the Database path,
username and password stored
                          * in the internal variables
                          * @see #setDBPath(Object path)
                          * @see #setUsername(Object username)
                          * @see #setPassword(Object password)
                          * @return The Database Connection
                          */
                        public void connectToDB() throws Exception {
                                    try{
                                                Class.forName(driverClass);
```

```
                                        if
((!Username.equals(""))&&(!Password.equals("")))DBConnection =
DriverManager.getConnection(DBPath,Username,Password);
                                        else DBConnection =
DriverManager.getConnection(DBPath);
                                        DBConnection.setReadOnly(true);
                                        }
                                catch (SQLException e){throw new
SQLException(e.toString()+ "*connection");}
                        };

                        /** Disconnects from the database.*/
                        public void disconnectFromDB() throws Exception {
                                try{
                                        DBConnection.close();
                                        }
                                catch (SQLException e){throw new
SQLException(e.toString()+"*disconnect");}
                        };

                        /** Sends a query to the database.
                         * @param query The Query Object.
                         * @return The result Object
                         */
                        public Object sendQuery(Object query) throws Exception {
                                ResultSet rs = null;
                                if (query instanceof Query){
                                        String myQuery =
(String)translateQuery((Query)query);
                                        Statement stmt =
DBConnection.createStatement();
                                        try{
                                                stmt.execute("USE
"+((Query)query).getDBName());
                                                rs =
stmt.executeQuery(myQuery);
                                                }
                                        catch(SQLException e){throw new
SQLException(e.toString()+"*SQL");}

                                }//end if
                                return rs;
                        }

}
```

2.14 EBCSCommunication Class

```
import java.io.*;
import javax.swing.*;

/** The <code>EBCSCommunication</code> class describes the communication
  * from the Client to the Server
  */
public class EBCSCommunication implements ClientServerCommunication{

           /** Receives the Action that is received by the Server
             * @param input_connection The object describing the input
connection from the Server
```

```
             * @return The Action
             */
            public Object readAction(Object input_conncetion) throws Exception{
                        int action=0;
                        try{
                                    ObjectInputStream
input=(ObjectInputStream)input_conncetion;
                                    action=input.readInt();
                           }
                        catch(Exception e){throw e;}
                        return new Integer(action);
            }

            /** Sends a Query
              * @param query The Query object
              * @param output_connection The object describing the output
connection to the Server
              */
            public void sendQuery(Object query, Object output_connection)throws
Exception{
                        ObjectOutputStream
output=(ObjectOutputStream)output_connection;
                        try{
                                    output.writeInt(Action.SEND_QUERY);
                                    output.writeObject(query);
                                    output.flush();
                                    }
                        catch(Exception e){throw e;}
            }

            /** Sends a Result
              * @param result The Result object
              * @param output_connection The object describing the output
connection to the Server
              * @param clientID The ID of the receiving Client
              */
            public void sendResult(Object result,Object output_connection,Object
clientID)throws Exception{

                        ObjectOutputStream
output=(ObjectOutputStream)output_connection;
                        Result res=(Result)result;
                                    try{

            output.writeInt(Action.SEND_RESULT);

            output.writeObject(clientID);
                                                output.flush();
                                                if (res.getRows()!=0){
                                                            for(int i =0;
i<res.getRows();i++){

            for(int k=0; k<res.getColumns();k++){

            ResultField RF = res.getResultField(i,k);

            if (RF.getType().equals(Field.GRAPHIC)){

                        SImage image=(SImage)RF.getValue();
```

```
                        output.writeObject(new Integer(SysConst.IMAGE_ON_STREAM));

                        output.writeObject(RF.getName());

                        output.writeInt(image.bytes.length);

                        output.flush();

                        BufferedOutputStream BOS =new
BufferedOutputStream((OutputStream)output_connection);

                        BOS.write(image.bytes,0,image.bytes.length);

                        BOS.flush();

            }

            else{

                                output.writeObject(RF);

                                output.flush();

                        }
                                                                                }

            output.writeObject(new Integer(SysConst.END_ROW));

            output.flush();
                                                                }//for

            output.writeObject(new Integer(SysConst.END_TRANSACTION));
                                                                output.flush();
                                                        }//if
                                                        else{

            output.writeObject(result);
                                                                output.flush();
                                                        }//else
                                        }
                                catch(Exception e){throw e;}
            }


            /** Returns a recived Result
              * @param input_connection The object describing the input
connection from the Server
              * @return The received Result
              */
            public Object receiveResult(Object input_connection)throws Exception{
                        try{
                                            JProgressBar progress= new
JProgressBar();
                                            progress.setIndeterminate(true);
                                            progress.setString("Receiving data
...");
                                            progress.setStringPainted(true);
```

```
                                        JFrame message= new JFrame("Result
Download");

           message.getContentPane().add(progress);
                                        message.pack();
                                        message.setVisible(true);

                                        int prog=0;
                                        Result r= new Result();
                                        ObjectInputStream input
=(ObjectInputStream)input_connection;
                                        Object DBID =input.readObject();
                                        Object data = input.readObject();
                                        if (data instanceof Result){
                                                  r=(Result)data;
                                        }//if
                                        else{
                                                   boolean exit = false;
                                                   int i=0;
                                                   while (exit==false){
                                                             if (data
instanceof ResultField){

           r.addResultField(i,(ResultField)data);

           prog++; progress.setValue(prog);message.repaint();
                                                   }
                                                   if (data
instanceof Integer){
                                                                        int
action =((Integer)data).intValue();

           switch (action){

           case SysConst.END_ROW : {

                                        i++;

                                        prog++;
progress.setValue(prog);message.repaint();

                                        };break;


           case SysConst.END_TRANSACTION : {

                                        exit=true;

                                        prog++;
progress.setValue(prog);message.repaint();

                                        };break;


           case SysConst.IMAGE_ON_STREAM : {

                                        String
name=(String)input.readObject();
```

```
                                                prog++;
progress.setValue(prog);message.repaint();

                                                int size =input.readInt();

                                                BufferedInputStream BIS = new
BufferedInputStream((InputStream)input_connection);

                                                int res=0;

                                                int count=0;

                                                byte[] bytes = new byte[size];

                                                //res=BIS.read(bytes,0,size);

                                                for(int n = 0; n<size; n++){

                                                           res=BIS.read();

                                                           bytes[n]=(new
Integer(res)).byteValue();

                                                           prog++;
progress.setValue(prog);message.repaint();

                                                }

                                                SImage image = new SImage();

                                                image.bytes=bytes;

                                                ResultField myRF = new
ResultField(name,Field.GRAPHIC,image);

                                                r.addResultField(i,myRF);

                                                prog++;
progress.setValue(prog);message.repaint();

                                                };break;
                                                                                    }
                                                                        }//if
                                                                        if (exit!=
true) data=input.readObject();
                                                                    }//while
                                                }//else
                                                message.dispose();
                                                return r;
                                    }
                          catch(Exception e){throw e;}
          }

          /** Receives a Database Structure
            * @param input_connection The object describing the input
connection from the Server
            * @return The Database Structure
            */
```

```
	public Object receiveDBStructure(Object input_connection)throws
Exception{
			try{
				ObjectInputStream input =
(ObjectInputStream)input_connection;
				return input.readObject();
			}
			catch(Exception e){System.out.println("Exception at
ClientThread 5:"+e);}
			return null;
	}

	/** Receives User's Info
	  * @param input_connection The object describing the input
connection from the Server
	  * @return The User's info
	  */
	public Object receiveUserInfo(Object input_connection)throws
Exception{
			ObjectInputStream input =
(ObjectInputStream)input_connection;
			User user=new User();
			try{
				user =(User)input.readObject();
				return user;
				}
			catch(Exception e){throw e;}
	}

	/** Sends a request of temination of the connection to the server
	  * @param output_connection The object describing the output
connection to the Server
	  */
	public void terminateConnection(Object output_connection)throws
Exception{
			try{
				ObjectOutputStream
output=(ObjectOutputStream)output_connection;
				output.writeInt(Action.SHUTDOWN);
				output.flush();
				}
			catch(Exception e){throw e;}
	}

	/** Sends the User Info to the Server
	  * @param info The User's Info
	  * @param output_connection The object describing the output
connection to the Server
	  */
	public void sendInfo(Object info,Object output_connection) throws
Exception{
			try{
				ObjectOutputStream
output=(ObjectOutputStream)output_connection;
				output.writeInt(Action.SEND_INFO);
				output.writeObject(((Info)info).clientName);
				output.writeInt(((Info)info).clientType);
				output.flush();
				}
```

```
                    catch(Exception e){throw e;}
          }
          //*********Messaging System********


          /** Receives a message from the Server
            * @param input_connection The object describing the input
connection from the Server
            * @return The received Message
            */
          public Object receive_message(Object input_connection) throws
Exception{
                    try{
                              ObjectInputStream
input=(ObjectInputStream)input_connection;
                              return input.readObject();
                              }
                    catch(Exception e){throw e;}
          }

          /** Sends a Message to the Server
            * @param output_connection The object describing the output
connection to the Server
            */
          public void send_message(Object message,Object output_connection)
throws Exception{
                    try{
                              Message msg=(Message)message;
                              ObjectOutputStream
out=(ObjectOutputStream)output_connection;

                              out.writeInt(Action.SEND_MESSAGE);
                              out.writeObject(msg);
                              out.flush();

                              }
                    catch(Exception e){throw e;}

          }

          /** Receives a GIS Position from the Server
            * @param input_connection The object describing the input
connection from the Server
            * @return The GIS Position
            */
          public Object receive_Position(Object input_connection) throws
Exception{
                    try{
                              ObjectInputStream input=(ObjectInputStream)
input_connection;
                              return input.readObject();
                              }
                    catch(Exception e){throw e;}
          }

          /** Sends a GIS Position to the Server
            * @param Position The GIS Position
            * @param output_connection The object describing the output
connection to the Server
```

```
            */
           public void send_Position(Object Position,Object output_connection)
throws Exception{
                       try{
                                  GISObject GISO = (GISObject)Position;
                                  ObjectOutputStream output=(ObjectOutputStream)
output_connection;
                                  output.writeInt(Action.SEND_POSITION);
                                  output.writeObject(GISO);
                                  output.flush();
                                  }
                       catch(Exception e){throw e;}

           }




}
```

2.15 EBSCCommunication Class

```
import java.io.*;
import javax.swing.*;

/** The <code>EBSCCommunication</code> class describes the communication
  * from the Server to the Client
  */
public class EBSCCommunication implements ServerClientCommunication{


           /** The constructor
             */
           public EBSCCommunication(){
           }

           /** Sends a result to a Client
             * @param result The Result to be sent
             * @param DBID The used database ID (included for future development
of the demo)
             * @param output_connection The object describing the output
connection to the client
             */
           public synchronized void sendResult(Object result, Object DBID,
Object output_connection) throws Exception{
                       ObjectOutputStream
output=(ObjectOutputStream)output_connection;
                       Result res=(Result)result;
                                  try{

           output.writeInt(Action.SEND_RESULT);
                                                      output.writeObject(DBID);
                                                      output.flush();
                                                      if (res.getRows()!=0){
                                                                 for(int i =0;
i<res.getRows();i++){

           for(int k=0; k<res.getColumns();k++){
```

```
            ResultField RF = res.getResultField(i,k);

            if (RF.getType().equals(Field.GRAPHIC)){

                        SImage image=(SImage)RF.getValue();

                        output.writeObject(new Integer(SysConst.IMAGE_ON_STREAM));

                        output.writeObject(RF.getName());

                        output.writeInt(image.bytes.length);

                        output.flush();

                        BufferedOutputStream BOS =new
BufferedOutputStream((OutputStream)output_connection);

                        BOS.write(image.bytes,0,image.bytes.length);

                        BOS.flush();

            }

            else{

                                    output.writeObject(RF);

                                    output.flush();

                        }
                                                                                    }

            output.writeObject(new Integer(SysConst.END_ROW));

            output.flush();
                                                                    }//for

            output.writeObject(new Integer(SysConst.END_TRANSACTION));
                                                                    output.flush();
                                                            }//if
                                                            else{

            output.writeObject(result);
                                                                    output.flush();
                                                            }//else
                                                }
                                    catch(Exception e){throw e;}
            }

            /** Sends the database structure to a Client
              * @param DBStructure The database structure to be sent
              * @param output_connection The object describing the output
connection to the client
              */
            public void sendDBStructure(Object DBStructure, Object
output_connection) throws Exception{
                        ObjectOutputStream output =
(ObjectOutputStream)output_connection;
```

```
                    try{
                                output.writeInt(Action.SEND_DBSTRUCTURE);
                                output.writeObject(DBStructure);
                                output.flush();
                                }
                    catch(Exception e){throw e;}
          }

          /** Receives a Query from one of the Clients
            * @param input_connection The object describing the input
connection from the client
            * @param The Query
            */
          public Object receiveQuery(Object input_connection) throws Exception{
                    try{
                                ObjectInputStream input=
(ObjectInputStream)input_connection;
                                Query query = (Query)input.readObject();
                                return query;
                                }
                    catch(Exception e){throw e;}
          }

          /** Notifys to a CentralClient that a Mobile Client has logged in
            * @param user The User that has logged in
            * @param output_connection The object describing the output
connection to the client
            */
          public void sendUserLOGIN(Object user, Object output_connection)
throws Exception{
                    try{
                                ObjectOutputStream output
=(ObjectOutputStream)output_connection;
                                output.writeInt(Action.SEND_USERLOGIN);
                                output.writeObject(user);
                                output.flush();
                                }
                    catch(Exception e){throw e;}
          }

          /** Notifys to a CentralClient that a Mobile Client has logged out
            * @param user The User that has logged out
            * @param output_connection The object describing the output
connection to the client
            */
          public void sendUserLOGOUT(Object user, Object output_connection)
throws Exception{
                    try{
                                ObjectOutputStream output
=(ObjectOutputStream)output_connection;
                                output.writeInt(Action.SEND_USERLOGOUT);
                                output.writeObject(user);
                                output.flush();
                                }
                    catch(Exception e){throw e;}
          }

          /** Receives the Action performad by the Client
```

```
             * @param input_connection The object describing the input
connection from the client
             * @param The Action
             */
            public Object readAction(Object input_conncetion) throws Exception{
                        int action=0;
                        try{
                                    ObjectInputStream
input=(ObjectInputStream)input_conncetion;
                                    action=input.readInt();
                           }
                        catch(Exception e){throw e;}
                        return new Integer(action);
            }

            /** Receives a Result from one of the Central Clients
             * @param input_connection The object describing the input
connection from the client
             * @param The Result
             */
            public Object receiveResult(Object input_connection) throws
Exception{
                        try{
                                                ResultInfo RI= new ResultInfo();

                                                Result r= new Result();
                                                ObjectInputStream input
=(ObjectInputStream)input_connection;
                                                Object clientID =input.readObject();
                                                Object data = input.readObject();
                                                if (data instanceof Result){
                                                            r=(Result)data;
                                                }//if
                                                else{
                                                            boolean exit = false;
                                                            int i=0;
                                                            while (exit==false){
                                                                        if (data
instanceof ResultField)

            r.addResultField(i,(ResultField)data);
                                                                        if (data
instanceof Integer){
                                                                                    int
action =((Integer)data).intValue();

            switch (action){

            case SysConst.END_ROW : {

                                                i++;

                                                };break;


            case SysConst.END_TRANSACTION : {

                                                exit=true;
```

```
                                        };break;

          case SysConst.IMAGE_ON_STREAM : {
                                        String
name=(String)input.readObject();
                                        int size =input.readInt();
                                        BufferedInputStream BIS = new
BufferedInputStream((InputStream)input_connection);
                                        int res=0;
                                        int count=0;
                                        byte[] bytes = new byte[size];
                                        //res=BIS.read(bytes,0,size);
                                        for(int n = 0; n<size; n++){
                                                res=BIS.read();
                                                bytes[n]=(new
Integer(res)).byteValue();
                                        }
                                        SImage image = new SImage();
                                        image.bytes=bytes;
                                        ResultField myRF = new
ResultField(name,Field.GRAPHIC,image);
                                        r.addResultField(i,myRF);
                                        };break;
                                                                        }
                                                }//if
                                                if (exit!=
true) data=input.readObject();
                                                }//while
                                        }//else
                                        RI.CCID=(Integer)clientID;
                                        RI.result=r;
                                        return RI;
                                }
                        catch(Exception e){throw e;}
          }

          /** Receives an Info object from one of the Clients
            * @param input_connection The object describing the input
connection from the client
            * @param The Info about the client
            */
```

```
            public Object receiveInfo(Object input_connection) throws Exception{
                        Info info=new Info();
                        try{
                                    ObjectInputStream
input=(ObjectInputStream)input_connection;
                                    info.clientName =(String)input.readObject();
                                    info.clientType = input.readInt();
                                    }
                        catch(Exception e){throw e;}
                        return info;
            }

            //*****Messaging System**********

            /** Receives a Message from one of the Clients
              * @param input_connection The object describing the input
connection from the client
              * @param The Message
              */
            public Object receive_message(Object input_connection) throws
Exception{
                        try{
                                    ObjectInputStream
input=(ObjectInputStream)input_connection;
                                    return input.readObject();
                                    }
                        catch(Exception e){throw e;}
            }

            /** Sends a Message to one of the Clients
              * @param message The Message to deliver
              * @param output_connection The object describing the output
connection to the client
              */
            public synchronized void send_message(Object message,Object
output_connection) throws Exception{
                        try{
                                    Message msg=(Message)message;
                                    ObjectOutputStream
out=(ObjectOutputStream)output_connection;

                                    out.writeInt(Action.SEND_MESSAGE);
                                    out.writeObject(msg);
                                    out.flush();

                                    }
                        catch(Exception e){throw e;}

            }

            /** Receives a GIS Position from one of the Clients
              * @param input_connection The object describing the input
connection from the client
              * @param The GIS Position
              */
            public Object receive_Position(Object input_connection) throws
Exception{
                        try{
```

```
                                    ObjectInputStream input=(ObjectInputStream)
input_connection;
                                    return input.readObject();
                                    }
                        catch(Exception e){throw e;}
            }

            /** Sends a GIS Position to one of the Clients
              * @param Position The GIS Position
              * @param output_connection The object describing the output
connection to the client
              */
            public synchronized void send_Position(Object Position,Object
output_connection) throws Exception{
                        try{
                                    GISObject GISO = (GISObject)Position;
                                    ObjectOutputStream output=(ObjectOutputStream)
output_connection;
                                    output.writeInt(Action.SEND_POSITION);
                                    output.writeObject(GISO);
                                    output.flush();
                                    }
                        catch(Exception e){throw e;}

            }


}
```

2.16 Field Class

```
import java.io.*;
/** The <code>Field</code> class holds the information of the fields
  * of a <code>Table</code>.
  */
public class Field implements Serializable{
            /** Constant indicating a TEXT field type*/
            public static final String TEXT = "TEXT";

            /** Constant indicating a GRAPHIC field type*/
            public static final String GRAPHIC = "GRAPHIC";

            /** Constant indicating a INT field type*/
            public static final String INT   = "INT";

            /** Constant indicating a FLOAT field type*/
            public static final String FLOAT = "FLOAT";

            /** Constant indicating a BOOLEAN field type*/
            public static final String BOOLEAN = "BOOLEAN";


            /** The field name variable*/
            private String name;
            /** The field type variable*/
            private String type;
```

```
            /** The field searcheable variable*/
            private boolean searcheable;

            private int size;


            private Field(){}
    /** Creates a Field object with a specific name, type and seracheable
option*/
            public Field(String name, String type, boolean searcheable,int size){
                        this.name=name;
                        this.type=type;
                        this.searcheable=searcheable;
                        this.size=size;
            }

    /** Compares two Field Objects
      * @param obj The Object to compare
      * @return True False
      */
    public boolean equals(Object obj){
                        if (obj instanceof Field){
                                    Field f=(Field)obj;
                                    if (name.equals(f.getName()))
                                                if (type.equals(f.getType()))
                                                            if
(searcheable==f.getSearcheable())
                                                                        if
(size==f.getSize())return true;
                                    return false;

                        }
                        else return false;
            }

            /** Returns the field name
              * @return the Field name
              */
            public String getName(){
                        return name;
            }

            /** Returns the type of the field. A list of valid type is described
              * in the class constants
              * @return The Field type
              */
            public String getType(){
                        return type;
            }

            /** Returns if the field can be used as a search value or not
              * @return Field searcheability*/
            public boolean getSearcheable(){
                        return searcheable;
            }

            public int getSize(){
                        return size;
            }
```

```
    /** Check that the String passed is a valid field TYPE
      * @return true false
      */
    static boolean checkType(String type){
                                if (type.equals(Field.TEXT)) return true;
                                if (type.equals(Field.INT)) return true;
                                if (type.equals(Field.FLOAT)) return true;
                                if (type.equals(Field.BOOLEAN)) return true;
                                if (type.equals(Field.GRAPHIC)) return true;
                                return false;
          }


}
```

2.17 GIS Class

```
import java.sql.*;

/** The <code>GIS</code> class represents the GIS Software and describes the
  * functionalities connected to it
  */
public class GIS implements GISCommunication{
          protected Connection dbConnection;
          protected Statement stmt;

          /** Start of the communication
            */
          public void connect() throws Exception{
          try{
                    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
          dbConnection = DriverManager.getConnection("jdbc:odbc:GIS_Object");
          stmt=dbConnection.createStatement();
                                }
                    catch(Exception e){throw e;}

          }

          /** End of the communication
            */
          public void disconnect() throws Exception{
                    try{
                                dbConnection.close();
                                }
                    catch(Exception e){throw e;}
          }

          /** Updating the value of an Object in the GIS
            * @param obj The Object in the GIS
            */
          public void updateObject(Object obj) throws Exception{
                    try{
                                GISObject GISO=(GISObject)obj;

                                String update = new String( "UPDATE Objektai "+

                                "SET LON = "+GISO.coordinate.getLon()+" , LAT =
"+GISO.coordinate.getLat()+
```

```
                                    " WHERE NAME = '"+GISO.name+"'");
                                    stmt.executeUpdate(update);
                                    }
                        catch(Exception e){throw e;}

            }

            /** Delete and Object form the GIS
              * @param obj The Object in the GIS
              */
            public void deleteObject(Object obj) throws Exception{
                        try{
                                    String update;
                                    GISObject GISO=(GISObject)obj;
                                    if (GISO.name.equals("*")){
                                                update= new String("DELETE
Objektai");
                                    }
                                    else update = new String( "DELETE FROM Objektai
WHERE NAME = '"+GISO.name+"'");

                                    stmt.executeUpdate(update);

                                    }
                        catch(Exception e){throw e;}

            }

            /** Create and Object in the GIS
              * @param obj The Object in the GIS
              */
            public void createObject(Object obj) throws Exception{
                        try{
                                    GISObject GISO=(GISObject)obj;

                                    String update = new String( "INSERT INTO
Objektai "+

                                    "VALUES
('"+GISO.name+"',"+GISO.coordinate.getLon()+","+GISO.coordinate.getLat()+")");
                                    stmt.executeUpdate(update);

                                    }
                        catch(Exception e){throw e;}

            }

}
```

2.18 GISCommunication Interface

```
/** The <code>GISCommunication</code> interface represents the GIS Software and
describes the
  * functionalities connected to it
  */
public interface GISCommunication{
```

```
            /** Start of the communication
              */
            public void connect() throws Exception;

            /** End of the communication
              */
            public void disconnect() throws Exception;

            /** Updating the value of an Object in the GIS
              * @param obj The Object in the GIS
              */
            public void updateObject(Object obj) throws Exception;

            /** Delete and Object form the GIS
              * @param obj The Object in the GIS
              */
            public void deleteObject(Object obj) throws Exception;

            /** Create and Object in the GIS
              * @param obj The Object in the GIS
              */
            public void createObject(Object obj) throws Exception;

}
```

GISObject Class

```
import java.io.*;

/** The <code>GISObject</code> class represents and object in the GIS system
  */
public class GISObject implements Serializable {
            /** The Object name*/
            public String name;
            /** The Object coordinate*/
            public Coordinate coordinate;

            /** Creates an instanec of the GISObject
              * @param name The object's name
              * @param c The object's coordinate*/
            public GISObject(String name, Coordinate c){
                        this.name=name;
                        coordinate = c;
            }
}
```

2.19 GPSCommunication Interface

```
/** The <code>GPSCommunication</code> interface represents the communication
with the GPS devide
  */
public interface GPSCommunication{
            /** Returns the coordinate read in the GPS device
              * @return The coordinate
              */
            public Coordinate getCoordinate();
}
```

2.20 GPSDevice Class

```
/** The <code>GPSDevice</code> class represents the GPS device
  * Note: Used as a simulation device in the demo version
  */
public class GPSDevice extends Thread implements GPSCommunication{
          protected float refLon, lon, refLat, lat;
          protected int state=0;
          protected MobileClient localClient;
          protected boolean exit=false;


          /** Creates an instance of the GPSDevice with starting coordinates
            * @param refLon The longitudinal coordinate
            * @param refLat The latitudinal coordinate
            * @param localClient The Mobile Client
            */
          public GPSDevice(float refLon,float refLat,MobileClient localClient){
                    this.localClient=localClient;
                    this.refLon=refLon;
                    this.refLat=refLat;
                    lon=refLon;
                    lat=refLat;
          }


          /** Returns the coordinate read in the GPS device
            * @return The coordinate
            */
          public Coordinate getCoordinate(){

                    // Algorithm created by "Sidabrinis Tinklas"

                    if (state == 0)
                    {
                              //varom is pradinio taso i desine
                              if (lon < refLon + 0.05) lon += 0.001;
                              else state = 1;
                    }
                    else if (state == 1)
                    {
                              // griztame is desines i pradzia
                              if (lon > refLon) lon -= 0.001;
                              else state = 2;
                    }
                    else if (state == 2)
                    {
                              // varome is refpointo i virsu
                              if (lat < refLat + 0.02) lat += 0.001;
                              else state = 3;
                    }
                    else
                    {
                              // varome is virsaus i refpointa
                              if (lat > refLat) lat -= 0.001;
                              else state = 0;
                    }

                    return new Coordinate(lon,lat);
```

```
            }

            /** Stops the running thread
              */
            public void halt(){
                        exit=true;
            }

            /** Starts the running thread
              */
            public void run(){
                        while(!exit){

            localClient.setCoordinate(this.getCoordinate());
                                    try{
                                                sleep(2000);
                                                }
                                    catch(InterruptedException e){}
                        }
            }
}
```

2.21 Info Class

```
/** The <code>Info</code> class holds information of the client
  */
public class Info{
            /** The typr of client
              * @see SysConst
              */
            public int clientType;

            /** The client name*/
            public String clientName;
}
```

2.22 InvalidStructureException Class

```
/** The <code>InvalidStructureException</code> is thrown when the file
  * holding the database structure describes an invalid structure*/
public class InvalidStructureException extends Exception{
            public InvalidStructureException(String file){
                        super("The structure in -"+file+"- is not a valid
structure");
            }
}
```

2.23 MainGUI Class

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

/** The <code>MainGUI</code> represents the main window of the Mobile Client
  */
public class MainGUI extends JFrame {
```

```
            Database db;
            MobileClient localClient;
            SearchPanel searchPanel;
            JToolBar statusBar;
            JLabel statusLb;
            MessageGUI msgPanel;


            /** Creates an instance of MainGUI
              * @param db The Database structure
              * @param localClient The MobileClient
              */
            public MainGUI(Database db,MobileClient localClient){
                        super("Car Tracking System");
                        this.localClient=localClient;
                        this.db=db;
                        searchPanel= new SearchPanel(db,localClient,this);
                        statusBar= new JToolBar();
                        statusLb= new JLabel("Connected");
                        statusLb.setBorder(new
SoftBevelBorder(BevelBorder.LOWERED));
                        statusBar.setFloatable(false);
                        statusBar.setBorder(new
SoftBevelBorder(BevelBorder.RAISED));
                        statusBar.add(statusLb);
                        msgPanel= new MessageGUI(localClient);
                        this.getContentPane().setLayout(new BorderLayout());
                        this.getContentPane().add(searchPanel,BorderLayout.NORTH);
                        this.getContentPane().add(statusBar,BorderLayout.SOUTH);
                        this.getContentPane().add(msgPanel,BorderLayout.CENTER);
                        this.addWindowListener(new WindowAdapter(){
                                    public void windowClosing(WindowEvent e){
                                                MainGUI
frame=(MainGUI)e.getWindow();
                                                frame.close();
                                    }
                        });

                        pack();
                        setVisible(true);

            }


            /** Closes the window and the application*/
            public void close(){
                        localClient.closeClient();
                        this.dispose();

            }

            /** Shows the Results in a separate window
              * @param res The result to show
              */
            public void showResult(Result res){
                                    new ResultGUI(res,localClient);
                                    if (!(res.isEmpty()))
localClient.addResult(res);
            }
```

```
            /** Appends a Message in the Message board
              * @param msg The Message
              */
            public void addMessage(Message msg){
                        msgPanel.addMessage(msg);
            }


}
```

2.24 Message Class

```
import java.io.*;

/** The <code>Message</code> class represents a message in the messaging
system*/
public class Message implements Serializable{
            protected String text;
            protected int sender;
            protected int receiver;

            /** Creates a Message with a specific text and with no receiver and
sender
              * @param Text The text message
              */
            public Message(String Text){
                        this.text=Text;
                        sender=-1;
                        receiver=SysConst.NO_RECEIVER;
            }

            /** Creates a Message with a specific text and sender , and with no
receiver
              * @param Text The text message
              * @param sender The sender Client ID
              */
            public Message(String text, int sender){
                        this(text);
                        this.sender=sender;
            }

            /** Creates a Message with a specific text, sender and receiver
              * @param Text The text message
              * @param sender The sender Client ID
              * @param receiver The receiver Clint ID
              */
            public Message(String text, int sender, int receiver){
                        this(text,sender);
                        this.receiver=receiver;
            }

            /** Sets the receiver of the message
              * @param receiver The receiver Client ID
              */
            public void setReceiver(int receiver){
                        this.receiver=receiver;
            }
```

```
            /** Sets the sender of the message
              * @param sender The sender Client ID
              */
            public void setSender(int sender){
                        this.sender=sender;
            }

            /** Returns the text of the message
              * @return The text
              */
            public String getText(){
                        return text;
            }

            /** Returns the sender of the message
              * @return The sender Client ID
              */
            public int getSender(){
                        return sender;
            }

            /** Returns the receiver of the message
              * @return The receiver Client ID
              */
            public int getReceiver(){
                        return receiver;
            }

            /** Check whether two instances of Message are equal
              * @return true/false
              */
            public boolean equals(Object obj){
                        if (obj instanceof Message){
                                    Message m=(Message)obj;
                                    if
((text.equals(m.getText()))&&(sender==m.getSender())&&(receiver==m.getReceiver()
))
                                                return true;
                                    else return false;
                        }
                        else return false;
            }

}
```

2.25 MessageGUI Class

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;

/** The <code>MessageGUI</code> class represents the GUI for the Messaging
System of the Mobile Client*/
public class MessageGUI extends JPanel implements ActionListener{

            protected JTextArea mainText,msgText;
            protected JButton sendBt;
            protected JPanel msgSend,msgList;
```

```
protected JScrollPane mainPane,msgPane;
protected MobileClient localClient;

/** Creates the GUI
  * @param localClient The MobileClient
  */
public MessageGUI(MobileClient localClient){
           this.localClient=localClient;

           msgSend= new JPanel();
           msgList= new JPanel();
           msgList.setLayout(new FlowLayout());
           msgSend.setLayout(new FlowLayout());
           msgSend.setBorder(new TitledBorder("Send Message"));
           msgList.setBorder(new TitledBorder("Message History"));

           mainText= new JTextArea();
           mainText.setLineWrap(true);
           mainText.setWrapStyleWord(true);
           mainText.setEditable(false);
           mainPane = new JScrollPane(mainText);
           mainPane.setPreferredSize(new Dimension (200,300));
           msgList.add(mainPane);

           msgText = new JTextArea();
           msgText.setLineWrap(true);
           msgText.setWrapStyleWord(true);
           msgPane = new JScrollPane(msgText);
           msgPane.setPreferredSize(new Dimension (130,50));
           msgSend.add(msgPane);

           sendBt = new JButton("Send");
           sendBt.addActionListener(this);
           msgSend.add(sendBt);


           //this.setPreferredSize(new Dimension(250,450));
           this.setBorder(new EmptyBorder(10,10,10,10));

           GridBagLayout gridbag = new GridBagLayout();
           GridBagConstraints c = new GridBagConstraints();
           this.setLayout(gridbag);

           c.gridx=0;
           c.gridy=0;
           gridbag.setConstraints(msgList,c);
           this.add(msgList);

           c.gridx=0;
           c.gridy=1;
           gridbag.setConstraints(msgSend,c);
           this.add(msgSend);

}

/** Adds a message to the message board
  * @param msg The message to add
  */
public void addMessage(Message msg){
```

```
                    mainText.append("Central Operator :
"+'\n'+msg.getText()+'\n'+"____________________"+'\n');
                    this.repaint();

          mainPane.getVerticalScrollBar().setValue(mainPane.getVerticalScrollBa
r().getMaximum());
          }

          /** Adds a local message to the message board
            * @param txt The message in text form
            */
          public void addLocalMsg(String txt){
                    mainText.append(localClient.getName()+" :
"+'\n'+txt+'\n'+"____________________"+'\n');
                    this.repaint();

          mainPane.getVerticalScrollBar().setValue(mainPane.getVerticalScrollBa
r().getMaximum());
          }


          /** Implementation of the ActionListener interface*/
          public void actionPerformed(ActionEvent e){
                    if (e.getSource() == sendBt){
                              localClient.sendMessage(msgText.getText());
                              addLocalMsg(msgText.getText());
                              msgText.setText("");
                    }
          }


}
```

2.26 MobileClient Class

```
import java.net.*;
import java.io.*;
import java.util.*;

/** The <code>MobileClient</code> class is the the main class of the Mobile
Client Component
    of the Car Tracking System. It holds informations about the Mobile Client
and is the actual
    Client application for the Mobile User*/
public class MobileClient extends Client{
          MainGUI main;
          protected GPSDevice GPS;
          protected boolean GISStarted=false;


          /** Creates an instance of the MobileClient with a specific Name and
IP address
            * @param Name The name of the MobileClient
            * @param IP The IP address where the Server is running (127.0.0.1
can be used for local connections)
            */
          public MobileClient(String Name,String IP){
                    super(Name,IP);
                    super.setClientType(SysConst.MOBILE_CLIENT);
```

```
                    float r1=(new Float(Math.random())).floatValue();
                    float r2=(new Float(Math.random())).floatValue();
                    GPS= new GPSDevice((r1/35.0f)+25.206518f,54.670831f
+(r2/35.0f),this);

          }//MobileClient


          /** Shows the results in a frame
            * @param res The <code>Result</code> object to be showed
            */
          public void showResult(Result res){
                    main.showResult(res);
          }

          /** Starts the GPS drevice reading
            */
          public void startGPS(){
                    GPS.start();
          }

          /** Sets the <code>Database</code> object used by the Mobile Client
            * @param db The database object
            */
          public void setDBStructure(Database db){
                    this.db=db;
                    main = new MainGUI(db,this);
                    MCT.setGUI(main);
          }

          /** Sets the coordinates of the Mobile Client
            * @param c The coordinate of the Mobile Client
            */
          public void setCoordinate(Coordinate c){
                    try{
                                        if (GISStarted==false){
                                                  gisCom.createObject(new
GISObject(info.clientName,c));
                                                  GISStarted=true;
                                        }
                                        else{gisCom.updateObject(new
GISObject(info.clientName,c));}

          MCT.getCommunication().send_Position(new
GISObject(info.clientName,c),output);
                                  }
                    catch(Exception e){System.out.println("Exception at GIS
Client: "+e);}
          }

          /** Sends a message to the Central Client through the Server.
            * @param text The message in the form of text
            */
          public void sendMessage(String text){
                    try{
                                  Message msg= new Message(text);

          MCT.getCommunication().send_message(msg,output);
                                  }
```

```
                    catch(Exception e){System.out.println("Msg Exception at
MobileClient "+e);}
          }

          /** Shows a message in the main GUI. It basically calls the
<code>addMessage(Message msg)</code>
            * of the <code>MainGUI</code> class.
            * @param msg The <code>Message</code> object to show
            */
          public void showMessage(Message msg){
                    main.addMessage(msg);
          }

          /** Closes the application and the connection to the Server*/
          public void closeClient(){
                    try{
                              super.closeClient();
                              gisCom.deleteObject(new
GISObject(info.clientName,new Coordinate(0,0)));
                              }
                    catch(Exception e){System.out.println("Exceprion at GIS
Client : "+e);}
          }

          /** The main function used to run the application*/
          public static void main(String[] args){
                    try{
                              MobileClient client= new
MobileClient(args[0],args[1]);
                              client.run();
                              client.startGPS();
                              }
                    catch( IndexOutOfBoundsException
e){System.out.println("Use : java MobileClient <username>");}
          }

}
```

2.27 MobileClientThread Class

```
import java.net.*;
import java.io.*;
import java.awt.*;
import javax.swing.*;

/** The <code>MobileClientThread</code> class is a thread run by a client
  * and takes care of receiving data from the Server.*/
public class MobileClientThread extends Thread{
          protected Socket server;
          protected Client localClient;
          protected ObjectInputStream input;
          protected User user;
          protected ClientServerCommunication CSCom;
          protected Component GUI;


          /** The constructor*/
          MobileClientThread(){}
```

```
            /** Creates an instance of the MobileClientThread class
              * @param sever The socket connection to the server
              * @param localClient Reference to the running client
              */
            public MobileClientThread(Socket server,Client localClient){
                        this.server=server;
                        CSCom= new EBCSCommunication();
                        this.localClient=localClient;
                        try{
                                    input=new
ObjectInputStream(server.getInputStream());
                                    }
                        catch (Exception e){System.out.println("Exception at
ClientThread 1"+e);}
            }

            public void setGUI(Component c){
                        GUI=c;
            }

            /** Returns the communication componet
              * @return The communication component
              */
            public ClientServerCommunication getCommunication(){
                        return CSCom;
            }

            /** Starts the Thread
              */
            public void run (){
                        try{
                                    while(input!=null){
                                                int
action=((Integer)CSCom.readAction(input)).intValue();
                                                switch (action){
                                                            case Action.SEND_RESULT :
{getResult();}break;
                                                            case
Action.SEND_DBSTRUCTURE : {getDBStructure();}break;
                                                            case
Action.SEND_USERLOGIN: {getUserLOGIN();};break;
                                                            case
Action.SEND_USERLOGOUT: {getUserLOGOUT();};break;
                                                            case Action.SEND_MESSAGE
: {getMessage();};break;
                                                            case Action.SEND_POSITION
: {getPosition();}break;
                                                }
                                    }
                                    System.out.print("Closed");
                                    }//try
                        catch(EOFException e){

            System.out.println("Closed");

            System.exit(0);

            }
```

```
                        catch(Exception e){System.out.println("Exception at
ClientThread 2"+e);}
            }

            /** Returns the User
              * @return The User
              */
            public User getUser(){
                        return user;
            }

            protected void getUserLOGIN(){
                        try{
                                    CentralClient
central=(CentralClient)localClient;
                                    user=(User)CSCom.receiveUserInfo(input);
                                    central.userLoggedIn(user);
                                    }
                        catch(Exception e){System.out.println("Exception at
ClientThread"+e);}
            }

            protected void getUserLOGOUT(){
                        try{
                                    CentralClient
central=(CentralClient)localClient;
                                    user=(User)CSCom.receiveUserInfo(input);
                                    central.userLoggedOut(user);
                                    }
                        catch(Exception e){System.out.println("Exception at
ClientThread"+e);}

            }

            protected void getPosition(){
                        try{
                                    CentralClient
central=(CentralClient)localClient;

            central.getPosition((GISObject)CSCom.receive_Position(input));
                                    }
                        catch(Exception e){System.out.println("GIS Exception at
Client Thread : "+e);}
            }

            protected void getResult(){
                        try{
                                    Result res=(Result)CSCom.receiveResult(input);
                                    localClient.showResult(res);
                                    }
                        catch(Exception e){System.out.println(e);}
            }

            protected void getDBStructure(){
                        try{
                                    Database db
=(Database)CSCom.receiveDBStructure(input);
                                    localClient.setDBStructure(db);
                                    }
```

```
                        catch(Exception e){System.out.println("Exception at
ClientThread"+e);}

            }

            /** Terminates the connection with the Server
              * @param output The output stream to the Sever */
            public void terminateConnection(ObjectOutputStream output){
                        try{
                                    System.out.print("Closing Connection....");
                                    CSCom.terminateConnection(output);
                                    }
                        catch(Exception e){System.out.println("Exception at
ClientThread 6:"+e);}
            }

            protected void getMessage(){
                        try{
                                    Message
msg=(Message)CSCom.receive_message(input);
                                    localClient.showMessage(msg);
                                    }
                        catch(Exception e){System.out.println("Msg Exception at
ClientThread "+e);}
            }



}
```

2.28 NotEnoughParameterException Class

```
/** The <code>NotEnoughParameterException</code> is thrown when a connection is
  * attempted to the database server, but the parameter used are not enough
  */
public class NotEnoughParameterException extends Exception{
            public NotEnoughParameterException(){
                        super("The Database request more parameters for
connection");
            }
}
```

2.29 OnLineUsersGUI Class

```
import javax.swing.*;
import javax.swing.AbstractListModel.*;
import java.awt.*;
import java.util.*;

/** The <code>OnLineUsersGUI</code> class is the GUI showing the on line users
of the system
  */
public class OnLineUsersGUI extends JPanel implements Observer{
            private CentralClient localClient;
            private JList users;
            private DefaultListModel listModel;
```

```
            /** Create and instance of the OnLineUsersGUI class
              * @param localClient The Central Client
              */
            public OnLineUsersGUI(CentralClient localClient){
                        this.localClient=localClient;
                        listModel = new DefaultListModel();
                        updateListModel();
                        users = new JList(listModel);
                        this.add(users);
            }

            private void updateListModel(){
                        listModel.removeAllElements();
                        for (int i =0; i<localClient.getUsersVector().size();i++){

            listModel.addElement(((User)localClient.getUsersVector().get(i)).getN
ame());
                        }
            }

            /** Implementation of the Observer interface
              */
            public void update(Observable o, Object arg){
                        updateListModel();
                        this.repaint();
            }
}
```

2.30 Query Class

```
import java.util.*;
import java.io.*;

/** The <code>Query</code> class describes a query made by the user
  */
public class Query implements Serializable{
            private Vector fields;
            private String DBName;
            private String tableName;

            /** Return the database name
              * @return The database name
              */
            public String getDBName(){
                        return DBName;
            }
            /** Returns the table where the query has been made
              * @return The table name
              */
            public String getTableName(){
                        return tableName;
            }

            /** Creates a Query object for a specific table in a specific
database
              * @param DBName The database name
              * @param tableName The table name
              */
            public Query(String DBName, String tableName){
```

```
                this.DBName=DBName;
                this.tableName=tableName;
                fields=new Vector();
        }

        /** Returns the number of fields in the query
          * @return The fields number
          */
        public int getFieldNo(){
                return fields.size();
        }

        /** Adds a new field to the query
          * @param RF The field to be added
          */
        public void addField(QueryField RF){
                fields.add(RF);
        }

        /** Returns a specific field in the query
          * @param index The index of the field
          */
        public QueryField getField(int index){
                return (QueryField)fields.get(index);
        }

        /** Returns wheter the Query is empty or not
          * @return true/false
          */
        public boolean isEmpty(){
                if (fields.size()==0) return true;
                else return false;
        }

}
```

2.31 QueryField Class

```
/** The <code>QueryField</code> class represents a field in the Query object
  */
public class QueryField extends ResultField{

        protected String criteria;


        /** Creates a QueryField object
          * @param name The field name
          * @param type The field type
          * @param criteria The field search criteria
          * @param value The field value
          */
        public QueryField(String name, String type,String criteria, Object
value){
                super(name,type,value);
                this.criteria=criteria;
        }

        /** Returns the field search criteria
          * @return The search criteria
```

```
            */
           public String getCriteria(){
                      return criteria;
           }

    /** Compares two QueryField Objects
     * @param obj The QueryField to compare
     * @return true false
     */
           public boolean equals(Object obj){
                      if (obj instanceof QueryField){
                                 QueryField temp=(QueryField)obj;
                                 if ( (super.equals((ResultField)temp)) &&
(criteria.equals(temp.getCriteria()))){
                                            return true;
                                 }
                                 else return false;
                      }
                      else return false;
           }

}//QueryField
```

2.32 QueryGUI Class

```
import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;

/** The <code>QueryGUI</code> class is the window used by clients to send
queries to the Server
  */
public class QueryGUI extends JFrame implements ActionListener{
           JButton sendBt,cancelBt;
           Vector components;
    Database DB;
           String[] strCriteria = {"=","LIKE"};
           Client localClient;

           /** Creates a QueryGUI object
             * @param DB The database
             * @param localClient The Client
             */
           public QueryGUI(Database DB, Client localClient){

                      super("Search on " + DB.getTable().getName());
                      this.localClient=localClient;
                      this.DB=DB;
                      components = new Vector();
                      Vector fields = DB.getTable().getFields();
                      JPanel pane = new JPanel();
                      GridBagLayout gridbag = new GridBagLayout();
                      GridBagConstraints c = new GridBagConstraints();
                      pane.setLayout(gridbag);
                      pane.setBorder(new EmptyBorder(10,10,10,10));

           pane.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
```

```
        int k=0;
                    for (int i=0; i < fields.size();i++){
                              Field f = (Field)fields.get(i);
                              if (f.getSearcheable() == true){
                                        c.anchor= GridBagConstraints.WEST;
                                        c.gridx=0;
                                        c.gridy=k;
                                        JLabel label=new
JLabel(f.getName());
                                        gridbag.setConstraints(label,c);
                                        pane.add(label);
                                        k++;
                                        int size=f.getSize();
                                        if
(f.getType().equals(Field.GRAPHIC)){
                                                  label=new JLabel("Image
Query is not part of this Demo");
                                                  c.anchor=
GridBagConstraints.WEST;
                                                  c.gridx=0;
                                                  c.gridy=k;

           gridbag.setConstraints(label,c);
                                                  pane.add(label);
                                                  k++;
                                        }//if
                                        else
                                                  if
(f.getType().equals(Field.BOOLEAN)){
                                                            String[] items
= {"","Yes","No"};
                                                            JComboBox combo
=new JComboBox(items);
                                                            c.anchor=
GridBagConstraints.WEST;
                                                            c.gridx=0;
                                                            c.gridy=k;

           gridbag.setConstraints(combo,c);

           pane.add(combo);

           components.add(new Element(f.getName(),f.getType(),"=",combo));
                                                            k++;
                                                            }
                                                  else {
                                                            JTextField
text;
                                                            JComboBox
criteria=new JComboBox(strCriteria);
                                                            if (f.getSize()
> 30)

text=new JTextField(30);
                                                            else text=new
JTextField(f.getSize());
                                                            JPanel
criteriaPanel= new JPanel();
```

```
            text.setText("");

            criteriaPanel.setLayout(new FlowLayout());

            criteriaPanel.add(criteria);

            criteriaPanel.add(text);
                                                                    c.anchor=
GridBagConstraints.WEST;
                                                                    c.gridx=0;
                                                                    c.gridy=k;

            gridbag.setConstraints(criteriaPanel,c);

            pane.add(criteriaPanel);
                                                                    k++;

            text.setText("");

            components.add(new Element(f.getName(),f.getType(),criteria,text));


                                                    }//else
                                    }//if

                        }//for
                        JPanel temp = new JPanel();
                        sendBt = new JButton("Send");
                        sendBt.addActionListener(this);
                        cancelBt = new JButton("Cancel");
                        cancelBt.addActionListener(this);
                        temp.setBorder(new EmptyBorder(10,0,0,0));
                        temp.setLayout(new FlowLayout());
                        temp.add(cancelBt);
                        temp.add(sendBt);
                        c.anchor= GridBagConstraints.EAST;
                        c.gridx=0;
                        c.gridy=k;
                        gridbag.setConstraints(temp,c);
                        pane.add(temp);

            setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

                        this.setContentPane(pane);
                        this.pack();
                        this.setVisible(true);
            }

            /** Implementation of the ActionListener interface*/
            public void actionPerformed(ActionEvent e){

                        if (e.getSource() == sendBt){
                                    try{
                                                Query myQuery=createQuery();
                                                if (myQuery==null)
            JOptionPane.showMessageDialog(this,"Some of the criteria have a wrong
format or no criteria has been selected");
                                                else{
```

```
            localClient.sendQuery(createQuery());
                                                this.dispose();
                                                }
                                    }
                                    catch(Exception ex){System.out.println(ex);}
                        }
                        else
                                    if (e.getSource() == cancelBt){
                                                this.dispose();
                                    }
            }

            /** automatically generates the look of the GUI
              */
            public Query createQuery(){
                        Query query=new
Query(DB.getName(),DB.getTable().getName());
                        QueryField rf;
                        for (int i=0;i<components.size();i++){
                                    Element e = (Element)components.get(i);
                                    if (e.component instanceof JTextField){
                                                JTextField t=
(JTextField)e.component;
                                                JComboBox tmp=(JComboBox)e.criteria;
                                                if (!(t.getText().equals(""))){
                                                            if
(e.Type.equals("INT")){
                                                                        try{
                                                                                    rf =
new QueryField(e.Name,e.Type,(String)tmp.getSelectedItem(),new
Integer(t.getText()));
                                                                                    }

            catch(NumberFormatException ex){return null;}
                                                                        }
                                                            else
                                                                        if
(e.Type.equals("FLOAT")){
                                                                                    try{

            rf = new QueryField(e.Name,e.Type,(String)tmp.getSelectedItem(),new
Float(t.getText()));

            }

            catch(NumberFormatException ex){return null;}
                                                                        }
                                                                        else{
                                                                                    rf =
new QueryField(e.Name,e.Type,(String)tmp.getSelectedItem(),t.getText());
                                                                                    }
                                                            query.addField(rf);
                                                }
                                    }//if
                                    else
                                                if (e.component instanceof
JComboBox){
```

```
                                                    JComboBox c=
(JComboBox)e.component;
                                                    String s=
(String)c.getSelectedItem();
                                                    if (!(s.equals(""))){
                                                              if
(s.equals("Yes"))
                                                                        rf=
new QueryField(e.Name,e.Type,(String)e.criteria,new Boolean("true"));
                                                              else rf= new
QueryField(e.Name,e.Type,(String)e.criteria,new Boolean("false"));

          query.addField(rf);

                                                    }
                                          }//if
                    }
                    if (query.getFieldNo()==0) return null;
                    else return query;

          }

          private class Element{
                    public String Name;
                    public String Type;
                    public Object component;
                    public Object criteria;
                    public Element(String Name,String type,Object
criteria,Object component){
                              this.criteria=criteria;
                              this.Type=type;
                              this.Name=Name;
                              this.component=component;
                    }
          }

}
```

2.33 Result Class

```
import java.util.*;
import java.io.*;

/**
 * The <code>Result</code> holds a result from the database, a row with
 * dynamic number of fields
 */
public class Result implements Serializable{
          /** Result Status OK*/
          public static final int OK = 1;
          /** Result Status Too broad*/
          public static final int TOO_BROAD = 2;

          protected Vector resultRow;
          protected int status;

    /** Creates a new empty Result Object*/
```

```
        public Result(){
                  resultRow=new Vector();
                  status = Result.OK;
        }
        /** Returns true if the Result is empty*/
        public boolean isEmpty(){
                  return resultRow.isEmpty();
        }
        /** Sets the status of the Result
          * @param status The result status
          */
        public void setStatus(int status){
                  this.status=status;
        }
/** Returns the status of the Result*/
        public int getStatus(){
                  return status;
        }

/** Returns a row of results
  * @param row The row index
  */
        public Vector getResultRow(int row){
                  return (Vector)resultRow.get(row);
        }


/** Returns one Field from the results
  * @param row - The row number in the result grid
  * @param col - the field number in the row of results
  * @return The Field
  */
        public ResultField getResultField(int row,int col){
                  try{
                            Vector VT = (Vector)resultRow.get(row);
                            ResultField temp=(ResultField)VT.get(col);
                            return temp;
                            }
                  catch(ArrayIndexOutOfBoundsException e){return null;}
        }



        private Iterator getIterator(){
                  return resultRow.iterator();
        }

        /** Adds a new ResultField in a specific row
          * @param row The row of results
          * @param field The field to be added
          */
        public void addResultField(int row,ResultField field){
                  try{
                            Vector temp=(Vector)resultRow.get(row);
                            temp.add(field);
                            }
                  catch(ArrayIndexOutOfBoundsException e){

                                                      Vector v=new Vector();
```

```
                                                    v.add(field);

                                                    resultRow.add(v);

            }
            }

            /** Returns the number of rows in the Result*/
            public int getRows(){
                        return resultRow.size();
            }

            /** Returns the number of columns in the Result*/
            public int getColumns(){
                        if (resultRow.size()!=0){
                                    Vector v=(Vector)resultRow.get(0);
                                    return v.size();
                        }
                        else
                                    return 0;
            }

    /** Returns a String representation of the Result object
      * @return String Representation*/
            public String toString(){
                        String s=new String();
                        s=(resultRow.toString());
                        return s;
            }
}//Result
```

2.34 ResultCache Class

```
import java.util.*;

/** The <code>ResultCache</code> class is a cache for Result objects
  */
public class ResultCache extends Result{
            public static int MAXLOG = 50;
            private int presentIndex;

            /** Creates a ResultCache object
              */
            public ResultCache(){
                        super();
                        presentIndex=0;
            }

            /** Adds a Result object to the cache
              * @param rs The result to be added
              */
            public void addResult(Result rs){
                        Vector v;
                        int rows=rs.getRows();
                        int columns=rs.getColumns();
                        for (int i=0;i<rows;i++){
                                    v = new Vector();
```

```
                                for (int k=0;k<columns;k++){
                                          v.add(rs.getResultField(i,k));
                                }//for
                                try{

            resultRow.setElementAt(v,presentIndex);
                                          }
                                catch(ArrayIndexOutOfBoundsException e){


            resultRow.add(presentIndex,v);

                                                                  }
                                presentIndex=((presentIndex+1)%MAXLOG);
                      }//for

            }

}
```

2.35 ResultField Class

```
/** The <code>ResultField</code> class extends the Field class and describes one
of the fields
  * in the Result Class*/
public class ResultField extends Field{

            private Object value;



    /** Creates a ResultField object
      * @param name The name of the Field
      * @param type The Type of the Field
      * @param value The value of the Field*/
            public ResultField(String name, String type, Object value){
                        super(name,type,false,0);
                        this.value=value;
            }

    /** Returns the value of the Field
      * @return The value
      */
            public Object getValue(){
                        return value;
            }

    /** Compares two ResultField Objects
     * @param obj The ResultField to compare
     * @return true false
     */
            public boolean equals(Object obj){
                        if (obj instanceof ResultField){
                                    ResultField temp=(ResultField)obj;
                                    if ( (super.equals((Field)temp)) &&
(value.equals(temp.getValue()))){
                                                return true;
                                    }
```

```
                                    else return false;
                        }
                        else return false;
            }

    /** Returns a String representation of the object
      * @return String Representation*/
            public String toString(){
                        return("<Name: "+super.getName()+" Value:
"+value.toString()+">");
            }
}//resultField
```

2.36 ResultGUI Class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.border.*;

/** The <code>ResultGUI</code> class is the window displayin the
<code>Result</code> objects
  */
public class ResultGUI extends JFrame implements ActionListener {

            JPanel pane;
            JScrollPane centerPane;
            JToolBar toolBar;
            JButton backBt;
            JButton forwardBt;
            JTextField countLb;
            Result rs;
            JPanel buttonPanel;
            JButton closeBt;
            JComboBox actionCb;
            JButton sendBt;
            MobileClient localClient;

            int rowNum = 0;

            /** Creates and display the ResultGUI
              * @param rs The Result to show
              * @param localClient The Mobile Client
              */
            public ResultGUI(Result rs, MobileClient localClient){
                        super("Result Window");
                        this.rs=rs;
                        this.localClient=localClient;
                        pane = new JPanel();
                        this.getContentPane().setLayout(new BorderLayout());
                        toolBar= new JToolBar();
                        backBt = new JButton(new ImageIcon("image/Back24.gif"));
                        forwardBt= new JButton(new
ImageIcon("image/Forward24.gif"));
                        backBt.addActionListener(this);
                        forwardBt.addActionListener(this);
                        countLb= new JTextField(10);
                        countLb.setEditable(false);
                        sendBt = new JButton("Send");
```

```
                    sendBt.addActionListener(this);

                    //*****DEMO VERSION ONLY*******
                              String[] actions={"","Checked","Alarm
On","Intruder Found","Guard Missing","Window Open"};
                              actionCb = new JComboBox(actions);
                              actionCb.setEditable(true);
                    //*****DEMO VERSION ONLY*******

                    toolBar.add(backBt);
                    toolBar.add(countLb);
                    toolBar.add(forwardBt);
                    toolBar.add(actionCb);
                    toolBar.add(sendBt);
                    this.getContentPane().add(toolBar,BorderLayout.NORTH);
                    createGUI(0);
                    centerPane =new JScrollPane(pane);
                    this.getContentPane().add(centerPane,BorderLayout.CENTER);

                    buttonPanel=new JPanel();
                    buttonPanel.setLayout(new FlowLayout());
                    buttonPanel.setBorder(new EmptyBorder(10,0,0,0));
                    closeBt= new JButton("Close");
                    closeBt.addActionListener(this);
                    buttonPanel.add(closeBt);
                    this.getContentPane().add(buttonPanel,BorderLayout.SOUTH);


          setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
                    this.pack();
                    this.setVisible(true);
          }

          /** Implementation of the ActionListener inteface
            */
          public void actionPerformed(ActionEvent e){
                    if (e.getSource() == backBt){
                              if (rowNum != 0){
                                        rowNum--;
                                        pane.removeAll();
                                        createGUI(rowNum);
                                        pane.repaint();
                                        repaint();

                              }
                    }//if
                    else
                              if (e.getSource() == forwardBt){
                                        if (rowNum < (rs.getRows()-1)){
                                                  rowNum++;
                                                  pane.removeAll();
                                                  createGUI(rowNum);
                                                  pane.repaint();
                                                  repaint();
                                        }
                              }
                              else
                                        if (e.getSource() == closeBt){
                                                  this.dispose();
```

```
                                        }
                                        else
                                                  if (e.getSource() ==
sendBt){
                                                            String msg= new
String("Object ID :"+(rs.getResultField(0,0)).getValue()+" action :
"+actionCb.getSelectedItem());

          localClient.sendMessage(msg);
                                                  }
          }



          /** Starts the process of automatic creation of the GUI based
            * on one of the Results
            * @param row The row index of the result to be showed from the
Result object
            */
          public void createGUI(int row){
                    JTextField text;
                    countLb.setText(" "+(row+1)+" of "+rs.getRows());
                    countLb.repaint();
                    int k=0;
                    GridBagLayout gridbag = new GridBagLayout();
                    GridBagConstraints c = new GridBagConstraints();
                    pane.setLayout(gridbag);
                    pane.setBorder(new EmptyBorder(10,10,10,10));

          pane.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);

                    if (!(rs.isEmpty())){
                              for (int i=0;i<rs.getColumns();i++){
                                        ResultField rf =
rs.getResultField(row,i);
                                        c.anchor= GridBagConstraints.WEST;
                                        c.gridx=0;
                                        c.gridy=k;
                                        JLabel label=new
JLabel(rf.getName());
                                        gridbag.setConstraints(label,c);
                                        pane.add(label);
                                        k++;
                                                  if
(rf.getType().equals(Field.BOOLEAN)){
                                                            c.anchor=
GridBagConstraints.WEST;
                                                            c.gridx=0;
                                                            c.gridy=k;
                                                            Boolean b
=(Boolean)rf.getValue();
                                                            if
(b.equals(new Boolean("true"))) text = new JTextField("Yes");
                                                            else text = new
JTextField("No");

          text.setEditable(false);

          gridbag.setConstraints(text,c);
```

```
                                                                pane.add(text);
                                                                k++;
                                                                }
                                                        else
                                                                if
(rf.getType().equals(Field.GRAPHIC)){
                                                                        int
Width=400;
                                                                        int
Height=400;
            SImage simage=(SImage)rf.getValue();
            ImageIcon image= new ImageIcon(simage.bytes);
            ScrollablePicture sp=new ScrollablePicture(image,5);
            JScrollPane jsp =new JScrollPane(sp);
            JPanel pa=new JPanel();
            pa.add(jsp);
            c.anchor= GridBagConstraints.WEST;
            c.gridx=0;
            c.gridy=k;
            gridbag.setConstraints(pa,c);
                                                                        if
(image.getIconWidth() < Width) Width=image.getIconWidth();
                                                                        if
(image.getIconHeight() < Height) Height=image.getIconHeight();
            jsp.setPreferredSize(new Dimension(Width+3,Height+3));
            pane.add(pa);
                                                                        k++;
                                                                }
                                                                else
                                                                        {
            String value;
                                                                        if
(rf.getType().equals(Field.INT)) {
            Integer integer=(Integer)rf.getValue();
            value=integer.toString();
            }
            else
                        if (rf.getType().equals(Field.FLOAT)){
                                    Float f=(Float)rf.getValue();
```

```
                                value=f.toString();
                        }
                        else value=(String)rf.getValue();
                                                                        if
(value.length() > 30){
            JTextArea textArea = new JTextArea(value);
            textArea.setEditable(false);
            textArea.setLineWrap(true);
            JScrollPane spane = new JScrollPane(textArea);
            spane.setPreferredSize(new Dimension(250, 50));
            JPanel p=new JPanel();
            p.add(spane);
            c.anchor= GridBagConstraints.WEST;
            c.gridx=0;
            c.gridy=k;
            gridbag.setConstraints(p,c);
            pane.add(p);
            k++;

            }//if
                                                                        else
{
            text=new JTextField(value);
            text.setEditable(false);
            c.anchor= GridBagConstraints.WEST;
            c.gridx=0;
            c.gridy=k;
            gridbag.setConstraints(text,c);
            pane.add(text);
            k++;
            }//else
```

```
                                                                    }//else

                                    }//for

                                    pane.repaint();
                                    this.setVisible(true);
                            }//if
                            else{
                                     if (rs.getStatus()== Result.OK )pane.add(new
JLabel("No Mach for the query found"));
                                     else
                                                if (rs.getStatus()==
Result.TOO_BROAD )pane.add(new JLabel("Too many results, please narrow your
search"));
                                    }
            }

}
```

2.37 ResultInfo Class

```
/** The <code>ResultInfo</code> class is used as a transport packet
  * for the Result Object in the network
  */
public class ResultInfo{
            /** The result Object*/
            public Result result;

            /** The Central Client ID*/
            public Integer CCID;
}
```

```
ScrollablePicture Class (by The SUN)
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

public class ScrollablePicture extends JLabel implements Scrollable {

    private int maxUnitIncrement = 1;

    public ScrollablePicture(ImageIcon i, int m) {
        super(i);
        maxUnitIncrement = m;
    }

    public Dimension getPreferredScrollableViewportSize() {
        return getPreferredSize();
    }

    public int getScrollableUnitIncrement(Rectangle visibleRect,
                                          int orientation,
                                          int direction) {
        //Get the current position.
        int currentPosition = 0;
        if (orientation == SwingConstants.HORIZONTAL)
            currentPosition = visibleRect.x;
        else
```

```
            currentPosition = visibleRect.y;

        //Return the number of pixels between currentPosition
        //and the nearest tick mark in the indicated direction.
        if (direction < 0) {
            int newPosition = currentPosition -
                             (currentPosition / maxUnitIncrement) *
                              maxUnitIncrement;
            return (newPosition == 0) ? maxUnitIncrement : newPosition;
        } else {
            return ((currentPosition / maxUnitIncrement) + 1) *
                   maxUnitIncrement - currentPosition;
        }
    }

    public int getScrollableBlockIncrement(Rectangle visibleRect,
                                           int orientation,
                                           int direction) {
        if (orientation == SwingConstants.HORIZONTAL)
            return visibleRect.width - maxUnitIncrement;
        else
            return visibleRect.height - maxUnitIncrement;
    }

    public boolean getScrollableTracksViewportWidth() {
        return false;
    }

    public boolean getScrollableTracksViewportHeight() {
        return false;
    }

    public void setMaxUnitIncrement(int pixels) {
        maxUnitIncrement = pixels;
    }
}
```

2.38 Sdatabase Class

```
import java.io.*;

/**
 * The <code>SDatabase</code> class holds the informations about specific
 * databases that the server can access.
 * @version 0.1
 */
public class SDatabase extends Database implements Serializable{


                    private DBServerCommunication dbCom;


                    /** Creates a SDatabase Object with the specific path and
no username or password*/
                    public SDatabase(String name,String path, String driver){
                              this(name,path,null,null,driver);
                    }
```

```
                    /** Creates a SDatabase Object with the specific path,
username and password*/
                    public SDatabase(String name,String path, String username,
String password, String driver){
                              dbCom=new EBConnection(driver);
                              dbCom.setDBPath(path);
                              dbCom.setUsername(username);
                              dbCom.setPassword(password);
                              this.name=name;
                    }
                    /** Connects to the Database using the Database path,
username and password stored
                      * in the internal variables.
                      * @throws Exception
                      */
                    public void connectToDB() throws Exception{
                              try{
                                        dbCom.connectToDB();
                                        }
                              catch(Exception e){throw e;}
                    }
                    /** Disconnects from the database.*/
                    public void disconnectFromDB() throws Exception{
                              try{
                                        dbCom.disconnectFromDB();
                                        }
                              catch(Exception e){throw e;}

                    }

                    /** Sends a query to the database.
                     * @param query The Query Object
                     * @return The result Object
                     */
                    public Object sendQuery(Object query) throws Exception{
                              try{
                                        return dbCom.sendQuery(query);
                                        }
                              catch(Exception e){throw e;}
                    }

                    /** Returns the Database Path
                      * @return The Path
                      */
                    public String getDBPath(){
                              return (String)dbCom.getDBPath();
                    }

                    /** Returns the Database Username
                      * @return the Username
                      */
                    public String getUsername(){
                              return (String)dbCom.getUsername();
                    }
                    /** Returns the Database Password
                      * @return The Password
                      */
                    public String getPassword(){
                              return (String)dbCom.getPassword();
```

```
                        }

        /** Compares two SDatabase Objects
          * @return true false
          */
           public boolean equals(Object O){
                                if (O instanceof SDatabase){
                                          SDatabase d=(SDatabase)O;
                                            if
(this.getDBPath().equals(d.getDBPath()))
                                                                  if
(this.getUsername().equals(d.getUsername()))
                                                                                  if
(this.getPassword().equals(d.getPassword()))
                                                                                                  if
(DBTable.equals(d.getTable()))

if (name.equals(d.getName())) return true;
                                          return false;
                                }
                                else return false;
                        }

                        /** Translates the SDatabase object into a Database object
                          * @return The Database object
                          */
                        public Database toDatabase(){
                                  Database db = new Database(this.getName());
                                  db.setTable(this.getTable());
                                  return db;
                        }

                        /** Translates a Query to an SQL sentence
                          * @param query The Query object
                          * @return The SQL statement
                          */
                        public Object translateQuery(Object query){
                                  return dbCom.translateQuery(query);
                        }

}// end Database class
```

2.39 SearchPanel Class

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

/** The <code>SearchPanel</code> class represents the Querying part of the Main
window
  * of the Mobile Client application
  */
public class SearchPanel extends JPanel implements ActionListener{

                        JButton queryBt;
                        JButton logBt,closeBt;
                        Database db;
                        MobileClient localClient;
                        MainGUI main;
```

```
                        /** Creates and istance of SearchPanel
                          * @param db The Database structure
                          * @param localClient The MobileClient
                          * @param main The main window
                          */
                        public SearchPanel(Database db,MobileClient localClient,
MainGUI main){

                                  this.main=main;
                                  this.localClient=localClient;
                                  this.db=db;
                                  this.setLayout(new FlowLayout());
                                  queryBt = new JButton("Send Query");
                                  queryBt.addActionListener(this);
                                  this.add(queryBt);

                                  logBt = new JButton("Show Result Log");
                                  logBt.addActionListener(this);
                                  this.add(logBt);

                                  closeBt = new JButton("Exit");
                                  closeBt.addActionListener(this);
                                  this.add(closeBt);

                        }

                        /** Implementation of the ActionListener interface
                          */
                        public void actionPerformed( ActionEvent e){
                                  if (e.getSource() == queryBt){
                                                      if (!(db.equals(null)))
                                                                new
QueryGUI(db,localClient);
                                                      else
JOptionPane.showMessageDialog(this, "The Database Structure is not loaded !");
                                  }
                                  else
                                            if (e.getSource() == logBt){
                                                                if
(localClient.getResultLog()== null){System.out.println("is null");}
                                                                else new
ResultGUI(localClient.getResultLog(),localClient);
                                            }
                                            else
                                                      if (e.getSource() ==
closeBt){
                                                                main.close();
                                                      }
                        }

}
```

2.40 Server Class

```
import java.util.*;
import java.sql.*;
import java.io.*;
```

```
import javax.swing.*;
import java.net.*;


/** The <code>Server</code> class is the main class of the Server Component
  * It holds all the informations about the database connections and handles
  * all the service request form the clients
  */
public class Server{

           /** Describes the READY status of the Server*/
           public static final int READY = 0;
           /** Describes the NO_QUERY_SERVICES status of the server*/
           public static final int NO_QUERY_SERVICES = 1;


           private int status;

           private Vector databases;

           private ServerSocket server_socket;

           private Vector mobileThreads;

           private Vector centralThreads;

           private Config config;


           /** Loads the configuration file if the configuration file is not
found
             * it will create a new file with default settings.
             *
             * Present Default Settings:
             *      - Server Port number = 8888
             *      - DB structure file name = database.dat
             */
           public void loadConfiguration(){
                      String filename="config.dat";
                      try{
                                 ObjectInputStream in = new
ObjectInputStream(new FileInputStream(filename));
                                 config=(Config)in.readObject();
                                 in.close();
                                 }//try
                      catch(FileNotFoundException e){
                                            try{
                                                       ObjectOutputStream out =
new ObjectOutputStream(new FileOutputStream(filename));
                                                       config = new Config();
                                                       config.setPort(8888);

           config.setDBFilePath("database.dat");
                                                       out.writeObject(config);
                                                       out.flush();
                                                       out.close();
                                                       }
                                            catch(IOException ex){}
```

```
                              }//catch
                    catch(IOException e){System.err.println("Exception at
Server: 1"+e);}//catch
                    catch(ClassNotFoundException
e){System.err.println("Exception at Server: "+e);}

          }



          /** Creates and instance of the Server, setting up the socket to
performe communication
            */
          public Server(){
                    databases= new Vector();
                    mobileThreads=new Vector();
                    centralThreads=new Vector();
          }

          /** Broad casts to all the Central Clients the user that has logged
in
            */
          public void broadcastUsersLOGIN(User user){
                    if (centralThreads.size()!= 0)
                              for (int i=0;i<centralThreads.size();i++){

          ((ServerThread)centralThreads.get(i)).sendUserLOGIN(user);
                              }
          }

          /** Broad casts to all the Central Clients the user that has logged
out
            */
          public void broadcastUsersLOGOUT(User user){
                    if (centralThreads.size()!= 0)
                              for (int i=0;i<centralThreads.size();i++){

          ((ServerThread)centralThreads.get(i)).sendUserLOGOUT(user);
                              }
          }

          /** Sends a list of all the online Mobile CLients to a specific
Central Client
            * @param CCID The Central Client ID
            */
          public void sendUsers(int CCID){
                    if (mobileThreads.size()!=0){
                              for (int i=0;i<mobileThreads.size();i++){
                                        User
tmpUser=((ServerThread)mobileThreads.get(i)).getUser();

          ((ServerThread)centralThreads.get(CCID)).sendUserLOGIN(tmpUser);
                              }
                    }
          }

          /** Sends a Result object to a Mobile Client
            * @param tmpRes The Result object
            * @param CID The Mobile Client ID
```

```
             */
            public void sendResult(Result tmpRes,Integer CID){
                        if (CID.intValue()== SysConst.BROADCAST){
                                    for(int i=0; i<mobileThreads.size();i++){
                                    ServerThread
STtmp=(ServerThread)mobileThreads.get(i);
                                    STtmp.sendResultToMobile(tmpRes);
                                    }
                        }
                        else{
                                    ServerThread
STtmp=(ServerThread)mobileThreads.get(CID.intValue());
                                    STtmp.sendResultToMobile(tmpRes);
                        }
            }


            /** Stores the Database structure read from a file to the
<code>Database</code>
              * class
              * @throws InvalidStructureException - the Database structure in the
file is not valid
              */
            public void storeDBStructure(String DBname,String file, String
path,String username, String password, String driver) throws
InvalidStructureException{
                        if (validateDBStructure(file)==false) throw new
InvalidStructureException(file);
                        else
                            fillInStructure(DBname,file,path,username,password,
driver);
                        try{
                                    ObjectOutputStream out = new
ObjectOutputStream(new FileOutputStream(config.getDBFilePath()));
                                    out.writeObject(databases);
                                    out.flush();
                                    out.close();
                                    }
                        catch (FileNotFoundException
e){System.out.println("Exception at Server "+e);}
                        catch (IOException e){System.out.println("Exception at
Server "+e);}

            }


            /** Returns a specific database connected to the server
              * @param index The index ID of the database
              * @return The Database*/
            public SDatabase getDatabase(int index){
                        return (SDatabase)databases.get(index);
            }

    //Populates the Database structure from the database structure file
    private void fillInStructure(String DBname,String file,String path, String
username, String password, String driver){
                        SDatabase DB =new SDatabase(DBname,path,username,password,
driver);
```

```
                    try{
                              boolean exit=false;
                              int field_index=-1;
                              String name="";
                              String type="";
                              int size=0;
                              boolean search=false;
                              BufferedReader Bfile = new BufferedReader(new
FileReader(file));
                              do{
                                        String line = Bfile.readLine();
                                        StringTokenizer ST = new
StringTokenizer(line);
                                        String token = ST.nextToken();
                                        if (token.equals("TABLENAME")){
                                                  DB.setTable(new
Table(ST.nextToken()));
                                                  }//if
                                                  else
                                                  if
(token.equals("<FIELD>")){

          field_index++;
                                                  }//if
                                                  else
                                                            if
(token.equals("NAME")){

          name=ST.nextToken();

          }//if
                                                                      else

          if (token.equals("TYPE")){

                    type=ST.nextToken();

          }//if

          else

                    if (token.equals("SEARCHEABLE")){

                              String temp=ST.nextToken();

                              if (temp.equals("Y")) search=true;

                              else search=false;

                    }//if

                    else

                              if (token.equals("SIZE")){


          size=Integer.parseInt(ST.nextToken());

                              }//if
```

```
                                    else
                                                if (token.equals("</FIELD>")){

            DB.getTable().getFields().add(new Field(name,type,search,size));
                                                }//if
                                                else
                                                            if
(token.equals("</TABLE>")){
                                                                        Bfile.close();
                                                                        exit=true;
                                                            }//if
                                       }//do
                                       while (exit==false);
                                    }//try
                        catch(FileNotFoundException
e){System.out.println("Exception at Server "+e.toString());}
                        catch(IOException e){System.out.println("Exception at
Server "+e.toString());}
                        databases.add(0,DB);//for demo version  only else use
"add(Object);"
            }
    // validates the database structure passed as a text file
            private boolean validateDBStructure(String file){


                        int table_count_tags=0;
                        int table_count=0;
                        int field_count_tags=0;
                        int name_count=0;
                        int type_count=0;
                        int search_count=0;
                        int other_count=0;
                        int tablename_count=0;
                        int size_count=0;


                        try{
                                    boolean exit = false;
                                    BufferedReader BR = new BufferedReader(new
FileReader(file));
                                    do{
                                                String line = BR.readLine();

                                                if (line != null){
                                                            StringTokenizer ST = new
StringTokenizer(line);
```

```
                                                        String token;
                                                        if (ST.hasMoreTokens()){

          token = ST.nextToken();
                                                        }//if
                                                        else {BR.close();return
false;}

                                                        if
(token.equals("<TABLE>")){

                                                                  table_count++;

          table_count_tags++;
                                                        }//if
                                                        else
                                                                  if
(token.equals("TABLENAME")){
                                                                                    if
((table_count_tags==1) && (table_count==1) && (field_count_tags==0) &&
(tablename_count==0)){

          tablename_count++;

if (ST.hasMoreTokens()){

String temp = ST.nextToken();

}//if

          else {BR.close();return false;}

}//if

else {BR.close();return false;}
                                                                  }//if
                                                                  else
                                                                      if
(token.equals("</TABLE>")){

          table_count_tags--;

          if(field_count_tags!=0) {BR.close();return false;}

          if(tablename_count!=1) {BR.close();return false;}

          }//if
                                                                                  else

if (token.equals("<FIELD>")){

                    if ((table_count_tags==1) && (table_count==1) &&
(tablename_count==1)){

                              field_count_tags++;

                    }//if

                    else {BR.close();return false;}
```

```
            }//if
            else
                        if (token.equals("</FIELD>")){
                                    if ((table_count_tags==1) && (table_count==1)){
                                                field_count_tags--;

                                                if
((name_count!=1)||(type_count!=1)||(search_count!=1) || (size_count!=1)){
                                                            BR.close();return false;
                                                           }

                                                else {
                                                                        name_count=0;
                                                                        type_count=0;
                                                                        search_count=0;
                                                                        size_count=0;
                                                             }
                                    }//if
                                    else {BR.close();return false;}

                        }//if
                        else
                                    if (token.equals("NAME")){

                                                if (field_count_tags==1){
                                                            name_count++;
                                                if (ST.hasMoreTokens()){
                                                                String temp =
ST.nextToken();
                                                           }//if
                                                           else {BR.close();return
false;}
                                                }//if
```

```
                                        else {BR.close();return false;}

                                }//if
                                else
                                        if (token.equals("TYPE")){
                                                String temp;

                                                if (field_count_tags==1){
                                                        type_count++;
                                                    if
(ST.hasMoreTokens()){
                                                        temp =
ST.nextToken();
                                                }//if
                                                else {BR.close();return
false;}
                                                        if
(Field.checkType(temp)!=true) {BR.close();return false;}
                                                }//if
                                                else {BR.close();return
false;}

                                        }//if
                                        else
                                                if
(token.equals("SEARCHEABLE")){
                                                        String temp;

                                                        if
(field_count_tags==1){

search_count++;
                                                            if
(ST.hasMoreTokens()){

temp = ST.nextToken();
```

```
          }//if
                                                                    else
{BR.close();return false;}
                                                                    if
(!temp.equals("Y")){
                                                                      if
(!temp.equals("N")){

             BR.close();return false;
                                                                      }
                                                                    }

                                                              }//if
                                                              else
{BR.close();return false;}

                                                        }//if
                                                        else
                                                              if
(token.equals("SIZE")){

          String temp;
                                                                    if
(field_count_tags==1){

           size_count++;

          if (ST.hasMoreTokens()){

                     temp = ST.nextToken();

          }//if

          else {BR.close();return false;}

          try{
```

```
                        int n = Integer.parseInt(temp);

                        if (!(n > 0)) {BR.close();return false;}

                        }

            catch(NumberFormatException e){BR.close();return false;}

            }//if
                                                                    }//if

                                                }//if
                                                else exit=true;
                                    }//do
                                while(exit==false);
                                if ((table_count==1) &&
(table_count_tags==0)){BR.close();return true;}
                                else {BR.close();return false;}

                                }
                        catch (FileNotFoundException
e){System.out.println("Exception at Server "+e.toString());}
                        catch (IOException e){System.out.println("Exception at
Server "+e.toString());}
                        return false;
            }

            public int getDBID(String name){
                        for(int i=0;i<databases.size();i++){
                                    SDatabase db =(SDatabase)databases.get(i);
                                    if (db.getName().equals(name)) return i;
                        }
                        return -1;
            }

            /** Reads the Structure of the Database and uses it to transform the
RecordSet object
              * received from the specific Database into a Result Object.
              * @param rs The ResultSet containig the result from the database
              * @param dbIndex The index ID of the database where the result come
from
              * @return A Result object
              */
            public Result fillResult(ResultSet rs, int dbIndex){
                        Result res=new Result();
                        SDatabase DB = getDatabase(dbIndex);
                        Vector fields = DB.getTable().getFields();
        try{
                                    int k=0;
                                    while (rs.next()) {
```

```
                                        for (int i=0;i<fields.size();i++){
                                                Field
f=(Field)fields.get(i);
                                                String name=f.getName();
                                                if
(f.getType().equals("TEXT")){

          res.addResultField(k,new
ResultField(name,f.getType(),rs.getString(name)));
                                                }//if
                                                else
                                                        if
(f.getType().equals("INT")){

          res.addResultField(k,new ResultField(name,f.getType(),new
Integer(rs.getInt(name))));
                                                        }//if
                                                        else
                                                                if
(f.getType().equals("FLOAT")){

          res.addResultField(k,new ResultField(name,f.getType(),new
Float(rs.getFloat(name))));

          }//if
                                                                else

          if (f.getType().equals("BOOLEAN")){

                    res.addResultField(k,new ResultField(name,f.getType(),new
Boolean(rs.getBoolean(name))));

          }//if

          else

                    if (f.getType().equals("GRAPHIC")){

                              byte[] b=rs.getBytes(name);

                              SImage si= new SImage();

                              if (b != null) si.bytes=b;

                              res.addResultField(k,new
ResultField(name,f.getType(),si));

                    }//if
                              }//for
                              k++;
                    }//while
                    return res;
                    }
                    catch (SQLException e){System.out.println("Exception at
Server "+e.toString());}
                    return null;
          }
```

```
//*************************************************************************
**************
           /** Creates a new database file from a text file
             * WARRING : Used only on demo version
             * @param filename The path and name of the texet file
             * @param path The Database URL
             * @param username The Username
             * @param password The Password
             * @param driver The jdbc driver
             */
           public void init(String filename,String name, String path, String
username, String password , String driver){

                                          try{

           storeDBStructure(name,filename,path,username,password , driver);
                                                    }
                                          catch(InvalidStructureException
e){System.out.println("Exception at Server "+e);}
                                          catch(Exception e
){System.out.println("Exception at Server "+e.toString());}
           }


//*************************************************************************
***************


    /** Initiate the Server and check the sets the server status */
           public void initiate(){
                      try{
                                 ObjectInputStream in = new
ObjectInputStream(new FileInputStream(config.getDBFilePath()));
                                 databases=(Vector)in.readObject();
                                 System.out.println("Database info loaded...");
                                 in.close();
                                 }
                      catch(FileNotFoundException
e){System.out.println(e);status=NO_QUERY_SERVICES;}
                      catch(ClassNotFoundException
e){System.out.println("Exception At Server : "+e);}
                      catch(IOException e){System.out.println("Exception At
Server : "+e);}

           }



    /** Runs the Server Application*/
           public void run(boolean newTxtfile, String filename,String name,
String path, String username, String password , String driver){

                      loadConfiguration();
                      System.out.println("Configuation loaded...");
                      if (newTxtfile==true) init(filename,name, path, username,
password , driver);
                      initiate();
```

```
                        System.out.println("Opening Socket ...");
                        //Opens Socket
                        try
                        {
                                    server_socket = new
ServerSocket(8888);//config.getPort());
                        }
                        catch(Exception e)
                        {
                                    System.err.println("Exception Server:");
                                    System.err.println(e);
                                    e.printStackTrace();
                        }

                        while (server_socket!=null){
                                    try{
                                                System.out.println("Server ready for
connections");
                                                Socket socket =
server_socket.accept();
                                                System.out.println("A Client has
connected.....");
                                                System.out.println("Users IP: " +
socket.getInetAddress().toString() + ", port : " + socket.getPort());
                                                ServerThread ST = new
ServerThread(this,socket);
                                                ST.start();
                                                }
                                    catch(Exception e){

            System.out.println("Exception at the Server :");

            System.out.println(e);
                                                                                    }
                        }
            }

            /** Adds a Mobile Client to the list of running ServerThread
              * @param ST The ServerThread handling the client communication
              */
            public void addMobileClient(ServerThread ST){
                        mobileThreads.add(ST);
                        ST.setThreadID(mobileThreads.indexOf(ST));

            };

            /** Adds a Central Client to the list of running ServerThread
              * @param ST The ServerThread handling the client communication
              */
            public void addCentralClient(ServerThread ST){
                        centralThreads.add(ST);
                        ST.setThreadID(centralThreads.indexOf(ST));

            };

            /** Notifys to the Server that a client has closed the connection
              * @param IP The IP address of the client
              * @param ST The server thread that served the client
              */
```

```
	public void mobileClientExit(String IP,ServerThread ST){
		System.out.println("The client "+IP+" has left...");
		mobileThreads.remove(ST);
	}

	/** Sends a Message to all Central Clients
	  * @param msg The message to be sent
	  */
	public void sendMsgToCC(Message msg){
		if (centralThreads.size()!= 0)
			for (int i=0;i<centralThreads.size();i++){

	((ServerThread)centralThreads.get(i)).sendMessage(msg);
			}

	}

	/** Sends a GIS position to all Central Clients
	  * @param GISO The GIS Object
	  */
	public void sendPosition(GISObject GISO){
		if (centralThreads.size()!= 0)
			for (int i=0;i<centralThreads.size();i++){

	((ServerThread)centralThreads.get(i)).sendPosition(GISO);
			}

	}

	/** Sends a message to Mobile Client
      * @param msg The message to be sent (Receiver information are in the
Message object)
      */
	public void sendMsgToMC(Message msg){
		if (mobileThreads.size()!=0)

	((ServerThread)mobileThreads.get(msg.getReceiver())).sendMessage(msg)
;
	}

	/** Broadcast a message to all Mobile Client
      * @param msg The message to be sent
      */
	public void broadcastMsgToMC(Message msg){
		if (mobileThreads.size()!=0)
			for (int i=0;i<mobileThreads.size();i++){

	((ServerThread)mobileThreads.get(i)).sendMessage(msg);
			}
	}

	/** Shutdown the Server and free all the threads
	  */
	public void shutServer(){
		if (centralThreads.size()!= 0)
			for (int i=0;i<centralThreads.size();i++){

	((ServerThread)centralThreads.get(i)).destroy();
			}
```

```
                    if (mobileThreads.size()!=0)
                              for (int i=0;i<mobileThreads.size();i++){

          ((ServerThread)mobileThreads.get(i)).destroy();
                              }
                    System.exit(0);

          }


}
```

2.41 ClientServerCommunication Class

```
import java.io.*;

/** The <code>ServerClientCommunication</code> interface describes the
communication
  * from the Server to the Client
  */
public interface ServerClientCommunication{
          //*****Query System*************

          /** Sends a result to a Client
            * @param result The Result to be sent
            * @param DBID The used database ID (included for future development
of the demo)
            * @param output_connection The object describing the output
connection to the client
            */
          public void sendResult(Object result, Object DBID, Object
output_connection) throws Exception;

          /** Sends the database structure to a Client
            * @param DBStructure The database structure to be sent
            * @param output_connection The object describing the output
connection to the client
            */
          public void sendDBStructure(Object DBStructure, Object
output_connection) throws Exception;

          /** Receives a Query from one of the Clients
            * @param input_connection The object describing the input
connection from the client
            * @param The Query
            */
          public Object receiveQuery(Object input_connection) throws Exception;

          /** Notifys to a CentralClient that a Mobile Client has logged in
            * @param user The User that has logged in
            * @param output_connection The object describing the output
connection to the client
            */
          public void sendUserLOGIN(Object user, Object output_connection)
throws Exception;

          /** Notifys to a CentralClient that a Mobile Client has logged out
            * @param user The User that has logged out
```

```
             * @param output_connection The object describing the output
connection to the client
             */
            public void sendUserLOGOUT(Object user, Object output_connection)
throws Exception;

            /** Receives the Action performad by the Client
             * @param input_connection The object describing the input
connection from the client
             * @param The Action
             */
            public Object readAction(Object input_connection) throws Exception;

            /** Receives a Result from one of the Central Clients
             * @param input_connection The object describing the input
connection from the client
             * @param The Result
             */
            public Object receiveResult(Object input_connection) throws
Exception;

            /** Receives an Info object from one of the Clients
             * @param input_connection The object describing the input
connection from the client
             * @param The Info about the client
             */
            public Object receiveInfo(Object input_connection) throws Exception;

            //*****Messaging System**********

            /** Receives a Message from one of the Clients
             * @param input_connection The object describing the input
connection from the client
             * @param The Message
             */
            public Object receive_message(Object input_connection) throws
Exception;

            /** Sends a Message to one of the Clients
             * @param message The Message to deliver
             * @param output_connection The object describing the output
connection to the client
             */
            public void send_message(Object message,Object output_connection)
throws Exception;

            //*****Tracking System***********

            /** Receives a GIS Position from one of the Clients
             * @param input_connection The object describing the input
connection from the client
             * @param The GIS Position
             */
            public Object receive_Position(Object input_connection) throws
Exception;

            /** Sends a GIS Position to one of the Clients
             * @param Position The GIS Position
```

```
             * @param output_connection The object describing the output
connection to the client
             */
           public void send_Position(Object Position,Object output_connection)
throws Exception;

}
```

2.42 ServerThread Class

```
import java.io.*;
import java.net.*;
import java.sql.*;

/** The <code>ServerTread</code> class is a thread run by the server in
  * order to be able to handle multiple concurrent request from the clients*/
public class ServerThread extends Thread{


           private Server server;
           private Socket socket;
           private ObjectInputStream input;
           private ObjectOutputStream output;
           private int threadID;
           private User user;
           private Info info;

           private ServerThread(){}

           private ServerClientCommunication SCCom;

           /** Creates an instance of the ServerThread Object
             * @param server Reference to the creating object
             * @param socket The communication socket assosiated to the client
             */
           public ServerThread(Server server , Socket socket){
                      this.info=new Info();
                      this.socket=socket;
                      this.server=server;
                      this.SCCom = new EBSCCommunication();

                      try{
                                 input = new
ObjectInputStream(socket.getInputStream());
                                 output= new
ObjectOutputStream(socket.getOutputStream());
                                 }
                      catch (IOException e){System.out.println("Exception at
ServerThread "+e);}
           }

           /** Sets the ID of the Thread
             * @param ID The thread ID
             */
           public void setThreadID(int ID){
                      threadID=ID;
           }
```

```
	/**------------ Only for Demo Version -----------------
	 * Used to send the DBStructure to the Client
	 * This function should be replaced by the Updating Functionality of
the system
	 */
	private void initClient(){
		try{
			SDatabase db = server.getDatabase(0);
			SCCom.sendDBStructure(db.toDatabase(),output);
			}
		catch(Exception e){System.out.println("Exception at
ServerThread : "+e);}
	}
    //*****************************************

	/** Return the information about the user connected to the Thread
	  * @return The User
	  */
	public User getUser(){
		return user;
	}

	private void sendInfo(){
		try{

			info=(Info)SCCom.receiveInfo(input);
			if (info.clientType==SysConst.CENTRAL_CLIENT){
				server.addCentralClient(this);
				server.sendUsers(threadID);
				}
			else{
				user= new
User(info.clientName,threadID);
				server.addMobileClient(this);
				server.broadcastUsersLOGIN(user);
				}

			}
		catch(Exception e){System.out.println("Exception at
ServerThread :"+e);}
	}

	/** Notifys the Client that a user has logged in
	  * @param user The user that has logged in
	  */
	public void sendUserLOGIN(User user){
		try{
			SCCom.sendUserLOGIN(user,output);
			}
		catch(Exception e){System.out.println("Exception at
ServerThread "+e);}
	}

	/** Notifys the Client that a user has logged out
	  * @param user The user that has logged out
	  */
	public void sendUserLOGOUT(User user){
		try{
			SCCom.sendUserLOGOUT(user,output);
```

```
                              }
                        catch(Exception e){System.out.println("Exception at
ServerThread "+e);}
            }


    /** Starts the Thread*/
            public void run(){
                        try{

                                    //**** Only for Demo Version
                                    initClient();
                                    //*************************

                                    while(input!=null){
                                                int
action=((Integer)SCCom.readAction(input)).intValue();
                                                switch (action){
                                                            case Action.SEND_QUERY
:{performQuery();}break;
                                                            case Action.SHUTDOWN :
{shutdown();}break;
                                                            case Action.SEND_INFO :
{sendInfo();};break;
                                                            case Action.SEND_RESULT :
{prepareResultToMobile();};break;
                                                            case Action.SEND_MESSAGE
: {getMessage();};break;
                                                            case Action.SEND_POSITION
: {getPosition();}break;
                                                }//switch
                                    }

            server.mobileClientExit(socket.getInetAddress().toString(),this);
                                    }//try
                        catch(Exception e){System.out.println("Exception at
ServerThread
1"+e);server.mobileClientExit(socket.getInetAddress().toString(),this);}
            }

            /** Prepares a result to be sent to the Mobile Client, calls the
Server.sendResult(...) function
              */
            protected void prepareResultToMobile(){
                        try{
                                    //**** In this version of the Demo the Database
ID has been ignored *****
                                    // in the final version the Result should
contain the origin Database
                                    ResultInfo
RI=(ResultInfo)SCCom.receiveResult(input);
                                    server.sendResult(RI.result,RI.CCID);
                                    }
                        catch(Exception e){System.out.println("Exception at
ServerThread u "+e);}
            }

            /** Sends a result to the Connected client
              * @param res The Result
```

```
            */
           public void sendResultToMobile(Result res){
                      //**** In this version of the Demo the Database ID has
been ignored *****
                      try{
                                 SCCom.sendResult(res,new Integer(0),output);
                                 }
                      catch(Exception e){System.out.println("Exception at
ServerThread "+e);}
           }

           /** Receives a GIS Obejct
             */
           protected void getPosition(){
                      try{
                                 GISObject
GISO=(GISObject)SCCom.receive_Position(input);
                                 server.sendPosition(GISO);
                                 }
                      catch(Exception e){}
           }

           /** Sends the position of a GIS Obejct to the connected client
             * @param GISO The GIS object
             */
           public void sendPosition(GISObject GISO){
                      try{
                                 SCCom.send_Position(GISO,output);
                                 }
                      catch(Exception e){System.out.println("GIS Exception at
ServerThread : "+e);}
           }


           /** Runs the shutdown procedure*/
           public void shutdown(){
                      try{
                                 if (info.clientType==SysConst.MOBILE_CLIENT){
                                            server.broadcastUsersLOGOUT(user);
                                            }
                                 input=null;
                                 output=null;
                                 socket.close();
                                 }
                      catch(IOException e){System.out.println("Exceptionat
ServerThread 2: "+e);}
           }



           /** Reads the query from the commuication stream, sends it to the
server. The result of
             * the query is then sent back to the connected client
             */
           public void performQuery(){
                      try{
                                 Query query = (Query)SCCom.receiveQuery(input);
                                 int DBID = server.getDBID(query.getDBName());
                                 SDatabase db = server.getDatabase(DBID);
```

```
                                    //String SQL
=(String)db.translateQuery((Query)query);
                                    db.connectToDB();
                                    Result res =
server.fillResult((ResultSet)db.sendQuery(query),0);
                                    if (res.getRows() > ResultCache.MAXLOG){
                                              res=new Result();
                                              res.setStatus(Result.TOO_BROAD);
                                    }
                                    SCCom.sendResult(res,query.getDBName(),output);
                                    db.disconnectFromDB();
                                    }
                          catch(Exception e){System.out.println("Exception at
ServerThread 3"+e);}

          }

          /** Reads a message from the communication Stream and performes the
correlated action
            */
          public void getMessage(){
                    try{
                                    Message
msg=(Message)SCCom.receive_message(input);
                                    msg.setSender(this.threadID);
                                    if (msg.getReceiver()== SysConst.NO_RECEIVER){
                                              server.sendMsgToCC(msg);
                                    }
                                    else{
                                              if (msg.getReceiver()==
SysConst.BROADCAST){

          server.broadcastMsgToMC(msg);
                                              }
                                              else server.sendMsgToMC(msg);
                                    }
                          }
                          catch(Exception e){System.out.println("Msg Exception at
ServerThread "+e);}

          }

          /** Sends a message to the connected client
            * @param msg The Message to be sent
            */
          public void sendMessage(Message msg){
                    try{
                                    SCCom.send_message(msg,output);
                                    }
                          catch(Exception e){System.out.println("Msg Exception at
Server Thread "+e);}
          }

}
```

2.43 SImage Class

```
import java.io.*;
```

```
/** The <code>SImage</code> class is the rapresentation of an image in terms of
bytes*/
public class SImage implements Serializable{
           /** The bytes forming the image*/
           byte[] bytes;
}
```

2.44 SysConst Class

```
/** The <code>SysConst</code> class holds all the System constants*/
public abstract class SysConst{

           public static final int MOBILE_CLIENT = 1;

           public static final int CENTRAL_CLIENT = 2;

           public static final int END_TRANSACTION = 3;

           public static final int END_ROW = 4;

           public static final int IMAGE_ON_STREAM = 5;

           public static final int NO_RECEIVER = -1;

           public static final int BROADCAST = -2;

}
```

2.45 Table Class

```
import java.util.*;
import java.io.*;

/** The <code>Table</code> class describes the table used in the database*/
public class Table implements Serializable{
           /** The Table name variable*/
           private String name;
           /** The array holding all the table fields objects*/
           private Vector fields;

           private Table(){};

           /** Creates a Table object with the spacific name*/
           public Table(String name){
                      this.name=name;
                      fields = new Vector();
           }

           /** Return the name of the table
             * @return The table name
             */
           public String getName(){
                      return name;
           }
    /** Compares two Table objects
      * @param obj The Table to compare
```

```
      * @return true false
      */
           public boolean equals(Object obj){
                     if (obj instanceof Table){
                               Table t=(Table)obj;
                               if (name.equals(t.getName()))
                                         if (fields.equals(t.getFields()))
return true;
                               return false;
                     }
                     else return false;
           }

           /** Returns the Vector holding the fields of the table*/
           public Vector getFields(){
                     return fields;
           }

           /** Return the one of the fields of the table stored in the
             * position described by the index
             * @return A table's Field
             */
           public Field getField(int index){
                     return (Field)fields.get(index);
           }
}
```

2.46 User Class

```
import java.io.*;

/** The <code>User</code> class represents a User of the System*/
public class User implements Serializable{
           protected int serverThreadID;
           protected String Name;

           /** Creates an empty User
             */
           public User(){}
           /** Creates a User
             * @param Name The user name
             * @param serverThreadID The ID of the Server Thread hadling the
communication with
                                        the client the user is using*/
           public User(String Name, int serverThreadID){
                     this.Name=Name;
                     this.serverThreadID=serverThreadID;
           }

           /** Returns the Name of the User
             * @return The name
             */
           public String getName(){
                     return Name;
           }

    /** Returns the Server Thread ID of Server Thread connected to the client
the user is using
      * @return The Server Thread ID
```

```
      */
           public int getServerThreadID(){
                     return serverThreadID;
           }

           /** Compares two User object and returns true is they are equal
             */
           public boolean equals(User u){
                     if ((Name.equals(u.getName())) && (serverThreadID ==
u.getServerThreadID()))
                               return true;
                     else return false;
           }
}
```

2.47 Users Class

```
import java.util.*;

/** The <code>Users</code> class is used as a container for the user's classes*/
public class Users extends Observable{
           private Vector usersVector;

           /** Creates an object of this class*/
           public Users(){
               this.usersVector= new Vector();
           }

           /** Adds a User
             * @param user The user
             */
           public void add(User user){
                     usersVector.add(user);
                     setChanged();
                     notifyObservers(usersVector);
           }

           /** Returns the Vector holding the users
             * @return The vector
             */
           public Vector getVector(){
                     return usersVector;
           }

           /** Returns a user in a specific position
             * @param index The position
             * @return The user
             */
           public User get(int index){
                     return (User)usersVector.get(index);
           }

           /** Removes a user
             * @param user The user to remove
             */
           public void remove(User user){
                     this.usersVector.remove(user);
                     setChanged();
                     notifyObservers(usersVector);
```

```
            }

            /** Returns true if the the User searched is found
              * @param name The User name
              */
            public boolean hasUser(String name){
                        for (int i=0;i<usersVector.size();i++){
                                    if ((
((User)usersVector.get(i)).getName()).equals(name))
                                                return true;
                        }
                        return false;
            }
}
```

3 Test Cases

3.1 CTSAllTests

```
import junit.framework.*;

public class CTSAllTests extends TestCase{

           public CTSAllTests(String name){
                      super(name);
           }

           public static Test suite(){
                      TestSuite suite=new TestSuite();
                      suite.addTest(ServerTest.suite());
                      suite.addTest(FieldTest.suite());
                      suite.addTest(TableTest.suite());
                      suite.addTest(DatabaseTest.suite());
                      suite.addTest(ResultTest.suite());
                      suite.addTest(EBConnectionTest.suite());
                      suite.addTest(MessageTest.suite());
                      return suite;
           }
}
```

3.2 DatabaseTest

```
import junit.framework.*;

public class DatabaseTest extends TestCase{
           private SDatabase DB1;
           private SDatabase DB2;
           private SDatabase DB3;
           private Table T1;
           private Query Q1;
           private String SQL1;

           public DatabaseTest(String name){
                      super(name);
           }

           protected void setUp(){
                      DB1= new
SDatabase("M","A","U","P","sun.jdbc.odbc.Jdbc.OdbcDriver");
                      T1=new Table("A");
                      T1.getFields().add(new Field("A","TEXT",true,5));
                      T1.getFields().add(new Field("B","TEXT",false,5));
                      T1.getFields().add(new Field("C","TEXT",true,5));
                      DB1.setTable(T1);

                      DB2= new
SDatabase("M","A","U","P","sun.jdbc.odbc.Jdbc.OdbcDriver");
                      T1=new Table("A");
                      T1.getFields().add(new Field("A","TEXT",true,5));
                      T1.getFields().add(new Field("D","TEXT",false,5));
                      T1.getFields().add(new Field("C","TEXT",true,5));
```

```
                    DB2.setTable(T1);

                    DB3= new
SDatabase("M","A","U","P","sun.jdbc.odbc.Jdbc.OdbcDriver");
                    T1=new Table("A");
                    T1.getFields().add(new Field("A","TEXT",true,5));
                    T1.getFields().add(new Field("B","TEXT",false,5));
                    T1.getFields().add(new Field("C","TEXT",true,5));
                    DB3.setTable(T1);

                    Q1= new Query("name", "Buildings");
                    Q1.addField(new QueryField("Name",Field.TEXT,"=",new
String("Mark")));
                    SQL1= new String("SELECT * FROM Buildings WHERE Name =
"+"'Mark' ");

          }

          public void QueryTranslationTest(){

                    assertEquals((String)DB1.translateQuery(Q1),SQL1);

          }

          public void equalsTest(){
                    boolean expected;
                    boolean result;

                    assertEquals(DB1,DB3);

                    expected=false;
                    result=DB1.equals(null);
                    assert(expected==result);

                    expected=false;
                    result=DB3.equals(DB2);
                    assert(expected==result);

          }

          public static Test suite(){
                    TestSuite suite = new TestSuite("Database Tests");
                    suite.addTest(new DatabaseTest("equalsTest"));
                    suite.addTest(new DatabaseTest("QueryTranslationTest"));
                    return suite;
          }
}
```

3.3 EBConnectionTest

```
import junit.framework.*;

public class EBConnectionTest extends TestCase{
          private Query Q1;
          private String SQL1;

          public EBConnectionTest(String name){
                    super(name);
          }
```

```
            protected void setUp(){
                        Q1= new Query("name", "Buildings");
                        Q1.addField(new QueryField("Name",Field.TEXT,"=",new
String("Mark")));
                        Q1.addField(new QueryField("ID",Field.TEXT,"=",new
Integer(1)));
                        SQL1= new String("SELECT * FROM Buildings WHERE Name =
'Mark'  AND ID = '1' ");

            }

            public void QueryTranslationTest(){
                        EBConnection con=new
EBConnection("sun.jdbc.odbc.Jdbc.OdbcDriver");
                        assertEquals((String)con.translateQuery(Q1),SQL1);
            }

            public static Test suite(){
                        TestSuite suite = new TestSuite("EBConnection Tests");
                        suite.addTest(new
EBConnectionTest("QueryTranslationTest"));
                        return suite;
            }
}
```

3.4 FieldTest

```
import junit.framework.*;

public class FieldTest extends TestCase{
            public FieldTest(String name){
                        super(name);
            }

    public void equalsTest(){
                        Field F1 = new Field("A","TEXT",true,5);
                        Field F2 = new Field("B","INT",true,5);
                        Field F3 = new Field("A","TEXT",true,5);

                        assertEquals(F1,F3);
                        boolean expected=false;
                        boolean result=F1.equals(null);
                        assert(expected==result);

                        expected=false;
                        result=F1.equals(F2);
                        assert(expected==result);

                        expected=false;
                        result=F3.equals(F2);
                        assert(expected==result);
            }

            public void checkTypeTest(){
                        boolean expected=true;
                        boolean result=Field.checkType("TEXT");
                        assert(expected==result);
                        expected=true;
```

```
                    result=Field.checkType("INT");
                    assert(expected==result);
                    expected=true;
                    result=Field.checkType("FLOAT");
                    assert(expected==result);
                    expected=false;
                    result=Field.checkType("TEX");
                    assert(expected==result);
          }

          public static Test suite(){
                    TestSuite suite = new TestSuite("Field Tests");
                    suite.addTest(new FieldTest("checkTypeTest"));
                    suite.addTest(new FieldTest("equalsTest"));
                    return suite;
          }
}
```

3.5 MessageTest

```
import junit.framework.*;

public class MessageTest extends TestCase{

          Message M1,M2,M3;
          User U1,U2,U3;

          public MessageTest(String name){
                    super(name);
          }

          protected void setUp(){
                    M1= new Message("Yes",1,2);
                    M2= new Message("Yes",1,2);
                    M3= new Message("Yes",3,2);
          }

          public void equalsTest(){
                    assertEquals(M1,M2);
                    boolean result = M1.equals(M3);
                    boolean expected = false;
                    assertEquals(result,expected);
          }

          public static Test suite(){
                    TestSuite suite = new TestSuite("Message Tests");
                    suite.addTest(new MessageTest("equalsTest"));
                    return suite;
          }

}
```

3.6 ResultTest

```
import junit.framework.*;

public class ResultTest extends TestCase{
          private Result R1;
          private Result R2;
```

```
            private ResultField RS1,RS2,RS3;


            public ResultTest(String name){
                                    super(name);
                        }

            protected void setUp(){
                        RS1=new ResultField("A","INT",new String("C"));
                        RS2=new ResultField("A","INT",new String("C"));
                        RS3=new ResultField("A","INT",new String("B"));
                        R1=new Result();

                        R2=new Result();
            }

            public void ResultFieldEqualsTest(){
                        assertEquals(RS1,RS2);
                        assert(!RS1.equals(RS3));
                        assert(!RS1.equals(null));
            }

            public void ResultGET_ADDResultFieldTest(){
                        R1.addResultField(0,RS1);
                        R1.addResultField(0,RS3);
                        assert(R1.getResultField(0,0).equals(RS1));
                        assert(R1.getResultField(0,1).equals(RS3));
                        ResultField result=R1.getResultField(1,2);
                        assert(result==null);


            }

            public static Test suite(){
                        TestSuite suite= new TestSuite("ResultTests");
                        suite.addTest(new ResultTest("ResultFieldEqualsTest"));
                        suite.addTest(new
ResultTest("ResultGET_ADDResultFieldTest"));
                        return suite;
            }
}
```

3.7 ServerTest

```
import junit.framework.*;
import java.io.*;
import java.util.*;

public class ServerTest extends TestCase{
            private SDatabase DB1;
            private SDatabase DB2;
            private Table T1;



            public ServerTest(String name){
                        super(name);
            }
```

```
	protected void setUp(){
			DB1= new
SDatabase("M","A","U","P","sun.jdbc.odbc.Jdbc.OdbcDriver");
			T1=new Table("Buildings");
			T1.getFields().add(new Field("ID","INT",true,5));
			T1.getFields().add(new Field("Name","TEXT",true,10));
			T1.getFields().add(new Field("Address","TEXT",false,30));
			T1.getFields().add(new Field("Remarks","TEXT",false,200));
			DB1.setTable(T1);


			DB2= new
SDatabase("M","A","s","P","sun.jdbc.odbc.Jdbc.OdbcDriver");
			T1=new Table("Build");
			T1.getFields().add(new Field("ID","INT",true,5));
			T1.getFields().add(new Field("Name","TEXT",false,10));
			T1.getFields().add(new Field("Address","TEXT",false,30));
			T1.getFields().add(new Field("Remarks","TEXT",false,200));
			DB2.setTable(T1);


	}


	public void testDatabaseStructureValidation(){
			Server myServer= new Server();
			myServer.loadConfiguration();
			try{

	myServer.storeDBStructure("M","DBStructure1.txt","","","","sun.jdbc.o
dbc.Jdbc.OdbcDriver");
				}
			catch(InvalidStructureException
e){fail("InvalidStructureException shouldn´t have been thrown");}

			try{

	myServer.storeDBStructure("M","DBStructure_test1.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
				fail("InvalidStructureException should have
been thrown");
				}
			catch(InvalidStructureException e){}

			try{

	myServer.storeDBStructure("M","DBStructure_test2.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
				fail("InvalidStructureException should have
been thrown");
				}
			catch(InvalidStructureException e){}
			try{

	myServer.storeDBStructure("M","DBStructure_test3.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
				}
```

```
                    catch(InvalidStructureException
e){fail("InvalidStructureException should´t have been thrown");}
                    try{

          myServer.storeDBStructure("M","DBStructure_test4.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test5.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test6.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test7.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test8.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test9.txt","","","","sun.j
dbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test10.txt","","","","sun.
jdbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{
```

```
          myServer.storeDBStructure("M","DBStructure_test11.txt","","","","sun.
jdbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test12.txt","","","","sun.
jdbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}
                    try{

          myServer.storeDBStructure("M","DBStructure_test13.txt","","","","sun.
jdbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}

                    try{

          myServer.storeDBStructure("M","DBStructure_test14.txt","","","","sun.
jdbc.odbc.Jdbc.OdbcDriver");
                              fail("InvalidStructureException should have
been thrown");
                              }
                    catch(InvalidStructureException e){}


          }



          public void FillInStructureTest(){
                    SDatabase expected;
                    SDatabase result;
                    Server myServer1= new Server();
                    myServer1.loadConfiguration();
                    try{

          myServer1.storeDBStructure("M","DBStructure1.txt","A","U","P","sun.jd
bc.odbc.Jdbc.OdbcDriver");
                              }
                    catch(InvalidStructureException e){}
                    expected=DB1;
                    result=myServer1.getDatabase(0);
                    assertEquals(expected,result);

                    Server myServer2= new Server();
                    myServer2.loadConfiguration();
                    try{

          myServer2.storeDBStructure("M","DBStructure_test3.txt","A","U","P","s
un.jdbc.odbc.Jdbc.OdbcDriver");
```

```
                              }
                        catch(InvalidStructureException e){}
                        result=myServer2.getDatabase(0);
                        assert(! result.equals(DB2));
                        assert(! result.equals(null));


            }

            public void WriteFileTest(){
                        SDatabase expected;
                        SDatabase result;
                        Vector databases;
                        Server myServer1= new Server();
                        myServer1.loadConfiguration();
                        try{

            myServer1.storeDBStructure("M","DBStructure1.txt","A","U","P","sun.jd
bc.odbc.Jdbc.OdbcDriver");
                              }
                        catch(InvalidStructureException e){}
                        expected=DB1;
                        try{//works onfy with the specific "database.dat" file
                                    ObjectInputStream in =new ObjectInputStream(new
FileInputStream("database.dat"));
                                    databases=(Vector)in.readObject();
                                    in.close();
                                    result=myServer1.getDatabase(0);
                                    assertEquals(expected,result);
                                    }
                    catch(FileNotFoundException e){fail("File not found");}
                    catch(IOException e){fail("IOException thrown");}
                    catch(ClassNotFoundException e){fail("class not found exception
thrown");}


            }



            public static Test suite(){
                        TestSuite suite = new TestSuite("Server Tests");
                        suite.addTest(new
ServerTest("testDatabaseStructureValidation"));
                        suite.addTest(new ServerTest("FillInStructureTest"));
                        suite.addTest(new ServerTest("WriteFileTest"));
                        return suite;
            }
}
```

3.8 TableTest

```
import junit.framework.*;
import java.util.*;

public class TableTest extends TestCase{
            private Table T1;
            private Table T2;
```

```
private Table T3;
private Table T4;

public TableTest(String name){
          super(name);
}

protected void setUp(){
          T1=new Table("A");
          T1.getFields().add(new Field("A","TEXT",true,5));
          T1.getFields().add(new Field("B","TEXT",false,5));
          T1.getFields().add(new Field("C","TEXT",true,5));
          T2=new Table("B");
          T2.getFields().add(new Field("A","TEXT",false,5));
          T2.getFields().add(new Field("B","INT",false,5));
          T2.getFields().add(new Field("C","TEXT",true,5));
          T3=new Table("A");
          T3.getFields().add(new Field("B","TEXT",false,5));
          T3.getFields().add(new Field("A","TEXT",true,5));
          T3.getFields().add(new Field("C","TEXT",true,5));
          T4=new Table("A");
          T4.getFields().add(new Field("A","TEXT",true,5));
          T4.getFields().add(new Field("B","TEXT",false,5));
          T4.getFields().add(new Field("C","TEXT",true,5));


}

public void equalsTest(){
          boolean expected;
          boolean result;

          Vector A= new Vector();
          A.add(new Field("A","TEXT",true,5));
          A.add(new Field("B","TEXT",false,5));
          A.add(new Field("C","TEXT",true,5));
          Vector B= new Vector();
          B.add(new Field("A","TEXT",true,5));
          B.add(new Field("B","TEXT",false,5));
          B.add(new Field("C","TEXT",true,5));

          assertEquals(T1,T4);
          expected=true;
          result=T1.getFields().equals(T4.getFields());
          assert(expected==result);

          expected=false;
          result=T1.equals(null);
          assert(expected==result);

          expected=false;
          result=T1.equals(T2);
          assert(expected==result);

          expected=false;
          result=T1.equals(T3);
          assert(expected==result);

}
```

```
        public static Test suite(){
                TestSuite suite= new TestSuite("Table Tests");
                suite.addTest(new TableTest("equalsTest"));
                return suite;
        }
}
```