**IT-SUBJECT:** THREAD

**TYPE:** ASSIGNMENT

**IDENTIFICATION:** TESTTHREADS

**COPYRIGHT:** *Michael Claudius*

**LEVEL:**  EASY

**TIME DURATION:** 1-2 hours

**SIZE:**  50 lines

**OBJECTIVE:** General thread understanding.

Child threads interface-references

Virtual methods

**REUIREMENTS:** Gosling Ch. 10 or BigJava ch. 19

**COMMANDS:** class Ping extends Thread

public void run()

sleep(delay\_time);

try {…} catch

(InterruptedException e){return;}

start();

public class PingPong implements Runnable

**IDENTIFICATION: THREADS/MC&LV**

*The mission*

The purpose of this assignment is to illustrate the thread-concept in Java.

*Prerequisites*

You must have read Ken Arnold & James Gosling: The Java Programming language Ch. 10 or similar literature in Cay S. Hrstman: BigJava Ch. 19.

*The objective*

The objective is to utilize the Thread class, Runnable interface and run child threads and grandchild threads.

*Program example*

public class MyText {

public MyText() {}

public void printOut(String text, int delayTime) {

try {

for (int i = 1; i < 100; i++) {

System.out.print(text + " ");

Thread.sleep(delayTime);

}

} catch (InterruptedException e) {

return;

}

}

}

public class PingPongRunnable implements Runnable {

private String text;

private int delayTime;

static public MyText myText = new MyText();

PingPongRunnable(String text, int delayTime) {

this.text = text;

this.delayTime = delayTime;

}

public void run() {

myText.printOut(text, delayTime);

{return;

} //end This thread

}

public static void main(String[] args) {

PingPongRunnable myPing = new PingPongRunnable("ping", 33);

System.out.print("Hej fra ping");

System.out.println(myPing.text);

Thread t1 = Thread(myPing);

t1.start();

System.out.println("Both Threads finished or are they...?");

}

}*Explanation*

The MyText is used as a Façade component for protection of the output medium. The printOut-method can easily be changed to use a text area on a Java GUI class. An object of this class can be viewed as a common resource for one or more threads. In order to print out, the threads should call the *printOut*-method.

The class PingPongRunnable implements the Runnable interface. The *run*-method is invoked when an object (a so called active object) of the class calls the *start*-method. Notice that PingPong has a shared resource as a static data field myText.

The *main*-method which is parent thread and acts as a coordinator for child thread t1 on myPing.

*Assigment 1: One child thread*

In Netbeans create a new project TestThreads with the classes.

Download the code of the classes from your teacher’s home page.

Check the difference between PingPong and PingPongRunnable.

Compile and run.

What is the order of the messages.

What goes on ?

*Assigment 2: Two child threads*

In *main* create an active object myPong, with a delay time on 100 milli seconds and the text Pong and use it the same way as myPing.

Compile and run.

What is the order of the messages from the main parent thread and the two child threads ?

What goes on ?

*Assigment 3: Use of join*

In order to let the parent thread (here the *main*) wait until the child threads have terminated one can use the *join*-method. In *main* add the sentences:

try

{ t1.join();

t2.join();

} catch (InterruptedException e) {return;}

before the last print out from the parent.

What is the order of the messages from the main parent thread and the two child threads ?

What goes on ?

*Critical comment. Unfortunately the messages from Ping and Pong are still mixed.*

*Assigment 4: Syncronized*

To prevent the mixing change the printOut-method must be a synchronized method

Compile and run!

*Assigment X: For the fast ones. One child and one grandchild child*

Create a new class Family which in the *run*-method is spawning a thread of the PingPongRunnable class and calling myText with the text “Mother”.

In *main* use Family to show how one can create grandchildren…..