Assignment 3 – Multithread Programming (9%)

Introduction

Assignment 3 total is 10 marks. You should begin Assignment 3 in Module 4; it is due at the end of Module 4. Check your Course Schedule for the precise due date. Directions for submitting Assignment 3 to your Open Learning Faculty Member for grading can be found in the Assignments Overview tab. An assignment marking criteria follows at the end of this document.

Instructions

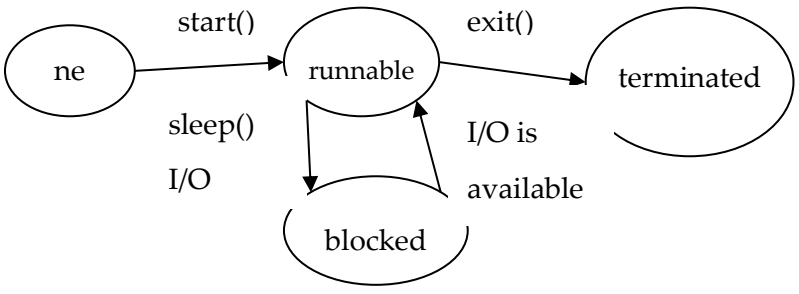
Through this assignment, you will learn how to use multiple threads in you Java application programs. The assignment is to write a multithread Java program that performs matrix multiplication. In the program, you need to calculate each element in the result matrix in a separate thread. For the convenience of programming, two sample matrices could be included in your code. However, your code should be generic with regard to matrix size. The sizes of three matrices should be stored in variable and used. The two operand matrices and the result matrix should be printed.

> You must submit Java files and screen shots that show how your program works. Refer to the marking criteria at the end of this document.

Thread and Runnable classes

The following five sections are prepared to help you understand how to use `Thread` and `Runnable` classes to implement multithread Java application programs.

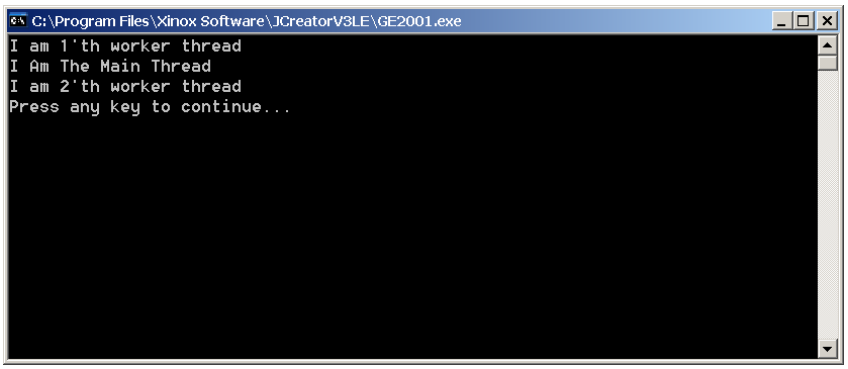
1. Java thread states



2. Two approaches:

2.1 extends Thread

```
class Worker extends Thread
{
   int id;  // not shared among threads; instance variable for an
object, i.e., thread

   Worker (int id) {
         this.id = id;
   }

   // need to be implemented; starting point of threads, when
start() is called
   public void run() {
         System.out.println("I am " + id + "'th worker thread");
   }
}

public class First
{
   public static void main(String args[]) {
         Worker runnerOne = new Worker(1);  // create a thread
         Worker runnerTwo = new Worker(2);
         runnerOne.start();  // start the thread
         runnerTwo.start();

         System.out.println("I am the main thread");
   }
}
```




```
//-----------------------------------------------------------------------------------------------
public class Second extends Thread
{
   int id;

   Second (int id) {
         this.id = id;
   }

   public void run() {
         System.out.println("I am " + id + "'th worker thread.");
   }

   public static void main(String args[]) {
         Second runnerOne = new Second(1);
         Second runnerTwo = new Second(2);
         runnerOne.start();
         runnerTwo.start();

         System.out.println("I am the main thread.");
   }
}
```

- Disadvantage
    - Cannot extend other class

2.2 implements Runnable

- Can extend other class
- Sort of multiple inheritance

- E.g., `public class ThreadApplet extends Applet implements Runnable`

```
class Worker implements Runnable
{
   int id;

   Worker (int id) {
         this.id = id;
   }

   public void run() {
         System.out.println("I am " + id + "'th worker thread.");
   }
}

public class Third
{
   public static void main(String args[]) {
         Runnable worker = new Worker(1);
         Thread runnerOne = new Thread(worker);
                  worker = new Worker(2);
         Thread runnerTwo = new Thread(worker);

         runnerOne.start();
         runnerTwo.start();

         System.out.println("I am the main thread.");
   }
}
//--------------------------------------------------------------------------
public class Fourth implements Runnable
{
   int id;

   Fourth (int id) {
         this.id = id;
   }
```

```
    public void run() {
          System.out.println("I am " + id + "'th worker thread.");
    }

    public static void main(String args[]) {
          Runnable worker = new Fourth(1);
          Thread runnerOne = new Thread(worker);
          Thread runnerTwo = new Thread(new Fourth(2));

          runnerOne.start();
          runnerTwo.start();

          System.out.println("I am the main thread.");
    }
}
```

3. Joining threads

```
public class Fifth implements Runnable
{
    int id;

    Fifth (int id) {
          this.id = id;
    }

    public void run() {
          try { Thread.sleep(1000); }  // sleep 1000 ms
          catch (InterruptedException ie) {}
          System.out.println("I am " + id + "'th worker thread.");
    }

    public static void main(String args[]) {
          Runnable worker = new Fifth(1);
          Thread runnerOne = new Thread(worker);
          Thread runnerTwo = new Thread(new Fifth(2));
```

```
        runnerOne.start();
        runnerTwo.start();

        try { runnerOne.join(); }  // wait until runnerOne
terminates
        catch(InterruptedException ie) {}

        System.out.println("The first worker done.");

        System.out.println("I am the main thread.");
    }
}
```

4. Thread cancellation

```
public class Sixth implements Runnable
{
    int id;

    Sixth (int id) {
        this.id = id;
    }

    public void run() {
        while (true) {
            try { Thread.sleep(1000); }  // sleep 1000 ms
            catch (InterruptedException ie) {  // if interrupted
                System.out.println("I (" + id + "'th worker) am
interrupted.");
                break;
            }
            System.out.println("I am " + id + "'th worker
thread.");
            if (Thread.currentThread().isInterrupted()) {  // if
interrupted
                System.out.println("I (" + id + "'th worker) am
interrupted.");
                break;
            }
        }
```

```
    }
```

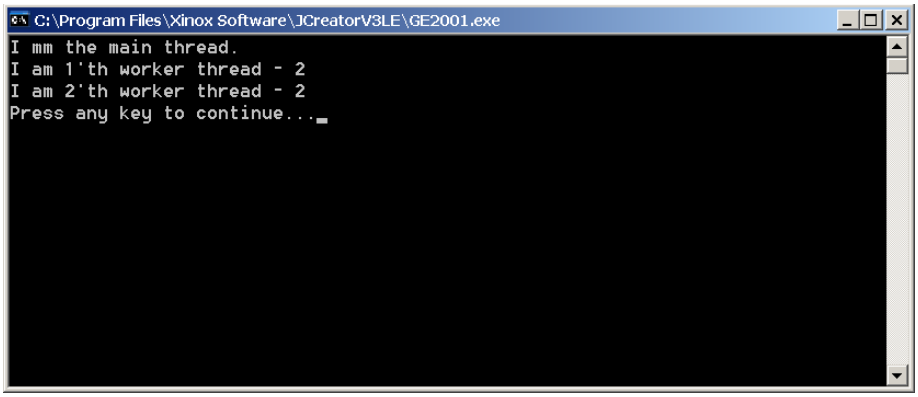
public static void main(String args[]) {

```
        Runnable worker = new Sixth(1);
        Thread runnerOne = new Thread(worker);
        Thread runnerTwo = new Thread(new Sixth(2));

        runnerOne.start();
        runnerTwo.start();

        runnerOne.stop();  // stop runnerOne; deprecated

        try { runnerOne.join(); }
        catch(InterruptedException ie) {}

        System.out.println("The first worker done.");

        runnerTwo.interrupt();  // interrupt runnerTwo

        System.out.println("I am the main thread.");
    }
}
```

5. Shared variable

```
public class Seventh implements Runnable
{
    static int common;  // shared among objects, i.e., threads
    int id;

    Seventh (int id) {
        this.id = id;
        common = id;
    }

    public void run() {
```

```
            System.out.println("I am " + id + "'th worker thread - " +
common);
    }

    public static void main(String args[]) {
            Thread runnerOne = new Thread(new Seventh(1));
            Thread runnerTwo = new Thread(new Seventh(2));

            runnerOne.start();
            runnerTwo.start();

            System.out.println("I mm the main thread.");
    }
}
```



Assignment Marking Criteria	Weighting
No syntax error: All requirements are fully implemented without syntax errors. Submitted screen shots will be reviewed with source code.	/5
Correct implementation: All requirements are correctly implemented and produce correct results Submitted screen shots will be reviewed with source code.	/5
Total	/10