5COSC019W - Tutorial 10 Exercises

1 Running a Thread

Guess what will happen when you attempt to compile and run the following code? (without actually compiling/running it)

```java
public class Background extends Thread {
    public static void main(String argv[]) {
        Background b = new Background();
        b.run();
    }

    public void start() {
        for (int i = 0; i < 10; i++) {
            System.out.println("Value of i = " + i);
        }
    }
}
```

1. A compile time error indicating that no run method is defined for the Thread class.
2. A run time error indicating that no run method is defined for the Thread class.
3. The code compiles and at run time the values 0 to 9 are printed out.
4. The code compiles but there is no output at runtime.

2 More on Problematic Threads

Guess what is wrong with the following code (without compiling/running it).

```java
class Thread1 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Hey I am thread: " + getName());

            try {
                Thread.sleep(1000);    // sleep 1 sec
            }
            catch (InterruptedException ex) {
                System.out.println("Thread 1 was interrupted");
            }
        }
    }
}

class Thread2 extends Thread {
    public void run() {
        while (true) {
            System.out.println("Hey I am thread: " + getName());

            try {
                Thread.sleep(1000);   // sleep 1 sec
            }
            catch (InterruptedException ex) {
                System.out.println("Thread 2 was interrupted");
            }
        }
    }
}

class ProblematicThreads {
    public static void main(String[] args) {
        Thread1 t1 = new Thread1();
        Thread2 t2 = new Thread2();

        t1.run();
        t2.run();
    }
}
```

Verify your guess by compiling/running the code. Fix the code so that it runs as expected.

3 Testing your knowledge on Threads

You create a class which implements `Runnable`. You have already written a `run` method for this class. You need a way to create a thread and have it execute the `run` method. Which of the following `start` methods should you use?

a)
```
public void start(){
    new Thread(this).start();
  }
```

b)
```
public void start(){
    Thread myT = new Thread();
    myT.start();
}
```

c)
```
public void start(){
    Thread myT = new Thread(this);
    myT.run();
}
```

4 Unsynchronised Threads

Consider the following code in which 20 threads are trying to increase or decrease the value of `x` so that it remains in the range $[0, 100]$. Each one of the threads checks before increasing/decreasing its value and if the operation results in out of range value then it should fail.

However, running the code shows that the range is not valid after all threads terminate.

Show how can you fix the code to run properly using `synchronized` blocks executed on a common object.

```java
class Increaser extends Thread {
    public void run() {
        if (UnsynchronisedThreads.x + 20 <= 100) {
            try {
                sleep(100);
            }
            catch (InterruptedException e) {
                System.out.println("thread increaser was interrupted");
            }

            System.out.println(getName() + " Increaser is executing " +
                                                 UnsynchronisedThreads.x);
            UnsynchronisedThreads.x = UnsynchronisedThreads.x + 20;

        }
    }
}


class Decreaser extends Thread {
    public void run() {
        if (UnsynchronisedThreads.x - 20 >= 0) {
            try {
                sleep(100);
            }
            catch (InterruptedException e) {
                System.out.println("thread decreaser was interrupted");
            }

            System.out.println(getName() + " Decreaser is executing " +
                                                 UnsynchronisedThreads.x);
            UnsynchronisedThreads.x = UnsynchronisedThreads.x - 20;

        }
    }
}

class UnsynchronisedThreads {
    public static int x = 0;   // range of x should remain within [0, 100]
    public static void main(String[] args) {
        Thread myThreads[] = new Thread[20];

        for (int i=0; i <= 9; i++) {
            myThreads[i] = new Increaser();
            myThreads[i+10] = new Decreaser();

            myThreads[i].start();
            myThreads[i+10].start();
        }


        /* Wait for all threads to finish */
        try {
            for (int i=0; i < 20; i++)
                myThreads[i].join();
        }
        catch (InterruptedException e){
            System.out.println("thread was interrupted");
            e.printStackTrace();
        }

        System.out.println("\n\n-> after all threads executed, x is: " +
                                           UnsynchronisedThreads.x);
    }
}
```

5 Challenge: Race Conditions (Data Races)

Write a program in which multiple threads add and remove elements from a `java.util.ArrayList`. Demonstrate that the list is being corrupted.