UE20CS352-OOADJ

Lab Assignment-8

Name: Vishwa Mehul Mehta

SRN: PES2UG20CS389

Section: F

Date: 28-03-2023

Problem Description:

Write a Java program that simulates a race between multiple runners. Each runner should be represented as a separate thread, and the program should output the current distance each runner has covered after each second. The distance each runner covers in each second should be determined randomly. The program should stop once one of the runners reaches a distance of 1000meters.The program should then print the top 3 runners of the race.

Code:

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Random;

public class RaceSimulator {
    public static void main(String[] args) {
        int numRunners = 5; // default number of runners
        if (args.length > 0) {
            numRunners = Integer.parseInt(args[0]);
        }
        List<Runner> runners = new ArrayList<>();
```

```
        for (int i = 1; i <= numRunners; i++) {
            runners.add(new Runner("Runner " + i));
        }
        for (Runner runner : runners) {
            runner.start(); // start each runner thread
        }
        try {
            for (Runner runner : runners) {
                runner.join(); // wait for each runner to
finish
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Collections.sort(runners,
Collections.reverseOrder()); // sort runners by distance
covered
        System.out.println("Top 3 runners:");
        for (int i = 0; i < 3; i++) {
            System.out.println((i + 1) + ". " +
runners.get(i));
        }
    }
}

class Runner extends Thread implements Comparable<Runner> {
    private static final int DISTANCE_TARGET = 1000;
    private static final int MIN_DISTANCE = 5;
    private static final int MAX_DISTANCE = 10;
    private static final Random random = new Random();
```

```
    private final String name;
    private int distanceCovered;

    public Runner(String name) {
        this.name = name;
    }

    public int getDistanceCovered() {
        return distanceCovered;
    }

    @Override
    public void run() {
        while (distanceCovered < DISTANCE_TARGET) {
            try {
                Thread.sleep(1);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            int distanceThisSecond =
random.nextInt(MAX_DISTANCE - MIN_DISTANCE + 1) +
MIN_DISTANCE;
            distanceCovered += distanceThisSecond;
            System.out.println(name + " covered " +
distanceThisSecond + " meters, total: " + distanceCovered);
        }
    }

    @Override
    public int compareTo(Runner other) {
```

```
        return Integer.compare(distanceCovered,
other.distanceCovered);
    }

    @Override
    public String toString() {
        return name + " (" + distanceCovered + "m)";
    }
}
```

Output:

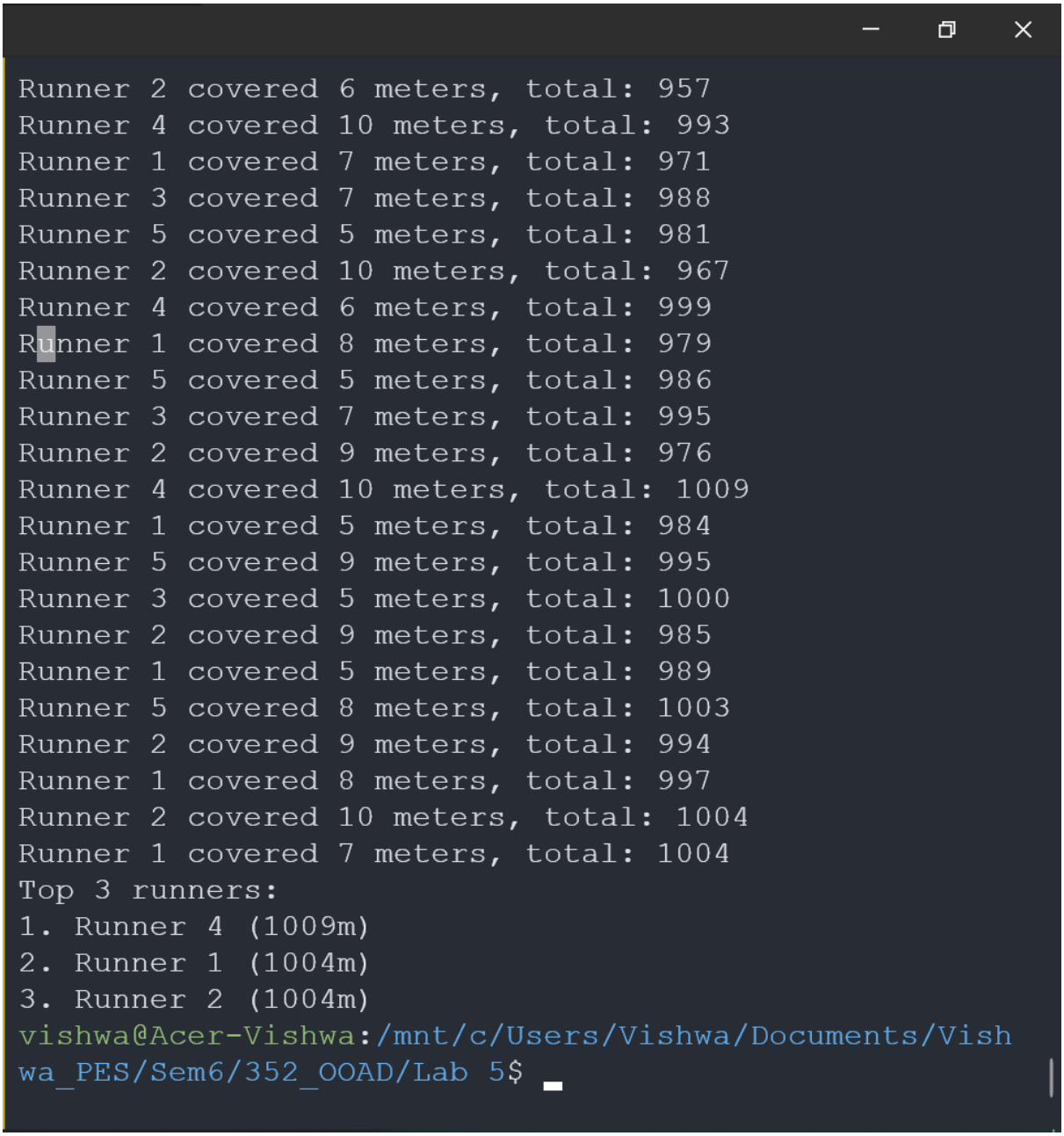
```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem6/352_OOAD/Lab 5$ java RaceSimulator 5
Runner 5 covered 10 meters, total: 10
Runner 3 covered 8 meters, total: 8
Runner 4 covered 6 meters, total: 6
Runner 2 covered 5 meters, total: 5
Runner 1 covered 9 meters, total: 9
Runner 5 covered 7 meters, total: 17
Runner 3 covered 7 meters, total: 15
Runner 1 covered 10 meters, total: 19
Runner 2 covered 5 meters, total: 10
Runner 4 covered 10 meters, total: 16
Runner 5 covered 6 meters, total: 23
Runner 3 covered 6 meters, total: 21
Runner 2 covered 5 meters, total: 15
Runner 4 covered 6 meters, total: 22
Runner 1 covered 6 meters, total: 25
Runner 5 covered 9 meters, total: 32
Runner 3 covered 9 meters, total: 30
Runner 2 covered 10 meters, total: 25
Runner 1 covered 7 meters, total: 32
Runner 4 covered 10 meters, total: 32
Runner 5 covered 7 meters, total: 39
Runner 2 covered 5 meters, total: 30
Runner 3 covered 5 meters, total: 35
Runner 4 covered 8 meters, total: 40
Runner 1 covered 10 meters, total: 42
Runner 5 covered 7 meters, total: 46
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vish
wa_PES/Sem6/352_OOAD/Lab 5$ java RaceSimulator 10
Runner 2 covered 9 meters, total: 9
Runner 10 covered 8 meters, total: 8
Runner 9 covered 6 meters, total: 6
Runner 6 covered 10 meters, total: 10
Runner 1 covered 6 meters, total: 6
Runner 3 covered 8 meters, total: 8
Runner 4 covered 9 meters, total: 9
Runner 5 covered 5 meters, total: 5
Runner 7 covered 9 meters, total: 9
Runner 8 covered 8 meters, total: 8
Runner 2 covered 10 meters, total: 19
Runner 10 covered 9 meters, total: 17
Runner 9 covered 7 meters, total: 13
Runner 6 covered 10 meters, total: 20
Runner 1 covered 5 meters, total: 11
Runner 4 covered 8 meters, total: 17
Runner 3 covered 6 meters, total: 14
Runner 5 covered 7 meters, total: 12
Runner 7 covered 8 meters, total: 17
Runner 8 covered 5 meters, total: 13
Runner 10 covered 5 meters, total: 22
Runner 2 covered 6 meters, total: 25
Runner 6 covered 5 meters, total: 25
Runner 9 covered 7 meters, total: 20
Runner 1 covered 7 meters, total: 18
Runner 4 covered 9 meters, total: 26
```

```
Runner 2 covered 10 meters, total: 983
Runner 6 covered 10 meters, total: 974
Runner 1 covered 10 meters, total: 1009
Runner 10 covered 10 meters, total: 982
Runner 8 covered 7 meters, total: 966
Runner 2 covered 6 meters, total: 989
Runner 6 covered 5 meters, total: 979
Runner 10 covered 5 meters, total: 987
Runner 8 covered 7 meters, total: 973
Runner 2 covered 8 meters, total: 997
Runner 6 covered 5 meters, total: 984
Runner 10 covered 7 meters, total: 994
Runner 8 covered 7 meters, total: 980
Runner 6 covered 7 meters, total: 991
Runner 2 covered 8 meters, total: 1005
Runner 8 covered 9 meters, total: 989
Runner 10 covered 5 meters, total: 999
Runner 6 covered 8 meters, total: 999
Runner 8 covered 10 meters, total: 999
Runner 10 covered 5 meters, total: 1004
Runner 6 covered 5 meters, total: 1004
Runner 8 covered 9 meters, total: 1008
Top 3 runners:
1. Runner 1 (1009m)
2. Runner 8 (1008m)
3. Runner 5 (1006m)
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vish
wa_PES/Sem6/352_OOAD/Lab 5$
```

```
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vishwa_PES/Sem6/352_OOAD/Lab 5$ java RaceSimulator 15
Runner 14 covered 9 meters, total: 9
Runner 2 covered 5 meters, total: 5
Runner 13 covered 10 meters, total: 10
Runner 11 covered 9 meters, total: 9
Runner 9 covered 5 meters, total: 5
Runner 1 covered 6 meters, total: 6
Runner 5 covered 6 meters, total: 6
Runner 10 covered 10 meters, total: 10
Runner 7 covered 8 meters, total: 8
Runner 12 covered 7 meters, total: 7
Runner 4 covered 10 meters, total: 10
Runner 8 covered 7 meters, total: 7
Runner 3 covered 7 meters, total: 7
Runner 14 covered 9 meters, total: 18
Runner 2 covered 6 meters, total: 11
Runner 13 covered 6 meters, total: 16
Runner 5 covered 9 meters, total: 15
Runner 11 covered 8 meters, total: 17
Runner 10 covered 8 meters, total: 18
Runner 9 covered 8 meters, total: 13
Runner 1 covered 8 meters, total: 14
Runner 4 covered 9 meters, total: 19
Runner 8 covered 9 meters, total: 16
Runner 12 covered 6 meters, total: 13
Runner 3 covered 6 meters, total: 13
Runner 7 covered 7 meters, total: 15
```

```
Runner 6 covered 9 meters, total: 973
Runner 1 covered 7 meters, total: 990
Runner 15 covered 10 meters, total: 988
Runner 2 covered 6 meters, total: 1000
Runner 12 covered 5 meters, total: 1003
Runner 7 covered 9 meters, total: 987
Runner 9 covered 6 meters, total: 990
Runner 14 covered 9 meters, total: 987
Runner 3 covered 7 meters, total: 1005
Runner 6 covered 8 meters, total: 981
Runner 1 covered 5 meters, total: 995
Runner 15 covered 7 meters, total: 995
Runner 7 covered 8 meters, total: 995
Runner 9 covered 10 meters, total: 1000
Runner 14 covered 9 meters, total: 996
Runner 6 covered 5 meters, total: 986
Runner 1 covered 10 meters, total: 1005
Runner 15 covered 10 meters, total: 1005
Runner 7 covered 5 meters, total: 1000
Runner 14 covered 9 meters, total: 1005
Runner 6 covered 7 meters, total: 993
Runner 6 covered 7 meters, total: 1000
Top 3 runners:
1. Runner 11 (1007m)
2. Runner 1 (1005m)
3. Runner 3 (1005m)
vishwa@Acer-Vishwa:/mnt/c/Users/Vishwa/Documents/Vish
wa_PES/Sem6/352_OOAD/Lab 5$
```



Summary:

Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

Threads can be created by using two mechanisms :

1. Extending the Thread class
2. Implementing the Runnable Interface

Thread creation by extending the Thread class
We create a class that extends the java.lang.Thread class. This class overrides the run() method available in the Thread class. A thread begins its life inside run() method. We create an object of our new class and call start() method to start the execution of a thread. Start() invokes the run() method on the Thread object.

Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.