**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

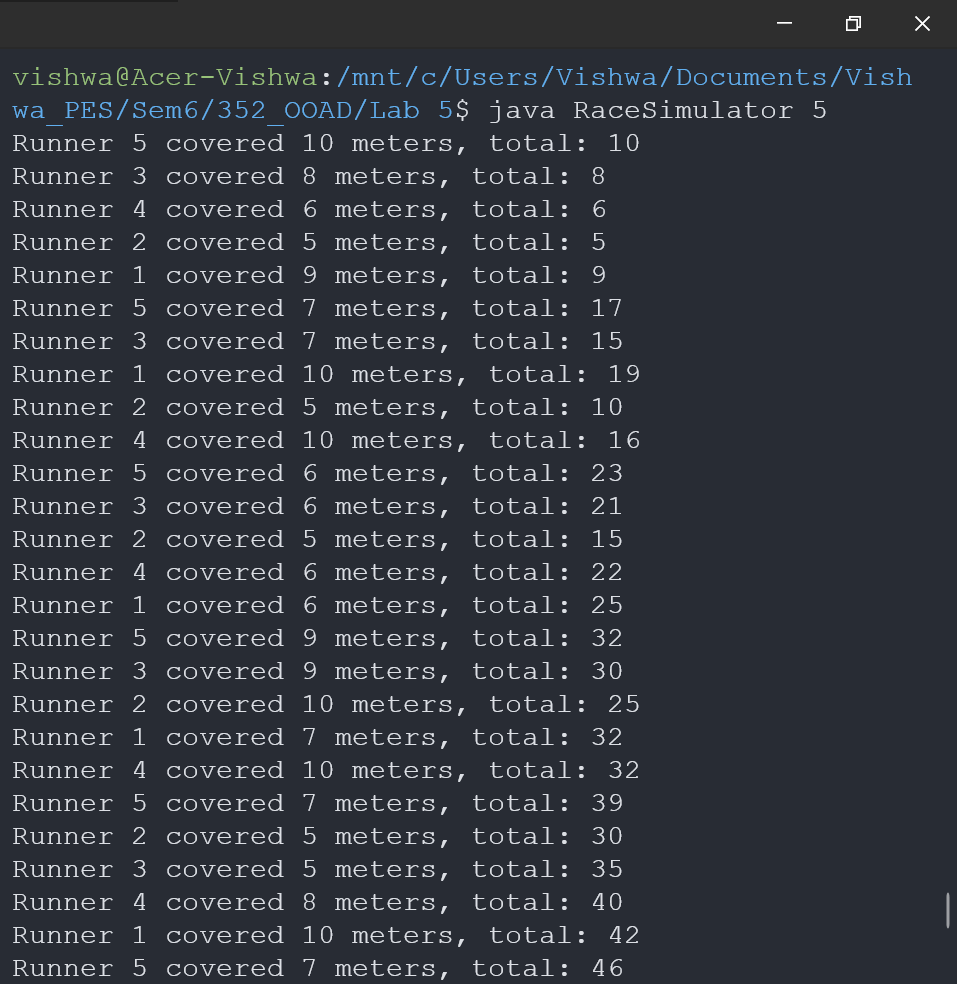
public String toString() {

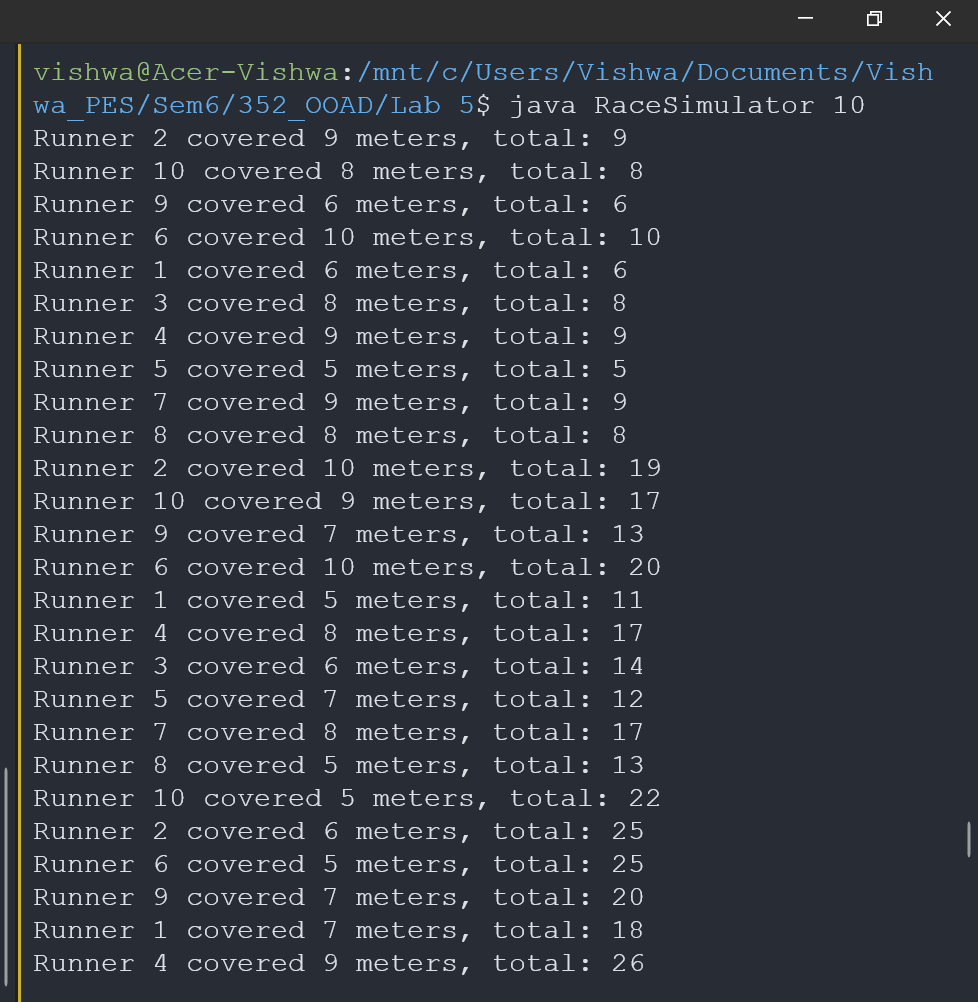
return name + " (" + distanceCovered + "m)";

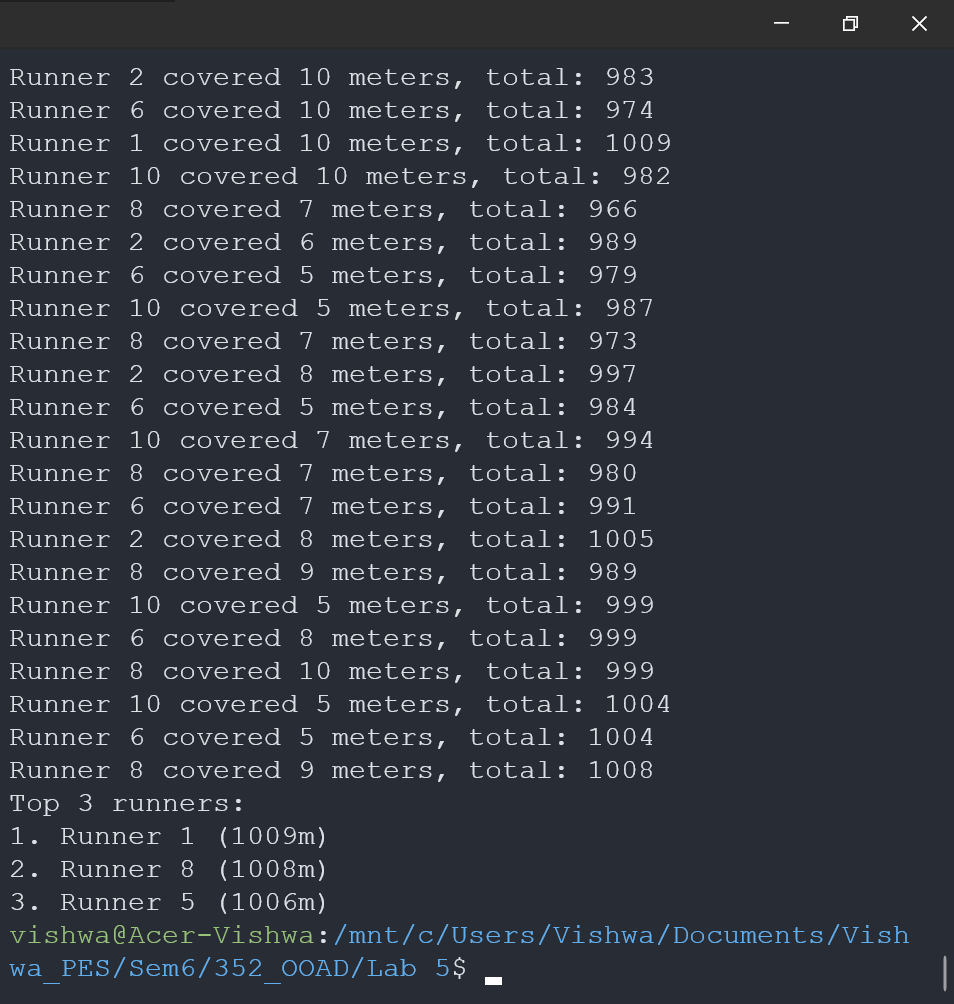
}

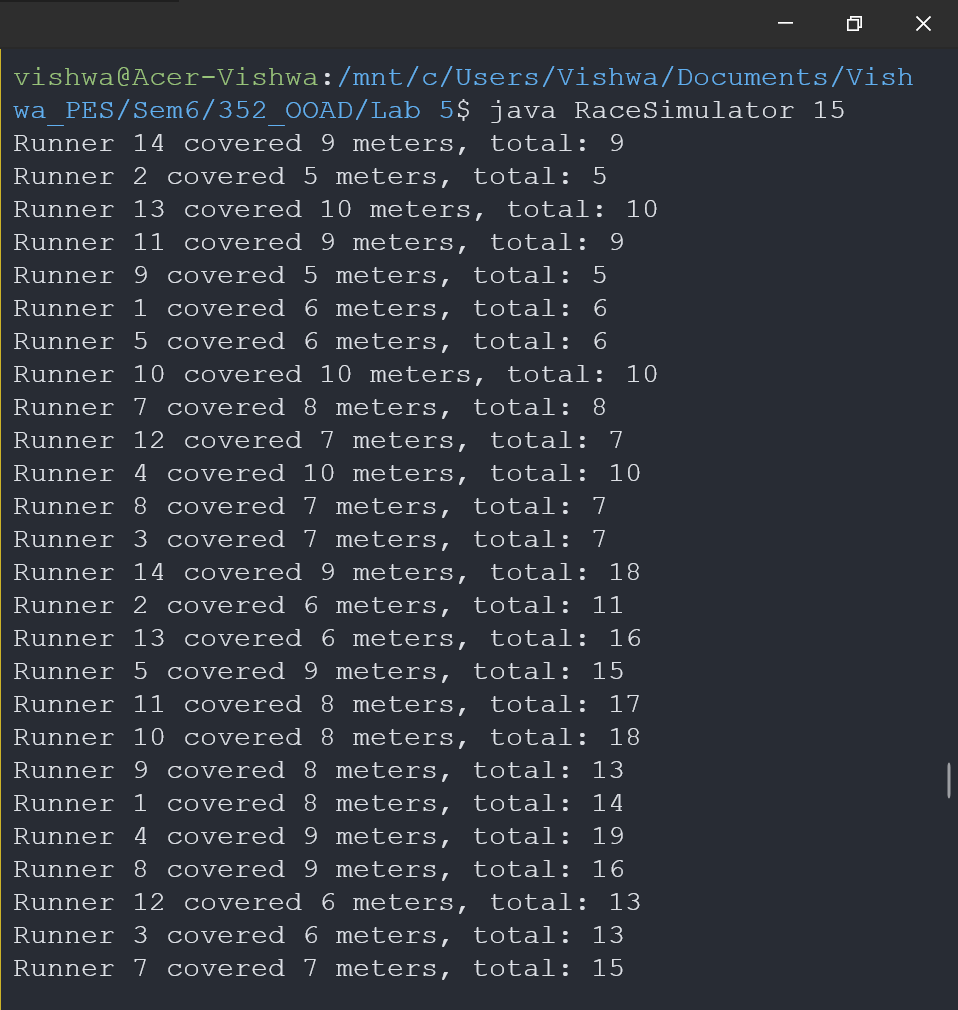
}

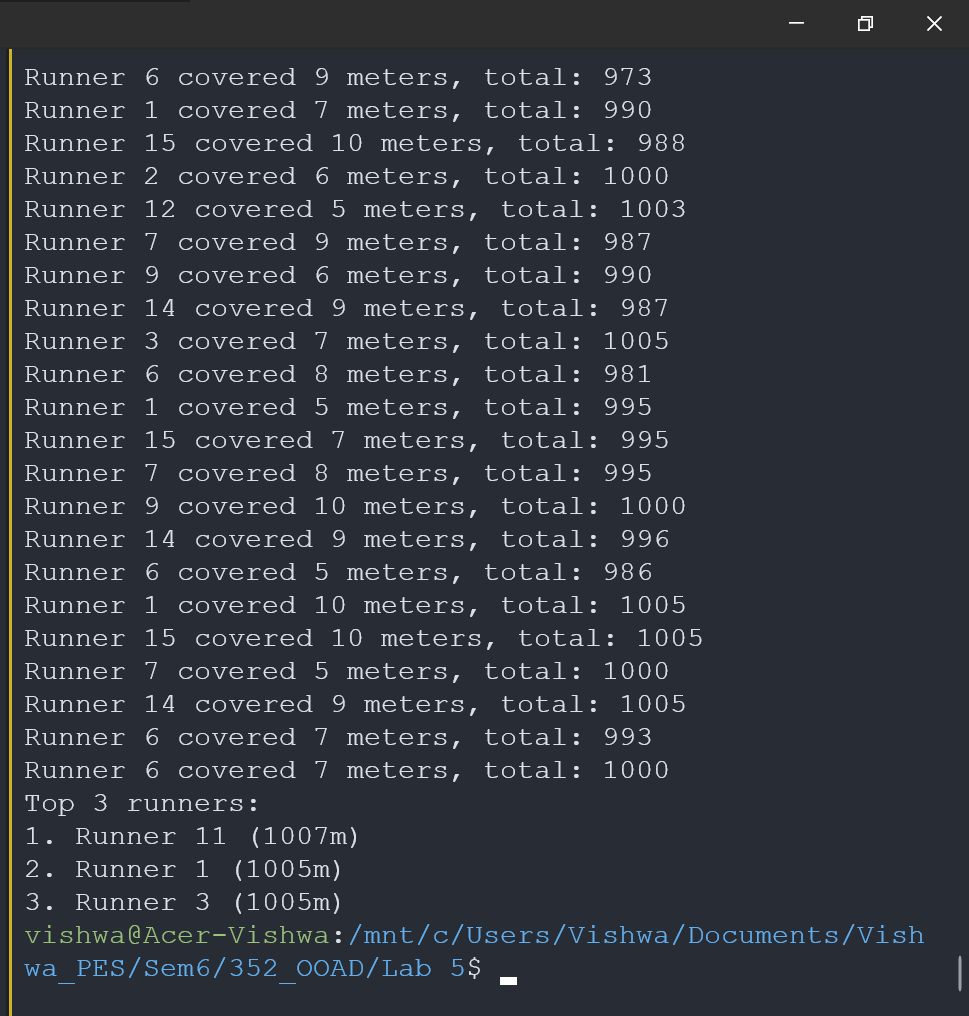
**Output:**

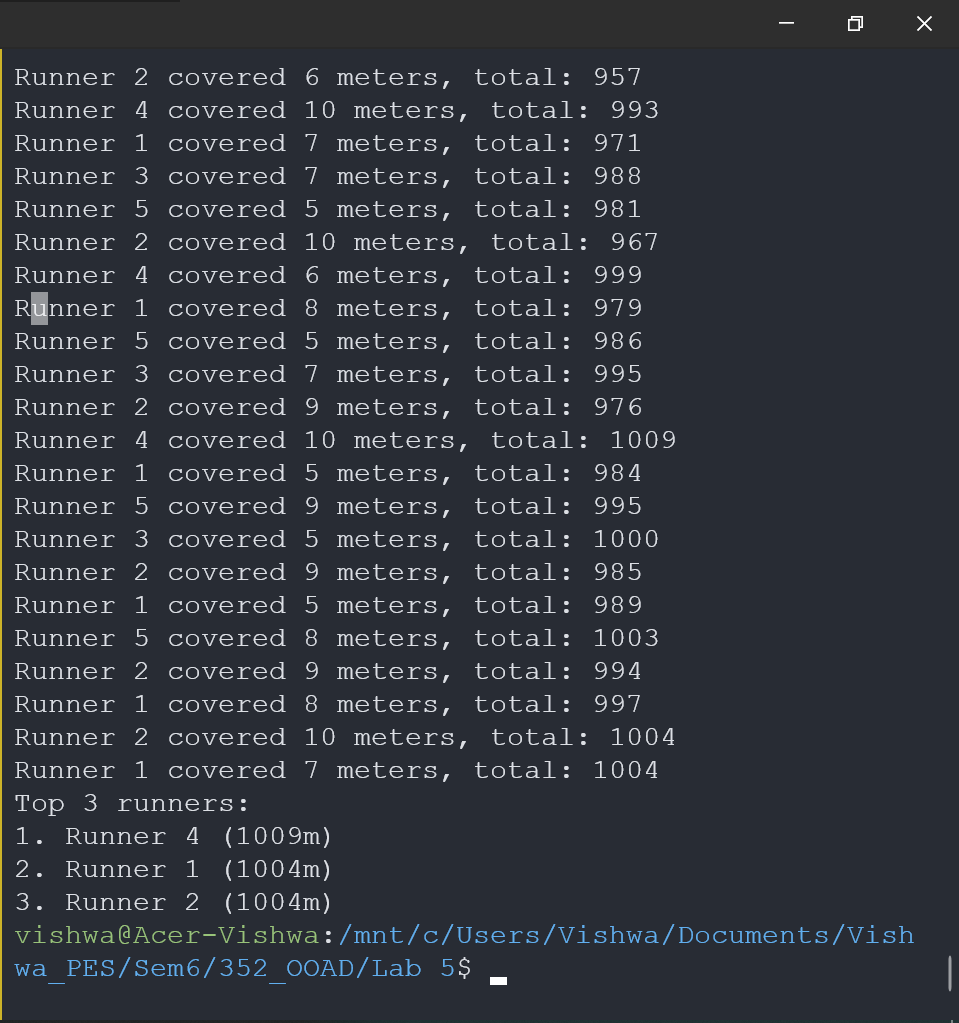












**Summary:**

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.