Object Oriented Analysis and Design using Java
Self-Learning: Multi-Threading in Java
Lab Week 8

Vanshika Goel	PES1UG20CS484	Section H	Roll No 40

Multithreading in Java is the process of running multiple threads at the same time.
Java Multithreading is commonly utilised in games, animation, and other applications.

Java provides excellent support for multithreaded programmes. The Thread class in Java provides multithreading capabilities. We can use Java Thread to create a lightweight process that performs some tasks. In our programme, we can create and start multiple threads. The Java runtime will create machine-level instructions and work with the operating system to execute them in parallel.
In an application, there are two types of threads: user threads and daemon threads. When we start an application, the primary user thread is generated. We have the ability to create multiple user threads as well as daemon threads. JVM terminates the programme once all user threads have been executed.

Threads can be generated via two mechanisms:
1. Adding to the Thread class
2. Putting the Runnable Interface into Action

The Thread Class vs. the Runnable Interface
1. Because Java does not support multiple inheritance, if we extend the Thread class, our class cannot extend any other class. Yet, by implementing the Runnable interface, we can still extend other base classes.
2. We may accomplish basic thread functionality by extending the Thread class, which has methods like yield(), interrupt(), and so on that are not available via the Runnable interface.
3. Using runnable returns an object that can be shared by several threads.

Advantages of Java Multithreading
1) It does not stall the user because threads are autonomous and can do many actions concurrently.
2) You can do multiple processes concurrently, which saves time.
3) Because threads are autonomous, an exception in one thread has no effect on other threads.

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 1000 meters. The program should then print the top 3 runners of the race.

Requirements:

1. The program should read input as to how many runners are running the race.
2. The program should create a separate thread for each runner (optionally add names for each thread to represent the runner).
3. Each thread should print the total distance run so far by the runner after each second. The distance each runner covers in each second should be determined randomly (approx. between 5 – 10 m).
4. The program should stop once one of the runners reaches a distance of 1000 meters. The program should then print the top 3 runners of the race.
5. Execute the code at least 3 times with different number of runners.

Code

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

public class Race {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Number of Runners in this Race: ");
        int n = sc.nextInt();

        List<Runner> runners = new ArrayList<>();
        for (int i = 0; i < n; i++) {
            runners.add(new Runner("Runner " + (i+1)));
        }
        for (Runner runner : runners) {
            runner.start();
        }
        try {
            for (Runner runner : runners) {
                runner.join();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
```

```
        }
        System.out.println("----------------------");
        Collections.sort(runners);
        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 String name;
    private int distance;
    private Random rand;

    public Runner(String name) {
        this.name = name;
        this.distance = 0;
        this.rand = new Random();
    }

    public void run() {
        while (distance <= 1000) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            int distanceCovered = rand.nextInt(6) + 5;
            distance += distanceCovered;
            System.out.println(name + " ran " + distance + " m.");
        }
    }

    public int compareTo(Runner other) {
        return Integer.compare(other.distance, this.distance);
    }

    public String toString() {
        return name + " ran distance: " + distance;
    }
}
```

Screenshots of Output:

For 5 runners in the race

```
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$ javac Race.java
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$ java Race
Number of Runners in this Race: 5
Runner 5 ran 9 m.
Runner 2 ran 5 m.
Runner 3 ran 5 m.
Runner 4 ran 8 m.
Runner 1 ran 6 m.
Runner 3 ran 10 m.
Runner 1 ran 15 m.
Runner 5 ran 18 m.
Runner 2 ran 13 m.
Runner 4 ran 15 m.
Runner 2 ran 22 m.
Runner 3 ran 17 m.
Runner 1 ran 20 m.
Runner 5 ran 23 m.
Runner 4 ran 21 m.
Runner 1 ran 28 m.
Runner 4 ran 30 m.
Runner 2 ran 28 m.
Runner 3 ran 22 m.
Runner 5 ran 28 m.
Runner 1 ran 36 m.
Runner 2 ran 35 m.
Runner 5 ran 38 m.
Runner 4 ran 39 m.
Runner 3 ran 29 m.
Runner 3 ran 37 m.
Runner 1 ran 45 m.
Runner 5 ran 43 m.
Runner 2 ran 40 m.
Runner 4 ran 45 m.
Runner 4 ran 51 m.
Runner 3 ran 43 m.
Runner 1 ran 52 m.
Runner 2 ran 46 m.
Runner 5 ran 53 m.
Runner 3 ran 49 m.
Runner 2 ran 55 m.
Runner 4 ran 57 m.
Runner 1 ran 61 m.
Runner 5 ran 63 m.
Runner 3 ran 57 m.
Runner 5 ran 73 m.
Runner 2 ran 65 m.
Runner 4 ran 62 m.
Runner 1 ran 70 m.
Runner 2 ran 70 m.

Runner 3 ran 989 m.
Runner 1 ran 1003 m.
Runner 5 ran 994 m.
Runner 4 ran 985 m.
Runner 3 ran 994 m.
Runner 5 ran 1000 m.
Runner 4 ran 995 m.
Runner 3 ran 1002 m.
Runner 5 ran 1007 m.
Runner 4 ran 1000 m.
Runner 4 ran 1007 m.
----------------------
Top 3 Runners:
1. Runner 4 ran distance: 1007
2. Runner 5 ran distance: 1007
3. Runner 2 ran distance: 1005
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$
```

For 7 Runners in the race:

```
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$ java Race
Number of Runners in this Race: 7
Runner 1 ran 10 m.
Runner 4 ran 9 m.
Runner 3 ran 7 m.
Runner 5 ran 9 m.
Runner 6 ran 7 m.
Runner 2 ran 5 m.
Runner 7 ran 6 m.
Runner 3 ran 15 m.
Runner 4 ran 14 m.
Runner 2 ran 13 m.
Runner 6 ran 16 m.
Runner 7 ran 16 m.
Runner 1 ran 17 m.
Runner 5 ran 18 m.
Runner 3 ran 24 m.
Runner 7 ran 25 m.
Runner 5 ran 26 m.
Runner 1 ran 24 m.
Runner 2 ran 18 m.
Runner 4 ran 22 m.
Runner 6 ran 21 m.
Runner 7 ran 34 m.
Runner 3 ran 30 m.
Runner 2 ran 24 m.
Runner 5 ran 33 m.
Runner 6 ran 29 m.
Runner 1 ran 32 m.
Runner 4 ran 27 m.
Runner 7 ran 42 m.
Runner 1 ran 39 m.
Runner 3 ran 37 m.
Runner 2 ran 30 m.
Runner 4 ran 33 m.
Runner 6 ran 36 m.
Runner 5 ran 39 m.
Runner 2 ran 37 m.
Runner 7 ran 50 m.
Runner 1 ran 45 m.
Runner 4 ran 42 m.
Runner 6 ran 45 m.
Runner 3 ran 47 m.
Runner 5 ran 45 m.
Runner 3 ran 57 m.
Runner 7 ran 58 m.
Runner 1 ran 55 m.
Runner 6 ran 53 m.
Runner 2 ran 42 m.
Runner 4 ran 51 m.
Runner 5 ran 51 m.
Runner 6 ran 63 m.
Runner 6 ran 980 m.
Runner 5 ran 990 m.
Runner 2 ran 1003 m.
Runner 7 ran 990 m.
Runner 3 ran 969 m.
Runner 6 ran 985 m.
Runner 5 ran 998 m.
Runner 7 ran 998 m.
Runner 3 ran 975 m.
Runner 6 ran 994 m.
Runner 5 ran 1004 m.
Runner 7 ran 1004 m.
Runner 6 ran 999 m.
Runner 3 ran 980 m.
Runner 6 ran 1006 m.
Runner 3 ran 985 m.
Runner 3 ran 991 m.
Runner 3 ran 1000 m.
Runner 3 ran 1008 m.
--------------------
Top 3 Runners:
1. Runner 4 ran distance: 1009
2. Runner 3 ran distance: 1008
3. Runner 6 ran distance: 1006
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$
```

For 3 Runners in the race:

```
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$ java Race
Number of Runners in this Race: 3
Runner 3 ran 6 m.
Runner 1 ran 8 m.
Runner 2 ran 10 m.
Runner 1 ran 15 m.
Runner 2 ran 15 m.
Runner 3 ran 11 m.
Runner 1 ran 22 m.
Runner 2 ran 25 m.
Runner 3 ran 17 m.
Runner 2 ran 34 m.
Runner 1 ran 30 m.
Runner 3 ran 23 m.
Runner 3 ran 29 m.
Runner 2 ran 41 m.
Runner 1 ran 36 m.
Runner 3 ran 37 m.
Runner 1 ran 41 m.
Runner 2 ran 51 m.
Runner 3 ran 42 m.
Runner 1 ran 47 m.
Runner 2 ran 61 m.
Runner 1 ran 55 m.
Runner 3 ran 47 m.
Runner 2 ran 66 m.
Runner 2 ran 71 m.
Runner 1 ran 60 m.
Runner 3 ran 57 m.
Runner 2 ran 77 m.
Runner 1 ran 68 m.
Runner 3 ran 64 m.
Runner 2 ran 87 m.
Runner 1 ran 74 m.
Runner 3 ran 74 m.
Runner 2 ran 95 m.
Runner 1 ran 83 m.
Runner 3 ran 83 m.
Runner 2 ran 103 m.
Runner 1 ran 89 m.
Runner 3 ran 89 m.
Runner 2 ran 108 m.
Runner 3 ran 96 m.
Runner 1 ran 95 m.
Runner 2 ran 115 m.
Runner 3 ran 104 m.
Runner 1 ran 105 m.
Runner 2 ran 123 m.
Runner 1 ran 113 m.
Runner 3 ran 110 m.
Runner 2 ran 132 m.
Runner 1 ran 120 m.

Runner 1 ran 1004 m.
Runner 3 ran 948 m.
Runner 2 ran 976 m.
Runner 3 ran 953 m.
Runner 2 ran 982 m.
Runner 3 ran 958 m.
Runner 2 ran 988 m.
Runner 3 ran 963 m.
Runner 2 ran 993 m.
Runner 3 ran 968 m.
Runner 2 ran 999 m.
Runner 3 ran 974 m.
Runner 2 ran 1008 m.
Runner 3 ran 982 m.
Runner 3 ran 991 m.
Runner 3 ran 997 m.
Runner 3 ran 1007 m.
--------------------
Top 3 Runners:
1. Runner 2 ran distance: 1008
2. Runner 3 ran distance: 1007
3. Runner 1 ran distance: 1004
Vanshikas-MacBook-Air /Users/vanshikagoel/Desktop/OOAD/Lab/Week 8$
```