

---

# **UFAZ OOP2 MULTITHREADING MINI-PROJECT REPORT**

---

October 8, 2018

Prepared by: Orxan Hashimli and Afrasiyab Khalili  
University: UFAZ

## TASK:

1. Write a **Counter** class that inherits from the **Thread** class; it has a String type attribute; its run() method counts from 1 to n by making a random pause of 0 to 5 seconds between two increments, it displays each incremented value with its name and then displays an end message. Test this class in a **TestCounter** class that starts several **Counter** objects.

## SOLUTION:

We have 3 classes:

- (a) **Counter** - it is a class which extends the Thread.  
So, we have two methods in our class:
  - i. **Counter()** - it is a constructor that takes two parameters( name of String and counter ).
  - ii. **run()** - it is a start point of Thread after calling **Thread.start()** method.
- (b) **TestCounter** - it is a main Class. We instantiate as many instance as we can. after instantiation we can awake our Thread objects just using **Thread.start()** method. For simplicity, we use **Thread.join()** method because we want our threads to wait each other. Of course we do this in **try, catch** block to know that there is mistake or not.
- (c) **Synchronized\_Counter** - It is doing the main task(displaying the string and incremented value).  
We use **for** loop inside the run() method then **try, catch** blocks to do that. To make a pause between increments we use **Thread.sleep()** method.  
Output will be like:

```

eclipse-workspace - AJP/src/Mini_Project_MultiThreading_aa/TestCounter.java - Eclipse
package Mini_Project_MultiThreading_aa;
import java.util.Scanner;

public class TestCounter {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("give the counter:");
        int n = scan.nextInt();
        Synchronized_counter a = new Synchronized_counter(n);
        Thread test1 = new Counter(a, "Afrasiyab");
        Thread test2 = new Counter(a, "Khalili");
        Thread test3 = new Counter(a, "Orxan");
        Thread test4 = new Counter(a, "Hashimli");
        test1.start();
        test2.start();
        test3.start();
        test4.start();
        try {
            test1.join();
            test2.join();
            test3.join();
            test4.join();
        }
        catch (Exception e) {}
        scan.close();
    }
}

<terminated> TestCounter (3) [Java Application] /usr/lib/jvm/java-8-or
give the counter:
4
0: Afrasiyab
0: Hashimli
0: Khalili
1: Afrasiyab
2: Afrasiyab
1: Khalili
0: Orxan
1: Hashimli
3: Afrasiyab
Afrasiyab: end in order 1
2: Khalili
1: Orxan
2: Hashimli
2: Orxan
3: Khalili
Khalili: end in order 2
3: Hashimli
Hashimli: end in order 3
3: Orxan
Orxan: end in order 4

```

8-10-08 08-47-31.png

2. Modify the **run()** method of the **Counter** class so that the thread displays the end message with its order of arrival. Test the change.

## SOLUTION:

Generally, we have 3 classes in this exercise.

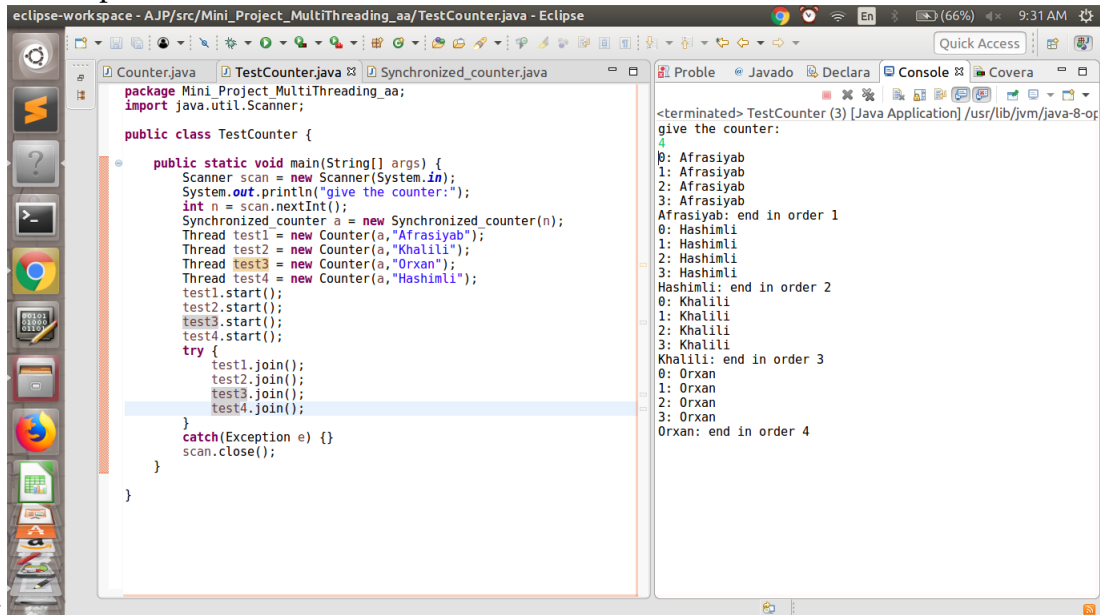
- (a) **Synchronized\_Counter** - it is our additional class that takes **counter** from the user.
- (b) **Counter** - it is a class that inherits **Thread** class.
- (c) **TestCounter** - it is our main class.

Procedure:

First, we are passing a value to the input of **constructor of Synchronized\_Counter**. Then I have another method in this class which is called **counter()** and it is **synchronized**. The main difference between (1) and (2) is that we are not using **synchronized** in our method (Picture.1), therefore we get output without order. So in the **Counter** class we create an object of **Synchronized\_Counter** class and we have the **run()** method that calls **counter()** method each time. Finally, In the Main, We instantiate an object which is instance of **Synchronized\_Counter** class, then several objects which are the instances of

Counter class and we give as a input instance of Synchronized\_Counter class when we instantiate Counter objects and then calling Thread.start(),Thread.join() with try,catch block.

Output will be like:



The screenshot shows the Eclipse IDE with a project named 'Mini\_Project\_MultiThreading\_aa'. The main editor displays the 'TestCounter.java' file. The code defines a 'Counter' class and a 'TestCounter' class. The 'TestCounter' class has a 'main' method that creates a 'Synchronized\_Counter' object and four 'Counter' objects, each associated with a thread. The threads are started and then joined in a try-catch block. The console on the right shows the output of the program, which lists the names of the threads and their completion order.

```
package Mini_Project_MultiThreading_aa;
import java.util.Scanner;

public class TestCounter {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("give the counter:");
        int n = scan.nextInt();
        Synchronized_Counter a = new Synchronized_Counter(n);
        Thread test1 = new Counter(a, "Afrasiyab");
        Thread test2 = new Counter(a, "Khalili");
        Thread test3 = new Counter(a, "Orxan");
        Thread test4 = new Counter(a, "Hashimli");
        test1.start();
        test2.start();
        test3.start();
        test4.start();
        try {
            test1.join();
            test2.join();
            test3.join();
            test4.join();
        }
        catch (Exception e) {}
        scan.close();
    }
}
```

<terminated> TestCounter (3) [Java Application] /usr/lib/jvm/java-8-or  
give the counter:  
4  
0: Afrasiyab  
1: Afrasiyab  
2: Afrasiyab  
3: Afrasiyab  
Afrasiyab: end in order 1  
0: Hashimli  
1: Hashimli  
2: Hashimli  
3: Hashimli  
Hashimli: end in order 2  
0: Khalili  
1: Khalili  
2: Khalili  
3: Khalili  
Khalili: end in order 3  
0: Orxan  
1: Orxan  
2: Orxan  
3: Orxan  
Orxan: end in order 4

-08 09-31-17.png