OOPM Lab
Lab Assingment number 09

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Write a JAVA program to demonstrate the concept of multithreading and synchronization.

Problem statement:

Create two threads such that one thread will print first 10 even natural numbers (0,2,4..) and another will print first 10 odd natural numbers(1,3,5..) in an ordered fashion.

To implement this you can use any of the following:

i. Synchronized method
ii. Synchronized block
iii. Static synchronization

Theory:

1. What is thread? How is it different from process?
   - **a.** A thread is a series of executed statements
   - **b.** A thread is a nested sequence of method calls
   - **c.** Its shares memory, files and per-process state
   - **d.** Process means any program is in execution. Process control block controls the operation of any process. Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc. A process can creates other processes which are known as Child Processes. Process takes more time to terminate and it is isolated means it does not share memory with any other process, while

     Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread have 3 states: running, ready, and blocked.

     Thread takes less time to terminate as compared to process and like process threads do not isolate.

2. What are the ways to implement multithreading? Give syntax.

   There are two ways to create a thread.

   - **a.** It can be created by extending the Thread class and overriding its run() method:

```
public class MyClass extends Thread {
    public void run() {
        System.out.println("This code is running in a thread");
    }
}
```

   - **b.** Another way to create a thread is to implement the Runnable interface:

```
public class MyClass implements Runnable {
```

```
    public void run() {
        System.out.println("This code is running in a thread");
    }
}
```

3. Define synchronization.
   a. Synchronization in java is the capability to control the access of multiple threads to any shared resource. Java Synchronization is better option where we want to allow only one thread to access the shared resource.
   b. The synchronization is mainly used to
      1. To prevent thread interference.
      2. To prevent consistency problem.

4. How to achieve synchronization of threads?

   There are two types of thread synchronization mutual exclusive and inter-thread communication.

   a. Mutual Exclusive
      1. Synchronized method.
      2. Synchronized block.
      3. static synchronization.
   b. Cooperation (Inter-thread communication in java)

   A) Synchronised method

   To make a method synchronized, add the synchronized keyword to its declaration

   B) Synchronised block

   You do not have to synchronize a whole method. Sometimes it is preferable to synchronize only part of a method. Java synchronized blocks inside methods makes this possible.

   C) Static Synchronisation

   Synchronized static methods are synchronized on the class object of the class the synchronized static method belongs to. Since only one class object exists in the Java VM per class, only one thread can execute inside a static synchronized method in the same class

5. Advantages of Multithreading.
   a. Improved throughput. Many concurrent compute operations and I/O requests within a single process.
   b. Simultaneous and fully symmetric use of multiple processors for computation and I/O
   c. Superior application responsiveness. If a request can be launched on its own thread, applications do not freeze or show the "hourglass". An entire application will not block, or otherwise wait, pending the completion of another request.
   d. Improved server responsiveness. Large or complex requests or slow clients don't block other requests for service. The overall throughput of the server is much greater.

e. Minimized system resource usage. Threads impose minimal impact on system resources. Threads require less overhead to create, maintain, and manage than a traditional process.

Program:

```
//code
import java.util.*;
import java.lang.*;

class Numbers {
   synchronized void printNumbers(int n) {
      for (int i=1 ; i<=10 ; i++) {
         System.out.println(n++);
         n++;
         try {
            Thread.sleep(400);
         }
         catch (Exception e) {
            System.out.println(e);
         }
      }
   }
}

class MyThread1 extends Thread {
   Numbers t;
   public MyThread1(Numbers t) {
      this.t = t;
   }
   public void run() {
      t.printNumbers(0);
   }
}

class MyThread2 extends Thread {
   Numbers t;
   public MyThread2(Numbers t) {
      this.t = t;
   }
   public void run() {
      t.printNumbers(1);
   }
}

public class ImplementationOfThreads {
   public static void main(String args[]) {
      Numbers obj = new Numbers();
      MyThread1 t1 = new MyThread1(obj);
      MyThread2 t2 = new MyThread2(obj);
      t1.start();
      t2.start();
   }
```

}

// output

```
E:\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 09\Code>javac ImplementationOfThreads.java

E:\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 09\Code>java ImplementationOfThreads
0
2
4
6
8
10
12
14
16
18
1
3
5
7
9
11
13
15
17
19

E:\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 09\Code>java ImplementationOfThreads
0
2
4
6
8
10
12
14
16
18
1
3
5
7
9
11
13
15
17
19

E:\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 09\Code>
```