OOP Lab: Experiment 9

Submitted By: Aryan Saxena
Batch: B1
SAP Id: 500082431
Roll No.: R214220274

Exercise 1: Write a program to implement the concept of threading by extending Thread Class and Runnable interface.

Code:

```java
class ThreadExtend extends Thread{
    public void run() {
        System.out.println("Thread Class is Running");
    }
}

class ThreadExtend2 implements Runnable{
    public void run() {
        System.out.println("Runnable Thread Class is running");
    }
}

public class Q1Thread {

    public static void main(String[] args) {
        ThreadExtend TE1 = new ThreadExtend();
        TE1.start();
        ThreadExtend2 Obj = new ThreadExtend2();
        Thread TE2 = new Thread(Obj);
        TE2.start();
    }

}
```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academic
em3-Java-OOP\LAB\Experiment 9\" ; if ($?) { javac Q1Thread.java } ; if ($?) { java Q1Threa
Thread Class is Running
Runnable Thread Class is running
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 9>
```

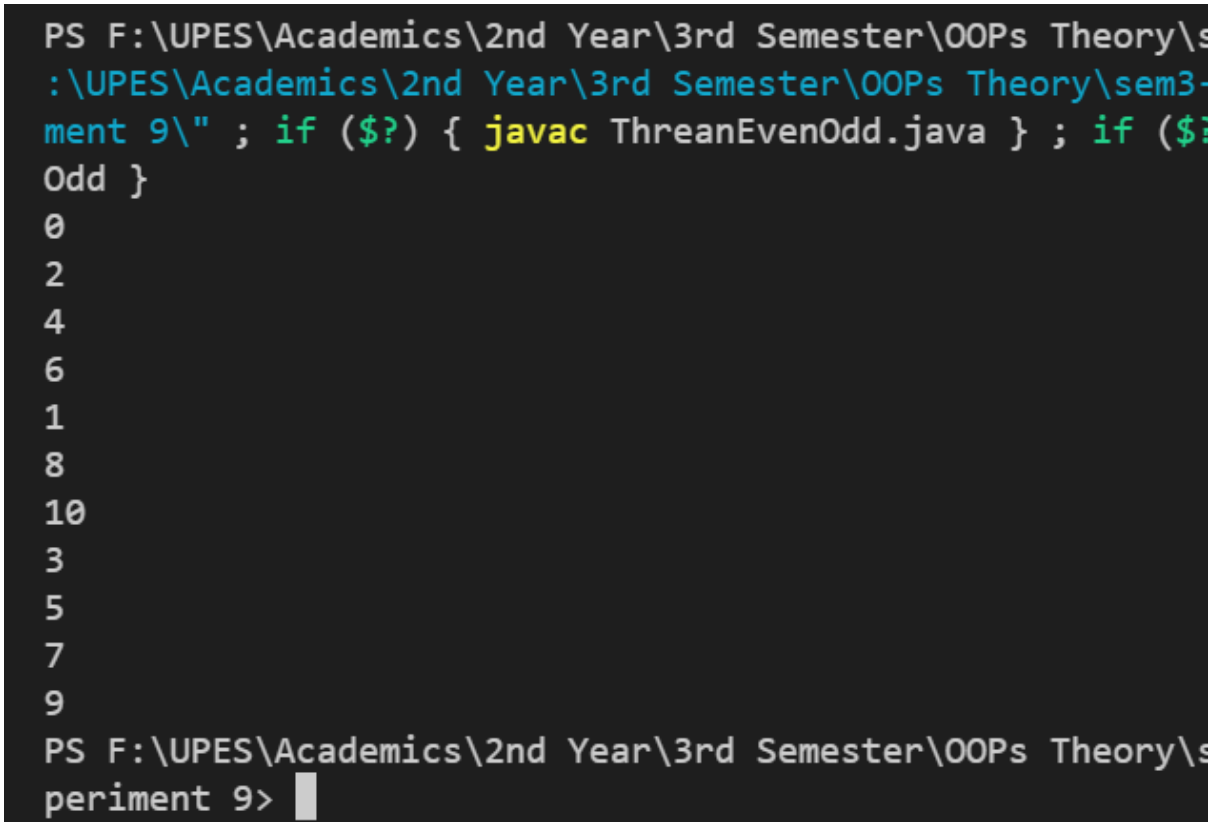
Exercise 2: Write a program for generating 2 threads, one for printing even numbers and the otherfor printing odd numbers.

Code:

```
class EvenThread extends Thread{
    public void run() {
        for(int i = 0; i<=10; i++)
        {
            if(i%2==0)
                System.out.println(i);
        }
    }
}

class OddThread extends Thread{
    public void run() {
        for(int i = 0; i<=10; i++)
        {
            if(i%2!=0)
                System.out.println(i);
        }
    }
}
public class ThreanEvenOdd
{
    public static void main(String[] args)
    {
        EvenThread TE = new EvenThread();
        TE.start();
        OddThread TO = new OddThread();
        TO.start();
    }
}
```

Output:


PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\s
:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-
ment 9\" ; if ($?) { javac ThreanEvenOdd.java } ; if ($
Odd }
0
2
4
6
1
8
10
3
5
7
9
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\s
periment 9>

Exercise 3: Write a program to launch 10 threads. Each thread increments a counter variable. Run the program with synchronization.

Code:

```
class CountingThread extends Thread
{
    static int counter = 0;
    public void run() {
        counter = inc(counter);
    }
    public synchronized int inc(int i)
      {
        i++;
        return i;
    }
}

public class ThreadSync
{
    public static void main(String[] args)
    {
        CountingThread TE1 = new CountingThread();
        CountingThread TE2 = new CountingThread();
        CountingThread TE3 = new CountingThread();
        CountingThread TE4 = new CountingThread();
        CountingThread TE5 = new CountingThread();
        CountingThread TE6 = new CountingThread();
        CountingThread TE7 = new CountingThread();
        CountingThread TE8 = new CountingThread();
        CountingThread TE9 = new CountingThread();
        CountingThread TE10 = new CountingThread();
        TE1.start();
        TE2.start();
        TE3.start();
        TE4.start();
        TE5.start();
        TE6.start();
        TE7.start();
        TE8.start();
        TE9.start();
        TE10.start();
        System.out.println(CountingThread.counter);
    }
}
```

Output:

```
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP> cd "f:\UPES\Academics\2nd
f ($?) { javac ThreadSync.java } ; if ($?) { java ThreadSync }
9
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs Theory\sem3-Java-OOP\LAB\Experiment 9>
```

Exercise 4: Write a Java program to create five threads with different priorities. Send two threads of the highest priority to sleep state. Check the aliveness of the threads and mark which thread is long lasting.

Code:

```
class ThreadClass extends Thread
{
    public void run() {};
}



public class ThreadLive
{
    public static void main(String[] args)
    {

        ThreadClass TE1 = new ThreadClass();
        ThreadClass TE2 = new ThreadClass();
        ThreadClass TE3 = new ThreadClass();
        ThreadClass TE4 = new ThreadClass();
        ThreadClass TE5 = new ThreadClass();
        TE1.setPriority(Thread.NORM_PRIORITY + 5);
        TE2.setPriority(Thread.MIN_PRIORITY);
        TE3.setPriority(Thread.NORM_PRIORITY + 3);
        TE4.setPriority(Thread.NORM_PRIORITY - 3);
        TE5.setPriority(Thread.NORM_PRIORITY + 4);
        TE1.start();
        TE2.start();
        TE3.start();
        TE4.start();
        TE5.start();
        System.out.println("Is thread one alive? " + TE1.isAlive());
        System.out.println("Is thread one alive? " + TE2.isAlive());
        System.out.println("Is thread one alive? " + TE3.isAlive());
        System.out.println("Is thread one alive? " + TE4.isAlive());
        System.out.println("Is thread one alive? " + TE5.isAlive());
        try {
            TE1.sleep(1000);
            TE5.sleep(1000);
        }
        catch(InterruptedException e) {
            System.out.println(e);
        }
        TE1.stop();
        TE2.stop();
        TE3.stop();
        TE4.stop();
        TE5.stop();
    }

}
```

Output:

```
Note: ThreadLive.java uses or overrides a deprec
Note: Recompile with -Xlint:deprecation for deta
Is thread one alive? false
Is thread one alive? false
Is thread one alive? false
Is thread one alive? false
Is thread one alive? false
PS F:\UPES\Academics\2nd Year\3rd Semester\OOPs
```