Topic- Thread Control and Priorties

Assignment-1

1)Create a thread which prints 1 to 10. After printing 5, there should be a delay of 5000 milliseconds before printing 6

Ans

```
class NumberPrinter extends Thread {
    @Override
    public void run() {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);


            // delay after printing 5
            if (i == 5) {
                try {
                    System.out.println("Delaying for 5 seconds...");
                    Thread.sleep(5000); // 5000 milliseconds = 5 seconds
                } catch (InterruptedException e) {
                    System.out.println("Thread interrupted");
                }
            }
        }
    }
}
public class DelayDemo {
    public static void main(String[] args) {
        NumberPrinter printer = new NumberPrinter();
        printer.start();
    }
```

}

Output

1

2

3

4

5

Delaying for 5 seconds...

6

7

8

9

10

Assignment-2

Q2)**Create two threads, one thread to display all even numbers between 1 & 20, another to display odd numbers between 1 & 20.**

```
class EvenThread extends Thread {
    @Override
    public void run() {
        System.out.println("Even numbers:");
        for (int i = 2; i <= 20; i += 2) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}

class OddThread extends Thread {
    @Override
    public void run() {
        System.out.println("Odd numbers:");
        for (int i = 1; i <= 20; i += 2) {
            System.out.print(i + " ");
        }
        System.out.println();
    }
}

public class EvenOddDemo {
    public static void main(String[] args) {
        EvenThread evenThread = new EvenThread();
```

```
        OddThread oddThread = new OddThread();

        evenThread.start();

        try {
            // Wait for even thread to complete before starting odd thread
            evenThread.join();
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted");
        }

        oddThread.start();
    } }
```

Output

```
Even numbers:
2 4 6 8 10 12 14 16 18 20
Odd numbers:
1 3 5 7 9 11 13 15 17 19
```

Assignment-3

3)**Create three threads- with different priorities - MAX, MIN, NORM- and start the threads at the same time. Observe the completion of the threads.**

Ans class PriorityThread extends Thread {

```
    private String threadName;


    public PriorityThread(String name) {
        this.threadName = name;
    }


    @Override
    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(threadName + " - Count: " + i +
                            " (Priority: " + getPriority() + ")");
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                System.out.println(threadName + " interrupted");
            }
        }
        System.out.println(threadName + " completed execution");
    }
}


public class PriorityDemo {
    public static void main(String[] args) {
        PriorityThread maxThread = new PriorityThread("MAX_PRIORITY");
```

```
        PriorityThread minThread = new PriorityThread("MIN_PRIORITY");
        PriorityThread normThread = new PriorityThread("NORM_PRIORITY");

        // Setting different priorities
        maxThread.setPriority(Thread.MAX_PRIORITY);   // 10
        minThread.setPriority(Thread.MIN_PRIORITY);   // 1
        normThread.setPriority(Thread.NORM_PRIORITY); // 5

        // Starting all threads at the same time
        maxThread.start();
        minThread.start();
        normThread.start();

        System.out.println("All threads started with different priorities");
    }
}
```

Ouptut

```
All threads started with different priorities
MAX_PRIORITY - Count: 1 (Priority: 10)
NORM_PRIORITY - Count: 1 (Priority: 5)
MIN_PRIORITY - Count: 1 (Priority: 1)
MAX_PRIORITY - Count: 2 (Priority: 10)
NORM_PRIORITY - Count: 2 (Priority: 5)
```