Assignment Week 3 Multithreading

Program 1: Create a thread by implementing runnable interface

```
public class _01_Thread_using_Runnable {
    public static void main(String[] args) {
        NewThread nt = new NewThread();
        Thread thread=new Thread(nt,"number thread");
        thread.start();
    }

    private static class NewThread implements Runnable {
        @Override
        public void run() {
            for(int i=0;i<10;i++){
                System.out.println("Child Thread: "+i);
                try{
                    Thread.sleep(1000);
                }
                catch(InterruptedException e){
                    e.printStackTrace();
                }
            }
        }
    }
}
```

Output-

```
Child Thread: 0
Child Thread: 1
Child Thread: 2
Child Thread: 3
Child Thread: 4
Child Thread: 5
Child Thread: 6
Child Thread: 7
Child Thread: 8
Child Thread: 9
```

Program 2: Print even and odd numbers using different threads

```
class PrintEvenThread implements Runnable {
    private int maxVal;

    public PrintEvenThread(int maxVal) {
        this.maxVal = maxVal;
    }

    @Override
    public void run() {
        for (int i = 2; i <= maxVal; i += 2) {
            System.out.println(Thread.currentThread().getName() + "=> " + i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
            }
        }
    }
}

class PrintOddThread implements Runnable {
    private int maxVal;

    public PrintOddThread(int maxVal) {
        this.maxVal = maxVal;
    }

    @Override
    public void run() {
        for (int i = 1; i <= maxVal; i += 2) {
            System.out.println(Thread.currentThread().getName() + "=> " + i);
            try {
                Thread.sleep(100);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}

public class _02_Even_and_Odd_Number {
    public static void main(String[] args) {
        int maxVal = 20;

        PrintEvenThread even = new PrintEvenThread(maxVal);
        PrintOddThread odd = new PrintOddThread(maxVal);

        Thread evenThread = new Thread(even, "Even");
        Thread oddThread = new Thread(odd, "Odd");

        evenThread.start();
        oddThread.start();
    }
}
```

Output-

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

Program 3: Create one user and two daemon threads

```
public class _03_Two_Daemon_and_One_UserThreads {

    public static void main(String[] args) {
        // Creating two daemon threads
        Thread dt1 = new Thread(new DaemonTask(), "Daemon Thread 1");
        Thread dt2 = new Thread(new DaemonTask(), "Daemon Thread 2");

        // Setting daemon status for the daemon threads
        dt1.setDaemon(true);
        dt2.setDaemon(true);

        // Creating a user thread
        Thread tUser = new Thread(new UserTask(), "User Thread 1");

        // Starting all threads
        dt1.start();
        dt2.start();
        tUser.start();
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.println("Exiting main thread... "+Thread.currentThread().getName());
    }

    // Task for daemon threads
    private static class DaemonTask implements Runnable {
        @Override
        public void run() {
            while (true) {
                System.out.println(Thread.currentThread().getName() + " is up and running...
");

                try {
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    // Task for user thread
    private static class UserTask implements Runnable {
        @Override
        public void run() {
            System.out.println(Thread.currentThread().getName() + " is up and running...");
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output-

```
Daemon Thread 2 is up and running...
User Thread 1 is up and running...
Daemon Thread 1 is up and running...
Daemon Thread 1 is up and running...
Daemon Thread 2 is up and running...
Exiting main thread... main
Daemon Thread 1 is up and running...
Daemon Thread 2 is up and running...
```

Program 4: Create a thread which contains below methods

- **getID()**
- **isAlive()**
- **currentThread()**
- **sleep(milliseconds)**

```
public class _04_ThreadMethods {
    public static void main(String[] args) {
        CustomThread nt = new CustomThread();
        System.out.println(nt.getID());
        System.out.println(nt.isAlive());
        System.out.println(nt.currentThread());
        nt.sleep(1000);

    }
}

class CustomThread implements Runnable {
    private Thread thread;

    public CustomThread() {
        thread = new Thread(this);
        thread.start();
    }

    public long getID() {
        return thread.getId();
    }

    public boolean isAlive() {
        return thread.isAlive();
    }

    public Thread currentThread() {
        return Thread.currentThread();
    }

    public void sleep(long milliseconds) {
        try {
            Thread.sleep(milliseconds);
        } catch (InterruptedException e) {
            System.out.println("Thread interrupted");
        }
    }
```

```
    @Override
    public void run() {
        System.out.println("Thread is running");
    }
}
```

Output-

```
21
true
Thread is running
Thread[#1,main,5,main]
```