Familiarization with threads

Aim

Implementation of threads, a program having threads - one thread prints all even numbers and finds the sum of first n even numbers another thread prints all odd numbers and finds the sum of first n odd numbers

Theory

Basic structure of thread

```
class NT implements Runnable {
        public void run() {// write function of thread here
        }
}
class ThreadDemo {
        public static void main(String args[]) {
                NT t=new NT(); // object t is created using class NT
                Thread tt= new Thread(t);//object t is converted into thread
                tt.start();// call the run() function
        }
}
```

In the above example modify run() function as given below, then run and check the output

```
public void run() {
        for(int i = 0; i <25 ; i=i+2)
                System.out.println("Even Thread: " + i);
        }
```

output

```
E:\JAVA_LAB\T01>java ThreadDemo
Even Thread: 0
Even Thread: 2
Even Thread: 4
:
:
Even Thread: 24
```

Again change the run() function// check the change in output

```
public void run() {
    try {
        for(int i = 0; i <20 ; i=i+2) {
                System.out.println("Even Thread: " + i);
                Thread.sleep(400);
                }
       } catch (InterruptedException e) {
                System.out.println("Child interrupted.");
       }
  }
//400 millisecond delay between displaying of even numbers
//Thread.sleep(400); command must be used inside the try catch statement, because it throws
InterruptedException
```

Another way to create thread

```
class NT implements Runnable {
        Thread t;
        NT() {//constructor
                t = new Thread(this);
                t.start(); // Start the thread by calling run()
        }
        // This is the entry point for the second thread.
        public void run() {
                }
}
class ThreadDemo {
        public static void main(String args[]) {
                new NT(); // create a new thread
                }
}
```

passing arguments to thread

```
class NT implements Runnable {
        Thread t;
        int j;
        NT(int i) {//constructor
                j=i;
                t = new Thread(this);
                t.start(); // Start the thread by calling run()
        }
        // This is the entry point for the second thread.
        public void run() {
                System.out.println("value of j: " +j);//prints 6
                }
}
class ThreadDemo {
        public static void main(String args[]) {
                new NT(6); // create a new thread and 6 is passed to thread
                }
}
```

Output
value of j: 6

creation of two threads in a program

```
class NT1 implements Runnable {
        public void run() {// write function of first thread here
        }
}
class NT2 implements Runnable {
        public void run() {// write function of second thread here
        }
```

```
}
class ThreadDemo {
        public static void main(String args[]) {
                NT1 t1=new NT1(); // object t is created using class NT1
                Thread tt1= new Thread(t1);//object t is converted into thread
                tt1.start();// call the run() function

                NT2 t2=new NT2(); // object t is created using class NT2
                Thread tt2= new Thread(t2);//object t is converted into thread
                tt2.start();// call the run() function
        }
}
```

The program given below will print the even numbers and odd numbers alternatively

```
class NT1 implements Runnable {
public void run() {
        try {
        for(int i = 0; i <20 ; i=i+2) {
                System.out.println("Even Thread: " + i);
                Thread.sleep(250);
                }
        } catch (InterruptedException e) {
                System.out.println("Child interrupted.");
                }
        }
}

class NT2 implements Runnable {
public void run() {
        try {
        for(int i = 1; i <20; i=i+2) {
                System.out.println("Odd Thread: " + i);
                Thread.sleep(250);
                }
        } catch (InterruptedException e) {
                System.out.println("Child2 interrupted.");
        }
        }
}
class ThreadDemo {
        public static void main(String args[]) {
        NT1 t1=new NT1(); // create a new thread
        NT2 t2=new NT2(); // create a new thread
        try {
                Thread tt1= new Thread(t1);
                Thread tt2= new Thread(t2);
                tt1.start();
                tt2.start();
```

```
                }
        catch (Exception e) {
                System.out.println("Main thread interrupted.");
                }
        }
}
```

output

```
Even Thread:0
Odd Thread:1
Even Thread:2
Odd Thread:3
Even Thread:4
Odd Thread:5
:
Even Thread:18
```

Modify the ThreadDemo class in the above program like this and check the output

```
class ThreadDemo {
        :
        :
                Thread tt1= new Thread(t1);
                Thread tt2= new Thread(t2);
                tt1.setPriority(Thread.NORM_PRIORITY+1);
                tt2.setPriority(Thread.NORM_PRIORITY-1);
                tt1.start();
                tt2.start();
        :
        :
}
```

```
// Demonstrate thread priorities.
class clicker implements Runnable {
        long click = 0;
        Thread t;
        private volatile boolean running = true;
        public void run() {
                while (running) {
                        click++;
                }
        }
        public void stop() {
                running = false;
        }
}
class HiLoPri {
public static void main(String args[]) {
        Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
```

```
        clicker hi = new clicker();
        clicker lo = new clicker();
        Thread t1=new Thread (hi);
        Thread t2=new Thread (lo);
        t1.setPriority(Thread.NORM_PRIORITY + 1);
        t2.setPriority(Thread.NORM_PRIORITY );
        t1.start();
        t2.start();
        try {
                Thread.sleep(7000);
                }
        catch (InterruptedException e) {
                System.out.println("Main thread interrupted.");
        }
        lo.stop();
        hi.stop();
        // Wait for child threads to terminate.
        try {
                t1.join();
                t2.join();
        } catch (InterruptedException e) {
        System.out.println("InterruptedException caught");
        }
        System.out.println("Low-priority thread: " + lo.click);
        System.out.println("High-priority thread: " + hi.click);
        }
}
```

verify the output of above program and find out the reason for it

---

Output of the lab experiment

Enter the value of N:5
Odd Thread:1
Even Thread:2
Odd Thread:3
Even Thread:4
Odd Thread:5
Even Thread:6
Odd Thread:7
Even Thread:8
Odd Thread:9
Even Thread:10
Sum of Even number is 30
Sum of Odd number is 25
