Thread

Two ways to create thread

- ☐ Using Runnable interface
- ☐ By Extending thread class

Extending Keyword

Methods:

- run()- block of code to be executed will be written inside run mentod
- start()- to start the execution of the thread
- sleep()- to suspend the execution of the time of a thread

Implementing thread using Runnable interface

In runnable interface there is only one run method
The most common use case of the Runnable interface is when we want only to override the run method.

Example:

```
package Assignment_2;
import java.lang.Thread;
public class ThreadusingRunnableInterface {
    public static void main(String[] args) {

        even1 ob1 =new even1();
        odd1 ob2=new odd1();
        Thread t1=new Thread(ob1,"even");
        Thread t2=new Thread(ob2,"odd");
        t1.getName();
        t2.getName();
        t2.setPriority(1);
        t1.setPriority(10);
        t1.start();
        t2.start();

    }
}

class even1 implements Runnable{
    @Override
    public void run() {
        try {
            for (int i = 1; i < 10; i++) {
                System.out.println(i * 2);
//                Thread.sleep(1000);

            }
```

```
        }
        catch (Exception e)
        {
            System.out.println("error");
        }
    }
}

class odd1 implements Runnable{
    @Override
    public void run() {
        try {
            for (int i = 1; i < 20; ) {
                System.out.println(i);
                i = i + 2;
            }
            Thread.sleep(10);
        }
        catch (Exception e)
        {
            System.out.println("error");
        }
    }
}
```

Thread class vs Runnable interface

- By extending thread, there is overhead of additional methods, i.e. they consume excess or indirect memory, computation time, or other resources.
- Since in Java, we can only extend one class, and therefore if we extend Thread class, then we will not be able to extend any other class. That is why we should implement Runnable interface to create a thread.
- Runnable makes the code more flexible as, if we are extending a thread, then our code will only be in a thread whereas, in case of runnable, one can pass it in various executor services, or pass it to the single-threaded environment.
- Maintenance of the code is easy if we implement the Runnable interface.