Subject:- Unix Operating System
System Lab Class :- TYIT

Name **PRN**
Aditi Sudhir Ghate **2020BTEIT00044**

4.2 Write a program which creates 3 threads, first thread printing even number, second thread printing odd number and third thread printing prime number in terminals.

Objectives:

1. To learn about threading in Linux/Unix and Java and

difference between them

2. Use of system call/library to write effective programs

Theory:

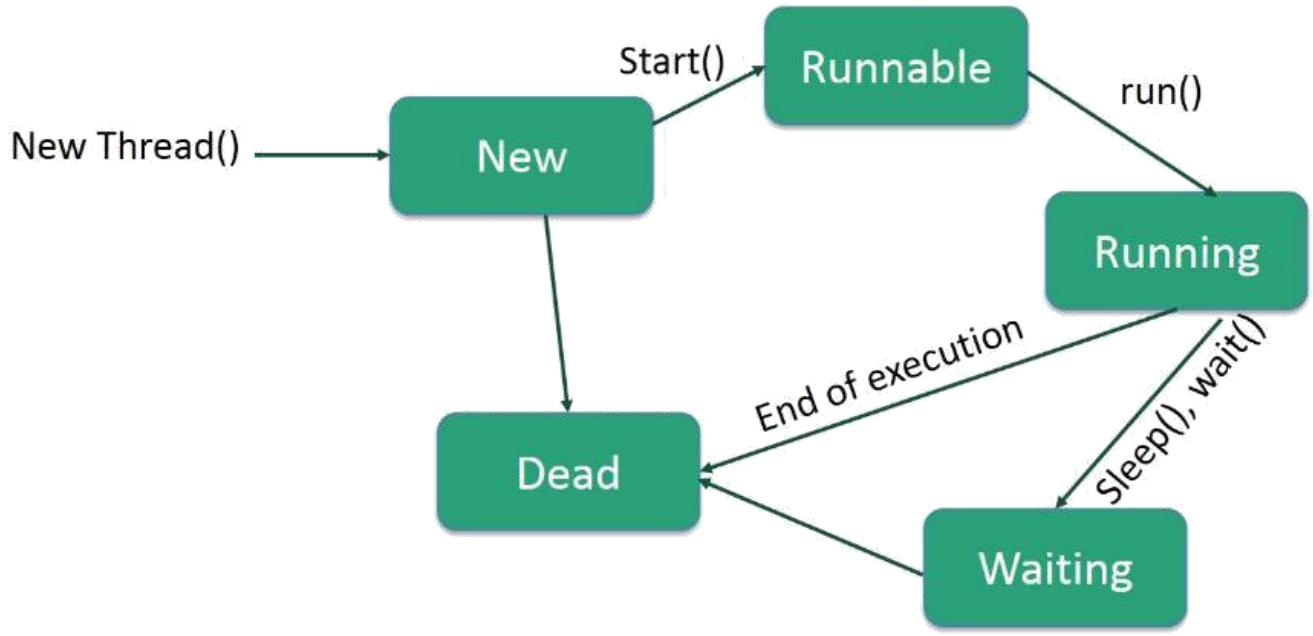
Java is a *multi-threaded programming language* which means we can develop multi-threaded program using Java. A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources specially when your computer has multiple CPUs.

By definition, multitasking is when multiple processes share common processing resources such as a CPU. Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads. Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.

Life Cycle of a Thread:

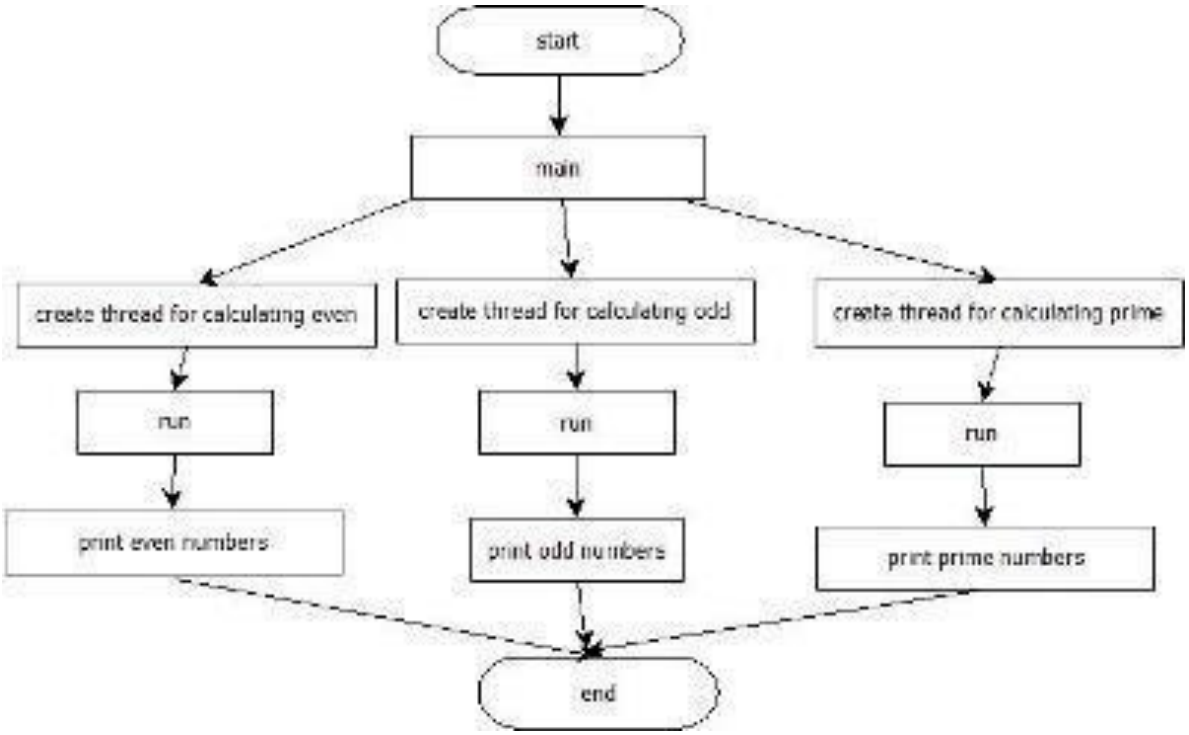
A thread goes through various stages in its life cycle. For example, a thread is born, started, runs, and then dies. The following diagram shows the complete life cycle of a thread.



Following are the stages of the life cycle:

- **New** − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.
- **Runnable** − After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
- **Waiting** − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
- **Timed Waiting** − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
- **Terminated (Dead)** − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

Flowchart:



Data Dictionary:

Sr Number	Variable/Function	Datatype	Use
1	t1	Thread1	Print odd numbers.
2	t2	Thread2	Print prime numbers.
3	tm	B4	Print even numbers.
4	i	int	Iterationg for loop.

Program:

```
public class B4 extends Thread
{
public static void main(String []args)
{
Thread1 t1 = new Thread1();
Thread2 t2 = new Thread2();
B4 tm = new B4();



tm.start();
t1.start();
t2.start();
}
public void run()
{
for(int i=0;i<=30;i=i+2)
{
System.out.println("Even: "+i);
```

```
}
}
}

class Thread1 extends Thread
{
public void run()
{
for(int i=1;i<=31;i=i+2)
{
System.out.println("Odd: "+i);
}
}
}
class Thread2 extends Thread
{
public void run()
{
for(int i=2;i<100;i++)
{
int count = 0;
for(int j=2;j<i;j++)
{
if(i%j==0)
{
count++;
}
}
if(count==0)
System.out.println("Prime: "+i);
}
}
}
```

Output:

```
aditi@aditi-Lenovo-ideapad-330S-14IKB-U:~/ADnOR/Assignments/4B$ javac B4.java
aditi@aditi-Lenovo-ideapad-330S-14IKB-U:~/ADnOR/Assignments/4B$ java B4
Prime: 2
Odd: 1
Odd: 3
Even: 0
Even: 2
Prime: 3
Even: 4
Odd: 5
Odd: 7
Odd: 9
Odd: 11
Odd: 13
Odd: 15
Odd: 17
Odd: 19
Odd: 21
Odd: 23
Odd: 25
Odd: 27
Odd: 29
Odd: 31
Even: 6
Prime: 5
Even: 8
Prime: 7
Prime: 11
Prime: 13
Prime: 17
Prime: 19
Prime: 23
Even: 10
Prime: 29
Prime: 31
Even: 12
Prime: 37
Even: 14
```

```
Even: 16
Even: 18
Even: 20
Even: 22
Prime: 41
Prime: 43
Even: 24
Prime: 47
Even: 26
Even: 28
Prime: 53
Even: 30
Prime: 59
Prime: 61
Prime: 67
Prime: 71
Prime: 73
Prime: 79
Prime: 83
Prime: 89
Prime: 97
```

Conclusion:

Multiple threads and their execution pattern studied in details. Need for mechanism to synchronize recognized.

References:

https://www.geeksforgeeks.org/multithreading-in-java/