Thread – Multithreading

1st - 10th Threads

APRIL / MAY

15th week
101-264

11 Saturday

Multithreading → It is the process of executing several task simultaneously.

There are two types of multitasking :—

Multitasking

- **Process based multitasking**
  - Facebook, Music, Chat
  - → Heavyweight
  - → Own memory address space
  - → Interprocess communication is expensive
- **Thread-based multitasking**
  - → text editor (m = memory)
    - edit text
    - print
  - → Light weight
  - → Address Space is shared.
  - → Interthread communication is cheap.

Benefits of Multi-threading :–

(I) Enables to do multi things at one time.

12 Sunday

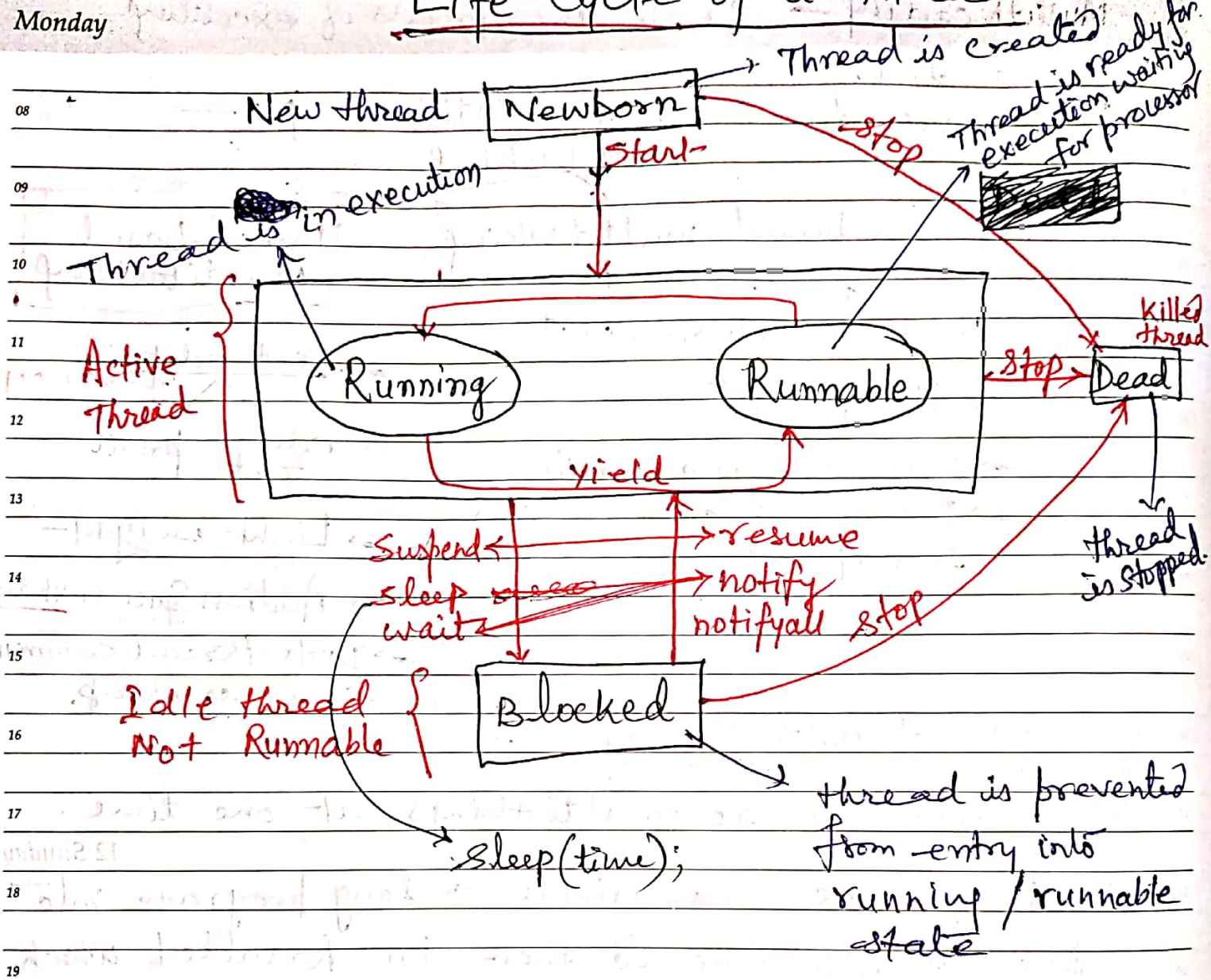
(II) Programmers can divide a long program into threads and execute them in parallel which will eventually increases speed of program execution.

(III) Improved performance and concurrency

(IV) Simultaneous access to multiple applications.

Notes

Life Cycle of a Thread



Thread Creation in Java :-

Main thread — Whenever a program starts execution from main function, one thread begins running which is called as the main thread of the program because it

Notes

APRIL

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MAY

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

14
Tuesday

is the one that is executed when your program begins.

We can create our own thread in java by two ways :—

(i) By implementing Runnable interface

(ii) By extending from Thread class.

(i) When a class is already extended from a base class and need to have multi-thread -ing capability, then we have to implement this class as :—

```
class thread-1 extends Apclass implements
                                    Runnable
{
——
——
}
```

(ii) When a class is directly extended from Thread class then :—

```
class thread2 extends Thread
{
——
——
}
```

Notes

15
Wednesday

16th week
105-260

April

Thread creation in java

Extending Thread class

```
class t2 extends Thread
{
    public void run()
    {
        System.out.println("Thread t2");
    }
public static void main(String args[])
{
    t2 obj1 = new t2();
    obj1.start();
}
}
```

Implementing Runnable interface

```
class t1 implements Runnable
{
    public void run()
    {
        System.out.println("Thread t1");
    }
public static void main(
        String args[])
{
    t1 obj1 = new t1();
    Thread t = new Thread(obj1);
    t.start();
}
}
```

Example to Run (complete) :- threadtest.java

```
class thread1 implements Runnable
{
    public void run()
    {
        for(int i=0; i<5; i++)
```

Notes

APRIL

S	M	T	W	T	F	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

MAY

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

16
Thursday

```
System.out.println("Thread 1: "+i);
    }
}
}
class Thread2 implements Runnable
{
    public void run()
    {
for(j=0; j<5; j++)
{
System.out.println("Thread 2: "+j);
}
    }
}
class threadtest {
    public static void main(String args[])
    {
Thread1 t1 = new Thread1();
Thread2 t2 = new Thread2();
Thread tobj1 = new Thread(t1);
Thread tobj2 = new Thread(t2);
    tobj1.start(); tobj2.start();
}
}
```

Notes

Use of isAlive() and join() method:-

- isAlive() method tests if the thread is alive or not. It returns true/false.

Syntax:- `public final boolean isAlive()`

- join() method waits for a thread to die. It causes currently running thread to stop executing until the thread it joins completes its task.

Example:- threadtest1.java

```
class Thread1 extends Thread
{
    public void run()
    {
        for(int i=0; i<=5; i++)
        {
            System.out.println("Status:" + i + isAlive());
        }
        System.out.println("Exit from thread1");
    }
}
class threadtest1{
    public static void main(String args[])
    {
        Thread1 t1 = new Thread1();
        t1.start();
        System.out.println("New Status:" + t1.isAlive()
```

Notes

o/p :-

NewStatus : true
Status : true
Status : true
Status : true
Status : true
Status : true
Exit from thread 1.

Now, let us modify the same program by calling join() method as :-

```
class thread1 extends Thread
{
    public void run()
    {
        for (int i=0; i<=5; i++)
        {
        System.out.println(" Status : " + isAlive());
        }
    System.out.println("Exit from thread 1");
    }
}
class threadtest1{
public static void main(String args[])
{
    thread1 t1 = new thread1();
    t1.start();
    try { t1.join();
    }
```

19 Sunday

Notes

```
        catch(Exception e)
        {
System.out.println("Error:"+e.toString());
        }
System.out.println("New Status:"+t1.isAlive());
    }
}
```

Output :-

Status : true
Status : true
Status : true
Status : true
Status : true
Status : true
Exit from thread 1
New Status : false

Explanation of above programs :—

main thread starts to call start method as

t1.start();

t1 started to call run() which print Status: true 6 times according to loop's order, then Exit from thread1.. After that Control transferred to try block and the Statement

System.out.println("New Status:"+t1.isAlive());

is paused till the t1.join();

21
Tuesday

does not finish its execution. After finishing execution of t1, the final new status of t1 will be :-

New Status : false.