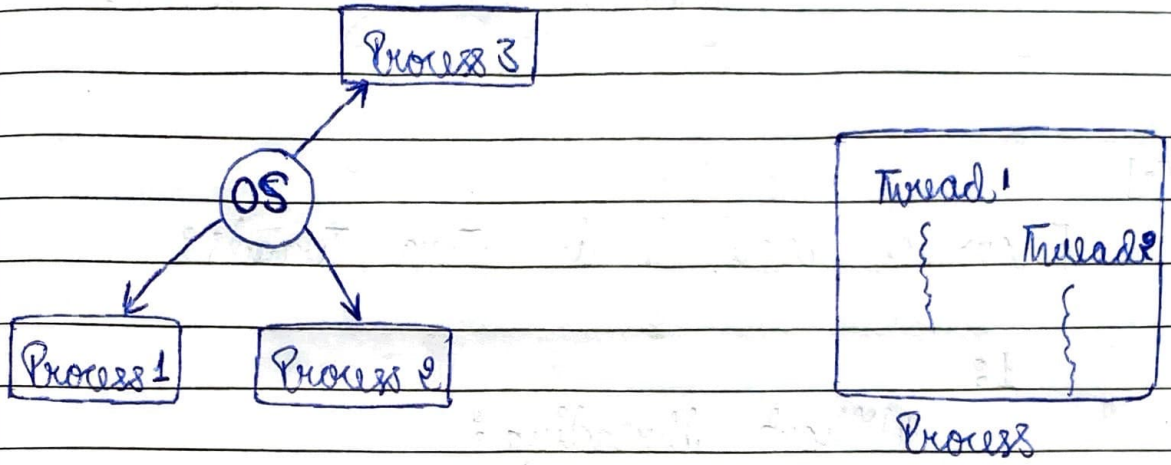
MULTITHREADING IN JAVA

Date :
Page no. 1

[•]

Multiprocessing and multithreading both are used to achieve multitasking



[•]

Multithreading :

a technique by which a single set of code can be used by several processors at different stages of execution.

[•]

→ **Threads :**

A direction or path that is taken while a program is being executed.

→ Threads use shared memory area.
→ Threads => faster context switching
→ A thread is lightweight where a process is heavyweight.

→ For Example :

A word processor can have one thread running in foreground as an editor and another in the background auto saving the document.

[•] Flow of control in Java Threads :

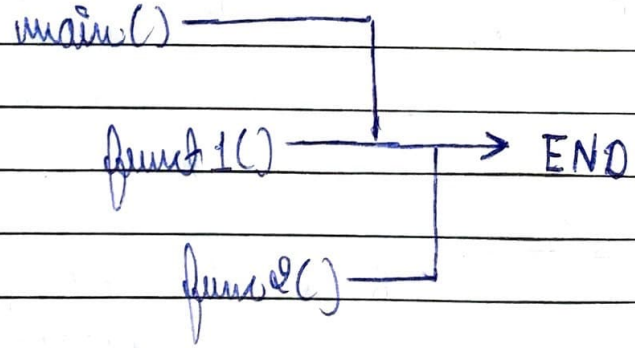
1 :

Without threading :

main() → func 1() → func 2() → END

2 :

With Threading



[•] Creating a Thread :

- There are two ways to create thread in Java.

1. By Extending Thread class.

2. By implementing Runnable Interface

→ Concurrency vs Parallelism

[•]

Concurrency :-

The ability of a program to execute multiple tasks simultaneously.

[•] Parallelism :-

An application where tasks are divided into smaller sub-tasks that are processed seemingly simultaneously or parallel.

Difference

[•]

Concurrency focuses on managing multiple tasks efficiently with one resource.

Parallelism utilizes multiple resources to execute tasks simultaneously, making process ~~function~~ faster.

[0]

Extending Thread Class : [Syntax]

Ex :

```
-> class MyThread1 extends Thread
   {
      @Override
      public void run()
      {
         while (i < 4000)
         {
            Sout ("Hello Thread 1");
         }
      }
   }

-> class MyThread2 extends Thread
   {
      @Override
      public void run()
      {
         while (i < 4000)
         {
            Sout ("Hello Thread 2");
         }
      }
   }
```

```
-> // Main Class
    public class Threading001
    {
        public static void main (String args[])
        {
            MyThread1 t1 = new MyThread1();
            MyThread2 t2 = new MyThread2();

            t1.start();
            t2.start();
        }
    }
```

[•]

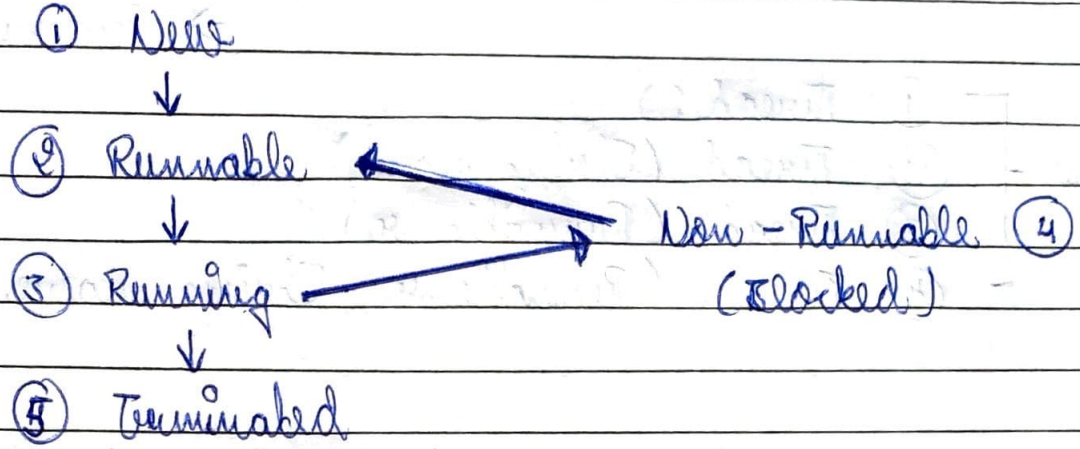
Syntax for Runnable Interface:

Example:

```
-> class MyRunnable1 implements Runnable
    {
        @Override
        public void run()
        {
            while (true)
            {
                Sout ("Thread 1");
            }
        }
    }
```

```
-> class MyRunnable2 implements Runnable
   {
      @Override
      public void run()
      {
         while (true)
         {
            Sout ("Thread 2");
         }
      }
   }

// Main class
        : Can not run directly start Method
->

public class Threading-002-Runnable.Interface
   {
      public static void main (String args[])
      {
         MyRunnable1 Obj1 = new MyRunnable1();
         Thread run1 = new Thread (Obj1);

         MyRunnable2 Obj2 = new MyRunnable2();
         Thread run2 = new Thread (Obj2);

         run1.start();
         run2.start();
      }
   }
```

[•] Life Cycle of a Thread :



① New :

Instance of thread is created which is not yet started by invoking start().

② Runnable :

After invocation of start() and before it is selected to be run by the scheduler.

③ Running :

After thread scheduler has selected it.

④ Non-Runnable :

Thread alive, not eligible to run.

⑤ Terminated :

run() method has exited.

[•] The Thread Class in Java :

Below are the commonly used Constructors of Thread Class :

→ Constructors of Thread

Simple -
1. Thread ()
2. Thread (String name)
3. Thread (Runnable r)
4. Thread (Runnable r, String name).

Thread Group -
5. Thread (ThreadGroup tg, Runnable target);
6. Thread (ThreadGroup tg, String name);
7. Thread (ThreadGroup tg, Runnable target, String name);
8. Thread (ThreadGroup tg, Runnable target, String name, long stackSize).

Ex :-

```
public class Thread implements Runnable
{
    Thread()
    {
        // Code.
    }
}
```

[•] Thread Methods in Java:

Below are some commonly used methods of Thread Class:

→ METHODS ←

[•] Basic Methods

→ ① run() ② start()
③ currentThread() ④ isAlive().

[•] Naming Methods

→ ① getName() ② setName(String name)

[•] Daemon Thread Methods:

→ ① isDaemon() ② setDaemon(String name)

[•] Priority Based Methods

→ ① getPriority ② setPriority(int priority)

[•] Preventing Thread execution Method.

→ ① Sleep() ② yield() ③ join().

[•] Deprecated Methods

→ ① suspend() ② resume()
③ stop() ④ destroy()

[•] Interrupting the Thread Methods.

→ ① interrupt() ② isInterrupted() ③ interrupted()