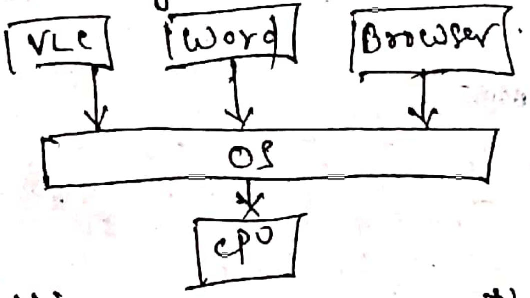
MultiThreading.

Q. Difference between Multitasking, multi processing & multi threading

Multitasking :- Performing multiple task at single time.



*) CPU executes one task at a time but cpu switches between the tasks so fast that it creates an illusion that ~~a~~ all tasks are executed at a time.

*) It increase the performance of cpu

~~Multi processing~~

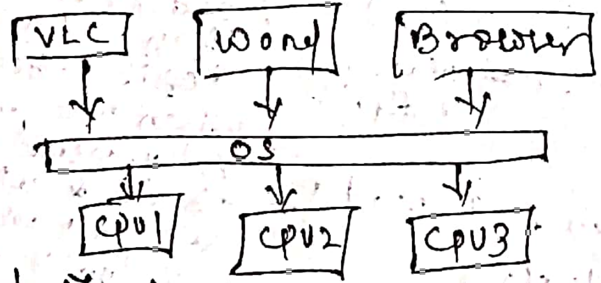
*) It is achieved by two ways

→ ~~Multi~~ process based Multitasking (Multi processing)

*) Thread based Multitasking (Multi Threading)

process Based Multitasking (Multi processing)

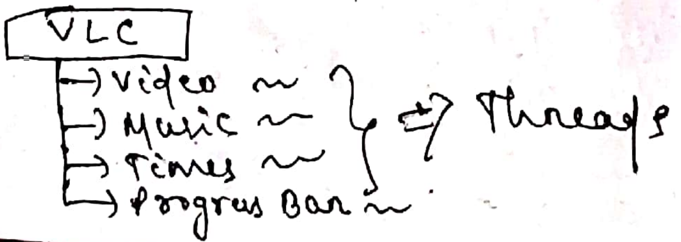
*) when one system is connected to multiple system. in order to complete the tasks.



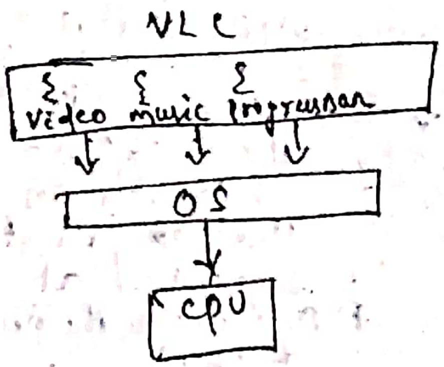
*) It is best suitable at System level or OS level.

Multithreading.

*) One ~~task~~ ~~or~~ task or process is divided into no. of threads. & threads are executing synyultaneously.



*) Executing multiple threads (sub process/ small tasks) at single time.



*) Used in s/w e.g. VLC
*) Used in Games.
*) Animations

*) Where to use MultiThreading.

e.g. VLC player.

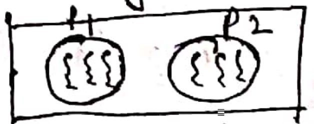
```
class VLC
{    p.s.v.m( )
        {  ~~void~~ play();
                    start();
}       }
class video -> Thread 1
    {  void play()
          {
    }     }
class Music( ) ——> Thread 2
    {  void start()
          {
    }     }
```

// If we call one by one method then it is not possible because when we ~~play~~ call video music is not there & when we call Music video is not there. so we have to execute parallely all the functions. So convert them to thread.

*) Multi Threading is best suitable at programming level.

*) Java provides Pre defined API for Multithreading.
e.g. Thread, Runnable, Thread Group.

Different between Process & Threads

Process	Threads
*) A program which is is Executing state	*) It is subpart of Process (small task).
*) 	{ }
*) Process is heavy weight	*) Thread is light weight.
*) ~~Context fo~~ process take more time for context switching	*) Thread takes less time for context switching
*) Takes more time for Interprocess Communication	*) Takes less time for inter thread communication.
*) Each processes has different Address space	*) Thread shares same address space.
*) Processes are not dependent on Each other	*) Threads are dependent ~~to~~ each other.
*) process doesnot required synchronization	*) Thread may required synchronization.
*) Resource Consumption is more	*) Resource Consumption is less.
*) Requires ~~less~~ More time for Creation	*) less time for creation
*) Process Requires more time for termination	*) less time for termination

~~Thread Life cycle~~

Thread creation

*) There are two ways to create Thread.
→ By using Thread (class)
→ " " Runnable (interface).

Using Thread (class)

```
        package java.lang;
  class Thread implements Runnable.
  {
     // constructors
     // methods
     - Run()
     - Start()
     - sleep()
     - join()
     - getName() & setName()
     - interrupted.
  }
```

create a thread:

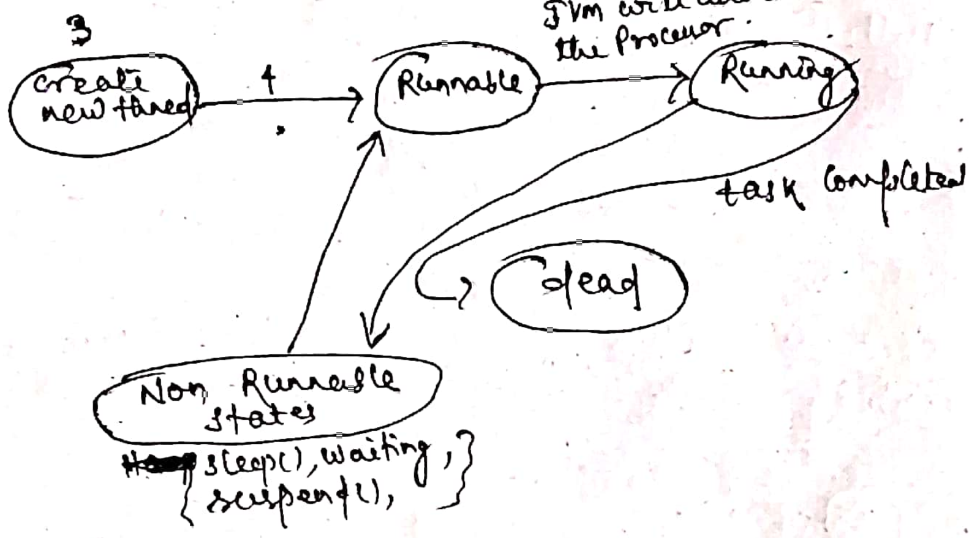
① Extend the thread class.

```
class test extends Thread.
{   public void run()     ② override the run() method
    {

    }
    p.s.v.m()   ③ create an object of the class.
    { Test ob = new Test();
      ob.start();
    }       ④ start the thread.
}
```

Life cycle of Thread.

*) 5 stages are there.



Runnable Interface.

```
package java.lang;
public interface Runnable
{ //method (only one method)
      run();
}
```

~~creating a thread using thread class~~

~~class test~~ er

```
public class test extends Thread
{   public void run()
  {  s.o.p ("sat");
     for (i=0 ; i<=10 ; i++)
     { s.o.p ("running...." + this.getName());
       s.o.p (i);
     }
  }
     p.s.v.m ( )
     { test t1 = new test();
       test t2 = new test();
       t1.start ();
       t2.start();
     }
}
```

Using Runnable interface.

```
class test implements Runnable
   {   public void run()
       {
         for ( i=0 ; i<=10; i++)
         {  s.o.p ("running - - - - )
            s.o.p(i),
       } }
         p.s.v.m( )
         { test t = new test();
         Thread th = new. Thread (t);}
         th.start();
```