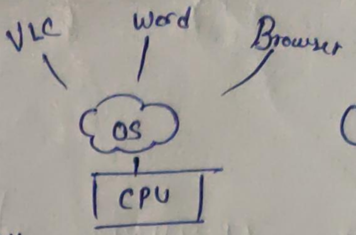
multi Threading in Java.

multitask: performing multiple task at single time.

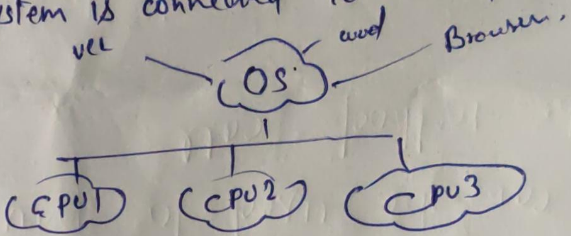


Switching

It increses the performance of CPU.

It can be achived via
1) Process based Multitask   multiprocessing
2) Thread based multitask   multithreading

Multiprocessing : when One system is connected to multiple processors.



It is suitable for system/os Level.

Multithread : executing multiple (small process) threads at single task.

VLC
Video
music
Timer
Progress bar.

uses   S/w.
games.
Animation.

VLC (program)

```
Clau VLC
  {
    psum( )
    {  ≡   playVideo( );
           playmusic( );
    }                create
                  ────────────→ thread ①
Clau video {
      Void playVideo( )
        {  ≡
        }
  }                         create.   thread ②
Clau music {  Void playmusic
          {  ≡
          }
  }
```

≡

Multithreading is best suited at programming level.

Java provides predefined API for multithreding

Thread
Runnable
Threadgroup
Threadpool
Concurrency

$\Bigg\}$ classes in util package.

Process	thread
A program in executing state.	It is subset of program
Context switch - takes long time	takes less time.
Interprocess Communication long time	less time.
Process share diff address space	thread share same address space.
Process are not depend on each other	thread depends on each other
Synchronization: Process does not require sync.	thread require sync.
Resource Consumption is more	less.
Time of creation (more)	less
Time of termination (more)	less

Thread life Cycle

How to create Thread in Java

Two ways
- Thread clam
- Runnable interface.

Thread clau have many constructors and methods.

```
Class Thread implements Runnable
{
   public void run ( ){≡
                      }
p Sync. void   start( ){≡
                      }
public static native Thread current Thread ( );

public final native boolean isAlive( );

  public final String getname()
                               {≡
                               }
public final Sync. void setName (String Nam)
                               {≡
                               }
```

isDaemon
setDaemon

getpriority.
setpriority.

THREADS.

2 ways to create Thread through—

1) Thread { class }

2) Runnable { interface } — better way. for a big project every class will extends some class multiple inheritence

java has predefined class $\boxed{\text{Thread}}$

containing constructors & methods

Basic {
- run — void
- start
- currentThread
- isAlive

Naming methods. {
- getName — (String)
- setName — (synchronized)

Daemon Thread methods {
- isDaemon — (boolean)
- setDaemon — (void)

Priority Based methods {
- get Priority (int)
- set Priority — (void)

methods who are responsible for pausing
(11) thread execution.

sleep(long millis)
yield
join
suspend
resume
stop
destroy

} Prevent Thread Execution Method

interrupt
isInterrupted
interrupted

} Interruppted a thread methods.

How to create thread using class (thread)

'java.lang package' step ①

```
class Test extends Thread
{
  public void run( )
  {
      // thread task
  }
  psvm(string[] arg)
  {
   Test t = new Test( );
   t.start();
  }
}
```

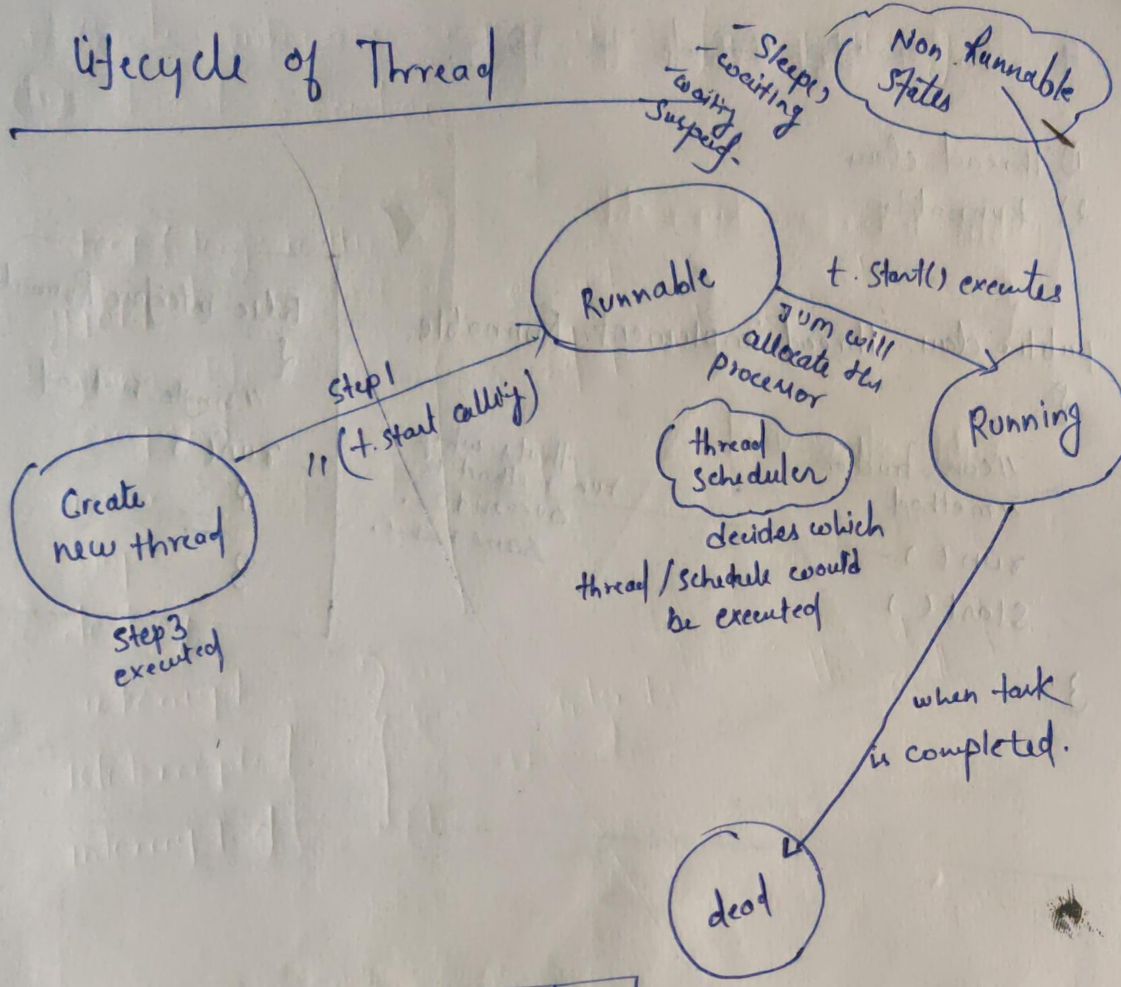
step ② override run method

when method in a subclass has the same name, same parameters and same return type then the method in the subclass is said to override the method in super class.

step 4. //start the thread

// thread is created

lifecycle of Thrread



Once thread is in dead state
it can not be executed again.

How to create threads in Java.

1) Thread class
2) Runnable

```
public class Thread implement Runnable
 {
     //constructor
     // method
     run()
     start()
 }
```

thats why run method is override.
same name.

```
Public interface Runnable
{
  // single method
  run()
   {
     =
   }
```

Method (I)

```
Class Test extends Thread
{
  public void run( )
   {
     // task of thread
     s.o.pln("task thread");
   }
   psvm(string[] arg)
   {
     Test t = new Test( );
   thread.
   start();
     t.start( );  ——— create callstack for t object.
     t.start( ); ——→ give exception  thread can not be invoke again.
   }
}
```

Method (II)

extends the thread class.

Override the run method.

```
class Test implements Runnable          step1
{ public void run( )
   {                                     step2
     // task of thread
     s.o.p( "task thread" );
   }
 psvm(String arg)                         step3
  { Test t= new Test( );
                 Thread (t);              step4
   { Thread th=new Thread(t);
     th.start( );  } }                    step5.
```

```
public class Test extends Thread
 {
  public void run ( )
   {
    S.o.pln (" Thread task");
   3
  psum(String [] args)
   { Test t=new Test ( );
     t. start( );
    t. start ( );    // again runtime exception.
```

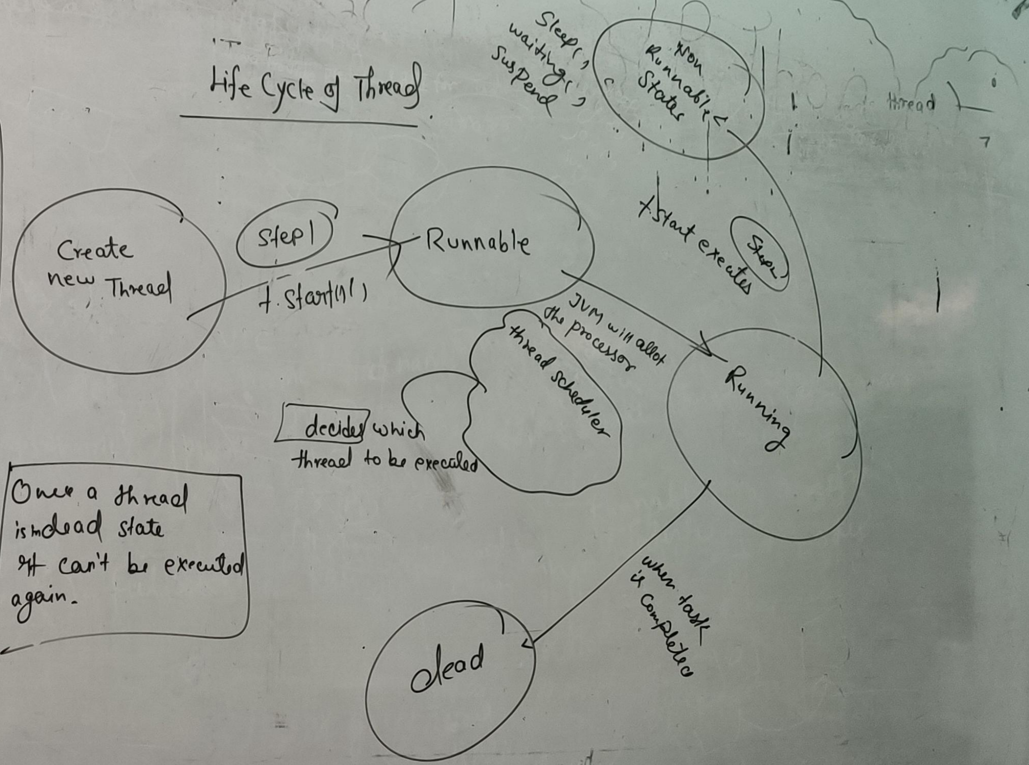
```
public class Test implements Runnable
 {
   public void run( )
    {
     S.o.pln (" task 2");
    3
  psum(String [] arg)
    {
     Test t=new Test ( );
    Thread th= new Thread (t);
     th. start ( );
   3
 3
```

MultiThreading in Java

titask

```
import java.util.*;
Class Test extends Thread        run()
{
    public void run( )
    {
        ____
        ____    thread task.
        ____
    }
    psvm(Strig[ ] arg)
    {
        Test t = new Test( );
        t.start( );
```

Life Cycle of Thread



Once a thread is dead state it can't be executed again.

How to create Thread

MultiThreading in Java

Method I

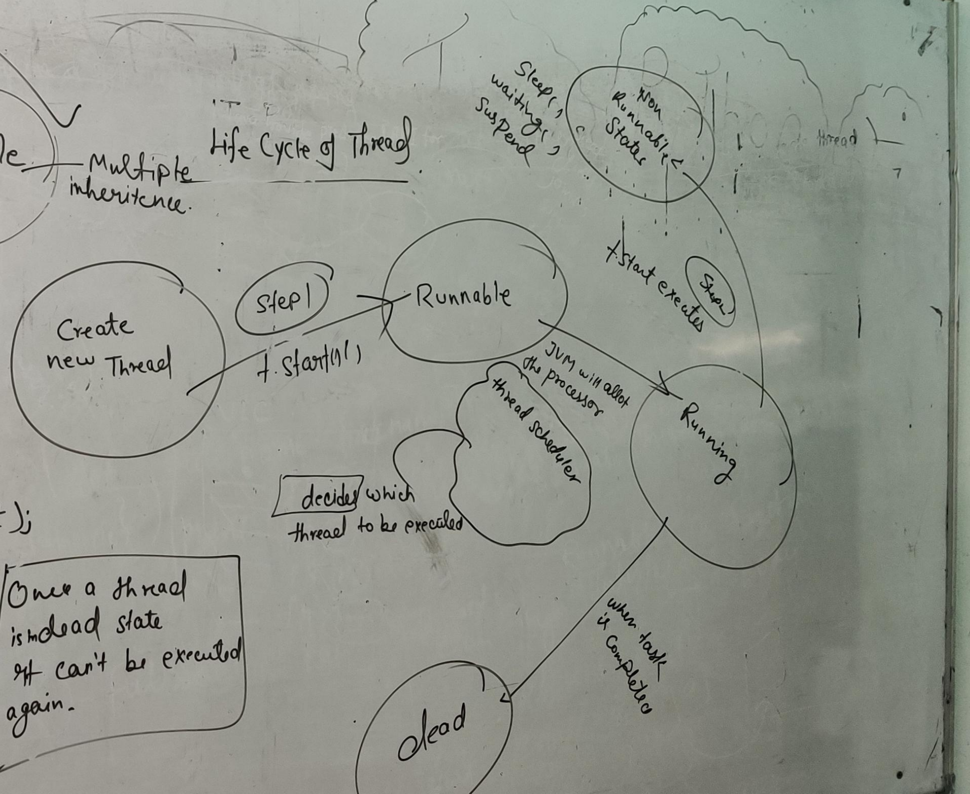
```
Class Test extends Thread
{
  public void run( )
  {
  S.o.pln(" task thread");
  }
  psvm(String[] arg)
  {
    Test t = new Test( );
    t.start( );
    t.start( );
  }
}
```

Method II

```
Class Test implements Runnable
{
  public void run( )
  {
  S.o.pln("task thread")
  }
  psvm(String[] arg)
  {
   Test T = new Test( );
  {
   Thread th = new Thread(t);
   th.start( );
   }
}
```

Multiple inheritence.

Life Cycle of Thread



Once a thread is in dead state it can't be executed again.