# ::Multithreading::

**Multitasking:-**When more than one task execution simultaneously is called multitasking, the main advantage of multitasking is reduced response time and improved performance.

**There are two types of multitasking:-**

1) Process-based Multitasking.
2) Thread based Multitasking(Multithreading).

**Process-based Multitasking:-**When more than one task is executed and all task is separate independent process or program is called process-based Multitasking.

**Example:-**

The best Example Operating System level we performed Process-based multitasking. In our window operating system performed several tasks simultaneously like listening to music from the media player, Programming practise on some editor/IDE, Internet browsing with chrome etc. when we closed media player there is no any impact on another running task because all task is a separate independent process.

**Key point of Process based Multitasking:-**

Each process has separate memory allocation.

Process is heavyweight; Heavyweight means maximum utilization of System resources.

Cost of communication between the processes is very expensive.

**Thread based Multitasking (Multithreading):-** When more than one task is executed and all task is separate independent part of same program is called Thread based Multitasking (Multithreading).

Programming level best uses of thread based multitasking.

VIKAS JAVA

## Key Point of Thread based Multitasking:-

Each thread shares common Memory address.

Thread based Multitasking is Lightweight; lightweight means minimum uses of system resources.

Cost of communication between the thread is very low.

## Application area of multithreading:-

Developed multimedia Graphics, animation, video games, web-server, application sever etc.

## Thread:-

Thread is flow of execution every thread has separate independent job.

The independent execution is known as thread.

A thread is lightweight process.

## There are two ways to create a Thread.

1) By Extending java.lang.Thread class.
2) By implementing java.lang.Runnable Interface.

## First approach creating to create Thread by extending Thread class:-

```
class MyThread extends Thread
{
public void run()
{
for(int i=0;i<10;i++)
{
System.out.println("Child Thread");
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
MyThread mt=new MyThread();
```

VIKAS JAVA

```
mt.start();➔ starting a thread
for(int i=0;i<10; i++)
{
System.out.println("main Thread");
}
}
}
```

**Result:-**Mixed-Output

## Overriding of run method:-

It is compulsory to override run method of Thread class otherwise we can't perform Multithreading.

When we are not overriding run method then start() method invoked Thread class run() method which is empty implementation hence we won't get any output.

**Example:-**

```
class MyThread extends Thread
{}
class ThreadDemo
{
public static void main(String[]args)
{
MyThread mt=new MyThread();
mt.start();
}
}
```

**Result:-**no output.

## Overriding of start() method:-

When we override start() method then our start method will be executed just like a normal method call and a new thread not be created. If we want to perform multithreading then the Thread class start() method must be invoked.

**Example:-**In this program there is no new thread will be created and the start method is executed first then remain code, we always get the same type of output.

VIKAS JAVA

```
class MyThread extends Thread
{
public void run()
{
System.out.println("run method MyThread class");
}
public void start()
{
for(int i=0;i<10;i++)
{
System.out.println("start method");
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
MyThread mt=new MyThread();
mt.start();
for(int i=0;i<10;i++)
{
System.out.println("main method");
}
}
}
```

**Result:-**

```
start method.........10times
then main method............10times
```

## Life-cycle of Thread:-

Basically there are four stage of thread Life Cycle but blocking/waiting/non-running mode we discussed later.

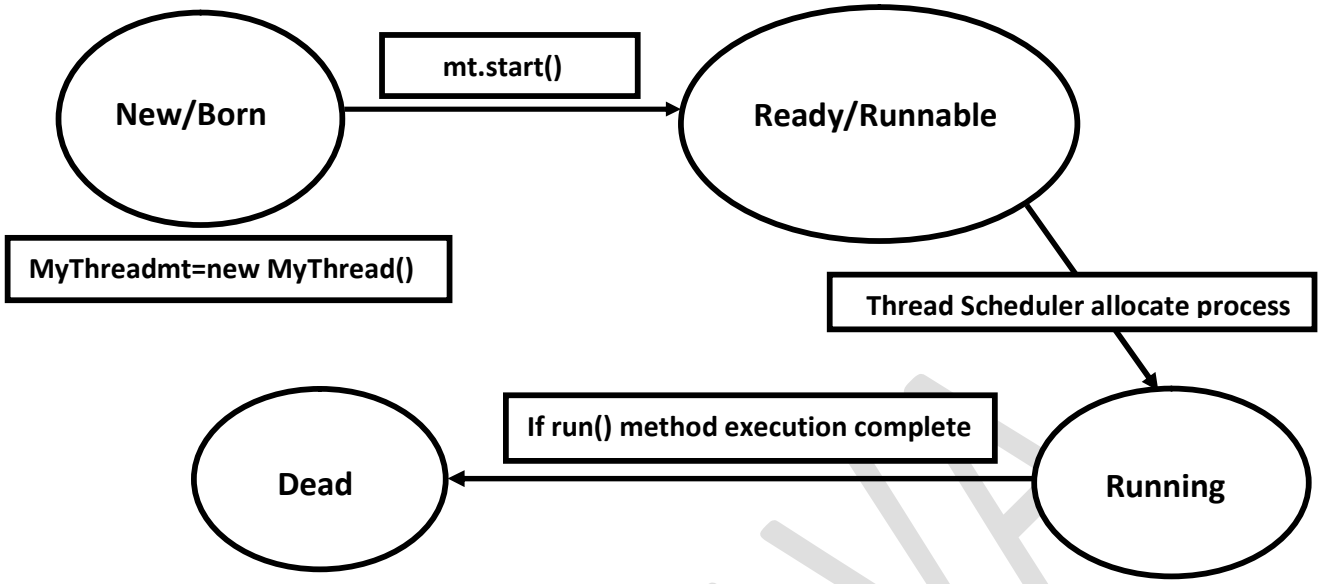
New/Born

Ready/Runnable

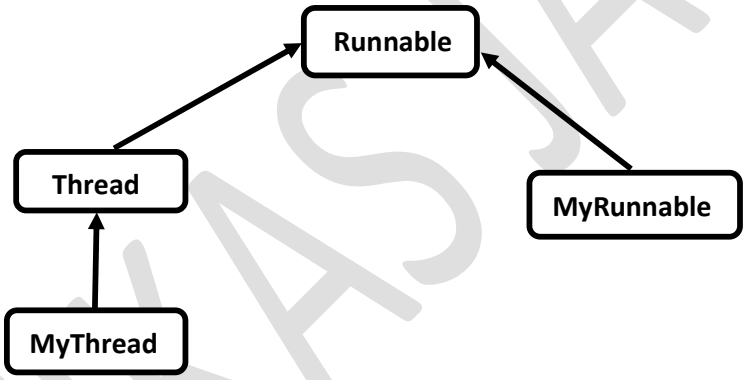
Running

Dead

VIKAS JAVA




## Creating a thread by implementing Runnable Interface:-




Runnableinterface present in java.lang package which contain only one run() method.

## The internal implementation of Multithreading:-

```
interface Runnable
{
public abstract void run();
}
class Thread implements Runnable
{
public void run()
{
}
}
```

```
class MyThread
{
public void run()
{
System.out.printlnln(“Hello Thread”);
}
}
```

**Example:-**

```
class MyRunnable implements Runnable
{
public void run()
{
for(int i=0;i<10;i++)
{
System.out.println("Child Demo");
}
}
}
class RunnableDemo
{
public static void main(String[]args)
{
MyRunnable r=new MyRunnable();
Thread t=new Thread(r);
t.start();
for(int i=0;i<10;i++)
{
System.out.println("main Thread");
}
}
}
```

**Result:-**mixed output

**Note:-**In this program when we try to start a thread by using r.start() then we will get compile time error saying:- Test.java:16: error: cannot find symbol
r.start();

**Symbol:-**method start()
location: variable r of type MyRunnable
because start method is not present in Runnable interface therefore we create Thread object and r passes as argument.

VIKAS JAVA

**Example:-**

```
MyRunnable r=new MyRunnable();
Thread t1=new Thread();
Thread t2=new Thread(r);
```

**Case1:-**t1.start():-In this case new thread will be created and which is responsible to execute Thread class run method which is empty implementation.

**Case2:-**t1.run() In this case no any new thread will be create and Thread class run method will be executed just like normal method call.

**Case3:-**t2.start():-In this case new thread will be created and which is responsible to MyRunnable class overriding run method will be executed.

**Case4:-**t2.run():-In this case there is no any new thread will be create and MyRunnable class run method will be executed just like normal method call.

**Case5:-**r.start():-In this case we will get compile time error because start method is not available Runnable interface.

**Case6:-**r.run():-In this case no any new thread will be created and MyRunnable class run method will be executed just like normal method call.

## Create Multithreading by extending Thread class and by implementing Runnable interface which one is best approach:-

**Approach First:-**In this case if we extending Thread class then we can't extends any other class so we missed the inheritance concept.

**Second approach:-**In this case If implements an interface so one possibility remain we can extends a class in future.

It is highly recommended to implements interface in place of extends class.

VIKAS JAVA

**Performed Single Task by multiple Threads:-**In this program multiple mt1, mt2 and mt3 are performed single task.

```
class MyThread extends Thread
{
public void run()
{
for(int i=0;i<10;i++)
{
System.out.println("MyThread Run...");
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
MyThread mt1=new MyThread();
MyThread mt2=new MyThread();
MyThread mt3=new MyThread();
mt1.start();
mt2.start();
mt3.start();
for(int i=0;i<10;i++)
{
System.out.println("main Thread");
}
}
}
```

**Output:-**Mixed Output

**Multiple Threads Performed Multiple Tasks:-**In this program multiple thread mt1, mt2 and mt3 are performed different task.

```
class MyThread1 extends Thread
{
public void run()
{
for(int i=0;i<3;i++)
{
System.out.println("MyThread1 Run");
}
}
}
```

VIKAS JAVA

```
class MyThread2 extends Thread
{
public void run()
{
for(int i=0;i<3;i++)
{
System.out.println("MyThread2 Run");
}
}
}
class MyThread3 extends Thread
{
public void run()
{
for(int i=0;i<3;i++)
{
System.out.println("MyThread3 Run");
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
MyThread1 mt1=new MyThread1();
MyThread2 mt2=new MyThread2();
MyThread3 mt3=new MyThread3();
mt1.start();
mt2.start();
mt3.start();
}
}
```

**Result:-**Mixed output.

## Setting and Getting name of Thread:-

In java Every Thread having some it may generate by JVM or user defined.

By default, main thread name is main it generates by JVM.
There are two following method of Thread class through which we get and set thread name.

1) public final String getNmae()

VIKAS JAVA

2) public final void setName(String ThreadName)

**Example:-**

```
class MyThread extends Thread
{
}
class ThreadDemo
{
public static void main(String[]args)
{
System.out.println(Thread.currentThread().getName());
Thread.currentThread().setName("abc");
System.out.println(Thread.currentThread().getName());
MyThread mt1=new MyThread();
MyThread mt2=new MyThread();
System.out.println(mt1.getName());
mt1.setName("xyz");
System.out.println(mt1.getName());
}
}
```

Result:- main
main
abc
Thread-0
xyz

**Note:-**We get current Thread object by using Thread.currentThread() method.

**Example:-**

```
class MyThread extends Thread
{
public void run()
{
for(int i=0;i<3;i++)
{
System.out.println("Run method executed by=
"+Thread.currentThread().getName());
}
}
}
class ThreadDemo
{
```

VIKAS JAVA

```
public static void main(String[]args)
{
MyThread mt=new MyThread();
mt.start();
for(int i=0;i<3;i++)
{
System.out.println("main method executed by=
"+Thread.currentThread().getName());
}
}
}
```

**Result:-**

```
main method executed by= main
Run method executed by= Thread-0
main method executed by= main
Run method executed by= Thread-0
main method executed by= main
Run method executed by= Thread-0
```

## Thread Priority:-

Every Thread in java has some priority it may be default generate by JVM or user defined thread priority.

The range of thread priority is 1 to 10. Where 1 (One) is lowest Thread priority and 10 (Ten) highest thread priority.

**Thread class defined some constant value for Thread priority.**

a) MIN_PRIORITY=0
b) NORM_PRIORITY=5
c) MAX_PRIORITY=10

**There are two following method of Thread class through which we get and set thread priority.**

1) public final int  getPriority()
2) public final void setPriority(int priority)

**Example:-**

```
class MyThread extends Thread
{
public void run()
{
for(int i=0;i<3;i++)
```

VIKAS JAVA

```
{
System.out.println(Thread.currentThread().getName()+" =
"+Thread.currentThread().getPriority());
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
System.out.println("current Thread
Priority="+Thread.currentThread().getPriority());
MyThread mt=new MyThread();
mt.setPriority(8);
mt.start();
for(int i=0;i<3;i++)
{
System.out.println(Thread.currentThread().getName()+" =
"+Thread.currentThread().getPriority());
}
}
}
```

**Result:-**

```
current Thread Priority=5
main = 5
Thread-0 = 8
main = 5
Thread-0 = 8
main = 5
Thread-0 = 8
```

The default thread priority only for main Thread is 5(Five) and remaining thread priority will be inherited from parent Thread to Child Thread.

If two or more thread same priority then we can't expect exact order of thread execution its depend upon Thread Scheduler.

**Example:-**

```
class MyThread extends Thread
{
public void run()
```

VIKAS JAVA

```
{
for(int i=0;i<3;i++)
{
System.out.println("Child thread");
}
}
}
class ThreadDemo
{
public static void main(String[]args)
{
System.out.println(Thread.currentThread().getPriority());
MyThread mt=new MyThread();
mt.setPriority(8);//Line-----17
mt.start();
for(int i=0;i<3;i++)
{
System.out.println("Main thread");
}
}
}
```

**Result:-**In this program Child Thread executed first then parent thread but when we comment line-17 then both thread priority is same because thread priority inherited from parent to child and main thread priority 5 so child thread priority also 5.

**Prevent Thread Execution:-**Based on our requirement we prevent the thread of execution by using following method:-

1) yield() method
2) join() method
3) sleep() method

## sleep() method:-

Based on our requirement if we want a thread don't performed any operation for a particular amount of time we used sleep() method.

sleep() method is static method so we can call this method directly by using class name.

sleep() method always throw checked exception so should compulsory handle it otherwise we will get compile time error.

VIKAS JAVA

**Prototype of sleep() method:-**

1) public static native void sleep(long millisecond )throws InterruptedException
2) public static void sleep(long millisecond, intnanosecond)throws InterruptedException

**Example:-**

```
class MyThread extends Thread
{
public void run()
{
for(int i=0;i<5;i++)
{
System.out.println("Number Iteration="+i);
try
{
Thread.sleep(1000);
}
catch(InterruptedException e)
{
System.out.println("I got interrupted");
}
}P
}
public static void main(String[]args)
{
MyThread mt=new MyThread();
mt.start();
}
}
```

**Result:-**run() method sop statement 5 times will be executed

VIKAS JAVA