Multithreading:-

Introduction

Multitasking

-The process of performing more than one task parallelly is called as Multitasking
-It is of 2 types
1.Process based multitasking
2.Thread based multitasking

process based multitasking

-The process of executing more than one task where each task is independent of other

Ex:browsing on google---Task-1
   watching something on youtube---Task-2
   using excel---Task-3
   typing something on word---Task-4
Thread Based Multitasking

-The process of executing more than one task parallelly where each task is an
 independent part of same program such type of processing is called as thread based multitasking
 or thread programming.

-For Ex: if we have a program which contains 1000lines of code but what we observed is
         next 500 is independent of first 500,but still it has to wait untill
         first 500 finishes its execution so due to this
       -perormance is decreasing
       -execution time is increasing
       -CPU utilisaion time is decreasing

      COPY THE CONTENT OF IMAGE
-Therefore in order to overcome above situations we will go for Thread programming

Thread:

-it is a flow of execution OR
-it is a small part of an Application   OR
-it is a light weight process

-We will create 1-thread for first 500lines
-We will create 2-thread for second 500lines and run both the threads parallely

-Programs which contains multiple threads is called as multi threaded program and
 such process is called as Multithreading

CREATION OF THREAD

-A thread can be created in two ways
1.By extending Thread class
2.By implementing Runnable Interface

Note:-

-In every program always there is one default thread ie main thread(main()).

CREATING THREAD BY EXTENDING THREAD CLASS

-Create our class which extends Thread class
-For defining thread we have to override run()

```
  public void run();
```

-it is a predefine method of thread class

-Using start() we can start execution of thread

```
  public synchronized void start();
```

Note:

Synchronized is a keyword which indicates that only one thread can access method at a time.

program

```
package Multithreding.com;
class Mythread extends Thread
{//@override run() of thread class for defining thread
      public void run()
      {
             //JOB of thread
            for(int i=0;i<=5;i++)
            {
                  System.out.println("MyThread");
            }
      }}
public class User {
       //default thread of everyprogram
      public static void main(String[] args) {
            //@job of main() thread
         Mythread t1=new Mythread();
         t1.start();//creates a thread by calling run() and make it available for
exe
         //after this stmt ther are two threads under exe main and mythread
         for(int i=0;i<=5;i++)
         {
             System.out.println("Main  Thread");
         }
      }
```

Explanation

-In above program execution starts from main thread and till t1.start() there is only one thread under execution once t1.start() is invoke it creates a thread and calls run()
-Now two threads are there for execution main thrread and mythread
-SO out of two thread which thread to select first for execution is decided by THREAD SCHEDULER.

-THREAD SCHEDULER is nothing but JVM and it is upto JVM whichever algorithm it follows and select a thread for execution.
-Since we don't know on what basis thread scheduler picks a thread for execution we cannot predict the output of program.

CREATING A THREAD BY IMPLEMENTING RUNNABLE INTERFACE:-

Runnable Interfae

```
public interface java.lang.Runnable {
  public abstract void run();
}

Program
------------
package Multithreding.com;
class Mythread implements Runnable
{//@override run() of thread class for defining thread
      public void run()
      {
             //JOB of thread
            for(int i=0;i<=5;i++)
            {
                  System.out.println("MyThread");
            }
      }}
public class User {
       //default thread of everyprogram
      public static void main(String[] args) {
            //@job of main() thread
         Mythread t1=new Mythread();
        // t1.start();//CTE because there is no start() in Mythread class
         Thread t2=new Thread(t1);
         t2.start();//creates a thread by calling run()
         //after this stmt ther are two threads under exe main amd mythread
         for(int i=0;i<=5;i++)
         {
             System.out.println("Main  Thread");
         }
      }

}
```

Q.Out of two ways of creating a Thread which one is preferrable?
A.Second way of thread creation is preferrable because when we create a thread by
  implementing Runnble interface at same time we can extend any other base class
also
  but if we create a thread by extending thread class sub class cannot extend any
other class.

```
//Multiple independent threads
package Multithreding.com;
public class Uset1 {
      public static void main(String[] args) {
    Mythread1 t=new Mythread1();
    Mythread11 t1=new Mythread11();
    Mythread12 t2=new Mythread12();
    t.start();
    t1.start();
    t2.start();
    for(int i=0;i<5;i++)
    {
      System.out.println("Poooja 1.0");
    }
}}
class Mythread1 extends Thread
{
      public void run()
      {
            for(int i=0;i<5;i++)
```

```
          {
            System.out.println("Poooja 2.0");
          }
}}
class Mythread11 extends Thread
{
      public void run()
      {
            for(int i=0;i<5;i++)
          {
            System.out.println("Poooja 3.0");
          }
}}
class Mythread12 extends Thread
{
      public void run()
      {
            for(int i=0;i<5;i++)
          {
            System.out.println("Poooja 4.0");
          }
}}
```

```
Output
--------
Poooja 1.0
Poooja 1.0
Poooja 1.0
Poooja 1.0
Poooja 1.0
Poooja 4.0
Poooja 4.0
Poooja 4.0
Poooja 4.0
Poooja 4.0
Poooja 3.0
Poooja 3.0
Poooja 3.0
Poooja 3.0
Poooja 3.0
Poooja 2.0
Poooja 2.0
Poooja 2.0
Poooja 2.0
Poooja 2.0
```

```
Execution
------------
main---->task-1
Mythread1--->task2
Mythread11---->task3
Mythread12------>task4
```

Q.Can we Resart a thread?
A.No We cannot restart a thread,if we do we will get IllegalThreadStateException

```
Ex:MyThread m1=new MyThread();
   m1.start();//Valid
   m1.start();//Exception
Exception in thread "main" java.lang.IllegalThreadStateExceptionMyThread
      at java.lang.Thread.start(Unknown Source)

MyThread
MyThread
MyThread
MyThread
      at Multithreding.com.User.main(User.java:19)
```

Internal Implementation of Threads

```
interface Runnable
{
 public abstract void run();
}
public class Thread implements Runnable
{
 public void run()
 {
   // No implementation
 }
}
class MyThread extends Thread
{
 public void run()
 {
   //define job of thread//
 }
}
```

Life Cycle of Thread

-Depending on different phases a Thread will be in any of one phase:
1.New
2.Runnable/Ready
3.Running
4.Blocked
5.Terminated

New:

When we create object of our thread class

```
MyThread m1=new Mythread();
```

Ready:

When we call run() by using start(),but before thread schedule picks
 that thread for execution

```
     m1.start();
```

Running

When thread scheduler(CPU) picks thread for execution(run() execution started)

Blocked

-If the running thread goes to sleeping state.

Terminated or dead

-When execution of run() is completed.

Thread.yield()

-yield() causes to pause current executing thread for giving chance to waiting threads
 of same priority.

-if there are no waiting threads or all threads are having low priority
 then same thread will continue its execution once again.