MULTITHREADING IN JAVA

What is multithreading?

The process of executing multiple tasks (also called threads) simultaneously is called multithreading. The primary purpose of multithreading is to provide simultaneous execution of two or more parts of a program to make maximum use of CPU time. A multithreaded program contains two or more parts that can run concurrently. It enables programmers to write in a way where multiple activities can proceed simultaneously within a single application.

What is Multitasking?

It is the way of executing multiple tasks at a time executing them concurrently over a specified period. Multitasking is done in two ways. These are:

1. Process-based multitasking: It is also called multiprocessing where each process has its address in memory, i.e., each process allocates separate memory area.
2. Thread-based multitasking: This thread-based multitasking is also termed as multithreading where threads share the same address space.

Life Cycle of a Thread

A thread goes through various stages of its life cycle. Example, first of all, a thread is born, started its tasks, run a sequence of tasks concurrently, and then dies. Here is the diagram of the various stages of a life cycle.

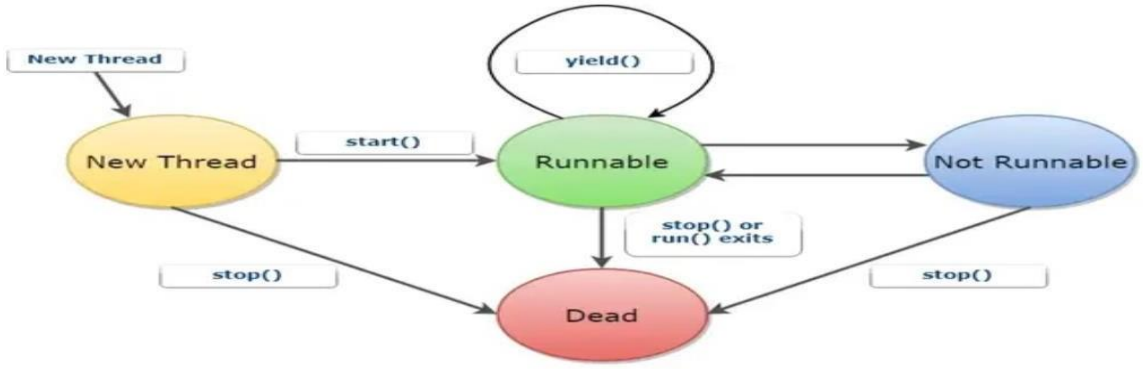

Fig: Life Cycle of a Thread in Java

1. New Thread: A new thread begins its life cycle in the new state. The process remains in thiscondition until the program starts the thread.
2. Runnable: As soon as the new thread starts, the thread status becomes Runnable. At this stage, athread is considered to execute its function or working.
3. Not Runnable: A Runnable thread when entered the time of waiting for the state for a specificinterval of time. That time, the thread is not in Runnable condition.
4. Dead / Terminated: The Runnable thread enters the end stage when it completes its tasks.

Thread properties

Java assigns each thread a priority that concludes that how a thread will be treated concerning others.Thread priorities are integers that specify the relative priority of one thread with another. Thread priorities are used for deciding when to switch from one running thread to another. It is called
a **context switch**.

Here is simple program to demonstrate

Multithreading in javaExample

```
class MultiThread
   extends Thread{
   public void run(){
      System.out.println("Running Thread Name: "+
      this.currentThread().getName()); System.out.println("Running
      Thread Priority: "+ this.currentThread().getPriority());
   }
}
public class MultiThrd {
   public static void main(String[] args)
      { MultiThread multiThread1 = new
      MultiThread();
      multiThread1.setName("First
      Thread");
      multiThread1.setPriority(Thread.MIN_PRIORITY);

      MultiThread multiThread2 = new
      MultiThread();
      multiThread2.setName("Second Thread");
      multiThread2.setPriority(Thread.MAX_PR
      IORITY);

      MultiThread multiThread3 = new
      MultiThread();
      multiThread3.setName("Third
      Thread");

      multiT
      hread1
```

```
        .start()
        ;
        multiT
        hread2
        .start()
        ;
        multiT
        hread3
        .start()
        ;
    }
}
```

Suspending, Resuming and Stopping Threads

Sometimes it is useful to suspend the execution of a thread. Example, suppose there is a thread that is used to display the time of day. If the user of that program does not want that clock, then the thread canbe suspended. Once suspended, restarting that thread is also a simple matter, which is called resuming the thread. Here are the lists of pre-defined functions used in Java for performing suspending, resumingand stopping threads respectively. They are:

1. suspend()
2. resume()
3. stop()