Java – Multithreading

* **New** − A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a **born thread**.
* **Runnable** − After a newly born thread is started, the thread becomes runnable. A thread in this state is executing its task.
* **Waiting** − Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
* **Timed Waiting** − A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
* **Terminated (Dead)** − A runnable thread enters the terminated state when it completes its task or otherwise terminates.

**Thread Priorities**

Every java thread has a priority that helps the operating system determine the order in which threads are scheduled.

Java thread priorities are the range between MIN\_PRIORITY (a constant of 1) and MAX\_PRIORITY (a constant of 10), By default, every thread is given priority NORM\_PRIORITY (a constant of 5). Create a

Tread by Implementing a Runnable Interface

If your class is intended to be executed as a thread, then you can achieve this by implementing a **Runnable** interface. You will need to follow four basic steps −

* Step 1

As a first step, you need to implement a **run ()** method provided by a **Runnable** interface. This method provides an entry point for the thread and you will put your complete business logic inside this method - “**run ()”**

* Step 2

Create Object reference of this class

* Step 3

Creating reference of the Thread class Object and pass in to it a constructor of this class

* Step 4

Call method **start ()** using Thread reference that was created in step 3

Method & Description

* **public void start()**

Starts the thread in a separate path of execution, then invokes the **run()** method on this **Thread** object.

* **public void run()**

If this **Thread** object was instantiated using a separate **Runnable** target, the **run()** method is invoked on that **Runnable** object.

* **public final void setName(String name)**

Changes the name of the **Thread** object. There is also a **getName()** method for retrieving the name.

* **public final void setPriority(int priority)**

Sets the priority of this **Thread** object. The possible values are between 1 and 10.

* **public final void join(long millisec)**

The current **Thread** invokes this method on a second thread, causing the current thread to block until the second thread terminates or the specified number of milliseconds passes.

* **public final boolean isAlive()**

Returns true if the **thread** is alive, which is any time after the **thread** has been started but before it runs to completion.

* **public static void sleep(long millisec)**

Causes the currently running **thread** to block for at least the specified number of milliseconds.

* **public static Thread currentThread()**

Returns a reference to the currently running **thread**, which is the **thread** that invokes this method.