**Lifecycle and States of a Thread in Java**

A thread in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

* New
* Runnable
* Inactive
* Timed Waiting
* Terminated

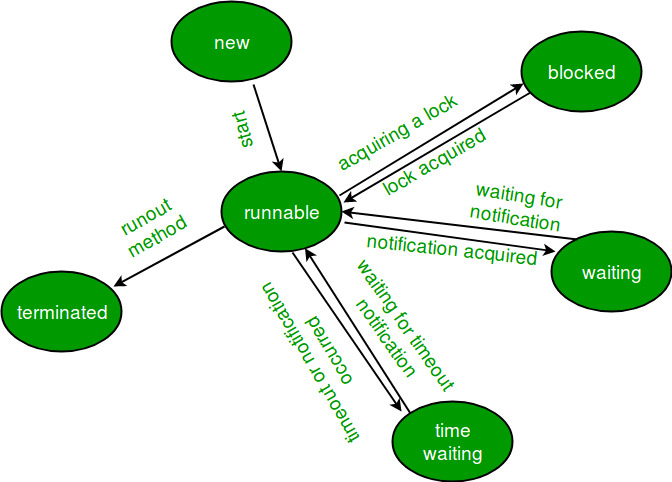


Figure: The diagram represents the lifecycle & various states of a thread.

**New Thread:**

When a new thread is created, it is in the new state. The thread has not yet started to run when thread is in this state. When a thread lies in the new state, it’s code is yet to be run and hasn’t started to execute.

**Runnable State:**

A thread that is ready to run is moved to runnable state. In this state, a thread might actually be running or it might be ready run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run.

A multi-threaded program allocates a fixed amount of time to each individual thread. Each and every thread runs for a short while and then pauses and relinquishes the CPU to another thread, so that other threads can get a chance to run. When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lies in runnable state.

**Inactive State:**

When a thread is temporarily inactive, then it’s in one of the following states:

**Blocked:**

For example, when a thread is waiting for I/O to complete, it lies in the blocked state. It’s the responsibility of the thread scheduler to reactivate and schedule a blocked/waiting thread. A thread in this state cannot continue its execution any further until it is moved to runnable state. Any thread in these states do not consume any CPU cycle.

**Waiting:**

A thread is in the blocked state when it tries to access a protected section of code that is currently locked by some other thread. When the protected section is unlocked, the schedule picks one of the thread which is blocked for that section and moves it to the runnable state. Whereas, a thread is in the waiting state when it waits for another thread on a condition. When this condition is fulfilled, the scheduler is notified and the waiting thread is moved to runnable state.

If a currently running thread is moved to blocked/waiting state, another thread in the runnable state is scheduled by the thread scheduler to run. It is the responsibility of thread scheduler to determine which thread to run.

**Timed Waiting:**

A thread lies in timed waiting state when it calls a method with a time out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to timed waiting state.

**Terminated State:**

A thread terminates because of either of the following reasons:

* Because it exists normally. This happens when the code of thread has entirely executed by the program.
* Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.

A thread that lies in terminated state does no longer consumes any cycles of CPU.