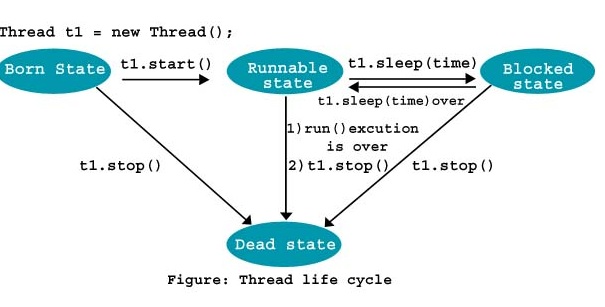
Thread Life cycle Java

**Generally, the life cycle gives the same meaning for Servlets, Applets and Threads. Different states, a thread (or applet/servlet) comes under from its object creation to object removal (garbage collection) is known as life cycle of thread (or applet/servlet). There are four main states in the Thread Life cycle.**

1. Born state
2. Runnable state
3. Blocked state
4. Dead state

**Bird view of the states**



**1. Born state** (of Thread Life cycle)

Thread t1 = new Thread();

In **born state**, the thread object is created, occupies memory but is inactive. In the above statement **t1** thread is created but is not eligible for microprocessor time as it is inactive. To make the thread active, call **start()** method on the thread object as **t1.start()**. This makes the thread active and now eligible for processor time slices. This state can be compared with start() method of **applets**.

**2. Runnable state** (of Thread Life cycle)

When the thread is active, the first and foremost job, it does implicitly, is calling **run()**method. When the thread is executing the run() method, we say, the thread is in **runnable state**. As it is a callback method, all the code expected to run by the thread, should be written here. In this state, the thread is **active**. This state can be compared to the **paint()** method of**applets**.

**3. Blocked state**(of Thread Life cycle)

The programmer can make a running thread to become inactive temporarily for some period. In this period (when inactive after starting), the thread is said to be in **blocked state**. The blocked state thread, as inactive, is not eligible to processor time. This thread can be brought back to the runnable state at any time. A thread can go a number of times from runnable state to blocked state and vice versa in its life cycle. This state can be compared with the**stop()** method of **applets**.

**4. Dead state**(of Thread Life cycle)

When the execution of run() method is over, as the job it is meant is done, it is brought to**dead state**. It is done implicitly by JVM. In dead state, the thread object is garbage collected. It is the end of the life cycle of thread. Once a thread is removed, it cannot be restarted again (as the thread object does not exist). This state can be compared with **destroy()** method of**applets**.

A thread can be killed and brought to **dead state**, anytime from any state, by calling explicitly**stop()** method.

**Number of ways a thread can be brought to blocked state**

A normal running thread can be brought, a number of ways, into blocked state, listed hereunder. By calling sleep() method.

1. By calling suspend() method.
2. When wait() method is called as in synchronization
3. When an I/O operation is performed, the thread is implicitly blocked by JVM.

**IllegalStateException:**This exception is thrown when you try to start a thread which is already started.