Thread Lifecycle

1. New State

A thread is created but has not started

Ex. Thread t = new Thread();

1. Runnable State

The thread is ready to run but may not be running immediately because the CPU may be executing some other threads.

Ex. After calling t.start(); , the thread moves to runnable state.

1. Running State

The thread is actively running.

The thread scheduler decides when a thread moves from the Runnable state to the running state

1. Blocked/Waiting State

The thread is waiting for signal to proceed.

It is temporarily inactive.

Ex. A thread may wait using t.wait();

1. Terminated

The thread has finished its execution and cannot run again.

Ex. Once the run() method completes, the thread enters the Terminated State.

package ThreadLifecycle;  
  
public class MyThread extends Thread {  
 @Override  
 public void run(){  
 System.*out*.println("Thread is in the Running State");  
 try{  
 Thread.*sleep*(1000); // Thread is moved to Wait state  
 } catch (InterruptedException e) {  
 System.*out*.println("Thread interrupted");  
 }  
 System.*out*.println("Thread is about to terminate");  
 }  
}

package ThreadLifecycle;  
  
import ThreadExamples.MyThread;  
  
public class ThreadLifecycleDemo {  
 public static void main(String[] args) {  
 //Thread is in New State  
 MyThread thread = new MyThread();  
 System.*out*.println("Thread is in New State");  
  
 // Move to RUNNABLE state  
 thread.start();  
 System.*out*.println("Thread is in Runnable state");  
  
 //Main thread waits for the thread to complete  
 try {  
 thread.join();  
 } catch (InterruptedException e){  
 e.printStackTrace();  
 }  
  
 System.*out*.println("Thread has terminated");  
 }  
}