Lesson 2 : Thread Creation/Lifecycle

Saturday, April 27, 2024 3:51 PM

Thread can be created by

1. Extending Thread class
2. Implementing Runnable interface (Functional interface, SAM is run())

Why 2 ways, because only one class can be extended at one time but multiple interfaces can be implmented

Thread class itself implements runnable interface. Thread class has following methods

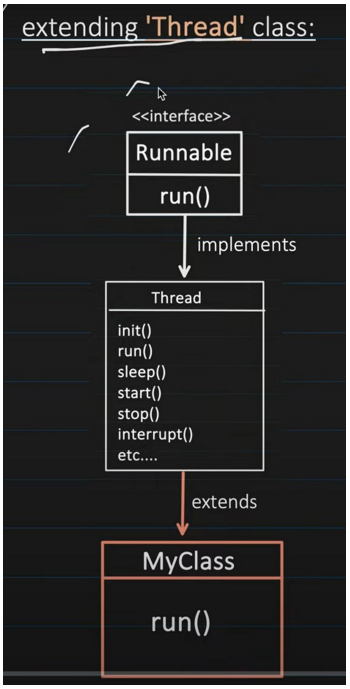
1. init()
2. run() (implemented from Runnable)
3. sleep(), start(), stop(), interrupt() etc

1st method :

Create a class that implements Runnable. Add implementation of run() method. Create object of the class and pass it as parameter in new Thread contructor. Start the thread by invoking start() method. Start() internally invokes run() method
Code example in repo

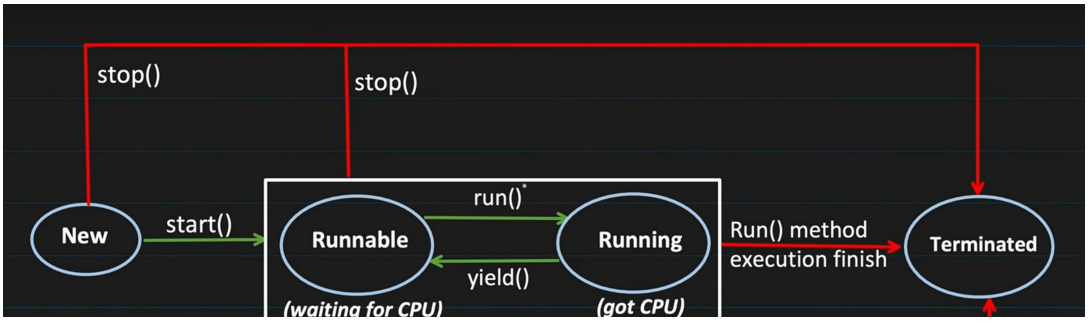
In production code usually threads are used by implementing runnable

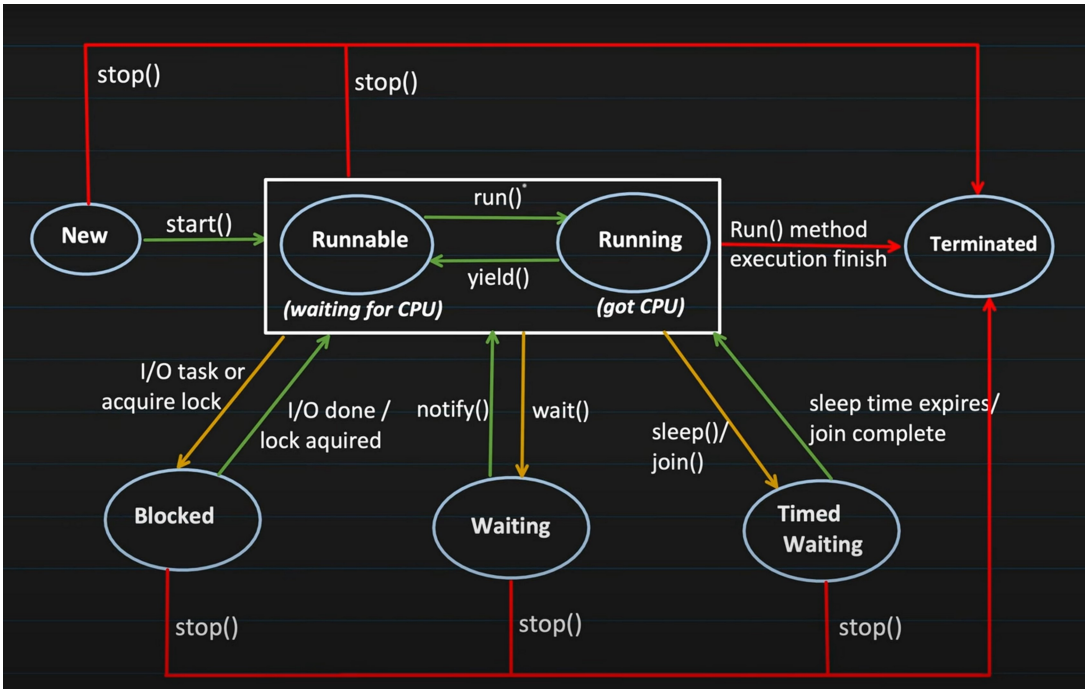
2nd method :



Create a class that extends Thread class, add your code in run() method.
Now in main create an instance of the class and invoke start() method.

THREAD LIFECYCLE





Runnable - Thread is either running or waiting for CPU time. Puts monitor lock
Blocked -- Waiting for I/O task like reading from a file/db, or waiting to acquire a lock. When thread is in blocked state it releases all the monitor locks
Waiting - when wait() is invoked, it becomes non-runnable. It goes back to runnable once we call notify() or notifyAll() method. Releases all the monitor locks
Timed Waiting - Thread waits for a specific period of time and comes back to runnable state, after specific conditions are met like sleep(), join(). Does not release monitor locks
Terminated - Life of thread is completed, it can not be started again.

MONITOR LOCKS : It helps to ensure that only 1 thread goes inside the particular section of code (a synchronized block or method). Each object has monitor lock. See code L2
So even if the methods are different but they share same instance of object thus monitor lock gets used. Note : All calls to synchronized methods must be made from **same instance of object** otherwise it will be treated as different locks.

Use of wait, notifyAll is shown in code examples.
Most things are present in code examples. Go through Producer Consumer solution it is very important