MultiThreading in Java

Enabling multiple threads or units of execution to run concurrently within a single program or process.

To achieve multithreading in java u have:

- Thread class
- Runnable Interface

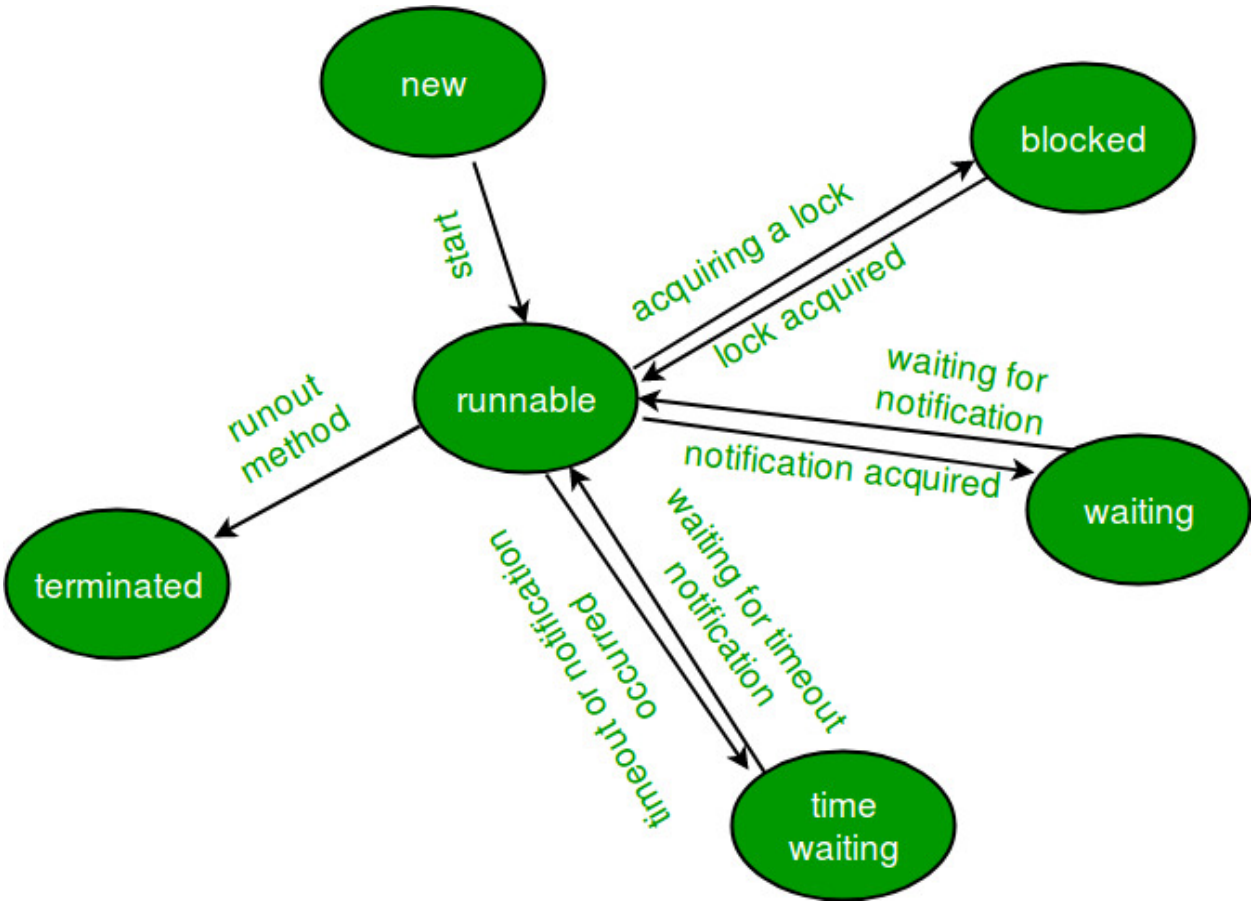
Thread Class:

- So class must inherit from thread class
- For writing the functionality of thread u have to override run method
- So now in main method u create a object of ur class and to start the thread u have to call start method(obj.start();)

Runnable Interface:

- So class must implement runnable interface and u have to override run method
- now u create an object of class inside ur main function and u also create a object of thread and attach the object(one u created be4)

States of thread:



- **NEW** :When u create object of thread it is new state
- **READY** :if u want thread to run u will call start method it will enter into ready state that is ready to run
- **RUNNING** : so it will call overridden run method so enter running state
- **TERMINATED**: when thread is killed or u want to end it ,it enters into terminated state it means u r killing that thread
- **WAITING**: now if u want to access a resource file or if the present thread is waiting for some other thread u will go to wait state
- **TIMED WAIT**: if thread wants to delay or work slowly u go to timed wait (delay of thread)→sleep or wait
- **BLOCKED**: if a thread wants to access a resource then it requests I/o that is when it goes to blocked state which is just like waiting state and once it gets notified by thread it runs again

Thread Priorities

There are threads in ready queue which are ready to run where jvm scheduler manages it

each thread is executed by cpu so the execution of thread is managed by scheduler where each thread gets equal time slice ans all threads share the cpu time

Java supports priorities from 1 to 10 so If a thread has highest priority then scheduler gives higher time slice than other……

min pri is 1 and max is 10 and avg is 5

Multithreading is provided by OS but java has its own multithreading environment that is jvm scheduler schedules the threads

Thread Class and its Constructors, methods

Synchronisation

Resource Sharing: multiple threads sharing same resource file

Critical section: lines of code that accessing the shared object i,e few lines from thread 1nd other few lines fo thread2 accessing shared object

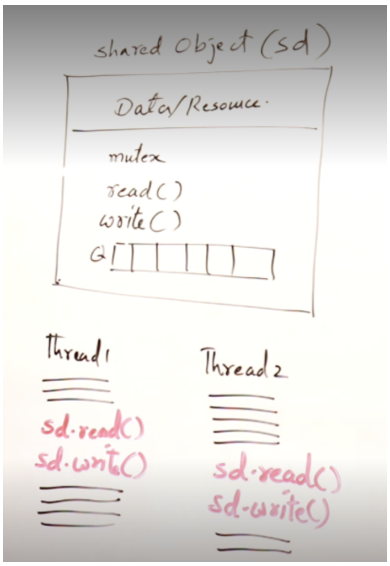
Mutual Exclusion: in the above example both the threads should access the shared object one after the another not simultaneously. so accessing of one thread prevents another

Mutex/lock : when a thread is using a resource it locks the resource file with mutex = 1 and if mutex = 0 it means a thread can access the resource → controlled by mutex

(atm - security)

Semaphore: it has a block queue with wait()→mutex = 1 and signal()→mutex = 1 so threads wanting to access the resource goes to block queue from ready queue once done comes back→controlled by os

Monitor:here java creates an object where all the above mechanism semaphore and mutex is in the object and is controlled by object



Inter thread communication

race condition- that is anyone thread can access resource there is no order

for prgms refer: https://medium.com/@aaqilrk/multithreading-in-java-7d5853153edd