**Thread Tutorial**

There are two ways in which you can accomplish

1.extend the thread class

2.implements the runnable interface

Extending the thread is not preferred because if you extend the thread class you will not be able to extend the any other class

And one more imp thing is there t's not as if

you really need that inherited Thread class behavior, because in order to use a thread

you'll need to instantiate one anyway

**what are the Diffrent Thread constructors ?**

■ Thread()

■ Thread(Runnable target)

■ Thread(Runnable target, String name)

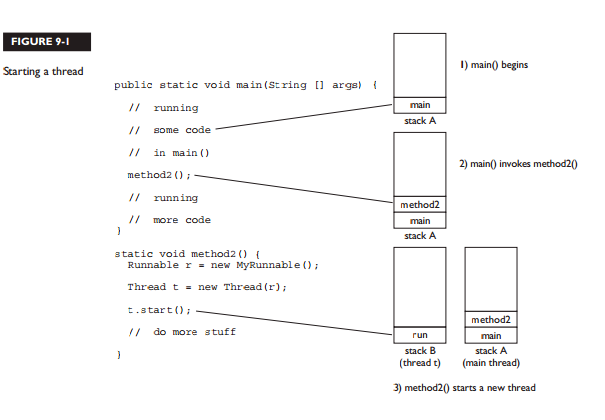
■ Thread(String name) You need to recognize all of them for the exam!

Once the start() method is called, the thread is considered to be alive (even though the run() method may not have actually started executing yet). A thread is considered dead (no longer alive) after the run() method completes. The isAlive() method is the best way to determine if a thread has been started but has not yet completed its run() method.

Very imp note :

Thread t = new Thread();

t.run(); // Legal, but does not start a new thread

****

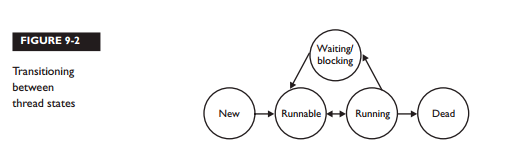
**Important point in threads**

Each thread will start, and each thread will run to completion.

A thread is done being a thread when its target run() method completes.

Once a thread has been started, it can never be started again.

**The order in which runnable threads are chosen to run is not guaranteed.**

****

1.New: thread instance has been created but start method is not yet been called

2.Runnable:this will enter when start method is called but the thread scheduler has not yet selected it for run

3.Running: this is state where execution of the thread is happening

4.waiting/blocking : this state can be entered say when in run method you called sleep,thread is waiting for another thread,thread is waiting for any IO resources from this state thread can enter it to runnable state when thread finishes the sleeping,another thread has notified that it can start it’s execution and IO resourace is available

5.when thread has finished it’s run method it’s called it’s in dead state

Don't rely on thread priorities when designing your multithreaded application. Because thread-scheduling priority behavior is not guaranteed, use thread priorities as a way to improve the effi ciency of your program, but just be sure your program doesn't depend on that behavior for correctness.

What is the yield method does it it put the thread to runnable state and puts another thread from pool which is off same as yielding thread to running state

Locking the objects :

1. Only methods (or blocks) can be synchronized, not variables or classes.
2. Each object has just one lock
3. Not all methods in a class need to be synchronized. A class can have both synchronized and non-synchronized methods.
4. If a thread goes to sleep, it holds any locks it has—it doesn't release them.
5. You can synchronize a block of code rather than a method.

class SyncTest {

public void doStuff() {

System.out.println("not synchronized");

synchronized(this) {

System.out.println("synchronized");

}

}

}

This is how you can have synchronized block