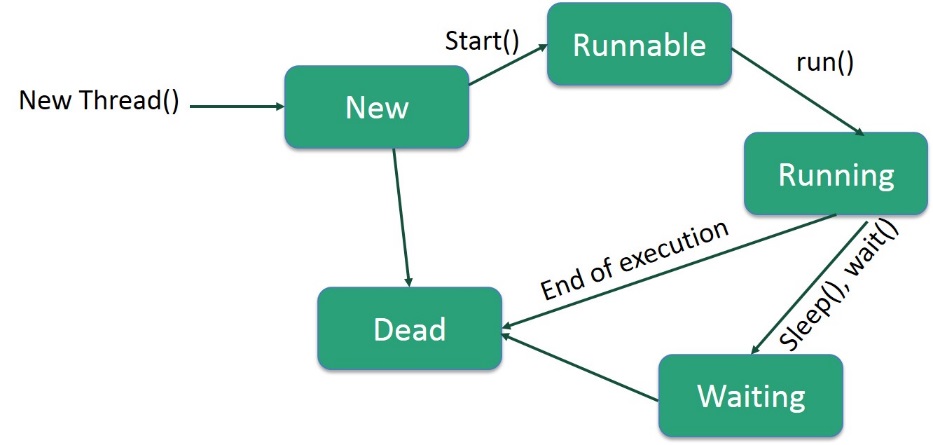
Thread:

<http://www.tutorialspoint.com/java/java_multithreading.htm>



Thread Priorities:

* MIN\_PRIORITY = 1
* NORM\_PRIORITY = 5
* MAX\_PRIORITY = 10

However, thread priorities cannot guarantee the order in which threads execute and very much platform dependent.

Example is on the web page:

class RunnableDemo implements Runnable {

Implementing Runnable interface:

Step 1

As a first step you need to implement a run() method provided by **Runnable** interface. This method provides entry point for the thread and you will put you complete business logic inside this method. Following is simple syntax of run() method:

public void run( )

Step 2:

At second step you will instantiate a **Thread** object using the following constructor:

Thread(Runnable threadObj, String threadName);

Step 3:

void start( );

If you need to work with the same data using threads, you have to use SYNCHRONIZED in order for one thread to work not concurrently. Only one thread will have an access, let’s say, to a file at certain point of time.

Interthread Communication:

These methods have been implemented as **final** methods in Object, so they are available in all the classes. All three methods can be called only from within a **synchronized** context.

public void wait()

public void notify()

public void nofifyAll()

Thread Deadlock:

Deadlock describes a situation where two or more threads are blocked forever, waiting for each other. Deadlock occurs when multiple threads need the same locks but obtain them in different order.