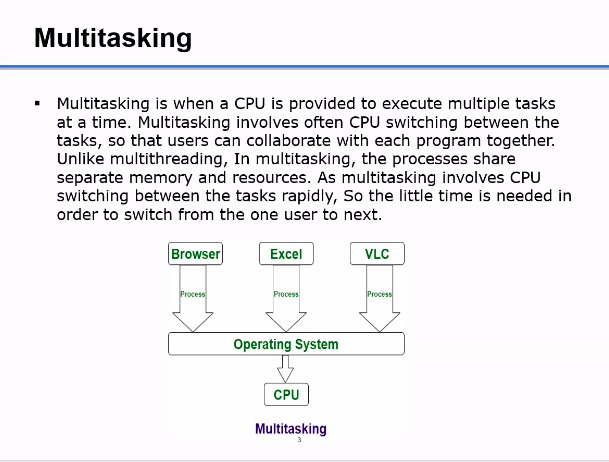
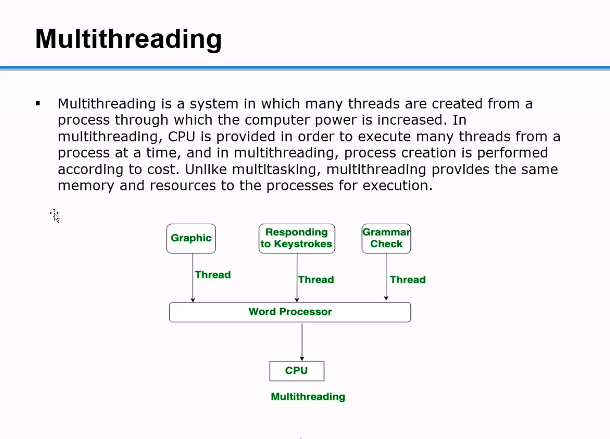
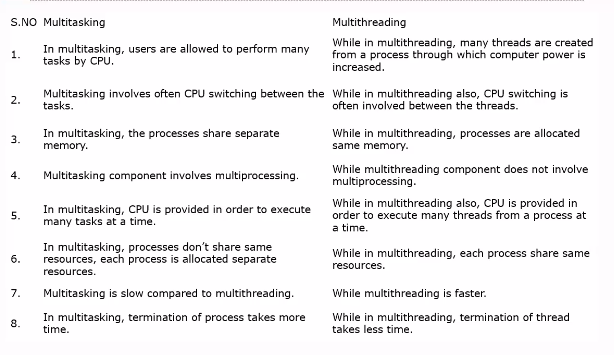
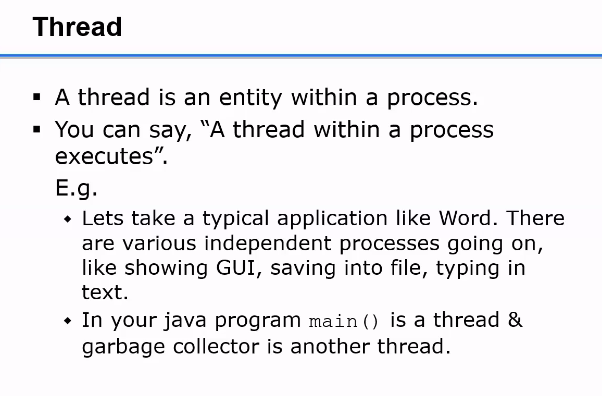
**Multitasking and Multithreading**

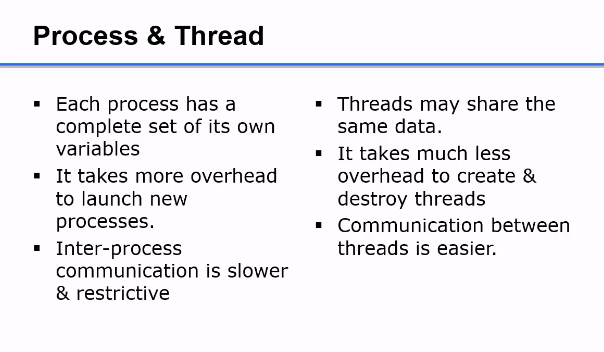


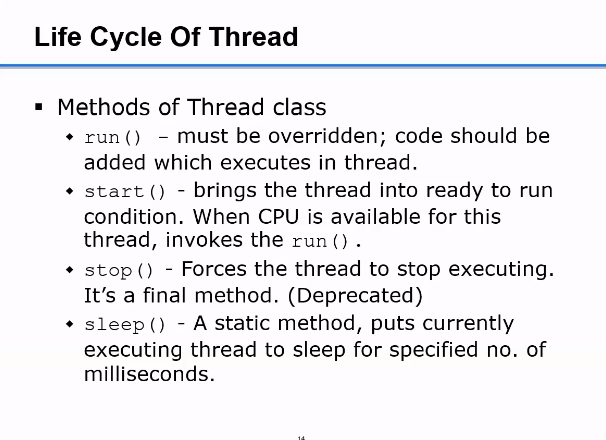


**Difference between multitasking and multithreading**



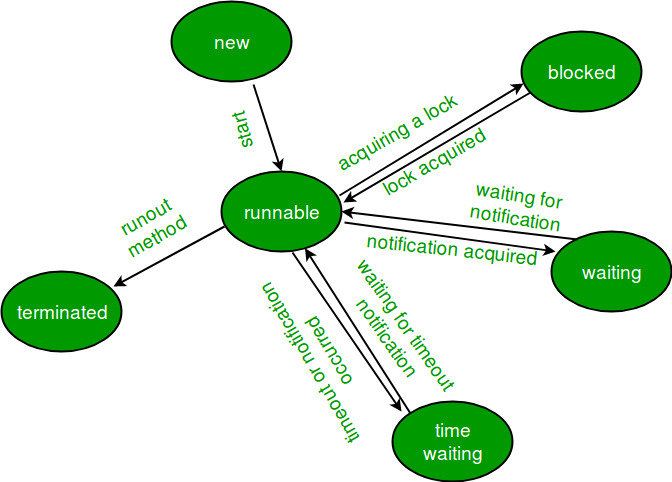






**Thread States**

* A [thread](http://www.geeksforgeeks.org/multithreading-in-java/) in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:
* New
* Runnable
* Blocked
* Waiting
* Timed Waiting
* Terminated



**Methods of Thread class**

Run () - must be overridden; code should be added which executes in thread.

Start () - brings the thread into ready to run condition. When CPU is available for this thread, invokes the run ().

Stop () - Forces the thread to stop executing. It’s a final method. (Deprecated)

Sleep () - A static method, puts currently executing thread to sleep for specified no. of milliseconds.

**Priorities to Threads**

Threads can be given different priorities from 1 to 10. The thread with more priority is given first preference by the thread scheduler to allocate microprocessor time, than the thread with less priority.

There are two methods that support priorities in Thread class.

1. with setPriority(int) method, we can set a priority to a thread.

2. with getPriority() we can retrieve the priority of a thread.

To give priorities, the Thread class defines three constant variables (called symbolic constants) as follows:

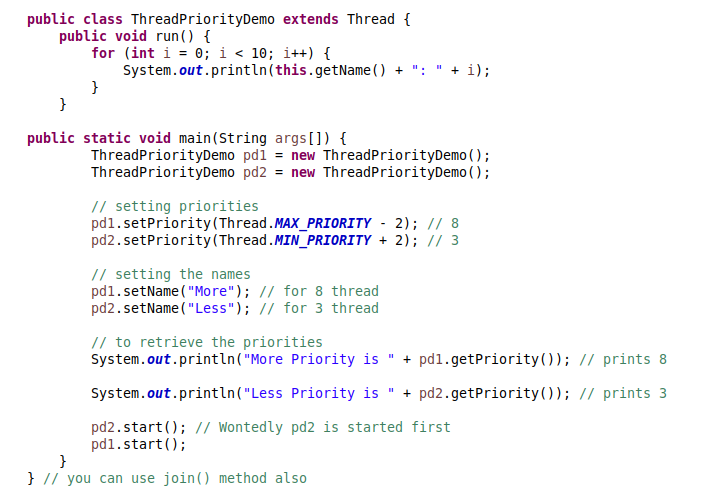
public static final int MIN\_PRIORITY = 1;

public static final int NORM\_PRIORITY = 5;

public static final int MAX\_PRIORITY = 10;

If we do not set a priority for a thread, JVM gives a default priority of 5 (called normal priority).

**Thread priority example code:**



**Output**

