Java AWT Event-Dispatching Thread explanation:

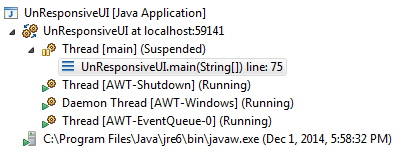
The Event Dispatching thread is a special thread that is managed by the AWT. Basically it is a thread that runs in an infinite loop processing event. The java.awt.EventQueue.invokeLater() method is a special way to provide some code that will run on the event queue. Writing a ui framework that is safe in a multithreading environment is very difficult so the AWT authors decided that they would only allow operations on GUI objects to occur on a single special thread. All event handlers will execute on this thread and all code that modifies the gui should also operate on this thread.

Now the AWT does not usually check that you are not issues gui commands from another thread (The WPF framework for C# does do this). so it is possible to write a lot of code and be pretty much agnostic to this and not run into any problems. But this can lead to undefined behavior so the best thing to do is to always ensure that gui code runs on the event dispatcher thread. invokeLater provides a mechanism to do this.

So a classic example is that you need to run a long running operation like downloading a file. So you launch a thread to perform this action then when it is completed you will use invokeLater to update the UI. If you didn't use invokeLater and instead you just updated the ui directly you might have a race condition and undefined behavior could occur.

javax.swing.SwingUtilities.invokeLater() is a wrapper for java.awt.EventQue- ue.invokeLater()

To run the constructor on the event-dispatching thread, invoke static method SwingUtilities.in-vokeLater() to asynchronously queue the constructor on the event-dispatching thread. The codes will be run after all pending events have been processed.



Basic OS synchronization mechanisms

**Mutex:**

Serialize execution of multiple threads by defining a “critical section” that can be executed by one thread at a time.

The thread that acquires the lock is also the one responsible to release it.

Implementation like Operating System or JVM block the threads using spin lock (busy wait) approach.

Two types of mutex: