Lecture #7

Topics: Java Threads - review
Java Monitors
Synchronization using Java monitors

Java Threads State Diagram

Java monitors

Java monitors are signal-and-continue monitors.
To construct a Java monitor, for all the service methods that should be executed inside the monitor use the "synchronized" keyword.

The synchronized keyword marks portions of a code as belonging to a monitor: a protected portion of code that enforces a one-thread-at-a-time-policy.
Every object in Java has associated with it a single lock. Ordinarily, when an object is being referenced (a method is invoked), the lock is ignored. When a method is declared as synchronized, however, calling the method requires owning the lock for the object.

Every object with synchronized methods is a monitor. In Java you cannot directly inspect or manipulate monitors. Instead you defined the scope of a monitor implicitly by specifying the code blocks and methods it contains through synchronized blocks or methods. **A synchronized method specified only the block of code.**

Java doesn't support named conditions. There is only one condition queue that will contain processes blocked for different reasons. When used inside a monitor, a notification object plays the role of a named variable. **A synchronized statement specifies an object and a block of code.**

By the time Q's turn to execute inside the monitor comes, the logical **condition** for which Q was waiting **may no longer hold**. The condition has to be checked again.

It is necessary to use a **while** loop instead of an **if** statement.

```
        While (condition) try {(wait( );} catch (InterruptedException e)  { }
```

Barging is possible.

A process that executes inside the monitor may signal when a condition becomes *True*. This is done using one of the two methods:

notify( )

notifyAll( )

A process that executes inside the monitor may block when a condition is not satisfied. This is done using the ***wait( )*** method.

Comments on Question 10 chapter 5.