

notify / notifyAll misunderstanding

Source: <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2004-07/1520.html>

From: VisionSet (*spam_at_ntlworld.com*)

Date: 07/13/04

Date: Tue, 13 Jul 2004 14:56:24 GMT

As I understand it, notify wakes up a waiting thread that then attempts to gain lock the object it is waiting for. Where as notifyAll wakes up all waiting threads and ONLY ONE gains the lock, the others go back to waiting. The following code does not seem to show this.

I have a record locking system that I have condensed to the simple example below. Record locks are placed on Integers that are managed by a Set. When lock is called then the record is put in the set if not already there. If the record is already there then wait() is called. When unlock() is called a record is removed from the set and notifyAll() is called.

In the example I have 2 threads that lock on record 5, and 8 that lock on record 6.

So after the 10 lock threads run, I will have 1 thread locked 5, 1 thread locked 6, 1 thread is waiting to lock 5, and 7 threads are waiting to lock 6.

Then a single thread unlocks record 5 and notifyAll is called.

So 8 waiting threads are then all notified and 1 arbitrarily gets the lock. If it is the 1 waiting to lock 5 then it will succeed since 5 has just been locked. But more likely it will be one of the 7 waiting to lock 6, which will be unsuccessful because 6 hasn't been unlocked.

SO! Why does it always succeed?

With notifyAll() I always get:

```
locked 5
locked 6
unlocked 5
locked 5
```

notify() on the other hand works as expected ie:

```
locked 5
locked 6
unlocked 5
```

but probably for the wrong reason since my understanding is obviously flawed.

```
import java.io.*;
import java.util.*;

public class Test {

    private Set lockSet = new HashSet();

    public void lock(int recNo) {
        Integer key = new Integer(recNo);
        synchronized(lockSet) {
            while(lockSet.contains(key)) {
                try{
                    lockSet.wait();
                }
                catch(InterruptedException ex) {}
            }
            lockSet.add(key);
            System.out.println("locked "+recNo);
        }
    }

    public void unlock(int recNo) {

        try {Thread.sleep(500);}
        catch(InterruptedException ex) {}

        Integer key = new Integer(recNo);
        synchronized(lockSet) {
            lockSet.remove(key);
            System.out.println("unlocked "+recNo);
            lockSet.notifyAll();
        }
    }

    public static void main(String[] args) throws Exception {

        Test test = new Test();
        test.testLocking();

    }

    private void testLocking() {

        lockIt(5);

        lockIt(6);
        lockIt(6);
        lockIt(6);

    }
}
```

```
lockIt(6);  
lockIt(6);  
lockIt(6);  
lockIt(6);  
lockIt(6);
```

```
lockIt(5);
```

```
unlockIt(5);
```

```
}
```

```
// these just fire up threads for lock & unlock
```

```
private void lockIt(final int x) {  
    new Thread(new Runnable() {  
        public void run() {  
            try {  
                lock(x);  
            }  
            catch(Exception e) {e.printStackTrace();}  
        }  
    }).start();  
}
```

```
private void unlockIt(final int x) {  
    new Thread(new Runnable() {  
        public void run() {  
            try {  
                unlock(x);  
            }  
            catch(Exception e) {e.printStackTrace();}  
        }  
    }).start();  
}
```

```
--
```

```
Mike W
```