CMSC 433 – Programming Language Technologies and Paradigms
Fall 2011

Locks & Conditions

Intrinsic Lock Example

```
public class RWDictionaryIntrinsicLock {
  Map<Integer, Integer> m = new TreeMap<Integer, Integer>();
  public synchronized Integer get(Integer key) {
    return m.get(key);
  }
  public synchronized Integer put(Integer key, Integer value) {
    return m.put(key, value);
  }
     …
}
```

Instrinsic Lock Example

- See RWDictionaryIntrinsicLock.java

Lock Interface

- High-level locking interface provides same memory visibility semantics as intrinsic locks,
  - Different locking semantics, scheduling algorithms, ordering guarantees, and performance characteristics

```
interface Lock {
    void lock();
    void lockInterruptibly() throws …;
    boolean tryLock();
    boolean tryLock(long time,TimeUnit unit) throws …;
    void unlock();
    Condition newCondition() throws …;
}
```

ReentrantLock

- Implements Lock
  - Can interrupt a thread waiting to acquire a lock
  - Can specify a timeout while waiting for a lock
  - Can poll for lock availability
  - Multiple wait-sets per lock via the Condition interface
- Outperforms built-in monitor locks in most cases, but slightly less convenient to use (requires finally block to release lock)

Lock Usage Idiom

- ReentrantLock not automatically released
  - Must release lock in finally block

```
Lock lock = new ReentrantLock();
…
lock.lock();
 try {
     // perform operations protected by lock
 } catch (Exception ex) {
     // restore invariants
 } finally {
      lock.unlock();
 }
```

Using Lock.tryLock()

```
public boolean fooWithTwoLocks() {
    while (true) {
      if (lock1.tryLock()) {
         try {
           if (lock2.tryLock()) {
             try {
               doWork();
             } finally { lock2.unlock(); }
           }
         } finally { lock1.unlock(); }
       }
       if (timeOut)  return false;
       NANOSECONDS.sleep(/* some amount */);
    }
}
```

Using Lock.tryLock()

```
boolean fooWithTimeBudget(String message) throws …{
    long nanosToLock = /* estimated time budget */ ;
    if (!lock.tryLock(nanosToLock, NANOSECONDS))
        return false;
    try {
        return doWork();
    } finally { lock.unlock(); }
}
```

Fairness

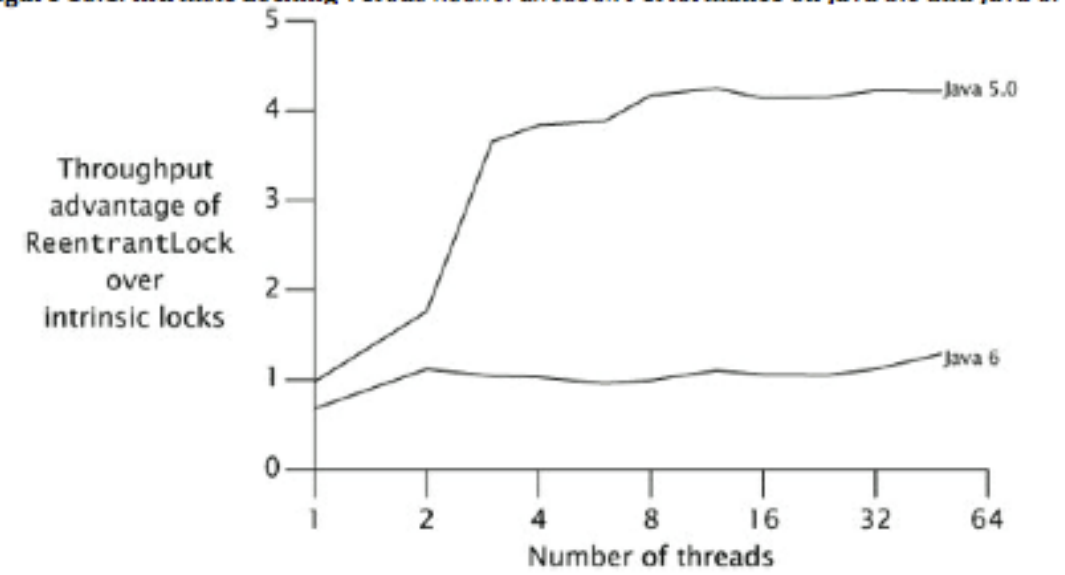
- ReentrantLock(boolean fair)
  - Creates ReentrantLock with/without fairness policy
- If true, under contention, the longest-waiting thread preferentially given access
- Otherwise no guarantees on access orderings

ReentrantLock Fairness Example

- See RWDictionaryReentrantLock .java

Intrinsic vs ReentrantLock

Figure 13.1. Intrinsic Locking Versus `ReentrantLock` Performance on Java 5.0 and Java 6.



Intrinsic vs ReentrantLock

- Performance differences are minimal
- May want to use ReentrantLocks only when you need the extra functionality they provide

ReadWriteLock

- ReadWriteLock defines a pair of locks
  - One for readers; one for writers

```
interface ReadWriteLock {
    Lock readLock();
    Lock writeLock();
}
```

ReentrantReadWriteLock

- ReentrantReadWriteLock class
  - Provides reentrant read and write locks
  - Allows writer to acquire read lock
  - Allows writer to downgrade to read lock
  - Supports “fair” and “writer preference” acquisition

ReentrantReadWriteLock Example

```
class RWDictionaryRWL {
    private final Map<String, Data> m = new TreeMap<String, Data>();
    private final ReentrantReadWriteLock rwl = new ReentrantReadWriteLock
    ();
    private final Lock r = rwl.readLock();
    private final Lock w = rwl.writeLock();
    public Data get(String key) {
          r.lock();
          try { return m.get(key); }
          finally { r.unlock(); }
      }
      public Data put(String key, Data value)
          w.lock();
          try { return m.put(key, value); }
          finally { w.unlock(); }
      }
    }
```

Read/Write Lock Example

- See RWDictionaryRWL.java

Condition Queues

- Each Java Object has 1 condition queue (wait set)
  - wait() and notify() manipulate condition queue
- Calling object.wait()
  - Adds current thread to object's condition queue
  - Releases lock on object
  - Puts current thread to sleep
- Calling object.notify() or object.notifyAll()
  - Wakes up one or more threads in condition queue
  - Allows threads to compete for lock on object

Intrinsic Condition Queue Example

```
class BoundedBufferPrim {
   final Object[] items = new Object[100];
   int putptr, takeptr, count;
```

Intrinsic Condition Queue Example

```
public synchronized void put(Object x) throws … {
  while (count == items.length)  wait();
  items[putptr] = x;
  if (++putptr == items.length)  putptr = 0;
  ++count;
   notifyAll();
}
```

Intrinsic Condition Queue Example

```
public synchronized Object take() throws …{
   while (count == 0) wait();
   Object x = items[takeptr];
   if (++takeptr == items.length) takeptr = 0;
   --count;
   notifyAll();
   return x;
}
```

Intrinsic Condition Queues

- Some limitations
  - Objects may be interested in multiple conditions, but have only one intrinsic condition queue
  - Notify() methods can't indicate reasons for wakeup

Condition Interface

- Improves on intrinsic condition queues
  - Multiple conditions per lock
  - Absolute and relative time-outs
  - Timed waits tell you why you returned
  - Convenient uninterruptible wait

Condition Interface

```
interface Condition {

    void await() throws IE;
    boolean await( long time, TimeUnit unit ) throws IE;
    long awaitNanos( long nanosTimeout) throws IE;
    void awaitUninterruptibly()
    boolean awaitUntil( Date deadline) throws IE;
    void signal();
    void signalAll();
}
```

Condition Example

```
class BoundedBufferCond {

    Object[] items = new Object[100];
    int putptr, takeptr, count;

    Lock lock = new ReentrantLock ();
    Condition notFull = lock.newCondition();
    Condition notEmpty = lock.newCondition();
```

Condition Example (cont.)

```
public void put( Object x) throws …{
    lock.lock();
    try {
      while (count == items.length) notFull.await();
      items[putptr] = x;
      if (++putptr == items.length) putptr = 0;
      ++count;
      notEmpty.signal();
    }  finally { lock.unlock(); }
}
```

Condition Example (cont.)

```
public Object take() throws …{
   lock.lock();
   try {
     while (count == 0) notEmpty.await();
     Object x = items[takeptr];
     if  (++takeptr == items.length)  takeptr = 0;
     --count;
     notFull.signal();
     return x;
   } finally { lock.unlock(); }
  }
}
```

Condition Example (cont.)

- See
  - BoundedBufferPrim.java
  - BoundedBufferCond.java

Ungraded Assignment

- Take a look at
  - RWDictionaryIntrinsicLock.java
  - RWDictionaryRWL.java
- The intrinsic locking approach performs better than the ReeentrantReadWriteLock approach.
- How might you modify the workload characteristics of these programs so that the ReeentrantReadWriteLock approach compares more favorably to the intrinsic locking approach