Administrivia

- Exam Thursday
  - on C++ (all) and Java (through threads)
  - practice exam posted
- Java threads project questions?
- C++ project commentary due Wednesday
  - grades for project code (90%) have been emailed

Last time - Java

- Daemon threads
- **Synchronized** methods and statement blocks
- Object locks
  - static synchronized method locks Class object
- **wait()** and **notify()**
  - only inside synchronized method or block

notify() vs. notifyAll()

- Very tricky to use notify() correctly
  - notifyAll() much safer
- Need:
  - all waiters are equal
  - each notify only needs to wake up 1 thread
  - handle `InterruptedException` correctly

**InterruptedException Example**

```java
synchronized (this) {
    while (!ready) {
        try { wait(); }
        catch (InterruptedException e) {
            notify();
            throw e;
        }
    // do whatever
}
```

**Handling InterruptedException**

```java
while (!ready) {
    try { wait(); }
    catch (InterruptedException e) {
        notify();
        throw e;
    }
    // do whatever
}
```
Deadlock

• Quite possible to create code that deadlocks
  – Thread 1 holds lock on A
  – Thread 2 holds lock on B
  – Thread 1 is trying to acquire a lock on B
  – Thread 2 is trying to acquire a lock on A
  – Deadlock!
• Not easy to detect when deadlock has occurred
  – other than by the fact that nothing is happening

A common multi-threading bug

• Threads might cache values
• Obtaining a lock forces the thread to get fresh values
• Releasing a lock forces the thread to flush out all pending writes
• volatile variables are never cached
• sleep(...) doesn’t force fresh values
• Many compilers don’t perform these optimizations
  – but some do (Hotspot?)
• Problem might also occur with multiple CPUs

Guidelines to simple/safe multi-threaded programming

• Synchronize access to shared data
• Don’t hold a lock on more than one object at a time
  – could cause deadlock
• Hold a lock for as little time as possible
  – reduces blocking waiting for locks
• While holding a lock, don’t call a method you don’t understand
  – e.g., a method provided by someone else, especially if you can’t be sure what it locks

Guidelines (cont.)

• Have to go beyond these guidelines for more complex situations
  – but need to understand threading and synchronization well
• Recommended book for more details:
  – Concurrent Programming in Java, by Doug Lea