Multithreading/java.util.concurrent

4-11-2006

Opening Discussion

- What did we talk about last class?
- Do you have any questions about the assignment?
- What do you remember about threads? I argue that multithreading is one of the most important things you can learn during your time at Trinity. Why?

Remember Threads?

- Previously we talked about multithreading and how we could do threads in Java. We did this using that basic Thread class and capabilities that were built into Java from the first version of the language.
- These capabilities give you everything you need to do multithreading, but they don't make it as easy as it could be in all cases.
- With Java 5.0 a new package was added to the libraries called java.util.concurrent. This package makes many of the common tasks that were challenging much easier.

Looking at the Library

- You should each bring up a window and look at the java.util.concurrent package. There are two packages below this that also contain useful classes.
- There are a few primary interfaces/classes in this package.
  - Executor, ExecutorService, ScheduledExecutorService, and Executors
  - Locks and Conditions
  - Synchronizer Utilities
  - BlockingQueues

Executors

- An Executor is something that can execute a Runnable object. The idea is that different implementations can choose to execute the Runnable object in different ways and potentially in different threads.
- The ExecutorService interface extends Executor and adds support for Callable objects which have the ability to return information, unlike Runnable.
- ScheduledExecutorService allows time information for when something is executed.
- The Executors class is a utility class that can be used to create different helpful objects including ExecutorServices.

Locks and Conditions

- Sometimes use of standard synchronized blocks isn't ideal because it is block scoped. Once the code leaves a block the lock on the object is relseased.
- For this purpose the java.util.concurrent.locks package has a Lock class that can be used to synchronize a resources across multiple blocks.
- There is also a Condition interface that can be used to give you the behavior of wait and notifyAll when you are using locks. The Lock object can give you Conditions and you can have more than one.

Synchronizer Utilities

- There are four other classes in java.util.concurrent that are used for synchronization purposes.
  - Semaphore – has a count of “permits” that threads can get. Blocks if you try to get one when all are used.
  - CountDownLatch – starts closed and blocks all threads that call await. It opens and starts those threads after a specified number of calls to countDown.
  - Exchanger – allows two threads to exchange a value by calling the method by that name. The first thread blocks until the second one is ready to exchange.
  - CyclicBarrier – similar to CountDownLatch but blocks a certain number of threads until that number have called await. It can be used multiple times.

Blocking Queues

- This is an interface for a Queue but the methods put and take will block if they can't complete and wait until they can.
- So if you try to put into a blocking queue with a fixed size and there are no more slots, it will block. If you try to take from one that is empty that will also block.
- There are 5 provided implementations of this interface. Let's go look at each of those.

Atomic Variables

- An atomic operation is one that can't be subdivided. For threading this means that other threads can interrupt it in the middle and see a bad state.
- The java.util.concurrent.atomic package contains classes that give you the ability to do atomic operations without having to do your own locking.
- These operations include not only get and set methods, but also methods that do basic operations.

Code

- Let's go back to our in class example and see what we can do. Unfortunately, drawing is something that doesn't parallelize very nicely.

Minute Essay

- We are considering putting parallelism into the project by having it so that different entities can update at the same time. How will this change the way you have to code?
- Remember that assignment #6 is due today.