



# Multithreading in Java

4/15/2007





# Opening Discussion

- Do you have any questions about the quiz?
- Let's look at solutions to the interclass problem.
- What could go wrong with our tree to mess up performance? It could become unbalanced and it degrades to a linked list.
- Do you have any questions about the assignment?





- I want us to code a BST together. We'll have one of you code and I'll walk through the process. These are really neat data structures so you really should get a feel for how they work.





- While support of threading was built into the original version of Java, some common threading tasks still aren't easy.
- There are some things that you find yourself doing rather frequently with threaded code and there are some limitations with the basic libraries that make it difficult to use at times.
- The `java.util.concurrent` library was added to make common parallel tasks easier. It is built around a few key concepts. Let's go look at the javadocs for this library.



# Executor Interface

- The heart of the concurrent library is the Executor interface. It provides a method for running a Runnable object, but tells you nothing about how it will be run.
- The commonly used subtype of Executor is ExecutorService that supports the Callable<T> interface and Future<T> objects.
- The Executors utility class gives you a way to create some commonly used ExecutorServices.





# Blocking Queues

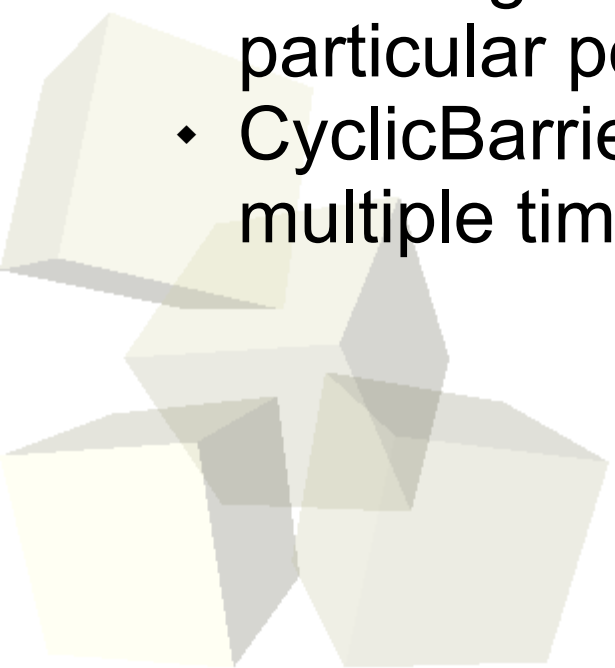
- One handy data structure in parallel applications is a blocking queue. This is a queue with a fixed number of slots in it.
- If you try to dequeue when the queue is empty, the thread will block until another thread adds something.
- If you try to enqueue when the queue is full the thread will block until another thread removes something.





# Coordinating Threads

- The concurrent library also provides a number of different classes to help with coordinating the activities of threads.
  - ♦ Semaphores – holds a number of “permits” that can be given out to threads.
  - ♦ CountdownLatch – stops threads until they have all hit the latch. Only works once.
  - ♦ Exchanger – two threads swap information at a particular point.
  - ♦ CyclicBarrier – like the CountdownLatch but works multiple times.





# Atomic and Locks

- There are two subpackages for `java.util.concurrent`: `atomic` and `locks`.
- The `java.util.concurrent.atomic` package provides data types that have atomic access methods. These methods can't be interrupted by other threads so they can be done in a thread safe way.
- In `java.util.concurrent.locks` you will find classes for locks that can be used in your program. Locking is basically what the `synchronized` keyword does, but that can't be shared across objects or methods.





# Multithreading Mergesort

- We talked about how we could multithread our mergesort. Let's go ahead and do it. Instead of using the standard thread library I want us to use the facilities in `java.util.concurrent`.





- What are some of the potential benefits of the streaming model Java uses for doing I/O? How could inheritance and polymorphism be put to use here?
- There are only six remaining class days.
- Interclass Problem – Write a remove method for our BST or integrate the BST into the drawing program.

