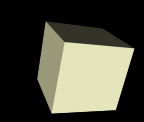
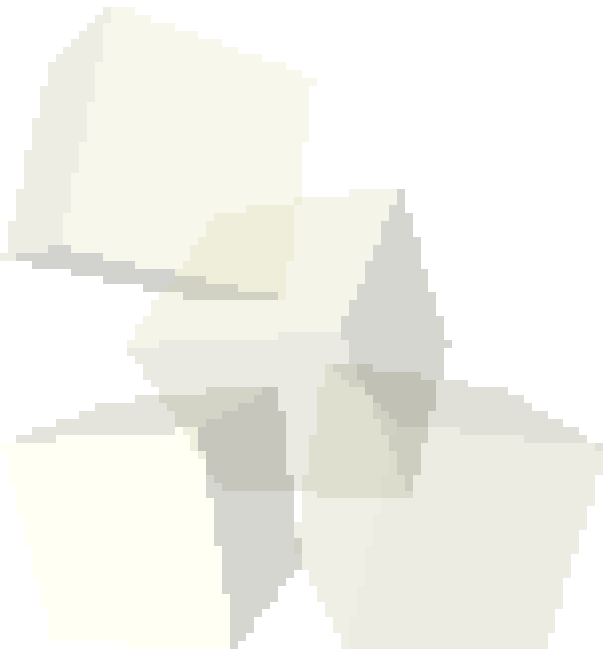


Stacks and Queues

2-25-2009



Opening Discussion

- Let's look at solutions to the interclass problem.
- Do you have any questions about the reading?
- Do you have any questions about the assignment?
- Minute Essays
  - How do you change the size of a button?
  - Why is it bad to call stop() on a thread?
  - How and why do you overclock a processor?

Synchronization

- The way to prevent two threads from accessing the same piece of memory at the same time is to synchronize the critical pieces of the code. You can put the synchronized keyword in front of methods or make synchronized blocks.
- Each object and class in Java can have a monitor that is locked when synchronized code is being executed. Only one thread can hold the lock on the monitor at a given time. This insures that you never have two threads executing critical code on a single object at the same time.
- Too much synchronization slows things down or causes deadlock.

Wait and Notify

- We can get even more control over how threads behave with the wait and notify methods.
- The wait method will stop the execution of a thread until some other thread tells it to continue execution. The notify and notifyAll methods are how threads tell other threads that they are supposed to wake up.
- All of these must be called by a thread that holds the monitor to the object they are being invoked on. Typically that means that are called from inside synchronized code.
- Wait should be called inside a while loop. Use notifyAll.

Thread Coding

- Sorting code can provide a great test for threading. In this case we want to use our slow sorts so that we can actually time how long it takes them to run.
- Let us create a command with some code that will create N (where N is an int variable) arrays of Doubles of length ARRAY_SIZE (you can declare that as a static final variable) then fill them with random values.
- We can sort them one at a time, then refill them and sort them in N threads. Use System.nanoTime to time how long each of those takes.

Abstract Data Types (ADTs)

- Today we will be working with the simplest forms of abstract data types. These are things that hold data and specify how you can interact with it and what happens when data is added or removed.
- In Java an ADT is basically an interface for a container with comments giving details on what happens with each method.
- Note that it doesn't specify how things happen. That is why it would be an interface. ADTs can be implemented in many different ways.

Stacks and Queues

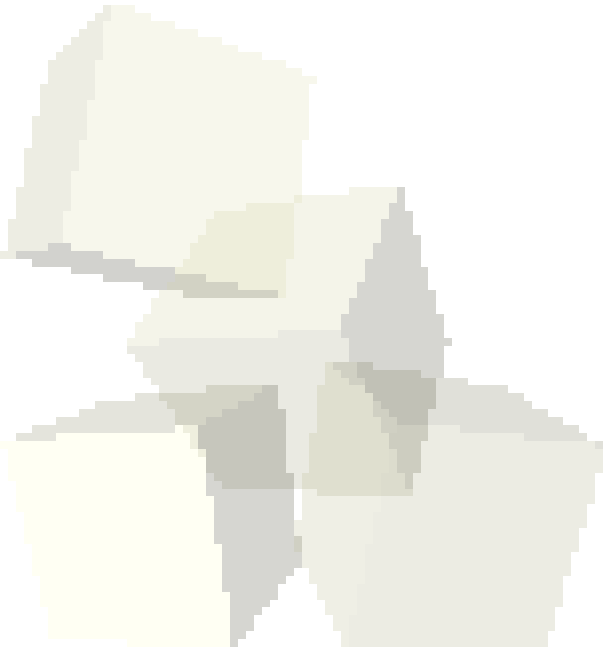
- The simplest forms of ADTs, they each require one method to add an element and one method to remove an element. For easy of use we typically also include two other methods.
- Methods of a stack
  - push
  - pop
  - peek
  - isEmpty
- Methods of a queue
  - enqueue
  - dequeue
  - peek
  - isEmpty

The Difference?

- Push and enqueue add items while pop and dequeue remove items. The difference is what item gets removed.
- A stack is last in, first out (LIFO). Just think of how you interact with a stack.
- A queue is first in, first out (FIFO). If you were British you would use the term queue instead of line for what you stand in when waiting for something.

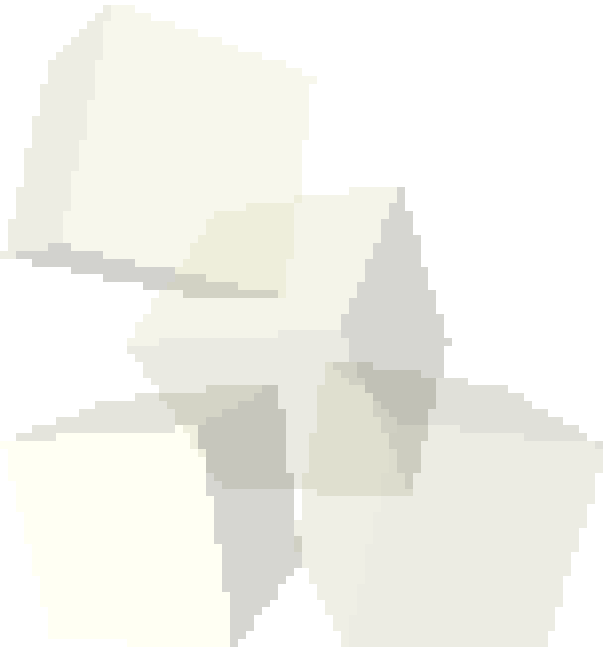
Array Based Stack

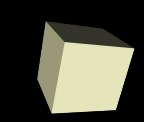
- Let's write an interface called MyStack with the methods we said should be in it. Make the interface generic so it can handle any type.
- Now let's write a class called ArrayStack and make it implement MyStack. Fill in the code for ArrayStack and add a main method to test that they work.



Array Based Queue

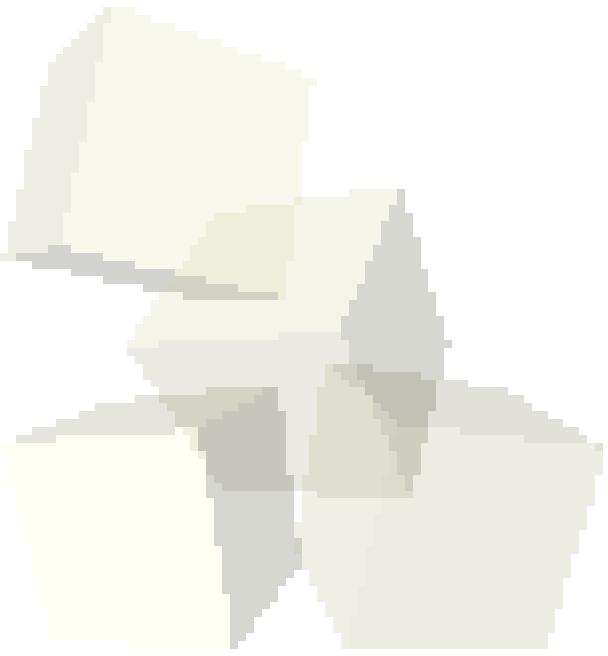
- Now we will do the same thing for a queue. Make a MyQueue interface and an ArrayQueue class.





Other Code

- One of the most standard applications of a stack is a reverse polish calculator (RPC).
- Let's make a class for a RPC then make a command that will use it.



Minute Essay

- We will re-implement the MyStack interface latter on using a linked list for the implementation. Can you describe how we might do that?
- Remember that design #3 is due today.
- Interclass problem – Edit the calculator GUI that you have made so that it implements an RPC calculator. You might want to use a JTextArea or a JList to display the stack.