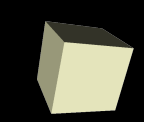
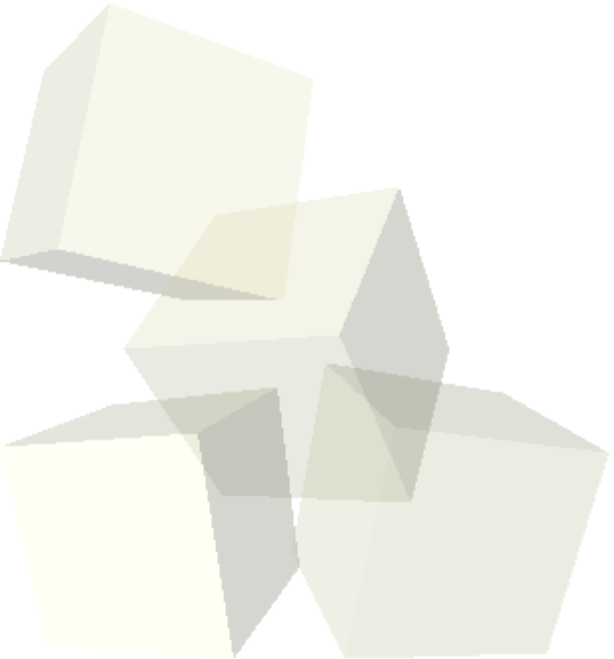
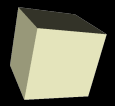


Java Basics

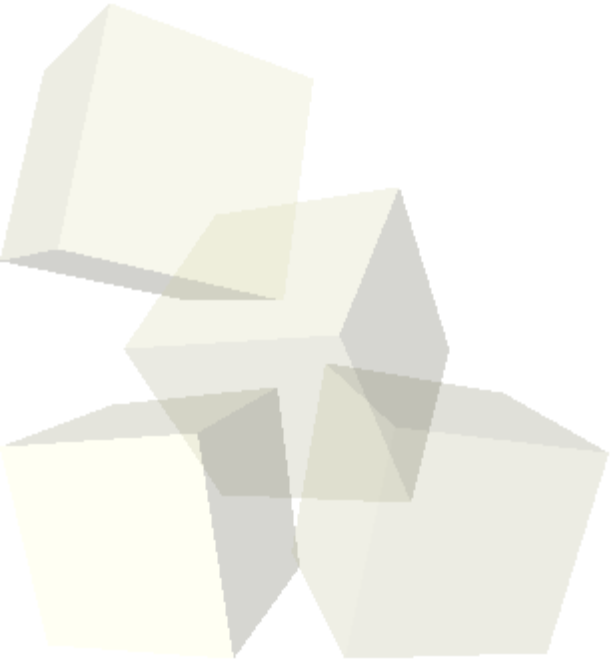
1/24/2008

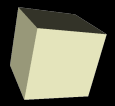




Opening Discussion

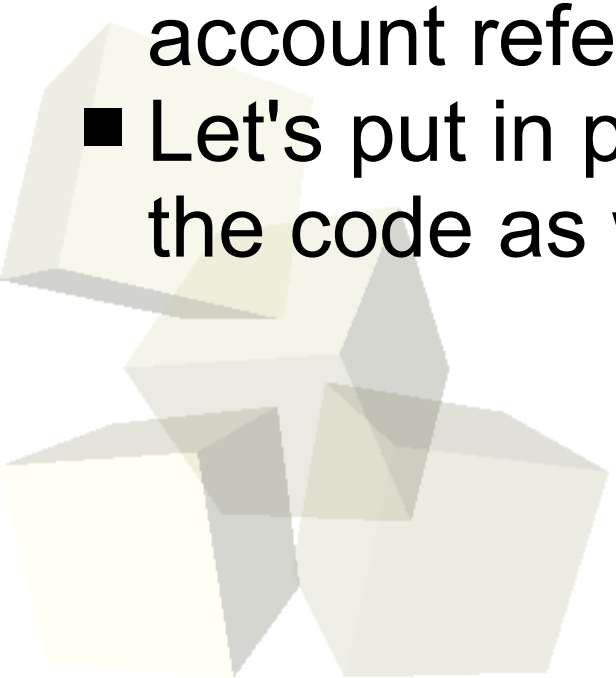
- Let's look at solutions to the interclass problem.
- What did we talk about last class?
- Do you have any questions about the reading?





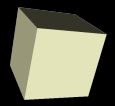
Code

- We want to continue our bank example that we worked on last time in two ways.
- First I want to have our bank make an account and try doing some things with it.
- Second we want to add customer information. Instead of adding that straight to the account, we should create a Customer class and have the account reference it.
- Let's put in proper documentation comments on the code as well.



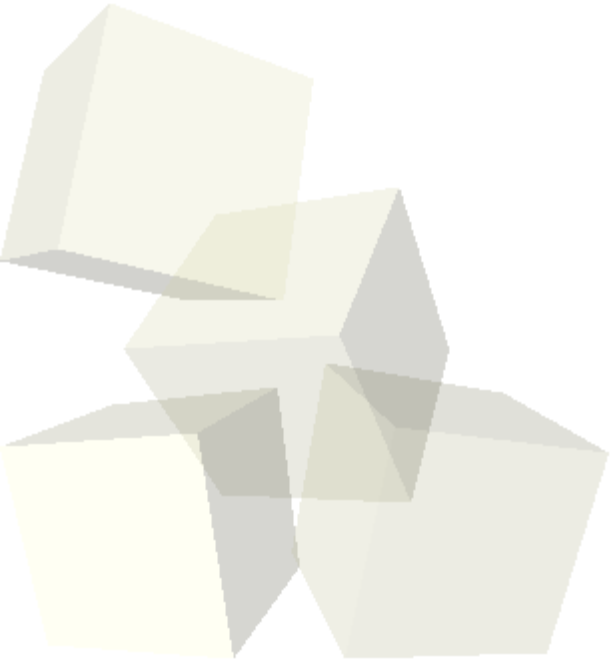
Java References vs. Pointers

- In Java when you declare an object you are really declaring a reference to an object. This is like a pointer but you can't do pointer arithmetic. To get a real object you use the new operator. New is like malloc and returns a heap object.
- All objects are gotten with new so all objects exist on the heap.
- When you call new it invokes a constructor.
- null is a universal symbol for references that don't point to anything.



More on Objects

- There is no operator overloading in Java. You write and call normal methods instead.
- Doing = or == with object references assigns or compares references. Think of them as pointers without the *.
- Use a copy constructor to copy and equals() for value comparison.
- No -> or * (deref) operator because things are implicitly dereferenced.

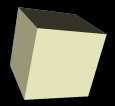


Garbage Collection

- Java has automatic garbage collection. There is no delete operator (no free).
- When you are done using an object and you have no more reachable references to it, the system can determine this and free up the memory for it.
- As a result, you can allocate objects much more freely because you don’t have to worry about freeing them.

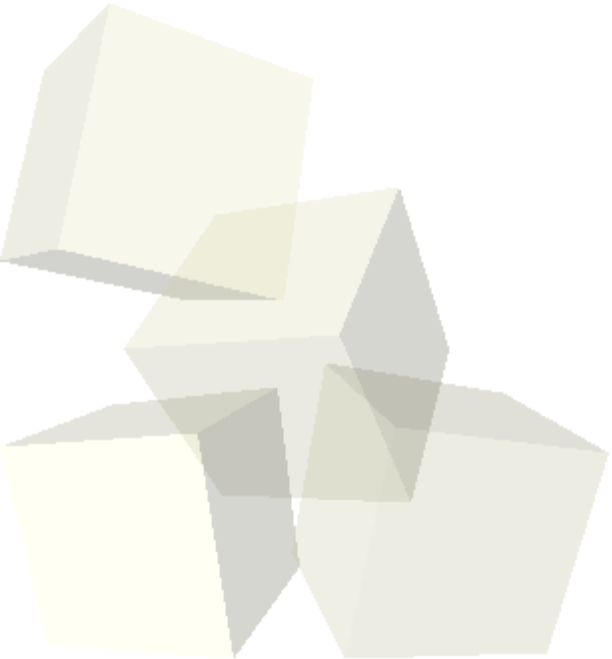
Arrays

- Arrays in Java are actually objects. An array type is denoted by placing [] after the normal type. When you do a new you are allocating the array object. If the items in it are objects you still have to allocate the individual objects.
- This is really nice for inclusion polymorphism.
- You use them like you would in C/C++, though they do have a length member that you can access to find out how many elements are in an array.



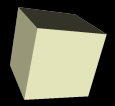
Strings

- Like arrays, strings are a class in Java, called String. Literal strings in Java are objects of that class too.
- This class is in the java.lang package. Note that java.lang is the one package that is implicitly imported so you don't have to use the import statement to refer to the classes in it directly.



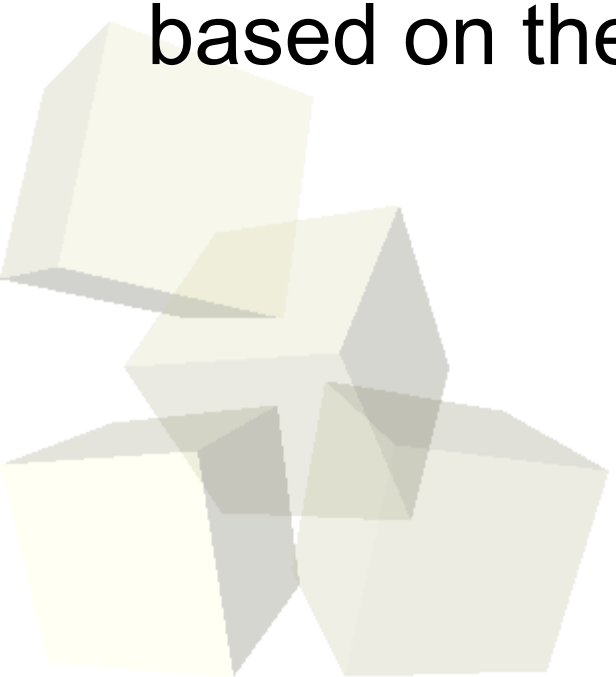
Exceptions

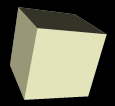
- Java tries to prevent sloppy error handling by providing exceptions instead of requiring programmers to check return codes.
- try blocks surround code that might throw exceptions.
- catch blocks follow a try block and specify what type it is waiting for.
- finally catches everything and always happens.
- If an exception type can be thrown but isn't caught, it must be in the throws clause of that method. Not true for RuntimeExceptions.



Immutable Objects

- In your reading you have inevitably come across the term immutable. What does this mean?
- What are the advantages and pitfalls of immutability?
- How can you write code that takes advantage of immutability?
- The entire paradigm of functional languages is based on the idea that data is immutable.





Minute Essay

- Do you have any significant questions about Java at this point? What do you see as the most significant differences between Java and C right now?
- Interclass Problem – Write a class that represents rational numbers and do some code to test it out.