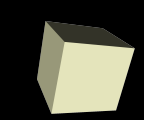
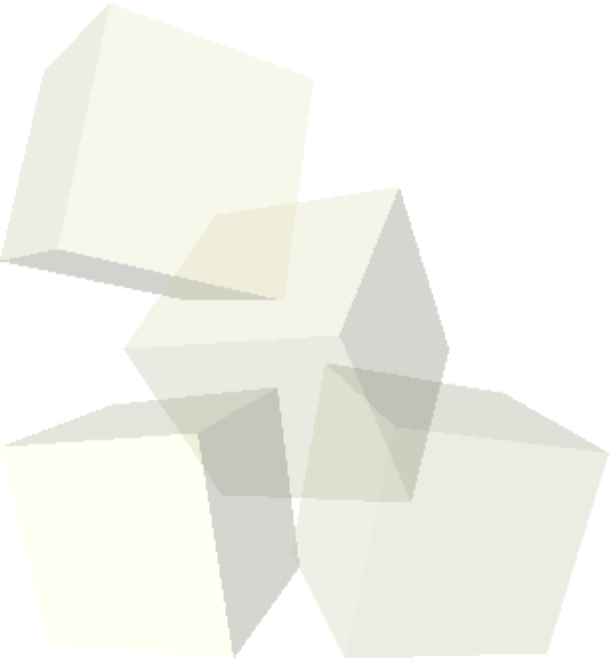


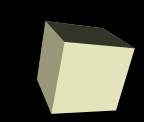
Java Basics

1-19-2006



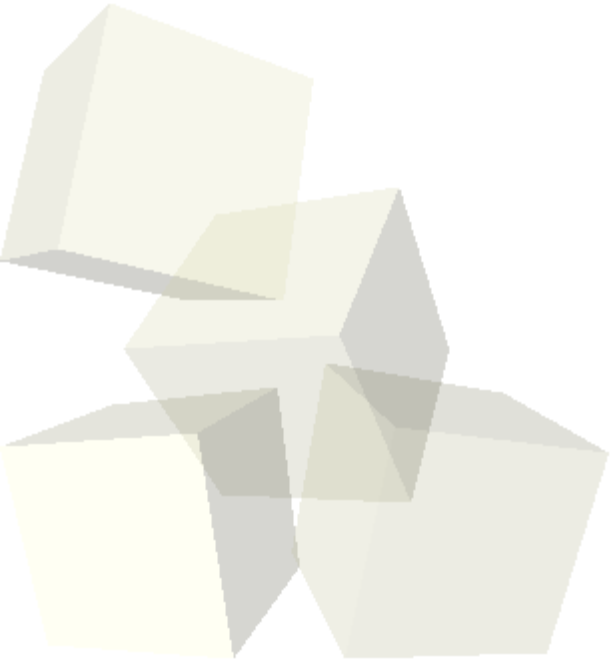
Opening Discussion

- Do you have any questions about the reading?
- Do you have any questions about the project?



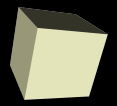
Code

- We want to continue our bank example that we worked on last time in two ways.
- First 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.



No Preprocessor Directives

- You import so you don’t have to type in full package names. This looks similar to #include in use, but it is quite different.
- No #define in Java. For constants use static final variables. For macros just use functions.
- There is also no conditional compilation in Java so #ifdef, #ifndef, etc. don’t exit. Assert was added in 1.4 but we won't be using it.

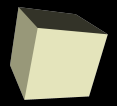


Meaning of static

- A static member is associated with a class, not the individual objects.
- In our blueprint analogy, a static member is something written on the blueprint or associated with the factory, not something that is carried with every object made from the blueprint.
- A simple example that is often used when you are trying to analyze the performance of programs is to count how many objects of a given type are created. You can do this with a static member and code in the constructors.

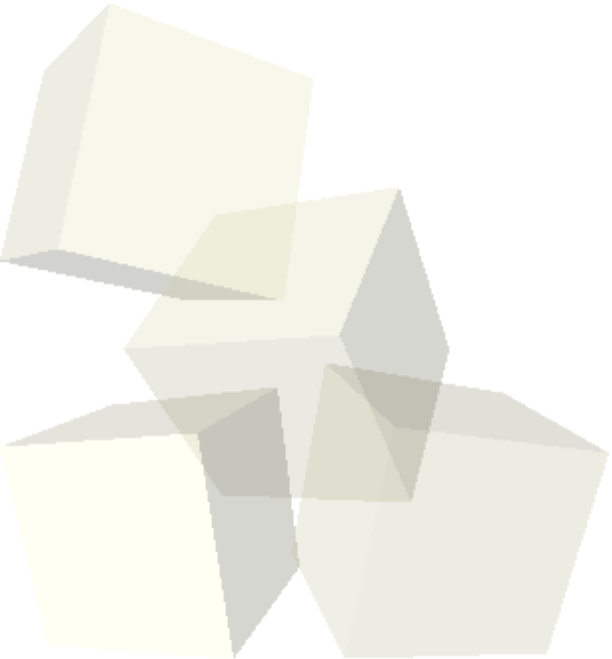
Java References vs. Pointers

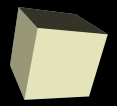
- In Java when you declare an object you are really declaring a reference to an object. This is like a pointer but you can't do pointer arithmetic. To get a real object you use the new operator. New is like malloc and returns a heap object.
- All objects are gotten with new so all objects exist on the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



More Code

- We need some constructors in these classes so that we can create them in a valid state. And then put a main in the code so we actually have a runnable program that we can test.



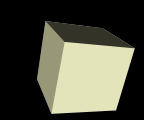


Primitive Types in Java

- Java is not purely object-oriented because it does have primitive types. These types are boolean, char, byte, short, int, long, float, and double.
- Note that booleans and chars are NOT ints in Java (though you can cast chars to ints). This is significant because the statement `if(v=3)` does not compile. This helps cut down on bugs but might seem restrictive in some cases.

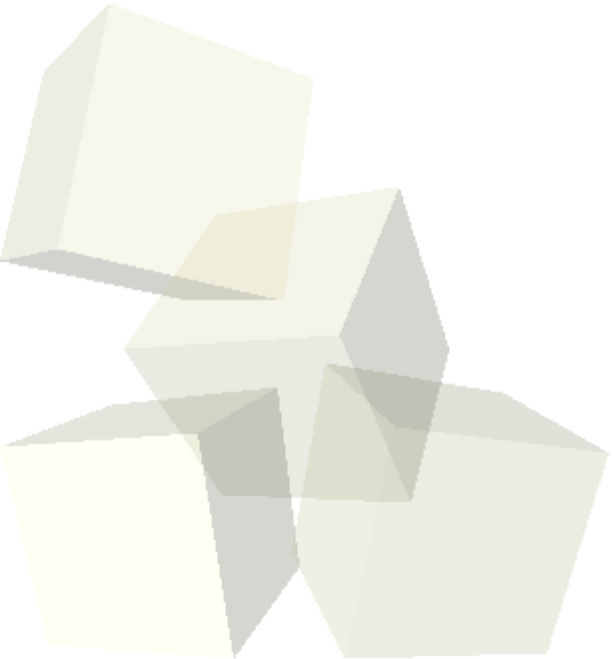
Primitives as Classes

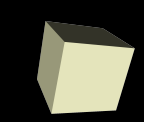
- When you need to represent a primitive type as a class there are some classes in java.lang that can help.
- They are classes like Integer and Double that are basically wrapper classes.
- They do have some nice functionality in static methods as well like Integer.parseInt(String s).
- These classes are immutable.



Code

- Let's write some code together now.





Minute Essay

- Are there any things we have talked about that aren't clear to you? Are you starting to see how objects get used in object-oriented programming?