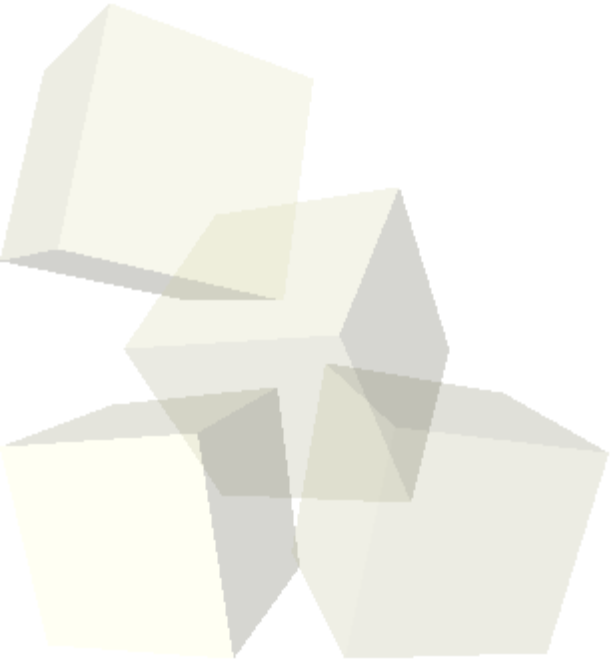
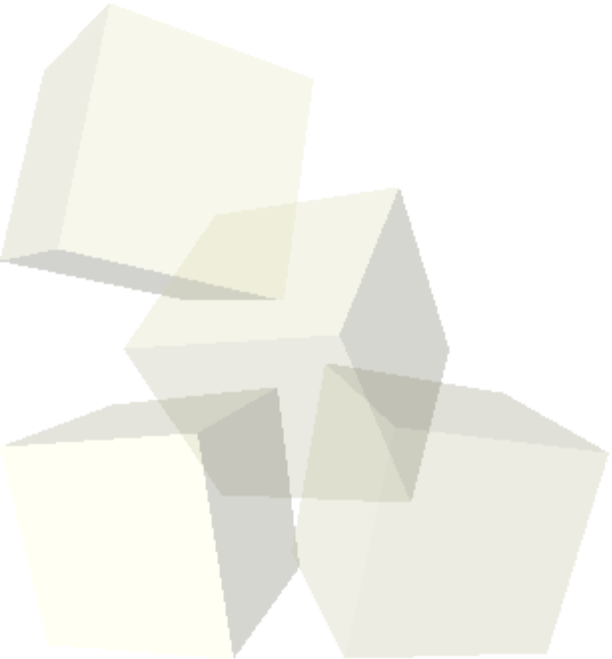
Inheritance

1/27/2009



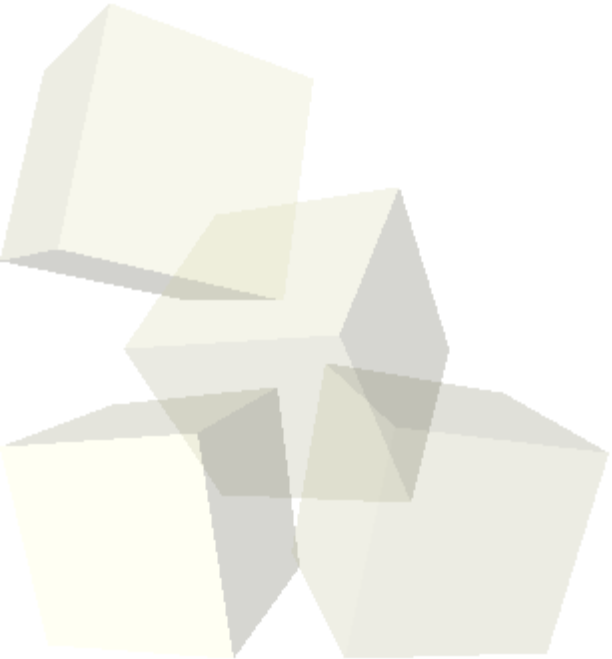
Opening Discussion

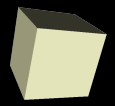
- Let's look at solutions to the interclass problem.
- What did we talk about last class?
- Do you have any questions about the reading?
- Do you have any questions about the assignment? Let's go over the requirements for this first design/assignment and what I'm expecting from you.



Minute Essay Responses

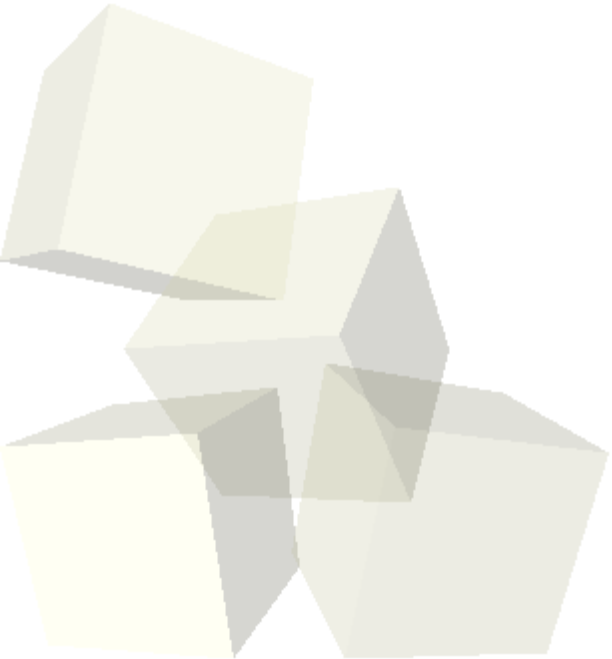
- What is the catch to Java?
- What makes Java better than other languages?
- How did Java get its name?





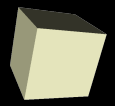
The API

- One of the greatest strengths of Java is the extensive library support that it has.
- When you program in Java you will probably need to keep the API open as a reference. It is far too big to try to memorize.
- Go to java.sun.com and look at the API for Java 6 SE.



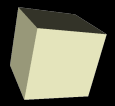
Types and Polymorphism

- What are types in programming languages? What are some examples of types you are used to?
- Polymorphism literally means many shapes. In a programming context it means many types. Polymorphic code is code that can work with more than one type.
- Universal polymorphism allows an infinite number of types. Ad hoc allows a finite number. Obviously the former is much more powerful and that is the type of polymorphism we will normally care about.
- Could you write polymorphic code in C?



Inheritance

- The primary way we get polymorphism in Java is through inheritance.
- We can specify that one class inherits from another class with the extends keyword in Java.
- Many class based object oriented languages include inheritance. It is a construct in languages that models the “is-a” relationship. You should only use inheritance when this relationship applies. Even when it does apply it isn't always the the right thing to do.



Two Sides of Inheritance

- Inheritance provides two functions.
- The original motivation for inheritance, and the root of the term, is that a subclass implicitly gets a copy of everything in the class that it is inheriting from. This means it has all data and functions. It can't directly access the things that are private.
- Inheritance also provides subtyping. If class B inherits from A, then any code that uses A will work with an object of type B. This is how we get our polymorphism in Java. We write code that works with supertypes and it automatically works with subtypes. This type of polymorphism is called inclusion polymorphism.

Inheritance Hierarchies

- The standard way of drawing out inheritance is through a tree-like hierarchy.
- In UML the arrows point from the subclass to the superclass. This is because the superclass doesn’t generally know of all of its subclasses but the subclasses know of the superclass.

Virtual Functions

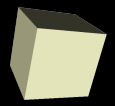
- One of the common powers of inheritance is that you don’t always have to use the methods defined by the superclass. You can override them in the subclass.
- Methods that can be overridden are called virtual methods. By default all methods in Java are virtual.
- A method invocation uses the definition “closest” to the actual class.

Final Keyword

- If you have a method that you don’t want to ever be overridden, you can declare it as final.
- You can also declare an entire class to be final in which case no subclasses can ever be written to inherit from it.
- The final keyword is greatly underused in Java. It requires thought, but should be used more.
- All immutable classes should be final. Otherwise subclasses might be created that aren't immutable and don't work with existing code.

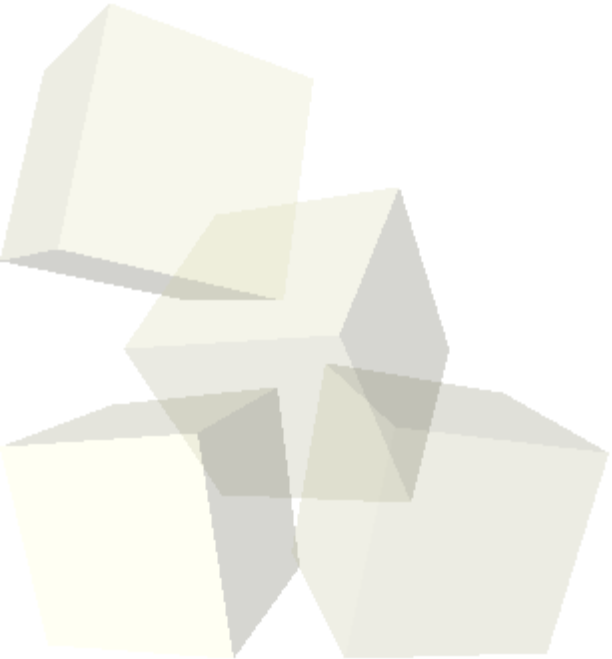
Abstract Keyword

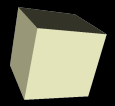
- You can declare a method in a class that doesn’t have an implementation. This method must be labeled as abstract.
- Any class that contains abstract methods must also be labeled as abstract.
- You use abstract methods when a superclass doesn’t have a good default implementation so all subclasses should override it and give their implementations.



Inclusion Polymorphism/Project

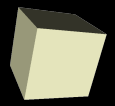
- Inclusion polymorphism is what allows my code to work with what you are going to be writing.
- You are going to create subtypes of the types I have defined. My code works with the supertypes and through inclusion polymorphism it will work with your subtypes as well.





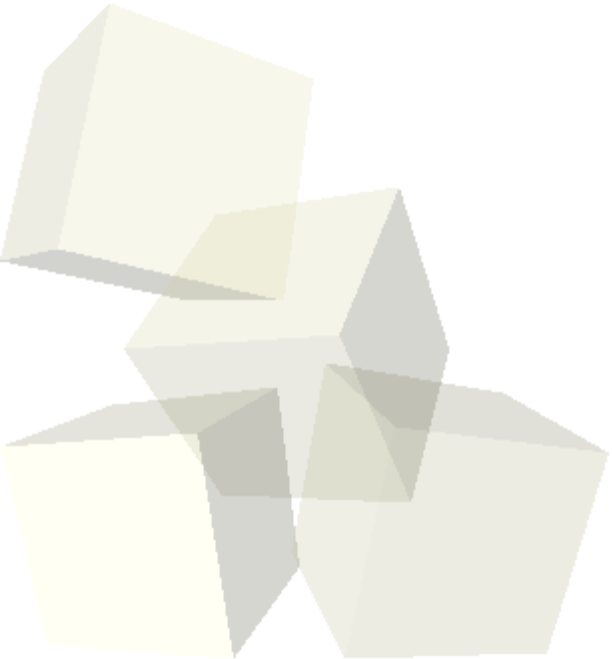
Restrictions in Java

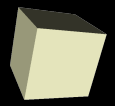
- Java places some restrictions on inheritance to simplify the language. The main restriction is that you can only extend one class. Doing otherwise, multiple inheritance, tends to make things very complex.
- There are times when you want to have a class be a subtype of two different types though. To allow this Java has a construct called an interface. Interfaces have no data (they can have static data) and all methods in them are abstract. They only define what you can do with them, not how to do it. You can implement as many interfaces as you want.



Interfaces Continued

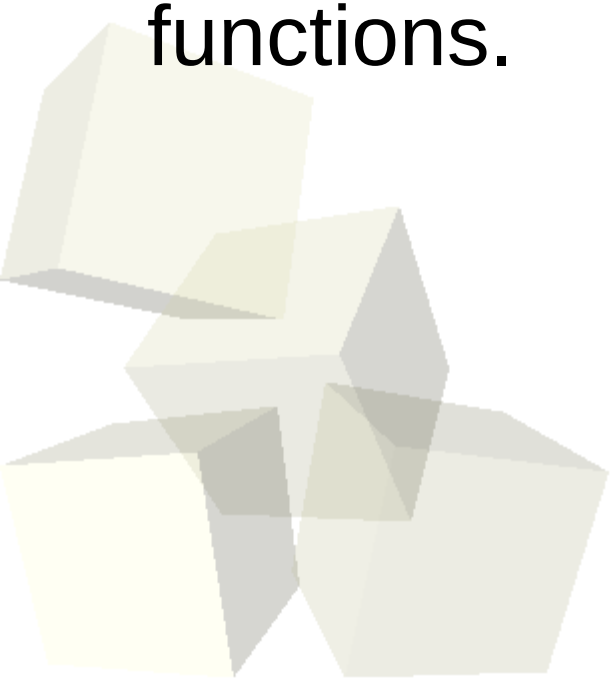
- Java allows multiple inheritance from interfaces because they can never create ambiguity.
- Implementing an interface only provides subtyping, not code reuse.
- Subtypes of interfaces need to implement all of the methods of that interface or they will be abstract.





Code

- Now I want you to watch me construct some code that uses inheritance and polymorphism.
- The reading runs you through the classic example of a shape. We can do something similar with ray-tracing code. I also want to use the example of a simple function of one variable as our supertype and then create subtypes for specific types of functions.



Minute Essay

- Does having me code in class help? Do you think I should try to take more time or less time to do coding? Remember that time spent coding isn't spent describing things in lecture.
- Interclass problem – Make a class hierarchy for furniture. Try to have it do something in a main to demonstrate that it works, but I'm not going for any type of functional application, just some print statements.