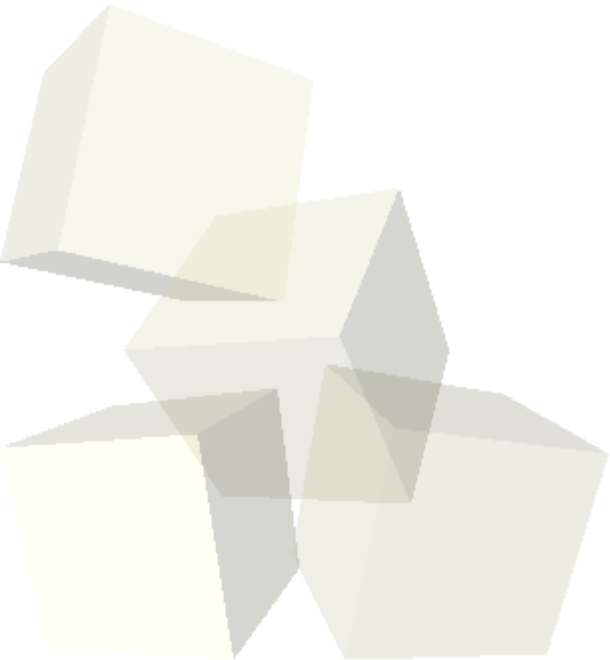
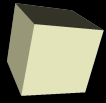




Remote Method Invocation

4/29/2008





Opening Discussion

- Solutions to the interclass problem.
- Do you have any questions about the assignment?
- Minute Essays
 - ◆ Alternatives to I/O streams.
 - ◆ How do you read '\n' from a stream?
 - ◆ Use of the decorator pattern in java.io and how it adds flexibility. Contrast with RandomAccessFile.
 - ◆ Open source Java.
 - ◆ Graphics libraries. Fractals in the drawing program. Difference between Graphics and Graphics2D.
 - ◆ Preparing for next semester's courses.

Networking with Multiple Messages

- We have played some with networking code now and seen how we can get computers to talk to one another.
- The socketing code we wrote is basically the way that networking really happens. Then again, machine language is the way programs really run, but that doesn't mean you want to write everything in machine code.
- Consider what we would need to do if we had programs that could send multiple types of information between them. For example, if our drawing program also had text chat and possibly sound clips and a whiteboard.



Remote Method Invocation (RMI)

- For complex networked applications Java has a wonderful package that allows “Remote Method Invocations”. This allows you to write code that looks like any normal code calling a method on an object, but the object might be located on a different machine.
- The RMI code will go through the effort of bundling information, sending it across the network, making the method call on the other machine, and sending the information back across the network.
- On the local machine you just call methods like normal with the exception they can throw a `RemoteException`.

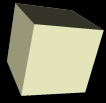


- To use RMI there are a few steps you have to take.
 - ◆ In your code you create remote interfaces that extend `java.rmi.Remote`. These specify the methods that can be called from other computers.
 - ◆ You then implement these interfaces with classes that extend a subclass of `RemoteObject`. Typically you extend `UnicastRemoteObject`.
 - ◆ Run `rmiregistry` on the server machine and using `java.rmi.Naming` to bind your primary class.
 - ◆ On the client use `java.rmi.Naming` to look up the server object and ask it for other objects you might need.
- Prior to Java 1.5 you had to use `rmic` to compile remote classes, but that is now automated.

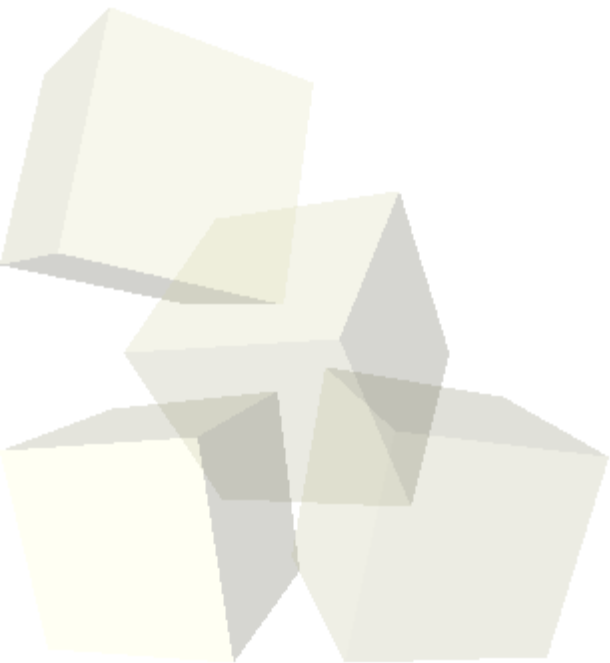


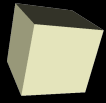
Object Passing in RMI

- Not everything can be passed or returned with RMI methods and different types are passed differently.
 - ◆ Primitives and Serializable objects are passed by value.
 - ◆ Remote objects are passed as by remote references.
 - ◆ Objects that aren't Serializable can't be passed and trying to do so will lead to an exception.
- These rules should make intuitive sense when you think about it. Inevitably RMI is using object serialization to pass things around. Primitives and normal serializable objects are passed completely across. Remote objects have stubs that are passed across.



- Let's try to think of something fun to do with RMI now. There is a bit of a mental learning curve, but once you get over that RMI makes it very easy to code networked applications of significant complexity.





- Describe the costs and benefits of using RMI as opposed to standard networking.
- Only one more day of class left.

