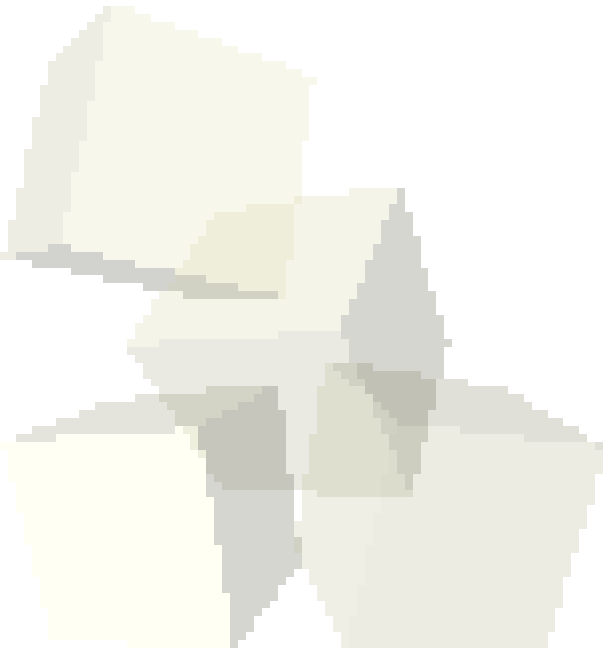




Generics, Enums, and Exceptions

2/3/2009





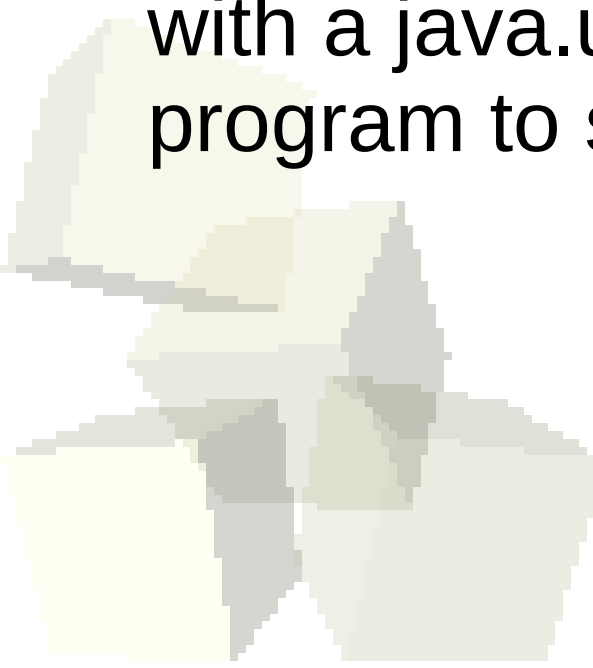
Opening Discussion

- Do you have any questions about the quiz?
- Let's look at solutions to the interclass problem.
- Minute essay comments
 - ◆ Why interfaces have no code reuse but allow multiple inheritance.
 - ◆ Feeling overwhelmed?
 - ◆ When would you use a local inner class?
 - ◆ Why make classes static?
 - ◆ My preferences: OS, PL, Skating rink.
 - ◆ How does computer hacking work?
 - ◆ Utility classes.
- Do you have any questions about the reading?
- Do you have any questions about the assignment?



Eclipse and Scanners

- Beginning this week we will start doing our coding in Eclipse. Let's open Eclipse and look around it quickly.
- With Eclipse we can't create objects and call methods on them the way we could in BlueJ. Instead, we will have to put in a main and run that.
- The easiest way to do simple text input in Java is with a `java.util.Scanner`. Let's write a little program to see how this works.



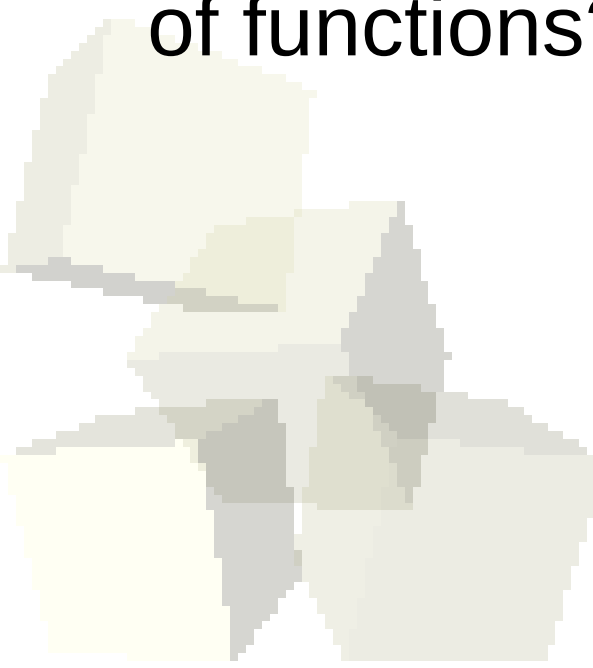


- The most significant feature added to Java 5.0 was that of generics. Generics provide a form of parametric polymorphism, typically for code that can take any type, but might be limited to a specific type for one instance.
- The most common use of this is for containers. Container classes typically should be able to hold anything, but any one container is generally intended to hold only one type.
- In practice, generics give you extra type safety and prevent you from doing a lot of type casts.



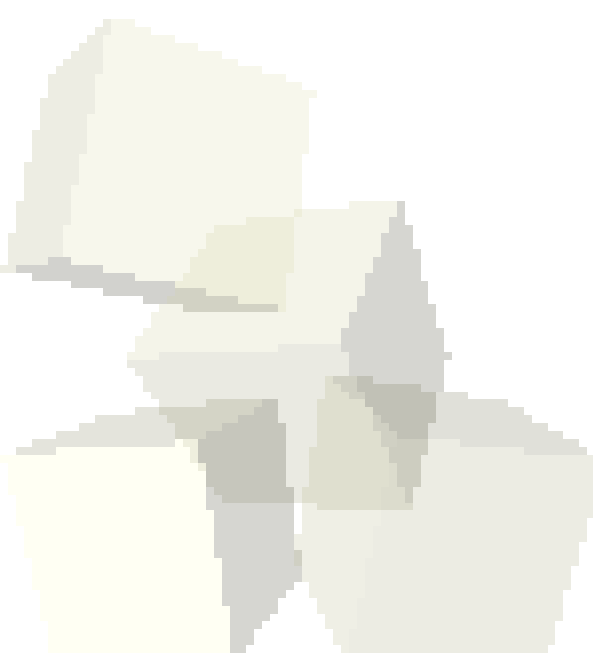
Making General Typesafe

- The problem with using the Object type for general polymorphism is that many different type checks have to be done at runtime and you lose static type safety.
- Using generics we can take our general function interface and make it static typesafe.
- How can you combine these more general types of functions?



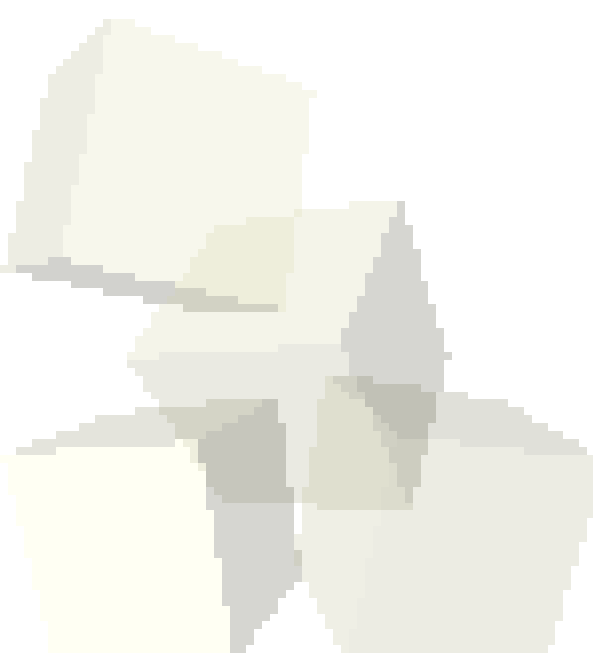


- C had enums. What were they supposed to do? What was the problem with them?
- Java includes enums as well. They serve the same goals, but lack the pitfalls.
- Java enum syntax can get quite complex, but the basic form is simple and very similar to C.



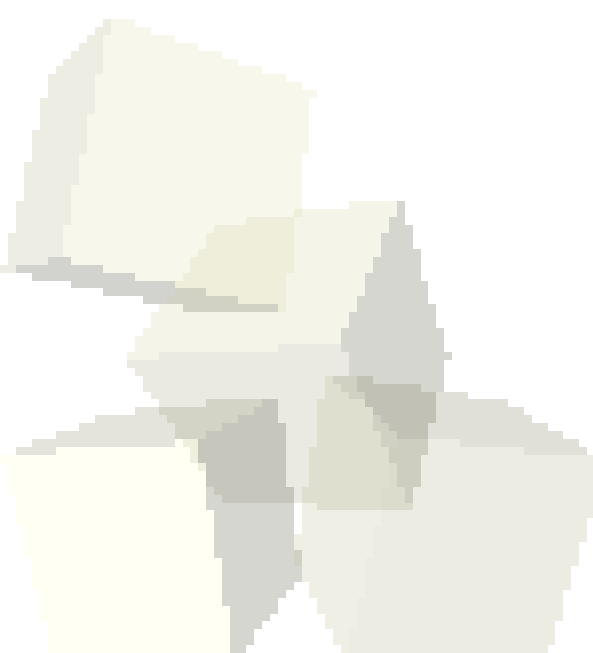


- How did you handle errors in C? (Consider the fopen function.)
- What are some problems with this method?





- Error handling in Java is done with exceptions, not return values or flags.
- Normal exceptions can't be ignored and they don't propagate. Runtime exceptions don't propagate.



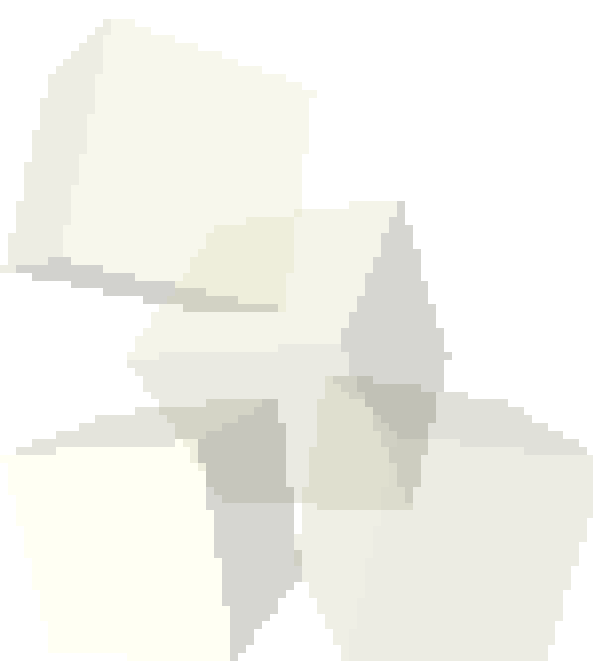


- For anything that isn't a RuntimeException you have to include handling code. For RuntimeExceptions it is optional.
- If you want to deal with a possible exception in the current method do this:
 - ◆ try {
 - statements
 - ◆ } catch(ExceptionType1 e) {
 - statements
 - ◆ } [catch(ExceptionType2 e) { ...} ...]
- If this method can't handle it you add a throws clause to the method and it will go up to the calling method.
 - ◆ Type name(args) throws ExType[,...] {...}



Additional Information

- Exceptions also have the advantage that they can provide additional information.
- Stack trace.
- Informative message.
- You can create your own exception classes. Strive to have them provide sufficient information for debugging.





- Do you have any questions about Java as a language? We are now moving from the language to libraries and problem solving.
- Interclass Problem – Write a program that uses a Scanner to read the contents of a file. The hint is that you will have a line something like `Scanner sc=new Scanner(new File("filename.txt"));`. You can decide what to do with the contents. Be creative. You might consider methods like `Double.parseDouble()`.