Let's look at some solutions to the interclass problem.
Do you have any questions about the project?
Early in the course I stressed how I wanted to feature creativity. Alice was a good tool to let you create interesting things quickly. Fundamentally it lacks power though.

Java provides a basically unlimited scope for creativity. It is limited only by your motivation.

I'd like to show you two projects that I've written in Java.

- The program you will use for submitting your assignments. This is short and sweet, but does something that we really need.
- A data plotting/analysis package that I've worked on for a while called SwiftVis.
There are basically two parts to any significant programming language: the language itself and the libraries for it.

The language describes the syntax and semantics of how you give instructions in your program. The library is a collection of code written by others that you can call on to help you do things.

Java has a fairly simple language, but very extensive libraries. We'll be covering the language in detail over the rest of the semester. We'll also see some parts of the libraries.

You can (and should) look at the API on-line to see the possibilities in the libraries.
Now that you know where to look for libraries, let's look at the language. We can do this by going back and looking at the interclass problem some.

A class contains methods, data members, and potentially other classes.

Inside a method we put statements. Basic statements end with semicolons.
- Variable declarations have type, variable name, and potentially initialization.
- Assignment statements have variable=expression.
- Method calls are also allowed as statements.
- We'll learn later about control flow statements.

Expressions have a type and a value.
• Just like in Alice, you can put comments in Java. They are probably more common in Java than in Alice.
• Single line comments can be added with //.
• Multiline comments can be added with /* and */.
• A comment that starts with /** is interpreted as being a special Javadoc comment. I'm not going to force you to learn how to do Javadoc comments, but they are what was used to create the API.
Java code, including libraries are arranged into packages.

Your code only sees the package it is in and java.lang. Other classes must be fully specified or imported. The import statement tells the Java compiler to look in a different location if it can't find something.

Eclipse can add import statements for you through the Ctrl-Alt-M or Ctrl-Alt-O keystrokes.
- We saw last time that we can use `System.out.println()` to print things.
- Your book likes `System.out.printf()`, but you won't see me use that much.
- Let's look in the API to see what `System.out` really is and what all we can do with it.
- Now let's write a bit of code to print some things.
The easiest way to do text input in Java is with the Scanner class. Technically this is java.util.Scanner so we will normally have an input statement.

Let's look at this in the API as well.

Now I'm going to write some code that uses both a Scanner and prints output. It might not all make sense at this point, but I want to do it for demonstration purposes.
What similarities do you see between Java and Alice? What things are different?

Remember that the project is due Friday.

Interclass Problem – Write code that uses input and output in Java, but isn't right from your book.