Let's look at solutions to the interclass problem.
Reference Types

- The “built in” primitive types are not the only types in Java. Classes define types as well. These are called reference types.
- We declare variables for reference types just like we would for primitive types.
- Reference types can either refer to an object of the proper type, or they can be null.
- Trying to use a null reference causes an exception.
- You have already used a reference type variable when you use Scanner. Scanner is a class in the Java libraries.
What is a Reference?

- Reference variables are called that for a reason. Instead of being an object, they refer to an object.
- This is in contrast to a primitive variable which actually is the thing you declare it to be.
- Because of this, a single reference variable could refer to several different objects over its life.
To make a new object we use the new keyword. The syntax for making an object is as follows:
- new Type(argumentList)

You will often see this on the right side of an assignment.

This tells Java to go out and get memory and set up an object of the specified type using the values that are passed in.

You have seen this already when working with a Scanner object to read values.
Once you have a reference variable that refers to a real object you can use the dot notation to call methods on that object.

This is what you have been doing with the Scanner object when you call methods like `nextDouble()`.
One of the most fundamental classes in Java is the String class. It represents any set of characters that you need to work with.

Let's look at String in the API.

The + operator is overloaded for String. Operators only work on primitives and, in this one instance, Strings.

Difference between null and empty Strings.

String literals are put in double quotes.

String objects are immutable. They don't change. Instead you build new String objects.

Checking equality.

- Use the equals method, not ==.
There are some other classes that are very fundamental to Java that your book doesn't mention.

These are the wrapper classes in Java that have names like primitives, but start with capital letters.

For our purposes, the main use of these classes are the helpful methods they in them. These include things like converting strings to numbers.
Write a little segment of code that has the user input a word and prints out how many letters are in the word.

Interclass Problem – Do problem 8.8. Warning: this isn't hard to program, but will take some thought.