Programming with Objects

Some Built-in Classes

Although the focus of object-oriented programming is generally on the design and implementation of new classes, it's important not to forget that the designers of Java have already provided a large number of reusable classes. Some of these classes are meant to be extended to produce new classes, while others can be used directly to create useful objects.

A string can be built up from smaller pieces using the + operator, but this is not always efficient. If `str` is a *String* and `ch` is a character, then executing the command "`str = str + ch;`" involves creating a whole new string that is a copy of `str`, with the value of `ch` appended onto the end. Copying the string takes some time. Building up a long string letter by letter would require a surprising amount of processing. The class *StringBuilder* makes it possible to be efficient about building up a long string from a number of smaller pieces. To do this, you must make an object belonging to the *StringBuilder* class. For example:

```
StringBuilder builder = new StringBuilder();
```

Like a *String*, a *StringBuilder* contains a sequence of characters. However, it is possible to add new characters onto the end of a `StringBuilder` without continually making copies of the data that it already contains. If `x` is a value of any type and `builder` is the variable defined above, then the command `builder.append(x)` will add `x`, converted into a string representation, onto the end of the data that was already in the builder. This can be done more efficiently than copying the data every time something is appended. A long string can be built up in a *StringBuilder* using a sequence of `append()` commands. When the string is complete, the function `builder.toString()` will return a copy of the string in the builder as an ordinary value of type *String*. The *StringBuilder* class is in the standard package `java.lang`, so you can use its simple name without importing it.

A number of useful classes are collected in the package `java.util`. For example, this package contains classes for working with collections of objects. Another class in this package, `java.util.Date`, is used to represent times. When a `Date` object is constructed without parameters, the result represents the current date and time, so an easy way to display this information is:

```
System.out.println( new Date() );
```

Of course, since it is in the package `java.util`, in order to use the `Date` class in your program, you must make it available by importing it with one of the statements "`import java.util.Date;`" or "`import java.util.*;`" at the beginning of your program.

The main point of all this, again, is that many problems have already been solved, and the solutions are available in Java's standard classes. If you are faced with a task that looks like it should be fairly common, it might be worth looking through a Java reference to see whether someone has already written a class that you can use.

The class "Object"

Class *Object* defines several instance methods that are inherited by every other class. These methods can be used with any object whatsoever.

The instance method `toString()` in class *Object* returns a value of type *String* that is supposed to be a string representation of the object. You've already used this method implicitly, any time you've printed out an object or concatenated an object onto a string. When you use an object in a context that requires a string, the object is automatically converted to type *String* by calling its `toString()` method.

Object-oriented Analysis and Design

Well-designed classes are software components that can be reused without editing. A well-designed class is not carefully crafted to do a particular job in a particular program. Instead, it is crafted to model some particular type of object or a single coherent concept. Since objects and concepts can recur in many problems, a well-designed class is likely to be reusable without modification in a variety of projects.

Take for example the multiple dice problems we have encountered, a dice class could be reused which would simplify the problem.