Chapter 6 - A First Look at Classes

Chapter 6 introduces the idea of using classes in Java programming, the ability to store data using constructors and instance fields, along with the methods that classes typically contain such as mutators, accessors, and calculators. Java is an object-oriented language, where an object is a software component that exists in memory with the ability to execute lines of instructed code, and classes are required to create these objects or instances.

A class file is a different file from the file that contains the main method. When creating a class, the first things that are required are the creation of instance fields and constructors. An object or instance can contain multiple instance fields, which are values that the object can hold. These instance fields can be of any data type such as int, double, boolean, String, etc. Note that these variables must be declared as private variables that are exclusive to the class in order to avoid unwanted tampering of data from other classes. When an object is created, the instance fields for the object needs to be filled in, and this is where constructors come into play. Constructors have the ability to evaluate arguments and set these arguments into different instance fields. This ability is, of course, dependant on the instructions written for the constructor. ProgramOneCSix.java and ProgramOneCSixClass.java show an example of how instance fields and constructors work.

Once the constructors and instance fields have been established, the class file can contain multiple methods that when called, can access or manipulate the instance fields. These methods are called accessors, mutators, and calculators and need to be public so that other Java files can access them. The following table highlights some of the differences between these methods:

Method	Command	Function
Accessor	getValue();	Accesses the data in a particular instance field
Mutator	setValue(x);	Changes the value of the data in a particular instance field
Calculator	calculate(x);	Manipulates the value of the data in a particular instance field depending on the instructions

*Note that the command names are not exclusive to the following examples. Calculator methods can be named anything that is relevant. The same is true for mutators and accessors, except that the get and set parts are usually required to provide clarity. ProgramTwoCSix.java and ProgramTwoCSixClass.java show an example of how these different methods work.

After creating a complete class file, the final thing to do is to use the class file in another Java program. In order to use these classes, an object needs to be created beforehand. The creation of this object is similar to when objects were created from the Scanner class and File class in previous examples. The commands for creating an object are all roughly similar.

ClassName objectName = new ClassName(Arguments);

Using ProgramTwoCSix.java and ProgramTwoCSixClass.java as an example, it is possible to evaluate the command above. The *ClassName* aspect of the command refers to the name of the class that the object is to be created from, in this case, it is the ProgramTwoCSixClass class. The *objectName* is the name of the object that has been created. It is possible to create more than one object for one class, and there are three objects in this scenario, *uWaterloo*, *uToronto*, and *mcMaster*. The *new ClassName(Arguments);* part of the command indicates the creation of the object, with the arguments in the bracket being arguments for the constructor. To utilize the methods that have been created in the class, the following command can be used.

objectName.methodName(x);

The *objectName* is the name of the object created beforehand, and the *methodName* is the name of the method that is going to be used (mutators, accessors, calculators). The arguments for these methods are placed inside of the brackets as introduced in chapter 5.

Objects can also be used as arguments in a Java method. When writing the method header, the data type specified should be the class name, and when the method is called, the argument should be the object. The method *outputCreator(x)* in ProgramTwoCSix.java shows an example of how to use objects as method arguments.