Chapters 8: Objects and Classes

Based on Introduction to Java Programming, Y. Daniel Liang, Brief Version, 9/E

Topics Covered

1. Creating Classes and Objects  
2. Constructors  
3. Reference Variables ↔ Objects  
4. Accessing Object’s Data and Methods  
5. Default Values for Data Fields  
6. Using Classes from Java Library  
7. static Variables and Methods, and Constants  
8. Visibility Modifiers: Four P’s  
9. Data Field Encapsulation  
10. Passing Objects to Methods  
11. Arrays of Objects  
12. Using Javadoc Style Comments
1 Creating Classes and Objects

An object represents an entity in the real world. For example: student, table, car, circle, university, book, bookstore.

Objects have:

- **state** represented by **data fields**
- **behavior** defined by **methods**

For example:

<table>
<thead>
<tr>
<th>circle has:</th>
<th>student has:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>state</strong></td>
<td><strong>state</strong></td>
</tr>
<tr>
<td>- radius</td>
<td>- name</td>
</tr>
<tr>
<td><strong>behavior</strong></td>
<td>- id</td>
</tr>
<tr>
<td>- getArea()</td>
<td>- grades</td>
</tr>
<tr>
<td>- getPerimeter()</td>
<td>- address</td>
</tr>
<tr>
<td>- setRadius()</td>
<td>- ...</td>
</tr>
<tr>
<td>- ...</td>
<td>- computeGPA()</td>
</tr>
<tr>
<td></td>
<td>- computeMajorGPA()</td>
</tr>
<tr>
<td></td>
<td>- getID()</td>
</tr>
<tr>
<td></td>
<td>- changeAddress()</td>
</tr>
<tr>
<td></td>
<td>- ...</td>
</tr>
</tbody>
</table>

A **class** is a template for creating objects of the same type. For example, a Circle class can be used to create multiple Circle objects.

A **constructor** is a special kind of method that is used to construct an object. A class can have multiple constructors (i.e. different ways of creating objects).

See TestSimpleCircle.java for an example of a class.
2 Constructors

A constructor is a special method that is used to construct/creat an object. Constructors have to obey several special rules:

- The name of the constructor is always the same as the name of the class, even if multiple constructors are present (uses method overloading).
- There is no return type, not even void.
- The constructor is called automatically when a new operator is used to create an object, ex. Circle c = new Circle();
The constructor that can be called with no arguments is called the default constructor or no-arg constructor. If a class has no constructors explicitly defined, Java provides a default constructor with empty body. It is provided only if NO constructors are defined in the class. Classes do not have to have default constructors, but it is usually a good idea to provide one. (Although there might be good reasons for to have such a constructor in some situations.)

3 Reference Variables ⇔ Objects

The statement

```
ClassName objectRefVariable;
```

declares a reference variable that can be used to store a memory address at which the actual object is stored.

In order to create an object, you need to use the new operator

```
objectRefVariable = new ClassName(...);
```

This is similar to how an array-name stores the memory addresses of the location where the array is stored. The actual array storage needs to be allocated using the new operator.

As with all the other declarations/creation statements, the above two lines can be combined into a single statement

```
ClassName objectRefVariable = new ClassName(...);
```

NOTE: observe the paranthesis in the last two statements. This is call to the constructor that may or may not take parameters.

Occasionally, you may want to create an object in memory (usually temporary object) that is not pointed to by any object reference variable. Those objects are called anonymous objects, since they do not have a name. For example

```
System.out.printf("The area of circle with radius %f is %f. \n", 
5, Circle(5).getArea() );
```

After this statement executes the object is still in the memory, but there is no way to access it. The memory that it occupies eventually gets reclaimed by Java garbage collection.

**Common error**  Confusing object assignment with reference variable assignment.

```
Circle c1 = Circle(5);
Circle c2 = Circle(17);
c1 = c2;
```

The last statement results in c1 and c2 pointing to the same Circle object in memory (the one with radius 17). It does not copy one Circle object to another.
4 Accessing Object’s Data and Methods

The **dot operator** or **object member access operator** is used to access data fields and methods of an object:

```
objectReferenceName.dataField
    accesses specific data field in the object

objectReferenceName.method()
    invokes a specific method on the object
```

The **instance variables** and **instance methods** are the variables and methods that can be accessed/invoked using a specific instance of the class (not using the class name). The object on which an instance method is invoked is called the **calling object**.

5 Default Values for Data Fields

When a variable of any kind is created the memory associated with that variable contains some values. The data fields of a newly created object are filled, by default, with zero bits. Depending on the type of the data field, the interpretation of zeroed memory is slightly different.

- numerical types (for both integers and floating point numbers) - set to number zero
- char variables - set to \u0000, which is a nul character, ‘\0’ or ‘’
- Boolean variables - set to false
- reference variables - set to null, which is an invalid memory address (from the point of view of the running program).

**NullPointerException** occurs if you try to use a reference variable without assigning an object to it first. You will see a lot of these.

6 Using Classes from Java Library

You have already seen some classes that come with Java. We will be using more and more of these.

- Math
- Arrays
- Date
- Random
7 static Variables and Methods, and Constants

A static variable (class variables) is shared by all objects of the class. Such variables represent the whole class of objects, rather than one instance of the class (object).

A static method (class method) is called on a class, rather than a specific object. Static methods do not have access to instance variables.

static keyword in the declaration of a variable or method indicates that it should be a static variable/method.

See CircleWithStaticMembers.java

All methods in the Math class are static. In fact, you cannot create an instance of that class.

Constants are shared among all instances of the class. They should be declared with final and static modifiers.

8 Visibility Modifiers: Four P’s

Data fields and methods

There are four modifiers that can be used with data fields / methods in a class definitions (only one of these can be specified at a time, but the modifiers can be different for different data fields / methods in the same class):

public any other class in the world can access the data fields / methods

package (default, when no other modifier is specified) any other class in the same package can access the data fields / methods

protected only members of the same class and subclasses (to be discussed in chapter 11) can access the data fields / methods

private only the members of the same class can access the data fields / methods

Constructors should be public in general. Exception: to prevent users of the class from creating an instance, for example, Math class has a private constructor.

Classes

The class itself can have access modifiers. Either public or nothing is used with a class. If no modifier is specified, only classes from within the same package can access the class. Each public class needs to be specified in its own file.
9 Data Field Encapsulation

The data fields should be private to the class. Why?
The class should provide

getters / accessors / get methods

and

setters / mutators / set methods

to allow for controlled access to the data fields.
See CircleWithPrivateDataFields.java.

10 Passing Objects to Methods

See IncrementExample.java

11 Arrays of Objects

See SortingCircles.java

12 Using Javadoc Style Comments

For Liang’s brief introduction about using Javadoc see the supplementary materials at http://www.cs.armstrong.edu/liang/intro9e/supplement/javadoccomments.pdf.

• Use /** ... */ instead of /* ... */ or // for comments if they should be used in the javadoc generated documentation.

• Run javadoc on your *.java files to generate HTML documentation.

• Some available tags (that you should start using) are:
  – @author [author name]: identifies the author(s) of a class or interface.
  – @version [version]: gives the version of a class or interface.
  – @param [parameter name] [parameter description]: describes the parameters in a method or constructor.
  – @return [description of return]: describes a return value from a method.