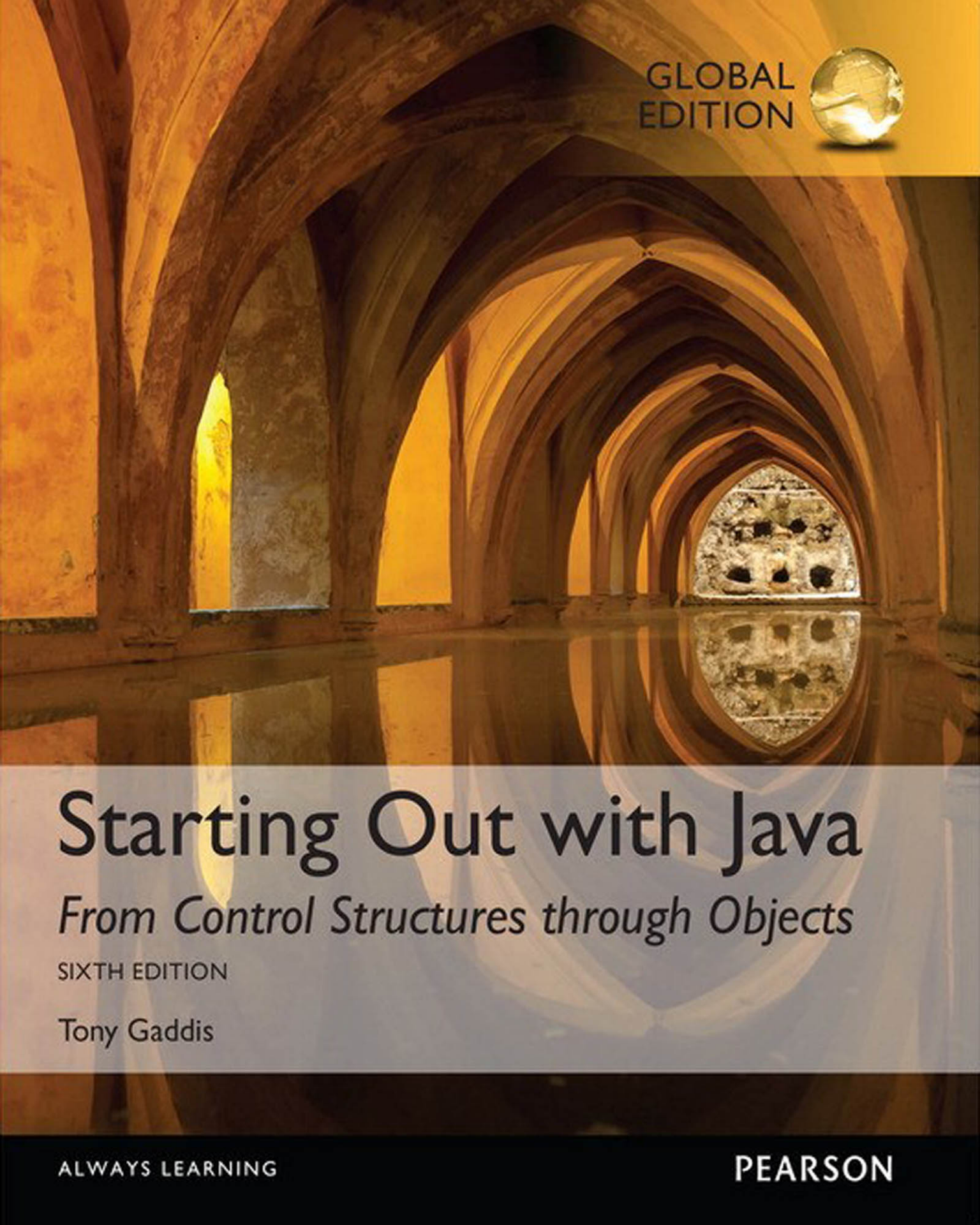


Starting Out with Java

From Control Structures through Objects

SIXTH EDITION

Tony Gaddis

ALWAYS LEARNING

PEARSON

STARTING OUT WITH

JAVA™

From Control Structures through Objects

```
12        Rectangle box = new Rectangle();
13
14        // Indicate what we are doing.
15        System.out.println("Sending the value 10.0 " +
16                           "to the setLength method.");
17
18        // Call the box object's setLength method.
19        box.setLength(10.0);
20
21        // Indicate we are done.
22        System.out.println("Done.");
23     }
24  }
```

Program Output

```
Sending the value 10.0 to the setLength method.
Done.
```

The program in Code Listing 6-3 must be saved as *LengthDemo.java* in the same folder or directory as the file *Rectangle.java*. The following command can then be used with the Sun JDK to compile the program:

```
javac LengthDemo.java
```

When the compiler reads the source code for `LengthDemo.java` and sees that a class named `Rectangle` is being used, it looks in the current folder or directory for the file *Rectangle.class*. That file does not exist, however, because we have not yet compiled *Rectangle.java*. So, the compiler searches for the file `Rectangle.java` and compiles it. This creates the file *Rectangle.class*, which makes the `Rectangle` class available. The compiler then finishes compiling `LengthDemo.java`. The resulting *LengthDemo.class* file may be executed with the following command:

```
java LengthDemo
```

The output of the program is shown at the bottom of Code Listing 6-3.

Let's look at each statement in this program's `main` method. First, the program uses the following statement, in line 12, to create a `Rectangle` object and associate it with a variable:

```
Rectangle box = new Rectangle();
```

Let's dissect the statement into two parts. The first part of the statement,

```
Rectangle box
```

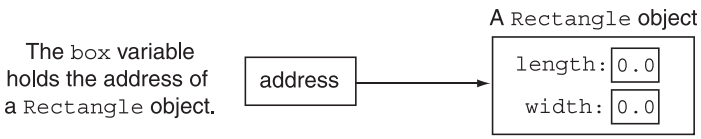
declares a variable named `box`. The data type of the variable is `Rectangle`. (Because the word `Rectangle` is not the name of a primitive data type, Java assumes it to be the name of a class.) Recall that a variable of a class type is a reference variable, and it holds the memory address of an object. When a reference variable holds an object's memory address, it is said

that the variable references the object. So, the variable box will be used to reference a Rectangle object. The second part of the statement is as follows:

```
= new Rectangle();
```

This part of the statement uses the key word new, which creates an object in memory. After the word new, the name of a class followed by a set of parentheses appears. This specifies the class that the object should be created from. In this case, an object of the Rectangle class is created. The memory address of the object is then assigned (by the = operator) to the variable box. After the statement executes, the variable box will reference the object that was created in memory. This is illustrated in Figure 6-11.

Figure 6-11 The box variable references a Rectangle class object



Notice that Figure 6-11 shows the Rectangle object's length and width fields set to 0. All of a class's numeric fields are initialized to 0 by default.

TIP: The parentheses in this statement are required. It would be an error to write the statement as follows:

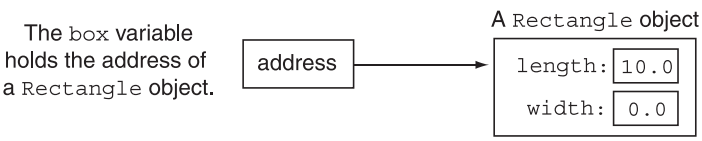
```
Rectangle box = new Rectangle; // ERROR!!
```

The statement in lines 15 and 16 uses the System.out.println method to display a message on the screen. The next statement, in line 19, calls the box object's setLength method as follows:

```
box.setLength(10.0);
```

This statement passes the argument 10.0 to the setLength method. When the method executes, the value 10.0 is copied into the len parameter variable. The method assigns the value of len to the length field and then terminates. Figure 6-12 shows the state of the box object after the method executes.

Figure 6-12 The state of the box object after the setLength method executes



Writing the `setWidth` Method

Now that we've seen how the `setLength` method works, let's add the `setWidth` method to the `Rectangle` class. The `setWidth` method is similar to `setLength`. It accepts an argument, which is assigned to the `width` field. Code Listing 6-4 shows the updated `Rectangle` class. The `setWidth` method is in lines 28 through 31. (This file is stored in the source code folder *Chapter 06\Rectangle Class Phase 2*.)

Code Listing 6-4 **(Rectangle.java)**

```
1  /**
2     Rectangle class, phase 2
3     Under construction!
4  */
5
6  public class Rectangle
7  {
8     private double length;
9     private double width;
10
11    /**
12       The setLength method stores a value in the
13       length field.
14       @param len The value to store in length.
15    */
16
17    public void setLength(double len)
18    {
19       length = len;
20    }
21
22    /**
23       The setWidth method stores a value in the
24       width field.
25       @param w The value to store in width.
26    */
27
28    public void setWidth(double w)
29    {
30       width = w;
31    }
32 }
```

The setWidth method has a parameter variable named w, which is assigned to the width field. For example, assume that box references a Rectangle object and the following statement is executed:

```
box.setWidth(20.0);
```

After this statement executes, the box object's width field will be set to 20.0.

Writing the getLength and getWidth Methods

Because the length and width fields are private, we wrote the setLength and setWidth methods to allow code outside the Rectangle class to store values in the fields. We must also write methods that allow code outside the class to get the values that are stored in these fields. That's what the getLength and getWidth methods will do. The getLength method will return the value stored in the length field, and the getWidth method will return the value stored in the width field.

Here is the code for the getLength method:

```
public double getLength()
{
    return length;
}
```

Assume that size is a double variable and that box references a Rectangle object, and the following statement is executed:

```
size = box.getLength();
```

This statement assigns the value that is returned from the getLength method to the size variable. After this statement executes, the size variable will contain the same value as the box object's length field.

The getWidth method is similar to getLength. The code for the method follows:

```
public double getWidth()
{
     return width;
}
```

This method returns the value that is stored in the width field. For example, assume that size is a double variable and that box references a Rectangle object, and the following statement is executed:

```
size = box.getWidth();
```

This statement assigns the value that is returned from the getWidth method to the size variable. After this statement executes, the size variable will contain the same value as the box object's width field.

Code Listing 6-5 shows the Rectangle class with all of the members we have discussed so far. The code for the getLength and getWidth methods is shown in lines 33 through 53. (This file is stored in the source code folder *Chapter 06\Rectangle Class Phase 3*.)

Code Listing 6-5 (Rectangle.java)

```
1  /**
2     Rectangle class, phase 3
3     Under construction!
4  */
5
6  public class Rectangle
7  {
8     private double length;
9     private double width;
10
11    /**
12       The setLength method stores a value in the
13       length field.
14       @param len The value to store in length.
15    */
16
17    public void setLength(double len)
18    {
19       length = len;
20    }
21
22    /**
23       The setWidth method stores a value in the
24       width field.
25       @param w The value to store in width.
26    */
27
28    public void setWidth(double w)
29    {
30       width = w;
31    }
32
33    /**
34       The getLength method returns a Rectangle
35       object's length.
36       @return The value in the length field.
37    */
38
39    public double getLength()
40    {
41       return length;
42    }
43
44    /**
45       The getWidth method returns a Rectangle
46       object's width.
```

```
47       @return The value in the width field.
48    */
49
50    public double getWidth()
51    {
52       return width;
53    }
54 }
```

Before continuing we should demonstrate how these methods work. Look at the program in Code Listing 6-6. (This file is also stored in the source code folder *Chapter 06\Rectangle Class Phase 3*.)

Code Listing 6-6 **(LengthWidthDemo.java)**

```
1  /**
2     This program demonstrates the Rectangle class's
3     setLength, setWidth, getLength, and getWidth methods.
4  */
5
6  public class LengthWidthDemo
7  {
8     public static void main(String[] args)
9     {
10       // Create a Rectangle object.
11       Rectangle box = new Rectangle();
12
13       // Call the object's setLength method, passing 10.0
14       // as an argument.
15       box.setLength(10.0);
16
17       // Call the object's setWidth method, passing 20.0
18       // as an argument.
19       box.setWidth(20.0);
20
21       // Display the object's length and width.
22       System.out.println("The box's length is " +
23                          box.getLength());
24       System.out.println("The box's width is " +
25                          box.getWidth());
26    }
27 }
```

Program Output

```
The box's length is 10.0
The box's width is 20.0
```