3

Introduction to Classes and Objects

You will see something new. Two things. And I call them Thing One and Thing Two.
—Dr. Theodor Seuss Geisel

Nothing can have value without being an object of utility.
—Karl Marx

Your public servants serve you right.
—Adlai E. Stevenson

Knowing how to answer one who speaks,
To reply to one who sends a message.
—Amenemope

OBJECTIVES

In this chapter you will learn:

- What classes, objects, methods and instance variables are.
- How to declare a class and use it to create an object.
- How to declare methods in a class to implement the class's behaviors.
- How to declare instance variables in a class to implement the class's attributes.
- How to call an object's methods to make those methods perform their tasks.
- The differences between instance variables of a class and local variables of a method.
- How to use a constructor to ensure that an object's data is initialized when the object is created.
- The differences between primitive and reference types.

Student Solution Exercises

3.5 What is the purpose of keyword `new`? Explain what happens when this keyword is used in an application.

ANS: The purpose of keyword `new` is to create an object of a class. When keyword `new` is used in an application, first a new object of the class to the right of `new` is created, then the class's constructor is called to ensure that the object is initialized properly.

3.7 Explain the purpose of an instance variable.

ANS: A class provides an instance variable (or several instance variables) when each object of the class must maintain information separately from all other objects of the class. For example, a class called `Account` that represents a bank account provides an instance variable to represent the balance of the account. Each `Account` object maintains its own balance, but does not know the balances of the bank's other accounts.

3.9 Explain how a program could use class `Scanner` without importing the class from package `java.util`.

ANS: If every use of a class's name in a program is fully qualified, there is no need to import the class. A class's fully qualified name consists of the class's package followed by the class name. For example, a program could use class `Scanner` without importing it if every use of `Scanner` in the program is specified as `java.util.Scanner`.

3.13 Create a class called `Invoice` that a hardware store might use to represent an invoice for an item sold at the store. An `Invoice` should include four pieces of information as instance variables—a part number (type `String`), a part description (type `String`), a quantity of the item being purchased (type `int`) and a price per item (`double`). Your class should have a constructor that initializes the four instance variables. Provide a *set* and a *get* method for each instance variable. In addition, provide a method named `getInvoiceAmount` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a `double` value. If the quantity is not positive, it should be set to `0`. If the price per item is not positive, it should be set to `0.0`. Write a test application named `InvoiceTest` that demonstrates class `Invoice`'s capabilities.

ANS:

```
1  // Exercises 3.13 Solution: Invoice.java
2  // Invoice class.
3
4  public class Invoice
5  {
6     private String partNumber;
7     private String partDescription;
8     private int quantity;
9     private double pricePerItem;
10
11    // four-argument constructor
12    public Invoice( String part, String description, int count,
13       double price )
14    {
15       partNumber = part;
16       partDescription = description;
17
18       if ( count > 0 ) // determine whether count is positive
19          quantity = count; // valid count assigned to quantity
20
21       if ( price > 0.0 ) // determine whether price is positive
```

```
22            pricePerItem = price; // valid price assigned to pricePerItem
23      } // end four-argument Invoice constructor
24
25      // set part number
26      public void setPartNumber( String part )
27      {
28         partNumber = part;
29      } // end method setPartNumber
30
31      // get part number
32      public String getPartNumber()
33      {
34         return partNumber;
35      } // end method getPartNumber
36
37      // set description
38      public void setPartDescription( String description )
39      {
40         partDescription = description;
41      } // end method setPartDescription
42
43      // get description
44      public String getPartDescription()
45      {
46         return partDescription;
47      } // end method getPartDescription
48
49      // set quantity
50      public void setQuantity( int count )
51      {
52         if ( count > 0 ) // determine whether count is positive
53            quantity = count; // valid count assigned to quantity
54
55         if ( count <= 0 ) // determine whether count is zero or negative
56            quantity = 0; // invalid count; quantity set to 0
57      } // end method setQuantity
58
59      // get quantity
60      public int getQuantity()
61      {
62         return quantity;
63      } // end method getQuantity
64
65      // set price per item
66      public void setPricePerItem( double price )
67      {
68         if ( price > 0.0 ) // determine whether price is positive
69            pricePerItem = price; // valid price assigned to pricePerItem
70
71         if ( price <= 0.0 ) // determine whether price is zero or negative
72            pricePerItem = 0.0; // invalid price; pricePerItem set to 0.0
73      } // end method setPricePerItem
74
75      // get price per item
76      public double getPricePerItem()
```

```
77      {
78         return pricePerItem;
79      } // end method getPricePerItem
80
81      // calculates and returns the invoice amount
82      public double getInvoiceAmount()
83      {
84         return getQuantity() * getPricePerItem(); // calculate total cost
85      } // end method getPaymentAmount
86   } // end class Invoice
```

```
1   // Exercises 3.13 Solution: InvoiceTest.java
2   // Application to test class Invoice.
3
4   public class InvoiceTest
5   {
6      public static void main( String args[] )
7      {
8         Invoice invoice1 = new Invoice( "1234", "Hammer", 2, 14.95 );
9
10        // display invoice1
11        System.out.println( "Original invoice information" );
12        System.out.printf( "Part number: %s\n", invoice1.getPartNumber() );
13        System.out.printf( "Description: %s\n",
14           invoice1.getPartDescription() );
15        System.out.printf( "Quantity: %d\n", invoice1.getQuantity() );
16        System.out.printf( "Price: %.2f\n", invoice1.getPricePerItem() );
17        System.out.printf( "Invoice amount: %.2f\n",
18           invoice1.getInvoiceAmount() );
19
20        // change invoice1's data
21        invoice1.setPartNumber( "001234" );
22        invoice1.setPartDescription( "Yellow Hammer" );
23        invoice1.setQuantity( 3 );
24        invoice1.setPricePerItem( 19.49 );
25
26        // display invoice1 with new data
27        System.out.println( "\nUpdated invoice information" );
28        System.out.printf( "Part number: %s\n", invoice1.getPartNumber() );
29        System.out.printf( "Description: %s\n",
30           invoice1.getPartDescription() );
31        System.out.printf( "Quantity: %d\n", invoice1.getQuantity() );
32        System.out.printf( "Price: %.2f\n", invoice1.getPricePerItem() );
33        System.out.printf( "Invoice amount: %.2f\n",
34           invoice1.getInvoiceAmount() );
35
36        Invoice invoice2 = new Invoice( "5678", "Paint Brush", -5, -9.99 );
37
38        // display invoice2
39        System.out.println( "\nOriginal invoice information" );
40        System.out.printf( "Part number: %s\n", invoice2.getPartNumber() );
41        System.out.printf( "Description: %s\n",
42           invoice2.getPartDescription() );
43        System.out.printf( "Quantity: %d\n", invoice2.getQuantity() );
```

```
44         System.out.printf( "Price: %.2f\n", invoice2.getPricePerItem() );
45         System.out.printf( "Invoice amount: %.2f\n",
46            invoice2.getInvoiceAmount() );
47
48         // change invoice2's data
49         invoice2.setQuantity( 3 );
50         invoice2.setPricePerItem( 9.49 );
51
52         // display invoice2 with new data
53         System.out.println( "\nUpdated invoice information" );
54         System.out.printf( "Part number: %s\n", invoice2.getPartNumber() );
55         System.out.printf( "Description: %s\n",
56            invoice2.getPartDescription() );
57         System.out.printf( "Quantity: %d\n", invoice2.getQuantity() );
58         System.out.printf( "Price: %.2f\n", invoice2.getPricePerItem() );
59         System.out.printf( "Invoice amount: %.2f\n",
60            invoice2.getInvoiceAmount() );
61
62      } // end main
63
64   } // end class InvoiceTest
```

```
Original invoice information
Part number: 1234
Description: Hammer
Quantity: 2
Price: 14.95
Invoice amount: 29.90

Updated invoice information
Part number: 001234
Description: Yellow Hammer
Quantity: 3
Price: 19.49
Invoice amount: 58.47

Original invoice information
Part number: 5678
Description: Paint Brush
Quantity: 0
Price: 0.00
Invoice amount: 0.00

Updated invoice information
Part number: 5678
Description: Paint Brush
Quantity: 3
Price: 9.49
Invoice amount: 28.47
```

3.15 Create a class called `Date` that includes three pieces of information as instance variables—a month (type `int`), a day (type `int`) and a year (type `int`). Your class should have a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a *set*

and a *get* method for each instance variable. Provide a method `displayDate` that displays the month, day and year separated by forward slashes (/). Write a test application named `DateTest` that demonstrates class `Date`'s capabilities.

ANS:

```
// Exercise 3.15 Solution: Date.java
// Date class with instance variables for the month, day and year.

public class Date
{
   private int month;
   private int day;
   private int year;

   // constructor
   public Date( int monthValue, int dayValue, int yearValue )
   {
      month = monthValue;
      day = dayValue;
      year = yearValue;
   } // end three-argument constructor

   // set the month
   public void setMonth( int monthValue )
   {
      month = monthValue;
   } // end method setMonth

   // return the month
   public int getMonth()
   {
      return month;
   } // return month

   // set the day
   public void setDay( int dayValue )
   {
      day = dayValue;
   } // end method setDay

   // return the day
   public int getDay()
   {
      return day;
   } // return day

   // set the year
   public void setYear( int yearValue )
   {
      year = yearValue;
   } // end method setYear

   // return the year
   public int getYear()
```

```
50      {
51         return year;
52      } // return year
53
54      // display the date
55      public void displayDate()
56      {
57         System.out.printf( "%d/%d/%d", getMonth(), getDay(), getYear() );
58      } // end method displayDate
59   } // end class Date
```

```
1   // Exercise 3.15 Solution: DateTest.java
2   // Application to test class Date.
3
4   public class DateTest
5   {
6      public static void main( String args[] )
7      {
8         Date date1 = new Date( 7, 4, 2004 );
9
10        System.out.print( "The initial date is: " );
11        date1.displayDate();
12
13        // change date values
14        date1.setMonth( 11 );
15        date1.setDay( 1 );
16        date1.setYear( 2003 );
17
18        System.out.print( "\nDate with new values is: " );
19        date1.displayDate();
20
21        System.out.println(); // output a newline
22     } // end main
23  } // end class DateTest
```

```
The initial date is: 7/4/2004
Date with new values is: 11/1/2003
```