

3

Introduction to Classes and Objects

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 method to make that method perform its task.
- 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.

*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

Assignment Checklist

Name: _____ Date: _____

Section: _____

Exercises	Assigned: Circle assignments		Date Due
Prelab Activities			
Matching	YES	NO	
Fill in the Blank	YES	NO	
Short Answer	YES	NO	
Programming Output	YES	NO	
Correct the Code	YES	NO	
Lab Exercises			
Exercise 1 — Modifying Class Account	YES	NO	
Exercise 2 — Modifying Class GradeBook	YES	NO	
Exercise 3 — Creating an Employee Class	YES	NO	
Debugging	YES	NO	
Postlab Activities			
Coding Exercises	1, 2, 3, 4, 5, 6, 7, 8, 9		
Programming Challenges	1, 2		

Prelab Activities

Matching

Name: _____ Date: _____

Section: _____

After reading Chapter 3 of *Java How to Program: 8/e*, answer the given questions. The questions are intended to test and reinforce your understanding of key concepts. You may answer the questions either before or during the lab.

For each term in the left column, write the letter for the description from the right column that best matches the term.

Term	Description
<u>D</u> 1. field	a) Used in a class instance creation expression to create an instance of a class.
<u>O</u> 2. calling method	b) Primitive type that represents a single-precision floating-point number.
<u>F</u> 3. reference	c) Causes Java to execute a method.
<u>A</u> 4. new keyword	d) Also known as an instance variable.
<u>G</u> 5. public method	e) A method that assigns a value to a private instance variable.
<u>L</u> 6. class declaration	f) A variable that refers to an object contains one of these as its value.
<u>M</u> 7. fully qualified class name	g) A method that can be accessed outside of the class in which it is declared.
<u>C</u> 8. method call	h) Default initial value of a reference-type variable.
<u>I</u> 9. parameter	i) Additional information a method requires to help it perform its task.
<u>E</u> 10. set method	j) Primitive type that represents a double-precision floating-point number.
<u>K</u> 11. default constructor	k) The compiler provides one of these for a class that does not declare any.
<u>N</u> 12. client of an object or a class	l) Encompasses all of the attributes and behaviors of a class.
<u>J</u> 13. double	m) Can be used to access a class if the class is not imported.
<u>H</u> 14. null	n) A class that calls any of an object's or class's methods.
<u>B</u> 15. float	o) Receives the return value from a method.

Prelab Activities

Name: _____

Fill in the Blank

Name: _____ Date: _____

Section: _____

Fill in the blanks for each of the following statements:

16. Each method can specify parameters that represent additional information the method requires to perform its task correctly.
17. Declaring instance variables with access modifier private is a form of information hiding.
18. Instance variables of the numeric primitive types are initialized to 0 and instance variables of type `boolean` are initialized to false.
19. Variables declared in the body of a particular method are known as local variables and can be used only in that method.
20. An `import` declaration is not required if you always refer to a class with its fully qualified class name.
21. Each parameter must specify both a(n) type and a(n) name.
22. The format specifier `%f` is used to output values of type float or double.
23. Programs use variables of reference types to store the location of objects in the computer's memory.
24. A(n) class normally consists of one or more methods that manipulate the attributes that belong to a particular object.
25. Classes often provide `public` methods to allow clients of the class to set or get the values of `private` instance variables.

Prelab Activities

Name: _____

Short Answer

Name: _____ Date: _____

Section: _____

Answer the given questions in the spaces provided. Your answers should be concise; aim for two or three sentences.

26. List the parts of a method header and why each one is important.

A method header consists of an access modifier, a return type, a method name and a parameter list. The access modifier determines whether this method can be called by methods of other classes. The return type specifies what type of information, if any, is returned by this method to the calling method. The name of the method is used by clients of the class to invoke the method. The parameter list specifies additional information that the method requires to perform its task.

27. How are constructors and methods similar? How are they different?

Constructors and methods are similar in that they both can have access modifiers that specify whether they are accessible outside the class in which they are declared (though they are most frequently declared `public`). Constructors and methods can both receive parameters. Constructors and methods differ in that constructors must have the same name as the class in which they are declared and constructors cannot return values. Also, constructors are called automatically when a new object of a class is created. Methods must be invoked explicitly.

28. What is the relationship between a client of an object and the object's `public` members?

A client of an object can access all of the object's `public` members—typically the `public` methods of the object's class. The client of the object cannot access any of that object's `private` members. A client typically calls an object's `public` methods to manipulate the object's `private` data.

29. What types of declarations are contained within a class declaration?

A class declaration contains declarations of the class's fields, constructors and methods. The variables represent the attributes of the class. The methods comprise the behaviors of the class. A constructor is used to initialize a new object of the class. (As you will learn later, a class can have multiple constructors.)

30. Distinguish between a primitive-type variable and a reference-type variable.

A primitive-type variable stores a value of its declared type. The primitive types are `boolean`, `byte`, `char`, `short`, `int`, `long`, `float` and `double`. A reference-type variable stores a reference to one object in memory. That object may contain many pieces of data and typically also has methods for manipulating that data. The object's methods can be invoked by preceding the method name with the name of the variable that contains the object's reference and a dot (`.`) separator.

Prelab Activities

Name: _____

Programming Output

Name: _____ Date: _____

Section: _____

For each of the given program segments, read the code and write the output in the space provided below each program. [Note: Do not execute these programs on a computer.]

Use the following class definition for *Programming Output Exercises 31–35*.

```
1 public class Account
2 {
3     private double balance; // instance variable that stores the balance
4
5     // constructor
6     public Account( double initialBalance )
7     {
8         // validate that initialBalance is greater than 0.0;
9         // if it is not, balance is initialized to the default value 0.0
10        if ( initialBalance > 0.0 )
11            balance = initialBalance;
12    } // end Account constructor
13
14    // credit (add) an amount to the account
15    public void credit( double amount )
16    {
17        balance = balance + amount; // add amount to balance
18    } // end method credit
19
20    // return the account balance
21    public double getBalance()
22    {
23        return balance; // gives the value of balance to the calling method
24    } // end method getBalance
25 } // end class Account
```

31. What is output by the following main method?

```
1 public static void main( String args[] )
2 {
3     Account account1 = new Account( 35.50 );
4
5     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
6 } // end main
```

Your answer:

account1 balance: \$35.50

Prelab Activities

Name: _____

Programming Output

32. What is output by the following main method?

```
1 public static void main( String args[] )
2 {
3     Account account1 = new Account( -20.17 );
4
5     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
6 } // end main
```

Your answer:

```
account1 balance: $0.00
```

33. What is output by the following main method?

```
1 public static void main( String args[] )
2 {
3     Account account1 = new Account( 15.33 );
4
5     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
6     System.out.println( "adding $2.53 to account1 balance" );
7
8     account1.credit( 2.53 );
9     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
10 } // end main
```

Your answer:

```
account1 balance: $15.33
adding $2.53 to account1 balance
account1 balance: $17.86
```

34. What is output by the following main method?

```
1 public static void main( String args[] )
2 {
3     Account account1 = new Account( 7.99 );
4
5     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
6     System.out.println( "adding -$1.14 to account1 balance" );
7
8     account1.credit( -1.14 );
9     System.out.printf( "account1 balance: %.2f\n", account1.getBalance() );
10 } // end main
```

Prelab Activities

Name: _____

Programming Output

Your answer:

```
account1 balance: $7.99  
adding -$1.14 to account1 balance  
account1 balance: $6.85
```


Prelab Activities

Name: _____

Correct the Code

Name: _____ Date: _____

Section: _____

Determine if there is an error in each of the following program segments. If there is an error, specify whether it is a logic error or a compilation error, circle the error in the program and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [Note: There may be more than one error in each program segment.]

Use the following class definitions for *Correct the Code Exercises 36–39*.

```

1  // Lab 2: GradeBook.java
2  // GradeBook class with a constructor to initialize the course name.
3
4  public class GradeBook
5  {
6      private String courseName; // course name for this GradeBook
7
8      // constructor initializes courseName with String supplied as argument
9      public GradeBook( String name )
10     {
11         courseName = name; // initializes courseName
12     } // end constructor
13
14     // method to set the course name
15     public void setCourseName( String name )
16     {
17         courseName = name; // store the course name
18     } // end method setCourseName
19
20     // method to retrieve the course name
21     public String getCourseName()
22     {
23         return courseName;
24     } // end method getCourseName
25
26     // display a welcome message to the GradeBook user
27     public void displayMessage()
28     {
29         // this statement calls getCourseName to get the
30         // name of the course this GradeBook represents
31         System.out.printf( "Welcome to the grade book for\n%s!\n",
32             getCourseName() );
33     } // end method displayMessage
34 } // end class GradeBook

```

35. The following code segment should create a new GradeBook object:

```
1  GradeBook gradeBook = Gradebook( "Introduction to Java", 25 );
```

Prelab Activities

Name: _____

Correct the Code

Your answer:

```
1 GradeBook gradeBook = new GradeBook( "Introduction to Java" );
```

- The new keyword is required to create an object and invoke its constructor.
- The GradeBook constructor accepts only one parameter of type String.
- Java is case sensitive. The GradeBook class has an uppercase B. Using a lowercase b is a compilation error.

36. The following code segment should set the GradeBook's course name:

```
1 setCourseName( gradeBook, "Advanced Java" )
```

Your answer:

```
1 gradeBook.setCourseName( "Advanced Java" );
```

- A method is invoked on an object starting with the name of the variable that contains the object's reference followed by the dot separator followed by the name of the method and its arguments in parentheses. The reference is not passed to the method as a parameter.
- Forgetting the semicolon at the end of a statement is a syntax error.

37. The following code segment should ask the user to input a course name. That should then be set as the course name of your gradeBook.

```
1 Scanner input = new Scanner( System.in );
2
3 System.out.println( "Please enter the course name:" );
4 inputName = Scanner.readLine();
5
6 gradeBook.setCourseName();
```

Your answer:

```
1 Scanner input = new Scanner( System.in );
2
3 System.out.println( "Please enter the course name:" );
4 String inputName = input.nextLine();
5
6 gradeBook.setCourseName( inputName );
```

- Variables must be declared before they are used. If inputName has not been declared previously in the application, it must be declared as a String here.
- The Scanner class does not declare a readLine method. The method for entering a line of text is called nextLine.

Prelab Activities

Name: _____

Correct the Code

- The input method `nextLine` must be invoked on a `Scanner` object. The variable `input` should be used to invoke `nextLine`.
- The `setCourseName` method requires a `String` parameter. Passing the method the variable `inputName` enables the method to set the name of the course to the input received from the user.

38. The following code segment should output the grade book's current course name:

```
1 System.out.printf( "The grade book's course name is: \n", gradeBook.courseName );
```

Your answer:

```
1 System.out.printf( "The grade book's course name is: %s\n",  
2     gradeBook.getCourseName() );
```

- There is no format specifier in the format string to act as a placeholder for the course name. The `%s` format specifier is used to output a `String` with the `printf` method.
- Variable `courseName` is a private variable of class `GradeBook` and cannot be accessed directly from outside the class. Method `getCourseName` must be used to obtain the course name.

Lab Exercises

Lab Exercise I — Modifying Class Account

Name: _____ Date: _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into five parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 3.1 and Fig. L 3.2)
5. Problem-Solving Tips

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. The source code for the template is available at www.pearsonhighered.com/deitel.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 3 of *Java How to Program: 8/e*. In this lab, you will practice:

- Creating methods.
- Invoking methods and receiving return values from methods.
- Testing a condition using an `if` statement.
- Outputting variables with a `printf` statement.

Description of the Problem

Modify class `Account` (Fig. L 3.1) to provide a method called `debit` that withdraws money from an `Account`. Ensure that the debit amount does not exceed the `Account`'s balance. If it does, the balance should be left unchanged and the method should print a message indicating "Debit amount exceeded account balance." Modify class `AccountTest` (Fig. L 3.2) to test method `debit`.

Sample Output

```
account1 balance: $50.00
account2 balance: $0.00

Enter withdrawal amount for account1: 25.67

subtracting 25.67 from account1 balance
account1 balance: $24.33
account2 balance: $0.00

Enter withdrawal amount for account2: 10.00

subtracting 10.00 from account2 balance
Debit amount exceeded account balance.
account1 balance: $24.33
account2 balance: $0.00
```

Lab Exercises

Name: _____

Lab Exercise I — Modifying Class Account

Program Template

```

1 // Lab 1: Account.java
2 // Account class with a constructor to
3 // initialize instance variable balance.
4
5 public class Account
6 {
7     private double balance; // instance variable that stores the balance
8
9     // constructor
10    public Account( double initialBalance )
11    {
12        // validate that initialBalance is greater than 0.0;
13        // if it is not, balance is initialized to the default value 0.0
14        if ( initialBalance > 0.0 )
15            balance = initialBalance;
16    } // end Account constructor
17
18    // credit (add) an amount to the account
19    public void credit( double amount )
20    {
21        balance = balance + amount; // add amount to balance
22    } // end method credit
23
24    /* write code to declare method debit. */
25
26    // return the account balance
27    public double getBalance()
28    {
29        return balance; // gives the value of balance to the calling method
30    } // end method getBalance
31
32 } // end class Account

```

Fig. L3.1 | Account.java.

```

1 // Lab 1: AccountTest.java
2 // Create and manipulate an Account object.
3 import java.util.Scanner;
4
5 public class AccountTest
6 {
7     // main method begins execution of Java application
8     public static void main( String args[] )
9     {
10        Account account1 = new Account( 50.00 ); // create Account object
11        Account account2 = new Account( -7.53 ); // create Account object
12
13        // display initial balance of each object
14        System.out.printf( "account1 balance: %.2f\n",
15                           account1.getBalance() );
16        System.out.printf( "account2 balance: %.2f\n\n",
17                           account2.getBalance() );
18    }

```

Fig. L3.2 | AccountTest.java. (Part I of 2.)

Lab Exercises

Name: _____

Lab Exercise I — Modifying Class Account

```
19 // create Scanner to obtain input from command window
20 Scanner input = new Scanner( System.in );
21 double withdrawalAmount; // withdrawal amount read from user
22
23 System.out.print( "Enter withdrawal amount for account1: " );
24 withdrawalAmount = input.nextDouble(); // obtain user input
25 System.out.printf( "\nsubtracting %.2f from account1 balance\n",
26     withdrawalAmount );
27 /* write code to withdraw money from account */
28
29 // display balances
30 System.out.printf( "account1 balance: $%.2f\n",
31     account1.getBalance() );
32 System.out.printf( "account2 balance: $%.2f\n\n",
33     account2.getBalance() );
34
35 System.out.print( "Enter withdrawal amount for account2: " );
36 withdrawalAmount = input.nextDouble(); // obtain user input
37 System.out.printf( "\nsubtracting %.2f from account2 balance\n",
38     withdrawalAmount );
39 /* write code to withdraw from account */
40
41 // display balances
42 System.out.printf( "account1 balance: $%.2f\n",
43     account1.getBalance() );
44 System.out.printf( "account2 balance: $%.2f\n",
45     account2.getBalance() );
46 } // end main
47
48 } // end class AccountTest
```

Fig. L 3.2 | AccountTest.java. (Part 2 of 2.)**Problem-Solving Tips**

1. Declare public method debit with a return type of void.
2. Use a parameter to enable the program to specify the amount the user wishes to withdraw.
3. In the body of method debit, use an if statement to test whether the withdrawal amount is more than the balance. Output an appropriate message if the condition is true.
4. Use another if statement to test whether the withdrawal amount is less than or equal to the balance. Decrement the balance appropriately.
5. If you have any questions as you proceed, ask your lab instructor for help.

Lab Exercises

Name: _____

Lab Exercise I — Modifying Class Account

Solution

```
1 // Lab 1: Account.java
2 // Account class with a constructor to
3 // initialize instance variable balance.
4
5 public class Account
6 {
7     private double balance; // instance variable that stores the balance
8
9     // constructor
10    public Account( double initialBalance )
11    {
12        // if initialBalance is not greater than 0.0,
13        // balance is still initialized to 0.0 by default
14        if ( initialBalance > 0.0 )
15            balance = initialBalance;
16    } // end Account constructor
17
18    // credits (adds) an amount to the account
19    public void credit( double amount )
20    {
21        balance = balance + amount; // add amount to balance
22    } // end method credit
23
24    // debits (subtracts) an amount from the account
25    public void debit( double amount )
26    {
27        if ( amount > balance )
28            System.out.println( "Debit amount exceeded account balance." );
29
30        if ( amount <= balance )
31            balance = balance - amount; // subtract amount to balance
32    } // end method debit
33
34    // returns the account balance
35    public double getBalance()
36    {
37        return balance; // gives the value of balance to the calling method
38    } // end method getBalance
39
40 } // end class Account
```

```
1 // Lab 1: AccountTest.java
2 // Create and manipulate an Account object.
3 import java.util.Scanner;
4
5 public class AccountTest
6 {
7     // main method begins execution of Java application
8     public static void main( String args[] )
9     {
10        Account account1 = new Account( 50.00 ); // create Account object
11        Account account2 = new Account( -7.53 ); v
12    }
```

Lab Exercises

Name: _____

Lab Exercise I — Modifying Class Account

```
13 // display initial balance of each object
14 System.out.printf( "account1 balance: $%.2f\n",
15     account1.getBalance() );
16 System.out.printf( "account2 balance: $%.2f\n\n",
17     account2.getBalance() );
18
19 // create Scanner to obtain input from command window
20 Scanner input = new Scanner( System.in );
21 double withdrawalAmount; // withdrawal amount read from user
22
23 System.out.print( "Enter withdrawal amount for account1: " );
24 withdrawalAmount = input.nextDouble(); // obtain user input
25 System.out.printf( "\nsubtracting $%.2f from account1 balance\n",
26     withdrawalAmount );
27 account1.debit( withdrawalAmount ); // subtract amount from account1
28
29 // display balances
30 System.out.printf( "account1 balance: $%.2f\n",
31     account1.getBalance() );
32 System.out.printf( "account2 balance: $%.2f\n\n",
33     account2.getBalance() );
34
35 System.out.print( "Enter withdrawal amount for account2: " );
36 withdrawalAmount = input.nextDouble(); // obtain user input
37 System.out.printf( "\nsubtracting $%.2f from account2 balance\n",
38     withdrawalAmount );
39 account2.debit( withdrawalAmount ); // subtract amount from account2
40
41 // display balances
42 System.out.printf( "account1 balance: $%.2f\n",
43     account1.getBalance() );
44 System.out.printf( "account2 balance: $%.2f\n",
45     account2.getBalance() );
46 } // end main
47
48 } // end class AccountTest
```


Lab Exercises

Name: _____

Lab Exercise 2 — Modifying Class GradeBook

Name: _____ Date: _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into five parts:

1. Lab Objectives
2. Problem of the Description
3. Sample Output
4. Program Template (Fig. L 3.3 and Fig. L 3.4)
5. Problem-Solving Tips

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description, and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. The source code for the template is available at www.pearsonhighered.com/deitel.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 3 of *Java How to Program: 8/e*. In this lab, you will practice:

- Declaring an instance variable.
- Providing a *set* method to modify an instance variable's value.
- Declaring methods with parameters.

Description of the Problem

Modify class `GradeBook` (Fig. L 3.3). Include a second `String` instance variable that represents the name of the course's instructor. Provide a *set* method to change the instructor's name and a *get* method to retrieve it. Modify the constructor to specify two parameters—one for the course name and one for the instructor's name. Modify method `displayMessage` such that it first outputs the welcome message and course name, then outputs "This course is presented by: " followed by the instructor's name. Modify the test application (Fig. L 3.4) to demonstrate the class's new capabilities.

Sample Output

```
Welcome to the grade book for
CS101 Introduction to Java Programming!
This course is presented by: Sam Smith

Changing instructor name to Judy Jones

Welcome to the grade book for
CS101 Introduction to Java Programming!
This course is presented by: Judy Jones
```

Lab Exercises

Name: _____

Lab Exercise 2 — Modifying Class GradeBook

Program Template

```

1  // Lab 2: GradeBook.java
2  // GradeBook class with a constructor to initialize the course name.
3
4  public class GradeBook
5  {
6      private String courseName; // course name for this GradeBook
7      /* write code to declare a second String instance variable */
8
9      // constructor initializes courseName with String supplied as argument
10     public GradeBook( String name )
11     {
12         courseName = name; // initializes courseName
13     } // end constructor
14
15     // method to set the course name
16     public void setCourseName( String name )
17     {
18         courseName = name; // store the course name
19     } // end method setCourseName
20
21     // method to retrieve the course name
22     public String getCourseName()
23     {
24         return courseName;
25     } // end method getCourseName
26
27     /* write code to declare a get and a set method for the instructor's name */
28
29     // display a welcome message to the GradeBook user
30     public void displayMessage()
31     {
32         // this statement calls getCourseName to get the
33         // name of the course this GradeBook represents
34         System.out.printf( "Welcome to the grade book for\n%s!\n",
35             getCourseName() );
36         /* write code to output the instructor's name */
37     } // end method displayMessage
38
39 } // end class GradeBook

```

Fig. L 3.3 | GradeBook.java.

```

1  // Lab 2: GradeBookTest.java
2  // GradeBook constructor used to specify the course name at the
3  // time each GradeBook object is created.
4
5  public class GradeBookTest
6  {
7      // main method begins program execution
8      public static void main( String args[] )
9      {
10         // create GradeBook object
11         GradeBook gradeBook1 = new GradeBook(
12             "CS101 Introduction to Java Programming" );

```

Fig. L 3.4 | GradeBookTest.java. (Part 1 of 2.)

Lab Exercises

Name: _____

Lab Exercise 2 — Modifying Class GradeBook

```
13
14     gradeBook1.displayMessage(); // display welcome message
15
16     /* write code to change instructor's name and output changes */
17
18     } // end main
19
20 } // end class GradeBookTest
```

Fig. L 3.4 | GradeBookTest.java. (Part 2 of 2.)

Problem-Solving Tips

1. In class GradeBook, declare a `String` instance variable to represent the instructor's name.
2. Declare a `public set` method for the instructor's name that does not return a value and takes a `String` as a parameter. In the body of the `set` method, assign the parameter's value to the variable that represents the instructor's name.
3. Declare a `public get` method that returns a `String` and takes no parameters. This method should return the instructor's name.
4. Modify the constructor to take two `String` parameters. Assign the parameter that represents the instructor's name to the appropriate instance variable.
5. Add a `System.out.printf` statement to method `displayMessage` to output the value of the instance variable you declared earlier.
6. If you have any questions as you proceed, ask your lab instructor for help.

Lab Exercises

Name: _____

Lab Exercise 2 — Modifying Class GradeBook

Solution

```
1 // Lab 2: GradeBook.java
2 // GradeBook class with a constructor to initialize the course name.
3
4 public class GradeBook
5 {
6     private String courseName; // course name for this GradeBook
7     private String instructorName; // name of course's instructor
8
9     // constructor initializes courseName with String supplied as argument
10    public GradeBook( String course, String instructor )
11    {
12        courseName = course; // initializes courseName
13        instructorName = instructor; // initializes instructorName
14    } // end constructor
15
16    // method to set the course name
17    public void setCourseName( String name )
18    {
19        courseName = name; // store the course name
20    } // end method setCourseName
21
22    // method to retrieve the course name
23    public String getCourseName()
24    {
25        return courseName;
26    } // end method getCourseName
27
28    // method to set the instructor name
29    public void setInstructorName( String name )
30    {
31        instructorName = name; // store the course name
32    } // end method setInstructorName
33
34    // method to retrieve the instructor name
35    public String getInstructorName()
36    {
37        return instructorName;
38    } // end method getInstructorName
39
40    // display a welcome message to the GradeBook user
41    public void displayMessage()
42    {
43        // this statement calls getCourseName to get the
44        // name of the course this GradeBook represents
45        System.out.printf( "Welcome to the grade book for\n%s!\n",
46            getCourseName() );
47        System.out.printf( "This course is presented by: %s\n",
48            getInstructorName() );
49    } // end method displayMessage
50
51 } // end class GradeBook
```

Lab Exercises

Name: _____

Lab Exercise 2 — Modifying Class GradeBook

```
1 // Lab 2: GradeBookTest.java
2 // GradeBook constructor used to specify the course name at the
3 // time each GradeBook object is created.
4
5 public class GradeBookTest
6 {
7     // main method begins program execution
8     public static void main( String args[] )
9     {
10        // create GradeBook object
11        GradeBook gradeBook1 = new GradeBook(
12            "CS101 Introduction to Java Programming", "Sam Smith" );
13
14        gradeBook1.displayMessage(); // display welcome message
15
16        System.out.println( "\nChanging instructor name to Judy Jones\n" );
17        gradeBook1.setInstructorName( "Judy Jones" );
18
19        gradeBook1.displayMessage(); // display welcome message
20    } // end main
21
22 } // end class GradeBookTest
```


Lab Exercises

Lab Exercise 3 — Creating an Employee Class

Name: _____ Date: _____

Section: _____

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into five parts:

1. Lab Objectives
2. Description of the Problem
3. Sample Output
4. Program Template (Fig. L 3.5 and Fig. L 3.6)
5. Problem-Solving Tips

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/** */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. The source code for the template is available at www.pearsonhighered.com/deitel.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 3 of *Java How to Program: 8/e*. In this lab, you will practice:

- Creating a class declaration.
- Declaring instance variables.
- Declaring a constructor.
- Declaring *set* and *get* methods.
- Writing a test application to demonstrate the capabilities of another class.

Description of the Problem

Using only programming techniques from this chapter and Chapter 2 of *Java How to Program: 8/e*, create a class called `Employee` that includes three pieces of information as instance variables—a first name (type `String`), a last name (type `String`) and a monthly salary (type `double`). Your class should have a constructor that initializes the three instance variables. Provide a *set* and a *get* method for each instance variable. If the monthly salary is not positive, set it to 0.0. Write a test application named `EmployeeTest` that demonstrates class `Employee`'s capabilities. Create two `Employee` objects and display the yearly salary for each `Employee`. Then give each `Employee` a 10% raise and display each `Employee`'s yearly salary again.

Sample Output

```
Employee 1: Bob Jones; Yearly Salary: 34500.00
Employee 2: Susan Baker; Yearly Salary: 37809.00

Increasing employee salaries by 10%
Employee 1: Bob Jones; Yearly Salary: 37950.00
Employee 2: Susan Baker; Yearly Salary: 41589.90
```

Lab Exercises

Name: _____

Lab Exercise 3 — Creating an Employee Class

Program Template

```
1 // Lab 3: Employee.java
2 // Employee class.
3
4 /* Begin class declaration of Employee class. */
5
6     /* Declare three instance variables here. */
7
8     /* Add a constructor that declares a parameter for each instance variable. Assign
9        each parameter value to the appropriate instance variable. Write code that
10       validates the value of salary to ensure that it is not negative. */
11
12     /* Declare set and get methods for the first name instance variable. */
13
14     /* Declare set and get methods for the last name instance variable. */
15
16     /* Declare set and get methods for the monthly salary instance variable. Write code
17        that validates the salary to ensure that it is not negative. */
18
19 /* End class declaration of Employee class. */
```

Fig. L 3.5 | Employee.java.

```
1 // Lab 3: EmployeeTest.java
2 // Application to test class Employee.
3
4 /* Begin class declaration of EmployeeTest class. */
5
6     /* Begin main method declaration. */
7
8     /* Create two Employee objects and assign them to Employee variables. */
9
10    /* Output the first name, last name and salary for each Employee. */
11
12    /* Give each Employee a 10% raise. */
13
14    /* Output the first name, last name and salary of each Employee again. */
15
16    /* End main method declaration */
17
18 /* End class declaration of EmployeeTest class. */
```

Fig. L 3.6 | EmployeeTest.java.

Problem-Solving Tips

1. Class `Employee` should declare three instance variables.
2. The constructor must declare three parameters, one for each instance variable. The value for the salary should be validated to ensure it is not negative.
3. Declare a `public set` and `get` method for each instance variable. The `set` methods should not return values and should each specify a parameter of a type that matches the corresponding instance variable (`String` for first name and last name, `double` for the salary). The `get` methods should receive no parameters and should specify a return type that matches the corresponding instance variable.

Lab Exercises

Name: _____

Lab Exercise 3 — Creating an Employee Class

4. When you call the constructor from the test class, you must pass it three arguments that match the parameters declared by the constructor.
5. Giving each employee a raise will require a call to the *get* method for the salary to obtain the current salary and a call to the *set* method for the salary to specify the new salary.
6. A salary is a dollar amount, so you should output the salary using the `%.2f` specifier to provide two digits of precision.
7. If you have any questions as you proceed, ask your lab instructor for help.

Solution

```
1  // Lab 3: Employee.java
2  // Employee class.
3
4  public class Employee
5  {
6      private String firstName;
7      private String lastName;
8      private double monthlySalary;
9
10     // constructor to initialize first name, last name and monthly salary
11     public Employee( String first, String last, double salary )
12     {
13         firstName = first;
14         lastName = last;
15
16         if ( salary >= 0.0 ) // determine whether salary is positive
17             monthlySalary = salary;
18     } // end three-argument Employee constructor
19
20     // set Employee's first name
21     public void setFirstName( String first )
22     {
23         firstName = first;
24     } // end method setFirstName
25
26     // get Employee's first name
27     public String getFirstName()
28     {
29         return firstName;
30     } // end method getFirstName
31
32     // set Employee's last name
33     public void setLastName( String last )
34     {
35         lastName = last;
36     } // end method setLastName
37
38     // get Employee's last name
39     public String getLastName()
40     {
41         return lastName;
42     } // end method getLastName
43
```

Lab Exercises

Name: _____

Lab Exercise 3 — Creating an Employee Class

```

44 // set Employee's monthly salary
45 public void setMonthlySalary( double salary )
46 {
47     if ( salary >= 0.0 ) // determine whether salary is positive
48         monthlySalary = salary;
49 } // end method setMonthlySalary
50
51 // get Employee's monthly salary
52 public double getMonthlySalary()
53 {
54     return monthlySalary;
55 } // end method getMonthlySalary
56
57 } // end class Employee

```



```

1 // Lab 3: EmployeeTest.java
2 // Application to test class Employee.
3
4 public class EmployeeTest
5 {
6     public static void main( String args[] )
7     {
8         Employee employee1 = new Employee( "Bob", "Jones", 2875.00 );
9         Employee employee2 = new Employee( "Susan", "Baker", 3150.75 );
10
11         // display employees
12         System.out.printf( "Employee 1: %s %s; Yearly Salary: %.2f\n",
13             employee1.getFirstName(), employee1.getLastName(),
14             12 * employee1.getMonthlySalary() );
15         System.out.printf( "Employee 2: %s %s; Yearly Salary: %.2f\n",
16             employee2.getFirstName(), employee2.getLastName(),
17             12 * employee2.getMonthlySalary() );
18
19         // increase employee salaries by 10%
20         System.out.println( "\nIncreasing employee salaries by 10%" );
21         employee1.setMonthlySalary( employee1.getMonthlySalary() * 1.10 );
22         employee2.setMonthlySalary( employee2.getMonthlySalary() * 1.10 );
23
24         // display employees with new yearly salary
25         System.out.printf( "Employee 1: %s %s; Yearly Salary: %.2f\n",
26             employee1.getFirstName(), employee1.getLastName(),
27             12 * employee1.getMonthlySalary() );
28         System.out.printf( "Employee 2: %s %s; Yearly Salary: %.2f\n",
29             employee2.getFirstName(), employee2.getLastName(),
30             12 * employee2.getMonthlySalary() );
31     } // end main
32
33 } // end class EmployeeTest

```

Lab Exercises

Name: _____

Debugging

Name: _____ Date: _____

Section: _____

The program in this section does not compile. Fix all the compilation errors so that the program will compile successfully. Once the program compiles, execute the program, and compare its output with the sample output; then eliminate any logic errors that may exist. The sample output demonstrates what the program's output should be once the program's code is corrected. The source code is available at the Web sites www.pearsonhighered.com/deitel.

Sample Output

```
Created John Smith, age 19
Happy Birthday to John Smith
```

Broken Code

```
1  // Person.java
2  // Creates and manipulates a person with a first name, last name and age
3
4  public class Person
5  {
6      private String firstName;
7      private String lastName;
8      private int age;
9
10     public void Person( String first, String last, int years )
11     {
12         firstName = first;
13         lastName = last;
14
15         if ( years < 0 )
16             age = years;
17     } // end Person constructor
18
19     public String getFirstName( String FirstName )
20     {
21         return firstName;
22     } // end method getFirstName
23
24     public setFirstName( String first )
25     {
26         firstName = first;
27     } // end method setFirstName
28
29     public String getLastName()
30     {
31         return;
32     } // end method getLastName
```

Fig. L 3.7 | Person.java. (Part I of 2.)

Lab Exercises

Name: _____

Debugging

```

33
34     public void setLastName( String last )
35     {
36         lastName = last;
37     } // end method setLastName
38
39     public int getAge()
40     {
41         return years;
42     } // end method getAge
43
44     public void setAge( int years )
45     {
46         if ( years > 0 )
47             age = years;
48     } // end method setAge
49 } // end class Person

```

Fig. L 3.7 | Person.java. (Part 2 of 2.)

```

1 // PersonTest.java
2 // Test application for the Person class
3
4 public class PersonTest
5 {
6     public static void main( String args[] )
7     {
8         Person person = Person( "John", "Smith", 19 );
9
10        System.out.printf( "Created %s %s, age %d\n",
11                           getFirstName(), getLastName(), getAge() );
12
13        person.setAge = person.getAge() + 1;
14        System.out.printf( "Happy Birthday to %s %s\n",
15                           person.getFirstName(), person.getLastName() );
16    } // end main
17 } // end class PersonTest

```

Fig. L 3.8 | PersonTest.java.

Debugging Exercise Solution

```

1 // Person.java
2 // Creates and manipulates a person with a first name, last name and age
3
4 public class Person
5 {
6     private String firstName;
7     private String lastName;
8     private int age;
9
10    public Person( String first, String last, int years )
11    {
12        firstName = first;
13        lastName = last;

```

Lab Exercises

Name: _____

Debugging

```
14
15     if ( years > 0 )
16         age = years;
17 } // end Person constructor
18
19 public String getFirstName()
20 {
21     return firstName;
22 } // end method getFirstName
23
24 public void setFirstName( String first )
25 {
26     firstName = first;
27 } // end method setFirstName
28
29 public String getLastName()
30 {
31     return lastName;
32 } // end method getLastName
33
34 public void setLastName( String last )
35 {
36     lastName = last;
37 } // end method setLastName
38
39 public int getAge()
40 {
41     return age;
42 } // end method getAge
43
44 public void setAge( int years )
45 {
46     if ( years > 0 )
47         age = years;
48 } // end method setAge
49 } // end class Person
```

```
1 // PersonTest.java
2 // Test application for the Person class
3
4 public class PersonTest
5 {
6     public static void main( String args[] )
7     {
8         Person person = new Person( "John", "Smith", 19 );
9
10        System.out.printf( "Created %s %s, age %d\n",
11            person.getFirstName(), person.getLastName(), person.getAge() );
12
13        person.setAge( person.getAge() + 1 );
14        System.out.printf( "Happy Birthday to %s %s\n",
15            person.getFirstName(), person.getLastName() );
16    } // end main
17 } // end class PersonTest
```

Lab Exercises

Name: _____

Debugging**Bugs in class Person**

- Line 10: A Constructor should not be declared with a return type. Otherwise, Java assumes it is a regular method of the class that must be called explicitly.
- Line 15: Logic error. A person's age should not be negative. The age should only be set if the value of years is greater than 0.
- Line 19: The `getFirstName` method should not declare a parameter.
- Line 24: All methods must be declared with a return type. The `setFirstName` method should specify return type `void`.
- Line 31: The `getLastName` method should return the value of the `lastName` instance variable.
- Line 41: The variable `years` is not a field or a local variable declared within this method, so `years` does not exist. The method should return the value of instance variable `age`.

Bugs in class PersonTest

- Line 8: The keyword `new` is missing before the constructor name.
- Line 11: Each of the method calls is missing `"person."` before the method name.
- Line 13: `setAge` is a method and must be invoked as a method. You cannot simply assign a value to `setAge`.

Postlab Activities

Coding Exercises

Name: _____ Date: _____

Section: _____

These coding exercises reinforce the lessons learned in the lab and provide additional programming experience outside the classroom and laboratory environment. They serve as a review after you have successfully completed the *Prelab Activities* and *Lab Exercises*.

For each of the following problems, write a program or a program segment that performs the specified action.

1. Write an empty class declaration for a class named Student.

```
1 public class Student
2 {
3 } // end class Student
```

2. Declare five instance variables in the class from *Coding Exercise 1*: A String variable for the first name, a String variable for the last name and three double variables that are used to store a student's exam grades.

```
1 public class Student
2 {
3     private String firstName;
4     private String lastName;
5     private double exam1;
6     private double exam2;
7     private double exam3;
8 } // end class Student
```

3. In the class from *Coding Exercise 2*, declare a constructor that takes five parameters—two Strings and three doubles. Use these parameters to initialize the instance variables declared earlier.

```
1 public class Student
2 {
3     private String firstName;
4     private String lastName;
5     private double exam1;
6     private double exam2;
7     private double exam3;
8
9     public Student( String first, String last, double firstExam, double secondExam,
10         double thirdExam )
11     {
12         firstName = first;
13         lastName = last;
14         exam1 = firstExam;
15         exam2 = secondExam;
16         exam3 = thirdExam;
17     } // end Student constructor
18 } // end class Student
```

Postlab Activities

Name: _____

Coding Exercises

4. Modify the class from *Coding Exercise 3* to include a *get* and a *set* method for each of the instance variables in the class.

```
1 public class Student
2 {
3     private String firstName;
4     private String lastName;
5     private double exam1;
6     private double exam2;
7     private double exam3;
8
9     public Student( String first, String last, double firstExam, double secondExam,
10         double thirdExam )
11     {
12         firstName = first;
13         lastName = last;
14         exam1 = firstExam;
15         exam2 = secondExam;
16         exam3 = thirdExam;
17     } // end Student constructor
18
19     public String getFirstName()
20     {
21         return firstName;
22     } // end method getFirstName
23
24     public void setFirstName( String first )
25     {
26         firstName = first;
27     } // end method setFirstName
28
29     public String getLastName()
30     {
31         return lastName;
32     } // end method getLastName
33
34     public void setLastName( String last )
35     {
36         lastName = last;
37     } // end method setLastName
38
39     public double getExam1()
40     {
41         return exam1;
42     } // end method getExam1
43
44     public void setExam1( double firstExam )
45     {
46         exam1 = firstExam;
47     } // end method setExam1
48
49     public double getExam2()
50     {
51         return exam2;
52     } // end method getExam2
53
54     public void setExam2( double secondExam )
55     {
```

Postlab Activities

Name: _____

Coding Exercises

```
56     exam2 = secondExam;
57 } // end method setExam2
58
59 public double getExam3()
60 {
61     return exam3;
62 } // end method getExam3
63
64 public void setExam3( double thirdExam )
65 {
66     exam3 = thirdExam;
67 } // end method setExam3
68 } // end class Student
```

5. Modify the class from *Coding Exercise 4* to include a `getAverage` method that calculates and returns the average of the three exam grades.

```
1  public class Student
2  {
3      private String firstName;
4      private String lastName;
5      private double exam1;
6      private double exam2;
7      private double exam3;
8
9      public Student( String first, String last, double firstExam, double secondExam,
10         double thirdExam )
11      {
12          firstName = first;
13          lastName = last;
14          exam1 = firstExam;
15          exam2 = secondExam;
16          exam3 = thirdExam;
17      } // end Student constructor
18
19      public String getFirstName()
20      {
21          return firstName;
22      } // end method getFirstName
23
24      public void setFirstName( String first )
25      {
26          firstName = first;
27      } // end method setFirstName
28
29      public String getLastName()
30      {
31          return lastName;
32      } // end method getLastName
33
34      public void setLastName( String last )
35      {
36          lastName = last;
37      } // end method setLastName
38
```

Postlab Activities

Name: _____

Coding Exercises

```

39 public double getExam1()
40 {
41     return exam1;
42 } // end method getExam1
43
44 public void setExam1( double firstExam )
45 {
46     exam1 = firstExam;
47 } // end method setExam1
48
49 public double getExam2()
50 {
51     return exam2;
52 } // end method getExam2
53
54 public void setExam2( double secondExam )
55 {
56     exam2 = secondExam;
57 } // end method setExam2
58
59 public double getExam3()
60 {
61     return exam3;
62 } // end method getExam3
63
64 public void setExam3( double thirdExam )
65 {
66     exam3 = thirdExam;
67 } // end method setExam3
68
69 public double getAverage()
70 {
71     return ( exam1 + exam2 + exam3 ) / 3;
72 } // end method getAverage
73 } // end class Student

```

6. Declare an empty test class to use the capabilities of your new Student class from *Coding Exercise 5*.

```

1 public class StudentTest
2 {
3 } // end class StudentTest

```

7. In the class from *Coding Exercise 6*, declare a main method that creates an instance of class Student.

```

1 public class StudentTest
2 {
3     public static void main( String args[] )
4     {
5         Student pupil = new Student( "Susan", "White", 83, 95, 90 );
6     } // end main
7 } // end class StudentTest

```

Postlab Activities

Name: _____

Coding Exercises

8. Add statements to the main method of *Coding Exercise 7* to test class `Student`'s *get* methods. Output the name and average for the student.

```
1 public class StudentTest
2 {
3     public static void main( String args[] )
4     {
5         Student pupil = new Student( "Susan", "White", 83, 95, 90 );
6
7         System.out.printf( "Hello %s %s. Your average grade is %.1f.\n",
8                             pupil.getFirstName(), pupil.getLastName(), pupil.getAverage() );
9     } // end main
10 } // end class StudentTest
```

9. Add statements to the main method of *Coding Exercise 8* that test the *set* methods of class `Student`, then output the new name and average of the `Student` object to show that the *set* methods worked correctly.

```
1 public class StudentTest
2 {
3     public static void main( String args[] )
4     {
5         Student pupil = new Student( "Susan", "White", 83, 95, 90 );
6
7         System.out.printf( "Hello %s %s. Your average grade is %.1f.\n",
8                             pupil.getFirstName(), pupil.getLastName(), pupil.getAverage() );
9
10        pupil.setExam1( 93 );
11        pupil.setExam2( 91 );
12        pupil.setExam3( 86 );
13
14        System.out.printf( "%s %s, your new grade is %.1f.\n",
15                            pupil.getFirstName(), pupil.getLastName(), pupil.getAverage() );
16    } // end main
17 } // end class StudentTest
```


Postlab Activities

Name: _____

Programming Challenges

Name: _____ Date: _____

Section: _____

The *Programming Challenges* are more involved than the *Coding Exercises* and may require a significant amount of time to complete. Write a Java program for each of the problems in this section. The answers to these problems are available at www.pearsonhighered.com/deitel. Pseudocode, hints or sample outputs are provided for each problem to aid you in your programming.

1. 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 (type `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.

Hints:

- To solve this exercise, mimic your solutions to *Lab Exercises 1–3*.
- The input values for the quantity and the price per item must be validated before they can be used to set the corresponding instance variables. This should be done both in the constructor and in the appropriate *set* methods.
- The method header for `getInvoiceAmount` should be `public double getInvoiceAmount()`.

Postlab Activities

Name: _____

Programming Challenges

- Your output should appear as follows:

```
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
```

Solution

```
1 // Programming Challenge 1: 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
```

Postlab Activities

Name: _____

Programming Challenges

```
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
```

Postlab Activities

Name: _____

Programming Challenges

```
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 // Programming Challenge 1: 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 );
```

Postlab Activities

Name: _____

Programming Challenges

```
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
```

Postlab Activities

Name: _____

Programming Challenges

2. 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.

Hints:

- To solve this exercise, mimic your solutions to *Lab Exercises 1–3*.
- For the purpose of this chapter, it is not necessary to validate the values passed to the constructor or the *set* methods.
- Your output should appear as follows:

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

Solution

```
1  // Programming Challenge 2: Date.java
2  // Date class with instance variables for the month, day and year.
3
4  public class Date
5  {
6      private int month;
7      private int day;
8      private int year;
9
10     // constructor
11     public Date( int monthValue, int dayValue, int yearValue )
12     {
13         month = monthValue;
14         day = dayValue;
15         year = yearValue;
16     } // end three-argument constructor
17
18     // set the month
19     public void setMonth( int monthValue )
20     {
21         month = monthValue;
22     } // end method setMonth
23
24     // return the month
25     public int getMonth()
26     {
27         return month;
28     } // return month
29
30     // set the day
31     public void setDay( int dayValue )
32     {
33         day = dayValue;
34     } // end method setDay
35
```

Postlab Activities

Name: _____

Programming Challenges

```
36 // return the day
37 public int getDay()
38 {
39     return day;
40 } // return day
41
42 // set the year
43 public void setYear( int yearValue )
44 {
45     year = yearValue;
46 } // end method setYear
47
48 // return the year
49 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 // Programming Challenge 2: 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
```

