Testing Using JUnit

Objectives

- To know what JUnit is and how JUnit works (§44.2).
- To create and run a JUnit test class from the command window (§44.2).
- To create and run a JUnit test class from NetBeans (§44.3).
- To create and run a JUnit test class from Eclipse (§44.4).

44.1 Introduction

JUnit is a tool for testing Java programs.

At the very beginning of this book in Section 2.16, we introduced software development process that includes requirements specification, analysis, design, implementation, testing, deployment, and maintenance. Testing is an important part of this process. This chapter introduces how to test Java classes using JUnit.

44.2 JUnit Basics

To test a class, you need to write a test class and run it through JUnit to generate a report for the class.

JUnit is the de facto framework for testing Java programs. JUnit is a third-party open-source library packed in a jar file. The jar file contains a tool called *test runner*, which is used to run test programs. Suppose you have a class named **A**. To test this class, you write a test class named **ATest**. This test class, called a *test runner*, contains the methods you write for testing class **A**. The test runner executes **ATest** to generate a test report, as shown in Figure 44.1.
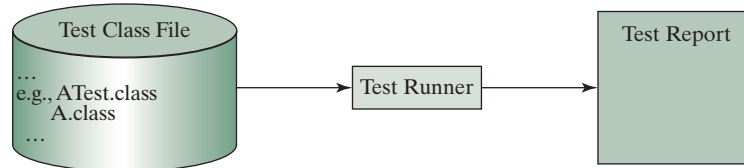

FIGURE 44.1 JUnit test runner executes the test class to generate a test report.

You will see how JUnit works from an example. To create the example, first you need to download JUnit from http://sourceforge.net/projects/junit/files/. At present, the latest version is junit-4.10.jar. Download this file to c:\book\lib and add it to the classpath environment variable as follows:

```
set classpath=.;%classpath%;c:\book\lib\junit-4.10.jar
```

To test if this environment variable is set correctly, open a new command window, and type the following command:

```
java org.junit.runner.JUnitCore
```

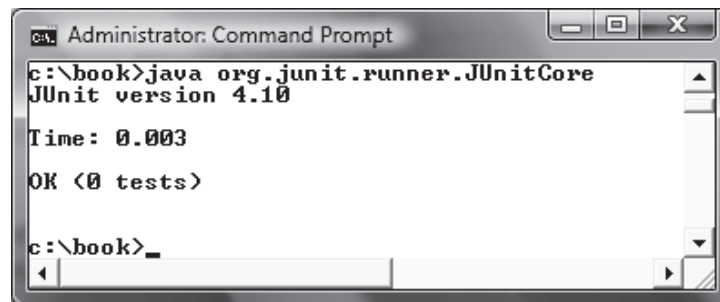You should see the message displayed as shown in Figure 44.2.

FIGURE 44.2 The JUnit test runner displays the JUnit version.
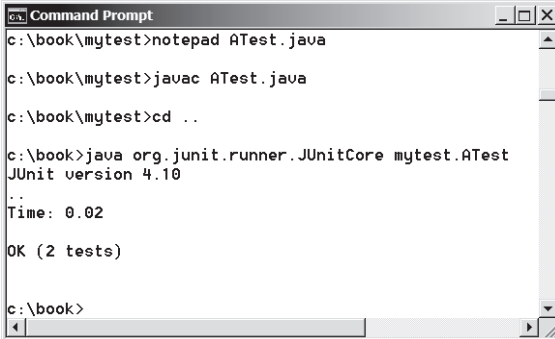
To use JUnit, create a test class. By convention, if the class to be tested is named **A**, the test class should be named **ATest**. A simple template of a test class may look like this:

```
1   package mytest;
2
3   import org.junit.*;
4   import static org.junit.Assert.*;
5
6   public class ATest {
7     @Test
8     public void m1() {
9       // Write a test method
10    }
11
12    @Test
13    public void m2() {
14      // Write another test method
15    }
16
17    @Before
18    public void setUp() throws Exception {
19      // Common objects used by test methods may be set up here
20    }
21  }
```

This class should be placed in a directory under mytest. Suppose the class is placed under c:\book\mytest. You need to compile it from the mytest directory and run it from c:\book as shown in the following screen shot.

Note the command to run the test from the console is:

```
java org.junit.runner.JUnitCore mytest.ATest
```

When this command is executed, **JUnitCore** controls the execution of **ATest**. It first executes the **setUp()** method to set up the common objects used for the test, and then executes test methods **m1** and **m2** in this order. You may define multiple test methods if desirable.

The following methods can be used to implement a test method:

assertTrue(booleanExpression)
The method reports success if the booleanExpression evaluates true.

assertEquals(Object, Object)
The method reports success if the two objects are the same using the **equals** method.

assertNull(Object)
The method reports success if the object reference passed is **null**.

fail(String)
The method causes the test to fail and prints out the string.

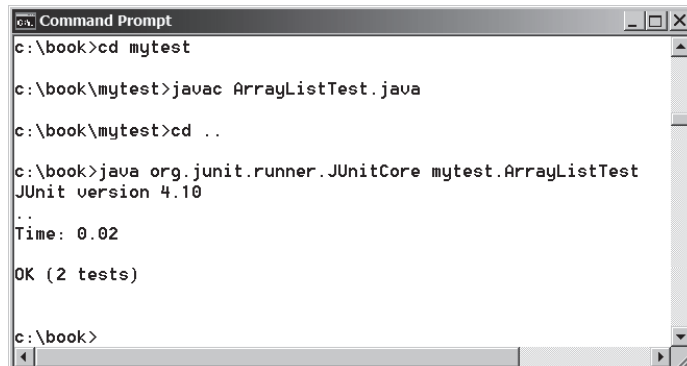Listing 44.1 is an example of a test class for testing `java.util.ArrayList`.

LISTING 44.1 ArrayListTest.java

```java
1  package mytest;
2
3  import org.junit.*;
4  import static org.junit.Assert.*;
5  import java.util.*;
6
7  public class ArrayListTest {
8    private ArrayList<String> list = new ArrayList<String>();
9
10   @Before
11   public void setUp() throws Exception {
12   }
13
14   @Test
15   public void testInsertion() {
16     list.add("Beijing");
17     assertEquals("Beijing", list.get(0));
18     list.add("Shanghai");
19     list.add("Hongkong");
20     assertEquals("Hongkong", list.get(list.size() - 1));
21   }
22
23   @Test
24   public void testDeletion() {
25     list.clear();
26     assertTrue(list.isEmpty());
27
28     list.add("A");
29     list.add("B");
30     list.add("C");
31     list.remove("B");
32     assertEquals(2, list.size());
33   }
34 }
```

A test run of the program is shown in Figure 44.3. Note that you have to first compile Array-ListTest.java. The `ArrayListTest` class is placed in the `mytest` package. So you should place ArrayListTest.java in the directory named `mytest`.


```
Command Prompt                                              _ □ ×
c:\book>cd mytest

c:\book\mytest>javac ArrayListTest.java

c:\book\mytest>cd ..

c:\book>java org.junit.runner.JUnitCore mytest.ArrayListTest
JUnit version 4.10
..
Time: 0.02

OK (2 tests)


c:\book>
```

FIGURE 44.3 The test report is displayed from running `ArrayListTest`.

No errors are reported in this JUnit run. If you mistakenly change

```
assertEquals(2, list.size());
```

in line 32 to

```
assertEquals(3, list.size());
```

Run **ArrayListTest** now. You will see an error reported as shown in Figure 44.4.

FIGURE 44.4   The test report reports an error.

  You can define any number of test methods. In this example, the two test methods **test-Insertion** and **testDeletion** are defined. JUnit executes **testInsertion** and **test-Deletion** in this order.

> **Note**
> The test class must be placed in a named package such as **mytest** in this example. The JUnit will not work if the test class is placed a default package.

Listing 44.2 gives a test class for testing the **Loan** class in Listing 10.2. For convenience, we create Loan.java in the same directory with LoanTest.java. The **Loan** class is shown in Listing 44.3.

LISTING 44.2   LoanTest.java

```
1   package mytest;
2
3   import org.junit.*;
4   import static org.junit.Assert.*;
5
6   public class LoanTest {
7     @Before
8     public void setUp() throws Exception {
9     }
10
11    @Test
12    public void testPaymentMethods() {
13      double annualInterestRate = 2.5;
14      int numberOfYears = 5;
15      double loanAmount = 1000;
16      Loan loan = new Loan(annualInterestRate, numberOfYears,
17        loanAmount);
18
```

```
19        assertTrue(loan.getMonthlyPayment() ==
20          getMonthlyPayment(annualInterestRate, numberOfYears,
21          loanAmount));
22        assertTrue(loan.getTotalPayment() ==
23          getTotalPayment(annualInterestRate, numberOfYears,
24          loanAmount));
25      }
26
27      /** Find monthly payment */
28      private double getMonthlyPayment(double annualInterestRate,
29          int numberOfYears, double loanAmount) {
30        double monthlyInterestRate = annualInterestRate / 1200;
31        double monthlyPayment = loanAmount * monthlyInterestRate / (1 -
32          (1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12)));
33        return monthlyPayment;
34      }
35
36      /** Find total payment */
37      public double getTotalPayment(double annualInterestRate,
38          int numberOfYears, double loanAmount) {
39        return getMonthlyPayment(annualInterestRate, numberOfYears,
40          loanAmount) * numberOfYears * 12;
41      }
42    }
```

LISTING 44.3 Loan.java

```
1  package mytest;
2
3  public class Loan {
4    private double annualInterestRate;
5    private int numberOfYears;
6    private double loanAmount;
7    private java.util.Date loanDate;
8
9    /** Default constructor */
10   public Loan() {
11     this(2.5, 1, 1000);
12   }
13
14   /** Construct a loan with specified annual interest rate,
15       number of years, and loan amount
16     */
17   public Loan(double annualInterestRate, int numberOfYears,
18       double loanAmount) {
19     this.annualInterestRate = annualInterestRate;
20     this.numberOfYears = numberOfYears;
21     this.loanAmount = loanAmount;
22     loanDate = new java.util.Date();
23   }
24
25   /** Return annualInterestRate */
26   public double getAnnualInterestRate() {
27     return annualInterestRate;
28   }
29
30   /** Set a new annualInterestRate */
31   public void setAnnualInterestRate(double annualInterestRate) {
32     this.annualInterestRate = annualInterestRate;
33   }
```

```
34
35     /** Return numberOfYears */
36     public int getNumberOfYears() {
37       return numberOfYears;
38     }
39
40     /** Set a new numberOfYears */
41     public void setNumberOfYears(int numberOfYears) {
42       this.numberOfYears = numberOfYears;
43     }
44
45     /** Return loanAmount */
46     public double getLoanAmount() {
47       return loanAmount;
48     }
49
50     /** Set a newloanAmount */
51     public void setLoanAmount(double loanAmount) {
52       this.loanAmount = loanAmount;
53     }
54
55     /** Find monthly payment */
56     public double getMonthlyPayment() {
57       double monthlyInterestRate = annualInterestRate / 1200;
58       double monthlyPayment = loanAmount * monthlyInterestRate / (1 -
59         (1 / Math.pow(1 + monthlyInterestRate, numberOfYears * 12)));
60       return monthlyPayment;
61     }
62
63     /** Find total payment */
64     public double getTotalPayment() {
65       double totalPayment = getMonthlyPayment() * numberOfYears * 12;
66       return totalPayment;
67     }
68
69     /** Return loan date */
70     public java.util.Date getLoanDate() {
71       return loanDate;
72     }
73   }
```

The **testPaymentMethods()** in **LoanTest** creates an instance of **Loan** (line 16–17) and
tests whether **loan.getMonthlyPayment()** returns the same value as **getMonthlyPayment
(annualInterestRate, numberOfYears, loanAmount)**. The latter method is defined
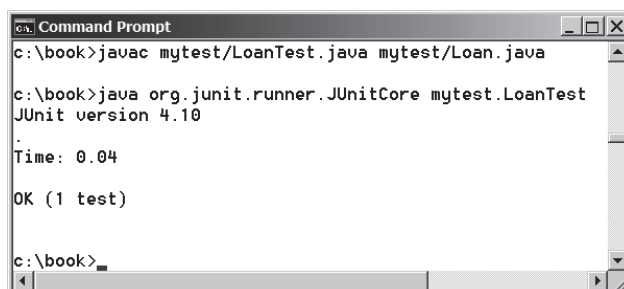in the **LoanTest** class (lines 28–34).

FIGURE 44.5   The JUnit test runner executes **LoanTest** and reports no errors.

The **testPaymentMethods()** also tests whether **loan.getTotalPayment()** returns the same value as **getTotalPayment(annualInterestRate, numberOfYears, loan-Amount)**. The latter method is defined in the **LoanTest** class (lines 37–41).

A sample run of the program is shown in Figure 44.5.

✓ **Check Point**

44.2.1 What is JUnit?

44.2.2 What is a JUnit test runner?

44.2.3 What is a test class? How do you create a test class?

44.2.4 How do you use the **assertTrue** method?

44.2.5 How do you use the **assertEquals** method?

44.3 Using JUnit from NetBeans

Key Point

JUnit is intergrated with NetBeans. Using NetBeans, the test program can be automatically generated and the test process can be automated.

An IDE such as NetBeans and Eclipse can greatly simplify the process for creating and running test classes. This section introduces using JUnit from NetBeans, and the next section will introduce using JUnit from Eclipse.

If you are not familiar with NetBeans, see Supplement II.B. Assume you have installed NetBeans 8 or higher. Create a project named **chapter44** as follows:

Step 1: Choose *File*, *New Project* to display the New Project dialog box.

Step 2: Choose Java in the Categories section and Java Application in the Projects section. Click *Next* to display the New Java Application dialog box.

Step 3: Enter **chapter44** as the Project Name and **c:\book** as Project Location. Click *Finish* to create the project as shown in Figure 44.6.

To demonstrate how to create a test class, we first create a class to be tested. Let the class be **Loan** from Listing 10.2. Here are the steps to create the **Loan** class under **chapter44**.
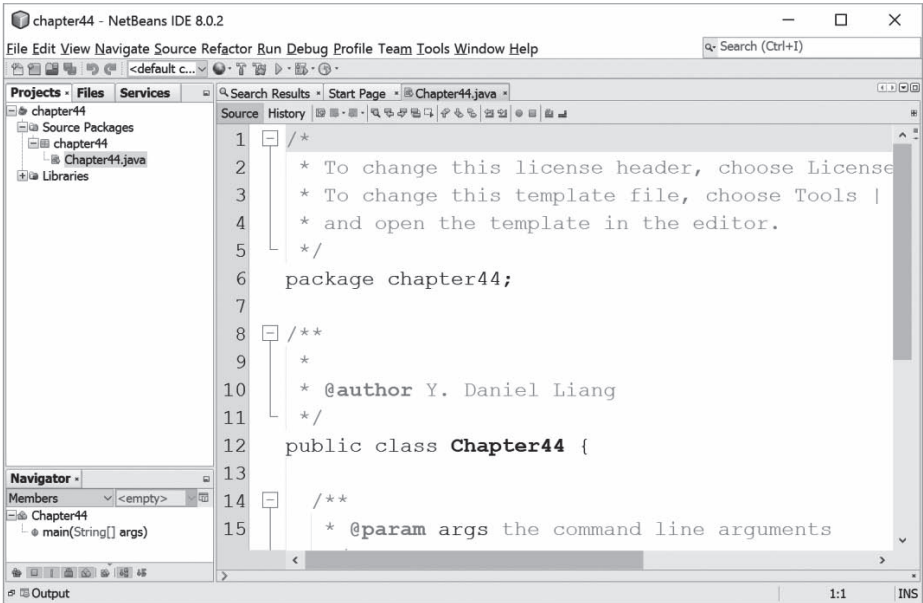

FIGURE 44.6 A new project named **chapter44** is created.

Step 1: Right-click the project node **chapter44** and choose *New*, *Java Class* to display the New Java Class dialog box.

Step 2: Enter **Loan** as Class Name and **chapter44** in the Package field and click *Finish* to create the class.

Step 3: Copy the code in Listing 10.2 to the **Loan** class and make sure the first line is **package chapter44**, as shown in Figure 44.7.
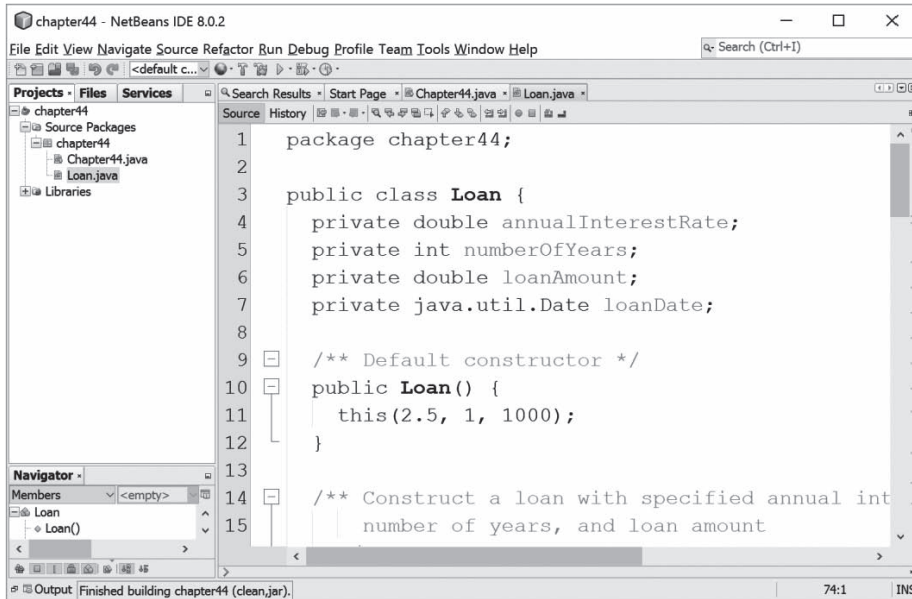

FIGURE 44.7 The **Loan** class is created.

Now you can create a test class to test the **Loan** class as follows:

Step 1: Right-click Loan.java in the project to display a context menu and choose *Tools*, *Create/Update Test* to display the Create Test dialog box, as shown in Figure 44.8.
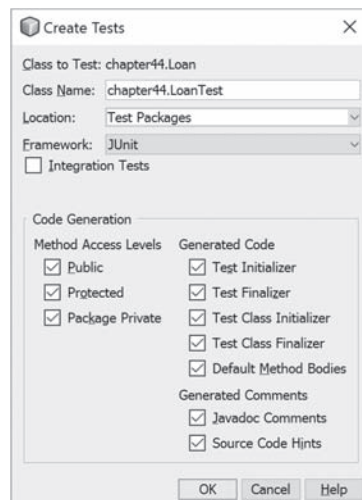

FIGURE 44.8 The Create Tests dialog box creates a Test class.

Step 2: Click OK. You will see the Select JUnit version dialog box displayed as shown in Figure 44.9. Choose Junit 4.x. Click *OK* to generate a test class named **LoanTest** as shown in Figure 44.10. Note that LoanTest.java is placed under the Test Packages node in the project.

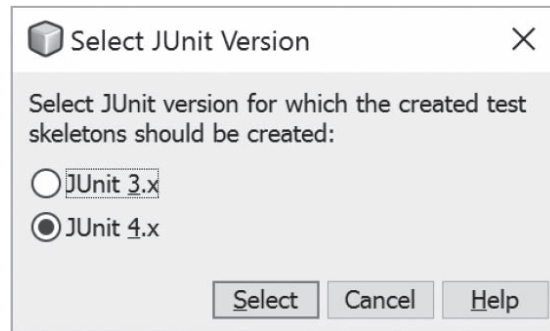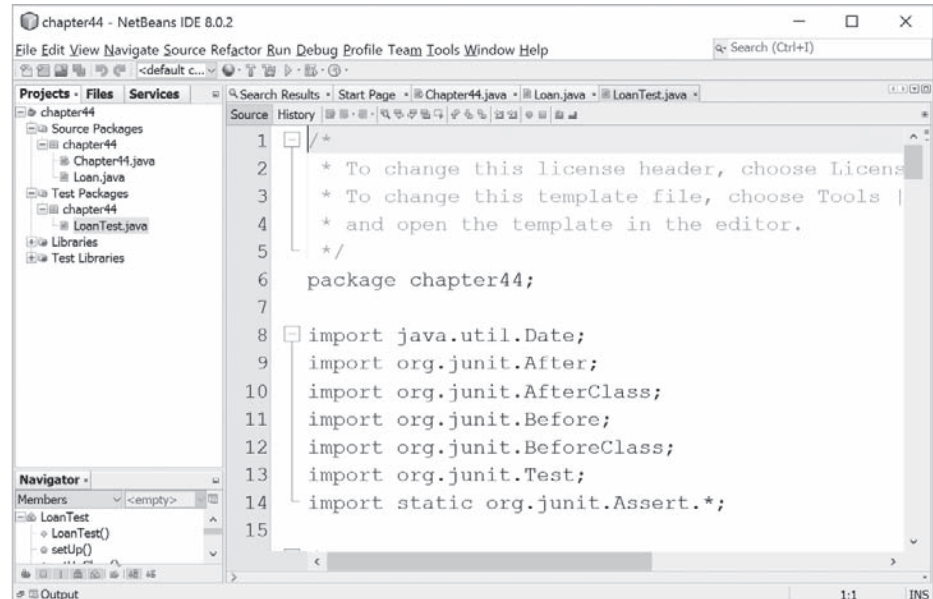**FIGURE 44.9** You should select JUnit 4.x framework to create test classes.

FIGURE 44.10 The **LoanTest** class is automatically generated.

You can now modify **LoanTest** by copying the code from Listing 44.2. Run LoanTest.java. You will see the test report as shown in Figure 44.11.
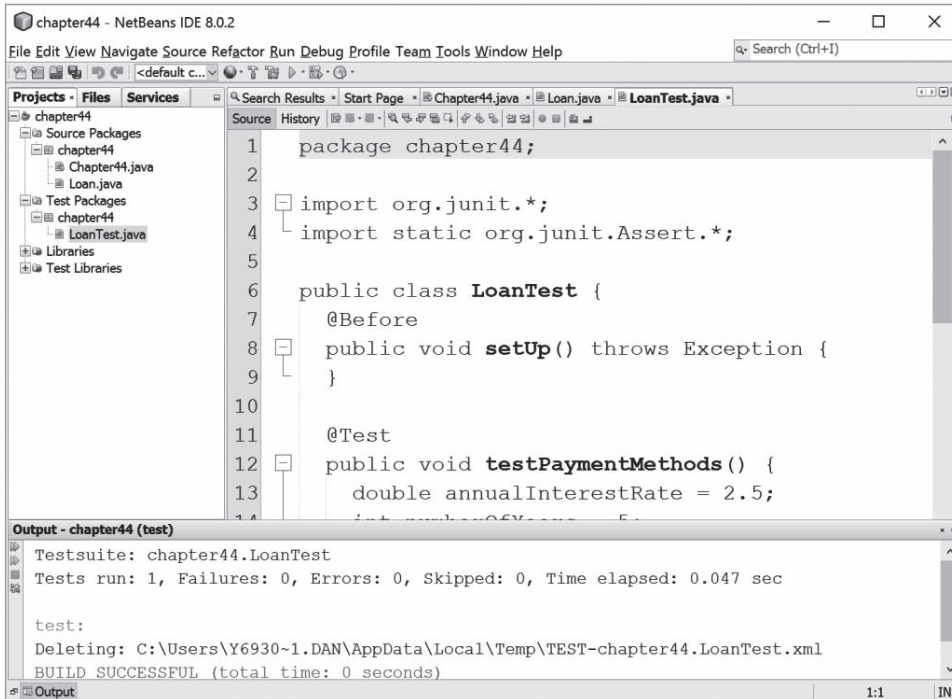

FIGURE 44.11 The test report is displayed after the **LoanTest** class is executed.

44.4 Using JUnit from Eclipse

JUnit is intergrated with Eclipse. Using Eclipse, the test program can be automatically generated and the test process can be automated.

Key Point

This section introduces using JUnit from Eclipse. If you are not familiar with Eclipse, see Supplement II.D. Assume you have installed Eclipse 4.5 or higher. Create a project named **chapter50** as follows:

Step 1: Choose *File*, *New Java Project* to display the New Java Project dialog box, as shown in Figure 44.12.

Step 2: Enter **chapter50** in the project name field and click *Finish* to create the project.

To demonstrate how to create a test class, we first create a class to be tested. Let the class be **Loan** from Listing 10.2. Here are the steps to create the **Loan** class under **chapter44**.
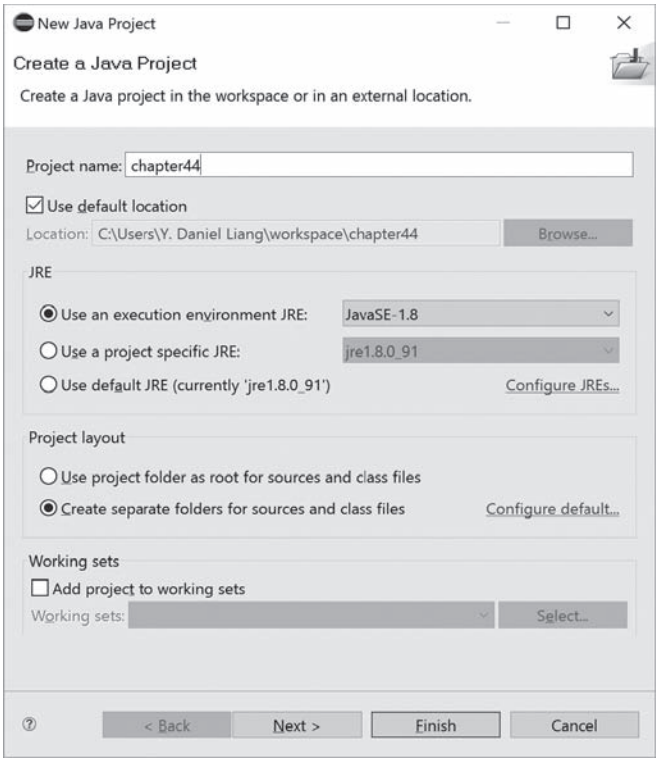
FIGURE 44.12 The New Java Project dialog creates a new project.

Step 1: Right-click the project node **chapter44** and choose *New*, *Class* to display the New Java Class dialog box, as shown in Figure 44.13.

Step 2: Enter **mytest** in the Package field and click *Finish* to create the class.

Step 3: Copy the code in Listing 10.2 to the **Loan** class and make sure the first line is **package mytest**, as shown in Figure 44.14.

Now you can create a test class to test the **Loan** class as follows:

Step 1: Right-click Loan.java in the project to display a context menu and choose *New*, *JUnit Test Case* to display the New JUnit Test Case dialog box, as shown in Figure 44.15.

Step 2: Click *Finish*. You will see a dialog prompting you to add JUnit 4 to the project build path. Click *OK* to add it. Now a test class named LoanTest is created as shown in Figure 44.16.

You can now modify **LoanTest** by copying the code from Listing 44.2. Run LoanTest.java. You will see the test report as shown in Figure 44.17.

KEY TERMS

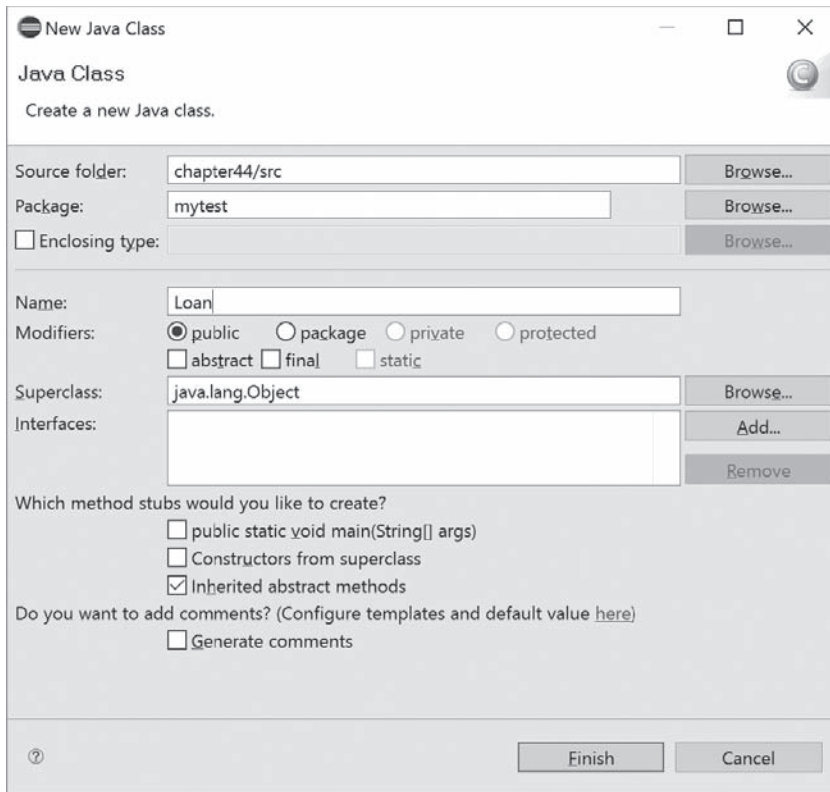|	
JUnit 44-2	test class 44-2
JUnitCore 44-2	test runner 44-2

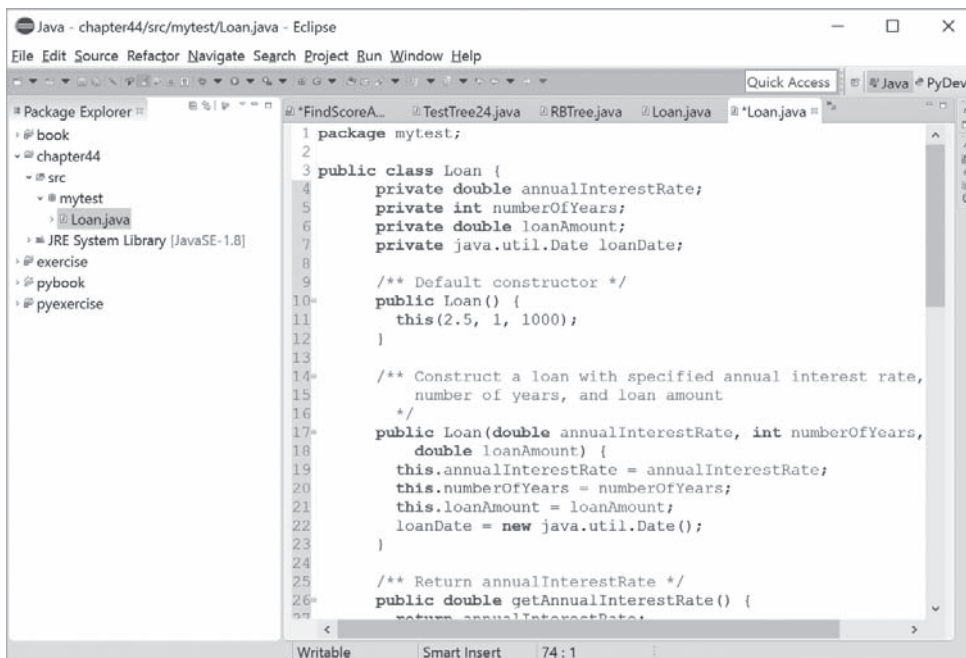**FIGURE 44.13** The New Java Class dialog creates a new Java class.

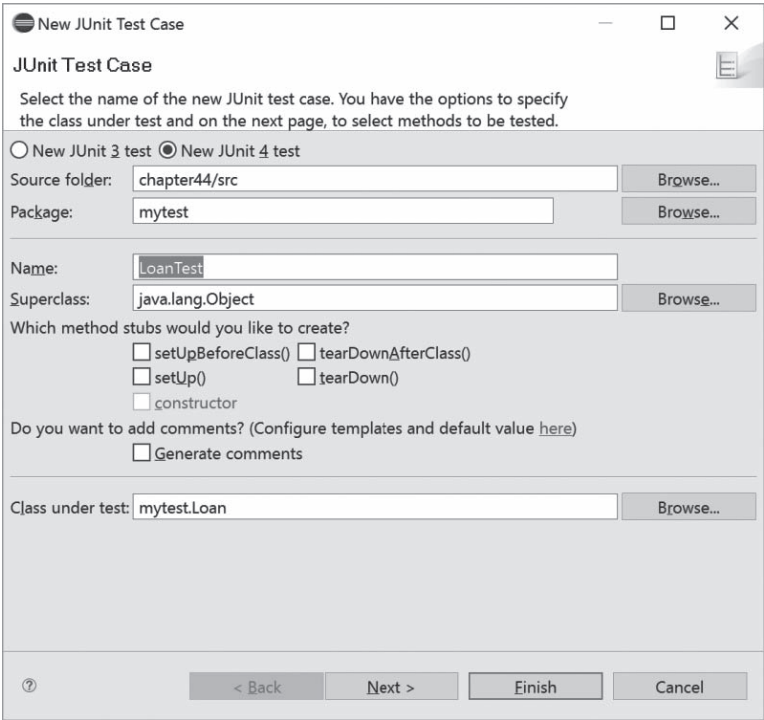**FIGURE 44.14** The Loan class is created.

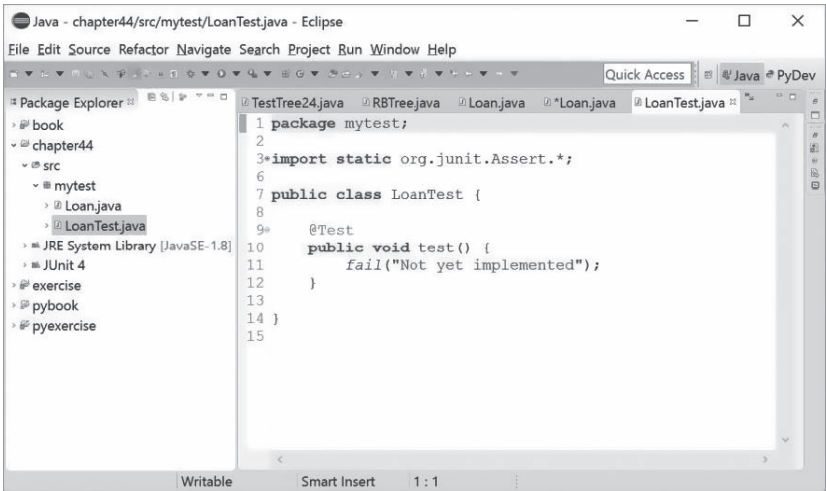**Figure 44.15** The New JUnit Test Case dialog box creates a Test class.

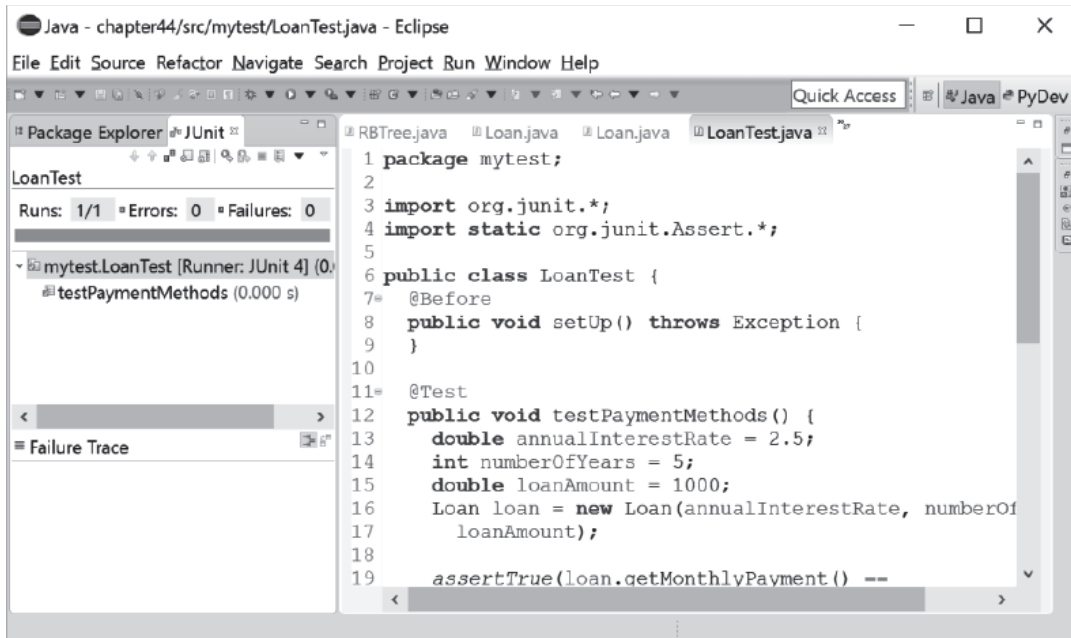**Figure 44.16** The **LoanTest** class is automatically generated.

FIGURE 44.17 The test report is displayed after the **LoanTest** class is executed.

CHAPTER SUMMARY

1. JUnit is an open-source framework for testing Java programs.

2. To test a Java class, you create a test class for the class to be tested and use JUnit's test runner to execute the test class to generate a test report.

3. You can create and run a test class from the command window or use a tool such as NetBeans and Eclipse.

QUIZ

Answer the quiz for this chapter online at the book Companion Website.

PROGRAMMING EXERCISES

MyProgrammingLab™

44.1 Write a test class to test the methods **length**, **charAt**, **substring**, and **indexOf** in the **java.lang.String** class.

44.2 Write a test class to test the methods **add**, **remove**, **addAll**, **removeAll**, **size**, **isEmpty**, and **contains** in the **java.util.HashSet** class.

44.3 Write a test class to test the method **isPrime** in Listing 6.7, PrimeNumberMethod. java.

44.4 Write a test class to test the methods **getBMI** and **getStatus** in the BMI class in Listing 10.4.