\*\*\*This is a bonus Web chapter

CHAPTER 50

Testing Using JUnit

Objectives

* To know what JUnit is and how JUnit works (§50.2).
* To create and run a JUnit test class from the command window (§50.2).
* To create and run a JUnit test class from NetBeans (§50.3).
* To create and run a JUnit test class from Eclipse (§50.4).

50.1 Introduction

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.

50.2 JUnit Basics

<key term>JUnit

<key term>test runner

<key term>test 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 class*, contains the methods you write for testing class A. The test runner executes ATest to generate a test report, as shown in Figure 50.1.



Figure 50.1

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 50.2.

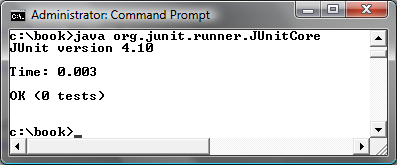


Figure 50.2

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:

package mytest;

import org.junit.\*;

import static org.junit.Assert.\*;

public class ATest {

@Test

public void m1() {

// Write a test method

}

@Test

public void m2() {

// Write another test method

}

@Before

public void setUp() throws Exception {

// Common objects used by test methods may be set up here

}

}

To run the test from the console, use the following command:

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

<key term>JUnitCore

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 50.1 is an example of a test class for testing java.util.ArrayList.

Listing 50.1 ArrayListTest.java

<Side Remark line 7: test class>

<Side Remark line 15: test method>

<Side Remark line 17: assertion>

<Side Remark line 20: assertion>

<Side Remark line 24: test method>

<Side Remark line 26: assertion>

<Side Remark line 32: assertion>

package mytest;

import org.junit.\*;

import static org.junit.Assert.\*;

import java.util.\*;

public class ArrayListTest {

private ArrayList<String> list = new ArrayList<String>();

@Before

public void setUp() throws Exception {

}

@Test

public void testInsertion() {

list.add("Beijing");

assertEquals("Beijing", list.get(0));

list.add("Shanghai");

list.add("Hongkong");

assertEquals("Hongkong", list.get(list.size() - 1));

}

@Test

public void testDeletion() {

list.clear();

assertTrue(list.isEmpty());

list.add("A");

list.add("B");

list.add("C");

list.remove("B");

assertEquals(2, list.size());

}

}

A test run of the program is shown in Figure 50.3. Note that you have to first compile ArrayListTest.java. The ArrayListTest class is placed in the mytest package. So you should place ArrayListTest.java in the directory named mytest.

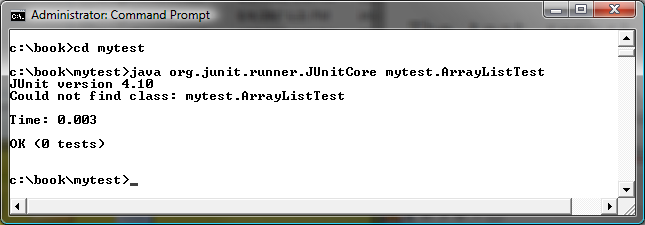


Figure 50.3

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 50.4.

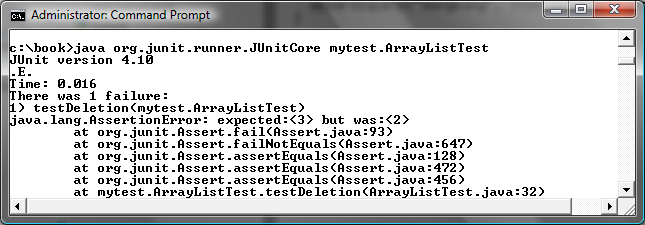


Figure 50.4

The test report reports an error.

You can define any number of test methods. In this example, two test methods testInsertion and testDeletion are defined. JUnit executes testInsertion and testDeletion 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 50.2 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 50.3.

Listing 50.2 LoanTest.java

<Side Remark line 6: test class>

<Side Remark line 12: test method>

<Side Remark line 19: assertion>

<Side Remark line 22: assertion>

<Side Remark line 28: compute monthly payment>

<Side Remark line 37: compute total payment>

package mytest;

import org.junit.\*;

import static org.junit.Assert.\*;

public class LoanTest {

@Before

public void setUp() throws Exception {

}

@Test

public void testPaymentMethods() {

double annualInterestRate = 2.5;

int numberOfYears = 5;

double loanAmount = 1000;

Loan loan = new Loan(annualInterestRate, numberOfYears,

loanAmount);

assertTrue(loan.getMonthlyPayment() ==

getMonthlyPayment(annualInterestRate, numberOfYears,

loanAmount));

assertTrue(loan.getTotalPayment() ==

getTotalPayment(annualInterestRate, numberOfYears,

loanAmount));

}

/\*\* Find monthly payment \*/

private double getMonthlyPayment(double annualInterestRate,

int numberOfYears, double loanAmount) {

double monthlyInterestRate = annualInterestRate / 1200;

double monthlyPayment = loanAmount \* monthlyInterestRate / (1 -

(1 / Math.pow(1 + monthlyInterestRate, numberOfYears \* 12)));

return monthlyPayment;

}

/\*\* Find total payment \*/

public double getTotalPayment(double annualInterestRate,

int numberOfYears, double loanAmount) {

return getMonthlyPayment(annualInterestRate, numberOfYears,

loanAmount) \* numberOfYears \* 12;

}

}

Listing 50.3 Loan.java

<Side Remark line 1: mytest package>

<Side Remark line 3: the Loan class>

<Side Remark line 56: getMonthlyPayment>

<Side Remark line 64: getTotalPayment>

package mytest;

public class Loan {

private double annualInterestRate;

private int numberOfYears;

private double loanAmount;

private java.util.Date loanDate;

/\*\* Default constructor \*/

public Loan() {

this(2.5, 1, 1000);

}

/\*\* Construct a loan with specified annual interest rate,

number of years, and loan amount

\*/

public Loan(double annualInterestRate, int numberOfYears,

double loanAmount) {

this.annualInterestRate = annualInterestRate;

this.numberOfYears = numberOfYears;

this.loanAmount = loanAmount;

loanDate = new java.util.Date();

}

/\*\* Return annualInterestRate \*/

public double getAnnualInterestRate() {

return annualInterestRate;

}

/\*\* Set a new annualInterestRate \*/

public void setAnnualInterestRate(double annualInterestRate) {

this.annualInterestRate = annualInterestRate;

}

/\*\* Return numberOfYears \*/

public int getNumberOfYears() {

return numberOfYears;

}

/\*\* Set a new numberOfYears \*/

public void setNumberOfYears(int numberOfYears) {

this.numberOfYears = numberOfYears;

}

/\*\* Return loanAmount \*/

public double getLoanAmount() {

return loanAmount;

}

/\*\* Set a newloanAmount \*/

public void setLoanAmount(double loanAmount) {

this.loanAmount = loanAmount;

}

/\*\* Find monthly payment \*/

public double getMonthlyPayment() {

double monthlyInterestRate = annualInterestRate / 1200;

double monthlyPayment = loanAmount \* monthlyInterestRate / (1 -

(1 / Math.pow(1 + monthlyInterestRate, numberOfYears \* 12)));

return monthlyPayment;

}

/\*\* Find total payment \*/

public double getTotalPayment() {

double totalPayment = getMonthlyPayment() \* numberOfYears \* 12;

return totalPayment;

}

/\*\* Return loan date \*/

public java.util.Date getLoanDate() {

return loanDate;

}

}

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).

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

A sample run of the program is shown in Figure 50.5.

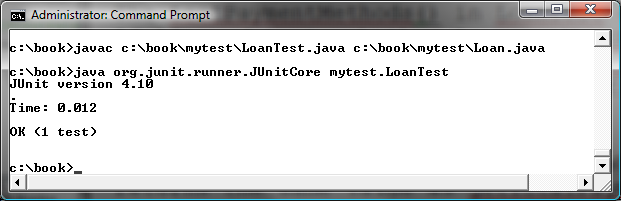


Figure 50.5

The JUnit test runner executes LoanTest and reports no errors.

50.3 Using JUnit from NetBeans

An IDE like 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 introduces using JUnit from Eclipse.

If you not familiar with NetBeans, see Supplement II.B. Assume you have installed NetBeans 7.0. Create a project named chapter50 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 chapter50 as the Project Name and c:\book as Project Location. Click *Finish* to create the project as shown in Figure 50.6.

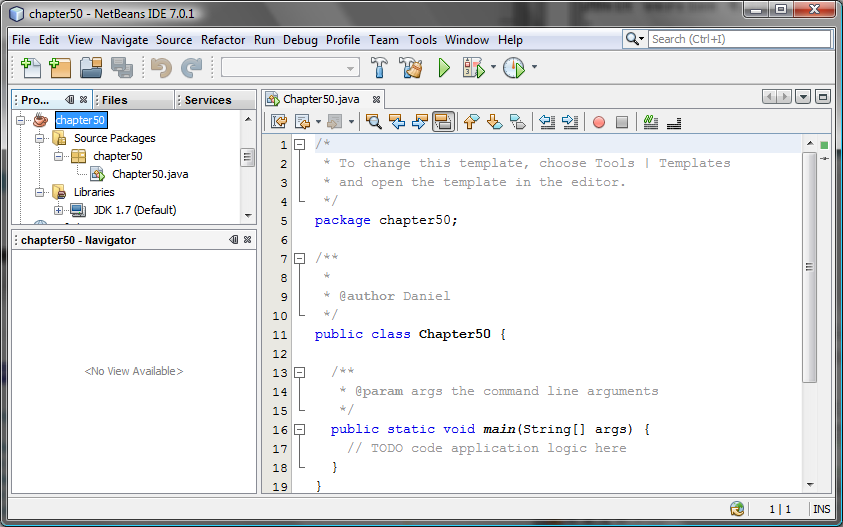


Figure 50.6

A new project named chapter50 is created.

To demonstrate how to create a test class, we first create class to be tested. Let the class be Loan from Listing 10.2. Here are the steps to create the Loan class under chapter50.

Step 1: Right-click the project node chapter50 and choose *New*, *Java Class* to display the New Java Class dialog box.

Step 2: Enter Loan as Class Name and chapter50 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 chapter50, as shown in Figure 50.7.

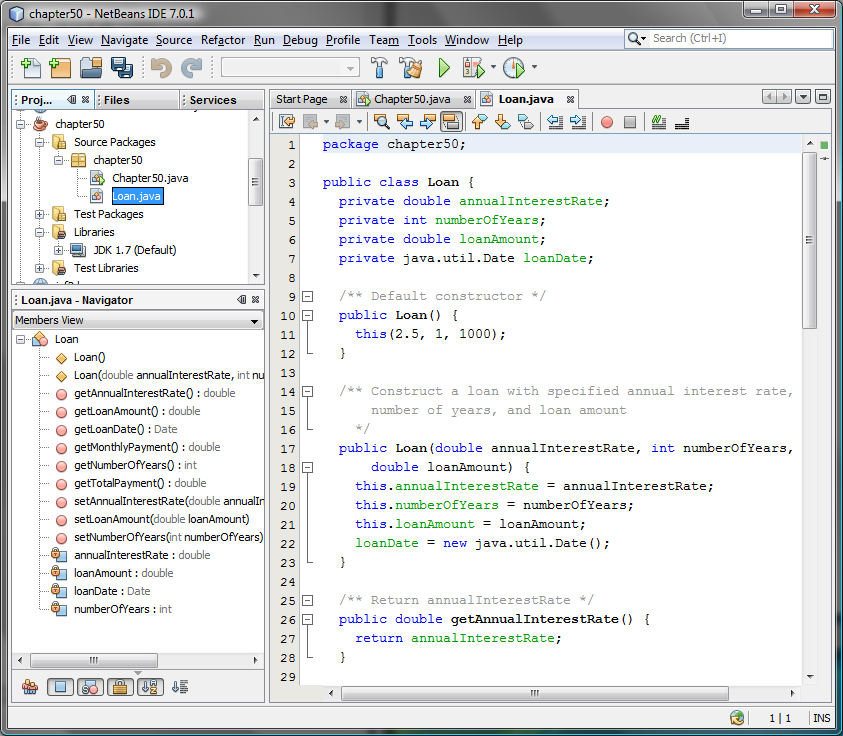


Figure 50.7

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 JUnit Test* to display the Select JUnit version dialog box, as shown in Figure 50.8.

Step 2: Choose JUnit 4.x. You will see the Create Tests dialog box displayed as shown in Figure 50.9. Click *OK* to generate a Test class named LoanTest as shown in Figure 50.10. Note that LoanTest.java is placed under the Test Packages node in the project.

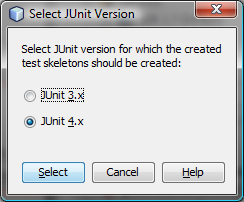


Figure 50.8

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

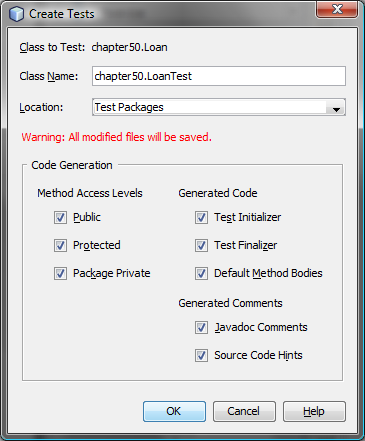


Figure 50.9

The Create Tests dialog box creates a Test class.

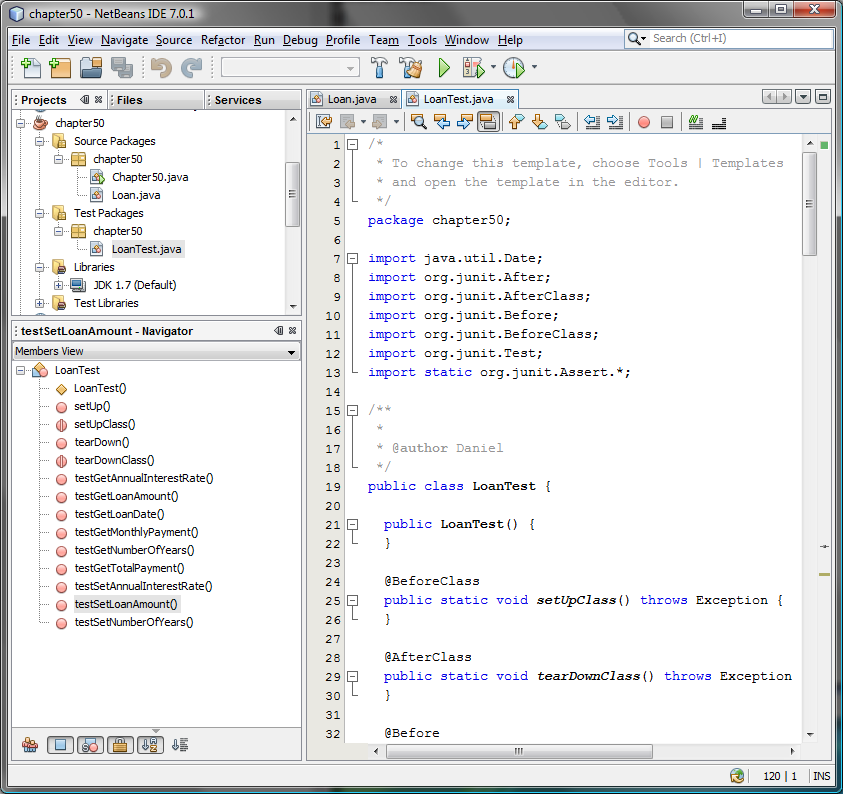


Figure 50.10

The LoanTest class is automatically generated.

You can now modify LoanTest by copying the code from Listing 50.2. Run LoanTest.java. You will see the test report as shown in Figure 50.11.

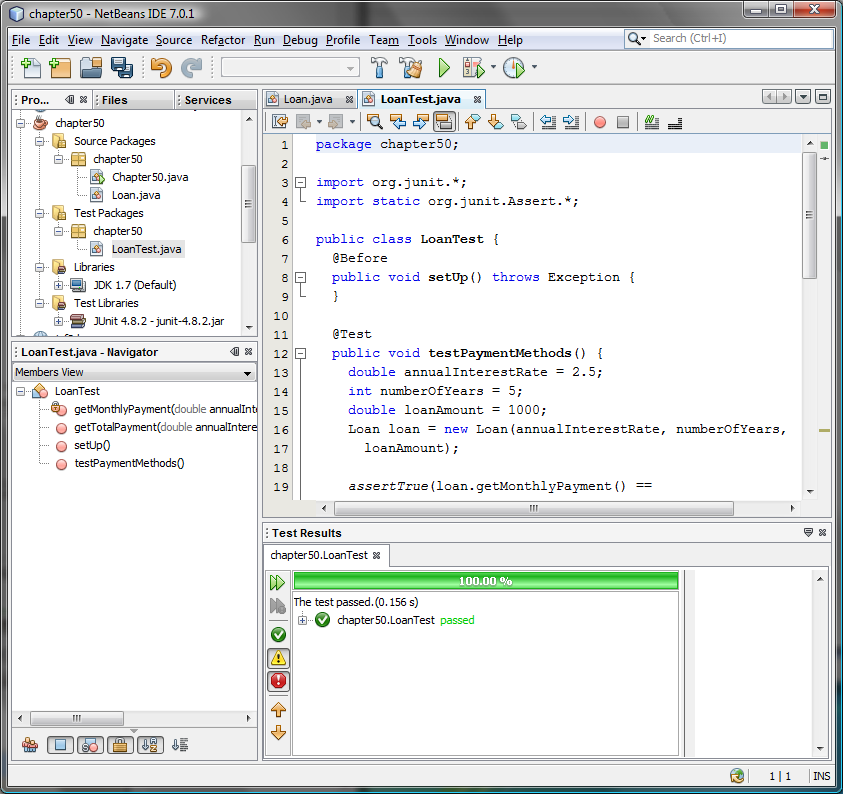


Figure 50.11

The test report is displayed after the LoanTest class is executed.

50.4 Using JUnit from Eclipse

This section introduces using JUnit from Eclipse. If you are not familiar with Eclipse, see Supplement II.D. Assume you have installed Eclipse 3.7. 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 50.12.

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

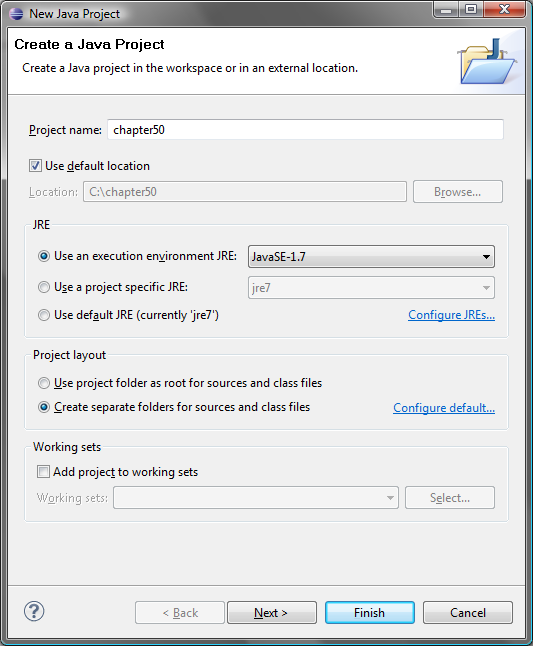


Figure 50.12

The New Java Project dialog creates a new project.

To demonstrate how to create a test class, we first create class to be tested. Let the class be Loan from Listing 10.2. Here are the steps to create the Loan class under chapter50.

Step 1: Right-click the project node chapter50 and choose *New*, *Class* to display the New Java Class dialog box, as shown in Figure 50.13.

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 50.14.

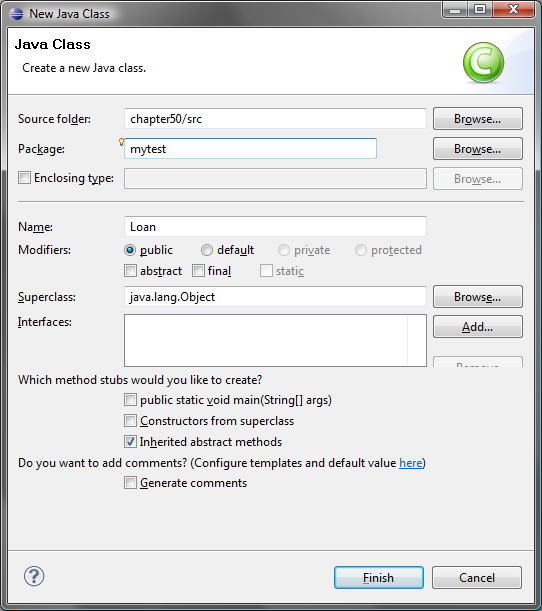


Figure 50.13

The New Java Class dialog creates a new Java class.

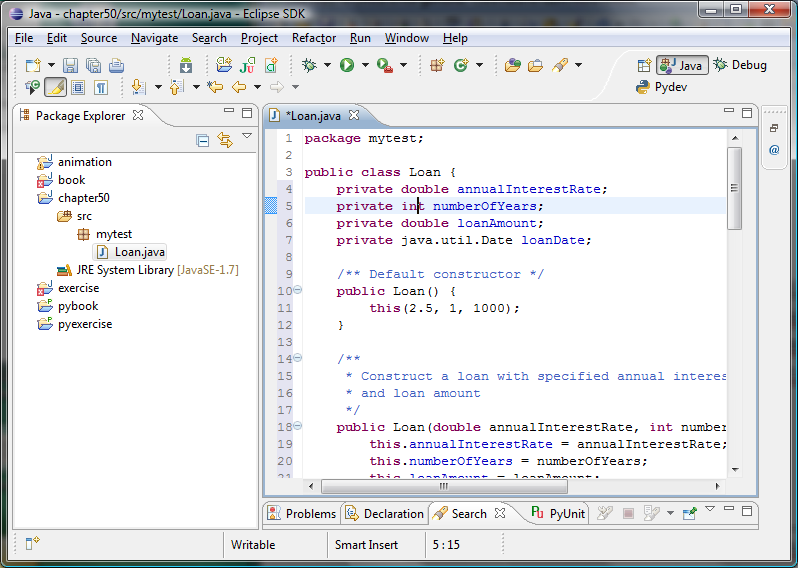


Figure 50.14

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 *New*, *JUnit Test Case* to display the New JUnit Test Case dialog box, as shown in Figure 50.15.

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 50.16.

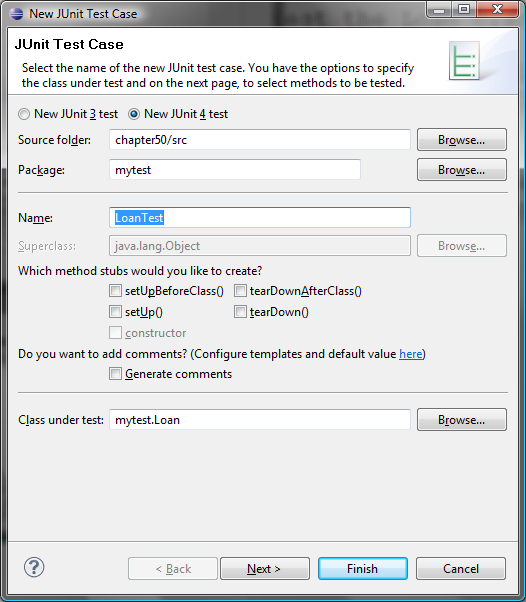


Figure 50.15

The New JUnit Test Case dialog box creates a Test class.

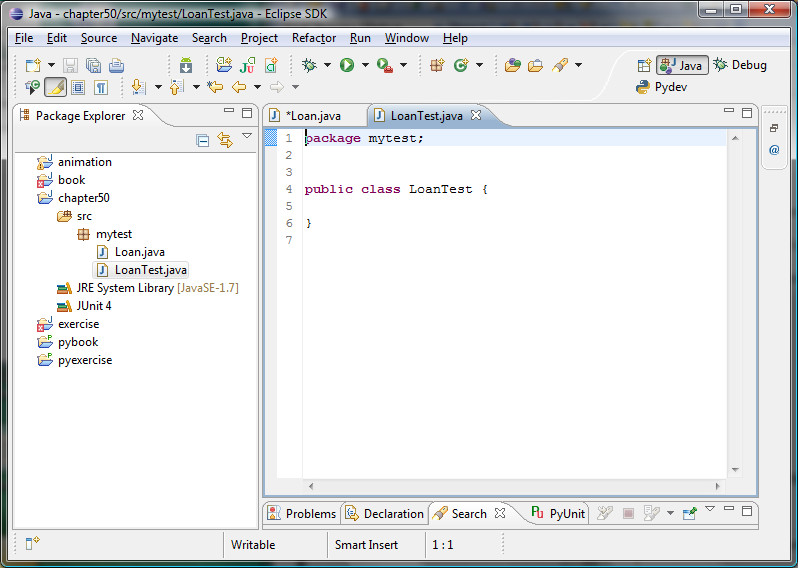


Figure 50.16

The LoanTest class is automatically generated.

You can now modify LoanTest by copying the code from Listing 50.2. Run LoanTest.java. You will see the test report as shown in Figure 50.17.

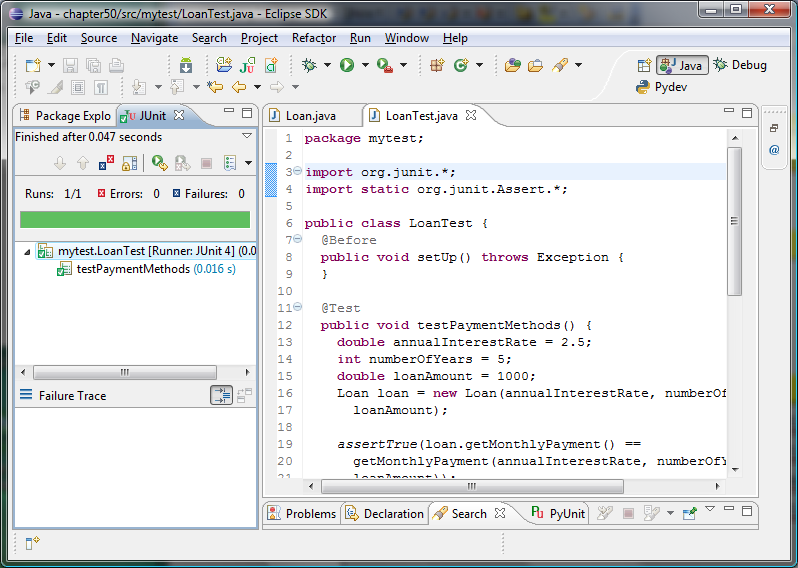


Figure 50.17

The test report is displayed after the LoanTest class is executed.

Key Terms

* **JUnit**
* **JUnitCore**
* **test class**
* **test runner**

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.

Test Questions

Do the test questions for this chapter online at **www.cs.armstrong.edu/liang/intro9e/test.html**.

Review Questions

50.1 What is JUnit?

50.2 What is a JUnit test runner?

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

50.4 How do you use the assertTrue method?

50.5 How do you use the assertEquals method?

Programming Exercises

50.1

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

50.2

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

50.3

Write a test class to test the method isPrime in Listing 5.7 PrimeNumberMethod.java.

50.4

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