Supplement: Testing Java Programs Using JUnit

### For Introduction to Java Programming By Y. Daniel Liang

JUnit is a simple framework for testing Java programs. You can download the latest version of JUnit from <a href="http://sourceforge.net/projects/junit/files/">http://sourceforge.net/projects/junit/files/</a>. At present, the latest version is junit-4.10.jar. Download this file and add it to the classpath as follows:

set CLASSPATH=.;%CLASSPATH%;%JUNIT\_HOME%\junit-4.10.jar

To use JUnit, create a test class. By convention, if the class to be tested is named  $\underline{A}$ , the test class should be named  $\underline{ATest}$ . A simple template of a test class may look like this:

package test;

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

java org.junit.runner.JUnitCore test.ATest

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

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

<u>assertTrue(booleanExpression)</u> The method reports success if the booleanExpression evaluates true.

<u>assertEquals(Object, Object)</u> The method reports success if the two objects are the same using the equals method.

<u>assertNull(Object)</u> The method reports success if the object reference passed is <u>null</u>.

 $\frac{fail(String)}{Causes}$  the test to fail and prints out the string.

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

package test;

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()); } }

#### <Output>

Testsuite: test.ArrayListTest <u>Setup</u> <u>Setup</u> Tests run: 2, Failures: 0, Errors: 0, Time elapsed: 0.172 sec

#### <End Output>

You can define any number of test methods. In this example, two test methods <u>testInsertion</u> and <u>testDeletion</u> are defined. JUnit executes testInsertion and testDeletion in this order.

An IDE like NetBeans and Eclipse can greatly simplify the process for creating and running test classes. Consider using NetBeans 6.0. Let us create a class Loan.java for Listing 9.2 in Chapter 9.

Create a new project named <u>JUnitDemo</u> and class <u>Loan</u> in the project, as shown in Figure 1. To create a test class for <u>Loan</u>, right-click Loan in the project pane to display a context menu. Choose **Tools** > **Create JUnit Test** to display a dialog box, as shown in Figure 2. Check all boxes to generate LoanTest. LoanTest.java now appears under the *Test Packages* node in the test project, as shown in Figure 3.

🗊 JUnitDemo - NetBeans IDE 6.0 Beta 1					
File Edit View Navigate Source Refactor Build Run Profile Versioning Tools Window Help					
1 🔁 🚰 🔩 😹 🏝 🗂 🦈 🦿   					
Proj       4 × Files	Bervices	SQL Command 1 × • • • • Q public class private dou private int private dou Ctrl+X Ctrl+C Ctrl+V F9	<pre>VirtualFormDemo x Stat Page x &amp; D LoanJava x &amp; D LoanTestjava x VirtualFormDemo x Stat Page x &amp; D LoanTestjava x VirtualFormDemo x Stat Page x &amp; D LoanTestjava x VirtualFormDemo x &amp; V V VirtualFormDemo x &amp; V VirtualFormPemo x &amp; V VirtuaF</pre>		
	Run File Debug File Profile File Add Delete Save As Template Find Usages Refactor	Shift+F6 Ctrl+Shift+F5 Delete Alt+F7	<pre>ct a loan with specified annual interest rate, of years and loan amount (double annualInterestRate, int numberOfYears, loanAmount) { alInterestRate = annualInterestRate; erOfYears = numberOfYears; Amount = loanAmount; = new java.util.Date();</pre>		
	File Members     Ctrl+F12     annualInterestRate */       File Hierarchy     Alt+F12     le getAnnualInterestRate;       Local History     Image: Ctrl + F12     Image: Ctrl + F12	annualInterestRate */ le getAnnualInterestRate() { nualInterestRate;			
	30         31         32         33         34         35         36         37         38         36         37         37         38         36         37         37         38         36         37         38         36         37         38         36         37         38         36         37         38         36         37         38         36<	<pre>this.ann } /** Return public int return nt } /** Set a r nublic voir INS</pre>	Add to Favorites  Create JUnit Tests Ctrl+Shift+U  Add to Palette  Apply Diff Patch Internationalization  umber Of Years;  hew number Of Years */		

# Figure 1

A test class can be created automatically in NetBeans.

🗊 Create Te	sts	×			
<u>C</u> lass to Test: Loan					
Class <u>N</u> ame:	LoanTest				
Location:	Test Package	es 💌			
Code Generation					
Method Ac	cess Levels	Generated Code			
Public		🔽 Test Initializer			
✓ Prot	ected	🔽 Test Finalizer			
Pack	age Private	Default <u>M</u> ethod Bodies			
		Generated Comments			
		☑ <u>l</u> avadoc Comments			
		Source Code Hints			
		OK Cancel <u>H</u> elp			

# Figure 2

You can specify the options for code generation.



### Figure 3

The test class LoanTest is generated.

The automatically generated test class contains the methods <u>setUpClass()</u>, <u>tearDownClass()</u>, <u>setUp()</u>, and <u>tearDown()</u>. These are the test class life-cycle methods. These methods are invoked in this order: <u>setUpClass()</u>, <u>setUp()</u>, test methods, <u>tearDown()</u>, and <u>tearDownClass()</u>. You can implement these methods to set up classes and objects before the test and destroy the objects and remove the classes after the test, if necessary.

NetBeans automatically generates test methods for each method. You need to modify the code to test the methods in the class.