JUnit

By Martin Shueh

Overview

- What is JUnit?
- What is JUnit features?
- Written by Erich Gamma and Kent Beck
- Available for free at SourceForge

Installation

- Download junit.zip
- Windows:
  - Unzip junit.zip to a directory reffered as %JUNIT_HOME%
  - set CLASSPATH=%CLASSPATH%;%JUNIT_HOME%\junit.jar

- Unix(bash):
  - Unzip the junit.zip to a directory referred to as $JUNIT_HOME.
  - export CLASSPATH=$CLASSPATH:$JUNIT_HOME/junit.jar

How to write and run simple test

1. Create a Subclass of TestCase:

```
package junit;
import java.util.*;
import junit.framework.*;
public class SimpleTest extends TestCase {
```

2. Write a test method to assert expected results on the object under test:

```
public void testEmptyCollection() {
Collection collection = new ArrayList();
assertTrue(collection.isEmpty()); }
```

3. Write a suite() method that uses reflection to dynamically create a test suite containing all the testXXX() methods:

```
public static Test suite() {
return new TestSuite(SimpleTest.class);
}
```

4. Write a main() method to conveniently run the test with the textual test runner:

```
public static void main(String args[]) {
  junit.textui.TestRunner.run(suite());
} }
```

5. Run the test:

- To run the test with the textual test runner used in main(), type:
  - java junit.SimpleTest
  - Output:
    - . Time: 0
    - OK (1 tests)

- To run the test with the graphical test runner, type:
  - java junit.swingui.TestRunner junitfaq.SimpleTest
  - Green bar will be displayed as passing result.

Test Fixture

- useful if you have two or more tests for a common set of objects.
- avoids duplicating the test code necessary to initialize and cleanup those common objects for each test.

- To create a test fixture:
  - define a setUp() method that initializes common objects and
  - a tearDown() method to cleanup those objects.

Fixture example

```
public class SimpleTest extends TestCase {
   private Collection collection;
   protected void setUp() {
        collection = new ArrayList();  }
   protected void tearDown() {
        collection.clear(); }
   public void testEmptyCollection() {
        assertTrue(collection.isEmpty()); }
   public void testOneItemCollection() {
        collection.add("itemA");
        assertEquals(1, collection.size()); } }
```

The method will execute in the following order:

setUp()
testOneItemCollection()
tearDown()
setUp()
testEmptyCollection()
tearDown()

Suite

Use to run several tests at once.

Provided by JUnit as an object call TestSuite which runs any number of test cases together

```
import junit.framework.*;
public class AllTests {
public static Test suite() {
TestSuite suite = new TestSuite();
suite.addTest(new SimpleTest("TestEmptyCollection"));
suite.addTest(new SimpleTest("TestOneCollection"));
return suite; }
public static void main(String[] args) {
junit.textui.TestRunner.run(suite()); } }
```

TestRunner

There are two type of TestRunner:

1. Textual version
2. Graphical version

To start textual version type:

Java junit.textui.TestRunner class name

To start graphical version type:

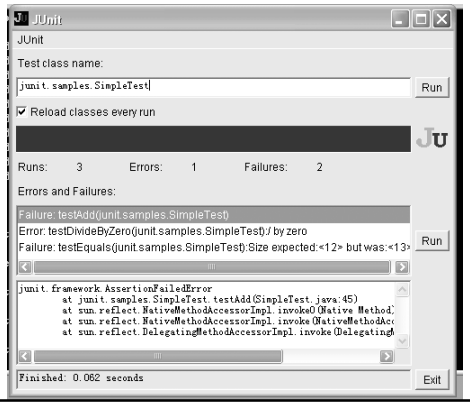
Java junit.awtui.TestRunner

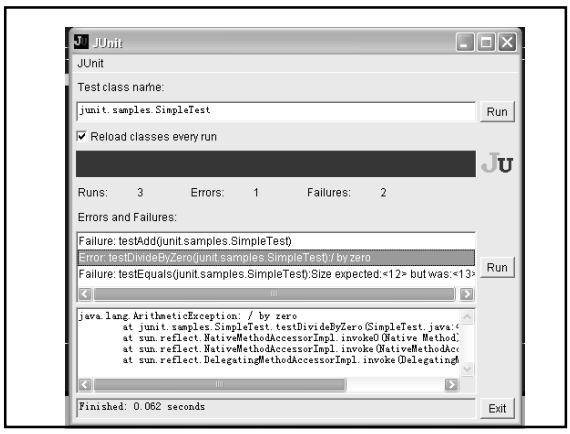
Graphical TestRunner

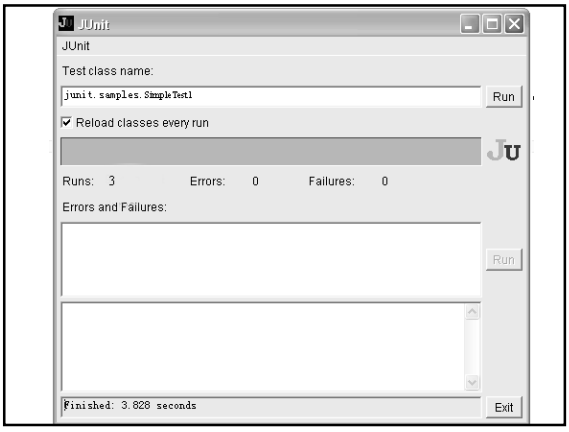
The graphical user interface presents a window with:

- a field to type in the name of a class with a suite method,
- a Run button to start the test,
- a progress indicator that turns from green to red in the case of a failed test,
- a list of failed tests.

Demo







Conclusion

JUnit is a very helpful for programmer.

It helps us to find errors quickly therefore save our time.

Use a fixture to save time if you have similar object.

Use suite to run some tests at once.

Use TestRunner to run and display the result

Sources

- www.junit.org
- www.google.com
- www.jguru.com