Lab: unit testing

In this lab, you will learn to implement classical Java unit testing in JUnit. You will use Eclipse's built-in JUnit implementation to write JUnit 4.X (or 3.X) test cases for Stutter.java. You will install EclEmma to facilitate generation of a code coverage report.

Code coverage is a common stopping criteria used in unit testing. The idea is: if you are executing every line of code at least once, you are probably making a reasonably good effort to find bugs. White-box testing is designed to achieve high code coverage. Concepts learned in this lab will serve as a foundation for a future lab on Android unit and integration testing.

Deliverables – none are due

- Link to your Stutter.java GitHub, including new commit with test cases
- Code coverage report demonstrating 100% coverage

Instructions

Setup:

1. Start Eclipse
2. Go to **Help → Install software**
3. In the wizard that appears, add the site http://update.eclemma.org/
4. From the list of available software, check EclEmma and install it.
5. When prompted, restart Eclipse. (**note**: Eclipse may be hidden on restart; click the Eclipse icon on the Windows taskbar to make it reappear)
6. Create a Stutter project – the Stutter.java code is on our course webpage.
7. Click **File->New->Other->JUnit Test Case.** Select **JUnit 4.X (or 3.X)** and call the class **StutterTest**.
8. Add a new test method to test the **Stutter.isDelimit** method. A test method has the annotation @Test above it. The method should call isDelimit with a delimiter and use org.junit.Assert.assertTrue to verify it returns true.
9. Add a second test method to test the **Stutter.isDelimit** method. The method should call isDelimit with a delimiter and use org.junit.Assert.assertFalse verify it returns false.
10. In the Package Explorer, right-click on **StutterTest**->**Run As->JUnit test**. Verify all tests pass.
11. Continue writing test cases until every Stutter method has at least one test.

Generate code coverage:

12. In the **Package Explorer** (JUnit is hiding it; look in the top left of the screen next to the JUnit summary of results), right-click on **StutterTest → Coverage As → JUnit test**
13. Open **Stutter.java** and notice now that the code is highlighted.
    a. Red code was not executed.
    b. Yellow code was partially executed (for instance: an if statement's condition was true, but never false).

    c. Green code was executed.
14. You will likely notice that it is tricky to cover the two lines in main() that read from System.in. You need a way to automatically send input to the console. A combination of System.setIn(InputStream in), ByteArrayInputStream, and String.getBytes() is sufficient to set the standard input to any string you would like. Use this combination to create two test cases to cover these lines.
15. Continue writing test cases until all lines are green.

Export the coverage report:

16. Go **to File->Export**
17. In the resulting dialog, expand **Run/Debug** and select **Coverage Session**.
18. Click **Next**.
19. In the resulting dialog, select your most recent coverage session. Select **HTML** as the format, and select a destination directory (put the results in a new directory called **emma**).
20. Click **Finish**.
21. Go to the **emma** directory you selected and open **index.html** to browse the results.
22. git commit -a && git push.

Resources

- JUnit 4.0 Assert: http://junit.sourceforge.net/javadoc/org/junit/Assert.html