Practical Unit Testing

JUnit

Atul Prakash
Unit Testing

• Software must be reliable for it to be useful

• You would not like a microwave oven or TV that keeps misbehaving. It could be even dangerous

• Same thing with software. Today, software controls your car, iPod, cellphone, your records at the university, etc. It is everywhere even though you don't see it.
Role of Testing in Development

• Typically, companies spend 50-60% of their software budgets on testing. More than on development.

• If bugs remain in the code after release, it is usually expensive to fix them.

• Bad publicity; releasing patches on a time cycle that is not under your control; legal liability; fixes may break other things; etc.
Unit Testing

• As you write each module (e.g., class), write the corresponding tests.
• If you find a bug in your code that is not uncovered by testing, add a test case as well that exhibits the bug first. That way, you will prevent the bug from creeping back again.
• Software changes all the time. Testing helps ensure that changes don't break old tests.
Testing we have seen so far...

- SpaceshipGameTest: we wrote that to test some of the functions provided by the game.
- Assignment 4:
  - SquareShipTest: test a square-shaped ship.
  - BirdOfPreyTest: test a ship of another shape
Making testing convenient

• Break your tests into small, independent tests. That way, you can say that your program passes \( m \) out of \( n \) tests. Fixes will be easier as well.

• What is it that you would like to see improved in the way we have been testing so far?
jUnit

- jUnit is a standard unit testing framework for Java.
- pyUnit: similar framework in Python
- cppUnit: a similar framework for C++
Sample Program and Test

- Download the zip file from Pragmatic Unit Testing book to a directory of your choice.
Add the files to jGrasp or Eclipse

- unzip the downloaded files to a directory.
- In jGrasp, go to Project -> New... and go to the directory containing the downloaded files. Create a project there called junittesting.
- Add the files "Largest.java" and "TestLargest.java" to the junittesting project.
```java
@Requires("list != null")
@Effects(post = "result is the largest number in the list")
public static int largest(int[] list) {
    int max = list[0];
    for (int i = 1; i < list.length; i++) {
        if (list[i] > max)
            max = list[i];
    }
    return max;
}
```
Take a few minutes ...

- Read through the tests that are there in the TestLargest.java. Understand what they are doing.
- Largest takes an int array and finds the largest value in that array
import junit.framework.*;

public class TestLargest extends TestCase {

    public TestLargest(String name) {
        super(name);
    }

    public void testOrder() {
        assertEquals(9, Largest.largest(new int[] {9,8,7}));
        assertEquals(9, Largest.largest(new int[] {8,9,7}));
        assertEquals(9, Largest.largest(new int[] {7,8,9}));
    }

    public void testEmpty() {
        try {
            Largest.largest(new int[] {});
            fail("Should have thrown an exception");
        } catch (RuntimeException e) {
            assertTrue(true);
        }
    }

    public void testOrder2() {
        int[] arr = new int[3];
        arr[0] = 8;
        arr[1] = 9;
        arr[2] = 7;
        assertEquals(9, Largest.largest(arr));
    }
}
JUnit Tests

- Class inherits from TestCase
- Constructor needed as given.
- It provides a few new methods:
  - assertEquals(value, expression)
  - assertTrue(expr)
  - assertFalse(expr)
  - fail(msg)
- Each test run till it either completes or an assertion fails
Assertions

- Assertions are simply a condition that you expect to be true at a given point in the program, based on its specs:
  - We saw:
    - assert expression
  - If the expression is false, the program quits.
jUnit Assertions

- jUnit provides a few additional ways of expressing assertions.
  - assertTrue(expr) is similar to:
    - assert expr
  - assertFalse(expr) is similar to:
    - assert !expr
  - assertEquals(val, expr) is similar to:
    - assert val == expr
  - fail(msg) is similar to:
    - assert false
    - But it also prints a msg.
Example Test

```java
48    public void testOrder2() {
49        int[] arr = new int[3];
50        arr[0] = 8;
51        arr[1] = 9;
52        arr[2] = 7;
53        assertEquals(9, Largest.largest(arr));
54    }
```
Equivalent test

```java
48    public void testOrder2() {
49        int[] arr = new int[3];
50        arr[0] = 8;
51        arr[1] = 9;
52        arr[2] = 7;
53        assertTrue(Largest.largest(arr) == 9);
54    }
```
JUnit: Pros

• Unlike our tests so far, JUnit is more flexible. If one test fails (because an assertion failed), it continues to the next test.

• No main required. JUnit provides one in its junit.textui.TestRunner class.
Digression: Java Code Organization

• Java provides a way to organize code
  • Not use a package: Default package
  • Using packages
Packages

- Naming packages
  - The package names matches the directory name and can be hierarchical
  - java.util has classes stored in
    - [path-to-java-packages]/java/util
Declaring your file to be within a package

- At the top of your source file, you add
  ```java
  package edu.um.eecs282.hello;
  
  class HelloWorld {
      // ...
  }
  ```

- For this example
  - File name = HelloWorld.java
  - Class name = HelloWorld.class
  - Location = [path-to]edu/um/eecs285/hello
CLASSPATH

- jar files: Packages in Java are often compressed into a jar file. E.g., junit-4.5.jar.
- To extract (you do not need to though), you can use the jar command:
  - jar xf junit-4.5.jar
- Java uses an environment variable called CLASSPATH to find jar files or other packages.
- Format:
  - dir1;dir2…;dirN on Windows
  - dir1:dir2:…:dirN on Unix/Mac OS.
jGrasp and jUnit

• You need to tell jGrasp where the jUnit package is.

• Go to Settings -> Path/ClassPath -> Project ...

• Add the directory containing junit's jar file to the path.
jGrasp and jUnit

- jUnit provides its own main method for running tests. From command-line (provided your CLASSPATH includes junit's jar file):
  
  - `java junit.textui.TestRunner TestLargest`

- Tell jGrasp to use that:

  - Settings->Compiler Settings -> Project <unittesting>
• Select the Flags/Args tab under Compiler
Modify FLAG2 for Run
Running the tests

- Compile the project and run TestLargest
- You should see the list of failed tests and the line number
```java
public static int largest(int[] list) {
    int max = list[0];
    for (int i = 1; i < list.length; i++) {
        if (list[i] > max) {
            max = list[i];
        }
    }
    return max;
}
```
Eclipse

- Eclipse is much easier. Simply select junit version 4 if you are given the choice (or the latest version) anywhere.

- File->Import -> General -> FileSystem

-
Select
Import resources from the local file system into an existing project.

Select an import source:

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Preferences

- CVS
- Run/Debug
- Tasks
- Team
- XML
Select From directory
Next steps

- Click on the directory "utj". You should see the files on the right side. Deselect all and select Largest.java and TestLargest.java.
Fix errors

- TestLargest.java may show errors. Hover your mouse over the word "junit" and select a fix to reorganize the project. You may have to add junit3 to the project in Eclipse.
Run the test

- Run the TestLargest as a JUnit test. You should see a window showing the failed tests. Clicking on the failed line should take you to that line in TestLargest.

- Select Project -> Run, when you have selected TestLargest.
Excerpted from the book, "Pragmatic Unit Testing in Java with JUnit"

```java
import junit.framework.*;

public class TestLargest extends TestCase {
    public TestLargest(String name) {
        super(name);
    }
}
```

Failure Trace

```
junit.framework.AssertionFailedError: expected:<9> but was:<2147483647>
    at TestLargest.testOrder(TestLargest.java:22)
```
Common Next Steps

• Read through your test and the code to see why the test failed.

• Set breakpoints and use a debugger if necessary to help identify the problems.

• For jGrasp, you will have to change the FLAG2 settings for the Debugger.