CS 2334: Lab 2
Unit Testing

Notes

- Rubric for each lab and project tells you what we are specifically looking for when we are grading your assignments

Specification to Working Implementation

A full specification tells us:

- Method prototypes
- Defines the type, meaning and expected values of the input parameters
- Defines the type, meaning and expected value of the return value
- Defines any side-effects (changes to the object on which the method was called)

Specification to Working Implementation

Implementation:

- Translation of the full specification into the code that makes it happen
- Once the implementation is complete, we are not done – we have to convince ourselves (and others) that the implementation behaves according to the specification

Unit Testing

- Formal process of testing our code base
- Create a set of test methods
  - Each method tests one or more aspects of our code
  - Provide a known set of inputs into a method
  - Checks the output from the method against an expected value
  - Checks the side effects (usually by calling other getter methods)

Unit Testing

The set of test methods should (together) verify that each part of our code is working properly

- Every “if” and “else” branch
- Every for/while loop
- Every case of a switch
- Every class

It feels like a lot of work, but it can dramatically improve the quality of our code

Unit Test Class

- A single unit test class contains a set of test methods
- Each test method includes:
  - Creating objects and/or initialization of primitives
  - Exercising of a set of methods
  - Comparing the results of the above method calls against expected values. This process is called *Assertion*

An Example

```
import org.junit.Test;
import org.junit.Assert;

public class IntTest {
	@Test
	public void test1() {
		int a = 5;				// Initialization
		int b = 37;

		int c = a + b;				// Use addition operator

		Assert.assertEquals(c, 42);	 // Fails if not true
	}
}
```

Example II

```
import org.junit.Test;
import org.junit.Assert;

public class DoubleTest {
	@Test
	public void test2() {
		double a = 5;				// Initialization
		double b = 37;

		double c = a + b;				// Use addition operator

		// The two values must be within 0.0001 of each other
		Assert.assertEquals(c, 42.0, 0.0001);  // Fails if not true
	}
}
```

Example III

```
import org.junit.Test;
import org.junit.Assert;

public class StringTest {
      @Test
      public void test3() {
            String foo = "Foo";          // Create object

            foo += "Bar";                // Modify object

            // Test result of operation
            Assert.assertTrue(foo.equals("FooBar"));
    }
}
```

Assert Class

For a more detailed discussion of the Assert class, see:

http://junit.sourceforge.net/javadoc/org/junit/Assert.html

(linked from the lab2 description)

Lab 2

Lab 2 consists of two key classes:

- Fruit objects store the name, weight and price of one type of fruit
- FruitBasket objects store a group of Fruit objects
  - Can ask about the total weight and cost of items in the basket
  - Can ask about the total weight and cost of items matching a particular name

Lab 2 Preparation

- Download lab2.zip
- Import into your Eclipse project

(details of how to do this are in the lab specification)

Demonstrate import

Unit Test Creation

Demonstrate unit test creation for Fruit class (FruitTest2)

Demonstrate testing of name, weight and price – at least two test methods

Lab 2 Requirements

- Produce a unit test class for FruitBasket
  - This class will have a set of test methods
  - Must test all aspects of the FruitBasket class
- As part of the testing process, you will discover a set of bugs in the FruitBasket class
  - Fix these bugs
  - All of your tests must pass
  - Note: we will test your code with our own unit tests. So, you must design your tests carefully to make sure you don't miss anything

Demonstrate: examine FruitBasket class and discuss specifications for a couple of the complicated methods

Submission

- Submit only one file: lab2.zip (casing matters)
- Due date: Friday, September 4th @11:59pm
- Submit to lab2 dropbox on D2L