**THE SETUP**

**Source:** https://github.com/8anksy/Spr16comp474p1.git

**Language:** Java

**Using:** JDK 1.8 and JUnit 4

**IDE:** Eclipse - Mars.1 Release (4.5.1)

In order to run this project you will need a Java IDE of your choosing. Clone the code from the GitHub source above. Run the program from the Main.java entry point with no parameters. The code will compile and test print out of a bill will show followed by results of the 3 test classes that were run.

**TESTS**

**TESTING OVERVIEW**

We started by testing our basic objects (Customer.java and ShoppingCart.java) to ensure that these components functioned correctly prior to being used in other methods. Afterwards, tested our more complex class, Register.java. We used equivalence partitioning to test the shopping cart size and the discounts based on cart size. We also have full conditional coverage for each combination of membership and tax status for the Customer object.

We had rounding error bugs initially that were discovered by testing for Register totals. We decided to implement BigDecimal objects to help with rounding to the nearest cent. We formatted these objects to use the *ROUND\_HALF\_EVEN* rule to two decimal places.

There are 3 test classes that test each of our objects (Customer.java, ShoppingCart.java, and Register.java). Their test methods are explained below:

**CustomerTests.java**

+ public void testMembership()

This method uses the constructor from the Customer.java class and tests that the membership parameters are assigned accordingly by using the getter methods.

+ public void testTaxExemption()

This method uses the constructor from the Customer.java class and tests that the membership parameters are assigned accordingly by using the getter methods.

**ShoppingCartTests.java**

+ public void CartMaxTest()

Tests the functional requirement that the cart size shall not exceed 50 items. The test starts with an empty cart and attempts to add 52 items to the cart, one by one. The test asserts that the cart size remains at 50 when it is capped.

+ public void TestCartSize()

Testing that our cart size is reported accurately. Starts with an empty cart, adds items, then removes items. Checks for expected cart size after each.

**RegisterTests.java (Each test starts with a ShoppingCart of size 4)**

+ public void TestCalcRawTotal()

Asserts that the subtotal of all items in the cart are equal to our expected value

+ public void TestWrongItem()

Tests that our cartSize is reported accurately when an item is in the cart that is not contained in the database.

+ public void TestCalcNonMemberDiscount()

Tests the calculated totals (before tax) for non-member customers. Also covers the “10 items or more” partition (9 items, 10 items, 11 items).

+ public void TestCalcMemberDiscount()

Tests the calculated totals (before tax) for non-member customers. Also covers the “more than 5 items” partition (4 items, 5 items, 6 items).

+ public void TestTaxTotal()

Tests the calculated totals after tax. Test coverage includes both exempt and non-exempt Customer types.

+ public void TestPrint()

Test output of the display of (receipt/tab/bill). Should contain the list of items, their prices, discounts, tax amount, and final total.