# SQA Assignment 5 - Spring 2016

**Due: Fri 22 Apr 2016 by 11:59 p.m. Assignment 5A, Web-CAT (your source and tests)**
**Wed 27 Apr 2016 by 11:59 p.m. Assignment 5B, Web-CAT (your source and tests)**
**The 5B submission will also be tested against reference tests.**

## Problem Description:

The objective of this assignment is to introduce you to the JUnit testing framework. In Assignment 4, you created numerous boundary value test cases based on a specification for a software module that calculates taxes. In this assignment you are to implement a module in Java and convert your test cases to JUnit tests. It is recommended that you use either Eclipse or jGRASP as the IDE for writing the Java source file and the corresponding test file. I will be using jGRASP in class which includes a plug-in for JUnit. For more information, see Using JUnit with jGRASP or at jGRASP web site (jgrasp.org) see "JUnit" under Tutorials.

**Tasks**

1. Based on the description in Assignment 4, write a Java class called TaxCalculator that includes a calculateTax method with the following signature:

   ```
   public int calculateTax(int status, int income)
   ```

   where *status* is 1 for single and 2 for joint, *income* is the reported income, and the return value is the calculated tax. If either of the arguments is invalid (e.g., a status of 0 or income of -10), a valid tax cannot be calculated so the method should abandon the calculation by throwing an `IllegalArgumentException` that includes a message describing the problem. For example, invoking calculateTax(0, 75000) should throw the following:

   ```
   new IllegalArgumentException("Status should be 1 or 2.")
   ```

   ***Note**: In keeping with IRS practices, we are working with integers. However, the calculations in Java will be done with double floating point arithmetic since income is multiplied by a double (i.e., the tax rate) which results a double. Be sure to "round" the result using Math.round() and then convert this to an int before returning the final result. Your test cases should catch mistakes in rounding.*

2. After you have created the TaxCalculator class, you should create a jGRASP project and add the source file to it. After successfully compiling TaxCalculator, use the **Tools > JUnit** menu or the JUnit buttons on the toolbar to create the corresponding test file, TaxCalculatorTest. If your TaxCalculator class is in the open project, the TaxCalculatorTest class will automatically be placed in the project. Otherwise, you can add the test file to the project by dragging it from the Browse tab and dropping in the project file in the Open projects pane.

3. Convert each of your test cases from Assignment 4 to JUnit test methods in TaxCalculatorTest. After each test method is added, compile and run the test file with JUnit, then make changes as appropriate. Keep in mind that Java's strong type checking and the int parameters in the calculateTax method may require you to adjust some of

your previous test cases. For most of your test cases, you will need to specify the expected result and actual result, where the actual result is the result of invoking calculateTax with the test input values for status and income. You should include the *status* and *income* for the test case in the optional first parameter (the error message) of the JUnit assert methods. Then when a test fails, JUnit will report the status and income along with the expected and actual results. Note that the test cases with invalid input values for status or income should pass if the calculateTax method correctly throws `IllegalArgumentException`.

4. As you construct the test cases and run them with JUnit, chances are that you will encounter some failures. A failure indicates that there is a defect in either the calculateTax method or in the test method. If all of your test cases pass, I recommend that you add a defect of two to calculateTax, and then run JUnit to make sure the appropriate test fails with a useful message. You can also add defects to your test methods to see them fail correctly.

## Deliverables

When you are confident that your TaxCalculator class and TaxCalculatorTest class are working properly, you are ready to submit them to Web-CAT, a web-based automatic testing and grading system. If you did not receive a Web-CAT password via email, you will need to login to Web-CAT (http://webcat.eng.auburn.edu) with your AU username and click "Forgot your password" to have it send you a link to updated your password.

You can submit a ZIP file containing your two .java files to Web-CAT via your browser (http://webcat.eng.auburn.edu). After you login to Web-CAT, on the "Home" page, you should see a list of assignments accepting submissions. Find **COMP 5710 Assignment 5A - JUnit,** click the "Submit" button, then "Browse" to locate the ZIP file containing your files and upload it. After 5 to 20 seconds you should see the results from Web-CAT. [Note that after Fri 22 Apr, you'll be submitting to **Assignment 5B – JUnit.**] In addition to running your test file, Web-CAT will run a set of reference tests against your program. If these tests turn up any defects, you have until Wed 27 Apr to correct you program and resubmit to Web-CAT.

Alternatively and recommended, you can submit your files directly from jGRASP. Click **Tools > Web-CAT > Configure** on the main menu, then copy/paste the following as the "Assignment Definition URL" and click OK:

http://webcat.eng.auburn.edu:8080/Web-CAT/WebObjects/Web-CAT.woa/wa/assignments/eclipse

[Note: Do not copy the link above into a browser; this is simply a link to an XML file that describes the assignments that are available to you for submission via Eclipse and jGRASP.]

After jGRASP has been configured for Web-CAT, you should have Web-CAT Submit button on the main tool bar  . If your .java files are in a jGRASP project, the Web-CAT Submit button  will be on the Open Projects tool bar in the Browse tab. After clicking the submit button, a dialog will pop up and allow you to submit files to Web-CAT from jGRASP.