Name: _____

Project 3, Part 2

White-box Unit Testing and Integration Testing

This is a GROUP assignment.

Due date:
This part of the assignment is due on FRIDAY, July 29, 2016 at 12:00pm (that's NOON, not midnight!).
No submissions will be accepted after the deadline.

The purpose of this assignment is to have you learn and practice the following:

1. Writing software methods and classes that conform with specifications/requirements.
2. Ability to analyze and create white-box unit tests for methods and classes based upon specifications/requirements and analysis of source code.
3. Ability to perform multiple methods of software integration testing.
4. Ability to employ automated unit testing tools.

Your assignment:
Your group will correct any errors in your classes that were detect in previous black-box testing, perform white-box unit testing, and then perform integration testing of the previously described software project.

Although this is a group project, the classes involved in this project were/are written INDIVIDUALLY and the source code will NOT be shared with the other members of the group except as explicitly specified in the description below. Instead, individual group members will be assigned to code specific classes, according to the specifications below, and will share ONLY the BYTECODE file (i.e. the .class file and not the .java source file) with group mates, who will then perform black-box unit testing. This will prevent group members for seeing the implementation of the module for which they will design black-box test cases or perform integration testing and will more closely simulate a real-world testing experience.

SUB-PART	ONE

For this portion of the assignment, you will correct your source code based on the unit testing done previously by your colleague:

Steps:

1. Obtain the black-box test results your colleague performed on your class file in the previous part of this assignment.
2. Correct your source code based on any failed tests that were reported to you.

   PLEASE NOTE that you are NOT supposed to perform any additional unit testing on your own class right now. Instead, correct the mistakes of which you have been made aware, perform some basic and simple run testing as you did before, and then stop. You will perform your own unit testing below, but don't do it right now.

SUB-PART	TWO

For this portion of the assignment, you will share your SOURCE CODE file (the .java source code file this time) with another member of your group, who will the perform white-box testing.

Steps:

1. Give your file to the remaining member of the group – that is, the member who did NOT previously perform the black-box testing. *(In the case of two-person groups, the other group member will of course be the same).*
2. Compile and white-box test the .java file that you received. You may employ any or all of the White-box testing techniques we discussed in class, but at a minimum you must:
   a. Create a UML activity diagram (i.e. flow graph) for any non-trivial method in the class you are testing (you should generally choose the more complex method).

b. Determine the list of linearly independent paths in the graph and list them. Are these a basis set? If not, what is the basis set?
c. Determine a set of test cases to cover each of the linearly independent paths.
d. Perform statement or branch level coverage for all other methods of the class under test.
e. Share your test results with the author of the class, and give him/her a chance to make corrections before moving on to sub-part three.

SUB-PART THREE

Each member of the group will now perform integration testing, as follows:

Group member whose last name is FIRST alphabetically:
Top-down integration testing *(using the driver file to be provided by the instructor).*

Group member whose last name is SECOND alphabetically:
Bottom-up integration testing.

Group member whose last name is LAST alphabetically:
Sandwich integration testing.
(If the group has only two members, do not perform sandwich testing)

When performing this integration, you will obviously have two or more iterations of integration testing. You should product a PDF that details the following:

1. Each phase of integration testing, indicating the objects that are included and any stubs/drivers that are employed. Include an illustration of this for each phase.
2. A report of integration testing errors found at each phase, the corrective action employed to fix these errors, and the (hopefully) repeat of the integration testing phase with successful tests.

ASSIGNMENT SUBMISSION

Assignment Submission:
You must submit this part of the assignment on D2L as a single ZIP file by the specified date and time. Your ZIP file should contain:

1. All ClassName.java files.
2. All ClassName.class files. (just in case I can't compile them myself for some reason!)
3. A .zip file called ClassName_WhiteBoxTesting.zip that contains all artifacts used in performing the white box tests described in sub-part two above. This, at a minimum, should include:
   a. A PDF containing the activity diagram/flow graph, listing of paths, and test cases for those paths. Call this file ClassName_FlowGraph.pdf.
   b. A copy of the test harness for this class. If you have embedded it in the main method of the class itself, that's fine. If you have it in a separate file, name that Class and file ClassName_WhiteBox.java.
   c. Any supporting stubs or drivers you have created.
4. Two or three zip files called TopDownIntegration.zip, BottomUpIntegration.zip, and (if required) SandwichIntegration.zip that contain the artifacts in sub-part three above.
5. Any other supporting documentation or material necessary for the instructor to grade this assignment.

You MUST name your file in the following manner:

`Group#_Project3_Part2.zip`

BONUS SUB-PART

You may NOT perform this step until AFTER sub-part two of this project has been completed and you have received the white-box test results from your colleague. For a SIGNIFICANT and INDIVIDUAL bonus, perform unit testing on your own class using JUnit Test.

Perform as thorough an exhaustive testing as you can using JUnit test, indicate the passing and failing test cases, and the corrective action. Assemble a written report of the JUnit testing, any necessary artifacts or generated test code, etc., and include it as a ZIP file named:

ClassName_JUnitTest_LastName.zip

Where "LastName" is the name of the INDIVIDUAL who did this testing.