4321/5321 Homework 5

Spring 2020

Question Weighting:

Question 1-5 - 20% each

Use the following approach for each problem **(all test must use the JUnitParams runner and read values from csv files).**

1. Develop CFG (reduced) and cyclomatic complexity.
2. Develop basis path set.
3. Determine significance on each variable.
4. Develop inputs and expected outputs from requirements, not code.
5. Add tests for missing Boundary Values not tested, including extreme range values.
6. Add tests for extreme range values for each variable that has a BV.
7. Add MCDC test cases for Multiple Condition Decision statements.
8. Add test cases to verify all table data.

**Submit the following in the PDF file - this is the evidence file**

1. Test case table snapshot
   1. Basis Path test cases (for problem 2-3 only)
      1. Use the line numbers in Eclipse for your basis path line.
      2. where tests are addition to basis path set use a "-" to indicate the basis path.
      3. Make sure all true is the first BP and tests are in correct order
      4. You do NOT need to submit the CFG with this homework
   2. Indicate tests for MCDC coverage with a "Statement 25 TFFT" - put this in the comment column. Note these the conditions are in the order for that statement.
2. JUnit pass indicator (green bar expanded)
3. JaCoCo statement green source line annotations (not summary)
4. PIT coverage annotation for the source method under test (all green lines) - use snipping tool. You must set PIT to **All mutators** to get credit.
5. Make sure to include the time stamp on your screen shots.

**Include in ZIP file**

1. Your evidence file (JaCoCo/JUnit screenshot)
2. JUnit test files (make sure problem number is referenced in the file name)
3. csv files used (make sure problem number is referenced in the file name)
4. For Problem 5 your Problem4ServerData.class file
5. For problem 3 and 4
   1. modified code files.

**Problem 1**) Test the Problem 1 source code (in the attached zip file). This problem implements problem 2 from HW 3 but with a very complex series of multiple condition decision statements. This problem demonstrates the usefulness of a good unit test for refactoring the code. You are to run the JUnit test, get complete JaCoCo coverage and get PIT all green for the method under test. This code should pass the tests without any changes required. You are simply using the existing tests to aggressively refactor the code.

**Test**

Use the test case table provided. You will be able to use the JUnit file from HW 4 - you may have to change the location of the csv file.

**Code**

Attach the corrected code as part of your submission. Explain what code changes were made and where (original source code statement numbers from Eclipse).

**PIT**

You must run PIT (PITClipse latest version) against your JUnit test file and get all green for the method under test (don't care about getters/setters). You must set PIT to **All mutators** to get credit.

**Problem 2**) Test the Problem 2 source code (in the attached zip file).

**Assumptions**

1. cart ranges from $0.00 to $10,000.00 both inclusive.
2. For result use Excel's currency format and **do not truncate**.
3. bonusPoints ranges from 0 to 999 both inclusive
4. use taxRate = 8.25%

**Test (80%)**

Use the following template for the test case table. Document how the multiple condition expression is tested using MCDC in the Comments column (see the Test case table below) and as described above. The comments column should also describe the basis path set.



Use the same inputs as shown above **only changing the value of cart** for the following test cases:

1. Basis path set
2. Missing BVs from the Basis path set
3. Extreme range values (for bonusPoints extreme ranges test cases use the cart value from the test case table above).

Test case design

1. The comparison threshold for result is 0.005 in JUnit
2. You must use Excel to compute the value of result in your table. Please see the test case tables from HW 3 to see how to use this function.

**PIT (20%)**

You must run PIT (PITClipse latest version) against your JUnit test file and get all green for the method under test (don't care about getters/setters). You must set PIT to **All mutators** to get credit.

**Problem 3**) Test the Problem 3 source code (in the attached zip file).

**Assumptions**

1. total ranges from $0.00 to $10,000.00 ft both inclusive. Significance of $0.01
2. For result use Excel's currency format and **do not truncate**.

**Test (80%)**

Use the following template for the test case table. Document how the multiple condition expression is tested using MCDC in the Comments column (see the Test case table below) and as described above. The comments column should also describe the basis path set.



Use the same inputs as shown above only changing the value of total for the following test cases:

1. Basis path set
2. Missing BVs from the Basis path set
3. Extreme range values

Test case design

1. The comparison threshold is 0.005 in JUnit
2. You must use Excel to compute the value of result in your table.
3. Make sure to properly test the multiple condition decision statement - you will get all Green from PIT even with an improper MCDC test. Review M06 MCDC expressions to determine what to do here.

**PIT (20%)**

You must run PIT (PITClipse latest version) against your JUnit test file and get all green for the method under test (don't care about getters/setters). You must set PIT to **All mutators** to get credit.

**Problem 4**) Test the Problem 4 source code (in the attached zip file). This problem implements the gas pump state diagram from HW 1 Problem 6 and uses a modified test case table (attached in Canvas) from HW 2 problem 1. This code has errors - you are to get the code to pass, get complete JaCoCo coverage (except the switch statement) and get PIT all green for the method under test.

**Assumptions**

1. This code has errors - you are to get the code to pass, get complete JaCoCo coverage (except the switch statement) and get PIT all green for the method under test.

**Test**

Use the test case table provided. Do NOT change test case values.

**Code (80%)**

Attach the corrected code as part of your submission. Explain what code changes were made and where (original source code statement numbers from Eclipse).

**PIT (20%)**

You must run PIT (PITClipse latest version) against your JUnit test file and get all green for the method under test (don't care about getters/setters). You must set PIT to **All mutators** to get credit. You will need to do something to the JUnit test method to help it get all green. No other clues will be given and only solutions that make test sense will be accepted (no hodge-podge solutions).

**Problem 5**) Use your test case table from problem 3.

Use EasyMock to mock the call to the server to get total.

EasyMock instructions

* + 1. Download EasyMock from the M14 Blackboard files
    2. InstallEasyMock in your project (add the EasyMock.jar to your Java Build path)

Execute the JUnit test. Create **Problem5ServerData.java** to define the signature for the getTotal() method (see slide 32 of M14 for how to do this). Follow the five steps shown in slide 35 of M14 to get EasyMock to work in your test environment.

**PIT** is not required.