Homework 5

Summer 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 2,4, and 5 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 and PIT screenshot)
2. Show snapshot of all mutators selected (PIT) from Window -> Preferences -> Pitest -> Mutators
3. JUnit test files (make sure problem number is referenced in the file name)
4. csv files used (make sure problem number is referenced in the file name)
5. For Problem 5 your Problem5ServerData.class file
6. For problem 3
   1. modified code files.

**Problem 1**) Test the Problem 1 source code (in the attached zip file).

**Assumptions**

1. use a significance of 0.1 on all doubles
2. speed is a double and ranges from 0 to 135 mph both inclusive.
3. distance is a double and ranges from 0 to 1,000 feet both inclusive

**Test** (80 % credit)

Use the following template for the test case table.



**PIT** (20% credit)

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).

This code is used to assist a small business in keeping count of boxes shipped. This company uses the railroad exclusively to ship their merchandise. A shipment is a set of 10 railroad cars each of which may hold either 360, 380, 400, 420, or 430 boxes. A shipment is a total of 4,000 boxes.

The railroad cycles through the 10 cars as represented by the following box capacity schedule:

Schedule 1: 380, 400, 420, 430, 380, 400, 430, 420, 380, 360

Schedule 2: 380, 420, 400, 430, 380, 400, 430, 420, 380, 360

Schedule 1 is used normally. Schedule 2 is only used when the shipment number is not divisible by 5 or when the shipment number is divisible by 10 but not by 25.

Given the box number in the car, the railroad car number (1-10) and the shipment number the software calculates the **absolute** box number of the **previous** box in the 4,000 box shipment.

Examples



**Assumptions**: Box In Car Number, RailRoad Car number, and Shipment Number are all positive integers. Assume Shipment Number has a maximum of 100,000 for testing purposes.

Does the code work? If so indicate that it does - if not update the code and retest. If you provide a code update you must attach that as part of your homework submittal.

**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 ShipmentNumber=1 for the first test case and ShipmentNumber=5 for the second.

**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 may ignore the one surviving mutant "Substituted 9 with 10 → SURVIVED" for statement 6.

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

This problem implements the binocular state diagram from HW 1 Problem 6 and uses the 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).

**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 (20%)**

Use the test case table from HW 2 Problem 1. 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 (0%)**

**Do NOT run PIT against this code.**

**Problem 4**) Test the Problem 4 source code (in the attached zip file).

**Assumptions**

1. Cart ranges from $0.00 to $20,000.00 both inclusive.
2. Use Excel's built in rounding to the Cent. Significance is to the Cent.

**Test (80%)**

Use the following template for the test case table.



Note: for the multiple condition decision statement MC/DC test cases you **MUST** use the masking solution.

**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 5**) Test the Problem 5 source code (in the attached zip file) using the test cases provided.

**Test (80%)**

The test cases provided must be modified to correctly test the code to get JaCoCo all green for the source method under test. You may **ONLY** change the test cases. The supplied test case table uses Excel VB. Please determine how best to enable the VB code for your use in developing and/or executing tese tests.

**Assumptions**

1. Cart ranges from $0.00 to $20,000.00 both inclusive.
2. Use Excel's built in rounding to the Cent. Significance is to the Cent.

**EasyMock**

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

EasyMock instructions

* + 1. Download EasyMock from the M14 Canvas 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 getCart() 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 (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.