5321 Homework 5
Fall 2019

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
    a. Basis Path test cases (for problem 1-2 only)
        i. Use the line numbers in Eclipse for your basis path line.
        ii. where tests are addition to basis path set use a "-" to indicate the basis path.
        iii. Make sure all true is the first BP and tests are in correct order
        iv. You do NOT need to submit the CFG with this homework
    b. Indicate tests for MCDC coverage with a "Statement 11 FFT" - put this in the comment column.
2. JUnit pass indicator (green bar expanded)
3. JaCoCo statement green source line annotations (not summary)
4. 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 4 your Problem4ServerData.class file
5. For problem 5
    a. both PIT html files
    b. Text explanation of PIT coverage analysis of the provided tests

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

Assumptions

1. cart ranges from $0.00 to $20,000.00 both inclusive.
2. For result use Excel's currency format and do not truncate.
3. memberPoints ranges from 0 to 10,000 both inclusive
4. items ranges from 1 to 50 both inclusive
5. use taxRate = 8.25%

Test
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.

Test Case	**Inputs**						**Exp Out**		
Number	**cart**	**coupon**	**memberPoints**	**items**	**member**	**taxRate**	**result**	**bPath**	**MCDC**
1	$5,000.01	TRUE	1,000	10	FALSE	8.25%	$4,059.38	8-9-22	TTFF

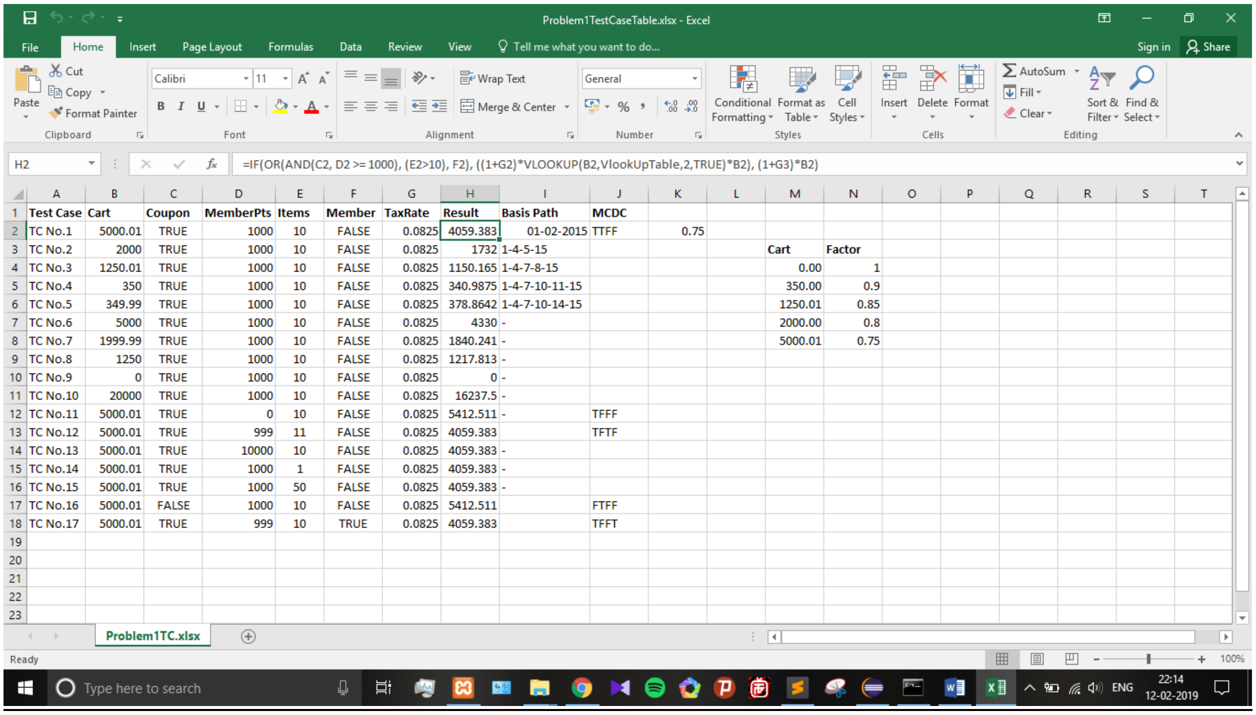
Use the same inputs as shown above **only changing the value of cart** for the following test cases:

1. Basis path set
2. Missing BVs from the Basis path set
3. Extreme range values (for memberPoints and items extreme ranges use cart value from above).

Test case design

1. The comparison threshold for result is 0.006 in JUnit
2. You must use VLOOKUP to compute the value of result in your table. Please see the test case table for problem 2 from HW 3 to see how to use this function. Note that the function references another tab that has the BVs for each ECP with the corresponding values of discount for each.

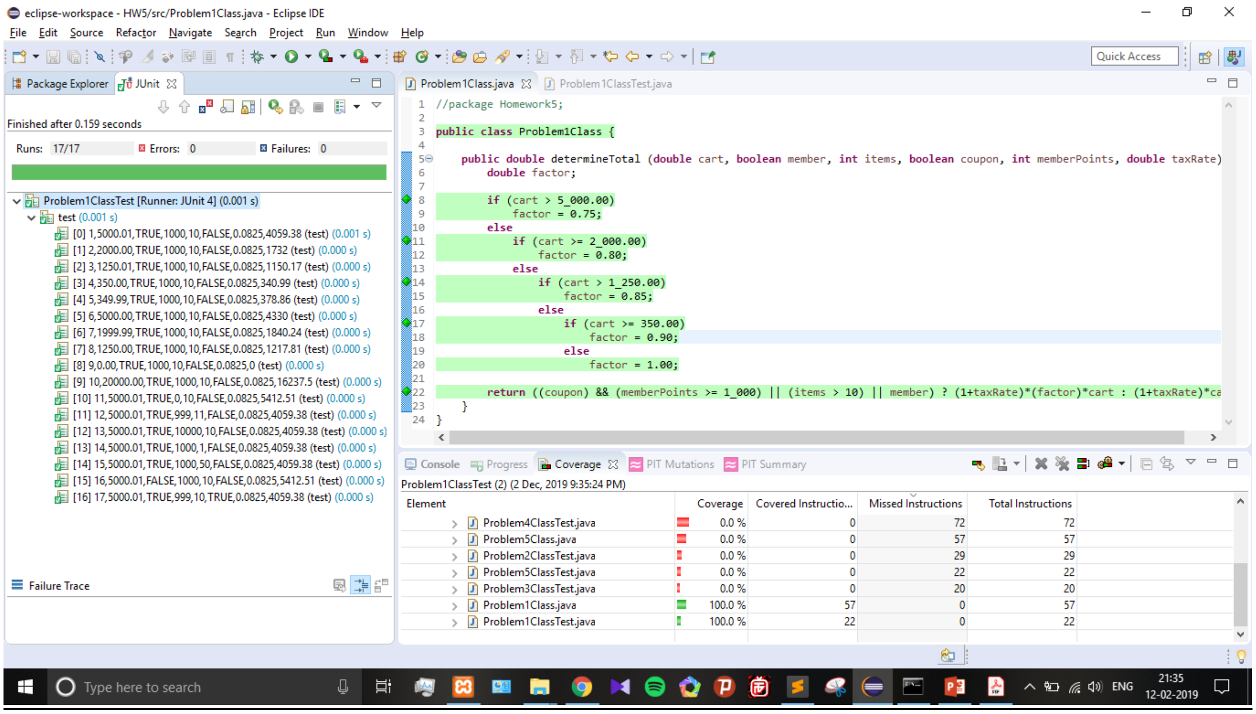
TEST CASE TABLE: (with **VLOOKUP**)



H2 =IF(OR(AND(C2, D2 >= 1000), (E2>10), F2), ((1+G2)*VLOOKUP(B2,VlookUpTable,2,TRUE)*B2), (1+G3)*B2)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Test Case	Cart	Coupon	MemberPts	Items	Member	TaxRate	Result	Basis Path	MCDC				
2	TC No.1	5000.01	TRUE	1000	10	FALSE	0.0825	4059.383	01-02-2015	TTFF	0.75			
3	TC No.2	2000	TRUE	1000	10	FALSE	0.0825	1732	1-4-5-15				Cart	Factor
4	TC No.3	1250.01	TRUE	1000	10	FALSE	0.0825	1150.165	1-4-7-8-15				0.00	1
5	TC No.4	350	TRUE	1000	10	FALSE	0.0825	340.9875	1-4-7-10-11-15				350.00	0.9
6	TC No.5	349.99	TRUE	1000	10	FALSE	0.0825	378.8642	1-4-7-10-14-15				1250.01	0.85
7	TC No.6	5000	TRUE	1000	10	FALSE	0.0825	4330	-				2000.00	0.8
8	TC No.7	1999.99	TRUE	1000	10	FALSE	0.0825	1840.241	-				5000.01	0.75
9	TC No.8	1250	TRUE	1000	10	FALSE	0.0825	1217.813	-					
10	TC No.9	0	TRUE	1000	10	FALSE	0.0825	0	-					
11	TC No.10	20000	TRUE	1000	10	FALSE	0.0825	16237.5	-					
12	TC No.11	5000.01	TRUE	0	10	FALSE	0.0825	5412.511	-	TFFF				
13	TC No.12	5000.01	TRUE	999	11	FALSE	0.0825	4059.383		TFTF				
14	TC No.13	5000.01	TRUE	10000	10	FALSE	0.0825	4059.383	-					
15	TC No.14	5000.01	TRUE	1000	1	FALSE	0.0825	4059.383	-					
16	TC No.15	5000.01	TRUE	1000	50	FALSE	0.0825	4059.383	-					
17	TC No.16	5000.01	FALSE	1000	10	FALSE	0.0825	5412.511		FTFF				
18	TC No.17	5000.01	TRUE	999	10	TRUE	0.0825	4059.383		TFFT				

JUNIT/JACOCO COVERAGE:



Finished after 0.159 seconds

Runs: 17/17 Errors: 0 Failures: 0

```
Problem1ClassTest [Runner: JUnit 4] (0.001 s)
  test (0.001 s)
    [0] 1,5000.01,TRUE,1000,10,FALSE,0.0825,4059.38 (test) (0.001 s)
    [1] 2,2000.00,TRUE,1000,10,FALSE,0.0825,1732 (test) (0.000 s)
    [2] 3,1250.01,TRUE,1000,10,FALSE,0.0825,1150.17 (test) (0.000 s)
    [3] 4,350.00,TRUE,1000,10,FALSE,0.0825,340.99 (test) (0.000 s)
    [4] 5,349.99,TRUE,1000,10,FALSE,0.0825,378.86 (test) (0.000 s)
    [5] 6,5000.00,TRUE,1000,10,FALSE,0.0825,4330 (test) (0.000 s)
    [6] 7,1999.99,TRUE,1000,10,FALSE,0.0825,1840.24 (test) (0.000 s)
    [7] 8,1250.00,TRUE,1000,10,FALSE,0.0825,1217.81 (test) (0.000 s)
    [8] 9,0.00,TRUE,1000,10,FALSE,0.0825,0 (test) (0.000 s)
    [9] 10,20000.00,TRUE,1000,10,FALSE,0.0825,16237.5 (test) (0.000 s)
    [10] 11,5000.01,TRUE,0,10,FALSE,0.0825,5412.51 (test) (0.000 s)
    [11] 12,5000.01,TRUE,999,11,FALSE,0.0825,4059.38 (test) (0.000 s)
    [12] 13,5000.01,TRUE,10000,10,FALSE,0.0825,4059.38 (test) (0.000 s)
    [13] 14,5000.01,TRUE,1000,1,FALSE,0.0825,4059.38 (test) (0.000 s)
    [14] 15,5000.01,TRUE,1000,50,FALSE,0.0825,4059.38 (test) (0.000 s)
    [15] 16,5000.01,FALSE,1000,10,FALSE,0.0825,5412.51 (test) (0.000 s)
    [16] 17,5000.01,TRUE,999,10,TRUE,0.0825,4059.38 (test) (0.000 s)
```

```java
//package Homework5;

public class Problem1Class {

    public double determineTotal (double cart, boolean member, int items, boolean coupon, int memberPoints, double taxRate)
        double factor;

        if (cart > 5_000.00)
            factor = 0.75;
        else
            if (cart >= 2_000.00)
                factor = 0.80;
            else
                if (cart > 1_250.00)
                    factor = 0.85;
                else
                    if (cart >= 350.00)
                        factor = 0.90;
                    else
                        factor = 1.00;

        return ((coupon) && (memberPoints >= 1_000) || (items > 10) || member) ? (1+taxRate)*(factor)*cart : (1+taxRate)*ca
    }
}
```

Problem1ClassTest (2) (2 Dec, 2019 9:35:24 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem4ClassTest.java	🟥 0.0 %	0	72	72
Problem5Class.java	🟥 0.0 %	0	57	57
Problem2ClassTest.java	🟥 0.0 %	0	29	29
Problem5ClassTest.java	🟥 0.0 %	0	22	22
Problem3ClassTest.java	🟥 0.0 %	0	20	20
Problem1Class.java	🟩 100.0 %	57	0	57
Problem1ClassTest.java	🟩 100.0 %	22	0	22

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

Assumptions

1. distance ranges from 0.0 to 5,000.0 ft both inclusive. Significance of 0.1
2. speed ranges from 0.0 to 150.0 mph both inclusive. Significance of 0.1
3. For brakingFactor use Excel's number format and do not truncate. Significance of 0.01

Test

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.

Test Case	Inputs			Exp Out			
Number	distance (ft)	speed (mph)	cruiseEngaged	applyBrakes	brakingFactor	bPath	MCDC
1	1,000.1	84.9	TRUE	FALSE	1.00	9-10-23	TTFT

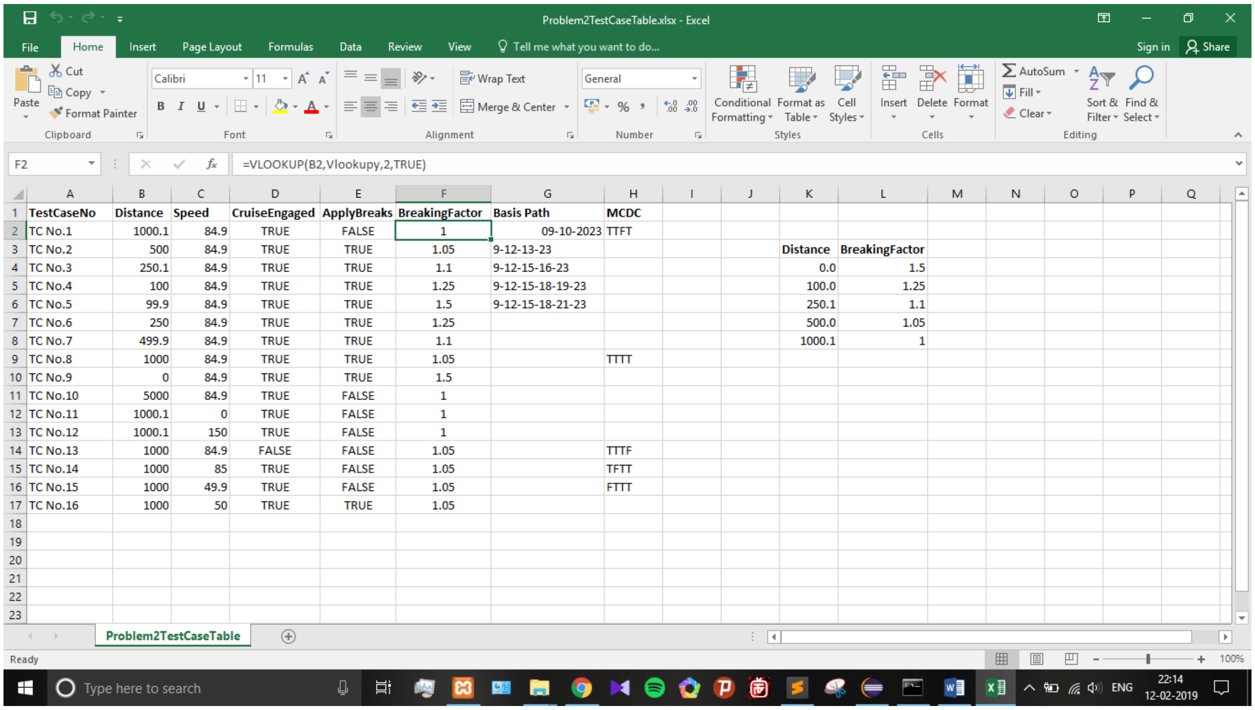
Use the same inputs as shown above **only changing the value of distance** for the following test cases:

1. Basis path set
2. Missing BVs from the Basis path set
3. Extreme range values (for speed extreme range values use the value of distance shown above).

Test case design

1. The comparison threshold for brakingFactor is 0.001 in JUnit
2. You must use VLOOKUP to compute the value of result in your table. Please see the test case table for problem 2 from HW 3 to see how to use this function. Note that the function references another tab that has the BVs for each ECP with the corresponding values of distance for each.

TEST CASE TABLE: (with **VLOOKUP**)



Problem2TestCaseTable.xlsx - Excel

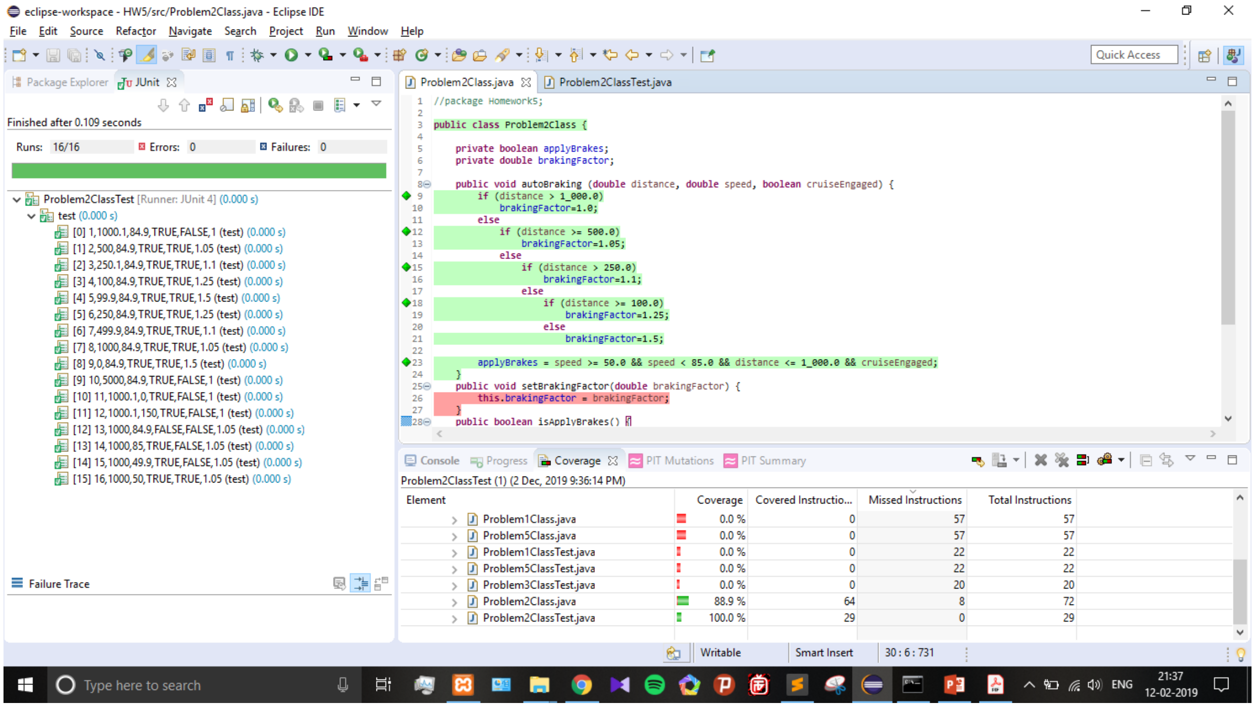
F2 =VLOOKUP(B2,Vlookupy,2,TRUE)

TestCaseNo	Distance	Speed	CruiseEngaged	ApplyBreaks	BreakingFactor	Basis Path	MCDC
TC No.1	1000.1	84.9	TRUE	FALSE	1	09-10-2023	TTFT
TC No.2	500	84.9	TRUE	TRUE	1.05	9-12-13-23	
TC No.3	250.1	84.9	TRUE	TRUE	1.1	9-12-15-16-23	
TC No.4	100	84.9	TRUE	TRUE	1.25	9-12-15-18-19-23	
TC No.5	99.9	84.9	TRUE	TRUE	1.5	9-12-15-18-21-23	
TC No.6	250	84.9	TRUE	TRUE	1.25		
TC No.7	499.9	84.9	TRUE	TRUE	1.1		
TC No.8	1000	84.9	TRUE	TRUE	1.05		TTTT
TC No.9	0	84.9	TRUE	TRUE	1.5		
TC No.10	5000	84.9	TRUE	FALSE	1		
TC No.11	1000.1	0	TRUE	FALSE	1		
TC No.12	1000.1	150	TRUE	FALSE	1		
TC No.13	1000	84.9	FALSE	FALSE	1.05		TTTF
TC No.14	1000	85	TRUE	FALSE	1.05		TFTT
TC No.15	1000	49.9	TRUE	FALSE	1.05		FTTT
TC No.16	1000	50	TRUE	TRUE	1.05		

Distance	BreakingFactor
0.0	1.5
100.0	1.25
250.1	1.1
500.0	1.05
1000.1	1

Problem2TestCaseTable

JUNIT/JACOCO COVERAGE:



eclipse-workspace - HW5/src/Problem2Class.java - Eclipse IDE

Finished after 0.109 seconds

Runs: 16/16 Errors: 0 Failures: 0

```
Problem2ClassTest [Runner: JUnit 4] (0.000 s)
  test (0.000 s)
    [0] 1,1000.1,84.9,TRUE,FALSE,1 (test) (0.000 s)
    [1] 2,500,84.9,TRUE,TRUE,1.05 (test) (0.000 s)
    [2] 3,250.1,84.9,TRUE,TRUE,1.1 (test) (0.000 s)
    [3] 4,100,84.9,TRUE,TRUE,1.25 (test) (0.000 s)
    [4] 5,99.9,84.9,TRUE,TRUE,1.5 (test) (0.000 s)
    [5] 6,250,84.9,TRUE,TRUE,1.25 (test) (0.000 s)
    [6] 7,499.9,84.9,TRUE,TRUE,1.1 (test) (0.000 s)
    [7] 8,1000,84.9,TRUE,TRUE,1.05 (test) (0.000 s)
    [8] 9,0,84.9,TRUE,TRUE,1.5 (test) (0.000 s)
    [9] 10,5000,84.9,TRUE,FALSE,1 (test) (0.000 s)
    [10] 11,1000.1,0,TRUE,FALSE,1 (test) (0.000 s)
    [11] 12,1000.1,150,TRUE,FALSE,1 (test) (0.000 s)
    [12] 13,1000,84.9,FALSE,FALSE,1.05 (test) (0.000 s)
    [13] 14,1000,85,TRUE,FALSE,1.05 (test) (0.000 s)
    [14] 15,1000,49.9,TRUE,FALSE,1.05 (test) (0.000 s)
    [15] 16,1000,50,TRUE,TRUE,1.05 (test) (0.000 s)
```

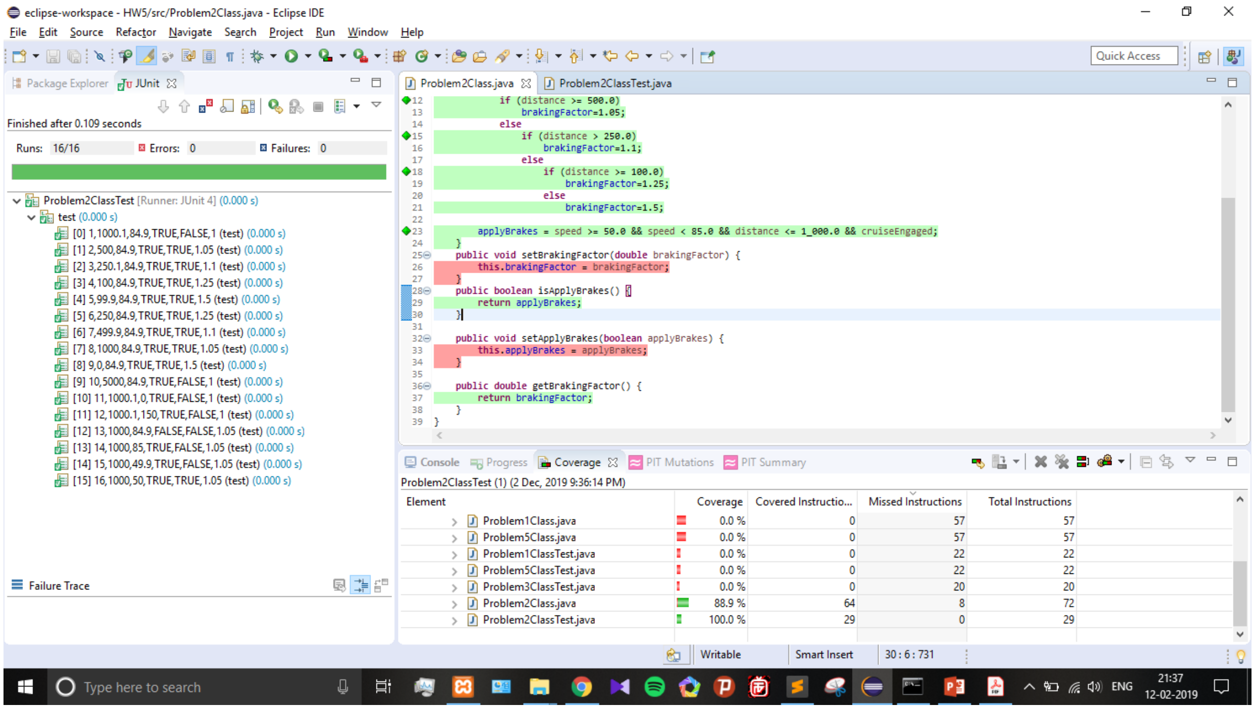
Problem2Class.java

```java
//package Homework5;

public class Problem2Class {

    private boolean applyBrakes;
    private double brakingFactor;

    public void autoBraking (double distance, double speed, boolean cruiseEngaged) {
        if (distance > 1_000.0)
            brakingFactor=1.0;
        else
            if (distance >= 500.0)
                brakingFactor=1.05;
            else
                if (distance > 250.0)
                    brakingFactor=1.1;
                else
                    if (distance >= 100.0)
                        brakingFactor=1.25;
                    else
                        brakingFactor=1.5;

        applyBrakes = speed >= 50.0 && speed < 85.0 && distance <= 1_000.0 && cruiseEngaged;
    }
    public void setBrakingFactor(double brakingFactor) {
        this.brakingFactor = brakingFactor;
    }
    public boolean isApplyBrakes() {
```

Problem2ClassTest (1) (2 Dec, 2019 9:36:14 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem1Class.java	0.0 %	0	57	57
Problem5Class.java	0.0 %	0	57	57
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem2Class.java	88.9 %	64	8	72
Problem2ClassTest.java	100.0 %	29	0	29

Writable Smart Insert 30 : 6 : 731



Finished after 0.109 seconds

Runs: 16/16 Errors: 0 Failures: 0

```
Problem2ClassTest [Runner: JUnit 4] (0.000 s)
  test (0.000 s)
    [0] 1,1000.1,84.9,TRUE,FALSE,1 (test) (0.000 s)
    [1] 2,500,84.9,TRUE,TRUE,1.05 (test) (0.000 s)
    [2] 3,250.1,84.9,TRUE,TRUE,1.1 (test) (0.000 s)
    [3] 4,100,84.9,TRUE,TRUE,1.25 (test) (0.000 s)
    [4] 5,99.9,84.9,TRUE,TRUE,1.5 (test) (0.000 s)
    [5] 6,250,84.9,TRUE,TRUE,1.25 (test) (0.000 s)
    [6] 7,499.9,84.9,TRUE,TRUE,1.1 (test) (0.000 s)
    [7] 8,1000,84.9,TRUE,TRUE,1.05 (test) (0.000 s)
    [8] 9,0,84.9,TRUE,TRUE,1.5 (test) (0.000 s)
    [9] 10,5000,84.9,TRUE,FALSE,1 (test) (0.000 s)
    [10] 11,1000.1,0,TRUE,FALSE,1 (test) (0.000 s)
    [11] 12,1000.1,150,TRUE,FALSE,1 (test) (0.000 s)
    [12] 13,1000,84.9,FALSE,FALSE,1.05 (test) (0.000 s)
    [13] 14,1000,85,TRUE,FALSE,1.05 (test) (0.000 s)
    [14] 15,1000,49.9,TRUE,FALSE,1.05 (test) (0.000 s)
    [15] 16,1000,50,TRUE,TRUE,1.05 (test) (0.000 s)
```

```java
            if (distance >= 500.0)
                brakingFactor=1.05;
            else
                if (distance > 250.0)
                    brakingFactor=1.1;
                else
                    if (distance >= 100.0)
                        brakingFactor=1.25;
                    else
                        brakingFactor=1.5;

        applyBrakes = speed >= 50.0 && speed < 85.0 && distance <= 1_000.0 && cruiseEngaged;
    }
    public void setBrakingFactor(double brakingFactor) {
        this.brakingFactor = brakingFactor;
    }
    public boolean isApplyBrakes() {
        return applyBrakes;
    }

    public void setApplyBrakes(boolean applyBrakes) {
        this.applyBrakes = applyBrakes;
    }

    public double getBrakingFactor() {
        return brakingFactor;
    }
}
```

Problem2ClassTest (1) (2 Dec, 2019 9:36:14 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem1Class.java	0.0 %	0	57	57
Problem5Class.java	0.0 %	0	57	57
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem2Class.java	88.9 %	64	8	72
Problem2ClassTest.java	100.0 %	29	0	29

Problem 3) Test the Problem 3 source code (in the attached zip file). **Do not use Basis path for this problem.**

Description

This code provides the absolute day number of the year for the preceding day.

1. Jan 02, 2019 would be 1
2. Dec 31, 2019 would be 364
3. Jan 01, 2020 would be 365
4. Jan 01, 2021 would be 366

It accounts for leap years.

Assumptions

1. range for year is 2019-2402 both inclusive
2. range for month is 1 to 12 both inclusive
3. range for day is 1 to 31 both inclusive
4. **DO NOT** perform extreme range tests on day, month, and/or year

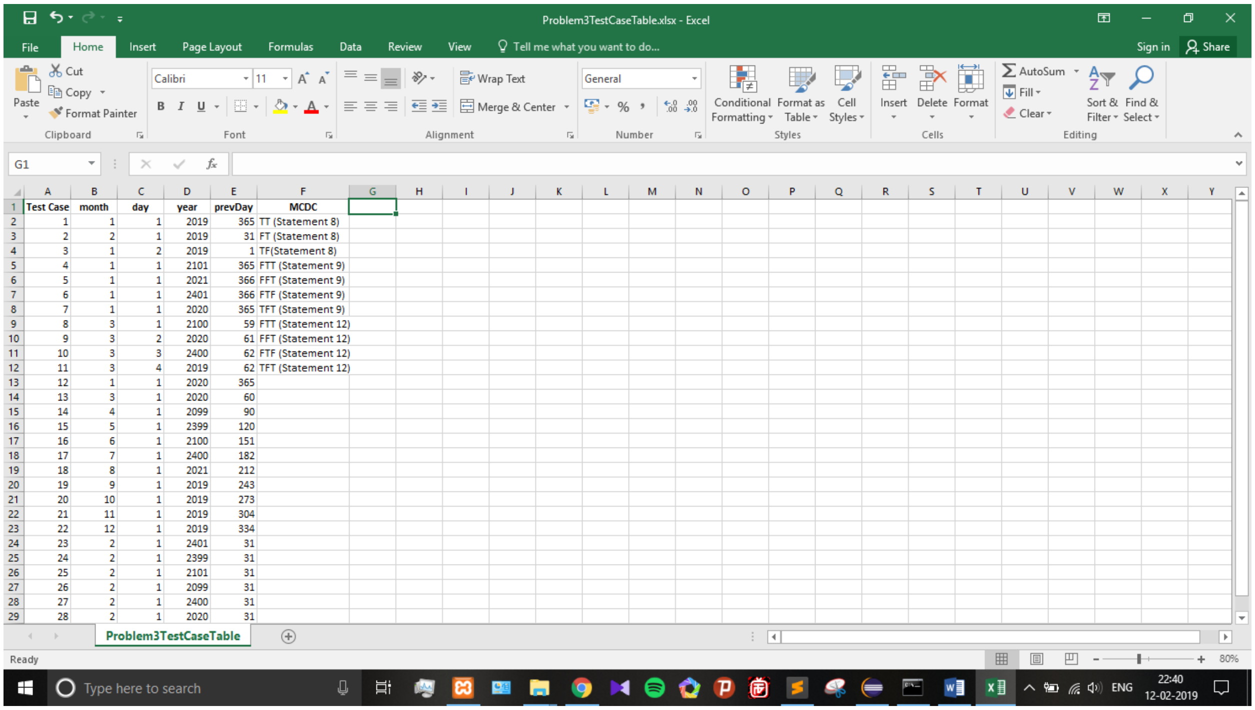
Test

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) - note that statements 8, 9, and 12 need to be described in this column.

Test Case	Inputs			Exp Out	
Number	month	day	year	result	MCDC

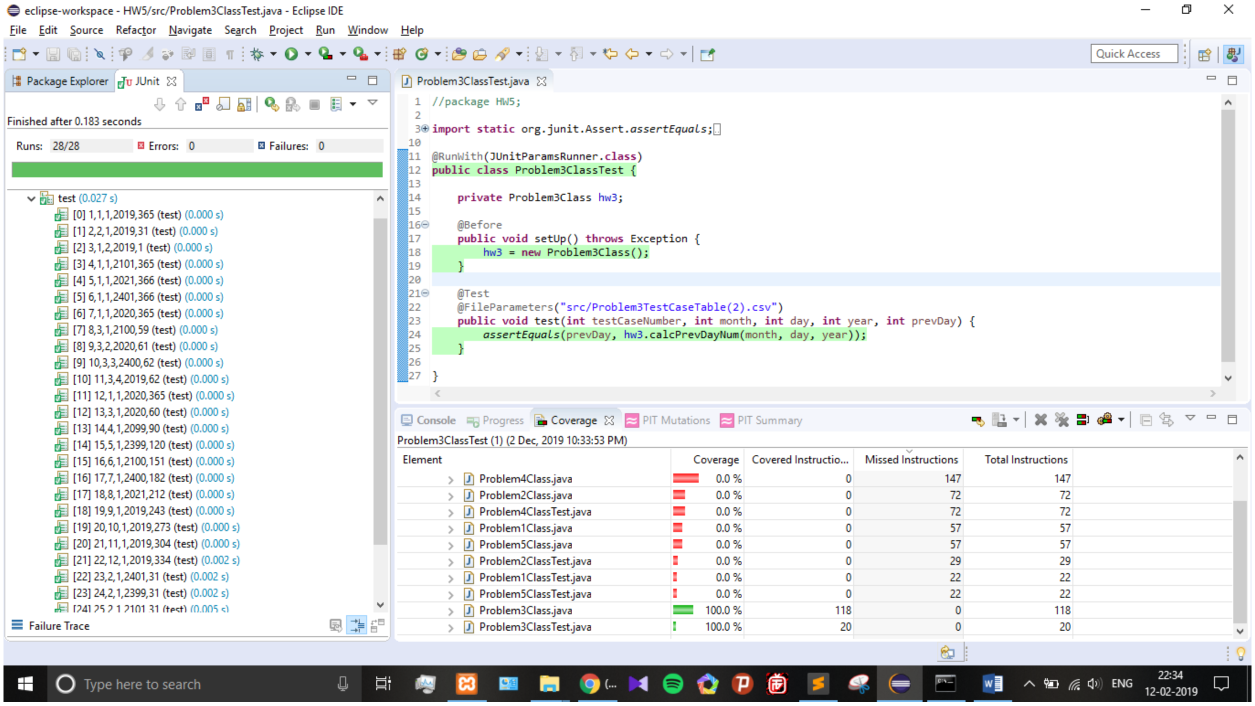
Test case design. In addition to testing for basis path, BVs, MCDC on multiple condition statements, your tests must test all table data. **Use the smallest year that fits the test case you are trying to use.**

TEST CASE TABLE:



Test Case	month	day	year	prevDay	MCDC
1	1	1	2019	365	TT (Statement 8)
2	2	1	2019	31	FT (Statement 8)
3	1	2	2019	1	TF(Statement 8)
4	1	1	2101	365	FTT (Statement 9)
5	1	1	2021	366	FFT (Statement 9)
6	1	1	2401	366	FTF (Statement 9)
7	1	1	2020	365	TFT (Statement 9)
8	3	1	2100	59	FTT (Statement 12)
9	3	2	2020	61	FFT (Statement 12)
10	3	3	2400	62	FTF (Statement 12)
11	3	4	2019	62	TFT (Statement 12)
12	1	1	2020	365	
13	3	1	2020	60	
14	4	1	2099	90	
15	5	1	2399	120	
16	6	1	2100	151	
17	7	1	2400	182	
18	8	1	2021	212	
19	9	1	2019	243	
20	10	1	2019	273	
21	11	1	2019	304	
22	12	1	2019	334	
23	2	1	2401	31	
24	2	1	2399	31	
25	2	1	2101	31	
26	2	1	2099	31	
27	2	1	2400	31	
28	2	1	2020	31	

JUNIT/JACOCO COVERAGE:



```
Finished after 0.183 seconds
Runs: 28/28    Errors: 0    Failures: 0

test (0.027 s)
  [0] 1,1,1,2019,365 (test) (0.000 s)
  [1] 2,2,1,2019,31 (test) (0.000 s)
  [2] 3,1,2,2019,1 (test) (0.000 s)
  [3] 4,1,1,2101,365 (test) (0.000 s)
  [4] 5,1,1,2021,366 (test) (0.000 s)
  [5] 6,1,1,2401,366 (test) (0.000 s)
  [6] 7,1,1,2020,365 (test) (0.000 s)
  [7] 8,3,1,2100,59 (test) (0.000 s)
  [8] 9,3,2,2020,61 (test) (0.000 s)
  [9] 10,3,3,2400,62 (test) (0.000 s)
  [10] 11,3,4,2019,62 (test) (0.000 s)
  [11] 12,1,1,2020,365 (test) (0.000 s)
  [12] 13,3,1,2020,60 (test) (0.000 s)
  [13] 14,4,1,2099,90 (test) (0.000 s)
  [14] 15,5,1,2399,120 (test) (0.000 s)
  [15] 16,6,1,2100,151 (test) (0.000 s)
  [16] 17,7,1,2400,182 (test) (0.000 s)
  [17] 18,8,1,2021,212 (test) (0.000 s)
  [18] 19,9,1,2019,243 (test) (0.000 s)
  [19] 20,10,1,2019,273 (test) (0.000 s)
  [20] 21,11,1,2019,304 (test) (0.000 s)
  [21] 22,12,1,2019,334 (test) (0.002 s)
  [22] 23,2,1,2401,31 (test) (0.002 s)
  [23] 24,2,1,2399,31 (test) (0.002 s)
  [24] 25,2,1,2101,31 (test) (0.005 s)
```

```java
//package HW5;

import static org.junit.Assert.assertEquals;

@RunWith(JUnitParamsRunner.class)
public class Problem3ClassTest {

    private Problem3Class hw3;

    @Before
    public void setUp() throws Exception {
        hw3 = new Problem3Class();
    }

    @Test
    @FileParameters("src/Problem3TestCaseTable(2).csv")
    public void test(int testCaseNumber, int month, int day, int year, int prevDay) {
        assertEquals(prevDay, hw3.calcPrevDayNum(month, day, year));
    }

}
```

Problem3ClassTest (1) (2 Dec, 2019 10:33:53 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem4Class.java	0.0 %	0	147	147
Problem2Class.java	0.0 %	0	72	72
Problem4ClassTest.java	0.0 %	0	72	72
Problem1Class.java	0.0 %	0	57	57
Problem5Class.java	0.0 %	0	57	57
Problem2ClassTest.java	0.0 %	0	29	29
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3Class.java	100.0 %	118	0	118
Problem3ClassTest.java	100.0 %	20	0	20

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

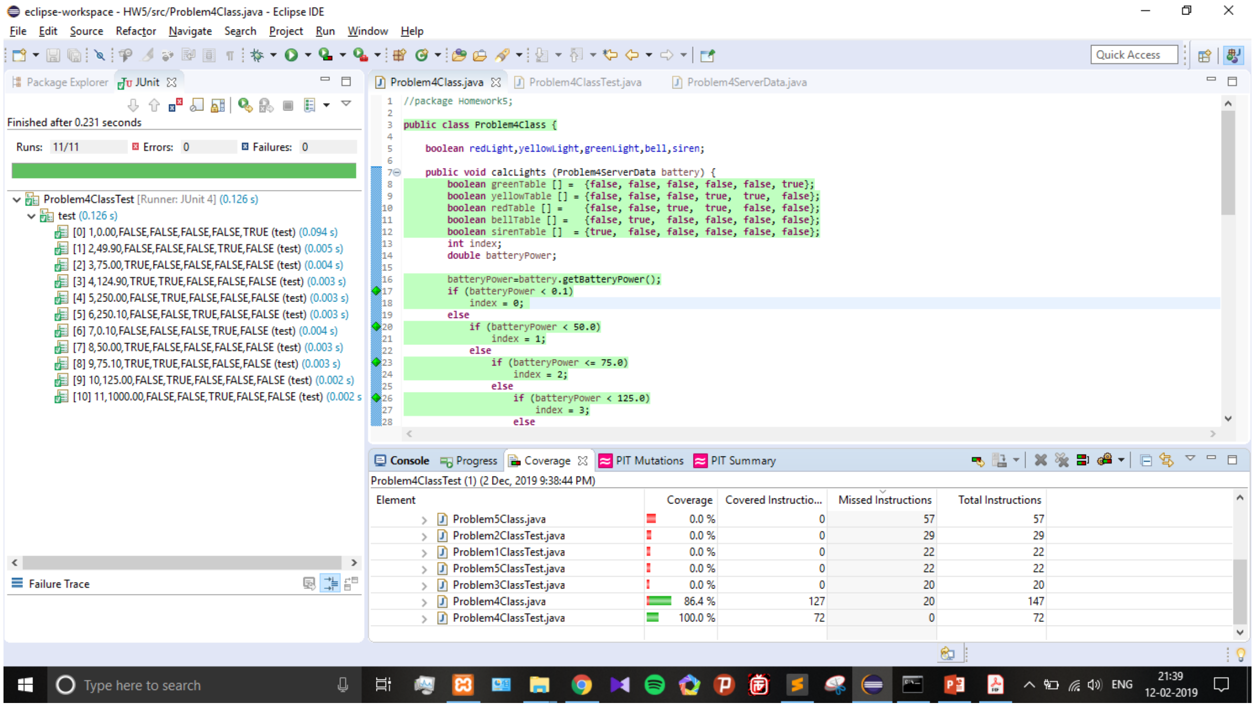
Use the supplied Excel test case table (which is the solution for problem 1 from HW 3).
Use Easy Mock to mock the call to the server to get batteryPower.

Easy Mock instructions

i. Download Easy Mock from the M14 Blackboard files
ii. Install Easy Mock in your project (add the EasyMock.jar to your Java Build path)

Execute the JUnit test. Create **Problem4ServerData.java** to define the signature for the getBatteryPower() method (see slide 32 of M14 for how to do this). Follow the five steps shown in slide 35 of M14 to get Easy Mock to work in your test environment.

<u>JUNIT/JACOCO COVERAGE:</u>

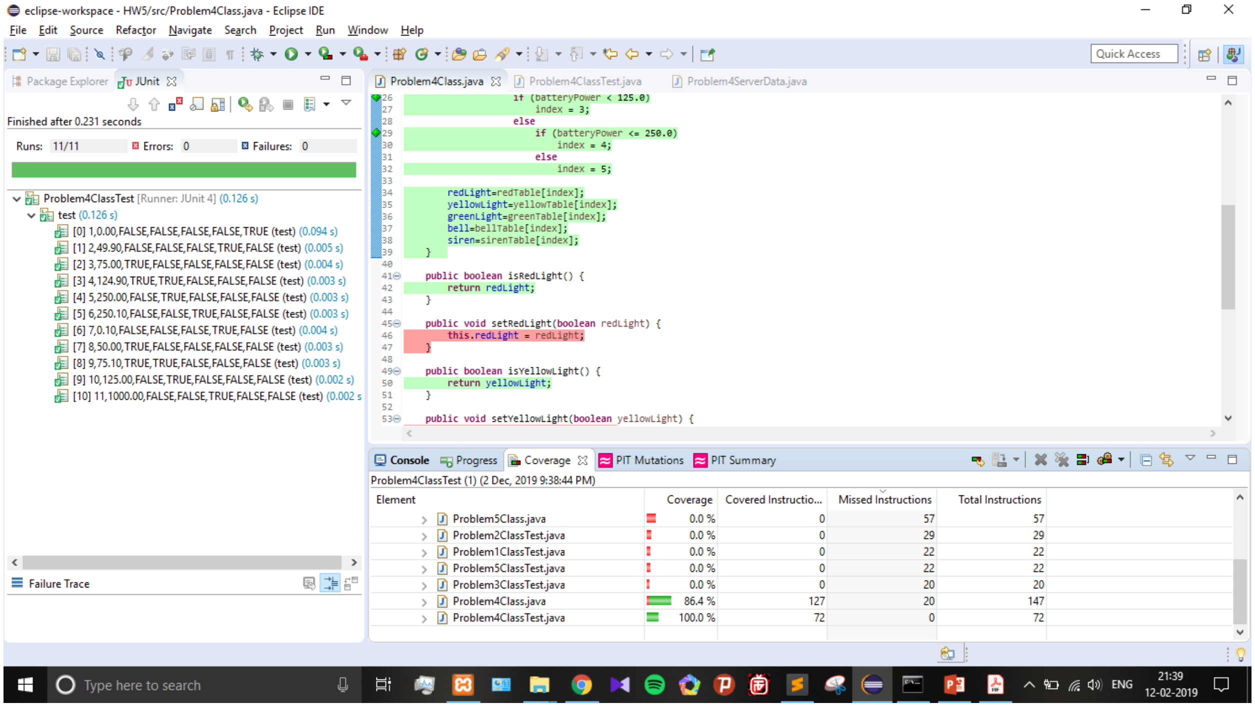


```
Finished after 0.231 seconds

Runs: 11/11    Errors: 0    Failures: 0

Problem4ClassTest [Runner: JUnit 4] (0.126 s)
  test (0.126 s)
    [0] 1,0.00,FALSE,FALSE,FALSE,FALSE,TRUE (test) (0.094 s)
    [1] 2,49.90,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.005 s)
    [2] 3,75.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.004 s)
    [3] 4,124.90,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [4] 5,250.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [5] 6,250.10,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.003 s)
    [6] 7,0.10,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.004 s)
    [7] 8,50.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.003 s)
    [8] 9,75.10,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [9] 10,125.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.002 s)
    [10] 11,1000.00,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.002 s
```

```java
//package Homework5;

public class Problem4Class {

    boolean redLight,yellowLight,greenLight,bell,siren;

    public void calcLights (Problem4ServerData battery) {
        boolean greenTable [] =  {false, false, false, false, false, true};
        boolean yellowTable [] = {false, false, false, true,  true,  false};
        boolean redTable [] =    {false, false, true,  true,  false, false};
        boolean bellTable [] =   {false, true,  false, false, false, false};
        boolean sirenTable []  = {true,  false, false, false, false, false};
        int index;
        double batteryPower;

        batteryPower=battery.getBatteryPower();
        if (batteryPower < 0.1)
            index = 0;
        else
            if (batteryPower < 50.0)
                index = 1;
            else
                if (batteryPower <= 75.0)
                    index = 2;
                else
                    if (batteryPower < 125.0)
                        index = 3;
                    else
```

Problem4ClassTest (1) (2 Dec, 2019 9:38:44 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem5Class.java	0.0 %	0	57	57
Problem2ClassTest.java	0.0 %	0	29	29
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem4Class.java	86.4 %	127	20	147
Problem4ClassTest.java	100.0 %	72	0	72



eclipse-workspace - HW5/src/Problem4Class.java - Eclipse IDE

Finished after 0.231 seconds

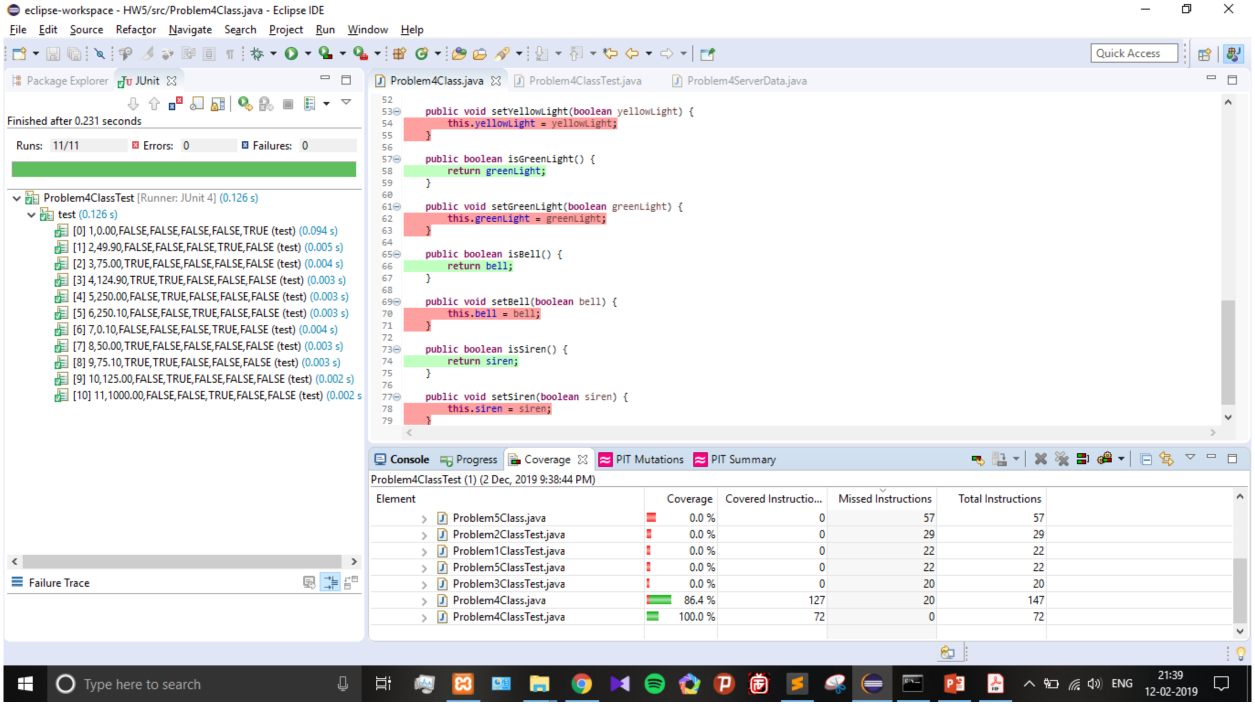
Runs: 11/11 Errors: 0 Failures: 0

```
Problem4ClassTest [Runner: JUnit 4] (0.126 s)
  test (0.126 s)
    [0] 1,0.00,FALSE,FALSE,FALSE,FALSE,TRUE (test) (0.094 s)
    [1] 2,49.90,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.005 s)
    [2] 3,75.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.004 s)
    [3] 4,124.90,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [4] 5,250.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [5] 6,250.10,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.003 s)
    [6] 7,0.10,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.004 s)
    [7] 8,50.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.003 s)
    [8] 9,75.10,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [9] 10,125.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.002 s)
    [10] 11,1000.00,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.002 s
```

```java
                if (batteryPower < 125.0)
                    index = 3;
                else
                    if (batteryPower <= 250.0)
                        index = 4;
                    else
                        index = 5;

        redLight=redTable[index];
        yellowLight=yellowTable[index];
        greenLight=greenTable[index];
        bell=bellTable[index];
        siren=sirenTable[index];
    }

    public boolean isRedLight() {
        return redLight;
    }

    public void setRedLight(boolean redLight) {
        this.redLight = redLight;
    }

    public boolean isYellowLight() {
        return yellowLight;
    }

    public void setYellowLight(boolean yellowLight) {
```

Problem4ClassTest (1) (2 Dec, 2019 9:38:44 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem5Class.java	0.0 %	0	57	57
Problem2ClassTest.java	0.0 %	0	29	29
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem4Class.java	86.4 %	127	20	147
Problem4ClassTest.java	100.0 %	72	0	72



eclipse-workspace - HW5/src/Problem4Class.java - Eclipse IDE

Finished after 0.231 seconds

Runs: 11/11 Errors: 0 Failures: 0

```
Problem4ClassTest [Runner: JUnit 4] (0.126 s)
  test (0.126 s)
    [0] 1,0.00,FALSE,FALSE,FALSE,FALSE,TRUE (test) (0.094 s)
    [1] 2,49.90,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.005 s)
    [2] 3,75.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.004 s)
    [3] 4,124.90,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [4] 5,250.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [5] 6,250.10,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.003 s)
    [6] 7,0.10,FALSE,FALSE,FALSE,TRUE,FALSE (test) (0.004 s)
    [7] 8,50.00,TRUE,FALSE,FALSE,FALSE,FALSE (test) (0.003 s)
    [8] 9,75.10,TRUE,TRUE,FALSE,FALSE,FALSE (test) (0.003 s)
    [9] 10,125.00,FALSE,TRUE,FALSE,FALSE,FALSE (test) (0.002 s)
    [10] 11,1000.00,FALSE,FALSE,TRUE,FALSE,FALSE (test) (0.002 s
```

```java

    public void setYellowLight(boolean yellowLight) {
        this.yellowLight = yellowLight;
    }

    public boolean isGreenLight() {
        return greenLight;
    }

    public void setGreenLight(boolean greenLight) {
        this.greenLight = greenLight;
    }

    public boolean isBell() {
        return bell;
    }

    public void setBell(boolean bell) {
        this.bell = bell;
    }

    public boolean isSiren() {
        return siren;
    }

    public void setSiren(boolean siren) {
        this.siren = siren;
    }
```

Problem4ClassTest (1) (2 Dec, 2019 9:38:44 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem5Class.java	0.0 %	0	57	57
Problem2ClassTest.java	0.0 %	0	29	29
Problem1ClassTest.java	0.0 %	0	22	22
Problem5ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem4Class.java	86.4 %	127	20	147
Problem4ClassTest.java	100.0 %	72	0	72

Problem 5) Test the Problem 5 source code (in the attached zip file) which is actually problem 1 from this assignment. Use the test case table provided to test the file. Run PIT against this test (use v1.1.9 - see the last line of the index.html file). Run JUnit and JaCoCo coverage against the tests. Take snapshots of each as required.

Determine what is wrong with the PIT coverage. Explain in your own words why the PIT coverage is so poor. Compare these test cases with problem 1 tests and take a PIT snapshot of each (both Problem 1 and this problem's test results).

Submit

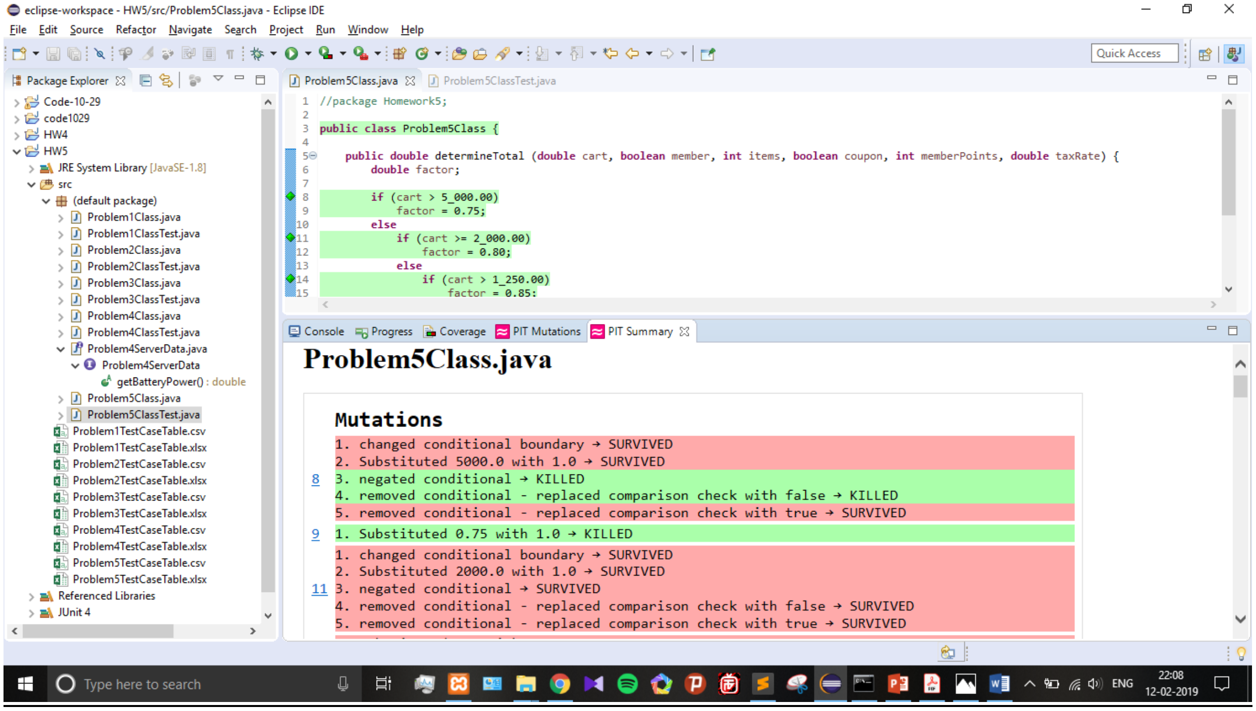
1. Text explanation of PIT coverage analysis of the provided tests (considering the JUnit and JaCoCo coverage indicated).
2. JUnit and JaCoCo coverage snapshot with timestamp
3. Two PIT html files (one each for Problem 1 and Problem 5 PIT results). Place these two files in your zip file in a folder named **Problem5 PIT results**.

To get the html files look under your Eclipse Workspace -> .metadata -> .plugins -> org.pitest.pitclipse.core -> html_results ... and find files: Problem1Class.java.html and Problem5Class.java.html.

Double check

1. Make sure that you have PIT v1.1.9 (see the last line in the PIT html results file).
2. Make sure that all Mutators is set. 50% deduction if not.

PROBLEM 5 – PIT MUTATION:

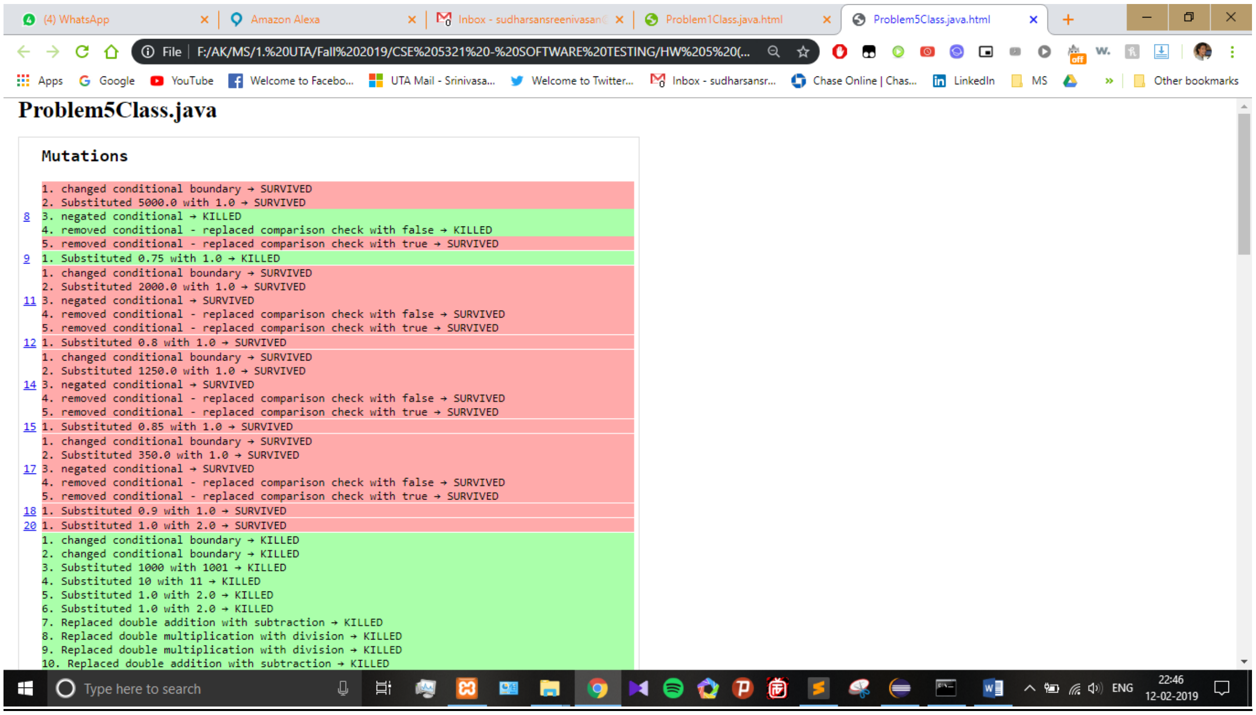


```java
//package Homework5;

public class Problem5Class {

    public double determineTotal (double cart, boolean member, int items, boolean coupon, int memberPoints, double taxRate) {
        double factor;

        if (cart > 5_000.00)
            factor = 0.75;
        else
            if (cart >= 2_000.00)
                factor = 0.80;
            else
                if (cart > 1_250.00)
                    factor = 0.85;
```

Problem5Class.java

Mutations

```
   1. changed conditional boundary → SURVIVED
   2. Substituted 5000.0 with 1.0 → SURVIVED
8  3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → SURVIVED
9  1. Substituted 0.75 with 1.0 → KILLED
   1. changed conditional boundary → SURVIVED
   2. Substituted 2000.0 with 1.0 → SURVIVED
11 3. negated conditional → SURVIVED
   4. removed conditional - replaced comparison check with false → SURVIVED
   5. removed conditional - replaced comparison check with true → SURVIVED
```

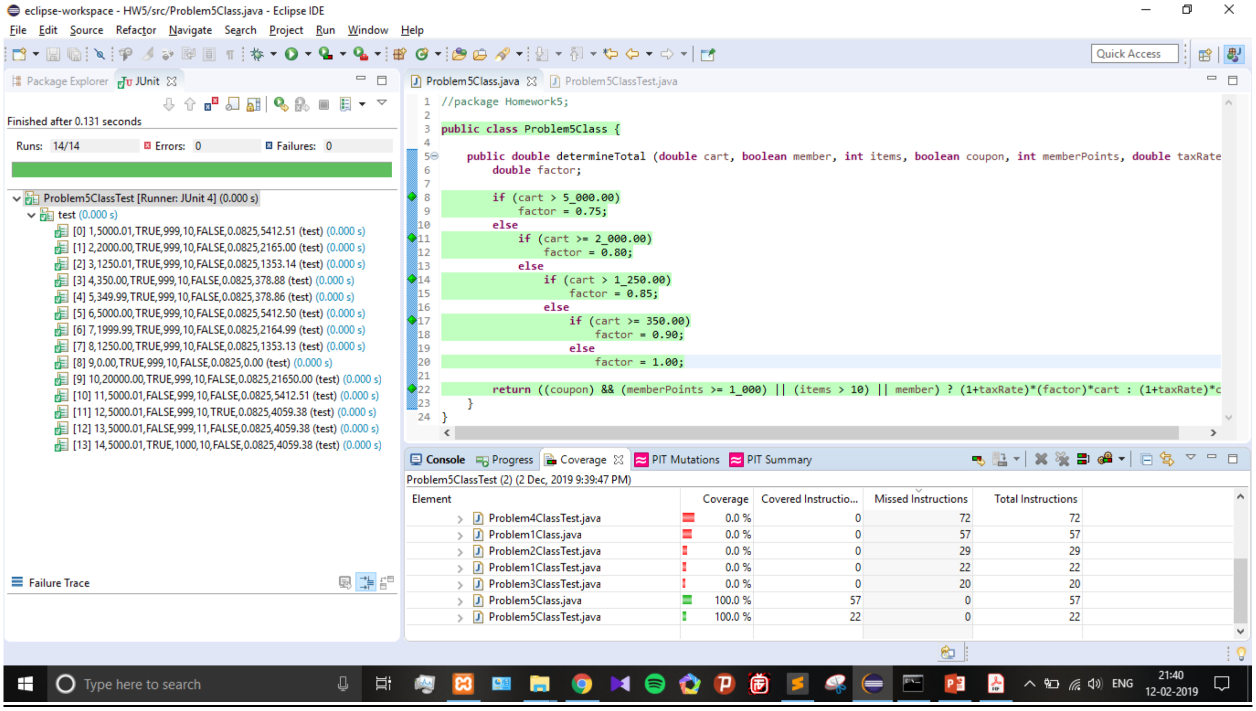


Problem5Class.java

Mutations

```
   1. changed conditional boundary → SURVIVED
   2. Substituted 5000.0 with 1.0 → SURVIVED
8  3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → SURVIVED
9  1. Substituted 0.75 with 1.0 → KILLED
   1. changed conditional boundary → SURVIVED
   2. Substituted 2000.0 with 1.0 → SURVIVED
11 3. negated conditional → SURVIVED
   4. removed conditional - replaced comparison check with false → SURVIVED
   5. removed conditional - replaced comparison check with true → SURVIVED
12 1. Substituted 0.8 with 1.0 → SURVIVED
   1. changed conditional boundary → SURVIVED
   2. Substituted 1250.0 with 1.0 → SURVIVED
14 3. negated conditional → SURVIVED
   4. removed conditional - replaced comparison check with false → SURVIVED
   5. removed conditional - replaced comparison check with true → SURVIVED
15 1. Substituted 0.85 with 1.0 → SURVIVED
   1. changed conditional boundary → SURVIVED
   2. Substituted 350.0 with 1.0 → SURVIVED
17 3. negated conditional → SURVIVED
   4. removed conditional - replaced comparison check with false → SURVIVED
   5. removed conditional - replaced comparison check with true → SURVIVED
18 1. Substituted 0.9 with 1.0 → SURVIVED
20 1. Substituted 1.0 with 2.0 → SURVIVED
   1. changed conditional boundary → KILLED
   2. changed conditional boundary → KILLED
   3. Substituted 1000 with 1001 → KILLED
   4. Substituted 10 with 11 → KILLED
   5. Substituted 1.0 with 2.0 → KILLED
   6. Substituted 1.0 with 2.0 → KILLED
   7. Replaced double addition with subtraction → KILLED
   8. Replaced double multiplication with division → KILLED
   9. Replaced double multiplication with division → KILLED
   10. Replaced double addition with subtraction → KILLED
```

PROBLEM 5 - JUNIT/JACOCO COVERAGE:



Finished after 0.131 seconds

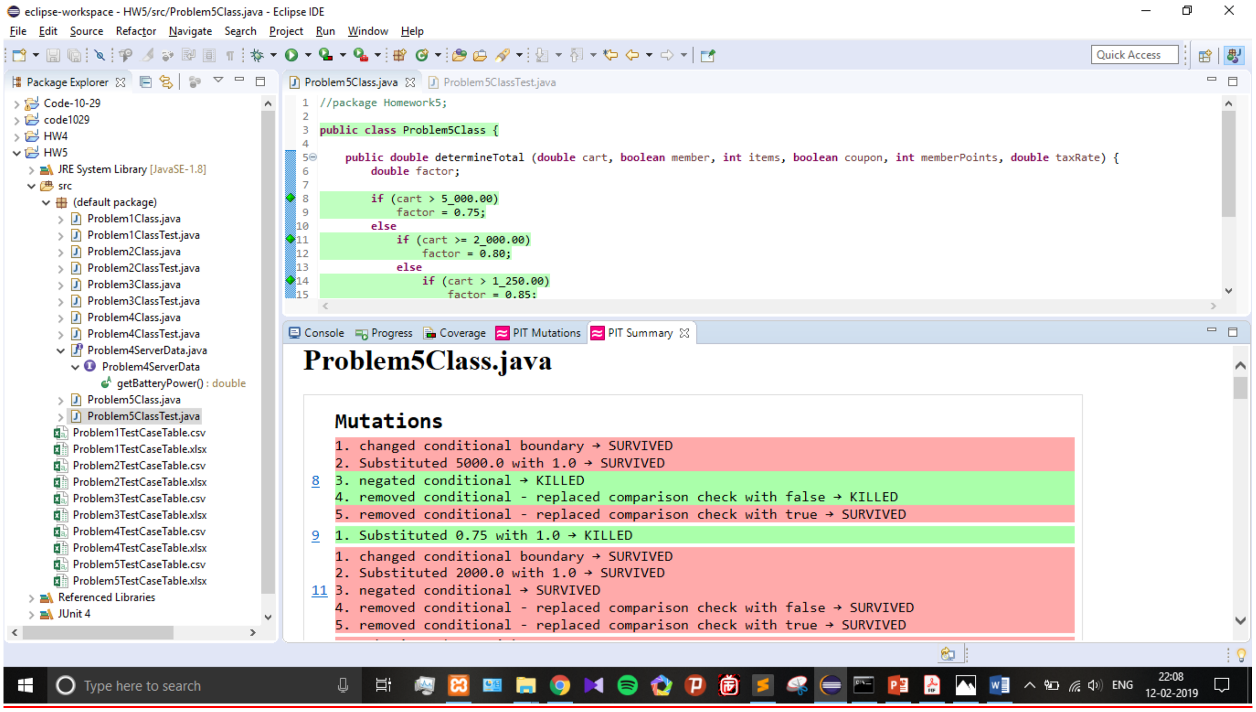
Runs: 14/14 Errors: 0 Failures: 0

```
Problem5ClassTest [Runner: JUnit 4] (0.000 s)
  test (0.000 s)
    [0] 1,5000.01,TRUE,999,10,FALSE,0.0825,5412.51 (test) (0.000 s)
    [1] 2,2000.00,TRUE,999,10,FALSE,0.0825,2165.00 (test) (0.000 s)
    [2] 3,1250.01,TRUE,999,10,FALSE,0.0825,1353.14 (test) (0.000 s)
    [3] 4,350.00,TRUE,999,10,FALSE,0.0825,378.88 (test) (0.000 s)
    [4] 5,349.99,TRUE,999,10,FALSE,0.0825,378.86 (test) (0.000 s)
    [5] 6,5000.00,TRUE,999,10,FALSE,0.0825,5412.50 (test) (0.000 s)
    [6] 7,1999.99,TRUE,999,10,FALSE,0.0825,2164.99 (test) (0.000 s)
    [7] 8,1250.00,TRUE,999,10,FALSE,0.0825,1353.13 (test) (0.000 s)
    [8] 9,0.00,TRUE,999,10,FALSE,0.0825,0.00 (test) (0.000 s)
    [9] 10,20000.00,TRUE,999,10,FALSE,0.0825,21650.00 (test) (0.000 s)
    [10] 11,5000.01,FALSE,999,10,FALSE,0.0825,5412.51 (test) (0.000 s)
    [11] 12,5000.01,FALSE,999,10,TRUE,0.0825,4059.38 (test) (0.000 s)
    [12] 13,5000.01,FALSE,999,11,FALSE,0.0825,4059.38 (test) (0.000 s)
    [13] 14,5000.01,TRUE,1000,10,FALSE,0.0825,4059.38 (test) (0.000 s)
```

```java
//package Homework5;

public class Problem5Class {

    public double determineTotal (double cart, boolean member, int items, boolean coupon, int memberPoints, double taxRate
        double factor;

        if (cart > 5_000.00)
            factor = 0.75;
        else
            if (cart >= 2_000.00)
                factor = 0.80;
            else
                if (cart > 1_250.00)
                    factor = 0.85;
                else
                    if (cart >= 350.00)
                        factor = 0.90;
                    else
                        factor = 1.00;

        return ((coupon) && (memberPoints >= 1_000) || (items > 10) || member) ? (1+taxRate)*(factor)*cart : (1+taxRate)*c
    }
}
```

Problem5ClassTest (2) (2 Dec, 2019 9:39:47 PM)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
Problem4ClassTest.java	0.0 %	0	72	72
Problem1Class.java	0.0 %	0	57	57
Problem2ClassTest.java	0.0 %	0	29	29
Problem1ClassTest.java	0.0 %	0	22	22
Problem3ClassTest.java	0.0 %	0	20	20
Problem5Class.java	100.0 %	57	0	57
Problem5ClassTest.java	100.0 %	22	0	22



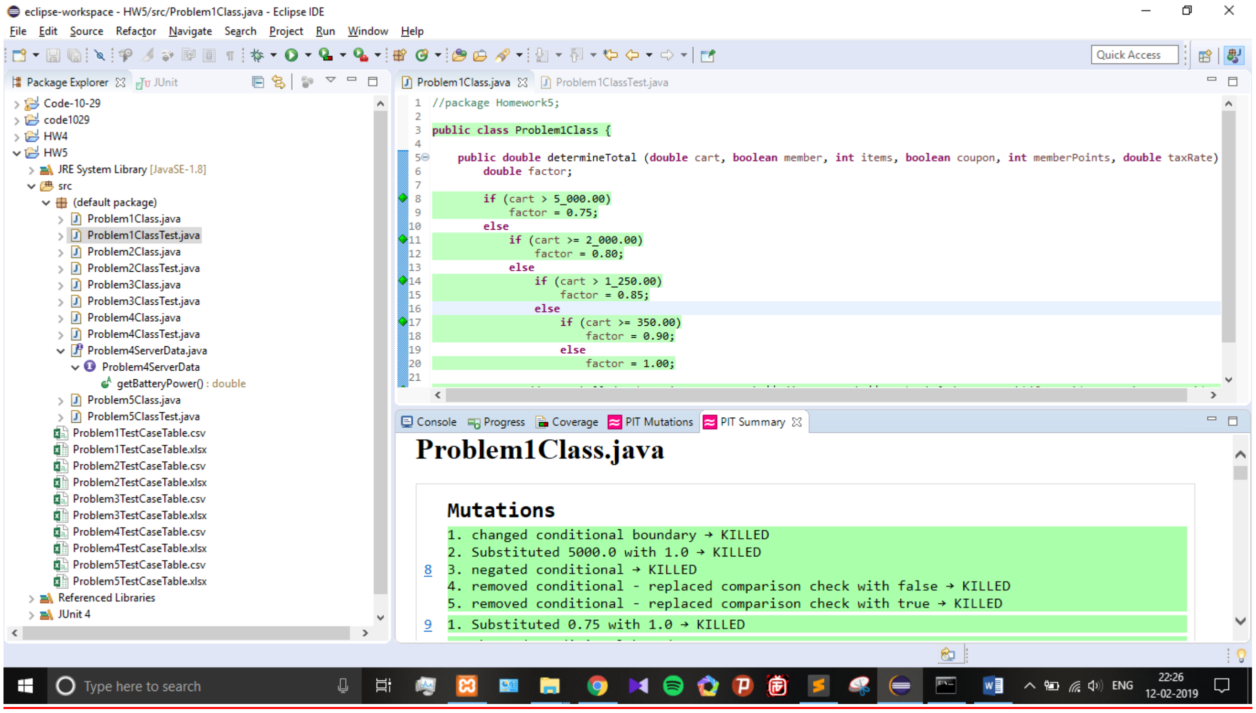
```java
//package Homework5;

public class Problem5Class {

    public double determineTotal (double cart, boolean member, int items, boolean coupon, int memberPoints, double taxRate) {
        double factor;

        if (cart > 5_000.00)
            factor = 0.75;
        else
            if (cart >= 2_000.00)
                factor = 0.80;
            else
                if (cart > 1_250.00)
                    factor = 0.85;
```

Problem5Class.java

Mutations

```
   1. changed conditional boundary → SURVIVED
   2. Substituted 5000.0 with 1.0 → SURVIVED
8  3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → SURVIVED
9  1. Substituted 0.75 with 1.0 → KILLED
   1. changed conditional boundary → SURVIVED
   2. Substituted 2000.0 with 1.0 → SURVIVED
11 3. negated conditional → SURVIVED
   4. removed conditional - replaced comparison check with false → SURVIVED
   5. removed conditional - replaced comparison check with true → SURVIVED
```

PROBLEM 1 – PIT MUTATION:

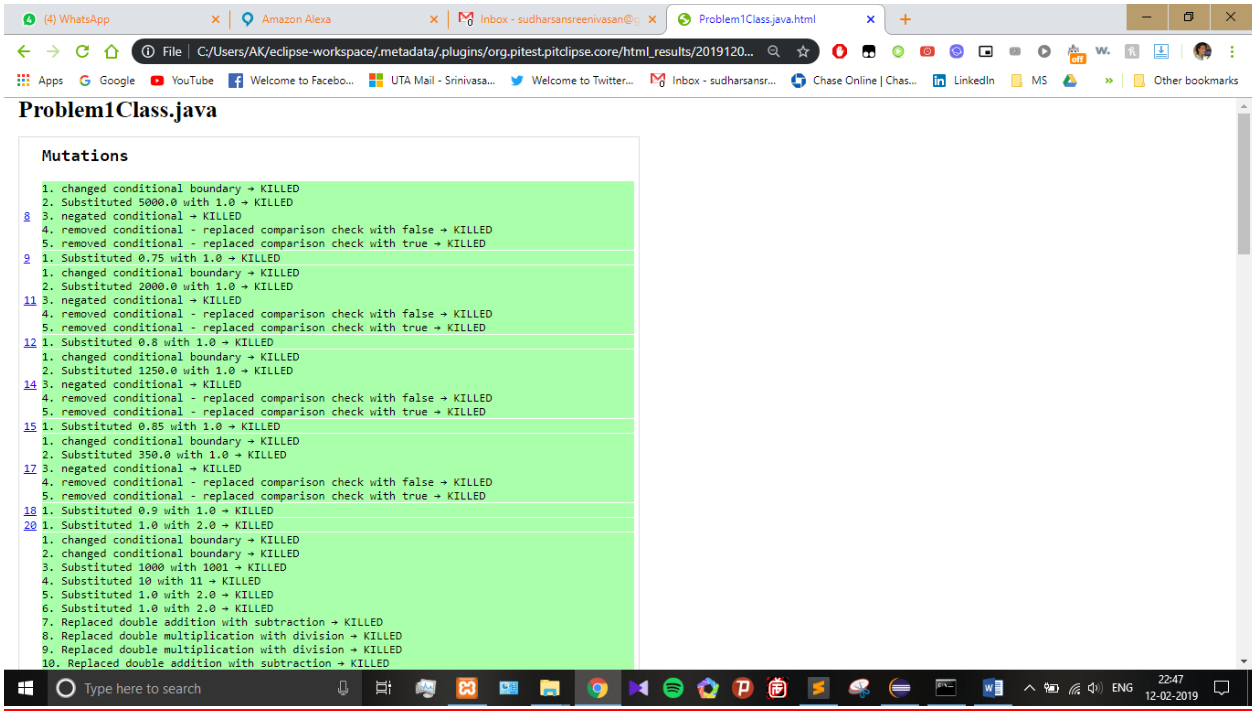


```java
//package Homework5;

public class Problem1Class {

    public double determineTotal (double cart, boolean member, int items, boolean coupon, int memberPoints, double taxRate)
        double factor;

        if (cart > 5_000.00)
            factor = 0.75;
        else
            if (cart >= 2_000.00)
                factor = 0.80;
            else
                if (cart > 1_250.00)
                    factor = 0.85;
                else
                    if (cart >= 350.00)
                        factor = 0.90;
                    else
                        factor = 1.00;

```

Problem1Class.java

Mutations

```
   1. changed conditional boundary → KILLED
   2. Substituted 5000.0 with 1.0 → KILLED
8  3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → KILLED
9  1. Substituted 0.75 with 1.0 → KILLED
```



Problem1Class.java

```
Mutations

   1. changed conditional boundary → KILLED
   2. Substituted 5000.0 with 1.0 → KILLED
8  3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → KILLED
9  1. Substituted 0.75 with 1.0 → KILLED
   1. changed conditional boundary → KILLED
   2. Substituted 2000.0 with 1.0 → KILLED
11 3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → KILLED
12 1. Substituted 0.8 with 1.0 → KILLED
   1. changed conditional boundary → KILLED
   2. Substituted 1250.0 with 1.0 → KILLED
14 3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → KILLED
15 1. Substituted 0.85 with 1.0 → KILLED
   1. changed conditional boundary → KILLED
   2. Substituted 350.0 with 1.0 → KILLED
17 3. negated conditional → KILLED
   4. removed conditional - replaced comparison check with false → KILLED
   5. removed conditional - replaced comparison check with true → KILLED
18 1. Substituted 0.9 with 1.0 → KILLED
20 1. Substituted 1.0 with 2.0 → KILLED
   1. changed conditional boundary → KILLED
   2. changed conditional boundary → KILLED
   3. Substituted 1000 with 1001 → KILLED
   4. Substituted 10 with 11 → KILLED
   5. Substituted 1.0 with 2.0 → KILLED
   6. Substituted 1.0 with 2.0 → KILLED
   7. Replaced double addition with subtraction → KILLED
   8. Replaced double multiplication with division → KILLED
   9. Replaced double multiplication with division → KILLED
   10. Replaced double addition with subtraction → KILLED
```

EXPLANATION FROM ABOVE OBSERVATION:

There is full PIT coverage for Problem 1 but not for Problem 5 due to the following difference in the test cases:

- Problem 5 does not extensively cover all boundary/MCDC solutions (boundary values for **member, memberpts**) whereas Problem 1 does.
- Test Cases provided for Problem 5 **does not cover full condition coverage for all variables**, but Problem 1 does.
- We can fix this by making sure all variables have condition coverage. For example, **TRUE for member, memberpts=1000** to ensure condition coverage.