

Kevin Bruzon

CS 3250

04/24/2022

### SHOWCASE 1 FINAL

For my showcase, the subject under test will be a Mortgage Calculator Web Application. The software was downloaded, and deployed locally as a Web App in order to conduct the automated testing.

The software was obtained from <https://github.com/mackenzieweaver/MortgageCalculator>. The authors of the software are Mackenzie Weaver, Josh Scott, and Mark Danny Carroll.

The software allows users to input information regarding a loan, and then calculates how much you will pay monthly, for the duration of the input term, as well as how much you will ultimately pay with interest included.

In order to perform automated testing of the software, Selenium will be used in Eclipse IDE using Java to create automated test cases. For testing purposes, the software was deployed on the local server, <http://127.0.0.1:8000/>. (using python -m http.server in CMD)

For my coverage criteria, I used Input Space Partitioning, creating 10 test cases that satisfy Base Choice Coverage, and 10 test cases that satisfy Pair-Wise Choice Coverage.

### ISP

#### Input and State Variables

- 5 Textboxes: Price, Down, Loan Amount, Interest Rate, Term(months)
- 2 Form Buttons: Calculate, Reset
- Many browser features: back, forward, reload, ...
- 4 Page features: Home, Solution, Code, Github

#### Characteristics

- Input is a number or not: [number, not number]
- Number compared to 0: [neg, 0, pos]
- Form button is clicked: [Calculate, Reset]
- Browser feature is clicked: [None, Back, Forward, Reload]
- Page feature is clicked: [None, Home, Solution, Code, Github]

\*\*\*All Partitions are COMPLETE and DISJOINT\*\*\*

## Applying BCC

- Input is a number or not: [number, not number] → Base: number
- Number compared to 0: [neg, 0, pos] → Base: pos
- Form button is clicked: [Calculate, Reset] → Base: Calculate
- Browser feature is clicked: [None, Back, Forward, Reload] → Base: None
- Page feature is clicked: [None, Home, Solution, Code, Github] → Base: None

## BCC Test Requirements

Base TR: {number, pos, Calculate, None, None}

TR 2: {**not number**, pos, Calculate, None, None}

TR 3: {number, **0**, Calculate, None, None}

TR 4: {number, **neg**, Calculate, None, None}

TR 5: {number, pos, **Reset**, None, None}

TR 6: {number, pos, Calculate, **Back**, None}

TR 7: {number, pos, Calculate, **Reload**, None}

TR 8: {number, pos, Calculate, **Forward**, None}

TR 9: {number, pos, Calculate, None, **Home**}

\*\*\*Redundant test case, same concept as TR6\*\*\*

\*\*\*Navigating Back is the same as navigating home\*\*\*

TR 10: {number, pos, Calculate, None, **Solution**}

\*\*\*Redundant test case, same concept as TR7\*\*\*

\*\*\*Navigating to current page is the same as reloading\*\*\*

TR 11: {number, pos, Calculate, None, **Code**}

TR 12: {number, pos, Calculate, None, **Github**}

## Applying PWC

TR	Page Ft	Browser Ft	Num comp to 0	Num or Not Num	Form Button
1	None	None	Neg	Not	Calculate
2	None	Back	0	Num	Reset
3	None	Reload	Pos	Not	Calculate
4	None	Forward	Neg	Num	Reset
5	Home	None	0	Num	Reset
6	Home	Back	Pos	Not	Calculate
7	Home	Reload	Neg	Num	Calculate
8	Home	Forward	0	Not	Reset
9	Solution	None	Pos	Num	Calculate
10	Solution	Back	Neg	Not	Reset
11	Solution	Reload	0	Num	Calculate
12	Solution	Forward	Pos	Not	Reset
13	Code	None	Neg	Num	Calculate
14	Code	Back	0	Not	Reset
15	Code	Reload	Pos	Num	Calculate
16	Code	Forward	Neg	Not	Reset
17	Github	None	0	Num	Calculate
18	Github	Back	Pos	Not	Reset
19	Github	Reload	Neg	Num	Reset
20	Github	Forward	0	Not	Calculate

## PWC Test Requirements:

**\*\*Since 10 Test Cases were created with BCC, Only using Bold TR for PWC Automated Tests\*\***

**TR1: {Not Number, Neg, Calc, None, None}**

TR2: {Number, 0, Reset, Back, None}

**TR3: {Not Number, Pos, Calc, Reload, None}**

**TR4: {Number, Neg, Reset, Forward, None}**

TR5: {Number, 0, Reset, None, Home}

\*\*\*Redundant, same concept as TR2\*\*\*

\*\*\*Navigating back is same as navigating to home\*\*\*

**TR6: {Not Number, Pos, Calc, Back, Home}**

**TR7: {Number, Neg, Calculate, Reload, Home}**

TR8: {Not Number, 0, Reset, Forward, Home}

TR9: {Number, Pos, Calculate, None, Solution}

**TR10: {Not Number, Neg, Reset, Back, Solution}**

TR11: {Number, 0, Calculate, Reload, Solution}

TR12: {Not Number, Pos, Reset, Forward, Solution}

TR13: {Number, Neg, Calculate, None, Code}

**TR14: {Not Number, 0, Reset, Back, Code}**

**TR15: {Number, Pos, Calculate, Reload, Code}**

TR16: {Not Number, Neg, Reset, Forward, Code}

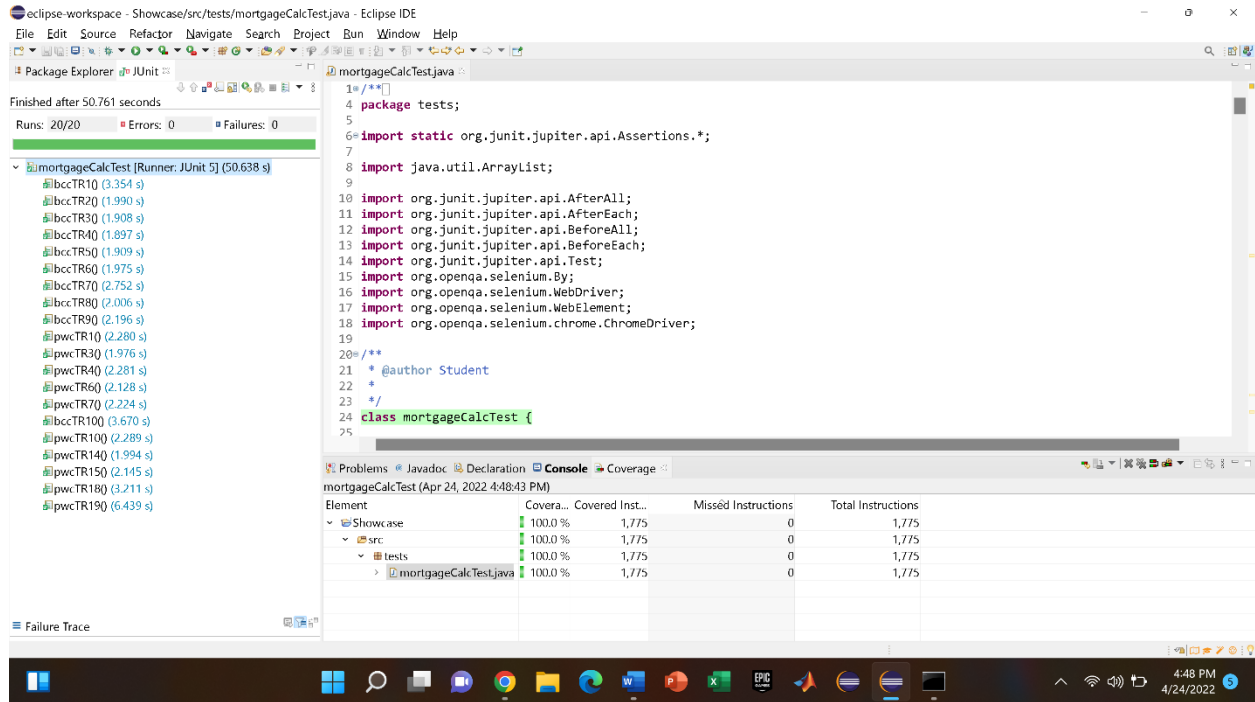
TR17: {Number, 0, Calculate, None, Github}

**TR18: {Not Number, Pos, Reset, Back, Github}**

**TR19: {Number, Neg, Reset, Reload, Github}**

TR20: {Not Number, 0, Calculate, Forward, Github}

## RESULTS



The screenshot displays the Eclipse IDE interface with the following components:

- Left Pane (Package Explorer):** Shows the test results for mortgageCalcTest [Runner: JUnit 5] (50.638 s). The results are as follows:

Test Case	Execution Time (s)
bccTR10	3.354
bccTR20	1.990
bccTR30	1.908
bccTR40	1.897
bccTR50	1.909
bccTR60	1.975
bccTR70	2.752
bccTR80	2.006
bccTR90	2.196
pwcTR10	2.280
pwcTR30	1.976
pwcTR40	2.281
pwcTR60	2.128
pwcTR70	2.224
bccTR100	3.670
pwcTR100	2.289
pwcTR140	1.994
pwcTR150	2.145
pwcTR180	3.211
pwcTR190	6.439

- Right Pane (Source Code):** Shows the source code of mortgageCalcTest.java. The code includes imports for JUnit and Selenium, and a class definition for mortgageCalcTest.

```
1 // **
2
3
4 package tests;
5
6 import static org.junit.jupiter.api.Assertions.*;
7
8 import java.util.ArrayList;
9
10 import org.junit.jupiter.api.AfterAll;
11 import org.junit.jupiter.api.AfterEach;
12 import org.junit.jupiter.api.BeforeAll;
13 import org.junit.jupiter.api.BeforeEach;
14 import org.junit.jupiter.api.Test;
15 import org.openqa.selenium.By;
16 import org.openqa.selenium.WebDriver;
17 import org.openqa.selenium.WebElement;
18 import org.openqa.selenium.chrome.ChromeDriver;
19
20 /**
21  * @author Student
22  *
23  */
24 class mortgageCalcTest {
25
```

- Bottom Pane (Coverage Report):** Shows the coverage report for mortgageCalcTest.java. The report is as follows:

Element	Coverage	Covered Inst...	Missed Instructions	Total Instructions
Showcase	100.0 %	1,775	0	1,775
src	100.0 %	1,775	0	1,775
tests	100.0 %	1,775	0	1,775
mortgageCalcTest.java	100.0 %	1,775	0	1,775

From the results, we can see that both BCC Test Cases and PWC Test Cases achieve 100% coverage on testing the input domain for the Mortgage Loan Calculator Web Application. This means that each line in our test code was successfully executed during the automated tests. Furthermore, this means that we have tested for 100% of the interface-based characteristics for this Web App, as our partitions consisted of interface-based characteristics. We can not that the inputs used for our Base Test Case, highlight the intended functionality of the Calculator as all inputs are accepted. The most crucial characteristics in order to preserve intended functionality, are Number, and Positive. Any test case that does not use Number and Positive, is meant to feed invalid inputs to see if there is a fault in the software, whether it be producing incorrect calculations, no calculations, navigating to a different page, or other faults that may exist. We know from the results that each test case was successfully executed (which includes those feeding invalid input), thus revealing that the software under test is not faulty when it comes to it's Input Domain.