Concordia University

Building a Multi-Function Scientific Calculator

Submitted by
Karan Bhalla

40047162

Submitted to Professor Pankaj Kamthan

SOEN:6011 Software Engineering Processes

Engineering and Computer Science Department

July 30, 2019

Contents

1 Changes from D1 to D2

1.1 Github Repository Url:

https://github.com/kbhalla22/SOEN6011_40047162_GroupB

1.2 Environment and Interface Requirements(Non Functional requirements)

1.2.1 Hardware(Version 1.0)

The application shall run on both IBM-compatible and Macintosh Hardware
Priority : 3

1.2.2 Software(Version 1.0)

The application shall be written in Java The application shall use only standard Java library functions(basic arithmetic). The application shall be usable on any system which supports the compiler, and shall not require any particular hardware or software.
Priority : 5

1.2.3 Operating System(Version 1.0)

The application shall be run on Windows, macOS and Linux based systems.
Priority : 3

1.2.4 Human Interfaces(Version 1.0)

The application shall function in a similar way as a regular calculator.
Priority : 4

1.2.4.1 Input The user shall input the number(s) when the application prompts to do so. The application shall ask the user to press the buttons available on the UI or the application shall allow the user to enter the numbers manually through his/her keyboard.
Priority : 5

1.2.4.2 Output The application shall display the results on the appropriate output area. Priority : 5

1.2.5 Response to undesirable events

1.2.5.1 Illegal Input Sequence Appropriate error message shall be printed, if an illegal sequence is detected.

1.2.5.2 Division By zero Division by zero shall be detected by the program and an appropriate error message is printed

1.2.5.3 Overflow/Underflow Overflow and underflow doesn't need to be detected

1.3 Reasons for selecting Algorithm 1:

1. This algorithm just computes one value (Sinx).
2. Has only 1 function call.
3. Need to implement only 1 more function.
4. Memory efficient as it uses primitive data types.
5. Simpler and fewer lines of code.

1.4 References For D1 :

1. Elizabeth Stapel, Trigonometric functions their graphs. Feb 2014. Accessed on July 27, 2019. [Online]. Available: https://www.purplemath.com/modules/triggrph2.htm
2. Anna Souza, Four Function Calculator Requirement Specification. Nov 2010. Accessed on July 27, 2019. [Online]. Available:

   http://www.mathcs.richmond.edu/~barnett/cs322/assignments/1999_fall/calculator_requirements.pdf

2 GitHub Repository(Visible to the TAs)

https://github.com/kbhalla22/SOEN6011_40047162_GroupB

PROBLEM 4

July 30, 2019

Name: Karan Bhalla
Student Id: 40047162
Concordia University

July 30, 2019

3 Debugger:

Debugging is a routine process of locating removing bugs, errors or abnormalities from programs. It helps to find subtle bugs that are not visible during code reviews or that only happens when a specific condition occurs.

3.1 Debugger used:

Eclipse Java Debugger

3.2 Advantages:

1. No need to rebuild the code to get additional information.
2. By using a debugger, a person can rewind the stack by dragging dropping yellow arrow.
3. No need to remove the println statements before deployment that otherwise would have been used for debugging purposes.
4. Provides useful information and correct location of the error through breakpoints.
5. A person can step into or step over a piece of code as per convenience.
6. The programmer can stop execution at a given point to investigate where it goes what the values are.

3.3 Disadvantages:

1. It is hard to debug multithreaded programs
2. The debuggers are not running real-time, so may not expose all the problems.
3. Learning curve: Debuggers are complicated tools. Maximizing their benefit will require time patience.
4. A sophisticated debugger may not be a good tool for beginner programmers.
5. A debugger will not solve bad code, practices or design.

4 Quality Attributes:

4.1 Correct[Wikipedia]:

Correctness from a software engineering perspective can be defined as the adherence to the specifications that determine how users can interact with the software and how the software should behave when it is used correctly. If the software behaves incorrectly, it might take a considerable amount of time to achieve the task or sometimes it is impossible to achieve it.

The project description clearly states the goal/ outcome of the project. In order to maintain the correctness of the software, only basic arithmetic operations and the Tan function is implemented. The software application displays results (double numbers) only when appropriate input is presented. The program supports decimal precision up to 16 decimal places.

4.2 Efficient:

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance.

In order to achieve efficiency, primitive data types are used in the application (Integer, Double) instead of user-defined data types as user-defined data types take up more space. In order to calculate the value of Tan(x), Only sin(x) is calculated and using this value, the value of Tan(x) can be stipulated.

4.3 Maintainable[2]:

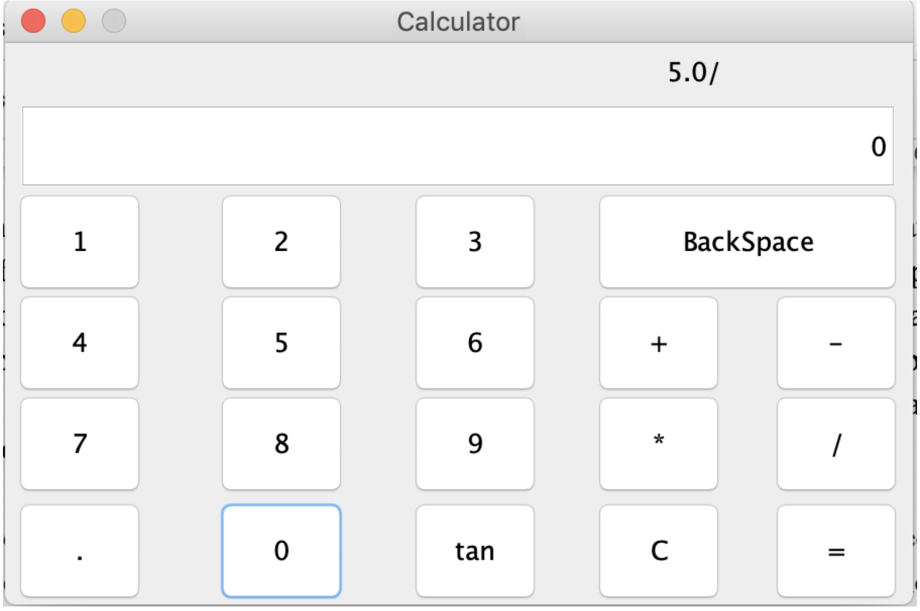
To a developer, maintainable code simply means âĂIJcode that is easy to modify or extendâĂİ. At the heart of maintainability is carefully constructed code that is easy to read; code that is easy to dissect in order to locate the particular component relating to a given change request; code that is then easy to modify without the risk of starting a chain reaction of breakages in dependant modules.

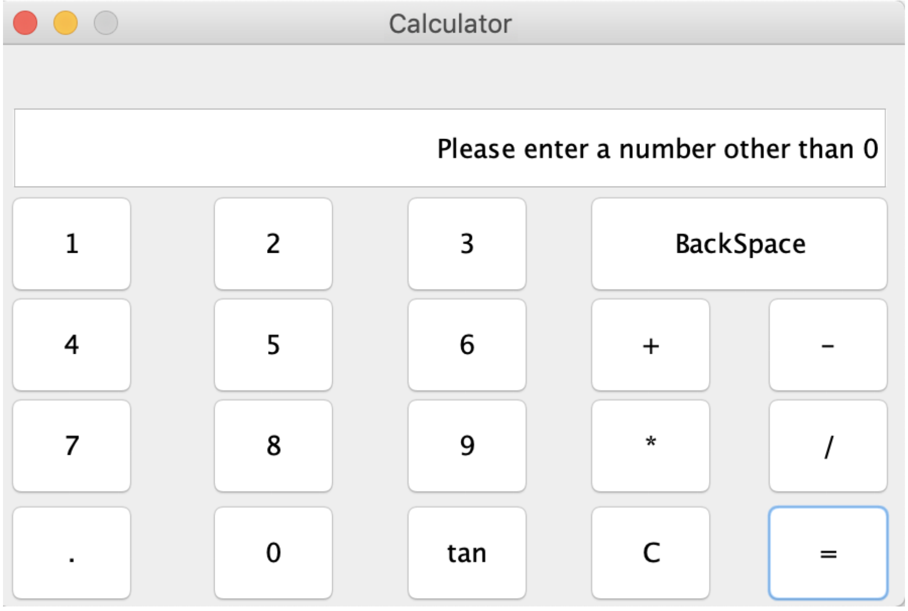
In order to make the code maintainable, the structure has been designed in such a way that in order to add more features, the programmer just needs to add a new button on the design panel. Moreover, all the logic has been built using switch cases which can be easily extended. Each switch case corresponds to a method call (functionality). The application is designed keeping in mind the ability to extend it in the future and in order to achieve this, a lot of emphases has been put on making the code more readable.

4.4 Robust:

Robust programming is a style of programming that focuses on handling unexpected termination and unexpected actions. It requires code to handle these terminations and actions gracefully by displaying accurate and unambiguous error messages. These error messages allow the user to more easily debug the program.

In order to make the code robust, proper error handling is used through the use of error messages. All instances of incorrect inputs, irregular button presses have been thoroughly checked and proper error messages.





Please enter a number other than 0

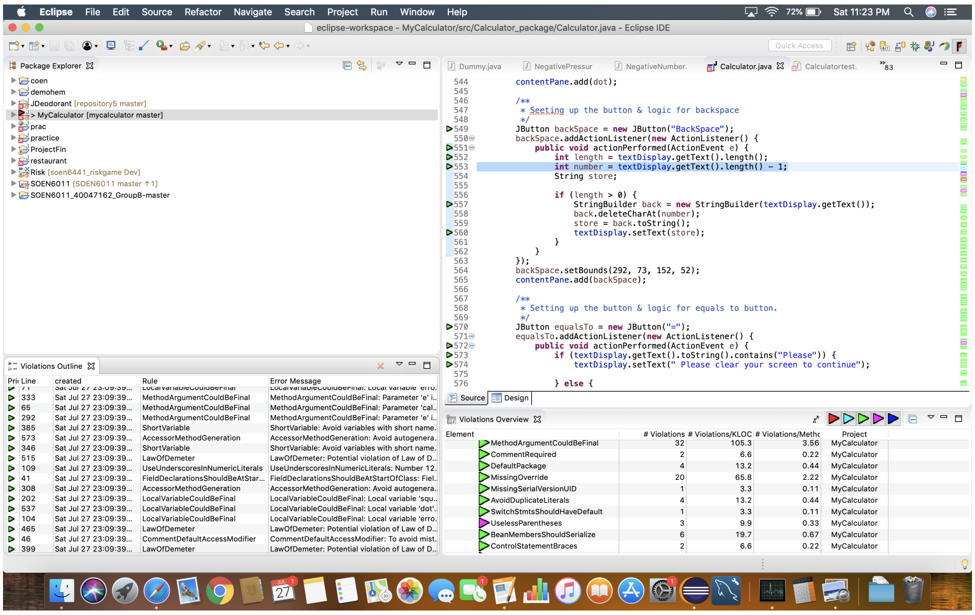
4.5 Usable:

The basic purpose of designing a GUI for the application was to make the application usable. The application is pretty much similar to a regular calculator with minimal buttons. The users can enter the data through the buttons or through the textbox available on the GUI. The users can obtain results by pressing the appropriate button.

5 Quality of Source Code:

5.1 Tool used: PMD

PMD is an open-source static source code analyzer that reports on issues found within application code. PMD includes built-in rule sets and supports the ability to write custom rules. For reference, a PMD report has also been generated is present in the submitted folder[Wikipedia].


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
72% Sat 11:23 PM
eclipse-workspace - MyCalculator/src/Calculator_package/Calculator.java - Eclipse IDE
Quick Access
Package Explorer
coen
demohem
JDeodorant [repository5 master]
> MyCalculator [mycalculator master]
prac
practice
ProjectFin
restaurant
Risk [soen6441_riskgame Dev]
SOEN6011 [SOEN6011 master ↑1]
SOEN6011_40047162_GroupB-master
Dummy.java
NegativePressur
NegativeNumber.
Calculator.java
Calculatortest.
»83
Source
Design
Violations Outline
Violations Overview

```
        contentPane.add(dot);

        /**
         * Seeting up the button & logic for backspace
         */
        JButton backSpace = new JButton("BackSpace");
        backSpace.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int length = textDisplay.getText().length();
                int number = textDisplay.getText().length() - 1;
                String store;

                if (length > 0) {
                    StringBuilder back = new StringBuilder(textDisplay.getText());
                    back.deleteCharAt(number);
                    store = back.toString();
                    textDisplay.setText(store);
                }
            }
        });
        backSpace.setBounds(292, 73, 152, 52);
        contentPane.add(backSpace);

        /**
         * Setting up the button & logic for equals to button.
         */
        JButton equalsTo = new JButton("=");
        equalsTo.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                if (textDisplay.getText().toString().contains("Please")) {
                    textDisplay.setText(" Please clear your screen to continue");

                } else {
```

Pri	Line	created	Rule	Error Message
▷	71	Sat Jul 27 23:09:39...	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'erro.
▷	333	Sat Jul 27 23:09:39...	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'e' i..
▷	65	Sat Jul 27 23:09:39...	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'cal..
▷	292	Sat Jul 27 23:09:39...	MethodArgumentCouldBeFinal	MethodArgumentCouldBeFinal: Parameter 'e' i..
▷	385	Sat Jul 27 23:09:39...	ShortVariable	ShortVariable: Avoid variables with short name.
▷	573	Sat Jul 27 23:09:39...	AccessorMethodGeneration	AccessorMethodGeneration: Avoid autogenera..
▷	346	Sat Jul 27 23:09:39...	ShortVariable	ShortVariable: Avoid variables with short name.
▷	515	Sat Jul 27 23:09:39...	LawOfDemeter	LawOfDemeter: Potential violation of Law of D...
▷	109	Sat Jul 27 23:09:39...	UseUnderscoresInNumericLiterals	UseUnderscoresInNumericLiterals: Number 12.
▷	41	Sat Jul 27 23:09:39...	FieldDeclarationsShouldBeAtStar...	FieldDeclarationsShouldBeAtStartOfClass: Fiel..
▷	308	Sat Jul 27 23:09:39...	AccessorMethodGeneration	AccessorMethodGeneration: Avoid autogenera..
▷	202	Sat Jul 27 23:09:39...	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'squ..
▷	537	Sat Jul 27 23:09:39...	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'dot'..
▷	104	Sat Jul 27 23:09:39...	LocalVariableCouldBeFinal	LocalVariableCouldBeFinal: Local variable 'erro.
▷	465	Sat Jul 27 23:09:39...	LawOfDemeter	LawOfDemeter: Potential violation of Law of D...
▷	46	Sat Jul 27 23:09:39...	CommentDefaultAccessModifier	CommentDefaultAccessModifier: To avoid mist.
▷	399	Sat Jul 27 23:09:39...	LawOfDemeter	LawOfDemeter: Potential violation of Law of D...

Element	# Violations	# Violations/KLOC	# Violations/Methc	Project
▷ MethodArgumentCouldBeFinal	32	105.3	3.56	MyCalculator
▷ CommentRequired	2	6.6	0.22	MyCalculator
▷ DefaultPackage	4	13.2	0.44	MyCalculator
▷ MissingOverride	20	65.8	2.22	MyCalculator
▷ MissingSerialVersionUID	1	3.3	0.11	MyCalculator
▷ AvoidDuplicateLiterals	4	13.2	0.44	MyCalculator
▷ SwitchStmtsShouldHaveDefault	1	3.3	0.11	MyCalculator
▷ UselessParentheses	3	9.9	0.33	MyCalculator
▷ BeanMembersShouldSerialize	6	19.7	0.67	MyCalculator
▷ ControlStatementBraces	2	6.6	0.22	MyCalculator

5.2 Advantages[Wikipedia]:

1. PMD can easily find Possible bugs like Empty try/catch/finally/switch blocks
2. Captures dead code- Unused local variables, parameters and private methods.
3. Captures empty if/while statements.
4. Produces Suboptimal codeâĂŤWasteful String/StringBuffer usage.
5. Identifies Classes with high Cyclomatic Complexity measurements.
6. Identifies Duplicate codeâĂŤCopied/pasted code can mean copied/pasted bugs, and decreases maintainability.

5.3 Disadvantages[1]:

1. PMD does not report compilation errors
2. it only can process well-formed source files

PROBLEM 6

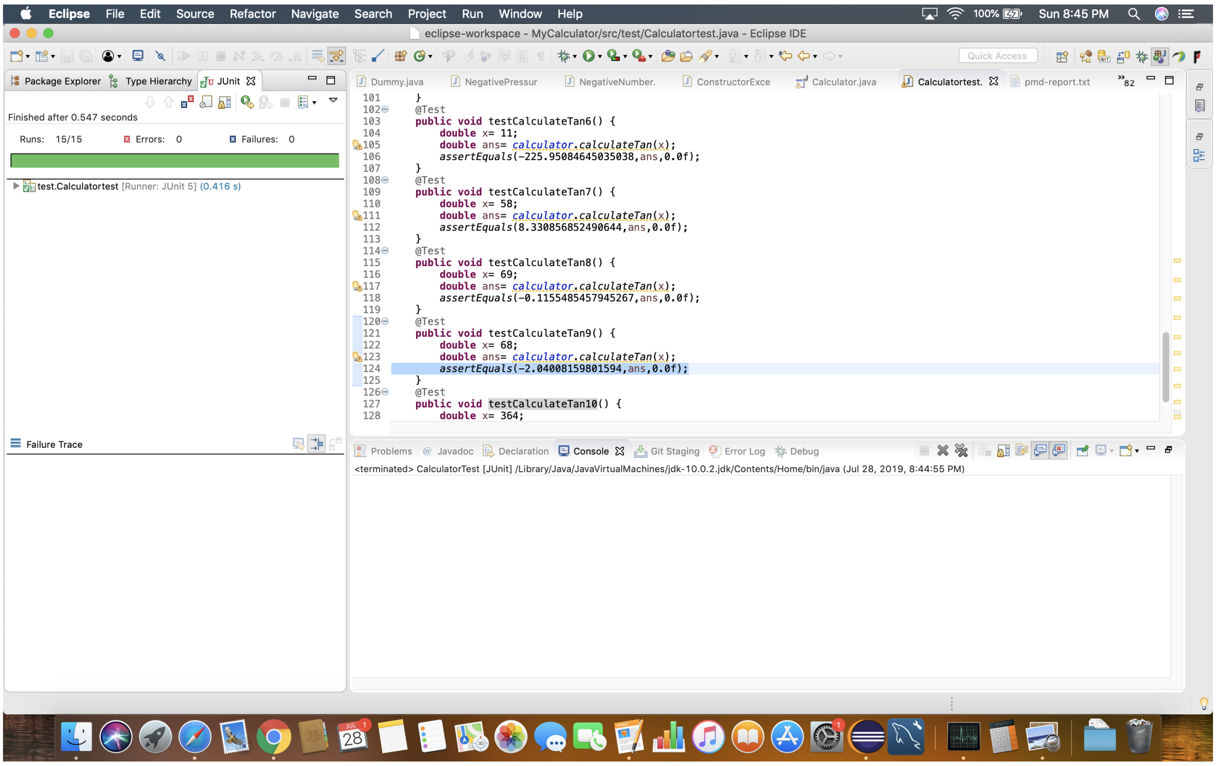
July 30, 2019

6 JUNIT test cases:

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

JUnit4 has been used to write the test cases.

The test cases named testAdd(), testSubtract(), testMultiply(), testDivide(), testCalculateTan1-10 correspond to requirement 4.2
Test case divideByZero() corresponds to requirement 5.2


Eclipse File Edit Source Refactor Navigate Search Project Run Window Help
100% Sun 8:45 PM
Quick Access
Package Explorer Type Hierarchy JUnit
Dummy.java NegativePressur NegativeNumber. ConstructorExce Calculator.java Calculatortest. pmd-report.txt
Failure Trace
Problems Javadoc Declaration Console Git Staging Error Log Debug

eclipse-workspace - MyCalculator/src/test/Calculatortest.java - Eclipse IDE

Finished after 0.547 seconds

Runs: 15/15 Errors: 0 Failures: 0

test.Calculatortest [Runner: JUnit 5] (0.416 s)

```
    }
    @Test
    public void testCalculateTan6() {
        double x= 11;
        double ans= calculator.calculateTan(x);
        assertEquals(-225.95084645035038,ans,0.0f);
    }
    @Test
    public void testCalculateTan7() {
        double x= 58;
        double ans= calculator.calculateTan(x);
        assertEquals(8.330856852490644,ans,0.0f);
    }
    @Test
    public void testCalculateTan8() {
        double x= 69;
        double ans= calculator.calculateTan(x);
        assertEquals(-0.1155485457945267,ans,0.0f);
    }
    @Test
    public void testCalculateTan9() {
        double x= 68;
        double ans= calculator.calculateTan(x);
        assertEquals(-2.04008159801594,ans,0.0f);
    }
    @Test
    public void testCalculateTan10() {
        double x= 364;
```

```
<terminated> CalculatorTest [JUnit] /Library/Java/JavaVirtualMachines/jdk-10.0.2.jdk/Contents/Home/bin/java (Jul 28, 2019, 8:44:55 PM)
```

References:

1. John Tobler, Checkstyle Vs PMD , Nov 2011. Accessed on: July 28, 2019. [Online]. Available: https://stackoverflow.com/questions/184563/checkstyle-vs-pmd

2. Robert Harvey, What characteristics make code maintainable?, Feb 2012. Accessed on: July 28, 2019. [Online]. Available: https://softwareengineering.stackexchange.com/questions/134855/what-characteristics-or-features-make-code-maintainable

3. Robert Hundt, United States Patent Application Publication. Oct 14, 2004