



SOEN 6011 - Software Engineering Processes

Summer 2019

Scientific Calculator- ETERNITY: FUNCTIONS

Project Report

Deliverable 3

Presented to

Instructor: PANKAJ KAMTHAN

By
Prashanthi Ramesh

August 3, 2019

Contents

1	Source Code Review	2
1.1	Manual Code Review	2
1.2	Automatic Code Review	4
2	Testing	5
2.1	Environment	5
2.2	Procedure	5
A	GitHub	8
A.1	Individual GitHub Link	8
A.2	Source Code Review Github Link of Himansi Patel	8
A.3	Testing Github Link of Nirav Patel	8
A.4	Team GitHub Link	8

Chapter 1

Source Code Review

Eternity Function F1: arccos(x) developed by Himansi Patel

1.1 Manual Code Review

Use of Static Methods

Non-static methods[2] defined in classes HelperFunction and ArccosFunction - calculatePI(), calculateFactorial(int number), calculatePower(double base, int exponent), arccos(double num) can be declared static.

```
public double calculatePI() {
    double pi = 0.0;
    for (int k = 0; k < 9999; k++) {
        double numerator = calculatePower(-1, k);
        double denominator = (2 * k) + 1;
        double value = numerator / denominator;
        pi = pi + value;
    }
    pi = 4 * pi;
    return pi;
}
```

Figure 1.1: A non-static method in HelperFunction class that can be declared as static

- Methods are declared static as they do not hold the state of an object of the class

Missing Javadoc Comments

Unit test methods in classes ArccosTest and HelperFunctionTest can include javadoc comments to quickly understand the purpose of the tests.

```

private ArccosFunction arccosTestObj = new ArccosFunction();

@Test
public void testArccos1() {
    double calculatedArccos = arccosTestObj.arccos(3);
    double mathArccos = Math.acos(3);
    assertEquals(calculatedArccos, mathArccos, 0.00000005);
}

@Test
public void testArccos2() {
    double calculatedArccos = arccosTestObj.arccos(-1.5);
    double mathArccos = Math.acos(-1.5);
    assertEquals(calculatedArccos, mathArccos, 0.00000005);
}

```

Figure 1.2: Missing comments in Unit Test Methods

Naming Conventions

Unit test methods in classes `ArccosTest` and `HelperFunctionTest` can include self-descriptive method names to quickly understand the purpose of the tests.

```

@Test
public void testArccos1() {
    double calculatedArccos = arccosTestObj.arccos(3);
    double mathArccos = Math.acos(3);
    assertEquals(calculatedArccos, mathArccos, 0.00000005);
}

@Test
public void testArccos2() {
    double calculatedArccos = arccosTestObj.arccos(-1.5);
    double mathArccos = Math.acos(-1.5);
    assertEquals(calculatedArccos, mathArccos, 0.00000005);
}

@Test
public void testArccos3() {
    double calculatedArccos = arccosTestObj.arccos(-0.9);
    double mathArccos = Math.acos(-0.9);
    double calculatedRoundedArccos = (double) Math.round(calculatedArccos * 100d) / 100d;
    double mathRoundedArccos = (double) Math.round(mathArccos * 100d) / 100d;
    assertEquals(calculatedRoundedArccos, mathRoundedArccos, 0.00000005);
}

```

Figure 1.3: Method names that can be changed to self-descriptive names

1.2 Automatic Code Review

- For automatic source code review CheckStyle[1] plugin for eclipse IDE was used.
- The Google Checks configuration was used to check if there are any deviations from the defined set of coding rules.
- The analysis showed no violations in the source code.

CheckStyle Plugin

Checkstyle inspects your Java source code and pointing out items that deviate from a defined set of coding rules.

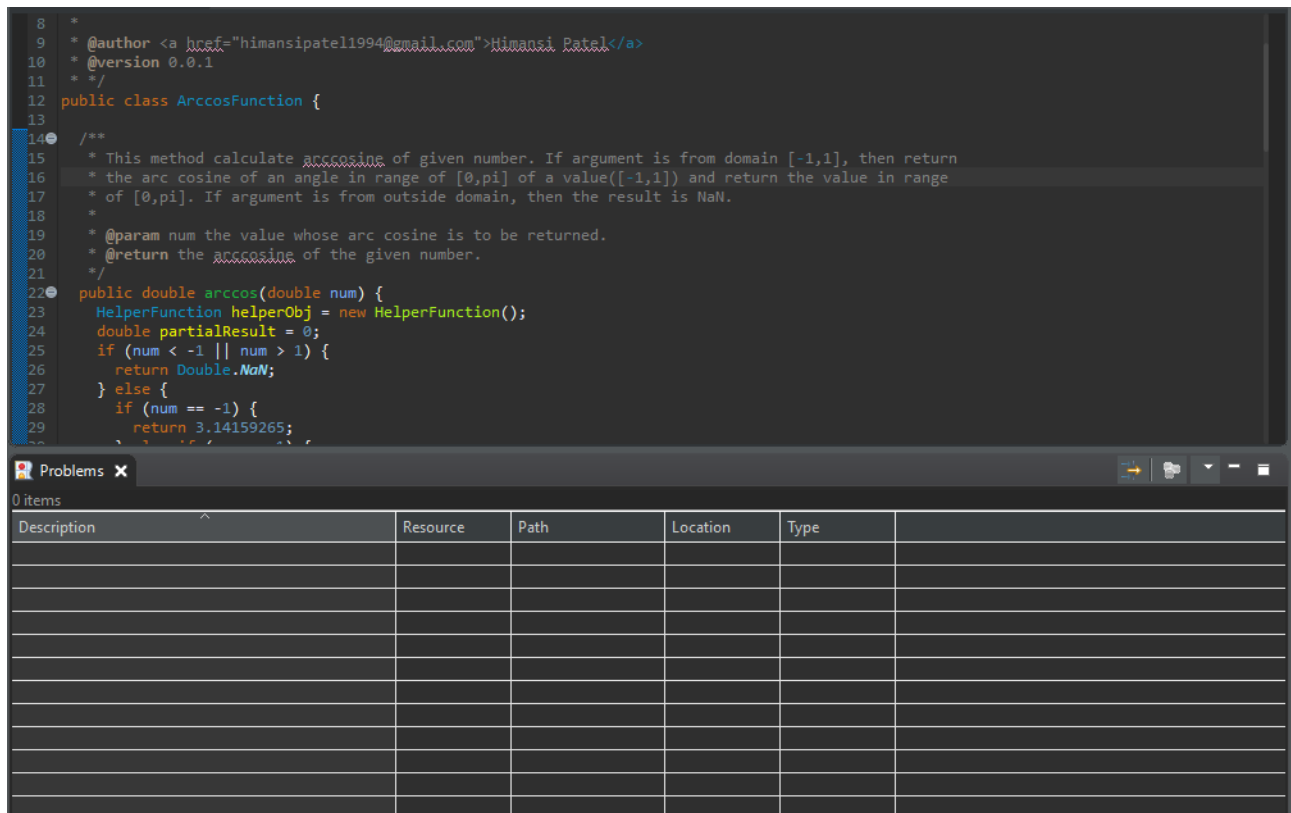


Figure 1.4: No violations generated by CheckStyle Plugin in Eclipse IDE

Chapter 2

Testing

Eternity Function F2: $\tan(x)$ developed by Nirav Patel

2.1 Environment

- Unit tests are created in Java in a separate project or separate source folder to keep the test code separate from the real code.
- The standard convention from the Maven and Gradle build tools is to use:

`src/test/java` - for test classes

2.2 Procedure

All Unit test cases created by the developer are working fine.

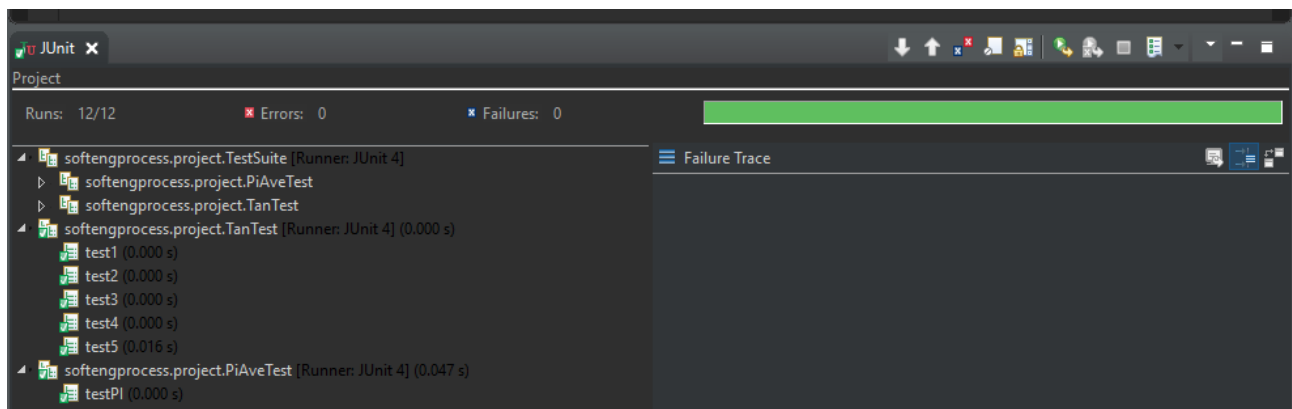
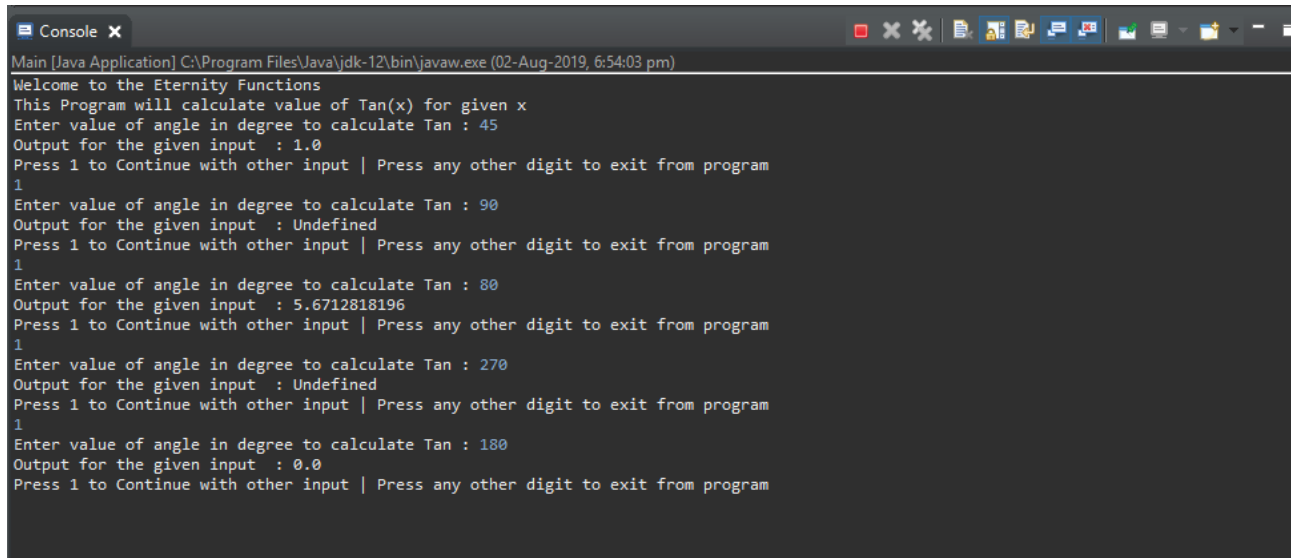
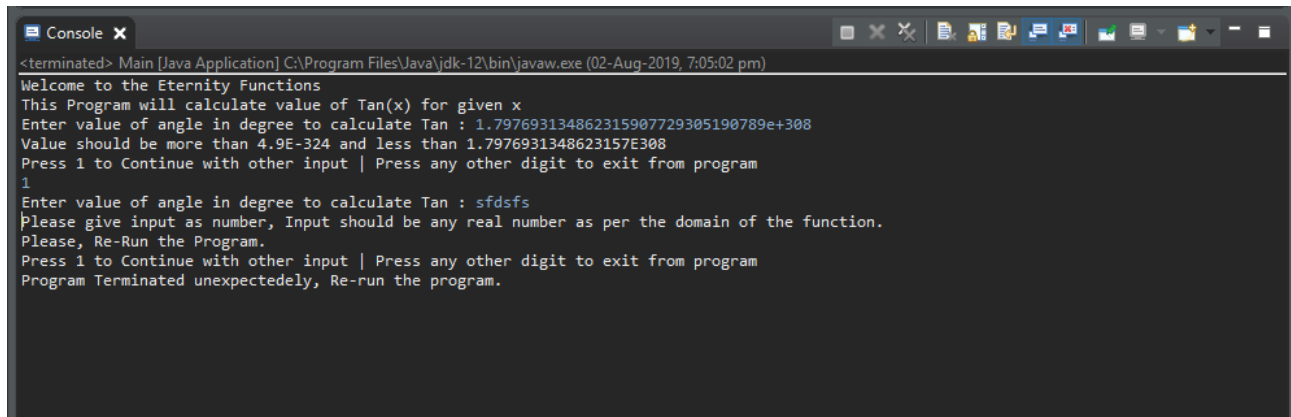


Figure 2.1: Test cases are passed in JUnit[3] in Eclipse IDE



```
Console X
Main [Java Application] C:\Program Files\Java\jdk-12\bin\javaw.exe (02-Aug-2019, 6:54:03 pm)
Welcome to the Eternity Functions
This Program will calculate value of Tan(x) for given x
Enter value of angle in degree to calculate Tan : 45
Output for the given input : 1.0
Press 1 to Continue with other input | Press any other digit to exit from program
1
Enter value of angle in degree to calculate Tan : 90
Output for the given input : Undefined
Press 1 to Continue with other input | Press any other digit to exit from program
1
Enter value of angle in degree to calculate Tan : 80
Output for the given input : 5.6712818196
Press 1 to Continue with other input | Press any other digit to exit from program
1
Enter value of angle in degree to calculate Tan : 270
Output for the given input : Undefined
Press 1 to Continue with other input | Press any other digit to exit from program
1
Enter value of angle in degree to calculate Tan : 180
Output for the given input : 0.0
Press 1 to Continue with other input | Press any other digit to exit from program
```

Figure 2.2: Test Cases that pass using JUnit are verified



```
Console X
<terminated> Main [Java Application] C:\Program Files\Java\jdk-12\bin\javaw.exe (02-Aug-2019, 7:05:02 pm)
Welcome to the Eternity Functions
This Program will calculate value of Tan(x) for given x
Enter value of angle in degree to calculate Tan : 1.797693134862315907729305190789e+308
Value should be more than 4.9E-324 and less than 1.7976931348623157E308
Press 1 to Continue with other input | Press any other digit to exit from program
1
Enter value of angle in degree to calculate Tan : sfdsf
Please give input as number, Input should be any real number as per the domain of the function.
Please, Re-Run the Program.
Press 1 to Continue with other input | Press any other digit to exit from program
Program Terminated unexpectedly, Re-run the program.
```

Figure 2.3: Test for valid input within the specified range in the requirement document

Apart from the test cases written by the developer, I have performed the following tests.

Table 2.1: Test Results

ID	Date	Input(degree)	Expected Output	Actual Output	State (Pass/Fail)
1	2 August, 2019	0	0	0	Pass
2	2 August, 2019	-360	0	0	Pass
3	2 August, 2019	-45	-1	-1	Pass
4	2 August, 2019	-90	Undefined	Undefined	Pass
5	2 August, 2019	-80	-5.67128	-5.67128	Pass
6	2 August, 2019	-270	Undefined	Undefined	Pass
7	2 August, 2019	-180	0	0	Pass
8	2 August, 2019	53	1.32704	1.32704	Pass
9	2 August, 2019	99	-6.31375	-6.31375	Pass
10	2 August, 2019	-99	6.31375	6.31375	Pass

Appendix A

GitHub

A.1 Individual GitHub Link

<https://github.com/PrashanthiRamesh/SOEN-6011-Project-Calculator/>

A.2 Source Code Review Github Link of Himansi Patel

<https://github.com/Himansipatel/SOEN-6011-Team-H-Himansi/>

A.3 Testing Github Link of Nirav Patel

<https://github.com/niravjdn/SOEN-6011-Project-Team-H-Nirav/>

A.4 Team GitHub Link

<https://github.com/niravjdn/SOEN-6011-Project/>

Bibliography

- [1] CheckStyle. *Eclipse Checkstyle Plugin*. 2019. URL: <https://checkstyle.org/eclipse-cs/#!/> (visited on 01/01/2019).
- [2] MIT. *Reading 4: Code Review*. 2015. URL: <http://web.mit.edu/6.005/www/fa15/classes/04-code-review/> (visited on 01/01/2015).
- [3] vogella. *Unit Testing with JUnit*. 2007. URL: <https://www.vogella.com/tutorials/JUnit/article.html> (visited on 01/01/2016).