Function F1: Arccos(x)
SOEN-6011 Project
Deliverable- 3
Problem- 5

Code Review Report of Function F2: $tan(x)$

Birva Shah (Student ID: 40070973)

2 August 2019

Github link: https://github.com/BirvaShah/SOEN6011_Project
Function F2 reviewed from Github link: https://github.com/Hetvishah1710/tanx

1 Code Review

Code review which is also called peer review is a phase in the software development process in which the programmers and peer reviewers come together to review a code. Their main aim is to find potential errors, check consistency of program design and to check adherence to coding standards and proper comments. This phase is relatively inexpensive rather than the more expensive process of handling, locating, and fixing bugs during later stages of development or after programs are delivered to users. There are various approaches and tools available for reviewing a source code, for example, PMD(Eclipse IDE plugin), Gerrit, Github, SpotBugs(Eclipse IDE Plugin), etc. We can also review a code manually.

2 Manual Code Review

I have reviewed F2 function source code manually as well as using tools. In manual code review, I have checked the coding convention styles that we as a team had decided to follow.

- The source code is compiling successfully.

- I have checked the input and output of the code for verification. The author has put an effort to use a Memento pattern to enhance the design of the function.

- Error handling is properly done for values outside the domain as shown in Figure 1. However, if the user input is not a number then the program throws un-handled exceptions as an error. I would suggest the author to use exception handling to handle NaN inputs.

```
**************************************
* Press 1 to compute tan value      *
* Press 2 to retrive previous result *
* Press 3 to store previous value    *
* Press 4 exit                       *
**************************************

Enter your choice :
1

Enter the value of x in degree :
90
Value of x should be within its domain!!
```

Figure 1

- The author has followed the coding styles particularly naming conventions of classes and functions as we had decided in a team. However, some variable names are found to be not adhering the purpose.

- From Figure 2, I would suggest the author to differentiate the main output and the output of supporting functions. Our calculator aims to give the clear output of the desired function only. Moreover, it is suggested that before committing the code, author should check the spelling/grammar errors seen in the output. So that the user do not misunderstand or misinterpret the output.

```
Tan [Java Application] C:\Program Files\Java\jdk1.8.0_161\bir
*****************************************
*        This is tan function calclator     *
*****************************************


***************************************
* Press 1 to compute tan value         *
* Press 2 to retrive previous result *
* Press 3 to store previous value      *
* Press 4 exit                         *
***************************************

Enter your choice :
1

Enter the value of x in degree :
25.236598745874566
****************************
*              Result           *
****************************
* value of sin :0.42636
* value of cos :0.90455
* value of tan :0.47135
****************************
```

Figure 2

3   Automated Code review using Tools

Automated code review software checks source code for compliance with a predefined set of rules or best practices. The use of analytical methods to inspect and review source code to detect bugs has been a standard development practice. This process can be accomplished both manually and in an automated fashion. With automation, software tools provide assistance with the code review and inspection process. The review program or tool typically displays a list of warnings (violations of programming standards). [**Wikipedia**]

3.1   Tools used to conduct code review

3.1.1   PMD:

PMD stands for Programming Mistake Detector.It is a free source code analysis tool which helps us to find the bugs in our java code and improve the code quality. I have used Eclipse PMD plugin to review the code.

- Figure 3 shows the number of violations detected in the source code of Function $tan(x)$. The number of violations are more because of the console (textual) output of the function. The function uses SystemPrintln rule to print the output which is considered as violation in any automated tool. However, we can ignore/remove such type of violations as the author may have chose to make a console application according to Project Description. Other violations such as DataFlowAnomalyAnalysis, LocalVariableCouldBeFinal can be removed from violations by Right-clicking on that violation in Violations Outline in PMD and disabling the rule. Such violations are important only in industrial based large projects, I have disabled above three unnecessary violations for this code review purpose.

Element	# Violations	# Violations/KLOC	# Violations/Method	Project
> ⊞ test	49	N/A	N/A	Assignment_2
∨ ⊞ main	75	477.7	4.17	Assignment_2
> Ⓙ Tan.java	60	458.0	6.00	Assignment_2
> Ⓙ DriverMemento.java	6	1000.0	3.00	Assignment_2
> Ⓙ CreateMemento.java	4	666.7	2.00	Assignment_2
> Ⓙ BeginMemento.java	5	357.1	1.25	Assignment_2

Figure 3

- Figure 4 shows some of the warning violations.

P	Line	created	Rule	Error Message
▶	235	Thu Aug 01 11:33:54 EDT 2019	DataflowAnomalyAnalysis	Found 'DD'-anomaly for variable 'fact' (lines '235'-'241').
▶	191	Thu Aug 01 11:33:54 EDT 2019	DataflowAnomalyAnalysis	Found 'DU'-anomaly for variable 'flag' (lines '191'-'206').
▶	217	Thu Aug 01 11:33:54 EDT 2019	DataflowAnomalyAnalysis	Found 'DD'-anomaly for variable 'output' (lines '217'-'224').
▶	165	Thu Aug 01 11:33:54 EDT 2019	DataflowAnomalyAnalysis	Found 'DU'-anomaly for variable 'flag' (lines '165'-'180').

Figure 4

- Figure 5 shows some of the important violations which can be fixed first to avoid future conflicts and violations.

P	Line	created	Rule	Error Message
▶	5	Thu Aug 01 11:35:07 EDT 2019	UnusedImports	Avoid unused imports such as 'javax.swing.plaf.synth.SynthSeparatorUI'
▶	20	Thu Aug 01 11:35:07 EDT 2019	ShortClassName	Avoid short class names like Tan

Figure 5

- Figure 6 shows some of the urgent violations for which the actions can be taken at high priority.

P	Line	created	Rule	Error Message
▷	140	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like r
▷	187	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like r
▷	169	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like n
▷	188	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like pi
▷	177	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like df
▷	21	Thu Aug 01 17:21:32 EDT 2019	CommentRequired	Field comments are required
▷	128	Thu Aug 01 17:21:32 EDT 2019	OnlyOneReturn	A method should have only one exit point, and that should be the last state...
▷	233	Thu Aug 01 17:21:32 EDT 2019	ShortVariable	Avoid variables with short names like r
▷	77	Thu Aug 01 17:21:32 EDT 2019	AvoidLiteralsInIfCondition	Avoid using Literals in Conditional Statements
▷	112	Thu Aug 01 17:21:32 EDT 2019	CommentSize	Comment is too large: Line too long
▷	151	Thu Aug 01 17:21:32 EDT 2019	AvoidDuplicateLiterals	The String literal ".#####" appears 4 times in this file; the first occurrence is ...

Figure 6: Tan.java

- Figure 7 shows critical violations with error messages. Author can consider looking into it as it may result in code failure.

P	Line	created	Rule	Error Message
▷	187	Thu Aug 01 17:15:00 EDT 2019	AvoidReassigningParameters	Avoid reassigning parameters such as 'r'
▷	161	Thu Aug 01 17:15:00 EDT 2019	AvoidReassigningParameters	Avoid reassigning parameters such as 'r'

Figure 7

- There are no **blocker violations** in code.

3

3.1.2   Checkstyle:

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to reduce the task of programmers. This makes it ideal for projects that want to enforce a coding standard. [**Sourceforge**]

I have ran the checkstyle plugin from Eclipse IDE to check the adherence of coding standards maintained in the source code. As shown in figure 8, there are 0 violations found. **This means that the author has taken care of writing a code by following the best practices and coding standards as defined in Checkstyle.**
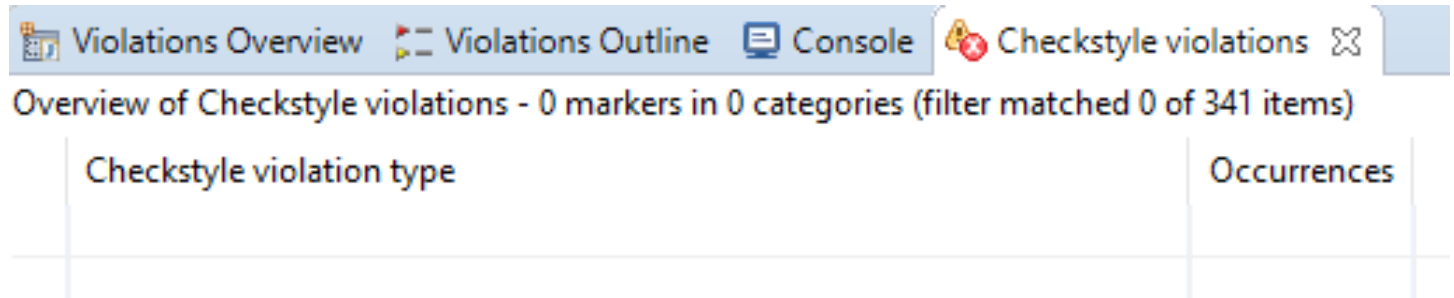

Figure 8

References

[1] Search Software Quality,
    https://searchsoftwarequality.techtarget.com/definition/code-review

[2] Wikipedia,
    https://en.wikipedia.org/wiki/Automated_code_review

[3] Sourceforge,
    https://checkstyle.sourceforge.io/