

SOEN6011 (DELIVERABLE 3)

<https://github.com/vik-sandhu/soen.6011>

Source Code Review

Author:
VIKRAMJIT SINGH

Student ID:
40075774

August 3, 2019



Contents

1	Source Code Review	$\sinh(x)$	2
1.1	Automatic source code review	2
1.2	Manual Source Code Review	3
2	Testing Review	$\log_b(x)$	4

Problem 5

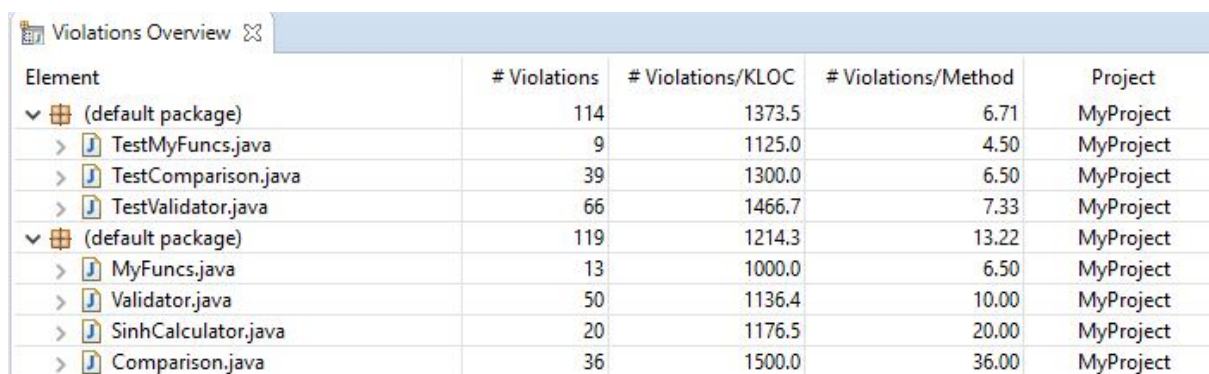
1 Source Code Review $\sinh(x)$

According to the professor's instructions, for problem 7, F2 reviews F3. So, for problem 5, I reviewed source code of $\sinh(x)$. I reviewed this source code both manually and automatically.

1.1 Automatic source code review

For Automated source code review, I used PMD (Programming Mistake Detector), which is a open source static source code analyzer that reports on issues found within application code. Issues announced by PMD are about inefficient code, or terrible programming propensities, which can decrease the exhibition and viability of the program.

I used PMD plugin in Eclipse IDE to review the source code. The report (text file) generated by PMD is attached within this deliverable(D3) under the name of pmd-report. The violations in the source code are listed in this report. The images below shows various violations shown on console.



The screenshot shows the 'Violations Overview' window in Eclipse. It contains a table with the following data:

Element	# Violations	# Violations/KLOC	# Violations/Method	Project
▼ [icon] (default package)	114	1373.5	6.71	MyProject
> [icon] TestMyFuncs.java	9	1125.0	4.50	MyProject
> [icon] TestComparison.java	39	1300.0	6.50	MyProject
> [icon] TestValidator.java	66	1466.7	7.33	MyProject
▼ [icon] (default package)	119	1214.3	13.22	MyProject
> [icon] MyFuncs.java	13	1000.0	6.50	MyProject
> [icon] Validator.java	50	1136.4	10.00	MyProject
> [icon] SinhCalculator.java	20	1176.5	20.00	MyProject
> [icon] Comparison.java	36	1500.0	36.00	MyProject

Figure 1: PMD console output

In the Figure 2

- Red arrow depicts Blocker Violations.
- Turquoise arrow depicts Critical Violations.
- Green arrow depicts Urgent Violations.

Priority	Line	created	Rule	Error Message
▶	4	Fri A...	ClassNamingCo...	ClassNamingConventions: The utility class name 'MyFuncs' doesn't match '[A-Z][a-zA-Z0-9]*(Utils? Helper)'
▶	11	Fri A...	AvoidReassignin...	AvoidReassigningParameters: Avoid reassigning parameters such as 's'
▶	16	Fri A...	LawOfDemeter	LawOfDemeter: Potential violation of Law of Demeter (object not created locally)
▶	11	Fri A...	ShortVariable	ShortVariable: Avoid variables with short names like s
▶	4	Fri A...	NoPackage	NoPackage: All classes, interfaces, enums and annotations must belong to a named package
▶	5	Fri A...	UseUtilityClass	UseUtilityClass: All methods are static. Consider using a utility class instead. Alternatively, you could add a private constructor or make the class abstract to silence this warning.
▶	30	Fri A...	MethodArgume...	MethodArgumentCouldBeFinal: Parameter 'n' is not assigned and could be declared final
▶	30	Fri A...	ShortVariable	ShortVariable: Avoid variables with short names like x
▶	30	Fri A...	ShortVariable	ShortVariable: Avoid variables with short names like n
▶	30	Fri A...	MethodArgume...	MethodArgumentCouldBeFinal: Parameter 'x' is not assigned and could be declared final
▶	14	Fri A...	ShortVariable	ShortVariable: Avoid variables with short names like i
▶	13	Fri A...	DataflowAnoma...	DataflowAnomalyAnalysis: Found 'DD'-anomaly for variable 'len' (lines '13'-'18').
▶	18	Fri A...	DataflowAnoma...	DataflowAnomalyAnalysis: Found 'DD'-anomaly for variable 'len' (lines '18'-'18').

Figure 2: PMD console output

- Pink arrow depicts Important Violations.
- Blue arrow depicts Warning Violations

1.2 Manual Source Code Review

After comprehending the entire source code, I came across various flaws in this source code which are listed below.

- Our team decided to abide by Google Java Style Guide. But coding standards are not followed throughout the project.
- Incorrect indentation throughout the project.
- Class Naming Conventions are also not followed, in the whole project.
- In the whole project, the variables names used are very small like i,x. Instead meaning full variables names should be used.
- The class Comparison has very high complexity. Complexity can be reduced using efficient loops or a better approach.
- Only variables that are final can contain underscore in their names, but in class Comparison contains many variables that are not final and contain underscore in their names. For example: value_positive, abs_x etc.
- Avoid using literals in conditional statements.
- Value of some local variables like len, count is never used.
- Resource Leak: Scanner sc is never closed.
- Public methods and constructors comments are missing.
- Some comments are too long.

Problem 7

2 Testing Review $\log_b(x)$

$\tan(x)$ (F2) is the function assigned to me. According to the professor's instructions, for problem 7, F2 reviews F4. So, for problem 7, I tested $\log_b(x)$.

I used JUnit to compute the result of the test cases. All the five test cases passed and it took just 0.031 seconds to all run the test cases which proves system efficiency. These test cases covered mostly all the functionality. The programmer had a clear and user friendly GUI.

*

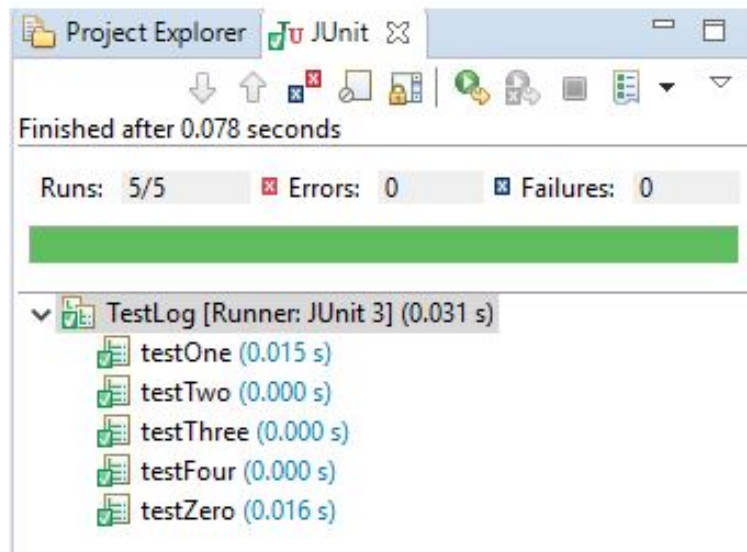


Figure 3: Test Output

During testing, I also found certain inconsistencies which are listed below.

- Missing Javadoc comments.
- Incorrect indentation.

- Some lines contained tab character.
- Should have added more test cases to show what happens when negative value is entered by the user.
- Though the test case testOne is passed in JUnit, but in this test case the output given by the function is wrong. Value of $\log_1(1)$ is undefined, but according to this program output is 0, which according to me is a major discrepancy.
- According to FR2, x can never be 0. But in test case testZero, user input x as 0 and still gets output from the program. In this case, output should be Math Error.

References

- [1] <https://en.wikipedia.org/wiki/PMD>
- [2] <https://en.wikipedia.org/wiki/JUnit>