SOEN 6011 PROJECT DELIVERY 2-2:
REVIEWS OF CALCULATOR OF FUNCTION 3(CF3) APPLICATION

August 2, 2019

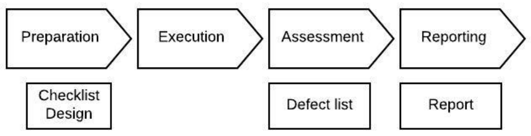
Xueying Li 40036265
https://github.com/raphealshirley/SOEN6011-PROJECT
Source code reviewed: Yanzhen Li
Test reviewed: Liangzhao Lin

1 Source Code Review

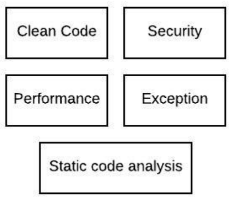
Code review is systematic examination of computer source code. It is intended to find mistakes overlooked in the initial development phase, improving the overall quality of software[1].

1.1 Code Review Methodology

Main phrases of code review are showing as figure 1a. It includes preparation, execution, assessment and reporting.
In this code reviewing process, an inspection was performed to find anomalies and to generate anomalies list though using Java Code Review Checklist[2].
Checklist is an important approach to assist the reviewer to generate a defect list. Utilizing Java Code Review Checklist to review could help evaluate the quality of the java code, and in our reviewing process, quality attributes taken into consideration are clean code, security, performance, static code analysis and error handling.And figure 1b shows all the attributes.



(a) Methodology Phases



(b) Quality Attributes

Figure 1: Code Review Methodology

1.2 Checklists

In our review process, a modified java code review check list is used based on[2]. Following gives the modified checklist.

Table 1: Checklist for Clean Code

Clean Code		
Category	Checklist Item	File-Line
Meaningful Names	Use Intention-Revealing Names	
	Pick one word per concept	
	Use Solution/Problem Domain Names	
Classes	Classes should be small!	
Functions	Functions should be small!	
	Do one Thing	
	Don’t Repeat Yourself (Avoid Duplication)	
Comments	Explain yourself in code	
Formatting	Make sure the code formatting is applied	
Exceptions	Use Exceptions rather than Return codes	
	Don’t return Null	

Table 2: Checklist for Security

Security		
Category	Checklist Item	File-Line
Fundamentals	Make class final if not being used for inheritance	
	Avoid duplication of code	
	Minimize the accessibility of classes and members	
Denial of Service	Input into a system should be checked for valid data size and range	
	Release resources (Streams, Connections, etc) in all cases	
Confidential Information	Purge sensitive information from exceptions (exposing file path, internals of the system, configuration)	
Input Validation	Validate inputs (for valid data, size, range, boundary conditions, etc)	

Table 3: Checklist for performance

Perfromance		
Category	Checklist Item	File-Line
Creating and Destroying Objects	Avoid creating unnecessary objects	
	Beware the performance of string concatenation	

Table 4: Checklist for Exception Handling

Exception Handling		
Category	Checklist Item	File-Line
Exceptions	Use checked exceptions for recoverable conditions and runtime exceptions for programming errors	
	Don’t ignore exceptions	

Table 5: Checklist for Static Code Analysis

Static Code Analysis		
Category	Checklist Item	File-Line
Static Code Analysis	Check static code analyzer report for the classes added/modified	

1.3 Assessment

1.3.1 Defect List

Following table gives a compete defect list.

Table 6: Defect list

Defect List		
Quality Attribute-Category	Checklist Item	File-Line
Clean Code-Meaningful Names	Use Intention-Revealing Names	log-16
Security-Fundamentals	Make class final if not being used for inheritance	log-5

1.3.2 Checkstyle Result

In this application, Checkstyle[3] is used for course code quality checking. Checktyle is a static code analysis tool used in software development for checking if Java source code complies with coding rules[3].Coding standards for checking is google coding standard according to Google Java Style Guide[4].The final result is *Checkstyle found no problems in the file(s).*

1.4 Report

A checklist-based infection source code reviewing is implemented. In overall, source code is proved to be relatively clean, secure, good-performed with good exception handling based on checklists. Although naming was not perfect, the output of checkstyle shows its good code style. Another small defect is that class would be more secure if is made final.

2 Testing

Unit Testing is implemented using Junit, and computer environment is Junit4 and Java 1.8. IDE is intellij.

2.1 Overall Test Results

All tests passed. And the calculation accuracy is all over 99.99 percents. 100 percent of the test is OK. There are no bugs to be reported.Following gives all OK tests which covers all the test cases.
In overall, test-ability is ensured as the test environment worked well. All test cases passed with result of OK and the accuracy was ensured.

2.2 Detailed Test Report

Following table 7 gives all the test cases with detailed infomation.

Table 7: Test Results - OK

Test Result-OK						
ID	Test Method	Test Date	Input	Expected Output	Actual Output	State (Pass/Fail)
1	LogGamma	30.Jul.2019	1	0	0	Pass
2		30.Jul.2019	1.53	-0.1192705602	-0.1192705602	Pass
3		30.Jul.2019	10.75	14.51947223	14.51947223	Pass
4		30.Jul.2019	100.65	362.1264291	362.1264291	Pass
5		30.Jul.2019	1.82E80	-0.06523734313	-0.06523734313	Pass
6	Gamma	30.Jul.2019	6	120	120	Pass
7		30.Jul.2019	4.000001	6.000007537	6.000007537	Pass
8		30.Jul.2019	-3.2	0.689056412	0.689056412	Pass
9		30.Jul.2019	-0.00001	-100,000.5772	-100,000.5772	Pass
10		30.Jul.2019	-42.232	-1.40581004E-51	-1.40581004E-51	Pass
11		30.Jul.2019	-242.232	-5.5611603E-474	-5.5611603E-474	Pass
12		30.Jul.2019	100.87	5.125761144E+157	5.125761144E+157	Pass
13		30.Jul.2019	-20.87	-2.306241516E-19	-2.306241516E-19	Pass
14	ln	30.Jul.2019	1	0	0	Pass
15		30.Jul.2019	2.5	0.91629073187	0.91629073187	Pass
16		30.Jul.2019	100.4	4.6091622073	4.6091622073	Pass
17		30.Jul.2019	200	5.2983173665	5.2983173665	Pass
18		30.Jul.2019	42812389.231312	17.572338087	17.572338087	Pass
19	exp	30.Jul.2019	4	54.5981500331	54.5981500331	Pass
20		30.Jul.2019	3.22655	25.1925924271	25.1925924271	Pass
21		30.Jul.2019	100.32	3.7018807e+43	3.7018807e+43	Pass
22		30.Jul.2019	1.554E82	4.73035380539	4.73035380539	Pass
23		30.Jul.2019	-3	0.04978706836	0.04978706836	Pass
24		30.Jul.2019	-10.54	0.00002645672	0.00002645672	Pass
25		30.Jul.2019	-1.42E44	0.65704681981	0.65704681981	Pass
26		30.Jul.2019	-1.762E-214	0.17170111795	0.17170111795	Pass
27	sine	30.Jul.2019	-6000	-0.4277195126	-0.4277195126	Pass
28		30.Jul.2019	-8900.054	-0.07790858957	-0.07790858957	Pass
29		30.Jul.2019	-10000.00001	-0.30561438888	-0.30561438888	Pass
30		30.Jul.2019	-1E6	0.34999350217	0.34999350217	Pass
31		30.Jul.2019	-1E8	-0.93163902711	-0.93163902711	Pass

References

[1] Java Code Review Checklist,Mahesh Chopker *https://dzone.com/articles/java-code-review-checklist?source=post*$_p$*age*

[2] Code review, Wikipedia $https://en.wikipedia.org/wiki/Code_review?utm_source = swifting.ioutm_medium = webutm_campaign = blog$

[3] checkstyle *https://checkstyle.sourceforge.io/*

[4] Google Java Style Guide *https://google.github.io/styleguide/javaguide.html*