PROBLEM 5 and 7. **Jingya Pan**
SOEN 6011 **40044079**
SOFTWARE ENGINEERING PROCESSES Due Date: 8/2/2019
Github address : https://github.com/panjingya/SOEN6011.git

1 Source code review

The source code is the implementation of function 6—exponential function. For this session, the Google's coding standards will be used as the criteria to perform the code reviewing. The aspects that included during the code review will be shown in the check list.

1.1 Check List

- Source File basics
- Source File Structure
- Formatting
- Naming
- Programming Practices
- Javadoc

1.2 Source File Basics

File Name	The names of the classes are 'Finalfunction', and 'Finalmain', which shows clearly the relationship between the two, but it is not case-sensitive, the purpose of the class does not be implied clearly.
File Encoding	Source files are encoded in UTF-8, which is compatible.

1.3 Source File Structure

Package statement	The package statement is not line-wrapped in the project and also obey the rule of column limt.
Class declaration	Each top-level class resides in a source file of its own, which is being done clearly in the project.

1.4 Formatting

Braces	Braces are used with if, else, and while statements, even when contains only a single statement, which made the code more readable and understandable.
Block indentation	Block indentation are consistent through the project.
One statement per line	One statement per line are guaranteed through the project.
Column limit	Column limit are reasonable through the project, no more than 100 characters.
Whitespace	Between consecutive classes a single blank line always appears to enhance the readaility; A horizontal whitespace appears after if, while, catch from an open parenthesis, which is consistent through the project.
Specific constructs	Some specific block are not being used properly. For example, in the 'if, else if, else' block. The end brace of 'if' preferably be put at the same line of the 'else if' statement; The end brace of 'else if' preferably be put at the same line of the 'else' statement.

1.5 Naming

Package names	Package name (sep) are all lowercase, which obey the rule.
Class names	Class name just use one capital letter, which can be improved to enhance readability; Preferably class names are written in UpperCamelCase.
Method names	Method name are all used lowercase letter in the project, which can be improved to enhance readability; Preferably method names are written in UpperCamelCase.
Constant names	Constant names are all used lowercase letter in the project, which can be improved to enhance readability; Preferably Constant names are written in uppercase letters.
Parameter names	Parameter names are all used one single letter to represent in the project, which made the meaning of the parameter not clear; Preferably Constant names can present the meaning, and are written in UpperCamelCase.

1.6 Programming Practices

Catch exceptions	When the program runs to catch, appropriate message will be shown in the console. For example, in the Finalfunction.java file line 41, System.out.println("Exception Raised = " + e);
Finalizers	did not override Object.finalize, which is a good practice.

1.7 Javadoc

Formatting	Detailed Javadoc blocks for each method are presented through the project.
Block tags	Block tags such as @author, @param, @return are properly used through the project.
The summary fragment	A detailed summary are provided in each javadoc block conveys the purpose of the function.
Indentation	The indentation for the javadoc need to be consistent through the project, which need to be improved.

2 Test case review

2.1 Abstract

The test case is based on function a^{b^x} (F7). According to the user manual, The application gives correct results for all the positive and negative numbers, for both powers and base. For the user input, three variables are needed to process the calculating. If the input are valid, the result will be displayed, otherwise the error messages will show up.

2.2 Testing Setup

Using the eclipse IDE to set up the project. After creating a new java project, I imported java files into the project just created. Junit 3 configuration is used to set up since in the project the import statement is like 'import junit.framework.TestCase'. Refreshing the project to eliminate all the warning, and start running the test cases.

2.3 Testing Result

There are 14 test cases altogether, followings are the results.

Input	Expected value	Obtained value	Status
a=3, b=2, x=2	81.0	81.0	Pass
a=2, b=0, x=0	2.0	2.0	Pass
a=2, b=5, x=0	2.0	2.0	Pass
a=0, b=5, x=9	0.0	0.0	Pass
a=0, b=0, x=0	0.0	0.0	Pass
a=0, b=0, x=2	1.0	1.0	Pass
a=5, b=0, x=6	1.0	1.0	Pass
a=0, b=0, x=0	0.0	0.0	Pass
a=0, b=1, x=0	0.0	0.0	Pass
a=2, b=0, x=0	2.0	2.0	Pass
a=4, b=0, x=0	4.0	4.0	Pass
a=2, b=-2, x=2	16.0	16.0	Pass
a=-2, b=4, x=2	65536.0	65536.0	Pass
a=-2, b=3, x=2	-512.0	-512.0	Pass

2.4 Problem

However the 'assertEquals' method are not using properly, even for now it does not influence the result. For the 'assertEquals' method the first parameter is expected value and the second parameter is the actual value, but now in the project, the two parameters are put in the wrong position.

3 Reference

Google Java Style Guide. (n.d.). Retrieved August 2, 2019, from https://google.github.io/styleguide/javaguide.html