Software Engineering Process **Amit Sachdeva**
Topic: **Deliverable 3 Code Review Question 5** **40084627**
Prof. P. Kamthan Due Date: 02/August/2019
My github Id: `https://github.com/amitsachdeva45/SEPPersonal/tree/master/Documentation/Deliverable3`

Code Review of Function 6 ab^x for Team I

Peer Details:

Name: Mahshad Saghaleini
Github Id: `https://github.com/MahshadS/Soen-6011`

1 Introduction and Objective

This project is about making a calculator for a function of ab^x. It should take a input from user and retrieve the result as accurate as possible. It must have exception as well as error handling keeping in mind the domain of the function. Users of this function are mostly mathematicians and even scientists, so it should also keep in mind the proper user interaction.
Objective of this review to check if the developer had used proper coding standards, proper quality tools. Addition to this, he/she had written every function from scratch without using any inbuilt function. The most important thing is to check the efficiency and maintainability of code.

2 Code review Approach

I have used manual style for code review without using any inbuilt tool. I have checked whether the peer is following proper coding standards, and is peer used any code quality tools to improve the coding style. Apart from this, I am checking whether she has used proper constraints and proper documentation to improve the readability of other developers. I have used a eclipse debugger for checking working of project.

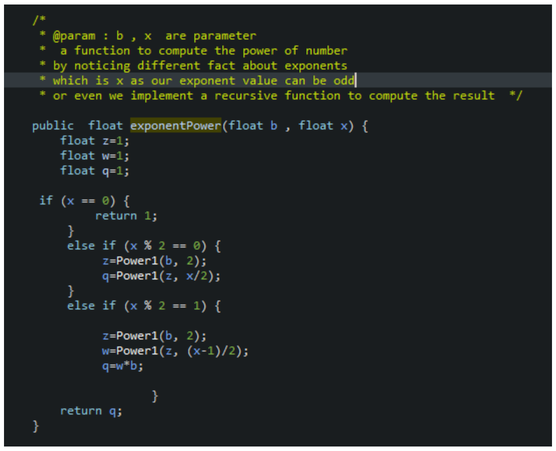
3 Coding Style

3.1 Functions/Methods

Coding Standards is properly used for writing Methods for example lowerCamelCase is used for declaring methods name with proper using access modifiers where needed as shown in figure 1. Along with that, naming of each method is properly clarifying its functionality and each method is handling only one task according to the defined coding standards.
Improvements: Two things can be improved in only main method by reducing the length of method as it is exceeding the defined number of lines of code i.e. approximately 25-40 lines per each method, and other thing is writing public methods before private methods.

3.2 Local Variables, Class Variables

All local variables are properly lowerCamelCase, and class variables are named properly defining its meaning with proper access modifiers i.e private and specifiers i.e. static, and final.
Improvements: All constant variables should be in **capital case** which can help in differentiating them for other class variables but few constant variables are written in lower case.



```
/*
 * @param : b , x  are parameter
 *  a function to compute the power of number
 * by noticing different fact about exponents
 * which is x as our exponent value can be odd
 * or even we implement a recursive function to compute the result  */

public  float exponentPower(float b , float x) {
    float z=1;
    float w=1;
    float q=1;

 if (x == 0) {
         return 1;
     }
     else if (x % 2 == 0) {
          z=Power1(b, 2);
          q=Power1(z, x/2);
     }
     else if (x % 2 == 1) {

          z=Power1(b, 2);
          w=Power1(z, (x-1)/2);
          q=w*b;

                 }
    return q;
}
```

Figure 1: Proper declaration of methods names and Java docs

3.3 Documentation, Class declaration

Java docs is used properly by giving proper description of every public method, explaining about every parameter passed in function and return type as shown in figure 1. Class is declared properly according to the coding standards.

3.4 Error Handling/Robust

Error handling is done by using try-catch block for example: For real number input, try catch is used to check so that user will only enter real number.

4 Code quality Tool

At some part of code, few useless spaces are present which can be removed by using coding quality tools like checkstyle. In addition to this, whole package is imported which can be avoided as we need only few functions from that package. Otherwise, everything is perfect.
Improvements: Checkstyle can be used to avoid this small things which indirectly increase burden on the code.

5 User Interface/Usability

Textual User interface (TUI) is used for working of the project, and user can select one of the option out of 5 options on command line for example addition, subtraction, multiplication, division and exponential. User can do any operation only once, after that it need to be restart.

6 Maintainability

To increase the maintainability, she has used separate methods for each functionality which enhance the ease to **maintain and understand** the code.

7 Final conclusion

All in all, she has consider all the factors which can be lead to go project structure by following proper norms, error handling, documentation which are key things to make a good project but few things can be done for example checkstyle can be used for improve code quality and remove useless code. Apart from this, Graphical user interface (GUI) and design pattern can be used to make the methods more independent and clear. So in the end, everything is perfect but few improvements can be plus point in the project.

8 References

- https://www.owasp.org/index.php/How_to_Write_an_Application_Code_Review_Finding
- https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/
- https://searchsoftwarequality.techtarget.com/definition/code-review