SOEN 6011: SOFTWARE ENGINEERING PROCESSES

Project Deliverable 2

GitHub Repository : `https://github.com/Hkohli30/SOEN_6011`
Himanshu Kohli
Student Id : 40070839
Applied Computer Science
Concordia University

Problem 4 and 6 **Himanshu Kohli**
SOEN 6011 **40070839**
Prof. Pankaj Kamthan Due Date: July 29, 2019

1 Debugger

1.1 Introduction

Debugging is a process of locating and removing bugs, errors and abnormalities form a programs. Bugs are subtle conditions which leads inefficient programs and can only be found when specific conditions occurs.

1.2 Debugger Used

The Eclipse IDE provides a bug finding tool which is combined in the views as Debug perspective which helps us to view execution of the program and the variables/constant at every stage of the program. Eclipse IDE Debugger is an source-level debugger or symbolic debugger which finds traps(conditions which doesn't allow the program to execute normally) in the system. Eclipse inbuilt debgger has many features which helps such as:[1] [2]

1. Single Stepping : A step by step execution of program helps the programmer understand, see and find the error by viewing the values of all the variables.
2. Breakpoints: Pausing the state of the program for examination
3. Tracepoints : Allows user to create conditional breakpoints to print messages without halting the breakpoints

1.3 Advantages and Disadvantages

1.3.1 Advantages

1. **Watches** in debugging eliminates the need to add additional print line statements to the source code and removing it after finding the bug.
2. It removes the tedious and time consuming process of rebuilding and executing the program while at most times **hot swapping** the changes.
3. Stepping in and out of some part of the code saves time for the user and increase programmer's efficiency to find bugs.

1.3.2 Disadvantages

1. For a smaller code which can be understood by print line statement it can be rather time consuming to use debugger
2. Overloads the system's resources in getting value of all variables at execution time.

2 Code Quality Checking Tool

2.1 Introduction

Code quality checking technically known as Static program analysis tool which is the analysis of the computer program without the actual execution of program. These software may vary and might target source code or the object code of the program. They have rules defined in their definition of the program and compare the source code with their definition and notify user of the modification which will make the program more feasible.[4]

2.2 Tool Used

PMD is an open source static source code analyzer that reports the issues found within the application. It compares the source with set of build in rules and report issues. Custom rules also can be defined by the user to handle specific issues or apply constrains. The issues reported are inefficient code or bad programming habits which reduces the maintainability and efficiency of the program. [5]

2.3 Advantages and Disadvantages

2.3.1 Advantages

1. Possibility of finding bugs using tool.
2. Removes redundancy in the application such as removing unwanted loops or code.
3. Sub optimal code : avoids the usage of strings and string buffer which takes memory but do not utilize it.
4. Dead Code : Reports the usage of code which is declared but not used in the program.
5. Reports over complicated errors such as replacing 'if' statement with loops.

2.3.2 Disadvantages

1. Sometimes report normal coding as error such as indentation errors or Java Docs needed errors
2. Does not support chain programming
3. Global variable declaration error.
4. Errors in normal access specifiers such as if the default access specifier is public and you specify public for a method, it will result in error.

2.4 Quality Attributes

The implemented program for function a^{b^x} aims to achieve all the following quality attributes and all the necessary steps taken to make the program more tenacious.

2.4.1 Correct

The program aims the results to be correct and handles larger and smaller values. The program implements log function and power function from scratch aiming to achieve actual calculator results. So two forms of power function were implemented one for decimal and one for integers for the exponent in order to achieve correct results. JUnit test cases for each and every value were tested and generated like-wise results for the values.

2.4.2 Efficient

The program implemented is efficient and only calculate values until it reaches the max value for double specifies it as positive or negative infinity, nullifying extra calculations whose results cannot be printed, saving computation time of the program. The bounding conditions are handled properly so that the function doesn't have to perform extra calculation and increase the efficiency in providing results. Try catch block maintains the state of the program while execution without having the program to be restarted making it efficient.

2.4.3 Maintainable

The program's implementation of Memento design pattern help the program memorize it's last result and make the program able to recall the result without any additional storage data structure. With using design patterns the program maintainability increases as everything divides into set of modules each pertaining to its own functionality which is easier to track, understand and modify. And the implementation of try catch helps the program.

2.4.4 Robust

The implementation of try catch block not only saves time but act as an shield to the erroneous inputs by the user such as a complex number or alphabetic values to the program. The program uses Taylor series to execute and converts b^x to $e^{x*\log b}$ where there are no values for $\log \leq 0$. And the program handles it without crashing making it secure and efficient.

2.4.5 Usability

The implementation of the Memento design pattern has divided the programming into modules which can be used for multiple other implementation of the program by just the main logic of the program. Even the logic part of the program is divided into modules which helps aim usability/re-usability of the program.

2.5 Traceability and Coverage

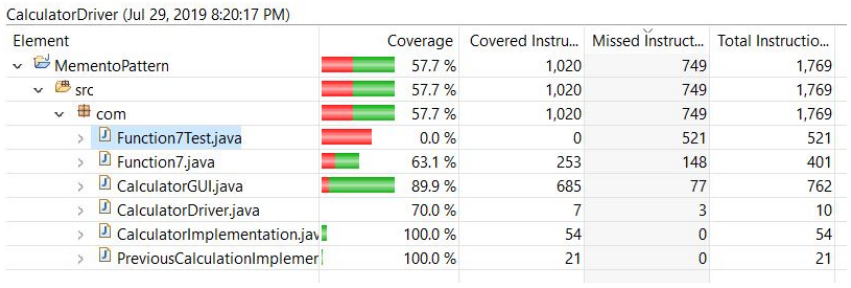
2.5.1 Test Case Traceability

Table 1: Test case traceability table.

Test Case	Test Case Name	Related Requirement(s)
1	testForPowerFunctionIntegerValues()	FRQ3
2	testForLogValue()	FRQ1, FRQ2
3	testForPowerFunctionLargeValues()	FRQ4, FRQ5
4	testForPowerFunctionDecimalValues()	FRQ3
5	testForEntireFunction()	FRQ8, FRQ7,FRQ6
6	signTestingForPowerFunction()	FRQ8
7	inputTestingForValidInputs()	FRQ1, FRQ3
8	inputTestingForInvalidInputs()	FRQ1, FRQ3

2.5.2 Code Coverage

Figure 1: Representation of the code coverage for random inputs

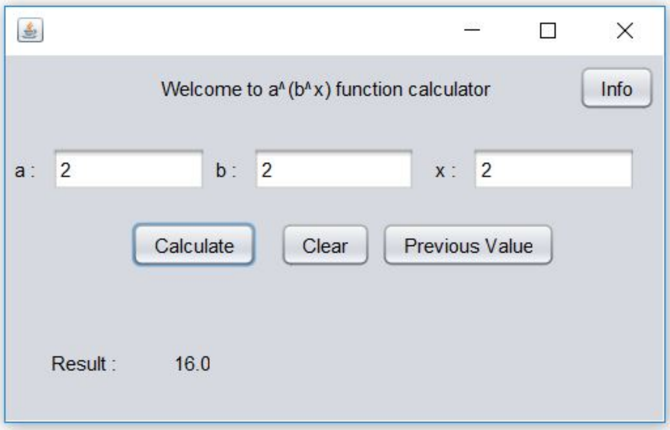


CalculatorDriver (Jul 29, 2019 8:20:17 PM)

Element	Coverage	Covered Instru...	Missed Instruct...	Total Instructio...
MementoPattern	57.7 %	1,020	749	1,769
src	57.7 %	1,020	749	1,769
com	57.7 %	1,020	749	1,769
Function7Test.java	0.0 %	0	521	521
Function7.java	63.1 %	253	148	401
CalculatorGUI.java	89.9 %	685	77	762
CalculatorDriver.java	70.0 %	7	3	10
CalculatorImplementation.jav	100.0 %	54	0	54
PreviousCalculationImplemer	100.0 %	21	0	21

2.5.3 Execution Screen Shot

Figure 2: Representation of the main screen



Welcome to a^(b^x) function calculator

a : 2 b : 2 x : 2

Result : 16.0

References

[1] Debugger https://en.wikipedia.org/wiki/Debugger

[2] Debugging the Eclipse IDE for Java Developers — The Eclipse Foundation https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php

[3] What are the advantages of using the Java debugger over println? https://softwareengineering.stackexchange.com/questions/168540/what-are-the-advantages-of-using-

[4] Static program analysis https://en.wikipedia.org/wiki/Static_program_analysis

[5] PMD (software) https://en.wikipedia.org/wiki/PMD_(software)

[6] List of system quality attributes https://en.wikipedia.org/wiki/List_of_system_quality_attributes