SOEN 6011- SOFTWARE ENGINEERING PROCESSES
PROJECT DELIVERABLE 2

Concordia University
Engineering and Computer Science

Jemish Kishor Paghadar (40080723)
Concordia University, Montreal

July 27 2019

Contents

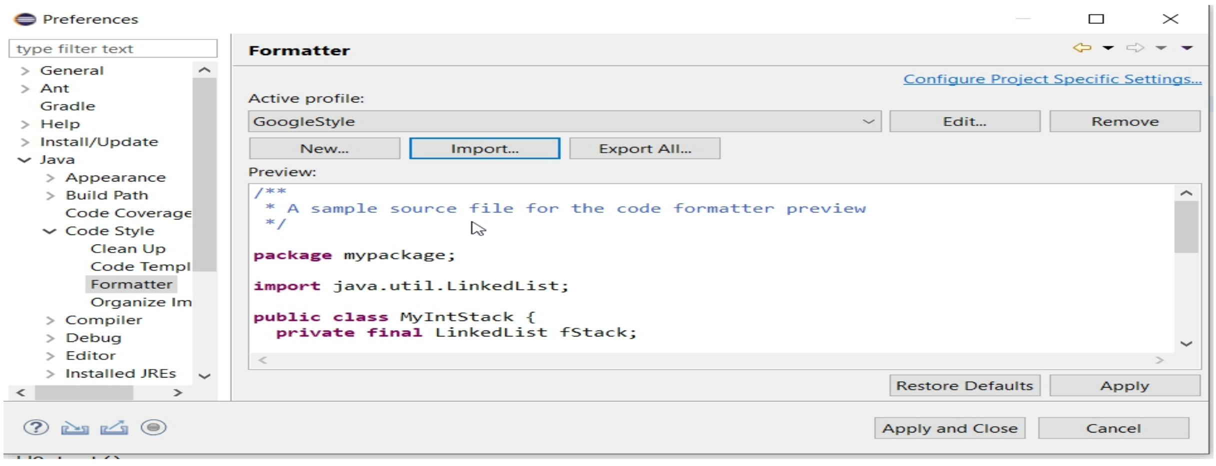
0.1 Problem 4

0.1.1 Function Implementation

The logarithm of a given number x is the exponent to which another fixed number, the base b, must be raised, to produce that number x as the logarithm is the inverse function to exponentiation. There are many different ways to implement logarithm functions such as using Maclaurin's Series, recursion method etc.
I have implemented logarithm function using Maclaurin's Series of natural logarithm without using any built-in functions which is mathematically represented as,
$\ln x = 2[(\frac{x-1}{x+1}) + \frac{1}{3}(\frac{x-1}{x+1})^3 + \frac{1}{5}(\frac{x-1}{x+1})^5 + ...]$

In general, $\ln x = 2\sum_{k=1}^{\infty} \frac{1}{2k-1}(\frac{x-1}{x+1})^{2k-1}$. So, I have implemented $log_b x$ as $\frac{\ln x}{\ln b}$.

We have followed 'standard' programming style for the implementation of source code which includes Javadoc API Documentation, proper indentation, naming and commenting conventions, consistence layout. We have used eclipse-java-Google style by importing it as a coding style to follow standard and improve readability and understandability. We have also used exception handling mechanism so that error message can be helpful.



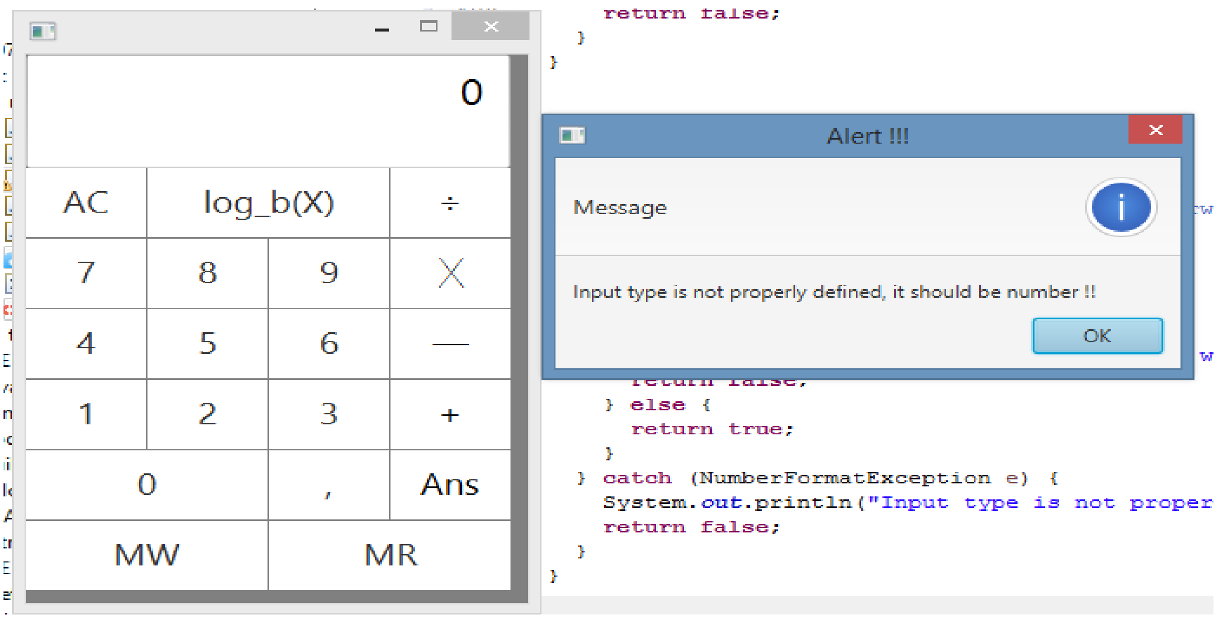
Active profile: GoogleStyle

```
/**
 * A sample source file for the code formatter preview
 */

package mypackage;

import java.util.LinkedList;

public class MyIntStack {
  private final LinkedList fStack;
```

Figure 1: Importing Google style to follow coding standard

I have used JavaFX for generating graphical user interface that is very easy to use and explore which is shown in below figure 2.
I have also used Memento design pattern which is used to restore state of an object to a previous state. Memento pattern uses three actor classes. Memento contains state of an object to be restored. Originator creates and stores states in Memento objects and

Caretaker object is responsible to restore object state from Memento. Using this pattern, I have implemented two functionality memory read that retrieves data from the memory and memory write stores the previous results in memory as shown in figure 3.



```
      return false;
   }
}
```

Alert !!!

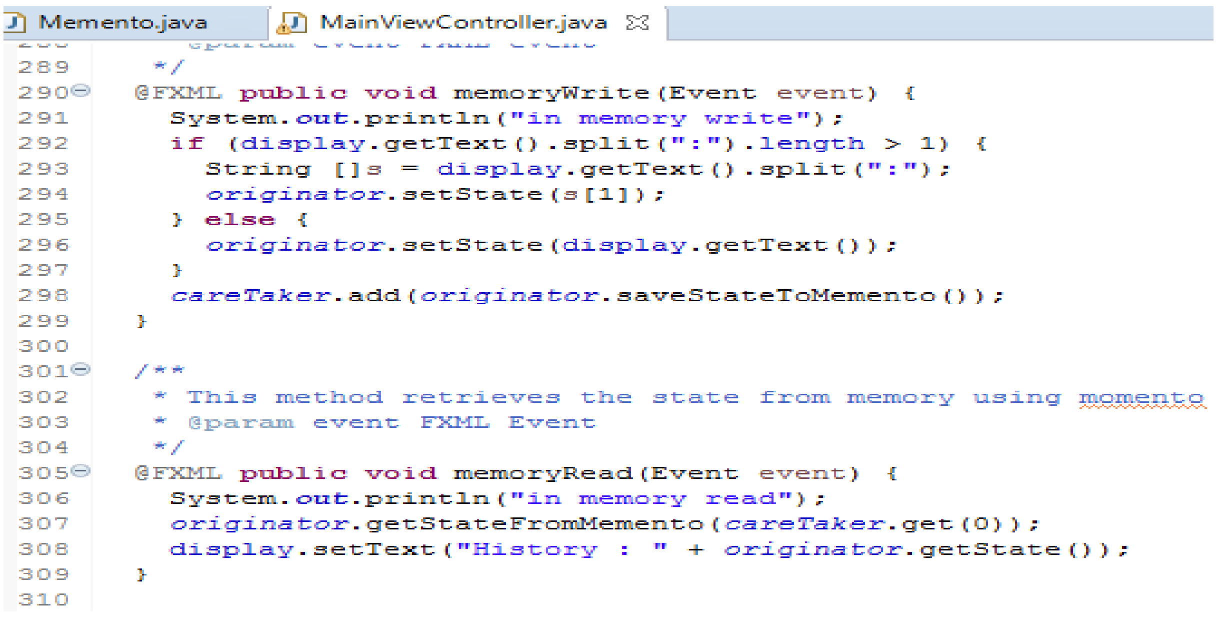
Message

Input type is not properly defined, it should be number !!

OK

```
         return false;
      } else {
         return true;
      }
   } catch (NumberFormatException e) {
   System.out.println("Input type is not proper
   return false;
   }
}
```

Figure 2: Graphical User interface of implementation



```
     * @param event FXML Event
     */
    @FXML public void memoryWrite(Event event) {
       System.out.println("in memory write");
       if (display.getText().split(":").length > 1) {
          String []s = display.getText().split(":");
          originator.setState(s[1]);
       } else {
          originator.setState(display.getText());
       }
       careTaker.add(originator.saveStateToMemento());
    }

    /**
     * This method retrieves the state from memory using momento
     * @param event FXML Event
     */
    @FXML public void memoryRead(Event event) {
       System.out.println("in memory read");
       originator.getStateFromMemento(careTaker.get(0));
       display.setText("History : " + originator.getState());
    }
```

Figure 3: Implementation and use of memento design pattern

0.1.2 Eclipse Debugger

- **Introduction :** Debugging is the routine process of locating and removing bugs, errors or abnormalities from programs. It helps to find subtle bug that are not visible during code reviews or that only happens when a specific condition occurs. Below are some quick tips and tools that will help you get started quickly with debugging Java project.
  Launching and Debugging a Java program in Eclipse: A Java program can be debugged simply by right clicking on the Java editor class file from Package explore or use the shortcut Alt + Shift + D, J instead.
  Breakpoints: A breakpoint is a signal that tells the debugger to temporarily suspend execution of your program at a certain point in the code. To define a breakpoint in your source code, right-click in the left margin in the Java editor and select Toggle Breakpoint. Alternatively, you can double-click on this position.
  Debug Perspective: The debug perspective offers additional views that can be used to troubleshoot an application like Breakpoints, Variables, Debug, Console etc. When a Java program is started in the debug mode, users are prompted to switch to the debug perspective.

- **Advantages**

- Helps you inspect your code during execution, making it a well-rounded tool.

- Allows you to use the same platform for both development and debugging.

- Shows relevant debugging information side-by-side, such as variables, breakpoints, threads, and call stacks.

- Allows to suspend and resume threads, step through the execution of the program, inspect values, and evaluate expressions.

- **Disadvantages**

- Eclipse does not provide Visual Debugger feature like NetBeans, that allows to debug the visual elements of Java and JavaFX GUI applications.

- Unlike Eclipse, IntelliJ provides inline debugger, that displays the value of variables within the code, right next the line where they were used.

0.1.3 Quality Attributes

- **Correct :** The correctness of a software program refers to the the ability of software program to perform their exact tasks, as defined by their specification. My program for the logarithm function is calculating the exact and accurate result of the function which helps to achieve accuracy which I have tested using unit testing framework. It is also free from error and in accordance with specifications.

- **Efficient :** The efficiency of program is measured in terms of time required to complete any task given to the system. My function calculates the result and responds in few seconds according to requirements. It also utilizes processor capacity, disk space and memory efficiently. As it takes less response time and efficient, it can be used in real time applications.

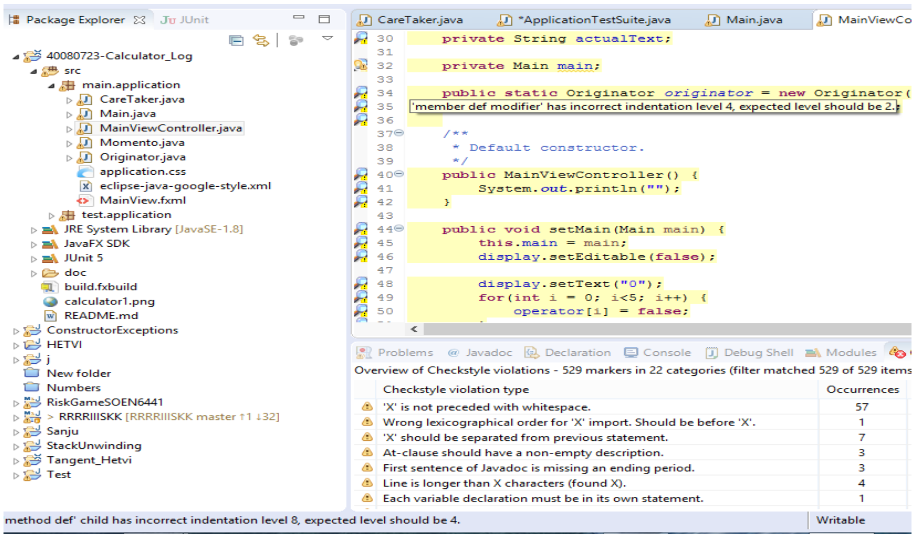
- **Maintainability :** It refers to the ease with which maintenance of a functional unit can be performed in accordance with prescribed requirements. As my function implements the different functionality independently with high cohesion and low coupling, it is very easy to cope with the changes and modify the program. It also supports reusability as any module can be reused in another program to save time and cost which helps to achieve the sustainability in further development.

- **Robust :** Robustness is the ability of a computer system to cope with errors during execution and cope with erroneous input. I have used exception handling mechanism and validation methods which handles unexpected inputs. This function implementation focuses on handling unexpected termination and unexpected actions. This program also handles these terminations and actions gracefully by displaying accurate and unambiguous error messages. These error messages allow the user to more easily debug the program.

- **Usable :** This can be measured in terms of ease of use. Application should be user friendly, easy to learn and navigation should be simple. This program provides consistent user interface standards or conventions. I have used graphical user interface which is easy for new or infrequent users to learn to use the system and I have used memento design pattern which allows to restore state of an object to a previous state.

0.1.4 Checkstyle

- **Introduction :** If your development team consists of more than one person, then obviously a common ground for coding standards (formatting rules, line lengths etc.) must be agreed upon - even if it is just for practical reasons to avoid superficial, format related merge conflicts. Checkstyle helps you define and easily apply those common rules. Checkstyle is an Open Source development tool to help you ensure that your Java code adheres to a set of coding standards. Checkstyle does this by inspecting your Java source code and pointing out items that deviate from a defined set of coding rules. Here is sample output of warnings generated by the plugin.



```
    private String actualText;

    private Main main;

    public static Originator originator = new Originator(
'member def modifier' has incorrect indentation level 4, expected level should be 2.

    /**
     * Default constructor.
     */
    public MainViewController() {
        System.out.println("");
    }

    public void setMain(Main main) {
        this.main = main;
        display.setEditable(false);

        display.setText("0");
        for(int i = 0; i<5; i++) {
            operator[i] = false;
```

Overview of Checkstyle violations - 529 markers in 22 categories (filter matched 529 of 529 items

Checkstyle violation type	Occurrences
'X' is not preceded with whitespace.	57
Wrong lexicographical order for 'X' import. Should be before 'X'.	1
'X' should be separated from previous statement.	7
At-clause should have a non-empty description.	3
First sentence of Javadoc is missing an ending period.	3
Line is longer than X characters (found X).	4
Each variable declaration must be in its own statement.	1

method def' child has incorrect indentation level 8, expected level should be 4.

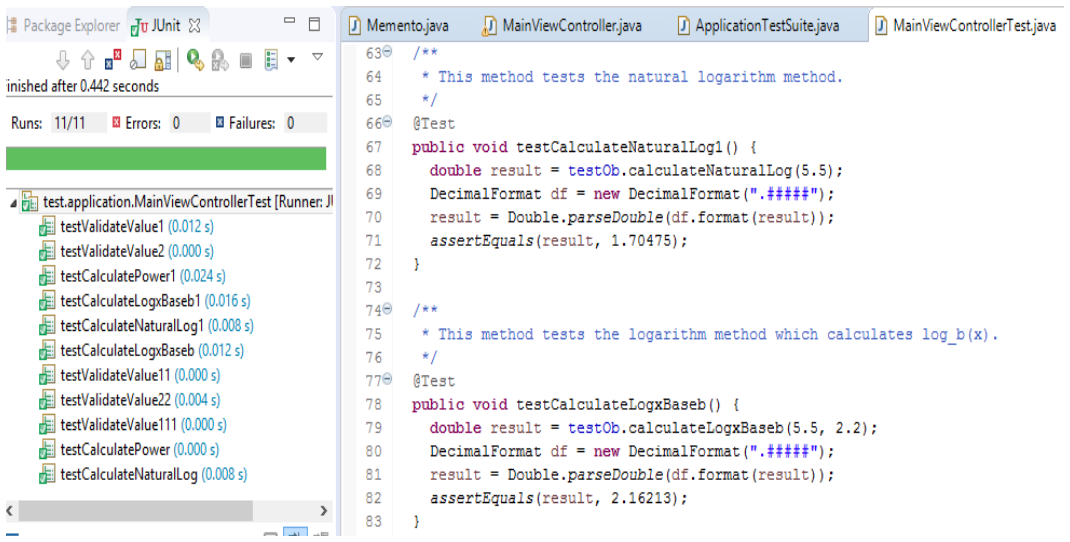
Figure 4: Warnings generated by Checkstyle Eclipse plugin

- **Advantages**
- By enforcing proper coding rules, helps to improve readability, re-usability, quality of the code and may reduce the cost of development.
- The set of rules used to check your code is highly configurable.
- Once defined a Checkstyle configuration can be used across multiple projects.
- **Disadvantages**
- Checkstyle is pretty much concerned with styles.It does not check for complicated rules like during the design of your classes, or for more special problems like implementing correctly the clone function.

0.2 Problem 6

0.2.1 Unit test

A unit test is a piece of code written by a developer that exercises a very small, specific area of functionality applied to one of the units of the code being tested. Usually, a unit test exercises some particular method in a particular context. The goal of unit testing is to isolate important parts (i.e. units) of the program and show that the individual parts are free of certain faults. I have used standard unit testing framework known as JUnit by which we can easily and incrementally build a test suite that will help us measure our progress, spot unintended side effects, and focus our development efforts. I have used best practices for writing unit test cases such as TDD(Test-driven development) approach and conventions. I have written test cases which are traceable to the requirement of the function. e.g, I have tested the validation method to ensure that input type and value is properly defined or not.



```
inished after 0.442 seconds
Runs: 11/11    Errors: 0    Failures: 0

test.application.MainViewControllerTest [Runner: J
    testValidateValue1 (0.012 s)
    testValidateValue2 (0.000 s)
    testCalculatePower1 (0.024 s)
    testCalculateLogxBaseb1 (0.016 s)
    testCalculateNaturalLog1 (0.008 s)
    testCalculateLogxBaseb (0.012 s)
    testValidateValue11 (0.000 s)
    testValidateValue22 (0.004 s)
    testValidateValue111 (0.000 s)
    testCalculatePower (0.000 s)
    testCalculateNaturalLog (0.008 s)
```

```java
/**
 * This method tests the natural logarithm method.
 */
@Test
public void testCalculateNaturalLog1() {
  double result = testOb.calculateNaturalLog(5.5);
  DecimalFormat df = new DecimalFormat(".#####");
  result = Double.parseDouble(df.format(result));
  assertEquals(result, 1.70475);
}

/**
 * This method tests the logarithm method which calculates log_b(x).
 */
@Test
public void testCalculateLogxBaseb() {
  double result = testOb.calculateLogxBaseb(5.5, 2.2);
  DecimalFormat df = new DecimalFormat(".#####");
  result = Double.parseDouble(df.format(result));
  assertEquals(result, 2.16213);
}
```

Figure 5: JUnit test case method implementation

0.2.2 Version Control System

To have a common repository for all project files available and updated remotely, distributed version control system(Git) is maintained.
Clone the repository from below link
https://github.com/Jemish27121997/SOEN6011_Function_Implementation.git

Bibliography

[1] https://www.eclipse.org/community/eclipse_newsletter/2017/june/article1.php

[2] https://raygun.com/blog/java-debugging-tools/

[3] https://checkstyle.org/eclipse-cs/#!/

[4] https://stackoverflow.com/questions/184563/checkstyle-vs-pmd

[5] https://github.com/Jemish27121997/SOEN6011_Function_Implementation