Concordia University

SOEN 6011 - Software Engineering Process

ETERNITY: FUNCTION ab^x

Deliverable 1

Avneet Kaur Pannu
Student ID : 40168576
https://github.com/avneet-kaur/SOEN-6011

Contents

1 Introduction

An exponential function is a function with the general form ab^x, $a \neq 0$, b is a positive real number and $b \neq 1$ .In an exponential function, a is constant, the base b is a constant, and the exponent x is a real variable.[1]

1.1 Domain

- The domain is all real numbers.
  $-\infty < x < +\infty$, $x \in R$ [1]

1.2 Co-Domain

- The co-domain is also set of all real numbers.

1.3 Characteristic

- **Exponential growth** : In the function f(x) = b^x when $b > 1$, the function represents exponential growth. In figure 1, it is evident on the left side. [3]
- **Exponential decay** : In the function f(x) = b^x when $0 < b < 1$, the function represents exponential decay. In figure 1, it is evident on the right side.[3]
- **Commutativity**: Exponential function is not commutative which means $x^y \neq y^x$ for $x \neq y$. For example, $0^1 = 0$ and $1^0 = 1$.
- **Natural Exponential Function**: When the base is chosen to be b=e, the function f(x) = e^x is called natural exponential function.[1]
- In the function f(x) = ab^x when $|a| > 1$, it increases the speed of either growth or decay, and $0 < |a| < 1$ decreases the speed of either growth or decay.[2]

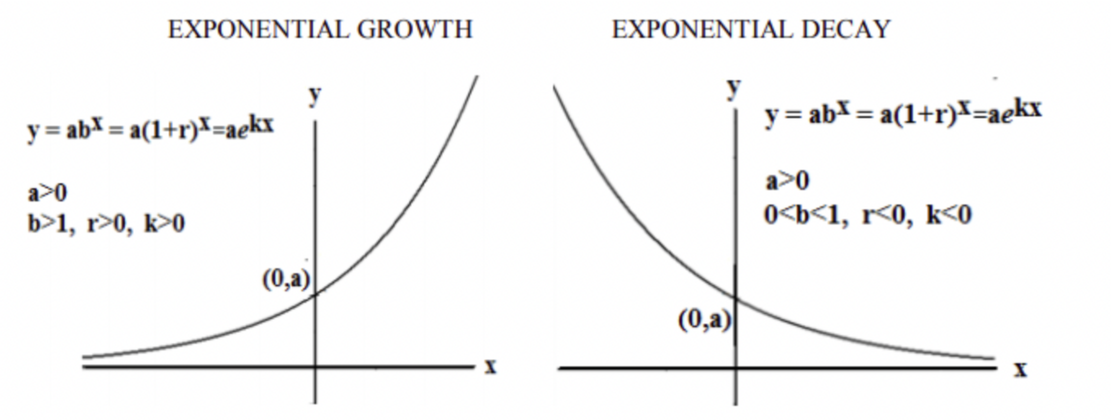


Figure 1: Exponential Growth and Exponential Decay

2 Functional Requirement

2.1 Definitions and abbreviations

Term	Definition
FR	Functional Requirement
NFR	Non-Functional Requirement
User	End user are the human users who interacts with the system
System	Application which is used for solving exponential function.

Table 1: Definitions and abbreviations

2.2 Assumptions

- The calculator must accepts the exponential constant like e in addition to the constants a and b.
-

2.3 Requirements

Functional Requirements

- **ID** :FR1
  Type :Functional
  Version Number :1.0
  Owner : Avneet

Priority : High
Difficulty : Easy
Description :The calculator should ask the user to input a, b, and x.
Rationale :In order to process function f(x) = ab^x and give output system needs input form the user.

- **ID** :FR2
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :When a user input is not a number, the system should provide an error message.
  Rationale :The only acceptable input for an exponential function calculation is a number.

- **ID** :FR3
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :If a user enters incorrect data, the system shouldn't shut down but rather prompt users to reenter their data.
  Rationale :The ability to perform calculations again and without closing the programme should be available to the user.

- **ID** :FR4
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : Whole numbers and rational numbers are accepted as user inputs.
  Rationale : The code does not handle irrational numbers.

For instance, π, $\sqrt{2}$

- **ID** :FR5
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : Fractional inputs must be entered as double values.
  Rationale : If a user wants to provide a base or exponent value of 1/2, they must do so as 0.5.

- **ID** :FR6
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : Base b is restricted to positive number.
  Rationale : In order to guarantee b^x is real number.

- **ID** :FR7
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : When any base value of b is raised to the power of x=0, the function's b^x portion must return the value 1.
  Rationale : For instance: 11 raised to the power 0 gives 1.

- **ID** :FR8
  Type :Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy

Description : When base value b=0 is raised to any exponent value, the b^x portion of the function must return 0.
Rationale : For instance, 0 raised to the power 11 yields 0.

Non-Functional Requirements

- **ID** :NFR1
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :An error message should be informative and relevant to the user.
  Rationale : The user should be able to resolve simple problems on their own by understanding error message to enhance usability.

- **ID** :NFR2
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :The command line interface ought to be user-friendly.
  Rationale : The system should be simple for the user to operate.

- **ID** :NFR3
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :The outcome must be accurate.
  Rationale : To enhance the accuracy of system. It is inappropriate to display incorrect ouptut to the user.

- **ID** :NFR4
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description :There should be no more than 5 seconds of calculation time.
  Rationale : In order to improve the performance of the system.

- **ID** :NFR5
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : System should be maintainable for the duration of its anticipated lifetime and able to accommodate new requirements in response to stakeholders' changing needs.
  Rationale : Since future changes to software systems are inevitable. Maintainable systems are therefore simpler to alter.

- **ID** :NFR6
  Type :Non-Functional
  Version Number :1.0
  Owner : Avneet
  Priority : High
  Difficulty : Easy
  Description : The system should be developed using widely used and standardised language.
  Rationale : Java is used to build a system which is platform independent hence make the system portable.

3 Algorithm

3.1 Pseudocode

Algorithm 1 Iterative Algorithm to calculate: ab^x

```
procedure calculateExponentialFunction(a, b, x)
    input:  String a, b, x
    output:  double res
    res = 1
    temp = 1
    if ((a || b) == "0") then return res
    else
        if (b == "e") then
            res = exponentialFunc(x,n)
            end
            return a * res
        else
            for temp <=x
            res = res * b
            temp = temp + 1
            end
            return a*res

procedure exponentialFunc(x, n)
    input:  int x, int n
    output:  double res
    power = 1
    factorial = 1
    recursive
    if (n==0) then
        return 1
    recursive = exponentialFunc(x, n-1)
    power = power * x
    factorial = factorial * n
    return (recursive + power/factorial)
```

Algorithm 2 Recursive Algorithm to calculate: ab^x

```
procedure calculateExponentialFunction(a, b, x)
    input:  string a, b, x
    output:  double res
    res=0
    if (a || b == 0) then
        return res
    else if (b=="e") then
        res = naturalExponential(x)
    else
        res = calculatePower(b,x)
    res = $a * res$
    return res

procedure naturalExponential(x)
    input:  int x
    output:  double exposum
    nterms = 25
    exposum = 1
    for i <= nterms
    exposum = 1+x * exposum/i
    end
    return exposum

procedure calculatePower(b, x)
    input:  double b, int x
    output:  double res
    if (($x < 0$) then
        return $1.0/powHelper(b, x)$
    return $powHelper(b, x)$

procedure powHelper(b, x)
    input:  double b, int x
    output:  double res
    if ($x == 0$) then return 1
    if ($x == 1$) then return b
    if ($xmod2 == 0$) then
        return $powerHandler(b * b, n/2)$
    else
        return $b * powerHandler(b * b, n/2)$
```

3.2 Description

Algorithm1

Description:

- **Case 1:** When base has been assigned with any number, then for loop will keep multiplying result variable by base variable until the power becomes zero. And then result is multiplied with constant a.
- **Case 2:** When base has been assigned with natural exponential constant "e". Then the function is computed using Taylor series. Since Taylor Series is a combination of multiple values like sum, power and factorial term, hence we will use static variables.

Rationale: Due to its $O(n^2)$ time complexity and linear space constraints in calculating exponential function and O(n) for power calculations, this algorithm was discarded.So, in the following algorithm 2 I tried to make it efficient.
Complexity: $O(n)$, $O(n^2)$
Advantages:

1. In iterative solution, there is no stack overflow exception where stack can take no more frames.
2. Iterative algorithms are easy to understand and readable by human.

Disadvantages:

1. For case 1, this algorithm is taking O(n) time complexity.
2. For case 2, this algorithm is taking $O(n^2)$ time complexity and O(n) Space complexity.

Algorithm2

Description:

- **Case 1:**When base is assigned any numeric value, calculatePower() function is called which in further calls powerhandler() function to consider cases where power is negative and positive.
- **Case 2:**When base has been assigned with natural exponential constant "e", we calculated the sum using for loop, and calculated it for n terms using Taylor Series.

Rationale: In Algorithm 1 case 1 is optimized to $O(logn)$ and case 2 is optimized to O(n). So the decision of implementing Algorithm 2 is made since its more time and space efficient.
Complexity: $O(logn)$, O(n)
Advantages:

1. Recursion add clarity to the code and reduce time while debugging code.
2. In this we tried to avoid stack overflow exception by handling all possible edge cases efficiently.

Disadvantages:

1. Recursive methods will often throw a StackOverflowException when processing big numeric values.

3.3 Mindmap for Pseudocode format Selection

4 Debugger, Quality Attributes, Checkstyle

4.1 Debugger

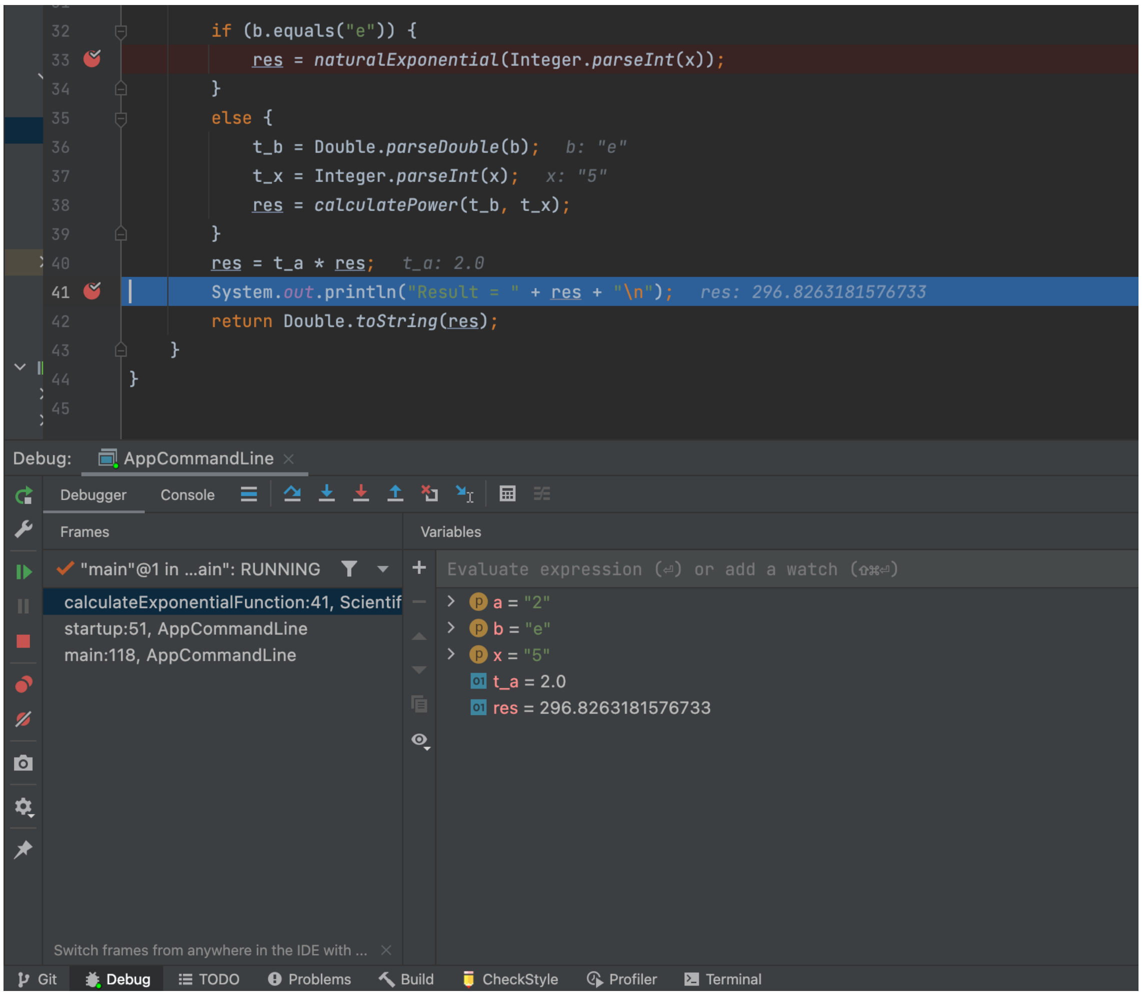
Description

A debugger is a tool that can be used to examine what is happening in a programme. This allows you to find bugs by carefully analysing how the programme is executed. One can examine the behaviour of code using a debugger without changing the source code.

Debugger Used: Used inbuilt debugger provided by IntelliJ IDEA.

Advantages

1. **Saves Time:** Debugger helps developer in identifying erroneous state of program. For example, if ode is breaking somewhere, then debugger comes to rescue to identify the root cause of error and hence saves time of developer.
2. **Bug free end product:** Debugger allows early fault detection and hence give enough time to developers to fix the error and release bug free software to customers.
3. **Watchpoints:** To edit code without restarting your program, Watchpoints comes to rescue and helps in inspecting data items or other objects.



```
        if (b.equals("e")) {
            res = naturalExponential(Integer.parseInt(x));
        }
        else {
            t_b = Double.parseDouble(b);   b: "e"
            t_x = Integer.parseInt(x);   x: "5"
            res = calculatePower(t_b, t_x);
        }
        res = t_a * res;   t_a: 2.0
        System.out.println("Result = " + res + "\n");   res: 296.8263181576733
        return Double.toString(res);
    }
}
```

```
"main"@1 in ...ain": RUNNING
calculateExponentialFunction:41, Scientif
startup:51, AppCommandLine
main:118, AppCommandLine

a = "2"
b = "e"
x = "5"
t_a = 2.0
res = 296.8263181576733
```

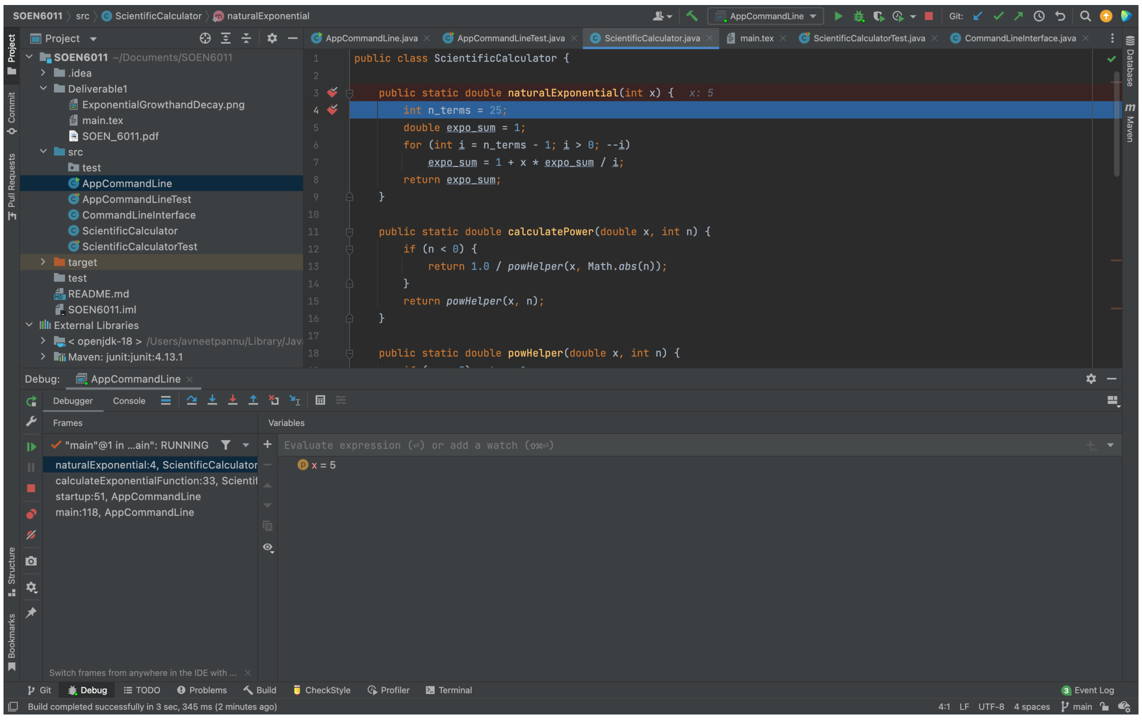
Figure 2: Snapshot representing usage of Line Breakpoint while debugging

Disadvantages

1. The downside of Debugger is that developers cannot debug microservices easily and serverless application in local environment.

2. Debugging remotely is also difficult since its dependent on permission, which require access and admin privileges to the server.

Snapshots



```java
public class ScientificCalculator {

    public static double naturalExponential(int x) {   x: 5
        int n_terms = 25;
        double expo_sum = 1;
        for (int i = n_terms - 1; i > 0; --i)
            expo_sum = 1 + x * expo_sum / i;
        return expo_sum;
    }

    public static double calculatePower(double x, int n) {
        if (n < 0) {
            return 1.0 / powHelper(x, Math.abs(n));
        }
        return powHelper(x, n);
    }

    public static double powHelper(double x, int n) {
```

```
"main"@1 in ...ain": RUNNING
naturalExponential:4, ScientificCalculator
calculateExponentialFunction:33, Scientif
startup:51, AppCommandLine
main:118, AppCommandLine
```

```
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
x = 5
```

Switch frames from anywhere in the IDE with ...

Build completed successfully in 3 sec, 345 ms (2 minutes ago)

Figure 3: Snapshot representing usage of Method Breakpoint

4.2 Quality Attributes

Robustness

1. Each time new functionality was released, a corresponding Unit case was developed and tested.
2. Included all potential edge cases for the calculating function.
3. Handled all the potential exceptions properly.

Maintainability

1. Each method that has been defined has been properly commented.
2. Global variables should not be used to ensure future maintenance.
3. Refactoring code in accordance with coding standards after each commit to a github branch.

Correctness

1. Adhering to java coding standards.
2. To check the correctness of all the functionality, JUnit test cases are implemented.
3. Displaying correct result while on successful calculation..

Usability

1. A command-line interface that is easy to use.
2. Showing a menu of the functions a calculator can perform.
3. Providing customers with a helpful error message that tries to assist them in identifying minor mistakes

4.3 Checkstyle

Description

To help programmers to adhere Java coding standard and to check the pragmatic quality of Java Code, development tool known as Checkstyle is used. It helps in checking Java code automatically. Also, developers can define there own set of rules and check code against their rules.
Checkstyle Used: Used Sun Code Conventions checkstyle provided by Checkstyle-IDEA.

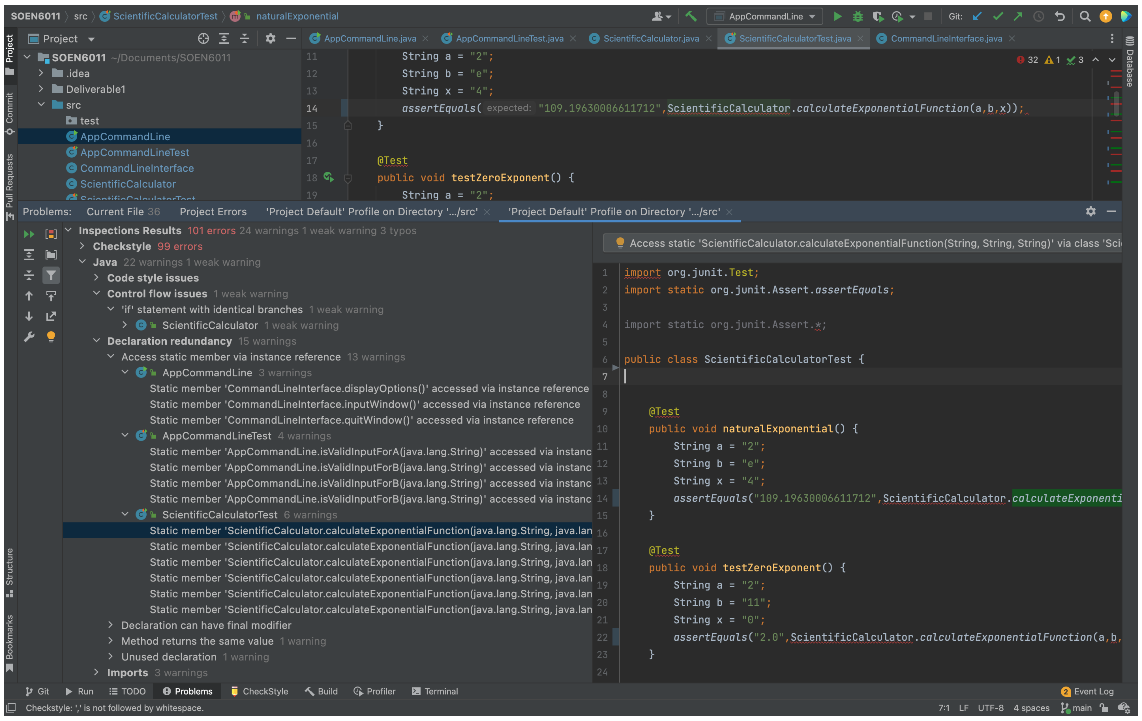
Advantages

1. Helps in resolving Class and method design problems.
2. Helps in resolving formatting issues.

Disadvantages

1. Configuration is time consuming, it needs external plugins to download.
2. To configure checkstyle, some people prefer making changes to maven configuration which is also error-prone and takes time to resolve.

Snapshots


SOEN6011 〉 src 〉 ScientificCalculatorTest 〉 naturalExponential
AppCommandLine
Git:
Project
SOEN6011 ~/Documents/SOEN6011
.idea
Deliverable1
src
test
AppCommandLine
AppCommandLineTest
CommandLineInterface
ScientificCalculator
ScientificCalculatorTest
AppCommandLine.java
AppCommandLineTest.java
ScientificCalculator.java
ScientificCalculatorTest.java
CommandLineInterface.java
32 1 3
Database
Problems: Current File 36 Project Errors 'Project Default' Profile on Directory '.../src' 'Project Default' Profile on Directory '.../src'
Git Run TODO Problems CheckStyle Build Profiler Terminal
2 Event Log
7:1 LF UTF-8 4 spaces main
Project Commit Pull Requests Structure Bookmarks

```
        String a = "2";
        String b = "e";
        String x = "4";
        assertEquals( expected: "109.19630006611712",ScientificCalculator.calculateExponentialFunction(a,b,x));
    }

    @Test
    public void testZeroExponent() {
        String a = "2";
```

```
Inspections Results 101 errors 24 warnings 1 weak warning 3 typos
  Checkstyle 99 errors
  Java 22 warnings 1 weak warning
    Code style issues
    Control flow issues 1 weak warning
      'if' statement with identical branches 1 weak warning
        ScientificCalculator 1 weak warning
    Declaration redundancy 15 warnings
      Access static member via instance reference 13 warnings
        AppCommandLine 3 warnings
          Static member 'CommandLineInterface.displayOptions()' accessed via instance reference
          Static member 'CommandLineInterface.inputWindow()' accessed via instance reference
          Static member 'CommandLineInterface.quitWindow()' accessed via instance reference
        AppCommandLineTest 4 warnings
          Static member 'AppCommandLine.isValidInputForA(java.lang.String)' accessed via instanc
          Static member 'AppCommandLine.isValidInputForB(java.lang.String)' accessed via instanc
          Static member 'AppCommandLine.isValidInputForB(java.lang.String)' accessed via instanc
          Static member 'AppCommandLine.isValidInputForB(java.lang.String)' accessed via instanc
        ScientificCalculatorTest 6 warnings
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
          Static member 'ScientificCalculator.calculateExponentialFunction(java.lang.String, java.lan
      Declaration can have final modifier
      Method returns the same value 1 warning
      Unused declaration 1 warning
  Imports 3 warnings
```

Access static 'ScientificCalculator.calculateExponentialFunction(String, String, String)' via class 'Sci

```
import org.junit.Test;
import static org.junit.Assert.assertEquals;

import static org.junit.Assert.*;

public class ScientificCalculatorTest {


    @Test
    public void naturalExponential() {
        String a = "2";
        String b = "e";
        String x = "4";
        assertEquals("109.19630006611712",ScientificCalculator.calculateExponenti
    }

    @Test
    public void testZeroExponent() {
        String a = "2";
        String b = "11";
        String x = "0";
        assertEquals("2.0",ScientificCalculator.calculateExponentialFunction(a,b,
    }

```

Checkstyle: ',' is not followed by whitespace.

Figure 4: Snapshot representing the usage of checkstyle

5 Unit Tests

5.1 Standard Guidelines

5.2 Traceability

Bibliography

[1] Zill, D. G., & Wright, W. S. (2011). Calculus: Early transcendentals. Jones and Bartlett Publishers.

[2] Rock, N. M. (2007). Standards driven math. student standards handbook. Team Rock Press.

[3] Exponential growth and decay - virtuallearningacademy.net. (n.d.). Retrieved August 3, 2022, from `https://virtuallearningacademy.net/VLA/LessonDisplay/Lesson6156/MATHALGIIBU17Exponential_Decay.pdf`

[4] GeeksforGeeks. (2022, July 19). Write a program to calculate POW(X,N). GeeksforGeeks. Retrieved August 5, 2022, from https://www.geeksforgeeks.org/write-a-c-program-to-calculate-powxn/

[5] Exponential Function Calculator. High accuracy calculation for life or science. (n.d.). Retrieved August 5, 2022, from https://keisan.casio.com/exec/system/1223447896