Scientific Calculator: Power Function x^y

Pavit Srivatsan, 40155323
Concordia University, SOEN 6011 - Software Engineering Processes

Introduction

A power function is of the form: f(x,y) = x^y ,where x and y are real numbers.

Features

1. When y is a non-negative integer, the domain is all real numbers: (- ∞,∞).
2. When y is a negative integer, the domain is all real numbers excluding zero ((- ∞, 0) $\cup$ (0,∞)).
3. When y is an irrational number and y $>$ 0, the domain is all non-negative real numbers and when y is an irrational number and y $<$ 0, the domain is all positive real numbers.
4. The codomain of the function is [- ∞,∞] and can be indeterminate.

Functional Requirements

1. System should prompt the user to enter the value of x and y, i.e. the base and the exponent.
2. System should display an error message when the double value entered by the user is not a number.
3. If the input entered is not valid, the system should prompt the user to input values again.
4. User should have the option to exit the program anytime during the use.

Non-Functional Requirements

1. The error message displayed should be appropriate and helpful for the user.
2. The result displayed should be as accurate as possible.
3. Calculation time should be less than 1 second.

Technical Specification

Risk Identified during Analysis Design

- The use of double types can pose a risk because of the possible loss of precision by using such data type.
- Testing must not only be limited to the main function but also its sub-functions

Development Environment

- **IDE:** Eclipse, **Development Language:** Java, **Testing Framework:** JUnit4
- **Debugger:** Eclipse Built-in Debugger, **Source Code Analysis:** SonarLint, **CheckStyle:** Google CheckStyle

Critical Decisions

- **Pseudocode Format:** Availability of multiple pseudocode formats made it hard to finalize a format to be consistent across the team.
- **User Interface:** The time constraint was a critical factor in deciding to choose a textual interface over a graphical one.
- **Code Style:** Availability of tools in Eclipse and IntelliJ marketplace for Google Checkstyle enabled us to choose Google Checkstyle.
- **Source Code Review:** SonarLint was easy to use and apply for Source Code Review. It was able to find some major technical issues with code such as Code Smell.

Implementation

```
Microsoft Windows [Version 10.0.19042.1165]
(c) Microsoft Corporation. All rights reserved.

D:\eclipse-workspace\SOEN6011_F7_Implementation\src>java Power.java
Enter the Base value:
4
Enter the Exponent value:
3
64.0

D:\eclipse-workspace\SOEN6011_F7_Implementation\src>java Power.java
Enter the Base value:
5
Enter the Exponent value:
-2
0.03845745325088501

D:\eclipse-workspace\SOEN6011_F7_Implementation\src>java Power.java
Enter the Base value:

Enter the Exponent value:

Exception : Entered input is not a double value

D:\eclipse-workspace\SOEN6011_F7_Implementation\src>
```

Lessons Learnt

Mathematical Terms:
Basic mathematical terms such as integer, real numbers, whole numbers, function, etc. were re-learnt.

Latex for Documentation:
Extensive documentation with the mandate to use Latex enabled to learn Latex and made it easier to use.

JUnit Testing:
Testing double values using JUnit had some depreciated methods. The current method involves adding a precision value.

Source code formatting:
Indentation of code was learnt with Google Checkstyle, which enabled the creation of a standard code.

References

https://www.overleaf.com/latex/templates/the-beamer-poster-version-of-the-radboud-university-mfddtqfwpspm

https://github.com/san089/SOEN-6011/blob/master/Poster/document.pdf