CSC 413 Project Documentation

Summer 2023

Juan Estrada

923058731

CSC-0413-01-Summer-2023-R4

https://github.com/csc413-SFSU-Souza/csc413-p1-jjestrada2

Table of Contents

1 Introduction

1.1 Project Overview

The project involves completing the development of a calculator software. The calculator is designed to evaluate simple mathematical expressions that include common operators such as addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^). It also recognizes the priority of mathematical expressions, ensuring that calculations are performed correctly. The software provides a user-friendly interface, allowing users to input their expressions and obtain accurate results.

1.2 Technical Overview

The technical overview of this project involves completing a software application using the principles of object-oriented design. The software follows a hierarchical structure, utilizing various classes to implement different functionalities and ensure code organization and reusability. The project aims to adhere to best practices and design patterns within the object-oriented programming paradigm.

Additionally, the project includes developing a graphical user interface (GUI) for the software. The GUI is designed with a user-friendly interface, considering aspects such as layout, aesthetics, and ease of interaction. Attention is given to proper user experience design principles, ensuring that the software is intuitive and easy to navigate for users.

1.3 Summary of Work Completed

To get the assignment working correctly, I made the following contributions:

1. Completed the abstract class "Operator" by creating a HashMap to store the operators' string values as keys and their respective initialization class as their value. This allowed for efficient retrieval and identification of operators.
2. Implemented the static methods of the abstract Operator class, such as "getOperator" and "check." These methods facilitated the validation and retrieval of operators based on the given expression.
3. Created subclasses of the abstract class "Operator" and implemented the corresponding abstract methods, such as "priority" and "execution." This allowed for defining the specific behavior and priority of each operator.
4. Completed the Operand class by adding its corresponding attribute (value), constructor, and methods like "getValue" and "check." This class provided the necessary functionality to represent and manipulate operands in the expression.
5. Implemented the user interface by completing the actionPerformed method. I added multiple conditional statements to handle different button clicks and provide the required functionality. For example, I enabled the erasure of text when the user pressed "C" or "CE," evaluated the expression when "=" was pressed, and concatenated characters by default when the button represented an operand or operator.
6. Addressed the evaluator class, which presented the most challenging part of the assignment. I fixed the initialization of the newOperator object and implemented an "if-else" conditional to handle parentheses operators and operands. I ensured the proper order of pushing them onto their respective stacks. Additionally, I created a while loop to process the operatorStack and operandStack, performing the necessary mathematical calculations to obtain the final result of the expression.

2 Development Environment

a. Java 20.0.1

b. intellij

3 How to Build/Import your Project

1. In your terminal type: git clone https://github.com/csc413-SFSU-Souza/csc413-p1-jjestrada2 Please clone your repo to a folder on your compute that does not require elevated privileges.
2. Open your preferred IDE (Integrated Development Environment) such as Eclipse, IntelliJ IDEA, or NetBeans.
3. Select import project.
4. Select the Calculator folder as the source root of the project
5. Keep the “Create project from existing resources” radio button selected
6. All default fields can be left alone here.
7. Now, you can build the project by selecting the "Build" or "Compile" option from the IDE's menu. This will compile the Java source code files into bytecode and generate the necessary class files.

4 How to Run your Project

After the project is successfully built, you can run the calculator application by locating the main class (in this case, EvaluatorUI.java) and right-clicking on it. Choose the "Run" or "Debug" option from the context menu. The calculator application should now launch and display the graphical user interface (GUI), allowing you to input mathematical expressions and evaluate them

5 Assumption Made

Initially, I assumed that all the code provided by the professor was accurate and couldn't be modified. However, as I delved deeper into the project, I came to the realization that my assumption was only partially correct.

6 Implementation Discussion

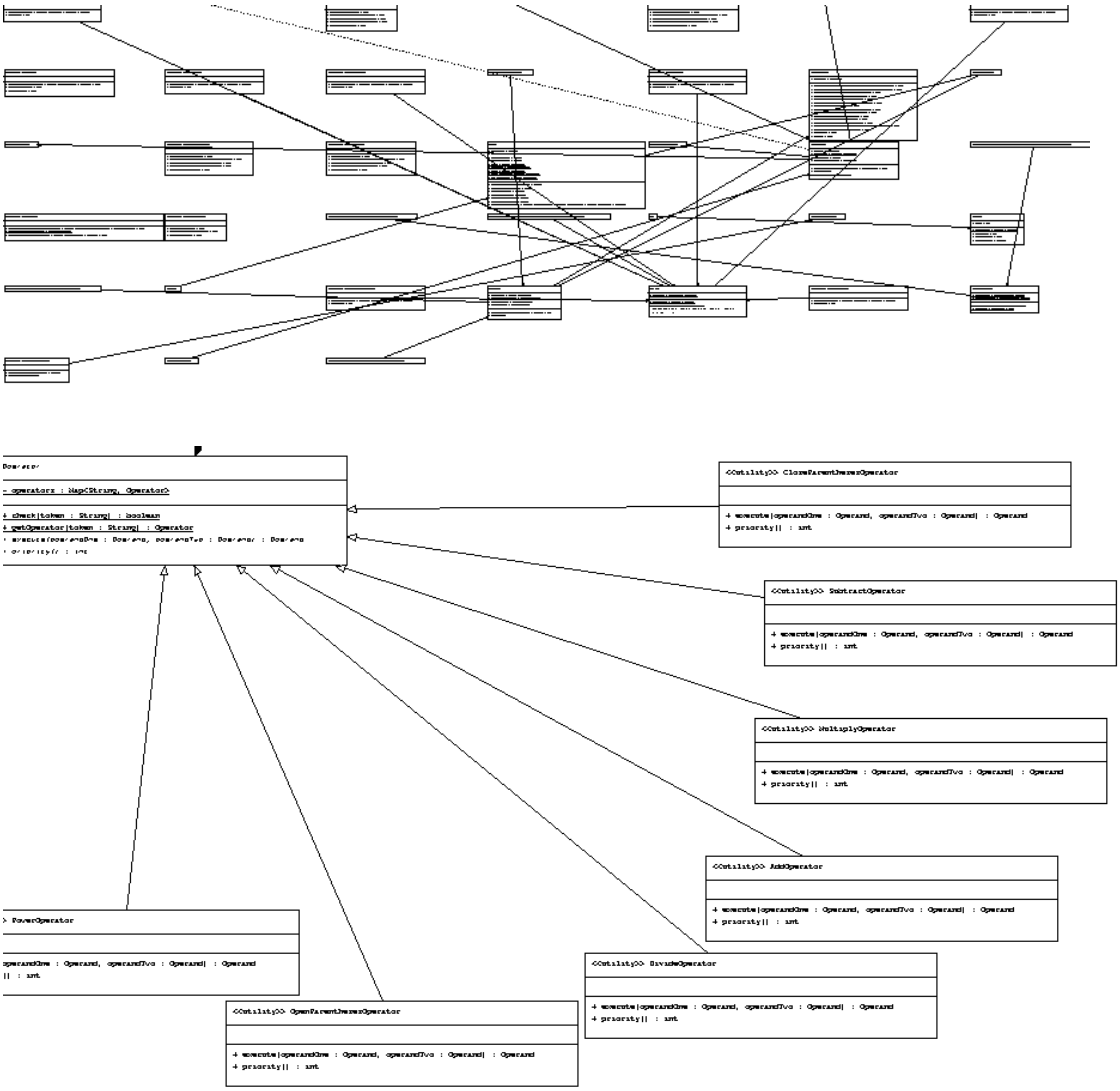
Some of the key design choices include:

- Object-Oriented Design: The project followed the object-oriented design paradigm, utilizing classes and objects to model the different components of the calculator. This design choice promotes encapsulation, modularity, and code reusability.

- Class Hierarchy: The use of class hierarchy was employed to represent the operators in the calculator. The abstract class "Operator" served as the base class, with concrete subclasses such as "OperatorAdd," "OperatorSubtract," and so on. This hierarchy allowed for easy extensibility and maintenance of the operator functionality.

- To handle the different operator types, a HashMap was used in the Operator class to store the operator strings as keys and their respective operator objects as values. This mapping approach provided a convenient and efficient way to retrieve the operator objects based on the input expression.

6.1 Class Diagram



7 Project Reflection

This project provided me with a valuable opportunity to evaluate my understanding of object-oriented programming (OOP) principles and concepts. It allowed me to revisit and refresh my knowledge of OOP, which I had studied a long time ago. As I worked on the assignment, I realized the importance of documentation in software development. The project highlighted how documentation plays a vital role in enhancing the understanding of a program.

8 Project Conclusion/Results

The project provided me with practical challenges that required creative problem-solving and critical thinking. I encountered various scenarios where I had to analyze requirements, make design decisions, and implement solutions using OOP principles. These challenges helped me refine my problem-solving skills and sharpen my ability to translate real-world problems into effective software solutions.