CSC 413 Project Documentation

Spring 2019

Nhan Nguyen

923100929

CSC413.01

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

Table of Contents

1 Introduction

1.1 Project Overview

This project is a calculator application that provides a user-friendly interface for performing arithmetic calculations. The calculator allows users to input mathematical expressions and obtain the evaluated results with ease. The calculator should support basic arithmetic operations such as addition, subtraction, multiplication, division, and power, as well as parentheses for grouping operations.

With its intuitive interface and comprehensive features, users can effortlessly perform mathematical operations and obtain accurate results. Whether you are a student, professional, or simply need a handy calculator, this application is designed to meet your needs efficiently and effectively.

1.2 Technical Overview

In technical terms, the project involves building a calculator application using Java programming language. The application follows the principles of object-oriented programming and it showcases the separation of concerns with distinct classes for expression evaluation, operator handling, operand management, and user interface.

The calculator employs the Shunting Yard algorithm to convert infix expressions into postfix notation, which can be easily evaluated. The algorithm assigns to each operator its correct operands, taking into account the order of precedence. This application incorporates the use of stacks to handle operator precedence and parentheses.

About the graphical user interface (GUI), the text field and buttons are organized in a visually pleasing manner, ensuring ease of use. By leveraging Swing components and event handling mechanisms, the calculator GUI facilitates a seamless interaction between users and the underlying evaluation logic.

More detailed information and technologies used will be provided below.

1.3 Summary of Work Completed

Several tasks were completed to ensure the proper functioning of the calculator application. The following contributions were made:

Designed and implemented the Operator hierarchy: The Operator class serves as the base class for different arithmetic operators such as addition, subtraction, multiplication, division, and power. Each operator subclass provides its own implementation for priority and execute methods.

Implemented the Operand class: The Operand class represents numeric values in the calculator. It handles the parsing of operands from the input expression and provides methods for retrieving and manipulating operand values.

Implemented the Evaluator class: The Evaluator class forms the core of the calculator application. It utilizes stacks to process operators and operands while evaluating the expression. The class incorporates the Shunting Yard algorithm and handles parentheses to ensure the correct order of operations.

Developed the EvaluatorUI class: The EvaluatorUI class is responsible for creating a graphical user interface (GUI) for the calculator. It utilizes Java Swing components to build the calculator interface, including buttons for input and display fields for the expression and result.

Integrated user input handling: The EvaluatorUI class listens for button clicks and handles user input. It appropriately updates the expression field and triggers the evaluation process when the equal (=) button is pressed.

2 Development Environment

a. Version of Java Used: Java 20 (Oracle JDK 20)
b. IDE Used: IntelliJ IDEA 2023.1.2 (Ultimate Edition)

3 How to Build/Import your Project

To import/build the calculator project, follow these steps:

1. Set up the Development Environment:
   i. Ensure you have Java Development Kit (JDK) installed on your computer. You can download the JDK from the official Oracle website.
   ii. Install an Integrated Development Environment (IDE) such as Eclipse, IntelliJ IDEA, or NetBeans. These IDEs provide a user-friendly environment for Java development.
2. Download the Project: Obtain the source code for the calculator project. This can be done by downloading the project files from a repository.
3. Open the Project in your IDE:
   i. Launch your chosen IDE and import the calculator project into it. This process may vary slightly depending on the IDE you are using.
   ii. Create a new Java project and configure it to use the existing source code files.
4. Build the Project: Ensure that the project builds successfully without any errors. If there are any compilation errors, review the code and resolve them.

4 How to Run your Project

After import/build the project, now we can run the calculator application, follow these steps:

1. Run the Application:
   i. Locate the main class file in the project, which contains the main method. In this case, it should be the EvaluatorUI.java file.
   ii. Right-click on that file and select "Run”.
   iii. The calculator application's graphical user interface (GUI) should appear on the screen.
2. Interact with the Calculator:
   i. Use the calculator's GUI to enter mathematical expressions and perform calculations.
   ii. Click on the number buttons to input digits, and use the operator buttons to perform addition, subtraction, multiplication, division, power, and parentheses.
   iii. The "C" button clears the entire expression, and the "CE" button clears the last entered character.
   iv. After entering an expression, click the "=" button to evaluate and display the result in the text field.

5 Assumption Made

a. Assumed the operands are integers: this application assumes that the operands used in the expressions are integers. It does not support decimal or floating-point numbers as operands.
b. Positive numbers only: The project assumes that the operands are positive numbers and does not handle negative inputs. It is designed to work with non-negative integer values.
c. Integer division: Since the operands are assumed to be integers, the division operation will perform integer division. This means that when dividing two integers, the result will be an integer without any fractional part. Any remainder will be discarded.
d. Limited operator support: The project includes support for basic arithmetic operators such as addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^), along with parentheses. It does not support more advanced mathematical functions or operators like sin, cos, etc.
e. Limited user interface: The graphical user interface (GUI) provided in the project assumes a basic layout with buttons representing the operands and operators. It does not include advanced features such as history tracking, saving value, or other function like table, graph, etc.

6 Implementation Discussion

a. Discuss design choice: The project was implemented following an object-oriented approach to ensure modularity and maintainability. The use of the Operator hierarchy allows for extensibility, as new operators can be easily added by creating their respective subclasses. The implementation of the Shunting Yard algorithm ensures that the calculator handles operator precedence correctly. Stacks were utilized to process operators.
b. Please see the picture of UML diagram in ***/documentation*** folder

7 Project Reflection

Throughout the development of the calculator project, I gained valuable insights into various aspects of software development and problem-solving. I had the opportunity to apply and enhance my knowledge of various programming concepts and technologies. Here are some key reflections on the project:

a. Programming Fundamentals: The project allowed me to reinforce my understanding of fundamental programming concepts such as variables, data types, control structures, and functions. It has been a long time since I last programmed in Java language. This provided me an opportunity to practice these concepts again.
b. Understanding and Implementing Algorithms: The project involved implementing the Shunting Yard algorithm to convert infix expressions to postfix notation. This algorithm required a clear understanding of stack operations and operator precedence. By implementing this algorithm, I enhanced my algorithmic thinking and problem-solving skills.
c. Object-Oriented Design: The project emphasized the use of object-oriented programming principles. Designing the Operator hierarchy and Operand class allowed for code reusability and maintainability. It helped me comprehend the significance of encapsulation, inheritance, and polymorphism in building modular and extensible software systems. OOP helped in organizing the codebase and promoting code reusability.

d. Error Handling and User Input Validation: To ensure the calculator handles errors gracefully, I incorporated exception handling mechanisms. It allowed me to learn how to anticipate and handle exceptional scenarios, such as invalid expressions or division by zero. Exception handling improved the overall robustness of the application.
e. Graphical User Interface (GUI) Development: The inclusion of a GUI component added a new dimension to the project. Implementing the EvaluatorUI class base on Java Swing allowed for a user-friendly calculator interface. It offered hands-on experience in GUI development and event handling.
f. Software Testing: Throughout the project, I recognized the importance of testing to ensure the correctness and reliability of the calculator. I follow unit tests to verify the functionality of individual components, as well as integration tests to validate the interactions between different modules. Testing helped me identify and fix issues early in the development cycle.
g. Version Control: I utilized version control systems like Git and use Github to manage the project's source code. It facilitated collaboration, allowed me to track changes, and provided a safety net in case of errors or regressions. Version control played a crucial role in maintaining code integrity and enabling easy code sharing.

In conclusion, the project served as a platform for my continuous learning and exploration of new concepts. It required researching and understanding different algorithms and techniques for expression evaluation. It helped me review my knowledge about Java programming language to prepare for the upcoming assignments. Additionally, it motivated me to stay updated with programming best practices and emerging technologies.

8 Project Conclusion/Results

In conclusion, the calculator project was successfully completed, resulting in a functional calculator application capable of evaluating arithmetic expressions. The project achieved the following outcomes:

a. Implementation of the core calculator functionality: The calculator can handle basic arithmetic operations such as addition, subtraction, multiplication, division, and power. It can also handle parentheses to ensure the correct order of operations.
b. User-friendly graphical interface: The EvaluatorUI class provides a visually appealing and intuitive interface for users to input expressions and view the results.
a. Error handling and input validation: The project incorporates appropriate error handling mechanisms to handle invalid expressions gracefully. It validates user input to ensure that only valid expressions are evaluated.
b. Modular and extensible design: The use of the Operator hierarchy and Operand class allows for easy extension of the calculator to support additional operators or functionality in the future.

Overall, the calculator project served as a valuable learning experience in implementing algorithms, object-oriented design, GUI development, and error handling. It demonstrated the successful application of these concepts to create a functional calculator application.