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 3](#_Toc522827688)

[1.1 Project Overview 3](#_Toc522827689)

[1.2 Technical Overview 3](#_Toc522827690)

[1.3 Summary of Work Completed 3](#_Toc522827691)

[2 Development Environment 3](#_Toc522827692)

[3 How to Build/Import your Project 3](#_Toc522827693)

[4 How to Run your Project 3](#_Toc522827694)

[5 Assumption Made 3](#_Toc522827695)

[6 Implementation Discussion 3](#_Toc522827696)

[6.1 Class Diagram 3](#_Toc522827697)

[7 Project Reflection 3](#_Toc522827698)

[8 Project Conclusion/Results 3](#_Toc522827699)

# Introduction

## Project Overview

This section only explains the project in high-level. Pretend the audience is not well versed in programming.

The project aims to develop a calculator application capable of evaluating arithmetic expressions. The calculator should support basic arithmetic operations such as addition, subtraction, multiplication, and division, as well as parentheses for grouping operations. The goal is to create a user-friendly calculator interface that allows users to input expressions and obtain the correct result.

## Technical Overview

When writing the technical overview, you can pretend the audience is an experience programmer.

In technical terms, the project involves building a calculator application using Java programming language. The application follows the principles of object-oriented programming and utilizes various data structures and algorithms to parse and evaluate arithmetic expressions. The calculator employs the Shunting Yard algorithm to convert infix expressions into postfix notation, which can be easily evaluated. The project also incorporates the use of stacks to handle operator precedence and parentheses.

## Summary of Work Completed

Work completed. Please discuss what you contributed to the assignment to get it working correctly. If you did not finish also list what was left not 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.

Created 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.

# Development Environment

a. Version of Java Used: Java 20 (Oracle JDK 20)

b. IDE Used: IntelliJ IDEA 2023.1.2 (Ultimate Edition)

# How to Build/Import your Project

Note saying things like hit the play button and/or click import project is not enough. You need to explain how to import and/or build the game.

# How to Run your Project

To run the calculator project, follow these steps:

1. Ensure you have Java Development Kit (JDK) installed on your machine.
2. Download the project source code and extract it to a directory.
3. Open a command prompt or terminal and navigate to the project directory.
4. Compile the Java source files by running the following command: javac edu/csc413/calculator/\*.java
5. Once the compilation is successful, run the application using the following command: java edu.csc413.calculator.evaluator.EvaluatorUI
6. The calculator GUI should appear, allowing you to input expressions, evaluate them and give the result.

# Assumption Made

assume the operands are integers

Assumed operands were just positive numbers.

no float values when dividing with whole numbers? since operands are ints anyway?

cannot have negative inputs

# Implementation Discussion

1. Discuss design choice made while implementing your assignment.

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.

1. Please include a UML diagram of your assignment. Files related to testing do not need to be included.

## Class Diagram

# Project Reflection

Throughout the development of the calculator project, I gained valuable insights into various aspects of software development and problem-solving. Here are some key reflections on the project:

1. 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.
2. 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.
3. Error Handling and User Input Validation: Implementing error handling and validation mechanisms was crucial to ensure that the calculator handled invalid expressions gracefully. This aspect provided me with insights into handling exceptions and user input validation effectively.
4. Graphical User Interface (GUI) Development: The inclusion of a GUI component added a new dimension to the project. Implementing the EvaluatorUI class using Java Swing allowed for a user-friendly calculator interface. It offered hands-on experience in GUI development and event handling.

# 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:

1. Implementation of the core calculator functionality: The calculator can handle basic arithmetic operations such as addition, subtraction, multiplication, and division. It can also handle parentheses to ensure the correct order of operations.
2. User-friendly graphical interface: The EvaluatorUI class provides a visually appealing and intuitive interface for users to input expressions and view the results.
3. 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.
4. 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.