CSC 413 Calculator Documentation

Andrew Hwang

918450486

CSC 413-02 Spring 2020

https://github.com/csc413-02-spring2020/csc413-p1-AndrewHwang97

Table of Contents

1 Introduction

1.1 Project Overview

The calculator program is a program designed to take mathematical expressions from the user from a keypad on the screen, and output the solutions on the text line. The way it solves these expressions is by solving expressions in the following order: parentheses, exponents, multiplication and division, and addition and subtraction. When a user finishes an expression, they can continue to add on expressions to the answer given.

1.2 Technical Overview

The calculator program is designed by using multiple classes, as well as abstract classes to achieve the solution. An abstract class named Operator was used to hold properties and functions of all operators, there are two abstract functions used in the class named priority() and execute(), that are later implemented in the child classes of Operator. There are six child classes that extend Operator that override these abstract functions. Priority() returns an integer giving a priority of the operator for the expression, and execute() performs the operation based on the operator. The Operand class is a class that holds properties of each operand. it holds its value, and has methods that can get the value of the operand and to check if it is a valid operand using ASCII values. The Evaluator class is the class that takes tokens from input and evaluates expressions given to it. It uses two stacks, one for operators and one for operands. The evaluator follows a loop:

> while there are more tokens to be scanned, if it is an operand, it is pushed to the operand stack, otherwise it is pushed to the operator stack given that it's valid. If the operator stack is not empty and there is not a right parentheses, and the operator at the top of the stack has a greater priority than the current one, then we process the stack, popping twice from the operand stack, and popping once from the operator stack, and pushing the result back into the operand stack, otherwise, we push the operator to the stack.

After the loop completes, there are still operators and operands left over in the stack. The program processes the stack one last time and returns the Integer value of the result. The GUI was implemented based on a series of buttons and a textbox for the input. One major method in the UI class performs actions based on user input. This was implemented using a switch statement to determine if a user pressed any of the function keys, otherwise it added the input to a String containing the input. The text box is updated to output what the String contains. Two methods were added to give functionality to the *C* and *CE* buttons. One method clears the whole expression, and the other removes an input from the expression. If the = key is pressed, the program uses an Evaluator object made, and performs the evaluation of the String text.

1.3 Summary of Work Completed

Skeleton Classes were given at the start of this project. the Operator class was implemented. The six classes for the operators were created and implemented as well. Operand class was completed by adding to the methods and giving value to the Operand. The evaluator class was worked on, specifically the loop of the Evaluator. As well as the addition of a method that processed both stacks. In the UI class, the actionPerformed method was worked on and completed as well as the addition of a clear function and a removeInput function. With the contributions made to the project, the Program works correctly and as intended.

2 Development Environment

For this project, Java version 10.0.2 was used. The Intellij IDE was used to complete this project.

3 How to Build/Import your Project

To build/import the project to the Intellij IDE, the project will need to be downloaded from github. Once the project is downloaded from github, open Intellij. Once Intellij is opened, Click on the "Project from existing sources" Under File > New. Find the folder containing the project, highlight the calculator folder, and hit the 'next' button. A series of options will appear. The default options are sufficient to use for the project.

4 How to Run your Project

Once the project has been imported to the IDE, It can be ran by right clicking the EvaluatorUI file under the calculator folder, and using the Run option. Once that is completed, the GUI for the calculator will appear on the screen.

5 Assumption Made

There were some assumptions made when designing and implementing the project. One Assumption made was that the user input on the calculator is valid and the expressions are valid. If a invalid expression is inputted, nothing occurs, however errors appear in the console. Another assumption made was that we assumed that the user would want to continue making expressions after the evaluation was performed. When a result is shown, and a user hits a number key, the result does not go away, it continues the expression. Assumptions were made for the C and CE keys, for this project the C key cleared the input text, and CE clears one input from the input text.

6 Implementation Discussion

For this project, organization of the project was a priority. Seperate classes and packages were used to achieve this. Adding the abstract class for the operators was used so that if there needs to be another operator added to the project, it would be simple to implement without having to change much code. Same goes for using a hashmap that held the operators. If another operator was added to the project, the addition would be simple as we would just need to add to the hashmap. The methods added to the project such as the clear() and CE functions were implemented to make the code more readable, as well as keeping the project organized. For the UI, a switch statement was used for actions for cleanliness of the code rather than using if-statements that would have been harder to read. The UML diagram can be found in the Documentation folder of this project.

7 Project Reflection

I feel like the calculator assignment was a good assignment to start off the course with. It was simple enough to do, but I also learned a lot from this assignment. In most of all the courses taken before 413, we do assignments from scratch and write them out. In this assignment, having starter code taught me how to work with code that is already there. I had to first try to understand what each file did, and how the calculator was supposed to work before I even started working on it. I feel like that is great since outside of school, we most likely won't be making programs from scratch, but working on existing code. The project was great as it gave a refresher on some of the design principles of Java, Object Oriented Programming and Abstract Classes, as well as gave some insight on how GUIs work. I also needed to use the debug tool a lot over the course of this assignment and I never realized how helpful it can be.

8 Project Conclusion/Results

For this project, we were given the task of creating a calculator program that gave users the ability to input mathematical expressions using a keypad on a GUI, and having the calculator perform evaluations of the expressions given to it. The project utilized multiple classes to make the calculator work. a Operator class was used as well as an Operand class. The Evaluator class was used and implemented both the Operator and Operand class to process the expressions. Some assumptions were made regarding input from the user and what the user expects the program to do. As a result of the work contributed to this project, the program is working correctly and as intended to.