CSC 413 Project Documentation

Spring 2023

Hoang-Anh Tran

922617784

Class.Section 02

https://github.com/csc413-SFSU-Souza/csc413-p1-htran31.git

Table of Contents

1 Introduction

1.1 Project Overview

Expression Evaluator is an object-oriented design project that uses the Java language to create two programs including an object that evaluates mathematical expressions and a graphical user interface for this object - Calculator. The purpose of the project is to practice, analyze, and apply synthetic mathematical expressions such as addition, subtraction, multiplication, and division to create a basic math calculator.

1.2 Technical Overview

The purpose of the project is to practice object-oriented design to create two programs:

1. An object that evaluates mathematical expressions
2. A GUI - a simple math calculator

1.3 Summary of Work Completed

Work Completed:

- Complete the Operand class (Operand.java) by constructing an operand from a string token and from an integer, return the integer value of the operand, and return true if the given token is a valid operand.
- Complete the Operator abstract superclass class (Operator.java)
  - Initialize an instance of a HashMap with all the Operators stored as values, keyed by their token
  - Create the individual Operator classes that be subclassed from Operator to implement each of the operations allowed in our expressions
  - Returns true if the specified token is a valid operator
- Complete the actionPerformed() function in EvaluatorUI.java
  - If not C, CE, or = buttons, then add value to the text field
  - If the C button, then delete one last character
  - If the CE button, then delete all the text field
  - If = button, execute the expression in the text field
- The Evaluator.java
  - successful execution of basic expressions (without parentheses)
  - return true results for expressions containing parentheses

Work Uncompleted: (Incomplete the Evaluator.java)

- Unpassed expressions containing parentheses.

2 Development Environment

a. Version of Java Used: Java 11 version 11.0.12
b. IDE used: IntelliJ IDEA

3 How to Build/Import your Project

1. On File, choose New, choose 'Project from Existing Sources...'
2. Choose the correct path of the 'calculator' file (under csc413-p1-htran31), then click OK.
   C:\Users\hoang\Desktop\csc413-p1-htran31\calculator\src\main
3. Choose 'Create project from existing sources', then click Next.
4. Put the name of the project, then click Next.
5. Click Next until click Create button (make sure all needed jars are in the Library Contents).

4 How to Run your Project

To evaluate expressions, choose the EvaluatorDriver.java:

- Run random expressions that you would like to evaluate:
  Right click, then select Run 'EvaluatorDriver.main()'
  (OR) Ctrl+Shift+F10
- Run auto the results from a HashMap list of expressions are given as command line arguments:
  1. Select Run/Debug Configuration Box
  2. Choose 'Edit Configurations...'
  3. Choose ‘EvaluatorDriver’
  4. At the box 'Program arguments', type "auto"
  5. Click Apply, then press OK

To run the GUI - Math Calculator, choose the EvaluatorUI.java:

Right click, then select Run EvaluatorUI.main()'
(OR) Ctrl+Shift+F10

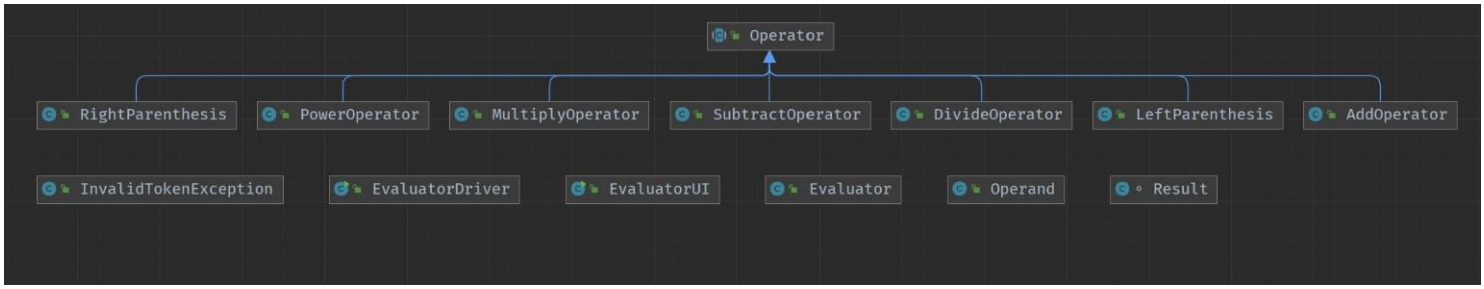
5 Assumption Made

- If an operand token is scanned, an Operand object is created from the token, and pushed to the operand Stack
- If an operator token is scanned, and the operator Stack is empty, then an Operator object is created from the token, and pushed to the operator Stack
- If an operator token is scanned, and the operator Stack is not empty, and the operator’s precedence is greater than the precedence of the Operator at the top of the Stack, then and Operator object is created from the token, and pushed to the operator Stack
- If the token is (, and Operator object is created from the token, and pushed to the operator Stack
- If the token is), the process Operators until the corresponding (is encountered. Pop the (Operator.
- If none of the above cases apply, process an Operator.
  - Pop the operand Stack twice (for each operand)
  - Pop the operator Stack
  - Execute the Operator with the two Operands
  - Push the result onto the operand Stack

When all tokens are read, process Operators until the operator Stack is empty.

6 Implementation Discussion

6.1 Class Diagram



Create new subclasses of the operator to set their precedence and return the compatible operation for the operands:

- AddOperator
- SubtractOperator
- MultiplyOperator
- DivideOperator
- PowerOperator
- LeftParenthesis
- RightParenthesis

7 Project Reflection

'Expression Evaluator' is a great project to practice object-oriented design using Java language. What I learned more about how to:

- use abstract superclass.
- assumes and parses pop() and push() logical operators and operands to the Stack.
- discover and execute the provided GUI Calculator.

However, I still haven't completed the project fully because I haven't figured out how to execute expressions containing parentheses. I am extremely happy that I executed their results correctly. However, I'm stuck figuring out how to 'pass' those expressions. I believe it's because I don't know how to properly push parentheses onto the operator stack.

Besides, I feel lucky to have found a way to implement the GUI because I was stuck for a long time controlling the C, CE, =, and ^ buttons. The order of operands like *, /, or ^ on execution also gives me a headache. Having learned from this project, I hope I start the project sooner and I should make an appointment with a tutor for help once.

8 Project Conclusion/Results

In conclusion, the Expression Evaluator project allows me to practice and get acquainted with the Java language better. The project helped me learn how to analyze logic when working with Stack, HashMap, and abstract classes. Moreover, I enjoyed my first time practicing with GUI in Java language. This provides an intuitive and interesting experience when learning to code.