CSC 413 Project 1 - Calculator Documentation
Summer 2024

Bryan Lee

922649673

Section 1-R4

https://github.com/csc413-SFSU-SU2024/calculator-BryanL43/tree/main

Introduction

Project Overview:

This is a simple calculator that evaluates simple math operations, including expressions with parenthesis. There is a display for individuals to interact with the calculator app that also displays errors when an expression is not valid.

Technical Overview:

This is an overly engineered calculator that solves mathematical equations by having abstract and encapsulated classes to represent mathematical operations. The operators and operands are stored in a stack and compiled based on priority given to the operators. Each computation is done within the subclasses of the operator and in the order of the stacks. Lastly, a GUI is created using the Swing library and the buttons are handled via a switch operation.

Summary of work completed:

My entire project is completely working. A big part of my development process is trial and error, especially with various methods of implementing parenthesis evaluation. The reading for the Oxford infix expression evaluator algorithm was also extremely useful in designing my code.
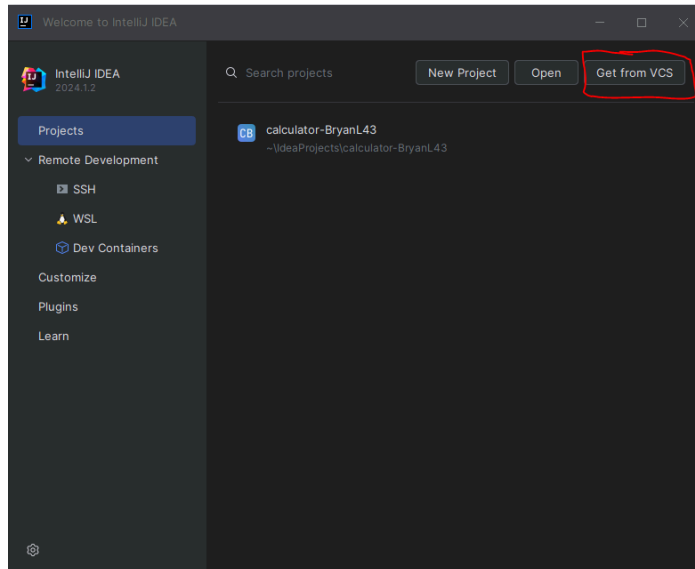
Development environment
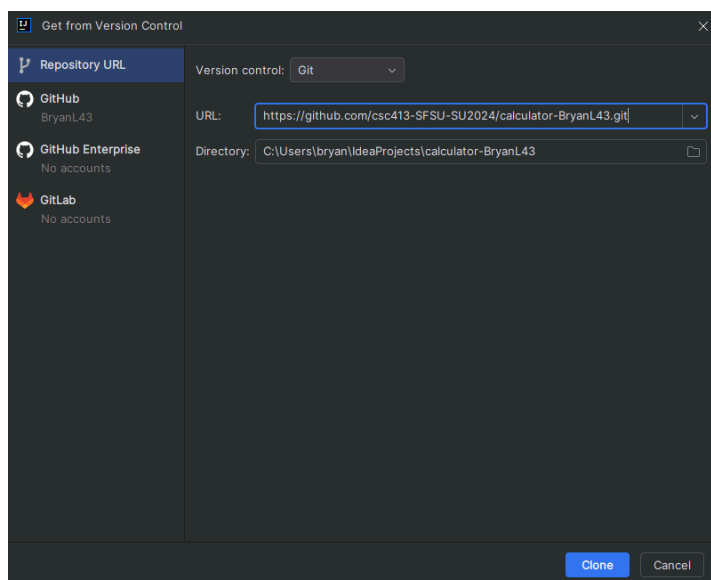   a. Version of Java used: Oracle OpenJDK 21.0.2
   b. IDE Used: IntelliJ

How to run/build your project
Method 1: Cloning from IntelliJ
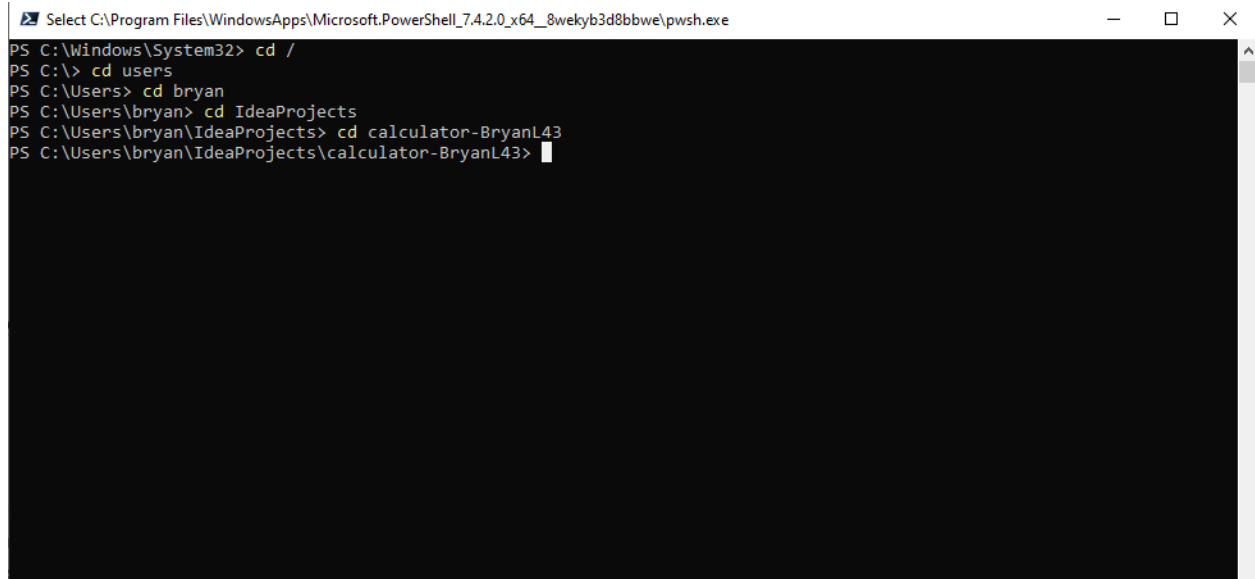Step 1: Click "Get from VCS"

Step 2: paste the copied git link and click clone.

Step 3: Navigate to the EvaluatorUI class and run the main method within it.

Method 2: Powershell

Step 1: Navigate to the directory of the downloaded file

```
Select C:\Program Files\WindowsApps\Microsoft.PowerShell_7.4.2.0_x64__8wekyb3d8bbwe\pwsh.exe                    —    □    ×
PS C:\Windows\System32> cd /
PS C:\> cd users
PS C:\Users> cd bryan
PS C:\Users\bryan> cd IdeaProjects
PS C:\Users\bryan\IdeaProjects> cd calculator-BryanL43
PS C:\Users\bryan\IdeaProjects\calculator-BryanL43> █
```

Step 2: Compile the program using the given commands (the tests are optional).
Assuming Windows OS.

Compile:

javac -d target .\calculator\evaluator\Operand.java

javac -d target .\calculator\operators*.java

javac -d target .\calculator\evaluator*.java

Compile Test:

javac -d target --class-path "./;.\lib\junit-platform-console-standalone-1.9.3.jar"
.\tests\operator*.java

javac -d target --class-path "./;.\lib\junit-platform-console-standalone-1.9.3.jar"
.\tests\operand*.java

javac -d target --class-path "./;.\lib\junit-platform-console-standalone-1.9.3.jar"
.\tests*.java

Run unit test:

java -jar .\lib\junit-platform-console-standalone-1.9.3.jar -cp target --scan-classpath

Step 3: Run the following command to deploy the calculator GUI

```
java -cp target calculator.evaluator.EvaluatorUI
```

Assumption made

I've assumed that the first operand cannot be a negative value but the result can be. Another assumption is that the parenthesis isn't treated as a multiplication operation i.e. 2(5) = 10 but simply as a separator for PEMDAS logic. The last assumption I made was creating subclasses for parenthesis, which was not implicitly stated in the instructions. However, this was a design choice I made as explained below.

Implementation discussion

Design choice:

A design choice I had to make was creating a new subclass called OperatorInitializer to put the created operators into the Hashmap instead of using a static block. This resolves the deadlock warning without interrupting other functionalities of the Operator class but requires a new method of initializeOperators to initialize the subclass and add to the hashmap. The initializeOperators method had to be executed within the getOperator and check method to run it. Another design choice I made was assigning the parenthesis as operators and pushing them to the operator stack. This was easier to interpret and ensure accuracy when developing the infix expression algorithm. The rest of the design is simply the given project structure.

UML Diagram:
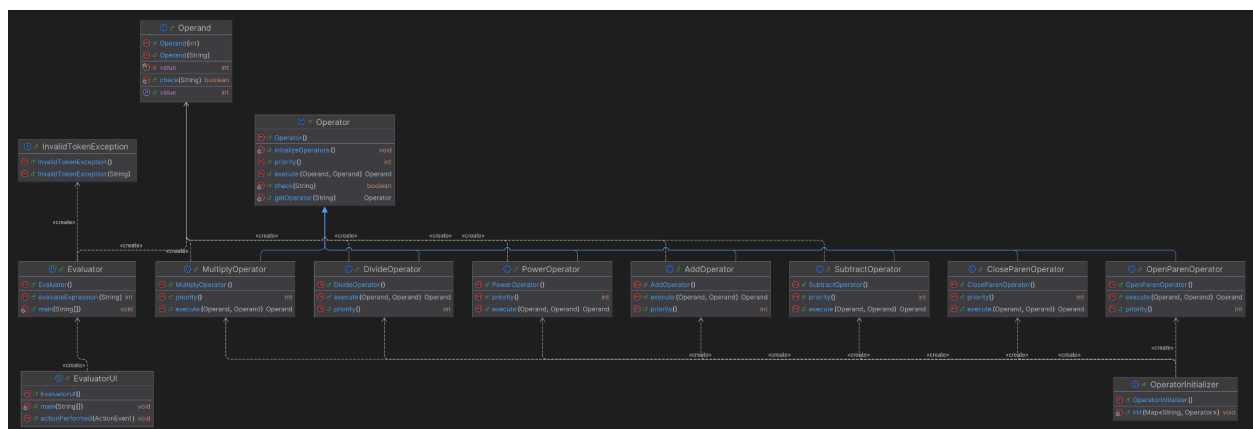
A better-quality image is located in the documentation folder > calculator-UML-Diagram.png

Project reflection

This project was helpful as a refresher to Java and the practical application of OOP. It gave me hands-on practice with abstract classes that most of my other courses only taught as theory. The project also gave me some practice with the Swing library for Java

GUI. I've used JavaFX before and it's similar to Swing so it wasn't too difficult to implement the UI features. Overall, I was able to finish this project fairly quickly and was able to refresh my Java knowledge.

Project conclusion and results

This project wasn't too difficult and was straightforward. I made careful design choices and even tested multiple ways to implement the parentheses evaluation, which is the most interesting part. I followed the given infix expression evaluator algorithm, primarily focusing on the action taken in the right parentheses, prompting my decision to implement the parentheses as an operator placeholder. Overall, I am confident I've completed the assignment and compiled my code with IntelliJ and Powershell.