CSC 413 Project Documentation

Spring 2023

Seyoung Kim

923109262

CSC413-02

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

Table of Contents

1 Introduction

1.1 Project Overview

This program calculates mathematical expressions. Its operators are ()*-*/, and, and its input and output are always integers.

1.2 Technical Overview

This program is a GUI calculator.

This program uses two stacks, operator and operand. Each Operator is implemented as a Class, and Operator class uses a hashmap to store operators. This program calculates infix expressions. Its operators are ()*-*/, and, and its input and output are always integers.

1.3 Summary of Work Completed

a. Operand

- Constructor creates an operand object from an integer or string token. It initializes its value to its token(parameter).
- Method getValue() returns its value.
- Static method check() checks if a given token is a valid operand.

b. Operator

- Hashmap operators store each operator (+-/*^()).
- Each operator class extends abstract class Operator.
- Method priority() returns its priority.
- Method execute() executes an operator given two operands.
- Static method getOperator() returns an operator object from Hashmap.
- Method getToken() returns its token.
- Static method check() checks if a given token is a valid operator.

c. Evaluator

- Method processOperator() processes the operator stack.
- Method evaluateExpression() calculates expression. If there is unvalid token, it throws InvalidTokenException.

d. EvaluatorUI

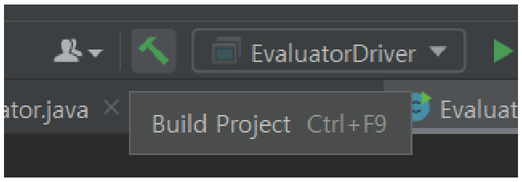
- Method actionPerformed() calculates the expression and prints its expression and result on the screen.

2 Development Environment

a. Version of Java Used: Oracle OpenJDK version 19.0.2

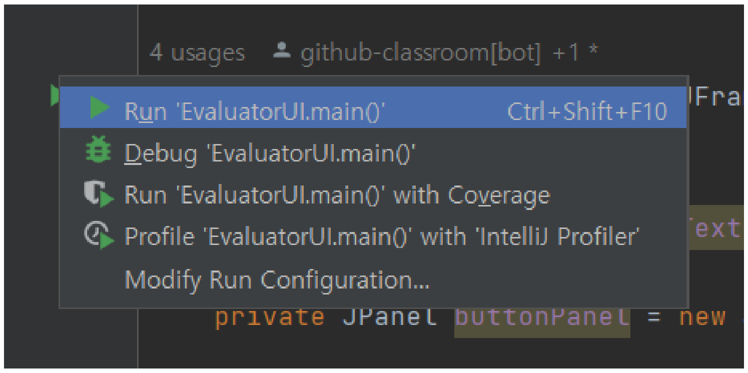
b. IDE Used: IntelliJ IDEA 2022.3.2 (Ultimate Edition)

3 How to Build/Import your Project



Download the code and press ‘Build Project’ button.

4 How to Run your Project


4 usages
github-classroom[bot] +1 *
Run 'EvaluatorUI.main()' Ctrl+Shift+F10
Debug 'EvaluatorUI.main()'
Run 'EvaluatorUI.main()' with Coverage
Profile 'EvaluatorUI.main()' with 'IntelliJ Profiler'
Modify Run Configuration...

```
private JPanel buttonPanel = new
```

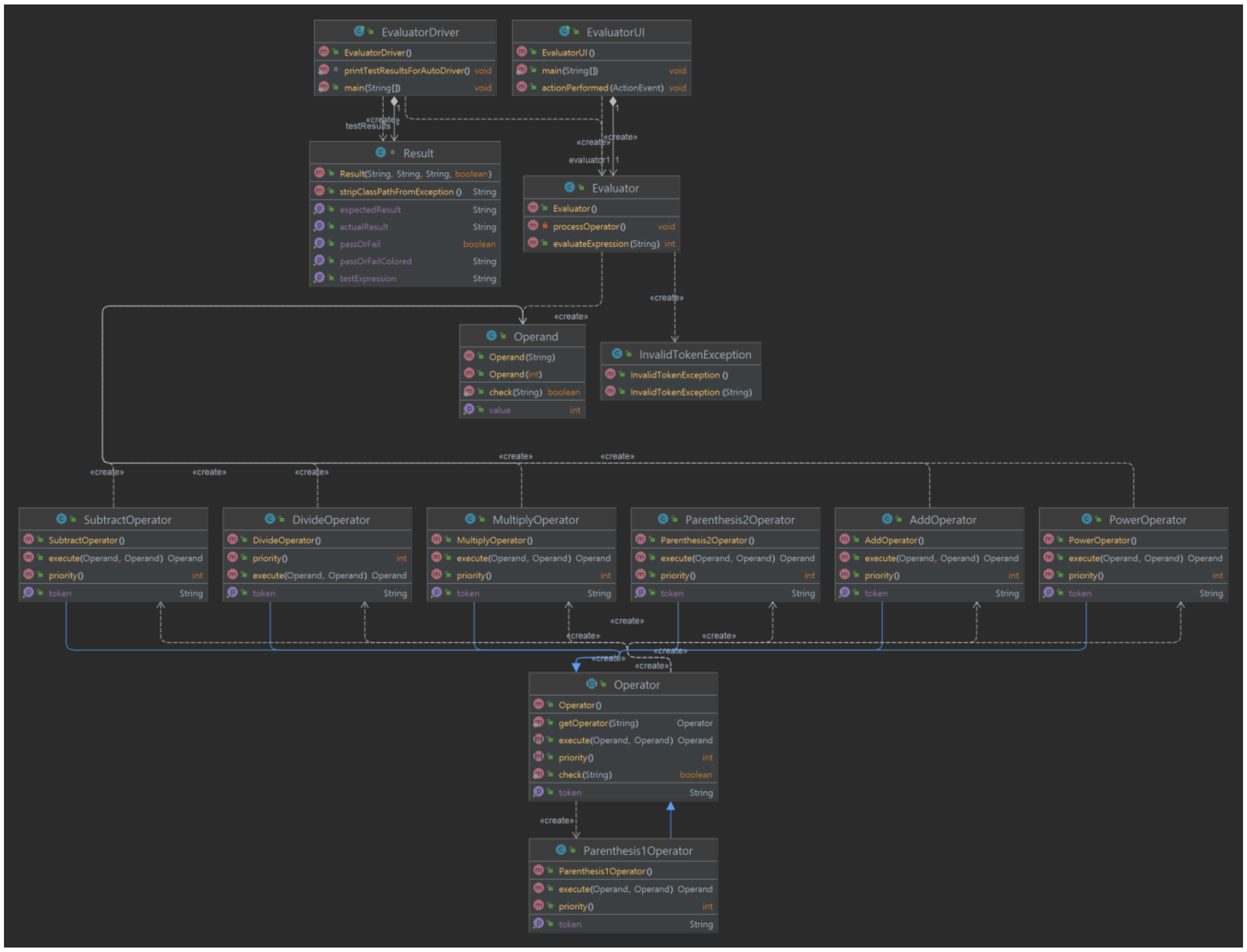
Click the ‘Run ‘EvaluatorUI.main()’’ button. EvaluatorUI file is in the src/main/java/edu.csc413.calculator/evaluator directory.

5 Assumption Made

a. All input operands are integers.

b. The wrong operators are not permitted.

5.1 Class Diagram

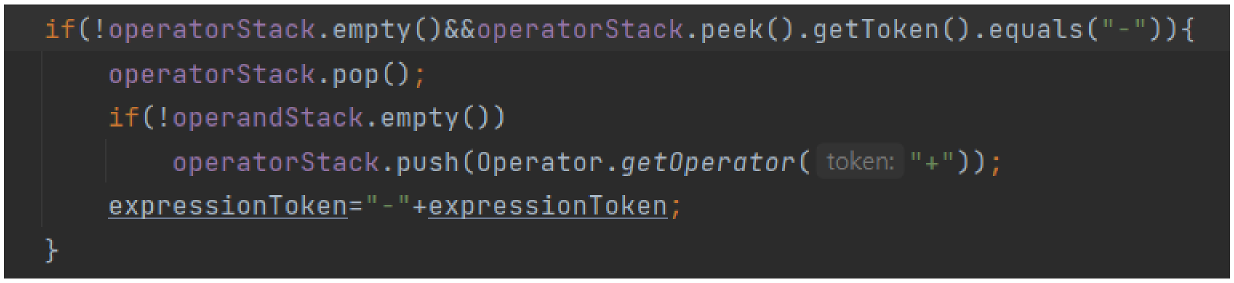


5.2 Design Choice

To use encapsulation, a variable in Operand class is private. In order to use this variable outside the class, getValue() method is public.

Class Operator is abstract, so each Operator class is implemented, which extends Operator class.

As processing operator is used repeatedly in evaluator class, I created the method processOperator() to prevent repetition.



```
if(!operatorStack.empty()&&operatorStack.peek().getToken().equals("-")){
    operatorStack.pop();
    if(!operandStack.empty())
        operatorStack.push(Operator.getOperator( token: "+"));
    expressionToken="-"+expressionToken;
}
```

I altered -number to +(-number) if operandStack is not empty in order to pass veryDifficultExpressionMixtureOfOperatorsNestedParensTest.

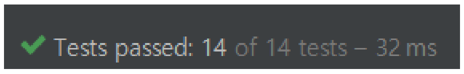
I utilized containsKey() function to avoid static checks in getToken() method.

I set the initialization block in Operator class to put each operator in the hashmap only once.

6 Project Reflection

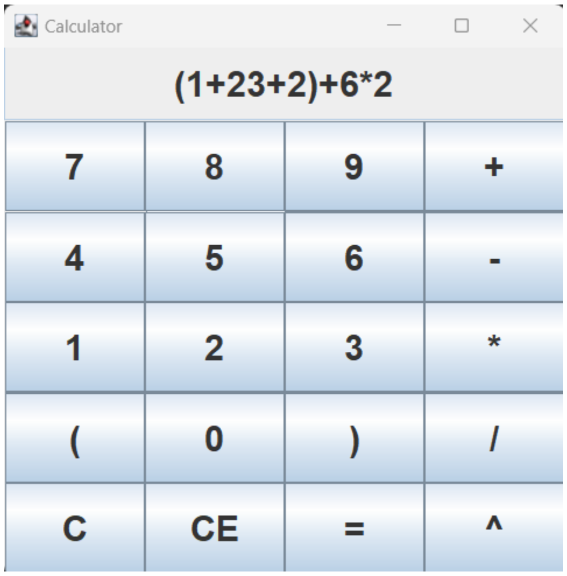
It was a great chance to learn how public, private, and protected access modifiers are used. Also, I was able to use Stack, and Hashmap data type to store data.

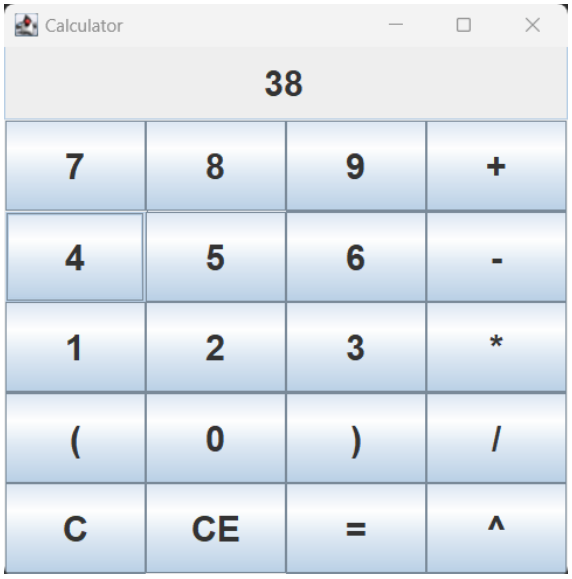
7 Project Conclusion/Results



```
✓ Tests passed: 14 of 14 tests – 32 ms
```

This program passes all tests.





Type the expression and press '=' button, and the result is printed on the screen.

C is for clear, clears entire expression.

CE is for clear expression, clears last entry up until the last operator.