Name: Ting Feng Student ID:922992561

Class: CSC413

Semester: Summer 23

A link to the repo: https://github.com/csc413-SFSU-Souza/csc413-p1-ChristineLoveCoding

2. Introduction

a. Project Overview

This is a simple calculator, which allows users to do Addition, subtraction, multiplication and division, exponent operations.

b. Technical Overview

The parent class is Operator, then We have five child classes AddOperator, SubtractOperator, DivideOpertor, MultiplyOpertaor and PowerOperator.

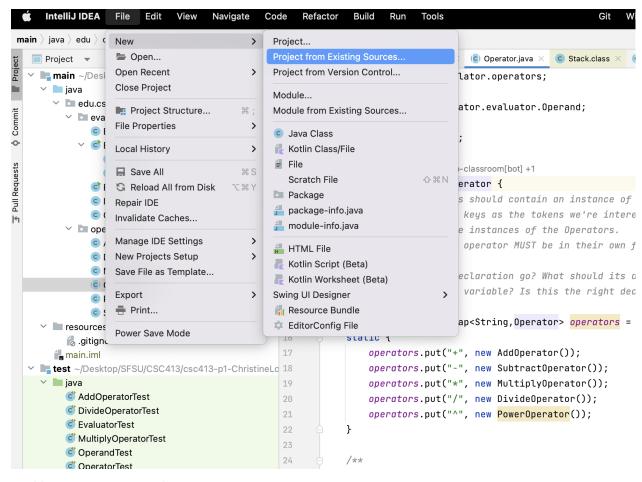
Evaluator class uses two Stacks to track Operands and Operators, so that higher priority operators will execute first. We use a recursive Algorithm to deal with nested parentheses.

c. Summary of work completed

I finished implementations of the operators, child of operator, evaluator and I have run all test cases in the test folder. I built a GUI that invoked the evaluator from the action handler. When the user input "C", it will clear all data on the input screen. When the user input "CE", the last integer will be removed.

- 3. Development environment.
- a. Version of Java Used: Java 17
- b. IDE Used: IntelliJ
- 4. How to build or import your game in the IDE you used.

Firstly, get into any project, click file and right click new, then select Import from Existing Sources. See this screenshot.



5. How to run your project

```
main / java / equ / csc413 / calculator / ev
                               🕃 😤 🔯 — 🔋 Evaluator/java 🗴 🄞 AddOperator/java 🗴 🔞 Operator/java 🗴 🔞 Operator/java 🗴 🕲 Operator/java 🗴 📽 Evaluator/Driver/java 🗴 📽 Evaluator/Driver/java
    main ~/Desktop/SFSU/CSC413/csc413-p1-ChristineL
                                                            package edu.csc413.calculator.evaluator;
     ✓ iava
       ✓ ■ edu.csc413.calculator
                                                            import javax.swing.*;
          evaluator
                                                            import java.awt.*;
              © Evaluator
                                                            import java.awt.event.ActionEvent;
              c EvaluatorDriver.java
                 EvaluatorDriver
                                                            import java.awt.event.ActionListener;
                 C Result
              © EvaluatorUI
              InvalidTokenException
                                                                                    tends JFrame implements ActionListener {
              Operand
                                                          Run 'EvaluatorUI.main()'

∨ □ operators

    AddOperator

                                                        Run 'EvaluatorUI.main()' with Coverage
              DivideOperator
                                                                                           essionTextField = new JTextField();
                                                           Modify Run Configuration...

    MultiplyOperator

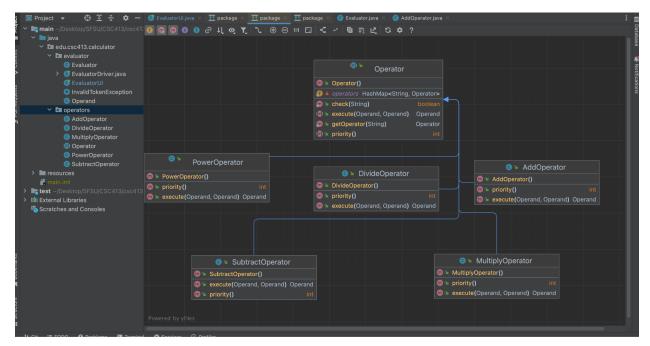
              Operator
                                                                private JPanel buttonPanel = new JPanel();
              SubtractOperator
                                                                // total of 20 buttons on the calculator,
          aitignore.
                                                                // numbered from left to right, top to bottom
                                                                // bText[] array contains the text for corresponding buttons

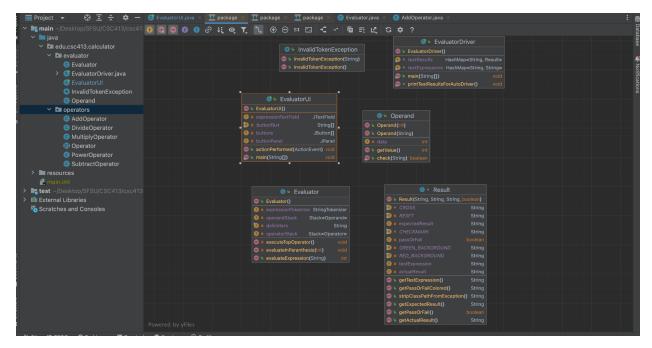
✓ Intest ~/Desktop/SFSU/CSC413/csc413-p1-ChristineL
                                                                5 usages
                                                                private static final String[] buttonText = {
          d AddOperatorTest
          © DivideOperatorTest
                                                                     "7", "8", "9", "+",
          C EvaluatorTest
                                                                     "4", "5", "6", "-",
          d MultiplyOperatorTest
                                                                     "1", "2", "3", "*",
          © OperandTest
                                                                     "(", "0", ")", "/",
          © OperatorTest
                                                                     "C", "CE", "=", "^"
          @ PowerOperatorTest
          SubtractOperatorTest
     > resources
     III External Libraries
                                                  25
                                                                 * C is for clear, clears entire expression
     Scratches and Consoles
                                                  26
                                                                 * CE is for clear expression, clears last entry up until the last operator.
                                                                 */
arks
```

- 6. Assumptions Made when designing and implementing your project At first sight, I would use hashmap and stack to implement the operator class.
- 7. Implementation Discussion
- a. Discuss design choices made while implementing your assignment.

After explorations, I decided to use recursion to handle nested parentheses. The skeleton uses an abstract Operator class to hide the Evaluator from knowing the details of each operator.

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





8. Project reflection

Testing is great and it really helps me understand the requirements of the class. The test covered the all the cases

And the ability to solve new problems is very important. Official online Java Doc really helps me understand things.

9. Project Conclusion and Results.

In the end, I finished the projects successfully. The most difficult part is how to deal with the nested parentness. I tried so many ways, and finally I achieved it by using a recursion Algorithm.