Muzaffar Sharapov

SFSU ID:921548703

February 4, 2021

Assignment 1 Documentation

Github Repository

https://github.com/sfsu-csc-413-spring-2021-roberts/assignment-1-calculator-mansursh

Project Introduction and Overview

In the evaluator/calculator project I had to implement object oriented design, infix expression evaluator, follow the skeleton project.

Tasks	Completed
Implement the eval method of the Evaluator class	x
String testing in EvaluatorTester.java. Passed test cases:	x
"1+1"	x
"7-1"	x
"1-7"	x
"2^3"	x
"3^2"	x
(1+1)^2+4	x
9*8/8	x
Used proper identifiers to explain methods	x
Requirement 1	x

Requirement 2	x
Requirement 3	x

Assumption

I assumed the skeleton that we were provided was correct and not misleading. Thus I assumed there were no specific test cases that would break my app from skeleton code.

Evaluator

In Evaluator class, the tricky part was eval method. In the test case with open and closing parenthesis I had to check if the top of operator stock was equal to open parenthesis “(” if true, then I had to get an index of (,) so I could solve inside the parentheses first.

EvaluatorUI

Even though I’ve developed many iOS apps in swift using programmatic UI. It seems GUI in Java language is harder to implement as Java language breaks down easier out of nowhere and I cannot even see where the problem is. It’s harder to debug in Java.
In the evaluatorUI class, actionPerformed method I tried many ways to make actionPerformed method work. But the only way I was able to make it work without crushes is by using an if statement for each button thus it took many lines of code. In EvaluatorUI i had get the a value of a button that was pressed and add to the TextField in case if there is a value already.

EvaluatorTester

In EvaluatorTester class I created simple test code that checks strings if they are solved correctly.

Operand

In operand most of the job was done already. In the boolean check method I used try catch statements to catch exceptions(non integer) when converting a string into int.

Operators

In Operator class. There was a hint: public abstract Operand execute(Operand fristOperand, Operand secondOperand); which indicates to use Strategy Design Pattern that we learned from our lectures. I created separate operator classes such as additionOperator.java, SubtractingOperator, to implement Strategy Design Pattern for the algorithm.

Execution and Development Environment

I executed and developed the app by using Visual Studio Code on my old MacBook. To create the Evaluator app, I had to break down the app into small parts, and I used separate playgrounds to implement each method. I had to visualize and draw the process, goals, and errors on the piece of paper. Use breaking points, debugging. I did my best to break, crush the app to see what error it would show, and resolved each error one by one.

Command Line Instructions to Compile and Execute

To compile we need to type in terminal

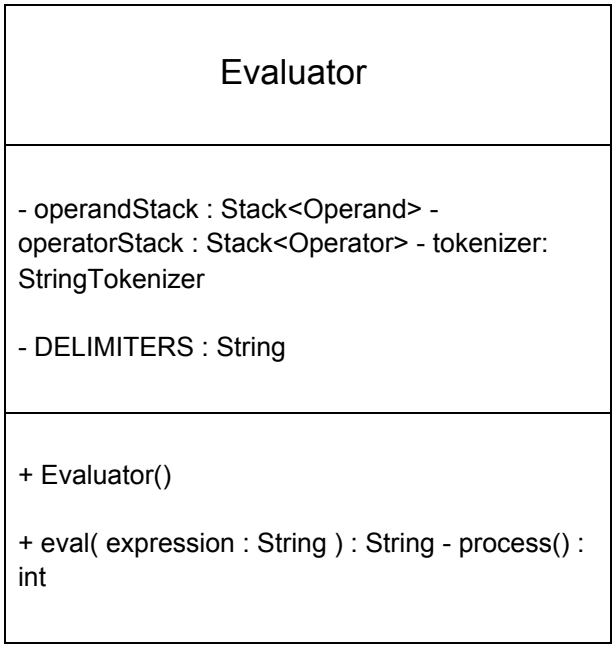
javac Evaluator.java javac EvaluatorUI.java

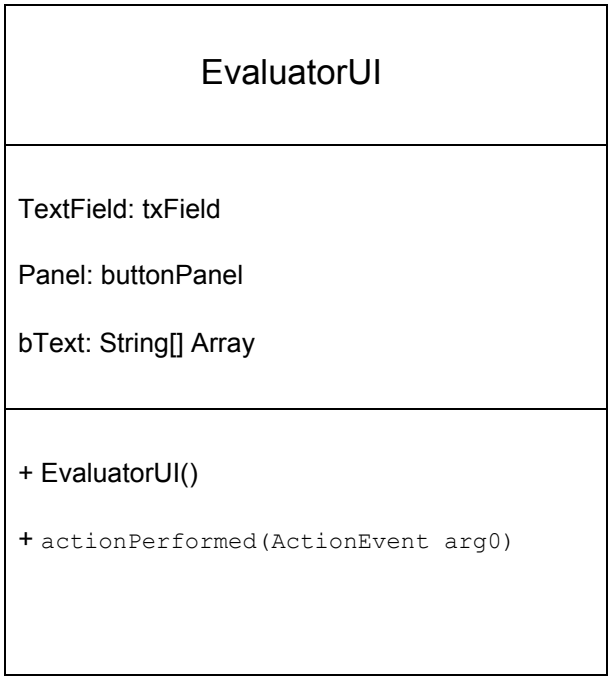
Code Organization

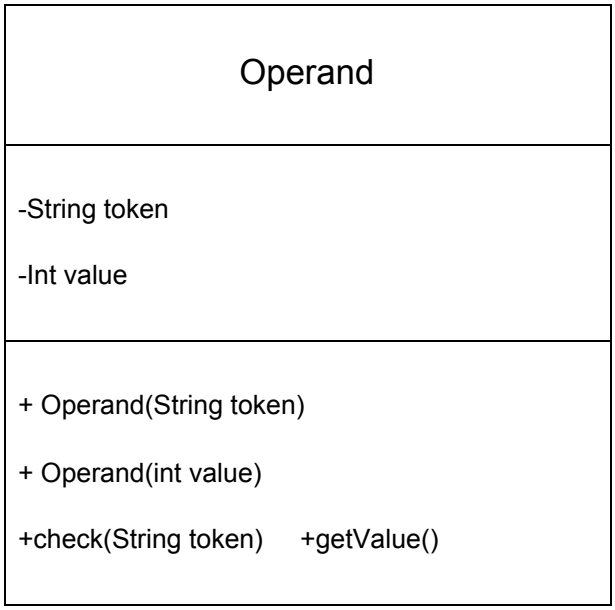
I chose to move the Operator and derived classes into a package, operators, in order to better organize the code.

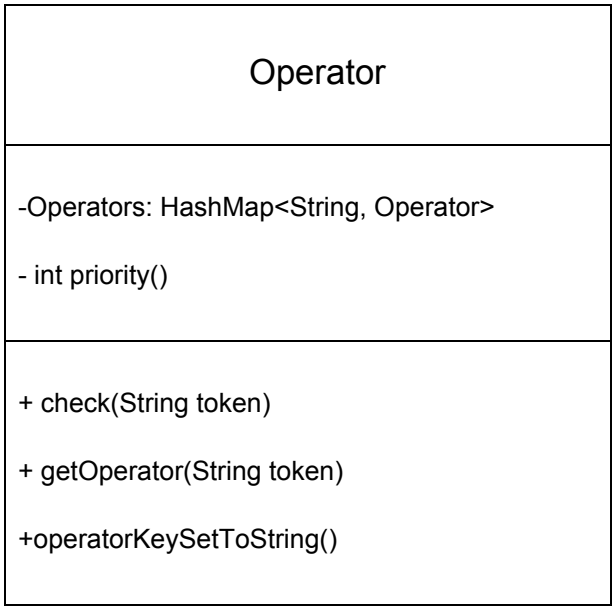
I made sure the structure of the code was correct. I removed extra spaces by using command + opt + F. I removed all the comments that I made and instead I took time to come up with meaningful names for classes, methods, variables.

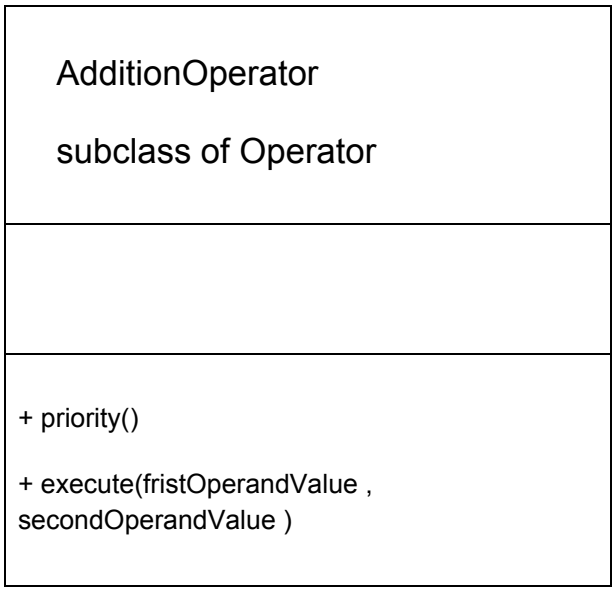
Class Diagram

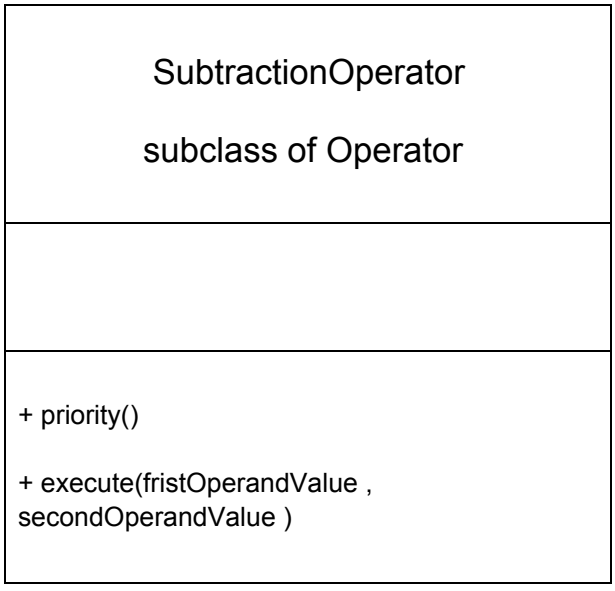


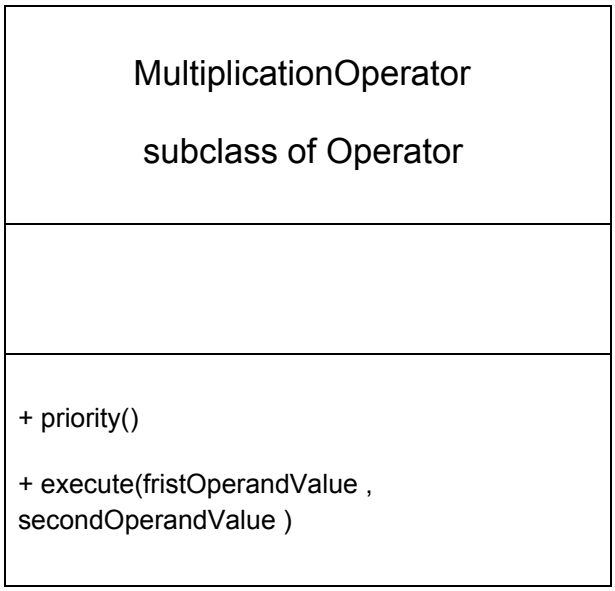


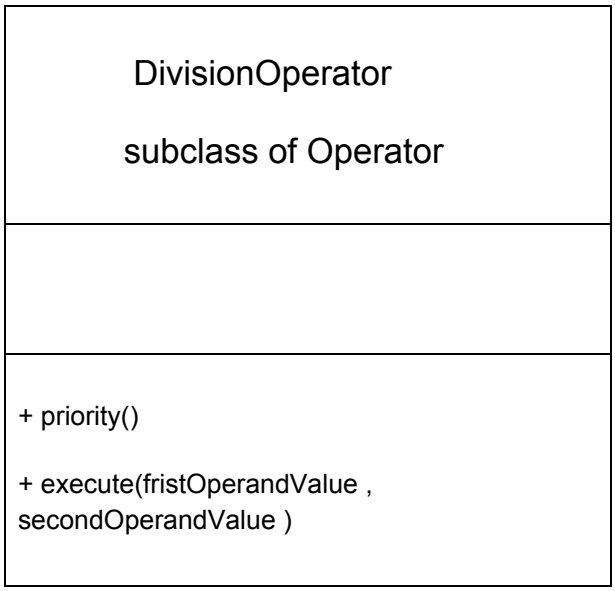

Operand
-String token
-Int value
+ Operand(String token)
+ Operand(int value)
+check(String token)
+getValue()

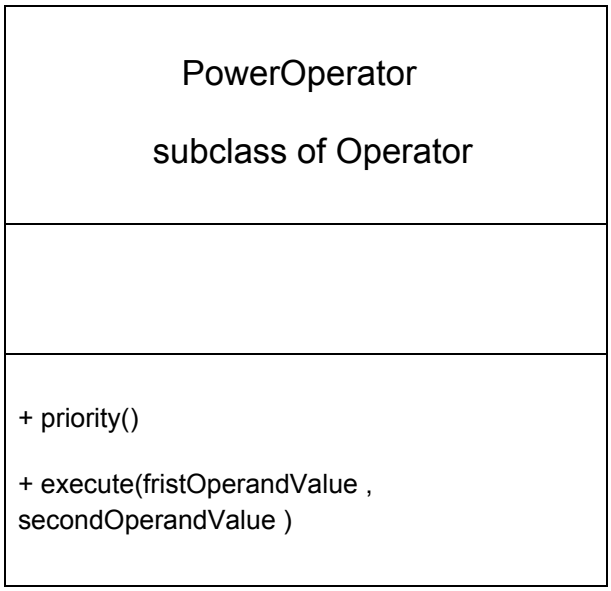

Operator
-Operators: HashMap<String, Operator>
- int priority()
+ check(String token)
+ getOperator(String token)
+operatorKeySetToString()

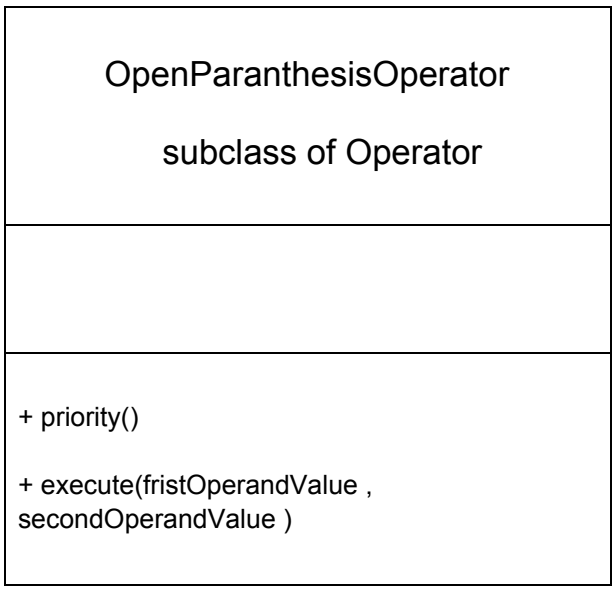

AdditionOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)

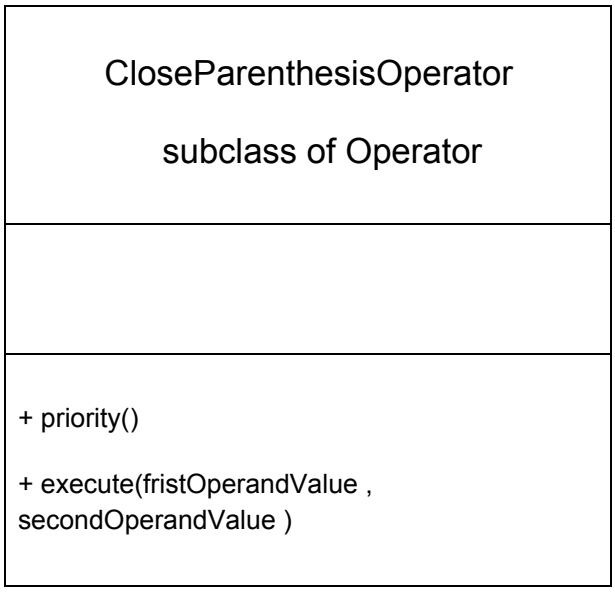

SubtractionOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)


MultiplicationOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)


DivisionOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)


PowerOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)


OpenParanthesisOperator
subclass of Operator
+ priority()
+ execute(fristOperandValue , secondOperandValue)



Results and Conclusion

At the end of this project I learned,practiced and tested my knowledge. The project taught me how to develop and implement object oriented design, Strategy pattern design.

Challenges

It was challenging for me to use my old macbook as it would glitch or stop working from time to time. At first it was challenging to follow the skeleton that we were provided. It is harder for me to follow a skeleton than to start from a scratch. The main challenge of the app was the eval method.

Future Work

The future work for this project would be to include more sophisticated math equations such as integrals, riemann sum, limits, sin, cos etc. Also apply code in EvaluatorUI to graph equations.