csc413-p1

Huy Nguyen

https://github.com/csc413-02-sp18/csc413-p1-huy9997

Calculator with GUI

This is the project contains a Calculator with a GUI written in Java. The project is complete and demonstrates a working Calculator that can be run through a Terminal/Console or GUI.

The work flow of this project is below.

Directions on how to run the code on console

- Clone the project please run "git clone https://github.com/csc413-02-sp18/csc413-p1-huy9997"

Running with Terminal

- javac EvaluatorTester.java
- java EvaluatorTester
- Enter input
- To close press control c
- You can run the GUI using the same commands listed above just replace the EvaluatorTester with EvaluatorUI.

Running the project through an IDE.

- Open the IDE then navigate to open the project.
- Make sure EvaluatorTester is running as the main method if you want to run it through console.
- Enter Input.
- If you want to run the GUI then right click the EvaluatorUI class then click run.

Pushing input with GUI or Console:

The Project starts at the top by gathering Information through the Console or the GUI. The Console contains a Scanner class where you can enter custom input as a String. The alternative would be to press custom input with the GUI. The GUI will then create a String of your input and push it into Evaluator.

Evaluating the input

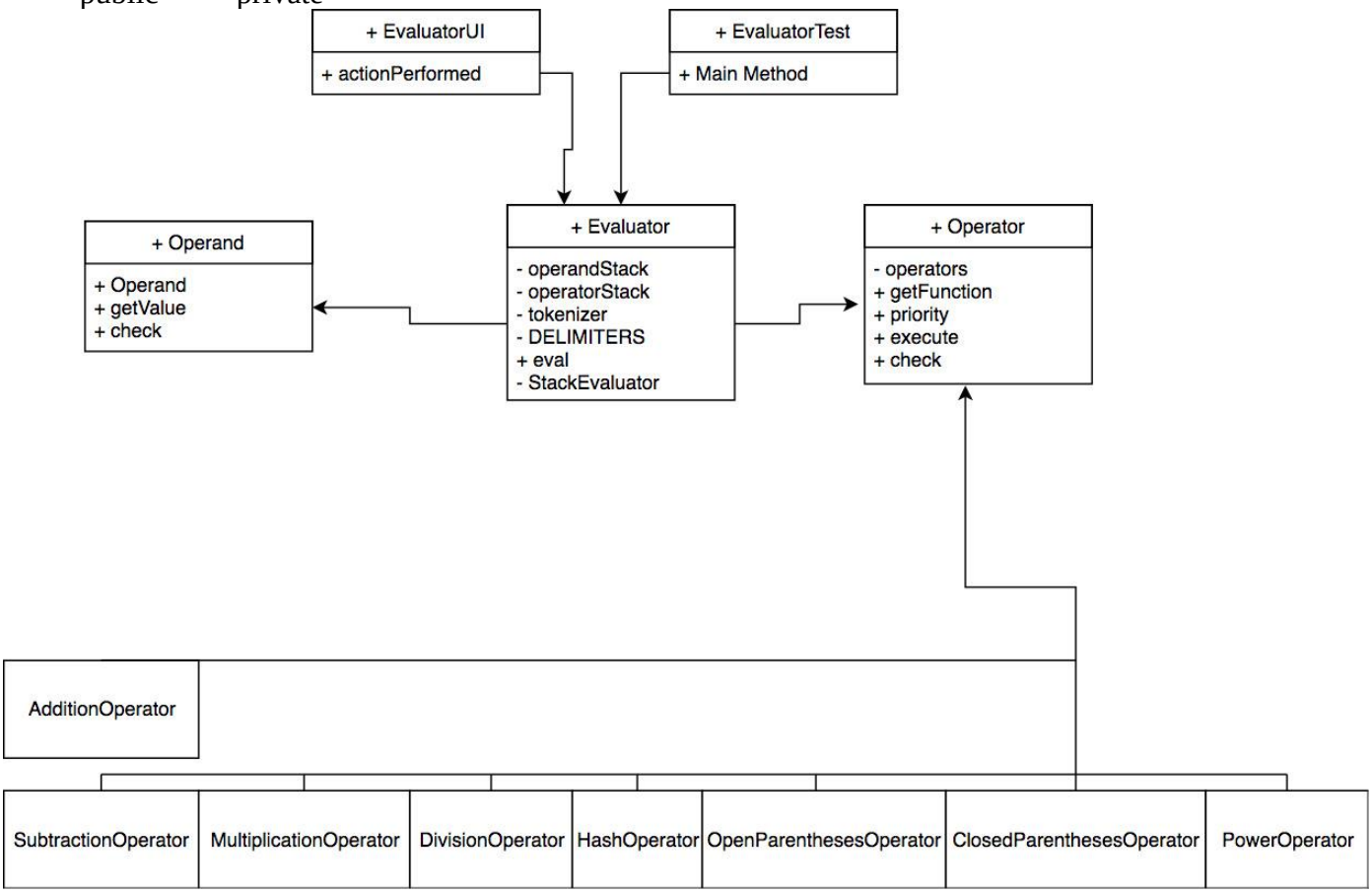
Once you have put in the input Evaluator class is class is called the String gathered from the GUI or Console will be passed in. Then the Evaluator class will then create two stacks OperandStack for the numbers and OperatorStack for all of the operators. I put a # function at the very bottom of the OperatorStack so does not call an Empty Stack Exception for trying to peek an empty OperatorStack. Then the code will continue to tokenize the entire string. I was provided some code implanted taking care of spaces and invalid tokens. If the user inputs a a+2 in will exit the code. If the a ")" is tokenized, then the class will begin to evaluate the operators by popping operator stack and the top two operands. It class will continue to do so until it reaches an "(".

During this process it will evaluate by calling the Operator.java class to see which OperatorStack which calls getFunction to get the function in the Hashmap that was called. That will call the class. An example would be if a "+" is in the OperatorStack. It would call Operator.java then call getFunction. Then getFunction will call the Hashmap to call the Addition class. The Addition class is where the actual it will be evaluated. It will take into consideration the priority going by PEMDAS. This is the same process for non "()" cases. The work flow of how this whole system works is provided in the image.

The other case of if the expression is without any "()". It will call the private function StackEvaluator to make sure that it will evaluate the code until you reach the "#" that is when it is empty. I made sure to add the "#" because I wanted to avoid the empty error. Once "#" in OperatorStack is reached than you know you have completed the stack.

Then you will get the result. You may keep entering new equations until you are done. You can see the flow chart at the top to see the Levels of the classes and the methods used in each class.

"+" = public "-" = private



Assumptions

I thought the project was a good not bad. It took some time to understand what the code given to use was doing. Then I went on to do try and do as many problems on a white board to understand the Algorithm implemented with two stacks. I thought it was going to be just like the calculator made in 220, but it was not the same. So doing the problems by hand and debugging by hand of what each step does helped a lot. I feel this was a good test to see where I was at coding wise with Java. I really enjoyed doing this project after I understood how everything worked because it was fun. I expected this project to much more difficult. I thought that I would not be able to finish with only about eight days. It was a little intimating seeing a GUI for the first time. I thought that aspect of the project was going to be much more difficult, same goes for pushing to Github. I overcame these challenges by just googling everything and watching videos.

Results

This project for the most part tested by ability to push to Github and work with the GUI. Those were the two main challenges I overcame. I just overcame them by doing research on the internet and readying up on documentation. I learned that the whiteboard is a very useful tool that can help you see where you are in your code. I do a good amount of test cases and problems on the white board tracing where I was in the code. Then I followed the code with print statements because I never used the Netbeans debugger. The print statements helped me see where the code was executing and where it was not.