Feras Alazzeh
ID: 916044212
CSC 413
Spring 2017

Documentation

Link to github repository:

https://github.com/CSC-413-SFSU-02/csc413_02_p1-FerasAlazzeh

Project Intro:

For this project, I was tasked with building a calculator. I created multiple classes to mirror different operations such as addition, subtraction, division, multiplication etc. I used a hash map to store these classes and create instances of these classes whenever I detected specific keys such as “+”, ”-“, ”/”, ”*” etc. I was able to get the calculator to work for all test cases. My calculator also handles some edge cases, such as when a user wants to input a negative number.

Command line instructions to compile and execute:

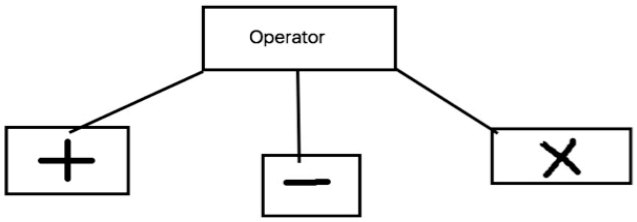
Press run in NetBeans, since it is a NetBeans project.

Assumptions:

I assumed that for every operator, I will have at least two operands. Before clarifying with you after class I did support most negative number cases however commented some of them out. However, I still do support a couple of cases such as “-6 + 5” will return -1, and 5-6 will return -1.

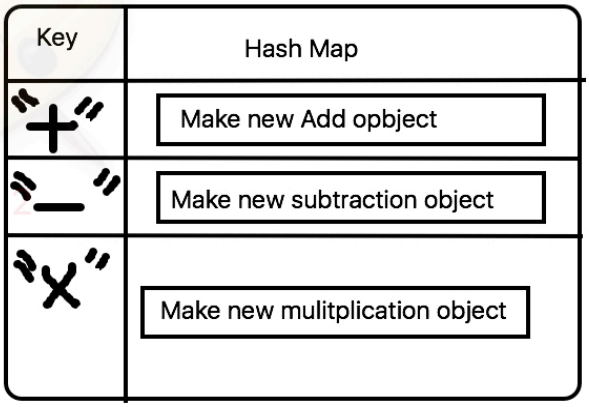
Implementation:

Every operator will have an execute class, and when called will execute specifcally to its object that we create using the hashmap to detect the token being parced through the expression
ex.
```
 Class Add{
Exectute(int x1, int x2){
  return x1+x2;
  }}
```



Operator Stack

Operand Stack



I implemented this calculator using the basic higher above extending an Operator class. I used polymorphism as explained above with the evaluator method because each operator evaluates operators differently. I used a hash map to create these operators and detect their appropriate object, ex add object, subtraction object. So we can use polymorphism to call the execute function. Than using a basic while loop I pushed operators and operands onto stacks while writing if statements to make sure we are mindful of the order of operations. Implementing parenthesis was similar. I pushed an object for an open parenthesis that was just an identifier. Than when I found a close parenthesis I would solve everything up until I work myself back to the "(".

Conclusion:

I learned that the architecture and design of the program is the most important thing. If I were to draw that diagram for myself and figure out the structure of my code, I could have finished this project faster, and could have made the code much cleaner. Actually, writing code and for loops, if statements and looking up functions is pretty basic. Anyone can do that, but having vision on building a well-structured program is a much harder and complex task.