Kevin Venegas-Mendoza

915142571

Summer 2019, CSC413

https://github.com/csc413-02-summer2019/csc413-p1-Kevinv234

**Introduction**

This software creates a calculator that can solve math equations with multiple operands within one equation. Our professor gave us 80 percent of the code and as students, we had to finish the rest. What I had to complete was the Operator.java and the Operand.java class. Once we completed that, we had to make classes for all operations that needed to be added such as Addition, Subtraction, Multiplication, Division, Power and Parenthesis operators. As soon as you make these classes, you want to finish graphical interface by finishing the UIs functionality.

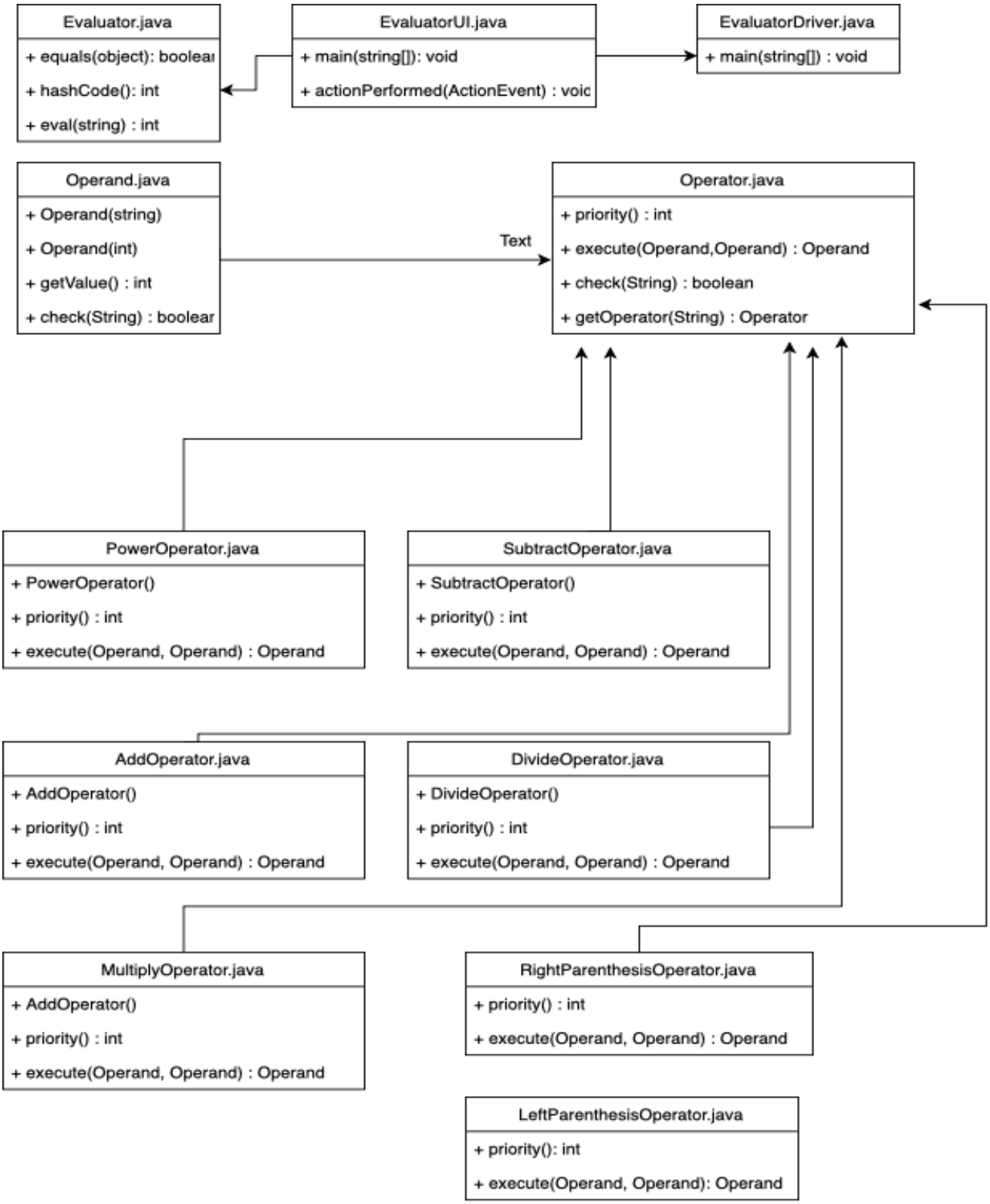
**Development Environment**

The version of java that I used is JDK SE 12.0.1, specifically for mac OSX. Once I downloaded the JDK I opened up my IDE to select the JRE. Without any JDK installed, any java program will not run. Upon importing the project, you want to select all Jar files that the professor provides, usually importing the project from GitHub does the automatically. JetBrains/IntelliJ makes really intuitive IDEs that allow for amazing customization and bench testing features.

**Build**

This project was specifically imported from a GIT repository. The file has 6 levels we must descend in order for us to reach the directory we want to be working in. You want to work in the directory level of edu.csc413.calculator where you see the evaluator and operators folder. Once you are in this directory, you are going to have three folders that are going to be crucial in reading the projects main function properly. One of the folders is for the operators, the other is for the evaluator, and the last one contains all the bench tests needed to test the functionality of the project. You can individually run the operator tests to test each function and in order for you to compile and run the entire project, you are going to want to run the main function within the evaluator.java. Another method you can use in order to test the function test cases for operand is editing the configuration and putting “auto” within the program arguments. If you want to test a specific case, you can type in a function within the program arguments. For us to test the evaluator.java you are going to want to run the EvaluatorTest.java. Once you test the evaluator, you can test the UI by running EvaluatorUI.java. With these build instructions you can run all files within the project.

## Implementation Discussion



Above is a UML map that offers a visual representation of the files and within the project. The EvaluatorUI is where the main program starts. As soon as the GUI is compiled, you have the UI running. When you start inputting numbers on the calculator, the evaluator driver class begins running. This checks that the operand and operator stack is correctly organized and there are no errors being made while data is being inputted into the stack. The evaluator is what properly solves problems when you want to evaluate the expression that are inputted. Implementing this project helped me understand the idea of abstraction and polymorphism. At first, it was difficult understanding the project because of the amount of files that were being implemented. I came from CSC340 where we only used at most 3 files. I have grown to like the idea of using more files because of the organization that it offers. I also liked the data structures being used in this file, stacks and hash maps made the organization of the data linear.

- I want to point out some of the errors, the button functionality for c and ce doesn't work properly. Only the backspace and the equal button works

**Reflection**

Learning java all over again was extremely difficult for me. I used C++ for a year after I learned Java. One thing that I am loving is the number of frameworks that Java offers and the functionality that this offers. This project was more fun than challenging, I was completely stuck for two days but once I wrapped my mind around the functionality of the program it came fast to me. Thank you for great projects Professor Souza.