CSC 413 Project Documentation

Summer 2020

Patricia Sarno

CSC 413.01

https://github.com/patricia-sarno/simple-calculator.git

Table of Contents

1 Introduction

This project uses Java and a GUI implementation to create a simple version of a calculator shown on the monitor.

1.1 Project Overview

This program takes in a simple mathematical expression such as “(4+6)*2” using one of the 5 mathematical operators: addition(+), subtraction(-), multiplication(*), division(/), and the power operator(^). We were already given an almost finished outline of the project. To complete the program, we had to finish implementing three main Java classes: Operand, Operator, and the Evaluator. The operand class essentially stores the operand given in two different ways, returns the operand as an integer and also checks to see if the operand given is an actual integer operand and not just some random letter or symbol. The operator class on the other hand, assigns each of the operators, including the open and closed parenthesis, and links them to their respective functions that we had to implement. This operator class also links the operators with the priority level when evaluating the expressions (PEMDAS). This operator class also allows us to be able to obtain the operator symbol and check if the character received is actually on the of 5 operators or a open and closed parenthesis. The Operand class and the Operator class both help the main Evaluator class carry out its’ functionality. The Evaluator takes in the actual expression and splits it into characters or “tokens”. So, it splits the operators from the operands and differentiates between two different operands and two different operators. By doing this, the Evaluator class is able to work through the expression and solve with the help of the operand and operator class. We also had implement the GUI. The GUI (Graphic User Interface), allows us to construct a visual calculator on the screen to make it look like we are actually using a calculator.

1.2 Technical Overview

As mentioned in the previous section, the three main classes that needed to be implemented were the Operand, Operator, and Evaluator classes. The operand class contains two constructors, one that takes in the operand as a string and stores it as an integer and one that takes in the operand as an integer value and stores it as an integer object. This ensures that we can use the operand value both as an integer or an integer object if we need to. Then, we have a getValue method in order for the Evaluator method to retrieve the operand as an integer. The operand class also makes sure that the token that is fed through is actually an integer using the parseInt method. The Operator class is where we match the 5 operators and the open and closed parenthesis with it’s given functionality and priority. This is all done using the HashMap and static block. For this class, we had to make subclasses for each of the operators’ functionalities. Once we declared the HashMap and created the subclasses, we assigned the keys, which are the operator symbols, to their respective subclasses within a static block. This operator class however only does the operation between two operands so it will do the operation between two operands with the highest priority. The evaluator class evaluates the string expression by splitting the expression into tokens and placing each token into an operator stack and an operand stack depending on what it is. Depending on the open and closed parenthesis and their priorities, we use the stack to find the results of two operands. We also had to implement the GUI, which was already mostly done for us. The only part we had to implement was the actionPerformed method which implements the equal operator, clear operator, and the CE operator.

1.3 Summary of Work Completed

I was able to get the program working by completing the Operand class, the Operator class, the Evaluator class and the EvaluatorUI class. In the Operand class, I finished implementing the two constructors, and the getValue and check methods. In the Operator class, I declared the HashMap and finished creating the static block that puts the keys with their assigned subclasses into the HashMap. Within the Operator class, I also made the subclasses: AddOperator, DivideOperator, LeftParenthesis, MultiplyOperator, PowerOperator, RightParenthesis, and SubtractOperator and assigned each one of them their priority and the functions they carry out. In the Evaluator class, I finished implementing the evaluateExpression method. In the EvaluatorUI class, I only had to implement the actionPerformed method that enables the equal and clear and CE buttons' functionality.

2 Development Environment

2.1 Version of Java Used

The java version that was used to implement this project is "13.0.02".

2.2 IDE Used

The IDE used to implement this project is IntelliJ IDEA Ultimate version 2020.1.2.

3 How to Build/Import your Project

To build/import my project, you would copy the repo link using either SSH or HTTPS.

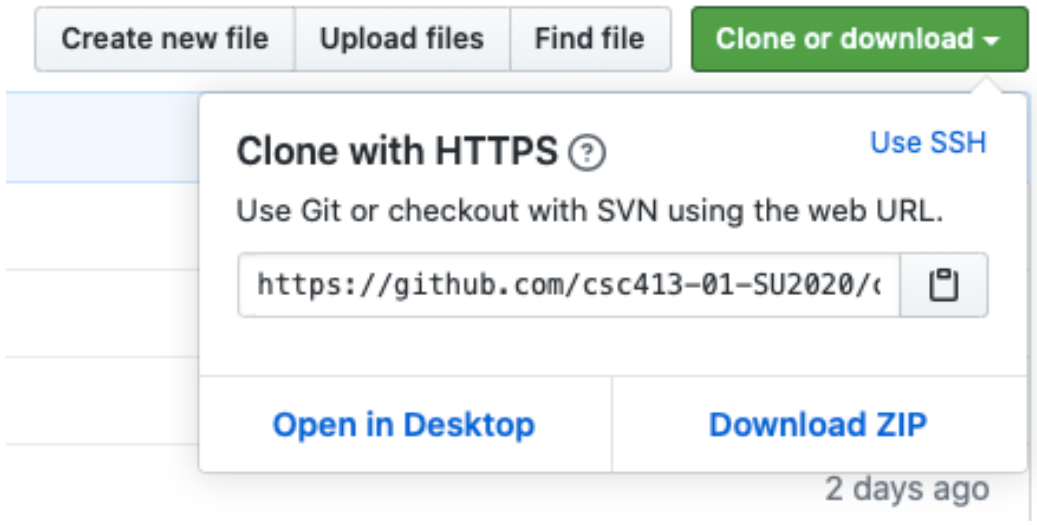
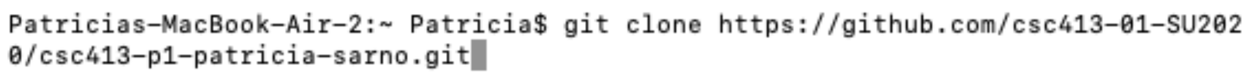


Figure 1, Repo Link Used to Clone.

Then, go on the terminal, navigate to the directory you want the project to be in and use the git clone function:

```
% git clone <link>
```



```
Patricias-MacBook-Air-2:~ Patricia$ git clone https://github.com/csc413-01-SU202
0/csc413-p1-patricia-sarno.git
```

Figure 2, How to Clone in the Terminal.

The project should appear on the directory of your choice. Then you open your IDE of your choice open the project file to import it. For this, make sure you click the calculator folder and not the csc413-p1-patricia-sarno project folder.

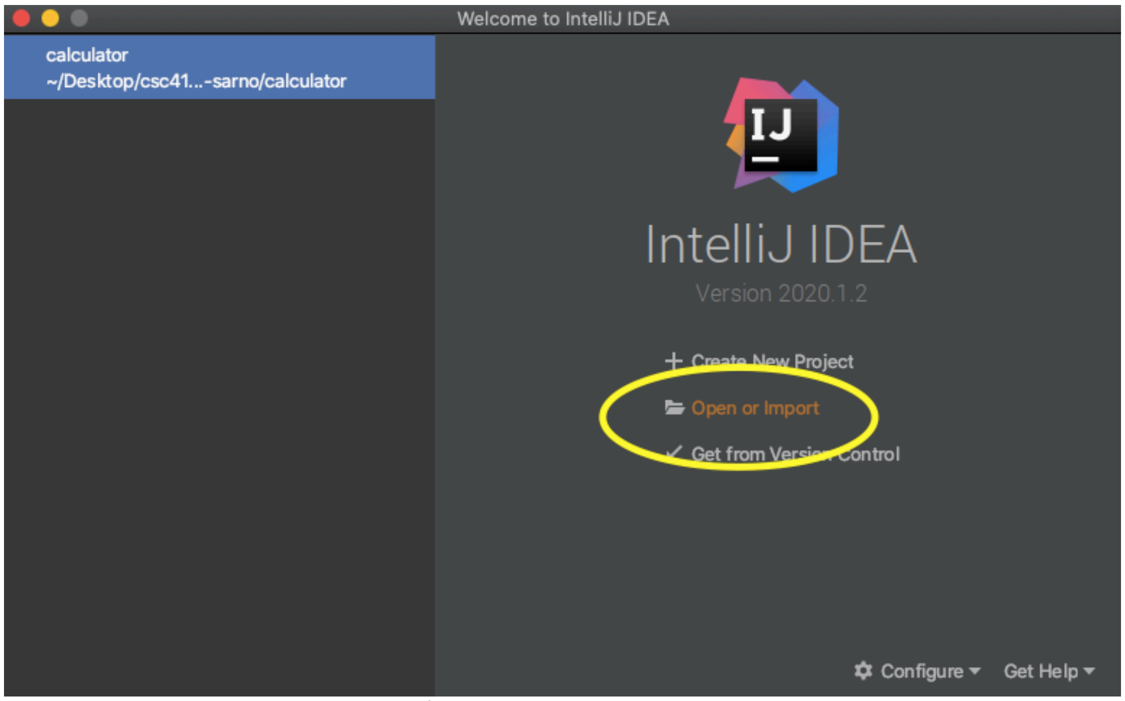


Figure 3, Importing/Opening a New Project on IntelliJ IDEA.

4 How to Run your Project

Once I finished my code and all the tests have passed, I can run my calculator program on the IDE by going to the EvaluatorDriver file and run the main function. To run the program and interact with the program through a graphical user interface (GUI), I would have to run the EvaluatorUI file instead.

5 Assumption Made

To make the project simpler, one assumption I had to make was to assume that this project only takes in positive integers.

6 Implementation Discussion

When I first cloned the project to my local repository, I was overwhelmed with the amount of code that was already given. It was hard to read and I didn't know where to start. Thankfully, the videos given were able to help and led me to start on the Operand class first because that is what the video said. The video helped explain what we had to do and how to implement the classes and already gave the check method to us. By giving us the parseInt function to turn a string into an integer in this check method, it gave me an idea to just use that for the constructor that took in the string token. By doing this I wouldn't have to convert anything into an ASCII decimal number which was what I was going to do because I saw that the video suggested that. I also needed to something to store the integer or integer object in the two constructors so I had to make a private integer data type outside of the constructor that can be used by both the constructors. Then, I would also use this data type to return the integer value in the getValue method. I was really confused because I tried to run the test for the Operand, and it wouldn't build and so I thought I was doing something wrong. But I just had to complete the other Operand and Evaluator classes first.

So, I moved on to the Operator class and I had to take this time to review on HashMaps and dig deeper into what the static block does. But, it was easy to follow what the video was saying and it gave us some code to declare the map, start the static block and create the subclasses. So, I just went ahead and created all the subclasses first and finished the static block. It wasn't until I started working on the Evaluator class that I realized that I had to add the parenthesis too. Then the rest of it was easy where we had to assign the operator to the given priority and implement the function of the operators when given two operands. What I did get confused with was the priority for the parenthesis because it wasn't given. For this, I had to ask a friend and they explained that the left/open parenthesis has a priority of 0 and the right/closed parenthesis has a priority of 4, which made sense because if an expression did have parenthesis, you would start on the expression with the parenthesis first and that is indicated by the left parenthesis. The right parenthesis would be last because that is what closes the expression and we would have to do the expression within the parenthesis first. In order to do the checks and the getOperator, I had to look up the HashMap API because the video mentioned to look more into the HashMap functions. This saved me a lot of time because I probably would have tried to write lines of code for something that I did with just one line of code looking at the HashMap API.

Next, I moved onto the Evaluator class. This one was the most confusing because I had no idea what I had to do and why the code already implemented into the class was doing there. I do remember doing an infix problem in my CSC 220 class using the stack data structure so I had to go look up the notes I had on that. It took a while to get some kind of pseudocode to work with and guide me with writing the code. It get's very confusing and I had to take into account the parenthesis which was really hard to figure out but my notes from my data structures class helped out a lot. And then I got really confused because when I finished my code and moved on to the already implemented code that was there, I realized that it was part of the same code that I already just wrote, so I just commented it out.

Then, I moved on the to the GUI implementation of this project which was so much harder and confusing because I had never worked with Java GUI before. All the functions were new to me and I did not understand what was going on. However, most of the code was already written and all I had to implement was the actionPerformed method. To evaluate the expression that is typed on the calculator, I needed to be able to call the evaluator class and execute the evaluateExpression function, so I created an Evaluator object. The easiest way I thought I could do this part was to use an if else statement where if a button is pressed, some function would be carried out. For example, if the equals ("=") button was pressed, the evaluator object would be used to call the evaluateExpression function method to evaluate the expression. However, the only thing that I messed up on was the CE button which is actually just a backspace button now.

6.1 Class Diagram

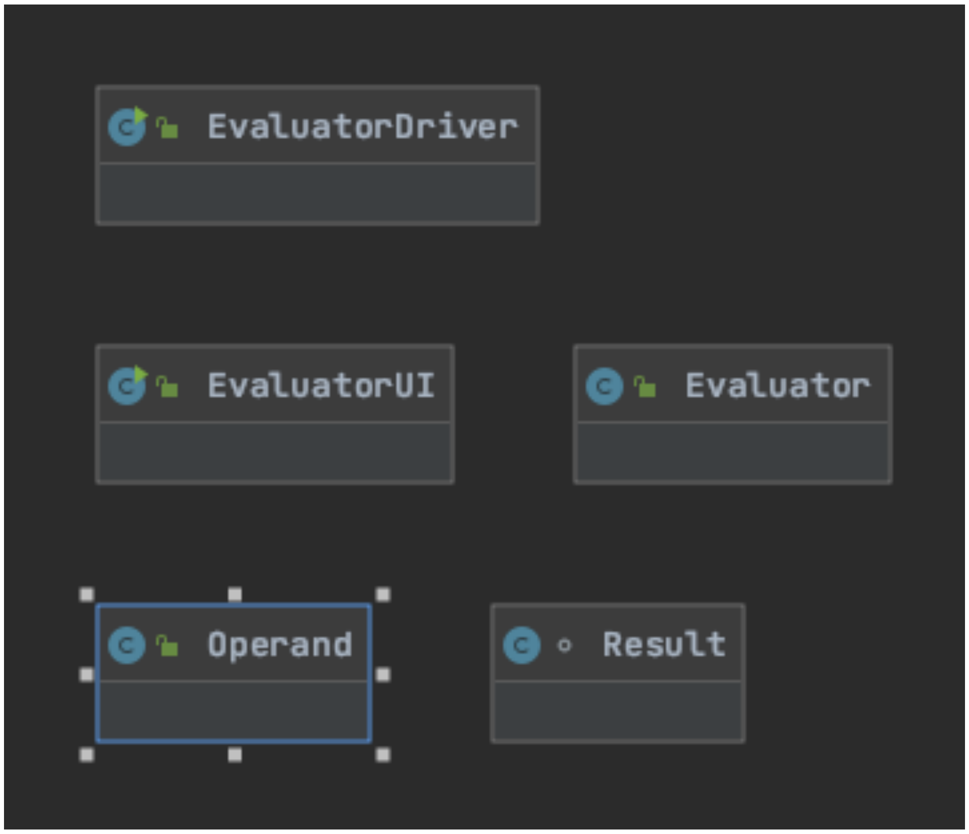


Figure 4, Class Diagram of Evaluator.

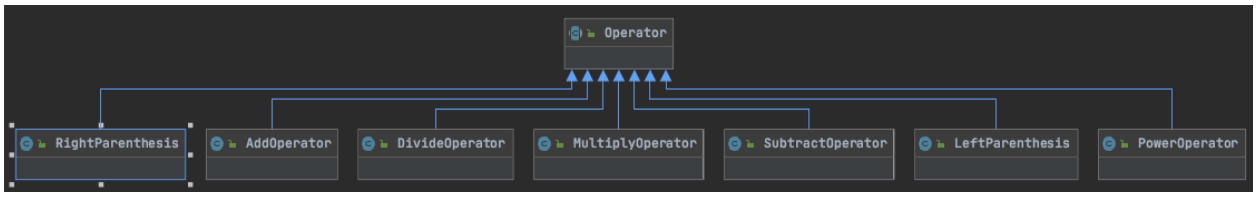


Figure 5, Class Diagram of Operator.

7 Project Reflection

This project was fun to work on. It was a nice review on my Java skills and it was simple and easy to understand. I'm glad that there were video's to help direct us where to start. When I first got the outlined code, it was hard to read it and understand what was going on and I think that is something I need to work on which is being able to understand code that is given to me and be able to work on top of that. This project also introduced me to working with GUI. I have worked with other graphical User Interfaces before but not in Java. It was very difficult, and this is something that I need to learn.

8 Project Conclusion/Results

I finished implementing all of the four main classes which were the Operand, Operator, Evaluator and the EvaluatorUI classes. When I ran all the tests, they all passed. This project was simple and helped review important concepts of Java. This is also a good assignment to let us know what we need to learn and catch up on, such as the GUI.

9 Sources

https://www.javatpoint.com/java-string-to-int
https://stackoverflow.com/questions/49904830/create-a-new-integer-object-that-holds-the-value-1
https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html#containsKey-java.lang.Object-
https://docs.oracle.com/javase/7/docs/api/java/awt/event/ActionEvent.html