Java Calculator Project Documentation

Spring 2021

Joshua Stone

Table of Contents

1 Introduction

1.1 Project Overview

For this project we are tasked with creating a fully functioning calculator using the Java programming language. Our calculator should provide all the functionality of a regular calculator, be aesthetically pleasing, and come up with the correct results when called upon by the user.

1.2 Technical Overview

For this project we developed a calculator with the help of OOP principles such as abstraction and inheritance. We used the hashmap data structure to hold our operators and we used a stack to process our operands.

1.3 Summary of Work Completed

The work I completed for this project consisted of completing the Evaluator, EvaluatorUI, Operand, and Operator classes. I also created various derived classes from the abstract Operator class. These derived classes include the AddOperator, CloseOperator, DivideOperator, MultiplyOperator, OpenOperator, PowerOperator, and SubstractOperator. My calculator program was also required to pass all unit tests and function within the EvaluatorDriver – which it did admirably!

2 Development Environment

Java version 8 used

Intellij IDEA 2020.3.2 Ultimate x64 IDE Used

3 How to Build/Import your Project

To import the project please use the following command in your terminal:

git clone https://github.com/csc413-sp21/csc413-p1-JoshLikesToCode.git

Once finished with this, launch your IDE and look for an “Import Project” function. This will prompt you to provide a source root folder for your project, you should select the folder named “calculator” as this source root folder. After this, click the “Create Project from existing resources” or similarly named button. Click next, leaving all fields as default, until you are prompted to select a Java JDK, if one is installed on the machine. If one is not installed, please install the Java JDK and follow all previous steps up until this point. If one is installed and you do not see a Java JDK selected, click the “+” symbol near the top of the UI and find where your Java JDK folder is installed and select it. Once finished with this step, proceed to click next until the project has successfully been imported.

4 How to Run your Project

Once the project has been cloned and successfully imported, you can then navigate to your IDE's "Build Project" dropdown menu pane and select the "Build" button. Once the IDE builds the project, you can then find the "Run" dropdown menu pane and select the "Run" button. This will launch the project and you should see a fully functioning calculator displayed.

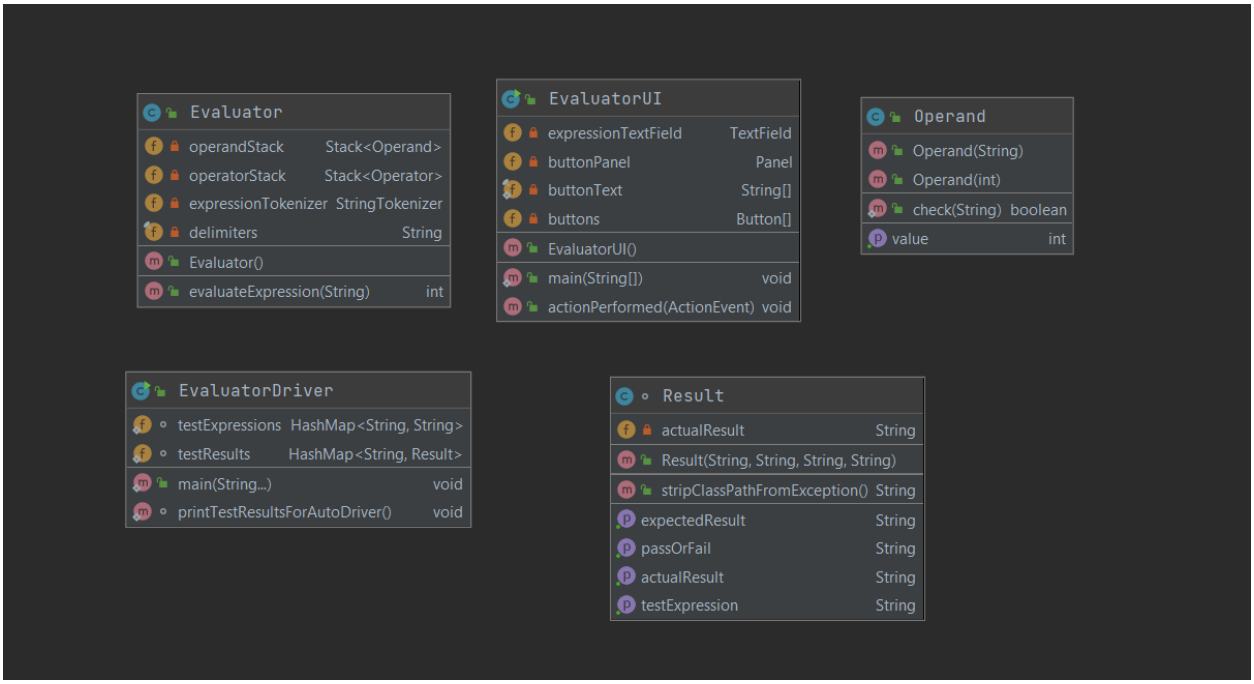
5 Assumption Made

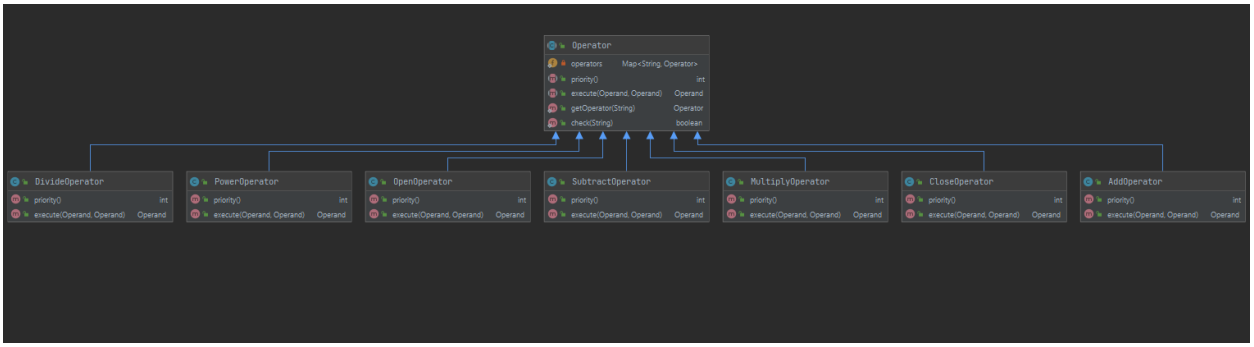
Some assumptions made when working on this project are the following: I assumed the unit tests and evaluator driver were working correctly (I changed the -17). I also assumed all code in the project that was not marked as TODO or was not left blank was functioning correctly.

6 Implementation Discussion

For the implementation of this project, I relied heavily on OOP principles. I tried to keep data members and data structures that weren't needed outside of their respective classes as private and provided the necessary public getter functions to make use of them if called upon. The Operator class is abstract and all other operators extend from it, this is because the AddOperator, SubtractOperators, etc have an **IS-A** relationship with the Operator class.

6.1 Class Diagram





7 Project Reflection

I had a lot of fun working with this project. In particular, it was challenging to bring myself back into OOP programming and paradigms, as I've been spending a lot of time with C lately. I will be the first to admit that my background with Java isn't that extensive, but luckily a lot of the same principles that apply to Java apply to C++ which is a programming language I am familiar with because of using it in community college.

The unit testing was another thing that was particularly new to me, but also very helpful. I'm used to creating driver classes, but I've never seen tests as extensive as the ones we used on this project. I can see how tests such as these can become very helpful to an aspiring software engineer such as myself.

Some challenges I ran into with this project were in figuring out how to retrieve data from our classes data structures. For example, at first I wanted to use an iterator in the Operator class' getOperator() function in order to retrieve the correct operator from the hashmap. That didn't exactly pan out and I figured I was simply over-thinking things, and it turns out, I was!

When I looked at some of hashmap's functions the get() function clearly stood out to me. I realized that I could simply use the token that's given to getOperator() as an argument and then feed that token into hashmap's get() function and it would then return the value that I was looking for in the hashmap.

Another challenge came with the figuring out the UI. I had difficulty at first figuring out how to display numbers after they were clicked or how to call their respective operators. My breakthrough with his came after I discovered the values given by actionEventObject.getAction() and expressionTextField.getText(). With this I was able to craft a switch that handled all the buttons and the entry of numbers into our textfield.

8 Project Conclusion/Results

In conclusion this project brought me up to speed with OOP design and gave me a glimpse of the power behind Java's JFrame class. After looking into JFrame a bit, it's safe to say that I haven't really dealt with anything like it in C or C++. In this regard, Java reminds me a lot like Swift with Xcode in its ability of being able to provide a fully functioning user interface along with the program.

As for the projects results, I was able to create a working calculator. Some things I noticed with my calculator is that leading zero's would be displayed quite a bit unless I setText("") which sometimes doesn't give the desired result when using the calculator's C function. Also, when the integer value in the calculator's expressionTextField.getText() overflows it does not display an error, which is something I tried working on but ultimately was not able to fix.