Karma Namgyal Ghale
921425775
February 16, 2022

Assignment 1 Documentation

Github Repository

https://github.com/sfsu-csc-413-fall-2022-roberts/assignment-1---calculator-Kng113

Project Introduction and Overview

The evaluator project required me to implement an infix expression evaluator given a code skeleton, and to integrate that evaluator into a partially implemented calculator user interface. The operators are assigned as per below precedence/priority. Implemented Initoperator to handle null and initialize before parsing the tokens. The program followed all the requirements and algorithms provided in the assignment.

Operator	**Priority**
AdditionOperator, SubtractionOperator	2

MultiplicationOperator, DivisionOperator	3
PowerOperator	4
LeftParenthesisOperator, RightParenthesisOperator	1
InitOperator	-1

Scope of Work

Task		**Completed**
Implement the eval method of the Evaluator class		X
Test the implementation with expressions that test all possible cases. The following expressions were used:		X
	1+(2*3/4^5)+2/5	X
	1+2	X
	1+1-(5+5*(2/2))	X
Implement methods in the abstract Operator class		X
	boolean check(String token)	X
	abstract int priority()	X
	abstract Operand execute(Operand operandOne, Operand operandTwo)	X
	Lookup mechanism for operators to prevent instantiation of the same operator more than once	X
Implement Operator subclasses		X
	Properly organize subclasses (I used a package named operators)	X
	Implement subclasses for the required operands: multiplication, division, addition, subtraction, exponentiation, parentheses)	X
Implement Operand class		X
	Package named operand is used.	

	Constructors (String and int parameters)	X
Task		**Completed**
	boolean check(String token)	X
	int getValue()	X
Complete implementation of the Calculator UI in the EvaluatorUI class		X
	Use the previously implemented Evaluator	X
	Implement the actionPerformed method to handle button presses	X

Execution and Development Environment

I used the IntelliJ IDE on my windows to complete this assignment, and tested in both the IDE and shell.

Compilation Result

Using the instructions provided in the assignment one specification:

```
> javac Evaluator.java
> java Evaluator
> javac EvaluatorUI.java
> java EvaluatorUI
```

No error messages or warnings were displayed, and the application ran as expected.

Assumptions

I assumed that all expressions provided to the Evaluator's eval method would be correct, so did not implement any error handling for invalid tokens.

Implementation

Evaluator

The Evaluator class provided us with a skeleton of an eval method that must be implemented according to the algorithm discussed in class, and linked from the specification. As I implemented this algorithm, I noted a few cases of duplicated code. Once I had the general algorithm working, I refactored this code into the process method, as the act of processing operators and operands from their respective stack was required in multiple places in the eval method.

Operator and the Strategy Software Design Pattern

Implementation for this project required us to understand and implement the Strategy pattern as described in class. The Strategy pattern allows a programmer to specify a family of algorithms as an abstract class, and to provide concrete implementation of that algorithm that may be used to vary behavior at runtime.

The Operator class provided the abstract base class for our implementation of this pattern, specifying an abstract method, execute, that would be implemented in derived classes to provide the varying behavior for each of the operators. The concrete classes that were created were: AdditionOperator, SubtractionOperator, MultiplicationOperator, DivisionOperator, ExponentiationOperator, and OpenParenthesesOperator.

Operand

The Operand class was fairly straightforward to implement, but did require some attention to the fact that an Operand could be instantiated with a String or an Integer. The constructor with String would just convert token to int and assign to the value and I have implemented the try and catch statement in the check method to validated if the token is an integer or not.

EvaluatorUI

The skeleton provided for the calculator UI required us to implement the actionPerformed method from the ActionListener interface. I used the sample code provided in class as a framework for starting my solution, but noticed that a lot of the logic was duplicated in a lengthy if/else block. I decided to refactor this code to a more concise form, removing the duplication in the example by simply inspecting the String value of the button pressed, and only explicitly creating actions for the special buttons “=“, “C”, and “CE”. The rest of the buttons simply appended text to the text field in order to build the expression.

The C and CE buttons were both implemented to clear the text field, as instructed in class.

The = button was responsible for taking the expression String from the text field, and evaluating that expression using the Evaluator class. After evaluating the expression, the result was coerced to a String, and placed back in the text field so that the user could continue building an expression from the previous result.

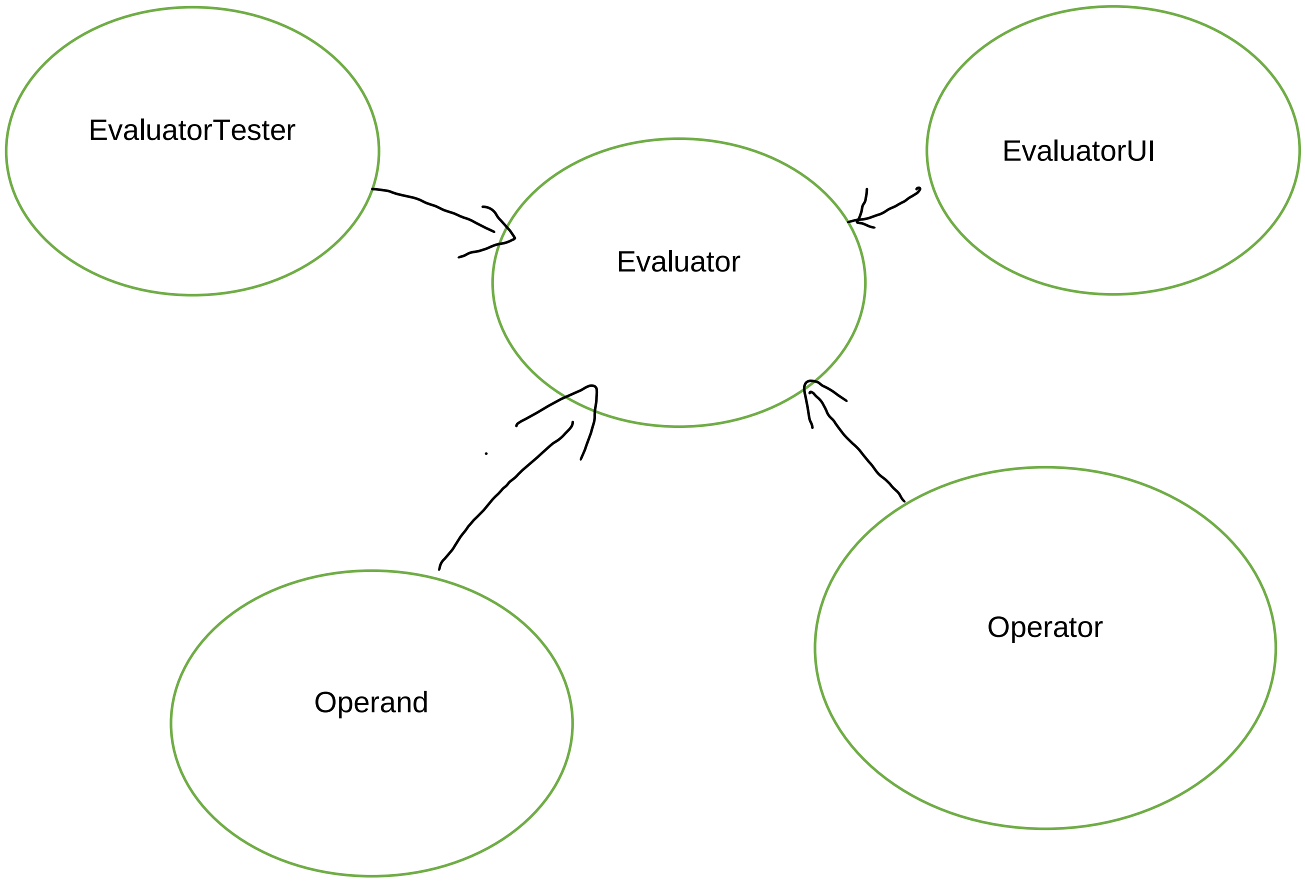
Code Organization

I chose to move the Operator and derived classes and Operand into new packages, operators and operands, in order to better organize the code.

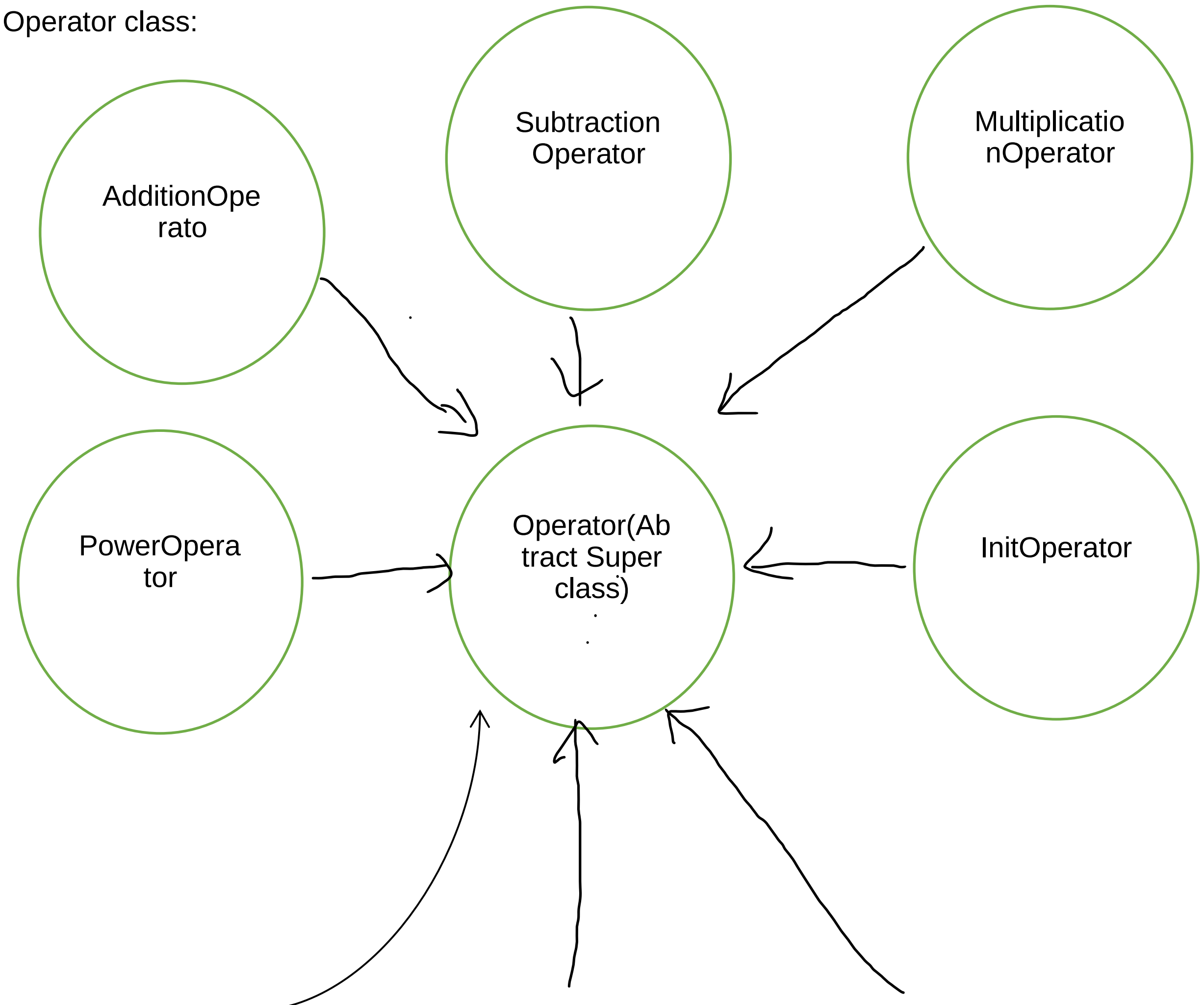
Class Diagram

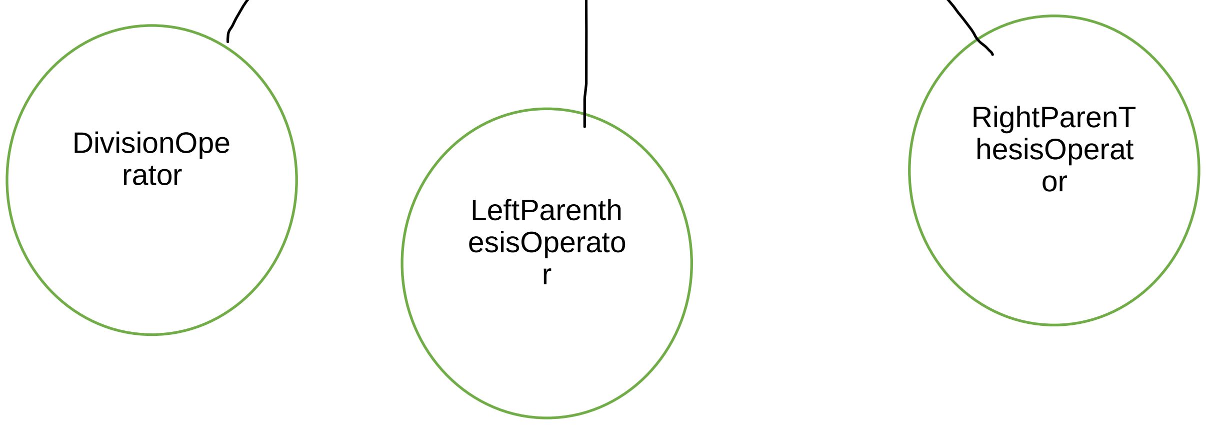
The following class diagram shows the details of all of the classes in this project, including the inheritance hierarchy:

Classes-



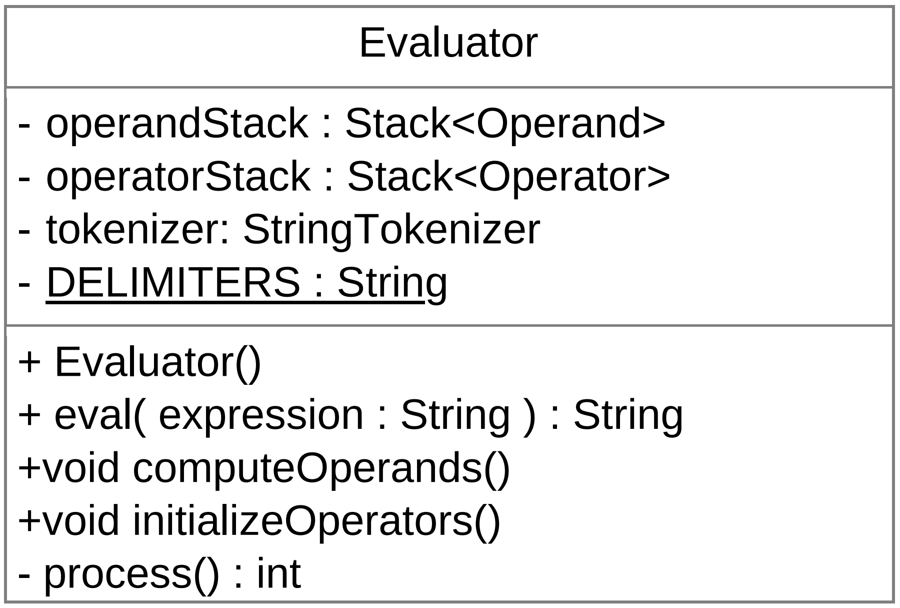
Operator class:





Results and Conclusion

The program was good exercise to learn more about hashmap data structure and OOP features. The implementation evaluates the mathematical expression within the scope of the requirements. Implementation has been tested on terminal as well as the java based graphical user interface. The assumptions and limitations of this implementation have been described in this document.


Evaluator
- operandStack : Stack<Operand>
- operatorStack : Stack<Operator>
- tokenizer: StringTokenizer
- DELIMITERS : String
+ Evaluator()
+ eval(expression : String) : String
+void computeOperands()
+void initializeOperators()
- process() : int

Challenges

The eval method was especially challenging for me to implement; it seemed like every change I made to the algorithm during development was having unintended side effects. I had trouble maintaining the operators HashMap into operatorStack. I tackled the problems consulting with friends and professor. Also, running the program through the debugger helped to see what were being pushed in operator and operand stacks.

Future Work

I think some interesting future work could be to expand the functionality of the EvaluatorUI class in order to implement all of the buttons present, and consider operation of scientific notation like sine, cosine and such. Work for better and simpler approaches to tackle the problem like duplication and repeated lines of same blocks of code.