CSC 413 Project Documentation

Fall 2018

Name: Kilan Rai

Student ID: 916002781

Class: CSC413-01

GitHub Repository Link:

<https://github.com/csc413-01-fa18/csc413-p1-kilanrai>

Table of Contents

[1 Introduction 3](#_Toc522827688)

[1.1 Project Overview 3](#_Toc522827689)

[1.2 Technical Overview 3](#_Toc522827690)

[1.3 Summary of Work Completed 3](#_Toc522827691)

[2 Development Environment 3](#_Toc522827692)

[3 How to Build/Import your Project 3](#_Toc522827693)

[4 How to Run your Project 3](#_Toc522827694)

[5 Assumption Made 3](#_Toc522827695)

[6 Implementation Discussion 3](#_Toc522827696)

[6.1 Class Diagram 3](#_Toc522827697)

[7 Project Reflection 3](#_Toc522827698)

[8 Project Conclusion/Results 3](#_Toc522827699)

Introduction: This project is about Assignment 1- Expression Evaluator and Calculator GUI in our class CSC413-01 in java language. In this project, I was provided almost complete version with four java source files, i.e. Evaluator.java, Operator.java, EvaluatorUI.java, and Operand.java. This project is created in java folder with “edu.csc413.calculator” as a main package. Within this main package, there are two package one is one is evaluator and other is operators. In this project, input is any numbers as operands and the set {+,=\*,/^(,)} as operators. When these inputs are entered, the mathematical expression will be evaluated. As result, we can see the result. For example, if we enter 2+3\*4 as input, we see on the display, “Result 14”. This can happen when we run EvaluatorDriver, but if we run EvaluatorUI, we can see on the console a 4 by 5 a mini calculator, where we can perform simple mathematical operation. In the mini calculator or GUI, all operators defined in this project and number from 0 to 9 are displayed in the 4 by 5 grid, including “ =, C, CE” symbols. The symbol = evaluates the expression, whereas C and CE clear the calculator.

Project Overviews: The purpose of this project is to practice object-oriented design to create two programs: (1) An object that evaluates mathematical expressions and (2) A GUI around the artifact from 1.

Technical Overview: When run EvaluatorDriver, ask Enter an Expression, i.e., a mathematical expresson. And when run EvaluatorUI, it will display a 4 by 5 mini calculator on the interface.

Summary of Work Completed: In Operand.java, I declared value as private int inside

public class Operanc{ }, converted string to integer and assigned to value inside the constructor of Operand, set value and return value. In Evaluator.java file, I made sure the public int eval(String expresson) function checks the left and right parenthesis and continue to evaluate the mathematical expression until it encounters the left open parenthesis. I also created a method named, operandStack.push(op.execute(v1,v2)) inside the while loop, that takes an operator as arguments and executes the while loop for completing the evaluation. In the Operator.java file, I declared private HashMap as follows and initialized the HashMap, operators

private static HashMap<String, Operator> operators. I added AddOperator.java, DivideOperator.java, etc. in this Operator.java source file. And in EvaluatorUI.java file, I set C, and CE to clear the mini calculator and = to evaluate the expression.

Development Environment: I used IntelliJ IDEA 2018.2.3x64

How to Build/import your project: on IntelliJ configure: click cvs, then select: checkout version control, then click GIT, then enter <https://github.com/csc413-01-fa18/csc413-p1-kilanrai>

In the URL for GIT clone repository or for cloning. Then Test, and Log in to GitHub account by entering user name in GitHub and password. Then it creates a Directory : C\Users\Kilan\csc413-p1-kilanrai

How to Run your project: on IntelliJ configure, there is a tab or select button where we can select either EvaluatorDriver, which prompts user to input an expression, or EvaluatorUI, which display a mini calculator for input by clicking, or All in calculator\_test which shows all test cases if it passes or fails.

Assumption Made: Evaluator class uses two utilities classes, I.e., Operand and Operator. The Evaluator class implements a single public method, public int eval(String expression) which takes a single String parameter that represents an infix mathematical expression, then parses and evaluates the expression, and returns the integer as result.

Implementation Discussion: The main implementation was in the Evaluator.java file. In this file, I added else if statements for checking the left parentheses and a while loop inside the if else statement to check the open left parenthesis. It will continue to evaluate the operators until it encounters the left open parentheses.

else if (toen.equals(“)”){

while(!(operatorStack.peek() instanceof LeftParenOperator)){

Operator oldOpr = operatorStack.pop();

Operand op2 = operand.Stack.pop();

Operand op1 = operandStack.pop();

operandStack.push(oldOpr.execute(op1,op2));

}

operatorStack.pop(); // get rid of left parentheses

}

Then after “Operator newOperator = Operator.getOpeator(token);” line I added

“!operatorStack.isEmpty()” within while loop condition to make sure stack is not empty.

Last thing I did in Evaluator.java was as follows:

while(!operatorStack.isEmpty()){

Operator op = operatorStack.pop();

Operand v2 = operandStack.pop();

Operand v1 = operandStack.pop();

operandStack.push(op.execute(v1,v2)); // takes an operator as an argument and execute them

}

Return operandStack.peep().getValue();

Next, I did a little bit more implementation in Operator.java. First, I declared private HashMap as follows:

private static HashMap<String, Operator> operators; Then I initialized the HashMap, operators inside the static:

static{

*operators* = new HashMap<>();  
*operators*.put("+", new AddOperator());  
*operators*.put("-", new SubtractOperator());  
*operators*.put("\*", new MultiplyOperator());  
*operators*.put("/", new DivideOperator());  
*operators*.put("^", new PowerOperator());  
*operators*.put("(", new LeftParenOperator());  
*operators*.put(")", new RightParenOperator());

}

I also added seven source file in the operator packages, which are AddOperator.java, SubtracOperator.java, MultiplyOperator.java, DivideOperator.java, etc..

And in the Operand.java file, I just declared a private int value, then I initialized in the constructor by converting the string to integer. In the EvaluatorUI.java, I set C and CE for clearing the calculator and = symbol for evaluating the expression.

Class Diagram: I don’t know how to make diagram so far. I tried but I could not figure out in this project. In the future, I will try my best. But in this assignment, I used as numbers as hierarchical class order.

1. Edu.csc413.calculator
2. evaluator:

Evaluator.java

EvaluatorDriver.java

EvaluatorUI.java

Operand.java

1. operators

AddOperator.java

DivideOperator.java

LeftParenOperator.java

MultiplyOperator.java

Operator.java

PowerOperator.java

RightParenOperator.java

SubtracOperator.java

Project Reflection: I feel a little bit overwhelming in this project since I have never prepared any documentation for any project so far. I feel a little bit nervous, too because I am not 100% sure what to write in the documentation in each table of contents. However, I tried my best as I understand the project how I implemented.

Project Conclusion: It was a kind of reviews for infix or Mathematical expression how to evaluate in the java which I think we did in csc220 a little bit. As result I feel little bit familiar with the implementation. But GUI part was little bit confusing initially for me. Somehow I was able to implement with lecture note.