***CSC 413 Project Documentation Summer 2021***

***Tamir Rasheed 918537300***

***CSC413.01***

***https://github.com/csc413-su21/csc413-p2-TamirRasheed***

Table of Contents

1 Introduction ....................................................................................................................................... 3-4

[1.1 Project Overview 3](#_Toc3540)

[1.2 Technical Overview 3](#_Toc3541)

[1.3 Summary of Work Completed 4](#_Toc3542)

[2 Development Environment 4](#_Toc3543)

[3 How to Build/Import your Project 4](#_Toc3544)

[4 How to Run your Project 4](#_Toc3545)

[5 Assumption Made 4](#_Toc3546)

[6 Implementation Discussion................................................................................................................ 5 6](#_Toc3547)

[6.1 Class Diagram ............................................................................................................................. 5 6](#_Toc3548)

[7 Project Reflection 6](#_Toc3549)

[8 Project Conclusion/Results ................................................................................................................ 6 7](#_Toc3550)

1 Introduction

## 1.1 Project Overview

In this project, the goal is to make an interpreter for a mock language X that is given to us (This mock language X can be thought of as a simplified version of what Java’s language is). Within this project’s code, the interpreter is responsible for making the byte codes (creations from the source files that are given to us (mock language X)) . The interpreter works with a virtual machine in which we are to make as well to run a program that is written within this mock language X in a java environment. A way to think of this, is that the goal of this program is that it can translate what it is given and with that do what the translation tells it to do. There are two files of the same mock language X caliber in which we are to test to see that the code within the program works like how it is supposed to.

## 1.2 Technical Overview

Within this project, the goal is to make an interpreter of a mock language in which is given to us (Mock Language X(This language can be thought of as a simplified version of Java’s language)) and code a virtual machine to show the interpreted version of the language. The way this is to be done is by processing byte codes in which is given to us from the source code files (this is the mock language x files(can be seen by any file with the extension .X)). With the given source code files, the interpreter we make + the virtual machine (in which we are to make as well) work together to run the program in mock language x within a java environment. The way these two programs work together is that the interpreter part of the program makes a new virtual machine call whenever interpreter is called upon and the virtual machine goes through the bytecodes and translates the given source code files. If all works, we should have a nice translator in a way!

## 1.3 Summary of Work Completed (still needs to be done)

I was able to complete the operands and the operators class (this is including the separate classes for each possible operator) and the gui is complete as well. However, I was unable to successfully produce a correct output for all possible inputs. The tests that were provided did all pass (except for some of the EvaluatorTest) so I know it wasn’t because of that and yet the output if there is more than one operator would always return null. I believe it has something to do with my evaluator class and how I call my method to go through the stacks. Additionally, the gui does produce input and most of the output correctly. However, it doesn’t show the results of the expression the user puts in after pressing the “=” sign. I believe this error occurs because I was unable to handle the exception correctly.

# Development Environment

I did my project on Intellij with OpenJDK-15. Everything works perfectly with that!

# How to Build/Import your Project

Simply go to the github link provided on the cover page and clone it onto your local system, then after doing that, open the main project folder with intellij (or whatever IDE you are using) to be able to have the whole project (with the mock languages, bytecode files, interpreter file and the virtual machine package all together).

# How to Run your Project

After importing the project, go over to the configuration settings on your ide and make sure the build is running from the Interpreter java file found within the interpreter folder (the main folder) and that the working input you are using is one of the x.cod extension files given. You cab simply just go over to the interpreter file and run it from there as well (in Intellij).

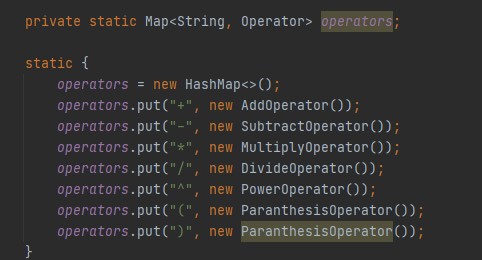
# Assumption Made

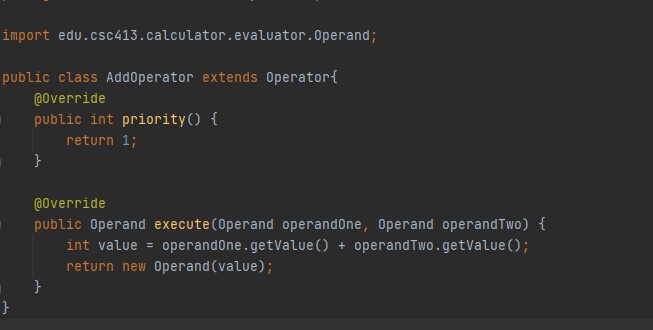
I tried my best to not make any assumptions while working on this project as I believe it is bad practice to assume anything when coding. However, after watching the videos, going through the given document multiple times and going through all the code myself, the only thing I can truly say I assumed was that the code showed in the video was ok and shouldn’t be edited(except where stated), that there were no errors within the given files of mock language x and that to follow the given document’s guidelines as closely as possible and to try to implement it in a way that the document shows. Another assumption I made was to not start this project late and to get a head start on it as this was a more complicated project when comparing to the last one given.

# Implementation Discussion

## Class Diagram

Here I’ll show a copy of my operator package and explain how I went about making the package. As can be seen by the screenshot below, I decided to make a hashmap with all the operators and make a separate class for each operator that function very similarly, just with different priorities (as can be seen from the project guidelines) and as well as a different math function (depending on the operand)





# Project Reflection

Throughout the process of trying to complete this project, I realized something. This is a summer course and with that time goes by really fast, even faster then a regular semester. The due date came by like nothing and because of that I was unable to complete the project with the right output. I was very close though, and because of that I am happy with my results. But, I am going to start ASAP with the next projects and not going to take any time I have available for granted.

# Project Conclusion/Results

In conclusion, the project doesn’t work as it is suppose to. I was very close yet very far in a way. As you can see by the screenshots below, I got the gui working correctly, and the result output is correct if its only one operand and one operator within the equation/expression, yet if there is two operators, the output returns null. Note that the gui is unable to show the results when clicking the “=” sign, like I’ve mentioned previously, I believe this is because I didn’t handle the exception properly.

