CSC 413 Project Documentation

Fall 2021

Ramit Singh

918213925

Class. Section #2

GitHub Repository Link

https://github.com/csc413-SFSU-Souza/csc413-p2-ramitsingh447

Table of Contents

1 Introduction 3

1.1 Project Overview 3

1.2 Technical Overview 3

1.3 Summary of Work Completed 3

2 Development Environment 3

3 How to Build/Import your Project 3

4 How to Run your Project 3

5 Assumption Made 3

6 Implementation Discussion 3

6.1 Class Diagram 3

7 Project Reflection 3

8 Project Conclusion/Results 3

1 Introduction
1.1 Project Overview
1.2 Technical Overview
1.3 Summary of Work Completed
2 Development Environment
3 How to Build/Import your Project
4 How to Run your Project
5 Assumption Made
6 Implementation Discussion
6.1 Class Diagram
7 Project Reflection
8 Project Conclusion/Results

1. Introduction

1.1. Project Overview

In this project, we have new Language called Language X. Made also interpreter to read and run the files in the new language being used. Extends the language to XCode as well and I made Virtual Machine that works with the interpreter to run the files. The interpreter comes with code that has execution for each function in the language so code works then and can be written in language X!

1.2. Technical Overview

In this project I implemented an interpreter for mock language X which can be the simplified version of java as well. For the interpreter, it should go and proceed the byte codes that are created from the source code with extension x. Then for the interpreter and other java files like virtual machine collaborate together to run the program written In the language X. Files have extension XCode including fib.x.cod.

1.3. Summary of Work Completed

I have implemented Runtime Stack class that records and process the stack of frames. Virtual Machine class is also used to operate the program. Also, it contains abstract functions for the byte code. I implemented the program class as well which stores the byte code from the source file and stores code in Array List as well all through. I implemented the resolve address label function that resolves the symbolics used in the functions. Bytecode Loader class was also Implemented and it loads the byte codes from the file into an array list. I created other byte code classes as well including halt, pop, go to, store, write, return code, load, call, label, false branch, Bop, and lit, dump and helps execute the program.

3. How to Build/Import your Project

Copy the link, go to the terminal, then git clone paste the link. Then open IntelliJ and click open or import option and repo is cloned choose your folder and then click ok option.

4. How to Run your Project

To run the project you go to the Interpreter class and edit the configuration and change the program arguments to fib.x.cod or factorial.x.cod.

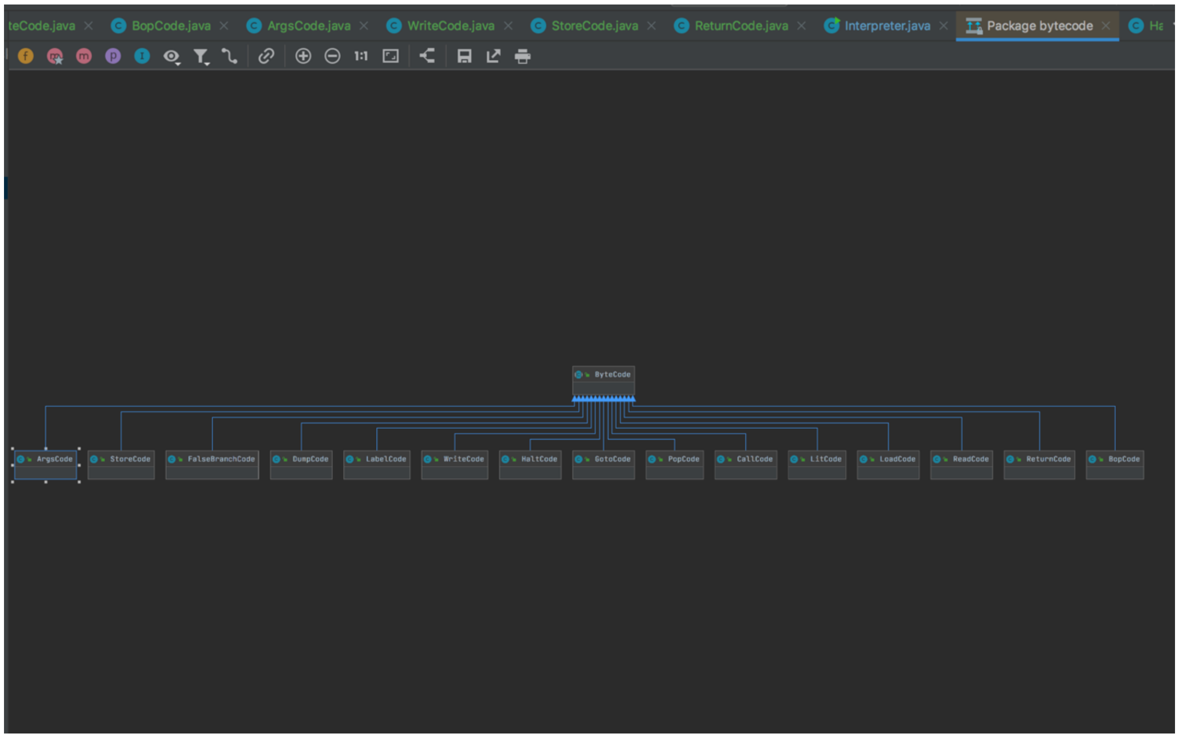
5. Assumption Made

I will assume that Bytecode source .cod files are to be tested.

6. Implementation Discussion

When I implement the project, I looked at first file we did and in our Table Code is where we first start off making subclasses that we needed to use.

6.1. Class Diagram



7. Project Reflection

This was one of my difficult projects I have done, even though I watched the videos it helped a lot more. I was having hard time getting fib.x.cod to run but I worked it out and everything else was perfect so I hope I get a better grade than what I got in ASMT 1.

8. Project Conclusion/Results

The project was fun to do and was helpful for my programming skills to build up. I put little more effort in this ASMT rather than the first one maybe because I started earlier on this one, which gave me more time to do all the things right, but other than that I believe everything should successfully run.