CSC 413 Project Documentation
Summer 2019

Student Melissa Estrada

Student ID Student ID 918591991

Class.Section
413.02

GitHub Repository Link
https://github.com/csc413-02-summer2019/csc413-p2-melissa4444

Table of Contents

1. Introduction

This project takes files with the extension of .x.cod as inputs and processes the byte codes to produce the output of the interpreter. This project implements an abstract class ByteCode that can be inherited by all byte code classes. The Virtual Machine class can carry out all the operations for byte codes and run time stack class that stores all the current values in the stack.

1.1. Project Overview

The interpreter loads the byte codes into an array list using the byte code loader and resolves the address before passing it to the Virtual Machine that has access to run time stack object which holds the elements in the run time stack as well as frame pointers in the stack, to execute the byte code operations. The virtual machine class also has a stack to store the address for the program to counter can be restored when it returns from the function. All the byte codes can access the stack indirectly via virtual machine.

1.2. Technical Overview

This project implements an interpreter to execute the byte codes. The Interpreter and the code table classes were already implemented and descriptions were given for each byte code operation. All byte code classes are implemented as well as the dump byte code that is used in the dumping state.

1.3. Summary of Work Completed

2. Development Environment

IntelliJ idea

java version "1.8.0_201"

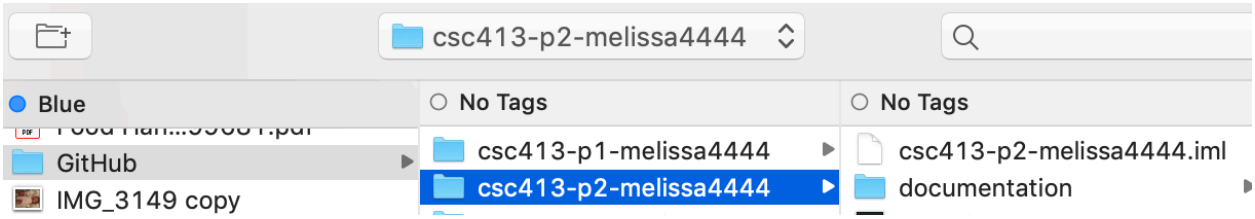
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)

Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)

3. How to Build/Import your Project

First you will have to click on the link that is provided on the cover page of this report. This will lead you to my GitHub Repository Link from there you will see the following image by clicking on the green button you can download the project.

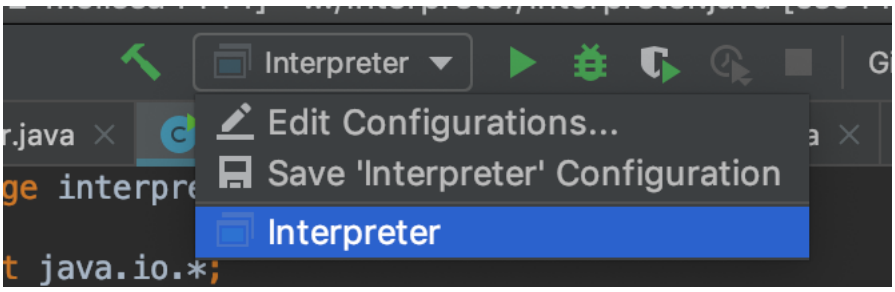
From there I personally used GitHub Desktop to sync the project. After doing this I launched IntelliJ IDEA. Then I choose the option to import a project. And make sure to choose the whole folder as the root.



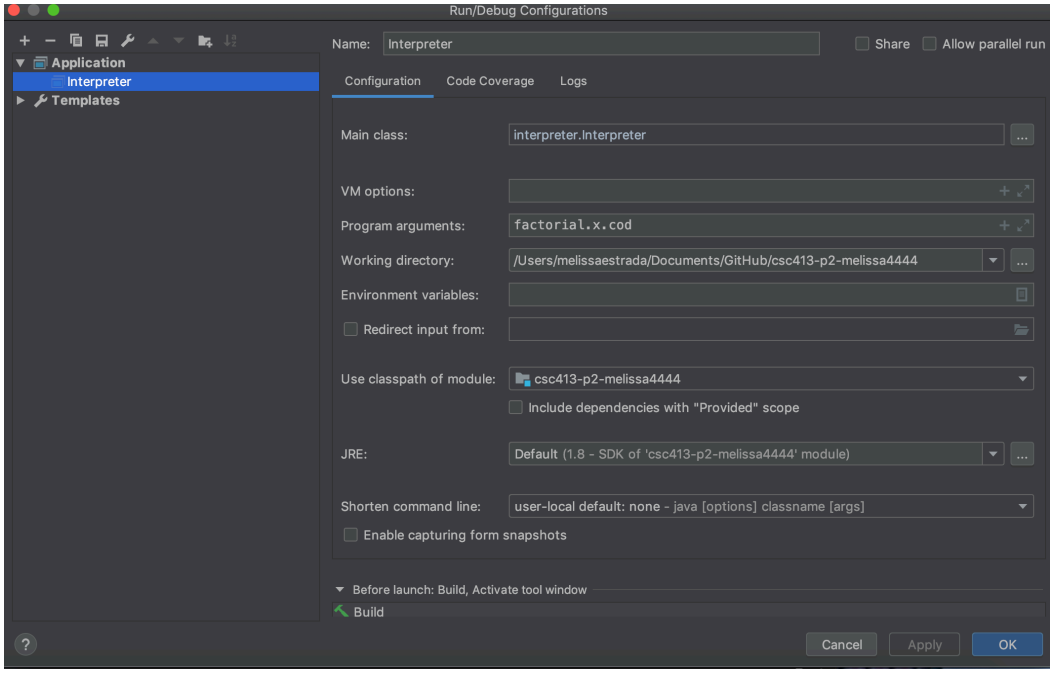
From there the project should be properly imported and built by using both Github Desktop and IntelliJ IDEA.

4. How to Run your Project

To rum my project you must go towards the top right corner like the following image, and click on the drop down arrow next to the word interpreter and choose edit configurations.



The following image should be displayed, go to the Program arguments section and add any of the .cod files. From there click apply.



Now go ahead and click on the green arrow to run.

Unfortunately my project does not build properly.

5. Assumption Made

Factorial can be calculated till input value of 12 after that there is overflow

No divide by 0 conditions

test files have no errors

6. Implementation Discussion

Interpreter Class implements the main method where it takes the input file name from command line arguments and store the byte code objects into an array list using byte code loader class object. The program array list then will be passed to Virtual Machine and the byte codes will be processed.

Code table class implements hash map for storing class names for byte codes. It contains the byte code strings that are keys and corresponding class names are the values.

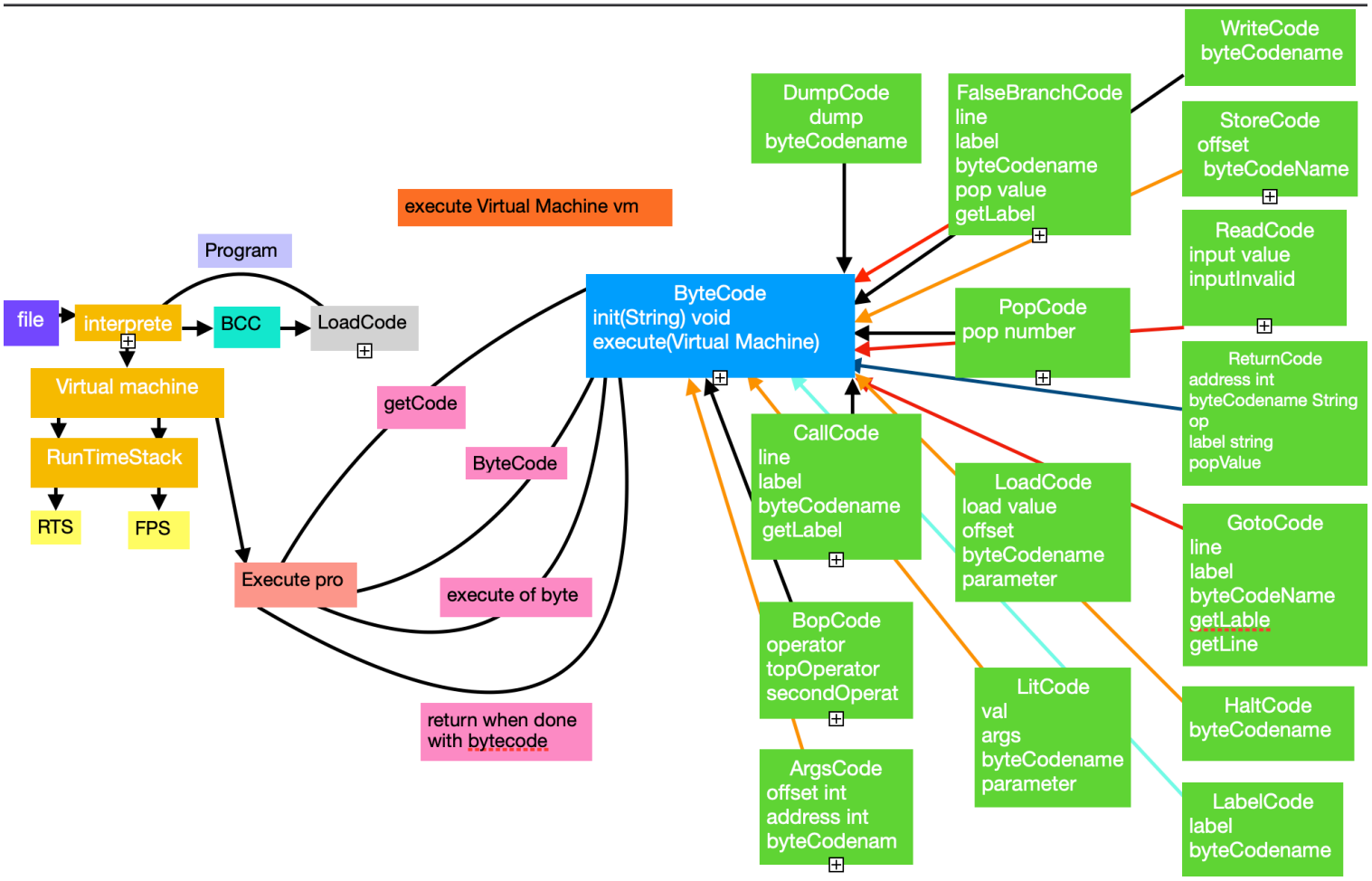
Program class implements a hash map to store the line numbers for label byte codes. The key will be name of the labels and line number of corresponding labels are mapped as values. It also implements a method to resolve the address of labels where it fetches the line numbers from the hash map and stores it in corresponding class variables for the goto, false branch, and call byte codes.

Byte code loader class reads the input file using buffered reader and stores the objects of byte codes in an array list program. The incoming line from buffered reader will be tokenized using String.split() method to store the arguments of byte codes which are the parameters for init functions of byte codes. The instants of byte code objects are created dynamically by using the (Byte Code) (Class.forName“interpreter.ByteCode.”+codeClass).newInstance()) where code class is the class name fetched from Code Table hash map. These objects of byte code are stored in an array list and address are resolved for GOTO, FALSEBRANCH, and CALL using resolveAddrs() method of Program class.

Virtual machine class is the array list of byte code objects are passed to the Virtual machine where PC counter is used to read the array list. The object of Virtual Machine itself is passed as a parameter for execute() method of byte codes using which byte code operations are performed. The Virtual Machine contains a flag used for dumping program state. The byte codes and run time stack are printed when the dump is true. The virtual machine contains object of run time stack using it to access the stack elements. The return address stack is also created in the virtual machine to store the pc counter value so that the address can be restored after returning from the function.

The ByteCode class is an abstract superclass in the Interpreter that is inherited by all byte codes classes. It contains three abstract methods one store the parameters of byte codes into private variables. The second abstract method access the stack elements vial the virtual machine object vm and performs the operation. The third returns the byte code name which is used for dumping. This class is inherited by all 15 sub classes each byte codes used to override the abstract method of the byte code class.

6.1. Class Diagram



7. Project Reflection

Learning how to use the String.split method as well as understanding the workflow of the virtual machine. Being able to research objects dynamically and being able to work with a complex array list. Even though my project does not work I learned a lot and still have a lot left to learn since I cannot get it to work.

8. Project Conclusion/Results

My conclusion is that this project is extremely complicated and it is one that requires a lot of research and time and not being afraid to ask for help. I kept having problems with returning a null for the getLabel() still not quite sure how to fix.