CSC 413 Assignment 2 Documentation

Summer 2021

Simon Wu

920698802

CSC413-01

Table of Contents

1 Introduction

1.1 Project Overview

This program will read a bytecode file and perform actions based on what is written. The program will first read through the file and create a list of actions that it finds on the file. Once it is done creating the list, it will then begin executing the actions in order until it reaches the end of the file or until it reaches an action that tells the program to stop execution.

1.2 Technical Overview

The program will perform actions based on the bytecode file that it is given. It performs these actions by using two stacks, and two ArrayLists. The stacks are “framePointer” and “returnAdresses”. The ArrayLists are “runTimeStack”, and “program”. The program will first parse through the bytecode file and create and initialize ByteCode objects for every line. Once the file has been parsed completely, the program will then resolve symbolic addresses with numerical ones for each bytecode if it is necessary. After the addresses are done being resolved, it is time for the program to start executing the bytecodes in the order that it was parsed. The “returnAdresses” stack will keep track of the positions in “program” that it will go back to. The “runTimeStack” will keep track of integers that are added or removed by the bytecode’s execution, and the “framePointer” will keep track of what integers in the “runTimeStack” belong to a function together.

1.3 Summary of Work Completed

- Implementation of Bytecode and abstraction of Bytecodse
- Implementation of Program’s resolveAddress()
- Implemented functions for the runTimeStack that would create behavior like stack methods
- Implemented methods for manipulating the framePointer stack
- Implemented the VirtualMachine class and created methods that would be used by the Bytecodes so it would perform actions on the runTimeStack

2 Development Environment

I am using OpenJDK-16.0.1 for my Java.

I used the IntelliJ Idea Ultimate edition as my IDE for this project.

3 How to Build/Import your Project

To import the project, you open the folder “csc413-p2-swu465”.

4 How to Run your Project

To run this program, you will have to right-click the Interpreter class and hit “Run ‘Interpreter.main()’”. After that you will have to edit the configuration of the Interpreter application that is created by. For this to run properly, you will need to put the bytecode file that you want the program to run inside the root folder and add it to the application’s program arguments by its filename.

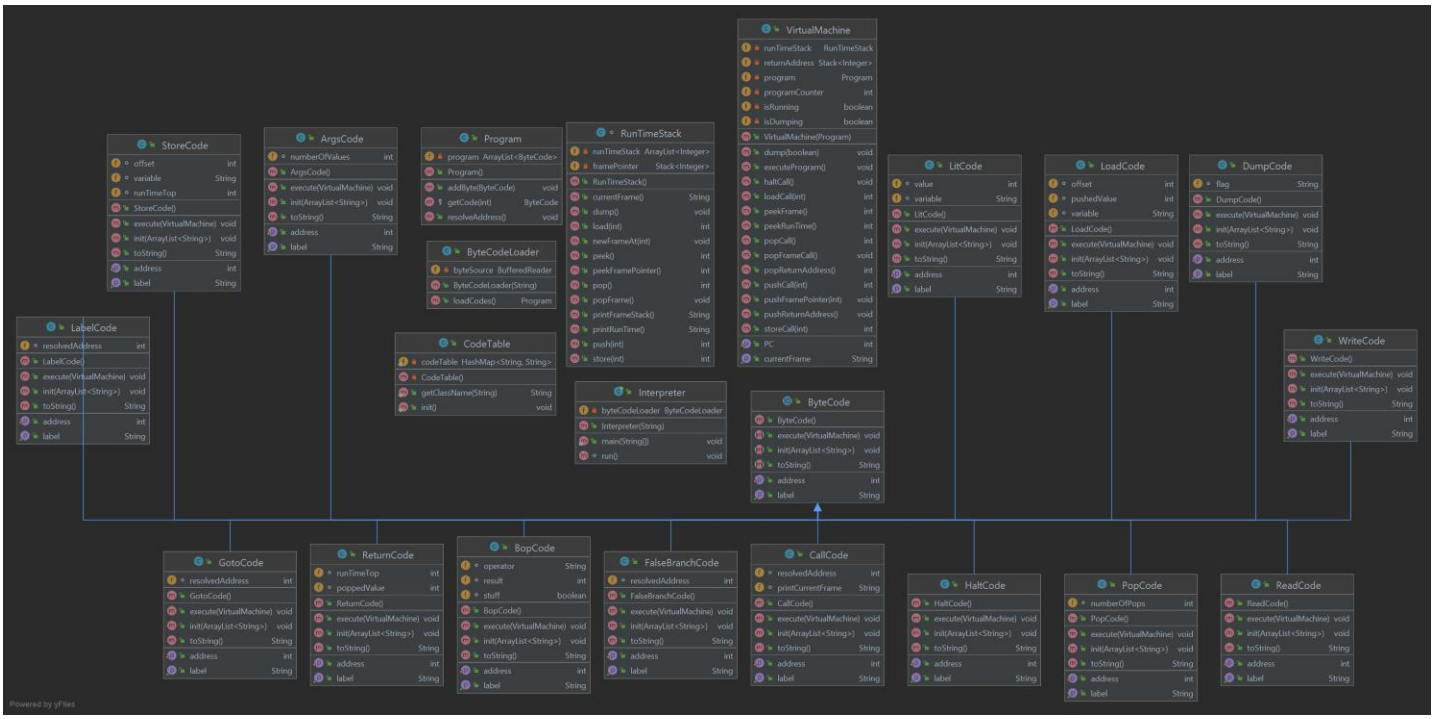
5 Assumption Made

- The user knows to give the program a valid bytecode
- The user knows what the program can and cannot handle

6 Implementation Discussion

For the most part, I tried my best to follow the requirements that was given to me. I started by working on the Program, RunTimeStack, VirtualMachine, and ByteCodeLoader classes. After I finished implementing the ByteCodeLoader class, I started implementing the ByteCode class and its abstractions. Like the last project, I created new methods to help me print out the entire stack instead of relying on the dump function for debugging. For many of the runTimeStack methods, I tried to account for error checking by using if statements to make sure things would go the way I wanted them to if I thought it was necessary.

6.1 Class Diagram



7 Project Reflection

This project as a concept was not very hard to understand. But I did have some troubles implementing it. The ByteCode abstraction was not very hard. I do not think resolving the symbolic addresses were very hard either, it was making sure that the ByteCodes executed correctly that was the problem. The program would be able to run without crashing, but sometimes the program would end early or infinitely loop. At first, problems were happening because I would sometimes be popping twice by accident. Or I would forget to push something onto the runTimeStack. The biggest trouble was getting the dump to work correctly. Right now, I think I have gotten the program to almost work completely as intended going from the sample file you have given but it is not quite there yet. I needed to adjust the

BOP bytebode and make some changes to the some of the methods in the RunTimeStack that you said needed some trimming.

8 Project Conclusion/Results

I believe the program is almost there. I have gotten it to stop crashing and I think I have gotten the dump function to work correctly now. But it is still not giving the exact same results that was shown in the sample file for Factorial when a 9 is given. I think there may still be some lingering bugs that need fixing but I think it is almost there.