CSC 413 Project Documentation

Fall 2019

Alejandro Zamora Ruiz

917841098

CSC 413.02

https://github.com/csc413-sp21/csc413-p2-zruiz95.git

Table of Contents

1 Introduction

1.1 Project Overview

This program's purpose is to read a file and perform a specific set of instructions based on text within that file. The program prompts the user for input in the form of an integer value which the program then reads and stores it into a stack. Once stored, the program manages the stack of values separated by functions and works down the list of each until the result is successfully calculated and returned.

1.2 Technical Overview

The program can read a source file with extension .x titled either "fib.x" or "factorial.x" in conjunction with files named "fib.x.cod" and "factorial.x.cod" which both contain bytecode and arguments for calculating a number. When we set the program up to read the files, running it will prompt us for a non-negative number and calculate the result based on the user's input. This value is stored into a stack and frames used to store the values that are generated by each of the bytecode files. The number will be run through bytecode files within the program that will eventually calculate the correct result.

1.3 Summary of Work Completed

An abstract class titled `ByteCode` was created along with fifteen subclasses that all extend `Bytecode.` An interface class titled `AddressClass` was created which is implemented by some of the `ByteCode` subclasses. The `ByteCodeLoader` class parses the program argument file into `ByteCode` objects that correspond to their class names. The arguments are then stored into an `ArrayList` that then moves these values to a Program object. I also implemented the Program class to resolve the `ByteCode` addresses that implements the interface named `AddressLable` which will let these objects be pointed towards the correct `ByteCode` when the program executes. The `RunTimeStack` class was implemented with various methods borrowed from the `StackClass` class which was designed to assit in managing the runtime stack. The `VirtualMachine` class is also implemented and used by the previous methods. The idea behind all these methods is to preserve encapsulation when implementing the `RunTimeStack` class methods and the `ByteCode` classes are not allowed to call methods from the `ByteCode` class directly. After all this was done, we added configurations for Fibonacci and Factorial so the program can run correctly

2 Development Environment

Java Version: `openjdk version "11.0.10" 2021-01-19`

IDE: IntelliJ IDEA 2020.3.2 (Ultimate Edition)

VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.

Operating System: Windows 10 Pro 20H2

Hardware:

- AMD Ryzen 7 2700x 3.7gHz
- 16GB RAM
- NVIDIA GeForce GTX 1650 4GB

3 How to Build/Import your Project

1. Open IntelliJ
2. Click Import Project
3. Find the folder titled “csc413-p2-zruiz95” and open it
4. Be sure to click on the “Create Project from Existing Sources” tick mark
5. Click “Next” until the folder is imported
6. Build the project using the IDE by clicking on *Build* ➔ *Build Project* in the top menu bar

4 How to Run your Project

1. Once project is finished building, go to *Run* ➔ *Edit configurations* in the menu bar
2. When the window opens, click the “+” at the top left of the window and select “Application”
3. Enter *Fibonacci* in the *Name* text field
    a. Enter Factorial in the *Name* text field when setting up Factorial
4. Type *“interpreter.Interpreter”* in the *Program Arguments:* text box
5. Type “fib.x.cod” in the *Program Arguments:* text box.
    a. Type “factorial.x.cod” in the *Program Arguments:* text box when setting up factorial
6. Once both configurations are created, click the *OK* button which closes the window
7. In the IDE, navigate to the *Run/Debug* dropdown menu and select the configuration of your choice, then click run. Repeat for other configurations

5 Assumption Made

Originally, I felt this project was going to be a challenge but now that it’s complete it was more of a challenge than anticipated. It’s a good think I started the project early or I would have missed the deadline. Overall, the project was a challenge because it tested parts of my ability to solve a problem beyond what other assignments in the past have done. I had to build fifteen different classes and make sure they were properly implemented in order to have the program function properly. This part of the assignment took the longest and needed the most of my attention so I would not loose my train of thought (Which is easy for me).

6 Implementation Discussion

All ByteCode subclasses will have their own dedicated files and the ByteCode classes are stored in the dame directory along with the interface whose job is to distinguish the byte code classes that need address resolution. ByteCodeLoader will then read the argument file and create new ByteCode objects.

6.1 Class Diagram

A full resolution copy of the class diagrams is included in the Documentation folder for convenience. I used IntelliJ to generate it vs using Lucid Chart.

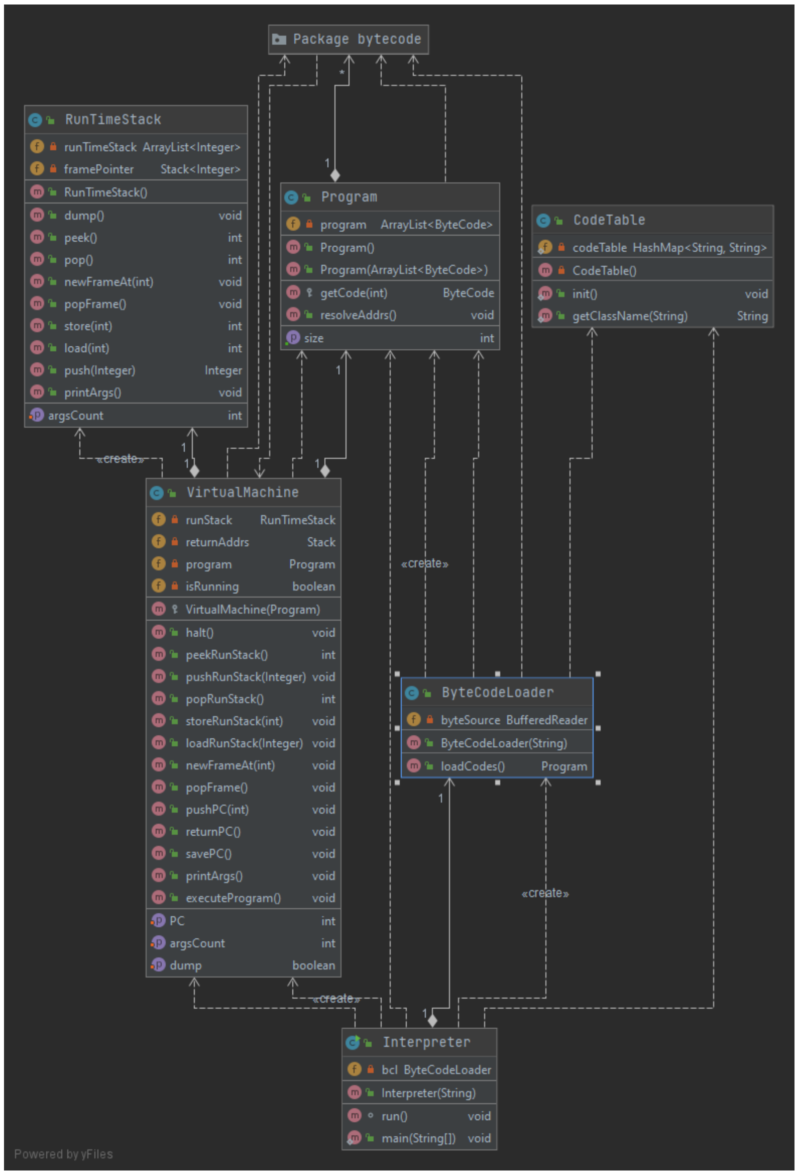

Figure 1: Interpreter Classes

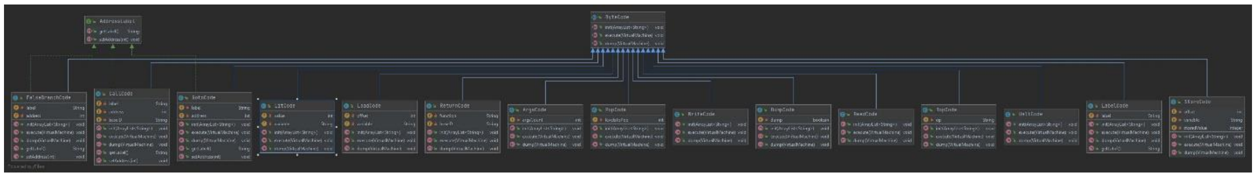

Figure 2: ByteCode Dependencies. Full resolution image included in the Documentation file

7 Project Reflection

This project was easily one of the most difficult assignments I have ever completed. When I opened the PDF instructions up and saw it was a full-on manual, I knew I was in for a real treat. Overwhelmed with the number of instructions that is needed to complete the assignment I began tackling it as best I could. The most time-consuming part was getting all fifteen subclasses in ByteCode running and making sure they are implemented with the rest of the program and countless builds and test code to get to the result. I’m surprised I didn’t go bald in pulling my hair out trying to figure out things like how to optimize RunTimeStack because I wanted to make sure the stack where all the number values were to be stored functioned as best as it could. Lots of effort and thought went in to getting popFrame and the store methods to function properly.

Overall, this project although difficult, allowed me to really flex my programming muscle and learn new ways to develop a program. I learned a lot about encapsulation, and even though the instruction manual was rather large and overwhelming, it provided a lot of welcomed help when needed.

8 Project Conclusion/Results

After spending way more time than I anticipated on this project, I was able to get the program to work but I’ll be honest, I did not test the project beyond making sure it outputted the correct values for each of the configuration. The amount of excitement that went through me when I ran both configurations and saw it returned the correct value is a feeling that will never get old, no matter how many projects or assignments I do.