CSC 413 Project Documentation

Fall 2019

Japheth Wun

921555593

CSC 413

https://github.com/csc413-su21/csc413-p2-finaldestroyer.git

Table of Contents

1 Introduction

1.1 Project Overview

For this program, we took in code from a source code and interpreted it from a mock language into byte code. Byte code can be considered as a set of instructions for the program to run. While interpreting the byte code interpreter, depending on the source file it will process different programs. Once interpreted, the program will take a user input and recursively find its fibonacci number. To do the functions to find the number, we take an external source codes and implement its texts found inside files.

1.2 Technical Overview

For this program, we take in some .cod files and read it. It would then interpret its instruction sets and create a program that will print out a fibonacci number. We ask for a user input and then we will print the fibonacci number for the given input.

1.3 Summary of Work Completed

The program itself was not completely finished since time was short. I was able to complete at least 80% of the work that was assigned I believe. It can not run yet but the coding logic is there. The thing that I would need to do to finish this program would be to clean up the code a bit and finish the program code. The program code itself was a bit confusing to do. I didn't really have time to test the program and fixx it, but I would need to look into why the file instructions are not being read into the interpreter.

2 Development Environment

Version of Java Used: JAVA 15
IDE Used: IntelliJ

3 How to Build/Import your Project

To import and build this project, you need to first download the file from the github repository and then unzip it. With an IDE, such as intelliJ, usually you go to file and you click on open, then search for the unzipped file of the project you downloaded from the repository. Once the class has been imported, you

can find the build function on top of the IDE. By building the code, you are able to access and use the program.

4 How to Run your Project

To run this program you would need to first go into the interpreter file. Then within the IDE, such as IntelliJ, there is always a run function on top of the IDE. Don't click run right away, you must click on its edit configuration and enter a user argument for the project to run.

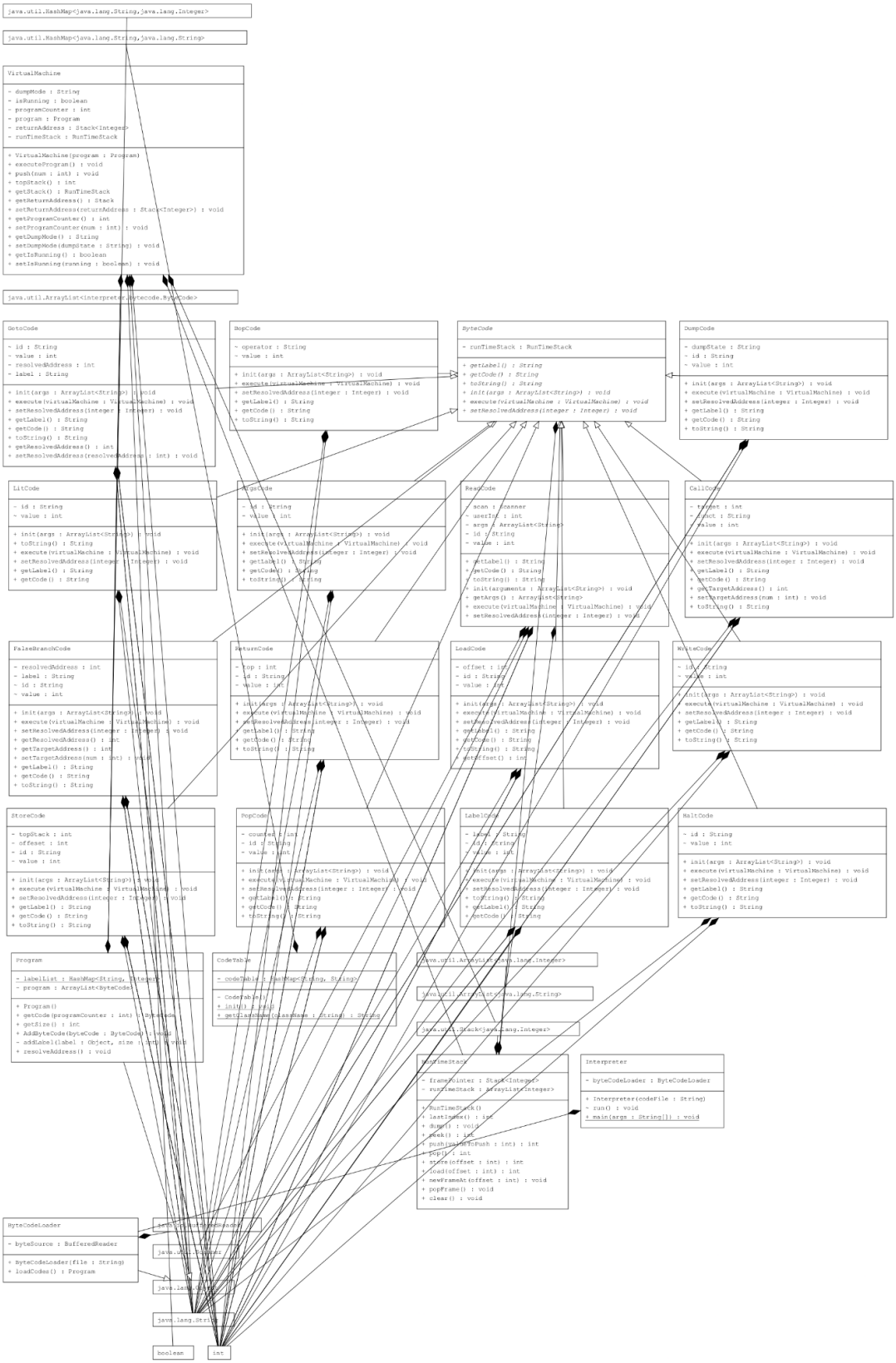
5 Assumption Made

There were many assumptions made for this project such as how it would take in code and how it would use those code. I assumed that by taking and reading a cod file, it would take instruction sets for the program to correctly work. When designing, coding and implementing the project, we would create multiple classes which represented the functions such as call, goto, pop, and dump, etc. . These functions would follow the instruction sets given from the file to complete a set of fibonacci numbers.

6 Implementation Discussion

The implementation of the program would first take a user input, then use it as a counter to find the fibonacci number for the given number. As it loops, it would create a list and then read the file, after reading the file it would put its instruction into another list for the code to work. It would first call the program and pop the functions/instruction set from the functions/instruction list.

6.1 Class Diagram


java.util.HashMap<java.lang.String,java.lang.Integer>
java.util.HashMap<java.lang.String,java.lang.String>
VirtualMachine
- dumpMode : String
- isRunning : boolean
- programCounter : int
- program : Program
- returnAddress : Stack<Integer>
- runTimeStack : RunTimeStack
+ VirtualMachine(program : Program)
+ executeProgram() : void
+ push(num : int) : void
+ topStack() : int
+ getStack() : RunTimeStack
+ getReturnAddress() : Stack
+ setReturnAddress(returnAddress : Stack<Integer>) : void
+ getProgramCounter() : int
+ setProgramCounter(num : int) : void
+ getDumpMode() : String
+ setDumpMode(dumpState : String) : void
+ getIsRunning() : boolean
+ setIsRunning(running : boolean) : void
java.util.ArrayList<interpreter.bytecode.ByteCode>
GotoCode
~ id : String
~ value : int
~ resolvedAddress : int
~ label : String
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
+ getResolvedAddress() : int
+ setResolvedAddress(resolvedAddress : int) : void
BopCode
~ operator : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
ByteCode
~ runTimeStack : RunTimeStack
+ getLabel() : String
+ getCode() : String
+ toString() : String
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
DumpCode
~ dumpState : String
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
LitCode
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ toString() : String
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
ArgsCode
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
ReadCode
~ scan : Scanner
~ userInt : int
~ args : ArrayList<String>
~ id : String
~ value : int
+ getLabel() : String
+ getCode() : String
+ toString() : String
+ init(arguments : ArrayList<String>) : void
+ getArgs() : ArrayList<String>
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
CallCode
~ target : int
~ funct : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ getTargetAddress() : int
+ setTargetAddress(num : int) : void
+ toString() : String
FalseBranchCode
~ resolvedAddress : int
~ label : String
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getResolvedAddress() : int
+ getTargetAddress() : int
+ setTargetAddress(num : int) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
ReturnCode
~ top : int
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
LoadCode
~ offset : int
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
+ getOffset() : int
WriteCode
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
StoreCode
~ topStack : int
~ offeset : int
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
PopCode
~ counted : int
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
LabelCode
~ label : String
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ toString() : String
+ getLabel() : String
+ getCode() : String
HaltCode
~ id : String
~ value : int
+ init(args : ArrayList<String>) : void
+ execute(virtualMachine : VirtualMachine) : void
+ setResolvedAddress(integer : Integer) : void
+ getLabel() : String
+ getCode() : String
+ toString() : String
Program
- labelList : HashMap<String, Integer>
- program : ArrayList<ByteCode>
+ Program()
+ getCode(programCounter : int) : ByteCode
+ getSize() : int
+ AddByteCode(byteCode : ByteCode) : void
- addLabel(label : Object, size : int) : void
+ resolveAddress() : void
CodeTable
- codeTable : HashMap<String, String>
- CodeTable()
+ init() : void
+ getClassName(className : String) : String
java.util.ArrayList<java.lang.Integer>
java.util.HashMap<java.lang.String,java.lang.String>
java.util.Stack<java.lang.Integer>
RunTimeStack
+ frameStack : Stack<Integer>
- runTimeStack : ArrayList<Integer>
+ RunTimeStack()
+ lastIndex() : int
+ dump() : void
+ peek() : int
+ push(numToPush : int) : int
+ pop() : int
+ store(offset : int) : int
+ load(offset : int) : int
+ newFrameAt(offset : int) : void
+ popFrame() : void
+ clear() : void
Interpreter
+ byteCodeLoader : ByteCodeLoader
+ Interpreter(codeFile : String)
+ run() : void
+ main(args : String[]) : void
ByteCodeLoader
- byteSource : BufferedReader
+ ByteCodeLoader(file : String)
+ loadCodes() : Program
java.io.BufferedReader
java.util.Scanner
java.lang.Object
java.lang.String
boolean
int

7 Project Reflection

As mentioned in the implementation, the design choices made was to take a number given by the user and print its fibonacci number. To do so, it would create sets of instructions for the program to go to produce the number that the user desired. But I wasn't able to complete this program and only do the code since its time was super limited. This project was extremely difficult for me since I lacked a lot of java experience since I'm still new to it. I had asked many of my classmates for a little bit of help and hints, but they seemed confused as well. I just believe that if I had more time and understood the project a bit better I could complete the project. I believe that if i was able to debug the program, it would be able to work within a few days.

8 Project Conclusion/Results

The conclusion of this project is that it does not run yet, but most of the logic for this program is there. The program’s functionalities are all there but it has not been cleaned and implemented correctly yet.