Documentation for Assignment 3
CSC 413-02

Tommy Tran

https://github.com/CSC-413-SFSU-02/csc413-p3-interpreter-tmytrn
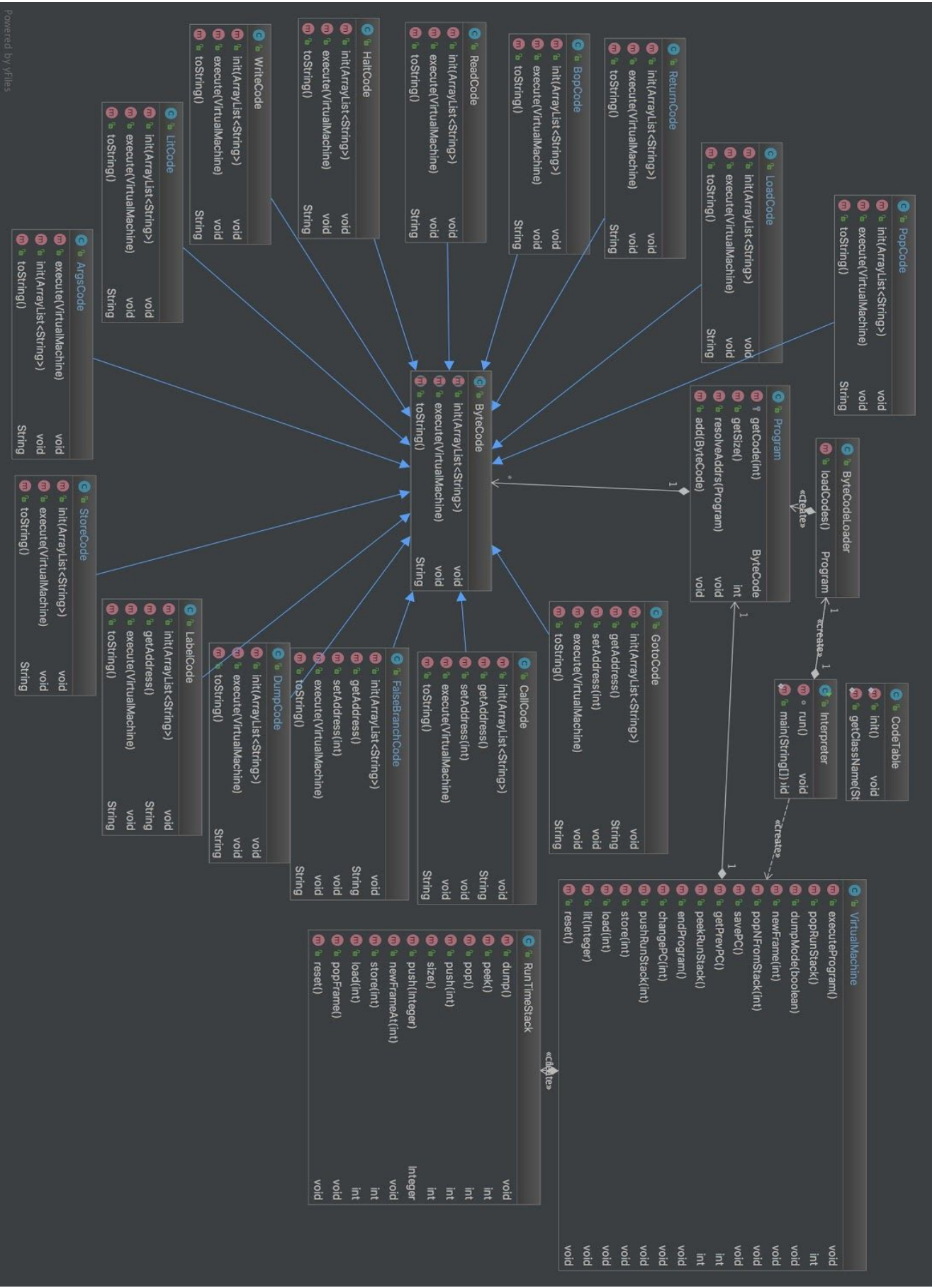
Introduction

The project in this repository is an Interpreter that will read and process .x.cod files. This Interpreter uses a Virtual Machine written by me. The virtual machine uses a set of bytecodes to read the .x.cod and perform the necessary operations that interpret the x code. To run this project, it's highly recommended to use the IntelliJ IDE, since the project was edited in this development environment.

To run this project, open in an IDE and press the play button. Make sure that an x.cod file is in the program arguments when running the Interpreter.main method.

Overview/Implementation

We were tasked with completing the ByteCodeLoader class by creating the necessary bytecode classes and storing them into a directory for easy access. After successfully finishing the ByteCodeLoadr, we had to ensure that the Program class resolved all of the code addresses to prep them for the VirtualMachine. After finishing the Program class, we needed to have the bytecodes access the RunTimeStack without breaking encapsulation. Lastly, we needed to implement a dump method to help print the current state of the runtime stack. Once any/all bugs are removed, the program should should run smoothly.

The UML diagram below shows the layout the project. Notice how all bytecodes inherit from the ByteCode abstract class.

UML Class Diagram

PopCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

ReturnCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

BopCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

LoadCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

ReadCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

HaltCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

WriteCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

LitCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

ArgsCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

ByteCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

Program
- getCode(int) : ByteCode
- getSize() : int
- resolveAddrs(Program) : void
- add(ByteCode) : void

ByteCodeLoader
- loadCodes() : Program

«create»

StoreCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

LabelCode
- init(ArrayList<String>) : void
- getAddress() : String
- execute(VirtualMachine) : void
- toString() : String

DumpCode
- init(ArrayList<String>) : void
- execute(VirtualMachine) : void
- toString() : String

FalseBranchCode
- init(ArrayList<String>) : void
- getAddress() : String
- setAddress(int) : void
- execute(VirtualMachine) : void
- toString() : String

CallCode
- init(ArrayList<String>) : void
- getAddress() : String
- setAddress(int) : void
- execute(VirtualMachine) : void
- toString() : String

GotoCode
- init(ArrayList<String>) : void
- getAddress() : String
- setAddress(int) : void
- execute(VirtualMachine) : void
- toString() : String

Interpreter
- run() : void
- main(String[]) : id

CodeTable
- init() : void
- getClassName(St...)

VirtualMachine
- executeProgram() : void
- popRunStack() : int
- dumpMode(boolean) : void
- newFrame(int) : void
- savePC() : void
- popNFromStack(int) : int
- getPrevPC() : int
- peekRunStack() : int
- endProgram() : void
- changePC(int) : void
- pushRunStack(int) : void
- store(int) : int
- load(int) : int
- lit(Integer) : void
- reset() : void

RunTimeStack
- dump() : void
- peek() : int
- pop() : int
- push(int) : int
- size() : int
- push(Integer) : Integer
- newFrameAt(int) : void
- store(int) : int
- load(int) : int
- popFrame() : void
- reset() : void

«create»

Assumptions

Some assumptions made about the project include how little code we were going to be given. A major assumption was that bytecodes needed to be made from scratch, according the the given hashmap in the CodeTable class. These bytecodes perform all the minute operations, while the VirtualMachine class simply executes each bytecode while reading the given x.cod file. Another assumption made is that the use of reflection is key to completing the assignment.

Results and Conclusions:

The result in this repository is a fully functional Interpreter program that will execute any x.cod files given to it. It fulfills all requirements given in the assignment.pdf. One of the toughest challenges encountered during this project was understanding the code given to us. This project has several moving parts in which one class heavily relies on the other for the program to work. Since several tasks throughout this project need to be completed, finding a workflow and portioning the work accordingly was quite a hurdle to overcome.