CSC 413 Project 2 Documentation

Summer 2024

Joshua Garcia

921986504

CSC 413 [02]

https://github.com/csc413-SFSU-SU2024/interpreter-JoshuaR503

Table of Contents

1 Introduction

1.1 Project Overview

Small project that is able to read and make sense of the instructions in a code file name for language called X

1.2 Technical Overview

This program is an interpreter that is able to process bytecodes from a source file for a programming language called X.

1.3 Summary of Work Completed

- Implemented all bytcode classes inside the bytecodes folder.
    - Implemented toString, execute and init methods for each class
    - Implemented ByteCode interface
- Implemented ByteCodeLoader to load files.
- Implemented CodeTable to map ByteCode structures
- Implemented Program resolve address
- Implemented virtual machine, which keeps track of the program state
- Implemented runtime stack
    - Manipulates stack content
    - Displays stack content
    - Keeps track of activation records

2 Development Environment

Amazon Corretto 20, InteliJ IDEA Ultimate Edition

3 How to Build/Import your Project

Clone github repository into the desired folder

On the root folder, run javac -cp . -d out interpreter/*.java interpreter/bytecodes/*.java interpreter/loaders/*.java interpreter/virtualmachine/*.java

4 How to Run your Project

```
javac -cp . -d out interpreter/*.java interpreter/bytecodes/*.java interpreter/loaders/*.java interpreter/virtualmachine/*.java
```

run either one of the following:

```
java -cp out interpreter.Interpreter factorial.verbose.cod
```

```
java -cp out interpreter.Interpreter functionArgsTest.cod
```

```
java -cp out interpreter.Interpreter fib.x.cod
```

```
java -cp out interpreter.Interpreter factorial.x.cod
```

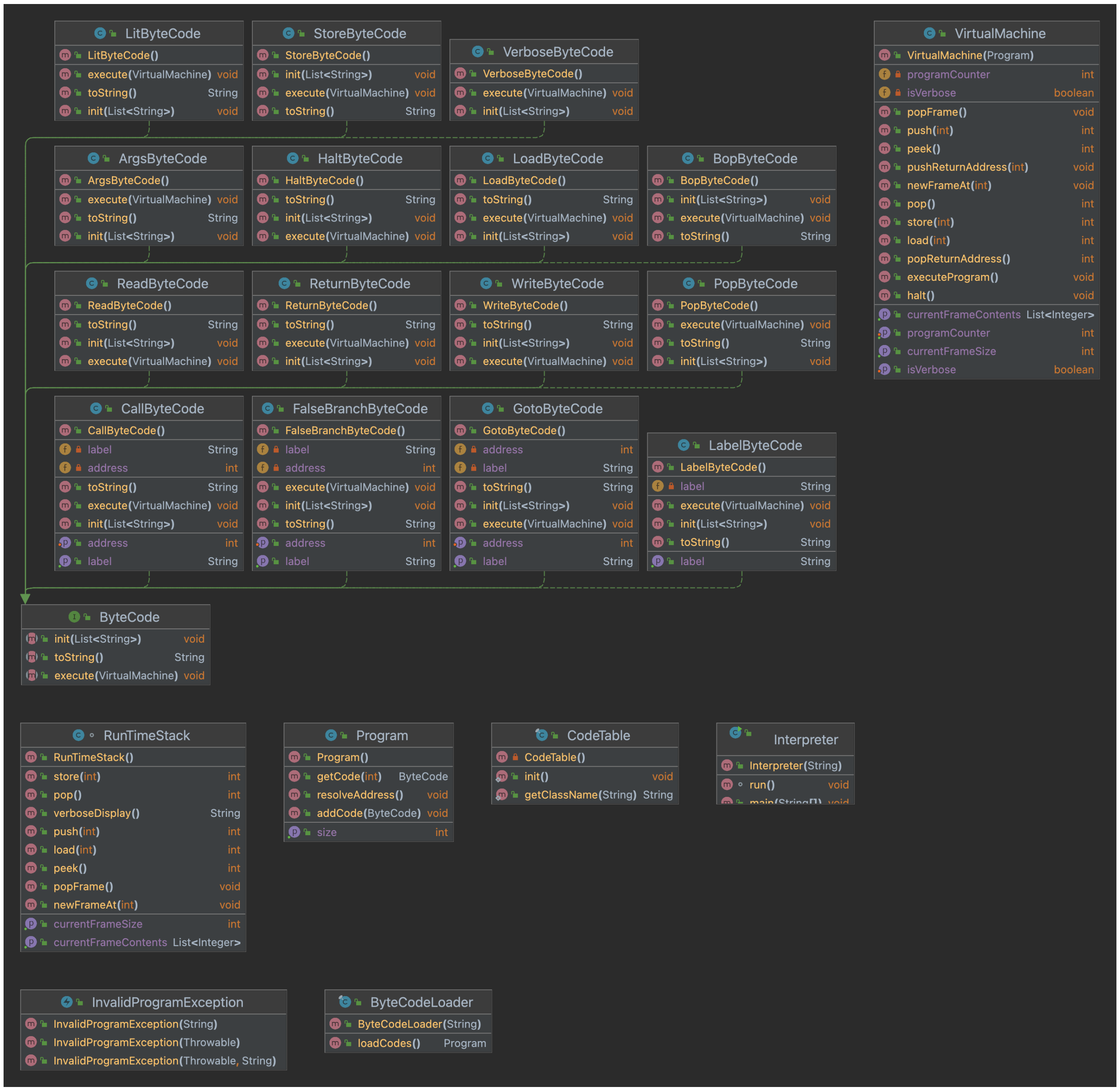
5 Assumption Made

.cod files do not contain any errors

6 Implementation Discussion

There wasn't much implementation left up to me as there was a very clear and strict way of doing things, which worked great. There was also some flexibility to do extra stuff, such as missing methods in the runtimestack class. I also dedided to create an interface for Call, Falsebranch, and Goto, since they shared the same methods. It made it easier to resolve the address.

6.1 Class Diagram



7 Project Reflection

Hands down one of the most challenging projects I’ve ever worked on. Lucky me was able to team up with some people and we worked together to complete this. I am not sure how far along they got, but I am glad I was able to find some help to tackle very challenging sections such as the bop operator, it remined me a lot about the previous project, and it helped me get through it. Trying to not break encapsulation was a truly challenging, so I had to find some workarounds. I am very nervous about what I had to do at CallByteCode, so we will see how it went. Not only this, but this project required me to pull several all nighters which is no bueno, but during the evening I work better and time does not seem to pass. Other than this I cannot believe I coded a Virtual Machine? I am pretty sure they are much more complex in the real world, but it was nice seeing how the concept works.

However, I didn’t fully get the interpreter to work, one of my outputs, fib.x.cod does not match the one given by Prof Anthony. I am not sure what went wrong but the displayed stack is not how it should be. I spent about 3 hours chasing down the bug but couldn’t found it. I though my implementation of store was wrong so I rewrote it. I did the same with pop and load, but no luck. I then thought it was verbose implementation but it didn’t work either. The output seems to work fine, it’s just the display of the stack which is wrong.

8 Project Conclusion/Results

I think the result of this project was ok. I wish I had spent more time working on understanding the .cod files because I think that would have been very valuable. Even though it works and compiles, I still have some questions and wish I had taken the time to System.out.print every single line to truly understand every single interaction. However, I do have a pretty good idea and I am satisfied with my current understanding, but wish I had just a little more.