CSC 413 Project Documentation

Spring 2019

Bahar Moattar

Student ID:

920006396

Class.Section:

413.02-Spring2019

GitHub Repository Link:

https://github.com/csc413-02-spring2019/csc413-02-baharmoattar.git

Table of Contents

1 Introduction

1.1 Project Overview

This project required to make the necessary changes to the provided classes of ByteCodeLoader, Program, RuntimeStack, and Vitural Machine to make an interpreter for mocking language X.

1.2 Technical Overview

The interpreter class in this project is meant to process byte code using source code to work like a simplified java. It works with The Virtual Machine to execute the program computing the nth Fibonacci number and factorial of a number through recursion.

1.3 Summary of Work Completed

In the project we required to make additional classes that were stored in a bytecode folder. In this folder, we made classes for x-machine bytecodes of HALT, POP, FALSEBRANCH, GOTO, STORE, LOAD, ARGS, CALL, RETURN, BOP, READ, WRITE, LABEL, and DUMP. All these classes made are a sub class for an abstract class called ByteCode. They inherited abstract functions of init() and execute() from ByteCode.

2 Development Environment

1. Version of Java Used: 1.8.0-191-b12
2. IDE Used: IntelliJ

3 How to Build/Import your Project

T his project has to be uploaded form the GitHub. There is a link in iLearn provided by instructor to direct it to the GitHub and my personal private depository.

1. Click on that link and go to the assignment page in GitHub.
2. Click on “Clone or Download” and copy the path.
3. In terminal environment login to your git account and type “git clone “the path address you copied”
4. This will make a version of my project in my computer and terminal to have access to.
5. Open the IDE (for me it was IntelliJ). And click on “Import Project”.
6. The window will open to brows in your computer to find the project you just made through the git clone in terminal.
7. Click on that and “open”.
8. In the next screen pick “Create project from existing sources”, “Next”.
9. Do not change the project name in the next screen and just hit “Next” again.
10. In the next screen make sure the path of this project in your computer is checked and

click “Next”.

11. “Finish”
12. When the project is loaded and imported to IntelliJ, you will see the whole package and classes that’s been transferred to it.
13. Each time you make some changes in your program through this IDE, open your terminal environment, type “git add . -> git commit -m “some message” -> git push” so all the changes have been made to the program will be saved in GitHub.

4 How to Run your Project

1. Finish making the classes and editing all you need to edit.
2. After you made sure there is no syntax error, we will need to add the source files.
3. open the “edit configuration” and make sure it was the correct project listed under “Application”
4. next click on the “interpreter” under the “Application”
5. on the right hang side change the “Name” (on top) to “fib.x.cod” or “factorial.x.cod”
6. on the “Program argument” section, make sure you have the same name as the name you just changed in “Name” section.
7. click apply, ok.
8. Click on the interpreter.java class and run the program.

5 Assumption Made

Making an interpreter to execute a mocking language was a hard project for me to do. Coding more in Java, I assumed I would have a lot of research to do because this is one of my first few Java projects.

With that being said, this project will be implemented based on these assumption:

1. The visual basic programming language will make the program run on any computer
2. The project will ensure accurate processing in computer
3. The facts and figures are reliable, the designed software to run primarily.

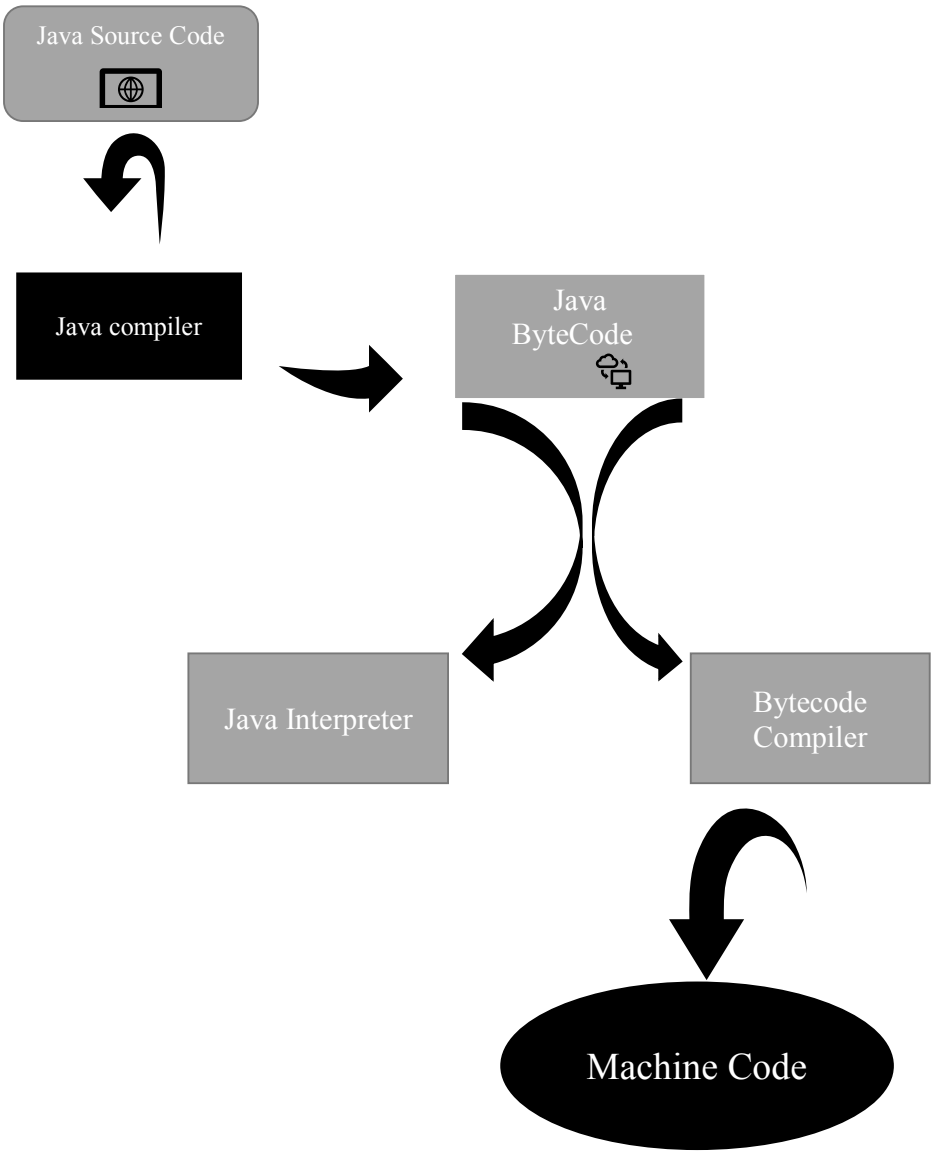
6 Implementation Discussion

6.1 Class Diagram

I tried to implement the program based off the pdf format instruction given to us. I tried not to change the program as little as possible. I added a about 16 sub classes to the abstract ByteCode class that would inherit the abstract methods of init() and execute() from it. Init() loads the bytecode and execute() executes the load in the virtual machine. I also had 4 other classes that needed to be completed, which were ByteCodeLoader, Program, Virtual Machine, and RunTimeStack. In ByteCodeLoader I was suppose to finish the method on loadCode(). I used the java.util.* so that I could use StringTokenizer what is built in Java class. I tried to combine it with readLine() so that it would read one line at a time and break that line into tokens. Using the tokens, I called method getClassName that is in the CodeTable class in order to match the tokens with the keys in the hashmap that's in CodeTable class. I tried dynamically load the bytecode and added newInstance() to create new instance of the class. In the program class, I created a new hashmap and made a method classed addCode to add the bytecodes to it. RunTimeStack records and process active frames. The methods within this class will pop frames off the stack or add frames onto the stack. Virtual Machine class is used to execute the program.

7 Project Reflection

The lines of codes that inspect other lines or classes to find any error. In this project the Interpreter Class is the class that's running all the other parts of the package, grab the information and return the arguments as a new mocked language.



8 Project Conclusion/Results

I'm not satisfied with the result I got from this program. When I run the program with fib.x.cod everything looks good and there is no error. But when I change it to factorial.x.cod and give the integer to the program, the answer I get is right but then it crashes right after. The error is coming from my PopCode() and prints this error message :

```
Exception in thread "main" java.lang.NumberFormatException: null
```

I will keep trying to figure out what is wrong.

So this is the new update for that class and the error I was getting. It looks like I was not doing my for loop right. The sign of the middle sentence instead of being i > was i <. And the loop was calling a wrong index number.

I think the result I'm getting now should be correct as far as both source files.