CSC 413 Project 2 Documentation

Semester 2024

Randy Chen

922525848

CSC413.02 Fall 2024

https://github.com/csc413-SFSU-FA2024/interpreter-Randychen10

Table of Contents

1 Introduction

In this assignment, we will be implementing an interpreter for the mock language X. This mock language can be thought of as a simplified version of java. Ther interpreter is responsible for processing byte codes that are created from the source code files with the extension of X. We will implement a virtual machine for this interpreter to work together to run a program written in X. We are given two sample programs, one being computing the nth number of a Fibonacci sequence and the other is finding the factorial of a number.

1.1 Project Overview

After looking at the files of this project, I see that we will be using parts of the pillars of object-oriented programming. This project requires encapsulation, inheritance, and abstraction. We need to create different byte code subclasses to help us translate this this language. We know that the byte code subclasses cannot access the run time stack directly, so how would we use the methods?

1.2 Technical Overview

The way I see it, it looks like we should start off by creating the runtime stack. Here we have to create multiple methods from the Java stack interface. Next, we want to create the ByteCodeLoader class which is responsible for loading bytecodes from the source code file into a data structure that stores the the entire program. After that, we want to create the CodeTable class, which is where we create a map and store all of our bytecodes inside. This will be used later on.

How would we store the bytecodes? Well we need to create a class that does it for us. We will call this the Program class. In this class, we create an ArrayList that can store all the bytecodes and ensure we can only bytecodes into it. After we create our CodeTable and Program classes, we have to start creating all the bytecode subclasses, which are in charge of the translation process.

We have the main ByteCode class, which stores all the abstract methods required for its subclasses. After that we create all the subclasses and extend it to bytecode so that we can modify the subclasses.

1.3 Summary of Work Completed

Everything that I have done for this assignment has been listed above, I followed step by step how I completed everything in 1.2 technical overview.

2 Development Environment

This project was made using the latest version of IntelliJ Idea Ultimate and JDK 22

3 How to Build/Import your Project

1. Open windows command prompt
2. (assuming you already have a CSC413 folder created for your calculator project) navigate to your CSC413 project directory using cd
3. Git Clone “project directory copied from github”
4. Open IntelliJ and load up a file
5. On the top right, look for “File”
6. Hover over File and press open
7. Look for your project in its directory and press open

4 How to Run your Project

1. Open IntelliJ
2. Go to interpreter.java
3. Look for the current file dropdown menu next to the run button
4. Click on the drop down menu, and hover over current file
5. Click on the three vertical dots that appear and select run with parameters
6. Enter the cod file you want to run in the program arguments box
7. Switch to that configuration and press run

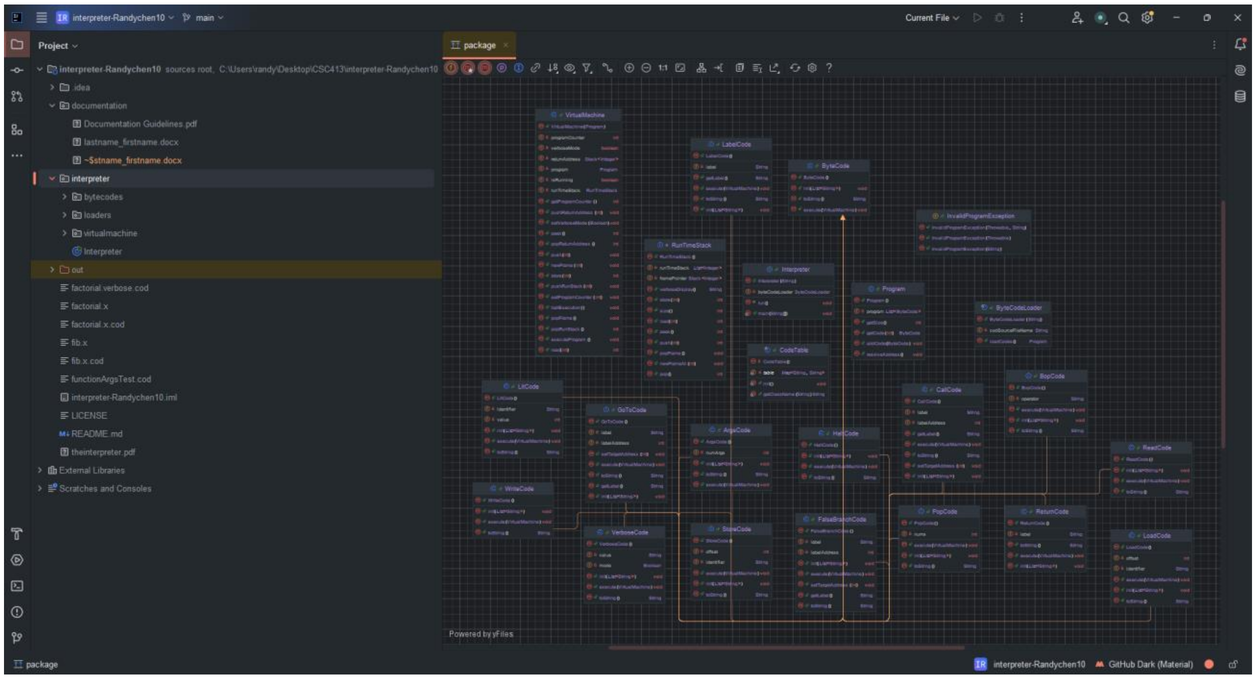
5 Assumption Made

I assumed this assignment was going to a hard and stressful project. Little did I know, we did most of it during class and it was a lot less that I expected.

6 Implementation Discussion

I started with creating the RunTimeStack because we were given a not so deadline deadline to finish it before the next class. After the RTS I went to work on the ByteCodeLoader since I was just follow whatever we got done in class. I then went to the CodeTable, which was really easy, I just had to add all the bytecode subclasses into a hashmap. Then I created the Program class which we did part of in class as well. I just finished it the day after we started it. Finally I did the bytecode subclasses and Virtual Machine at the same time because we could not call anything from the subclasses to the RTS, we had to go throught the VM which I created methods that directly called the methods from the RTS.

6.1 Class Diagram



7 Project Reflection

After spending about 3 weeks working on this project, I am officially done. The biggest struggle about this project was the 15 different bytecode subclasses we had to code. Even though there was not really much code to go into them, a couple of the subclasses constantly gave me issues and constantly threw errors at me. I tried to do everything I could, but unfortunately there are still bugs in my program.

8 Project Conclusion/Results

Now that I am finished with the project, I can say that it compiles and that there might be a few bugs that I never got to fix, but I am done.