CSC 413 Project Documentation

Summer 2021

Anmol Burmy

921143454

CSC 0413-01

GitHub Repository Link -

https://github.com/csc413-su21/csc413-p2-Burmy

Table of Contents

1 Introduction

1.1 Project Overview

This Java project is an Interpreter that reads and processes a mock language X. This interpreter uses a Virtual Machine, and uses a set of bytecodes to read the x.cod files and perform the necessary operations that interpret the x code.

1.2 Technical Overview

This Java project is an Interpreter that reads and processes a mock language X. You can think of the mock language X as a simplified version of Java. The interpreter is responsible for processing byte codes that are created from the source code files with the extension x. The interpreter and the Virtual Machine will work together to run a program written in the Language X. The two sample programs are a recursive version of computing the nth Fibonacci number and recursively finding the factorial of a number. These files have the extension x.cod.

1.3 Summary of Work Completed

- Implemented all the ByteCode classes listed in the Supported ByteCodes. And made sure to create the correct abstractions for the ByteCodes.
- Completed the implementation of ByteCodeLoader class, which loads the byte codes from the files into an array list.
- Completed the implementation of Program class, which is responsible for storing all the ByteCodes read from the source file.
- Completed the implementation of RuntimeStack class, which is responsible for recording and processing the stack of active frames.
- Completed the implementation of VirtualMachine class, which is responsible for executing the given program.

2 Development Environment

This project was developed using IntelliJ IDEA with Java version 16.0.1.

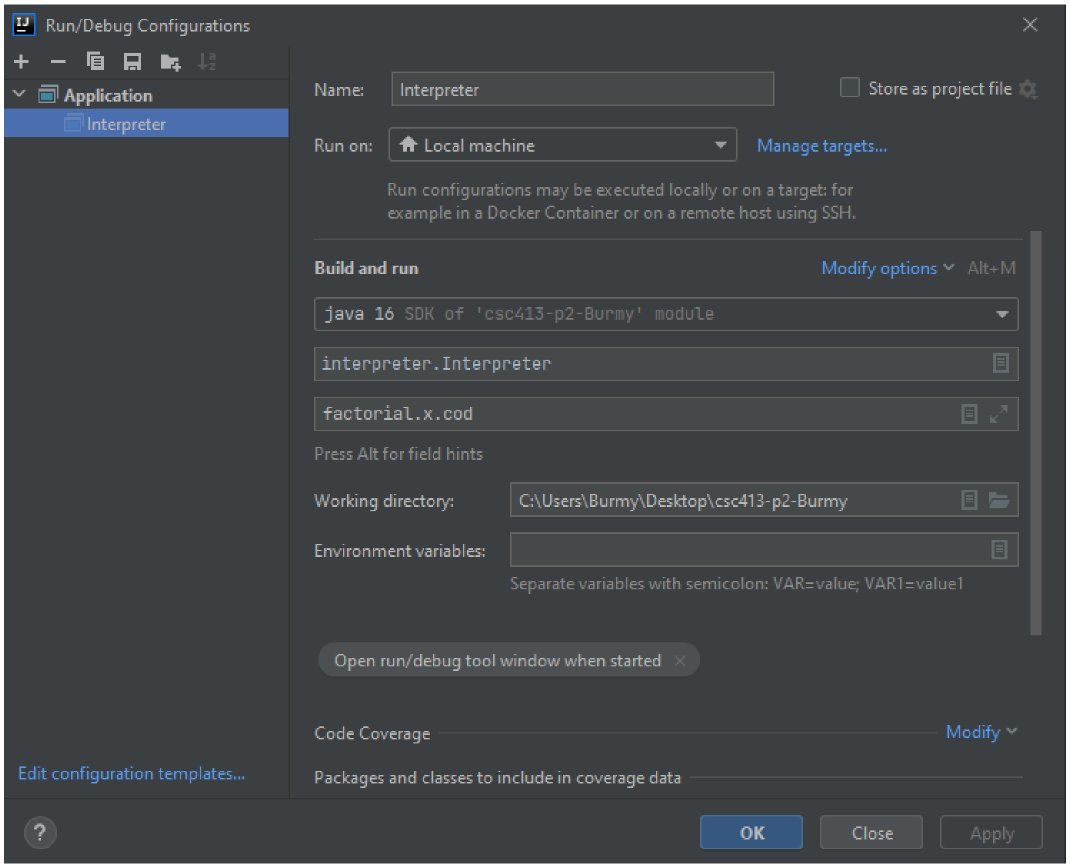
3 How to Build/Import your Project

To build/import the project, go to my repo and click the green “Clone or Download” button on your repo’s home page. Then select either HTTPS or SSH, If you are not sure what SSH keys are then use the HTTPS method. Copy the link and then open your terminal. This will either be Git Bash on Windows, Terminal on Mac or Linux, or the terminal for your Windows Linux Subsystem if you have it installed. In your terminal type: git clone repo_url_you_copied. Once the repo is cloned, follow the steps below to import your project into IntelliJ IDEA.

1. Select Import Project
2. Select the “csc413-p2-Burmy” folder as the source root of your project.
3. Keep the “Create project from existing resources” radio button selected.
4. Now keep clicking “NEXT” until the import is finished.

4 How to Run your Project

When executing this project you will need to create the run configurations and set the command line arguments. The Interpreter only is able to handle the .x.cod files. A sample picture of a run configuration is given below:



Name: Interpreter
Store as project file
Run on: Local machine
Run configurations may be executed locally or on a target: for example in a Docker Container or on a remote host using SSH.
Build and run
java 16 SDK of 'csc413-p2-Burmy' module
interpreter.Interpreter
factorial.x.cod
Press Alt for field hints
Working directory: C:\Users\Burmy\Desktop\csc413-p2-Burmy
Environment variables:
Separate variables with semicolon: VAR=value; VAR1=value1
Open run/debug tool window when started
Code Coverage
Packages and classes to include in coverage data

After you have filled in the required configuration, you can just right click on the “Interpreter” class and click “Run Interpreter” from the dropdown menu. Or you can also select/open the class in the window and press the play button on top of the editor window.

5 Assumption Made

I assumed that you had to know and understand the algorithms of factorials and fibonacci sequences.

6 Implementation Discussion

For this project, we had to load in the test code in order for it to run as a program in the virtual machine. There is a ByteCode abstract class and from this many concrete classes were created for each type of code. These byte codes were loaded in and saved as their respective bytecode object.

We have a ByteCodeLoader class. This class is responsible for loading bytes from the source code file into a data structure that stores the entire program. The ArrayList was used to store bytecodes. The ArrayList was contained inside of a Program Object. Adding and Getting bytecodes will go through the Program class.

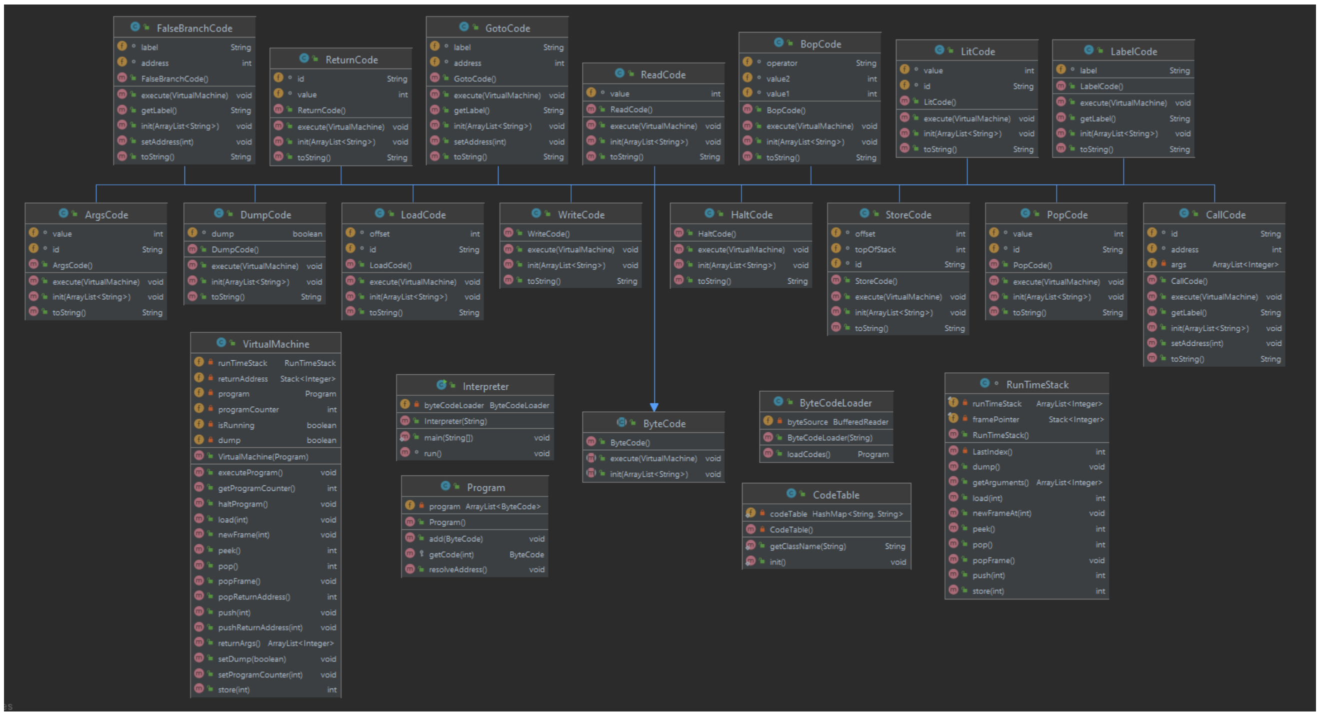
The Program class is responsible for storing all the bytecodes read from the source file. ByteCodes are stored in an ArrayList which has a designated type of ByteCode. It is to ensure only ByteCode, and its subclass can only be added to ArrayList.

We also have a RunTimeStack class. This class is responsible for recording and processing the stack of active frames. This class contains two data structures used to help the VirtualMachine Class execute the program. These data structures are maintained private and are - Stack Frame Pointer, which is used to record the beginning of each activation record (frame) when calling functions and ArrayList runStack, which is used to represent the runtime stack. It will be an ArrayList because we will need to access all locations of the runtime stack.

Finally, There is a VirtualMachine class. This class is used for executing the given program. The VirtualMachine is the controller of all the programs. All operations need to go through this class.

The Interpreter class and the CodeTable class were provided with implementation. The Interpreter class is the entry point to the Interpreter project. The ByteCodeLoader class uses the CodeTable class. It merely stores a HashMap which allows us to have a mapping between bytecodes as they appear in the source code and their respective classes in the Interpreter project.

6.1 Class Diagram



7 Project Reflection

Going into this assignment I already assumed this project was going to be much more difficult than the first one as the Professor had mentioned in the videos. I couldn't figure out what I was supposed to do or even what files I was supposed to edit. So first I tried to understand the assignment because I feel like understanding how the program should work beforehand eases the process as we know what to expect. And after reading through project instructions, the videos provided by the professor and some help from my classmates in discord, made me understand the assignment very well.

In addition, I also spent a lot of time getting my output as close as the one provided in the instructions of the assignment. There were a lot of things to look at even if you got the program working.

8 Project Conclusion/Results

I think this project was fairly difficult and needed a lot of time to be built and precise with the instructions that were provided. After having some difficulty on how to actually start the assignment, I was able to finish and create an interpreter that can process the language X. I tried to make the code as readable and understandable as possible by adding comments and making the code clean, so people or even I can understand what the program does in the future.