CSC 413 Project Documentation

Spring 2021

Daryl Stronge

917844673

413.02

https://github.com/csc413-sp21/csc413-p2-DSnoNintendo

Table of Contents

1. Introduction

For my first project for CSC 413, I created an interpreter that can read and execute a low-level programming language called Bytecode.

1.1. Project Overview

The program reads text files containing Bytecode and executes each line, one by one. The two algorithms executed for this project were calculating factorials and calculating fibonacci sequences

1.2. Technical Overview

The program was made in Java 12 using object-oriented programming

1.3. Summary of Work Completed

To complete the project, I had to alter starter code, write new algorithms, and create 10+ new classes to execute each line of Bytecode.

2. Development Environment

The program was developed using Java in the IntelliJ Idea IDE

3. How to Build/Import your Project

1. Open terminal.

2. Make sure Git is installed by entering “Git” in terminal. If there is no error, Git is installed.

3. Make sure Java is installed by entering command “Java --version” in terminal. The program has been tested for Java 12.

4. enter command: “clone https://github.com/csc413-sp21/csc413-p2-DSnoNintendo.git”

4. How to Run your Project

1. Open cloned folder in a Jave IDE (IntelliJ, Eclipse, etc.)

2. Navigate to interpreter/Interpreter.Java

3. Configure the Interpreter class to take“fibonacci.x.cod” or “factorial.x.cod” as the argument for the “main” method. You can also write your own Bytecode program (see “theinterpreter.pdf”).

4. Run the program

5. Assumptions Made

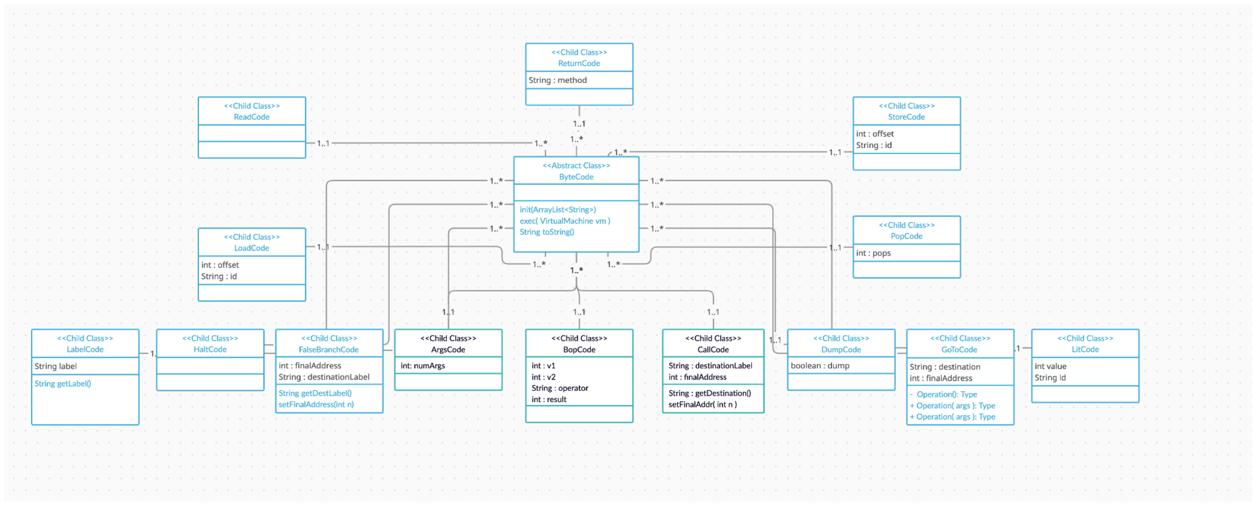
- You understand factorials and fibonacci sequences

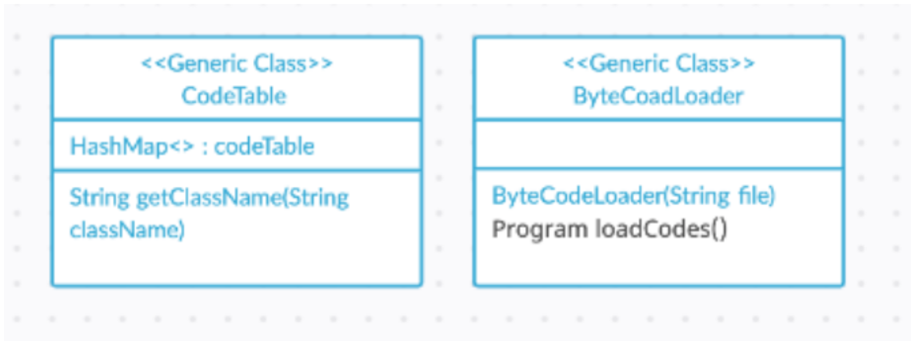
- You know how to configure arguments for the main method of a Java class

6. Implementation Discussion

The project was implemented using multiple objects and classes. An abstract class for Bytecode was created and a child class was made for each Bytecode. I wrote algorithms to execute the Bytecode, dump the runtime stack, and load each Bytecode to be executed in the order it was written in text files.

6.1. Class Diagram





7. Project Reflection

This was probably the hardest assignment I've done since I started studying computer science, but at the same time, it taught me a lot. Writing this program helped me understand how abstracted our traditional programming languages are. It also helped understand how runtime stacks operate.

8. Project Conclusion/Results

After days of writing and debugging algorithms I was able to successfully execute the Bytecode in the test files that were given to me to calculate fibonacci sequences and factorials. This was a very valuable learning lesson that will help me write my future programs more elegantly in future.