

CSC 413 Project 2 Documentation
Summer 2023

Juan Estrada

923058731

CSC-0413-01-Summer-2023-R4

[https://github.com/csc413-SFSU-
Souza/csc413-p2-jjestrada2](https://github.com/csc413-SFSU-Souza/csc413-p2-jjestrada2)

Table of Contents

1	Introduction	3
1.1	Project Overview.....	3
1.2	Technical Overview	3
1.3	Summary of Work Completed	3
2	Development Environment.....	3
3	How to Build/Import your Project	4
4	How to Run your Project.....	4
5	Assumption Made.....	5
6	Implementation Discussion	5
6.1	Class Diagram.....	5
7	Project Reflection.....	9
8	Project Conclusion/Results	9

1 Introduction

1.1 Project Overview

The project is an interpreter for a programming language. An interpreter is a program that reads and executes source code written in a specific programming language. In this case, the interpreter is designed to execute bytecode instructions, which are a low-level representation of the source code. The interpreter simulates the behavior of a virtual machine and performs operations based on the bytecode instructions. The project aims to provide a means of executing programs written in this programming language without the need for a traditional compiler.

1.2 Technical Overview

The interpreter is implemented using Java programming language. It consists of several classes that work together to load, execute, and interpret bytecode instructions. The main components of the interpreter include:

ByteCodeLoader: This class is responsible for loading the bytecode instructions from a file. It reads the file line by line, tokenizes each line, and creates instances of the corresponding bytecode classes based on the opcode.

Program: This class represents the program to be executed. It stores a list of bytecode instructions in the order they appear in the source code. It also provides methods to access and manipulate the bytecode instructions.

VirtualMachine: This class simulates the behavior of a virtual machine. It has a runtime stack to store values during program execution. It executes the bytecode instructions one by one, performing the required operations based on the opcode. It also handles control flow instructions like branching and function calls.

ByteCode (and its subclasses): The ByteCode class is an abstract class that defines the common structure and methods for all bytecode instructions. Each specific bytecode instruction (e.g., ArgsCode, CallCode, BopCode) extends the ByteCode class and provides its own implementation of the load and execute methods.

1.3 Summary of Work Completed

In the project, I contributed to the overall structure of the interpreter and implemented the ByteCodeLoader class, which is responsible for loading bytecode instructions from a file. I also worked on the VirtualMachine class, implementing the execution of bytecode instructions and handling control flow. Additionally, I contributed to the implementation of the bytecode instructions.

2 Development Environment

a) Java 20.0.1

b) IntelliJ

3 How to Build/Import your Project

1. In your terminal type: `git clone https://github.com/csc413-SFSU-Souza/csc413-p2-jjestrada2`

Please clone your repo to a folder on your computer that does not require elevated privileges.

2. Open your preferred IDE (Integrated Development Environment) such as Eclipse, IntelliJ IDEA, or NetBeans.

3. Select import project.

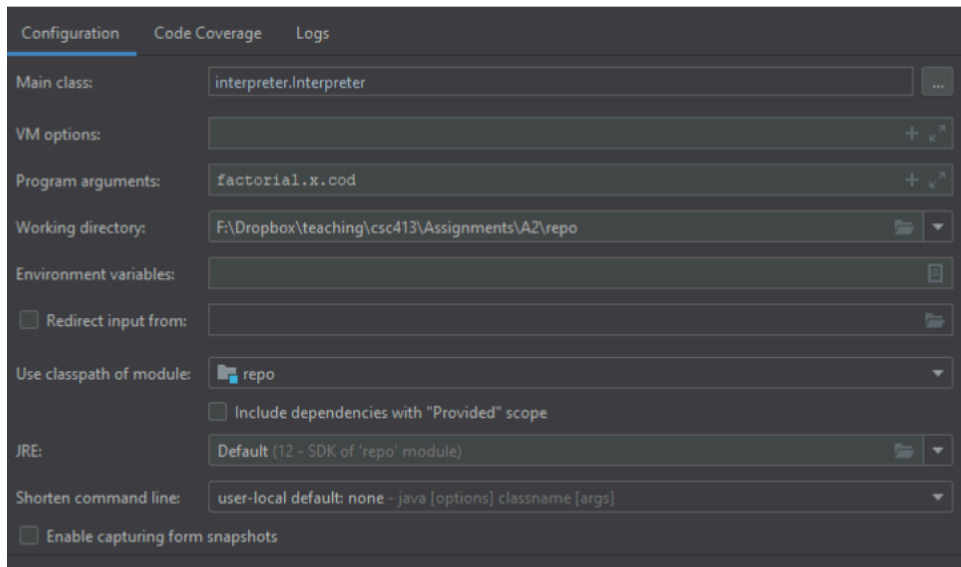
4. When importing the interpreter project you will use the root of your repository as the source. DO NOT USE the interpreter folder as the root. This will cause the project to not work and in the end force you to change the file structure.

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. The `.x` are not going to be used for this project, and are present only as a reference.

A sample picture of a run configuration is given below:

Note that the working directory and the Use Classpath of module values may not be the same. The purpose of the image it show where the filename of the ByteCode source file goes.



5 Assumption Made

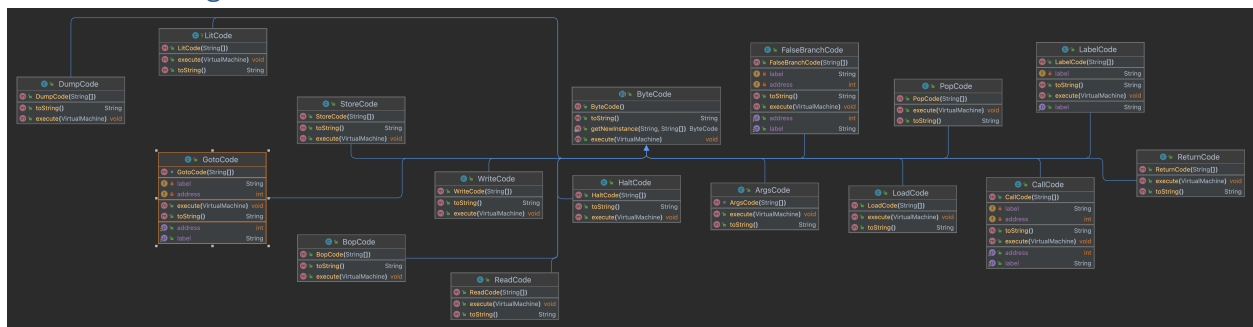
Initially, I assumed that all the code provided by the professor was accurate and couldn't be modified. However, as I delved deeper into the project, I came to the realization that my assumption was only partially correct.

6 Implementation Discussion

Object-Oriented Design: The interpreter was implemented using an object-oriented design approach. This allowed for modular and encapsulated code, with each bytecode instruction being represented by a separate class. This design choice promotes code reusability, maintainability, and readability.

Abstract ByteCode Class: The use of an abstract ByteCode class provided a common interface and structure for all bytecode instructions. It enforced the implementation of necessary methods such as load and execute in each bytecode subclass. This design facilitated easy addition of new bytecode instructions in the future.

6.1 Class Diagram



InvalidProgramException

  InvalidProgramException(Throwable, String)

  InvalidProgramException(String)

  InvalidProgramException(Throwable)

ByteCodeLoader

  ByteCodeLoader(String)



  loadCodes() Program

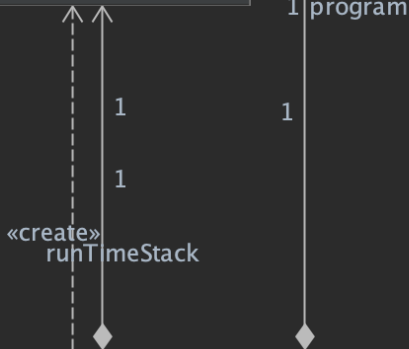
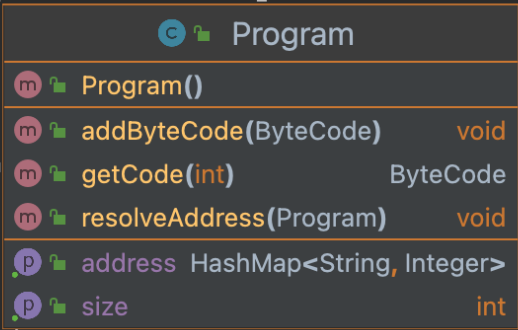
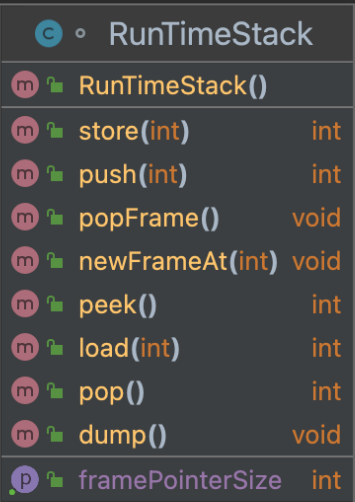
ByteCode

  ByteCode()

  toString() String

   getNewInstance(String, String[]) ByteCode

  execute(VirtualMachine) void



7 Project Reflection

Lessons Learned:

- I successfully implemented the core functionality of the interpreter, including the bytecode loader, virtual machine execution, and handling of various bytecode instructions.
- Effective planning and task allocation are crucial for successful project execution.
- Thorough testing and debugging are essential to identify and resolve issues early in the development process. This allowed me to achieve a more robust and reliable interpreter.

8 Project Conclusion/Results

In conclusion, the project successfully delivered a functional interpreter capable of executing bytecode instructions. I consider the project a valuable learning experience. I achieved my primary objectives and gained insights into the implementation of an interpreter, addressing challenges, and making significant progress.