CSC 413 Project 2 Documentation

Summer 2023

Pedro Grande

921149265

CSC 413.01

https://github.com/csc413-SFSU-Souza/csc413-p2-PFGrande/tree/main

Table of Contents

1 Introduction

1.1 Project Overview

The Interpreter application is programmed to interpret a simple programming language. It runs programs written in .cod files and it has the ability to read and write to the terminal.

1.2 Technical Overview

The Interpreter parses .cod files to create instances of ByteCode classes. The ByteCodes are used to execute commands and manipulate the Run Time Stack. The Run Time Stack stores numerical values declared by the programmer writing the .cod files. The Virtual Machine class prevents the ByteCodes from directly manipulating the run time stack. The virtual machine class is the class that executes the ByteCodes in cod files. The virtual machine class includes functions used by the ByteCodes to manipulate the RunTimeStack and to check if their execution is viable.

The Program class stores the ByteCodes in a list for the virtual machine to cycle through. The program counter in the virtual machine class is the index for which a ByteCode is executed. A method in the class resolves the addresses for labels declared in cod program files. These labels are used as keys in a hashmap pointing to an index in the program's list. When jumping, the virtual machine's class program counter is set to the value corresponding to the label. Another way to jump in the program is using the return stack.

The virtual machine contains the return stack, it used to store the address from which a function was called from. When the CALL ByteCode is executed, the Virtual machine stores the value of the program counter in the return stack, and then sets it to the value returned by the hashmap.

Through the virtual machine, ByteCodes can create new frame pointers, update the return address, check the frame boundary, and push and pop values from the frame stack.

There are ByteCodes that are not allowed to operate across frame boundaries. The virtual machine is able to check the size of a frame and determines whether a ByteCode is modifying values outside its scope. Frame pointers serve as scopes. The frame pointer stack stores the starting position of a frame to isolate function arguments from other variables in the program.

1.3 Summary of Work Completed

A majority of the ByteCodes function as intended and can be printed in a readable format. I was unable to complete the Dump function. This function would have output ByteCodes during their execution.

2 Development Environment

a. Java 14.0.2
b. IntelliJ IDEA

3 How to Build/Import your Project

4 How to Run your Project

a. Put name of cod file as argument for program
b. Use Interpreter.java as main class
c. Build and run

5 Assumption Made

6 Implementation Discussion

6.1 Class Diagram

7 Project Reflection

8 Project Conclusion/Results