CSC 413 Project Documentation

Spring 2019

Kevin Fung

915857298

413.03

https://github.com/csc415-03-spring2019/csc413-p2-kefung2

Introduction

This project is the second assignment from CSC 413 which we will need do a translator from a list of computer language for the computer to read. In this case it would be the make up language X. Two of the file are done for us, and we have to finish the rest.

In the project we have to create a file for each of the bytecode, and implement their function. We also have to implement the ByteCodeLoader class, which read the .cod file. The other two file are the RunTimeStack and the VirtualMechine class, which act as a middle man between the interpreter class and the byte codes. Also, we are not allow to break the encapsulation of the class.

Development Environment

This project is done IntelliJ Ultimate Edition, with Language level 8 and SDK 1.8.

Link:

https://www.jetbrains.com/idea/download/#section=windows

How to Build or Import and Run the Project

To get this project, you first go to the github link, and use the SSH link to clone the repo, or you can download the file it self, then to import this project, you press import project and choice the file that say csc413-p2-username as the root of the project. To run this project go to run and then edit config. Under the program argument area, type either fib.x.cod or factorial.x.cod, then press ok. Then you can run the Interpreter class to start the program.

Assumptions Made when designing and Implementing you Project

Some assumptions I made was I think I need a good amount of knowledge for computer language for the project, because the bytecode are something that was talk about in CSC 256 class, so I think that might help me out on this project. I also know that this project is going to be hard and complicated, so I plan to start as early as I can.

Implementation Discussion

In this project we are suggested to start by creating all the bytecode, and leave them empty until the end. Next was the ByteCodeLoader class, which load the .cod file and read the line from it. From the last project I know that I should use a tokenizer for read the text, and it can be use as a loop condition until the end of the file. When it ask us to create a new instance, I was not sure how to do it, so I have to look up online, and later on, the professor told us that there is a code in the pdf that show us how to do it, and I change my code to it afterward. Another problem I ran in to when I was coding this part was it did not exit the loop and it keep storing the same value whenever loadcode was read. Soon I notice that I never set a ending condition for the loop to end, and I need to clear the argument list everytime to loop start again. The next class that we need to code is the Program class, and mainly the resolveAddrs function. I understand that if the byte code is wither goto, false branch or call code, it will look for it label, however I could not think of a way to do it, after some tries.

Next is the RunTimeStack class, which contain function that we use for RTS like pop, peek, ect. On the pdf there are all the function header given to us, and we just have to implement it. Some of the function was easy to understand and some is harder. I code the function that I am familiar with first, and then for the one that I do not know I try to follow the description on the

pdf and try to implement it. The next class that we need to implement would be the VirtualMachine class. On the pdf there is a base code for the function execute for this class, so I copy it over, and see what other function I need to add to this class. I notice that all of the variable in this class are private, so I know I have to make some getter and setter function for them, so i can have some access to make changes to the variable. The professor also mention that when we turn in the project the dumping should be set to OFF and turn ON when it see the dump code to turn it on, so I added a new data field call isDumping, and put the comment out line inside a if statement, with isDumping as it argument. And I make a setter function for isDumping, so when the dump code is call, it will change isDumping to ON.

The last thing that we have to implement are the bytecode files itself. First thing I have to do was to create a abstract class for the bytecode, and have all the other subclass extend from this one class, and after that I created a subclass for each bytecode, and begin implementing each of them. The bytecode subclasses are:

HALT:

Which stop the program, when it execute, I did this by changing the while loop condition isRunning in the VirtualMachine class from true to false, to stop the program.

POP:

Which pop the top n level of the RTS. I did this by using a for loop to pop the stack n times.

FALSEBRANCH:

Which pop the top of the stack.

GOTO:

Which go to the label.

STORE:

Which pop the top value from the stack, and store it into the frame stack, and use as a comment.

LOAD:

Which push the nth frame onto the top of the RTS.

LIT:

Which push a value to the RTS, if there is a variable name with it, it initialize 0 and put the name to the RTS, and the name will equal to the value is initialize with which is 0.

ARGS:

Which set up a frame nth place down from the top of the RTS.

CALL:

Which transfer control to another bytecode.

RETURN:

Which return from the current function back to the previous place where we call this.

BOP:

Which pop the top 2 value of the stack and perform a math or logic operation on them. I did this with a switch statement.

READ:

Which read a input from the user. This can be done by using a Scanner class.

WRITE:

Which peek the value on the top of the stack to the output.

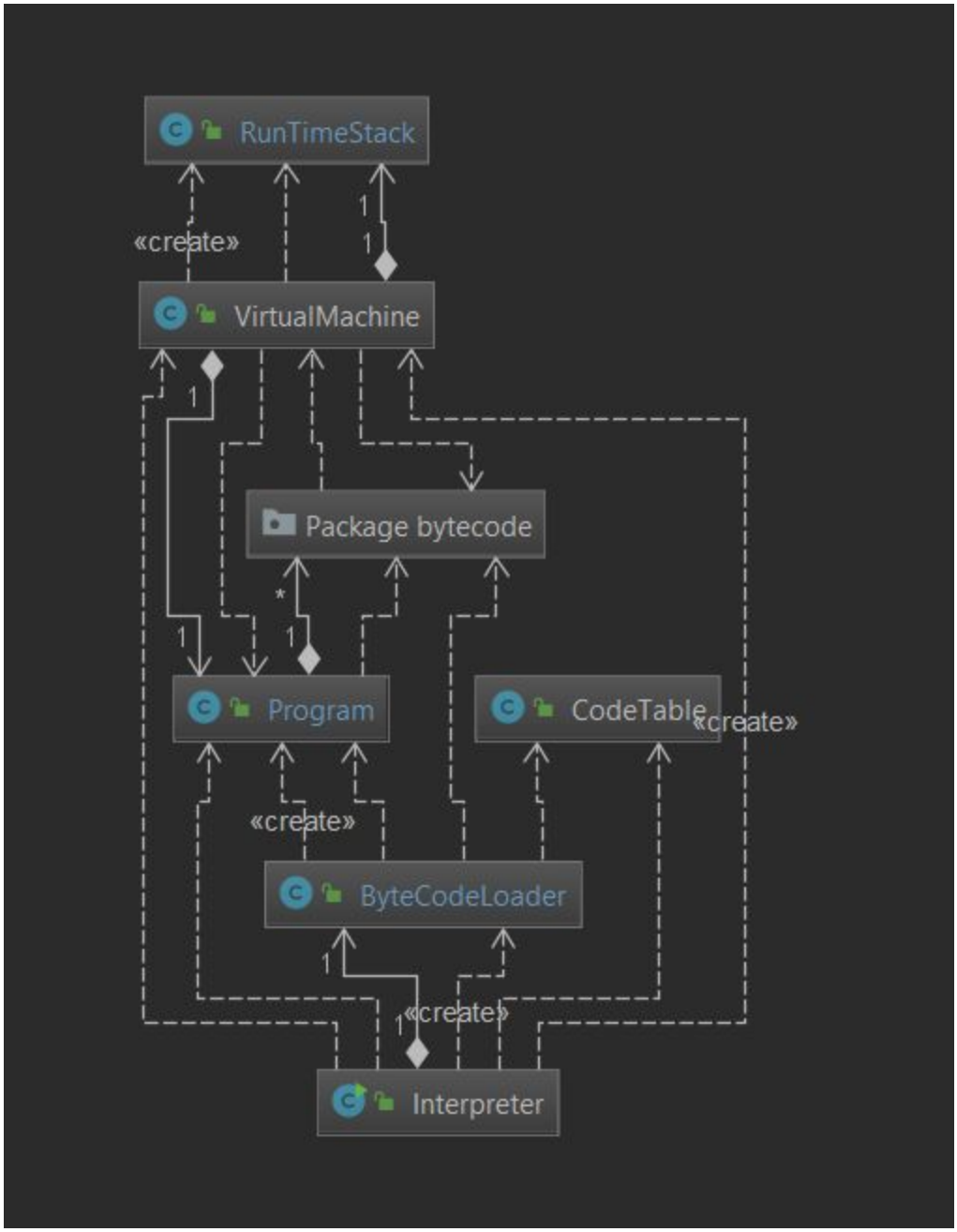
LABEL:

Which label 3 branches(falsebrance, goto, call), and we can jump to that location easier.

DUMP:

Which set the variable in VirtualMachine isDumping, to true or false, to control if the program will dump or not. I did this with a setter function to control the Dumping

Here is a diagram that shows how this project is connected together:



As you can see from this diagram, the Interpreter class, act as a starting point, then call ByteCodeLoader class to read the .cod file, and while it is running, it will look up the name of the bytecode from the CodeTable class, then it create a new instance from it, it then passes the instance it made to the Program Class. The Program will then go to the bytecode folder to gain

information about what each bytecode is suppose to do, and in the VirtualMachine class it will run those code, and gain information from the RunTimeStack class to to get a right output for the result.

Project Reflection

This project was a struggle for me. When I got the assignment first thing I did was read over the the pdf, and try to get some understanding of the project, and I tried to start implement some of the function, but I couldn't due to other class work, so I end up starting a little later than I planned. When I had problem, I would look it up on slack, or ask my friends for help. This was helpful to some extent but it could not help me finish the project. I was hoping that my friends and reading off slack was enough to help me on some problem, but for the next project, instead of just reading every comment on slack and see if anyone have the same problem a me, or just asking my friends, I should just ask for help on slack, this way it might be more helpful.

Project Conclusion and Results

This project overall went horrible and it will not run correctly. There was not enough time to work on the project, and I still have to save some time for my other classes work. I did my best to work on the project whenever it is possible, but there was not enough for me, since I spended a lot of time looking for stuff online that can help with the project. I understand what some function should do, but I was not sure how to implement them, so I just try to implement it by following the word on the pdf, and see if it make sense or not.. I did everything I can, but it only able to run up until it ask me for a input, after I put in a number, it will just give me an empty stack error, so I added a if statement, to make it exit the program instead of crashing. In

the end, I think this project is challenging, and I was not able to complete it within the given time.