Assignment 2 – The Interpreter

CSC 413 Project Documentation

Summer 2020

Name - Rajdeep Singh

ID: 920258882

CSC 413-02 Summer 2020

GitHub Repository Link:

https://github.com/csc413-02-SU2020/csc413-p2-Rajdeep0303.git

Table of Contents

Introduction

Project Overview

This project is an interpreter program for the mock language X. The main goal of the program is to interprets byte codes of programs written in mock language X. Language X is a made up programming language. Language X comes with an interpreter, a compiler and a debugger. This program reads a file and performs specific functions. The user enter the integer value to run the program and program run the stack value that separated by functions and work to calculate the final result.

Technical Overview

This program was design, verify, and implement the functionality of an interpreter. The interpreter functions on a programming language knows as Language X. The two main files the interpreter and the virtual machine work together to run a program written in the mock language X. The goal of this project was to build an interpreter for a language. The interprets byte code of programs written in mock language X. The user will enter the integer value to run the program and the program will read this value into a stack. The run the stack value that separated by functions and work to calculate the final result. The project structure was provided, we have to work with files which already implemented and creating abstract class Bytecode, extended and implemented by the different files. The interpreter is an assembly like interpreter that run and processes bytecode as it runs.

Summary of Work Completed

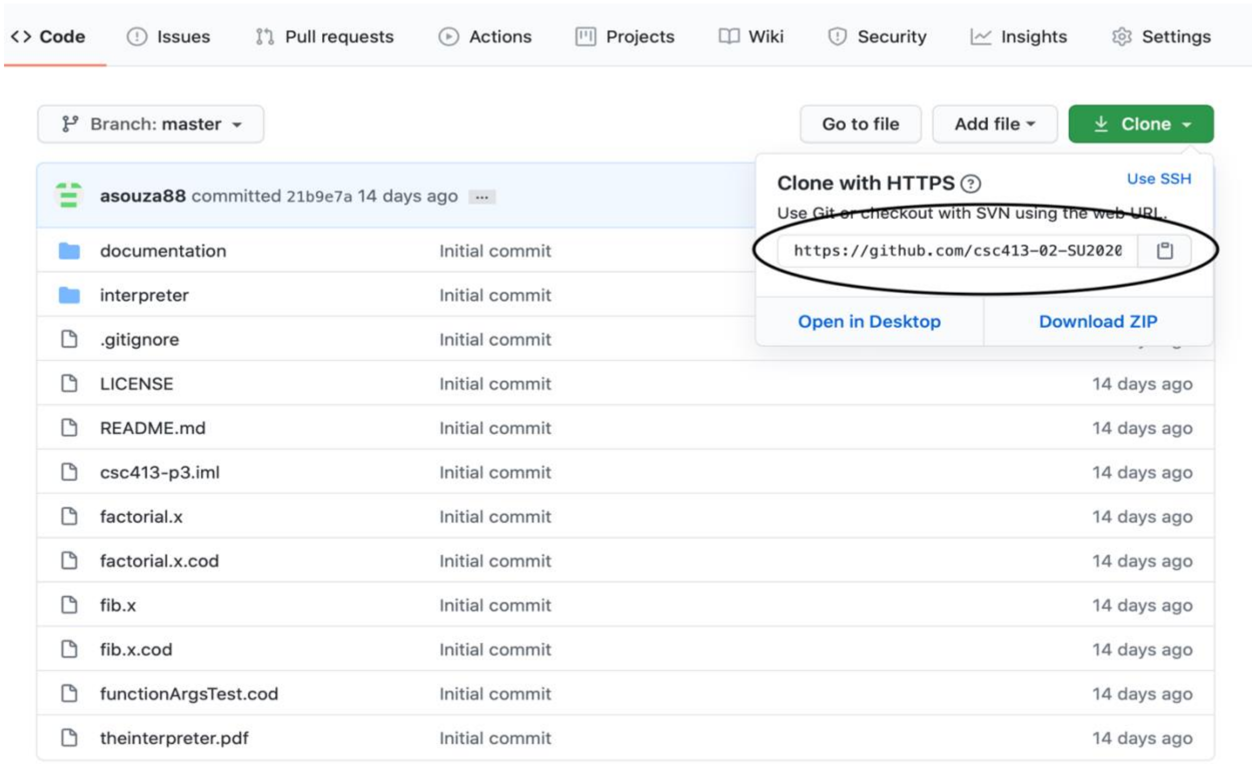
For this project, I created 16 subclasses that extends byte codes folder. I created the all ByteCode classes and leaving them empty. After, creating the all ByteCode classes, I start implement the Program, RunTimeStack, ByteCodeLoader, and VirtualMachine classes and in end, I start implement the empty subclasses. I implemented “Program” class for storing all the byte codes, and implemented the “RunTimeStack” class for processing the stack of active frames. Implemented the “Virtual Machine” class to do all the work to run the methods. Implemented the ByteCodeLoader class to parse the program argument file into ByteCode objects and their arguments. Also, created an interface class called AddressLabel which will be implemented by a select few of the ByteCode.

Development Environment

For this project, I use IntelliJ IDEA Ultimate to compile it, java version 2020.1.2.
Java version 13.0.2 was used to develop this project in IntelliJ IDEA was used to develop this project.

How to Build/Import your Project

To be able to build the project in IntelliJ IDEA Ultimate, go to GitHub link where you can see the repository is located. Copy the link to import, or you can download the zip file containing the whole project.

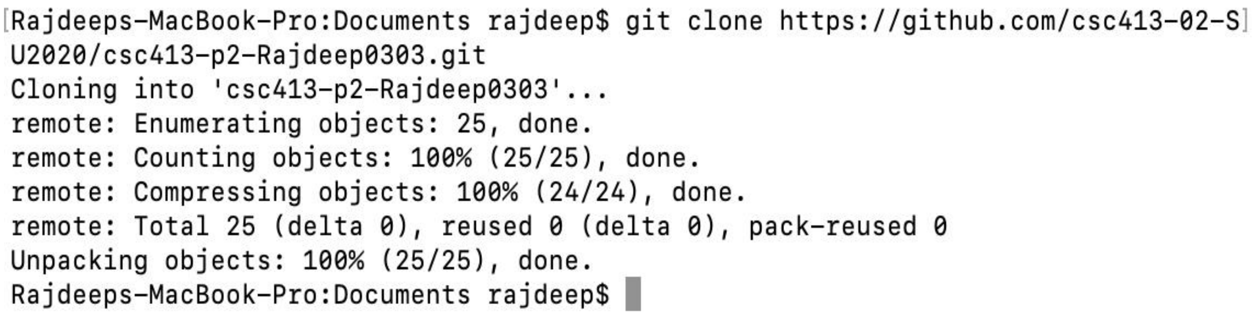


asouza88 committed 21b9e7a 14 days ago

documentation	Initial commit	
interpreter	Initial commit	
.gitignore	Initial commit	
LICENSE	Initial commit	14 days ago
README.md	Initial commit	14 days ago
csc413-p3.iml	Initial commit	14 days ago
factorial.x	Initial commit	14 days ago
factorial.x.cod	Initial commit	14 days ago
fib.x	Initial commit	14 days ago
fib.x.cod	Initial commit	14 days ago
functionArgsTest.cod	Initial commit	14 days ago
theinterpreter.pdf	Initial commit	14 days ago

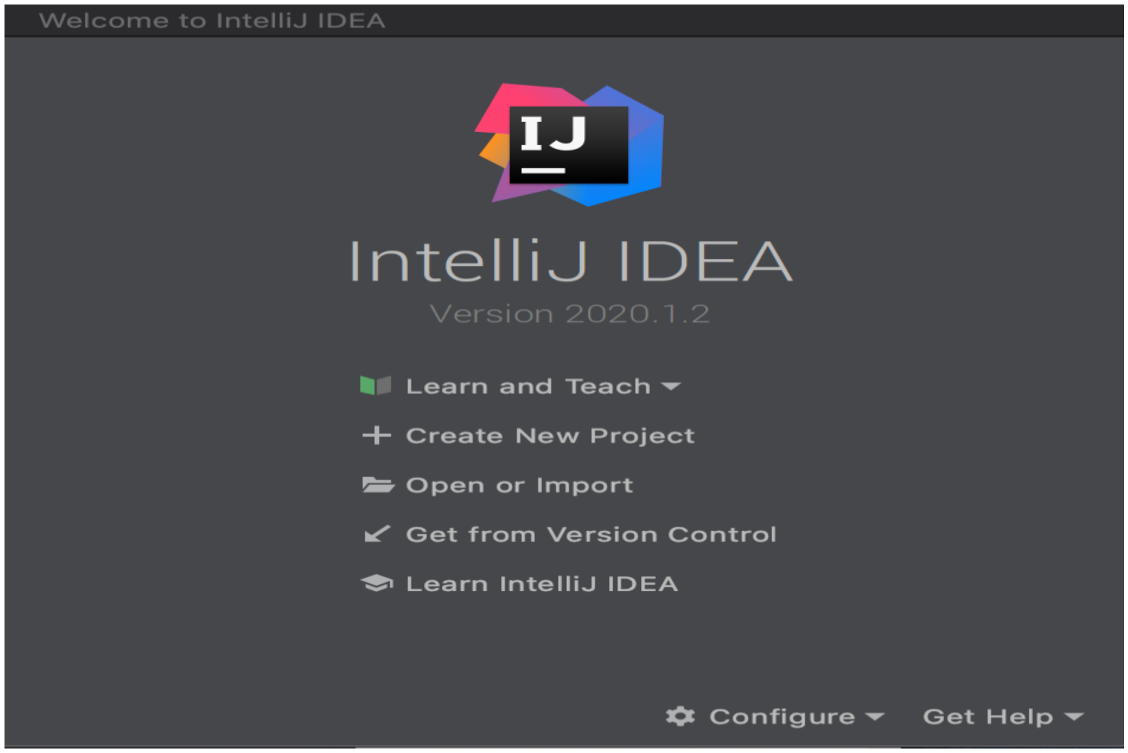
Click on "Clone or Download" button. Then select the HTTPS or SSH if you have an SSH keys set up.

Then, copy the link in the terminal. For example, git clone repo_link_here

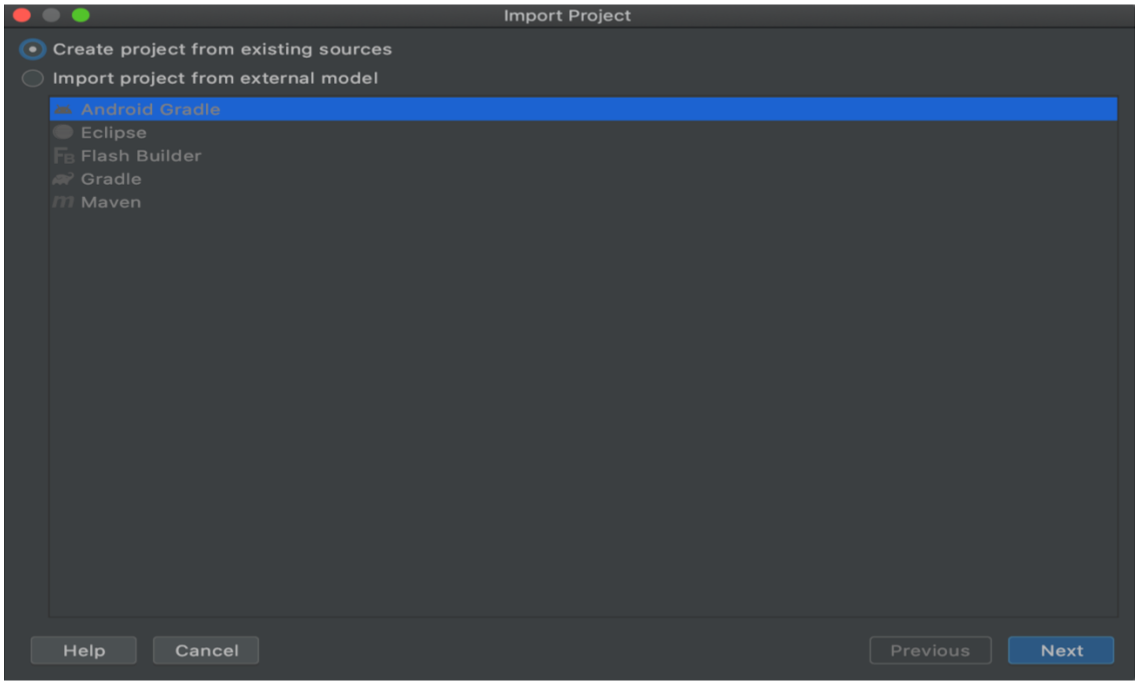


```
Rajdeeps-MacBook-Pro:Documents rajdeep$ git clone https://github.com/csc413-02-S
U2020/csc413-p2-Rajdeep0303.git
Cloning into 'csc413-p2-Rajdeep0303'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 25 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (25/25), done.
Rajdeeps-MacBook-Pro:Documents rajdeep$
```

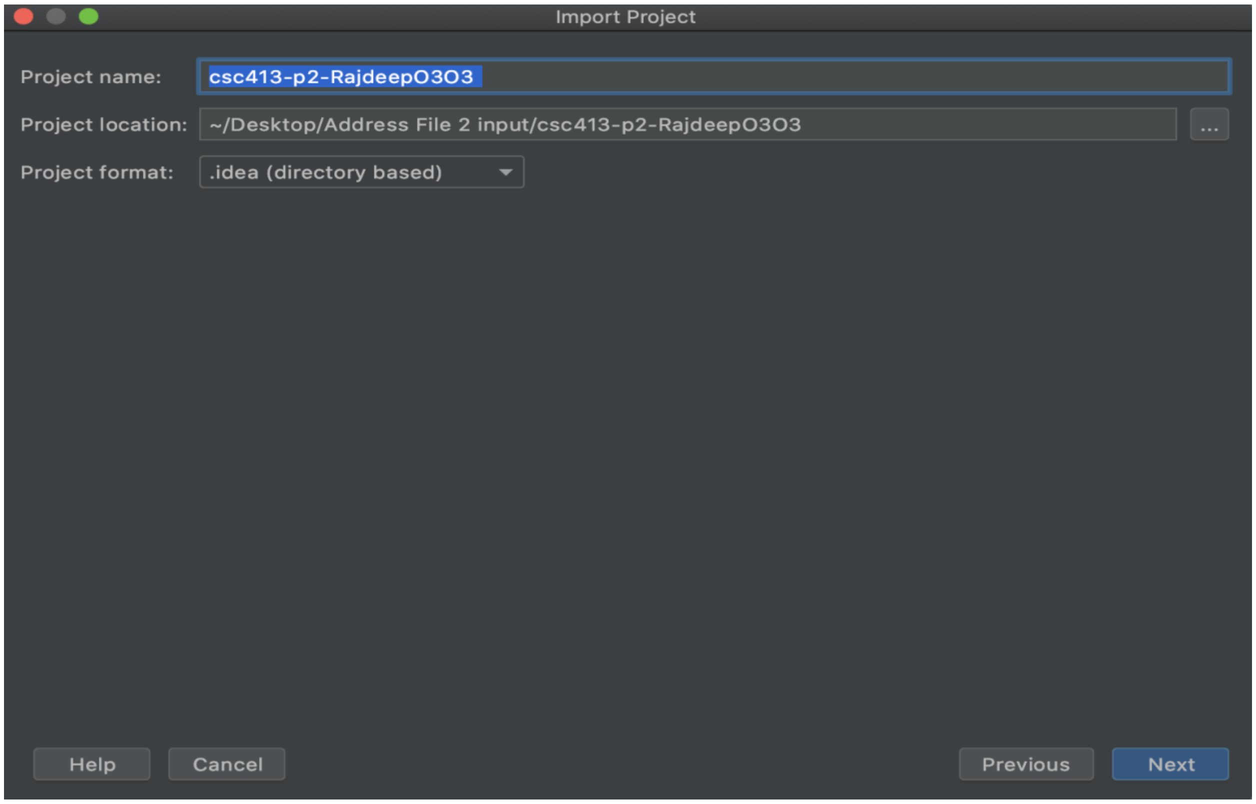
Once the repo is cloned, follow the following steps to import your project into IntelliJ. Steps are very easy, just follow the direction.



Click on “Import Project”



Click on “Create project from existing resources” and then click “Next” button.

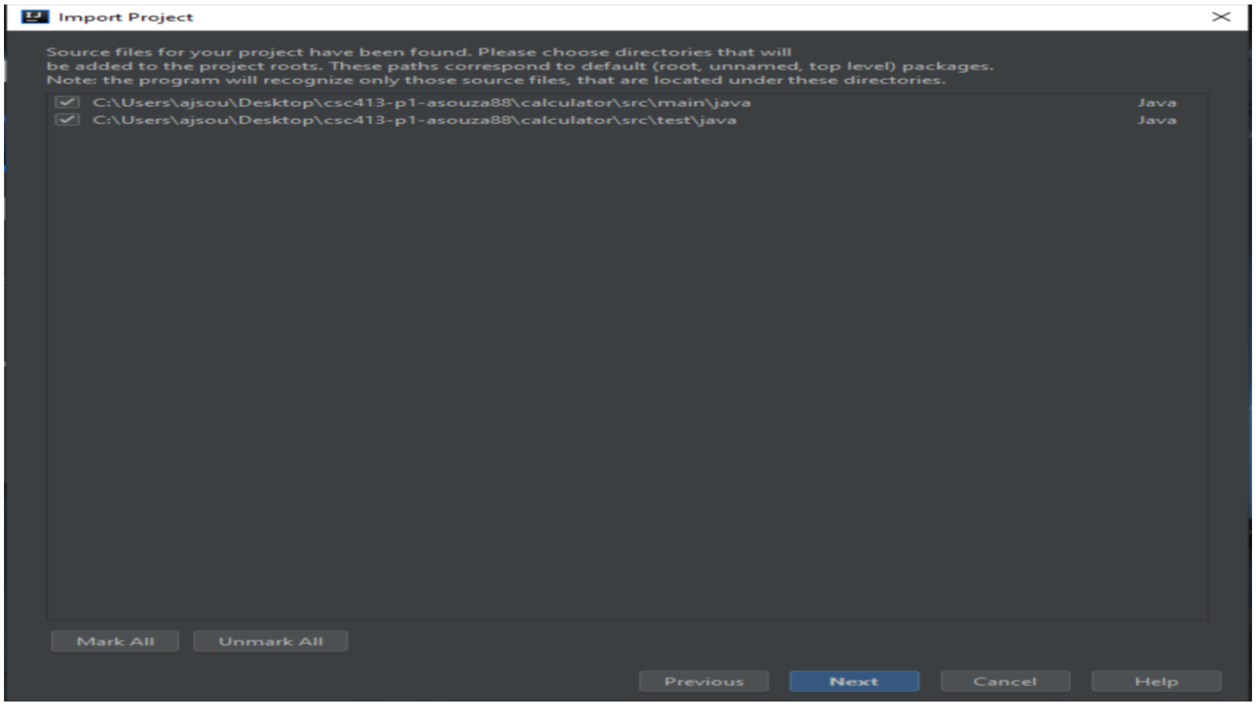


Project name: csc413-p2-RajdeepO3O3

Project location: ~/Desktop/Address File 2 input/csc413-p2-RajdeepO3O3

Project format: .idea (directory based)

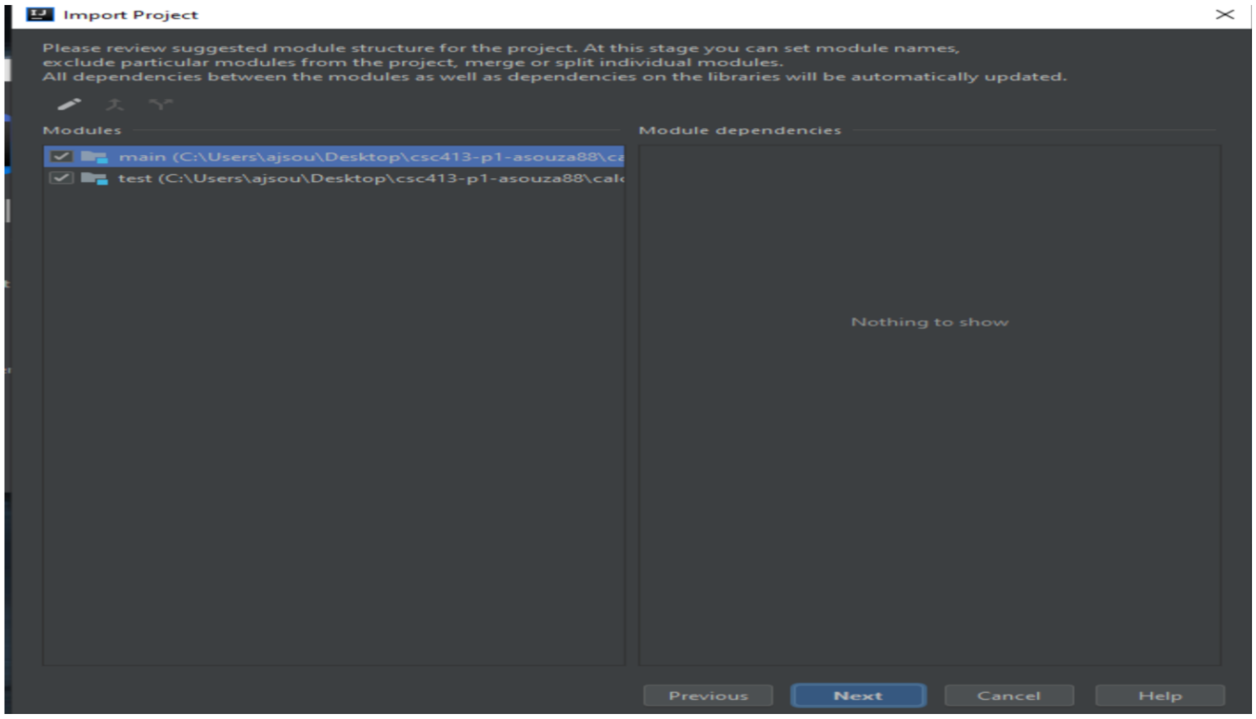
Leave the all default fields alone.



Source files for your project have been found. Please choose directories that will be added to the project roots. These paths correspond to default (root, unnamed, top level) packages. Note: the program will recognize only those source files, that are located under these directories.

✓ C:\Users\ajsou\Desktop\csc413-p1-asouza88\calculator\src\main\java Java
✓ C:\Users\ajsou\Desktop\csc413-p1-asouza88\calculator\src\test\java Java

Mark both items. One is the source code folder and second folder contain source of unit tests. Then, click on “Next” button.


Import Project

Please review suggested module structure for the project. At this stage you can set module names, exclude particular modules from the project, merge or split individual modules.
All dependencies between the modules as well as dependencies on the libraries will be automatically updated.

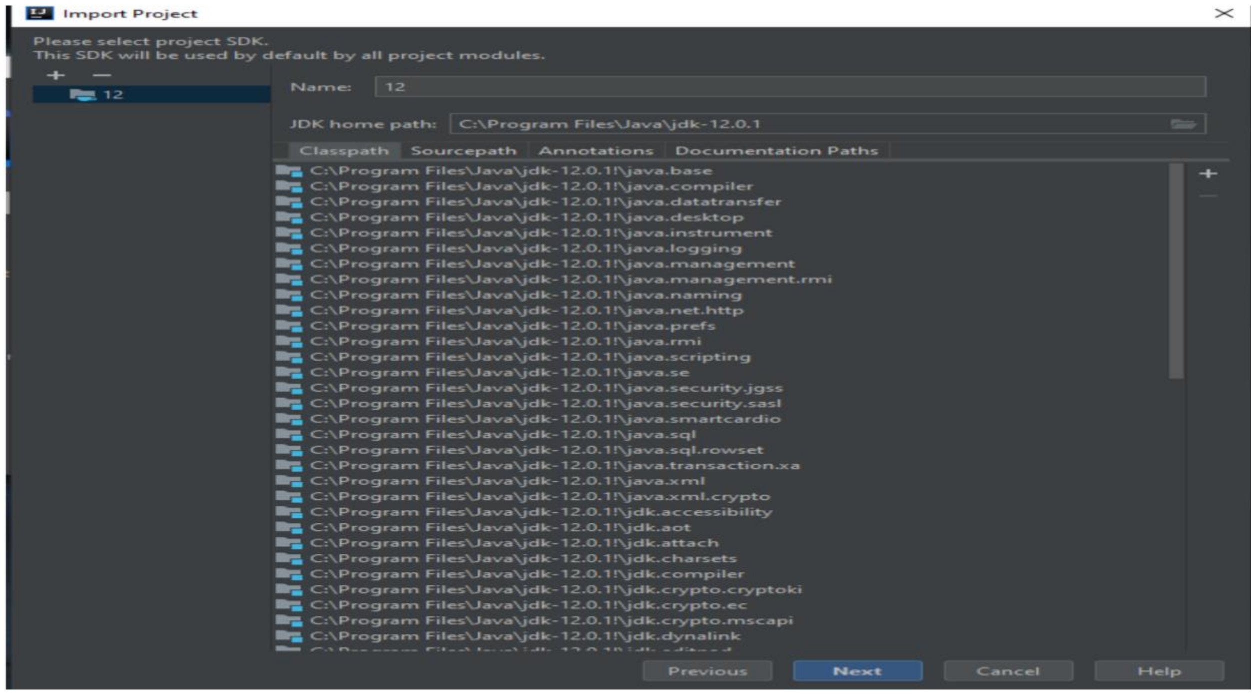
Modules

- [✓] main (C:\Users\ajsou\Desktop\csc413-p1-asouza88\ca
- [✓] test (C:\Users\ajsou\Desktop\csc413-p1-asouza88\calc

Module dependencies

Nothing to show

Click “Next” and all default fields can be left alone here.


Import Project

Please select project SDK.
This SDK will be used by default by all project modules.

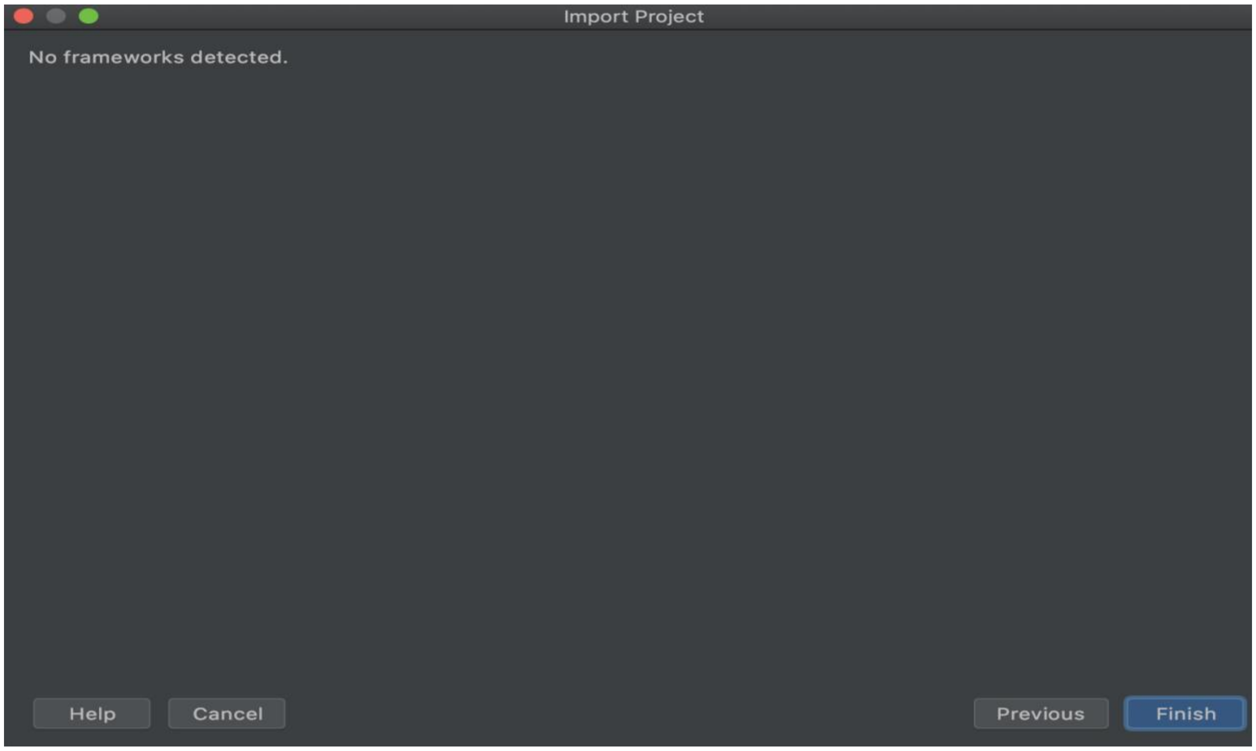
12

Name: 12

JDK home path: C:\Program Files\Java\jdk-12.0.1

```
C:\Program Files\Java\jdk-12.0.1!\java.base
C:\Program Files\Java\jdk-12.0.1!\java.compiler
C:\Program Files\Java\jdk-12.0.1!\java.datatransfer
C:\Program Files\Java\jdk-12.0.1!\java.desktop
C:\Program Files\Java\jdk-12.0.1!\java.instrument
C:\Program Files\Java\jdk-12.0.1!\java.logging
C:\Program Files\Java\jdk-12.0.1!\java.management
C:\Program Files\Java\jdk-12.0.1!\java.management.rmi
C:\Program Files\Java\jdk-12.0.1!\java.naming
C:\Program Files\Java\jdk-12.0.1!\java.net.http
C:\Program Files\Java\jdk-12.0.1!\java.prefs
C:\Program Files\Java\jdk-12.0.1!\java.rmi
C:\Program Files\Java\jdk-12.0.1!\java.scripting
C:\Program Files\Java\jdk-12.0.1!\java.se
C:\Program Files\Java\jdk-12.0.1!\java.security.jgss
C:\Program Files\Java\jdk-12.0.1!\java.security.sasl
C:\Program Files\Java\jdk-12.0.1!\java.smartcardio
C:\Program Files\Java\jdk-12.0.1!\java.sql
C:\Program Files\Java\jdk-12.0.1!\java.sql.rowset
C:\Program Files\Java\jdk-12.0.1!\java.transaction.xa
C:\Program Files\Java\jdk-12.0.1!\java.xml
C:\Program Files\Java\jdk-12.0.1!\java.xml.crypto
C:\Program Files\Java\jdk-12.0.1!\jdk.accessibility
C:\Program Files\Java\jdk-12.0.1!\jdk.aot
C:\Program Files\Java\jdk-12.0.1!\jdk.attach
C:\Program Files\Java\jdk-12.0.1!\jdk.charsets
C:\Program Files\Java\jdk-12.0.1!\jdk.compiler
C:\Program Files\Java\jdk-12.0.1!\jdk.crypto.cryptoki
C:\Program Files\Java\jdk-12.0.1!\jdk.crypto.ec
C:\Program Files\Java\jdk-12.0.1!\jdk.crypto.mscapi
C:\Program Files\Java\jdk-12.0.1!\jdk.dynalink
```

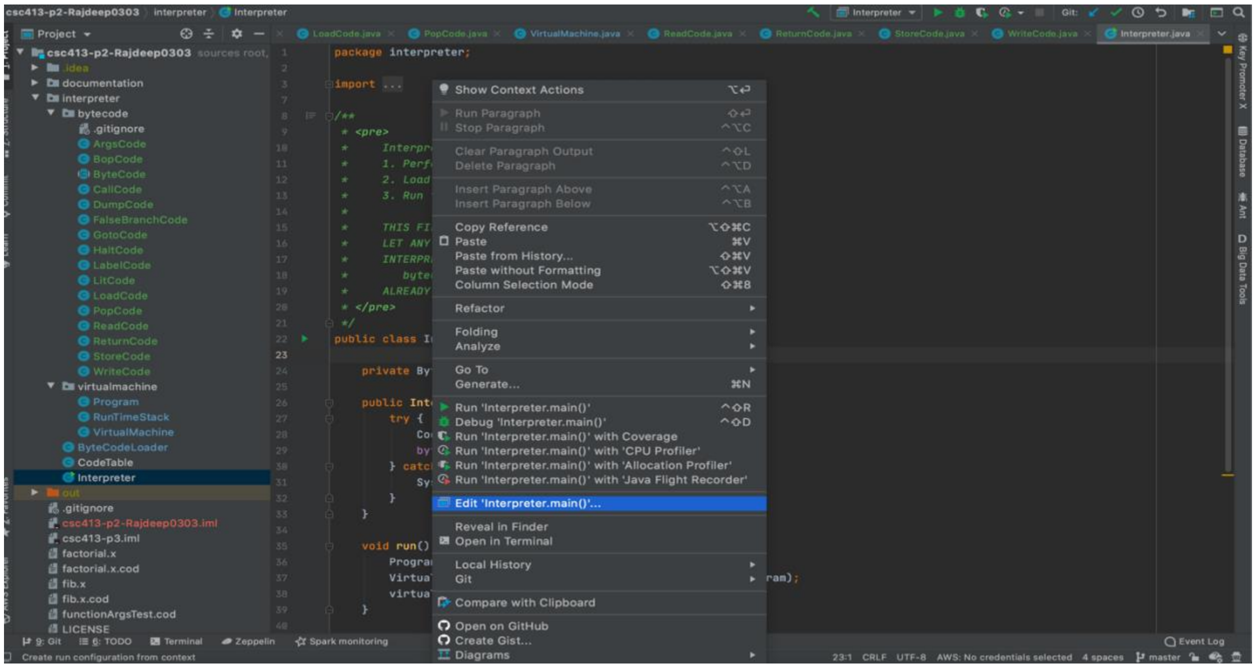
If you see a screen like this then click “Next” otherwise java JDK need to be installed before going forward. If you could not see this screen then click on + symbol on the top left. After JDK add click “Next”.



No frameworks detected.

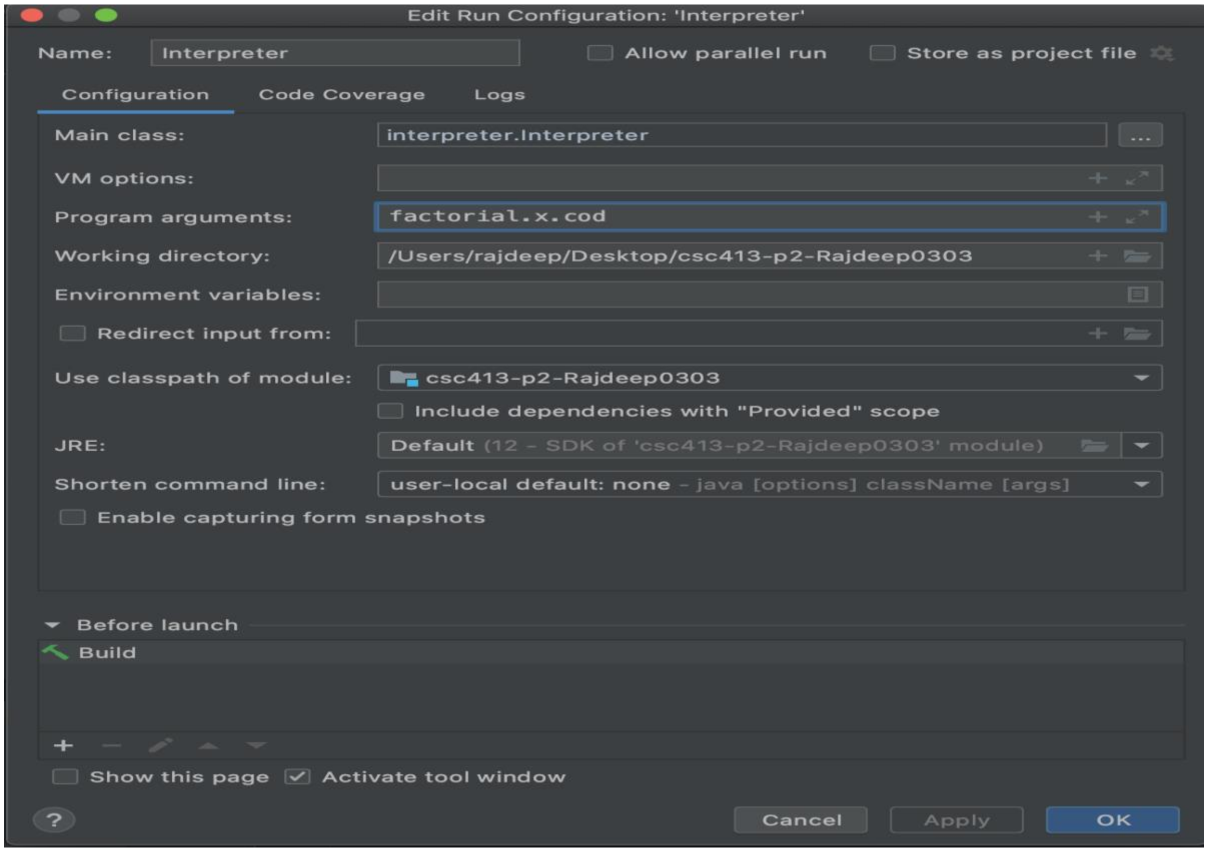
Click “Next” because this window should empty because we do not have any framework.

How to Run your Project



```
package interpreter;

import ...

/**
 * <pre>
 *     Interpr
 *     1. Perf
 *     2. Load
 *     3. Run
 *
 *     THIS FI
 *     LET ANY
 *     INTERPR
 *         byte
 *     ALREADY
 * </pre>
 */
public class I

    private By

    public Int
        try {
            Co
            by
        } catc
            Sy
        }
    }

    void run()
        Progra
        Virtua
        virtua                ram);
    }
```

While the project is open in IntelliJ, click “Interpreter.main()…” Edit Configurations



Edit Run Configuration: 'Interpreter'

Name: Interpreter

Allow parallel run

Store as project file

Configuration | Code Coverage | Logs

Main class: interpreter.Interpreter

VM options:

Program arguments: factorial.x.cod

Working directory: /Users/rajdeep/Desktop/csc413-p2-Rajdeep0303

Environment variables:

Redirect input from:

Use classpath of module: csc413-p2-Rajdeep0303

Include dependencies with "Provided" scope

JRE: Default (12 - SDK of 'csc413-p2-Rajdeep0303' module)

Shorten command line: user-local default: none - java [options] className [args]

Enable capturing form snapshots

Before launch

Build

Show this page

✓ Activate tool window

- In the “Program arguments:” text field, type in “factorial.x.cod”
- Repeat steps 3-5 with “fib” for the name and “fib.x.cod” for the program arguments
- Click APPLY then OK
- In the top right corner of the IntelliJ IDEA window, click the dropdown menu and select either “fib” or “factorial”

Assumption Made

Some assumptions, I had during the project:-

- I though, interpreter should be outputting binary numbers after GOTO.
- I assumed that the project will show the X program output.
- Byte code classes cannot call the dump method.
- Number of arguments in the source code are valid.
- Interpreter.java and CodeTable.java should not be modified.
- Some byte code classes may have init methods.
- Only need to print the integer value and result.
- Overflow have to be managed.
- Underflow do not have to be managed.
- Some classes may have additional methods or constructors.

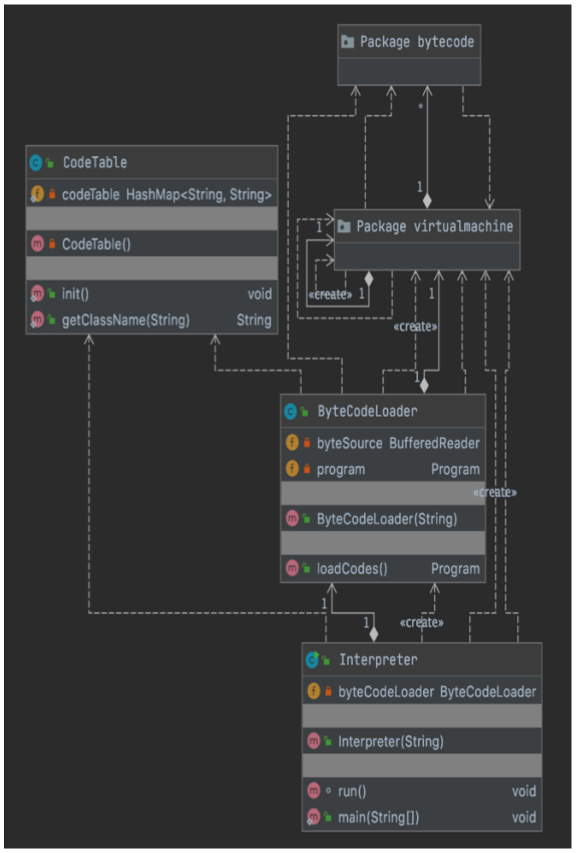
Implementation Discussion

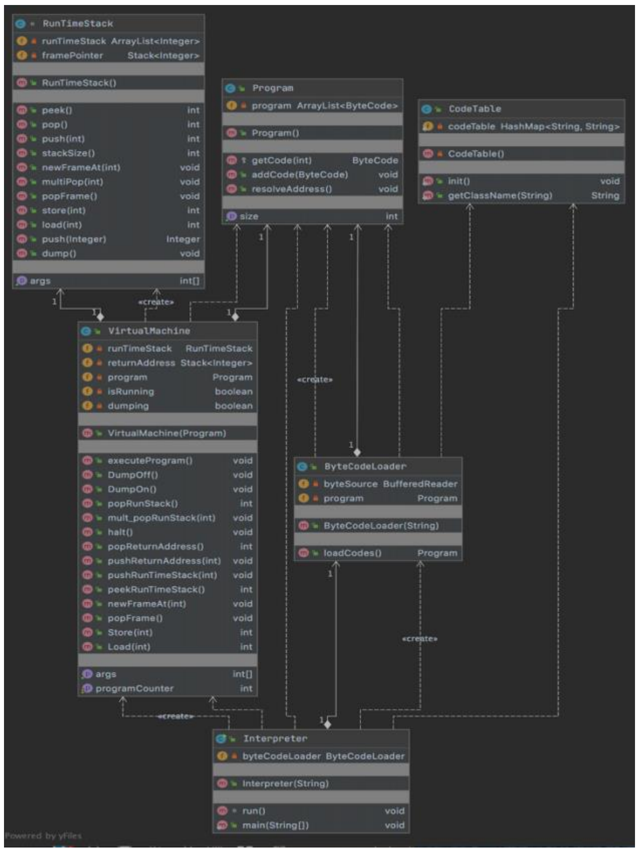
Interpreter is the main class that starts the program. “Interpreter.java” and “CodeTable.java” classes was already implemented of us. “VirtualMachine.java” class is making the virtual machine to do all the work to run methods from RunTimeStack and executes the X program. This class is like a controller of this project. “RunTimeStack.java” class help VirtualMachine to execute the program. No getters and setters need for this class. There are several functions in this class. For example, peek(), push(), dump(),push(),pop(), store(), load(), newFrameAt(), and popFrame(). “ByteCodeLoader.java” class reading the ByteCodes from the source code file. I use ArrayList to store the ByeCode. “Program.java” class where we need resolveAddress(). This resolveAddress() will resolves all the addresses of particular byte codes. This class store all the ByteCode read from the source file.

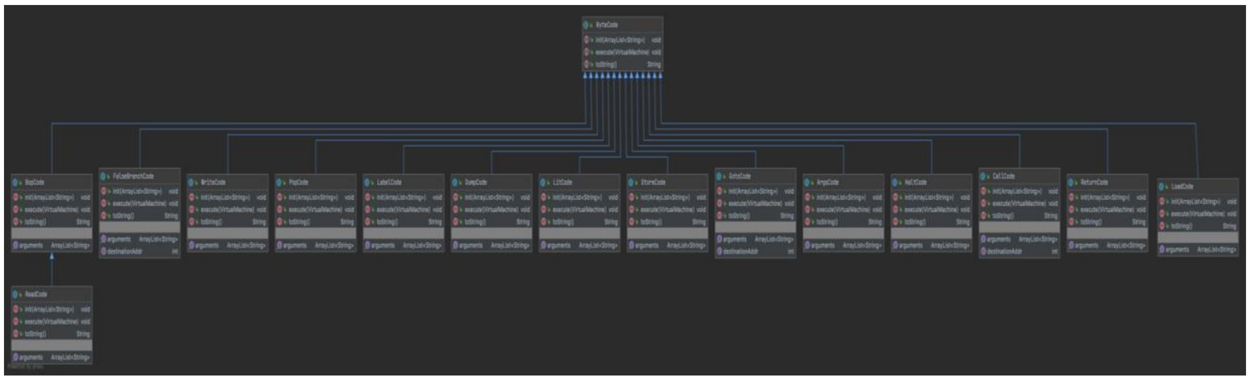
“ByteCode.java” class have following descriptions:-

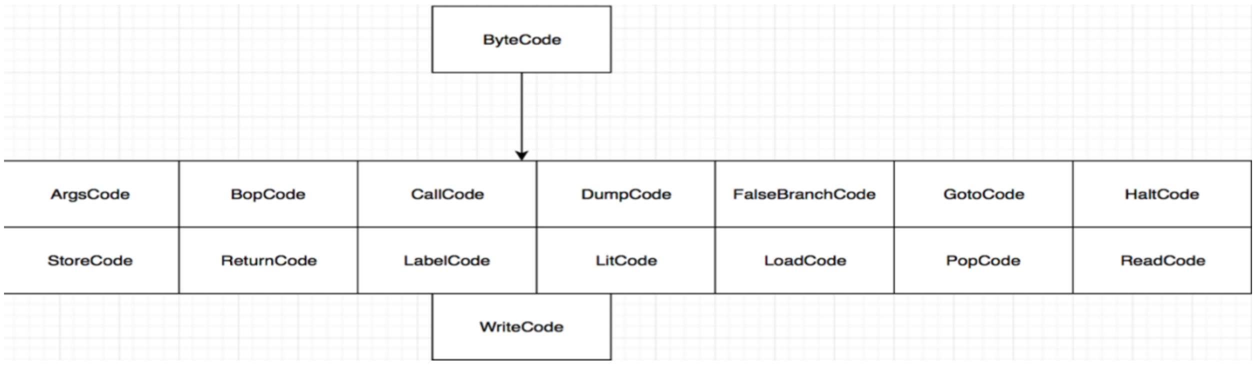
1. ByteCode.java class initialize the functions for byte codes.
2. HaltCode.java main goal is to stop execution and it is not allowed to kill the interpreter program.
3. PopCode.java class pops the top of the stack and remove the value from the run time stack.
4. FalseBranchCode.java class pops the top of the stack. It have one argument and when nothing to pop or value is not 0 then, it move to next byte code.
5. GotoCode.java initializes execution. It has one argument. Also, if dump is on then, it required to be dumped.
6. StoreCode.java class pops the top of the stack. The value stores in the offset. The offset starts from the beginning of the frame.
7. LoadCode.java class push the value form an offset from the current frame. The offset starts from the beginning of the frame.
8. LitCode.java class push the literal value. It take one or two arguments.
9. ArgsCode.java class executed previous call byte code. It has one argument. Also, if dump is on then, it required to be dumped.
10. DumpCode.java class used to set the state. It used to turn dumping ON and OFF.
11. CallCode.java class track the control where it going to return and it transfers control to the indicated function.
12. ReturnCode.java class return from the current functions. It return zero to one argument.
13. BopCode.java class remove two value or it pops two top level of the stack. It push one value.
14. ReadCode.java class read a integer enter by user and push the value to the stack.
15. LabelCode.java class target for branches. It have one argument. It used to address the resolution.
16. WriteCode.java class write the value of the top of the stack and display information to the console.

Class Diagram









Project Reflection

Working on this project since the beginning, I learn so many things though out the project. Due to difficulty of this project, I had to use the debugging skills during the project. The main problem, I was getting was to match the output to the professor output because, I was showing the labels for falsebranch and goto codes but I have to hide the addresses. I was able to implement an interpreter to execute ByteCodes of a language X. I spend lots of time on coding and debugging the program. Debugging helps me to encounter several errors. This project was very hard for me to understand, I read the pdf 4 times to better understanding. This project was my longer projects in writing code so far. I took help on the slack and some classmate to understand the project better. This project really improve my coding skills, problem solving and understanding the OOP principles due to difficulty level of the project.

Project Conclusion/Results

Overall, this project give me an excellent knowledge of slack, encapsulation, inheritance, etc. The project itself performs very well but, it take lots of time to code the four major classes that we need to implement. My interpreter successfully compiles and prints out the Byte Codes. This program displays how each byte code operate in the X program. I wish more tools and hints were provided to implement and debug the project. After debugging, I was successfully able to interpret the bytecode files (factorial.x.cod and fib.x.cod. I feel better and confident after finishing the entire project, and I have a better understanding of the OOP principles. My main focus was not to break the encapsulation within the code. I am proud that, I complete this project with all requirements given in the assignment pdf. In the last, this project was very helpful because I learned about programming in both abstract idea and technically.