CSC 413 Project Documentation

Fall 2018

Keith Rich

(With some help from David Dean)

915 799 591

CSC 413.01

https://github.com/csc413-02-fa18/csc413-p2-GuantanamoBae.git

Table of Contents

1 Introduction

1.1 Project Overview

Creating a program that reads a unknow language, compiles said language, then finally displays the output though a virtual machine AKA a computer.

1.2 Technical Overview

Said language is read from files provided. Then passed to an interpreter that loads the files into the predetermined bytecodes. Then it is passed to a virtual machine that works as a computer within a computer and runs this unknown, made up language.

1.3 Summary of Work Completed

Built and initiated methods for each of the fifteen bytecodes. Implemented all the working methods and functions in ByteCodeLoader, Program, RuntimeStack, and finally virtual machine.

2 Development Environment

Intellj ultimate IDEA 2018.2.3

3 How to Build/Import your Project

Using your terminal cd to the folder you wish to hold your repository at. Once their git clone URL the entire project. Go to your IDE of your choice and import the entire repository that has been cloned. Once there use no wrappers and continue with default setting to import your project from existing sources. Once imported using this system of file tree: repo > interpreter > interpreter > then edit configurations. Once there change the “program arguments” to the file that you wish to read. Example fib.x.cod. Then under the user instructions of the manual find out how to build and run your IDE.

4 How to Run your Project

Most of what was explained above will go into details about how to run the program. Once it is imported edit the configurations to the specified file you with to read.

5 Assumption Made

I thought for a very long time that the difficulty of this project would only be understanding how to implement the project via the twenty-one pages of PDF. Realization that it is not only a difficult project to wrap your head around in general, but a very difficult to code all the aspects of it as well. As forewarned in class, this is a major step from just being able to implements a simple recursive function, or garbage collector in CSC 340

6 Implementation Discussion

As discussed in class the first step was creating and implementing all the bytecodes. Certain bytecodes must do certain thing, for example the bytecode for dump was rather extensive due in part because it must remove all the objects in both the runtime stack and the frames that were created in the program. Whereas the load bytecode was there strictly to assignment a variable. Once the bytecodes were built and the methods that go along with them were initiated it came to the act of loading those bytecodes into a HashMap that was placed in “Code table” using a class called Bytecode Loader. Rather self-explanatory bytecodeLoader is. Program had to contain a method called “resolve address” in which it help determine where the frame is and where to place the variable. Runtime stack was next and was where one would implement the functions that go along with what you would expect a vector or array to have. Where you could pop from the top of the stack, or peek at what object was held in the front of the stack. The majority of the coding for this assignment comes from the virtual machine class. Must like the Runtime stack it also includes certain function like push, pop and peek that you would expect to find in the runtime stack class. However, this class contains the constructors or rather the placements of where the new from will be going to as well as the bulk of store and load for the actual program.

6.1 Class Diagram

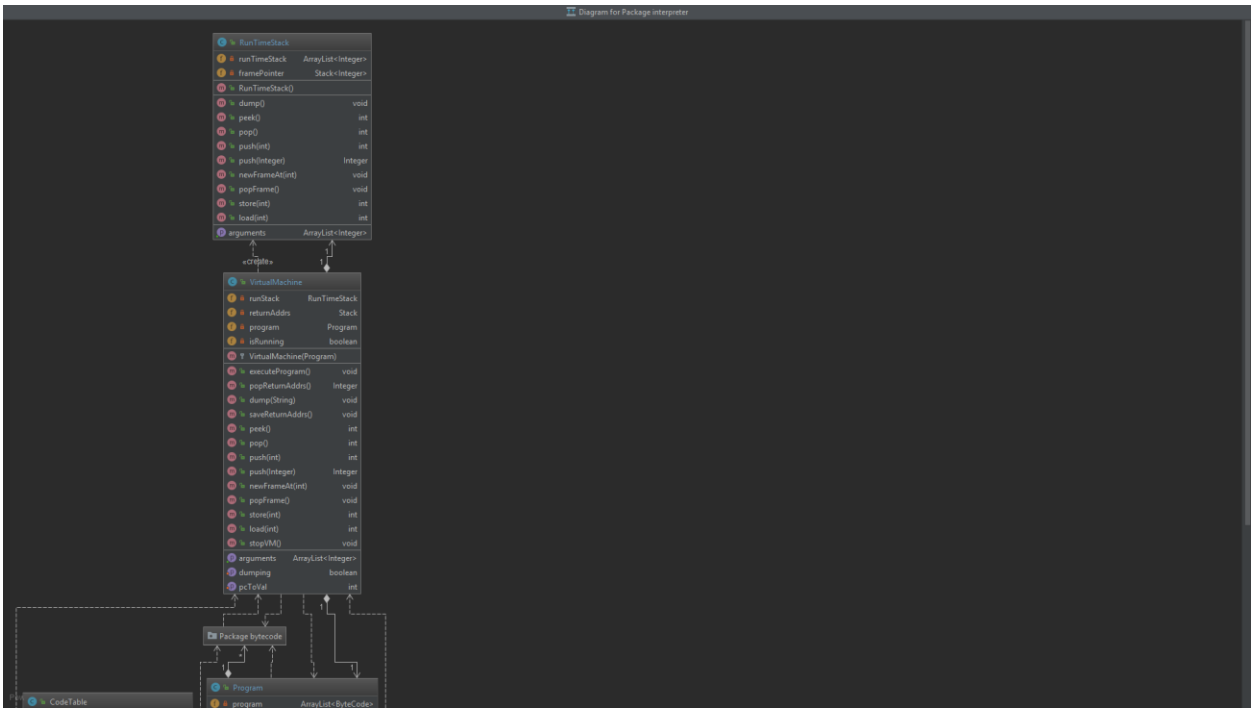


Diagram for Package interpreter
RunTimeStack
runTimeStack ArrayList<Integer>
framePointer Stack<Integer>
RunTimeStack()
dump() void
peek() int
pop() int
push(int) int
push(Integer) Integer
newFrameAt(int) void
popFrame() void
store(int) int
load(int) int
arguments ArrayList<Integer>
«create»
VirtualMachine
runStack RunTimeStack
returnAddrs Stack
program Program
isRunning boolean
VirtualMachine(Program)
executeProgram() void
popReturnAddrs() Integer
dump(String) void
saveReturnAddrs() void
peek() int
pop() int
push(int) int
push(Integer) Integer
newFrameAt(int) void
popFrame() void
store(int) int
load(int) int
stopVM() void
arguments ArrayList<Integer>
dumping boolean
pcToVal int
Package bytecode
Program
program ArrayList<ByteCode>
CodeTable

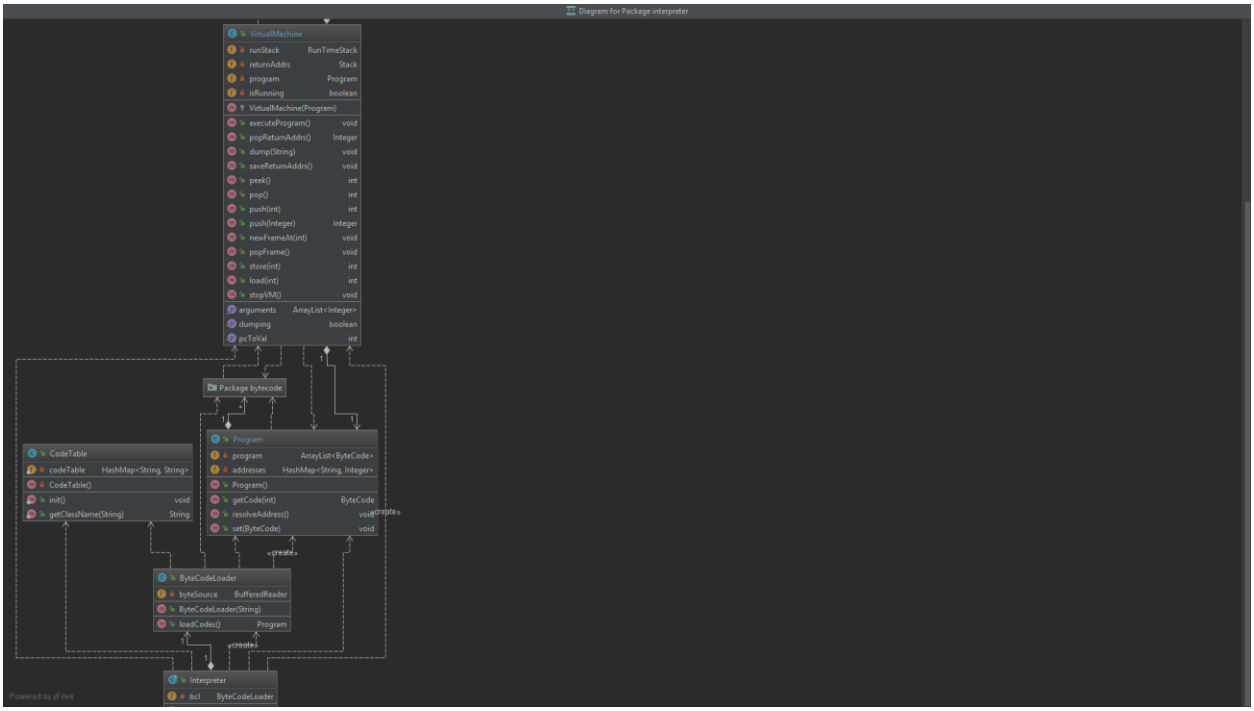


Diagram for Package interpreter
VirtualMachine
runStack RunTimeStack
returnAddrs Stack
program Program
isRunning boolean
VirtualMachine(Program)
executeProgram() void
popReturnAddrs() Integer
dump(String) void
saveReturnAddrs() void
peek() int
pop() int
push(int) int
push(Integer) Integer
newFrameAt(int) void
popFrame() void
store(int) int
load(int) int
stopVM() void
arguments ArrayList<Integer>
dumping boolean
pcToVal int
Package bytecode
CodeTable
codeTable HashMap<String, String>
CodeTable()
init() void
getClassName(String) String
Program
program ArrayList<ByteCode>
addresses HashMap<String, Integer>
Program()
getCode(int) ByteCode
resolveAddress() void
set(ByteCode) void
«create»
«create»
ByteCodeLoader
byteSource BufferedReader
ByteCodeLoader(String)
loadCodes() Program
«create»
Interpreter
bcl ByteCodeLoader
Powered by yFiles

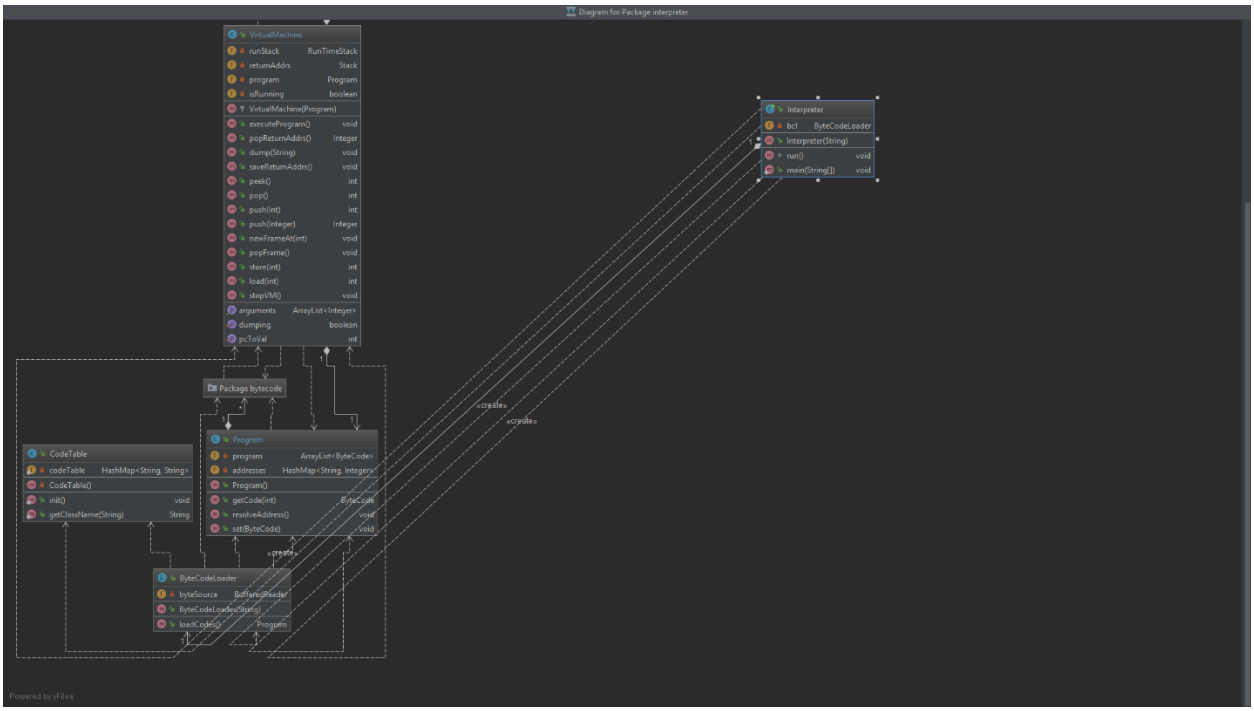
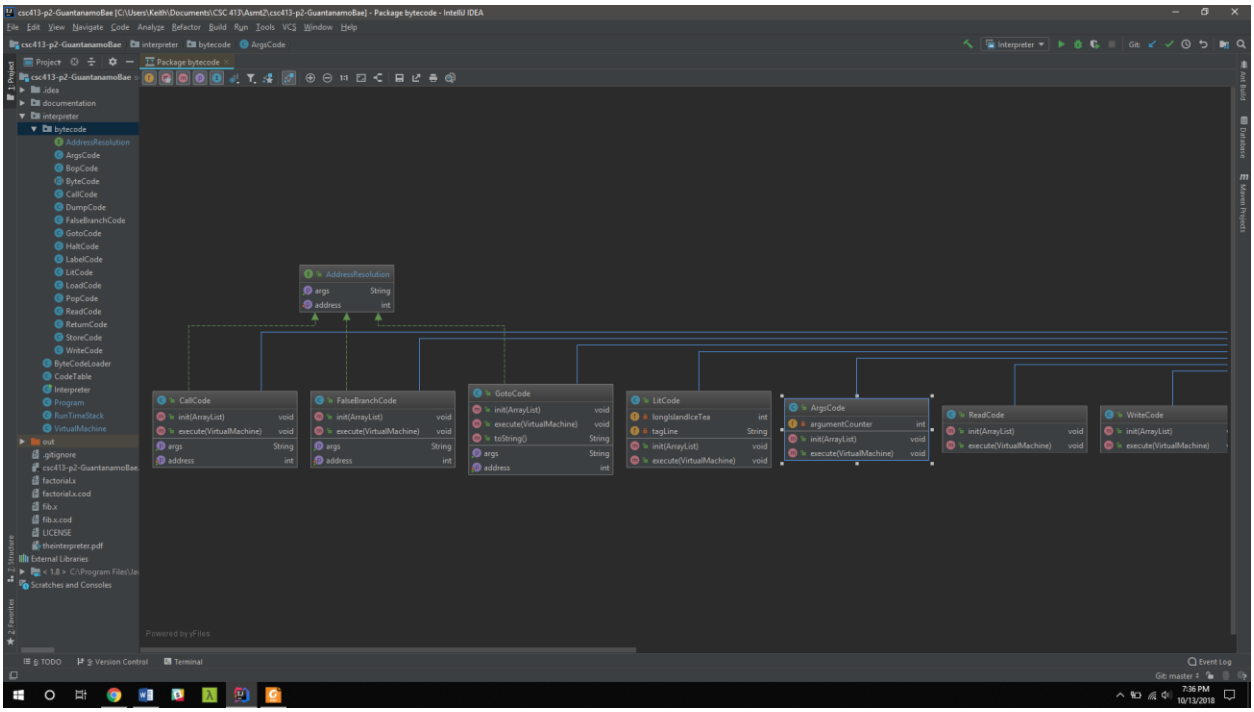
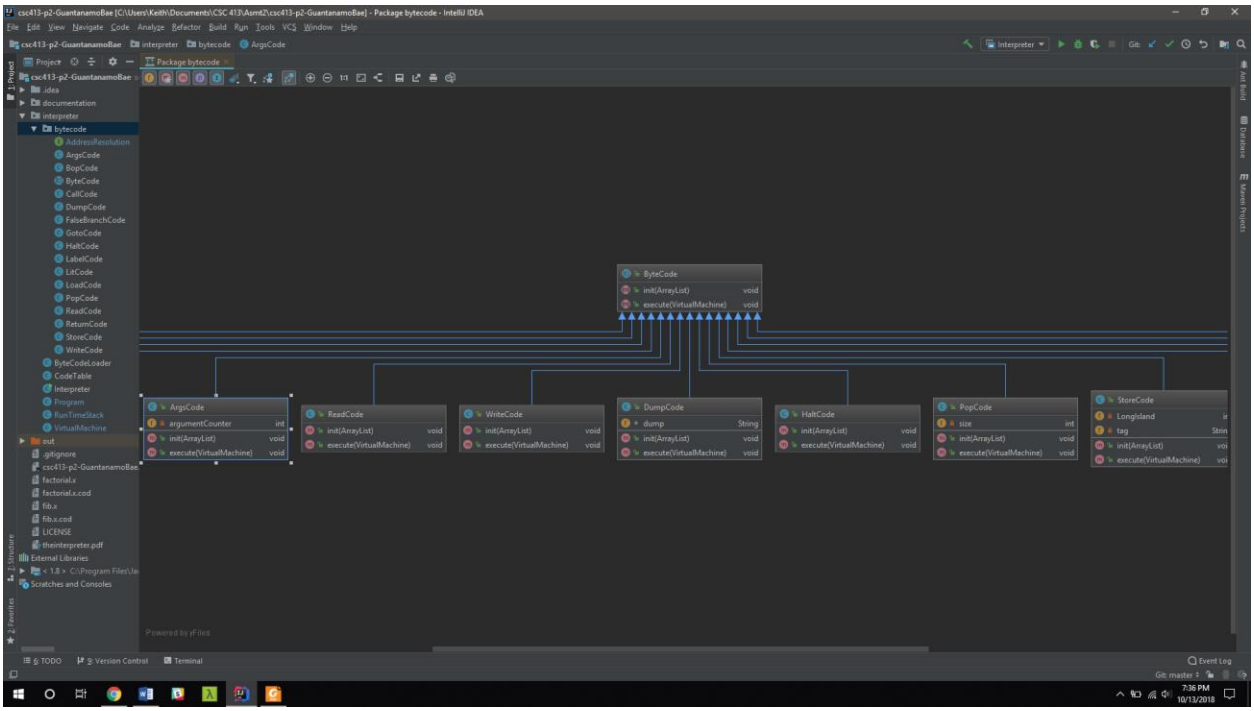
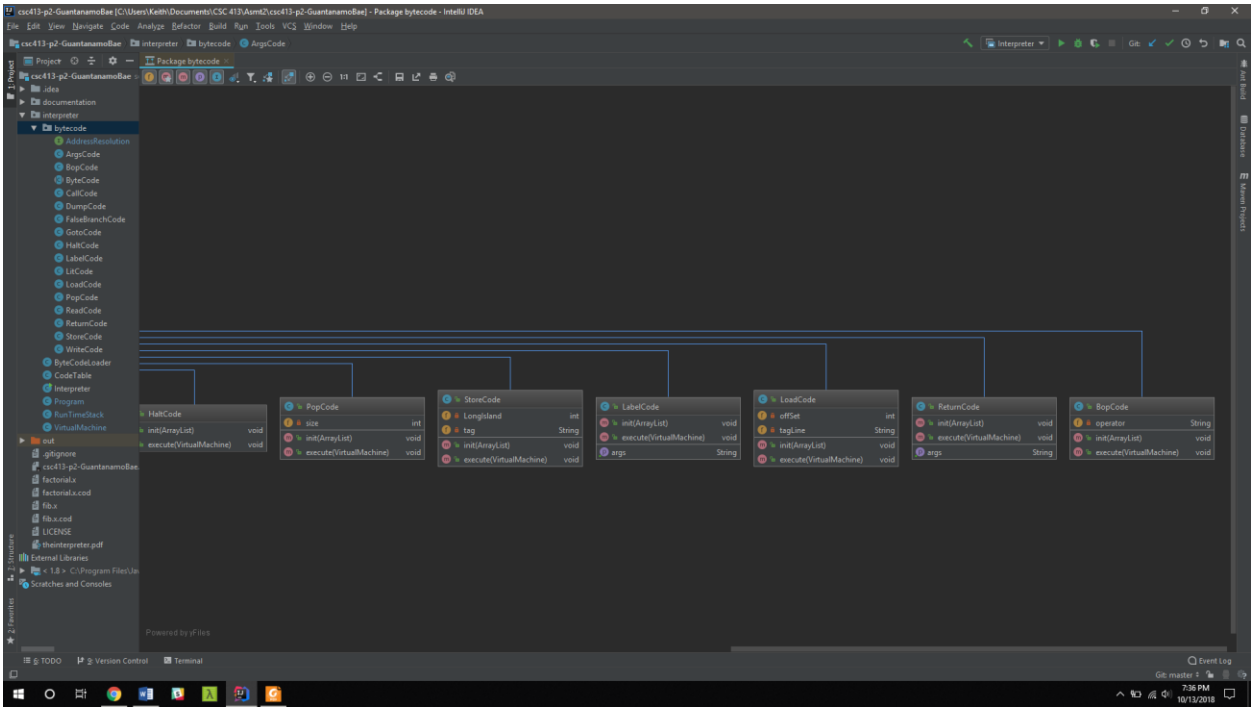


Diagram for Package interpreter
VirtualMachine
runStack RunTimeStack
returnAddrs Stack
program Program
isRunning boolean
VirtualMachine(Program)
executeProgram() void
popReturnAddrs() Integer
dump(String) void
saveReturnAddrs() void
peek() int
pop() int
push(int) int
push(Integer) Integer
newFrameAt(int) void
popFrame() void
store(int) int
load(int) int
stopVM() void
arguments ArrayList<Integer>
dumping boolean
pcToVal int
Interpreter
bcl ByteCodeLoader
Interpreter(String)
run() void
main(String[]) void
Package bytecode
«create»
«create»
CodeTable
codeTable HashMap<String, String>
CodeTable()
init() void
getClassName(String) String
Program
program ArrayList<ByteCode>
addresses HashMap<String, Integer>
Program()
getCode(int) ByteCode
resolveAddress() void
set(ByteCode) void
«create»
ByteCodeLoader
byteSource BufferedReader
ByteCodeLoader(String)
loadCodes() Program
Powered by yFiles


csc413-p2-GuantanamoBae [C:\Users\Keith\Documents\CSC 413\Asmt2\csc413-p2-GuantanamoBae] - Package bytecode - IntelliJ IDEA
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
csc413-p2-GuantanamoBae interpreter bytecode ArgsCode
Interpreter
Git:
Project
Package bytecode
csc413-p2-GuantanamoBae
.idea
documentation
interpreter
bytecode
AddressResolution
ArgsCode
BopCode
ByteCode
CallCode
DumpCode
FalseBranchCode
GotoCode
HaltCode
LabelCode
LitCode
LoadCode
PopCode
ReadCode
ReturnCode
StoreCode
WriteCode
ByteCodeLoader
CodeTable
Interpreter
Program
RunTimeStack
VirtualMachine
out
.gitignore
csc413-p2-GuantanamoBae
factorial.x
factorial.x.cod
fib.x
fib.x.cod
LICENSE
theinterpreter.pdf
External Libraries
< 1.8 > C:\Program Files\Ja
Scratches and Consoles
AddressResolution
args String
address int
CallCode
init(ArrayList) void
execute(VirtualMachine) void
args String
address int
FalseBranchCode
init(ArrayList) void
execute(VirtualMachine) void
args String
address int
GotoCode
init(ArrayList) void
execute(VirtualMachine) void
toString() String
args String
address int
LitCode
longIslandIceTea int
tagLine String
init(ArrayList) void
execute(VirtualMachine) void
ArgsCode
argumentCounter int
init(ArrayList) void
execute(VirtualMachine) void
ReadCode
init(ArrayList) void
execute(VirtualMachine) void
WriteCode
init(ArrayList) void
execute(VirtualMachine) void
Powered by yFiles
TODO
Version Control
Terminal
Event Log
Git: master
7:36 PM
10/13/2018


csc413-p2-GuantanamoBae [C:\Users\Keith\Documents\CSC 413\Asmt2\csc413-p2-GuantanamoBae] - Package bytecode - IntelliJ IDEA
ByteCode
init(ArrayList) void
execute(VirtualMachine) void
ArgsCode
argumentCounter int
ReadCode
WriteCode
DumpCode
dump String
HaltCode
PopCode
size int
StoreCode
LongIsland
tag String

0


csc413-p2-GuantanamoBae [C:\Users\Keith\Documents\CSC 413\Asmt2\csc413-p2-GuantanamoBae] - Package bytecode - IntelliJ IDEA
HaltCode
PopCode
StoreCode
LabelCode
LoadCode
offSet int
tagLine String
ReturnCode
BopCode
operator String
args String

7 Project Reflection

The project was extremely challenging. Like the PDF says, it takes a few reads to fully wrap your head around the program requirements. However, if I were to be compliantly honest I could have used a dumbing down PDF to read though as well. My initial thoughts were, and these thoughts were discussed with other classmates; that if I fully know what the program should do inside and out then implementation should come easy. As it came to be, you really can't know what you're getting into until you start working on it. Or in other words, sampled some of the code. Having already pushed the assignment to GitHub I have washed my hands of it. I understand that most students keep procrastinating the project until they run out of time, but procrastination was never an issue with me on this assignment. Most of my other classes I am taking have suffered because of the time I spent on this project. If I had a chance to do everything over again, I would have looked at the code first. Then read the PDF over and over again. Then looked at the code again and read it again. My mistake was spending too much time trying to know the in's-and-outs about this project before even looking at the code. At the end of the day I think myself, and most of the students, weather they admit it or not don't really have a full understanding of the project and how to incorporate it into the boiler plate code provided. As I stated earlier in the documentation, this is a giant leap from implementing a garbage collector in CSC340.

8 Project Conclusion/Results

My code complies and gives me the correct output from what I can see. I am sure you can relate to the feeling of being completed and the elated feeling that comes with it. This is now my third write up where I have modified this conclusion. It started with me complaining about not having any rhyme or reason for bracket placement or number values. The second amendment was me complaining about random brackets or commas but at least I had correct values. Then lastly and hopefully this is my last time I redo this documentation my complaints are back to normal political crap. As I have said many times in this documentation, it was a hard project to complete. Hopefully one in the coming years I can think back on.