Brandon Watanabe
921988506
17 May, 2022

Assignment 5 Documentation

Github Repository

https://github.com/sfsu-csc-413-fall-2022-roberts/assignment-5---debugger-BrandonWatanabeStudentAccount

Project Introduction and Overview

For this project, I had to implement a Debugger into the previous project, Interpreter. The debugger allows for the user to step through the code much like in an IDE.

For this project I had to implement a Parser which builds an Abstract Syntax Tree from tokens made by the previous assignment. I had to add functionality so the compiler can accept new tokens as well as a new switch statement.

Summary of Technical Work

The Interpreter takes in a file and if the user enters debug mode it runs Debugger. Debugger takes the file and adds the correct extensions then loads in the DebuggerCodeTable and runs the DebuggerVirtualMachine and handles the execution of DebuggerCommands. DebuggerCodes creates a table of codes and their corresponding debuggerByteCodes that are sent to the ByteCodeLoader. The ByteCodeLoader creates the instances of the ByteCodes. The DebuggerShell handles the user commands. The user commands all are subclassed of DebuggerCommand. The DebuggerCommands each talk to the DebuggerVirtualMachine which holds the FunctionEnvironmentRecord stack and handles execution of debugger byte codes. The FunctionEnvironmentRecord handles the output for each line.

Scope of Work

Task	**Completed**
Implemented and updated in Debugger.java	X
Public Debugger()	X
public void run()	X
Implemented and updated in DebuggerCodeTable.java	TimestampTree
public static void init()	X
public static void populateMap()	X
public static String get(String code)	X

Updated FunctionEnvironmentRecord.java	X
public void print()	X
public void beginScope()	X
public void setFunctionInfo(String functionName, int startingLineNumber, int endingLineNumber)	X
public void setFuncitonLineNumber(int currentFuncitonLineNumber)	X
public void enter(String symbol, int value)	X
public void pop(int count)	X
public void setRecordListVariables()	X
Updated in ui package	X
ContinueCommand	X
DebuggerCommand	X
DebuggerShell	X
Entry	X
ExitCommand	X
HelpCommand	X
InvalidInputCommand	X

ListCommand	X
LocalsCommand	X
SetCommand	X
SourceCommand	X
StepCommand	X
Implemented and updated in debuggercodes package	X
CallDebuggerCode	X
FormalDebuggerCode	X
FunctionDebuggerCode	X
LabelDebuggerCode	X
LineDebuggerCode	X
LitDebuggerCode	X
PopDebuggerCode	X
ReturnDebuggerCode	X

Execution and Development Environment

I used Java 17 on the IntelliJ IDE on my Windows 10 computer to complete this assignment and tested in both the IDE and shell.

Compilation Result

Using the instructions provided in the assignment one specification:

```
  ->    1: program {boolean j int i
        2:   int factorial(int n) {
        3:       if (n < 2) then
        4:          { return 1 }
        5:       else
        6:          {return n*factorial(n-1) }
        7:   }
        8:   while (1==1) {
        9:       i = write(factorial(read()))
        10:   }
        11: }
Type ? for help
?
set
list
locals
source
step
continue
exit
Type ? for help
set
Enter line number:
2
Type ? for help
set
Enter line number:
4
Type ? for help
list
  * 2:   int factorial(int n) {
  * 4:          { return 1 }
Type ? for help
locals
Type ? for help
source
  ->    1: program {boolean j int i
      * 2:   int factorial(int n) {
        3:       if (n < 2) then
      * 4:          { return 1 }
        5:       else
        6:          {return n*factorial(n-1) }
        7:   }
        8:   while (1==1) {
```

```
        9:      i = write(factorial(read()))
        10:   }
        11: }
Type ? for help
continue
< -, -, -, -, -1 >

< (), -1, -1, Read, -1 >

Please enter an integer:
1
RETURN
< (), -1, -1, Read, -1 >

< -, -, -, -, -1 >

< (), -1, -1, Write, -1 >

< ( <dummyFormal,0> ), -1, -1, Write, -1 >

1
Exception in thread "main" java.util.EmptyStackException
at java.base/java.util.Stack.peek(Stack.java:101)
at java.base/java.util.Stack.pop(Stack.java:83)
at interpreter.RunTimeStack.popFrame(RunTimeStack.java:73)
at interpreter.VirtualMachine.popFrame(VirtualMachine.java:82)
at interpreter.bytecode.ReturnCode.execute(ReturnCode.java:49)
at
interpreter.bytecode.debuggercodes.ReturnDebuggerCode.execute(ReturnDebuggerCode.java:
11)
at
interpreter.debugger.DebuggerVirtualMachine.executeProgram(DebuggerVirtualMachine.java:4
5)
at interpreter.debugger.ui.ContinueCommand.execute(ContinueCommand.java:18)
at interpreter.debugger.Debugger.run(Debugger.java:33)
at interpreter.Interpreter.main(Interpreter.java:38)

Process finished with exit code 1
```

I could not get past the second return bytecode as I the framePointers Stack was empty and I wasn’t sure how to fix this so that it runs.

Assumptions

I assumed that my Assignment 4 was good enough to complete this assignment but I am not sure that it was.

Implementation

Debugger

The Debugger takes in the file and runs the Interpreter constructor. Then it adds the correct file extensions to the file, then runs the DebuggerCodeTable initialization method. The run method creates a Program object from the DebugerCodeTable using the ByteCodeLoader. Then it creates a DebuggerVirtualMachine object and a DebuggerShell Object. It then runs the shell prompt and executes the commands in a loop till exit.

DebugerCodeTable

The DebugerCodeTable class stores the relations between the bytecode names and their respective ByteCode subclass. The debugger versions are extensions on the base byte code subclass. To do this, first, I created an init method that runs the private method to populate the HashMap that stores this data. This method reads a file with all of the different types of bytecodes and the name of the subclass. It then takes these and assigns them to the HashMap. I also implemented a get method to access the HashMap. To get items from the DebugerCodeTable, there is a get method that checks if the code is in the table and if not then it takes it from CodeTable.

FunctionEnvironmentRecord

The FunctionEnvironmentRecord keeps track of the state of the function throughout the debugging process. It stores the current line number, the start and end line numbers of the function, the name of the function, and any formals it may have.

DebuggerShell

The DebuggerShell scans the sourceFile and parses each line, putting them into Entry objects which hold the line number, the actual source line, and if there is a breakpoint on that line. It also runs the prompt for the user to input commands. These commands are taken in and sent to their respective DebuggerCommand subclass for executuion.

DebuggerCommand

The DebuggerCommand class is an abstract class for the various command subclasses. These subclasses are ran after a user input and each have their own implementation of the execute method. These commands talk to the shell to advance the current line, check if the current line has a break point, prints out the source code, and talks to the DebuggerVirtualMachine.

DebuggerVirtualMachine

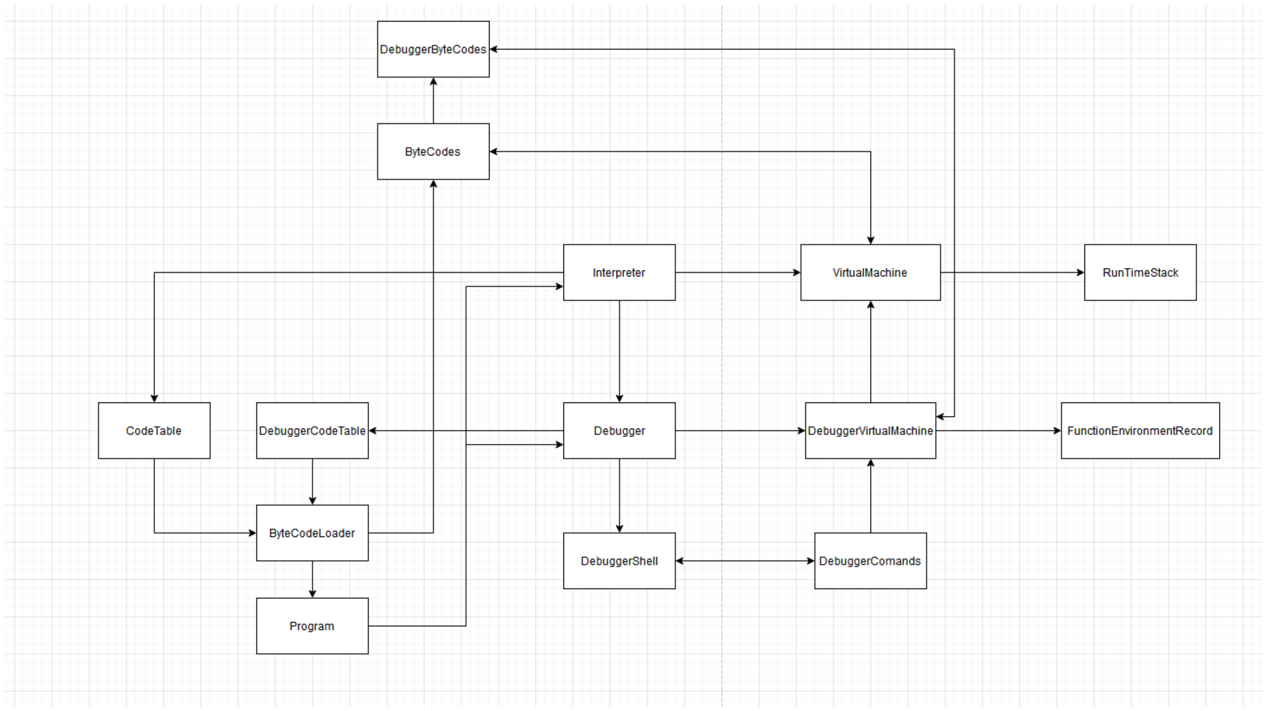
The DebuggerVirtualMachine was by far the hardest part of this project. I had much trouble with this and implementing the various debugger byte codes to make sure it all functions properly.

The DebuggerVirtualMachine acts a bridge between the debugger and the virtual machine. It has a stack of FunctionEnvironmentRecords and updates them according to the byte codes read from the file.

Code Organization

I implemented all of the new classes and methods in the same style as the ones provided so that it was easy to read and understand what additions were made.

Class Diagram



Results and Conclusion

This project really challenged me and my knowledge of object oriented programming. It was definitely the hardest in this class and I could not finish it all. I think that I just had a misunderstanding of how the frames and runtime stack functioned and my Assignment 4 had some errors that likely held me back.

Challenges

The most challenging portion of this project for me was implementing the DebuggerVirtualMachine and the Debugger Byte Codes. I kept getting errors every line that I had to go along and fix. I think that since I did not have a great foundation from Assignment 4, it hindered my ability to do this project.

Future Work

A way to work on this in the future would be to finish it and make it completely functional. I could then add more debugger functionality. I could also make the output make more sense and be easier to read.