# ICSI 311 Assignment 8 – Start the Interpreter

**If you have bugs or missing features in this, you will need to fix them before you can continue on to new assignments. This is very typical in software development outside of school.**

**You must submit .java files. Any other file type will be ignored. Especially “.class” files.**

**You must not zip or otherwise compress your assignment. Blackboard will allow you to submit multiple files.**

***You must submit buildable .java files for credit.***

Submit, as you have been, all of your existing .java files plus your new interpreter.java.

With the AST complete, we can start getting ready to interpret our AST.

Create a new class (Interpreter.java).

We will need storage for our variables. Create HashMaps that map name (string) to data type for our variables. We will need three hash maps: string->Integer, string->Float and string->string.

In addition to the variable storage, we also need to know where the labels are. Create another hash map string->Node. Use a visitor pattern to walk the statementsnode searching for labeledstatements. Replace the LabeledStatementNode in the AST with the child node. Populate the map with the name and the child node.

One more optimization that we will add here – walk the tree again (two different walks), this time looking for FOR statements and their matching NEXT statements. Add a Node reference to NEXT so that we know where to go when NEXT executes. Add a Node reference to the FOR that references the node AFTER next.

Finally, since it is possible (and, in fact, good form) to put the DATA at the end of your program, we need to pre-process these, too. Walk the tree, removing the DATA statements. Insert their contents into a Java collection that we can use for READ.

We have a little bit of a problem with our design. There is no good way to know what the next instruction should be. We could loop over our StatementsNode, but when we have gosub and goto, we need to go out of order. Add a “Next” member to StatementNode, of type StatementNode. This will indicate the statement to run after this statement.

Finally, one more visitor pattern needs to be run over the StatementsNode – set the “Next” element for each to the next item in the StatementsNode’s statements member. For example, if you have a program:

A=4

PRINT A

The assignment node’s next should point to the print node. The print node’s next should be null.

These four walks should be methods on the interpreter class. One method should call both – you can call it “initialize”.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rubric | Poor | OK | Good | Great |
| Comments | None/Excessive (0) | “What” not “Why”, few (5) | Some “what” comments or missing some (7) | Anything not obvious has reasoning (10) |
| Variable/Function naming | Single letters everywhere (0) | Lots of abbreviations (5) | Full words most of the time (8) | Full words, descriptive (10) |
| Variable Storage | None (0) |  | Significantly Attempted (10) | Correct (15) |
| Label optimization | None (0) |  | Significantly Attempted (10) | Correct (15) |
| For/Next optimization | None (0) |  | Significantly Attempted (10) | Correct (20) |
| NextInstruction | None(0) |  | Significantly Attempted (10) | Correct (15) |
| Read optimization | None(0) |  | Significantly Attempted (10) | Correct (15) |