CS224 – Project 3: Jack Symbol Table

Background

When compiling code and encountering an identifier which represents a variable, we need to know what that variable stands for. We need to know if the variable is a field, an argument to a function, or perhaps a local variable. We need to know the type of the variable as well. All that information can be stored in a symbol table.

Objective

You will complete the SymbolTable class which will implement as described on p236 of the text. Note that you will want to implement the symbol table using two separate hash tables in order to deal with scoping. One hash table will hold variables that have class scope and a second hash table will handle variables that have scope just within the method or function that is being compiled.

You will then modify the CompilationEngine so that every time an identifier is encountered you will print out the type, kind, and index for that variable, and whether the variable is being defined (stored into the symbol table) or being used (retrieved from the symbol table).

Criteria for Success

I have provided a method for the CompilationEngine in the starter code which prints the variable information out to the xml file. I have also provided test files so that you can compare your output to see if it is correct.

Resources

The relevant reading for this project is in Chapter 11 in the nand2tetris text. I have also supplied a zip file containing starter files. You will need to replace the JackTokenizer file with the one that you completed for Project 1, and replace and extend the appropriate methods in the CompilationEngine with your work from Project 2. Do **not** replace the entire ComplilationEngine file since I have included the additional method `symbolInfo` to print to the xml file.

Tools

The TextComparer is supplied on phoenix. Run the TextComparer in the terminal supplying the two files that are being compared as arguments.

Hints

You are given a class STEntry which the fields type, kind, and index. A SymbolTable should be a `Hastable<String,STEntry>` where each identifier is associated with a STEntry containing the information about that identifier.

You will need counters in the SymbolTable class to keep track of how many static, fields, arguments, and local variables have been defined so that you can determine the index values as you defined new variables.

In the CompilationEngine, as you encounter identifiers that represent variables you will need to use the symbolInfo method to write to the xml file. The `use` parameter should be given a value of either Define or Use.

Submission and Assessment

Submit in Canvas your entire JackCompiler project as a single ZIP archive. Your project will be graded using the rubric provided in the Canvas submission.