CS224 – Project 2: Jack Parser

Background

The next step in writing a compiler for the Jack language is to write a parser. We start with a grammar which describes the syntax of the language. The parser then groups the tokens into a syntax tree which we will represent using xml.

Objective

You will complete the CompilationEngine class which will implement a recursive descent parser and produces the syntax tree as xml. The grammar for Jack is given in figure 10.5 of your text.

Criteria for Success

Use the same test files that were provided for the tokenizer. The parse results are given there as well. When you compare your resulting xml files with the test files, using TextComparer they should be identical.

Resources

The relevant reading for this project is Chapter 10 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.

Tools

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

Hints

Each parser method should end with the token advanced to the token **after** the last token that is parsed for that method. It is important to be consistent with this. For some methods this will happen naturally because of a while loop. For other methods you will have to take care to do an extra call to `tokenizer.advance()`

Submission and Assessment

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