**Jack Tokenizer Project Report**

The Jack tokenizer takes .jack files and breaks them down into their tokens. In my approach, I put more emphasis on the constructor and had the file open and processed to make it more workable. In the constructor, methods are called to open the file and separate its symbols, keywords, and other elements by spaces. I then added them to a list after stripping the spaces away and placing each element into its own respective index called ‘tokens’. Once the tokens were all in one place, I could incrementally assign a type to each token and write to a file in the following format:

<’tokens’>

<type> token </type>

<type> token </type>

…

<’/tokens’>

A challenge faced was organizing the file’s contents, particularly ignoring the comments. The majority of time spent on this project was simply ignoring the comments and splitting the larger elements into its smaller pieces (i.e. splat.sploosh(gravy[i]). Once the file was separated into its smaller parts and placed into an array, the project became trivial to complete.

I have a newfound appreciation for the work that goes into the creation of compilers. This project is only a small part of what could be a whole compiler. I also have a deeper understanding of how a real compiler decides what pieces of grammar perform which function.

Below is an example output generated with the help of JackTokenizer: