CPSC326 | Homework 1: Lexer | 2/3/2025 | Arjuna Herbst

I designed tests that involved invalid ID's and symbols, as well as symbols with no space between them since this was something I struggled with. One of the main issues I had when implementing the nextToken method was keeping track of the column correctly. I was getting the correct TokenType and returning it, but with the wrong column value. In the end, I modified the read() method not to increment the column so that I could manually increment column where I needed to and it helped me visualize where it would be when reading through multiple character tokens, especially comments and string values.

I also struggled with doubles with a dot after, e.g. "32.1.", but it turned out mostly what I was doing wrong was throwing errors when I really needed to be breaking out of my read loop to return TokenType DOT.

To describe the tests that I created, it's simplest if I just show what strings I created as input for my Lexer:

"Return struct new And" → This test was created to ensure my Lexer wouldn't have problems with upper vs lowercase characters in reserved words.

"7&&(}xy_0.2" → This test is similar to `tokensWithNoSpaces`, which was provided in the source code. However, that test caused me a lot of issues, so I wanted to create a similar test with a different order of token types to ensure my Lexer could handle input like this.

"var!=b_2\n\"hi\" 8.23." → This test combines a bunch of different token types, including keywords, newlines, doubles followed directly by a DOT. These were all things I struggled with handling while implementing nextToken so I wanted to make a test that would combine all of them

" %" → This is the first of 2 "negative" test cases, it involves skipping past a whitespace and then handling an invalid symbol.

"a_0xy_ $" → Lastly, I made a test to ensure my lexer can handle ID's with multiple '_', as well as another invalid symbol.