Brandon Watnabe
921988506
February 16, 2022

Assignment 2 Documentation

Github Repository

https://github.com/sfsu-csc-413-fall-2022-roberts/assignment-2---lexer-BrandonWatanabeStudentAccount.git

Project Introduction and Overview

For this project, I had to modify a Lexer class to add certain tokens for it to recognize. This is to teach us about Lexical Analysis and how compilers work.

Scope of Work

Task		Completed
Add tokens to tokens file		X
Run TokensSetup.java to auto generate TokenTypes.java and Tokens.java		X
Update Token class to include the line number		X
	public Token(int leftPosition, int rightPosition, int lineNumber, Symbol sym)	X
	public int getLineNumber()	X
Modify methods in Lexer class		X
	public Token newIdToken(String id, int startPosition, int endPosition, int lineNumber)	X
	public Token newNumberToken(String number, int startPosition, int endPosition, int lineNumber)	X
	public Token makeToken(String s, int startPosition, int endPosition, int lineNumber)	X
	public Token nextToken()	X
	public static void main(String [] args)	X

Execution and Development Environment

I used the IntelliJ IDE on my Windows to complete this assignment and tested in both the IDE and shell.

Compilation Result

Using the instructions provided in the assignment one specification:

> javac lexer/setup/TokenSetup.java

> java lexer.setup.TokenSetup

> javac lexer/Lexer.java

```
> java lexer.Lexer simple.x

READLINE:   program { int i int j

Program     left:0       right:6       line:1       Program

{           left:8       right:8       line:1       LeftBrace

int         left:10      right:12      line:1       Int

i           left:14      right:14      line:1       Identifier

int         left:16      right:18      line:1       Int

j           left:20      right:20      line:1       Identifier

READLINE:   i = i + j + 7

i           left:0       right:0       line:2       Identifier

=           left:2       right:2       line:2       Assign

i           left:4       right:4       line:2       Identifier

+           left:6       right:6       line:2       Plus

j           left:8       right:8       line:2       Identifier

+           left:10      right:10      line:2       Plus
```

7 left:12 right:12 line:2 INTeger

READLINE: j = write(i)

j left:0 right:0 line:3 Identifier

= left:2 right:2 line:3 Assign

write left:4 right:8 line:3 Identifier

(left:9 right:9 line:3 LeftParen

i left:10 right:10 line:3 Identifier

) left:11 right:11 line:3 RightParen

READLINE: }

} left:0 right:0 line:4 RightBrace

No error messages or warnings were displayed, and the application ran as expected.

Assumptions

I assumed that the provided Classes and methods worked as intended and that I would not have to modify the majority of them.

Implementation

Lexer

The Lexer class was given to me with the main method commented out. I was required to implement this method, allow it to accept any file using commandline arguments at runtime, and reformat how the output is printed to the console. I also needed to add functionality to print out a line number. To start, I updated the method to check if any arguments were given, if not, it will

print out a usage message to let the user know that they need to add a file path. Then, I use args[0] as the file name so that the file name is not hard coded anymore to allow the user to use any file they want. I then had to implement the line number into all the methods that make the tokens and handle them so that the line number would be added to each token when processed. After that, I formatted the output to make it more legible and added the line number and kind of token as well.

Token

The Token class was also partially implemented for me. All I needed to do was to add the line number in the constructor and create a getLineNumber method.

Code Organization

I did not change the organization of the code from the original.

Results and Conclusion

This project taught me how lexers work and how all the lines of code that I write are interpreted and handled under the hood. This was an interesting insight into how it all works.

Challenges

The most challenging part for me was implementing the Utf16StringLit and TimestampLit to be read as tokens. I am still working on how to do that and therefore the assignment is not completed yet. Another part that took some time to figure out was formatting the printf statement as I had forgotten how to use printf properly.

Future Work

I think that a way to further develop this project would be to add in functionality for more tokens such as more reserved words and floats.