https://github.com/917wiszniewskibianca/lftc ->**scanner.py**

1. __init__(self, file_info)

Purpose: Initializes a Scanner object.

Parameters:

file_info: The information about the file to be scanned.

Actions:

Initializes instance variables like _lineCount, _file, _SymbolTable, _programInternalForm, and _tokens.

Calls readTokens() and scan() methods.

Catches and prints a ValueError if it occurs during initialization.

2. readTokens(self)

Purpose: Reads tokens from the "token.in" file and stores them in the _tokens list.

Actions:

Opens the "token.in" file, reads each line, and appends the stripped line to the _tokens list.

3. writeToFile(self)

Purpose: Writes the Program Internal Form (PIF) and Symbol Table (ST) to respective output files.

Actions:

Writes the elements of _programInternalForm to "PIF.out".

Writes the Symbol Table as a string to "ST.out".

4. scan(self)

Purpose: Scans the file, tokenizes each line, classifies tokens, and updates the Program Internal Form and Symbol Table.

Actions:

Splits the file into lines, tokenizes each line, classifies tokens, and updates the Program Internal Form and Symbol Table accordingly.

Calls the writeToFile() method to save the results to output files.

5. tokenizeLine(self, line_string)

Purpose: Tokenizes a given line.

Parameters:

line_string: The line to be tokenized.

Returns: A list of tokens.

Actions:

Uses regular expressions to split the line into tokens, filters out spaces, and processes special cases like '==' and '<='.

("[^"]+" | [a-zA-Z0-9]+|[^a-zA-Z0-9"\s]+) -> regex for splitting lines
"[^"]+" -> match a string between double quotes
[a-zA-Z0-9]+ ->match alphanumeric char
[^a-zA-Z0-9"\s]+ -> looks for one or more chars that are not alphanumeric or quotes
[^] – everything that is not
^" [] – must start with "
$ must end with
*** 0 or more**

6. classifyToken(self, token)

Purpose: Classifies a token as an identifier, constant, string constant, character, or integer.

Parameters:

token: The token to be classified.

^[a-zA-Z]+[a-zA-Z0-9]*$ ->match a variable name that should start with a letter and can be followed by one or more alphanumeric values
'^"[a-zA-Z0-9\s]+"$' -> match a string constant
^\'[a-zA-Z0-9\'$]' ->match a single char
'^0$|^(\+|-)?[1-9][0-9]*$' -> match a digit, either 0 or a +/- non zero digit

Returns:

1 for an identifier.

2 for a string constant.

3 for a character.

4 for an integer.

0 if the token cannot be classified.