**FLCD Scanner Documentation**

**Link to GitHub repository**: https://github.com/ralucaioanaianos/Formal-Languages-and-Compiler-Design/tree/main/lab\_3

**Program Requirements:**

Statement: Implements a scanner (lexical analyzer): Implement the scanning algorithm and use ST from lab 2 for the symbol table.

**Scanner:**

The Scanner receives the text file where the program is written as String.

separators: the list of separators from “token.in”

operators: the list of operators from “token.in”

reservedWords: the list of reserved words from “token.in”

pif: tokens correspond to value -1, identifiers to the position in the table, int constants to -2 and string constants to -3

symTable: the SymTable that contains identifiers and constants

fileToCheck: the file with the program

tokens: all the tokens from the program

**Methods:**

readFile(self):  
 *"""  
 parses the given file line by line and tokenizes each line* ***:return****: the list of errors in the file, each error indicates its line and the syntax error  
 """*

readTokenFile(self):  
 *"""  
 parses “token.in”, extracts all the tokens and separates them into operators, separators and reserved words.* ***:return****:-  
 """*

writeToPifFile(self):  
 *"""  
 writes what is needed in “pif.out”* ***:return****:  
 """*

isReservedWord(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is a reserved word, False otherwise  
 """*

isIdentifier(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is an identifier, False otherwise  
 """*

isStringConstant(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is a string constant, False otherwise  
 """*

isConstant(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is a constant, False otherwise  
 """*

isPartOfOperator(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is part of an operator, False otherwise  
 """*

isSeparator(self, token):  
 *"""* ***:param*** *token: the token to verify* ***:return****: True if token is a separator, False otherwise  
 """*

tokenizeLine(self, line: str):  
 *"""  
 takes every line, extracts every token from it, analyzes it and verifies whether there are syntax errors or not* ***:param*** *line: the line to tokenize* ***:return****: the syntax error if it exists on the line, None otherwise  
 """*

**SymTable:**

getIdentifiers(self):  
 *"""* ***:return****: the table of identifiers  
 """*

getConstants(self):  
 *"""* ***:return****: the table of constants  
 """*

addIdentifierToTable(self, key):  
 *"""* ***:param*** *key: the key to add to the table of identifiers* ***:return****: the index where key was inserted in the table of identifiers  
 """*

addConstantToTable(self, key):  
 *"""* ***:param*** *key: the key to add to the table of constants* ***:return****: the index where key was inserted in the table of constants  
 """*

getIdentifierPosition(self, key):  
 *"""* ***:param*** *key: the key to find in the table of identifiers* ***:return****: the index where the key is found in the table of identifiers, None if it does not exist  
 """*

getConstantPosition(self, key):  
 *"""* ***:param*** *key: the key to find in the table of constants* ***:return****: the index where the key is found in the table of constants, None if it does not exist  
 """*

\_\_str\_\_(self):  
 *"""* ***:return****: the string representation of SymTable  
 """*