```
https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-915-Micu-
AlexiaClaudia/tree/main/1-Mini-Language-And-Scanner

SCANNER CLASS

COMPONENTS

- I chose to have 4 lists to classify tokens: operators, separators, reservedWords
or types (for differentiating between identifiers and constants)

- My constants and identifiers are held in separate symbol tables -> contantTable,
identifierTable

- I have a program internal form

- and a boolean to check for lexical errors (and for each paranthesis type)


METHODS

Scanner(filepath) - initializes all components that are needed
ReadFile() - returns the entirea read file
CreateListOfTokens - reads the file and extracts the apparitions of this pattern
in the file. Then it filters out the empty strings.
Detect: decides if a token is a - reserved word (or array type) - these are
clasified together
                                - a constant
                                - an identifier (could still be a constant if it's
in quotes)
                                - an operator
                                - a separator
                                - none of the above (it could still be a constant
if it's in quotes)

SCAN method:
 - keeps track of the lexical corectness, location and paranthesis + quote
managing
 - iterates the list of tokens, detects each token
 - FOR : - identifiers: if it's in quotes -> constant
                        if it's not in quotes and already exists in symbol table
or the previous element is a type-> all good
                        if it's not in quotes -> lexical error
         - constants: adds them if they don't already exist
         - separator: if it's in quotes -> const
                      handles paranthesis
         - operator : if it's in quotes -> const
         - reservedWords : if it's in quotes -> const
         - else if it's not in quotes -> unidentified and lexical error
Based on the islexicallyCorrect, in quotes and paranthesis handlers, it prints a
message on whether the file was lexically correct.
```