https://github.com/DragosteSergiu/Facultate/blob/main/Lab3FLCD.py

DOCUMENTATION

```
-------------------------------------------------------------------------------------------------------------------------------|
CLASS: 'Scanner'
-------------------------------------------------------------------------------------------------------------------------------|
SUMMARY - It is used for tokenizing the source code written in the custom made programming language.
FIELDS      - The class 'Scanner' is build using five fields. This fields are the following:
            - 'Path', is a field that represents the actual path of the file in which the source code to be scanned is written.
            - 'Handler', is a field that takes as argument an object of type 'Handler', which is used to perform the reading of the
              special characters, reserved words, and operators stored in the file 'Token.in', also to perform the tokenization of the input file,
              verification of the identifiers and constants and to retrieve the errors found in the program.
            - 'SymbolTable', is a field that takes as argument an object of type 'SortedTable', which is used to store all identifiers
              that are found.
            - 'ProgramInternalForm' is a field that takes as argument an object of type 'ProgramInternalForm', which is used to store all tokens
              that are found.
            - 'ErrorsList' is a field that takes as argument an object of type 'ErrorsList', which is used to store all irregularities found.

OUTPUT - The execution of the main method of this class, named 'execute', returns three entities, which are the following:
            - 'SymbolTable', is represented as an object of type 'SortedTable', which is composed of elements of type 'Element',
              and its usage is to store variables and constants found in the program.
              The values that are returned by this field are written in file 'ST.out'.
            - 'ProgramInternalForm', is represented as an object of type 'ProgramInternalForm', which is composed of elements of type 'PIFElement',
              and its usage is to store all tokens that occur in the program, with their value in 'Symbol Table' and in file 'Token.in'.
              The values that are returned by this field are written in file 'PIF.out'.
            - 'ErrorsList', is represented as an object of type 'ErrorsList', which is composed of elements of type 'Error',
              and its usage is to store all irregularities found in the source code.
              The values that are returned by this field are written in file 'errors.out'.

METHODS     - The class has a constructor and three methods, which are the following:
            - 'execute()', is the main method of the class, it takes no arguments, and basically it performs all the logic of the scanner, such as:
              reads line by line the program written in the file given as parameter for field 'path', calls methods from the class 'Handler',
              receives and processes values returned by the called methods, and updates the fields 'SymbolTable', 'ProgramInternalForm', and 'ErrorsList'.
```

- 'symbolTable', is a getter that returns the value of the field 'SymbolTable'.
- 'ProgramInternalForm', is a getter that returns the value of the field 'ProgramInternalForm'.

CLASS: 'Handler'

SUMMARY - It is used to perform the actual process of tokenization, for verification of identifiers and for error detection.
FIELDS - The class 'Handler' is build using one fields. The field is the following:
- 'filename', is a field that represents the name of the file where the reserved words, separators and operators are written.
OUTPUT - The main method of this class is represented by the method named 'parse', which splits a given string in tokens and returns a list of tokens and a list of errors wrapped in an object of type 'ErrorsList'.
METHODS - The class has a constructor and fourteen methods, which are the following:
- 'readTokens()', is a method which reads the content from the file which name is stored in the only field of the class, and it stores the content in the dictionary named 'TOKENS'.
- 'writeInFile()', is a method that writes a given string in a specified file, it takes as parameters two string which are 'filename', representing the name of the file' and 'string', representing the content that will be written in the file.
The method is called just before the execution of the main method 'execute()' from class 'Scanner' is over.
- 'updateProgramInternalForm()', is a method which syncronizes the values of the elements from a list of an object 'ProgramInternalForm' with the values from a list of an object 'SortedTable', it takes as parameters two list.
- 'parse()', is the main method of the class, and it is used for tokenization of the lines from given file, for error detection. It takes three parameters: 'line', represents the line to be tokenized, 'number', represents the number of the line to be tokenized, and 'identifiers', represents all identifiers which are previously stored in the 'SymbolTable'. The function returns a list of tokens and an object of type 'ErrorsList'.
- 'parseComposedSeparators()', is a method which is used in the main method of this class, it performs an additional tokenize, which verifies and corrects the list of tokens if there exists an composed separator such as ':=', '==', etc. It takes as parameter one values, which represents the list of tokens and returns a modified version of that list.
- 'parseOperatos()', is a method which is used in the main method of this class, it performs an additional tokenize, which verifies if there is an assign statement that takes exactly one constant numeric value on the right hand side, and if it is true performs and a concatenation between the sign(if there is a sign '+', or '-') and the constant numeric value. It takes as parameter one values, which is the list of tokens and returns a modified version of that list.
- 'verifyErrors()', is a method which is used for calling all the verification methods of the class. The method takes the same parameters as method 'parse() 'and it returns and object of type 'ErrorsList'. This method is called in the main method of this class, 'parse()'.
- 'verifyAssign()', is a method which verifies if the tokens of a given

line represent an assign statement, and if it is true, it checks if
the identifiers are already stored in 'SymbolTable' or if the format of the constants is valid. The function takes as parameter two values,
'tokens' and 'identifier'. It returns an object of type 'Error' if there is the case. This method is used in the method 'verifyErrors()'.
- 'verifyIdentifiers()', is a method which verify if a given string is a valid identifier, and if it is not, returns an object of type 'Error'.
It takes as parameter one value, which is the list of tokens of the given line.
- 'verifyNumberOfParenthesis()', is a method which verifies if the parenthesis are used in a proper way, if they are closed correctly.
It returns an object of type 'Error' if there is an irregularity in the writing of the parenthesis.
It takes as parameter one value, which is the list of tokens of the given line. It is used in method 'verifyErrors()'
- 'verifyVariableDeclaration()', is a method which verifies if the given tokens form a variable declaration statement, and if it true, it
checks if the syntax is correct. It takes as parameter one value, which is the list of tokens of the given line, and returns an object of type
'Error'. It is used in methods 'verifyErrors()'.
- 'isValidIdentifier()', is a method which performs a verification of a given string, it checks if the string respects all constraints of a valid
identifier. It takes as parameter one string and it returns a boolean value.
- 'isValidNumericConstant()', is a method which performs a verification of a given string, it checks if the string respects all constraints of a
valid numeric constant. It takes as a parameter one string and it returns a boolean value.
- 'isValidStringConstant()', is a method which performs a verification of a given string, it checks if the string respects all constraints of a
valid string constant. It takes as a parameter one string and it returns a boolean value.