Github link: https://github.com/dacianf/FLDC

Class Controller

Implementation details:
    Reads the input file with the code line by line, gets each token from each line and checks if it is part of the specifications.
    If the token is:
        * an identifier or a constant it will be added in both symbol table and program internal form;
        * a reserved word, it will be added just in the internal form
    If it is not one from the above it will throw an error with the error's location
    After the whole input file is parsed, the `program internal form` and the `symbol table` will be written into files.

Methods:
    * public run(String fileName)
        -PRE:
            fileName - a valid sequence of characters which describes a valid file
        -POST:
            given file will be read and split in tokens
        -RETURN:
            nothing - if the file is lexical correct
            throws error - if there exists a lexical error

    * public saveData()
        -PRE:
            -
        -POST:
            Program internal form will be saved in pif.out
            Symbol table will be saved in st.out
        -RETURN:
            -

Class Scanner

Implementation details:
    Scanner uses method `getTokens` to get the tokens out of a given line. For doing this, it gets the characters from the line, splits them and checks it they match of the categories from the specifications.

Methods:
    * public getTokens(String line)

- PRE:
  token - a valid sequence of characters
- POST:
  given line is parsed from the position where this function was when it return the result at last call
- RETURN:
  token - a sequence of characters which represent the found token

Class Specification

Implementation details:
Specification class contains a list with each separators, operators and reserved words and a dictionary which contains each word with its codification.

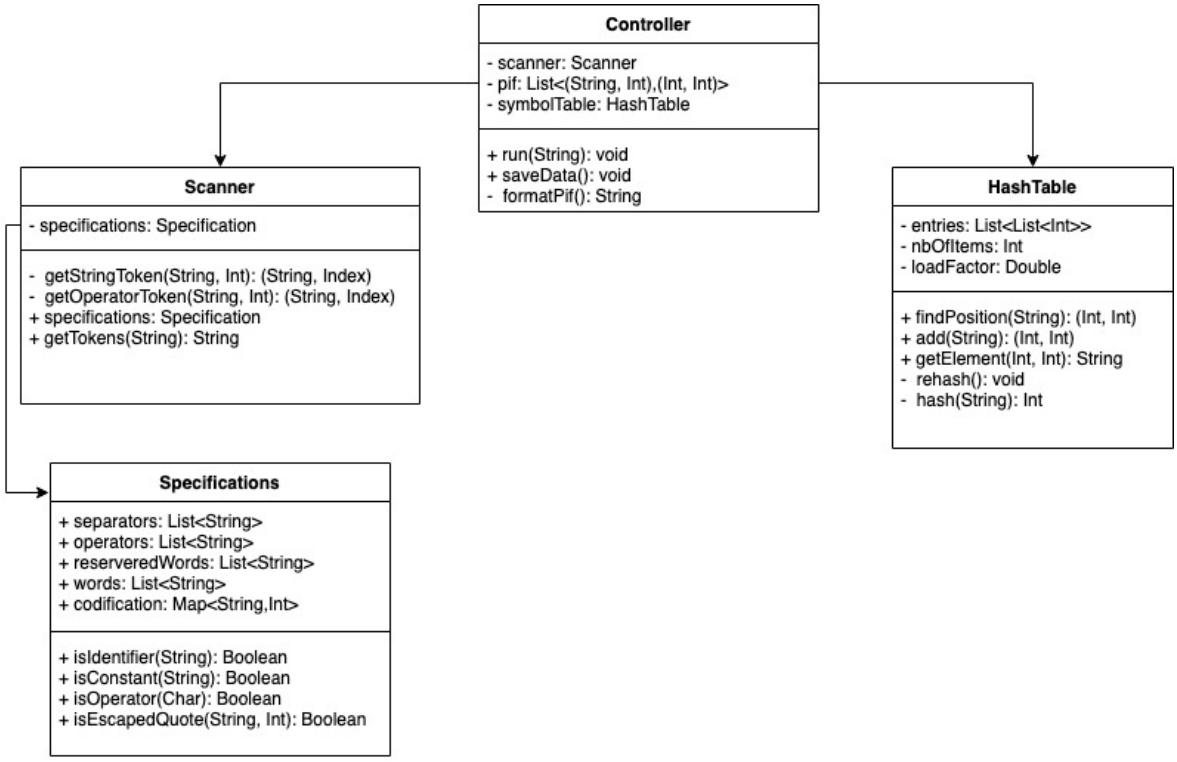
Methods:
* public getTokens(String token)
  - PRE:
    token - a valid sequence of characters
  - POST:
    -
  - RETURN:
    true - if the given token match the regex for an identifier
    false - otherwise

* public isConstant(String token)
  - PRE:
    token - a valid sequence of characters
  - POST:
    -
  - RETURN:
    true - if the given token match the regex for a constant
    false - otherwise

* public isOperator(String char)
  - PRE:
    token - a valid character
  - POST:
    -
  - RETURN:
    true - if the given char is in the list of operators
    false - otherwise

* public isEscapedQuote(String line, Integer index)
  - PRE:

line - a valid sequence of characters
index - a positive integer < line.size
- POST:
-
- RETURN:
true - if the given position in the line contains the escape quote
false - otherwise



Correct example:

Input:

```
main(){
    [int] numbers = [100]
    int n
    citeste "C", n
    int tmp = n
    int i = 0
    while tmp > 0 {
        citeste "S", numbers[i]
        i++
        tmp --
    }
```

```
        int sum = 0
        for int i = 0; i < maxLenght; i++ {
            sum += numbers[i]
        }
        afiseaza sum
    }
```

Output:

pif.out
Program internal form:
(('main', 33), (-1, -1))
(('(', 6), (-1, -1))
((')', 7), (-1, -1))
(('{', 4), (-1, -1))
(('[', 2), (-1, -1))
(('int', 35), (-1, -1))
((']', 3), (-1, -1))
((' ', 9), (-1, -1))
(('numbers', 0), (17, 0))
((' ', 9), (-1, -1))
(('=', 21), (-1, -1))
((' ', 9), (-1, -1))
(('[', 2), (-1, -1))
(('100', 1), (27, 0))
((']', 3), (-1, -1))
(('int', 35), (-1, -1))
((' ', 9), (-1, -1))
(('n', 0), (1, 0))
(('citeste', 37), (-1, -1))
((' ', 9), (-1, -1))
(('"C"', 1), (23, 0))
((',', 13), (-1, -1))
((' ', 9), (-1, -1))
(('n', 0), (1, 0))
(('int', 35), (-1, -1))
((' ', 9), (-1, -1))
(('tmp', 0), (10, 0))
((' ', 9), (-1, -1))
(('=', 21), (-1, -1))
((' ', 9), (-1, -1))
(('n', 0), (1, 0))
(('int', 35), (-1, -1))
((' ', 9), (-1, -1))

(('i', 0), (23, 1))
((' ', 9), (-1, -1))
(('=', 21), (-1, -1))
((' ', 9), (-1, -1))
(('0', 1), (24, 0))
(('while', 42), (-1, -1))
((' ', 9), (-1, -1))
(('tmp', 0), (10, 0))
((' ', 9), (-1, -1))
(('>', 23), (-1, -1))
((' ', 9), (-1, -1))
(('0', 1), (24, 0))
((' ', 9), (-1, -1))
(('{', 4), (-1, -1))
(('citeste', 37), (-1, -1))
((' ', 9), (-1, -1))
(('"S"', 1), (12, 0))
((',', 13), (-1, -1))
((' ', 9), (-1, -1))
(('numbers', 0), (17, 0))
(('[', 2), (-1, -1))
(('i', 0), (23, 1))
((']', 3), (-1, -1))
(('i', 0), (23, 1))
(('++', 29), (-1, -1))
(('tmp', 0), (10, 0))
((' ', 9), (-1, -1))
(('--', 30), (-1, -1))
(('}', 5), (-1, -1))
(('int', 35), (-1, -1))
((' ', 9), (-1, -1))
(('sum', 0), (16, 0))
((' ', 9), (-1, -1))
(('=', 21), (-1, -1))
((' ', 9), (-1, -1))
(('0', 1), (24, 0))
(('for', 41), (-1, -1))
((' ', 9), (-1, -1))
(('int', 35), (-1, -1))
((' ', 9), (-1, -1))
(('i', 0), (23, 1))
((' ', 9), (-1, -1))
(('=', 21), (-1, -1))
((' ', 9), (-1, -1))
(('0', 1), (24, 0))
((';', 12), (-1, -1))

((' ', 9), (-1, -1))
(('i', 0), (23, 1))
((' ', 9), (-1, -1))
(('<', 19), (-1, -1))
((' ', 9), (-1, -1))
(('maxLenght', 0), (2, 0))
((';', 12), (-1, -1))
((' ', 9), (-1, -1))
(('i', 0), (23, 1))
(('++', 29), (-1, -1))
((' ', 9), (-1, -1))
(('{', 4), (-1, -1))
(('sum', 0), (16, 0))
((' ', 9), (-1, -1))
(('+=', 31), (-1, -1))
((' ', 9), (-1, -1))
(('numbers', 0), (17, 0))
(('[', 2), (-1, -1))
(('i', 0), (23, 1))
((']', 3), (-1, -1))
(('}', 5), (-1, -1))
(('afiseaza', 38), (-1, -1))
((' ', 9), (-1, -1))
(('sum', 0), (16, 0))
(('}', 5), (-1, -1))

st.out
Symbol table
(represented as hash table)
hash: 0
	[]
hash: 1
	['n']
hash: 2
	['maxLenght']
hash: 3
	[]
hash: 4
	[]
hash: 5
	[]
hash: 6
	[]
hash: 7
	[]
hash: 8

	[]
hash: 9
	[]
hash: 10
	['tmp']
hash: 11
	[]
hash: 12
	['"S"']
hash: 13
	[]
hash: 14
	[]
hash: 15
	[]
hash: 16
	['sum']
hash: 17
	['numbers']
hash: 18
	[]
hash: 19
	[]
hash: 20
	[]
hash: 21
	[]
hash: 22
	[]
hash: 23
	['"C"', 'i']
hash: 24
	['0']
hash: 25
	[]
hash: 26
	[]
hash: 27
	['100']
hash: 28
	[]
hash: 29
	[]
hash: 30
	[]
hash: 31

[]

Example with lexical error:

Input:
```
main(){
    a = "
    int c = 2
    for int i=0, i < 2a; i++{
        afiseaza(i)
    }
}
```

Output:
```
raise Exception:
        Lexical error: Unknown token '2a' at line 4 on position 14
```