https://github.com/LaviniaGalan/FLCD/tree/master/Lab3

In my implementation, I use the following classes:

- **Symbol Table –** represented as a hash table, for which the collisions are solved by using a linked list. The hash function is the sum of the ascii code of the characters from the token, modulo M.

- **PIF –** the class for the Program Internal Form entity. It is represented as a list of Pairs of the token code and their position in the Symbol Table (the position is (-1, -1) if the token is a keyword, a separator or an operator and thus, it does not appear in the symbol table).
  - It has only one method:

    - void add(Integer code, Pair<Integer, Integer> position): adds in the program internal form the pair (code, position).

- **Classifier –** it is used for classifying a token, which can be a reserved word, a separator, an operator, an identifier or a constant.
  - In the Classifier class, there are 3 lists, each of them containing the keywords, the separators and, respectively, the operators that appear in the language specifications.
  - It also contains a Map which is populated in the constructor and associates each atom with an integer number (code) - the codification table.
  - The methods used for classification are:

  - boolean isReservedWord(String token) - returns true if the token given as parameter is a reserved word and false otherwise.

  - boolean isOperator(String token) - returns true if the token given as parameter is an operator and false otherwise.

  - boolean isSeparator(String token) - returns true if the token given as parameter is a separator and false otherwise.

  > (For these 3 methods, the checking is done by verifying if the token appears in the list of corresponding atoms.)

  - boolean isIdentifier(String token) - checks if a token is identifier and returns true if the token respects the rule for identifiers and false otherwise.

  - boolean isConstant(String token) - checks if a token is constant (numeric, character or string); returns true if the token respects the rules for constants and false otherwise.

  > (The last 2 methods check the matching of the token with a regex expression.)

  - Other methods:

  - int getCode(String token) - returns the code of the given token from the codification table.

- **Scanner –** it scans the source code, extract the tokens from it and creates the symbol table and the program internal form. The tokens are retrieved by scanning the file line by line, and then each line, character by character.
  - In the Scanner class, there are a Symbol Table, a PIF and a Classifier.
  - The methods are:

- void scanFile(String fileName) - scans the file given as parameter line by line, creating the list of tokens from the file.

- List<String> tokenize(String line) - extracts all the tokens from the line given as parameter and returns a list which contains these tokens.

- String extractStringConstant(String line, int currentPosition) - this method is called when at the currentPosition in the line, there is a quote mark ("). It parses the rest of the line until the matching quote mark is found or until there are no more characters in the line. It returns the substring that starts at that quote mark and ends at the matching quote mark or at the end of the line.

- String extractMinus(String line, int currentPosition) - this method is called when at the currentPosition in the line, there is a minus sign(-). For a minus, there are two possibilities: it is an operator or it is a part of a numerical constant. The method analyses these cases and return "-" if it is an operator or the numerical constant, if that is the case.

- String extractOperator(String line, int currentPosition) - this method checks if at the currentPosition in the line, there is a symbol that can be an operator, that is, a symbol that is an operator itself, or a symbol that is a part of a compound operator. The method performs a look-ahead and retrieves the compound operator, if that is the case. It returns the found operator, or empty string if there is no operator.

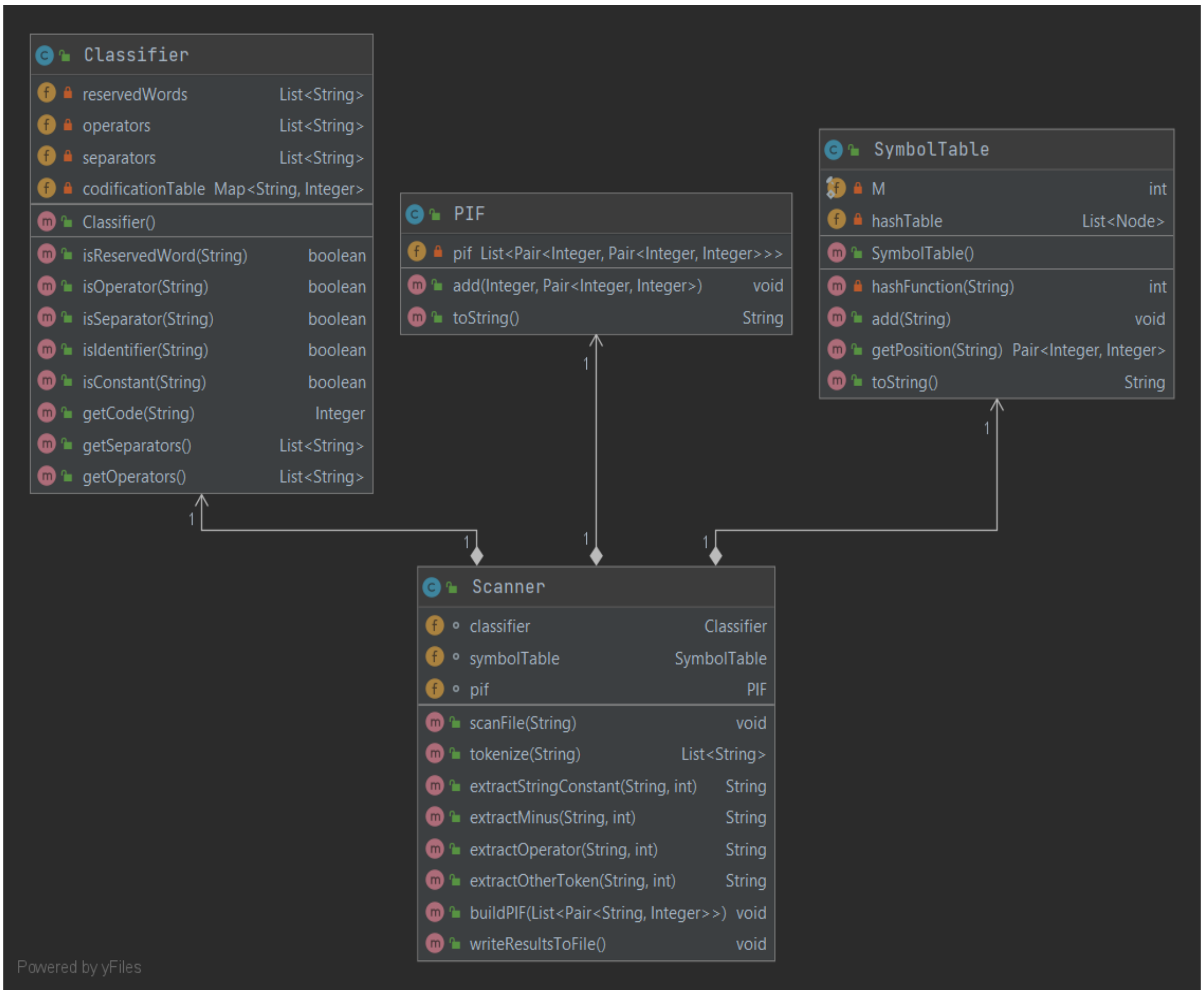
- String extractOtherTokens(String line, int currentPosition) - When a line is tokenized, if the character that is currently processed haven't been in the situations described above, then this method is called. It extracts the token that starts at the currentPosition and ends at the first separator or operator found.

> The tokenize method scans the line using an index (the current position in that line). It analyses the character from the current position and depending on its type, it calls the methods that extract and return a token. This token is added in the list of tokens and the index of the current position will be placed immediately after that token in the line.

- boolean buildPIF(List<Pair<String, Integer>> tokens): the list given as parameter consists of pairs of String (the actual text of the token) and an Integer, which denotes the line of the source file at which that token appears. It checks every token from the list as follows: if the token is identifier, separator or operator, it is added in PIF with the code from the codification table and the position (-1, -1). If the token is identifier or constant, it is added in the Symbol Table (if it has not been added already) and in PIF with the code 0 for identifier, 1 for constants and the position in the ST. If the token could not be placed in one of these 5 categories, then it is invalid, and a message will be displayed, saying that an error was found at the line at which the invalid token was found. If an error was found, it returns true. Else, returns false.

- void writeResultsToFile(): this method writes the symbol table and the PIF in the corresponding output files.

The class diagram is:


Classifier
reservedWords List<String>
operators List<String>
separators List<String>
codificationTable Map<String, Integer>
Classifier()
isReservedWord(String) boolean
isOperator(String) boolean
isSeparator(String) boolean
isIdentifier(String) boolean
isConstant(String) boolean
getCode(String) Integer
getSeparators() List<String>
getOperators() List<String>
PIF
pif List<Pair<Integer, Pair<Integer, Integer>>>
add(Integer, Pair<Integer, Integer>) void
toString() String
SymbolTable
M int
hashTable List<Node>
SymbolTable()
hashFunction(String) int
add(String) void
getPosition(String) Pair<Integer, Integer>
toString() String
1
1
1
1
1
1
Scanner
classifier Classifier
symbolTable SymbolTable
pif PIF
scanFile(String) void
tokenize(String) List<String>
extractStringConstant(String, int) String
extractMinus(String, int) String
extractOperator(String, int) String
extractOtherToken(String, int) String
buildPIF(List<Pair<String, Integer>>) void
writeResultsToFile() void
Powered by yFiles