https://github.com/biancaszekely32/FLCD

Documentation – Lab3

➢ Language

The Language class defines the rules and properties of the programming language being analyzed. It includes information about operators, separators, reserved words, and provides methods to determine the type of a token.

Fields:

- *operators*: A list containing operators like addition, subtraction, multiplication, division, modulus, comparison operators, logical operators, etc.
- *separators*: A list of characters used to separate tokens, such as brackets, parentheses, braces, colons, semicolons, commas, quotes, spaces, newlines, and tabs.
- *reservedWords*: A list of predefined words in the language, such as control flow statements (if, else, for, while), I/O operations (read, write), data types (integer, string), etc.

  tokens: A hash map that associates tokens (e.g., identifiers, constants) with their respective codes. It facilitates easy retrieval of the code for a given token.

Methods:

- *loadListOfTokens()*: Initializes the possibleTokens hash map by associating each token type with a unique code. It also assigns codes to reserved words, operators, and separators.
- *getCode(String token)*: Returns the code associated with a given token.
- *isOperator(String token)*: Checks if a token is an operator.
- *isPartOfOperator(char op):* Determines if a character is part of an operator, considering cases like ">= or <=".
- *isSeparator(String token):* Checks if a token is a separator.
- *isReservedWord(String token):* Determines if a token is a reserved word.
- *isIdentifier(String token):* Validates if a token conforms to the language's identifier pattern; It checks if the identifier starts with a letter from a-z or from A-Z and it can be followed by another letters or by digits, and it can contain '_'.
- *isConstant(String token):* Checks if a token matches any of the defined constant patterns (integer, character, or string).

  -integer: can be either 0, or can be a number formed with digits, or by positive or negative numbers formed with digits

  -character: it could be a letter, a digit or a any of the charcaters _?!#*,./%-+=<>;)(}{

  -string: it can contain charcters or integers, or any of these characters _?!#*,./%-+=<>;)(}{

Program Internal Form

The ProgramInternalForm class represents the Program Internal Form (PIF) used in lexical analysis. It manages a list of pairs, where each pair contains a token and a pair of codes.

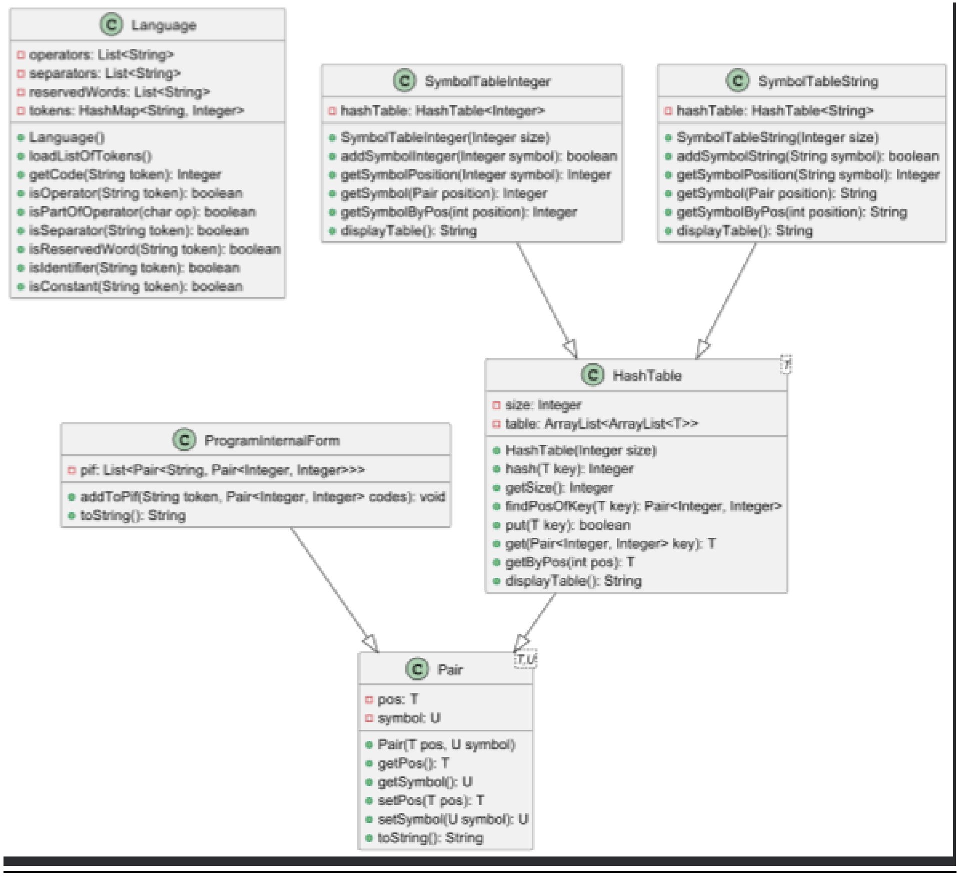
Fields:

- *pif:* A list containing pairs of tokens and their associated codes.

Methods:

- *public void addToPif(String token, Pair<Integer, Integer> codes):* Adds a new pair to the Program Internal Form.
- *@Override public String toString():* Generates a string representation of the Program Internal Form.

Class Diagram


Language
operators: List<String>
separators: List<String>
reservedWords: List<String>
tokens: HashMap<String, Integer>
Language()
loadListOfTokens()
getCode(String token): Integer
isOperator(String token): boolean
isPartOfOperator(char op): boolean
isSeparator(String token): boolean
isReservedWord(String token): boolean
isIdentifier(String token): boolean
isConstant(String token): boolean
SymbolTableInteger
hashTable: HashTable<Integer>
SymbolTableInteger(Integer size)
addSymbolInteger(Integer symbol): boolean
getSymbolPosition(Integer symbol): Integer
getSymbol(Pair position): Integer
getSymbolByPos(int position): Integer
displayTable(): String
SymbolTableString
hashTable: HashTable<String>
SymbolTableString(Integer size)
addSymbolString(String symbol): boolean
getSymbolPosition(String symbol): Integer
getSymbol(Pair position): String
getSymbolByPos(int position): String
displayTable(): String
HashTable
T
size: Integer
table: ArrayList<ArrayList<T>>
HashTable(Integer size)
hash(T key): Integer
getSize(): Integer
findPosOfKey(T key): Pair<Integer, Integer>
put(T key): boolean
get(Pair<Integer, Integer> key): T
getByPos(int pos): T
displayTable(): String
ProgramInternalForm
pif: List<Pair<String, Pair<Integer, Integer>>>
addToPif(String token, Pair<Integer, Integer> codes): void
toString(): String
Pair
T,U
pos: T
symbol: U
Pair(T pos, U symbol)
getPos(): T
getSymbol(): U
setPos(T pos): T
setSymbol(U symbol): U
toString(): String