In addition to the old SymbolTable we added a new scanner for the language.

This scanner implements a series of regular expressions in order to check for tokens, identifiers and constants in the written language code.

Language Scanner(filename,tokenFileName)

Attributes:

tokens: List<String> = we store in the code all the tokens read from the tokenFile

programContent: String = we store in this variable the whole program that we want to scan

symbolTable: SymbolTable = the symbol table which stores identifiers and constants scanned from the program

pifTable: List<Pair<String<Pair<Integer,Integer>>> = a list in which we store all the read tokens and symbolTable values, in the order in which they were scanned in the program. If we scan an identifier and constant multiple times, it will still be added with its according location in the pifTable

currentIndex: Int = starting with 0 we use it to iterate our program in the scanning process

lineCounter: Int = starting with 1, we use it to denote the currentLine the scanner is at. It is used when displaying where a lexical error took place

scanProgram = scans the whole program until we reached its end, using a series of checks

The checks are done using regular expressions, and until we can get to that check we will skip unnecessary whitespace using the takeoverSpace() function

verifyStringConstant regex = ^\"[a-zA-Z0-9_ ]*\"

^ = start of string

\" = matches the starting quote, same for the ending quote

[a-zA-Z0-9 _]* = any alphabetical,numerical, space or underscore for 0 or more times

verifyIntConstant = here, I used a more classical approach. I check if the currentIndex points in the programContent at a digit, or at a plus/minus sign followed by a digit. Until we reach the end of this rule, we construct our number from the start. In the end, the number is added to the consant table in the SymTable. In case of the number starting with 0, the calculation rule will just ignore it until we reach a non-zero digit.

verifyTokenFromList = the tokens from the tokenFile are then checked and are added in the pifTable with indices (-1,-1) since they are not added in the symTable like the constants or identifiers.

verifyIdentifier = ^[a-zA-Z_][a-zA-Z0-9_]* This regex denotes an identifier as a succession of characters, the first one always being an alphabetical one. Then, they can be followed by both alphabetical and numerical ones.