Scanner Documentation

Project Requirements:

Based on the specification of the mini-language, implement a scanner that will take as input a text file containing the source program and will produce as output the following:

- PIF (Program Internal Form)
- ST (Symbol Table)

In addition, the program should be able to determine the lexical errors.

Other restrictions:

a) identifiers: length at most 8 characters
b) symbol table: unique for identifiers and constants
c) symbol table organization: hashing table

Used technologies & data structures:

For this laboratory, I've used Java as programming language and Intellij as IDE.

Regarding the data structures, the SymbolTable is represented on a hash table, where each element is a pair consisting of (key, linked-list), where the linked list contains the values hashed to the same key. Also it has a variable m - the number of positions. The hash function is computed as the sum of the ascii code of the identifier/ constant % m. The search function, finds the position in the hash for the given parameter, and then check if it is in the linked list. If it is, then returns the position on which it is placed on the linked list, otherwise -1. The add function, finds the position in the hash for the given parameter, and then checks if it is already there. If it is, returns -1, otherwise returns the position oh which it is placed.

I declared my PIF (Program Internal Form) as a list of lists (with the size equal to 2). Each element of the list is a list of the form [code, ST_pos]. The code for identifiers is 0 and for constants is 1. For separators, operators and reserved words, the code is equal to the corresponding value from the codification table. Regarding the ST_pos, for separators, operators and reserved words the value is -1, while for identifiers and constants it is equal to the position from the Symbol Table.

The codification table is a hash map, in which the key is represented by: identifier, constant and the multitude of reserved words, operators and separators. The value for identifier is 0, for constant is 1, and for the following elements it increases by 1 with each entry, starting from 2.

Project description:

The language instructions are read line by line from a .txt file. Each line is processed using a tokenGenerator function, which returns a list of all the tokens found: operators, separators, string tokens (in quotes) and remaining tokens (reserved words, identifiers and constants).

We go then through the list and check: if the current token is an operator, separator or a reserved word, it is added to PIF, as the pair [code - from the codification table, -1]; if the current token is a valid identifier, it is added to the symbol table, then added to pif as the pair [0, symbol table key]; if the current token is a constant, it is again added to the symbol table in the same manner, then to pif as the pair [1, symbol table key]; otherwise, a message is printed, notifying a lexical error