DOCUMENTATION: FORMAL LANGUAGES AND COMPILER DESIGN
LAB 4

HTTPS://GITHUB.COM/MPELAEZ9/FORMAL-LANGUAGES-AND-COMPILER-DESIGN

I had already implemented all the funciontions that are being asked in this second part in the first part, so I refer again to my Lab3 Documentation:

The project comprises three Java classes along with the Main class, which serves only to initiate the scanner. One of the remaining classes is the Pair class, possessing only two attributes: key and value. This class will be employed to hold the token (key) and its corresponding position in the Symbol Table (value) within the PIF.

MyHashTable, which I made in LAB2, is also being used in this Project. It is already explained in the first Documentation.

The newest part of this incremental Project is the MyScanner class. Its purpose is to read the input file and examinate and catalog each token the file contains. This class contains: a separator list where the separators from the token are being stored, a token list that has stored the tokens contained in the file, regular and special relational lists (introduced by hand), and the PIF and MyHashTable (the ST).

Here are the main methods that it holds explained so we can understand better how the lexical analysis on a source code file goes:

1. *readTokens():* Responsible for reading the tokens and reserved words from the `token.in` file and storing them in the `tokenList`.

2. *readSeparatorsOperators():* Reads the operator and separator tokens from the `token.in` file and stores them in the `separatorList()`.

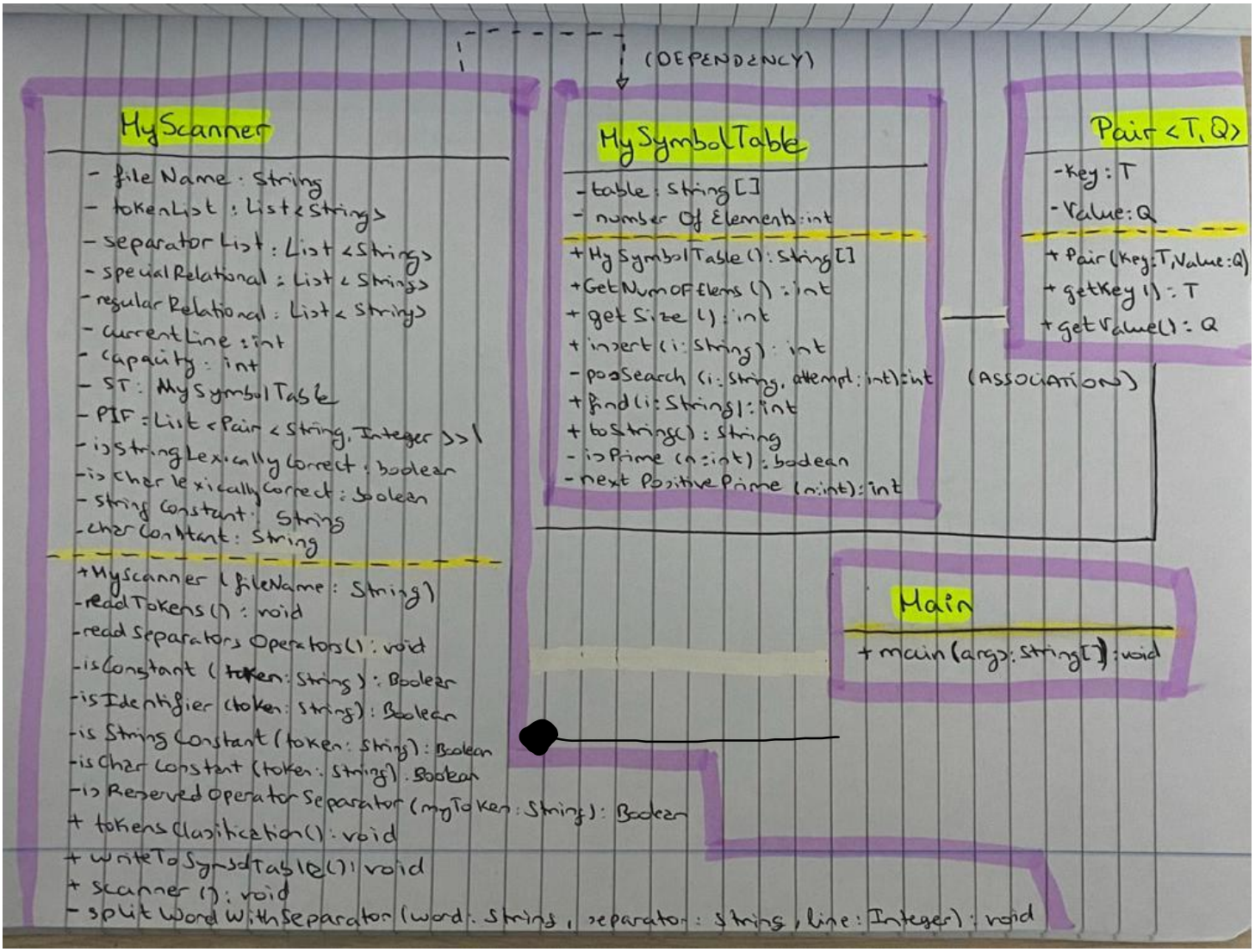
3. *tokensClasification()*: Classifies the tokens into reserved operators, identifiers, and constants. It then inserts the last two into the `SymbolTable`. Additionally, it prints the Program Internal Form (PIF) and checks for any lexical errors.

4. *Scanner():* Reads every word from the input file, stores it, and keeps track of the line it is on.

5. *writeSymbolTable():* Prints the Symbol Table (ST) to the `st.out` file.

6. *splitWordWithSeparator():* Divides words that have a separator. For example, if the word is "a>=b", it will be divided into "a", ">=" and "b".

Class Diagram:



LEYEND:

- \- Private
- \+ Public
- First part of the box → attributes
- Second part of the box → methods
- Parameters → ()
- Return → (): _____
- Dependency ---- >
- Association _____

CONCLUSIONS:

- MyScanner has an association relationship with Pair and a dependecy relationship with MySymbolTable (the Scanner uses the Symbol Table).
- MySymbolTable has an association relationship with Pair because of the instance that it contains.
- Lastly, Main has a composition relationship with MyScanner, because it creates an instance of MyScanner in its main method.