https://github.com/ComanacDragos/ToyLanguageCompiler

```
class Scanner
    //program split by newline
    List<String> programLines;
    //map which encodes each token that can appear in the program
    Map<String, Integer> tokenEncode;
    //tokens of the program -- first column of PIF
    List<String> tokens;
    //the position of each token in the symbol table -- second column in PIF
    List<Integer> tokensPositionInSymbolTable;
    //the line of each token in the program -- third line in PIF
    List<Integer> tokensLines;


    SymbolTable symbolTable = new SymbolTableBSTImpl();


    //patterns corresponding to each constant and ID
    Map<Type, String> patterns;
```

Receives the program and outputs the FIP and SymbolTable to a directory corresponding to the program name

- program and tokens are read from file
- each line is split by the set of simple operators and by the white spaces that are followed by at least 2 quotes
- empty lines are removed
- look ahead is applied to create composed tokens
- the token is processed
- FIP and Symbol table are written to files

public Scanner(String program)

Receives a token and a line

PIF is represented by the 3 lists: tokens, tokensLines, tokensPositionInSymbolTable

Classifies the token and adds it to the PIF otherwise it throws a LexicalError at the given line

- if the token is an operator separator or reserved word it is added to the PIF with the given line and the position -1
- if it is an id or a constant it is added to the PIF with the corresponding type (id or constant) and to the Symbol table according to the pattern that the token matches
- otherwise a lexical error is thrown

private void processToken(String token, Integer line)

```
//read the lines from a file
    public List<String> readFile(String file)

//write to a file the content
    public void writeToFile(String file, String content)

Types corresponding to the types of values in the symbol table
public enum Type

    Type factory that generates the corresponding Value class given a token and a type
public class TypeFactory
```