https://github.com/ComanacDragos/ToyLanguageCompiler

Statement: Implement a scanner (lexical analyzer): Implement the scanning algorithm and use ST from lab 2 for the symbol table.

Input: Programs p1/p2/p3/p1err and token.in (see Lab 1a)

Output: PIF.out, ST.out, message "lexically correct" or "lexical error + location"

Deliverables: input, output, source code, documentation

Details:

- ST.out should give information about the data structure used in representation
- If there exists an error the program should give a description and the location (line and token)

```
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)

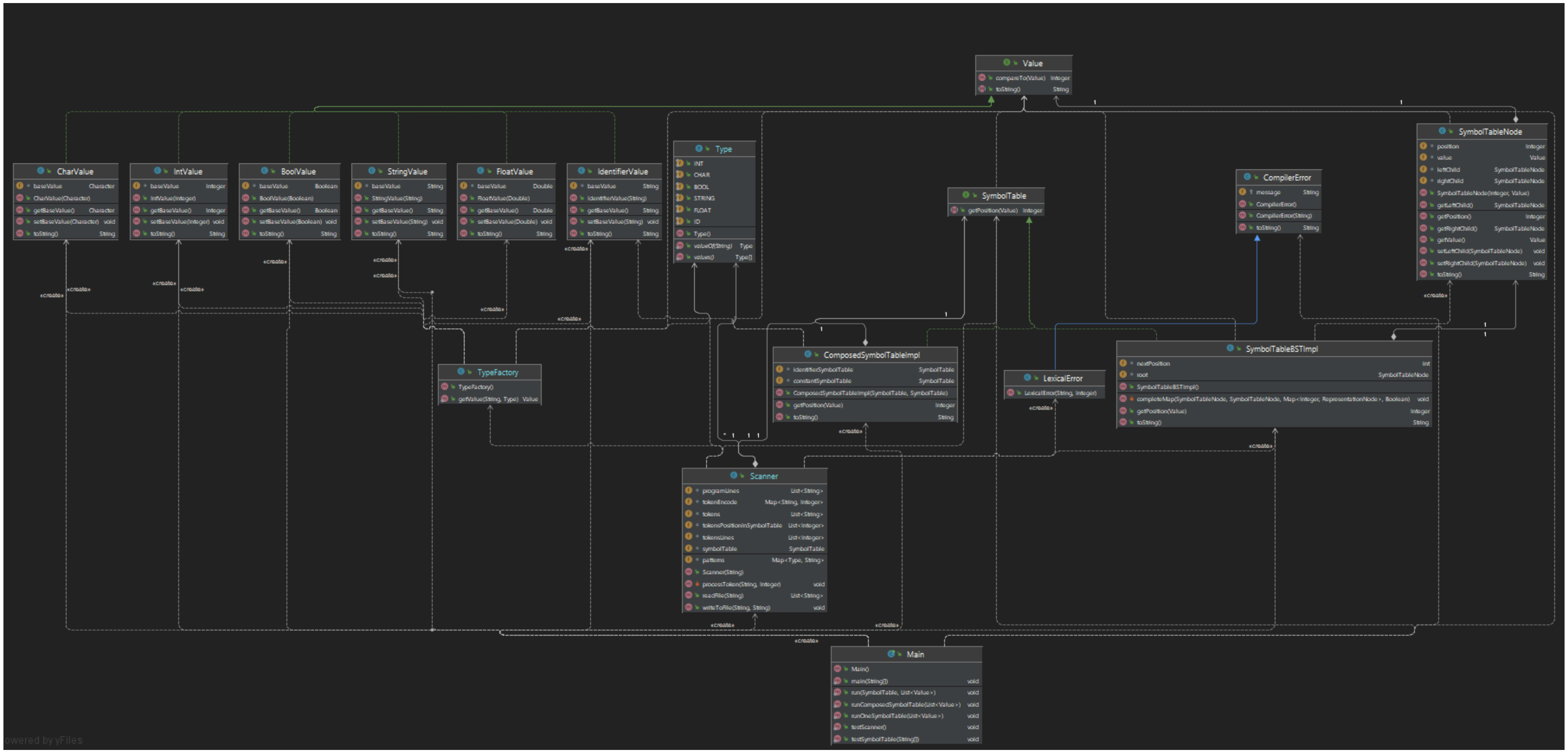
```
//read the lines from a file
    public List<String> readFile(String file)


//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
```


Value
compareTo(Value) Integer
toString() String
CharValue
baseValue Character
CharValue(Character)
getBaseValue() Character
setBaseValue(Character) void
toString() String
IntValue
baseValue Integer
IntValue(Integer)
getBaseValue() Integer
setBaseValue(Integer) void
toString() String
BoolValue
baseValue Boolean
BoolValue(Boolean)
getBaseValue() Boolean
setBaseValue(Boolean) void
toString() String
StringValue
baseValue String
StringValue(String)
getBaseValue() String
setBaseValue(String) void
toString() String
FloatValue
baseValue Double
FloatValue(Double)
getBaseValue() Double
setBaseValue(Double) void
toString() String
IdentifierValue
baseValue String
IdentifierValue(String)
getBaseValue() String
setBaseValue(String) void
toString() String
Type
INT
CHAR
BOOL
STRING
FLOAT
ID
Type()
valueOf(String) Type
values() Type[]
SymbolTable
getPosition(Value) Integer
CompilerError
message String
CompilerError()
CompilerError(String)
toString() String
SymbolTableNode
position Integer
value Value
leftChild SymbolTableNode
rightChild SymbolTableNode
SymbolTableNode(Integer, Value)
getLeftChild() SymbolTableNode
getPosition() Integer
getRightChild() SymbolTableNode
getValue() Value
setLeftChild(SymbolTableNode) void
setRightChild(SymbolTableNode) void
toString() String
TypeFactory
TypeFactory()
getValue(String, Type) Value
ComposedSymbolTableImpl
identifierSymbolTable SymbolTable
constantSymbolTable SymbolTable
ComposedSymbolTableImpl(SymbolTable, SymbolTable)
getPosition(Value) Integer
toString() String
LexicalError
LexicalError(String, Integer)
SymbolTableBSTImpl
nextPosition int
root SymbolTableNode
SymbolTableBSTImpl()
completeMap(SymbolTableNode, SymbolTableNode, Map<Integer, RepresentationNode>, Boolean) void
getPosition(Value) Integer
toString() String
Scanner
programLines List<String>
tokenEncode Map<String, Integer>
tokens List<String>
tokensPositionInSymbolTable List<Integer>
tokensLines List<Integer>
symbolTable SymbolTable
patterns Map<Type, String>
Scanner(String)
processToken(String, Integer) void
readFile(String) List<String>
writeToFile(String, String) void
Main
Main()
main(String[]) void
run(SymbolTable, List<Value>) void
runComposedSymbolTable(List<Value>) void
runOneSymbolTable(List<Value>) void
testScanner() void
testSymbolTable(String[]) void
«create»
1
*
owered by yFiles

Testing

Input:

```
int a=9;
int     b=6;
if(a>b){
    >>"a is the maximum";
}else{
    >>bbbb+"b is the maximum" ;
}


 >>0.0

    >>1.3
    <<=0.1
    >>+0.001
    >>-3
    >>-11111.1
    !=
    <=
    >=
    ==
    ^
    !a%bbb
-'a'
!"aa aa"
```

Output:

FIP:

```
token,position,line
int,-1,1
id,0,1
=,-1,1
constant,1,1
;,-1,1
int,-1,2
id,2,2
=,-1,2
constant,3,2
;,-1,2
if,-1,3
(,-1,3
id,0,3
>,-1,3
id,2,3
),-1,3
{,-1,3
>>,-1,4
constant,4,4
;,-1,4
},-1,5
else,-1,5
{,-1,5
>>,-1,6
id,5,6
+,-1,6
constant,6,6
;,-1,6
},-1,7
>>,-1,10
constant,7,10
>>,-1,12
constant,8,12
<<,-1,13
=,-1,13
constant,9,13
>>,-1,14
constant,10,14
>>,-1,15
constant,11,15
>>,-1,16
constant,12,16
!=,-1,17
<=,-1,18
>=,-1,19
==,-1,20
^,-1,21
!,-1,22
id,0,22
%,-1,22
id,13,22
-,-1,23
constant,14,23
!,-1,24
constant,15,24
```

ST:

```
position,value,parent,sibling
0,a,-1,-1
1,9,0,2
2,b,0,1
3,6,1,-1
4,"""a is the maximum""",3,-1
5,bbbb,2,-1
6,"""b is the maximum""",4,-1
7,0.0,6,15
8,1.3,7,11
9,0.1,8,-1
10,0.001,9,-1
11,-3,7,8
12,-11111.1,11,-1
13,bbb,5,-1
14,"'a'",12,-1
15,"""aa aa""",6,7
```

Error program:

```
a=9;
a+012
b='aa';
if (a>b){
   >>"a is the maximum
}else{
   >>"b is the maximum"
}
```

```
Output: Lexical error at line: 2 for token: '012'
```