Parser

Github link:

https://github.com/CimpeanAndreea/FLCD/tree/master/Lab5

Problem statement

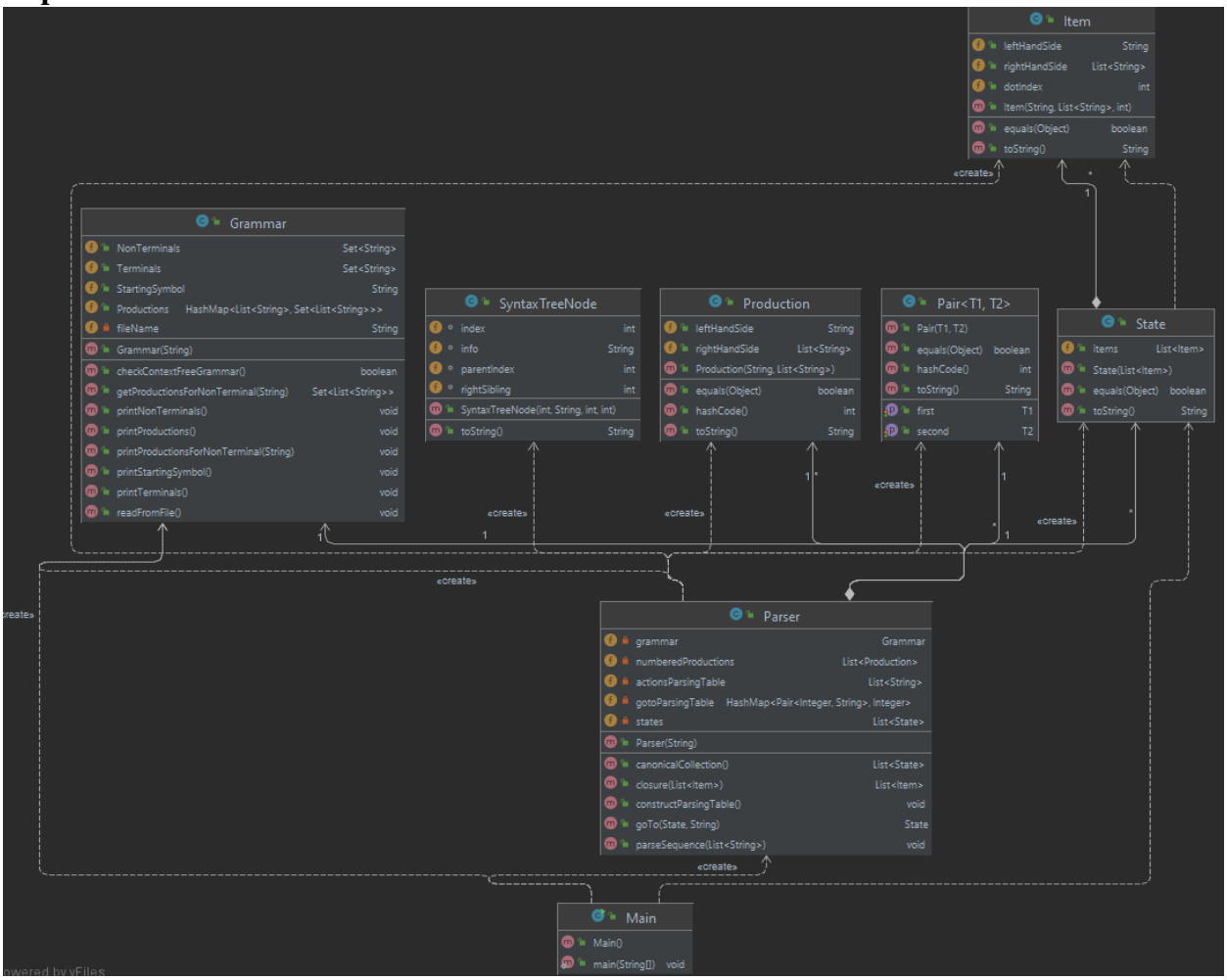
PART 3: Deliverables

1. Algorithms corresponding to *parsing table* (if needed) and *parsing strategy*

2. Class *ParserOutput* - DS and operations corresponding to choice 2.a/2.b/2.c (Lab 5) (required operations: transform parsing tree into representation; print DS to screen and to file)

Remark: If the table contains conflicts, you will be helped to solve them. It is important to print a message containing row (symbol in LL(1), respectively state in LR(0)) and column (symbol) where the conflict appears. For LL(1), values (α,i) might also help.

Implementation



Continuing the previous work 2 more functions were added to the Parser class and one more class **SyntaxTreeNode.** In order to parse a given sequence, after building the canonical collection of the LR(0) parser, there needs to be build a parsing table, with 2 parts: ACTION and GOTO, having one row for each obtained state, one column for the action part and one column for each symbol from the terminals and nonterminals of the grammar in the goto part.

function constructParsingTable()

description: construct the parsing table following the next steps:

1. If there is an item in the state containing dot somewhere besides the last position, then action(state) = shift
2. If there is an item [A->Beta.] in the state (that has dot at the end) then action(state) = reduce P, where P is the number of the production A->Beta from the grammar
3. If [S`->S.] belongs to the state then action(state)=accept
4. if goto(si, X) = sj then goto(si, X)=sj
   Conflicts:

If there are 2 items in a state that have the property from step 2, there is a reduce-reduce conflict, if there is an item with the property from 1 and another with a property from 2, there is a shift-reduce conflict. If there are conflicts print a message.

post: completes the 2 lists from the parser