Link to github repository : <https://github.com/paie2993/FormalLanguagesParser> (Andrei Șaramet + Adrian Popescu from 936/1)

Link to reviewed repository: <https://github.com/rad0o/FLCD/tree/main/lab5> (Radu Savin from 936/1)

Since the entries in the parse table have 2 different forms (i.e.

- for pairs of the form (nonterminal, terminal) the entry is a pair of type (right side of production, index of production)

- for pairs of the form (terminal, terminal), the entry is just an action (pop or accept))

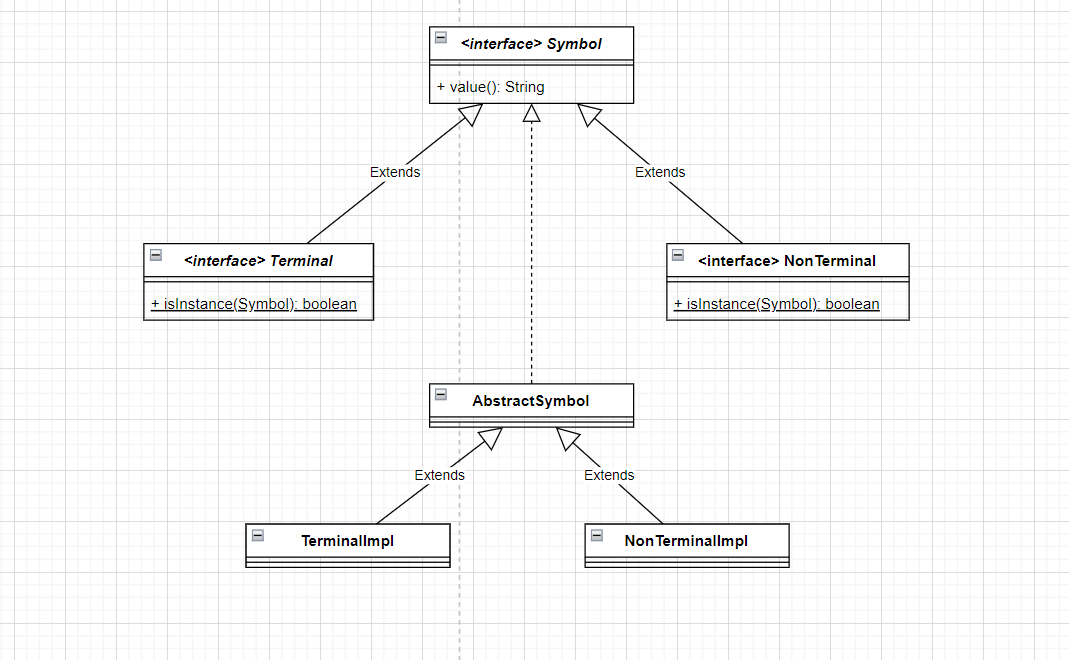
the table is implemented using two hash maps, with keys the above-listed pairs and entries of type NextMove (which stores pairs of the form (right side of production, index of production)) and Action (an enumeration with values pop and accept), respectively.

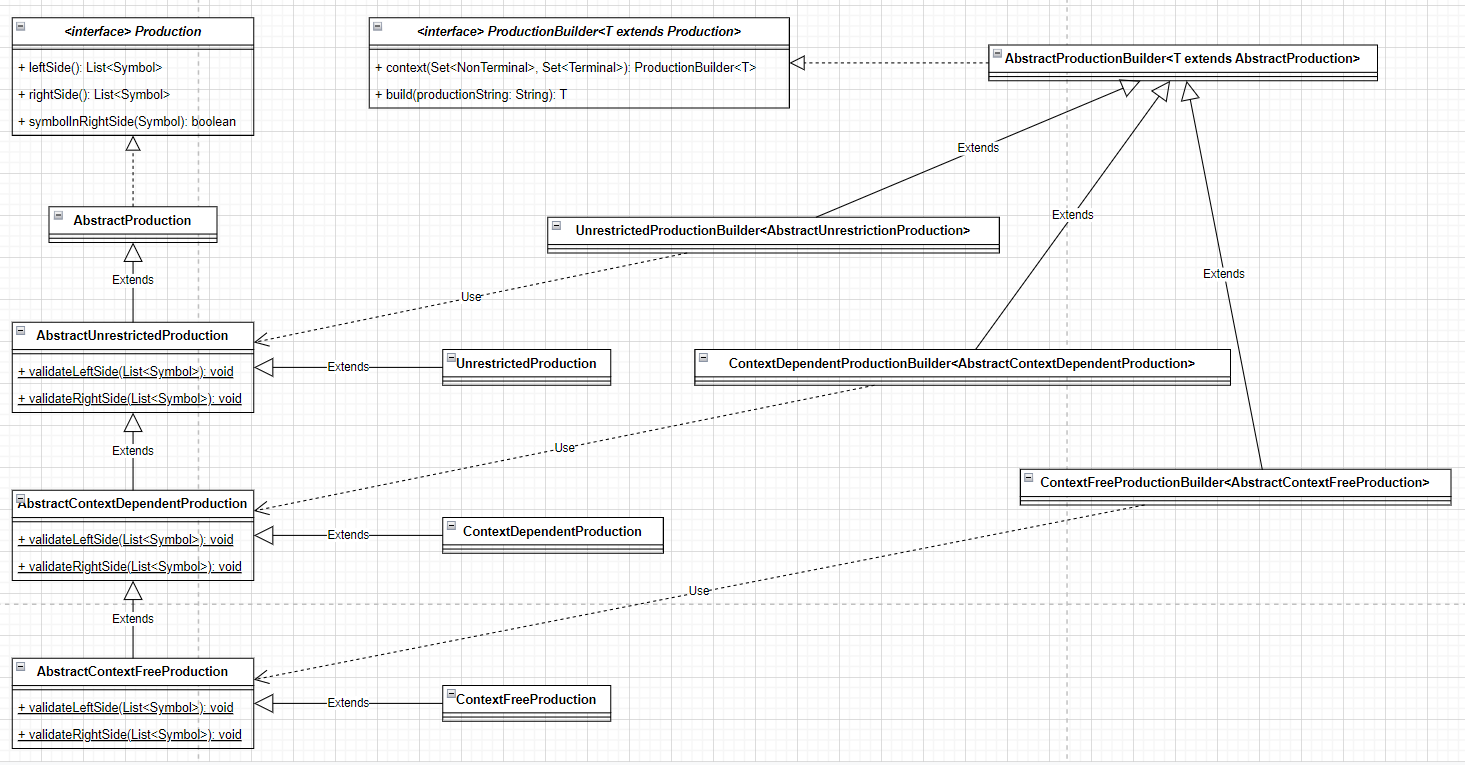
As the parser accepts only lists of strings (which can store symbols of a word or tokens of a program internal form), there have been defined two readers, i.e., one for reading a word from a file and another for reading a program internal form, specifically the first column of the PIF, the list of tokens.

For the ParserOutput class, the chosen implementation was the father-sibling table. The table is implemented as a list with entries of type TableEntry (which models triples of the form (symbol, index of parent, index of left sibling). A list data structure is used as each entry stores two indexes for this list.

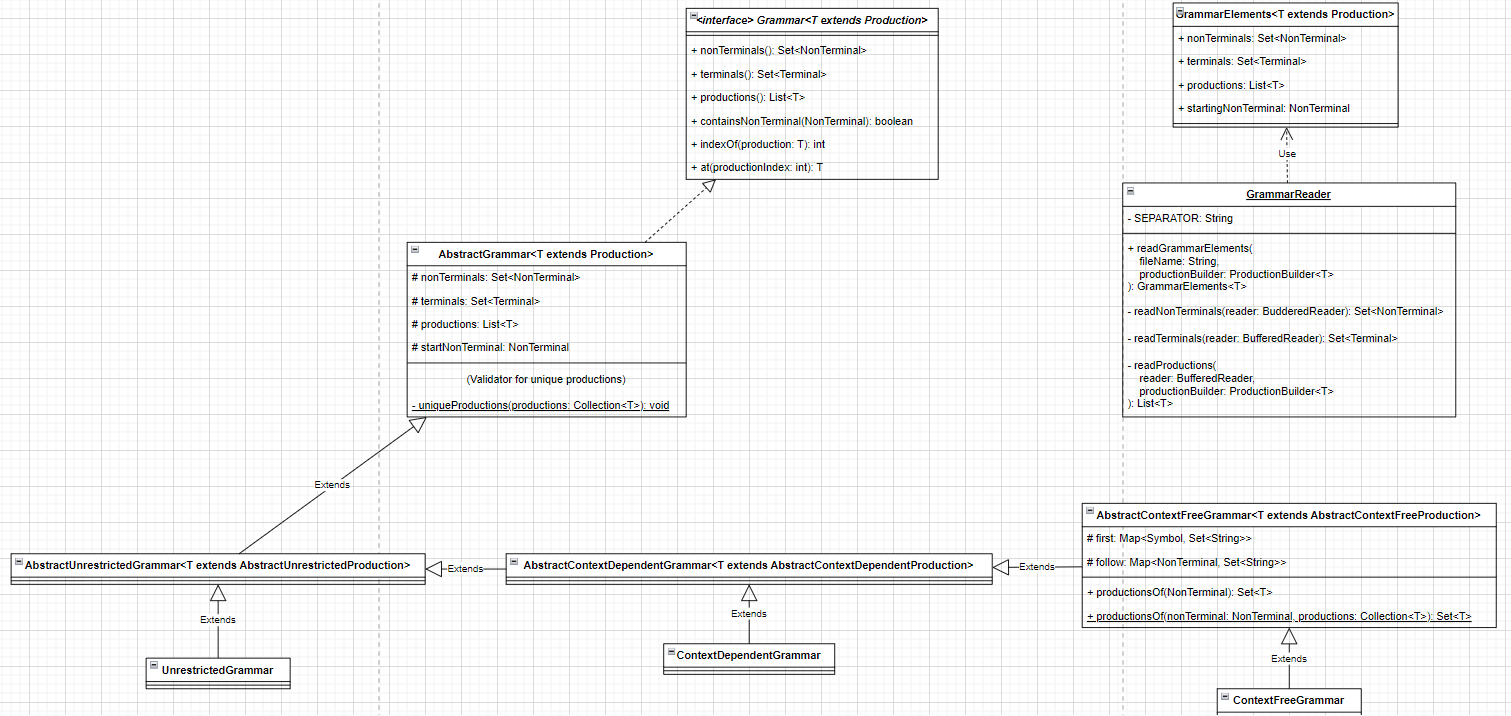
Furthermore, two other implementations are provided for the ParserOutput. The first one, namely production string, stores a list of productions, while the second, called derivation string, stores a list of sentential forms (modelled as lists of symbols).

Symbol hierarchy:

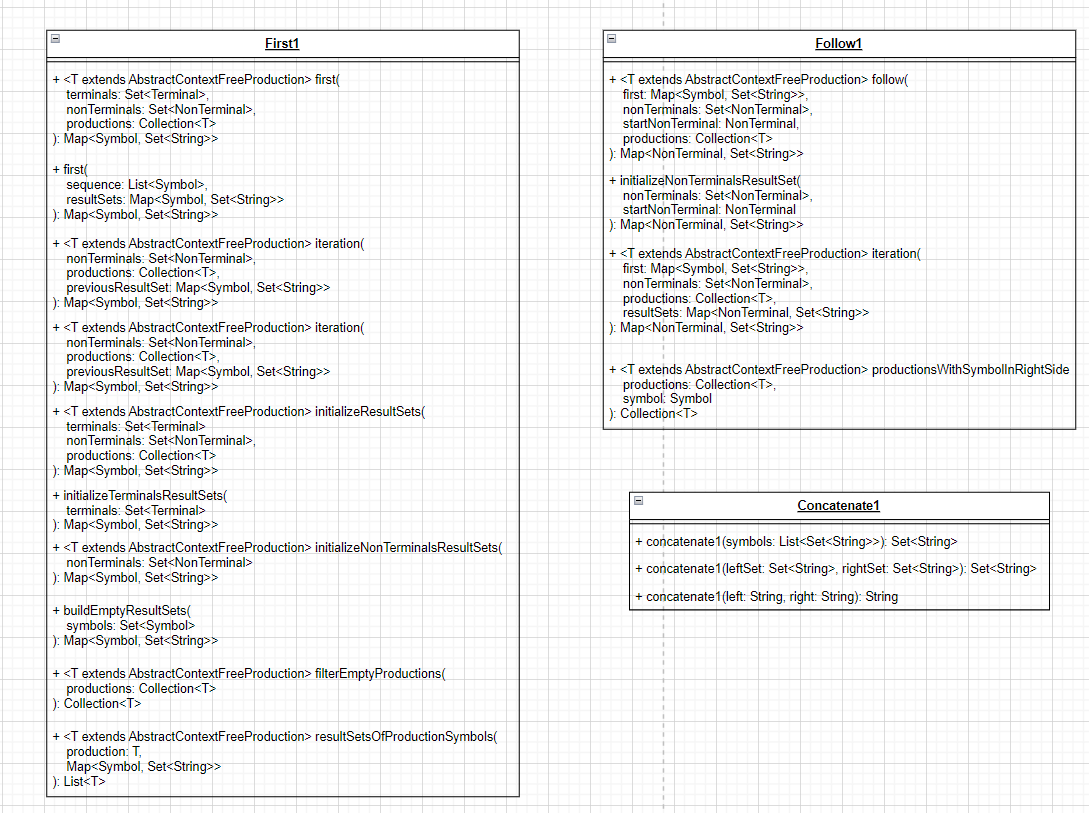


Production hierarchy: 

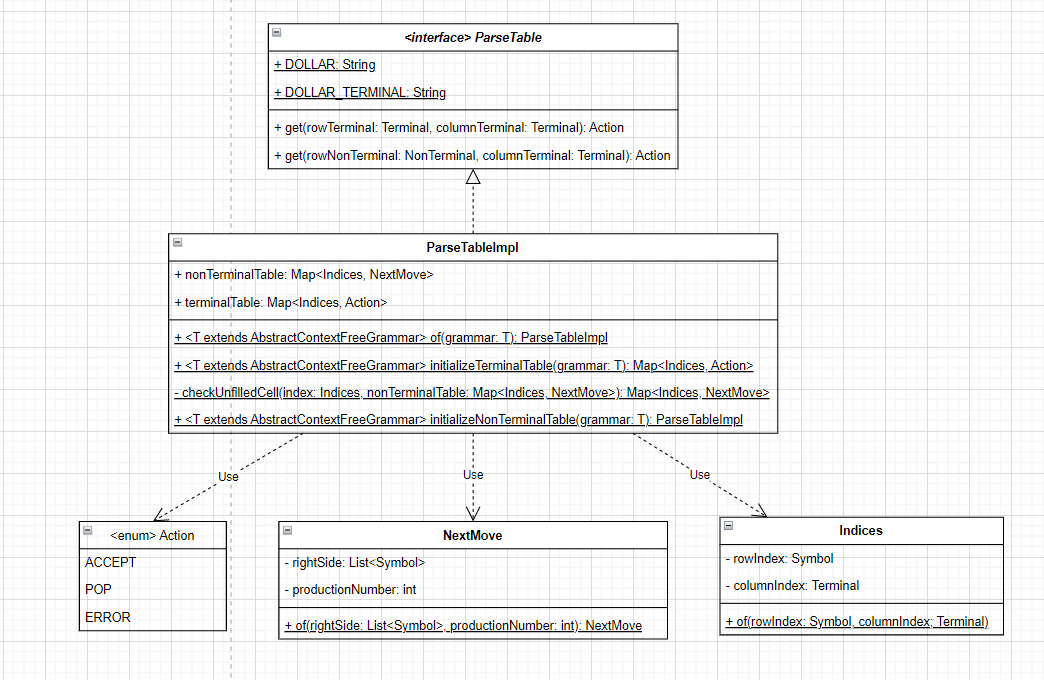
Grammar hierarchy:



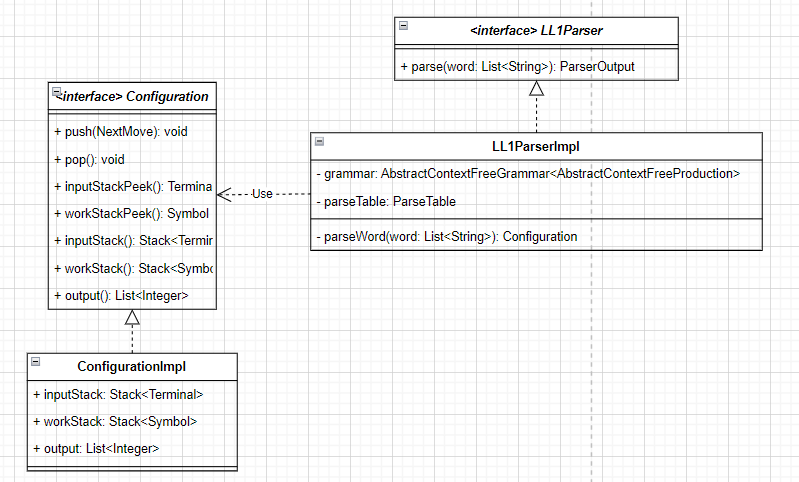
First, follow, concatentate:



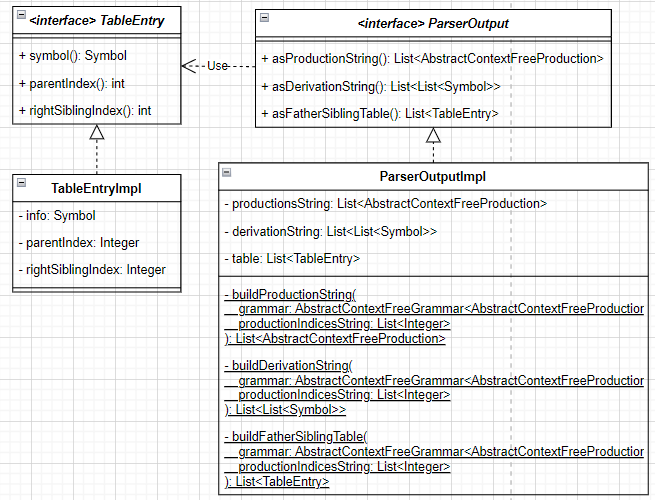
Pare Table:



LLI1 Parser:



Parser output:



File readers and writers:

