Github Link: https://github.com/916RasnitaRadu/SEM5-FLCD/tree/main/Lab6

Documentation

Grammar

The Grammar class implements the grammar for a formal language. It has a field for each (N, E, P, S) set of the grammar and those are: *non-terminals*, *terminals*, *productions* and *the starting symbol*. The starting symbol is kept as a map between, where the key is the left-hand side of the production and the value is the right-hand side of the production represented as a list of strings.

Most of the methods are for parsing, such as:

- parseLine(String line) – parse a line of the file and return a list of words
- fromFile(String fileName) – read the input grammar file, creates the (N, E, P, S) tuple and verifies if the grammar is context free by calling the validate() function
- parseRules(List<String> rules) – takes the set of the productions and creates the map from the class

There are also other methods:

- validate(List<String> N, List<String> E, Map<String, List<String>> P, String S) – checks if the grammar is context free
- isNonTerminal(String value)
- isTerminal(String value)
- getProductionsFor(String nonterminal) – returns the list of productions for a non-terminal

Parser

The Parser class is implementing the LL(1) algorithm, with the FIRST and FOLLOW algorithms. The class is composed of: one object of type Grammar, the FIRST set of terminals and the FOLLOW set of terminals.

Operations:

- generateFirst( ) – method that generates a set for each non-terminal that contains all terminals from which we can start a sequence (from the given non-terminal)
- generateFollow( ) – method that builds for each non-terminal which contains the “first of what’s after”, namely all the non-terminals into which we can go from the given non-terminal
- innerLoop(Set<String> initialSet, List<String> items, Set<String> additionalSet) – method for computing the FIRST set of a sequence of symbols within a production rule.