Lab 5

Link Github: https://github.com/913-groza-vlad/Formal-Languages-and-Compiler-Design/tree/main/Lab5

The Grammar class represents a grammar and provides methods to initialize the grammar from a file, parse its content and check whether it is a context-free grammar.

The Grammar class has the following attributes:
- nonTerminals, which is a set containing the non-terminal symbols of the grammar
- terminals, which is a set containing the terminal symbols of the grammar
- productions, which is a map representing production rules, where each key is a non-terminal symbol, and the corresponding value is a set of production rules
- startSymbol, the start symbol of the grammar
- filename, the name of the file containing the grammar rules

The class contains methods for representing each of its fields in string format and the also the whole grammar (so we can display the grammar and its content) and the methods:

private Set<String> parseLine(String line):
This method parses a line from the grammar file to extract a set of elements enclosed in curly braces.

public boolean checkCFG():
Checks whether the grammar adheres to the rules of a context-free grammar. In order to return true, the following conditions must be met: the start symbol of the grammar must appear on the left-hand side of at least one production rule, all left-hand side symbols of production rules must be non-terminals and all symbols in the right-hand side must be either non-terminals, terminals, or the symbol epsilon.

public void readGrammarFromFile():
It reads the grammar rules from a specified file, initializes the grammar attributes (non-terminals, terminals, start symbol and productions) and populates the

Grammar object with the parsed information. The method reads the first two lines from the file, extracting non-terminals and terminals from lines enclosed in curly braces. Then, on the third line there is the start symbol of the grammar. Then, the productions are read by: skipping the line containing P = { header, then iterate through the remaining lines, parsing production rules and adding them to the set of productions for each non-terminal.