**FLCD Lab. 5**

**- documentation -**

**GitHub Repository**:

<https://github.com/adrianPascan/FLCD_Lab2>

**Grammar**

1. **input file format**

For a grammar G=(N,Σ,P,S),

NONTERMINAL ϵ N

TERMINAL ϵ Σ

SYMBOL ϵ N U Σ

file := nonterminals “\n” terminals “\n” startSymbol “\n” productions

nonterminals := NONTERMINAL | NONTERMINAL “ ” nonterminals

terminals := TERMINAL | TERMINAL “ ”terminals

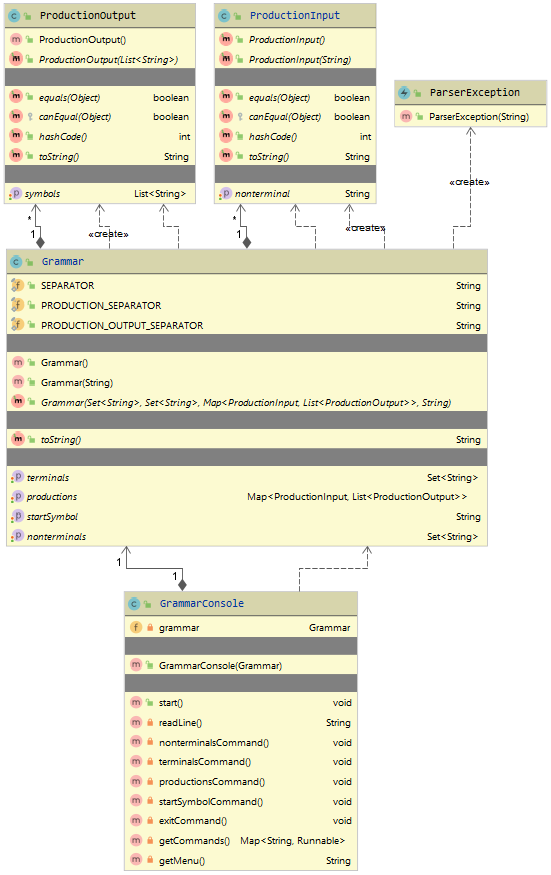
startSymbol := NONTERMINAL

productions := production | production “\n” productions

production := NONTERMINAL “:” symbols

symbols := NONTERMINAL “ “ symbols | TERMINAL “ “ symbols

1. **diagram**

****

1. **classes**
2. **ParserInput**
   * auxiliary class for Grammar class
   * represents the input of the transition function
   * fields
     + nonterminal: string
3. **ParserOutput**
   * auxiliary class for Grammar class
   * represents the output of the transition function
   * fields
     + symbols: list of string
4. **Grammar**
   * class implementing a grammar
   * fields
     + nonterminals: set of string
     + terminals: set of string
     + productions: map from ParserInput instance to list of ParserOutput instance
     + startSymbol: string
   * constructors
     + Grammar()
       - Returns a new instance of Grammar class, ‘g’, with no parameters.
       - preconditions: true
       - postconditions: ‘g’ has no parameters
       - return: ‘g’
     + Grammar (filePath)
       - Returns a new instance of Grammar class, ‘g’, with parameters read from a file.
       - preconditions: ‘filePath’, string – a valid file path
       - postconditions: ‘productions’ and ‘startSymbol’ of ‘g’ are valid (i.e. ‘startSymbol’ ϵ ‘nonterminals’, input of ‘productions’ ϵ ‘nonterminals’ and elements of outputs of ‘productions’ ϵ ‘nonterminals’ U ‘terminals’)
       - return: ‘g’
5. **GrammarException**
   * exception class
   * extends RuntimeException
6. **GrammarConsole**
   * class implementing a console for Grammar class
   * fields
     + grammar: Grammar instance
   * methods and functions
     + start()
       - Runs the GrammarConsole instance, ‘gc’.
       - preconditions: true
       - postconditions: ‘gc’ is running
     + readLine()
       - Reads the next line ‘l’ from the console and returns it.
       - preconditions: true
       - postconditions: ‘l’ is the next line read
       - return: ‘l'
     + getCommands()
       - Returns a map ‘m’ from a string to function references of GrammarConsole instance ‘gc’.
       - preconditions: true
       - postconditions: ‘m’ maps to function references of ‘gc’
       - return: ‘m'
     + getMenu()
       - Returns a string ‘s’ representing the codes associated with each command description of GrammarConsole instance ‘gc’
       - preconditions: true
       - postconditions: ‘s’ is a string representation of the menu
       - return: ‘s’
     + nonterminalsCommand()
       - Writes the nonterminals ‘nonterminals’ of ‘grammar’ to the console.
       - preconditions: true
       - postconditions: ‘nonterminals’ were written to the console
     + terminalsCommand()
       - Writes the terminals ‘terminals’ of ‘grammar’ to the console.
       - preconditions: true
       - postconditions: ‘terminals’ was written to the console
     + productionsCommand()
       - Reads a nonterminal ‘nt’ from the console and writes the productions ‘prods’ from ‘nt’ of ‘grammar’ to the console.
       - preconditions: true
       - postconditions: ‘prods’ were written to the console
     + startSymbolCommand()
       - Writes the start symbol ‘ss’ of ‘g’ to the console.
       - preconditions: true
       - postconditions: ‘ss’ was written to the console
     + exitCommand()
       - Writes exiting messages ‘em’ to the console.
       - preconditions: true
       - postconditions: ‘em’ were written to the console