Descendent recursive parser documentation

- Data structures:

The grammar is a class having the following attributes:

terminals, non-terminals - represented by a list of strings
productions - represented by a tuple of 2 elements, the non-terminal on the left side, and the elements on the right, included in the production
check_CFG - bool value (true if the grammar is CFG, false otherwise)

The class has the following methods:

read_from_file(f) - reads a grammar from an input file f and also checks if it is CFG or not, updating the attribute check_CFG upon decision
get_non_terminals - returns the non terminals
get_terminals - returns the terminals
get_productions - returns the productions
is_non_terminal(el) - checks if el is in the non terminals list
is_terminal(el) - checks if el is in the terminals list
get_productions_for_a_given_non_terminal(el) - returns the productions for a given non_terminal el
get_is_CFG - returns the attribute check_CFG

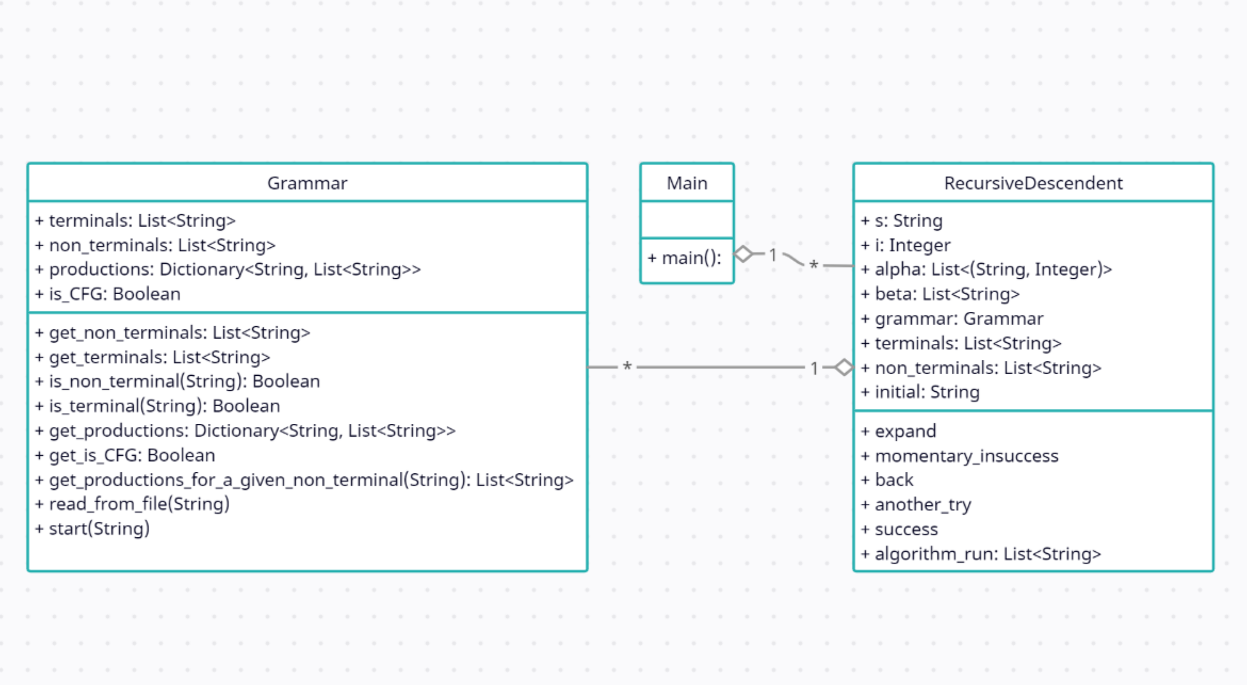
The recursive descendant is a class having the following attributes:

s - state of the parsing which can be:
“q” = normal state
”f” = final state, corresponding to success
”b” = back state
”e” = error state, corresponding to insuccess
i - position of current symbol in input sequence
alpha = working stack, stores the way the parse is built
beta = input stack, part of the tree to be built
grammar - also a class, explained above
File = the file to be read from

The class has the following methods:

expand, advance, momentary insuccess, back, another try, success - corresponding to moves used in the algorithm
algorithm_run - the recursive descent parsing

- The class diagram:


Grammar
+ terminals: List<String>
+ non_terminals: List<String>
+ productions: Dictionary<String, List<String>>
+ is_CFG: Boolean
+ get_non_terminals: List<String>
+ get_terminals: List<String>
+ is_non_terminal(String): Boolean
+ is_terminal(String): Boolean
+ get_productions: Dictionary<String, List<String>>
+ get_is_CFG: Boolean
+ get_productions_for_a_given_non_terminal(String): List<String>
+ read_from_file(String)
+ start(String)
Main
+ main():
1
*
*
1
RecursiveDescendent
+ s: String
+ i: Integer
+ alpha: List<(String, Integer)>
+ beta: List<String>
+ grammar: Grammar
+ terminals: List<String>
+ non_terminals: List<String>
+ initial: String
+ expand
+ momentary_insuccess
+ back
+ another_try
+ success
+ algorithm_run: List<String>