grammar.py

class Production:

`def __init__(self, left_term, right_term):` - class constructor

`def get_left_term(self):` - getter for left term

`def get_right_term(self):` - getter for right term

`def set_left_term(self, new_left):` - setter for left term

`def set_right_term(self, new_right)` - setter for right term

`def __str__(self)` - toString method

`def __eq__(self, other):` - equals method

Class Grammar:

start_symbol – the start symbol for the grammar

terminals – set of terminals

non-terminals – set of non-terminals

productions – list of productions for the grammar

`def __init__(self, start_symbol, terminals, non_terminals, productions)`
- constructor

- getter methods and setter methods

`def is_terminal(self, symbol)` - checks if a symbol is terminal

`def is_non_terminal(self, symbol)` - checks if symbol is non-terminal

- toString method;

grammar_utils.py

`def read_grammar_from_file(in_file)` - reads the grammar rules from a specified path stored in the variable “in_file”

config.py

class Config:

state – holds the state of the program during parsing, it is represented through an enum in the state_type.py file

index – is the parsing index

work_stack – the stack to be built during parsing

input_stack – holds the input stack for the parsing and is initialised with the start symbol

toString method

recursive_descent_alg.py

`def get_next_prod(prod, prods)` - generates the next production for parsing

`def recursive_descent(grammar, sequence)` - parses the productions according to the recursive decent paradigm, building a tree with terminal and non-terminal values while checking the input grammar for correctness.