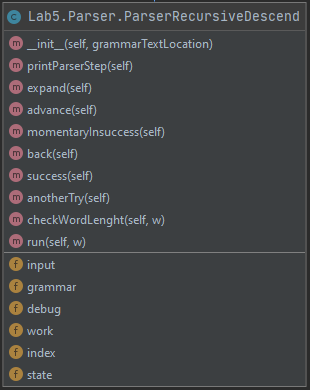
Lab 6 – Part 2

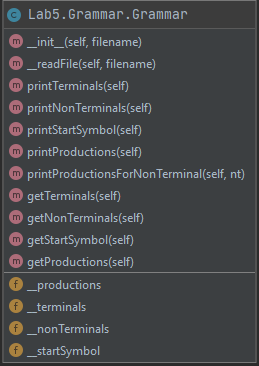
# Recursive descendent parser – implemented in team with Andrei Onița

Github link: <https://github.com/the-coding-cloud/FLCD/tree/main/Lab5>

* Finished implementing another try until the beginning of the lab
* Implemented the parsing strategy (some minor issues have to be solved)

# UML Diagram





# Grammar

## Data Structure

We created the class grammar which stores the grammar from a text file

Constructor(file)

* Pre:
* In: file: string - location of Grammar definition
* Out: a Grammar class
* Post:

Creates a Grammar that initialises with the terminals, nonTerminals, productions, startSymbol, File.

The terminals, nonTerminals, productions are lists.

## Operations:

Function readFile(filename)

* Pre: an initialised fileName path
* In: - fileName : String
* Out: -
* Post:
* Desc: reads the Grammar and loads it in the class lists,

# Parser Recursive Descen**t**

## Data Structure

We created the class ParserRecursiveDescent

Constructor(file)

* Pre:
* In: file: string - location of Grammar definition
* Out: a Parser class
* Post:

Creates a Parser that initialises with the work and input stacks, index, state

## Operations:

Function expand(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: expands the non terminal into it s first production of terminals

Function advance(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: puts one terminal in the work stack

Function momentaryInsuccess(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: state is change to b

Function back(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: goes back one index

Function success(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: state set to s

Function anotherTry(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: parses the last non terminal to the next set of terminals in its production, or pops the non terminal from work stack to input stack

Function expand(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: expands the non terminal into it s first production of terminals

Function advance(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: puts one terminal in the work stack

Function momentaryInsuccess(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: state is change to b

Function back(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: goes back one index

Function success(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: state set to s

Function anotherTry(filename)

* Pre:
* In:
* Out: -
* Post:
* Desc: parses the last non terminal to the next set of terminals in its production, or pops the non terminal from work stack to input stack