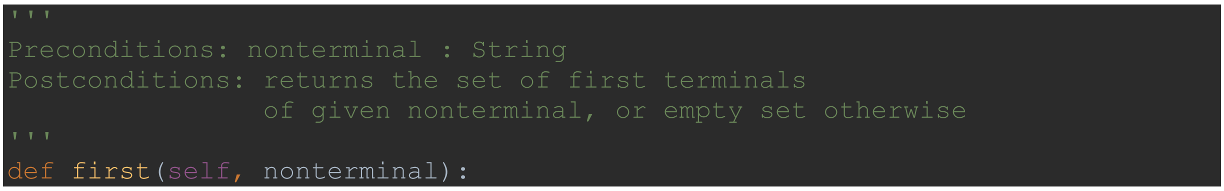
Lab 6 & 7 – FLCD

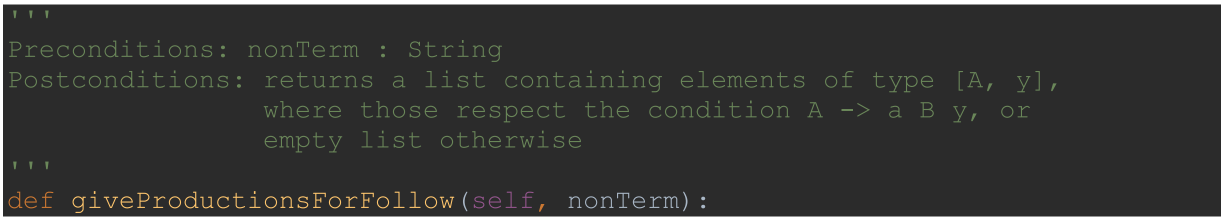
Team Mirza Lorena, Moldovan Vasilica

LL(1) Parser

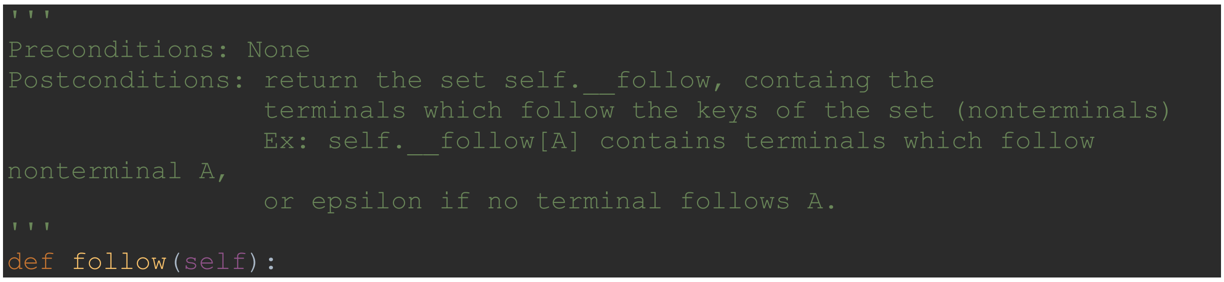
Github link: https://github.com/mirzalorena/flcd_lab5



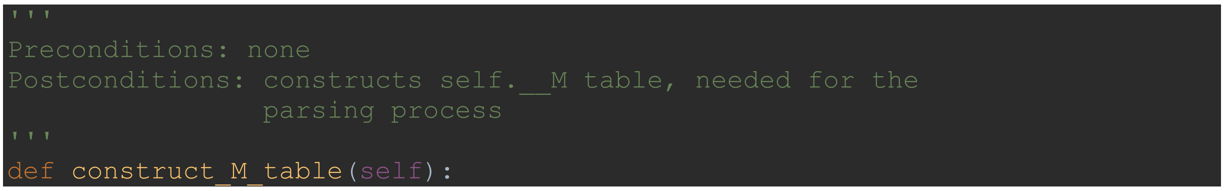
```
'''
Preconditions: nonterminal : String
Postconditions: returns the set of first terminals
                of given nonterminal, or empty set otherwise
'''
def first(self, nonterminal):
```



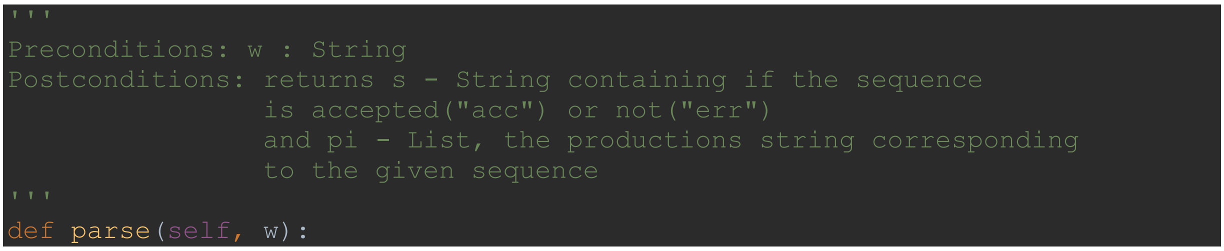
```
'''
Preconditions: nonTerm : String
Postconditions: returns a list containing elements of type [A, y],
                where those respect the condition A -> a B y, or
                empty list otherwise
'''
def giveProductionsForFollow(self, nonTerm):
```



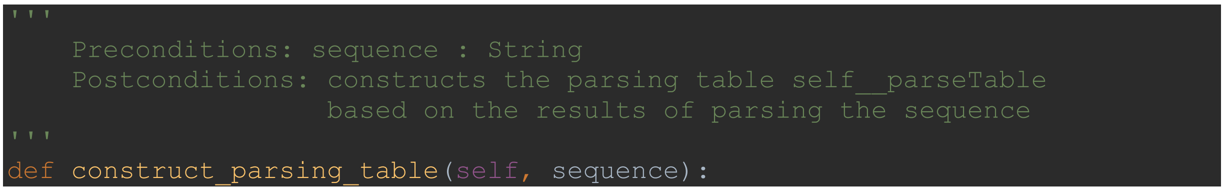
```
'''
Preconditions: None
Postconditions: return the set self.__follow, containg the
                terminals which follow the keys of the set (nonterminals)
                Ex: self.__follow[A] contains terminals which follow
nonterminal A,
                or epsilon if no terminal follows A.
'''
def follow(self):
```



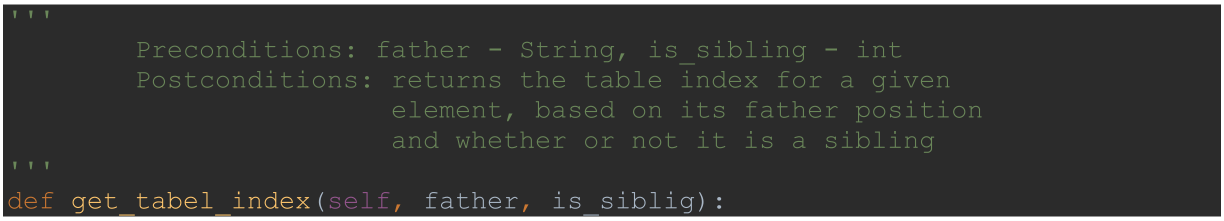
```
'''
Preconditions: none
Postconditions: constructs self.__M table, needed for the
                parsing process
'''
def construct_M_table(self):
```



```
'''
Preconditions: w : String
Postconditions: returns s - String containing if the sequence
                is accepted("acc") or not("err")
                and pi - List, the productions string corresponding
                to the given sequence
'''
def parse(self, w):
```

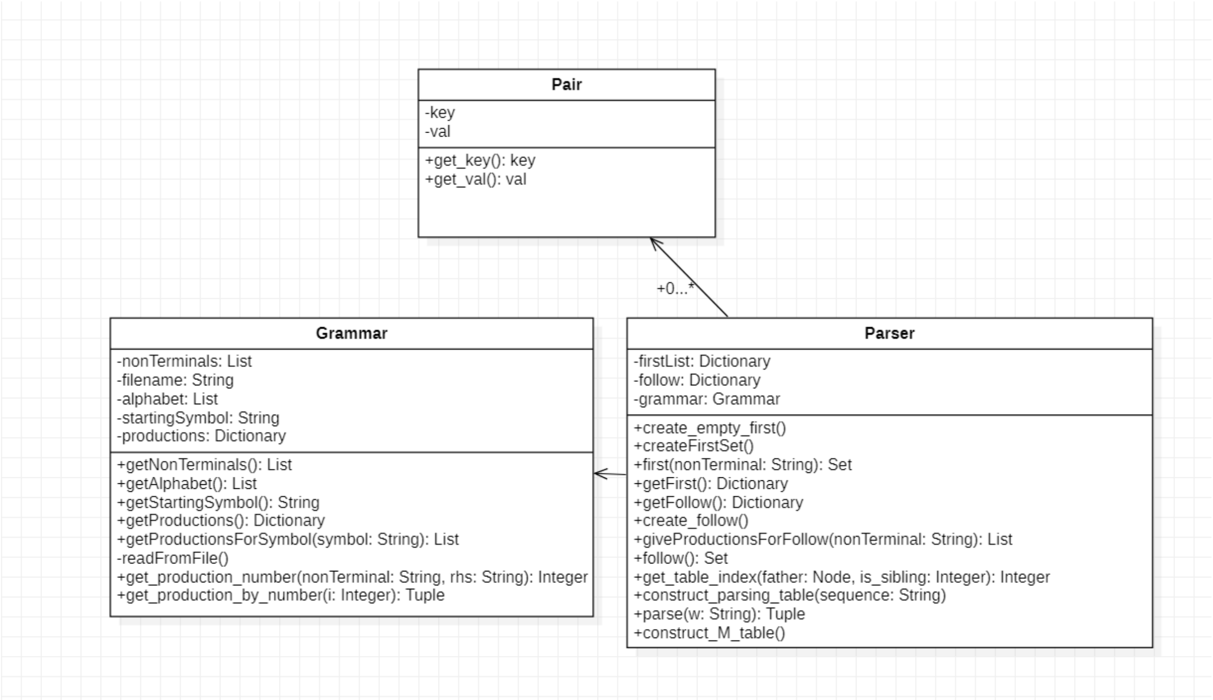


```
'''
    Preconditions: sequence : String
    Postconditions: constructs the parsing table self__parseTable
                    based on the results of parsing the sequence
'''
def construct_parsing_table(self, sequence):
```



```
'''
        Preconditions: father - String, is_sibling - int
        Postconditions: returns the table index for a given
                        element, based on its father position
                        and whether or not it is a sibling
'''
def get_tabel_index(self, father, is_siblig):
```

UML Diagram:



Pair
-key
-val
+get_key(): key
+get_val(): val
+0...*
Grammar
-nonTerminals: List
-filename: String
-alphabet: List
-startingSymbol: String
-productions: Dictionary
+getNonTerminals(): List
+getAlphabet(): List
+getStartingSymbol(): String
+getProductions(): Dictionary
+getProductionsForSymbol(symbol: String): List
-readFromFile()
+get_production_number(nonTerminal: String, rhs: String): Integer
+get_production_by_number(i: Integer): Tuple
Parser
-firstList: Dictionary
-follow: Dictionary
-grammar: Grammar
+create_empty_first()
+createFirstSet()
+first(nonTerminal: String): Set
+getFirst(): Dictionary
+getFollow(): Dictionary
+create_follow()
+giveProductionsForFollow(nonTerminal: String): List
+follow(): Set
+get_table_index(father: Node, is_sibling: Integer): Integer
+construct_parsing_table(sequence: String)
+parse(w: String): Tuple
+construct_M_table()