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):
```

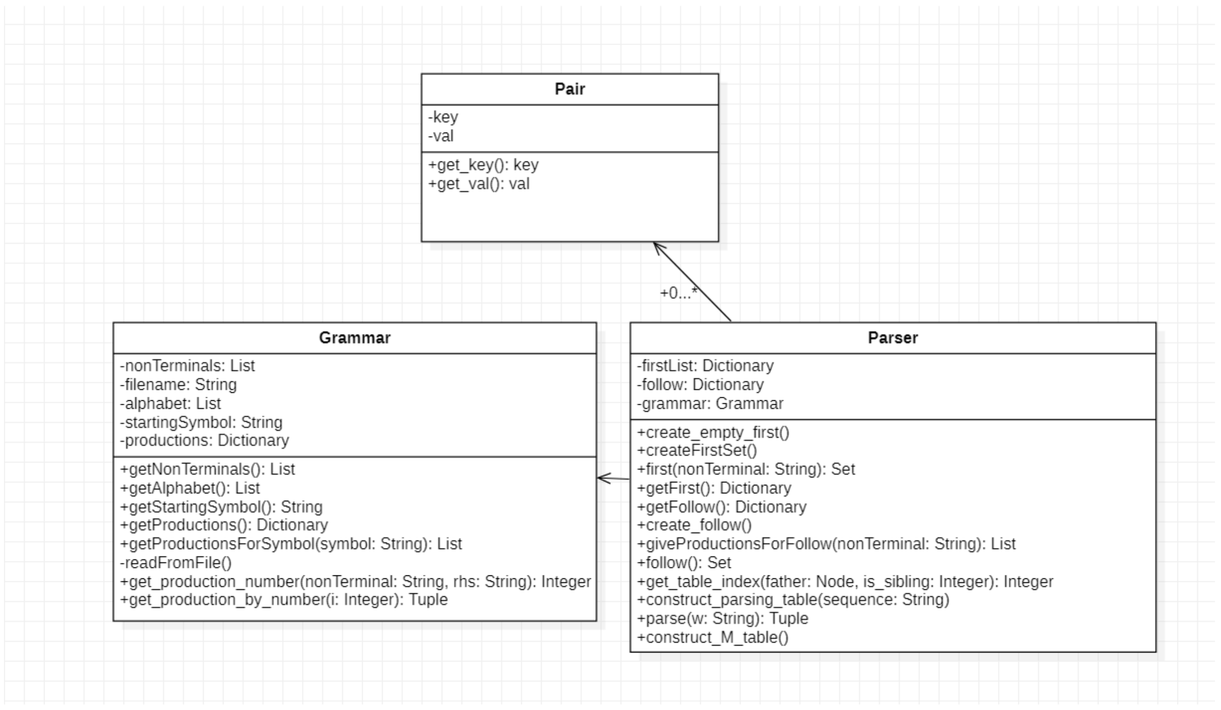
```
'''
Preconditions: none
Postconditions: constructs self.__M table, needed for the
                parsing process
'''
def construct_M_table(self):
```

```
'''
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:



Grammar:

```
N--{S,A,B,C,D}
E--{+,*,a,(,)}
S--S
S->B A
A->+ B A
A->epsilon
B->D C
C->* D C
C->epsilon
```

```
D->( S )
D->a
```

For:

```
W = individual a ; a = 2 ;"
```

We obtain:

('err', []) for ('S', 'individual')

For:

```
W = "a + ( a * a )"
```

We obtain:

('acc', [1, 4, 8, 6, 2, 4, 7, 1, 4, 8, 5, 8, 6, 3, 6, 3])

And the Parsing Table: (crt number, Node, Father, Sibling):

{1, {S, {-1, -1}}}

{2, {B, {1, -1}}}

{3, {A, {1, 2}}}

{4, {D, {2, -1}}}

{5, {C, {2, 4}}}

{6, {a, {4, -1}}}

{7, {epsilon, {5, -1}}}

{8, {+, {3, -1}}}

{9, {B, {3, 8}}}

{10, {A, {3, 9}}}

{11, {D, {9, -1}}}

{12, {C, {9, 11}}}

{13, {(, {11, -1}}}

{14, {S, {11, 13}}}

{15, {), {11, 14}}}

{16, {B, {14, -1}}}

{17, {A, {14, 16}}}

```
{18,  {D,  {16, -1}}}

{19,  {C,  {16, 18}}}

{20,  {a,  {18, -1}}}

{21,  {*,  {19, -1}}}

{22,  {D,  {19, 21}}}

{23,  {C,  {19, 22}}}

{24,  {a,  {22, -1}}}

{25,  {epsilon,  {23, -1}}}

{26,  {epsilon,  {17, -1}}}
```