Lark

docs source parser

Parser IDE

full screen

Load Grammar

Parser: Earley

Keep all tokens

```
//tokens
START_PROGRAM    : "Start "
FINISH_PROGRAM   : "Finish"
VAR              : "var "
WHILE            : "while"
IF               : "if "
ELSE             : "else "
DEF              : "def"
RETURN           : "return"
CLASS            : "class"
EXTENDS          : "extends"
PRINT            : "print"
PRIVATE          : "Private"
PUBLIC           : "Public"
PROTECTED        : "Protected"

LEFT_CURRLY_BRACES     : /{/
RIGHT_CURRLY_BRACES    : /}/
LEFT_PARENTHESIS  : /\(/
RIGHT_PARENTHESIS : /\)/

COMMA     : /,/
COLON     : /:/

NEW_LINE: /\n+/
WHITESPACE: (" " | /\t/ )+

GREATER_THAN       : />/
LESS_THAN          : /</
LESS_EQUAL         : /<=/
GREATER_EQUAL      : />=/
NOT_EQUAL          : /!=/
EQUAL_COMPARATION  : /==/
EQUAL              : /=/

PLUS               : /\+/
MINUS              : /-/
ASTERISK           : /\*/

SLASH              : /\//

INT_CTE       : "int"
FLOAT_CTE     : "float"
DOUBLE_CTE    : "double"
STRING_CTE    : "String"

IDENTIFIER: /[a-zA-Z_][a-zA-Z0-9_]*/
CONST_INT: /\d+/
CONST_FLOAT: /\d+(\.\d+)?/
STRING          : /".*"/


%ignore WHITESPACE
%ignore NEW_LINE

// rules
start: programa

programa : START_PROGRAM bloque_global FINISH_PROGRAM

bloque_global : LEFT_CURRLY_BRACES bg1 RIGHT_CURRLY_BRACES
bg1 : estatuto_global bg1 |

estatuto_global : asignacion | condicion | escritura | class_declaration | cic

asignacion : IDENTIFIER EQUAL expresion NEW_LINE

condicion  : IF LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS bloque condicion2
condicion1 : ELSE bloque |

bloque: LEFT_CURRLY_BRACES b1 RIGHT_CURRLY_BRACES
b1 : estatuto b1 |

estatuto    : asignacion
            | condicion
            | escritura
            | ciclo
            | vars

// TO do
tipo: INT_CTE
    | FLOAT_CTE
    | STRING_CTE

ciclo : WHILE LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS bloque

escritura : PRINT LEFT_PARENTHESIS esc1 esc2 RIGHT_PARENTHESIS NEW_LINE
esc1: expresion
    | STRING
esc2: COMMA esc1 |

vars    : VAR vars2
vars2   : IDENTIFIER vars3
vars3   : COMMA vars2
        | COLON tipo NEW_LINE

expresion    : exp expresion1
expresion1   : expresion2 exp |
expresion2   : GREATER_THAN
             | LESS_THAN
             | EQUAL_COMPARATION
             | NOT_EQUAL
             | GREATER_EQUAL
             | LESS_EQUAL |

exp : termino ex1
ex1 : ex2 exp |
ex2 : PLUS | MINUS

factor  : factor1 | factor2
factor1 : LEFT_PARENTHESIS expresion RIGHT_PARENTHESIS
factor2 : factor3 var_cte
factor3 : PLUS | MINUS |

termino : factor t1
t1 : t2 termino |
t2 : ASTERISK | SLASH


class_declaration : class | class_extends

class : LESS_THAN CLASS GREATER_THAN IDENTIFIER bloque_class

//to do
class_extends : //LESS_THAN CLASS GREATER_THAN IDENTIFIER EXTENDS class_cte bl

function  : DEF IDENTIFIER LEFT_PARENTHESIS function1 RIGHT_PARENTHESIS LEFT_C
function1 : var_cte COMMA function1 |
function2 : RETURN var_cte

bloque_class : LEFT_CURRLY_BRACES bc1 content_class bc2 content_class bc3 cont
bc1 : PRIVATE COLON |
bc2 : PUBLIC COLON |
bc3 : PROTECTED COLON |

content_class : cc1 |
cc1 : cc2 content_class
cc2 : vars | function | asignacion

// to do
var_cte : IDENTIFIER
        | CONST_INT
        | CONST_FLOAT
        | STRING
        | //OBJECT
```

```
Start
{
  <class> Person
  {
    Public
      var num : int
      var spreak : String
      num = 4
      speak = "hola mundo"
  }
  var aux : int
  aux = 4

  while (aux > 0) {
    print("hola mundo" )
    print("\n")
    aux = aux - 1
  }
}
Finish
```