

Gramática DECAF aumentada

<foo>	No terminal
foo	Terminal
[a]	Cero o una
a*	Cero o más
a+	Uno o más
a b	a ó b
(a)	Agrupación
a*,	Cero o más a separadas por coma

id = letter (letter|digit)*
num = digit (digit)*
char = letter

```

<program>          → 'class' 'Program' '{' (<declaration>)* '}'
<declaration>      → <structDeclaration>
                   | <varDeclaration>
                   | <methodDeclaration>

<varDeclaration>   → <varType> id ';' | <varType> id '[' num ']' ';'
<structDeclaration> → 'struct' id '{' (<varDeclaration>)* '}'
<varType>          → 'int'
                   | 'char'
                   | 'boolean'
                   | 'struct' id
                   | <structDeclaration>
                   | 'void'

<methodDeclaration> → <methodType> id '(' (<parameter>)*, ')' <block>
<methodType>       → 'int'
                   | 'char'
                   | 'boolean'
                   | 'void'

<parameter>        → <parameterType> id
<parameterType>    → <parameterType> id '[' ']'
                   | 'int'
                   | 'char'
                   | 'boolean'

<block>            → '{' (<varDeclaration>)* (<statement>)* '}'
<statement>        → 'if' '(' <expression> ')' <block> ['else' <block>]
                   → 'while' '(' <expression> ')' <block>
                   → 'return' [<expression>] ';'
                   → <methodCall> ';'
                   → <block>

<location>         → <location> '=' <expression>
<expression>       → [<expression>] ';'
                   → (id | id '[' <expression> ']' ) [ '.' <location> ]
                   | <methodCall>
                   | <literal>
                   | <expression> <op> <expression>
                   | '-' <expression>
                   | '!' <expression>
                   | '(' <expression> ')'

<methodCall>       → id '(' (<arg>)*, ')'
<arg>               → <expression>
<op>                → <arith_op> | <rel_op> | <eq_op> | <cond_op>
<arith_op>          → '+' | '-' | '*' | '/' | '%'
<rel_op>             → '<' | '>' | '<=' | '>='
<eq_op>              → '==' | '!='
<cond_op>            → '&&' | '||'
<literal>            → <int_literal> | <char_literal> | <bool_literal>
<int_literal>        → num
<char_literal>       → ''' char '''
<bool_literal>       → 'true' | 'false'

```