PRIEMERA GRAMATICA:

/\*\*

 \* Ejemplo mi primer proyecto con Jison utilizando Nodejs en Ubuntu

 \*/

%lex

%options case-insensitive

%%

\s+                           //ignorando los espacios en blanco

"//".\*                        // ignore comment line

[/][\*][^\*]\*[\*]+([^/\*][^\*]\*[\*]+)\*[/]       //ignore comment Multilinea

"public"            return 'public\_';

"class"             return 'class\_';

"interface"         return 'interface\_';

"void"              return 'void\_';

"static"            return 'static\_';

"main"              return 'main\_';

"args"              return 'args\_';

"int"               return 'int\_';

"String"            return 'string\_';

"boolean"           return 'boolean\_';

"double"            return 'double\_';

"char"              return 'char\_';

"for"               return 'for\_';

"do"                return "do\_";

"while"             return 'while\_';

"if"                return 'if\_';

"else"              return 'else\_';

"System"            return 'system\_';

"out"               return 'out\_';

"println"           return 'println\_';

"print"             return 'print\_';

"this"              return 'this\_';

"false"             return 'false\_';

"true"              return 'true\_';

"break"             return 'break\_';

"continue"          return 'continue\_';

"return"            return 'return\_';

"("                 return 'parAbre';

")"                 return 'parCierra';

","                 return 'coma';

"{"                 return 'llaveAbre';

"}"                 return 'llaveCierra';

";"                 return 'puntoComa';

"."                 return 'punto';

"["                 return 'cocheteAbre';

"]"                 return 'corcheteCierra';

"="                 return 'igual';

"+"                 return 'mas';

"-"                 return 'menos';

"\*"                 return 'por';

"/"                 return 'division';

"&&"                return 'and\_';

"||"                return 'or\_';

"!"                 return 'not\_';

"^"                 return 'xor\_';

"<"                 return 'menorQue';

">"                 return 'mayorQue';

">="                return 'mayorIgual';

"<="                return 'menorIgual';

"=="                return 'igualIgual';

"!="                return 'diferenteA';

\"[^\"]\*\"                    { yytext = yytext.substr(1,yyleng-2); return 'cadena'; /\*//"\*/ }

/\* Espacios en blanco \*/

[ \r\t]+            {}

\n                  {}

[0-9]+\b                            return 'entero';

[0-9]+("."[0-9]+)?\b                return 'decimal';

("-")[0-9]+("."[0-9]+)?\b           return 'negativo';

([a-zA-Z\_])[a-zA-Z0-9\_]\*            return 'identificador';

<<EOF>>                             return 'EOF';

.     {

            console.error('Error léxico: ' + yytext + ', line: ' + yylloc.first\_line + ', column: ' + yylloc.first\_column);

      }

/lex

%left 'or\_'

%left 'and\_'

%left 'mayorQue' 'menorQue'

%left 'mas' 'menos'

%left 'por' 'division'

%left uMenos

%right 'not\_'

%left 'MAS' 'MENOS'

%left 'POR' 'DIVIDIDO'

%left UMENOS

%start INI

%%

INI: INSTRUCCIONES EOF

;

INSTRUCCIONES: INSTRUCCIONES CLASE

             | CLASE

;

CLASE: public\_ TIPO\_CLASE identificador llaveAbre SENTENCIAS\_GLOBALES llaveCierra

     | public\_ TIPO\_CLASE identificador llaveAbre llaveCierra

;

TIPO\_CLASE: class\_

          | interface\_

;

SENTENCIAS\_GLOBALES: SENTENCIAS\_GLOBALES SENTENCIAG

                  | SENTENCIAG

;

SENTENCIAG: METODO

          | DECLARACION\_VARIABLE puntoComa

          | METODO\_MAIN

;

METODO: public\_ TIPO identificador parAbre PARAMETROS parCierra  BLOQUE\_SENTENCIAS

;

METODO\_MAIN: public\_ static\_ void\_ main\_ parAbre string\_ cocheteAbre corcheteCierra args\_ BLOQUE\_SENTENCIAS

;

DECLARACION\_VARIABLE: DECLARACION\_VARIABLE  coma DECLARACION

                    | DECLARACION

;

DECLARACION: TIPO EXPRESION

;

TIPO:int\_

    | double\_

    | string\_

    | char\_

    | void\_

    | boolean\_

;

BLOQUE\_SENTENCIAS: llaveAbre LISTA\_SENTENCIAS llaveCierra

                 | llaveAbre llaveCierra

;

LISTA\_SENTENCIAS: LISTA\_SENTENCIAS SENTENCIAS

                | SENTENCIAS

;

SENTENCIAS: IF

          | FOR

          | RETURN

          | DO\_WHILE

          | WHILE

          | break\_ puntoComa

          | continue\_ puntoComa

          | ASIGNACION\_VARIABLE     puntoComa

          | DECLARACION\_VARIABLE   puntoComa

          | PRINT

          | LLAMADAS

;

PRINT: system\_ punto out\_ punto IMPRESION parAbre EXPRESION parCierra puntoComa

;

IMPRESION: println

         | print\_

;

ASIGNACION\_VARIABLE: identificador igual EXPRESION

                   | this\_ identificador igual EXPRESION

;

LLAMADAS: identificador parAbre PARAMETROS parCierra puntoComa

;

IF: if\_ parAbre CONDICION parCierra BLOQUE\_SENTENCIAS

  | if\_ parAbre CONDICION parCierra BLOQUE\_SENTENCIAS ELSEIF

  | if\_ parAbre CONDICION parCierra BLOQUE\_SENTENCIAS ELSE

;

ELSEIF: else\_ if\_ parAbre CONDICION parCierra BLOQUE\_SENTENCIAS

      | else\_ if\_ parAbre CONDICION parCierra BLOQUE\_SENTENCIAS ELSEIF

      | else\_ if\_ [arAbre CONDICION parCierra BLOQUE\_SENTENCIAS ELSE

;

ELSE: else\_ BLOQUE\_SENTENCIAS

;

FOR: for\_ parAbre DECLARACION\_VARIABLE puntoComa EXPRESION puntoComa EXPRESION parCierra BLOQUE\_SENTENCIAS

;

WHILE: while\_ parAbre CONDICION parCierra  BLOQUE\_SENTENCIAS

;

DO\_WHILE:  do\_ BLOQUE\_SENTENCIAS while\_ parAbre CONDICION parCierra puntoComa

;

RETURN: return\_ puntoComa

      | return\_ EXPRESION puntoComa

;

PARAMETROS: PARAMETROS coma PARAMETRO

          | PARAMETRO

;

PARAMETRO: TIPO identificador

         | ARITMETICA

         | cadena

;

EXPRESION: ARITMETICA

         | cadena

         | LLAMADAS

         | this\_ punto identificador

;

CONDICION: EXPRESION menorQue EXPRESION

          |EXPRESION menorIgual EXPRESION

          |EXPRESION igual EXPRESION

          |EXPRESION mayorQue EXPRESION

          |EXPRESION mayorIgual EXPRESION

          |EXPRESION igualIgual EXPRESION

          |EXPRESION diferenteA EXPRESION

          |EXPRESION menorQue EXPRESION LOGICOS CONDICION

          |EXPRESION menorIgual EXPRESION LOGICOS CONDICION

          |EXPRESION igual EXPRESION LOGICOS CONDICION

          |EXPRESION mayorQue EXPRESION LOGICOS CONDICION

          |EXPRESION mayorIgual EXPRESION LOGICOS CONDICION

          |EXPRESION igualIgual EXPRESION LOGICOS CONDICION

          |EXPRESION diferenteA EXPRESION LOGICOS CONDICION

;

LOGICOS: and\_

       | or\_

       | not\_

       | xor\_

;

ARITMETICA : E

;

E: E mas T

 | E menos T

 | T

;

T: T por F

 | T division F

 | F

;

F: identificador

 | entero

 | decimal

 | negativo

 | parAbre E parCierra

;