MANUAL TECNICO

VICTOR VASQUEZ CORONADO
201114301

GRAMATICA

ARCHIVO JSON

LEXICO.JFLEX

Ingresamos nuestro package

```
package Compilador;

importamos por aca las librerías mas importantes que se va a
utilizar

import java_cup.runtime.Symbol;
import javax.swing.*;
import java.io.*;
import java.awt.*;


/* segunda parte: declaramos las directivas y los macros
El codigo entre %{  y %} sera copiado integramente en el
    analizador generado.
*/
%%
%class JavaLexicoJSON
%public
%full
%unicode
%line
%column
%char
%cup

/*
   Activamos la compatibilidad con Java CUP para analizadores
   sintacticos(parser)
    Activar el contador de lineas, variable yyline
    Activar el contador de columna, variable yycolumn
*/


Letra = [a-zA-Zñáéíóú]
FinLinea = \r|\n|\r\n|\n\r
EspacioBlanco = {FinLinea} | [ \t\f]
Numero = 0|[1-9][0-9]*
ID = [a-zA-Z]([a-zA-Z]*[0-9]* |"_"|{Letra})*
URL = ([a-zA-Z]+)":" "\\" {ID}"."{ID}
SIGNO = "+"|"-"

comilla = ["\""]

%%
/* OPERADORES Y SIGNOS */
  Macro declaraciones
```

```
    Declaramos expresiones regulares que despues usaremos en las
    reglas lexicas.

Si se desea agregar algún signo únicamente hay que agregarlo al
programa para que este funcione correctamente

":" {return new Symbol(sym.DSPTS, new token(yycolumn, yyline,
yytext()));}
"," {return new Symbol(sym.COMA, new token(yycolumn, yyline,
yytext()));}
"{" {return new Symbol(sym.ABRIRLLAVE, new token(yycolumn, yyline,
yytext()));}
"}" {return new Symbol(sym.CERRARLLAVE, new token(yycolumn,
yyline, yytext()));}
"[" {return new Symbol(sym.ABRECOR, new token(yycolumn, yyline,
yytext()));}
"]" {return new Symbol(sym.CIERRACOR, new token(yycolumn, yyline,
yytext()));}


/* PALABRAS RESERVADAS */
Aca van todas las palabras reservadas del lenguaje existen un
monton y están están en symbol sym cada una
Si se encuentra un numero, se imprime, se regresa un token numero
        que representa un entero y el valor que se obtuvo de la
cadena yytext
        al convertirla a entero. yytext es el token encontrado.

"mapa" {return new Symbol(sym.MAPA, new token(yycolumn, yyline,
yytext()));}
"Tamano" {return new Symbol(sym.TAMANO, new token(yycolumn,
yyline, yytext()));}
"x" {return new Symbol(sym.X, new token(yycolumn, yyline,
yytext()));}
"y" {return new Symbol(sym.Y, new token(yycolumn, yyline,
yytext()));}
"objetos" {return new Symbol(sym.OBJETOS, new token(yycolumn,
yyline, yytext()));}
"Posicion" {return new Symbol(sym.POSICION, new token(yycolumn,
yyline, yytext()));}
"Nombre" {return new Symbol(sym.NOMBRE, new token(yycolumn,
yyline, yytext()));}
"Bonificacion" {return new Symbol(sym.BONIFICACION, new
token(yycolumn, yyline, yytext()));}
"valor" {return new Symbol(sym.VALOR, new token(yycolumn, yyline,
yytext()));}
"sentido" {return new Symbol(sym.SENTIDO, new token(yycolumn,
yyline, yytext()));}
"Imagen" {return new Symbol(sym.IMAGEN, new token(yycolumn,
yyline, yytext()));}
"enemigos" {return new Symbol(sym.ENEMIGOS, new token(yycolumn,
yyline, yytext()));}
"Movimiento" {return new Symbol(sym.MOVIMIENTO, new
token(yycolumn, yyline, yytext()));}
```

```
"Dano" {return new Symbol(sym.DANO, new token(yycolumn, yyline,
yytext()));}
"heroe" {return new Symbol(sym.HEROE, new token(yycolumn, yyline,
yytext()));}
"Armadura" {return new Symbol(sym.ARMADURA, new token(yycolumn,
yyline, yytext()));}
"meta" {return new Symbol(sym.META, new token(yycolumn, yyline,
yytext()));}
"muros" {return new Symbol(sym.MUROS, new token(yycolumn, yyline,
yytext()));}


/* EXPRESIONES */
Expresiones regulares reciben para formanr una palabra reservada

{Numero} {return new Symbol(sym.NUMERO, new token(yycolumn,
yyline, yytext()));}
{ID} {return new Symbol(sym.ID, new token(yycolumn, yyline,
yytext()));}
{URL} {return new Symbol(sym.URL, new token(yycolumn, yyline,
yytext()));}
{SIGNO} {return new Symbol(sym.SIGNO, new token(yycolumn, yyline,
yytext()));}
{FinLinea} {/* ignorar */}
{EspacioBlanco} {/* ignorar */}
{comilla} {/* ignorar */}
. {System.err.println("ERROR LEXICO, caracter invalido: " +
yytext() + " ["+ yyline + " : "+ yycolumn + "]");

      JOptionPane.showMessageDialog(null, "Error léxico, carácter
no reconocido: " + yytext() );
}
```

SINTACTICO.CUP

```
package Compilador;

import java_cup.runtime.*;
import java.util.ArrayList;
import java.io.FileReader;
import javax.swing.JOptionPane;


action code {:
  ArrayList<Instruccion> instrucciones = new ArrayList();
:}

parser code {:
   //esta es la manera en que se puede acceder a un objeto que se genera durante la etapa del
parsing
/* Codigo del parser, se copia integramente a la clase final.
   Agregamos el manejo de errores. */
```

```
   public ArrayList<Instruccion> getInstrucciones(){
   return action_obj.instrucciones;
   }

   @Override
   public void syntax_error(Symbol sy) {
      token t=(token)sy.value;
      done_parsing();
      report_error("ERROR SINTACTICO: " + t.getCadena() + " [" +t.getRow()+ ":" +t.getCol()+ "]"
,null);
   }
:}

   /* Cuando se encuentra un error de donde el sistema no puede
      recuperarse, se lanza un error fatal. Se despliega el mensaje
      de error y se finaliza la ejecucion. */
   public void report_fatal_error(String message, Object info) {
      report_error(message, info);
      System.exit(1);
   }


/*---------------------------------TERMINALES-----------------------------*/

terminal DSPTS, COMA, ABRIRLLAVE, CERRARLLAVE, ABRECOR, CIERRACOR, SIGNO;
terminal MAPA, TAMANO, X, Y, OBJETOS, POSICION, BONIFICACION, VALOR, SENTIDO, IMAGEN;
terminal ENEMIGOS, MOVIMIENTO, DANO, HEROE, ARMADURA, META, MUROS, NOMBRE,
NUMERO;
terminal token ID;
terminal token URL;

/*-----------------------------NO TERMINALES----------------------------*/

non terminal inicio, instrucciones, muros, muro;
non terminal mapa, valores, heroe, meta, imagen, objetos, objeto, bonificacion, finales, enemigos,
enemigo;

/*-------------------------------GRAMATICAS-------------------------------*/
VALOR DE ARRANQUE EN LA GRAMATICA
start with inicio;

/*----------------------ESTRUCTURA DE PROGRAMA--------------------------*/
inicio ::= ABRIRLLAVE instrucciones
        |instrucciones
        |CERRARLLAVE
        ;

/*--------------------DEFINICION DE INSTRUCCIONES-------------------------*/
ACA VA LA ESTRUCTURA DE LA GRAMATICA QUE ES MUY IMPORTANTE
LA GRAMATICA ESTA SENCILLA DE EXPLICARLA Y ENTENDERLA
```

```
instrucciones ::= mapa inicio
            | heroe inicio
            | meta inicio
            | objetos inicio
            | enemigos inicio
            | muros inicio
           ;

valores ::= X DSPTS NUMERO COMA Y DSPTS NUMERO
         ;

imagen ::= IMAGEN DSPTS URL
       ;

mapa ::= MAPA DSPTS ABRIRLLAVE TAMANO DSPTS ABRIRLLAVE valores CERRARLLAVE
CERRARLLAVE COMA
      ;

heroe ::= HEROE DSPTS ABRIRLLAVE MOVIMIENTO DSPTS ID COMA DANO DSPTS NUMERO COMA
ARMADURA DSPTS NUMERO COMA POSICION DSPTS ABRIRLLAVE valores CERRARLLAVE COMA
imagen COMA NOMBRE DSPTS ID CERRARLLAVE COMA
       ;

meta ::= META DSPTS ABRIRLLAVE POSICION DSPTS ABRIRLLAVE valores CERRARLLAVE COMA
imagen CERRARLLAVE COMA
     ;

objetos ::= OBJETOS DSPTS ABRECOR objeto
       | objeto
       | CIERRACOR COMA
       ;

objeto ::= ABRIRLLAVE POSICION DSPTS ABRIRLLAVE valores CERRARLLAVE COMA bonificacion
imagen finales
       ;

bonificacion ::= BONIFICACION DSPTS ABRIRLLAVE VALOR DSPTS NUMERO COMA SENTIDO DSPTS
SIGNO CERRARLLAVE COMA
           ;

finales ::= CERRARLLAVE COMA
       | CERRARLLAVE
       | CIERRACOR COMA
       ;

enemigos ::= ENEMIGOS DSPTS ABRECOR enemigo
         | enemigo CIERRACOR COMA
        ;

enemigo ::= ABRIRLLAVE MOVIMIENTO DSPTS ID COMA DANO DSPTS NUMERO COMA POSICION
DSPTS ABRIRLLAVE valores CERRARLLAVE COMA imagen finales
        ;
```

```
muros ::= MUROS DSPTS ABRECOR muro
      | muro
      | CIERRACOR
      ;

muro ::= ABRIRLLAVE POSICION DSPTS ABRIRLLAVE valores CERRARLLAVE finales
     ;
```

ARCHIVO MUV

LEXICO.JFLEX

```
package CompiladorMUV;
import java_cup.runtime.Symbol;
import javax.swing.*;
import java.io.*;


/* segunda parte: declaramos las directivas y los macros */
%%
%class JavaLexicoMUV
%public
%full
%unicode
%line
%column
%char
%cup

/*
LineTerminator = \r|\n|\r\n|\n\r
WhiteSpace = {LineTerminator} | [ \t\f]
ValorEntero = 0|[1-9][0-9]*
Comentario = [//] [a-z][a-z]*[0-9]*

a-zA-Zñáéíóú
*/


Letra = [a-zA-Zñáéíóú]
FinLinea = \r|\n|\r\n|\n\r
EspacioBlanco = {FinLinea} | [ \t\f]
Numero = 0|[1-9][0-9]*
ID = [a-zA-Z]([a-zA-Z]*[0-9]* |"_"|{Letra})*
URL = ([a-zA-Z]+)":" "\\" {ID}"."{ID}


/*comilla = ["\""]*/
```

```
%%
/* OPERADORES Y SIGNOS */
":" {return new Symbol(sym.DOSPUNTOS, new tkMUV(yycolumn, yyline, yytext()));}
";" {return new Symbol(sym.PUNTOYCOMA, new tkMUV(yycolumn, yyline, yytext()));}
"," {return new Symbol(sym.COMA, new tkMUV(yycolumn, yyline, yytext()));}
"{" {return new Symbol(sym.LLAVEABRE, new tkMUV(yycolumn, yyline, yytext()));}
"}" {return new Symbol(sym.LLAVECIERRA, new tkMUV(yycolumn, yyline, yytext()));}
"(" {return new Symbol(sym.PARENTABRE, new tkMUV(yycolumn, yyline, yytext()));}
")" {return new Symbol(sym.PARENTCIERRA, new tkMUV(yycolumn, yyline, yytext()));}
"==" {return new Symbol(sym.COMPARACION, new tkMUV(yycolumn, yyline, yytext()));}
"=" {return new Symbol(sym.IGUAL, new tkMUV(yycolumn, yyline, yytext()));}
"+" {return new Symbol(sym.MAS, new tkMUV(yycolumn, yyline, yytext()));}
"-" {return new Symbol(sym.MENOS, new tkMUV(yycolumn, yyline, yytext()));}
"*" {return new Symbol(sym.MULTIPLICACION, new tkMUV(yycolumn, yyline, yytext()));}
"^" {return new Symbol(sym.POTENCIA, new tkMUV(yycolumn, yyline, yytext()));}
"/" {return new Symbol(sym.DIVISION, new tkMUV(yycolumn, yyline, yytext()));}
"%" {return new Symbol(sym.MODULO, new tkMUV(yycolumn, yyline, yytext()));}
">" {return new Symbol(sym.MAYOR, new tkMUV(yycolumn, yyline, yytext()));}
"<" {return new Symbol(sym.MENOR, new tkMUV(yycolumn, yyline, yytext()));}
"&&" {return new Symbol(sym.AND, new tkMUV(yycolumn, yyline, yytext()));}
"||" {return new Symbol(sym.OR, new tkMUV(yycolumn, yyline, yytext()));}
"!=" {return new Symbol(sym.NOT, new tkMUV(yycolumn, yyline, yytext()));}



/* PALABRAS RESERVADAS */
"public" {return new Symbol(sym.PUBLICO, new tkMUV(yycolumn, yyline, yytext()));}
"void" {return new Symbol(sym.VOID, new tkMUV(yycolumn, yyline, yytext()));}
"Integer" {return new Symbol(sym.INTEGER, new tkMUV(yycolumn, yyline, yytext()));}
"bool" {return new Symbol(sym.BOOL, new tkMUV(yycolumn, yyline, yytext()));}
"String" {return new Symbol(sym.STRING, new tkMUV(yycolumn, yyline, yytext()));}
"movleft" {return new Symbol(sym.MOVLEFT, new tkMUV(yycolumn, yyline, yytext()));}
"movright" {return new Symbol(sym.MOVRIGHT, new tkMUV(yycolumn, yyline, yytext()));}
"movup" {return new Symbol(sym.MOVUP, new tkMUV(yycolumn, yyline, yytext()));}
"movdown" {return new Symbol(sym.MOVDOWN, new tkMUV(yycolumn, yyline, yytext()));}
"say" {return new Symbol(sym.SAY, new tkMUV(yycolumn, yyline, yytext()));}
"convert" {return new Symbol(sym.CONVERT, new tkMUV(yycolumn, yyline, yytext()));}
"wait" {return new Symbol(sym.WAIT, new tkMUV(yycolumn, yyline, yytext()));}
"if" {return new Symbol(sym.IF, new tkMUV(yycolumn, yyline, yytext()));}
"for" {return new Symbol(sym.FOR, new tkMUV(yycolumn, yyline, yytext()));}
"while" {return new Symbol(sym.WHILE, new tkMUV(yycolumn, yyline, yytext()));}
"switch" {return new Symbol(sym.SWITCH, new tkMUV(yycolumn, yyline, yytext()));}
"case" {return new Symbol(sym.CASE, new tkMUV(yycolumn, yyline, yytext()));}
"break" {return new Symbol(sym.BREAK, new tkMUV(yycolumn, yyline, yytext()));}
"default"  {return new Symbol(sym.DEFAULT, new tkMUV(yycolumn, yyline, yytext()));}
"true"  {return new Symbol(sym.TRUE, new tkMUV(yycolumn, yyline, yytext()));}
"false"  {return new Symbol(sym.FALSE, new tkMUV(yycolumn, yyline, yytext()));}



/* EXPRESIONES */
```

```
{Numero} {return new Symbol(sym.NUMERO, new tkMUV(yycolumn, yyline, yytext()));}
{ID} {return new Symbol(sym.ID, new tkMUV(yycolumn, yyline, yytext()));}
{URL} {return new Symbol(sym.URL, new tkMUV(yycolumn, yyline, yytext()));}
("\""){ID}("\"")  {return new Symbol(sym.CADENA, new tkMUV(yycolumn, yyline, yytext()));}
{FinLinea} {/* ignorar */}
{EspacioBlanco} {/* ignorar */}
. {System.err.println("ERROR LEXICO, caracter invalido: " + yytext() + " ["+ yyline + " : "+ yycolumn +
"]");

    JOptionPane.showMessageDialog(null, "Error léxico, carácter no reconocido: " + yytext() );

}
```

SINTACTICO.CUP

```
package CompiladorMUV;

import java_cup.runtime.*;
import java.util.ArrayList;
import java.io.FileReader;
import javax.swing.JOptionPane;


action code {:
  ArrayList<instrucMUV> instrucciones = new ArrayList();
:}

parser code {:
   //esta es la manera en que se puede acceder a un objeto que se genera durante la etapa del
parsing

/* Codigo del parser, se copia integramente a la clase final.
   Agregamos el manejo de errores. */

   public ArrayList<instrucMUV> getInstrucciones(){
   return action_obj.instrucciones;
   }

   @Override
   public void syntax_error(Symbol sy) {
      tkMUV t=(tkMUV)sy.value;
      done_parsing();
      report_error("ERROR SINTACTICO: " + t.getCadena() + " [" +t.getRow()+ ":" +t.getCol()+ "]"
,null);
   }
:}


/*--------------------------------TERMINALES----------------------------*/
```

```
terminal PUBLICO, VOID, ID, INTEGER, BOOL, STRING, PARENTABRE, PARENTCIERRA;
terminal LLAVEABRE, LLAVECIERRA, COMPARACION, IGUAL, MOVLEFT, MOVRIGHT;
terminal MOVUP, MOVDOWN, SAY, CONVERT, WAIT, PUNTOYCOMA, MAS, MENOS,
MULTIPLICACION;
terminal DIVISION, POTENCIA, MODULO, MAYOR, MENOR, AND, OR, NOT, COMA, IF;
terminal FOR, WHILE, SWITCH, CASE, BREAK, DEFAULT, DOSPUNTOS, VERDADERO, FALSO;
terminal COMENTARIO, CADENA;
terminal URL, FALSE, TRUE;

terminal Integer NUMERO;


/*------------------------------NO TERMINALES----------------------------*/


non terminal inicio, methodos, metodo, cuerpo, elemento, declaracion, asignacion, if_, for_,
while_;
non terminal switch_, metodos, tipo, l_id, expr_list, bool, condicion, conectorCond;
non terminal op, comp, equal, conectorLogico, incremento, expr_part, l_casos;
non terminal caso, param, convert1, convert2;

non terminal Integer expr, factor, term, term2;

precedence left OR;
precedence left AND;
precedence left NOT;
precedence left COMPARACION;
precedence left MAYOR, MENOR;
precedence left MAS, MENOS;
precedence left MULTIPLICACION, DIVISION;
precedence left POTENCIA, MODULO;
precedence right IGUAL;

/*------------------------------GRAMATICAS------------------------------*/
start with inicio;

/*----------------------ESTRUCTURA DE PROGRAMA--------------------------*/
inicio ::= methodos
                    |error
                    ;

/*--------------------DEFINICION DE INSTRUCCIONES-----------------------*/
methodos ::= methodos metodo
                    |metodo
                    ;

metodo ::= PUBLICO VOID ID PARENTABRE PARENTCIERRA LLAVEABRE cuerpo LLAVECIERRA
                    |COMENTARIO
                    ;
```

```
cuerpo ::= elemento cuerpo
                    |elemento
                    ;

elemento ::= asignacion
                    |declaracion
                    |COMENTARIO
                    |if_
                    |for_
                    |while_
                    |switch_
                    |metodos
                    ;

declaracion ::= tipo l_id PUNTOYCOMA
                    ;

l_id ::= ID COMA l_id
                    |ID
                    ;

tipo ::= INTEGER
                    |BOOL
                    |STRING
                    ;

asignacion ::= tipo ID IGUAL expr_list PUNTOYCOMA
                            |ID IGUAL expr_list PUNTOYCOMA
                            |ID IGUAL metodos
                            ;

bool ::= VERDADERO
                    |FALSO
                    ;

if_ ::= IF PARENTABRE condicion PARENTCIERRA LLAVEABRE cuerpo LLAVECIERRA
                    ;

condicion ::= PARENTABRE condicion PARENTCIERRA conectorCond
                    |comp op comp conectorCond
                    ;

comp ::= ID
                    |NUMERO
                    ;

op ::= COMPARACION
          |MAYOR equal
          |MENOR equal
          ;

equal ::= IGUAL
```

```
                    |
                    ;

conectorCond ::= conectorLogico condicion
                            |
                            ;

conectorLogico ::= AND
                                    |OR
                                    |NOT
                                    ;

for_ ::= FOR PARENTABRE ID IGUAL expr_list PUNTOYCOMA ID op expr_list PUNTOYCOMA
incremento PARENTCIERRA LLAVEABRE cuerpo LLAVECIERRA
                            ;

incremento ::= ID IGUAL expr_list
                            |ID MAS MAS
                            |ID MENOS MENOS
                            ;

expr_list ::= expr_list expr_part
                            |expr_part
                            ;

expr_part ::= expr
                            ;

expr ::= expr MAS factor
                    |expr MENOS factor
                    |factor
                    ;

factor ::= factor MULTIPLICACION term2
                    |factor DIVISION term2
                    |term2
                    ;

term2 ::= term2 POTENCIA term
        |term2 MODULO term
                     |term
                     ;

term ::= PARENTABRE expr PARENTCIERRA
                    |NUMERO
                    |ID
                    |CADENA
                    |bool
                    ;

while_ ::= WHILE PARENTABRE condicion PARENTCIERRA LLAVEABRE cuerpo LLAVECIERRA
                    ;
```

```
switch_ ::= SWITCH PARENTABRE ID PARENTCIERRA LLAVEABRE l_casos LLAVECIERRA
                        ;

l_casos ::= caso l_casos
          |caso
          ;

caso ::= CASE NUMERO DOSPUNTOS cuerpo BREAK PUNTOYCOMA
          |DEFAULT DOSPUNTOS cuerpo BREAK PUNTOYCOMA
          ;

metodos ::= MOVDOWN PARENTABRE comp PARENTCIERRA PUNTOYCOMA
          |MOVUP PARENTABRE comp PARENTCIERRA PUNTOYCOMA
          |MOVRIGHT PARENTABRE comp PARENTCIERRA PUNTOYCOMA
          |MOVLEFT PARENTABRE comp PARENTCIERRA PUNTOYCOMA
          |WAIT PARENTABRE comp PARENTCIERRA PUNTOYCOMA
          |SAY PARENTABRE param PARENTCIERRA PUNTOYCOMA
          |CONVERT PARENTABRE convert1 COMA convert2 PARENTCIERRA PUNTOYCOMA
          ;

param ::= CADENA
          |ID
          ;

convert1 ::= ID
          |CADENA
          |NUMERO
          ;

convert2 ::= INTEGER
          |STRING
          |BOOL
          ;
```

DIAGRAMA GENERAL DEL PROYECTO

Diagrama general del proyecto