```
package ejemplo.jflex;
import java.util.Stack;

/* Aqui arriva se importan todos los paquetes que necesite el codigo */

/**
 * Esta clase es un ejemplo simple de un lexer.
 */
%%

%public
%class MiLexico
%unicode
%type MiToken
%line
%column

%{
   /**************************************************************************
   * En esta sección se puede incluir código que se copiará textualmente
   * como parte de la definición de la clase del analizador léxico.
   * Típicamente serán variables de instancia o nuevos métodos de la clase.
   **************************************************************************/

   /* Variables para reconocer Strings */
   StringBuffer string = new StringBuffer();
   int string_yyline = 0;
   int string_yycolumn = 0;

   Stack<Character> comentarioPila = new Stack<>();
   int nivelComentario = 0;
   private MiToken token(String nombre) {
      return new MiToken(nombre, this.yyline, this.yycolumn);
   }
   private MiToken token(String nombre, Object valor) {
      return new MiToken(nombre, this.yyline, this.yycolumn, valor);
   }
   private MiToken token(String nombre, int line, int column, Object valor) {
      return new MiToken(nombre, line, column, valor);
   }

%}

/*Aqui se definen los PATRONES con sus expresiones regulares asociadas*/

LineTerminator = \r|\n|\r\n
WhiteSpace     = {LineTerminator} | [ \t\f]
IDENTIFICADOR = [a-zA-Z_][a-zA-Z0-9_]*
```

```
LIT_INTEGER = [0-9]+
LIT_FLOAT = [0-9]+\.[0-9]+|[0-9]+\.|\.[0-9]+
LIT_ARRAY = "\[" \s* {LIT_FLOAT} (\s* , \s* {LIT_FLOAT})* \s* "\]"
PAREN_ABIERTO     = \(
PAREN_CERRADO    = \)
CORCHETE_ABIERTO   = \[
CORCHETE_CERRADO  = \]
LLAVE_ABIERTO     = \{
LLAVE_CERRADO    = \}
COMMA          = ,
ASIGNACION     = :=
COMENTARIO_LINEA  = \$.*


/* Defino un estado para los strings*/
%state CADENA
/* Defino un estado para los comentarios*/
%state COMENTARIO


%%

<YYINITIAL> {
/* palabras reservadas */
"DECLARE.SECTION"     { return token("DECLARE_SECTION", yytext()); }
"ENDDECLARE.SECTION"  { return token("ENDDECLARE_SECTION", yytext()); }
"PROGRAM.SECTION"     { return token("PROGRAM_SECTION", yytext());}
"ENDPROGRAM.SECTION"  { return token("ENDPROGRAM_SECTION", yytext()); }
"LOOP WHEN"          { return token("LOOP_WHEN", yytext()); }
"BACKWARD_LOOP WHEN" { return token("BACKWARD_LOOP_WHEN", yytext()); }
"THEN"               { return token("THEN", yytext()); }
"END_LOOP"           { return token("END_LOOP", yytext()); }
"CONDITION"          { return token("CONDITION", yytext()); }
"BACKWARD_CONDITION" { return token("BACKWARD_CONDITION", yytext()); }
"ELSE"               { return token("ELSE", yytext()); }
"ELSE_BACKWARD"      { return token("ELSE_BACKWARD", yytext()); }
"END"                { return token("END", yytext()); }
"BREAK"              { return token("BREAK", yytext()); }
"CONTINUE"           { return token("CONTINUE", yytext()); }
"promedio_ponderado" { return token("PROMEDIO_PONDERADO", yytext()); }



/* Tipos de datos */
"intiger"     { return token("TYPE_INTEGER", yytext()); }
"float"       { return token("TYPE_FLOAT", yytext()); }
"boolean"     { return token("TYPE_BOOLEAN", yytext()); }
"float_array" { return token("TYPE_FLOAT_ARRAY", yytext()); }
```

```
/* Operadores */
":" { return token("DECLARACION", yytext()); }
"+"  { return token("OP_SUM", yytext()); }
"-" { return token("OP_SUB", yytext()); }
"*"  { return token("OP_MUL", yytext()); }
"/"  { return token("OP_DIV", yytext()); }
"=="   { return token("OP_IGUAL", yytext()); }
"!="  { return token("OP_NEQ", yytext()); }
">"   { return token("OP_MA", yytext()); }
"<"   { return token("OP_ME", yytext()); }
">="  { return token("OP_MAI", yytext()); }
"<="  { return token("OP_MEI", yytext()); }
"and" { return token("OP_AND", yytext()); }
"or" { return token("OP_OR", yytext()); }
"not" { return token("OP_NOT", yytext()); }
"[*" { throw new Error("Error: Se encontró '[*' sin haber abierto un (*."); }
"{*" { throw new Error("Error: Se encontró '{*' sin haber abierto un [*."); }
"*)" { throw new Error("Error: Se encontró '*)' sin haber abierto un comentario."); }
"*}" { throw new Error("Error: Se encontró '*}' sin haber abierto un comentario."); }
"*]" { throw new Error("Error: Se encontró '*]' sin haber abierto un comentario."); }


/* Símbolos */
{PAREN_ABIERTO}     { return token("PAREN_ABIERTO", yytext()); }
{PAREN_CERRADO}    { return token("PAREN_CERRADO", yytext()); }
{CORCHETE_ABIERTO}   { return token("CORCHETE_ABIERTO", yytext()); }
{CORCHETE_CERRADO}  { return token("CORCHETE_CERRADO", yytext()); }
{LLAVE_ABIERTO}     { return token("LLAVE_ABIERTO", yytext()); }
{LLAVE_CERRADO}    { return token("LLAVE_CERRADO", yytext()); }
{COMMA}          { return token("COMMA", yytext()); }
{ASIGNACION}     { return token("ASIGNACION", yytext()); }

/* Literales */
{LIT_INTEGER} { return token("LIT_INTEGER", Integer.parseInt(yytext())); }
{LIT_FLOAT}   { return token("LIT_FLOAT", Float.parseFloat(yytext())); }
"true"   { return token("LIT_BOOLEAN", yytext() ); }
"false"  { return token("LIT_BOOLEAN", yytext()); }
{LIT_ARRAY} { return token("LIT_ARRAY", yytext()); }

/* Identificadores */
{IDENTIFICADOR} { return token("IDENTIFICADOR", yytext()); }

/*Comentarios de una linea*/
{COMENTARIO_LINEA} { /* Ignorar */ }

/* espacios en blanco */
{WhiteSpace}        { /* ignore */ }
```

```
\"              { string.setLength(0);
                  string_yyline = this.yyline;
                  string_yycolumn = this.yycolumn;
                  yybegin(CADENA);
                 }

 /* Comentarios multilínea */
"(*" { comentarioPila.push('('); nivelComentario = 1; yybegin(COMENTARIO); }
}
/* Manejo de comentarios */
<COMENTARIO> {
/* Apertura de comentarios anidados */
   "(*" { comentarioPila.push('('); nivelComentario = 1; }
   "[*" { comentarioPila.push('['); nivelComentario = 2;}
   "{*" { comentarioPila.push('{'); nivelComentario = 3;}

/* Cierre de comentarios con validación de jerarquía */
   "*)" {
      if (!comentarioPila.isEmpty() && comentarioPila.peek() == '(' && nivelComentario == 1){
         comentarioPila.pop();
         nivelComentario = 3;
         if (comentarioPila.isEmpty()) yybegin(YYINITIAL);
      } else {
         throw new Error("Error de balanceo: Se esperaba cierre para " +
comentarioPila.peek());
      }
   }
   "*]" {
      if (!comentarioPila.isEmpty() && comentarioPila.peek() == '[' && nivelComentario == 2){
         comentarioPila.pop();
         nivelComentario = 1;
         if (comentarioPila.isEmpty()) yybegin(YYINITIAL);
      } else {
         throw new Error("Error de balanceo: Se esperaba cierre para " +
comentarioPila.peek());
      }
   }
   "*}" {
      if (!comentarioPila.isEmpty() && comentarioPila.peek() == '{' && nivelComentario == 3){
         comentarioPila.pop();
         nivelComentario = 2;
         if (comentarioPila.isEmpty()) yybegin(YYINITIAL);
      } else {
         throw new Error("Error de balanceo: Se esperaba cierre para " +
comentarioPila.peek());
      }
   }
   /* Fin de archivo */
```

```
   <<EOF>> {
      if (!comentarioPila.isEmpty()) {
         throw new Error("Error: Fin de archivo con comentario no cerrado. Se esperaba
cierre para " + comentarioPila.peek());
      }
   }
[^] { /* Ignorar otros caracteres dentro del comentario */ }

}

<CADENA> {
  \"                { yybegin(YYINITIAL);
                      return token("LIT_STRING",
                                   string_yyline, string_yycolumn,
                                   string.toString());
                    }
  \\\"              { string.append("\"") ; }

  /* Fin de archivo */
  <<EOF>>           { throw new Error("Fin de archivo dentro de la cadena: \n" +
                                      string.toString()); }

  /* Cualquier otro carácter */
  [^]               { string.append(yytext()); }
}

/* fallback de errores */
[^]                 { throw new Error("Carácter inválido <"+yytext()+">"); }
```