```
package analizadores;
import java_cup.runtime.Symbol;
import excepciones.Errores;
import analizadores.TokenLista;
import analizadores.Tokens;

%%

%class Scanner
%public
%line
%column
%cup
%ignorecase

%{
   private Symbol symbol(int type, Object value, Tokens tokenTipo) {
      TokenLista.guardar(tokenTipo, yytext(), yyline+1, yycolumn+1);
      return new Symbol(type, yyline+1, yycolumn+1, value);
   }

   private Symbol symbol(int type, Tokens tokenTipo) {
      TokenLista.guardar(tokenTipo, yytext(), yyline+1, yycolumn+1);
      return new Symbol(type, yyline+1, yycolumn+1, yytext());
   }
%}

ENTERO  = [0-9]+
DECIMAL = [0-9]+\.[0-9]+
ID      = [a-zA-Z][a-zA-Z0-9_]*
CADENA  = \"([^\\\"]|\\.)*\"
CHAR    = \'([^\\\']|\\.)\'
ESPACIO = [ \t\r\n\f]+
COMENTARIO_LINEA = "//".*
COMENTARIO_BLOQUE = "/*" [^]*? "*/"

%%

<YYINITIAL> {
   "start_with" { return symbol(sym.START_WITH, Tokens.START_WITH); }
   "void"     { return symbol(sym.VOID, Tokens.VOID); }
   "var"      { return symbol(sym.VAR, Tokens.VAR); }
   "int"      { return symbol(sym.INT, Tokens.INT); }
   "double"   { return symbol(sym.DOUBLE, Tokens.DOUBLE); }
   "bool"     { return symbol(sym.BOOL, Tokens.BOOL); }
   "char"     { return symbol(sym.CHAR, Tokens.CHAR); }
   "string"   { return symbol(sym.STRING, Tokens.STRING); }

   "if"       { return symbol(sym.IF, Tokens.IF); }
   "else"     { return symbol(sym.ELSE, Tokens.ELSE); }
   "switch"   { return symbol(sym.SWITCH, Tokens.SWITCH); }
   "case"     { return symbol(sym.CASE, Tokens.CASE); }
   "default"  { return symbol(sym.DEFAULT, Tokens.DEFAULT); }
   "while"    { return symbol(sym.WHILE, Tokens.WHILE); }
   "do"       { return symbol(sym.DO, Tokens.DO); }
   "for"      { return symbol(sym.FOR, Tokens.FOR); }
   "break"    { return symbol(sym.BREAK, Tokens.BREAK); }
```

```
"continue"  { return symbol(sym.CONTINUE, Tokens.CONTINUE); }
"return"    { return symbol(sym.RETURN, Tokens.RETURN); }

"print"     { return symbol(sym.PRINT, Tokens.PRINT); }
"println"   { return symbol(sym.PRINTLN, Tokens.PRINTLN); }
"length"    { return symbol(sym.LENGTH, Tokens.LENGTH); }
"round"     { return symbol(sym.ROUND, Tokens.ROUND); }
"toString"  { return symbol(sym.TOSTRING, Tokens.TOSTRING); }
"find"      { return symbol(sym.FIND, Tokens.FIND); }

"new"       { return symbol(sym.NEW, Tokens.NEW); }
"list"      { return symbol(sym.LIST, Tokens.LIST); }
"append"    { return symbol(sym.APPEND, Tokens.APPEND); }
"remove"    { return symbol(sym.REMOVE, Tokens.REMOVE); }

"true"      { return symbol(sym.TRUE, "true", Tokens.TRUE); }
"false"     { return symbol(sym.FALSE, "false", Tokens.FALSE); }

"++"        { return symbol(sym.MASMAS, Tokens.INCREMENTO); }
"--"        { return symbol(sym.MENOSMENOS, Tokens.DECREMENTO); }
"+"         { return symbol(sym.MAS, Tokens.SUMA); }
"-"         { return symbol(sym.MENOS, Tokens.RESTA); }
"**"        { return symbol(sym.POTENCIA, Tokens.POT); }
"*"         { return symbol(sym.POR, Tokens.MULT); }
"/"         { return symbol(sym.DIV, Tokens.DIV); }
"%"         { return symbol(sym.MOD, Tokens.MOD); }

"("         { return symbol(sym.PAR_A, Tokens.PAR_A); }
")"         { return symbol(sym.PAR_C, Tokens.PAR_C); }
"{"         { return symbol(sym.LLAVE_A, Tokens.LLAVE_A); }
"}"         { return symbol(sym.LLAVE_C, Tokens.LLAVE_C); }
"["         { return symbol(sym.COR_IZQ, Tokens.COR_IZQ); }
"]"         { return symbol(sym.COR_DER, Tokens.COR_DER); }

";"         { return symbol(sym.PTCOMA, Tokens.PUNTO_COMA); }
":"         { return symbol(sym.DOSPTOS, Tokens.DOS_PUNTOS); }
"="         { return symbol(sym.IGUAL, Tokens.ASIGN); }
","         { return symbol(sym.COMA, Tokens.COMA); }
"."         { return symbol(sym.PUNTO, Tokens.PUNTO); }

"=="        { return symbol(sym.IGUALIGUAL, Tokens.IGUAL); }
"!="        { return symbol(sym.DIFERENTE, Tokens.DIF); }
"<="        { return symbol(sym.MENORIGUAL, Tokens.MENORIGUAL); }
">="        { return symbol(sym.MAYORIGUAL, Tokens.MAYORIGUAL); }
"<"         { return symbol(sym.MENOR, Tokens.MENOR); }
">"         { return symbol(sym.MAYOR, Tokens.MAYOR); }
"||"        { return symbol(sym.OR, Tokens.OR); }
"&&"        { return symbol(sym.AND, Tokens.AND); }
"!"         { return symbol(sym.NOT, Tokens.NOT); }
"^"         { return symbol(sym.XOR, Tokens.XOR); }

{ENTERO}    { return symbol(sym.ENTERO, yytext(), Tokens.ENTERO); }
{DECIMAL}   { return symbol(sym.DECIMAL, yytext(), Tokens.DECIMAL); }
{ID}        { return symbol(sym.ID, yytext(), Tokens.ID); }

{CADENA}    {
   String val = yytext();
```

```
        String contenido = val.substring(1, val.length()-1);
        // Procesar secuencias de escape
        contenido = contenido.replace("\\n", "\n")
                             .replace("\\t", "\t")
                             .replace("\\r", "\r")
                             .replace("\\'", "'")
                             .replace("\\\"", "\"")
                             .replace("\\\\", "\\");
        TokenLista.guardar(Tokens.LIT_STRING, val, yyline+1, yycolumn+1);
        return new Symbol(sym.LIT_STRING, yyline+1, yycolumn+1, contenido);
    }

    {CHAR}     {
        String val = yytext();
        String contenido = val.substring(1, val.length()-1);
        // Procesar secuencias de escape
        contenido = contenido.replace("\\n", "\n")
                             .replace("\\t", "\t")
                             .replace("\\'", "'")
                             .replace("\\\\", "\\");
        TokenLista.guardar(Tokens.LIT_CHAR, val, yyline+1, yycolumn+1);
        return new Symbol(sym.LIT_CHAR, yyline+1, yycolumn+1, contenido);
    }

    {ESPACIO}          { /* Ignorar */ }
    {COMENTARIO_LINEA}  { /* Ignorar */ }
    {COMENTARIO_BLOQUE} { /* Ignorar */ }
}

[^] {
    Errores.agregar("Lexico", "Caracter no reconocido: " + yytext(), yyline+1, yycolumn+1);
    TokenLista.guardar(Tokens.ERROR, yytext(), yyline+1, yycolumn+1);
    System.err.println("Error léxico: " + yytext() + " en línea " + (yyline+1));
}
```