

```
1 import java_cup.runtime.*;
2 import java.util.ArrayList;
3
4 terminal ALL, CLL, AP, CP, AC, CC;
5 terminal COMA, PYC, ASIG, MAS, MENOS, POR, DIV;
6 terminal INVERSA, TRANSPUESTA, ADJUNTA, PRINT;
7 terminal String IDENT;
8 terminal Double NUMERO;
9
10 non terminal linea, lineaExp;
11 non terminal double[][] comando;
12 non terminal double[][] matriz, exp;
13 non terminal ArrayList<ArrayList<Double>> procesarMatriz;
14 non terminal ArrayList<Double> fila;
15
16 precedence left MAS, MENOS;
17 precedence left POR, DIV;
18
19 lineaExp ::= lineaExp linea | linea;
20
21 linea ::= exp:v PYC | PYC;
22
23 exp ::= PRINT AP exp:v {:Matrices.print(v);;} CP
24       | INVERSA AP exp:v {:if(Matrices.filas(v)==Matrices.columnas(v)){
25                             RESULT = Matrices.inversa(v);
26                         }else{
27                             System.out.println(Matrices.ERROR_INVERSA);
28                             System.exit(-1);
29                         };} CP
30       | TRANSPUESTA AP exp:v {:RESULT = Matrices.transpuesta(v);;} CP
31       | ADJUNTA AP exp:v {:if(Matrices.filas(v)==Matrices.columnas(v)){
32                             RESULT = Matrices.adjunta(v);
33                         }else{
34                             System.out.println(Matrices.ERROR_ADJUNTA);
35                             System.exit(-1);
36                         };} CP
37       | exp:v1 MAS exp:v2 {:if((Matrices.columnas(v1)==Matrices.columnas(v2))&&
38                             (Matrices.filas(v1)==Matrices.filas(v2))){
39                             RESULT = Matrices.suma(v1,v2);
40                         }else{
41                             System.out.println(Matrices.ERROR_SUMA);
42                             System.exit(-1);
43                         };}
44       | exp:v1 MENOS exp:v2
45       | exp:v1 POR exp:v2 {:if(Matrices.columnas(v1)==Matrices.filas(v2)){
46                             RESULT = Matrices.producto(v1,v2);
47                         }else{
48                             System.out.println(Matrices.ERROR_PROD);
49                             System.exit(-1);
50                         };;}
51       | exp:v1 DIV exp:v2
52       | IDENT:a ASIG exp:v {:TablaSimbolos.insertar(a, v);;}
53       | IDENT:a {:double[][] aux = TablaSimbolos.buscar(a);
54                   if(aux==null){
55                       System.out.println(TablaSimbolos.ERROR_NOEXISTE);
56                       System.exit(-1);
57                   }
58                   RESULT = aux;;}
59       | AP exp:v CP {:RESULT=v;;
```

```
59 | matriz:v {:RESULT = v;:};
60
61
62 matriz ::= AC procesarMatriz:v CC {:try{
63     double[][] a = Matrices.toArray(v);
64     RESULT=a;
65     }catch (IndexOutOfBoundsException e){
66
67         System.out.println(Matrices.ERROR_FILAS);
68         System.exit(-1);
69     }
70     :}
71     |ALL procesarMatriz:v CLL {:try{
72         double[][] a = Matrices.toArray(v);
73         RESULT=a;
74         }catch (IndexOutOfBoundsException e){
75
76         System.out.println(Matrices.ERROR_FILAS);
77         System.exit(-1);
78     }
79     :};
80 procesarMatriz ::= fila:v PYC procesarMatriz:m {:ArrayList<ArrayList<Double>> aux
= new ArrayList<>(); aux.add(v);aux.addAll(m); RESULT = aux;:}
81     | fila:v {:ArrayList<ArrayList<Double>> aux = new ArrayList<>();
aux.add(v); RESULT = aux;:}
82     | ALL fila:v CLL COMA procesarMatriz:m
{:ArrayList<ArrayList<Double>> aux = new ArrayList<>(); aux.add(v);aux.addAll(m);
RESULT = aux;:}
83     | ALL fila:v CLL {:ArrayList<ArrayList<Double>> aux = new
ArrayList<>(); aux.add(v); RESULT = aux;:} ;
84
85 fila ::= NUMERO:a COMA fila:v {:ArrayList<Double> aux = new ArrayList<>();
aux.add(a); aux.addAll(v); RESULT=aux;:}
86     | NUMERO:a {:ArrayList<Double> aux = new ArrayList<>(); aux.add(a); RESULT
= aux;:};
87
```