```
/* Terminals (tokens returned by the scanner). */
terminal Token                  PR_VARS, CORCHETE_A, CORCHETE_C, LLAVE_A, LLAVE_C, PARENTESIS_A,
                                PARENTESIS_C, PR_STRING, PR_INT, PR_BOOLEAN;
terminal Token                  COMA, ID, ASIGNACION, S_SUMA, S_RESTA, S_DIV, S_MUL, PR_OR, PR_AND,
                                COMPARADOR, ENTERO, BOOLEAN;
terminal Token                  PR_INSTRUCCIONES, PR_IF, PR_ELSE, PR_WHILE, PR_PINTAR, PR_RANGE, FINAL,
                                CADENA;

non terminal                    S, dcls, dcl, varListString, varListInt, varListBoolean, instrLienzos, instrLienzo;
non terminal String             asigString;
non terminal Integer            asigInt;
non terminal Boolean            varBoolean;
non terminal Nodo               posicion, expresion;
non terminal List<Instruccion>      instrs, restoIf;
non terminal Instruccion        instr;

/* precedence */

precedence left     PR_OR, PR_AND;
precedence left     COMPARADOR;
precedence left     S_SUMA, S_RESTA;
precedence left     S_MUL, S_DIV;

/* The grammar */

start with S;

S ::=                           PR_VARS CORCHETE_A dcls CORCHETE_C instrLienzos
                               |instrLienzos
                               ;

dcls ::=                        dcls dcl
                               |dcl
                               |error FINAL dcl
                               |error CORCHETE_C instrLienzos
                               ;

dcl ::=                         PR_STRING varListString FINAL
                               |PR_INT varListInt FINAL
                               |PR_BOOLEAN varListBoolean FINAL
                               ;

varListString ::=                   varListString COMA ID:idVariable asigString:nodo {:
                                   :}
                               |ID:idVariable asigString:nodo {:
                                   :}
                               ;

asigString ::=                      ASIGNACION expresion:derecha {:
                                        RESULT =  derecha.evaluar(variables).toString();
                                   :}
                               |
                               ;

varListInt ::=                      varListInt COMA ID:idVariable asigInt:nodo {:
                                   :}
                               |ID:idVariable asigInt:nodo {:
                                   :}
```

```
                        ;

asigInt ::=             ASIGNACION expresion:derecha {:
                                RESULT = Integer.parseInt(derecha.evaluar(variables).toString());
                        :}
                        |
                        ;

varListBoolean ::=          varListBoolean COMA ID:idVariable varBoolean:nodo {:
                        :}
                        |ID:idVariable varBoolean:nodo {:
                        :}
                        ;

varBoolean ::=          ASIGNACION expresion:derecha {:
                                RESULT = (boolean) derecha.evaluar(variables);
                        :}
                        |
                        ;

instrLienzos ::=            instrLienzos instrLienzo
                        |instrLienzo
                        ;

instrLienzo ::=         PR_INSTRUCCIONES PARENTESIS_A ID:id PARENTESIS_C CORCHETE_A
                        instrs:instrucciones CORCHETE_C {:
                        :}
                        |error CORCHETE_A instrs
                        |error CORCHETE_C instrLienzo
                        ;

instrs ::=              instrs:instrucciones instr:instruccion {:
                                RESULT = instrucciones;
                        :}
                        |instr:instruccion {:
                                RESULT = instrucciones;
                        :}
                        |error FINAL instr
                        ;

instr ::=               ID:id ASIGNACION expresion:valorDerecha FINAL {:
                        :}
                        |PR_PINTAR:valor PARENTESIS_A expresion:idColor COMA expresion:idImagen COMA
                        posicion:posX COMA posicion:posY PARENTESIS_C FINAL {:
                                RESULT = retorno;
                        :}
                        |PR_WHILE:valor PARENTESIS_A expresion:condiciones PARENTESIS_C LLAVE_A
                        instrs:instrucciones LLAVE_C {:
                        :}
                        |PR_IF:valor PARENTESIS_A expresion:condiciones PARENTESIS_C LLAVE_A
                        instrs:instruccionesSi LLAVE_C restoIf:instruccionesSiNo {:
                                RESULT = retorno;
                        :}
                        ;

restoIf::=              PR_ELSE LLAVE_A instrs:instrucciones LLAVE_C {:
                                RESULT = instrucciones;
                        :}
```

```
                        |   {:
                                RESULT = instrucciones;
                            }
                          :}
                        ;

posicion ::=                expresion:nodo {:
                                    RESULT = nodo;
                          :}
                        |expresion:izq PR_RANGE:operador expresion:der {:
                                    RESULT = nodo;
                          :}
                        ;

expresion ::=               expresion:izq S_SUMA:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq S_RESTA:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq S_MUL:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq S_DIV:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq PR_AND:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq PR_OR:operador expresion:der {:
                                RESULT = nodo;
                          :}
                        |expresion:izq COMPARADOR:operador expresion:der {:
                                    RESULT = nodo;
                          :}
                        |ID:valor   {:
                                RESULT = nodo;
                          :}
                        |BOOLEAN:valor {:
                                RESULT = nodo;
                          :}
                        |CADENA:valor{:
                                RESULT = nodo;
                          :}
                        |ENTERO:valor {:
                                RESULT = nodo;
                          :}
                        ;
```