Manual Tecnico:
Gramatica de movimiento:
Lexico:

```
package com.example.proyecto1_23.Parser;

    import java_cup.runtime.Symbol;
    import java.util.*;

%%

%class lexer

%public
%line
%column
%cup
%cupdebug


//espacios
ESPACIOS = [ \r\t\b\n\f]

//elemenos fundamentales
COMILLA = "\""
COMA = ","
PUNTO = "."
PUNTOCOMA = ";"
LITERALES = [a-zA-Z]
NUMEROS = [0-9]

/*cadenas*/
ENTEROS = {NUMEROS}+
DECIMAL = ({ENTEROS}+{PUNTO}{ENTEROS}+)
CADENA = {LITERALES}+({LITERALES}|{ENTEROS})*
CADENAS = {COMILLA}({ESPACIOS}|{CADENA}|{DECIMAL}|{ENTEROS})*{COMILLA}


/*OPeradores aritmeticos*/
SUMA = "+"
RESTA = "-"
MULTIPLICACION = "*"
DIVISION = "/"

/*Elementos de agrupacion*/
PARENTESISABIERTO = "("
PARENTESISCERRADO = ")"
LLAVESABIERTO = "{"
LLAVESCERRADO = "}"
CORCHETESABIERTO = "["
```

```
CORCHETESCERRADO = "]"

/*comentarios*/
COMENTARIOS = "#".*


/*Definicion de movimientos*/

UP = "up"
DOWN = "down"
LEFT = "left"
RIGHT = "right"
FLOOR = "FLOOR"
CEIL = "CEIL"
PUSH = "push"


%{
  //  private ReporteErrores tabla = new ReporteErrores();

   // public void setTabla(ReporteErrores tabla){
  //      this.tabla = tabla;
  //  }

%}


%%
  {COMA}                  {return new Symbol(sym.COMA,yyline+1,yycolumn+1, yytext());}
  {PUNTO}                 {return new Symbol(sym.PUNTO, yyline+1,yycolumn+1, yytext());}
  {PUNTOCOMA}             {return new Symbol(sym.PUNTOCOMA,yyline+1,yycolumn+1,
yytext());}

  /*valores artimeticos*/
  {SUMA}                          {return new Symbol(sym.SUMA, yyline+1, yycolumn+1,
yytext());}
  {RESTA}                 {return new Symbol(sym.RESTA, yyline+1, yycolumn+1,
yytext());}
  {MULTIPLICACION}        {return new Symbol(sym.MULTIPLICACION, yyline+1,
yycolumn+1, yytext());}
  {DIVISION}                      {return new Symbol(sym.DIVISION, yyline+1,
yycolumn+1, yytext());}
  {PARENTESISABIERTO}     {return new Symbol(sym.PARENTESISABIERTO, yyline+1,
yycolumn+1, yytext());}
  {PARENTESISCERRADO}     {return new Symbol(sym.PARENTESISCERRADO, yyline+1,
yycolumn+1, yytext());}
  {LLAVESABIERTO}         {return new Symbol(sym.LLAVESABIERTO, yyline+1,
yycolumn+1, yytext());}
```

```
  {LLAVESCERRADO}              {return new Symbol(sym.LLAVESCERRADO, yyline+1,
yycolumn+1, yytext());}
  {CORCHETESABIERTO}           {return new Symbol(sym.CORCHETESABIERTO, yyline+1,
yycolumn+1, yytext());}
  {CORCHETESCERRADO}                {return new Symbol(sym.CORCHETESCERRADO,
yyline+1, yycolumn+1, yytext());}

  {ENTEROS}                 {return new Symbol(sym.ENTEROS, yyline+1,yycolumn+1, new
Double(yytext()));}
  {DECIMAL}                 {return new Symbol(sym.DECIMAL, yyline+1,yycolumn+1, new
Double(yytext()));}
  {CADENAS}                 {return new Symbol(sym.CADENAS ,yyline+1,yycolumn+1,yytext());}



  {UP}            {return new Symbol(sym.UP, yyline+1, yycolumn+1, yytext());}
  {DOWN}            {return new Symbol(sym.DOWN, yyline+1, yycolumn+1, yytext());}
  {LEFT}          {return new Symbol(sym.LEFT, yyline+1, yycolumn+1, yytext());}
  {RIGHT}           {return new Symbol(sym.RIGHT, yyline+1, yycolumn+1, yytext());}
  {FLOOR}           {return new Symbol(sym.FLOOR, yyline+1, yycolumn+1, yytext());}
  {CEIL}          {return new Symbol(sym.CEIL, yyline+1, yycolumn+1, yytext());}
  {PUSH}          {return new Symbol(sym.PUSH, yyline+1, yycolumn+1, yytext());}


  {ESPACIOS}                {/*NADA*/}
  {COMENTARIOS} {/*NADA*/}



   0.(0)+.{DECIMAL}  {/*tabla.agregarError(yytext(), yyline, yycolumn, "Lexico");*/
                return new Symbol(sym.ERROR, yyline+1, yycolumn+1, yytext());}
   0.(0)+.{ENTEROS} {/*tabla.agregarError(yytext(), yyline, yycolumn, "Lexico");*/
               return new Symbol(sym.ERROR, yyline+1, yycolumn+1, yytext());}
  /*SI NO VIENE NADA, ENTONCES GENERA UN ERROR*/
   [^]     {/*tabla.agregarError(yytext(), yyline+1, yycolumn+1, "ERROR LEXICO");*/
         return new Symbol(sym.ERROR, yyline+1, yycolumn+1, yytext());}
```

Sintactico:

```
package com.example.proyecto1_23.Parser;
import com.example.proyecto1_23.Ocurrencias.movimientoObjetos;
import com.example.proyecto1_23.Vista.movimiento;
import java_cup.runtime.Symbol;
import ReporteErrores.*;
import java.util.ArrayList;
parser code
{:

    public parser(lexer lexerForma) {
        super(lexerForma);
```

```
    }
    public void syntax_error(Symbol s) {
            tabla.agregarError(String.valueOf(s.value), s.left, s.right, "ERRORLEXICO");
        }
    //cantidad de moviminetos
    private movimientoObjetos movimientosIndividuales;
    private int cantidadMovimientos;
    private ArrayList<movimientoObjetos> movimientosLista;


    private ArrayList<String> tipoMovimientoLista= new ArrayList<>();
    private ArrayList<Double> cantidadMovimientoLista= new ArrayList<>();
    private ArrayList<Integer> columnasLista= new ArrayList<>();
    private ArrayList<Integer> filasLista= new ArrayList<>();


    public ArrayList<String> getListaMovimientos(){
        return this.tipoMovimientoLista;
    }
    public ArrayList<Double> getListaCantidadMovimientos(){
            return this.cantidadMovimientoLista;
    }

    public ArrayList<Integer> getcolumnasLista(){
                return this.columnasLista;
    }
    public ArrayList<Integer> getfilasLista(){
                    return this.filasLista;
    }



    public ArrayList<movimientoObjetos> getListMoveGame(){
        return this.movimientosLista;
    }



    //private ReporteErrores tabla = new ReporteErrores();
    private movimiento graficas;
    public void mostrarValoresMovimiento(String tipo) {

    }
   // public void setTabla(ReporteErrores tabla){
  //          this.tabla = tabla;
  //   }
```

```
:}


//terminales:
terminal PUNTOCOMA, ESPACIOS, COMENTARIOS, PARENTESISABIERTO,
PARENTESISCERRADO, LLAVESABIERTO, LLAVESCERRADO, CORCHETESABIERTO,
CORCHETESCERRADO, PUNTO, COMA, SUMA, RESTA, MULTIPLICACION, DIVISION,
ERROR, UP, DOWN, LEFT, RIGHT, CEIL, FLOOR, PUSH;
terminal Double  ENTEROS,DECIMAL;
terminal String CADENAS;

// seccion de no termimales
non terminal s, inicioMovimiento, tipoMovimiento, estructuraMovmiento;
non terminal Double expresiones;
non terminal Double expresionIndividual;
non terminal Double funcionesAproximacion;
// para los operadores aritmeticos se tiene una precedencia:

precedence left  SUMA, RESTA;
precedence left  MULTIPLICACION, DIVISION;
precedence left  CEIL, FLOOR;


//estado inicial del analizador sintactico
start with s;
//inicio estados

s::=  inicioMovimiento s
   | inicioMovimiento
;

//movimiento
inicioMovimiento ::= tipoMovimiento estructuraMovmiento PUNTOCOMA

;

// MOVIMIENTOS
tipoMovimiento::= UP:e
            {:
            tipoMovimientoLista.add(String.valueOf(e));
               columnasLista.add(eright);
               filasLista.add(eleft);
            System.out.println(e+String.valueOf(eleft)+String.valueOf(eright)); RESULT = e;  :}
              | DOWN:e {: tipoMovimientoLista.add(String.valueOf(e) );
                        columnasLista.add(eright);
                        filasLista.add(eleft);

              System.out.println(e+String.valueOf(eleft)+String.valueOf(eright)); RESULT = e;  :}
```

```
            | LEFT:e {:  tipoMovimientoLista.add(String.valueOf(e));
                        columnasLista.add(eright);
                        filasLista.add(eleft);
                        System.out.println(e+String.valueOf(eleft)+String.valueOf(eright)); RESULT =
e;  :}
            | RIGHT:e {: tipoMovimientoLista.add(String.valueOf(e));
                    columnasLista.add(eright);
                    filasLista.add(eleft);
                    System.out.println(e+String.valueOf(eleft)+String.valueOf(eright));RESULT = e;  :}
;

estructuraMovmiento::= PARENTESISABIERTO expresiones:e1 PARENTESISCERRADO
                    /*INGRESO DE VALORES DIRECTOS A LA FUNCION*/
                    {:  cantidadMovimientoLista.add(Double.valueOf(e1)); System.out.println(e1+"  ");
RESULT = e1;  :}
                |

;
//------------------------------------------------------------------------ ELEMENTOS BASICOS

expresionIndividual::=
      ENTEROS:n {:RESULT = n;:}
   | DECIMAL:d {:RESULT = d;:}
;

expresiones::=
   expresionIndividual:n {: RESULT = n; :}
      | expresiones:e1 SUMA:o expresiones:e2
                {: RESULT = e1 + e2;:}
      |expresiones:e1 RESTA:o expresiones:e2
                {: RESULT = e1 - e2;:}
      | expresiones:e1 MULTIPLICACION:o expresiones:e2
                {: RESULT = e1 * e2;:}
      |  expresiones:e1 DIVISION:o expresiones:e2
                {: RESULT = e1 / e2;:}
      | RESTA expresiones:e1
            {:RESULT = Double.valueOf(-e1);:}
      | PARENTESISABIERTO expresiones:e PARENTESISCERRADO
                {:RESULT = e;:}
       | funcionesAproximacion:funciones
       {:RESULT = funciones;:}
;
funcionesAproximacion::=
            FLOOR PARENTESISABIERTO expresiones:e PARENTESISCERRADO
              {:RESULT = Math.floor(e);:}
           | CEIL PARENTESISABIERTO expresiones:e PARENTESISCERRADO
              {:RESULT = Math.ceil(e);:}
         ;
```