Nombre: **MoveTheDot**

Carrera: **Lic. en Sistemas Informáticos**
Materia: **Diseño de Compiladores**
Cátedra: **Ing. Pablo Pandolfo**
Año de Curso: **2020** Turno: **Noche**
Entrega: Parcial
Fecha requerida: **31/08/2020** Fecha entregada: **31/08/2020**

## Grupo

| ID/Matrícula | APELLIDO, Nombres | Correo Electrónico |
|---|---|---|
| 501 - 11219 | Candenas, Nereo | nereo.candenas@comunidad.ub.edu.ar |
| 501 - 11220 | Jinkus, Iván | ivan.jinkus@comunidad.ub.edu.ar |
| 501 - 11221 | Martin, Mariano | mariano.martin@comunidad.ub.edu.ar |
| 501 - 11222 | Ferreira, Gabriel | gabriel.ferreira@comunidad.ub.edu.ar |

## Grilla de calificación

| Concepto | Funcionalidad | Contenidos | Resolución | Formato | Ampliaciones |
|---|---|---|---|---|---|
| **Sobresaliente (10)** | | | | | |
| **Distinguido (9-8)** | | | | | |
| **Bueno (7-6)** | | | | | |
| **Aprobado (5-4)** | | | | | |
| **Insuficiente (3-2-1)** | | | | | |
| **Reprobado (0)** | | | | | |
| **NOTA** | | | | | |

| Comentario adicional del Profesor: |
|---|
| |

## Contenido

# Desarrollo

## Descripción del Lenguaje

Proponemos desarrollar un lenguaje de programación básico que llamaremos **MoveTheDot**, y lo basaremos en el movimiento del juego de la viborita.
Este lenguaje nos dará la oportunidad de crear un conjunto de instrucciones para mover un punto en un tablero de coordenadas, con la opción de seleccionar su punto de inicio hasta la posición deseada o final, según el conjunto de instrucciones ingresado.

## Objetivo:

Desarrollar un lenguaje básico con un conjunto limitado de instrucciones y su compilador utilizando las herramientas provistas por la cátedra.

**Con el Lenguaje se podrá realizar las siguientes acciones:**

- ✓ Dimensionar un tablero predefinido.
- ✓ Definir el punto de inicio para el punto, o dejarlo por defecto en (0,0)
- ✓ Indicar siguiente movimiento

## Acciones disponibles:

El Lenguaje contará con las siguientes instrucciones de movimiento del punto:

***ubicar(num,num)*** ubica al dot en las coordenadas definidas en los valores indicados.

***mov_lat(+/-num)***: realiza un desplazamiento en sentido horizontal del punto según el valor ingresado.
hacia la derecha si el valor es positivo,
hacia la izquierda si el valor es negativo,
no se mueve si es cero

***mov_ver(+/-num)***: realiza un desplazamiento en sentido vertical del punto según el valor ingresado.
hacia la arriba si el valor es positivo,
hacia la abajo si el valor es negativo,
no se mueve si es cero

## Estructuras de control:

**inicio**, **fin**: para delimitar las estructuras de control
**repetir**(num) : repite el bloque la cantidad de veces del valor num

## Tipo de datos:

num=enteros

## Errores en Ejecución:

- Si el valor del movimiento no es un número entero.
- Si los movimientos superan la capacidad de la grilla.
- Si no se registra Inicio y Fin
- Si instrucciones fuera de orden

## Repositorio del código fuente en GIT

https://github.com/nereocandenas/CompiMTD

## Archivos de definición  jflex y CUP

### MoveTheDot.jflex

```
package ar.edu.ub.dc.mtd;
import java_cup.runtime.Symbol;
%%
%public
%class Scanner
%standalone
%cup
%%
"i"|"I"|"inicio"|"Inicio"|"INICIO" {return new Symbol(sym.INICIO);}
"f"|"F"|"fin"|"Fin"|"FIN" {return new Symbol(sym.FIN);}
"r"|"R"|"repetir"|"Repetir"|"REPETIR" {return new Symbol(sym.REPETIR);}
"u"|"U"|"ubicar"|"Ubicar"|"UBICAR" {return new Symbol(sym.UBICAR);}
"l"|"L"|"mov_lat"|"Mov_Lat"|"MOV_LAT" {return new Symbol(sym.LATERAL);}
"v"|"V"|"mov_ver"|"Mov_Ver"|"MOV_VER" {return new Symbol(sym.VERTICAL);}
"(" {return new Symbol(sym.PAR_A);}
")" {return new Symbol(sym.PAR_C);}
"[" {return new Symbol(sym.COR_A);}
"]" {return new Symbol(sym.COR_C);}
"+" {return new Symbol(sym.SUMA);}
"-" {return new Symbol(sym.RESTA);}
"," {return new Symbol(sym.COMA);}
[:digit:]+ { return new Symbol(sym.ENTERO, new Integer(yytext())); }
[ \t\r\n]+ {;}
```

### MoveTheDot.jflex

```
package ar.edu.ub.dc.mtd;
import java_cup.runtime.*;
action code {: static int x = 0;
                       static int y = 0;

                       public static void locate_it(int xx, int yy) {
                         x = xx;
```

```
                            y = yy;
                          }

                          public static void move_it(int xx, int yy) {
                            x += xx;
                            y += yy;
                          }

                          public static void show_it() {
                                          System.out.println("Finaliza en x:" + x + ", y:" + y);
                                          System.out.println("Fin de ejecución.");
                          }
:}

/* Terminales */
terminal INICIO, FIN, UBICAR, VERTICAL, LATERAL, PAR_A, PAR_C, SUMA, RESTA, REPETIR,
COR_A, COR_C, COMA;

/* Terminales con atributo asociado */
terminal Integer ENTERO;

non terminal PROGRAMA, CUERPO, LINEAS, LINEA1;

start with PROGRAMA;

/* Programa vacio */
PROGRAMA ::= INICIO FIN
            {:
            System.out.println("Programa vacio");
            show_it();
            :};

/* Programa con contenido */
PROGRAMA ::= INICIO CUERPO FIN
                  {: show_it(); :};

PROGRAMA ::= INICIO LINEA1 CUERPO FIN
                  {: show_it(); :};

LINEA1 ::= UBICAR PAR_A ENTERO:n1 COMA ENTERO:n2 PAR_C
                  {:
                   System.out.println("Se ubica en x:"+n1.intValue()+", y:"+n2.intValue());
                   locate_it(n1.intValue(),n2.intValue());
                   :};
```

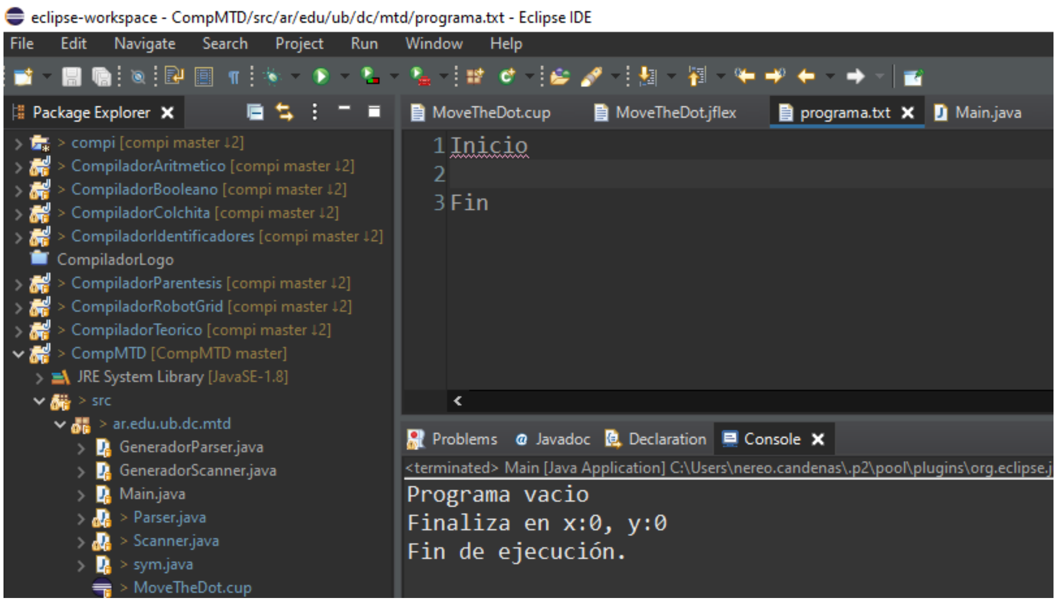
```
CUERPO ::= LINEAS
                {: :};

CUERPO ::= CUERPO LINEAS
                {: :};

LINEAS ::= VERTICAL PAR_A SUMA ENTERO:n1 PAR_C
                {:
                 System.out.println("Sube: "+n1.intValue());
                 move_it(0,n1.intValue());
                 :};

LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A VERTICAL PAR_A SUMA ENTERO:n2
PAR_C COR_C
                        {:
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Sube: "+n2.intValue());
                        move_it(0,n2.intValue());
                        }
                        :};

LINEAS ::= VERTICAL PAR_A ENTERO:n1 PAR_C
                {:
                 System.out.println("Sube: "+n1.intValue());
                 move_it(0,n1.intValue());
                 :};

LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A VERTICAL PAR_A ENTERO:n2 PAR_C
COR_C
                        {:
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Sube: "+n2.intValue());
                        move_it(0,n2.intValue());
                        }
                        :};

LINEAS ::= VERTICAL PAR_A RESTA ENTERO:n1 PAR_C
                {:
                System.out.println("Baja: "+n1.intValue());
                  move_it(0,-n1.intValue());
                  :};
```

```
LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A VERTICAL PAR_A RESTA ENTERO:n2
PAR_C COR_C
                        {:
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Baja: "+n2.intValue());
                        move_it(0,-n2.intValue());
                        }
                        :};

LINEAS ::= LATERAL PAR_A SUMA ENTERO:n1 PAR_C
     {:
      System.out.println("Derecha: "+n1.intValue());
      move_it(n1.intValue(),0);
      :};

LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A LATERAL PAR_A SUMA ENTERO:n2
PAR_C COR_C
                        {:
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Derecha: "+n2.intValue());
                        move_it(n2.intValue(),0);
                        }
                        :};

LINEAS ::= LATERAL PAR_A ENTERO:n1 PAR_C
     {:
      System.out.println("Derecha: "+n1.intValue());
      move_it(n1.intValue(),0);
      :};

LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A LATERAL PAR_A ENTERO:n2 PAR_C
COR_C
                        {:
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Derecha: "+n2.intValue());
                        move_it(n2.intValue(),0);
                        }
                        :};

LINEAS ::= LATERAL PAR_A RESTA ENTERO:n1 PAR_C
                {:
                 System.out.println("Izquierda: "+n1.intValue());
                 move_it(-n1.intValue(),0);
                 :};
```

```
LINEAS ::= REPETIR PAR_A ENTERO:n1 PAR_C COR_A LATERAL PAR_A RESTA ENTERO:n2
PAR_C COR_C
                    {:
                    for (int i = 0; i < n1; i++) {
                    System.out.println("Rep("+i+")-> Izquierda: "+n2.intValue());
                    move_it(-n2.intValue(),0);
                    }
                    :};
*****************************************************************************
```

## Capturas De Pantallas

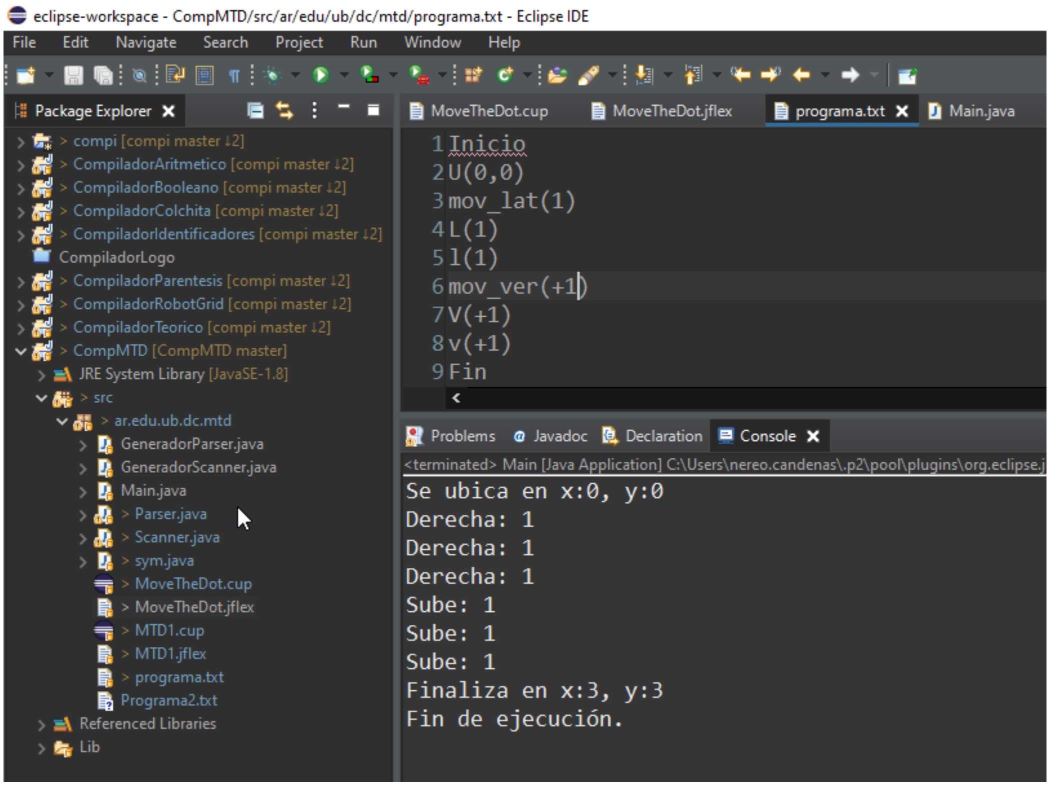
### Ejemplo 1: Programa Vacío


eclipse-workspace - CompMTD/src/ar/edu/ub/dc/mtd/programa.txt - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
MoveTheDot.cup
MoveTheDot.jflex
programa.txt
Main.java
compi [compi master ↓2]
CompiladorAritmetico [compi master ↓2]
CompiladorBooleano [compi master ↓2]
CompiladorColchita [compi master ↓2]
CompiladorIdentificadores [compi master ↓2]
CompiladorLogo
CompiladorParentesis [compi master ↓2]
CompiladorRobotGrid [compi master ↓2]
CompiladorTeorico [compi master ↓2]
CompMTD [CompMTD master]
JRE System Library [JavaSE-1.8]
src
ar.edu.ub.dc.mtd
GeneradorParser.java
GeneradorScanner.java
Main.java
Parser.java
Scanner.java
sym.java
MoveTheDot.cup
Problems
Javadoc
Declaration
Console

```
Inicio

Fin
```

```
<terminated> Main [Java Application] C:\Users\nereo.candenas\.p2\pool\plugins\org.eclipse.j
Programa vacio
Finaliza en x:0, y:0
Fin de ejecución.
```

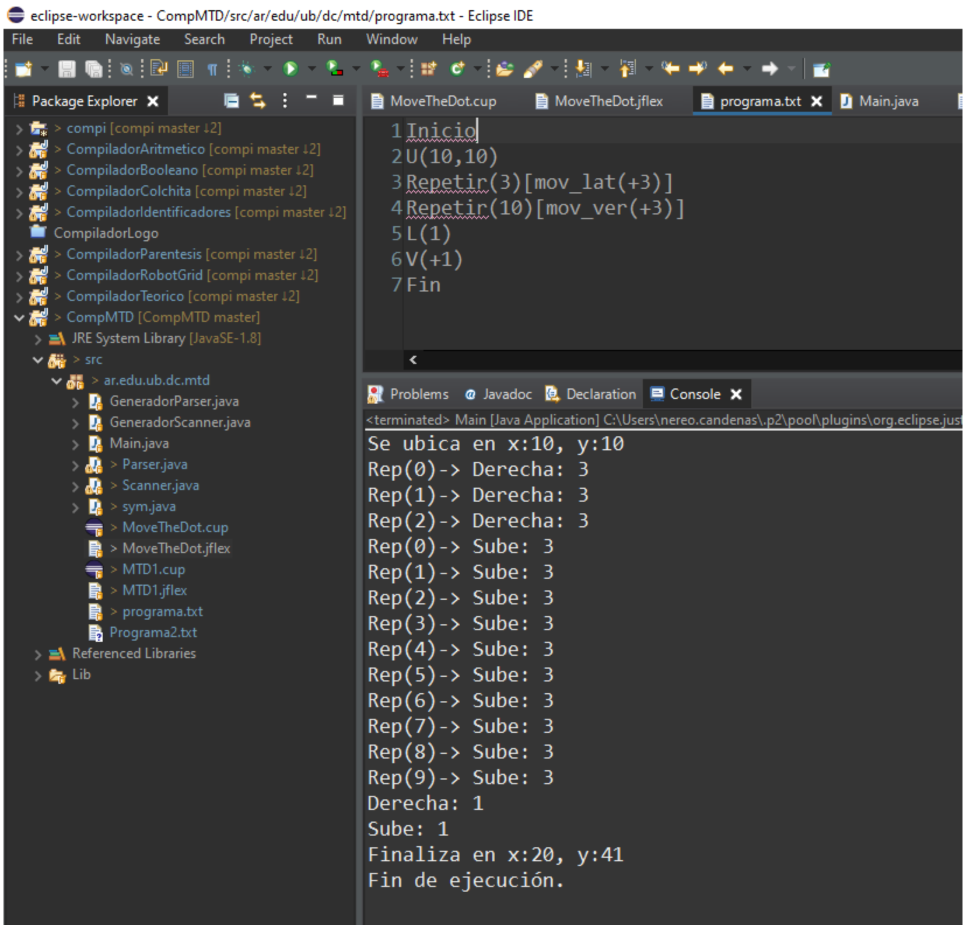
## Ejemplo 2: Programa Sin repetición


eclipse-workspace - CompMTD/src/ar/edu/ub/dc/mtd/programa.txt - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
compi [compi master ↓2]
CompiladorAritmetico [compi master ↓2]
CompiladorBooleano [compi master ↓2]
CompiladorColchita [compi master ↓2]
CompiladorIdentificadores [compi master ↓2]
CompiladorLogo
CompiladorParentesis [compi master ↓2]
CompiladorRobotGrid [compi master ↓2]
CompiladorTeorico [compi master ↓2]
CompMTD [CompMTD master]
JRE System Library [JavaSE-1.8]
src
ar.edu.ub.dc.mtd
GeneradorParser.java
GeneradorScanner.java
Main.java
Parser.java
Scanner.java
sym.java
MoveTheDot.cup
MoveTheDot.jflex
MTD1.cup
MTD1.jflex
programa.txt
Programa2.txt
Referenced Libraries
Lib
MoveTheDot.cup MoveTheDot.jflex programa.txt Main.java
Problems Javadoc Declaration Console
<terminated> Main [Java Application] C:\Users\nereo.candenas\.p2\pool\plugins\org.eclipse.j

```
Inicio
U(0,0)
mov_lat(1)
L(1)
l(1)
mov_ver(+1)
V(+1)
v(+1)
Fin
```

```
Se ubica en x:0, y:0
Derecha: 1
Derecha: 1
Derecha: 1
Sube: 1
Sube: 1
Sube: 1
Finaliza en x:3, y:3
Fin de ejecución.
```

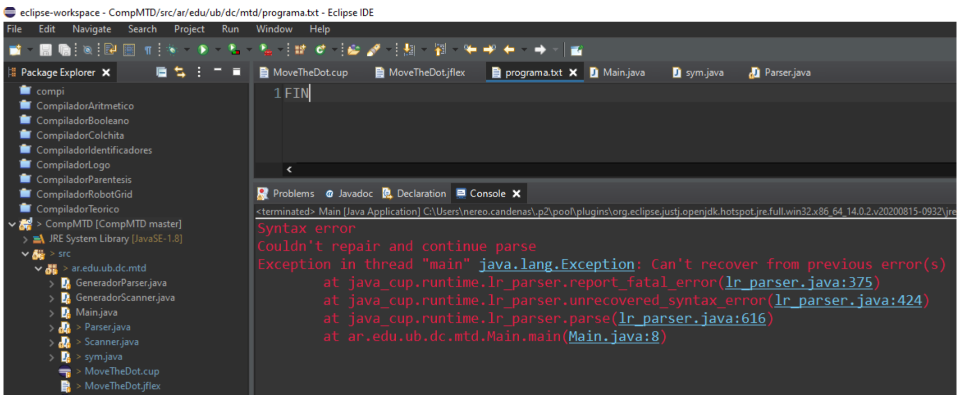
## Ejemplo 3: Programa Con repetición


eclipse-workspace - CompMTD/src/ar/edu/ub/dc/mtd/programa.txt - Eclipse IDE

```
Inicio
U(10,10)
Repetir(3)[mov_lat(+3)]
Repetir(10)[mov_ver(+3)]
L(1)
V(+1)
Fin
```

```
<terminated> Main [Java Application] C:\Users\nereo.candenas\.p2\pool\plugins\org.eclipse.just
Se ubica en x:10, y:10
Rep(0)-> Derecha: 3
Rep(1)-> Derecha: 3
Rep(2)-> Derecha: 3
Rep(0)-> Sube: 3
Rep(1)-> Sube: 3
Rep(2)-> Sube: 3
Rep(3)-> Sube: 3
Rep(4)-> Sube: 3
Rep(5)-> Sube: 3
Rep(6)-> Sube: 3
Rep(7)-> Sube: 3
Rep(8)-> Sube: 3
Rep(9)-> Sube: 3
Derecha: 1
Sube: 1
Finaliza en x:20, y:41
Fin de ejecución.
```

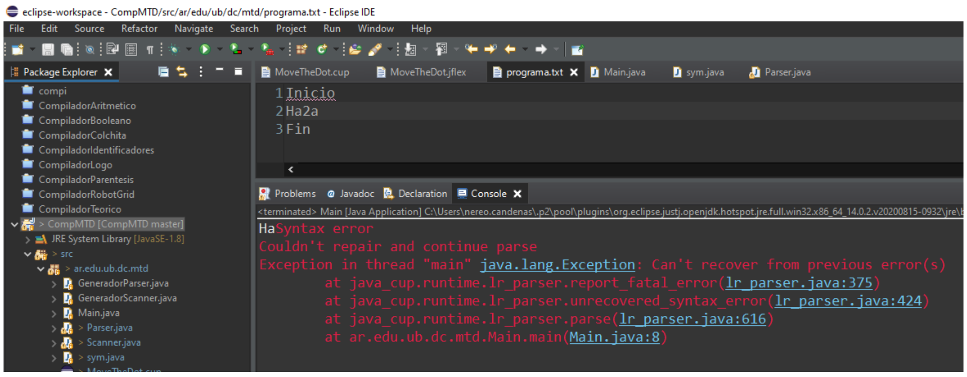
## Ejemplo 4: Programa incompleto (Syntax Error)


eclipse-workspace - CompMTD/src/ar/edu/ub/dc/mtd/programa.txt - Eclipse IDE

```
FIN
```

```
<terminated> Main [Java Application] C:\Users\nereo.candenas\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre
Syntax error
Couldn't repair and continue parse
Exception in thread "main" java.lang.Exception: Can't recover from previous error(s)
	at java_cup.runtime.lr_parser.report_fatal_error(lr_parser.java:375)
	at java_cup.runtime.lr_parser.unrecovered_syntax_error(lr_parser.java:424)
	at java_cup.runtime.lr_parser.parse(lr_parser.java:616)
	at ar.edu.ub.dc.mtd.Main.main(Main.java:8)
```

## Ejemplo 5: Programa erroneo (Syntax Error)



```
Inicio
Ha2a
Fin
```

```
<terminated> Main [Java Application] C:\Users\nereo.candenas\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\
HaSyntax error
Couldn't repair and continue parse
Exception in thread "main" java.lang.Exception: Can't recover from previous error(s)
	at java_cup.runtime.lr_parser.report_fatal_error(lr_parser.java:375)
	at java_cup.runtime.lr_parser.unrecovered_syntax_error(lr_parser.java:424)
	at java_cup.runtime.lr_parser.parse(lr_parser.java:616)
	at ar.edu.ub.dc.mtd.Main.main(Main.java:8)
```

************************************************************************

# Archivos JAVA

## sym.java

```
//----------------------------------------------------
// The following code was generated by CUP v0.11a beta 20060608
// Sat Oct 17 19:28:26 ART 2020
//----------------------------------------------------

package ar.edu.ub.dc.mtd;

/** CUP generated class containing symbol constants. */
public class sym {
  /* terminals */
  public static final int NUM = 15;
  public static final int PAR_A = 7;
  public static final int COMA = 14;
  public static final int POS = 9;
  public static final int INICIO = 2;
  public static final int COR_C = 13;
  public static final int VERTICAL = 5;
  public static final int FIN = 3;
  public static final int EOF = 0;
  public static final int COR_A = 12;
  public static final int error = 1;
  public static final int UBICAR = 4;
  public static final int LATERAL = 6;
  public static final int REPETIR = 11;
  public static final int NEG = 10;
  public static final int PAR_C = 8;
}
```

*************************************************************************

**Parser.java**

```
//----------------------------------------------------
// The following code was generated by CUP v0.11a beta 20060608
// Sat Oct 17 19:28:26 ART 2020
//----------------------------------------------------

package ar.edu.ub.dc.mtd;

import java_cup.runtime.*;

/** CUP v0.11a beta 20060608 generated parser.
  * @version Sat Oct 17 19:28:26 ART 2020
  */
public class Parser extends java_cup.runtime.lr_parser {

  /** Default constructor. */
  public Parser() {super();}

  /** Constructor which sets the default scanner. */
  public Parser(java_cup.runtime.Scanner s) {super(s);}

  /** Constructor which sets the default scanner. */
  public Parser(java_cup.runtime.Scanner s, java_cup.runtime.SymbolFactory sf)
{super(s,sf);}

  /** Production table. */
  protected static final short _production_table[][] =
    unpackFromStrings(new String[] {
    "\000\023\000\002\002\004\000\002\002\004\000\002\002" +
    "\005\000\002\002\006\000\002\005\010\000\002\003\003" +
    "\000\002\003\004\000\002\004\007\000\002\004\015\000" +
    "\002\004\006\000\002\004\014\000\002\004\007\000\002" +
    "\004\015\000\002\004\007\000\002\004\015\000\002\004" +
    "\006\000\002\004\014\000\002\004\007\000\002\004\015" +
    "" });

  /** Access to production table. */
```

```
public short[][] production_table() {return _production_table;}

/** Parse-action table. */
protected static final short[][] _action_table =
  unpackFromStrings(new String[] {
  "\000\105\000\004\004\004\001\002\000\014\005\011\006" +
  "\010\007\013\010\012\015\014\001\002\000\004\002\006" +
  "\001\002\000\004\002\001\001\002\000\010\007\013\010" +
  "\012\015\014\001\002\000\004\011\101\001\002\000\004" +
  "\002\000\001\002\000\004\011\070\001\002\000\004\011" +
  "\057\001\002\000\004\011\021\001\002\000\012\005\017" +
  "\007\013\010\012\015\014\001\002\000\012\005\ufffc\007" +
  "\ufffc\010\ufffc\015\ufffc\001\002\000\004\002\uffff\001\002" +
  "\000\012\005\ufffb\007\ufffb\010\ufffb\015\ufffb\001\002\000" +
  "\004\021\022\001\002\000\004\012\023\001\002\000\004" +
  "\016\024\001\002\000\006\007\026\010\025\001\002\000" +
  "\004\011\043\001\002\000\004\011\027\001\002\000\010" +
  "\013\032\014\031\021\030\001\002\000\004\012\041\001" +
  "\002\000\004\021\036\001\002\000\004\021\033\001\002" +
  "\000\004\012\034\001\002\000\004\017\035\001\002\000" +
  "\012\005\ufff9\007\ufff9\010\ufff9\015\ufff9\001\002\000\004" +
  "\012\037\001\002\000\004\017\040\001\002\000\012\005" +
  "\ufff5\007\ufff5\010\ufff5\015\ufff5\001\002\000\004\017\042" +
  "\001\002\000\012\005\ufff7\007\ufff7\010\ufff7\015\ufff7\001" +
  "\002\000\010\013\046\014\045\021\044\001\002\000\004" +
  "\012\055\001\002\000\004\021\052\001\002\000\004\021" +
  "\047\001\002\000\004\012\050\001\002\000\004\017\051" +
  "\001\002\000\012\005\ufff3\007\ufff3\010\ufff3\015\ufff3\001" +
  "\002\000\004\012\053\001\002\000\004\017\054\001\002" +
  "\000\012\005\uffef\007\uffef\010\uffef\015\uffef\001\002\000" +
  "\004\017\056\001\002\000\012\005\ufff1\007\ufff1\010\ufff1" +
  "\015\ufff1\001\002\000\010\013\062\014\061\021\060\001" +
  "\002\000\004\012\067\001\002\000\004\021\065\001\002" +
  "\000\004\021\063\001\002\000\004\012\064\001\002\000" +
  "\012\005\ufffa\007\ufffa\010\ufffa\015\ufffa\001\002\000\004" +
  "\012\066\001\002\000\012\005\ufff6\007\ufff6\010\ufff6\015" +
  "\ufff6\001\002\000\012\005\ufff8\007\ufff8\010\ufff8\015\ufff8" +
  "\001\002\000\010\013\073\014\072\021\071\001\002\000" +
  "\004\012\100\001\002\000\004\021\076\001\002\000\004" +
  "\021\074\001\002\000\004\012\075\001\002\000\012\005" +
  "\ufff4\007\ufff4\010\ufff4\015\ufff4\001\002\000\004\012\077" +
  "\001\002\000\012\005\ufff0\007\ufff0\010\ufff0\015\ufff0\001" +
  "\002\000\012\005\ufff2\007\ufff2\010\ufff2\015\ufff2\001\002" +
  "\000\004\021\102\001\002\000\004\020\103\001\002\000" +
  "\004\021\104\001\002\000\004\012\105\001\002\000\010" +
  "\007\ufffd\010\ufffd\015\ufffd\001\002\000\012\005\107\007" +
  "\013\010\012\015\014\001\002\000\004\002\ufffe\001\002" +
  "" });

/** Access to parse-action table. */
public short[][] action_table() {return _action_table;}

/** <code>reduce_goto</code> table. */
protected static final short[][] _reduce_table =
  unpackFromStrings(new String[] {
  "\000\105\000\004\002\004\001\001\000\010\003\014\004" +
  "\015\005\006\001\001\000\002\001\001\000\002\001\001" +
  "\000\006\003\105\004\015\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
  "\001\001\000\004\004\017\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
  "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
  "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
  "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
  "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
  "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
  "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
  "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
  "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
  "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\002" +
  "\001\001\000\002\001\001\000\002\001\001\000\002\001" +
  "\001\000\002\001\001\000\002\001\001\000\002\001\001" +
  "\000\002\001\001\000\002\001\001\000\002\001\001\000" +
  "\002\001\001\000\002\001\001\000\002\001\001\000\004" +
  "\004\017\001\001\000\002\001\001" });

/** Access to <code>reduce_goto</code> table. */
public short[][] reduce_table() {return _reduce_table;}

/** Instance of action encapsulation class. */
protected CUP$Parser$actions action_obj;

/** Action encapsulation object initializer. */
protected void init_actions()
  {
      action_obj = new CUP$Parser$actions(this);
    }

  /** Invoke a user supplied parse action. */
  public java_cup.runtime.Symbol do_action(
    int                        act_num,
    java_cup.runtime.lr_parser parser,
    java.util.Stack            stack,
    int                        top)
    throws java.lang.Exception
  {
    /* call code in generated class */
    return action_obj.CUP$Parser$do_action(act_num, parser, stack, top);
  }

  /** Indicates start state. */
  public int start_state() {return 0;}
  /** Indicates start production. */
  public int start_production() {return 0;}

  /** <code>EOF</code> Symbol index. */
  public int EOF_sym() {return 0;}

  /** <code>error</code> Symbol index. */
  public int error_sym() {return 1;}

}

/** Cup generated class to encapsulate user supplied action code.*/
class CUP$Parser$actions {

 static int x = 0;
                                        static int y = 0;

                                        public static void locate_it(int xx, int yy) {
                                          x = xx;
                                          y = yy;
                                        }

                                        public static void move_it(int xx, int yy) {
                                          x += xx;
                                          y += yy;
                                        }
                                        public static void show_it() {
                                                    System.out.println("Finaliza en x:" + x + ", y:" + y);
                                                    System.out.println("Fin de ejecución.");
                                        }

  private final Parser parser;

  /** Constructor */
  CUP$Parser$actions(Parser parser) {
    this.parser = parser;
  }

  /** Method with the actual generated action code. */
  public final java_cup.runtime.Symbol CUP$Parser$do_action(
    int                        CUP$Parser$act_num,
    java_cup.runtime.lr_parser CUP$Parser$parser,
    java.util.Stack            CUP$Parser$stack,
    int                        CUP$Parser$top)
    throws java.lang.Exception
    {
      /* Symbol object for return from actions */
      java_cup.runtime.Symbol CUP$Parser$result;

      /* select the action based on the action number */
      switch (CUP$Parser$act_num)
        {
          /*. . . . . . . . . . . . . . . . . . . .*/
          case 18: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A LATERAL PAR_A NEG NUM
PAR_C COR_C
            {
              Object RESULT =null;
                        int                                        n1left                                        =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
                        int                                        n1right                                       =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
                        Integer            n1            =            (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
                        int                                        n2left                                        =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int                                        n2right                                       =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Integer            n2            =            (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;
```

```
                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Izquierda: "+n2.intValue());
                        move_it(-n2.intValue(),0);
                        }

              CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 17: // LINEAS ::= LATERAL PAR_A NEG NUM PAR_C
            {
              Object RESULT =null;
                    int n1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                    int n1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                    Integer n1 = (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

                     System.out.println("Izquierda: "+n1.intValue());
                     move_it(-n1.intValue(),0);

              CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 16: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A LATERAL PAR_A NUM PAR_C
COR_C
            {
              Object RESULT =null;
                    int n1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).left;
                    int n1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                    Integer n1 = (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                    int n2left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
```

```
                int                                    n2right                                    =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                Integer           n2           =           (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;

                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Derecha: "+n2.intValue());
                        move_it(n2.intValue(),0);
                        }

          CUP$Parser$result        =        parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 15: // LINEAS ::= LATERAL PAR_A NUM PAR_C
        {
          Object RESULT =null;
                int                                    n1left                                     =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int                                    n1right                                    =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Integer           n1           =           (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

      System.out.println("Derecha: "+n1.intValue());
      move_it(n1.intValue(),0);

          CUP$Parser$result        =        parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 14: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A LATERAL PAR_A POS NUM
PAR_C COR_C
        {
          Object RESULT =null;
                int                                    n1left                                     =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
```

```
              int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
              Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
              int n2left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
              int n2right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
              Integer n2 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;

                     for (int i = 0; i < n1; i++) {
                     System.out.println("Rep("+i+")-> Derecha: "+n2.intValue());
                     move_it(n2.intValue(),0);
                     }

        CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
      }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 13: // LINEAS ::= LATERAL PAR_A POS NUM PAR_C
       {
        Object RESULT =null;
              int n1left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
              int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
              Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

      System.out.println("Derecha: "+n1.intValue());
      move_it(n1.intValue(),0);

        CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
       }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
```

```
        case 12: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A VERTICAL PAR_A NEG NUM PAR_C COR_C
          {
            Object RESULT =null;
                int n1left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
                int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
                Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
                int n2left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                int n2right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                Integer n2 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;

                        for (int i = 0; i < n1; i++) {
                        System.out.println("Rep("+i+")-> Baja: "+n2.intValue());
                        move_it(0,-n2.intValue());
                        }

            CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 11: // LINEAS ::= VERTICAL PAR_A NEG NUM PAR_C
          {
            Object RESULT =null;
                int n1left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

                System.out.println("Baja: "+n1.intValue());
                 move_it(0,-n1.intValue());
```

```
              CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 10: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A VERTICAL PAR_A NUM PAR_C
COR_C
            {
              Object RESULT =null;
                        int n1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).left;
                        int n1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-7)).right;
                        Integer n1 = (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-7)).value;
                        int n2left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
                        int n2right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
                        Integer n2 = (Integer)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;

                                          for (int i = 0; i < n1; i++) {
                                          System.out.println("Rep("+i+")-> Sube: "+n2.intValue());
                                          move_it(0,n2.intValue());
                                          }

              CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-9)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 9: // LINEAS ::= VERTICAL PAR_A NUM PAR_C
            {
              Object RESULT =null;
                        int n1left =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                        int n1right =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
```

```
              Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

              System.out.println("Sube: "+n1.intValue());
              move_it(0,n1.intValue());

          CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 8: // LINEAS ::= REPETIR PAR_A NUM PAR_C COR_A VERTICAL PAR_A POS NUM PAR_C COR_C
        {
          Object RESULT =null;
              int n1left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).left;
              int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-8)).right;
              Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-8)).value;
              int n2left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).left;
              int n2right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)).right;
              Integer n2 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-2)).value;

                  for (int i = 0; i < n1; i++) {
                  System.out.println("Rep("+i+")-> Sube: "+n2.intValue());
                  move_it(0,n2.intValue());
                  }

          CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-10)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
        }
      return CUP$Parser$result;

      /*. . . . . . . . . . . . . . . . . . . .*/
      case 7: // LINEAS ::= VERTICAL PAR_A POS NUM PAR_C
        {
            Object RESULT =null;
                int n1left = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int n1right = ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Integer n1 = (Integer)((java_cup.runtime.Symbol) CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

                 System.out.println("Sube: "+n1.intValue());
                 move_it(0,n1.intValue());

            CUP$Parser$result = parser.getSymbolFactory().newSymbol("LINEAS",2, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-4)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
          }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 6: // CUERPO ::= CUERPO LINEAS
            {
              Object RESULT =null;

              CUP$Parser$result = parser.getSymbolFactory().newSymbol("CUERPO",1, ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 5: // CUERPO ::= LINEAS
            {
              Object RESULT =null;

              CUP$Parser$result = parser.getSymbolFactory().newSymbol("CUERPO",1, ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 4: // LINEA1 ::= UBICAR PAR_A NUM COMA NUM PAR_C
            {
              Object RESULT =null;
                int                                   n1left                                   =
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).left;
                int                                   n1right                                  =
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)).right;
                Integer          n1          =          (Integer)((java_cup.runtime.Symbol)
  CUP$Parser$stack.elementAt(CUP$Parser$top-3)).value;
                int                                   n2left                                   =
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int                                   n2right                                  =
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Integer          n2          =          (Integer)((java_cup.runtime.Symbol)
  CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;

                 System.out.println("Se ubica en x:"+n1.intValue()+", y:"+n2.intValue());
                 locate_it(n1.intValue(),n2.intValue());

              CUP$Parser$result        =        parser.getSymbolFactory().newSymbol("LINEA1",3,
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-5)),
  ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 3: // PROGRAMA ::= INICIO LINEA1 CUERPO FIN
            {
              Object RESULT =null;
                 show_it();
              CUP$Parser$result     =     parser.getSymbolFactory().newSymbol("PROGRAMA",0,
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-3)),
  ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;

          /*. . . . . . . . . . . . . . . . . . . .*/
          case 2: // PROGRAMA ::= INICIO CUERPO FIN
            {
              Object RESULT =null;
                 show_it();
              CUP$Parser$result     =     parser.getSymbolFactory().newSymbol("PROGRAMA",0,
  ((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-2)),
  ((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
            }
          return CUP$Parser$result;
```

```
        /*. . . . . . . . . . . . . . . . . . . .*/
        case 1: // PROGRAMA ::= INICIO FIN
         {
           Object RESULT =null;

             System.out.println("Programa vacio");
             show_it();

           CUP$Parser$result = parser.getSymbolFactory().newSymbol("PROGRAMA",0,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        return CUP$Parser$result;

        /*. . . . . . . . . . . . . . . . . . . .*/
        case 0: // $START ::= PROGRAMA EOF
         {
           Object RESULT =null;
                int start_valleft =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).left;
                int start_valright =
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)).right;
                Object start_val = (Object)((java_cup.runtime.Symbol)
CUP$Parser$stack.elementAt(CUP$Parser$top-1)).value;
                RESULT = start_val;
           CUP$Parser$result = parser.getSymbolFactory().newSymbol("$START",0,
((java_cup.runtime.Symbol)CUP$Parser$stack.elementAt(CUP$Parser$top-1)),
((java_cup.runtime.Symbol)CUP$Parser$stack.peek()), RESULT);
         }
        /* ACCEPT */
        CUP$Parser$parser.done_parsing();
        return CUP$Parser$result;

        /* . . . . . .*/
        default:
          throw new Exception(
            "Invalid action number found in internal parse table");

      }
    }
  }
```

**************************************************************************

## Scanner.java

```
/* The following code was generated by JFlex 1.6.0 */

package ar.edu.ub.dc.mtd;

import java_cup.runtime.Symbol;


/**
 * This class is a scanner generated by
 * <a href="http://www.jflex.de/">JFlex</a> 1.6.0
 * from the specification file <tt>src/ar/edu/ub/dc/mtd/MoveTheDot.jflex</tt>
 */
public class Scanner implements java_cup.runtime.Scanner {

  /** This character denotes the end of file */
  public static final int YYEOF = -1;

  /** initial size of the lookahead buffer */
  private static final int ZZ_BUFFERSIZE = 16384;

  /** lexical states */
  public static final int YYINITIAL = 0;

  /**
   * ZZ_LEXSTATE[l] is the state in the DFA for the lexical state l
   * ZZ_LEXSTATE[l+1] is the state in the DFA for the lexical state l
   *                  at the beginning of a line
   * l is of the form l = 2*k, k a non negative integer
   */
  private static final int ZZ_LEXSTATE[] = {
     0, 0
  };

  /**
   * Translates characters to character classes
   */
  private static final String ZZ_CMAP_PACKED =
    "\11\0\2\50\2\0\1\50\22\0\1\50\7\0\1\40\1\41\1\0"+
    "\1\44\1\46\1\45\2\0\12\47\7\0\1\30\1\27\1\7\1\0"+
    "\1\20\1\12\2\0\1\2\2\0\1\32\1\36\1\6\1\10\1\21"+
    "\1\0\1\14\1\0\1\22\1\24\1\37\4\0\1\42\1\0\1\43"+
    "\1\0\1\35\1\0\1\26\1\25\1\4\1\0\1\15\1\11\2\0"+
    "\1\1\2\0\1\31\1\33\1\3\1\5\1\16\1\0\1\13\1\0"+
    "\1\17\1\23\1\34\u05e9\0\12\47\206\0\12\47\306\0\12\47\u019c\0"+
    "\12\47\166\0\12\47\166\0\12\47\166\0\12\47\166\0\12\47\166\0"+
    "\12\47\166\0\12\47\166\0\12\47\166\0\12\47\166\0\12\47\140\0"+
    "\12\47\166\0\12\47\106\0\12\47\u0116\0\12\47\106\0\12\47\u0746\0"+
    "\12\47\46\0\12\47\u012c\0\12\47\200\0\12\47\246\0\12\47\6\0"+
    "\12\47\266\0\12\47\126\0\12\47\206\0\12\47\6\0\12\47\u89c6\0"+
    "\12\47\u02a6\0\12\47\46\0\12\47\306\0\12\47\26\0\12\47\126\0"+
    "\12\47\u0196\0\12\47\u5316\0\12\47\u0586\0\12\47\u0bbc\0\12\47\200\0"+
    "\12\47\74\0\12\47\220\0\12\47\u0116\0\12\47\u01d6\0\12\47\u0176\0"+
    "\12\47\146\0\12\47\u0216\0\12\47\u5176\0\12\47\346\0\12\47\u6c74\0"+

"\62\47\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\uffff\0\
uffff\0\uffff\0\uffff\0\u280f\0";

  /**
   * Translates characters to character classes
   */
  private static final char [] ZZ_CMAP = zzUnpackCMap(ZZ_CMAP_PACKED);

  /**
   * Translates DFA states to action switch labels.
   */
  private static final int [] ZZ_ACTION = zzUnpackAction();

  private static final String ZZ_ACTION_PACKED_0 =
    "\1\0\1\1\2\2\2\3\2\4\2\5\1\6\1\1"+
    "\1\7\1\1\1\10\1\11\1\12\1\13\1\14\1\15"+
    "\1\16\1\17\1\20\15\0\1\3\32\0\1\2\2\0"+
    "\1\5\4\0\1\4";

  private static int [] zzUnpackAction() {
    int [] result = new int[72];
    int offset = 0;
    offset = zzUnpackAction(ZZ_ACTION_PACKED_0, offset, result);
    return result;
  }

  private static int zzUnpackAction(String packed, int offset, int [] result) {
    int i = 0;       /* index in packed string  */
    int j = offset;  /* index in unpacked array */
    int l = packed.length();
    while (i < l) {
      int count = packed.charAt(i++);
    int value = packed.charAt(i++);
    do result[j++] = value; while (--count > 0);
  }
  return j;
}


/**
 * Translates a state to a row index in the transition table
 */
private static final int [] ZZ_ROWMAP = zzUnpackRowMap();

private static final String ZZ_ROWMAP_PACKED_0 =
  "\0\0\0\51\0\122\0\173\0\244\0\315\0\366\0\u011f"+
  "\0\u0148\0\u0171\0\51\0\u019a\0\51\0\u01c3\0\51\0\51"+
  "\0\51\0\51\0\51\0\51\0\51\0\u01ec\0\u0215\0\u023e"+
  "\0\u0267\0\u0290\0\u02b9\0\u02e2\0\u030b\0\u0334\0\u035d\0\u0386"+
  "\0\u03af\0\u03d8\0\u0401\0\u042a\0\51\0\u0453\0\u047c\0\u04a5"+
  "\0\u04ce\0\u04f7\0\u0520\0\u0549\0\u0572\0\u059b\0\u05c4\0\u05ed"+
  "\0\u0616\0\u063f\0\u0668\0\u0691\0\u06ba\0\u06e3\0\u070c\0\u0735"+
  "\0\u075e\0\u0787\0\u07b0\0\u07d9\0\u0802\0\u082b\0\u0854\0\51"+
  "\0\u087d\0\u08a6\0\51\0\u08cf\0\u08f8\0\u0921\0\u094a\0\51";

private static int [] zzUnpackRowMap() {
  int [] result = new int[72];
  int offset = 0;
  offset = zzUnpackRowMap(ZZ_ROWMAP_PACKED_0, offset, result);
  return result;
}

private static int zzUnpackRowMap(String packed, int offset, int [] result) {
  int i = 0;  /* index in packed string  */
  int j = offset;  /* index in unpacked array */
  int l = packed.length();
  while (i < l) {
    int high = packed.charAt(i++) << 16;
    result[j++] = high | packed.charAt(i++);
  }
  return j;
}

/**
 * The transition table of the DFA
 */
```

```
private static final int [] ZZ_TRANS = zzUnpackTrans();

private static final String ZZ_TRANS_PACKED_0 =
  "\1\2\1\3\1\4\6\2\1\5\1\6\1\7\1\10"+
  "\6\2\1\11\1\12\4\2\2\13\1\14\1\15\1\2"+
  "\1\16\1\15\1\17\1\20\1\21\1\22\1\23\1\24"+
  "\1\25\1\26\1\27\54\0\1\30\50\0\1\30\2\0"+
  "\1\31\43\0\1\32\50\0\1\32\1\33\63\0\1\34"+
  "\50\0\1\34\2\0\1\35\55\0\1\36\50\0\1\36"+
  "\1\0\1\37\26\0\1\40\50\0\1\41\2\0\1\42"+
  "\107\0\1\26\51\0\1\27\1\0\1\43\51\0\1\44"+
  "\51\0\1\45\53\0\1\45\60\0\1\46\53\0\1\47"+
  "\30\0\1\50\51\0\1\51\102\0\1\52\50\0\1\53"+
  "\53\0\1\54\15\0\1\55\53\0\1\56\56\0\1\57"+
  "\53\0\1\60\34\0\1\61\53\0\1\62\76\0\1\63"+
  "\50\0\1\64\50\0\1\65\14\0\1\66\51\0\1\67"+
  "\65\0\1\70\53\0\1\71\54\0\1\72\52\0\1\73"+
  "\51\0\1\74\2\0\1\75\46\0\1\74\4\0\1\75"+
  "\43\0\1\76\4\0\1\77\16\0\1\100\53\0\1\100"+
  "\41\0\1\101\51\0\1\102\61\0\1\103\51\0\1\103"+
  "\62\0\1\104\37\0\1\105\63\0\1\106\40\0\1\107"+
  "\43\0\1\110\51\0\1\110\53\0\1\13\44\0\1\15"+
  "\57\0\1\13\42\0\1\15\34\0";

private static int [] zzUnpackTrans() {
 int [] result = new int[2419];
 int offset = 0;
 offset = zzUnpackTrans(ZZ_TRANS_PACKED_0, offset, result);
 return result;
}

private static int zzUnpackTrans(String packed, int offset, int [] result) {
 int i = 0;      /* index in packed string  */
 int j = offset;  /* index in unpacked array */
 int l = packed.length();
 while (i < l) {
  int count = packed.charAt(i++);
  int value = packed.charAt(i++);
  value--;
  do result[j++] = value; while (--count > 0);
 }
 return j;
}
```

```
/* error codes */
private static final int ZZ_UNKNOWN_ERROR = 0;
private static final int ZZ_NO_MATCH = 1;
private static final int ZZ_PUSHBACK_2BIG = 2;

/* error messages for the codes above */
private static final String ZZ_ERROR_MSG[] = {
  "Unkown internal scanner error",
  "Error: could not match input",
  "Error: pushback value was too large"
};

/**
 * ZZ_ATTRIBUTE[aState] contains the attributes of state <code>aState</code>
 */
private static final int [] ZZ_ATTRIBUTE = zzUnpackAttribute();

private static final String ZZ_ATTRIBUTE_PACKED_0 =
  "\1\0\1\11\10\1\1\11\1\1\1\11\1\1\7\11"+
  "\2\1\15\0\1\11\32\0\1\11\2\0\1\11\4\0"+
  "\1\11";

private static int [] zzUnpackAttribute() {
  int [] result = new int[72];
  int offset = 0;
  offset = zzUnpackAttribute(ZZ_ATTRIBUTE_PACKED_0, offset, result);
  return result;
}

private static int zzUnpackAttribute(String packed, int offset, int [] result) {
  int i = 0;       /* index in packed string  */
  int j = offset;  /* index in unpacked array */
  int l = packed.length();
  while (i < l) {
    int count = packed.charAt(i++);
    int value = packed.charAt(i++);
    do result[j++] = value; while (--count > 0);
  }
  return j;
}

/** the input device */
private java.io.Reader zzReader;
```

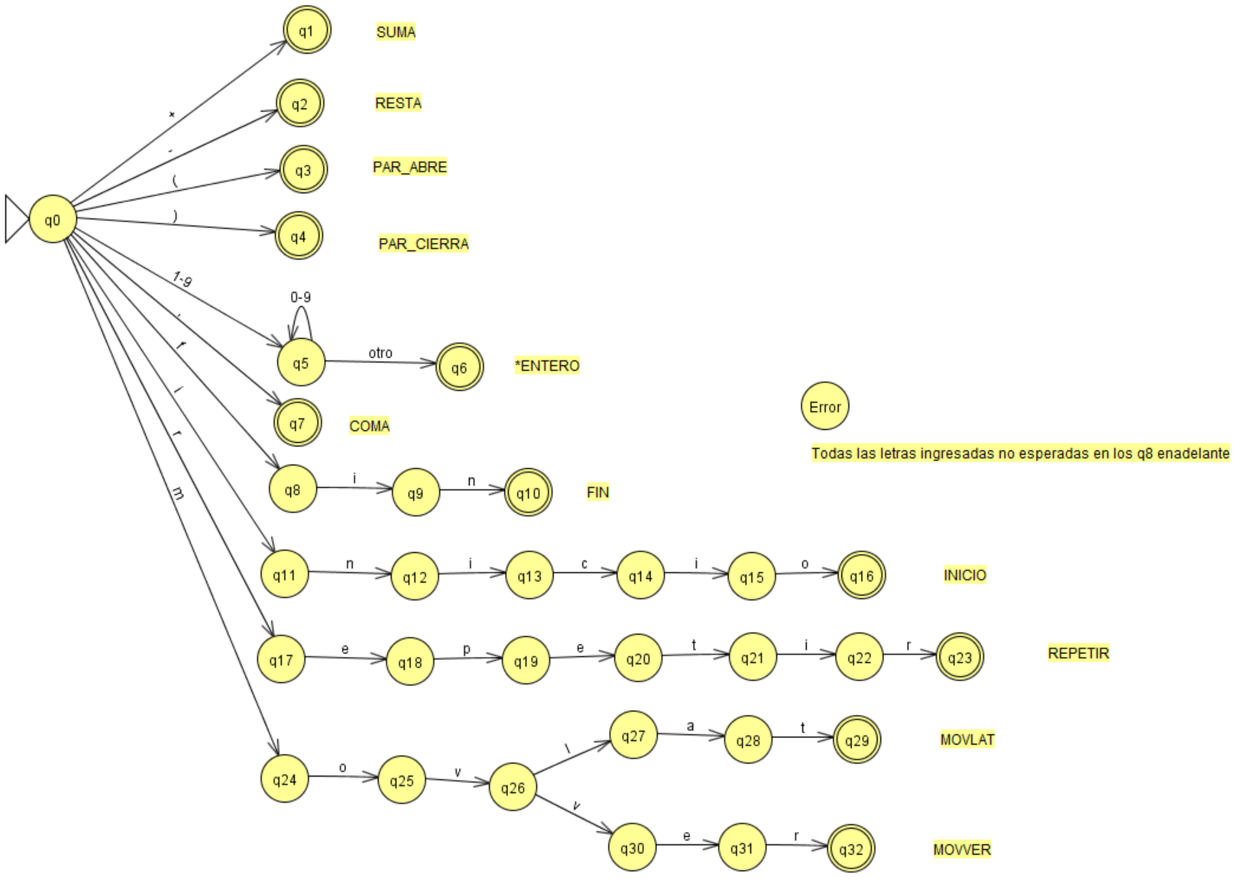
```
/** the current state of the DFA */
private int zzState;

/** the current lexical state */
private int zzLexicalState = YYINITIAL;

/** this buffer contains the current text to be matched and is
    the source of the yytext() string */
private char zzBuffer[] = new char[ZZ_BUFFERSIZE];

/** the textposition at the last accepting state */
private int zzMarkedPos;

/** the current text position in the buffer */
private int zzCurrentPos;

/** startRead marks the beginning of the yytext() string in the buffer */
private int zzStartRead;

/** endRead marks the last character in the buffer, that has been read
    from input */
private int zzEndRead;

/** number of newlines encountered up to the start of the matched text */
private int yyline;

/** the number of characters up to the start of the matched text */
private int yychar;

/**
 * the number of characters from the last newline up to the start of the
 * matched text
 */
private int yycolumn;

/**
 * zzAtBOL == true <=> the scanner is currently at the beginning of a line
 */
private boolean zzAtBOL = true;

/** zzAtEOF == true <=> the scanner is at the EOF */
private boolean zzAtEOF;
```

```
/** denotes if the user-EOF-code has already been executed */
private boolean zzEOFDone;

/**
 * The number of occupied positions in zzBuffer beyond zzEndRead.
 * When a lead/high surrogate has been read from the input stream
 * into the final zzBuffer position, this will have a value of 1;
 * otherwise, it will have a value of 0.
 */
private int zzFinalHighSurrogate = 0;


/**
 * Creates a new scanner
 *
 * @param   in  the java.io.Reader to read input from.
 */
public Scanner(java.io.Reader in) {
  this.zzReader = in;
}


/**
 * Unpacks the compressed character translation table.
 *
 * @param packed   the packed character translation table
 * @return         the unpacked character translation table
 */
private static char [] zzUnpackCMap(String packed) {
  char [] map = new char[0x110000];
  int i = 0;  /* index in packed string  */
  int j = 0;  /* index in unpacked array */
  while (i < 354) {
    int  count = packed.charAt(i++);
    char value = packed.charAt(i++);
    do map[j++] = value; while (--count > 0);
  }
  return map;
}


/**
 * Refills the input buffer.
 *
```

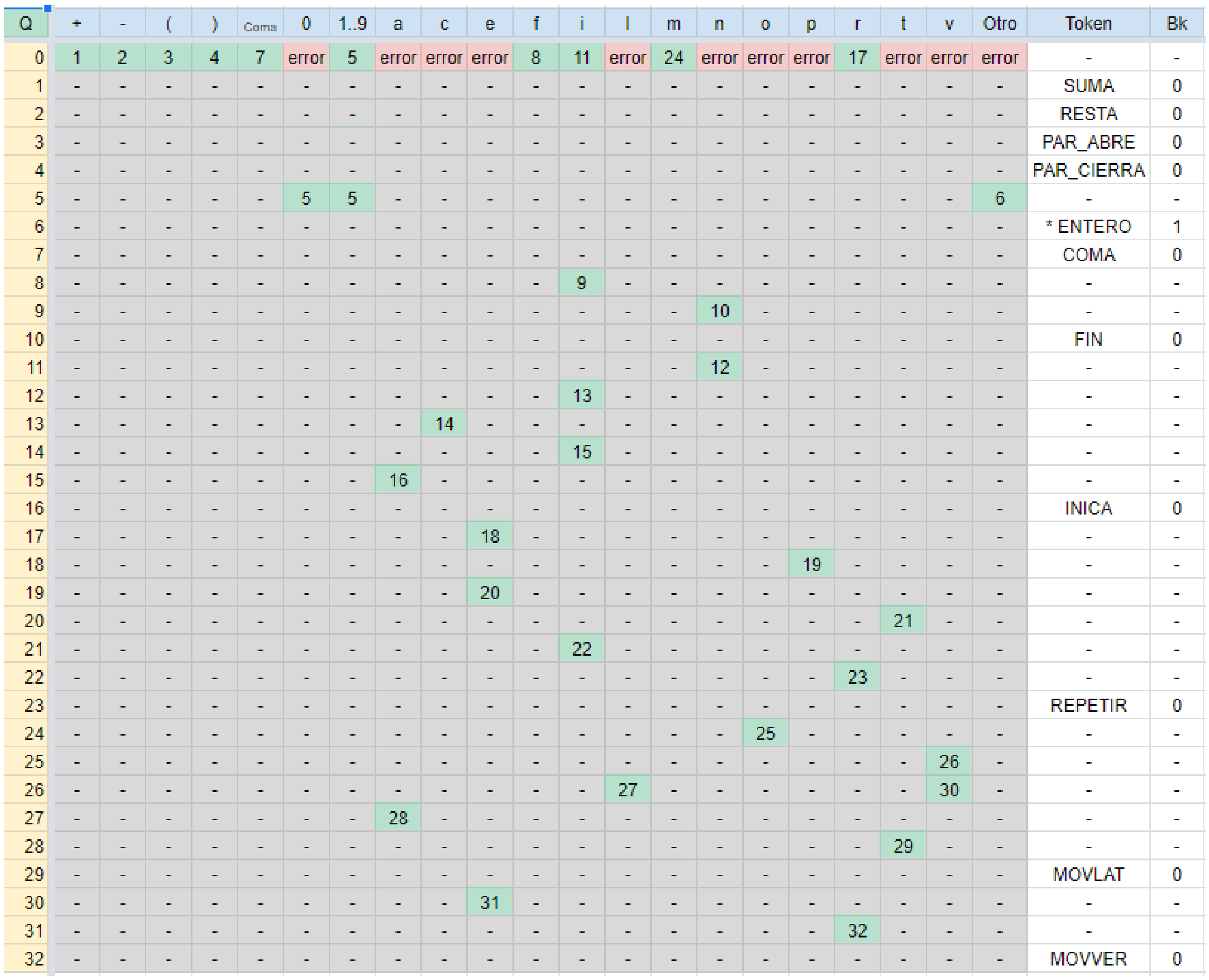
```
   * @return      <code>false</code>, iff there was new input.
   *
   * @exception   java.io.IOException  if any I/O-Error occurs
   */
  private boolean zzRefill() throws java.io.IOException {

    /* first: make room (if you can) */
    if (zzStartRead > 0) {
      zzEndRead += zzFinalHighSurrogate;
      zzFinalHighSurrogate = 0;
      System.arraycopy(zzBuffer, zzStartRead,
                       zzBuffer, 0,
                       zzEndRead-zzStartRead);

      /* translate stored positions */
      zzEndRead-= zzStartRead;
      zzCurrentPos-= zzStartRead;
      zzMarkedPos-= zzStartRead;
      zzStartRead = 0;
    }

    /* is the buffer big enough? */
    if (zzCurrentPos >= zzBuffer.length - zzFinalHighSurrogate) {
      /* if not: blow it up */
      char newBuffer[] = new char[zzBuffer.length*2];
      System.arraycopy(zzBuffer, 0, newBuffer, 0, zzBuffer.length);
      zzBuffer = newBuffer;
      zzEndRead += zzFinalHighSurrogate;
      zzFinalHighSurrogate = 0;
    }

    /* fill the buffer with new input */
    int requested = zzBuffer.length - zzEndRead;
    int totalRead = 0;
    while (totalRead < requested) {
      int numRead = zzReader.read(zzBuffer, zzEndRead + totalRead, requested - totalRead);
      if (numRead == -1) {
        break;
      }
      totalRead += numRead;
    }

    if (totalRead > 0) {
      zzEndRead += totalRead;
      if (totalRead == requested) { /* possibly more input available */
        if (Character.isHighSurrogate(zzBuffer[zzEndRead - 1])) {
          --zzEndRead;
          zzFinalHighSurrogate = 1;
        }
      }
      return false;
    }

    // totalRead = 0: End of stream
    return true;
  }


  /**
   * Closes the input stream.
   */
  public final void yyclose() throws java.io.IOException {
    zzAtEOF = true;            /* indicate end of file */
    zzEndRead = zzStartRead;  /* invalidate buffer    */

    if (zzReader != null)
      zzReader.close();
  }


  /**
   * Resets the scanner to read from a new input stream.
   * Does not close the old reader.
   *
   * All internal variables are reset, the old input stream
   * <b>cannot</b> be reused (internal buffer is discarded and lost).
   * Lexical state is set to <tt>ZZ_INITIAL</tt>.
   *
   * Internal scan buffer is resized down to its initial length, if it has grown.
   *
   * @param reader   the new input stream
   */
  public final void yyreset(java.io.Reader reader) {
    zzReader = reader;
    zzAtBOL  = true;
    zzAtEOF  = false;
    zzEOFDone = false;
    zzEndRead = zzStartRead = 0;
    zzCurrentPos = zzMarkedPos = 0;
    zzFinalHighSurrogate = 0;
    yyline = yychar = yycolumn = 0;
    zzLexicalState = YYINITIAL;
    if (zzBuffer.length > ZZ_BUFFERSIZE)
      zzBuffer = new char[ZZ_BUFFERSIZE];
  }


  /**
   * Returns the current lexical state.
   */
  public final int yystate() {
    return zzLexicalState;
  }

  /**
   * Enters a new lexical state
   *
   * @param newState the new lexical state
   */
  public final void yybegin(int newState) {
    zzLexicalState = newState;
  }

  /**
   * Returns the text matched by the current regular expression.
   */
  public final String yytext() {
    return new String( zzBuffer, zzStartRead, zzMarkedPos-zzStartRead );
  }


  /**
   * Returns the character at position <tt>pos</tt> from the
   * matched text.
   *
   * It is equivalent to yytext().charAt(pos), but faster
   *
   * @param pos the position of the character to fetch.
   *            A value from 0 to yylength()-1.
   *
   * @return the character at position pos
   */
```

```
public final char yycharat(int pos) {
  return zzBuffer[zzStartRead+pos];
}


/**
 * Returns the length of the matched text region.
 */
public final int yylength() {
  return zzMarkedPos-zzStartRead;
}


/**
 * Reports an error that occured while scanning.
 *
 * In a wellformed scanner (no or only correct usage of
 * yypushback(int) and a match-all fallback rule) this method
 * will only be called with things that "Can't Possibly Happen".
 * If this method is called, something is seriously wrong
 * (e.g. a JFlex bug producing a faulty scanner etc.).
 *
 * Usual syntax/scanner level error handling should be done
 * in error fallback rules.
 *
 * @param   errorCode  the code of the errormessage to display
 */
private void zzScanError(int errorCode) {
  String message;
  try {
    message = ZZ_ERROR_MSG[errorCode];
  }
  catch (ArrayIndexOutOfBoundsException e) {
    message = ZZ_ERROR_MSG[ZZ_UNKNOWN_ERROR];
  }

  throw new Error(message);
}


/**
 * Pushes the specified amount of characters back into the input stream.
 *
 * They will be read again by then next call of the scanning method
 *
 * @param number  the number of characters to be read again.
 *                This number must not be greater than yylength()!
 */
public void yypushback(int number)  {
  if ( number > yylength() )
    zzScanError(ZZ_PUSHBACK_2BIG);

  zzMarkedPos -= number;
}


/**
 * Contains user EOF-code, which will be executed exactly once,
 * when the end of file is reached
 */
private void zzDoEOF() throws java.io.IOException {
  if (!zzEOFDone) {
    zzEOFDone = true;
    yyclose();
  }
}


/**
 * Resumes scanning until the next regular expression is matched,
 * the end of input is encountered or an I/O-Error occurs.
 *
 * @return      the next token
 * @exception   java.io.IOException  if any I/O-Error occurs
 */
public java_cup.runtime.Symbol next_token() throws java.io.IOException {
  int zzInput;
  int zzAction;

  // cached fields:
  int zzCurrentPosL;
  int zzMarkedPosL;
  int zzEndReadL = zzEndRead;
  char [] zzBufferL = zzBuffer;
  char [] zzCMapL = ZZ_CMAP;

  int [] zzTransL = ZZ_TRANS;
  int [] zzRowMapL = ZZ_ROWMAP;
  int [] zzAttrL = ZZ_ATTRIBUTE;

  while (true) {
    zzMarkedPosL = zzMarkedPos;

    zzAction = -1;

    zzCurrentPosL = zzCurrentPos = zzStartRead = zzMarkedPosL;

    zzState = ZZ_LEXSTATE[zzLexicalState];

    // set up zzAction for empty match case:
    int zzAttributes = zzAttrL[zzState];
    if ( (zzAttributes & 1) == 1 ) {
     zzAction = zzState;
    }


    zzForAction: {
     while (true) {

      if (zzCurrentPosL < zzEndReadL) {
       zzInput = Character.codePointAt(zzBufferL, zzCurrentPosL, zzEndReadL);
       zzCurrentPosL += Character.charCount(zzInput);
      }
      else if (zzAtEOF) {
       zzInput = YYEOF;
       break zzForAction;
      }
      else {
       // store back cached positions
       zzCurrentPos  = zzCurrentPosL;
       zzMarkedPos   = zzMarkedPosL;
       boolean eof = zzRefill();
       // get translated positions and possibly new buffer
       zzCurrentPosL  = zzCurrentPos;
       zzMarkedPosL   = zzMarkedPos;
       zzBufferL      = zzBuffer;
       zzEndReadL     = zzEndRead;
       if (eof) {
        zzInput = YYEOF;
        break zzForAction;
       }
       else {
           zzInput = Character.codePointAt(zzBufferL, zzCurrentPosL, zzEndReadL);
           zzCurrentPosL += Character.charCount(zzInput);
         }
       }
       int zzNext = zzTransL[ zzRowMapL[zzState] + zzCMapL[zzInput] ];
       if (zzNext == -1) break zzForAction;
       zzState = zzNext;

       zzAttributes = zzAttrL[zzState];
       if ( (zzAttributes & 1) == 1 ) {
         zzAction = zzState;
         zzMarkedPosL = zzCurrentPosL;
         if ( (zzAttributes & 8) == 8 ) break zzForAction;
       }

     }
  }

  // store back cached position
  zzMarkedPos = zzMarkedPosL;

  switch (zzAction < 0 ? zzAction : ZZ_ACTION[zzAction]) {
     case 1:
       { System.out.print(yytext());
       }
     case 17: break;
     case 2:
       { return new Symbol(sym.INICIO);
       }
     case 18: break;
     case 3:
       { return new Symbol(sym.FIN);
       }
     case 19: break;
     case 4:
       { return new Symbol(sym.REPETIR);
       }
     case 20: break;
     case 5:
       { return new Symbol(sym.UBICAR);
       }
     case 21: break;
     case 6:
       { return new Symbol(sym.LATERAL);
       }
     case 22: break;
     case 7:
       { return new Symbol(sym.VERTICAL);
       }
     case 23: break;
     case 8:
       { return new Symbol(sym.PAR_A);
       }
     case 24: break;
     case 9:
       { return new Symbol(sym.PAR_C);
       }
     case 25: break;
     case 10:
       { return new Symbol(sym.COR_A);
       }
     case 26: break;
     case 11:
       { return new Symbol(sym.COR_C);
       }
     case 27: break;
     case 12:
       { return new Symbol(sym.POS);
       }
     case 28: break;
     case 13:
       { return new Symbol(sym.NEG);
       }
     case 29: break;
     case 14:
       { return new Symbol(sym.COMA);
       }
     case 30: break;
     case 15:
       { return new Symbol(sym.NUM, new Integer(yytext()));
       }
     case 31: break;
     case 16:
       { ;
       }
     case 32: break;
     default:
       if (zzInput == YYEOF && zzStartRead == zzCurrentPos) {
          zzAtEOF = true;
          zzDoEOF();
            { return new java_cup.runtime.Symbol(sym.EOF); }
        }
        else {
          zzScanError(ZZ_NO_MATCH);
        }
      }
    }
  }

  /**
   * Runs the scanner on input files.
   *
   * This is a standalone scanner, it will print any unmatched
   * text to System.out unchanged.
   *
   * @param argv   the command line, contains the filenames to run
   *               the scanner on.
   */
  public static void main(String argv[]) {
    if (argv.length == 0) {
      System.out.println("Usage : java Scanner [ --encoding <name> ] <inputfile(s)>");
    }
    else {
      int firstFilePos = 0;
      String encodingName = "UTF-8";
      if (argv[0].equals("--encoding")) {
        firstFilePos = 2;
        encodingName = argv[1];
        try {
          java.nio.charset.Charset.forName(encodingName); // Side-effect: is encodingName valid?
        } catch (Exception e) {
          System.out.println("Invalid encoding '" + encodingName + "'");
          return;
        }
      }
      for (int i = firstFilePos; i < argv.length; i++) {
        Scanner scanner = null;
        try {
          java.io.FileInputStream stream = new java.io.FileInputStream(argv[i]);
          java.io.Reader reader = new java.io.InputStreamReader(stream, encodingName);
          scanner = new Scanner(reader);
          while ( !scanner.zzAtEOF ) scanner.next_token();
        }
        catch (java.io.FileNotFoundException e) {
         System.out.println("File not found : \""+argv[i]+"\"");
        }
        catch (java.io.IOException e) {
         System.out.println("IO error scanning file \""+argv[i]+"\"");
         System.out.println(e);
        }
        catch (Exception e) {
         System.out.println("Unexpected exception:");
         e.printStackTrace();
        }
      }
    }
  }
}
```

## Árbol de Parsing


q0
+
-
(
)
1-9
,
f
i
r
m
q1
SUMA
q2
RESTA
q3
PAR_ABRE
q4
PAR_CIERRA
0-9
q5
otro
q6
*ENTERO
q7
COMA
Error
Todas las letras ingresadas no esperadas en los q8 enadelante
q8 i q9 n q10
FIN
q11 n q12 i q13 c q14 i q15 o q16
INICIO
q17 e q18 p q19 e q20 t q21 i q22 r q23
REPETIR
q24 o q25 v q26
l q27 a q28 t q29
MOVLAT
v q30 e q31 r q32
MOVVER

## Tabla de transición



| Q | + | - | ( | ) | Coma | 0 | 1..9 | a | c | e | f | i | l | m | n | o | p | r | t | v | Otro | Token | Bk |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 7 | error | 5 | error | error | error | 8 | 11 | error | 24 | error | error | error | 17 | error | error | error | - | - |
| 1 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | SUMA | 0 |
| 2 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | RESTA | 0 |
| 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | PAR_ABRE | 0 |
| 4 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | PAR_CIERRA | 0 |
| 5 | - | - | - | - | - | 5 | 5 | - | - | - | - | - | - | - | - | - | - | - | - | - | 6 | - | - |
| 6 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | * ENTERO | 1 |
| 7 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | COMA | 0 |
| 8 | - | - | - | - | - | - | - | - | - | - | - | 9 | - | - | - | - | - | - | - | - | - | - | - |
| 9 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 10 | - | - | - | - | - | - | - | - |
| 10 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | FIN | 0 |
| 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 12 | - | - | - | - | - | - | - | - |
| 12 | - | - | - | - | - | - | - | - | - | - | - | 13 | - | - | - | - | - | - | - | - | - | - | - |
| 13 | - | - | - | - | - | - | - | - | 14 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 14 | - | - | - | - | - | - | - | - | - | - | - | 15 | - | - | - | - | - | - | - | - | - | - | - |
| 15 | - | - | - | - | - | - | - | 16 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | INICA | 0 |
| 17 | - | - | - | - | - | - | - | - | - | 18 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 18 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 19 | - | - | - | - | - | - |
| 19 | - | - | - | - | - | - | - | - | - | 20 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 20 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 21 | - | - | - | - |
| 21 | - | - | - | - | - | - | - | - | - | - | - | 22 | - | - | - | - | - | - | - | - | - | - | - |
| 22 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 23 | - | - | - | - | - |
| 23 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | REPETIR | 0 |
| 24 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 25 | - | - | - | - | - | - | - |
| 25 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 26 | - | - | - |
| 26 | - | - | - | - | - | - | - | - | - | - | - | - | 27 | - | - | - | - | - | - | 30 | - | - | - |
| 27 | - | - | - | - | - | - | - | 28 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 28 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 29 | - | - | - | - |
| 29 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | MOVLAT | 0 |
| 30 | - | - | - | - | - | - | - | - | - | 31 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 31 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 32 | - | - | - | - | - |
| 32 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | MOVVER | 0 |