Instituto Tecnológico de Costa Rica

Sede San Carlos

Escuela de Computación

Lenguajes de Programación

**Primer Proyecto Programado: Análisis Sintáctico**

Oscar Víquez Acuña

Walter Chavarría Mora 201128928

Cristiám Alberto Vega Fernández 201030999

19 de Abril de 2013

# Análisis del lenguaje

Para la creación del Analizador Sintáctico se trabajó bajo el lenguaje de programación JAVA, el cual es un lenguaje orientado a objetos diseñado para ser multiplataforma y poder ser empleado el mismo programa en diversos sistemas operativos. Java es un lenguaje relativamente sencillo, debido a que prácticamente toda la funcionalidad se encuentra en clases que forman parte del API de java. Constantemente están surgiendo nuevos apis, que proporcionan nuevas extensiones a las características del lenguaje.

Estas características, junto con el hecho de que sea un lenguaje libre, pudiéndose utilizar el compilador y la máquina virtual de forma gratuita le augura un gran futuro.

Algunas de las características con las que cuenta el lenguaje a implementar es que no permite la definición de variables anidadas, esto quiere decir que a la hora de la creación del código se pueden declarar variables globales y variables locales pero no variables anidadas que estén dentro de otros métodos. La definición de clases, de métodos, de funciones y de constructores, en este apartado a lo que se refiere es que se permite la creación de las clases, la creación de métodos y a su vez la creación de funciones que en este caso es por medio de las palabras reservadas para realización de procedimientos. También permite la herencia ya que permite la creación de nuevas clases basadas en clases existentes.

Además un identificador es el nombre de variables, métodos, clases e interfaces. Un identificador es una secuencia ilimitada de caracteres alfabéticos o dígitos (unicode) que comienzan con un carácter alfabético. Los nombres pueden incluir el carácter subrayado (\_) y el dólar ($). Toda variable debe ser declarada antes de poder ser utilizada, además el lenguaje no soporta el uso de arreglos, aunque si permite el uso de bucles(while) anidados.

Permite el uso de import, a los paquetes y clase de las bibliotecas se accede utilizando la palabra reservada import.

# Soluciones e implementación

Para dar una completa y efectiva solución al problema planteado en dicho proyecto se trabajó en fases o etapas, cada etapa correspondiente a una etapa interna del análisis sintáctico que realiza normalmente un compilador: Las etapas realizadas fueron las siguientes:

1. Identificación de los tokens: En la gramática dada para la realización del proyecto se hizo un análisis identificando aquellos símbolos correspondientes a tokens y aquellos que únicamente desencadenan otras reglas, también conocidos durante el proyecto como tokens no terminales. Dichos tokens se almacenaron en una clase con un correspondiente valor numérico para conseguir una identificación de los mismos más sencilla.

2. Preparación de la Gramática: La gramática brindada no se encontraba completamente en **bnf** o **ebnf**, por lo tanto se realizó la conversión de las reglas que aún se encontraban en **ebnf** al sistema **bnf** para tenerlas a las mismas en un único sistema. La conversión se realizó en múltiples ocasiones porque al generar el archivo “.cup” daba errores.

3. Generar Archivo Flex: Para la generación del archivo **“.flex”**  se siguieron las reglas encontradas en la página oficial. **Ver anexo 1.**

4. Scanner: Se encarga de leer el código que el usuario ingresa el editor y verificar que los lexemas tengan su correspondiente en la clase que contiene los tokens identificados anteriormente, si esto no ocurre muestra un error de identificación de token, el error será indicado por la excepción creada para el scanner.

5. Generar Archivo Cup: Para la generación del cup se tomó en cuenta las reglas de la gramática brindada por el profesor y las modificaciones que a esta se le realizaron para utilizar el sistema bnf. **Ver anexo 2.**

6. Parser: Se encarga de verificar que se cumpla con las reglas definidas para el lenguaje creado, si encontrara que un carácter no se encuentra en la posición correcta este enviará un ParserException que indicara el carácter leído la posición donde se encuentra y lo que esperaba encontrar. Si se corre correctamente el parser este estará generando el árbol AST, ya que cada lectura de regla que hace va utilizando y asignando a la clase correspondiente.

7. Realizar las Clases: La realización de las clases se hizo según la gramática que se ingresó finalmente al archivo, cup los no terminales que iniciaban una regla(los que se encuentran al lado izquierdo de la asignación) se incluyeron como reglas abstractas las cuales heredan de la clase principal del AST, las que se encuentran al lado derecho de las asignaciones se crearon como clases concretas las cuales heredan de las abstractas que corresponden según su regla.

8. Impresión del AST: Para la impresión se utiliza lo brindada por el profesor en clase la única modificación fue la de agregar al final (en las hojas del árbol) el valor correspondiente a los id’s, números y cadenas de texto, esto por medio del valor que trae el id el cual se encuentra en la clase como symbol lo cual nos permite utilizar su valor.

9. Otras Soluciones: Al pasar la gramática de ebnf a bnf surgieron inconvenientes, los cuales se intentaron arreglar pero la forma de trabajar del cup no permitió conseguir un trabajo efectivo.

Las siguientes implementaciones se aplicaron al código y a los ejemplos utilizados:

1. La regla que realizaba la identificación de los tipos de datos no se pudo realizar efectivamente porque al generar el cup daba errores ya que existía la posibilidad de tener dos identificadores juntos, cuando esto no era permitido por el cup, Para solucionar este problema se realizó la sustitución de la regla type en la demás reglas por un identificador para poder tomar cualquier identificador, brindándonos la posibilidad de mediante una comprobación poder obtener cualquier tipo asignado. Por este motivo al probar código los tipos asignados en el código del editor no pueden ser int, string o bool , deben contener cualquier otro carácter extra, esto porque el scanner los detecta pero las reglas del cup no los admiten generando con esto un error.
2. Cuando se desea probar un código por errores que la gramática genera y los cuales no pudieron corregirse, se debe agregar siempre a la primera asignación de un método o clase unos corchetes a ambos lados, encerrarla entre corchetes. Posterior a este cambio las demás asignaciones que se encuentren dentro del mismo método servirán correctamente.

# Resultados obtenidos

|  |  |
| --- | --- |
| **Objetivo o problema a resolver** | **Resultados(Total, Parcial, Nulo)** |
| Editor | Total |
| Scanner | Total |
| Parser | Total |
| AST | Total |

# Conclusiones

Hay momentos que se cree que la creación de un compilador es algo sencillo de realizar pero a la hora de la práctica se llega a la conclusión que estamos equivocados. La creación de un compilador requiere de mucho esfuerzo mental y físico, a la hora de la realización del análisis sintáctico se puede denotar que lleva su complejidad porque hay que analizar la gramática del lenguaje que se quiere compilar y eso conlleva mucho esfuerzo porque la gramática es muy extensa. Fue muy complicado de implementar y eso que utilizamos herramientas de que ayudan a simplificar procesos como es el JFLEX y el JavaCup.

También he notado como todas nuestras materias se va complementando y enlazando, por ejemplo, en [matemáticas](http://www.monografias.com/Matematicas/index.shtml) discretas vimos la representación de árboles, los cuales usamos aquí. Igualmente en estructura de datos I, vimos [métodos](http://www.monografias.com/trabajos11/metods/metods.shtml) de ordenamiento que las gramáticas de los compiladores usan. Por lo tanto, no parece tan complicado crear un compilador, sólo se necesitan los conocimientos adecuados y dedicarle su tiempo para tener [éxito](http://www.monografias.com/trabajos15/llave-exito/llave-exito.shtml).

# Anexos

## 1) Anexo 1

import java\_cup.runtime.\*;

%%

%class Scanner

%unicode

%public

%ignorecase

/\*%debug\*/

%cup //para que funcione con el parser

%line

%column

%type Symbol

%scanerror Exception

//%function nextToken

%eofval{

return symbol(sym.EOF,new String("EOF"));

%eofval}

%{

StringBuffer string = new StringBuffer();

private Symbol symbol(int type) {

return new Symbol(type, yyline, yycolumn);

}

private Symbol symbol(int type, Object value) {

return new Symbol(type, yyline, yycolumn, value);

}

boolean errorScanner=false;

String datos="";

String datos2="";

int countComment=0;

%}

LineTerminator = \r|\n|\r\n

InputCharacter = [^\r\n]

WhiteSpace = {LineTerminator} | [ \t\f]

/\* comments \*/

Comment = {EndOfLineComment}

EndOfLineComment = "//" {InputCharacter}\* {LineTerminator}

Identifier = [:jletter:] [:jletterdigit:]\* | [$][:jletter:][:jletterdigit:]\* | \_[:jletter:][:jletterdigit:]\*

DecIntegerLiteral = 0 | 0.[0-9][0-9]\* | [1-9][0-9]\* | [1-9][0-9]\*.[0-9][0-9]\*

//Char = '[:jletter:]'

%state STRING

%state NESTEDCOMMENT

%%

/\* Palabras reservadas \*/

<YYINITIAL> "INT" { datos+="- Lexema:'"+yytext()+"' - Token:'INT':- Valor:'"+sym.TINT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'INT':- Valor:'"+sym.TINT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TINT,yytext()); }

<YYINITIAL> "STRING[]" { datos+="- Lexema:'"+yytext()+"' - Token:'STRING':- Valor:'"+sym.TSTRING+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'STRING':- Valor:'"+sym.TSTRING+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TSTRING,yytext()); }

<YYINITIAL> "BOOLEAN" { datos+="- Lexema:'"+yytext()+"' - Token:'BOOLEAN':- Valor:'"+sym.TBOOL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'BOOLEAN':- Valor:'"+sym.TBOOL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TBOOL,yytext()); }

/\*<YYINITIAL> "CHAR" {

datos+=- Lexema:'"+yytext()+"' - Token:'CHAR':- Valor:'"+sym.TCHAR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'CHAR':- Valor:'"+sym.TCHAR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TCHAR,yytext()); }

\*/

<YYINITIAL> "WHILE" { datos+="- Lexema:'"+yytext()+"' - Token:'WHILE':- Valor:'"+sym.CICLO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'WHILE':- Valor:'"+sym.CICLO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.CICLO,yytext());}

<YYINITIAL> "IMPORT" { datos+="- Lexema:'"+yytext()+"' - Token:'IMPORT':- Valor:'"+sym.IMPORT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'IMPORT':- Valor:'"+sym.IMPORT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.IMPORT,yytext());}

<YYINITIAL> "PUBLIC" { datos+="- Lexema:'"+yytext()+"' - Token:'PUBLIC':- Valor:'"+sym.PUBLIC+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'PUBLIC':- Valor:'"+sym.PUBLIC+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PUBLIC,yytext());}

<YYINITIAL> "STATIC" { datos+="- Lexema:'"+yytext()+"' - Token:'STATIC':- Valor:'"+sym.STATIC+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'STATIC':- Valor:'"+sym.STATIC+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.STATIC,yytext());}

<YYINITIAL> "VOID" { datos+="- Lexema:'"+yytext()+"' - Token:'VOID':- Valor:'"+sym.VOID+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'VOID':- Valor:'"+sym.VOID+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.VOID,yytext());}

<YYINITIAL> "MAIN" { datos+="- Lexema:'"+yytext()+"' - Token:'MAIN':- Valor:'"+sym.MAIN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'MAIN':- Valor:'"+sym.MAIN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MAIN,yytext());}

<YYINITIAL> "EXTENDS" { datos+="- Lexema:'"+yytext()+"' - Token:'EXTENDS':- Valor:'"+sym.EXTENDS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'EXTENDS':- Valor:'"+sym.EXTENDS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.EXTENDS,yytext());}

<YYINITIAL> "IMPLEMENTS" { datos+="- Lexema:'"+yytext()+"' - Token:'IMPLEMENTS':- Valor:'"+sym.IMPLEMENTS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'IMPLEMENTS':- Valor:'"+sym.IMPLEMENTS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.IMPLEMENTS,yytext());}

<YYINITIAL> "RETURN" { datos+="- Lexema:'"+yytext()+"' - Token:'RETURN':- Valor:'"+sym.RETURN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'RETURN':- Valor:'"+sym.RETURN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.RETURN,yytext());}

<YYINITIAL> "SYSTEM" { datos+="- Lexema:'"+yytext()+"' - Token:'SYSTEM':- Valor:'"+sym.SYSTEM+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'SYSTEM':- Valor:'"+sym.SYSTEM+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.SYSTEM,yytext());}

<YYINITIAL> "OUT" { datos+="- Lexema:'"+yytext()+"' - Token:'OUT':- Valor:'"+sym.OUT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'OUT':- Valor:'"+sym.OUT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.OUT,yytext());}

<YYINITIAL> "PRINTLN" { datos+="- Lexema:'"+yytext()+"' - Token:'PRINTLN':- Valor:'"+sym.PRINTLN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'PRINTLN':- Valor:'"+sym.PRINTLN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PRINTLN,yytext());}

<YYINITIAL> "CLASS" { datos+="- Lexema:'"+yytext()+"' - Token:'CLASS':- Valor:'"+sym.CLASS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'CLASS':- Valor:'"+sym.CLASS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol (sym.CLASS,yytext());}

<YYINITIAL> "IF" { datos+="- Lexema:'"+yytext()+"' - Token:'IF':- Valor:'"+sym.CONDI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'IF':- Valor:'"+sym.CONDI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.CONDI,yytext());}

<YYINITIAL> "ELSE" { datos+="- Lexema:'"+yytext()+"' - Token:'ELSE':- Valor:'"+sym.SINO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'ELSE':- Valor:'"+sym.SINO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.SINO,yytext());}

<YYINITIAL> "LENGTH" { datos+="- Lexema:'"+yytext()+"' - Token:'LENGTH':- Valor:'"+sym.LENGTH+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'LENGTH':- Valor:'"+sym.LENGTH+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.LENGTH,yytext());}

<YYINITIAL> "TRUE" { datos+="- Lexema:'"+yytext()+"' - Token:'TRUE':- Valor:'"+sym.TRUE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'TRUE':- Valor:'"+sym.TRUE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TRUE,yytext());}

<YYINITIAL> "FALSE" { datos+="- Lexema:'"+yytext()+"' - Token:'FALSE':- Valor:'"+sym.FALSE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'FALSE':- Valor:'"+sym.FALSE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.FALSE,yytext());}

<YYINITIAL> "THIS" { datos+="- Lexema:'"+yytext()+"' - Token:'THIS':- Valor:'"+sym.THIS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'THIS':- Valor:'"+sym.THIS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.THIS,yytext());}

<YYINITIAL> "NEW" { datos+="- Lexema:'"+yytext()+"' - Token:'NEW':- Valor:'"+sym.NEW+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'NEW':- Valor:'"+sym.NEW+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.NEW,yytext());}

<YYINITIAL> "IN" { datos+="- Lexema:'"+yytext()+"' - Token:'IN':- Valor:'"+sym.IN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'IN':- Valor:'"+sym.IN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.IN,yytext());}

<YYINITIAL> "READ" { datos+="- Lexema:'"+yytext()+"' - Token:'READ':- Valor:'"+sym.READ+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'READ':- Valor:'"+sym.READ+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.READ,yytext());}

<YYINITIAL> "EXIT" { datos+="- Lexema:'"+yytext()+"' - Token:'EXIT':- Valor:'"+sym.EXIT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'EXIT':- Valor:'"+sym.EXIT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.EXIT,yytext());}

<YYINITIAL> {

/\* identificadores \*/

{Identifier} { datos+="- Lexema:'"+yytext()+"' - Token:'ID':- Valor:'"+sym.ID+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'ID':- Valor:'"+sym.ID+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.ID,yytext()); }

/\* literales enteros positivos \*/

{DecIntegerLiteral} { datos+="- Lexema:'"+yytext()+"' - Token:'NUM':- Valor:'"+sym.NUM+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'NUM':- Valor:'"+sym.NUM+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.NUM,yytext()); }

\" { string.setLength(0); yybegin(STRING); }

/\*{Char} { datos+="- Lexema:'"+yytext()+"' - Token:'TCHAR':- Valor:'"+sym.TCHAR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'TCHAR':- Valor:'"+sym.TCHAR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.TCHAR,yytext()); }\*/

/\* operadores \*/

"+" { datos+="- Lexema:'"+yytext()+"' - Token:'+':- Valor:'"+sym.SUMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'+':- Valor:'"+sym.SUMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.SUMA,yytext()); }

"-" { datos+="- Lexema:'"+yytext()+"' - Token:'-':- Valor:'"+sym.REST+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'-':- Valor:'"+sym.REST+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.REST,yytext()); }

"/" { datos+="- Lexema:'"+yytext()+"' - Token:'/':- Valor:'"+sym.DIV+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'/':- Valor:'"+sym.DIV+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.DIV,yytext()); }

"\*" { datos+="- Lexema:'"+yytext()+"' - Token:'\*':- Valor:'"+sym.MULT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'\*':- Valor:'"+sym.MULT+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MULT,yytext()); }

"=" { datos+="- Lexema:'"+yytext()+"' - Token:'=':- Valor:'"+sym.ASIGN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'=':- Valor:'"+sym.ASIGN+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.ASIGN,yytext()); }

"!" { datos+="- Lexema:'"+yytext()+"' - Token:'!':- Valor:'"+sym.NEG+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'!':- Valor:'"+sym.NEG+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.NEG,yytext()); }

"!=" { datos+="- Lexema:'"+yytext()+"' - Token:'!=':- Valor:'"+sym.DIFE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'!=':- Valor:'"+sym.DIFE+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.DIFE,yytext()); }

"||" { datos+="- Lexema:'"+yytext()+"' - Token:'||':- Valor:'"+sym.OR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'||':- Valor:'"+sym.OR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.OR,yytext()); }

"&&" { datos+="- Lexema:'"+yytext()+"' - Token:'&&':- Valor:'"+sym.CLASS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'&&':- Valor:'"+sym.CLASS+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.AND,yytext()); }

"==" { datos+="- Lexema:'"+yytext()+"' - Token:'==':- Valor:'"+sym.COMPA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'==':- Valor:'"+sym.COMPA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.COMPA,yytext()); }

"<" { datos+="- Lexema:'"+yytext()+"' - Token:'<':- Valor:'"+sym.MENOR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'<':- Valor:'"+sym.MENOR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MENOR,yytext()); }

">" { datos+="- Lexema:'"+yytext()+"' - Token:'>':- Valor:'"+sym.MAYOR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'>':- Valor:'"+sym.MAYOR+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MAYOR,yytext()); }

"<=" { datos+="- Lexema:'"+yytext()+"' - Token:'<=':- Valor:'"+sym.MENORIGUAL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'<=':- Valor:'"+sym.MENORIGUAL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MENORIGUAL,yytext()); }

">=" { datos+="- Lexema:'"+yytext()+"' - Token:'=>':- Valor:'"+sym.MAYORIGUAL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'=>':- Valor:'"+sym.MAYORIGUAL+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.MAYORIGUAL,yytext()); }

/\* otros simbolos válidos \*/

"(" { datos+="- Lexema:'"+yytext()+"' - Token:'(':- Valor:'"+sym.PARENRedI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'(':- Valor:'"+sym.PARENRedI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENRedI,yytext()); }

")" { datos+="- Lexema:'"+yytext()+"' - Token:')':- Valor:'"+sym.PARENRedD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:')':- Valor:'"+sym.PARENRedD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENRedD,yytext()); }

"[" { datos+="- Lexema:'"+yytext()+"' - Token:'[':- Valor:'"+sym.PARENCuaI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'[':- Valor:'"+sym.PARENCuaI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENCuaI,yytext()); }

"]" { datos+="- Lexema:'"+yytext()+"' - Token:']':- Valor:'"+sym.PARENCuaD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:']':- Valor:'"+sym.PARENCuaD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENCuaD,yytext()); }

"{" { datos+="- Lexema:'"+yytext()+"' - Token:'{':- Valor:'"+sym.PARENCurI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'{':- Valor:'"+sym.PARENCurI+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENCurI,yytext()); }

"}" { datos+="- Lexema:'"+yytext()+"' - Token:'}':- Valor:'"+sym.PARENCurD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'}':- Valor:'"+sym.PARENCurD+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PARENCurD,yytext()); }

"." { datos+="- Lexema:'"+yytext()+"' - Token:'.':- Valor:'"+sym.PUNTO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'.':- Valor:'"+sym.PUNTO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PUNTO,yytext()); }

";" { datos+="- Lexema:'"+yytext()+"' - Token:';':- Valor:'"+sym.PUNTOyCOMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:';':- Valor:'"+sym.PUNTOyCOMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PUNTOyCOMA,yytext()); }

"," { datos+="- Lexema:'"+yytext()+"' - Token:',':- Valor:'"+sym.COMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:',':- Valor:'"+sym.COMA+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.COMA,yytext()); }

".\*" { datos+="- Lexema:'"+yytext()+"' - Token:'.\*':- Valor:'"+sym.PUNTOyASTERISCO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'.\*':- Valor:'"+sym.PUNTOyASTERISCO+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.PUNTOyASTERISCO,yytext()); }

"/\*" { countComment++;yybegin(NESTEDCOMMENT);}

/\* commentarios \*/

{Comment} { /\* ignore \*/ } /\*ya no se hace porque se realiza abajo \*/

/\* espacios en blanco \*/

{WhiteSpace} { /\* ignore \*/ }

}

<STRING> {

\" { yybegin(YYINITIAL);

datos+="- Lexema:'"+yytext()+"' - Token:'STRING':- Valor:'"+sym.STRING+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

System.out.println("- Lexema:'"+yytext()+"' - Token:'STRING':- Valor:'"+sym.STRING+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

return symbol(sym.STRING, string.toString()); }

[^\n\r\"\\]+ { string.append( yytext() ); }

\\t { string.append('\t'); }

\\n { string.append('\n'); }

\\r { string.append('\r'); }

\\\" { string.append('\"'); }

\\ { string.append('\\'); }

}

<NESTEDCOMMENT>{

"/\*" {countComment++;}

"\*/" {countComment--;if(countComment==0){yybegin(YYINITIAL);}}

[^] {/\*Ignore\*/}

}

/\* caracteres no válidos \*/

.|\n {datos2+="\n[Error Scanner] Caracter no permitido '"+yytext()+"' - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n";

/\*System.out.println("Error caracter inválido: <" + yytext() + "> en fila: " + yyline + " columna: " + yycolumn );\*/

errorScanner=true;

//throw new Error("\n[Error Scanner] Caracter no permitido '"+yytext()+" - Fila:'"+yyline+" - Columna:'"+yycolumn+"'\n");

}

## 2) Anexo 2

import java\_cup.runtime.\*;

import AST.\*;

parser code {:

Scanner scanner;

AAST raiz; //Raiz para la creacion del arbol

public parser(java.io.Reader input){

scanner = new Scanner(input);

}

public boolean errorState=false;

public String datos3="";

public String errores(int sys) //Detecta si se encontro algun error y lo identifica por el symbolo

{

errorState=true;

if (sys == 2)

return("NUM");

else if (sys == 3)

return("COMA");

else if (sys == 4)

return("PUNTO");

else if (sys == 5)

return("PUNTOyCOMA");

else if (sys == 6)

return("PUNTOyASTERISCO");

else if (sys == 7)

return("IMPORT");

else if (sys == 8)

return("PUBLIC");

else if (sys == 9)

return("STATIC");

else if (sys == 10)

return("VOID");

else if (sys == 11)

return("MAIN");

else if (sys == 12)

return("EXTENDS");

else if (sys == 13)

return("IMPLEMENTS");

else if (sys == 14)

return("RETURN");

else if (sys == 15)

return("SYSTEM");

else if (sys == 16)

return("OUT");

else if (sys == 17)

return("PRINTLN");

else if (sys == 18)

return("CLASS");

else if (sys == 19)

return("CICLO");

else if (sys == 20)

return("CONDI");

else if (sys == 21)

return("SINO");

else if (sys == 22)

return("IN");

else if (sys == 23)

return("READ");

else if (sys == 24)

return("EXIT");

else if (sys == 25)

return("LENGTH");

else if (sys == 26)

return("TRUE");

else if (sys == 27)

return("FALSE");

else if (sys == 28)

return("THIS");

else if (sys == 29)

return("NEW");

else if (sys == 30)

return("SUMA");

else if (sys == 31)

return("REST");

else if (sys == 32)

return("DIV");

else if (sys == 33)

return("MUL");

else if (sys == 34)

return("ASIGN");

else if (sys == 35)

return("OR");

else if (sys == 36)

return("AND");

else if (sys == 37)

return("NEG");

else if (sys == 38)

return("COMPA");

else if (sys == 39)

return("DIFE");

else if (sys == 40)

return("MENOR");

else if (sys == 41)

return("MENORIGUAL");

else if (sys == 42)

return("MAYORIGUAL");

else if (sys == 43)

return("MAYOR");

else if (sys == 44)

return("STRING");

else if (sys == 45)

return("ID");

else if (sys == 46)

return("TINT");

else if (sys == 47)

return("TSTRING");

else if (sys == 48)

return("TBOOL");

else if (sys == 49)

return("PARENRedD");

else if (sys == 50)

return("PARENRedI");

else if (sys == 51)

return("PARENCuaD");

else if (sys == 52)

return("PARENCuaI");

else if (sys == 53)

return("PARENCurD");

else if (sys == 54)

return("PARENCurI");

else

return("error");

}

public void syntax\_error(Symbol cur\_token2) //en error muestra lo que se espera que leyera

{

if (cur\_token.sym == 2)

report\_error("Se esperaba un Entero", cur\_token2);

else if (cur\_token.sym == 3)

report\_error("Se esperaba un Coma", cur\_token2);

else if (cur\_token.sym == 4)

report\_error("Se esperaba una Punto", cur\_token2);

else if (cur\_token.sym == 5)

report\_error("Se esperaba una PuntoyComa", cur\_token2);

else if (cur\_token.sym == 6)

report\_error("Se esperaba una PuntoyAsterisco", cur\_token2);

else if (cur\_token.sym == 7)

report\_error("Se esperaba parentesis Import", cur\_token2);

else if (cur\_token.sym == 8)

report\_error("Se esperaba parentesis Public", cur\_token2);

else if (cur\_token.sym == 9)

report\_error("Se esperaba un punto y Static", cur\_token2);

else if (cur\_token.sym == 10)

report\_error("Se esperaba una Void", cur\_token2);

else if (cur\_token.sym == 11)

report\_error("Se esperaba un Main", cur\_token2);

else if (cur\_token.sym == 12)

report\_error("Se esperaba una Extends", cur\_token2);

else if (cur\_token.sym == 13)

report\_error("Se esparaba un Implements", cur\_token2);

else if (cur\_token.sym == 14)

report\_error("Se esperaba un Return", cur\_token2);

else if (cur\_token.sym == 15)

report\_error("Se esperaba un System", cur\_token2);

else if (cur\_token.sym == 16)

report\_error("Se esperaba un Out", cur\_token2);

else if (cur\_token.sym == 17)

report\_error("Se esperaba un Println", cur\_token2);

else if (cur\_token.sym == 18)

report\_error("Se esperaba un Class", cur\_token2);

else if (cur\_token.sym == 19)

report\_error("Se esperaba un Ciclo", cur\_token2);

else if (cur\_token.sym == 20)

report\_error("Se esperaba un Condi", cur\_token2);

else if (cur\_token.sym == 21)

report\_error("Se esperaba un Sino", cur\_token2);

else if (cur\_token.sym == 22)

report\_error("Se esperaba un In", cur\_token2);

else if (cur\_token.sym == 23)

report\_error("Se esperaba un Read", cur\_token2);

else if (cur\_token.sym == 24)

report\_error("Se esperaba un Exit", cur\_token2);

else if (cur\_token.sym == 25)

report\_error("Se esperaba un Length", cur\_token2);

else if (cur\_token.sym == 26)

report\_error("Se esperaba un True", cur\_token2);

else if (cur\_token.sym == 27)

report\_error("Se esperaba un False", cur\_token2);

else if (cur\_token.sym == 28)

report\_error("Se esperaba un This", cur\_token2);

else if (cur\_token.sym == 29)

report\_error("Se esperaba un New", cur\_token2);

else if (cur\_token.sym == 30)

report\_error("Se esperaba un Suma", cur\_token2);

else if (cur\_token.sym == 31)

report\_error("Se esperaba un Rest", cur\_token2);

else if (cur\_token.sym == 32)

report\_error("Se esperaba un Div", cur\_token2);

else if (cur\_token.sym == 33)

report\_error("Se esperaba un Mul", cur\_token2);

else if (cur\_token.sym == 34)

report\_error("Se esperaba un Asign", cur\_token2);

else if (cur\_token.sym == 35)

report\_error("Se esperaba un Or", cur\_token2);

else if (cur\_token.sym == 36)

report\_error("Se esperaba un And", cur\_token2);

else if (cur\_token.sym == 37)

report\_error("Se esperaba un Neg", cur\_token2);

else if (cur\_token.sym == 38)

report\_error("Se esperaba un Compa", cur\_token2);

else if (cur\_token.sym == 39)

report\_error("Se esperaba un Dife", cur\_token2);

else if (cur\_token.sym == 40)

report\_error("Se esperaba un Menor", cur\_token2);

else if (cur\_token.sym == 41)

report\_error("Se esperaba un Menorigual", cur\_token2);

else if (cur\_token.sym == 42)

report\_error("Se esperaba un Mayorigual", cur\_token2);

else if (cur\_token.sym == 43)

report\_error("Se esperaba un Mayor", cur\_token2);

else if (cur\_token.sym == 44)

report\_error("Se esperaba un String", cur\_token2);

else if (cur\_token.sym == 45)

report\_error("Se esperaba un Id", cur\_token2);

else if (cur\_token.sym == 46)

report\_error("Se esperaba un TINT", cur\_token2);

else if (cur\_token.sym == 47)

report\_error("Se esperaba un TSRING", cur\_token2);

else if (cur\_token.sym == 48)

report\_error("Se esperaba un TBOOL", cur\_token2);

else if (cur\_token.sym == 49)

report\_error("Se esperaba un Parentesis Redondo Izquierdo", cur\_token2);

else if (cur\_token.sym == 50)

report\_error("Se esperaba un Parentesis Redondo Derecho", cur\_token2);

else if (cur\_token.sym == 51)

report\_error("Se esperaba un Parentesis Cuadrado Derecho", cur\_token2);

else if (cur\_token.sym == 52)

report\_error("Se esperaba un Parentesis Cuadrado Izquierdo", cur\_token2);

else if (cur\_token.sym == 53)

report\_error("Se esperaba un Corchete Derecho", cur\_token2);

else if (cur\_token.sym == 54)

report\_error("Se esperaba un Corchete Izquierdo", cur\_token2);

else

report\_error("Error: ", cur\_token2);

}

public void report\_error(String message, Object info) { //Muestra elerror y la exception del un error de cup

StringBuffer m = new StringBuffer("Error ");

if (info instanceof java\_cup.runtime.Symbol) {

m.append( "(" + errores(cur\_token.sym) +")" );

m.append(" en fila "+ cur\_token.left + " columna " + cur\_token.right );

if (((java\_cup.runtime.Symbol)info).value != null)

m.append(". Lexema: " + cur\_token.value.toString());

}

m.append(" : "+message);

m.append(" y en su lugar viene " + ((java\_cup.runtime.Symbol)info).value.toString());

datos3+="\n"+m.toString();

System.out.println(m);

}

public void report\_fatal\_error(String message, Object info) {

report\_error(message, info);

throw new RuntimeException("Error Fatal de Sintaxis!!!"+info.value);

}

:}

///////////// Lista de tokens terminales

//terminal Integer NUM;

terminal COMA,PUNTO, PUNTOyCOMA,PUNTOyASTERISCO,IMPORT,PUBLIC,STATIC,VOID,MAIN,EXTENDS,IMPLEMENTS, RETURN, SYSTEM,OUT,PRINTLN,CLASS,CICLO,CONDI,SINO,IN,READ,EXIT,LENGTH,TRUE,FALSE,THIS,NEW,SUMA,REST,DIV,MULT,ASIGN,OR,AND,NEG,COMPA,DIFE,MENOR,MENORIGUAL,MAYORIGUAL,MAYOR;

//terminal String STRING, ID;

terminal TINT,TSTRING,TBOOL,PARENRedI,PARENRedD,PARENCuaD,PARENCuaI,PARENCurD,PARENCurI;

//terminal Symbol STRING,ID,NUM;

terminal String STRING,ID,NUM;

///////////// Lista de tokens no terminales

non terminal AProgramAST program;

non terminal AVarImportDeclAST varimportdecl;

non terminal AImportDeclAST importdecl;

non terminal AImportDecl2AST importdecl2;

non terminal AVarClassDeclAST varclassdecl;

non terminal AClassDeclAST classdecl;

non terminal AClassDecl2AST classdecl2;

non terminal AClassDecl3AST classdecl3;

non terminal AClassDecl4AST classdecl4;

non terminal ATypeNameAST typename;

non terminal ATypeName2AST typename2;

non terminal AMainClassAST mainclass;

non terminal ABodyDeclAST bodydecl;

non terminal ABodyDecl2AST bodydecl2;

non terminal AVarDeclAST vardecl;

non terminal AConstrDeclAST constrdecl;

non terminal AVarVarDeclAST varvardecl;

non terminal AVarStatementAST varstatement;

non terminal AMethodDeclAST methodecl;

non terminal ATypeVoidAST typevoid;

non terminal AReturnExpAST returnexp;

non terminal AFormalListAST formallist;

non terminal AVarTypeIdAST vartypeid;

/\*type, \*/

non terminal AStatementAST statement;

non terminal AElseStatementAST elsestatement;

non terminal ACuadExpAST cuadexpe;

non terminal AExpTypeAST exptype;

non terminal AThisIdAST thisid;

non terminal AExpAST exp;

non terminal AVarDistiExpAST vardistiexp;

non terminal ADistiExpAST distiexp;

non terminal AExp\_SimplAST exp\_simpl;

non terminal AExpListAST explist;

non terminal AVarComaExpAST varcomaexp;

non terminal AOpAST op;

/////////////Precendence para evitar errores en el cup

precedence left ID;

precedence left OR, AND;

precedence left MENOR, MAYOR, MENORIGUAL, MAYORIGUAL, DIFE, COMPA;

precedence left SUMA, REST;

precedence left MULT, DIV;

precedence left PUNTO;

precedence left PARENCuaI;

precedence left SINO;

start with program;

// Regla 1)

program ::= varimportdecl:i varclassdecl:c mainclass:m

{:

parser.raiz = new ImpClssMnAST(i,c,m);

:}

;

//Regla 2)

varimportdecl ::= importdecl:d varimportdecl:s

{:

RESULT= new VarImportDeclAST(d,s);

:}

|

{:

RESULT = null;

:}

;

// Regla 3)

importdecl ::= IMPORT typename:tn importdecl2:imp1 PUNTOyCOMA

{:

RESULT= new ImportDeclAST(tn,imp1);

:}

;

// Regla 4)

importdecl2 ::= PUNTOyASTERISCO

{:

RESULT= new ImportDecl2AST();

:}

|

{:

RESULT= null;

:}

;

// Regla 5)

varclassdecl ::= varclassdecl:cds classdecl:cd

{:

RESULT= new VarClassDeclAST(cd,cds);

:}

|

{:

RESULT= null;

:}

;

// Regla 6)

classdecl ::= classdecl2:cd1 PARENCurI bodydecl:bd PARENCurD

{:

RESULT= new ClassDeclBodyAST(cd1,bd);

:}

;

// Regla 7)

classdecl2::= CLASS ID:id1 classdecl3:cd2

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new ClassDecl2AST(ident1,cd2);

:}

|

{:

RESULT= null;

:}

;

// Regla 8)

classdecl3::= classdecl4:cd3 ID:id1

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new ClassDecl3AST(ident1,cd3);

:}

;

// Regla 9) --------FALTA-----

classdecl4::= EXTENDS

{:

RESULT= new ClassDecl4ExtendsAST();

:}

|

IMPLEMENTS

{:

RESULT= new ClassDecl4ImplementsAST();

:}

;

// Regla 10)

typename ::= ID:id1 typename2:tn

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new TypeNameAST(ident1,tn);

:}

;

// Regla 11)

typename2 ::= typename2:tns PUNTO ID:id1

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new TypeName2AST(ident1,tns);

:}

|

{:

RESULT= null;

:}

;

// Regla 12)

mainclass ::= CLASS ID:id1 PARENCurI PUBLIC STATIC VOID MAIN PARENRedI TSTRING ID:id2 PARENRedD PARENCurI statement:s PARENCurD PARENCurD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

Symbol ident2 = new Symbol(sym.ID,id2left,id2right,id2);

RESULT= new MainClassAST(ident1,ident2,s); //t,i2,s);

:}

;

// Regla 13)

bodydecl ::= bodydecl2:bd1 bodydecl:bd

{:

RESULT= new BodyDeclAST(bd1,bd);

:}

|

{:

RESULT= null;

:}

;

// Regla 14)

bodydecl2 ::= vardecl:vd

{:

RESULT= new BodyDecl2VarDeclAST(vd);

:}

| constrdecl:cd

{:

RESULT= new BodyDecl2ConstDeclAST(cd);

:}

| methodecl:md

{:

RESULT= new BodyDecl2MethoDeclAST(md);

:}

| classdecl:cd2

{:

RESULT= new BodyDecl2ClassDeclAST(cd2);

:}

;

// Regla 15)

vardecl ::= ID:id1 ID:id2 PUNTOyCOMA

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

Symbol ident2 = new Symbol(sym.ID,id2left,id2right,id2);

RESULT= new VarDeclAST(ident1,ident2);

:}

;

// Regla 16)

constrdecl ::= PUBLIC ID:id1 PARENRedI formallist:fl PARENRedD PARENCurI varvardecl:vd varstatement:s PARENCurD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new ConstrDeclAST(ident1,fl,vd,s);

:}

;

// Regla 17)

varvardecl ::= varvardecl:vds vardecl:vd

{:

RESULT= new VarVarDeclAST(vd,vds);

:}

|

{:

RESULT= null;

:}

;

// Regla 18)

varstatement ::= varstatement:ss statement:s

{:

RESULT= new VarStatementAST(s,ss);

:}

|

{:

RESULT= null;

:}

;

// Regla 19)

methodecl ::= PUBLIC typevoid:iv ID:id1 PARENRedI formallist:fl PARENRedD PARENCurI varvardecl:vd varstatement:s returnexp:r PARENCurD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new MethodDeclAST(ident1,iv,fl,vd,s,r);

:}

;

// Regla 20) --------FALTA-----

typevoid::= ID:id1

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new TypeVoidIdAST(ident1);

:}

| VOID

{:

RESULT= new TypeVoidVoidAST();

:}

;

// Regla 21)

returnexp::= RETURN exp:e PUNTOyCOMA

{:

RESULT= new ReturnExpAST(e);

:}

|

{:

RESULT= null;

:}

;

// Regla 22)

formallist ::= ID:id1 ID:id2 vartypeid:fls

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

Symbol ident2 = new Symbol(sym.ID,id2left,id2right,id2);

RESULT= new FormalListAST(ident1,ident2,fls);

:}

|

{:

RESULT= null;

:}

;

// Regla 23)

vartypeid ::= vartypeid:fls COMA ID:id1 ID:id2

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

Symbol ident2 = new Symbol(sym.ID,id2left,id2right,id2);

RESULT= new VarTypeIdAST(ident1,ident2,fls);

:}

|

{:

RESULT= null;

:}

;

/\*// Regla 24)

type ::= TINT PARENCuaI PARENCuaD

| TBOOL

| TINT

//| ID

;

\*/

// Regla 24)

statement ::= ID:id1 ASIGN exptype:ed PUNTOyCOMA

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new StatAsigAST(ident1,ed);

:}

| thisid:t PUNTO ID:id1 PARENRedI explist:el PARENRedD PUNTOyCOMA

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new StatExpListAST(ident1,t,el);

:}

| ID:id1 cuadexpe:ce ASIGN exptype:ed PUNTOyCOMA

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new StatCuadExpAST(ident1,ce,ed);

:}

| SYSTEM PUNTO EXIT PARENRedI NUM:n PARENRedD PUNTOyCOMA

{:

Symbol num = new Symbol(sym.NUM,nleft,nright,n);

RESULT= new StatExitAST(num);

:}

| PARENCurI varstatement:s PARENCurD

{:

RESULT= new StatVarStatAST(s);

:}

| CONDI PARENRedI exp:e PARENRedD statement:s elsestatement:es

{:

RESULT= new StatIfElseAST(e,s,es);

:}

| CICLO PARENRedI exp:e PARENRedD statement:s

{:

RESULT= new StatWhileAST(e,s);

:}

| SYSTEM PUNTO OUT PUNTO PRINTLN PARENRedI exp:e PARENRedD PUNTOyCOMA

{:

RESULT= new StatPrintlnAST(e);

:}

;

// Regla 25)

elsestatement::= SINO statement:s

{:

RESULT= new ElseStatementAST(s);

:}

|

{:

RESULT= null;

:}

;

// Regla 26)

cuadexpe::= PARENCuaI exp:e PARENCuaD

{:

RESULT= new CuadExpAST(e);

:}

;

// Regla 27)

exptype::= exp:e {:

RESULT= new ExpTypeExpAST(e);

:}

| PARENRedI ID:id1 PARENRedD SYSTEM PUNTO IN PUNTO READ PARENRedI PARENRedD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new ExpTypeReadAST(ident1);

:}

;

// Regla 28) --------FALTA-----

thisid::= THIS

{:

RESULT= new ThisIdThisAST(); ////////////////////////Estosssssssssssssssssssssssssssssssss

:}

| ID:id1

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new ThisIdIdAST(ident1); ////////////////////////Estosssssssssssssssssssssssssssssssss

:}

;

// Regla 29)

exp ::= exp\_simpl:es vardistiexp:ei

{:

RESULT= new ExpAST(es,ei);

:}

;

// Regla 30)

vardistiexp ::= vardistiexp:e distiexp :t

{:

RESULT= new VarDistiExpAST(t,e);

:}

|

{:

RESULT= null;

:}

;

// Regla 31)

distiexp ::= PUNTO ID:id1 PARENRedI explist:el PARENRedD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new DistiExpListAST(ident1,el);

:}

| op:o exp:e

{:

RESULT= new DistiExpOpAST(o,e);

:}

| PARENCuaI exp:e PARENCuaD

{:

RESULT= new DistiExpCuadrAST(e);

:}

| PUNTO LENGTH

{:

RESULT= new DistiExpLengthAST();//(p,l);

:}

;

// Regla 32)

exp\_simpl ::= ID:id1

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new Exp\_SimpIdAST(ident1);;

:}

| TRUE

{:

RESULT= new Exp\_SimpTrueAST();

:}

| FALSE

{:

RESULT= new Exp\_SimpFalseAST();

:}

| NUM:n

{:

Symbol num = new Symbol(sym.ID,nleft,nright,n);

RESULT= new Exp\_SimpNumAST(num);

:}

| THIS

{:

RESULT= new Exp\_SimpThisAST();

:}

| NEW TINT PARENCuaI exp:e PARENCuaD

{:

RESULT= new Exp\_SimpNewIntAST(e);

:}

| NEW ID:id1 PARENRedI explist:e PARENRedD

{:

Symbol ident1 = new Symbol(sym.ID,id1left,id1right,id1);

RESULT= new Exp\_SimpNewIdAST(ident1,e);

:}

| NEG exp:e

{:

RESULT= new Exp\_SimpNegAST(e);

:}

| PARENRedI exp:e PARENRedD

{:

RESULT= new Exp\_SimpExpAST(e);

:}

| STRING:s

{:

Symbol stri = new Symbol(sym.STRING,sleft,sright,s);

RESULT= new Exp\_SimpStringAST(stri);

:}

;

// Regla 33)

explist ::= exp:e varcomaexp:l

{:

RESULT= new ExpListAST(e,l);

:}

|

{:

RESULT= null;

:}

;

// Regla 34)

varcomaexp ::= varcomaexp:l COMA exp:e

{:

RESULT= new VarComaExpAST(e,l);

:}

|

{:

RESULT= null;

:}

;

// Regla 35) --------FALTA-----

op ::= SUMA

{:

RESULT= new OpSumAST();

:}

| REST

{:

RESULT= new OpRestAST();

:}

| MULT

{:

RESULT= new OpMulAST();

:}

| DIV

{:

RESULT= new OpDivAST();

:}

| DIFE

{:

RESULT= new OpDifeAST();

:}

| COMPA

{:

RESULT= new OpCompaAST();

:}

| MENOR

{:

RESULT= new OpMenorAST();

:}

| MENORIGUAL

{:

RESULT= new OpMenorIgualAST();

:}

| MAYORIGUAL

{:

RESULT= new OpMayorIgualAST();

:}

| MAYOR

{:

RESULT= new OpMayorAST();

:}

| OR

{:

RESULT= new OpOrAST();

:}

| AND

{:

RESULT= new OpAndAST();

:}

;