Instituto Tecnológico de Costa Rica

Escuela de Computación
Compiladores e Intérpretes

Daniel Berrocal Ramírez
201049486

Jorge Rojas Aragonés
200969830

San Carlos, 19 de Abril de 2013

Tabla de contenido

Objetivo General:

- Por medio del análisis de una gramática BNF, realizar la implementación de la primera etapa de un compilador funcional para el lenguaje JAVA.

Objetivos Específicos:

- Aplicar conceptos de análisis sintáctico y semántico en el proceso de desarrollo.
- Generar una herramienta que canalice todos los conceptos estudiados del tema.

Análisis del Lenguaje:

Se solicita realizar la primera fase de un compilador para una gramática MiniJava, por medio de esta gramática el usuario podrá escribir código java, debido a que no es la gramática total de java el usuario no cuenta con todas las bondades del lenguaje no obstante la gramática MiniJava es suficiente para crear un programa con sus respectivas clases. Desarrollar un compilador para lenguaje java en lenguaje java brinda ciertas características beneficiosas para la implementación final, para mencionar, aumenta considerablemente la portabilidad del compilador.

Los programas que el usuario final podrá escribir tendrán ciertas características propias del lenguaje java, por ejemplo, código limpio, legible y ordenado; podrá crear desde funciones simples hasta interfaces graficas de usuario, también podrá implementar herencia entre otras cualidades que permita la gramática MiniJava.

Soluciones e Implementación:

Para llevar a cabo la elaboración de este proyecto, se pasó por varias etapas de desarrollo, esto para no recargar la labor a realizar en un solo punto, en conjunto se decidió dividir el proyecto en varias etapas, pensando en un mejor resultado al final, cada etapa se realizó en un tiempo considerable dependiendo de la etapa que se tratase, a continuación se describe cada una de las etapas:

Investigación:

En esta etapa se realizó una intensa búsqueda de todo tipo de información que proporciona algún apoyo para la confección del sistema. Se investigó sobre la gramática y ejemplos de archivos JFLEX y CUP para tener una idea más clara de la labor a realizar.

Planteo de la Solución:

En dicha etapa, con toda la información que se recaudó en la etapa de investigación, se procede a plantear posibles maneras de llegar a una solución, discutiendo, probando, y elaborando pequeñas pruebas de código, mutuamente se llegó a la decisión de cuál posible solución implementar.

Desarrollo:

Teniendo en mente de qué manera darle solución al problema, se inicia con el desarrollo de la solución, en esta etapa se implementa todo el conocimiento adquirido en clase y la información recaudada en la etapa de investigación. Como primer paso, se realiza la interfaz gráfica del compilador y se genera el scanner con la herramienta JFLEX, luego se le realizaron los cambios necesarios a la gramática proporcionada esto para que fuese compatible con la herramienta CUP y se nos facilitara el trabajo al generar parser, por último se

realizan los cambios indispensables al archivo .CUP para volver a generar parser, esto para generar AST.

Prueba:

En esta etapa, se le realizaron distintas pruebas al sistema para buscar todo tipo de errores y generar las posibles soluciones para dichos errores.

Al final se adjuntan los archivos de configuración JFLEX y CUP para su correspondiente revisión.

Resultados Obtenidos:

Durante todo el proceso de desarrollo del Proyecto se generaron ciertos resultados, algunos esperados y otros no muy esperados, ya que el desarrollo se dio en etapas marcadas por lapsos de tiempos reducidos a una semana cada lapso.
Para cada etapa se generaron resultados diferentes que afectaron el desarrollo.

Primera Etapa: Scanner y Editor Funcionales

En esta primera etapa se realizaron con éxito total todos los objetivos correspondientes a la etapa, el scanner se generó utilizando la herramienta JFLEX y el editor se desarrolló en NetBeans, la interfaz gráfica es muy sencilla para facilitar el uso de la misma.

Segunda Etapa: Parser Funcional

Esta etapa presento ciertos problemas que evitaron la entrega del avance correspondiente a la fecha pactada, los problemas se dieron en la confección del archivo .CUP, esto debido a problemas en la traducción en la gramática y en la modificación de las reglas. Dichos problemas fueron solucionados con el inconveniente del atraso en el tiempo de entrega.

Tercera Etapa: AST Funcional

Debido a los problemas que se dieron en la segunda etapa y el tiempo que se invirtió en solucionar dichos problemas, generaron un atraso considerable en el tiempo de desarrollo y en el lapso que se tenía para entregar AST, por esta razón no se logró entregar a tiempo el avance correspondiente.

Conclusiones:

Durante el desarrollo de este proyecto, se logran rescatar muchos aspectos importantes para cualquier estudiante de la carrera de ingeniería en computación, pero el que más importancia parece tener es, el impacto a nivel académico, que se da cuando se analizan los procesos que lleva a cabo un compilador, una herramienta que como programadores utilizamos a diario y nunca nos detenemos y pensamos analíticamente su funcionamiento. Por esta razón es importante la asignación de este tipo de proyectos en los cuales se inculca una manera de pensar analítica, ayudando esto a la formación de futuros ingenieros profesionales.

Bibliografía:

JFlex User's Manual (Enero 31, 2009). En *The Fast Lexical Analyser Generator.* Consultado el 28 de Marzo del 2013, en http://jflex.de/manual.html

Breve Introducción a CUP (Septiembre 2 del 2012). En *Universidad Carlos III de Madrid*. Consultado el 4 de Abril del 2013, en http://www.it.uc3m.es/~luis/fo1/p/CUP.html

Abstract Syntax Tree (Noviembre 20 del 2006). En *Eclipse*. Consultado el 14 de Abril del 2013, en http://www.eclipse.org/articles/Article-JavaCodeManipulation_AST/index.html

Anexos:

Contenido del archivo JFLEX:

```
/**
 * Scanner para la gramática de prueba del curso de Compiladores e Intérpretes.
 */
%%

%class Scanner
%unicode
%debug
%line
%column
%type Symbol // cambiar nombre del token
%function nextToken // funcion principal

/*%eofval{
        return symbol(sym.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);
 }
%}

LineTerminator = \r|\n|\r\n
InputCharacter = [^\r\n]
WhiteSpace     = {LineTerminator} | [ \t\f]

/* comments */
Comment = {TraditionalComment} | {EndOfLineComment}

TraditionalComment = "/*" [^*] ~"*/" | "/*" "*"+ "/"
```

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

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


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

%state STRING

%%

/* Palabras reservadas */

<YYINITIAL> "INT"             { return symbol(sym.INT); }
<YYINITIAL> "STRING"          { return symbol(sym.STRING); }
<YYINITIAL> "boolean"         { return symbol(sym.BOOLEAN); }
<YYINITIAL> "Class"           { return symbol(sym.Class); }
<YYINITIAL> "else"            { return symbol(sym.Else); }
<YYINITIAL> "if"              { return symbol(sym.If); }
<YYINITIAL> "true"            { return symbol(sym.True); }
<YYINITIAL> "false"           { return symbol(sym.False); }
<YYINITIAL> "while"           { return symbol(sym.While); }
<YYINITIAL> "main"            { return symbol(sym.Main); }
<YYINITIAL> "public"          { return symbol(sym.Public); }
<YYINITIAL> "static"          { return symbol(sym.Static); }
<YYINITIAL> "extends"         { return symbol(sym.Extends); }
<YYINITIAL> "length"          { return symbol(sym.Lenght); }
<YYINITIAL> "INTEGER_LITERAL"  { return symbol(sym.INTEGER_LITERAL); }
<YYINITIAL> "STRING_CONSTANT"  { return symbol(sym.STRING_CONSTANT); }
<YYINITIAL> "new"             { return symbol(sym.New); }
<YYINITIAL> "out"             { return symbol(sym.Out); }
<YYINITIAL> "println"         { return symbol(sym.Println); }
<YYINITIAL> "return"          { return symbol(sym.Return); }
<YYINITIAL> "system"          { return symbol(sym.System); }
<YYINITIAL> "this"            { return symbol(sym.This); }
<YYINITIAL> "void"            { return symbol(sym.Void); }
<YYINITIAL> "import"          { return symbol(sym.Import); }
<YYINITIAL> "implemets"       { return symbol(sym.Implements); }
<YYINITIAL> "exit"            { return symbol(sym.Exit); }
<YYINITIAL> "in"              { return symbol(sym.In); }
<YYINITIAL> "read"            { return symbol(sym.Read); }
```

```
<YYINITIAL> {
  /* identificadores */
  {Identifier}                { return symbol(sym.ID,yytext()); }}

  /* literales enteros positivos */

  {DecIntegerLiteral}          { return symbol(sym.NUM); }
  \"                          { string.setLength(0); yybegin(STRING); }

  /* operadores */

  "="                          { return symbol(sym.Igual_Asig); }
  "*"                         { return symbol(sym.MULT); }
  "/"                         { return symbol(sym.Div); }
  "+"                          { return symbol(sym.Suma); }
  "-"                         { return symbol(sym.Resta); }

  /* operadores logicos*/

  "||"                        { return symbol(sym.OR); }
  "&&"                           { return symbol(sym.AND); }
  "!"                         { return symbol(sym.Negacion); }

  /* operadores condicionales*/

  "!="                         { return symbol(sym.Diferencia); }
  "=="                          { return symbol(sym.Igual_Comp); }
  "<"                         { return symbol(sym.Menor); }
  ">"                         { return symbol(sym.Mayor); }
  "<="                          { return symbol(sym.Menor_Igual); }
  ">="                          { return symbol(sym.Mayor_Igual); }

  /* otros simbolos válidos */

  "("                         { return symbol(sym.PARENT_on); }
  ")"                         { return symbol(sym.PARENT_off); }
  "["                         { return symbol(sym.Corchete_on); }
  "]"                         { return symbol(sym.Corchete_off); }
  "{"                         { return symbol(sym.Llave_on); }
```

```
  "}"                   { return symbol(sym.Llave_off); }
  ";"                   { return symbol(sym.DOTCOMMA); }
  ","                   { return symbol(sym.COMA); }
  "."                   { return symbol(sym.Punto); }


  /* commentarios */
  {Comment}                 { /* ignore */ }

  /* espacios en blanco */
  {WhiteSpace}              { /* ignore */ }


<STRING> {
  \"                    { yybegin(YYINITIAL);
                          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('\\'); }
}

/* caracteres no válidos */
.|\n                        { System.out.println("Error caracter inválido: <" + yytext() + "> en
fila: " + yyline + " columna: " + yycolumn );
                              /*throw new Error("Caracter no permitido <"+
                                  yytext()+">");*/ }
```

Contenido del archivo CUP:

```
import java_cup.runtime.*;
import AST.*;

parser code {:

	Scanner scanner;
	AST raiz;

	public parser(java.io.Reader input){
	scanner = new Scanner(input);

	}

public String errores(int sys)
{
	if (sys == 0)
		return("Class");
	else if (sys == 1)
		return("ID");
	else if (sys == 2)
		return("Llave_on");
	else if (sys == 3)
		return("Llave_off");
	else if (sys == 4)
		return("Public");
	else if (sys == 5)
		return("Static");
	else if (sys == 6)
		return("Else");
	else if (sys == 7)
		return("Void");
	else if (sys == 8)
		return("Main");
	else if (sys == 9)
		return("PARENT_on");
	else if (sys == 10)
		return("PARENT_off");
	else if (sys == 11)
		return("Corchete_on");
```

```
else if (sys == 12)
        return("Corchete_off");
else if (sys == 13)
        return("STRING");
else if (sys == 14 )
        return("Extends");
else if (sys ==15 )
        return("Return");
else if (sys == 16)
        return("INT");
else if (sys == 17)
        return("BOOLEAN");
else if (sys == 18)
        return("If");
else if (sys ==19 )
        return("DOTCOMMA");
else if (sys ==20 )
        return("COMA");
else if (sys == 21)
        return("Punto");
else if (sys == 22)
        return("While");
else if (sys == 23)
        return("System");
else if (sys == 24)
        return("Out");
else if (sys == 25)
        return("Println");
else if (sys == 26)
        return("Lenght");
else if (sys == 27)
        return("INTEGER_LITERAL");
else if (sys ==28 )
        return("True");
else if (sys == 29)
        return("False");
else if (sys ==30 )
        return("This");
else if (sys ==31 )
        return("New");
```

```
else if (sys == 32)
        return("Negacion");
else if (sys == 33)
        return("STRING_CONSTANT");
else if (sys == 34)
        return("Suma");
else if (sys == 35)
        return("Resta");
else if (sys ==36 )
        return("MULT");
else if (sys ==37 )
        return("Div");
else if (sys ==38 )
        return("Diferencia");
else if (sys == 39)
        return("Igual_Comp");
else if (sys == 40)
        return("Igual_Asig");
else if (sys == 41)
        return("Menor");
else if (sys == 42)
        return("Menor_Igual");
else if (sys ==43 )
        return("Mayor");
else if (sys == 44)
        return("Mayor_Igual");
else if (sys ==45 )
        return("OR");
else if (sys ==46 )
        return("AND");
else if (sys == 47)
        return("EOF");
else if (sys == 48)
        return("ERROR");
else if (sys == 49)
        return("NUM");
else if (sys == 50)
        return("STR");
else if (sys == 51)
        return("In");
```

```
        else if (sys == 52)
                return("Read");
        else if (sys == 53)
                return("Import");
        else if (sys == 54)
                return("Implements");
        else if (sys == 55)
                return("Exit");
        else
                return("error");
}

public void syntax_error(Symbol cur_token2)
{
        if (cur_token.sym == 0)
                report_error("No se esperaba un class", cur_token2);
        else if (cur_token.sym == 2)
                report_error("No se esperaba un ID", cur_token2);
        else if (cur_token.sym == 3)
                report_error("No se esperaba una llave abierta", cur_token2);
        else if (cur_token.sym == 4)
                report_error("No se esperaba una llave cerrada", cur_token2);
        else if (cur_token.sym == 5)
                report_error("No se esperaba un public", cur_token2);
        else if (cur_token.sym == 6)
                report_error("No se esperaba un else", cur_token2);
        else if (cur_token.sym == 7)
                report_error("No se esperaba un void", cur_token2);
        else if (cur_token.sym == 8)
                report_error("No se esperaba un main", cur_token2);
        else if (cur_token.sym == 9)
                report_error("No se esperaba un parentesis abierto", cur_token2);
        else if (cur_token.sym == 10)
                report_error("No se esperaba un parentesis cerrado", cur_token2);
        else if (cur_token.sym == 11)
                report_error("No se esperaba un corchete abierto", cur_token2);
        else if (cur_token.sym == 12)
                report_error("No se esperaba un cochete cerrado", cur_token2);
        else if (cur_token.sym == 13)
                report_error("No se esperaba un string", cur_token2);
```

```
else if (cur_token.sym == 14)
        report_error("No se esperaba un extendes", cur_token2);
else if (cur_token.sym == 15)
        report_error("No se esperaba un return", cur_token2);
else if (cur_token.sym == 16)
        report_error("No se esperaba un int", cur_token2);
else if (cur_token.sym == 17)
        report_error("No se esperaba un boolean", cur_token2);
else if (cur_token.sym == 18)
        report_error("No se esperaba un if", cur_token2);
else if (cur_token.sym == 19)
        report_error("No se esperaba un punto y coma", cur_token2);
else if (cur_token.sym == 20)
        report_error("No se esperaba una coma", cur_token2);
else if (cur_token.sym == 21)
        report_error("No se esperaba un punto", cur_token2);
else if (cur_token.sym == 22)
        report_error("No se esperaba un while", cur_token2);
else if (cur_token.sym == 23)
        report_error("No se esperaba un system", cur_token2);
else if (cur_token.sym == 24)
        report_error("No se esperaba un out", cur_token2);
else if (cur_token.sym == 25)
        report_error("No se esperaba un println", cur_token2);
else if (cur_token.sym == 26)
        report_error("No se esperaba un length", cur_token2);
else if (cur_token.sym == 27)
        report_error("No se esperaba un INTEGER_LITERAL", cur_token2);
else if (cur_token.sym == 28)
        report_error("No se esperaba un true", cur_token2);
else if (cur_token.sym == 29)
        report_error("No se esperaba un false", cur_token2);
else if (cur_token.sym == 30)
        report_error("No se esperaba un this", cur_token2);
else if (cur_token.sym == 31)
        report_error("No se esperaba una new", cur_token2);
else if (cur_token.sym == 32)
        report_error("No se esperaba una negacion ", cur_token2);
else if (cur_token.sym == 33)
        report_error("No se esperaba un STRING_CONSTANT", cur_token2);
```

```
else if (cur_token.sym == 34)
        report_error("No se esperaba una suma", cur_token2);
else if (cur_token.sym == 35)
        report_error("No se esperaba una resta ", cur_token2);
else if (cur_token.sym == 36)
        report_error("No se esperaba una multiplicacion", cur_token2);
else if (cur_token.sym == 37)
        report_error("No se esperaba una division", cur_token2);
else if (cur_token.sym == 38)
        report_error("No se esperaba una diferencia", cur_token2);
else if (cur_token.sym == 39)
        report_error("No se esperaba una comparacion", cur_token2);
else if (cur_token.sym == 40)
        report_error("No se esperaba una asignacion", cur_token2);
else if (cur_token.sym == 41)
        report_error("No se esperaba un menor ", cur_token2);
else if (cur_token.sym == 42)
        report_error("No se esperaba un menor igual", cur_token2);
else if (cur_token.sym == 43)
        report_error("No se esperaba un mayor", cur_token2);
else if (cur_token.sym == 44)
        report_error("No se esperaba un mayor igual", cur_token2);
else if (cur_token.sym == 45)
        report_error("No se esperaba un or", cur_token2);
else if (cur_token.sym == 46)
        report_error("No se esperaba un and", cur_token2);
else if (cur_token.sym == 47)
        report_error("No se esperaba un fin de linea", cur_token2);
else if (cur_token.sym == 48)
        report_error("No se esperaba un ERROR", cur_token2);
else if (cur_token.sym == 49)
        report_error("No se esperaba un numero", cur_token2);
else if (cur_token.sym == 50)
        report_error("No se esperaba una string", cur_token2);
else if (cur_token.sym == 51)
        report_error("No se esperaba una in", cur_token2);
else if (cur_token.sym == 52)
        report_error("No se esperaba una read", cur_token2);
else if (cur_token.sym == 53)
        report_error("No se esperaba un import", cur_token2);
```

```
        else if (cur_token.sym == 54)
                report_error("No se esperaba un implements", cur_token2);
        else if (cur_token.sym == 55)
                report_error("No se esperaba un exit", cur_token2);
        else
                report_error("Error: ", cur_token2);
}

public void report_error(String message, Object info) {
        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());

        System.out.println(m);
}


public void report_fatal_error(String message, Object info)    {
        report_error(message, info);
        throw new RuntimeException("Error Fatal de Sintaxis!!!");
}

:}

terminal Symbol ID;
terminal Symbol Llave_off;
terminal Symbol Llave_on;
terminal Symbol PARENT_off;
terminal Symbol PARENT_on;
terminal Symbol BOOLEAN;
terminal Symbol INT;
```

```
terminal Symbol DOTCOMMA;
terminal Symbol INTEGER_LITERAL;
terminal Symbol Corchete_off;
terminal Symbol Corchete_on;
terminal Symbol True;
terminal Symbol False;
terminal Symbol STRING_CONSTANT;
terminal Symbol Negacion;
terminal Symbol Suma;
terminal Symbol Resta;
terminal Symbol Public;
terminal Symbol Static;
terminal Symbol Extends;
terminal Symbol Return;
terminal Symbol If;
terminal Symbol COMA;
terminal Symbol Punto;
terminal Symbol While;
terminal Symbol System;
terminal Symbol Out;
terminal Symbol Println;
terminal Symbol Length;
terminal Symbol Else;
terminal Symbol Void;
terminal Symbol Main;
terminal Symbol STRING;
terminal Symbol MULT;
terminal Symbol Div;
terminal Symbol Diferencia;
terminal Symbol Igual_Comp;
terminal Symbol Class;
terminal Symbol This;
terminal Symbol New;
terminal Symbol Igual_Asig;
terminal Symbol Menor;
terminal Symbol Menor_Igual;
terminal Symbol Mayor;
terminal Symbol Mayor_Igual;
terminal Symbol OR;
terminal Symbol AND;
```

```
// terminal Symbol ERROR;
// terminal Symbol NUM;
// terminal Symbol STR;
terminal Symbol In;
terminal Symbol Read;
terminal Symbol Import;
terminal Symbol Implements;
terminal Symbol Exit;


non terminal Program;
non terminal ImportDecl;
non terminal ImportDecla;
non terminal ImportDecls;
non terminal ImportDeclsa;
non terminal TypeName;
non terminal TypeNames;
non terminal MainClass;
non terminal ClassDecl;
non terminal ClassDecla;
non terminal ClassDeclsa;
non terminal ClassDeclsb;
non terminal ClassDeclsc;
non terminal Statement;
non terminal Statementa;
non terminal Statementb;
non terminal Statementc;
non terminal Statementd;
non terminal Statements;
non terminal Statementsa;
non terminal BodyDecl;
non terminal Exp;
non terminal Exps;
non terminal Expsa;
non terminal Expsb;
non terminal ExpList;
non terminal ExpLists;
non terminal ExpListsa;
non terminal Type;
non terminal Types;
```

```
non terminal Typesa;
non terminal TypeNamesa;
non terminal VarDecl;
non terminal VarDecla;
non terminal ConstrDecl;
non terminal ConstrDecls;
non terminal ClassDecls;
non terminal MethodDecl;
non terminal MethodDecls;
non terminal MethodDecla;
non terminal MethodDeclb;
non terminal Exp_Simpl;
non terminal Op;
non terminal FormalList;


precedence left Return;
precedence left COMA;
precedence left Punto;
precedence left Class;
precedence left PARENT_on;
precedence left Else;
precedence left Llave_off;
precedence left Llave_on;
precedence left INT;
precedence left Suma;
precedence left Resta;
precedence left Div;
precedence left MULT;
precedence left Public;
precedence left BOOLEAN;
precedence left ID;

start with Program;

// Regla 01)
Program ::= ImportDecl ClassDecl MainClass ;

// Regla 02)
ImportDecl ::= ImportDecls ImportDecla ;
```

// Regla 03)
ImportDecla ::= ImportDecl | ;

// Regla 04)
ImportDecls ::= Import TypeName ImportDeclsa DOTCOMMA ;

// Regla 05)
ImportDeclsa ::= Punto MULT | ;

// Regla 06)
TypeName ::= ID TypeNames ;

// Regla 07)
TypeNames ::= Punto ID TypeNamesa ;

// Regla 08)
TypeNamesa ::= TypeNames | ;

// Regla 09)
MainClass ::= Class ID Llave_on Public Static Void Main PARENT_on STRING
Corchete_on Corchete_off ID PARENT_off Llave_on Statement Llave_off Llave_off ;

// Regla 10)
ClassDecl ::= ClassDecls ClassDecla ;

// Regla 11)
ClassDecla ::= | ClassDecl ;

// Regla 12)
ClassDecls ::= Class ID ClassDeclsc Llave_on BodyDecl Llave_off ;

// Regla 13)
ClassDeclsa ::= Extends | Implements ;

// Regla 14)
ClassDeclsb ::= ClassDeclsa ID ;

// Regla 15)
ClassDeclsc ::= ClassDeclsb | ;

```
// Regla 16)
BodyDecl ::= VarDecl | ConstrDecl | MethodDecl | ClassDecl ;

// Regla 17)
VarDecl ::= Type ID  VarDecla ;

// Regla 18)
VarDecla ::= VarDecl DOTCOMMA | ;

// Regla 19)
ConstrDecl ::= Public ID PARENT_on  FormalList PARENT_off Llave_on VarDecl
Statement Llave_off  ConstrDecls ;

// Regla 20)
ConstrDecls ::= ConstrDecl | ;

// Regla 21)
MethodDecl ::= Public MethodDecla ID PARENT_on  FormalList  PARENT_off
Llave_on VarDecl Statements MethodDeclb Llave_off  MethodDecls ;

// Regla 22)
MethodDecla ::= Type | Void ;

// Regla 23)
MethodDeclb ::= Return Exp DOTCOMMA | ;

// Regla 24)
MethodDecls ::= MethodDecl | ;

// Regla 25)
FormalList ::= Type ID Types | ;

// Regla 26)
Type ::= INT  Corchete_on  Corchete_off
    | BOOLEAN
    | INT
    | ID ;

// Regla 27)
```

```
Types ::= COMA Type ID Typesa ;

// Regla 28)
Typesa ::= | Types ;

// Regla 29)
Statement ::= Llave_on Statements Llave_off
          | If PARENT_on  Exp PARENT_off Statement Statementa
          | While PARENT_on  Exp PARENT_off Statement
          | System Punto Out Punto Println PARENT_on  Exp PARENT_off DOTCOMMA
          | System Punto Exit PARENT_on INTEGER_LITERAL PARENT_off
DOTCOMMA
        | ID Statementc Igual_Asig Statementd
          | Statementb Punto ID  PARENT_on  ExpList PARENT_off ;

// Regla 30)
Statementa ::= Else Statement | ;

// Regla 31)
Statementb ::= This | ID ;

// Regla 32)
Statementc ::= Corchete_on  Exp  Corchete_off | ;

// Regla 33)
Statementd ::= Exp | PARENT_on  Type PARENT_off System Punto In Punto Read
PARENT_on PARENT_off DOTCOMMA ;

// Regla 34)
Statements ::= Statement Statementsa ;

// Regla 35)
Statementsa ::= | Statements ;

// Regla 36)
Exp ::= Exp_Simpl Exps ;

// Regla 37)
Exps ::= PARENT_on Expsa PARENT_off Expsb ;
```

// Regla 38)
Expsa ::= Op Exp | Corchete_on Exp Corchete_off | Punto Length | Punto ID PARENT_on ExpList PARENT_off ;

// Regla 39)
Expsb ::= Exps | ;

// Regla 40)
Exp_Simpl ::= INTEGER_LITERAL
| True
| False
| ID
| This
| New INT Corchete_on Exp Corchete_off
| New ID PARENT_on ExpList PARENT_off
| Negacion Exp
| PARENT_on Exp PARENT_off
| STRING_CONSTANT ;

// Regla 41)
ExpList ::= Exp ExpLists | ;

// Regla 42)
ExpLists ::= COMA Exp ExpListsa ;

// Regla 43)
ExpListsa ::= | ExpLists ;

// Regla 44)
Op ::= Suma | Resta | MULT | Div | Diferencia | Igual_Comp | Menor | Menor_Igual | Mayor_Igual | Mayor | OR | AND ;