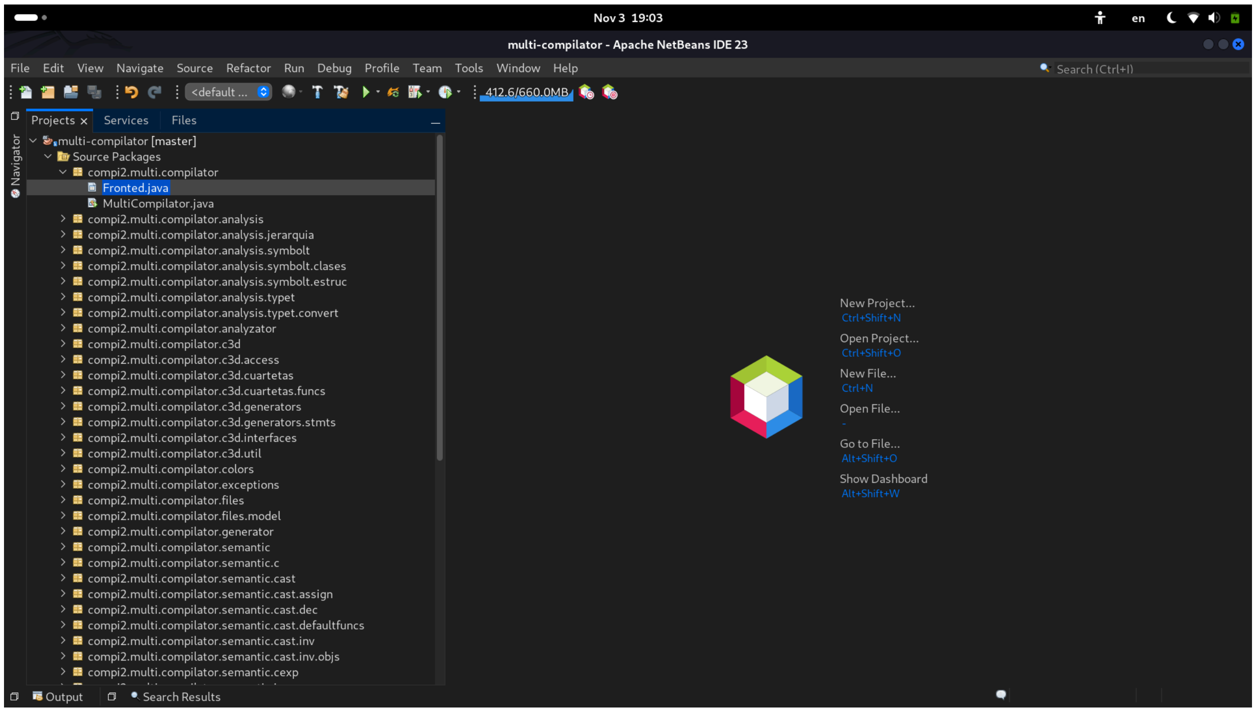
MULTI COMPILER

TECHNIC MANUAL

ÍNDICE

TECNOLOGÍA USADA EN EL PROYECTO

- Ide: ApacheNetBeans IDE 16


Nov 3 19:03
en
multi-compilator - Apache NetBeans IDE 23
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default ...
412.6/660.0MB
Projects
Services
Files
Navigator
New Project... Ctrl+Shift+N
Open Project... Ctrl+Shift+O
New File... Ctrl+N
Open File...
Go to File... Alt+Shift+O
Show Dashboard Alt+Shift+W
Output
Search Results

```
multi-compilator [master]
  Source Packages
    compi2.multi.compilator
      Fronted.java
      MultiCompilator.java
    compi2.multi.compilator.analysis
    compi2.multi.compilator.analysis.jerarquia
    compi2.multi.compilator.analysis.symbolt
    compi2.multi.compilator.analysis.symbolt.clases
    compi2.multi.compilator.analysis.symbolt.estruc
    compi2.multi.compilator.analysis.typet
    compi2.multi.compilator.analysis.typet.convert
    compi2.multi.compilator.analyzator
    compi2.multi.compilator.c3d
    compi2.multi.compilator.c3d.access
    compi2.multi.compilator.c3d.cuartetas
    compi2.multi.compilator.c3d.cuartetas.funcs
    compi2.multi.compilator.c3d.generators
    compi2.multi.compilator.c3d.generators.stmts
    compi2.multi.compilator.c3d.interfaces
    compi2.multi.compilator.c3d.util
    compi2.multi.compilator.colors
    compi2.multi.compilator.exceptions
    compi2.multi.compilator.files
    compi2.multi.compilator.files.model
    compi2.multi.compilator.generator
    compi2.multi.compilator.semantic
    compi2.multi.compilator.semantic.c
    compi2.multi.compilator.semantic.cast
    compi2.multi.compilator.semantic.cast.assign
    compi2.multi.compilator.semantic.cast.dec
    compi2.multi.compilator.semantic.cast.defaultfuncs
    compi2.multi.compilator.semantic.cast.inv
    compi2.multi.compilator.semantic.cast.inv.objs
    compi2.multi.compilator.semantic.cexp
```

- JFlex 1.9.1
  - Enlace para la descarga: https://jflex.de/download.html

- Java-cup-11b
  - Enlace para la descarga: https://www2.cs.tum.edu/projects/cup/

REQUERIMIENTOS PARA EL FUNCIONAMIENTO DEL PROYECTO

- **JAVA**

openjdk 23-ea 2024-09-17
OpenJDK Runtime Environment (build 23-ea+36-Debian-1)
OpenJDK 64-Bit Server VM (build 23-ea+36-Debian-1, mixed mode, sharing)

- **C++**

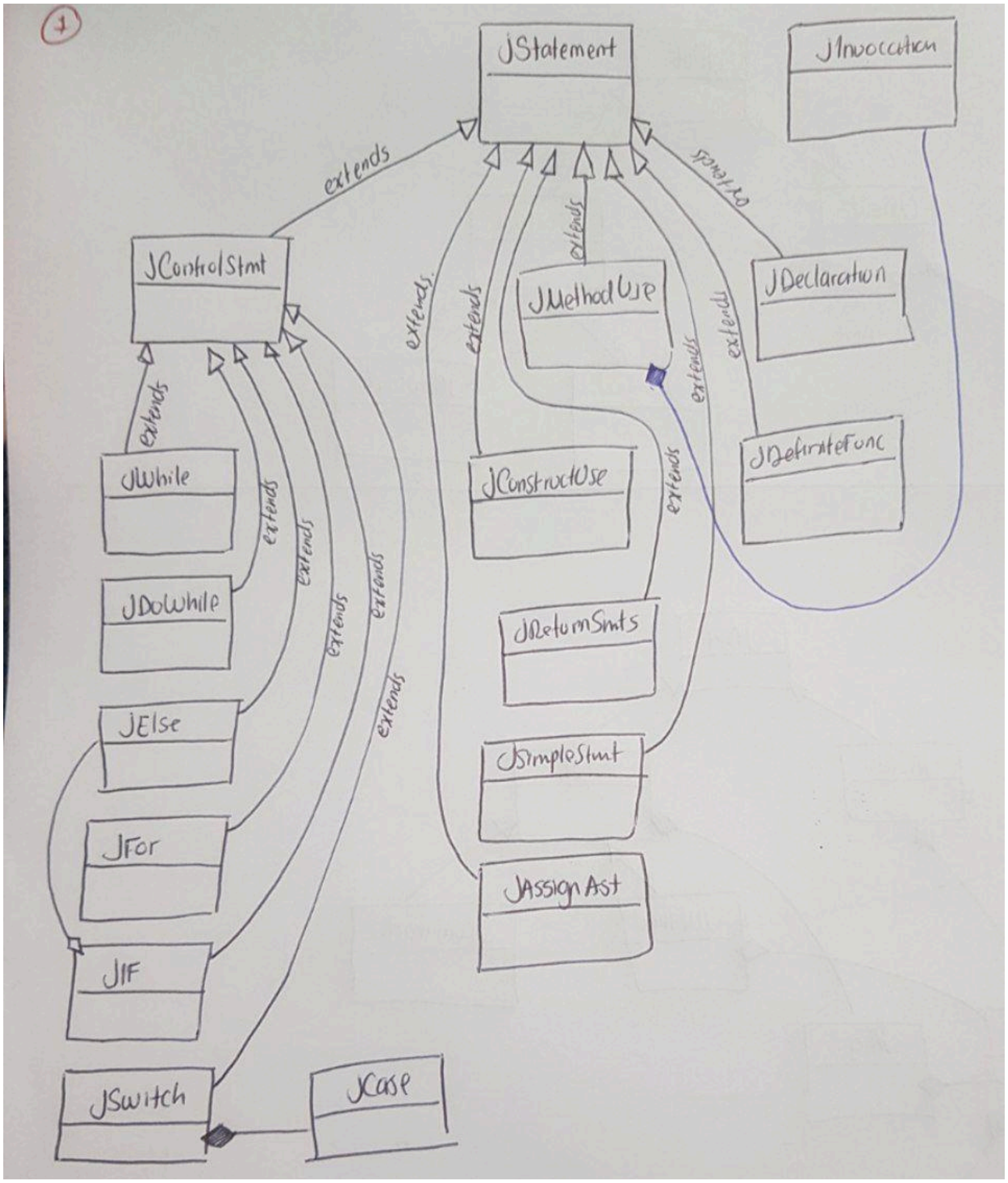
g++ (Debian 14.2.0-3) 14.2.0
Copyright (C) 2024 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.

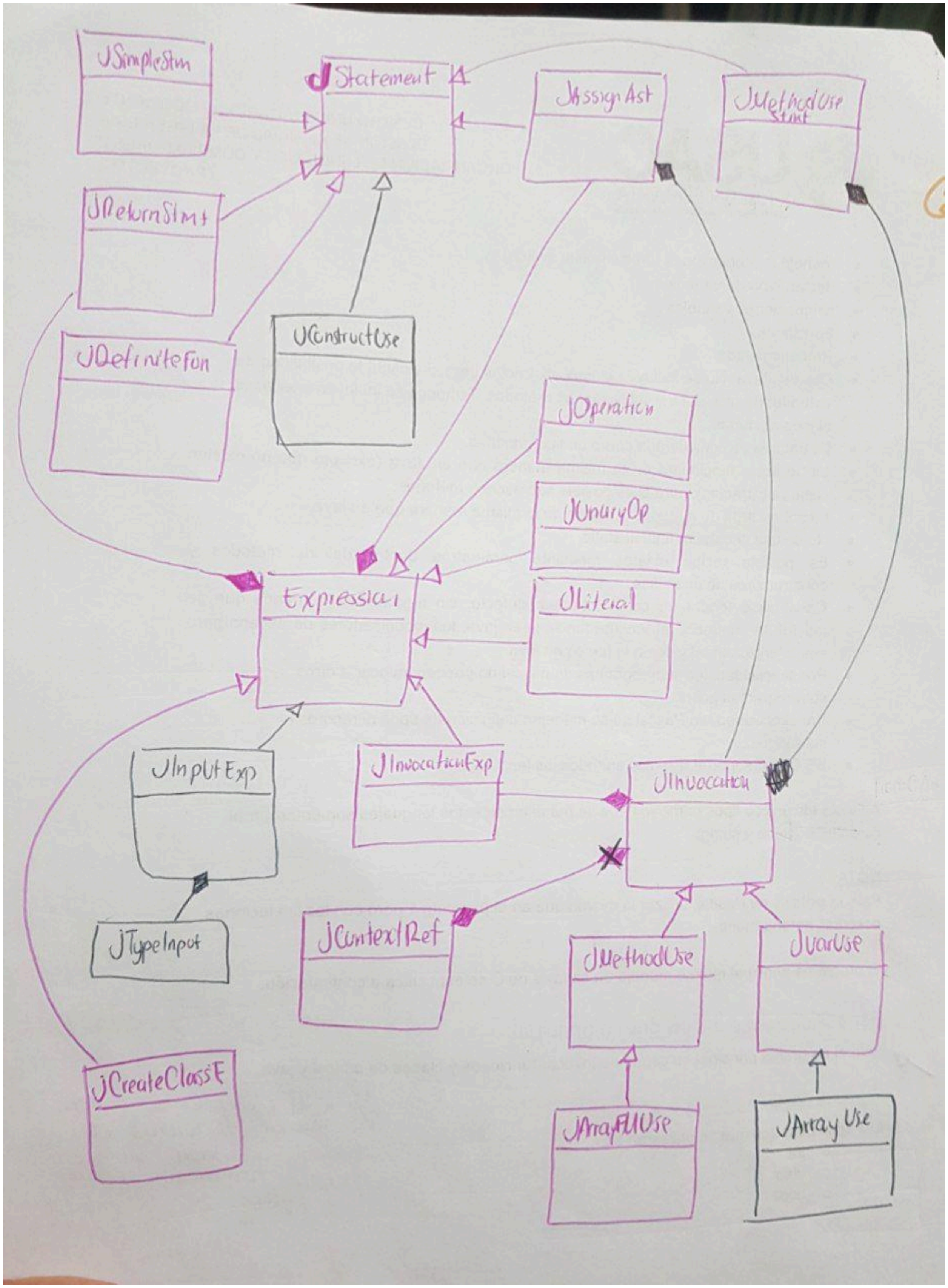
- NASM Assembler

Version 2.16.03

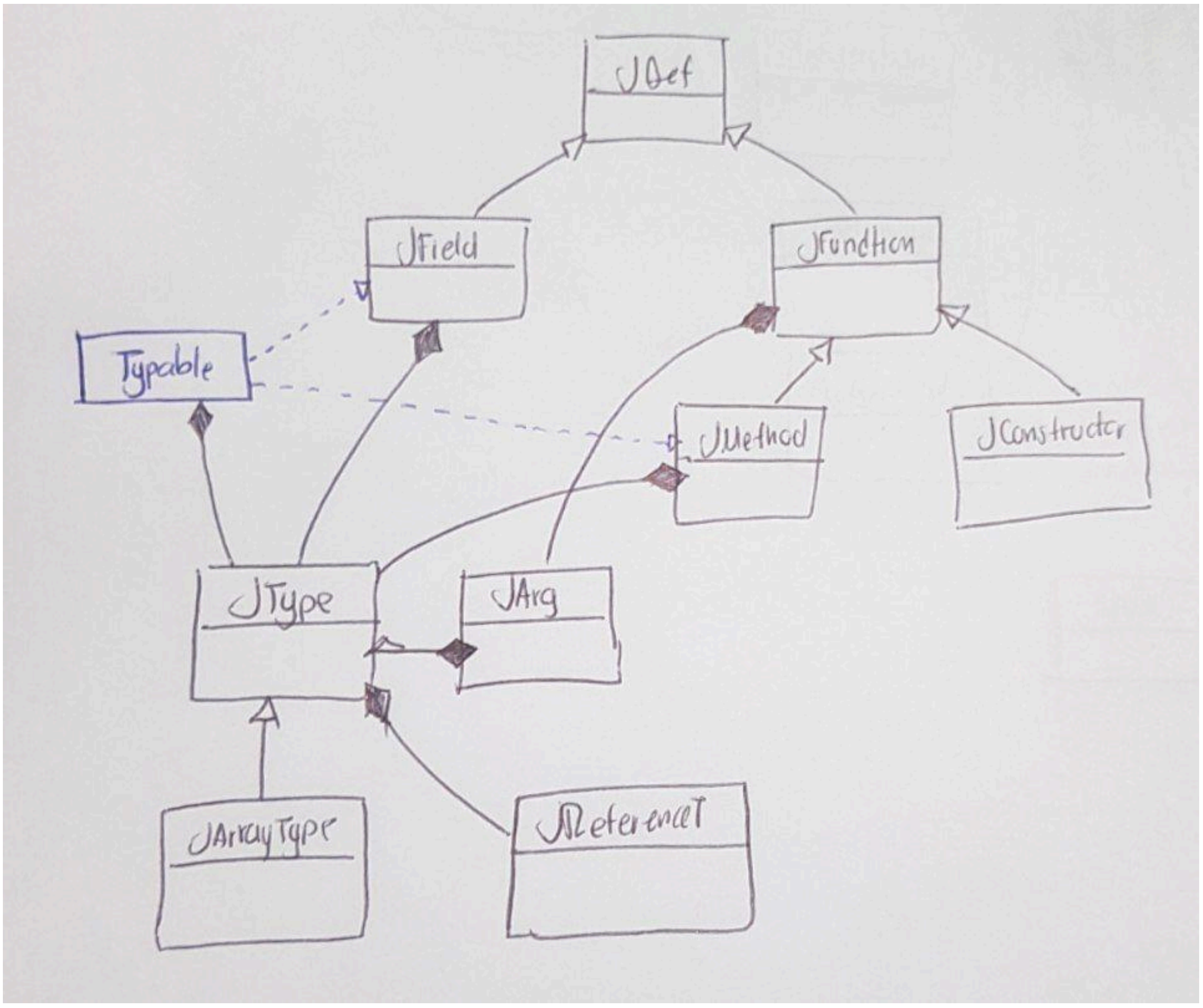
DIAGRAMA DE CLASES

ESTRUCTURAS DE JAVA

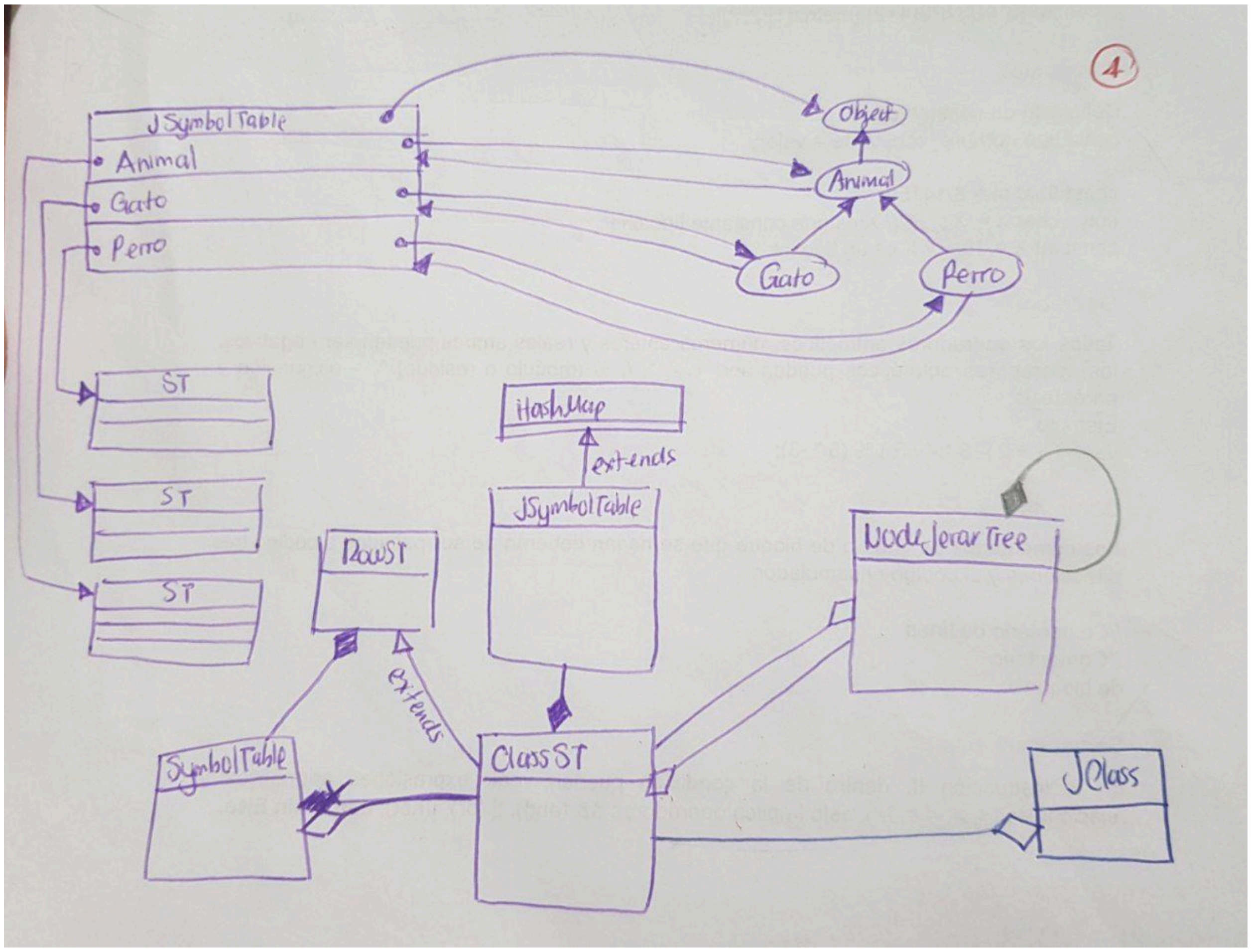



JSimpleStm
Statement
JAssignAst
JMethodUse Stmt
JReturnStmt
JConstructUse
JDefiniteFun
JOperation
JUnaryOp
Expression
JLiteral
JInputExp
JInvocationExp
JInvocation
JTypeInput
JContextRef
JMethodUse
JVarUse
JCreateClassE
JArrayFUUse
JArray Use

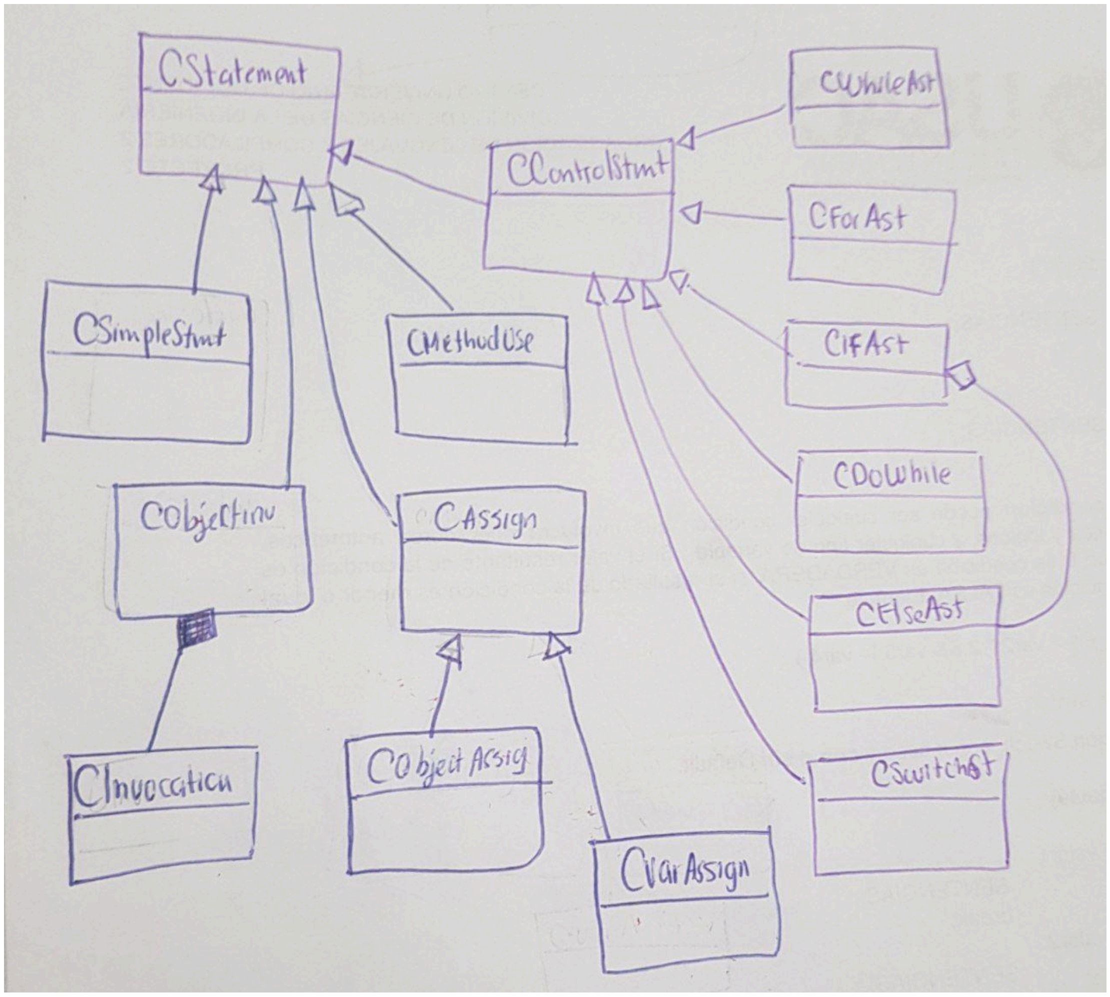
DEFINICIONES EN JAVA



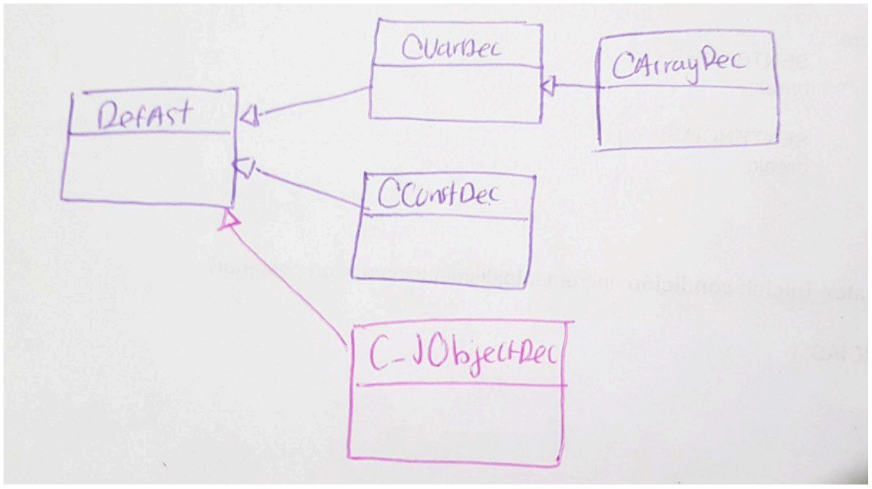
ESTRUCTURA DE UNA TABLA DE SÍMBOLOS ANIDADA PENSADA PARA OBJETOS



ESTRUCTURAS DEL LENGUAJE PRINCIPAL



DEFINICIONES EN EL LENGUAJE PRINCIPAL



ESPECIFICACIONES DEL LENGUAJE

PALABRAS RESERVADAS PARA INVOCAR LENGUAJES

- %%JAVA
- %%PASCAL
- %%PROGRAMA

TIPOS DE DATOS PRIMITIVOS

Las expresiones regulares para estos dependen de cada tipo de lenguaje.

- boolean
- string
- int
- real/float
- char

SÍMBOLOS COMUNES

Son símbolos encontrados en los tres lenguajes manejados.

OPERACIONES

Símbolo	Nombre
+	PLUS
-	MINUS
*	TIMES
/	DIV
%	MODULE
^	POWER

DELIMITADORES

Símbolo	Nombre
(	LPAREN
)	RPAREN

[	LBRACK
]	RBRACK
{	LBRACE
}	RBRACE

OTROS

Símbolo	Nombre
;	SEMICOLON
:	COLON
,	COMMA
.	DOT

ESPECIFICACIONES PARA EL LEXER

PASCAL

Es case insensitive, lo que significa que es insensible a mayúsculas y minúsculas.

PALABRAS RESERVADAS

- AND
- ARRAY
- BEGIN
- BOOLEAN
- BREAK
- CASE
- CHAR
- CONST
- CONTINUE
- DO
- DOWNTO
- ELSE
- FOR
- FUNCTION
- IF
- IN
- INTEGER
- MOD
- NOT
- OF
- OR
- PACKED
- PROCEDURE
- PROGRAM
- REAL
- REPEAT
- RETURN
- SET
- STRING
- THEN
- TO
- TYPE
- UNTIL
- VAR
- WHILE
- WITH

SÍMBOLOS

Símbolo	Nombre
=	EQUALS
<>	DIFFERENT
:=	ASSIGNATION

TIPOS DE DATOS

Comentario de una linea	// (carácter)*
Comentario de varias lineas	\{* (carácter)* *\}
String	‘(carácter)*’
Char	‘(carácter)’
Booleano	1|0
Int	(número)+
Real	(número)+.(número)+

JAVA

Es case sensitive, por lo cual las palabras reservadas tienen que ser escritas tal cual aparecen.

PALABRAS RESERVADAS

★ break
★ boolean
★ case
★ char
★ continue
★ class
★ default
★ do
★ else
★ extends
★ **false**
★ float
★ for
★ if
★ int
★ return
★ new
★ null
★ private
★ protected
★ public
★ String
★ switch
★ super
★ this
★ void
★ **true**
★ while
★ print
★ println

SÍMBOLOS

Símbolo	Nombre
&&	AND
||	OR
!	NOT
=	ASSIGNATION
==	EQUALS
!=	DIFFERENT

TIPOS DE DATOS

Comentario de una linea	{ (carácter)* }
Comentario de varias lineas	\{* (carácter)* *\}
String	“ (carácter)* ”
Char	‘ (carácter) ’
Booleano	true | false
Int	(número)+
Float	(número)+.(número)+

MAIN (C)

Es case sensitive

PALABRAS RESERVADAS

- arreglo
- break
- case
- char
- clrscr
- const
- continue
- do
- else
- float
- for
- getch
- if
- include
- int
- JAVA
- main
- PASCAL
- print
- println
- printf
- scanf
- string
- switch
- void
- while

SÍMBOLOS

Símbolo	Nombre
&&	AND
||	OR
!	NOT
=	ASSIGNATION
==	EQUALS
!=	DIFFERENT
&	AMPERSAND
#	HASH

ESPECIFICACIONES PARA EL PARSER

```
start with s;
s ::=   pascal_block:pb
      java_block
      main_program
      ;
```

PRECEDENCIAS

```
/*booleanos*/
precedence left THEN;
precedence left ELSE;
precedence left OR;
precedence left AND;
precedence left NOT;

/*operaciones*/
precedence left PLUS, MINUS;
precedence left TIMES, DIV, BARRA, MOD;
precedence left POWER;
precedence right UMINUS;

precedence right SEMICOLON;

precedence left IF;
precedence left ELSE;
```

GRAMÁTICA PARA PASCAL

```
pascal_block ::= PASCAL_SECTION pcontent
      ;
pcontent ::= functions_b procedure_b
      ;
```

ÚTILES

```
var_type ::= INTEGER
           | REAL
```

```
             | BOOLEAN
             | CHAR
             | STRING
             | ID
             ;

id_list ::= id_list COMA ID
             | ID
             ;

range ::= expression DOT DOT DOT expression
             ;

arr_range ::= CORCHETE_L expression DOT DOT expression CORCHETE_R
             ;

expression_list ::= expression_list COMA expression
             | expression
             ;
```

BLOQUE DE VARIABLES

```
var_b ::= VAR list_vardec
             | /* emtpy */
             ;

list_vardec ::= list_vardec var_dec
             | var_dec
             ;

var_dec ::= id_list COLON var_type SEMICOLON
             | id_list COLON range SEMICOLON
             | id_list COLON ARRAY arr_range OF var_type SEMICOLON
             | id_list EQUALS RECORD record_b END SEMICOLON
             ;
```

EXPRESIONES

```
/*precedencia de booleanos*/
```

```
precedence left THEN;
precedence left ELSE;
precedence left OR;
precedence left AND;
precedence left NOT;

/*precedencia de operaciones*/
precedence left PLUS, MINUS;
precedence left TIMES, DIV, MOD;
precedence right UMINUS;

expression ::= expression AND expression
            | expression AND THEN expression
            | expression OR expression
            | expression OR ELSE expression
            | expression NOT expression
            | bool_exp
            ;

bool_exp ::= arit_exp EQUALS arit_exp
            | arit_exp DIFFERENT arit_exp
            | arit_exp GRATER arit_exp
            | arit_exp LESS arit_exp
            | arit_exp GRATER_EQUALS arit_exp
            | arit_exp LESS_EQUALS arit_exp
            | arit_exp
            ;

arit_exp ::=
            arit_exp PLUS arit_exp
            | arit_exp MINUS arit_exp
            | arit_exp TIMES arit_exp
            | arit_exp DIV arit_exp
            | arit_exp MOD arit_exp
            | literals
            | ID CORCHETE_L arit_exp CORCHETE_R
            | ID DOT list_access
            | ID PARENTESIS_L id_list PARENTESIS_R
            | ID
            %prec UMINUS
            | PARENTESIS_L expression PARENTESIS_R
            %prec UMINUS
            | PLUS arit_exp
```

```
             %prec UMINUS
             | MINUS arit_exp
             ;

list_access ::=  ID DOT list_access
             | ID
             ;

literals ::= BOOLEAN_LIT
             | INTEGER_LIT
             | REAL_LIT
             | CHAR_LIT
             | STRING_LIT
             ;
```

BLOQUE DE INSTRUCCIONES

CONDICIONAL

```
precedence left IF;
precedence left ELSE;

conditional ::= if_stmt pos_if
      ;
if_stmt ::= IF PARENTESIS_L expression PARENTESIS_R THEN
      ;

pos_if ::= simple_stmt elif_stmt
      | simple_stmt SEMICOLON
      | complex_stmt
      | BEGIN block_stmt END SEMICOLON
      | BEGIN block_stmt END elif_stmt
      ;

elif_stmt ::=  ELSE IF PARENTESIS_L expression PARENTESIS_R THEN pos_elif
elif_stmt
      | ELSE IF PARENTESIS_L expression PARENTESIS_R THEN pos_elif
      | else_stmt
      ;
```

```
pos_elif ::= simple_stmt else_stmt
       | simple_stmt SEMICOLON
       | BEGIN block_stmt END SEMICOLON
       | BEGIN block_stmt END else_stmt
       ;

else_stmt ::= ELSE statements_wc
       | /* empty */
       ;

statements_wc ::= simple_stmt SEMICOLON
       | BEGIN block_stmt END SEMICOLON
       ;
```

CASE

```
case_stmt ::= CASE PARENTESIS_L expression PARENTESIS_R OF case_block ELSE
single_stm END SEMICOLON
       ;

case_block ::= labels COLON single_stm SEMICOLON
       ;

labels ::= labels COMA expression
       | expression
       ;

single_stm ::= ID ASSIGNATION expression
       | ID PARENTESIS_L expression_list PARENTESIS_R
       | ID DOT ID
       | definite_fun
       ;
```

CICLOS

```
while_stmt ::= WHILE PARENTESIS_L expression PARENTESIS_R DO statements
       ;

for_stmt ::= FOR ID ASSIGNATION expression TO expression DO statements
       ;

repeat_stmt ::= REPEAT statements UNTIL expression SEMICOLON
```

```
      ;
```

OTRAS INSTRUCCIONES

```
statements ::= simple_stmt
      | simple_stmt SEMICOLON
      | complex_stmt
      | BEGIN block_stmt END SEMICOLON
      ;

block_stmt ::= list_stmts
      | /* empty */
      ;

list_stmts ::= simple_stmt
      | simple_stmt SEMICOLON
      | complex_stmt
      | simple_stmt SEMICOLON more_stmts
      | complex_stmt more_stmts
      ;

more_stmts ::= simple_stmt
      | simple_stmt SEMICOLON
      | simple_stmt SEMICOLON more_stmts
      | complex_stmt more_stmts
      | complex_stmt
      ;

simple_stmt ::= BREAK
      | CONTINUE
      | ID ASSIGNATION expression
      | ID PARENTESIS_L expression_list PARENTESIS_R
      | ID DOT ID ASSIGNATION expression
      | definite_fun
      ;

complex_stmt ::= conditional
      | case_stmt
      | while_stmt
      | for_stmt
      | repeat_stmt
      ;
```

FUNCIONES

```
functions_b ::= functions
      | /* empty */
      ;

functions ::= functions function_dec
      ;

function_dec ::= function_dec FUNCTION ID PARENTESIS_L arguments
PARENTESIS_R COLON var_type SEMICOLON
            var_b
            BEGIN block_stmt END SEMICOLON
      ;
```

PROCEDIMIENTOS

```
procedure_b ::= procedures
      | /* empty */
      ;

procedures ::= procedures procedure_dec
      | procedure_dec
      ;

procedure_dec ::= PROCEDURE ID PARENTESIS_L arguments PARENTESIS_R
SEMICOLON
            var_b
            BEGIN block_stmt END SEMICOLON
      ;
```

PARÁMETROS Y ARGUMENTOS

```
arguments ::= arguments COMA type_arg
      | type_arg
      | /* empty */
      ;
```

```
type_arg ::= VAR id_list COLON var_type
       | id_list COLON var_type
       ;
```

GRAMÁTICA PARA JAVA

```
java_block ::= JAVA_SECTION jcontent
       ;
```

ÚTILES

```
jmodificator ::= PUBLIC
       | PRIVATE
       | PROTECTED
       ;

jtype ::= INT_TKN jbracks_list
       | STRING_TKN jbracks_list
       | FLOAT_TKN jbracks_list
       | BOOLEAN_TKN jbracks_list
       | CHAR_TKN jbracks_list
       | ID jbracks_list
       | INT_TKN
       | STRING_TKN
       | FLOAT_TKN
       | BOOLEAN_TKN
       | CHAR_TKN
       | ID
       ;

jbracks_list ::= jbracks_list jbrack
       | jbrack
       ;

jbrack ::= LBRACK RBRACK
       ;
```

```
jarray_access ::= LBRACK jexp:e RBRACK
      | jarray_access LBRACK jexp:e RBRACK
      ;

jexp_list ::= jexp_list COMMA jexp
      | jexp
      ;

jliterals ::= INTEGER_LIT
      | STRING_LIT
      | BOOLEAN_LIT
      | CHAR_LIT
      | FLOAT_LIT
      ;
```

CLASES

```
jcontent ::= jcontent jclass
      | /* empty */
      ;

jclass ::= PUBLIC CLASS ID jherence LBRACE jinternal_block RBRACE
      ;

jherence ::= EXTENDS ID
      | /* empty */
      ;
```

MÉTODOS Y ATRIBUTOS

```
jinternal_block ::= jinternal jinternal_block
      | /* empty */
      ;

jinternal ::=
            /* declarations */
      jmodificator jtype ID joptions_dec
      | jmodificator VOID ID jmethod_dec

            /* constructor */
      | jmodificator ID LPAREN jargs RPAREN LBRACE jstmts_block RBRACE
      ;

joptions_dec ::=
            /*simple field*/
      SEMICOLON
            /*simple and declaration field*/
      | ASSIGNATION jexp SEMICOLON
            /*method*/
      | jmethod_dec
      ;

jmethod_dec ::= LPAREN jargs RPAREN LBRACE jstmts_block RBRACE
      ;
```

PARÁMETROS

```
jargs ::= jlist_args
      | /* empty */
      ;

jlist_args ::= jlist_args COMMA jarg
      | jarg
      ;

jarg ::= jtype ID
      ;
```

BLOQUE DE INSTRUCCIONES

```
jstmts_block ::= jstmts_block jstmt
      | jstmt
      ;

jstmt ::= jcontrol_stmts
      | jsimple_stmts SEMICOLON
      | jdeclaration SEMICOLON
      | jmethod_use SEMICOLON
      | jconstruct_use SEMICOLON
      | jassign SEMICOLON
      | jdefinite_funcs SEMICOLON
      ;

jcontrol_stmts ::= jif_stmt
      | jwhile_stmt
      | jdo_while_stmt
      | jfor_stmt
      | jswitch_stmt
      ;

jsimple_stmts ::= BREAK
      | CONTINUE
      | RETURN jexp
      ;

jdefinite_funcs ::= PRINT LPAREN jexp_list RPAREN
      | PRINTLN LPAREN jexp_list RPAREN
      ;
```

CONDICIONAL

```
jif_stmt ::= IF LPAREN jexp RPAREN LBRACE jstmts_block RBRACE jelif_stmt
      ;

jelif_stmt ::= ELSE LBRACE jstmts_block RBRACE
      | ELSE jif_stmt
      | /* empty */
      ;
```

CICLOS

```
jwhile_stmt ::= WHILE LPAREN jexp RPAREN LBRACE jstmts_block RBRACE
      ;

jdo_while_stmt ::= DO LBRACE jstmts_block RBRACE WHILE LPAREN jexp RPAREN
SEMICOLON
      ;

jfor_stmt ::= FOR RPAREN jfor_reduced_stmt SEMICOLON jexp SEMICOLON
```

```
jfor_reduced_stmt RPAREN LBRACE jstmts_block RBRACE
      ;

jfor_reduced_stmt ::= jdeclaration
      | jmethod_use
      | jassign
      | /* empty */
      ;
```

SWITCH

```
jswitch_stmt ::= SWITCH LPAREN jexp RPAREN LBRACE jswitch_cases RBRACE
      ;

jswitch_cases ::= jswitch_cases jcase
      | jcase
      ;

jcase ::= CASE jexp COLON jstmts_block
      | DEFAULT COLON jstmts_block
      ;
```

ASIGNACIONES

```
jassign ::= ID ASSIGNATION jexp
      | ID PLUS PLUS
      | ID MINUS MINUS
      | THIS jaccess ASSIGNATION jexp
      | THIS jaccess PLUS PLUS
      | THIS jaccess MINUS MINUS
      | SUPER jaccess ASSIGNATION jexp
      | SUPER jaccess PLUS PLUS
      | ID jaccess ASSIGNATION jexp
      | ID jaccess PLUS PLUS
      | ID LPAREN RPAREN jaccess ASSIGNATION jexp
      | ID LPAREN RPAREN jaccess PLUS PLUS
      | ID LPAREN RPAREN jaccess MINUS MINUS
      | ID LPAREN jexp_list RPAREN jaccess ASSIGNATION jexp
      | ID LPAREN jexp_list RPAREN jaccess PLUS PLUS
      | ID LPAREN jexp_list RPAREN jaccess MINUS MINUS
      | ID jarray_access jaccess ASSIGNATION jexp SEMICOLON
      | ID jarray_access jaccess PLUS PLUS
      | ID jarray_access jaccess MINUS MINUS
      ;
```

OTROS

```
jdeclaration ::= jtype ID ASSIGNATION jexp
      | jtype ID
      ;

jmethod_use ::= THIS jaccess
      | SUPER jaccess
      | ID jaccess
      | ID LPAREN RPAREN
      | ID LPAREN jexp_list RPAREN
      | ID LPAREN RPAREN jaccess
      | ID LPAREN jexp_list RPAREN jaccess
      | ID jarray_access jaccess
      ;

jconstruct_use ::= THIS LPAREN RPAREN
      | THIS LPAREN jexp_list RPAREN
      | SUPER LPAREN RPAREN
      | SUPER LPAREN jexp_list RPAREN
      ;

jaccess ::= DOT jcomplex_access jaccess
      | DOT jcomplex_access
      ;

jcomplex_access ::= ID
      | ID LPAREN RPAREN
      | ID LPAREN RPAREN jarray_access
      | ID LPAREN jexp_list RPAREN
      | ID LPAREN jexp_list RPAREN jarray_access
      | ID jarray_access
      ;
```

EXPRESIONES

```
jexp ::= jexp AND jexp
      | jexp OR jexp
      | NOT jexp
      | jbool_exp
      ;

jbool_exp ::= jarit_exp:e1 EQUALS:o jarit_exp:e2j
      | jarit_exp:e1 DIFFERENT:o jarit_exp:e2j
      | jarit_exp:e1 GRATER:o jarit_exp:e2j
      | jarit_exp:e1 LESS:o jarit_exp:e2j
      | jarit_exp:e1 GRATER_EQUALS:o jarit_exp:e2j
      | jarit_exp:e1 LESS_EQUALS:o jarit_exp:e2j
      | jarit_exp:ej
      ;

jarit_exp ::= jarit_exp:e1 PLUS:o jarit_exp:e2
      | jarit_exp:e1 MINUS:o jarit_exp:e2
      | jarit_exp:e1 TIMES:o jarit_exp:e2
      | jarit_exp:e1 DIV:o jarit_exp:e2
      | jarit_exp:e1 MOD:o jarit_exp:e2
      | jarit_exp:e1 POWER:o jarit_exp:e2
      | jliterals:e
      | NULL_LIT
      | ID:i jarray_access
      | ID:i jarray_access jaccess
      | ID:i LPAREN jexp_list:l RPAREN
      | ID:i LPAREN RPAREN
      | ID:i LPAREN jexp_list:l RPAREN jaccess
      | ID:i LPAREN RPAREN jaccess
      | ID:i jaccess
      | ID:i
      | THIS jaccess
      | SUPER jaccess
      | NEW ID LPAREN RPAREN
      | NEW ID LPAREN jexp_list RPAREN
      %prec UMINUS
      | LPAREN jexp:e RPAREN
      %prec UMINUS
      | PLUS:o jarit_exp:e
      %prec UMINUS
      | MINUS:o jarit_exp:e  ;
```

GRAMÁTICA PARA EL PROGRAMA PRINCIPAL (C)

```
main_program ::=
      MAIN_SECTION
      cimports
      cconst_b
      cvars_b
      VOID MAIN LPAREN RPAREN LBRACE cstmts RBRACE
      ;
```

ÚTIL

```
ctype ::= INT_TKN
      | CHAR_TKN
      | FLOAT_TKN
      | STRING_TKN
      | BOOLEAN_TKN
      ;

carray_dims ::= carray_dims LBRACK cexp RBRACK
      | LBRACK cexp RBRACK
      ;

cparams ::= cexp COMMA cparams
      | cexp
      ;

cexp_list ::= cexp_list COMMA cexp
      | cexp
      ;

cliterals ::= STRING_LIT
      | FLOAT_LIT
      | INTEGER_LIT
      | BOOLEAN_LIT
      | CHAR_LIT
      ;
```

SECCIÓN INICIAL

```
cimports ::= cimports HASH INCLUDE STRING_LIT
      | /* empty */
      ;

cconst_b ::= cconst_b CONST ctype ID ASSIGNATION cexp SEMICOLON
      | /* empty */
      ;

cvars_b ::= cvars_b cvars_dec
      | /* empty */
      ;

cvars_dec ::= ctype ID SEMICOLON
      | ctype ID ASSIGNATION cexp SEMICOLON
      | ctype ARRAY carray_dims SEMICOLON
      | c_jclass_init
      ;

c_jclass_init ::= JAVA DOT ID c_java_construct SEMICOLON
      | JAVA DOT ID ARRAY carray_dims SEMICOLON
      ;

c_java_construct ::= c_j_construct COMMA c_java_construct
      | c_j_construct SEMICOLON
      ;

c_j_construct ::= ID LPAREN RPAREN
      | ID LPAREN cparams RPAREN
      ;
```

BLOQUE DE INSTRUCCIONES

```
cstmts ::= ccontrol_stmts
      | single_stmt SEMICOLON
      | def_functions_stmt SEMICOLON
      | c_jinvocation SEMICOLON
      | c_pinvocation SEMICOLON
      | cassign SEMICOLON
      ;

single_stmt ::= BREAK
      | CONTINUE
      ;

def_functions_stmt ::=  SCANF LPAREN STRING_LIT COMMA AMPERSAND ID RPAREN
      | PRINT LPAREN cexp_list RPAREN
      | CLEAR LPAREN RPAREN
      | GETCH LPAREN RPAREN
      ;

ccontrol_stmts ::= cif_stmt
      | celif_stmt
      | cswitch_stmt
      | cfor_stmt
      | cwhile_stmt
      | cdo_while_stmt
      ;

cassign ::= c_jinvocation ASSIGNATION cexp
      | c_pinvocation ASSIGNATION cexp
      | ID ASSIGNATION cexp
      ;
```

CONDICIONAL

```
cif_stmt ::= IF LPAREN cexp RPAREN LBRACE cstmts RBRACE celif_stmt
      ;

celif_stmt ::= ELSE cif_stmt
      | ELSE LBRACE cstmts RBRACE
      ;
```

SWITCH

```
cswitch_stmt ::= SWITCH LPAREN cexp RPAREN LBRACE ccases RBRACE
      ;
```

```
ccases ::= ccases ccase
      | ccase
      ;

ccase ::= CASE cexp COLON cstmts BREAK SEMICOLON
      | DEFAULT COLON cstmts BREAK SEMICOLON
      ;
```

CICLOS

```
cfor_stmt ::= FOR LPAREN cfor_reduced_stmt SEMICOLON
            cexp SEMICOLON cfor_reduced_stmt RPAREN LBRACE cstmts RBRACE
      ;

cfor_reduced_stmt ::= c_jinvocation
      | c_pinvocation
      | cassign
      | /* empty */
      ;


cwhile_stmt ::= WHILE LPAREN cexp RPAREN LBRACE cstmts RBRACE
      ;

cdo_while_stmt ::= DO LBRACE cstmts RBRACE
      WHILE LPAREN cexp RPAREN SEMICOLON
      ;
```

INVOCACIONES

```
c_jinvocation ::= JAVA DOT ID jaccess
      | JAVA DOT ID carray_dims jaccess
      ;

c_pinvocation ::= PASCAL DOT ID LPAREN RPAREN
      | PASCAL DOT ID LPAREN cparams RPAREN
      ;
```

EXPRESIONES

```
cexp ::= cexp AND cexp
      | cexp OR cexp
      | NOT jexp
      | cbool_exp
      ;

cbool_exp ::= carit_exp EQUALS carit_exp
      | carit_exp DIFFERENT carit_exp
      | carit_exp GRATER carit_exp
      | carit_exp LESS carit_exp
      | carit_exp GRATER_EQUALS carit_exp
      | carit_exp LESS_EQUALS carit_exp
      | carit_exp
      ;

carit_exp ::= carit_exp PLUS carit_exp
      | carit_exp MINUS carit_exp
      | carit_exp TIMES carit_exp
      | carit_exp DIV carit_exp
      | carit_exp MOD carit_exp
      | carit_exp POWER carit_exp
      | cliterals
      | c_jinvocation
      | c_pinvocation
      | ID:i
      %prec UMINUS
      | LPAREN cexp RPAREN
      %prec UMINUS
      | PLUS carit_exp
      %prec UMINUS
      | MINUS carit_exp
      ;
```

CONVERSIÓN IMPLÍCITA DE TIPOS PRIMITIVOS

Operadores Aritméticos

Suma

boolean (0)	0
char (1)	1
integer (2)	2
longint (3)	3
real (4)	4
string (5)	ERROR

No es posible realizar operaciones con un tipo personalizado

+	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	1	2	3	4	5
char	1	1	2	3	4	5
integer	2	2	2	3	4	5
longint	3	3	3	3	4	5
real	4	4	4	4	4	5
string	5	5	5	5	5	5

No es posible operar con tipos definidos por el usuario.

Resta

boolean (0)	0
char (1)	1
integer (2)	2
longint (3)	3
real (4)	4
string (5)	ERROR

No es posible realizar operaciones con un tipo personalizado

-	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	1	2	3	4	ERROR
char	1	1	2	3	4	ERROR
integer	2	2	2	3	4	ERROR
longint	3	3	3	3	4	ERROR
real	4	4	4	4	4	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible operar con tipos definidos por el usuario.

Multiplicación (*)

*	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	1	2	3	4	ERROR
char	1	1	2	3	4	ERROR
integer	2	2	2	3	4	ERROR
longint	3	3	3	3	4	ERROR
real	4	4	4	4	4	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible operar con tipos definidos por el usuario.

División (“/”)

/	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	4	4	4	4	4	ERROR
char	4	4	4	4	4	ERROR
integer	4	4	4	4	4	ERROR
longint	4	4	4	4	4	ERROR
real	4	4	4	4	4	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible operar con objetos.

Módulo (mod %)

div	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	2	2	2	2	ERROR	ERROR
char	2	2	2	2	ERROR	ERROR
integer	2	2	2	2	ERROR	ERROR
longint	2	2	2	2	ERROR	ERROR
real	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible operar con objetos.

Potencia (^)

div	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	4	4	4	4	4	ERROR
char	4	4	4	4	4	ERROR
integer	4	4	4	4	4	ERROR
longint	4	4	4	4	4	ERROR
real	4	4	4	4	4	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible operar con objetos.

Operadores Relacionales

Igualdad

=	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	0	0	0	0	ERROR
char	0	0	0	0	0	ERROR
integer	0	0	0	0	0	ERROR
longint	0	0	0	0	0	ERROR
real	0	0	0	0	0	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	0

Un objeto puede ser comparado por otro igual o un null y regresa un booleano.

Desigualdad

<>	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	0	0	0	0	ERROR
char	0	0	0	0	0	ERROR
integer	0	0	0	0	0	ERROR
longint	0	0	0	0	0	ERROR
real	0	0	0	0	0	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	0

Un objeto puede ser comparado por otro igual o un null y regresa un booleano.

Comparadores

> >= <= <	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	0	0	0	0	ERROR
char	0	0	0	0	0	ERROR
integer	0	0	0	0	0	ERROR
longint	0	0	0	0	0	ERROR
real	0	0	0	0	0	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible realizar operaciones con un objeto.

Operadores Booleanos

and, and then, or, or else	boolean (0)	char (1)	integer (2)	longint (3)	real (4)	string (5)
boolean	0	0	0	0	0	ERROR
char	0	0	0	0	0	ERROR
integer	0	0	0	0	0	ERROR
longint	0	0	0	0	0	ERROR
real	0	0	0	0	0	ERROR
string	ERROR	ERROR	ERROR	ERROR	ERROR	ERROR

No es posible realizar operaciones con un objeto.

Negación (Not)

boolean (0)	0
char (1)	0
integer (2)	0
longint (3)	0
real (4)	0
string (5)	ERROR

No es posible realizar operaciones con un tipo personalizado