archivo_compilado -> paquete libreria clase_interface_declaracion

paquete-> K_PACKAGE qualified_name EOL

libreria->(librerias'|vacio)
librerias'-> librerias(librerías'|vacio)
librerias-> K_REQUIRE qualified_name EOL

clase_interface_declaracion-> (clase_declaration|interface_declaration)

clase_declaration-> K_CLASS Identificador extend implement bodyclase K_END

extend-> (extends|vacio)
extends-> K_EXTENDS Identificador

implement-> (implements|vacio)
implements -> K_IMPLEMENTS qualified_name (implements'|vacio)
implements'->im (implements'|vacio)
im->(COMA qualified_name)

interface_declaration-> K_INTERFACE Identificador bodyclase K_END

bodyclase->(bodyclass|vacio)
bodyclass-->bodyuses(bodyclass|vacio)
bodyuses->(acciones| definir_variables| definir_arreglo| definir_propiedad| ciclos| clase_interface_declaracion| metodos| constructor)

acciones->
(imprimir|asignacion_especial|retorno|conversion_variable|accion_sistema|condicional|invocar_metodo_funcion|crear_objeto|referencia_objeto|inspeccionar_objeto|matematica_especial|accion_num|ajuste_array|ordenar_arreglo|busqueda_array|split|manejo_archivos|accion_arreglo|union|asignación)

imprimir-> K_IMPRIMIR valor EOL

asignacion_especial-> ASIGNACION_ESP LPAR bodyasign RPAR EOL
bodyasign->identificadores COMA valor

retorno-> K_RETORNO EOL

retorno_valor-> K_RETORNO valor EOL

conversion_variable-> CONVERSION LPAR identificadores RPAR EOL

accion_sistema-> ACCIONSYS EOL

condicional-> LLAIZQ bodyexpresiones LLADER K_IF LPAR condicion_exp RPAR (elseb|vacio)
bodyexpresiones-> (bodyexp|vacio)
bodyexp-> bodyexp'(bodyexp|vacio)
bodyexp'->(acciones|ciclos|definir_var_local|retorno_valor)

ciclos-> (ciclo_times|ciclo_each)

ciclo_times-> DecimalLiteral K_TIMES K_DO LLAIZQ bodyexpresiones LLADER

ciclo_each-> identificadores K_EACH K_DO DELIM var_local DELIM LLAIZQ bodyexpresiones LLADER

condicion_exp-> condicion cond
cond->(cond'|vacio)
cond'->OP_LOG condición_exp
condicion-> valor OP_REL valor

elseb-> K_ELSE LLAIZQ bodyexpresiones LLADER

definir_var_local-> var_local LPAR body_var_local RPAR EOL
body_var_local-> TIPO COMA (valor|vacio)

invocar_metodo_funcion-> K_INVOKE qualified_name LPAR (parametros|vacio) RPAR EOL

parametros-> valor param'
param'->(secuencia|vacio)
secuencia-> COMA parametros

crear_objeto-> qualified_name PUNTO K_NEW LPAR (parametros|vacio) RPAR EOL

referencia_objeto-> REFERENCIA LPAR identificadores RPAR EOL

inspeccionar_objeto-> K_INSPECCIONAR LPAR identificadores RPAR EOL

usar_metodo-> qualified_name LPAR (parametros|vacio) RPAR

matematica_especial-> MATEMATICA LPAR identificadores COMA valor RPAR EOL

accion_num-> NUM LPAR identificadores COMA valor RPAR EOL

ajuste_array-> K_RESIZE LPAR identificadores RPAR EOL

ordenar_arreglo->K_ORDENAR MODO_ORD LPAR identificadores RPAR EOL

busqueda_array-> identificadores K_WHERE identificadores TIPOBUSQUEDA valor EOL

split-> K_SPLIT LPAR string RPAR identificadores EOL

manejo_archivos-> K_BEGIN K_DIR LPAR manejo RPAR EOL
manejo-> ACCIONARCHIVO COMA string

accion_arreglo->ACCIONARREGLO LPAR identificadores COMA valor RPAR EOL

union->string K_UNION LPAR identificadores RPAR string EOL

asignacion-> identificadores ASIGNACION expresion EOL

expresion->valor expresion'
expresion'->(OP_ARI expresion)|vacio

definir_variables-> (definir_variable|definir_var_local)

definir_variable-> K_DEF K_VAR Identificador LPAR var_exp EOL

var_exp-> TIPO COMA VISIBILIDAD COMA (MODIFICADOR|vacio) COMA (valor|vacio) RPAR

definir_arreglo-> K_DEF K_ARRAY identificadores LPAR DecimalLiteral RPAR LLAIZQ
(exp_arreglo|vacio) LLADER EOL
exp_arreglo-> valor exp_arreglo'
exp_arreglo'->(COMA exp_arreglo)|vacio

definir_propiedad-> K_DEF K_PROPIEDAD Identificador LPAR Identificador RPAR LLAIZQ
body_propiedad LLADER

body_propiedad-> get (set|vacio)
get-> K_GET LLAIZQ Identificador LLADER
set-> K_SET LLAIZQ K_VALUE LLADER

constructor-> K_DEF K_INITIALIZE LPAR param_tipo RPAR LLAIZQ bodyexpresiones LLADER

param_tipo->(parametros_tipos|vacio)
parametros_tipos-> TIPO valor param_t'
param_t'->(COMA parametros_tipos)|vacio

metodos-> (funcion|metodo)

funcion-> K_DEF K_FUNC Identificador LPAR param_tipo RPAR LLAIZQ bodyexpresiones LLADER

metodo-> K_DEF K_VOID Identificador LPAR param_tipo RPAR LLAIZQ bodyexpresiones LLADER

valor-> (char|string|DecimalLiteral|booleano|var_local|usar_metodo|qualified_name)

var_local-> DOUBLEDOT Identificador

comentario-> # CaracteresNoSaltoLinea

identificador->(Identificador|var_local)

Programa Ejemplo

```
package programa.ejemplo;
#Esto es un comentario
require libreria1;
require libreria2;
class main extends heredada implements interfaz,interfac,iterfas
def initialize (){
	puts 'Nueva clase';
}
def var variable1 (int,private,final,20);
:local (int,);
def array arreglo(5){10,2,4,5,6,4,5,6};
:local2 (string,'jojoj');
add(:local,metodo());
FLOAT(variable1);
def property propiedad(variable1){
	get {variable1}
	set {value}
}
x=a.b.c[3];
#holabb
{
:local3 (string,'jojoj');
	main.new();
	round(variable1,0);
	beep;
	{
		sort asc (arreglo);
		beep;
	} if (variable1<=:local and x==y )
} if (variable1>:local)
else
```

```
{
	inspect(:local);
	cos(x,90);
	arreglo each do |:x|{
	ref(arreglo);
	{
		main.new();
		round(variable1,0);
		beep;
	} if (variable1<=:local and x==y )
	}
}
def void prueba(int x, string s)
{
	10 times do {
	x=x+1;
	break;
	}
	'string' union (x) 'mas texto';
}
invoke prueba(variable1,:local);

interface interfaz
def func test(){
	arreglo where arreglo like 10;
	split('o') :local2;
	begin dir (open,'direciconarchivo',x);
	push (arreglo,30);
	resize(arreglo);
	{
	:local4 (string,'jojoj');
		main.new();
		round(variable1,0);
		beep;
	} if (variable1<=:local and x==y )
	return x;
}
end
end
```