Gramática Analizador Sintáctico Servidor

s -> s_import clase

s_import -> s_import IMPORT ID id_import ;

| λ

id_import -> PUNTO ID id_import

| PUNTO MULTIPLICACION

| λ

clase -> visibilidad final CLASS ID { codigo }

codigo -> codigo metodo

| codigo visibilidad creacion

| codigo constructor

| λ

constructor -> visibilidad ID () parametros) { sentencias }

metodo -> visibilidad STRING ID (parametros) { sentencias_metodos }

| visibilidad INT ID (parametros) { sentencias_metodos }

| visibilidad BOOLEAN ID (parametros) { sentencias_metodos }

| visibilidad CHAR ID (parametros) { sentencias_metodos }

| visibilidad DOUBLE ID (parametros) { sentencias_metodos }

| visibilidad OBJECT ID (parametros) { sentencias_metodos }

| visibilidad VOID ID (parametros) { sentencias }

sentencias_metodos -> variable sentencias_metodos

| ciclo_for sentencias_metodos

| ciclo_while sentencias_metodos

| ciclo_do_while sentencias_metodos

| sentencia_if sentencias_metodos

| sentencia_switch sentencias_metodos

| llamada sentencias_metodos

| ID INCREMENTO; sentencias_metodos

| ID DECREMENTO; sentencias_metodos

| RETURN resultado;

| RETURN ID;

| λ

sentencias_if_else -> variable sentencias_if_else

| ciclo_for sentencias_if_else

| ciclo_while sentencias_if_else

| ciclo_do_while sentencias_if_else

| sentencia_if sentencias_if_else

| sentencia_switch sentencias_if_else

| llamada sentencias_if_else

| ID INCREMENTO; sentencias_if_else

| ID DECREMENTO; sentencias_if_else

| RETURN resultado;

| RETURN ID;

| BREAK;

| λ

sentencias -> variable sentencias

| ciclo_for sentencias

| ciclo_while sentencias

| ciclo_do_while sentencias

| sentencia_if sentencias

| sentencia_switch sentencias

| llamada sentencias

| ID INCREMENTO PUNTO_COMA sentencias

| ID DECREMENTO PUNTO_COMA sentencias

| BREAK;

| λ

sentencia_switch -> SWITCH (ID) { sentencia_case }

| SWITCH (ID) { sentencia_case sentencia_default }

sentencia_default -> DEFAULT: sentencias

sentencia_case -> CASE ENTERO: sentencias sentencia_case

| CASE CADENA: sentencias sentencia_case

| λ

ciclo_do_while -> DO { sentencias } WHILE (param_sentencia);

llamada -> ID (param_llamada);

param_llamada -> parametro_llamada

| λ

parametro_llamada -> resultado, parametro_llamada

| resultado

| ID

| ID, parametro_llamada

sentencia_if -> IF (param_sentencia) { sentencias_if_else } sentencia_else

sentencia_else -> ELSE { sentencias_if_else }

| ELSE sentencia_if

| λ

ciclo_while ::=

WHILE (param_sentencia) { sentencias }

;

param_sentencia -> bandera

| bandera logicos param_sentencia

| ID

| ID logicos param_sentencia

| ID relacionales resultado logicos param_sentencia

| ID relacionales resultado

| (param_sentencia)

| (param_sentencia) logicos param_sentencia

ciclo_for -> FOR (INT ID = ENTERO; ID relacionales valor; ID aumento) { sentencias }

aumento -> ++

| --

valor -> expresion

| ID

logicos -> LOGICO_Y

| LOGICO_O

relacionales -> >

| <

| >=

| <=

| ==

| !=

variable -> creacion

| asignacion

creacion -> var_int

| var_double

| var_string

| var_char

| var_boolean

| var_object

var_int -> INT ID;

| INT ID = expresion;

| INT ID = ID;

var_int -> DOUBLE ID;

| DOUBLE ID = expresion;

| DOUBLE ID = ID;

var_int -> STRING ID;

| STRING ID = CADENA;

| STRING ID = ID;

var_int -> CHAR ID;

| CHAR ID = LETRA;

| CHAR ID = ID;

var_int -> BOOLEAN ID;

| BOOLEAN ID = bandera;

| BOOLEAN ID = ID;

var_object -> OBJECT ID;

| OBJECT ID = NEW OBJECT ();

| OBJECT ID = ID;

| ID ID;

| ID ID = NEW ID (param_llamada);

| ID ID = ID;

asignacion -> ID = ID;

| ID = resultado;

| THIS . ID = ID;

resultado -> expresion

| CADENA c

| LETRA

| bandera

| NEW OBJECT ()

| NEW ID (param_llamada)

c -> + CADENA c

| + x c

| + ID c

| λ

bandera -> TRUE

| FALSE

numero ::=

ENTERO

| DECIMAL

parametros -> tipo_parametro ID , parametros

| tipo_parametro ID

| λ

tipo_parametro -> INT

| BOOLEAN

| STRING

| CHAR

| DOUBLE

| OBJECT

| ID:var

visibilidad -> PRIVATE

| PUBLIC

| PROTECTED

| λ

final -> FINAL

| λ

x -> t x_prima

x_prima -> + t x_prima

| - t x_prima

| λ

t -> p t_prima

t_prima -> * p t_prima

| / p t_prima

| λ

p -> - e

| e

e -> numero

| (x)

expresion -> numero

| expresion + expresion

| expresion - expresion

| expresion * expresion

| expresion / expresion

| - expresion

| (expresion)