Attribute Grammar

Nodo	Predicados	Reglas Semánticas
programa → *elemento*:elemento*		
definicion → *nombre*:String *tipo*:tipo		
funcion:elemento → *string*:String *parametros*:definicion* *tipo*:tipo *atributo*:atributo* *sentencia*:sentencia*	Simple(tipo) OR tipo==VOID Simple(parametros$_i$.getTipo())	sentencia$_i$.miFuncion = funcion
struct:elemento → *string*:String *definicion*:definicion*		
atributo:elemento → *definicion*:definicion		
if:sentencia → *condic*:expresion *verdadero*:sentencia* *falso*:sentencia*	condic.tipo==tipoint	sentencia$_i$.miFuncion = if.miFuncion
read:sentencia → *expresion*:expresion	simple(expresion.tipo) expresion.modificable == true	
return:sentencia → *expresion*:expresion	expresión.tipo == return.función.tipo Si return.miFuncion.tipo == VOID expresion == null	
print:sentencia → *expresion*:expresion	simple(expresion.tipo)	
while:sentencia → *expresion*:expresion *sentencia*:sentencia*	expresion.tipo == tipoint	sentencia$_i$.miFuncion = while.miFuncion
invocarSentencia:sentencia → *string:String* *expresion:expresion*	|expresion| == |invocarSentencia.defFuncion.parametro| expresion$_i$.tipo == invocarSentencia.defFuncion.parametro$_i$.tipo	

expresionBinaria:expresion, sentencia → *left*:expresion *string*:String *right*:expresion	left.tipo == right.tipo Si string ≠ '=' && left.tipo = tipoint right.tipo == tipoint Si string ≠ '=' && left.tipo = tiporeal right.tipo == tiporeal si string == '=' left.modificable == true left.tipo == right.tipo simple(left.tipo)	Si (operador == "+" OR operador == "-" OR operador == "/" OR operador == "*" OR operador == "=") expresionBinaria.tipo = left.tipo sino expresionBinaria.tipo = tipoint expresionBinaria.modificable = false
expresionBinaria:expresion, sentencia → *left*:expresion *string*:String *right*:expresion	left.tipo == right.tipo Si string ≠ '=' && left.tipo = tipoint right.tipo == tipoint Si string ≠ '=' && left.tipo = tiporeal right.tipo == tiporeal si string == '=' left.modificable == true left.tipo == right.tipo simple(left.tipo)	Si (operador == "+" OR operador == "-" OR operador == "/" OR operador == "*" OR operador == "=") expresionBinaria.tipo = left.tipo sino expresionBinaria.tipo = tipoint expresionBinaria.modificable = false
invocarFuncion:expresion → *string*:String *expresion*:expresion*	|expresion| == |invocarFuncion.defFuncion.parametro| $expresion_i$.tipo == $invocarFuncion.defFuncion.parametro_i$.tipo node.defFuncion.tipo ≠ VOID	invocarFuncion.tipo = invocarFuncion.defFuncion.tipo invocarFuncion.modificable = false
litent:expresion → *valor*:int		litent.tipo = tipoint litent.modificable = false
litchar:expresion → *valor*:String		litent.tipo = tipochar litchar.modificable = false

litreal:expresion → *valor*:String		litent.tipo = tiporeal litreal.modificable = false
var:expresion → *nombre*:String		var.tipo = definicion.tipo var.modificable = true
cast:expresion → *tipo*:tipo *expresion*:expresion	simple(tipo) simple(expresion.tipo) tipo ≠ expresion.tipo	cast.modificable = false
expresionUnaria:expresion → *expresion*:expresion	expresion.tipo == tipoint	expresionUnaria.tipo = tipoint expresionUnaria.modificable = false
expresionLogica:expresion → *left*:expresion *string*:String *right*:expresion	left.tipo == right.tipo left.tipo == tipoint	expresionLogica.tipo = tipoint expresioLogica.modificable = false
accesoArray:expresion → *contenedor*:expresion *posicion*:expresion	contenedor.tipo == array posicion.tipo == tipoint	accesoArray.tipo = contenedor.tipo.tipo accesoArray.modificable = true
accesoStruct:expresion → *contenedor*:expresion *atributo*:String	contenedor.tipo == tipoident contenedor.tipo.definicion.definicion [nombre==atributo] ≠ null	accesoStruct.tipo = contenedor.definicion.definicion[atributo].tipo accesoStruct.modificable = true
entreParentesis:expresion → *contenido*:expresion		entreParentesis.tipo = contenido.tipo entreParentesis.modificable = false
array:tipo → *litent*:litent *tipo*:tipo		
tipoint:tipo → λ		
tipovoid:tipo → λ		
tiporeal:tipo → λ		
tipochar:tipo → λ		
tipoident:tipo → *tipo*:String		

Recordatorio de operadores (para cortar y pegar): ⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ∩ ⊂ ⊄ Σ ∃ ∀

Funciones auxiliares:
simple(tipo) = (tipo == int) OR (tipo == real) OR (tipo == char)

Atributos

Categoría Sintáctica	Nombre del atributo	Tipo Java	Heredado/Sintetizado	Descripción
expresion	tipo	Tipo	Sintetizado	Tipo de la expresión (operaciones que permite)
expresion	modificable	boolean	Sintetizado	Indica si la expresión puede aparecer a la izquierda de una asignación
sentencia	miFuncion	Funcion	Heredado	Indica a la sentencia la función donde se encuentra