

Attribute Grammar

Attributes

Symbol	Attribute Name	Java Type	Inherited/Synthesized	Description
expression	tipo	Tipo	Synthesized	Tipo of the expression
expression	Lvalue	Boolean	Synthesized	True if the expression can appear to the left of an assignment
funcionSentencia	tipo	Tipo	Synthesized	Tipo of the funcionSentencia
declaracionfuncion	tipo	Tipo	Synthesized	Tipo of the declaracionfuncion
Sentencia	padre	declaracionfuncion	Inherited	Se usara tanto para los returns

Rules

Node	Predicates	Semantic Functions
program → declaraciones*		
declaracionstructs: d eclaraciones → nombre:string definicion*		
declaracionglobales: declaraciones → definicion	Declaracionglobales.tipo == definicion.tipo;	
declaracionfuncion: d eclaraciones → nombre:string argumento:definicion* tipo? variablesLocales:definicion* sentencia*	Definicion*.getTipo().tipoSimple() If(Tipo.exist()) Tipo.tipoSimple()	If(tipo.exist){ Declaracionfuncion.tipo = tipo; }else{ Declaracionfuncion.tipo = VoidTipo }
definicion → IDENT:string tipo		
printSentencia: sente ncia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	
readSentencia: sente ncia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	
printspSentencia: sen tencia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	

printlnSentencia: sentencia → expression*	expression*.getTipo().tipoSimple() expression*.lvalue == true	
functionSentencia: sentencia → nombre:string argumento:expression*	Expression.lenght == funcionSentencia.getDeclaracionfuncion().getTipo().get().params.length For(var a : expression){ a.tipo() == funcionSentencia.getDeclaracionfuncion().getTipo().get().params[indice].Tipo(); }	FunctionSentencia.tipo = funcionSentencia.getDeclaracionfuncion().getTipo().get()
asignacionSentencia: sentencia → left:expression expression	sameTipo(left.tipo, expression.tipo) left.tipo.tipoSimple() left.lvalue == true	
returnSentencia: sentencia → expression?	If(returnSentencia.declaracionfucnion().getTipo().isEmpty()){ expression.isEmpty() } If(returnSentencia.declaracionFuncion.getTipo().isPresent()){ Expression.isPresent() } ReturnSentecia.getDeclaFuncion().getTipo == expression.get().getTipo();	
ifSentencia: sentencia → condicion:expression entonces:sentencia* otro:sentencia*	Condicion.getTipo() == int;	
whileSentencia: sentencia → condicion:expression entonces:sentencia*	Condicion.getTipo() == int;	
intExpresion: expresion → intValue:int		IntExpresion.tipo = int; IntExpresion.lvalue = false;
realExpresion: expresion → doubleValue:double		RealExpresion.tipo = float; RealExpresion.lvalue = false;
identificadorExpresion: expression → name:string		IdentificadorExpresion.tipo = identificadorExpresion.getDefinicion().getTipo() IdentificadorExpresion.lvalue = false
charExpresion: expresion → charValue:char		charExpresion.tipo = char CharExpresion.lvalue = false

accesoArrayExpresion: expression → acceso: expression indice: expression	Acceso.getTipo() == Arraytipo Indice.getTipo() == IntTipo	ArrayTipo tipoArray = (ArrayTipo) indice.getTipoExpresion(); AccesoArrayExpresion.setTipoExpresion(tipoArray.getTipo()); AccesoArrayExpresion.lvalue = True
parentesisExpresion: expression → expression		ParetesisExpresion.tipo = expression.tipo ParentisisExpresion.lvalue = false
castExpresion: expres sion → tipo expression	Tipo.tipoSimple() Expression.tipo != tipo	CastExpresion.tipo = tipo CastExpresion.lvalue = false
negacionExpresion: e xpression → expression	expression.tipo == int	NegacionExpresion.tipo = expression.tipo Arithmetic.lvalue = false
arithmeticExpresion: expression → left: expression operator: string right: expression	NullOVoid(Left.getTipoexpresion()) NullOVoid(Right.getTipoExpresion()) sameTipo(left.tipo, right.tipo) Left.getTipoExpresion() == IntTipo FloatTipo Right.getTipoExpresion() == IntTipo FloatTipo	arithmeticExpresion.tipo = left.tipo arithmetic.lvalue = false
logicExpresion: expr ession → left: expression operator: string right: expression	SameTipo(left.tipo, right.tipo)	LogicExpresion.tipo = IntTipo LogicExpresion.lvalue = false
boolExpresion: expre ssion → left: expression operator: string right: expression	SameTipo(left.tipo, right.tipo) Left.getTipoExpresion() == IntTipo Right.getTipoExpresion() == IntTipo	boolExpresion.tipo = IntTipo boolExpresion.lvalue = false
acederCap: expressio n → left: expression right: string	Left.tipo == StringTipo ((StringTipo) Left.type).getdefinitions(). foreach(def -> def.getIdENT().equals(right)	AccederCap.tipo = def.tipo AccederCap.lvalue = true
funcionExpresion: ex pression → nombre: string argumentos: expression *	Expression.lenght == funcionSentencia.getDeclaracionfuncion().getTip o().get().params.length For(var a : expression){ a.tipo() == funcionSentencia.getDeclaracionfuncion().getTip o().get().params[indice].Tipo(); }	FuncionExpresion.tipo = funcionExpresion.getDeclaracionfu ncion.getTipo().get() FuncionExpresion.lvalue = false
intTipo: tipo → name: string		
floatTipo: tipo → name: string		

charTipo: tipo → name:string		
arrayTipo: tipo → tamArray:int tipo		
StringTipo: tipo → name:string		

Operators samples (cut & paste if needed):

$\Rightarrow \Leftrightarrow \neq \emptyset \in \notin \cup \cap \subset \not\subset \sum \exists \forall$

Auxiliary Functions

Name	Description
TipoSimple()	Comprueba que el tipo sea simple
NullOVoid()	Comprueba que el tipo pueda ser null o V