Attribute Grammar

Attributes

Symbol	Attribute Name	Java Type	Inherited/ Synthesized	Description
variable	varDef	VarDefinition	synthesized	Guardar referencia
varDefinition	scope	int	synthesized	Local o Global
FunctionCall (both)	func	FunctionDefinition	synthesized	Guardar referencia
run	func	functionDefinition	synthesized	Guardar referencia
structype	structDef	structDefinition	synthesized	Guardar referencia
fieldDef	structDef	structDefinition	synthesized	Guardar referencia

Sumado a las siguientes tablas y Mapas:

VarDefinitions, structDefinitions, fieldDefinitions, functionDefinitions, functionCreations

Rules

Node	Predicates	Semantic Functions
program → name:string definition* functionCreation* functionDefinition* run		
functionCreation → name:string	Not FunctionCreation in functionCreations	FunctionCreations.add(functionCreation)
varDefinition:definition → name:string type	Not VarDefinition in varDefinitions (from Top)	VarDefinitions.add(varDefinition) setScope(varDefinitions.scope)

structDefinition:definition → name:string fieldDefinition*	Not StructDefinition in structDefinitions	StructDefinitions.add(structDefinition)
		for field in fields:
		  field.setStructDef = structDef

fieldDefinition:definition → name:string type	Not fieldDefinition in fieldDefinitions	FieldDefinitions.add(fieldDefinition)
functionDefinition:definition → name:string parameters:varDefinition* type? locals:varDefinition* statement*	Not FunctionDef in funcDefs	FuncDefinitions.add(funcDefinition)
print:statement → expression*		
println:statement → expression*		
read:statement → expression*		
functionCallStatement:statement → name:string expression*	FunctionCall in FunctionDefinitions	FunctionCall.funcDef(funcDefFound)
assignment:statement → left:expression right:expression		

conditional:statement → expression ifStatements:statement* elseStatements:statement*		
loop:statement → fromStatements:statement* expression loopStatements:statement*		
return:statement → expression?		
run → name:string expression*	Run in FunctionCreations	Run.setFuncDef(funcDefFound)
intType:type → ε		
realType:type → ε		
charType:type → ε		
arrayType:type → intValue:int type		
structType:type → name:string		
voidType:type → ε		
variable:expression → name:string		
intLiteral:expression → intValue:int		
realLiteral:expression → floatValue:float		
charLiteral:expression → name:string		
functionCallExpression:expression → name:string expression*	FunctionCall in FunctionDefinitions	FunctionCall.funcDef(funcDefFound)
structAccess:expression → expr:expression name:string		
arrayAccess:expression → left:expression right:expression		
cast:expression → castType:type expression		
arithmeticBinary:expression → left:expression operator:string right:expression		
arithmeticUnary:expression → operator:string expr:expression		
logicBinary:expression → left:expression operator:string right:expression		
logicUnary:expression → operator:string expr:expression		

relationalBinary:expression → left:expression operator:string right:expression		

Operators samples (cut & paste if needed):
⇒ ⇔ ≠ ∅ ∈ ∉ ∪ ∩ ⊂ ⊄ ∑ ∃ ∀