Program ::= *Decl*$^{+}$

Decl ::= *VariableDecl* | *FunctionDecl* | *ConstDecl* | *ClassDecl* | *IntefaceDecl*

VariableDecl ::= *Variable* **;**

Variable ::= *Type* **ident**

ConstDecl ::= **static** *ConstType* **ident** *;*

ConstType ::= **int** | **double** | **boolean** | **string**

Type ::= **int** | **double** | **boolean** | **string** | **ident** | *Type* **[ ]**

FunctionDecl ::= *Type* **ident (** *Formals* **)** *StmtBlock* | **void ident (** *Formals* **)** *StmtBlock*

Formals ::= *Variable* **,** *Formals* | *Variable*

ClassDecl ::= **class ident** < **extends ident**> < **implements ident**$^{+}$ **,** > **{** *Field** **}**

Field ::= *VariableDecl* | *FunctionDecl* | *ConstDecl*

InterfaceDecl ::= **interface ident {** *Prototype** **}**

Prototype ::= *Type* **ident (** *Formals* **) ;** | **void ident (** *Formals* **) ;**

StmtBlock ::= **{** *VariableDecl** *ConstDecl** *Stmt** **}**

Stmt ::= < *Expr* > **;** | *IfStmt* | *WhileStmt* | *ForStmt* | *BreakStmt* | *ReturnStmt* | *PrintStmt*

  | *StmtBlock*

IfStmt ::= **if (** *Expr* **)** *Stmt* < **else** *Stmt* >

WhileStmt ::= **while (** *Expr* **)** *Stmt*

ForStmt ::= **for (** *Expr* **;** *Expr* **;** *Expr* **)** *Stmt*

ReturnStmt ::= **return** *Expr* **;**

BreakStmt ::= **break ;**

PrintStmt ::= **System.out.println (** *Expr*$^{+}$ **,) ;**

Expr ::= *LValue* **=** *Expr* | *Constant* | *LValue* | **this** | **(** *Expr* **)** | *Expr* **-** *Expr* | *Expr* **/** *Expr*

  | *Expr* **%** *Expr* | **-** *Expr* | *Expr* **>** *Expr* | *Expr* **>=** *Expr* | *Expr* **!=** *Expr* | *Expr* **||** *Expr*

  | **!** *Expr* | **New (ident)**

LValue ::= **ident** | *Expr* **. ident**

Constant ::= **intConstant** | **doubleConstant** | **booleanConstant** | **stringConstant** | **null**

[Gramática Original]

Program ::= Decl+

Decl ::= VariableDecl I FunctionDecl I ConstDecl I ClassDecl I IntefaceDecl

VariableDecl ::= Variable **;**

Variable ::= Type **ident**

Type ::= **int** I **double** I **boolean** I **string** I **ident** I Type **[]**

FunctionDecl ::= Type **ident (** Formals **)** StmtBlock I **void ident (** Formals **)** StmtBlock

Formals ::= Variable **,** Formals I Variable

ClassDecl ::= **class ident** < **extends ident** > < **implements ident+ ,** > **{** Field* **}**

Field ::= VariableDecl I FunctionDecl

InterfaceDecl ::= **interface ident {** Prototype* **}**

Prototype ::= Type **ident (** Formals **) ;** I **void ident (** Formals **) ;**

StmtBlock ::= **{** VariableDecl* Stint* **}**

Stmt ::= < Expr > **;** I IfStmt I WhileStmt I ForStmt I BreakStmt I ReturnStmt I PrintStmt | StmtBlock

IfStmt ::= **if (** Expr **)** Stmt < **else** Stmt >

WhileStmt ::= **while (** Expr **)** Stmt

ForStmt ::= **for (** Expr **;** Expr **;** Expr **)** Stmt

ReturnStmt ::= **return** Expr **;**

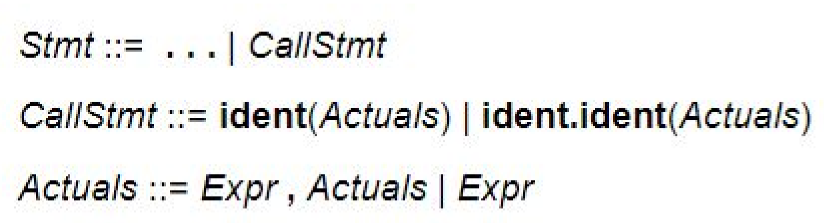
BreakStmt ::= **break ;**

PrintStmt ::= **System.out.println (** Expr+ **,) ;**

Expr ::= LValue **=** Expr I *Constant* I LValue I **this** I **(** Expr**)** I Expr **-** Expr I *Expr **I** Expr* | Expr **%** Expr I **-** Expr I Expr **>** Expr I Expr **>=** Expr I Expr **I=** Expr I Expr **II** Expr | **!** Expr I **New (ident)**

LValue ::= **ident** I Expr. **ident**

Constant ::= **static int intConstant** I **static double doubleConstant** | **static bool boolConstant** I **static string stringConstant** I **null**



Stmt ::= **. . .** | *CallStmt*

CallStmt ::= **ident**(*Actuals*) | **ident.ident**(*Actuals*)

Actuals ::= *Expr* , *Actuals* | *Expr*

[Gramática Modificada]

Inicio' ::=	Program
Program ::=	Decl
Decl ::=	ClassDecl Decl1
Decl ::=	InterfaceDecl Decl1
Decl ::=	ConstDecl Decl1
Decl ::=	FunctionDecl1 Decl1
Decl ::=	type Variable DECL2
DECL2::=	; Decl1
DECL2::=	FunctionDecl Decl1
Decl1 ::=	Decl
Decl1 ::=	ε
VariableDecl ::=	Variable **;**
Variable ::=	TypeArray **ident**
ConstDecl ::=	**static** ConstType **ident ;**
ConstType ::=	**int**
ConstType ::=	**double**
ConstType ::=	**boolean**
ConstType ::=	**string**
Type ::=	ConstType
Type ::=	**ident**
TypeArray: :=	**[ ] TypeArray**
TypeArray: :=	ε
FunctionDecl ::=	**(** Formals **)** StmtBlock
**FunctionDecl1 ::=	**void ident (** Formals **)** StmtBlock
Formals ::=	type Variable Formals1
Formals1 ::=	**,** Formals
Formals1 ::=	ε
ClassDecl ::=	**class ident** ClassDecl1 classDecl2 **{** Field **}**
ClassDecl1 ::=	**extends ident**
ClassDecl1 ::=	ε
ClassDecl2 ::=	**implements ident** ClassDecl3
ClassDecl2 ::=	ε
ClassDecl3 ::=	, **ident** ClassDecl3
ClassDecl3::=	ε
Field ::=	type Variable Field2
Field ::=	FunctionDecl1Field
Field ::=	ConstDecl Field
Field ::=	ε

Field2::=	; Field	
Field2::=	FunctionDecl Field	
InterfaceDecl ::=	**interface ident {** Prototype **}**	
Prototype ::=	**void ident (** Formals **) ;** Prototype	
Prototype ::=	Type TypeArray **ident (** Formals **) ;** Prototype	
Prototype ::=	ε	
StmtBlock ::=	**{** StmtBlock1 ConstDecl StmtBlock2 **}**	
StmtBlock1 ::=	Type VariableDecl StmtBlock1	
StmtBlock1 ::=	ε	
StmtBlock2 ::=	Stmt StmtBlock2	
StmtBlock2 ::=	ε	
Stmt ::=	**;**	
Stmt ::=	IfStmt	
Stmt ::=	WhileStmt	
Stmt ::=	ForStmt	
Stmt ::=	BreakStmt	
Stmt ::=	ReturnStmt	
Stmt ::=	PrintStmt	
Stmt ::=	StmtBlock	
Stmt::=	Lvalue Stmt0	
Stmt ::=	Expr **;**	
Stmt0::=	CallStmt	
Stmt0::=	Expr1;	PROGRAM.MAIN.COSA()
CallStmt ::=	(Expr Actuals)	
Actuals::=	, Expr Actuals	
Actuals ::=	ε	
IfStmt ::=	**if (** Expr **)** Stmt ElseStmt	
ElseStmt ::=	**else** Stmt	
ElseStmt ::=	ε	
WhileStmt ::=	**while (** Expr **)** Stmt	
ForStmt ::=	**for (** Expr **;** Expr **;** Expr **)** Stmt	
ReturnStmt ::=	**return** Expr **;**	
BreakStmt ::=	**break ;**	
PrintStmt ::=	**System.out.println (** PrintStmt2 **) ;**	
PrintStmt2::=	Expr PrintStmt3	
PrintStmt3::=	**,** Expr PrintStmt3	
PrintStmt3::=	ε	
Expr ::=	A Factor Expr1	
Expr1::=	Operacion Expr	
Expr1::=	ε	
A :==	!	
A :==	-	
A :==	ε	
Operacion ::=	=	
Operacion ::=	>	
Operacion ::=	>=	
Operacion ::=	!=	
Operacion ::=	||	

Operacion ::=	%
Operacion ::=	/
Operacion ::=	-
Factor ::=	Constant
Factor ::=	LValue
Factor ::=	**(** Expr **)**
Factor ::=	**New (ident)**
LValue ::=	**ident** LValue1
LValue ::=	this **. ident**
LValue1 ::=	**.ident** LValue1
LValue1 ::=	ε
Constant ::=	**intConstant**
Constant ::=	**doubleConstant**
Constant ::=	**boolConstant**
Constant ::=	**stringConstant**
Constant ::=	**null**