Program ::= Decl+

Decl ::= VariableDecl|FunctionDecl|CalssDecl|InterfaceDecl

VariableDecl ::= Variable;

Variable ::= Type ident

Type ::= int|double|bool|string|ident|Type[]

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

Formals ::= Variable+,|ε

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

Field ::= VariableDecl|FunctionDecl

InterfaceDecl ::= interface ident {Prototype\*}

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

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

StmtBlock ::= {VariableDecl\* Smtm\*}

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

WhileStmt ::= while (Expr) Stmt

ForStmt ::= for (<Expr>; Expr; <Expr>)Stmt

ReturnStmt ::= return <Expr>;

BreakStmt ::= break;

PrintStmt ::= Print (Expr+,);

Expr ::= LValue = Expr|Constant|LValue|this|Call|(Expr)|Expr+Expr|Expr-Expr|

Expr\*Expr|Expr/Expr|Expr%Expr|-Expr|Expr<Expr|Expr<=Expr|Expr>Expr|Expr>=Expr|

Expr==Expr|Expr!=Expr|Expr&&Expr|Expr||Expr|!Expr|New(ident)|NewArray(Expr,Type)|

ReadInteger()|ReadLine()|Malloc(Expr)

LValue ::= ident|Expr.ident|Expr[Expr]

Call ::= ident(Actuals)|Expr.ident(Actuals)|Expr.LibCall(Actuals)

LibCall ::= GetByte(Expr)|SetByte(Expr,Expr)

Actuals ::= Expr+,|ε

Constant ::= intConstant|doubleConstant|boolConstant|stringConstant|null