Gramática

Program ::= VarDecList FunDecList ClassDecList

VarDecList ::= { VarDec }

VarDec ::= (**var** | **val**) Identifier (**:** Type)? (= Exp) END

FunDecList ::= { FunDec }

FunDec ::= **fun** Identifier **(** [ParamList] **)** (**:** Type)?{ Block }

ClassDecList ::= { ClassDec }

ClassDec ::= **class** Identifier **{** {VarDec} [END] { ENDL } **}**

ParamList ::= Parameter { **,** Parameter }

Parameter ::= Identifier **:** Type

Type ::= **Int** | **Boolean** | **Unit**

Block ::= VarDecList StmtList

StmtList ::= { Statement [END] { ENDL }}

Statement ::= Exp END
| **if** (Exp) { Block } [**else** { Block }]
| **while** (Exp) { Block }
| **for** (Identifier **in** Exp **..** Exp) { Block }
| **return** [Exp] END
| **print** (Exp)
| **println** (Exp)
| Identifier = Exp

Exp ::= LogicAnd { || LogicAnd }

LogicAnd ::= Equality { **&&** Equality }

Equality ::= Comparison { (== | **!**=) Comparison }

Comparison ::= Term { (< | <= | > | >=) Term }

Term ::= Factor { (+ | -) Factor }

Factor ::= Unary { (***** | /) Unary }

Unary ::= (**!** | -) Unary | Primary

Primary ::= IntegerLiteral (INT_LITERAL)
| (**true** | **false**)
| **this**
| Identifier
| (Exp)
| Primary **.** Identifier
| Identifier **(** [ArgumentList] **)**

ArgumentList ::= Exp { **,** Exp }

END ::= **;** | **ENDL**

Identifier ::= ID