Gramática

Program ::= VarDecList ClassDecList FunDecList
VarDecList ::= {VarDec}
ClassDecList ::= {ClassDec}
FunDecList ::= {FunDec}
VarDec ::= (**var** | **val**) Identifier [**:** Type] = Exp END
ClassDec ::= **class** Identifier([ClassParamList]) {[VarDecList]}
FunDec ::= **fun** Identifier([FunParamList])[: Type] {Block}
FunParamList ::= FunParameter {, FunParameter}
FunParameter ::= Identifier : Type
Type ::= **Int** | **Boolean** | **Unit**
Block ::= VarDecList StmtList
StmtList ::= {Statement END {ENDL}}
Statement ::= IfStatement
| PrintStatement
| FCallStm
| AssignStatement
| ForStatement
| WhileStatement
| ReturnStatement
IfStatement ::= **if**(Exp){Block}[**else**{Block}]
PrintStatement ::= (**print** | **println**)(Exp)
FCallStm ::= Identifier([Exp {, Exp}])
AssignStatement ::= Identifier = Exp
ForStatement ::= **for**(Identifier **in** Exp..Exp){Block}
WhileStatement ::= **while**(Exp){Block}
ReturnStatement ::= **return**[Exp]
Exp ::= LogicAnd {|| LogicAnd}
LogicAnd ::= Equality { Equality}
Equality ::= Comparison {(== | !=) Comparison}
Comparison ::= Term {(<|≤|>|≥) Term}
Term ::= Factor {(+ | −) Factor}
Factor ::= Unary {(∗ | /) Unary}
Unary ::= (! | −) Unary | Primary
Primary ::= IntegerLiteral
| BooleanLiteral
| Identifier
| Identifier(Arguments)
| Identifier.Identifier
| (Exp)
BooleanLiteral ::= **true** | **false**
Arguments ::= [Exp {, Exp}]
END ::= **;** | **ENDL**