Gramática

Program	::=	{ Declaration [END] { ENDL }}
Declaration	::=	VarDecl | FunDecl | ClassDecl
VarDecl	::=	("var" | "val") Identifier (":" Type)? "=" Expression END
FunDecl	::=	"fun" Identifier "(" [ParamList] ")" (: Type)? Block
ClassDecl	::=	"class" Identifier "{" Declaration [END] { ENDL }"}"
ParamList	::=	Parameter { "," Parameter }
Parameter	::=	Identifier ":" Type
Type	::=	"Int" | "Boolean" | "Unit"
Block	::=	"{" Statement [END] { ENDL }"}"
Statement	::=	("var" | "val") Identifier (: Type)? = Expression END
		| Expression END
		| "if" "("Expression")" Statement ["else" Statement]
		| "while" "("Expression")" Statement
		| "for" "(" Identifier "in" Expression ".." Expression")" Statement
		| "return" [Expression] END
		| { Statement [END] { ENDL }}
		| "print" (LogicOr)
		| "println" (LogicOr)
Expression	::=	Assignment
Assignment	::=	Identifier "=" Assignment | LogicOr
LogicOr	::=	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
		| "(" Expression ")"
		| Primary "." Identifier
		| Identifier "(" [ArgumentList] ")"
ArgumentList	::=	Expression { , Expression }
END	::=	SEMICOLON | ENDL
Identifier	::=	ID