Grammar

The grammar below uses the notation x^* to denote that x might appear an arbitrary number of times, and the notation $x^?$, which means that x is optional.

Program	→	*TopLevelDecl**
TopLevelDecl	→	*FunctionDecl*
	|	*ClassDecl*
FunctionDecl	→	*Type id* (*ParamList*$^?$) *Block*
VarDecl	→	*Type id* ;
ParamList	→	*Type id ParamRest**
ParamRest	→	, *Type id*
BaseType	→	**boolean**
	|	**int**
	|	*id*
Type	→	*BaseType*
	|	*BaseType* [ ]
Block	→	{ *BlockStatement** }
BlockStatement	→	*Statement*
	|	*Type* id ;
Statement	→	*Block*
	|	**if** (*Exp*) *Statement* **else** *Statement*
	|	**while** (*Exp*) *Statement*
	|	**return** *Exp*;
	|	*Exp*;
	|	*ExpL*= *Exp*;
Exp	→	*Exp op Exp*
	|	! *Exp*
	|	- *Exp*
	|	*Exp* . **length**
	|	*id* (*ExpList*$^?$)
	|	**true**
	|	**false**
	|	⟨integer literal⟩
	|	(*Exp*)
	|	**this**
	|	**null**
	|	**new** *id* ( )
	|	**new** *BaseType* [*Exp*] [ ]*
	|	*Exp* . *id* (*ExpList*$^?$)
	|	*ExpL*
ExpL	→	*id*
	|	*Exp*[*Exp*]
	|	*Exp* . *id*
ExpList	→	*Exp ExpRest**
ExpRest	→	, *Exp*
id	→	⟨identifier⟩
op	→	&& | + | - | * | / | < | ==
ClassDecl	→	**class** *id* { *MemberDecl** }
	|	**class** *id* **extends** *id* { *MemberDecl** }
MemberDecl	→	*VarDecl*
	|	*MethodDecl*
MethodDecl	→	*Type id* (*ParamList*$^?$) *Block*