IC Grammar

```
program    ::= classDecl*
classDecl  ::= class CLASS [extends CLASS] '{' (field|method)* '}'
field      ::= type ID (',' ID)* ';'
method     ::= [static] (type|void) ID '(' [formals] ')' '{' stmt* '}'
formals    ::= type ID (',' type ID)*
type       ::= int | boolean | string | CLASS | type '[' ']'

stmt       ::= location '=' expr ';'
             | call ';'
             | return [expr] ';'
             | if '(' expr ')' stmt [else stmt]
             | while '(' expr ')' stmt
             | break ';'
             | continue ';'
             | '{' stmt* '}'
             | type ID ['=' expr] ';'

expr       ::= location
             | call
             | this
             | new CLASS '(' ')'
             | new type '[' expr ']'
             | expr '.' length
             | expr binop expr
             | unop expr
             | literal
             | '(' expr ')'

call       ::= staticCall | virtualCall
staticCall ::= CLASS '.' ID '(' [expr (',' expr)*] ')'
virtualCall::= [expr '.'] ID '(' [expr (',' expr)*] ')'
location   ::= ID | expr '.' ID | expr '[' expr ']'

binop      ::= '+' | '-' | '*' | '/' | '%' | '&&' | '||'
             | '<' | '<=' | '>' | '>=' | '==' | '!='
unop       ::= '-' | '!'
literal    ::= INTEGER | STRING | true | false | null
```