```
Program  ::=  (ClassDeclaration)* eot

ClassDeclaration  ::=
       class id  {
           MemberDeclaration*
       }

MemberDeclaration ::= (public | private)?  static?  Type id (
             | ;
             | (ParameterList?) {
                   Statement* (return Expression ;)?
               }
             )

Type  ::=  PrimitiveType | ReferenceType

PrimitiveType ::= int([])?
                | boolean
                | void

ReferenceType ::= id([])?

ParameterList  ::=  Type id (, Type id)*

Args ::= (ArgumentList?)

ArgumentList  ::=  Expression (, Expression)*

Reference  ::=  BaseRef ReferenceTail

BaseRef  ::=  this | RefSegment

ReferenceTail :: = (. id([Expression])?)*

RefSegment ::= id([Expression])?

Statement  ::=
         { Statement* }
       |  ReferenceStatement
       |  PrimitiveType id = Expression;
       |  if ( Expression ) Statement (else Statement)?
       |  while ( Expression ) Statement

ReferenceStatement ::=
       |  this ReferenceTail ReferenceStatementTail
       |  id ( (id = Expression; | ReferenceTail ReferenceStatementTail)
               |  [ (] id = Expression; | Expression] ReferenceTail
ReferenceStatementTail)
            )

ReferenceStatementTail ::= (= Expression | Args);

Expresion ::= DisjunctionExpression
```

```
DisjunctionExpression ::= ConjunctionExpression (|| ConjunctionExpression)*

ConjunctionExpression ::= EqualityExpression (&& EqualityExpression)*

EqualityExpression ::= RelationalExpression ((== | !=) RelationalExpression)*

RelationalExpression ::= AdditiveExpression ((<= | < | > | >=)
AdditiveExpression)*

AdditiveExpression ::= MultiplicativeExpression ((+ | -)
MultiplicativeExpression)*

MultiplicativeExpression ::= TerminalExpresion ((* | /) TerminalExpresion)*

TerminalExpresion  ::=
          Reference (Args)?
       |  (- | !) Expression
       |  (Expression)
       |  new  (id (() | [Expression]) | int [Expression])
       |  num | true  |  false
```