program:


```
program  ::= classlist?
```

no references

classlist:


```
classlist
        ::= classdecl+
```

referenced by:

- classbody
- program

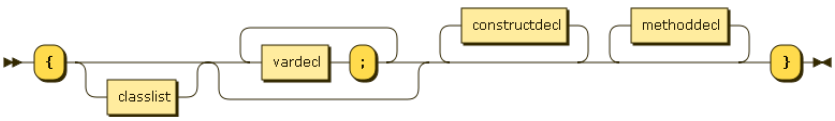
classdecl:


```
classdecl
        ::= 'class' ident ( 'extends' ident )? classbody
```

referenced by:

- classlist

classbody:

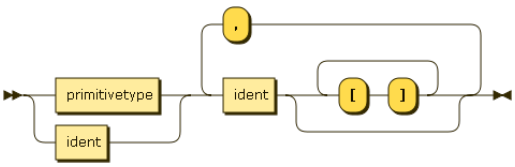



```
classbody
        ::= '{' classlist? ( vardecl ';' )* constructdecl* methoddecl* '}'
```

referenced by:

- classdecl

vardecl:

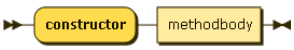



```
vardecl  ::= ( primitivetype | ident ) ident ( '[' ']' )* ( ',' ident ( '[' ']' )* )*
```

referenced by:

- classbody
- statement

constructdecl:

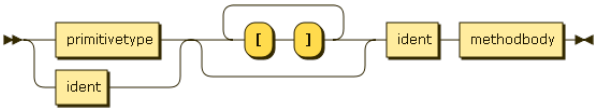



```
constructdecl
        ::= 'constructor' methodbody
```

referenced by:

- classbody

methoddecl:

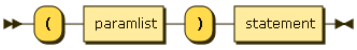


```
methoddecl
        ::= ( primitivetype | ident ) ( '[' ']' )* ident methodbody
```

referenced by:

- classbody

methodbody:

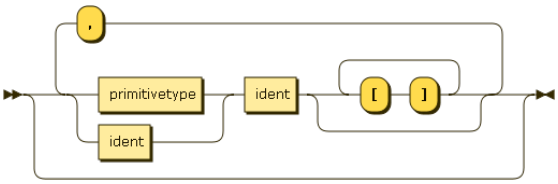



```
methodbody
        ::= '(' paramlist ')' statement
```

referenced by:

- constructdecl
- methoddecl

paramlist:

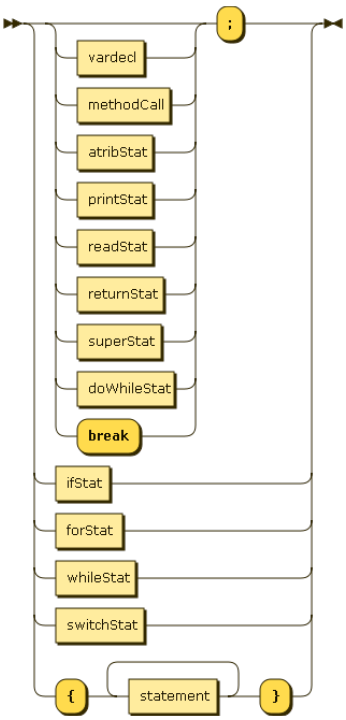



```
paramlist
        ::= ( ( primitivetype | ident ) ident ( '[' ']' )* ( ',' ( primitivetype | ident ) ident ( '[' ']' )* )* )?
```

referenced by:

- methodCall
- methodbody

statement:

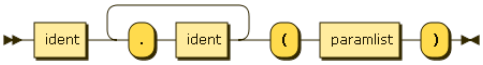



```
statement
        ::= ( vardecl | methodCall | atribStat | printStat | readStat | returnStat | superStat | doWhileStat | 'break' )? ';'
          | ifStat
          | forStat
          | whileStat
          | switchStat
          | '{' statement+ '}'
```

referenced by:

- doWhileStat
- forStat
- ifStat
- methodbody
- statement
- switchCaseStat
- whileStat

methodCall:

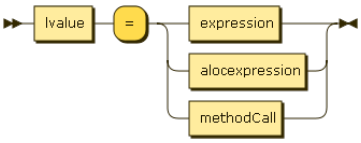



```
methodCall
        ::= ident ( '.' ident )+ '(' paramlist ')'
```

referenced by:

- atribStat
- factor
- statement

atribStat:




```
atribStat
        ::= lvalue '=' ( expression | alocexpression | methodCall )
```

referenced by:

- forStat
- statement

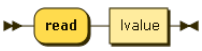
printStat:


```
printStat
        ::= 'print' expression
```

referenced by:

- statement

readStat:

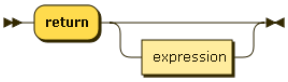



```
readStat ::= 'read' lvalue
```

referenced by:

- statement

returnStat:

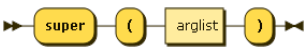



```
returnStat
        ::= 'return' expression?
```

referenced by:

- statement

superStat:

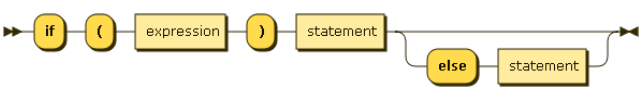



```
superStat
        ::= 'super' '(' arglist ')'
```

referenced by:

- statement

ifStat:

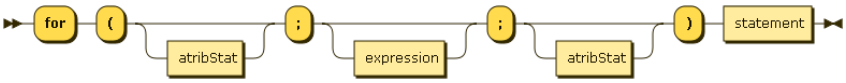



```
ifStat   ::= 'if' '(' expression ')' statement ( 'else' statement )?
```

referenced by:

- statement

forStat:




```
forStat  ::= 'for' '(' atribStat? ';' expression? ';' atribStat? ')' statement
```

referenced by:

- statement

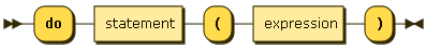
whileStat:


```
whileStat
         ::= 'while' '(' expression ')' statement
```

referenced by:

- statement

doWhileStat:

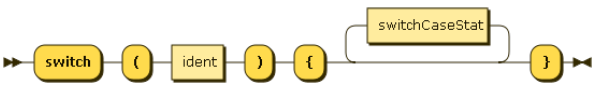



```
doWhileStat
         ::= 'do' statement '(' expression ')'
```

referenced by:

- statement

switchStat:

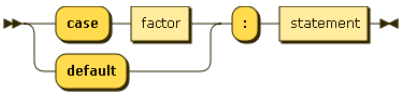



```
switchStat
         ::= 'switch' '(' ident ')' '{' switchCaseStat* '}'
```

referenced by:

- statement

switchCaseStat:

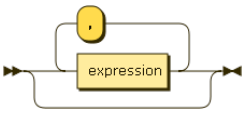



```
switchCaseStat
         ::= ( 'case' factor | 'default' ) ':' statement
```

referenced by:

- switchStat

arglist:

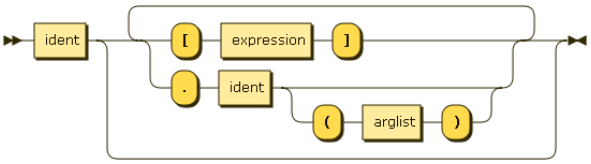



```
arglist  ::= ( expression ( ',' expression )* )?
```

referenced by:

- alocexpression
- lvalue
- superStat

lvalue:

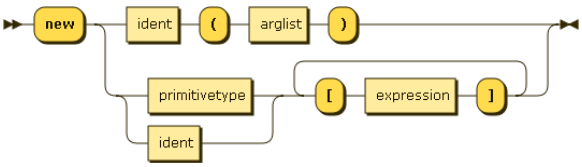



```
lvalue   ::= ident ( '[' expression ']' | '.' ident ( '(' arglist ')' )? )*
```

referenced by:

- atribStat
- factor
- readStat

alocexpression:

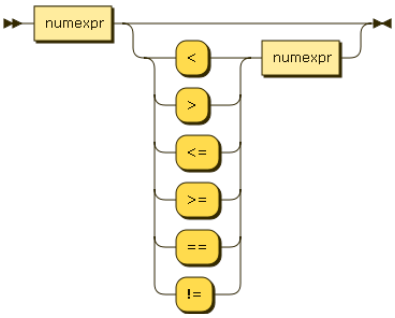



```
alocexpression
         ::= 'new' ( ident '(' arglist ')' | ( primitivetype | ident ) ( '[' expression ']' )+ )
```

referenced by:

- atribStat

expression:

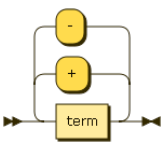



```
expression
         ::= numexpr ( ( '<' | '>' | '<=' | '>=' | '==' | '!=' ) numexpr )?
```

referenced by:

- alocexpression
- arglist
- atribStat
- doWhileStat
- factor
- forStat
- ifStat
- lvalue
- printStat
- returnStat
- whileStat

numexpr:

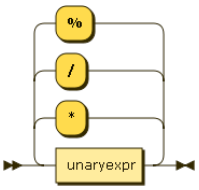



```
numexpr  ::= term ( ( '+' | '-' ) term )*
```

referenced by:

- expression

term:

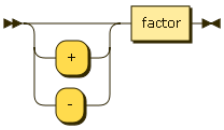



```
term     ::= unaryexpr ( ( '*' | '/' | '%' ) unaryexpr )*
```

referenced by:

- numexpr

unaryexpr:

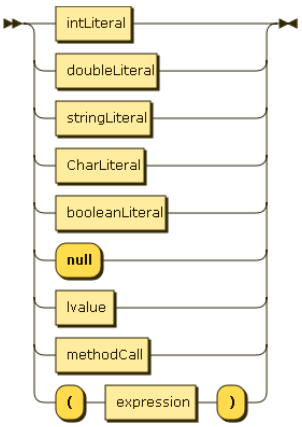



```
unaryexpr
         ::= ( '+' | '-' )? factor
```

referenced by:

- term

factor:

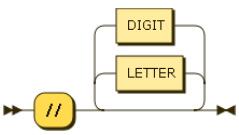



```
factor   ::= intLiteral
           | doubleLiteral
           | stringLiteral
           | CharLiteral
           | booleanLiteral
           | 'null'
           | lvalue
           | methodCall
           | '(' expression ')'
```

referenced by:

- switchCaseStat
- unaryexpr

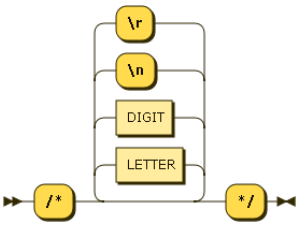
lineComment:




```
lineComment
         ::= '//' ( LETTER | DIGIT )*
```

no references

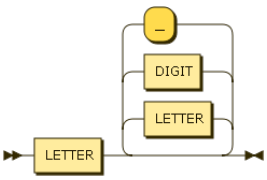
blockComment:




```
blockComment
        ::= '/*' ( LETTER | DIGIT | '\n' | '\r' )* '*/'
```

no references

ident:

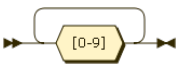



```
ident    ::= LETTER ( LETTER | DIGIT | '_' )*
```

referenced by:

- alocexpression
- classdecl
- lvalue
- methodCall
- methoddecl
- paramlist
- switchStat
- vardecl

DIGIT:

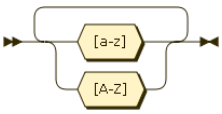



```
DIGIT    ::= [0-9]+
```

referenced by:

- blockComment
- ident
- intLiteral
- lineComment
- stringLiteral

LETTER:

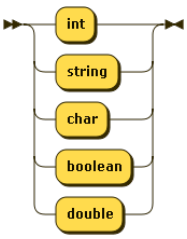



```
LETTER   ::= [a-zA-Z]+
```

referenced by:

- blockComment
- ident
- lineComment
- stringLiteral

primitivetype:

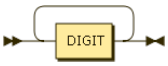



```
primitivetype
          ::= 'int'
            | 'string'
            | 'char'
            | 'boolean'
            | 'double'
```

referenced by:

- alocexpression
- methoddecl
- paramlist
- vardecl

intLiteral:

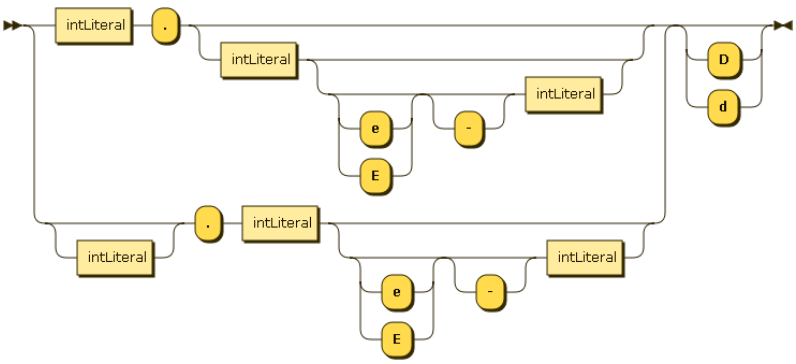



```
intLiteral
        ::= DIGIT+
```

referenced by:

- doubleLiteral
- factor

doubleLiteral:

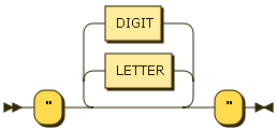



```
doubleLiteral
        ::= ( intLiteral '.' ( intLiteral ( ( 'e' | 'E' )? '-'? intLiteral )? )? | intLiteral? '.' intLiteral ( ( 'e' | 'E' )? '-'? intLiteral )? ) ( 'D' | 'd' )?
```

referenced by:

- factor

stringLiteral:

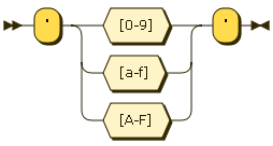



```
stringLiteral
        ::= '"' ( LETTER | DIGIT )* '"'
```

referenced by:

- factor

CharLiteral:

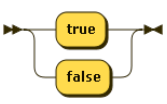



```
CharLiteral
        ::= "'" [0-9a-fA-F] "'"
```

referenced by:

- factor

booleanLiteral:




```
booleanLiteral
        ::= 'true'
          | 'false'
```

referenced by:

- factor

... generated by Railroad Diagram Generator