Minijavac, no left recursion, with precedence

```
Program -> MainClassDecl ClassDeclList
MainClassDecl -> class ID { public static void main ( String [ ] ID ) { StmtList } }
ClassDeclList -> ClassDecl ClassDeclList | EPSILON
ClassDecl -> class ID OptExtends { ClassVarDeclList MethodDeclList }
OptExtends -> extends ID | EPSILON
ClassVarDeclList -> ClassVarDec ClassVarDecList | EPSILON
MethodDecList -> MethodDecl MethodDecList | EPSILON
ClassVarDecl -> Type ID ;
MethodDecl -> public Type ID ( OptFormal ) { StmtList return Expr ; }
OptFormal: := Formal FormalList | EPSILON
FormalList -> , FormalList | EPSILON
OptExprList -> Expr ExprList | EPSILON
ExprList -> , Expr ExprList | EPSILON
Formal -> Type ID
Type -> int | boolean | ID
StmtList -> Stmt StmtList | EPSILON
Stmt -> Type ID = Expr ; | { StmtList } | if ( Expr ) Stmt else Stmt  | while ( Expr
) Stmt  | System.out.println ( Expr ) ; | ID = Expr ;
Expr -> Term1 Expr’
Expr’ -> or Term1 Expr’ | EPSILON
Term1 -> Term2 Term1’
Term1’ -> && Term2 Term1’ | EPSILON
Term2 -> Term3 Term2’
Term2’ -> == Term3 Term2’ | != Term3 Term2’ | EPSILON
Term3 -> Term4 Term3’
Term3’ -> > Term4 Term3’ | >= Term4 Term3’ | < Term4 Term3’ | <= Term4 Term3’ |
EPSILON
Term4 -> Term5 Term4’
Term4’ -> + Term5 Term4’ | - Term5 Term4’ | EPSILON
Term5 -> UnaryExpr Term5’
Term5’ -> * UnaryExpr Term5’ | / UnaryExpr Term5’ | EPSILON
UnaryExpr -> CallExpr | - UnaryExpr | ! UnaryExpr
CallExpr -> LiteralExpr CallExpr’ | new ID ( ) CallExpr’
CallExpr’ -> . ID ( OptExprList ) CallExpr' | EPSILON
ParenExpr -> ( Expr )
LiteralExpr -> ID | Integer | true | false | ParenExpr | this | null
```

First Sets

Non-Terminal Symbol	First Set
class	class
ID	ID
{	{
public	public
static	static
void	void
main	main
(	(
String	String
[	[
]	]
)	)
}	}
ε	ε
MethodDeclList	MethodDeclList
extends	extends
ClassVarDec	ClassVarDec
ClassVarDecList	ClassVarDecList
;	;
OptFormal	OptFormal
return	return
,	,
int	int
boolean	boolean
=	=
if	if
else	else
while	while
System.out.println	System.out.println
or	or
&&	&&
==	==
!=	!=
>	>
>=	>=
<	<
<=	<=
+	+
-	-
*	*
/	/
!	!
new	new
.	.
CallExpr'	CallExpr'

Integer	Integer
true	true
false	false
this	this
null	null
MainClassDecl	class
ClassDeclList	ε, class
ClassDecl	class
OptExtends	extends, ε
ClassVarDeclList	ClassVarDec, ε
MethodDecList	ε, public
MethodDecl	public
FormalList	,, ε
OptExprList	ε, -, !, new, ID, Integer, true, false, this, null, (
ExprList	,, ε
Type	int, boolean, ID
StmtList	ε, {, if, while, System.out.println, ID, int, boolean
Stmt	{, if, while, System.out.println, ID, int, boolean
Expr’	or, ε
Term1’	&&, ε
Term2’	==, !=, ε
Term3’	>, >=, <, <=, ε
Term4’	+, -, ε
Term5’	*, /, ε
UnaryExpr	-, !, new, ID, Integer, true, false, this, null, (
CallExpr	new, ID, Integer, true, false, this, null, (
CallExpr’	., ε
ParenExpr	(
LiteralExpr	ID, Integer, true, false, this, null, (
Program	class
ClassVarDecl	int, boolean, ID
Formal	int, boolean, ID
Term5	-, !, new, ID, Integer, true, false, this, null, (
Term4	-, !, new, ID, Integer, true, false, this, null, (
Term3	-, !, new, ID, Integer, true, false, this, null, (
Term2	-, !, new, ID, Integer, true, false, this, null, (
Term1	-, !, new, ID, Integer, true, false, this, null, (
Expr	-, !, new, ID, Integer, true, false, this, null, (

Follow Sets

Non-Terminal Symbol	Follow Set
Program	$
MainClassDecl	class, $
ClassDeclList	$
ClassDecl	class, $
OptExtends	{
ClassVarDeclList	MethodDeclList
MethodDecList	
ClassVarDecl	
MethodDecl	public
FormalList	
OptExprList	)
ExprList	)
Formal	
Type	ID
StmtList	}, return
Stmt	else, {, if, while, System.out.println, ID, int, boolean, }, return
Expr	), ;, ,
Expr’	), ;, ,
Term1	or,), ;, ,
Term1’	or,), ;, ,
Term2	&&, or,), ;, ,
Term2’	&&, or,), ;, ,
Term3	==, !=, &&, or,), ;, ,
Term3’	==, !=, &&, or,), ;, ,
Term4	>, >=, <, <=, ==, !=, &&, or,), ;, ,
Term4’	>, >=, <, <=, ==, !=, &&, or,), ;, ,
Term5	+, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
Term5’	+, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
UnaryExpr	*, /, +, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
CallExpr	*, /, +, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
CallExpr’	*, /, +, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
ParenExpr	., *, /, +, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,
LiteralExpr	., *, /, +, -, >, >=, <, <=, ==, !=, &&, or,), ;, ,

Predict Set

```
1      Program → MainClassDecl ClassDeclList  class
2      MainClassDecl → class ID { public static void main ( String [ ] ID ) {
StmtList } } class
3      ClassDeclList → ClassDecl ClassDeclList      class
4      ClassDeclList → ε  $
5      ClassDecl → class ID OptExtends { ClassVarDeclList MethodDeclList }    class
6      OptExtends → extends ID   extends
7      OptExtends → ε     {
8      ClassVarDeclList → ClassVarDec ClassVarDecList      ClassVarDec
9      ClassVarDeclList → ε      MethodDeclList
10     MethodDecList → MethodDecl MethodDecList     public
11     MethodDecList → ε
12     ClassVarDecl → Type ID ;  int, boolean, ID
13     MethodDecl → public Type ID ( OptFormal ) { StmtList return Expr ; }   public
14     FormalList → , FormalList ,
15     FormalList → ε
16     OptExprList → Expr ExprList     -, !, new, ID, Integer, true, false, this,
null, (
17     OptExprList → ε    )
18     ExprList → , Expr ExprList      ,
19     ExprList → ε )
20     Formal → Type ID   int, boolean, ID
21     Type → int   int
22     Type → boolean     boolean
23     Type → ID    ID
24     StmtList → Stmt StmtList  {, if, while, System.out.println, ID, int, boolean
25     StmtList → ε }, return
26     Stmt → Type ID = Expr ;   int, boolean, ID
27     Stmt → { StmtList }{
28     Stmt → if ( Expr ) Stmt else Stmt      if
29     Stmt → while ( Expr ) Stmt      while
30     Stmt → System.out.println ( Expr ) ;   System.out.println
31     Stmt → ID = Expr ; ID
32     Expr → Term1 Expr’ -, !, new, ID, Integer, true, false, this, null, (
33     Expr’ → or Term1 Expr’    or
34     Expr’ → ε    ), ;, ,
35     Term1 → Term2 Term1’      -, !, new, ID, Integer, true, false, this, null, (
36     Term1’ → && Term2 Term1’  &&
37     Term1’ → ε   or, ), ;, ,
38     Term2 → Term3 Term2’      -, !, new, ID, Integer, true, false, this, null, (
39     Term2’ → == Term3 Term2’  ==
40     Term2’ → != Term3 Term2’  !=
41     Term2’ → ε   &&, or, ), ;, ,
42     Term3 → Term4 Term3’      -, !, new, ID, Integer, true, false, this, null, (
43     Term3’ → > Term4 Term3’   >
44     Term3’ → >= Term4 Term3’  >=
45    Term3’ → < Term4 Term3’   <
46    Term3’ → <= Term4 Term3’  <=
47    Term3’ → ε   ==, !=, &&, or, ), ;, ,
48    Term4 → Term5 Term4’      -, !, new, ID, Integer, true, false, this, null, (
49    Term4’ → + Term5 Term4’   +
50    Term4’ → - Term5 Term4’   -
51    Term4’ → ε   >, >=, <, <=, ==, !=, &&, or, ), ;, ,
52    Term5 → UnaryExpr Term5’  -, !, new, ID, Integer, true, false, this, null, (
53    Term5’ → * UnaryExpr Term5’      *
54    Term5’ → / UnaryExpr Term5’      /
55    Term5’ → ε   +, -, >, >=, <, <=, ==, !=, &&, or, ), ;, ,
56    UnaryExpr → CallExpr      new, ID, Integer, true, false, this, null, (
57    UnaryExpr → - UnaryExpr   -
58    UnaryExpr → ! UnaryExpr   !
59    CallExpr → LiteralExpr CallExpr’ID, Integer, true, false, this, null, (
60    CallExpr → new ID ( ) CallExpr’ new
61    CallExpr’ → . ID ( OptExprList ) CallExpr'    .
62    CallExpr’ → ε      *, /, +, -, >, >=, <, <=, ==, !=, &&, or, ), ;, ,
63    ParenExpr → ( Expr )      (
64    LiteralExpr → ID   ID
65    LiteralExpr → Integer     Integer
66    LiteralExpr → true true
67    LiteralExpr → falsefalse
68    LiteralExpr → ParenExpr   (
69    LiteralExpr → this this
70    LiteralExpr → null null
```