Precedence: Consider the following expression:

3 + 4 * 5

Is the value of the expression **(3 + 4)*5 =35**?

Or, is the value of the expression **3 + (4 * 5)=23**?

Multiplication has *precedence* over addition and, hence, the value of the expression **3 + 4 * 5=23**.

The addition operator **+** and the multiplication operator ***** are *binary* operators: both operators require *two* operands. The addition operator looks left and right for the nearest operands and finds **3** on the left and **4** on the right. Similarly, the multiplication operator looks left and right for the nearest operands and finds **4** on the left and **5** on the right. Operand **4** is in contention. Operand **4** is simultaneously the right operand of the addition operator and the left operand of the multiplication operator. Both operators cannot be executed simultaneously. One operation must be performed first. The rules of precedence determine which operation is performed first and which operator is bound to operands in contention.

Associativity: When operators have the same precedence, associativity governs the order of evaluation. For example, **a + b + c + d** is evaluated **((a + b) +c) + d**. The addition operator associates to the left. Correctly parenthesized expressions that coerce the order of operations of a left-associative operator accumulate on the left.

Consider **a=b=c=d**. The expression **a=b=c=d** is evaluated **a=(b=(c=d))**. First **d** is assigned to **c**, then **c** is assigned to **b** and, finally, **b** is assign to **a**. The assignment operators associate to the right.

```
-----------------------------------------------------------------------
11  L  +, -        number, number        addition, subtraction
       +           string, any           string concatenation
-----------------------------------------------------------------------
10  L  <<          integer, integer      left shift
       >>          integer, integer      right shift with sign
extension
       >>>         integer, integer      right shift with zero
extension
-----------------------------------------------------------------------
 9  L  <, <=       number, number        less than, less than or equal
to
       >, >=       number, number        greater than, greater than or
equal to
       instanceof  reference, type       type comparison
-----------------------------------------------------------------------
 8  L  ==          primitive, primitive  equal (have identical values)
       !=          primitive, primitive  not equal (have different
values)
       ==          reference, reference  equal (refer to the same
object)
       !=          reference, reference  not equal (refer to different
objects)
-----------------------------------------------------------------------
 7  L  &           integer, integer      bitwise AND
       &           boolean, boolean      boolean AND
-----------------------------------------------------------------------
 6  L  ^           integer, integer      bitwise XOR
       ^           boolean, boolean      boolean XOR
-----------------------------------------------------------------------
 5  L  |           integer, integer      bitwise OR
       |           boolean, boolean      boolean OR
-----------------------------------------------------------------------
 4  L  &&          boolean, boolean      conditional AND
-----------------------------------------------------------------------
 3  L  ||          boolean, boolean      conditional OR
-----------------------------------------------------------------------
 2  R  ?:          boolean, any, any     conditional (ternary) operator
-----------------------------------------------------------------------
 1  R  =           variable, any         assignment
       *=, /=, %=  variable, any         assignment with operation
       +=, -=
       <<=, >>=,
       >>>=,
       &=, ^=, |=
```

Precedence	Associativity	Operator	Operand Types	Operation Performed
15	left-to-right	**.**	object, member	object member access
		[]	array, int	array element access
		(args)	method, arglist	method invocation
		++	variable	post-increment
		--	variable	post-decrement
14	right-to-left	**++**	variable	pre-increment

		--	variable	pre-decrement
		+	number	unary plus
		-	number	unary minus
		!	Boolean	Boolean NOT
13	right-to-left	**new**	class, arglist	object creation
		(type)	type, any	cast (type conversion)
12	left-to-right	*****	real | integer, real | integer	multiplication
		/	real | integer, real | integer	division
		%	integer, integer	remainder
11	left-to-right	**+**	real | integer, real | integer	addition
		-	real | integer, real | integer	subtraction
		+	string, any	string concatentation