**Java Operator Precedence Table**

This page lists C operators in order of *precedence* (highest to lowest). Their *associativity* indicates in what order operators of equal precedence in an expression are applied.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Associativity** |
| ( )  [ ] .  ++ -- | Parentheses (function call) (see Note 1)  Brackets (array subscript)  Postfix increment/decrement (see Note 2) | left-to-right |
| ++ --  + -  ! ~  (*type*) | Prefix increment/decrement  Unary plus/minus  Logical negation/bitwise complement  Cast (convert value to temporary value of *type*) | right-to-left |
| \* / % | Multiplication/division/modulus | left-to-right |
| + - | Addition/subtraction | left-to-right |
| << >> | Bitwise shift left, Bitwise shift right | left-to-right |
| < <=  > >= | Relational less than/less than or equal to  Relational greater than/greater than or equal to | left-to-right |
| == != | Relational is equal to/is not equal to | left-to-right |
| & | Bitwise AND | left-to-right |
| ^ | Bitwise exclusive OR | left-to-right |
| | | Bitwise inclusive OR | left-to-right |
| && | Logical AND | left-to-right |
| | | | Logical OR | left-to-right |
| ? : | Ternary conditional | right-to-left |
| =  += -=  \*= /=  %= &=  ^= |=  <<= >>= | Assignment  Addition/subtraction assignment  Multiplication/division assignment  Modulus/bitwise AND assignment  Bitwise exclusive/inclusive OR assignment  Bitwise shift left/right assignment | right-to-left |
| , | Comma (separate expressions) | left-to-right |
| **Note 1:** Parentheses are also used to group sub-expressions to force a different precedence; such parenthetical expressions can be nested and are evaluated from inner to outer.  **Note 2:** Postfix increment/decrement have high precedence, but the actual increment or decrement of the operand is delayed (to be accomplished sometime before the statement completes execution). So in the statement **y = x \* z++;** the current value of **z** is used to evaluate the expression (*i.e.,* **z++** evaluates to **z**) and **z** only incremented after all else is done. | | |
|  | | |

**Java Operator Precedence Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Precedence** | **Operator** | **Type** | **Associativity** |
| 15 | () [] · | Parentheses Array subscript Member selection | Left to Right |
| 14 | ++ -- | Unary post-increment Unary post-decrement | Left to Right |
| 13 | ++ -- + - ! ~ ( *type* ) | Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast | Right to left |
| 12 | \*  /  % | Multiplication Division Modulus | Left to right |
| 11 | + - | Addition Subtraction | Left to right |
| 10 | << >> >>> | Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension | Left to right |
| 9 | < <= > >= instanceof | Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only) | Left to right |
| 8 | == != | Relational is equal to Relational is not equal to | Left to right |
| 7 | & | Bitwise AND | Left to right |
| 6 | ^ | Bitwise exclusive OR | Left to right |
| 5 | | | Bitwise inclusive OR | Left to right |
| 4 | && | Logical AND | Left to right |
| 3 | || | Logical OR | Left to right |
| 2 | ? : | Ternary conditional | Right to left |
| 1 | = += -= \*= /= %= | Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment | Right to left |

*Larger number means higher precedence*.