**Operator Precedence in Java**

Java has well-defined rules for specifying the order in which the operators in an expression are evaluated when the expression has several operators. For example, multiplication and division have a higher precedence than addition and subtraction. Precedence rules can be overridden by explicit parentheses.  
  
Precedence order. When two operators share an operand the operator with the higher precedence goes first. For example, 1 + 2 \* 3 is treated as 1 + (2 \* 3), whereas 1 \* 2 + 3 is treated as (1 \* 2) + 3 since multiplication has a higher precedence than addition.  
Associativity. When an expression has two operators with the same precedence, the expression is evaluated according to its associativity. For example x = y = z = 17 is treated as x = (y = (z = 17)), leaving all three variables with the value 17, since the = operator has right-to-left associativity (and an assignment statement evaluates to the value on the right hand side). On the other hand, 72 / 2 / 3 is treated as (72 / 2) / 3 since the / operator has left-to-right associativity. Some operators are not associative: for example, the expressions (x <= y <= z) and x++-- are invalid.

Precedence and associativity of Java operators. The table below shows all Java operators from highest to lowest precedence, along with their associativity. Most programmers do not memorize them all, and even those that do still use parentheses for clarity.

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **Operator** | **Description** | **Associativity** |
| **16** | []  .  () | access array element  access object member  parentheses | left to right |
| **15** | ++  -- | unary post-increment  unary post-decrement | not associative |
| **14** | ++  --  +  -  !  ~ | unary pre-increment  unary pre-decrement  unary plus  unary minus  unary logical NOT  unary bitwise NOT | right to left |
| **13** | ()  new | cast  object creation | right to left |
| **12** | \* / % | multiplicative | left to right |
| **11** | + -  + | additive  string concatenation | left to right |
| **10** | << >>  >>> | shift | left to right |
| **9** | < <=  > >=  instanceof | relational | not associative |
| **8** | ==  != | equality | left to right |
| **7** | & | bitwise AND | left to right |
| **6** | ^ | bitwise XOR | left to right |
| **5** | | | bitwise OR | left to right |
| **4** | && | logical AND | left to right |
| **3** | || | logical OR | left to right |
| **2** | ?: | ternary | right to left |
| **1** | = += -=  \*= /= %=  &= ^= |=  <<= >>= >>>= | assignment | right to left |