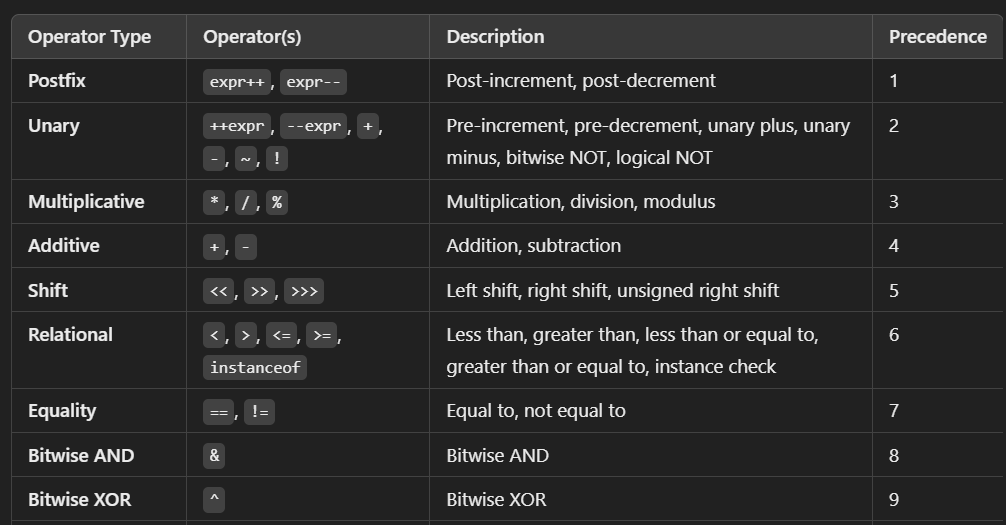
Chatgpt note (dsa)

In Java, operator precedence defines the order in which operators are evaluated in an expression. It dictates how operators like arithmetic, logical, relational, etc., are grouped and applied to operands.

Here is a detailed table covering Java's operator precedence, from the highest to the lowest:



**Additional Details:**

* **Postfix and Prefix Operators:**
* Postfix (expr++, expr--) has the highest precedence, meaning it is evaluated first in an expression, but they also return the value before incrementing or decrementing.
* Prefix (++expr, --expr) is evaluated next, modifying the variable first before returning the value.
* **Unary Operators:**
* These include the increment (++), decrement (--), unary plus (+), unary minus (-), logical NOT (!), and bitwise NOT (~). These operators work on a single operand and affect the operand's value or logical state.
* **Multiplicative Operators:**
* \*, /, and % perform multiplication, division, and modulo operations. They have higher precedence than addition and subtraction operators.
* **Relational and Equality Operators:**
* These operators are used to compare values. Relational operators (<, >, <=, >=) have higher precedence than equality operators (==, !=).
* **Bitwise and Logical Operators:**
* Bitwise operators (&, ^, |) operate at the bit level and have precedence over logical AND (&&) and logical OR (||), which deal with boolean expressions.
* **Assignment Operators:**
* Assignment operators, including =, +=, -=, \*=, and /=, have the lowest precedence in the operator hierarchy.

**Associativity:**

Most operators in Java follow **left-to-right** associativity, meaning they are evaluated from left to right in an expression. The only exceptions are:

* **Unary plus, unary minus, prefix increment/decrement, and logical NOT** (!), which are right to left.
* **Ternary conditional operator** (? :) also evaluates right to left.

**Example:**

Consider the following expression:

int x = 5 + 3 \* 2 - 4 / 2;

This would be evaluated as follows:

* First, 3 \* 2 (multiplication) is evaluated because multiplication has a higher precedence than addition and subtraction.
* Then, 4 / 2 (division) is evaluated because division also has higher precedence.
* Finally, the addition (5 + 6) and subtraction (11 - 2) are performed.

The result would be 9.

By understanding operator precedence and associativity, you can ensure that your expressions are evaluated as expected in Java!



Same precedence ( expression is evaluated left to right)

Revision:

exp++, exp-- HIGH

++exp, --exp , [+exp, -exp, ~exp **(Bitwise NOT)**, !exp][Unary] ( right to left)

\*, /, % (Arithmetic Operators)

+, -

Shift

<, >, <=, >= (relational)

== , != (equality)

Bitwise

&& (Logical operators)

||

?: ternary (right to left)

(Assignment Operators) LOW (right to left)

Arithmetic Operators

Unary

Binary

Relational Operators

Logical Operators

Assignment Operators

Mnemonic: "Aunt Uma Bought Red Lovely Apples"

**unary operator** (+, -, ~, !) in Java with explanation:

**✅ + Unary Plus**

(Usually does **nothing**, used to show positivity)

int a = +5; // same as just int a = 5;

System.out.println(a); // Output: 5

**✅ - Unary Minus**

(Used to make a number negative)

int a = 5;

int b = -a;

System.out.println(b); // Output: -5

**✅ ~ Bitwise NOT**

(Inverts each bit — works on integers)

int a = 5; // Binary: 0000 0101

int b = ~a; // 1111 1010 → -6 (in 2's complement)

System.out.println(b); // Output: -6

**✅ ! Logical NOT**

(Used to **reverse** a boolean value)

boolean flag = true;

System.out.println(!flag); // Output: false

**CamelCase**

* First letter **lowercase**, every next word starts with **uppercase**.
* **Used for:** variables, functions, and object names.
* **Example:** studentName, totalMarks, getUserInfo

**✅ PascalCase**

* **Each word starts with an uppercase** letter, including the first.
* **Used for:** class names, interfaces, and sometimes constructors.
* **Example:** Student, EmployeeDetails, AccountManager