Java - Operators and Expressions

What is Operators ?

Operators in Java are special symbols that act on one, two, or three operands to compute a result, enabling arithmetic, comparisons, logic, assignments, bit manipulation, and more.

Definition

- An operator is a symbol that performs a specific operation on operands (values/variables) and produces a result.
- Java groups operators into categories like arithmetic, assignment, relational, logical, unary, bitwise/shift, ternary, and type comparison.

Types of Operator's with examples

1) Arithmetic operators

- Purpose: basic math on numbers.
- Operators: `+, -, *, /, %`
- Example:
  - int a = 10, b = 3;
  - int sum = a + b; // 13
  - int rem = a % b; // 1

2) Assignment operators

- Purpose: assign or update variable values.
- Operators: `=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=`
- Example:

- int x = 5;
- x += 3; // x becomes 8
- x <<= 1; // x becomes 16 (explained)

Simple rule yaad rakho:

- `x << n` = `x * (2^n)`
- Example: `8 << 1 = 8 * 2 = 16`

3) Relational (comparison) operators

- Purpose: compare two values, result is boolean.
- Operators: `==, !=, <, <=, >, >=`
- Example:
  - boolean pass = marks >= 33;

4) Logical operators

- Purpose: combine boolean expressions, short-circuiting for && and ||.
- Operators: `&&, ||, !`
- Example:
  - if (isAdult && hasID) { /* allow */ }
  - boolean safe = !isNull;

5) Unary operators

- Purpose: operate on a single operand (sign, increment, decrement, boolean negate).
- Operators: +, -, ++, --, !
- Example:
  - `++i` // pre-increment , `i++` // post increment
  - `--i` // pre-decrement , `i--` //post-decrement
  - flag = !flag;

6) Bitwise operators

- Purpose: bit-level operations on integral types (byte, short, int, long).
- Operators: `& (AND), | (OR), ^ (XOR), ~ (NOT)`
- Example:
  - int mask = a & b;

- int toggled = x ^ 1;

7) Shift operators

- Purpose: shift bits left/right; >>> is logical right shift.
- Operators: `<<, >>, >>>`
- Example:
  - int d2 = n << 1; // multiply by 2
  - int h = n >> 1; // arithmetic divide by 2

8) Ternary (conditional) operator

- Purpose: compact if-else expression.
- Syntax: condition ? expr1 : expr2
- Example:
  - char grade = marks >= 90 ? 'A' : 'B';

> Arithmetic Operator's cannot work with booleans and % operator's can work on floats & doubles .

Precedence of Operators

Operator precedence in Java defines the order in which parts of an expression are evaluated; operators with higher precedence execute before those with lower precedence, and associativity breaks ties when precedence is equal.

Core idea

- Precedence : decides which operator goes first in mixed expressions
- (ex.- * before +, so `10 + 2 * 5` is 20, not 60).
- Associativity : when operators share the same precedence, evaluation order is determined by left-to-right or right-to-left rules
- (ex.- `72 / 2 / 3` is (72 / 2) / 3 due to left-to-right

Here is table Operator's Precedence

Java Operator Precedence Table

Operator	Description	Associativity

`()`
`[]`
`.`
	method invocation
	array subscript
	member access/selection
		left-to-right

`++` `--`	unary postfix increment unary postfix decrement	right-to-left

`++`
`--`
`+`
`-`
`!`
`~`
`( type )`
`new`
	unary prefix increment
	unary prefix decrement
	unary plus
	unary minus
	unary logical negation
	unary bitwise complement
	unary cast
	object creation
		right-to-left

`*`
`/`
`%`
	multiplication
	division
	modulus (remainder)
		left-to-right

`+` `-`	addition or string concatenation subtraction	left-to-right

`<<`
`>>`
`>>>`
	left shift
	arithmetic/signed right shift (sign bit duplicated)
	logical/unsigned right shift (zero shifted in)
		left-to-right

`<`
`<=`
`>`
`>=`
`instanceof`
	less than
	less than or equal to
	greater than
	greater than or equal to
	type comparison
		left-to-right

`==` `!=`	is equal to (equality) is not equal to (inequality)	left-to-right
`&`	bitwise AND boolean logical AND (no short-circuiting)	left-to-right
`^`	bitwise exclusive OR boolean logical exclusive OR	left-to-right
`|`	bitwise inclusive OR boolean logical inclusive OR (no short-circuiting)	left-to-right
`&&`	logical/conditional AND (short-circuiting)	left-to-right
`||`	logical/conditional OR (short-circuiting)	left-to-right
`?:`	conditional/ternary (if-then-else)	right-to-left

`=`
`+=`
`-=`
`*=`
`/=`
`%=`
`&=`
`^=`
`|=`
`<<=`
`>>=`
`>>>=`
	assignment
	addition assignment
	subtraction assignment
	multiplication assignment
	division assignment
	modulus/remainder assignment
	bitwise AND assignment
	bitwise exclusive OR assignment
	bitwise inclusive OR assignment
	bitwise left shift assignment
	bitwise arithmetic/signed right shift assignment
	bitwise logical/unsigned right shift assignment
		right-to-left

Result Data Types After Arithmetic Operation

following table summarise the resulting data after arithmetic operations on them —

byte + short ⇒ int

short + int ⇒ int

long + float ⇒ float

int + float ⇒ float

char + short ⇒ int

long + double ⇒ double

float + double ⇒ double

char + int ⇒ int

And Many more.....