**Java Operators - 20 Interview Questions**

**1. What are the different types of operators in Java? List them with examples.**

Java offers several categories of operators that allow you to manipulate data and control program logic. These include:

1. **Arithmetic Operators**: +, -, \*, /, % — perform basic math.  
   Example: int sum = a + b;
2. **Relational Operators**: ==, !=, >, <, >=, <= — compare two values.  
   Example: if (x < y)
3. **Logical Operators**: &&, ||, ! — combine multiple boolean expressions.  
   Example: (x > 5 && y < 10)
4. **Assignment Operators**: =, +=, -=, \*=, /=, %= — assign and modify values.  
   Example: a += 3; // a = a + 3
5. **Unary Operators**: ++, --, +, -, ! — operate on a single operand.  
   Example: x++
6. **Bitwise Operators**: &, |, ^, ~, <<, >> — operate on binary digits.  
   Example: a & b

These operators are used in expressions, control structures, loops, and methods to write efficient and meaningful code.

**2. What is the difference between == and = in Java?**

= is the assignment operator, used to assign values to variables.

int a = 5; // assigns value 5 to variable a

== is a comparison operator, used to compare two values for equality.

int a = 5, b = 6;

System.out.println(a == b); // prints false

Common mistake: using = in conditional expressions, which can lead to logical errors or compile-time issues.  
Correct:

if (a == b) { /\* do something \*/ }

**3. Explain the use of the % (modulus) operator with an example.**

The % operator returns the **remainder** after division. It is useful in a wide range of scenarios such as checking for even/odd numbers or cycling through a range.

Example:

int a = 17;

int b = 5;

System.out.println(a % b); // Output: 2

Common use cases:

* Check even/odd: num % 2 == 0
* Divisibility tests: if (num % 3 == 0)
* Index wrapping in arrays: index % array.length

**4. How does the ++ operator work? Differentiate between a++ and ++a.**

The ++ operator increases the value of a variable by 1.

* a++ (post-increment): use the value first, then increment.
* ++a (pre-increment): increment first, then use the value.

Example:

int a = 5;

System.out.println(++a); // 6

System.out.println(a++); // 6 (a becomes 7 after this)

System.out.println(a); // 7

Use this operator in loops, counters, and conditional logic to control iterations and flow.

**5. What is the role of logical operators in Java?**

Logical operators (&&, ||, !) are used to combine multiple boolean expressions and control the flow of execution based on multiple conditions.

* && (logical AND): returns true only if both conditions are true.
* || (logical OR): returns true if at least one condition is true.
* ! (logical NOT): inverts the boolean value.

Example:

if (age > 18 && citizen) {

System.out.println("Eligible to vote");

}

They’re essential in complex decision-making and flow control.

**6. Explain bitwise operators with examples.**

Bitwise operators operate on bits (0s and 1s) of integer types:

* & (AND), | (OR), ^ (XOR), ~ (NOT)
* << (left shift), >> (right shift)

Example:

int a = 5; // 0101

int b = 3; // 0011

System.out.println(a & b); // 1 (0001)

Bitwise operators are useful in low-level programming like cryptography, compression, and optimization.

**7. What is a ternary operator in Java?**

The ternary operator (? :) is a shorthand for if-else statements.  
Syntax:

condition ? expr1 : expr2

Example:

int max = (a > b) ? a : b;

It returns expr1 if the condition is true, else returns expr2. It’s useful for simple conditional assignments.

**8. What is operator precedence? Give examples.**

Operator precedence defines the order in which operators are evaluated.  
For example, \* has higher precedence than +:

int result = 2 + 3 \* 4; // 2 + 12 = 14

Use parentheses to override default precedence:

int result = (2 + 3) \* 4; // 5 \* 4 = 20

Understanding precedence avoids logic errors in expressions.

**9. Explain short-circuit evaluation with logical operators.**

Short-circuiting means Java skips evaluating the second condition if the result is already known:

* &&: if first condition is false, second is skipped.
* ||: if first condition is true, second is skipped.

Example:

if (x != 0 && 10/x > 1) // avoids divide-by-zero error

Used for optimizing conditions and avoiding unnecessary computations.

**10. What are compound assignment operators? Give examples.**

Compound assignment operators combine arithmetic operations with assignment:

* +=, -=, \*=, /=, %=

Example:

int a = 10;

a += 5; // a = a + 5 => 15

They simplify code and are frequently used in loops, counters, and cumulative logic.

**11. Can operators be overloaded in Java?**

No, Java does not support operator overloading, unlike C++. Each operator has predefined behavior.  
For example, + is overloaded internally for String concatenation but cannot be customized for custom objects.  
This restriction maintains simplicity and avoids confusion.

**12. How do relational operators work in Java?**

Relational operators compare two operands and return a boolean result:

* >, <, >=, <=, ==, !=

Example:

if (a >= b)

System.out.println("a is greater or equal to b");

Used in if, while, and for conditions for decision-making.

**13. What is the result of dividing an integer by zero?**

In Java:

* Integer division by zero causes a runtime ArithmeticException.
* Floating-point division by zero results in Infinity or NaN.

Example:

int a = 5;

int b = 0;

System.out.println(a / b); // throws ArithmeticException

Always validate divisor before performing division.

**14. What is the difference between | and || in Java?**

* | is a bitwise OR or boolean OR without short-circuit.
* || is a logical OR with short-circuit behavior.

Example:

boolean result = true | false; // evaluates both

boolean result2 = true || false; // skips evaluating false

Prefer || for performance and logical conditions.

**15. Can we use arithmetic operators on characters in Java?**

Yes, Java allows arithmetic operations on char since it's internally treated as an integer (Unicode value).

Example:

char ch = 'A';

System.out.println(ch + 1); // prints 66

Useful in character manipulation, loops, and encoding.

**16. How does the instanceof operator work?**

instanceof is used to test whether an object is an instance of a specific class or subclass.

Example:

if (obj instanceof String) {

System.out.println("It's a String");

}

Returns true if the object is compatible with the class type.

**17. What is the use of shift operators in Java?**

Shift operators move bits left or right:

* << shifts left (multiplies by 2)
* >> shifts right (divides by 2, preserves sign)
* >>> shifts right (fills with 0, unsigned)

Example:

int x = 4; // 0100

System.out.println(x << 1); // 8

Used in performance-critical applications.

**18. Explain the precedence and associativity of operators.**

Precedence determines evaluation order. Associativity resolves ties:

* Left-to-right: +, -, \*, /
* Right-to-left: assignment operators

Example:

int x = 5;

x += 3 \* 2; // x = x + (3 \* 2) = 11

Understanding both is critical for writing correct expressions.

**19. How are logical NOT and bitwise NOT different?**

* ! (logical NOT) reverses boolean value.
* ~ (bitwise NOT) inverts bits.

Example:

boolean flag = true;

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

int a = 5; // 0101

System.out.println(~a); // -6

Choose based on data type and logic requirement.

**20. What happens when you mix data types in an expression?**

Java performs type promotion:

* Smaller types (byte, short) are promoted to int.
* Mixed types are converted to the larger type.

Example:

int a = 5;

double b = 6.2;

System.out.println(a + b); // 11.2

Ensure compatibility when mixing types to avoid precision loss or errors.