JAVA OPERATORS

Introduction to Operators

In Java, **operators** are special symbols used to perform operations on variables and values.

Java operators are **fundamental building blocks** of programming.
Understanding operators helps in:

- Writing correct logic
- Avoiding runtime errors
- Improving performance

Example:

int a = 10 + 5;

Here:

- + → Operator
- 10 and 5 → Operands
- = → Assignment Operator

Types of Operators in Java

Category	**Operators**
Arithmetic	**+ - * / %**
Relational	**> < >= <= == !=**
Logical	**&&**
Bitwise	**&**
Assignment	**= += , -=, *=, /= , %=**
Ternary	**?:**
Increment and decrement	**++ and --**

Let’s discuss all one by one :

Arithmetic Operators

Arithmetic operators are used to perform basic mathematical calculations.

List of Arithmetic Operators

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)

Example Program

```
class ArithmeticDemo {
    public static void main(String[] args) {
        int a = 20, b = 6;
        System.out.println(a + b);  // 26
        System.out.println(a - b);  // 14
        System.out.println(a * b);  // 120
        System.out.println(a / b);  // 3
        System.out.println(a % b);  // 2
    }
}
```

Important Note

- **Division of two integers gives an integer result**
- **% is useful in checking even/odd numbers**

- Modulus operator is used to find the remainder.

Relational Operators

Relational operators are used to compare two values.
They always return boolean (true / false).

List of Relational Operators

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
==	Equal to
!=	Not equal to

Example Program

```
class RelationalDemo {
   public static void main(String[] args) {
      int x = 10, y = 20;
      System.out.println(x > y);   // false
      System.out.println(x < y);   // true
      System.out.println(x == y);  // false
      System.out.println(x != y);  // true
   }
}
```

Common Mistake

❌ = is assignment
✅ == is comparison

Logical Operators

Logical operators are used to combine multiple conditions.

List of Logical Operators

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

Example Program

```
class LogicalDemo {

   public static void main(String[] args) {

       int age = 20;

       System.out.println(age > 18 && age < 25); //
true

       System.out.println(age < 18 || age > 60); //
false

       System.out.println(!(age > 18));        // false

   }

}
```

Bitwise Operators

Bitwise operators work on binary representation of numbers.

List of Bitwise Operators

Operator	Meaning
&	Bitwise AND
^	Bitwise XOR
~	Bitwise Complement
<<	Left Shift
>>	Right Shift

Example

```
class BitwiseDemo {
   public static void main(String[] args) {
      int a = 5, b = 3;
      System.out.println(a & b); // 1
      System.out.println(a | b); // 7
      System.out.println(a ^ b); // 6
      System.out.println(a << 1); // 10
      System.out.println(a >> 1); // 2
   }
}
```

Binary Explanation

- 5 → 101

- **3 → 011**

Assignment Operators

Assignment operators are used to assign values to variables.

List of Assignment Operators

Operator	**Meaning**
=	**Assign**
+=	**Add and assign**
-=	**Subtract and assign**
***=**	**Multiply and assign**
/=	**Divide and assign**
%=	**Modulus and assign**

Example Program

```
class AssignmentDemo {
    public static void main(String[] args) {
        int a = 10;
        a += 5;   // a = a + 5
        a -= 3;   // a = a - 3
        a *= 2;   // a = a * 2
        System.out.println(a); // 24
    }
}
```

---

7. Ternary Operator

Ternary operator is a shorthand for if–else.

Syntax

condition ? value1 : value2;

Example

```
class TernaryDemo {
   public static void main(String[] args) {
      int a = 10, b = 20;
      int max = (a > b) ? a : b;
      System.out.println(max); // 20
   }
}
```

Advantages

✔ Short code
✔ Easy readability
✔ Faster execution

---

8. Operator Precedence & Associativity

Operator Precedence

It defines which operator is evaluated first.

Associativity

It defines direction of evaluation (Left → Right or Right → Left).

Operator Precedence Table (High → Low)

Precedence	Operator
Highest	()
	! ~ ++ --

	* / %
	+ -
	<< >> >>>
	< <= > >=
	== !=
	&
	^
	`
	&&
	`
	?:
Lowest	= += -= *=

Example

```
int result = 10 + 5 * 2;

System.out.println(result);
```

✓ Output: 20
(because * has higher precedence than +)

Associativity Example

```
int a = 10, b = 5, c = 2;

int result = a - b - c;
```

✓ Evaluated Left → Right
✓ (10 - 5) - 2 = 3

Increment and Decrement Operators in Java

In Java, increment (++) and decrement (--) operators are unary operators used to increase or decrease the value of a variable by 1.

1️⃣Increment Operator (++)

Increases the value of a variable by 1.

🔹 Types of Increment

a) Pre-Increment (++a)

- **First increases the value**
- **Then uses the updated value**

```
int a = 5;
System.out.println(++a); // 6
System.out.println(a);   // 6
```

b) Post-Increment (a++)

- **First uses the current value**
- **Then increases the value**

```
int a = 5;
System.out.println(a++); // 5
System.out.println(a);   // 6
```

2️⃣Decrement Operator (--)

Decreases the value of a variable by 1.

🔹 Types of Decrement

a) Pre-Decrement (--a)

- First decreases the value
- Then uses the updated value

```
int a = 5;

System.out.println(--a); // 4

System.out.println(a);   // 4
```

b) Post-Decrement (a--)

- First uses the current value
- Then decreases the value

```
int a = 5;

System.out.println(a--); // 5

System.out.println(a);   // 4
```

🔁 Difference Between Pre and Post

Operator	Action Order	Example (a=5)	Output
++a	Increment → Use	++a	6
a++	Use → Increment	a++	5
--a	Decrement → Use	--a	4
a--	Use → Decrement	a--	5

▯ Important Points (Exam / Viva)

- ++ and -- work only with variables, not constants.

- **5++; // ❌ Invalid**
- **Used in loops (for, while)**
- **Increase/decrease value by exactly 1**
- **Faster and cleaner than a = a + 1**