Logical Operators in Java (AND, OR, NOT)

1. Introduction

- Logical operators are used to **combine two or more conditions** (boolean expressions) and return a boolean result (`true` or `false`).
- They are mostly used in **decision-making** (if-else, loops).

---

2. What are Logical Operators?

- Operators that work on **boolean values** (`true/false`).
- They help evaluate multiple conditions together.

---

3. Types of Logical Operators

3.1 Logical AND (`&&`)

- Returns `true` only if **both conditions are true**.
- If any one is false then output is false .
- Syntax:

```
(condition1 && condition2)
```

- Example:

```
if (age >= 18 && citizen == true) {
    System.out.println("Eligible to vote");
}
```

3.2 Logical OR (`||`)

- Returns `true` if **at least one condition is true**.
- Returns `false` only if **both conditions are false**.
- Syntax:

```
(condition1 || condition2)
```

- Example:

```
if (marks >= 40 || graceMarks == true) {
    System.out.println("Student Passed");
}
```

3.3 Logical NOT (`!`)

- Reverses the boolean value.
- Basically they are changes each others.
- Syntax:

```
!(condition)
```

- Example:

```
if (!(isLoggedIn)) {
    System.out.println("Please login first");
}
```

4. Truth Tables

4.1 AND (`&&`)

A	B	A && B
true	true	true
true	false	false
false	true	false

A	B	A && B
false	false	false

4.2 OR (`||`)

A	B	A || B
true	true	true
true	false	true
false	true	true
false	false	false

4.3 NOT (`!`)

A	!A
true	false
false	true

5. Difference Between Logical & Bitwise Operators

- `&&` vs `&` → `&&` is **short-circuiting** (stops evaluation if first condition is false), while `&` always evaluates both.
- `||` vs `|` → `||` is **short-circuiting** (stops evaluation if first condition is true), while `|` always evaluates both.

6. Precedence of Logical Operators

Order of evaluation:

1. `!` (NOT) – Highest
2. `&&` (AND)
3. `||` (OR) – Lowest

Example:

```
boolean result = true || false && !false;
// Evaluated as: true || (false && true) → true || false → true
```

7. Use Cases in Java Programs

- Validations (login, eligibility checks)
- Complex conditions in loops
- Input checks (null, empty, etc.)

8. Common Mistakes & Best Practices

✅ Use `&&` and `||` instead of `&` and `|` unless bitwise operation is intended.

✅ Always use parentheses to make conditions clear.

❌ Don't forget that `!` only applies to one condition.

❌ Avoid long, unreadable condition chains.

---

9. Example Programs

Example 1: AND Operator

```
public class AndExample {
    public static void main(String[] args) {
        int age = 20;
        boolean citizen = true;

        if (age >= 18 && citizen) {
            System.out.println("You are eligible to vote.");
        } else {
            System.out.println("You are not eligible to vote.");
        }
    }
}
```

Example 2: OR Operator

```
public class OrExample {
    public static void main(String[] args) {
        int marks = 35;
        boolean graceMarks = true;

        if (marks >= 40 || graceMarks) {
            System.out.println("You passed the exam.");
        } else {
            System.out.println("You failed the exam.");
        }
    }
}
```

Example 3: NOT Operator

```
public class NotExample {
    public static void main(String[] args) {
        boolean isLoggedIn = false;

        if (!isLoggedIn) {
            System.out.println("Please login first.");
        } else {
            System.out.println("Welcome to your account.");
        }
    }
}
```

10. Summary

- Logical operators help combine multiple conditions.
- `&&` (AND) → True only if **all conditions are true**.
- `||` (OR) → True if **at least one condition is true**.
- `!` (NOT) → Reverses the result.
- Use `&&` and `||` for short-circuiting efficiency.
- Logical operators are **essential in decision-making and validation** in Java.