Conditionals in java (If Else Ladder , Switch Statement's ....)

What are Conditionals?

- Conditional statements are constructs that evaluate a **boolean expression** (true/false) and direct the flow of execution accordingly.
- They help in making decisions such as:
  - Executing a block *if* a condition is true.
  - Choosing between two alternative paths.
  - Handling multiple conditions.

Types of Conditionals in Java

1. The `if` Statement

- Simplest form of conditional.
- Executes a block of code if the given condition evaluates to `true`.

Syntax:

```
if (condition) {
    // code block if condition is true
}
```

Example:

```
int age = 20;
if (age >= 18) {
    System.out.println("Eligible to vote");
}
```

Key Points:

- The `condition` must return a boolean (`true` or `false`).
- If condition is false, the block is skipped.

---

2. The `if-else` Statement

- Provides two-way decision-making.
- If the `if` condition is false, `else` block runs.

Syntax:

```
if (condition) {
    // code if true
} else {
    // code if false
}
```

Example:

```
int number = 7;
if (number % 2 == 0) {
    System.out.println("Even number");
} else {
    System.out.println("Odd number");
}
```

---

3. The `if-else-if` Ladder

- Used when multiple conditions need to be checked sequentially.
- Program checks each condition from top to bottom; executes the first true block, then skips the rest.

Syntax:

```
if (condition1) {
    // code block 1
} else if (condition2) {
    // code block 2
} else if (condition3) {
    // code block 3
} else {
    // default case
}
```

Example:

```
int marks = 85;
if (marks >= 90) {
```

```
    System.out.println("Grade A");
} else if (marks >= 75) {
    System.out.println("Grade B");
} else if (marks >= 50) {
    System.out.println("Grade C");
} else {
    System.out.println("Failed");
}
```

---

4. Nested `if` Statements

- An `if` inside another `if`.
- Useful when one condition depends on another.

Example:

```
int age = 25;
if (age >= 18) {
    if (age >= 21) {
        System.out.println("Eligible for marriage");
    } else {
        System.out.println("Eligible to vote but not for marriage");
    }
}
```

5. The `switch` Statement

- Used when multiple possible values of a single variable need different execution paths.
- Cleaner than multiple `if-else` statements for discrete values.

Syntax:

```
switch (variable) {
    case value1:
        // code block 1
        break;
    case value2:
        // code block 2
        break;
    default:
        // default block
}
```

Example:

```
int day = 5;
switch (day) {
    case 1: System.out.println("Monday");
    break;
    case 2: System.out.println("Tuesday");
    break;
    case 3: System.out.println("Wednesday");
    break;
    case 4: System.out.println("Thursday");
    break;
    case 5: System.out.println("Friday");
    break;
    case 6: System.out.println("Saturday");
    break;
    case 7: System.out.println("Sunday");
    break;
    default: System.out.println("Invalid day");
}
```

Notes:

- `break;` prevents fall-through to the next case.
- `default` in switch is optional but recommended.
- Valid with `int`, `char`, `String`, and `enum` in Java.

Switch Expressions (Morden Java 12+)

- A modern upgrade of `switch`.
- Uses `>` and can **return values** directly.

Example:

```
String day = switch (5) {
    case 1 -> "Monday";
    case 2 -> "Tuesday";
    case 3 -> "Wednesday";
    case 4 -> "Thursday";
    case 5 -> "Friday";
    case 6 -> "Saturday";
    case 7 -> "Sunday";
    default -> "Invalid";
};
System.out.println(day);
```

---

6. The Ternary Operator (`?:`)

- A shorthand version of an `if-else`.
- Evaluates a condition and chooses one of two values.

Syntax:

```
javavariable = (condition) ? valueIfTrue : valueIfFalse;
```

Example:

```
int number = 10;
String result = (number % 2 == 0) ? "Even" : "Odd";
System.out.println(result);
```

Quick Recap

- `if` → simple decision
- `if-else` → two-way decision
- `if-else-if` → multiple decisions
- **Nested if** → decision inside a decision
- `switch` → cleaner choice among fixed values
- **Switch expressions (Java 12+)** → modern and concise
- **Ternary** `?:` → compact one-line conditional