1. Pre-requisite: Boolean Expressions (Asking True/False Questions)

Before a program can make a decision, it needs to **ask a question**. In Java, these questions must always evaluate to a `boolean` value: either `true` or `false`.

To build these questions, we introduce **two new sets of operators**:

Relational Operators (Comparing Values)

Operator	Meaning	Example (`int age = 20;`)	Result
`==`	Equal to *(double equals!)*	`age == 18`	`false`
`!=`	Not equal to	`age != 18`	`true`
`>` / `<`	Greater than / Less than	`age > 18`	`true`
`>=` / `<=`	Greater/Less than or equal to	`age <= 20`	`true`

> **Teacher's Note:** Remember the **Golden Rule** from Phase 2:
>
> - `=` → assign a value (put the value in the box)
> - `==` → compare two values (are these two the same?) Mixing these up is the **#1 beginner bug**.

Logical Operators (Combining Questions)

Sometimes one question isn't enough. We use these to **chain conditions together**:

- `&&` (AND) → true only if **both sides** are true
- `||` (OR) → true if **at least one** side is true
- `!` (NOT) → flips `true` to `false` and `false` to `true`

---

2. The `if` Statement (The Basic Decision)

The `if` statement is the simplest way to make a decision.

- If the boolean expression inside `()` is `true`, Java runs the code inside `{}`.
- If it's `false`, Java **skips that block entirely**.

```
int score = 85;

// Question: Is the score 80 or higher?
if (score >= 80) {
    System.out.println("You passed the test!");
}
```

---

3. The `if-else` Statement (The Backup Plan)

What if we want something specific to happen when the condition is `false`? That's where `else` comes in.

- Guarantees that **one of the two blocks** runs, but **never both**.

```
boolean hasTicket = false;

if (hasTicket) {
    System.out.println("Welcome to the concert!");
} else {
    System.out.println("Sorry, you cannot enter.");
}
```

> **Tip:** In Java, `if (hasTicket == true)` can simply be written as `if (hasTicket)` — cleaner and easier to read.

---

4. The `else if` Chain (Multiple Options)

Life rarely has just **two options**. When you have several possibilities, use an `else if` chain.

- Java evaluates them **from top to bottom**.
- The **first condition that is true** runs, and the rest are skipped.

```
int temperature = 75;

if (temperature > 90) {
    System.out.println("It's boiling outside!");
} else if (temperature > 70) {
    // This runs because 75 > 70
    System.out.println("The weather is perfect.");
} else if (temperature > 50) {
    System.out.println("Bring a light jacket.");
} else {
    System.out.println("It is freezing!");
}
```

Tasks

Task

Look at this code. What will the program print out?

```
int age = 15;
boolean hasVIPPass = true;

if (age >= 18) {
    System.out.println("Come on in!");
```

```
} else if (hasVIPPass == true) {
    System.out.println("Right this way, VIP!");
} else {
    System.out.println("Sorry, come back when you are older.");
}
```

(Answer: It prints `"Right this way, VIP!"`. The first condition (`age >= 18`) is `false`, so Java skips it. It checks the `else if`, which is `true`, runs that block, and then instantly skips the rest!)

Related Topics

- Type Casting
- Loops