**Java Control Flow Statements – Detailed Notes**

**🧭 What are Control Flow Statements?**

Control Flow statements **determine the order** in which statements are **executed** in a Java program. Instead of running code from top to bottom, control flow allows decisions, loops, and branches.

**🔰 Categories of Control Flow Statements:**

1. **Conditional (Decision-making) Statements**
2. **Looping (Iteration) Statements**
3. **Jump (Branching) Statements**

**1️⃣ Conditional (Decision-Making) Statements**

Used to make **decisions** in your code.

**A. if Statement**

if (condition) {

// block of code if condition is true

}

✅ **Example:**

int age = 18;

if (age >= 18) {

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

}

🧠 **Real-life Analogy:**  
"If it's raining, take an umbrella."

**B. if-else Statement**

if (condition) {

// true block

} else {

// false block

}

✅ **Example:**

int marks = 35;

if (marks >= 35) {

System.out.println("Pass");

} else {

System.out.println("Fail");

}

**C. if-else-if Ladder**

Used for multiple conditions.

if (condition1) {

// block 1

} else if (condition2) {

// block 2

} else {

// default block

}

✅ **Example:**

int marks = 85;

if (marks >= 90) {

System.out.println("Grade A");

} else if (marks >= 75) {

System.out.println("Grade B");

} else {

System.out.println("Grade C");

}

**D. switch Statement**

Replaces long if-else-if ladders when checking a single variable for multiple values.

switch (variable) {

case value1:

// code

break;

case value2:

// code

break;

...

default:

// default code

}

✅ **Example:**

int day = 3;

switch (day) {

case 1: System.out.println("Monday"); break;

case 2: System.out.println("Tuesday"); break;

case 3: System.out.println("Wednesday"); break;

default: System.out.println("Invalid");

}

📌 **Important Points:**

* break exits the switch.
* default is optional.

**2️⃣ Looping (Iteration) Statements**

Used to repeat code until a condition is false.

**A. while Loop**

while (condition) {

// code to execute repeatedly

}

✅ **Example:**

int i = 1;

while (i <= 5) {

System.out.println(i);

i++;

}

📌 Use when the **number of iterations is not known in advance.**

**B. do-while Loop**

Executes the loop body **at least once**.

do {

// code

} while (condition);

✅ **Example:**

int i = 1;

do {

System.out.println(i);

i++;

} while (i <= 5);

**C. for Loop**

Best when the **number of iterations is known**.

for (initialization; condition; update) {

// code

}

✅ **Example:**

for (int i = 1; i <= 5; i++) {

System.out.println(i);

}

**D. Enhanced for-each Loop**

Used for arrays and collections.

for (type variable : array) {

// code

}

✅ **Example:**

int[] nums = {10, 20, 30};

for (int num : nums) {

System.out.println(num);

}

**3️⃣ Jump (Branching) Statements**

Used to **change the normal flow** of execution.

**A. break Statement**

Used to **exit a loop** or a switch early.

✅ **Example in loop:**

for (int i = 1; i <= 10; i++) {

if (i == 5) break;

System.out.println(i);

}

**B. continue Statement**

Skips the current iteration and moves to the next.

✅ **Example:**

for (int i = 1; i <= 5; i++) {

if (i == 3) continue;

System.out.println(i);

}

**C. return Statement**

Exits from a method and optionally returns a value.

✅ **Example:**

public static int square(int x) {

return x \* x;

}

**🔄 Flowchart Summary**

Start

↓

[Condition?]

↓ ↘

True False

↓ ↓

[Block] Skip

↓

[Next Step]

**🔍 Real-world Analogy Summary**

| **Java Concept** | **Real Life Example** |
| --- | --- |
| if | If it's sunny, go outside |
| if-else | If hungry, eat else sleep |
| switch | Menu selection in a restaurant |
| while | Keep drinking water while thirsty |
| for | Count from 1 to 10 |
| do-while | Eat at least once before checking hunger |
| break | Walk out of class if bored |
| continue | Skip a question in an exam |
| return | Give back result from function/method |

**💡 Best Practices**

* Always include a **termination condition** in loops.
* Use switch instead of long if-else ladders where appropriate.
* Avoid infinite loops unless intentional.
* Use {} brackets even for single-line blocks for readability.