## **Conditional statements in java**

**if Statement**

The if statement is a single conditional based statement that executes only if the provided condition is true.

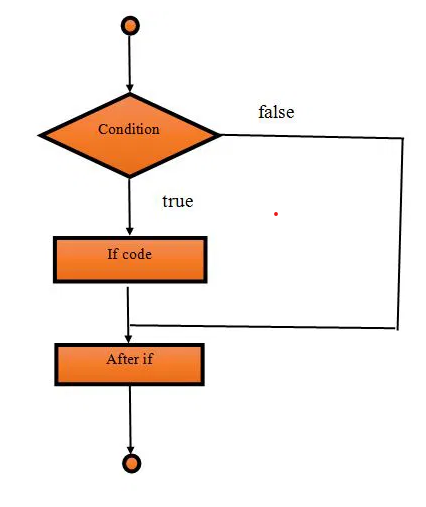
**Syntax:**

if(condition)

{

//code

}



**if-else Statement**

The if-else statement is used for testing condition. If the condition is true, if block executes otherwise else block executes.

It is useful in the scenario when we want to perform some operation based on the **false** result.

The else block execute only when condition is **false**.

**Syntax:**

if(condition)

{

//code for true

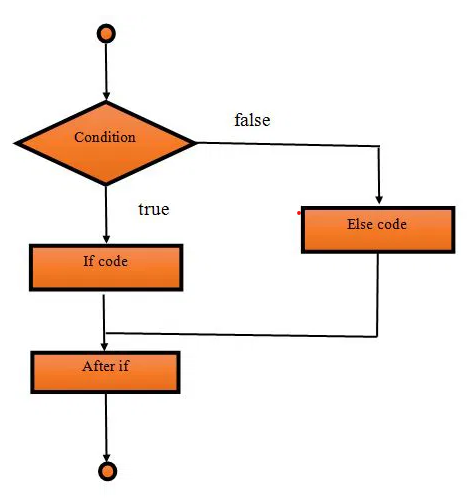
}

else

{

//code for false

}



**if-else-if ladder Statement**

In Java, the if-else-if ladder statement is used for testing conditions. It is used for testing one condition from multiple statements.

When we have multiple conditions to execute then it is recommend to use if-else-if ladder.

**Syntax:**

if(condition1)

{

//code for if condition1 is true

}

else if(condition2)

{

//code for if condition2 is true

}

else if(condition3)

{

//code for if condition3 is true

}

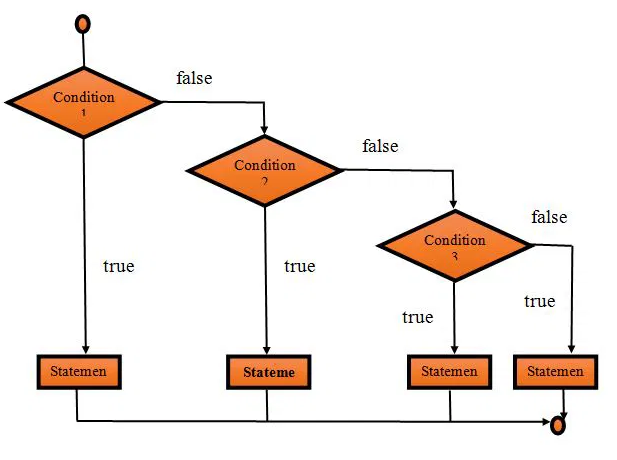
...

else

{

//code for all the false conditions

}



**Nested if statement**

In Java, the Nested if statement is a if inside another if. In this, one if block is created inside another if block when the outer block is true then only the inner block is executed.

**Syntax:**

if(condition)

{

//statement

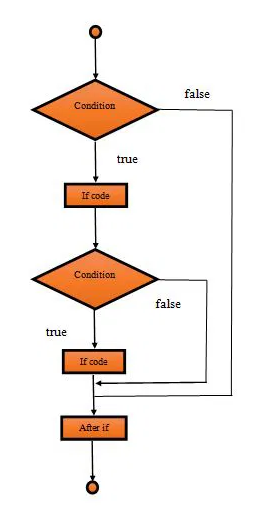
if(condition)

{

//statement

}

}



**Switch statement**

In Java, the switch statement is used for executing one statement from multiple conditions. it is similar to an if-else-if ladder.

Switch statement consists of conditional based cases and a default case.

In a switch statement, the expression can be of **byte, short, char and int** type.

From **JDK-7, enum, String** can also be used in switch cases.

Following are some of the rules while using the switch statement:

1. There can be one or N numbers of cases.
2. The values in the case must be unique.
3. Each statement of the case can have a break statement. It is optional.

**Syntax:**

Following is the syntax to declare the switch case in Java.

switch(expression)

{

case value1:

//code for execution;

break; //optional

case value2:

// code for execution

break; //optional

......

......

......

......

Case value n:

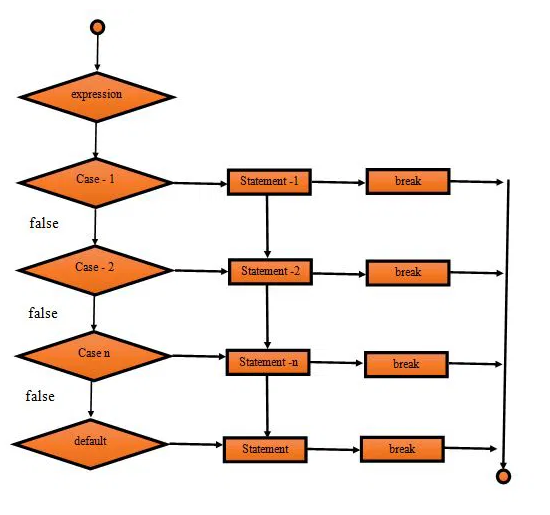
// code for execution

break; //optional

default:

code for execution when none of the case is true;

}

  
🔄 **Multiple Case Labels in Java switch Statements**In Java, starting from Java 14, you can use multiple case labels in a more concise and readable way using the "enhanced switch" (also known as switch expressions). **✅ Old Way (Traditional switch)**int day = 6;

switch (day) {

case 6:

case 7:

System.out.println("Weekend");

break;

case 1:

case 2:

case 3:

case 4:

case 5:

System.out.println("Weekday");

break;

}  
  
**✅ New Way (Java 14+: Multiple Case Labels with ->)**With enhanced switch, you can list multiple case labels in one line, separated by commas:  
int day = 6;

switch (day) {

case 1, 2, 3, 4, 5 -> System.out.println("Weekday");

case 6, 7 -> System.out.println("Weekend");

default -> System.out.println("Invalid day");

}  
🟡 This is more readable and avoids unnecessary break statements.  
  
**💡 Summary**

| **Feature** | **Traditional switch** | **Enhanced switch (Java 14+)** |
| --- | --- | --- |
| Multiple case labels | Yes, but stacked | Yes, comma-separated |
| Arrow (->) syntax | ❌ | ✅ |
| Fallthrough by default | ✅ | ❌ (not allowed) |
| Needs break | ✅ | ❌ |
| Can return a value | ❌ | ✅ |

**✅ Guess the Output – Java Conditional Statements**

**1. Simple if Statement**

int x = 3;

if (x > 5)

System.out.println("High");

System.out.println("Done");

**2] if-else-if Ladder**

int score = 85;

if (score >= 90)

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

else if (score >= 80)

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

else

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

**3] Nested if Statements**

int num = 15;

if (num > 10) {

if (num % 5 == 0)

System.out.println("Divisible by 5");

else

System.out.println("Not divisible by 5");

} else {

System.out.println("Less than or equal to 10");

}

**4] No Braces Confusion**

int a = 5;

if (a > 0)

System.out.println("Positive");

System.out.println("Check Complete");

**5] Equality Check**

int a = 10;

int b = 20;

if (a == b)

System.out.println("Equal");

else

System.out.println("Not Equal");

**6] Multiple Conditions (&&)**

int age = 25;

int income = 40000;

if (age > 18 && income > 30000)

System.out.println("Eligible");

else

System.out.println("Not Eligible");

**7] String Comparison Mistake**

String a = "hello";

if (a == "hello")

System.out.println("Matched");

else

System.out.println("Not Matched");

**8] Multiple Conditions (||)**

boolean hasLicense = false;

boolean hasLearnerPermit = true;

if (hasLicense || hasLearnerPermit)

System.out.println("Can Drive");

else

System.out.println("Cannot Drive");

**9] Basic Case Matching**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 day");

}  
  
**10] Fallthrough Without break**  
int number = 2;

switch (number) {

case 1:

System.out.println("One");

case 2:

System.out.println("Two");

case 3:

System.out.println("Three");

default:

System.out.println("Default");

}

**11] default in the Middle**

int x = 4;

switch (x) {

case 1:

System.out.println("A");

break;

default:

System.out.println("Default");

case 2:

System.out.println("B");

}

**12] String in switch (Java 7+)**  
String fruit = "Apple";

switch (fruit) {

case "Banana":

System.out.println("Banana");

break;

case "Apple":

System.out.println("Apple");

break;

default:

System.out.println("Unknown Fruit");

}

**13] Char in switch**  
char grade = 'B';

switch (grade) {

case 'A':

System.out.println("Excellent");

break;

case 'B':

System.out.println("Good");

break;

case 'C':

System.out.println("Average");

break;

default:

System.out.println("Poor");

}

**14] Case Sensitivity**

char option = 'y';

switch (option) {

case 'Y':

System.out.println("Yes");

break;

case 'N':

System.out.println("No");

break;

default:

System.out.println("Invalid");

}

**15] Multiple Case Labels (Java 14+)**

int status = 404;

switch (status) {

case 200, 201 -> System.out.println("Success");

case 400, 404 -> System.out.println("Client Error");

default -> System.out.println("Other");

}

## **Introduction to loops in java**

Loop is an important concept of a programming that allows to iterate over the sequence of statements.

Loop is designed to execute particular code block till the specified condition is true or all the elements of a collection(array, list etc) are completely traversed.

The most common use of loop is to perform repetitive tasks.

For example if we want to print table of a number then we need to write print statement 10 times. However, we can do the same with a single print statement by using loop.

Loop is designed to execute its block till the specified condition is true.

Java provides mainly three loop based on the loop structure.

1. for loop
2. while loop
3. do while loop

**For Loop**

The for loop is used for executing a part of the program **repeatedly**. When the number of execution is fixed then it is suggested to use for loop. For loop can be categories into two type.

1. for loop
2. for-each loop

**Syntax:**

for(initialization;condition;increment/decrement)

{

//statement

}

**For loop Parameters:**

To create a for loop, we need to set the following parameters.

**1) Initialization**

It is the initial part, where we set initial value for the loop. It is executed only once at the starting of loop. It is optional, if we don’t want to set initial value.

**2) Condition**

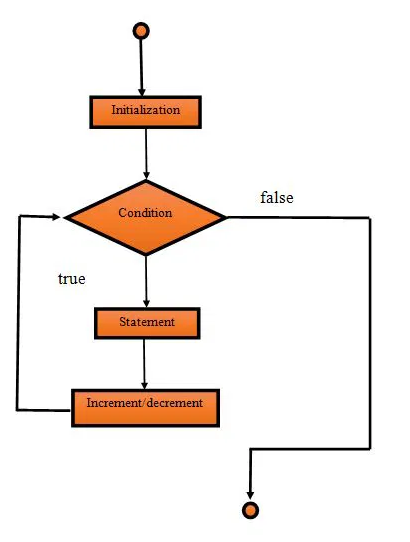
It is used to test a condition each time while executing. The execution continues until the condition is false. It is optional and if we don’t specify, loop will be inifinite.

**3) Statement**

It is loop body and executed every time until the condition is false.

**4) Increment/Decrement**

It is used for set increment or decrement value for the loop.



**for-each Loop**

In Java, for each loop is used for traversing array or collection elements. In this loop, there is no need for increment or decrement operator.

**Syntax:**

for(Type var:array)

{

//code for execution

}   
  
**1. Traditional (C‑Style) for Loop**

for ( <init> ; <condition> ; <update> ) {

// body

}

**1.1 Components**

1. **Initialization (<init>)**
   * Executed once before the loop starts.
   * You can declare one or more loop variables here.
   * Example: int i = 0, j = 10;
2. **Condition (<condition>)**
   * Evaluated before each iteration.
   * If it returns true, the loop body executes; if false, the loop exits.
   * Must be a boolean expression (or convertible to boolean).
3. **Update (<update>)**
   * Executed after each iteration of the loop body.
   * Typically used to increment/decrement loop variables.
   * Can be multiple comma‑separated statements: i++, j--.
4. **Body**
   * One or more statements enclosed in {}.
   * If you omit {}, only the next single statement is treated as the body.

**1.2 Rules & Tips**

* **Scopes:** Variables declared in <init> are only visible inside the loop.
* **Multiple variables:** Allowed if they share the same type:

for (int i = 0, j = 10; i < j; i++, j--) { … }

* **Empty clauses:** You can omit any part—just keep the semicolons.
  + Infinite loop: for (;;) { … }
  + No update: for (int i = 0; i < 10; ) { … i++; }
* **Break & Continue:**
  + break; exits the entire loop.
  + continue; skips to the update step, then rechecks the condition.
* **Nested loops:** Inner loops can shadow outer loop variables but avoid confusion.

**1.3 Example: Summing Even Numbers**

int sum = 0;

for (int i = 1; i <= 100; i++) {

if (i % 2 != 0) continue; // skip odd

sum += i;

}

System.out.println("Sum of evens: " + sum);  
  
**2. Enhanced (For‑Each) for Loop**

for ( ElementType element : collectionOrArray ) {

// body using element

}

**2.1 When to Use**

* **Iterating arrays**

int[] nums = {1,2,3};

for (int n : nums) { … }

* **Iterating any Iterable<T>** (e.g., List, Set)

List<String> names = List.of("Alice","Bob");

for (String name : names) { … }

**2.2 Rules & Tips**

* **Read‑only element:** The loop variable element is a copy of each item—you cannot reassign into the underlying array/collection.
* **Cannot modify structure:** Don’t add or remove elements from the collection inside the loop (will throw ConcurrentModificationException).
* **Simpler syntax:** No index or iterator code—you focus on the logic.
* **Supports multi‑dimensional arrays:**

int[][] matrix = { {1,2}, {3,4} };

for (int[] row : matrix)

for (int cell : row)

System.out.println(cell);

**2.3 Example: Printing Non‑Empty Strings**

String[] words = {"apple", "", "banana", " ", "cherry"};

for (String w : words) {

if (w == null || w.trim().isEmpty()) continue;

System.out.println(w);

}

**While Loop**

Like for loop, while loop is also used to execute code repeatedly. a control statement. It is used for iterating a part of the program several times. When the number of iteration is not fixed then while loop is used.

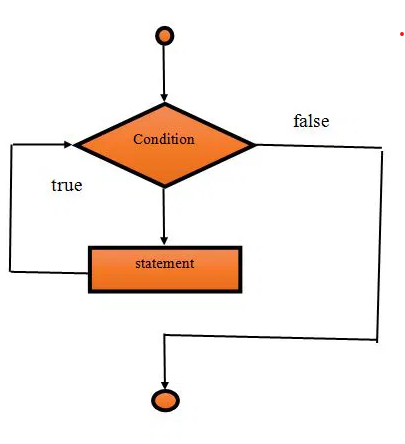
**Syntax:**

while(condition)

{

//code for execution

}



**Example for infinite while loop**

A while loop which conditional expression **always returns true** is called infinite while loop. We can also create infinite loop by passing **true** literal in the loop.

Be careful, when creating infinite loop because it can issue memory overflow problem.

public class WhileDemo2

{

public static void main(String[] args)

{

while(true)

{

System.out.println("infinitive while loop");

}

}

}

**do-while loop**

In Java, the do-while loop is used to execute statements again and again. This loop **executes at least once** because the loop is executed before the condition is checked. It means loop **condition evaluates after executing** of loop body.

The main **difference between while and do-while loop** is, in do while loop condition evaluates after executing the loop.

**Syntax:**

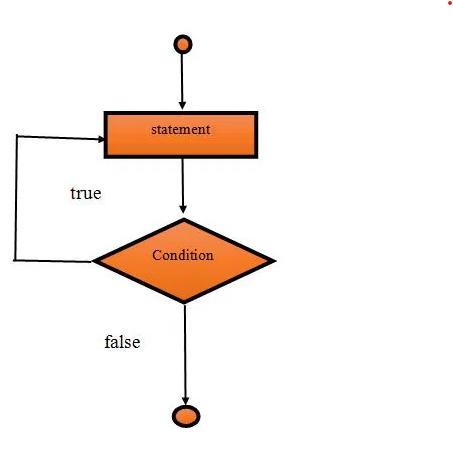
do

{

//code for execution

}

while(condition);



## **Assignments for this week:**

**1] Electricity Bill Calculator**

**📝 Assignment:**

Input units consumed.

Calculate bill using slab:

* 0–100 → ₹2/unit
* 101–300 → ₹3/unit
* 301–500 → ₹5/unit
* 501+ → ₹7/unit

Use switch for:

* Customer Type: Residential / Commercial
* Add 10% tax for commercial customers

✅ Sample Input:

Enter customer type (1 for Residential, 2 for Commercial): 1

Enter units consumed: 350

✅ Sample Output:

Base Bill: ₹1050.0

Final Bill (Residential): ₹1050.0

**2. Gym Workout Tracker**

**Loop Used:** for loop  
**Concepts:** Iteration, conditional logic

**📝 Assignment:**

Track the calories burned for 7 days of workouts.

* User enters daily workout time (in minutes).
* For each day, calculate calories burned (assume 5 calories per minute).
* At the end, show total and average calories burned.

✅ Sample Input:

Enter workout time for 7 days (in minutes):

30

45

60

0

20

50

40

✅ Sample Output:

Total calories burned: 1225

Average per day: 175.0

**3. Bank Loan EMI Tracker**

**Loop Used:** do-while  
**Concepts:** Loop control, arithmetic, condition checking

**📝 Assignment:**

* Ask user for total loan amount and EMI per month.
* Simulate monthly payment until loan is repaid.
* Show month-wise balance and total months required.

✅ Sample Input:

Enter loan amount: 10000

Enter EMI per month: 2000

✅ Sample Output:

Month 1: Remaining balance = 8000

Month 2: Remaining balance = 6000

Month 3: Remaining balance = 4000

Month 4: Remaining balance = 2000

Month 5: Remaining balance = 0

Total months to repay loan: 5

**4. Fuel Consumption Log**

**Loop Used:** for loop  
**Concepts:** Input collection, average calculation

**📝 Assignment:**

Log fuel used each day for a vehicle over n days.

* Ask for number of days.
* Input fuel consumed each day.
* Output total and average fuel usage.

✅ Sample Input:

Enter number of days: 4

Enter fuel consumed each day:

5.0

6.5

4.2

7.3

✅ Sample Output:

Total fuel consumed: 23.0 liters

Average per day: 5.75 liters

**5. Print right-angled triangle pattern  
Assignment:** Print a right-angled triangle pattern using asterisks (\*).  
  
✅ Sample Input:

**How many rows to print: 5**

✅ Sample Output:

**\***

**\*\***

**\*\*\***

**\*\*\*\***

**\*\*\*\*\***

**6. Factorial of a number  
Assignment:** Calculate the factorial of a given positive integer n.  
**Explanation:** The factorial of a number n is the product of all positive integers from 1 to n.

✅ Sample Input:

**Input number to calculate factorial: 5**

✅ Sample Output:  
**5!** = **120  
  
Calculation:**

5!=5×4×3×2×15!=5×4×3×2×15!=1205!=120

**Explanation:**

* Factorial of a non-negative integer **n** is the product of all positive integers from **1** to **n**.
* By definition, **0! = 1**.