

**1.What are the Conditional Operators in Java?**

=> There are three types of conditional operators in Java. These include:

* Conditional **AND**
* Conditional **OR**
* **Ternary Operator**

They are used when a condition comprises more than one boolean expression. For instance, if we want to print a number only if it is greater than 2 and less than 5, then we will use conditional operators to combine the 2 expressions.

**Logical-and operator (&&)**

It is used when we want the condition to be true if both the expressions are true.

**Syntax**

if(condition - 1 && condition - 2) {

statement;

}

**Example**   
Print the number if the input value is greater than 5 and less than 10.

**Code**

if (val > 5 && val < 10) {

System.out.print(val);

}

**Case -** 1: val = 3   
Output- No output   
Explanation- The input value is less than 10 but it is not greater than 5.

**Case - 2:** val = 7   
Output- 7   
Explanation- The input value is both less than 10 and greater than 5.

**Case - 3:** val = 13   
Output- No output   
Explanation- The input value is greater than 5 but it is not less than 10.

**Logical-or operator (||)**

This operator is used when we are satisfied as long as any one of the boolean expressions is evaluated as true.

**Syntax**

if(condition - 1 || condition - 2) {

statement;

}

**Example**   
Print the number if the input value is greater than 10 or less than 5.

**Code**

if (val < 5 || val > 10) {

System.out.print(val);

}

**Case - 1:** val = 3   
Output- 3   
Explanation- The input value is less than 5. It is enough to satisfy the condition so the second condition won't be tested and the val will be printed.

**Case - 2:** val = 7   
Output- No output   
Explanation- Both the conditions are evaluated as false.

**Ternary operator (?:)**

It is a smaller version for the if-else statement. If the condition is true then the statement - 1 is executed else the statement - 2 is executed.

**Syntax**   
condition ? statement - 1 : statement - 2; Example

val % 2 == 1 ? System.out.println(“Value entered is odd) : System.out.println(“Value entered

is even);

**Case - 1:** val = 1   
Output (without ternary operator) - Value entered is odd Output (without ternary operator) - Value entered is odd

**Case - 2:** val = 2   
Output (without ternary operator) - Value entered is even Output (without ternary operator) - Value entered is even

**2.What are the types of operators based on the number of operands?**

=>There are two types of operators based on the number of operands:

**Unary operators:** An operator that operates on only one operand. Examples: ++, --, !, etc.

**Binary operators:** An operator that operates on two operands. Examples: +, -, \*, /, etc.

**3. What is the use of Switch case in Java programming?**

=>Let’s say we have a variable. Now, we want to do multiple operations on it based upon what value it is storing. In such cases the switch statement use.

It is like an if-else ladder with multiple conditions, where we check for equality of a variable with various values.

It works with byte, short, int, long, enum types, String and some wrapper types like Byte, Short, Int, and Long. Since Java 7, we use strings in the switch statement.

**Syntax**

switch (expression) {

case x:

// code

break;

case y:

// code

break; .

.   
. default:

// code

}   
**Note:** The case value must be literal or constant, and must be unique.

**Example**   
Write a program using switch statements to check if the input lowercase character is vowel or consonant.

**Code**   
switch (ch) { case ‘a’:

System.out.println(“Vowel”);

break;

case ‘e’:

System.out.println(“Vowel”);

break;

case ‘i’:

System.out.println(“Vowel”);

break;

case ‘o’:

System.out.println(“Vowel”);

break;

case ‘u’:

System.out.println(“Vowel”);

break;

default:

System.out.println(“Consonant”);

}

Case-1: ch = ‘e’ Output- Vowel

Case-2: ch = ‘w’ Output - Consonant

**4.What are the priority levels of arithmetic operation in Java?**

=>In Java, arithmetic operations follow a specific order of precedence. The following is the list of precedence levels for arithmetic operations in Java, from highest to lowest:

1. **Parentheses ()**
2. **Unary operators (e.g., unary plus (+), unary minus (-))**
3. **Multiplication (\*), division (/), and modulus (%), left to right**
4. **Addition (+) and subtraction (-), left to right**

When an expression contains multiple operations, operations with higher precedence are performed before operations with lower precedence. Parentheses can be used to control the order of evaluation and to force an expression to be evaluated in a specific order.

**5.What are the conditional Statements and use of conditional statements in Java?**

=>**Conditional statements** are a fundamental concept in programming that allow you to execute a certain block of code based on certain conditions. In Java, there are three main types of conditional statements:

1.if statement

2.if-else statement

3.switch statement

Conditional statements are used in Java to control the flow of execution based on certain conditions. They are used to make decisions, repeat actions, and to implement algorithms. They are essential for writing programs that can adapt and make decisions based on input and changing conditions.

**6. What is the syntax of if else statement?**

=>In Java, the syntax of an if-else statement is as follows:

if (condition - 1) {

statement - 1

} else if (condition - 2) {

statement - 2

} else {

statement - 3

}

**The condition-1** is a boolean expression that evaluates to either true or false. If the condition is true, **the statement-1** is executed.

If the **condition-1** is false, then **condition-2** boolean expression that evaluates to either true or false . If the **condition-2** is true, the **statement-2** is executed.

If the **conditrion-2** is also false then the **statement-3** is executed.

**7. What are the 3 types of iterative statements in java?**

=> In Java, we have 3 types of iterative statements -

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

**The while loop**

A while loop is a loop that runs through its body, known as a while statement, as long as a predetermined condition is evaluated as true.

Syntax

while (condition)

statement;

Example -

Print the first 10 natural numbers.

Code   
int i = 1;   
while (i <= 10) {

System.out.print(i + “ “);

i = i + 1; }

Output- 1 2 3 4 5 6 7 8 9 10

**The for loop**

Unlike while loop, in for loop we have 3 parts in the for header.

Syntax

for (init-statement; condition; final-expression) {

statement

}

Example -  
Print the first 10 natural numbers.

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

System.out.println(i + “ “);

}

Output- 1 2 3 4 5 6 7 8 9 10

**The do-while loop**

Unlike while and for loop, do-while loop tests for the condition at the end of each execution for the next iteration. In other words, the loop is executed at least once before the condition is checked. Other than that everything is the same as in the while loop.

Syntax

do {

statement;

} while (condition); Example -   
Code

int idx = 15;

do {

System.out.print(idx + “ ”);

} while (idx < 5);

**Output**- 15

**8.Write the difference between for loop and do-while loop?**

=> **For loop** and **do-while** loop are two types of loop structures in programming languages. The main difference between the two is in how they control the execution of the loop:

**For loop**: The for loop is used when you know the number of times you want to execute the loop body. The for loop consists of three parts: an initialization expression, a termination condition, and an iteration expression. The loop body is executed only if the termination condition is true.

for (initialization; termination; iteration) {

// code to be executed

}

**Do-while loop:** The do-while loop is used when you want to execute the loop body at least once, and then repeat the loop as long as the condition is true. The loop body is executed first, and then the condition is checked. If the condition is true, the loop body is executed again.

do {

// code to be executed

} while (condition);

**In conclusion**, the for loop is best suited for a situation where you know the number of times you want to execute the loop body, while the do-while loop is best suited for a situation where you want to execute the loop body at least once, and then repeat the loop based on a condition.

**9. Write a program to print numbers from 1 to 10.**

=> Here's a program in Java that prints numbers from 1 to 10:

public class Main {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

**Output:**

12345678910

Here's an explanation of the code:

**for (int i = 1; i <= 10; i++):** This is the for loop. It starts with an initialization expression int i = 1 to initialize the loop variable i to 1. The termination condition i <= 10 specifies that the loop should continue to execute as long as i is less than or equal to 10. The iteration expression i++ increments i by 1 after each iteration of the loop.

**System.out.print(i):** This line of code is inside the loop body. It prints the value of i to the console on each iteration of the loop.

When the program is run, it will print the numbers 1 to 10.