**Q-1: what are the conditional operator in java ?**

**Ans:** conditional operators play a crucial role in creating logical expressions that yield boolean results. They allow developers to control the flow of execution within a Java program based on specific conditions. Let’s delve into the details:

1. **Conditional AND (**&&**)**:
   * The conditional AND operator is denoted by &&.
   * It operates between two Boolean expressions.
   * The result is true only if **both** expressions are true; otherwise, it returns false.

**Conditional OR (||):**

* The conditional OR operator is denoted by ||.
* It also works with two Boolean expressions.
* The result is true if **either** of the expressions is true; otherwise, it returns false.

**Ternary Operator (**?:**)**:

* The ternary operator is unique because it accepts **three operands**.
* It evaluates Boolean expressions and assigns a value to a variable based on the condition.
* Syntax: variable = (condition) ? expression1 : expression2
* If the condition is true, expression1 is executed; otherwise, expression2 is executed.

**Q-2: what are the types of operators based on the number of operands ?**

Soln: Unary Operators:

* + These operators work on **one operand**.
  + Examples include:
    - **Increment (**++**)**: Increases the value of a variable by 1.
    - **Decrement (**--**)**: Decreases the value of a variable by 1.
    - **Unary Plus (**+**)**: Represents a positive value.
    - **Unary Minus (**-**)**: Represents a negative value.
    - **Logical NOT (**!**)**: Negates a boolean value.

1. Binary Operators:
   * Binary operators operate on **two operands**.
   * Common binary operators include:
     + **Arithmetic Operators** (e.g., +, -, \*, /, %): Perform mathematical calculations.
     + **Comparison Operators** (e.g., ==, !=, <, >, <=, >=): Compare values.
     + **Logical Operators** (e.g., &&, ||): Combine boolean expressions.
     + **Bitwise Operators** (e.g., &, |, ^, ~, <<, >>): Manipulate individual bits.
2. Ternary Operator (?:):
   * The ternary operator is unique because it uses **three operands**.
   * Syntax: variable = (condition) ? expression1 : expression2
   * If the condition is true, expression1 is assigned; otherwise, expression2 is assigned.

**Q-3: what is the use of switch case in java programming ?**

Soln:

switch-case statement is used as a control flow mechanism to make decisions based on the value of a variable. It is particularly useful when you have a single variable or expression whose value can lead to different execution paths in your code

Code :

public class Main {

public static void main(String[] args) {

int number = 44;

String size;

switch (number) {

case 29:

size = "Small";

break;

case 42:

size = "Medium";

break;

case 44:

size = "Large"; // Matches the value 44

break;

case 48:

size = "Extra Large";

break;

default:

size = "Unknown";

break;

}

System.out.println("Size: " + size); // Output: Size: Large

}

}

**Q-4: what are the priority levels of arithmetic operation in java ?**

Soln:

arithmetic operations follow the standard precedence rules, where certain operators have higher precedence than others. The priority levels of arithmetic operations in Java, listed from highest to lowest precedence, are as follows:

* **Parentheses**: Expressions within parentheses are evaluated first.
* Postfix operators: For example, ++ and --, which increment or decrement values after the current expression is evaluated.
* **Unary operators**: Unary operators such as +, -, ++, --, !, and ~.
* Multiplicative operators: These include \* (multiplication), / (division), and % (modulo).
* **Additive operators:** These include + (addition) and - (subtraction).
* Shift operators: These include << (left shift), >> (right shift with sign extension), and >>> (right shift with zero extension).
* **Relational operators:** These include <, <=, >, and >=.
* Equality operators: These include == (equal to) and != (not equal to).

**Q-5: what are the conditional statement and use of conditional statement in java ?**

**Soln: c**onditional statements are used to execute different blocks of code based on certain conditions. The main conditional statements in Java are if, else if, and else. These statements allow you to control the flow of your program based on whether certain conditions are true or false.

**if statement:** It is used to execute a block of code only if the specified condition is true.

if (condition) {

// block of code to be executed if the condition is true

}

**if-else statemen**t: It is used to execute one block of code if the specified condition is true, and another block of code if the condition is false.

if (condition) {

// block of code to be executed if the condition is true

} else {

// block of code to be executed if the condition is false

}

**else if statement:** It is used to specify multiple conditions. If the first condition is false, it checks the next condition and executes the block of code associated with the first true condition.

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if condition2 is true

} else {

// block of code to be executed if neither condition1 nor condition2 is true

}

**Q-6: what is the syntax of if else statement ?**

Soln:

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if condition2 is true

} else {

// block of code to be executed if neither condition1 nor condition2 is true

}

**Q-7: What are the 3 types of iteratives statements in java ?**

Soln:

there are three main types of iterative statements (also known as loops) used for executing a block of code repeatedly:

* **for loop:** The for loop repeatedly executes a block of code until a specified condition evaluates to false. It consists of three parts: initialization, condition, and iteration expression.
* for (initialization; condition; iteration) {
* // block of code to be executed
* }
* **while loop:** The while loop repeats a block of code as long as a specified condition is true.
* while (condition) {
* // block of code to be executed
* }
* **do-while loop**: The do-while loop is similar to the while loop, but it ensures that the block of code is executed at least once before checking the condition.
* do {
* // block of code to be executed
* } while (condition);

**Q-8: write the difference between for loop and do-while loop**

Soln: The main difference between a for loop and a do-while loop in Java lies in when the loop condition is evaluated relative to the execution of the loop body:

* **For Loop:**
  + In a for loop, the loop condition is evaluated before entering the loop body. If the condition is false initially, the loop body will not execute at all.
  + The for loop is typically used when the number of iterations is known before entering the loop, and it often involves iterating over a range of values.

Example:

java

Copy code

for (int i = 0; i < 5; i++) {

// Loop body

}

* **Do-While Loop:**
  + In a do-while loop, the loop condition is evaluated after executing the loop body. This guarantees that the loop body will execute at least once, regardless of the condition.
  + The do-while loop is useful when you need to execute the loop body at least once and then check the condition for further iterations.

Example:

java

Copy code

int i = 0;

do {

// Loop body

i++;

} while (i < 5);

**Q-9: write a program to print number from 1 to 10.**

Soln: public class PrintNumbers {

public static void main(String[] args) {

System.out.println("Numbers from 1 to 10:");

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

System.out.print(i + " ");

}

}

}