1. What are the Conditional Operators in Java?
Ans: Conditional Operators are used when a condition comprises more than one boolean expression. For example, if we want to print a number that is greater than 2 and less than 5, we use conditional operators to combine the two expressions. There are 3 types of conditional operators in Java, i.e, Logical-AND, Logical-OR and Ternary Operator.

Logical-AND Operator: (&&)

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

Syntax:

```
if( expression1 && expression2){
        statement;
}
```

Example:

```
if( res>10 && res<25){
    System.out.println(res);
}
```

Logical-OR Operator:(||)

It is used when we are satisfied as long as **any one** of the boolean expression is evaluated as true.

Syntax:

```
if( expression1 || expression2){
        statement;
}
```

Example:

```
if( res<10 || res>25){
    System.out.println(res);
}
```

Ternary Operator: (?:)

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

Syntax:

```
condition ?  statement1 : statement2;
```

Example:

```
int result;
result = (m1>m2) ? m1 : m2;
```

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

Ans: There are **two types** of operators based on the number of operands. They are as follows:
i. Unary: Unary operators perform an action with a **single operand**. For example, if we want to create negative numbers at the time of initialization itself, then we can use the negative (-) unary operator and the script language recognizes this operator. It reverses the sign of the expression from positive to negative or vice-versa.
Example: `a = -10;`
Similarly, the Prefix operators increment or decrement the variable prior to dereferencing the object, while the Postfix operators increment or decrement the variable after referencing it.
Example:

```
a=1;
b = a++; // b will equal 1, a will equal 2;
a = 1;
b = ++a; // b will equal 2, a will equal 2;
a= 1;
b = a--; // b will equal 1, a will equal 0;
```

Ii. Binary: Binary operators perform actions with **two operands**. In a complex expression, (two or more operands) the order of evaluation depends on precedence rules. A space needs to be inserted before and after a binary operator. The binary arithmetic operators that are supported are listed below:

Symbol	Operation	Example	Description
+	Addition	a + b	Add the two operands
-	Subtraction	a - b	Subtract the second operand from the first operand
*	Multiplication	a * b	Multiply the two operands
/	Division	a / b	Divide the first operand by the second operand
^	Power	a ^ b	Raise the first operand by the power of the second operand

%	Modulo	a % b	Divide the first operand by the second operand and yield the remainder portion

3. What is the use of Switch case in Java programming?
Ans: The **switch statement** is a multi-way branch statement. In simple words, the Java switch statement executes one statement from multiple conditions(or cases). It is like an if-else-if ladder statement. It provides an easy way to dispatch execution to different parts of code based on the value of the expression. Basically, the expression can be a byte, short, char, or int primitive data types. It basically tests the equality of variables against multiple values(cases).
Syntax:
Here, the expression may take many values, based on each value(separated here by a **case**), a particular piece of code is executed and then a **break** keyword is used to terminate the execution of that case.

```
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.

4. What are the priority levels of arithmetic operation in Java?
Ans: **Operator Priority Level Table in Java**
The table below lists the priority level of operators in Java; the higher it appears in the table, the higher its priority.

Operators	Priority
postfix increment and decrement	`++ --`
prefix increment and decrement, and unary	`++ -- + - ~ !`
multiplicative	`* / %`
additive	`+ -`
shift	`<< >> >>>`
relational	`< > <= >= instanceof`
equality	`== !=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`

logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

5. What are the conditional Statements and use of conditional statements in Java?
Ans: Conditional statements are branches of code that can be executed depending on another condition. I.e, whether that condition is true or false. In Java, these clauses are called decision or selection statements.
There are **five** types of Java conditional statements:-
i. If statement
ii. If-Else statement
iii. If-Else-If Ladder statement
iv. Nested If statement
v. Switch statement
The use of conditional statements is as follows:
To branch the execution of a program based on user input, we need to use something known as a conditional statement. So, we can execute one branch of code if the condition is true and another if it is false. It gives the user an option to execute the desired branch of code if there are multiple conditions and only of them is satisfied depending on the input.
Conditional statements are mostly used in decision-making scenarios which means these statements take a decision on the basis of some conditions. The conditional statements are also referred to as branching statements because the program takes a decision based on the result of the assessed condition. A very simple example of conditional statements from our daily life can be if today is Friday, then tomorrow will be Saturday.

6. What is the syntax of if else statement?
Ans: An if-else statement is designed to give us this functionality in our code. It executes statements based on if some condition is true or false. If the condition is true, the if statement is executed, otherwise the else statement is executed.
The **syntax** of an if-else statement is:

```
if (condition) {
statement - 1
}
else {
statement - 2
}
```

7. What are the 3 types of iterative statements in Java?
Ans: In Java, the three types of iterative statements are:
a. 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;
}
```

b. the for loop - Unlike the while loop, in for loop we have 3 parts in the for header.
Syntax:

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

c. 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.

Syntax:

```
do {
statement;
} while(condition);
```

8. Write the difference between for loop and do-while loop?
Ans: The **syntax** of the **for** loop is:

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

Init-statement: In general, this statement is used to initialize or assign a starting value to a variable that may be altered over the course of the loop. It is executed only once at the start.
Condition: Similar to the while condition, it serves as a loop control. The loop block is executed until the condition evaluates to true.
Final-expression: It is evaluated after each iteration of the loop. It is generally to update the values of the loop variables.

FLOW OF FOR LOOP:
i. Init-statement is executed, i.e, a variable is defined and initialized.
ii. Next, the condition is evaluated or checked, if it is true then only the statement/body of the loop is executed, otherwise, the loop terminates. If the condition is false on the first iteration, then the **for** body is not executed at all.
iii. If the condition is true, the **for** body executes.
iv. In the last step, the final-expression is evaluated.

The syntax of the **do-while** loop is:

```
do {
statement;
} while(condition);
```

Unlike for loop, the 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.
For example,

```
int idx = 15;
do {
System.out.print(idx + " ");
} while (idx < 5);
```

idx is assigned a value of 15. Once the flow enters the do loop, it prints the value of idx without checking any condition. Though it is false here the statement will execute once. Next, the flow enters the while condition, here the value of idx is checked, the condition is found to be false and the do-while loop stops execution.

9. Write a program to print numbers from 1 to 10.
Ans:

```
public class Solution {
        public static void main(String []args) {
           for(int i=1;i<=10; ++i) {
              System.out.println(i);
           }
        }
```

```
}
```