## 1) What are the conditional operators in Java?

Ans:
There are three main conditional operators in Java:

1. The ternary operator (? :): This operator is also known as the conditional operator. It takes three operands and evaluates a boolean expression. The syntax is as follows:

```
Condition ? expression1 : expression2
```

If the condition is true, the operator returns the value of expression1; otherwise, it returns the value of expression2.

2. The logical AND operator (&&): This operator evaluates two boolean expressions and returns true if both expressions are true. If either or both expressions are false, it returns false. The syntax is as follows:

```
expression1 && expression2
```

3. The logical OR operator (||): This operator evaluates two boolean expressions and returns true if at least one of the expressions is true. If both expressions are false, it returns false. The syntax is as follows:

```
expression1 || expression2
```

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

Ans:
Based on the number of operands:

1. Unary operators: Unary operators work with a single operand. They perform operations on a single value or variable. Examples of unary operators include:

- Unary minus (-): Negates the value of an operand.
- Unary plus (+): Represents the positive value of an operand.
- Increment (++): Increases the value of an operand by 1.
- Decrement (--): Decreases the value of an operand by 1.
- Logical complement (!): Flips the logical state of an operand.

2. Binary operators: Binary operators work with two operands. They perform operations between two values or variables. Examples of binary operators include:

- Arithmetic operators: Addition (+), subtraction (-), multiplication (*), division (/), and modulus (%).

- Assignment operators: Assign a value to a variable, such as the equals sign (=), plus equals (+=), minus equals (-=), etc.

- Relational operators: Compare two values and return a boolean result, such as greater than (>), less than (<), equal to (==), not equal to (!=), etc.

- Logical operators: Combine boolean expressions and return a boolean result, such as logical AND (&&), logical OR (||), etc.

- Bitwise operators: Perform operations on individual bits of binary numbers, such as bitwise AND (&), bitwise OR (|), bitwise XOR (^), etc.

3. Ternary operator: The ternary operator is a special operator that takes three operands. It evaluates a boolean expression and returns one of two values based on the result of the evaluation. The syntax of the ternary operator is as follows:
condition ? expression1 : expression2

If the condition is true, the operator returns the value of expression1; otherwise, it returns the value of expression2.

## 3) What is the use of the Switch case in Java programming?

Ans:

The switch case statement in Java is used to control the flow of execution based on the value of a variable or an expression. It provides a concise and efficient way to handle multiple possible values and execute different blocks of code depending on the match.

Here are some uses of the switch case statement in Java programming:

1. Multiple Branching: The switch case statement allows you to specify multiple cases, each representing a possible value of the variable or expression being evaluated. It provides an alternative to writing multiple if-else statements, making the code more readable and manageable, especially when there are many possible values to consider.

2. Simplifying Decision-Making: When you have a variable or expression whose value can result in different actions, the switch case statement simplifies decision-making. Each case represents a specific action to be performed, reducing the need for complex nested if-else structures.

3. Code Organization: By using a switch case, you can organize your code into distinct blocks associated with different cases. This can make the code more modular and easier to understand, as related actions are grouped together.

4. Default Case: The switch case statement allows you to include a default case. If none of the cases matches the value being evaluated, the code inside the default case will be executed. This provides a fallback option for handling unexpected or undefined values.

It's important to note that the value being evaluated in the switch case statement should be a byte, short, char, int, String, or an enum. Java does not allow the use of long, float, double, or boolean types as the switch expression.

## 4) What are the priority levels of arithmetic operations in Java ?

Ans:
The priority levels of arithmetic operators in Java:

1. Multiplication (*)
2. Division (/)
3. Modulus (%)
4. Addition (+)
5. Subtraction (-)

## 5) What are the conditional statements and use of conditional statements in Java?

Ans:
In Java, there are two main conditional statements:

1. if Statement: The if statement is used to execute a block of code if a specified condition is true. The basic syntax is as follows:

```java
if (condition) {
    // Code to be executed if the condition is true
}
```

The condition is an expression that evaluates to either true or false. If the condition is true, the code block within the if statement is executed. If the condition is false, the code block is skipped, and the program continues to the next statement after the if statement.

The if statement can be extended with optional else if and else blocks to handle multiple conditions.

```
if (condition1) {
    // Code to be executed if condition1 is true
} else if (condition2) {
    // Code to be executed if condition2 is true
} else {
    // Code to be executed if none of the conditions are true
}
```

In this case, the conditions are evaluated in order, and the code block corresponding to the first true condition is executed. If none of the conditions are true, the code inside the else block is executed (if present).

2. switch Statement: The switch statement is used to select one of many code blocks to execute based on the value of a variable or expression. It provides a concise way to handle multiple cases. The basic syntax is as follows:

```
switch (variable/expression) {
    case value1:
        // Code to be executed if variable/expression matches value1
        break;
    case value2:
        // Code to be executed if variable/expression matches value2
        break;
    default:
        // Code to be executed if variable/expression doesn't match any case
}
```

The variable or expression inside the switch statement is evaluated, and the corresponding code block is executed if a match is found. The break statement is used to exit the switch block after a match. If no match is found, the code inside the default block is executed (if present).

The uses of conditional statements in Java,

1. Decision Making: Conditional statements allow you to make decisions in your program based on specific conditions. They enable you to execute different blocks of code depending on whether a condition is true or false.

2. Flow Control: Conditional statements help control the flow of execution in your program. By using conditional statements, you can determine which code blocks should be executed and which ones should be skipped, based on the evaluation of conditions.

3. Handling User Input: Conditional statements are commonly used to handle user input. They allow you to validate user input and perform different actions or provide appropriate feedback based on the input value.

4. Error Handling: Conditional statements can be used to handle error conditions or exceptional cases in your program. They enable you to identify specific error conditions and execute error-handling code blocks to handle those situations gracefully.

5. Loop Termination: Conditional statements are often used in conjunction with loops to control the termination of the loop. By evaluating a condition, you can determine whether to continue or exit the loop.

6. Algorithmic Logic: Conditional statements are essential for implementing complex algorithmic logic. They allow you to define intricate decision-making processes and execute specific code blocks based on various conditions, enabling you to solve complex problems.

## 6) What is the syntax of the if-else statement?

Ans:
The syntax of the if-else statement in Java is as follows:

```
if (condition) {
    // Code to be executed if the condition is true
} else {
    // Code to be executed if the condition is false
}
```

The if-else statement consists of the following parts:

1. The `if` keyword is used to start the if-else statement.

2. `condition` is a boolean expression that is evaluated. If the condition evaluates to true, the code block inside the if statement is executed. If the condition evaluates to false, the code block inside the else statement is executed.

3. The code block inside the if statement is enclosed within curly braces `{}`, and it contains the code to be executed if the condition is true.

4. The `else` keyword is followed by another code block enclosed within curly braces `{}`, which contains the code to be executed if the condition is false.

## 7) What are the three types of iterative statements in Java?

Ans:
In Java, there are three types of iterative statements, also known as loops:

1. for Loop: The for loop is used to execute a block of code repeatedly for a fixed number of iterations. It consists of three parts: initialization, condition, and increment/decrement. Here's the syntax:

```
for (initialization; condition; increment/decrement) {
    // Code to be executed in each iteration
}
```

The initialization part is executed once at the beginning, setting the initial value of the loop variable. The condition is evaluated before each iteration, and if it evaluates to true, the code block inside the for loop is executed. After each iteration, the increment/decrement part is executed to update the loop variable. If the condition becomes false, the loop terminates.

2. while Loop: The while loop is used to execute a block of code repeatedly as long as a condition is true. It only consists of the condition part. Here's the syntax:

```
while (condition) {
    // Code to be executed in each iteration
}
```

The condition is evaluated before each iteration, and if it evaluates to true, the code block inside the while loop is executed. If the condition becomes false, the loop terminates. It's important to ensure that the condition eventually becomes false to avoid infinite loops.

3. do-while Loop: The do-while loop is similar to the while loop, but the condition is evaluated after each iteration. This ensures that the code block is executed at least once. Here's the syntax:

```
do {
    // Code to be executed in each iteration
} while (condition);
```

The code block inside the do-while loop is executed first, and then the condition is evaluated. If the condition evaluates to true, the loop continues to the next iteration. If the condition becomes false, the loop terminates.

## 8) What is the difference between for loop and a do-while loop?

Ans:
The differences between a for loop and a do-while loop in Java:

For Loop:
1. Consists of initialization, condition, and increment/decrement parts.
2. The initialization part is executed once at the beginning of the loop.
3. The condition is evaluated before each iteration.
4. If the condition is false at the beginning, the loop body is not executed.
5. The increment/decrement part is executed after each iteration.
6. The loop may not execute the loop body at all if the condition is initially false.
7. Used when the number of iterations is known or when iterating over a range.

Do-While Loop:
1. Consists of a loop body and a condition.
2. The loop body is executed first before the condition is checked.
3. The condition is evaluated after each iteration.
4. The loop body is guaranteed to execute at least once, regardless of the condition.
5. If the condition is false after the first iteration, the loop is terminated.
6. Used when you want to ensure that the loop body is executed at least once, typically when input validation or menu-driven programs are involved.
7. Provides a way to exit the loop based on a condition evaluated after the loop body execution.

## 9) Write a program to print numbers from 1 to 10?

Ans:
Java program to print the numbers between 1 to 10:

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

In this program, we use a for loop to iterate from 1 to 10. The loop starts with an initialization `int i = 1`, and the condition `i <= 10` specifies that the loop should continue as long as `i` is less than or equal to 10. After each iteration, the value of `i` is incremented by 1 using the `i++` statement.

Within the loop body, we use `System.out.println(i)` to print the value of `i` on a new line. This will output the numbers from 1 to 10.