List of Concepts Involved:

- Java Operators
- Conditional Statements
- If else ternary
- Switch case
- Loops intro
- For loop, while loop,do-while loop
- More on loops
- Scanner class and user Input in java

Topic 1: Conditional operators

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. We have 3 types of conditional operators - logical-and, logical-or and ternary operator.

Logical-and operator (&&)

It is used when we want the condition to be true iff 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.

Common misconception: ‘&’ v/s ‘&&’

& -> compares bitwise
&& -> logical and

The first one parses through all the conditions, despite their evaluation as true or false. However the latter traverses through the second condition only if the first condition is evaluated as true, otherwise it negates the entire condition. For instance,

```
System.out.println(false & 5/0==2);
```

It gives us a runtime error as 5 is being divided by 0, which isn't a valid operation. However if we write-

```
System.out.println(false && 5/0==2);
```

We get “false” as our output. This is so because our first condition is false, which is enough to make the entire condition false in case of logical and.

Try this
Write a program to print the value of input if it is even and divisible by 3.

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.

Common misconception: '|' v/s '||'

| -> compares bitwise
|| -> logical or

The first one parses through all the conditions, despite their evaluation as true or false. However the latter traverses through the second condition only if the first condition is evaluated as false, otherwise it validates the entire condition. For instance,

```
System.out.println(true | 5/0==2);
```

It gives us a runtime error as 5 is being divided by 0, which isn't a valid operation. However if we write-

```
System.out.println(true || 5/0==2);
```

We get "true" as our output. This is so because our first condition is true, which is enough to make the entire condition true in case of logical or.

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

Without ternary operator

```
if (val % 2 == 1) {
     System.out.println("Value entered is odd");
} else {
     System.out. println("Value entered is even");
}
```

With ternary operator

```
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

Try this

1. Write a short program that gives the following as output -
   For each multiple of 3, print "Fizz" instead of the number.
   For each multiple of 5, print "Buzz" instead of the number.
   For numbers which are multiples of both 3 and 5, print "FizzBuzz" instead of the number.
   Otherwise print the number itself.

2. Write a short program that prints each number from 1 to 100 on a new line, except if the number is a multiple of 5 or 7.

Topic 2: If statement

In real life we often encounter situations where our actions are governed by some sort of conditions. For instance, if the weather is rainy, we carry an umbrella. Here carrying an umbrella is an action which is performed only when the condition of the weather being rainy is fulfilled.

An if statement is based on the same principle. It executes a statement based upon if some condition is true.

Syntax

```
if (condition) {
      statement;
}
```

Example:-

```
if(marks > 33) {
     System.out.print("Pass");
}
```

Case - 1: marks = 85
Output - Pass
Explanation - Since the marks are greater than 80 i.e. the condition inside 'if' parentheses is true, we get "Pass" printed in our output.

Case - 2: marks = 70
Output - No output
Explanation - Since the marks are not greater than 80 i.e. the condition inside if parentheses is false, the statement in it's block is skipped i.e. we get no output.

Topic 2: If-else statement

Sometimes we encounter situations where we have 2 types of actions that we can perform. For example, if it's tuesday i'll eat veg burger otherwise i'll eat non-veg burger.
An if else statement is designed to give us this functionality in our code. It executes statements based upon if some condition is true or false. If the condition is true, the if statement is executed, otherwise the else statement is executed.

Syntax

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

Example -
To illustrate the if-else statement, we can create a grading system. We'll assume that the score ranges between 0 to 100 inclusive. A score above 33 gets a "Pass" verdict, otherwise it's "Fail".

To solve this problem, we can use an if else statement to execute different actions for failing and passing grades:

```
if (score > 33) {
     System.out.println("Pass");
} else {
     System.out. println("Fail");
}
```

Case - 1: grade = 60
Output - Pass
Explanation - Since the score is more than 33 i.e. the condition inside 'if' parentheses is true, we get "Pass" printed.

Case - 2: grade = 20
Output - Fail
Explanation - Since the score is not more than 33 i.e. the condition inside 'if' parentheses is false, we get "Fail" printed.

Try this

1. Find if the input value is odd or even. If it's odd print "Odd", otherwise print "Even".
   Note: Input value will be between 1 and 10^6.

2. Find if the input character is 'a' or not.
   Note: Input characters will be lowercase alphabets.

Topic 2: If-else if statement

It works on the same principle as if-else statements, except here we can have multiple conditions. If the condition inside the if block is true, then a code/ statement is executed, but if it is false, it moves to the else-if block and will check if it is true and will execute the statement in the else-if block. But when the condition in this block is false, it will execute the statement in the final else block.

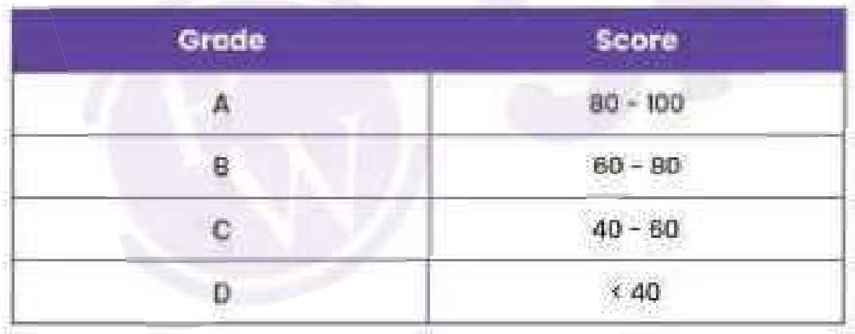
We can have multiple else-if blocks too.

Syntax

```
if (condition - 1) {
     statement - 1
} else if (condition - 2) {
     statement - 2
} else {
     statement - 3
}
```

Example

To understand this, we can further expand our grading system. Apart from pass and fail now it will give grades based on the score.



Grade	Score
A	80 - 100
B	60 - 80
C	40 - 60
D	< 40

To solve this problem, we can use an if - else if - else statement to execute different actions depending upon the score:

```
if(score > 80) {
     System.out.println(“A”);
} else if (score > 60) {
     System.out.println(“B”);
} else if (score > 40) {
     System.out.println(“C”);
} else {
     System.out.println(“D”);
}
```

Case-1: grade = 81
Output- A
Explanation - Explanation - Since the score is more than 80 i.e. the condition inside if parentheses is true, “A” gets printed.

Case-2: grade = 61
Output- B
Explanation - Since the score is less than 80, the first block is skipped and since it is more than 60 i.e. the condition inside else-if parentheses is true, “B” gets printed.

Case-3: grade = 41
Output - C
Explanation- Since the score is 41, the first 2 blocks are skipped and then the condition for the third block is checked. It turns out that it is true, so “C” gets printed and the rest is skipped.

Case 4: grade = 21
Output- D
Explanatio - Since all the conditions are falsified by the input, the else-block is run and we get “D” as output.

Try this

1. Write a program to identify people as “Child” (age < 12), “Teenager” (12 <= age < 18) or “Adult” (age >= 18).
2. Print the maximum of 3 numbers a, b, c taken as input.

Topic 2: Nested if-else

It is simply an if-else statement inside another if-else statement.

Syntax

```
if (condition - 1) {
     if (condition - 2) {
           statement - 1
     } else {
           statement - 2
     }
} else {
     statement - 3
}
```

Example:-

```
if (score > 33) {
     if(marks > 80) {
           System.out.print(“Gracefully ”);
     }
     System.out.println(“Pass”);
} else {
      System.out. println(“Fail”);
}
```

Note - Watch Your Curly Braces

Topic 3: Switch statement

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 comes into play.

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, you can 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

Try this
Write a program to print the day name based upon the day number.
1 - Monday, 2 - Tuesday, etc.

Topic 4: Introduction to Iterative statements/Loops

Assume someone comes up to you and says “I want you to give me a program that can give me all the numbers between 1 and 10000”. In such a situation writing all the numbers from 1 to 10000 isn’t a feasible solution. That’s where loops come in. They help you perform a task repeatedly, until a certain condition is met. In our example, the task would be to print the value of the number, and the condition would be till the time it is less than 10000. In Java, we have 3 types of iterative statements -

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