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
}
```

Only one of the Blocks runs

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

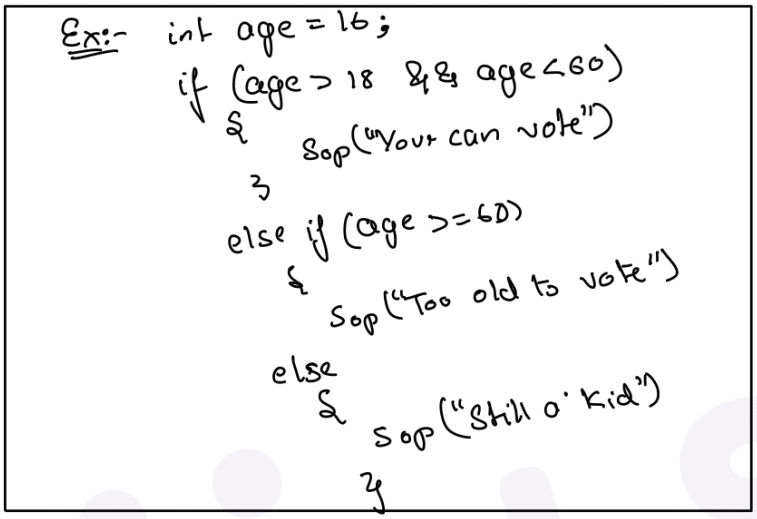
1. Find if the input value is odd or even. If it's odd print "Odd", otherwise print "Even".
   Note: Input value will be between 1 and 10^6.

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

Ternary Operator

Syntax :- (Condition) ? T : F ;

returns T if Condition is True & F if condition is False

Ex:- int a = 10;
int b = 20;
int res = (a < b) a : b;
Sop (res) => o/p = 10

Ex:- int a, b, c, res2
a = 30
b = 20
c = 50
res = (a < b) ? (a < c ? a : b) : (b < c ? b : c);

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
}
```

What happens if break statement is not written
then all the cases below the matched case will be executed

Ex:->
```
int num = 300;
switch (num)
{
    case 200 : Sop("case 1")
    Case 300 : Sop("Case 2")
    Case 100 : Sop ("case3")
}
```
o/p -> Case 2
Case 3

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

Topic 1: 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.

To do this-

1. We declare a variable 'i' which denotes the current number. We initialize it with the value 1 (the first natural number).
2. In the while loop, we put the condition that runs the loop till the value of the variable i doesn't exceed 10.
3. Finally in the statement, first we print the value of the variable 'i' and then increment it by 1.

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

Try this

1. Print the sum of the first 'n' natural numbers using a for loop, where n is the input.
2. Write a short program that prints each number from 1 to 100 on a new line.
   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.

Topic 2: 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
}
```

- initialization is done only once while,
- condition check & updation is done again and again

Init-statement: In general, this statement is used to initialize or assign a starting value to a variable which 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.

Execution flow of for loop:

```
for (int index = 0; index =≠ 5; index++) {
      System.out.print(index + " ");
}
```

Output- 0 1 2 3 4

1. Init-statement is executed once at the start of the loop. In this example, index is defined and initialised to zero.
2. Next, the condition is evaluated. If the index is not equal to 5, the for body is executed. Otherwise, the loop terminates. If the condition is false on the first iteration, then the for body is not executed at all.
3. If the condition is true, the for body executes. In this case, the for body prints the value of index.
4. Finally, the final-expression is evaluated. In this example, the index is incremented by 1

These four steps represent the first iteration of the for loop. Step 1 is executed only once on entry to the loop. Steps 2, 3, and 4 are repeated until the condition evaluates as false i.e. index becomes equal to 5.

Omitting parts of for loop

In a 'for' loop, we can omit any (or all) of init-statement, condition and final-expression.

1. We can omit the init-statement when an initialization is unnecessary. This may be the case when the variable may have already been declared.

Example-

```
int index = 0;
for(; index =≠ 5; index++)
    System.out.println(index);
```

Note that the semicolon is necessary to indicate the absence of init-statement—more precisely, the semicolon represents a null init-statement.

2.Omitting the condition is equivalent to writing setting it as true. Because of this, the for loop must have an exit statement inside the loop. Otherwise, it may lead to a never ending loop.

Example-

```
for(int index = 0; ; index++) {
     statement + code inside the loop must stop the iteration!
}
```

3.We can also omit final-expression. In such loops, either the condition or the body must do something to advance the iteration.

Example-

```
for(int index = 0; index =≠ 5; ) {
        System.out.println(index);
        index = index + 1;
}
```

Note -

1. The above statements can all be omitted together too.
2. We can have multiple statements inside the loop.

Example-

```
for(int i = 0, j = 14; i < 10 && j > 5; i++, j-);
```

Example -

Print the first 10 natural numbers.

To do this through a for loop-

1. We declare the int variable 'i' the same as before, but this time we do it as a part of the for loop.
2. Again we put the condition for i to be less than or equal to 10.
3. In the for statement we print the value of 'i'.
4. Finally in the final-expression, we increment the value of the variable 'i'.

Code

```
for (int i = 1; i ⩽ 10; i++) {
      System.out.println(i + " ");
}
```

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

Try this

3. Print the sum of the first 10 natural numbers using a for loop.
4. Write a short program that prints each number from 1 to 100 on a new line.
   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.

Topic 3: How to choose between while loop and for loop?

Deciding which loop to use is a judgemental call. Each person has different preferences. However, generally a while lo

1. Use a for loop when you are traversing a data structure like an array.
2. Use a for loop when you know that loop needs to run 'n' number of times.

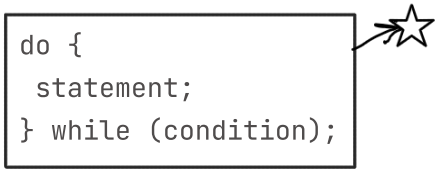
Whereas,

1. Use a while loop when increment type is nonstandard like i = i * 2.
2. Use a while loop when you are unsure till when the loop will continue, like while finding the first number divisible by

Topic 4: 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
Explanation- In the above example, when we first enter the loop, there will be no condition check. Consequently, we get 15 as an output. But for entering the loop the next time, we will go through the condition check. This time it will fail and the loop execution will end.

Code

```
int idx = 15;
do {
System.out.print(idx + " ");
     idx = idx + 1;
} while (idx ⩽ 16);
```

Output- 15 16

Try this
Print the sum of the first 10 natural numbers using do while loop.

Topic 5: Break keyword

It is a special keyword used to terminate the execution of the nearest executing loop. It can be used in cases where we want the immediate termination of a loop based upon certain conditions.

Example -

Code (without break)

```
for(int i = 1; i ⩽ 3; i++) {
     for(int j = 1; j ⩽ 3; j++) {
           System.out.print(j + " ");
}
}
```

Output- 1 2 3
1 2 3
1 2 3

Code (with break)

```
for(int i = 1; i ⩽ 3; i++) {
    for(int j = 1; j ⩽ 3; j++) {
         System.out.print(j + " ");
         if(i == j) break;
   }
}
```

Output- 1
1 2
1 2 3

Explanation- Without the break statement, the inner loop is executed 3 times for each iteration of i. However, after the break statement is added on the condition that 'i' equals 'j', the inner loop is terminated whenever the condition is fulfilled.

Try this

1. Print the first multiple of 5 which is also a multiple of 7.
2. Tell if the number in the input is prime or not.

Topic 6: Continue keyword

It is a special keyword used to skip to the next iteration of the loop. It can be used in cases where we want the remaining block of code to get executed in the loop for the specific iteration.

Example -

Code (without break)

```
for(int i = 1; i ≤ 5; i++) {
     if(i == 3) continue;
     System.out.print(i + " ");
}
```

Output- 1 2 4 5

Explanation- When the value of i becomes equal to 3, the continue statement makes the loop jump onto the next iteration, skipping the remainder of the code, which in this case is printing 'i'.

Try this

1. Print all values between 1 and 100, except if it's a multiple of 3.
2. Print all factors of the number in the input.

Topic 7: Using labels with continue and break keywords

In this we label a specific loop, just like we name a variable, and then use the continue or break statement to apply continue/break on that specific loop.

Example

```
first:
   for(int i = 0; i < 3; i++) {
      for(int j = 0; j < 3; j++) {
            if(i == 1 && j == 1)
                  continue first;
            System.out. println(i + " " + j);
        }
 }
```

Output

0 0
0 1
0 2
1 0
2 0
2 1
2 2

Explanation
As soon as we reach the continue statement, unlike normal scenarios, we move on to the next iteration of the outer loop that we labelled as “first”.

```
second:
       for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++) {
                 if(i == 1 && j == 1)
                      break second;
            System.out. println(i + “ “ + j);
       }
 }
```

Output
0 0
0 1
0 2
1 0

Explanation
As soon as we reach the break statement, unlike normal scenarios, we break out of the outer loop that we labelled as “first”.

Topic: Scanner class and User input in Java

In java to take the inputs from the keyboard we use Scanner class.
Scanner class is present in java.util package
To use this scanner class in our programs we need to import the Scanner class as shown below:

```
import java.util.Scanner
```

To take integer input from the user we use nextInt().
Similarly to take String input from the user we use next().

Write a program to accept 2 integer input from the user and perform addition operation?

```
import java.util.Scanner;

class TestApp {
        public static void main(String[] args) {
                Scanner scan = new Scanner(System.in);
                System.out.println("Enter First no::");
                int a = scan.nextInt();
                System.out.println("Enter Second no::");
                int b = scan.nextInt();

                System.out.println("The sum of 2 numbers is :"+(a+b));
       }
}
```

Step 1: Creating Object of Scanner Class
use new key word to create an object
new class_name(method);

Step 2: Store that Object in a variable
type_of_object variable_name = ...

Keyword to create obj
class
type of object class.

Write a program to accept String input from the user and display it on the console?

```
import java.util.Scanner;

class TestApp {
        public static void main(String[] args) {

                Scanner scan =new Scanner(System.in);
                System.out.println("Enter the username:: ");
                String name = scan.next();

                System.out.println("Enter the password:: ");
                String pwd= scan.next();

                System.out.println("The username is ::"+name);
                System.out.println("The password is ::"+pwd);
       }
}
```