Day 3 of Learning Java

Java Control Statements

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear. However, Java provides statements that can be used to control the flow of execution Java code. Such statements are called control flow statements.

It encompasses all mechanisms that control the execution path of the program, including:

- **Branching:** Decisions are made based on conditions (e.g., if, else, switch).
- **Loops:** Repeating blocks of code (for, while, do-while).
- **Jump Statements :** Continue & Break Statements

Decision- Making Statements / Selection

Decision-Making Statements (also called Selection Statements or Conditional Statements) are used to control the flow of the program based on conditions.

The decisions and execution of blocks is based on the whether a condition is true or false.

Simple if Statement

The if statement executes a block of code only if the condition is true.

Syntax:

- if (condition) {
  // Code to execute if condition is true
  }

if-else Statement

The if-else statement executes if block of code if the condition is true and else block if the condition is false.

Syntax:

- if (condition) {
  // Code to execute if condition is true
  } else {
  // Code to execute if condition is false
  }

Nested if-else Statement

A nested if-else statement is when an if-else block is placed inside another if or else block.

Syntax:

- if (condition1) {
  // Code if condition1 is true
  if (condition2) {
  // Code if condition2 is true
  } else {
  // Code if condition2 is false
  }
  } else {
  // Code if condition1 is false
  }

if-else-if ladder

The if-else-if statement contains the if-statement followed by multiple else-if statements. It allows a program to check several conditions and execute the block of code associated with the first true condition.

Syntax:

- if (condition1) {
  // Code to execute if condition1 is true
  } else if (condition2) {
  // Code to execute if condition2 is true
  } else {
  // Code to execute if none of the conditions are true
  }

Switch Statement

The switch statement allows us to execute a block of code among many alternatives. It is often used as an alternative to an if-else-if ladder.

Syntax:

- switch (expression) {
  case value1:
  // Code to execute if expression == value1
  break;
  case value2:
  // Code to execute if expression == value2
  break;
  default:
  // Code to execute if no cases match

}

Research:

1. What are the advantages of using a switch statement over multiple if-else statements?
2. What happens if we omit the break statement in a switch statement?

Repetition / Iteration / Looping Statement

Looping Statements are used to repeatedly execute a block of code as long as a certain condition is true. They allow you to avoid writing the same code multiple times.

For Loop

A for loop repeats a block of code if a condition is true. It's used when you know exactly how many times the code should run. The number of repetitions is determined by a set range or condition.

Syntax:

- for (initialization; condition; increment/decrement) {
  // Code to execute
  }

while Loop

The while loop is used when the number of iterations is not known in advance, and the loop runs as long as the condition is true. The condition is checked before each iteration.

Syntax:

- while (condition) {
  // Code to execute
  }

do-while Loop

A do-while loop is a control flow statement that executes a block of code at least once regardless of whether the condition is true or false initially and then repeats the execution as long as a specified condition is true.

do-while loop is also known as an "exit-controlled loop" because the condition is checked after the code block is executed.

o marks the start of the loop and executes the code inside the block. The condition is checked after the loop executes. If the condition evaluates to true, the loop will repeat. If it's false, the loop will exit.

Syntax:

- do {
  // Code to be executed
  } while (condition);

for-each Loop / Enhanced for loop

The for-each loop is used to iterate over elements in an array or a collection (such as a list or set) without needing to use an index.

Syntax:

- for (datatype variable : collection) {
  // Code to be executed
  }

Jump Statements

Jump statements are used to transfer the control/execution of the program to a different part of the program. The jump statements help control the flow of execution in loops, switch cases, and methods in Java.

There are three main types of jump statements:

Break Statement

The break statement is used to exit from a loop or switch statement before it completes its normal execution.

It breaks only the inner loop in the case of the nested loop. The break statement cannot be used independently in the Java program, i.e., it can only be written inside the loop or switch statement.

Continue Statement

The continue statement is used to skip the current iteration of a loop and proceed to the next iteration.

Return Statement

The return statement is used to exit from a method and optionally return a value to the calling code.

In methods: Exits the method and returns control to the caller.

Revision

1. Write a Java program that acts as a basic calculator. The program should:
   Take two integers as input from the user.
   Provide a menu to the user with options to add, subtract, multiply, or divide the numbers.
   Use if-else or switch statements to perform the corresponding operation.
   Print the result of the chosen operation.

   Example Output:

```
>> Enter the first number: 10
>> Enter the second number: 5
>> Choose an operation:
>> 1. Add
>> 2. Subtract
>> 3. Multiply
>> 4. Divide
>> Enter your choice (1-4): 3
>> Result: 10 * 5 = 50
>>
```

2. Write a program that calculates the factorial of a given number using a for loop or while loop. The factorial of a number n is the product of all positive integers less than or equal to n (i.e., n! = n * (n-1) * ... * 1).

3. Write a program that prints the multiplication table for a given number. The program should:
   Take an integer n as input.
   Use a for loop to print the multiplication table for n from 1 to 10.