DAY1

JAVA Programming Concepts: Notes with Examples

1. What is Programming?

Definition: Programming is the process of giving a set of instructions to a computer to perform a specific task or solve a problem. Think of it like writing a recipe for a computer to follow.

Example: Imagine you want a computer to add two numbers and display the result.

- **Instruction 1:** Get the first number (e.g., 5).
- **Instruction 2:** Get the second number (e.g., 3).
- **Instruction 3:** Add the first and second numbers.
- **Instruction 4:** Show the result.

In a programming language like Java, this might look like:

Java

```
public class AdditionExample {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 3;
        int sumResult = num1 + num2;
        System.out.println(sumResult); // Output: 8
    }
}
```

---

2. Importance of Programming

Concept: Programming is crucial because it allows us to automate repetitive tasks, build software applications (like websites, mobile apps, games), analyze large amounts of data, and control machines. It's how we tell computers to do useful things for us efficiently and accurately.

Example:

- **Automation:** Instead of manually sending "Happy Birthday" emails to 100 employees, you can write a program that automatically sends personalized emails on their birthdays.
- **Websites:** When you browse Google, Facebook, or Amazon, you're interacting with programs written by developers.
- **Data Analysis:** Scientists use programming to process vast datasets to discover patterns, for instance, predicting weather patterns or analyzing medical research.

DAY1

3. Programming Paradigms

Concept: Programming paradigms are fundamental styles or approaches to how you structure your code and solve problems. The video mentioned two main types: Imperative and Declarative.

a. Imperative Programming (How to do)

Concept: This paradigm focuses on *how* a program should execute. You provide explicit step-by-step instructions that change the program's state.

Example:

- **Procedural Programming:** Organizing code into reusable blocks called methods (or functions in other languages).
  - **Scenario:** Calculate the area of a rectangle.

Java

```
public class RectangleAreaCalculator {
  public static double calculateRectangleArea(double length, double width) {
    double area = length * width;
    return area;
  }

  public static void main(String[] args) {
    // Call the method
    double myArea = calculateRectangleArea(10.0, 5.0);
    System.out.println(myArea); // Output: 50.0
  }
}
```

DAY1

- **Object-Oriented Programming (OOP):** Organizing code around "objects" which are instances of "classes" (blueprints) that combine data (attributes/fields) and behavior (methods).
    - **Scenario:** Represent a car with its make, model, and ability to start.

Java

```
public class Car {
    String make;
    String model;

    // Constructor
    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }

    // Method
    public String start() {
        return "The " + this.make + " " + this.model + " is starting.";
    }

    public static void main(String[] args) {
        // Create an object (instance) of the Car class
        Car myCar = new Car("Toyota", "Camry");
        System.out.println(myCar.start()); // Output: The Toyota Camry is starting.
    }
}
```

DAY1

b. Declarative Programming (What to do)

Concept: This paradigm focuses on *what* needs to be done, rather than specifying the exact steps to do it. You describe the desired result, and the system figures out how to achieve it.

Example:

- **Functional Programming (using Java 8 Streams and Lambdas):** While Java is primarily object-oriented, it incorporates functional programming elements with Streams API and lambda expressions, allowing you to express *what* to do with collections of data.
    - **Scenario:** Filter even numbers from a list.

Java

```
import java.util.Arrays;

import java.util.List;

import java.util.stream.Collectors;


public class FunctionalFilterExample {

  public static void main(String[] args) {

    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6);


    // Using a declarative approach with Java Streams

    List<Integer> evenNumbers = numbers.stream()

                        .filter(n -> n % 2 == 0) // What: filter numbers where they are even

                        .collect(Collectors.toList());

    System.out.println(evenNumbers); // Output: [2, 4, 6]


    // Contrast with an imperative approach (traditional loop)

    /*

    List<Integer> resultImperative = new java.util.ArrayList<>();

    for (int num : numbers) {

      if (num % 2 == 0) {

        resultImperative.add(num);

      }

    }

    System.out.println(resultImperative);
    */
  }
}
```

- **Database Programming (SQL - Structured Query Language):** This is a prime example of declarative programming, as it's language-agnostic for the most part. You declare what data you want, not how to retrieve it.
  - **Scenario:** Select all users from a database table named 'Users' who are older than 30.

SQL

```
SELECT *
FROM Users
WHERE Age > 30;
```

You don't tell the database *how* to search the table; you just tell it *what* you want.

---

4. Choosing a Programming Language

Concept: The "best" programming language doesn't exist; the right choice depends on your project's requirements, your career goals, and the specific domain you're working in.

Examples:

- **Enterprise Applications/Android Apps/Big Data:** Java is a very strong choice due to its robustness, scalability, and ecosystem.
- **Web Development (Backend):** Java (Spring Boot), Python (Django, Flask), Node.js (JavaScript), Ruby (Ruby on Rails), PHP.
- **Mobile Apps:** Swift/Objective-C (iOS), Kotlin/Java (Android), React Native/Flutter (cross-platform).
- **Data Science/Machine Learning:** Python (with libraries like Pandas, NumPy, scikit-learn), R.
- **Game Development:** C++, C#, Java (less common for AAA games, but used in some indie games and mobile games like Minecraft).
- **System Programming/Performance:** C, C++.