🧠 GUIA DE ESTUDO – TEMA 07

Programação Orientada a Objetos (POO) – Conceitos e Prática

💡 Objetivo

Compreender os conceitos iniciais da Programação Orientada a Objetos (POO) e aplicá-los na reestruturação de um código que utiliza arrays para manipulação de dados, transformando-o em uma versão orientada a objetos.

🔍 Parte 1 – Revisando os Conceitos-Chave de POO

Conceito	Definição
Classe	Um molde para criação de objetos, definindo atributos (dados) e métodos (ações).
Objeto	Uma instância de uma classe. É criado com new.
Atributos	As características de um objeto (ex: nome, idade, salário).
Métodos	As ações ou comportamentos de um objeto (ex: exibirDados, calcularSalario).
Instância	O ato de criar um objeto a partir de uma classe.
Abstração	Foco nas informações essenciais do objeto, ignorando os detalhes complexos.
Encapsulamento	Proteção dos dados, mantendo atributos privados e acesso via métodos públicos (get/set).

🔧 Parte 2 – Analisando o Código Tema6

O código atual usa uma **matriz bidimensional de Strings** para armazenar dados como ID, Nome, Idade e Salário. Isso simula um "banco de dados", mas **não utiliza POO**.

Problemas desse modelo:

- Os dados estão acoplados e sem estrutura clara.
- Não há encapsulamento.
- O código não é reutilizável nem modular.
- Difícil de expandir (ex: adicionar CPF, cargo etc.).

🛠️ Parte 3 – Refatorando com POO

Vamos criar uma classe chamada Pessoa com os atributos necessários e métodos úteis.
Classe Modelo:

```
public class Pessoa {
    private int id;
    private String nome;
    private int idade;
    private double salario;

    // Construtor
    public Pessoa(int id, String nome, int idade, double salario) {
        this.id = id;
        this.nome = nome;
        this.idade = idade;
        this.salario = salario;
    }

    // Métodos Getters
    public int getId() { return id; }
    public String getNome() { return nome; }
    public int getIdade() { return idade; }
    public double getSalario() { return salario; }

    // Método para exibir dados
    public void exibirDados() {
        System.out.println("ID: " + id + ", Nome: " + nome + ", Idade: " + idade + ", Salário: R$" +
salario);
    }
}
```

🚀 Parte 4 – Aplicando a Classe no Programa Principal

```
public class Principal {

    public static void main(String[] args) {
        // Criando objetos Pessoa
        Pessoa[] pessoas = {
            new Pessoa(1, "Joao", 25, 2500.00),
            new Pessoa(2, "Maria", 30, 3000.00),
            new Pessoa(3, "Carlos", 35, 4000.00)
        };

        // Exibir os dados (método de cada objeto)
        System.out.println("=== DADOS PESSOAIS ===");
        for (Pessoa p : pessoas) {
            p.exibirDados();
        }

        // Estatísticas simples
        double somaSalarios = 0;
        int somaIdades = 0;
```

```
        for (Pessoa p : pessoas) {
            somaSalarios += p.getSalario();
            somaIdades += p.getIdade();
        }

        System.out.println("\nMédia de Idade: " + (somaIdades / pessoas.length));
        System.out.println("Média Salarial: R$ " + (somaSalarios / pessoas.length));
    }
}
```

🧾 Parte 5 – Exercícios de Fixação

1. Crie um método aumentarSalario(double percentual) dentro da classe Pessoa. Depois criar uma outra classe para instanciar 3 pessoas e aumentar o salário de todos em 10%.
2. Crie outro atributo chamado cargo e atualize o construtor e a exibição.

📌 Resumo Visual

Termo	Exemplo no Código
Classe	Pessoa
Objeto	new Pessoa(...)
Atributo	nome, idade, salario
Método	exibirDados(), getIdade()
Instância	Pessoa p = new Pessoa(...)

🧩 Dica Extra

Você pode usar a ferramenta **UML (Unified Modeling Language)** para representar graficamente a estrutura da classe Pessoa!