Resumo para Prova – Orientação a Objetos (POO)

RESUMO PARA PROVA – ORIENTAÇÃO A OBJETOS (POO)

**O que é Programação Orientada a Objetos?**

É um paradigma de programação que organiza o código em objetos, que são instâncias de classes. Cada objeto possui atributos (dados) e métodos (funções). **LEMBRA DISSO**

PRINCIPAIS CONCEITOS

1. Classe

- Um molde ou estrutura para criar objetos.

- Define os atributos e comportamentos que o objeto terá.

**Exemplo:**

**class Livro:**

**def \_\_init\_\_(self, titulo, autor):**

**self.titulo = titulo**

**self.autor = autor**

2. Objeto

- Uma instância de uma classe, como um objeto real

**Exemplo:**

**meu\_livro = Livro("Dom Casmurro", "Machado de Assis")**

3. Atributos

- Características do objeto, geralmente definidos no \_\_init\_\_, sempre colocado dentro do ()

**Exemplo:**

**self.titulo = titulo**

4. Métodos

- Ações que os objetos podem realizar (funções dentro da classe).

**Exemplo:**

**def exibir\_info(self):**

**return f"{self.titulo} por {self.autor}"**

**Mostro o autor e o titulo**

PILARES DA POO

1. Encapsulamento

Proteger os dados e permitir acesso controlado por métodos.

**Exemplo:**

**class Conta:**

**def \_\_init\_\_(self):**

**self.\_saldo = 0**

**def depositar(self, valor):**

**self.\_saldo += valor**

**#LEMBRAR DO GET PARA ENCAPSULAMENTO**

**def get\_saldo(self):**

**return self.\_saldo**

2. Herança

Uma classe pode herdar atributos e métodos de outra classe (classe pai).

**Exemplo:**

**class Animal:**

**def falar(self):**

**print("Algum som")**

**#CACHORRO RECEBE OS ATRIBUTOS DE ANIMAL**

**class Cachorro(Animal):**

**def falar(self):**

**print("Latir")**

3. Polimorfismo

Permite usar o mesmo método com comportamentos diferentes em classes diferentes.

**Exemplo:**

**def fazer\_animal\_falar(animal):**

**animal.falar() # Pode ser um gato, cachorro, etc.**

4. Abstração

Esconder detalhes internos e mostrar apenas o necessário.

**Exemplo:**

**minha\_conta.saque(100)**

OUTROS CONCEITOS IMPORTANTES

**Método \_\_init\_\_()**

Construtor da classe – é chamado automaticamente ao criar um objeto.

**self**

Referência ao próprio objeto – usada dentro da classe para acessar atributos e métodos.

**Modificadores de Acesso**

- Público: nome

- Protegido: \_nome

- Privado: \_\_nome (evita acesso externo direto)

DICAS, LEMBRA LETICIA

- Sempre defina \_\_init\_\_ com os atributos principais.

- Use super().\_\_init\_\_() quando estiver trabalhando com herança, para puxar os atributos do pai

- Nomes de atributos e métodos devem ser coerentes e padronizados.

- Teste com objetos reais para garantir que as classes funcionam corretamente.

- Treine criar classes com pelo menos atributos, métodos, e um caso de herança.