# POO

Introdução à programação Orientada a Objetos

Conceito:

POO é um paradigma de programação que organiza o código em torno de objetos, que combina atributos e comportamentos (métodos).

Essa área facilita a modularidade, reutilização e manutenção de um código.

1. Classes: Moldes que definem atributos de objetos. Facilitam a organização e reutilização do código

2. Objetos: Instâncias de classes que representam entidades com estado e comportamento.

3. Abstração: Oculto detalhes desnecessários, de maneira que outras classes reutilizem os métodos da classe principal de maneira automata

4. Heronça: Permite que classes herdem características e métodos em comum com uma classe "principal" ou "mãe".

5. Encapsulamento: Protege os dados de um objetos, controlando o acesso por meio de métodos.

6. Polimorfismo: Permite que metodos tenham diferentes comportamentos dependendo do contexto, promovendo flexibilidade.

# 2. POLIMORFISMO:

▸ Conceito:

↳ O Polimorfismo, um dos princípios fundamentais da Programaçaõ Orientada a Objectos (POO), é a capacidade de objetos de diferentes classes responderem a métodos com o mesmo nome, mas com comportamentos distintos, estando de acordo com as suas implementaçoẽs especificas.

Derivado do grego, o termo "polimorfismo" significa "muitas formas". Na POO, ele permite que um método ou operaçaõ seja executado de diferentes formas, dependendo do contexto ou do tipo particular do objeto que o invoca. Assim, ele representa o quarto, e ultimo, pilar da Programaçaõ Orientada a Objetos.

▸ Polimorfismo de Sobrecarga (Overloading):

↳ Acontece quando uma classe contém múltiplos métodos com o mesmo nome, mas possui assinaturas diferentes. Em outras palavras, o mesmo nome de método é utilizado várias vezes, porém, cada versão do método aceita distintos tipos ou quantidade de argumentos.

▸ Polimorfismo de Sobrescrita (Overriding):

↳ Ocorre quando uma classe filha redefine um método da classe mãe, mantendo o mesmo nome, tipo de retorno e parâmetros. A implementação do método na classe filha pode ser diferente da que há na classe mãe. Basicamente, quer dizer que a assinatura é a mesma, entretanto, as classes são distintas.

# 3. Classes Abstratas

▸ Conceito:

↳ Classes abstratas são classes que não podem ser instanciadas diretamente, elas servem de modelo para outras classes.

▸ Características e regras:

↳ Na linguagem Java, uma classe é declarada abstrata usando a palavra-chave "abstract".
Elas podem ter métodos abstratos e não abstratos, sendo os abstratos sem corpo.
Os métodos abstratos são obrigatóriamente implementados na classe filha.

▸ Classe abstrata e Classe comum:

↳ A classe abstrata não pode ser instanciada, pode conter métodos abstratos, no geral é a classe mãe. A classe comum pode ser instânciada, não contém métodos abstratos (se uma classe possui um método abstrato ela é obrigatóriamente uma classe abstrata), geralmente é usada para criar objetos e pode ou não ser a classe mãe.

# 4. Interface

- Definição em POO:

↳ Uma interface é um tipo que estabelece um contrato entre classes, definindo um conjunto de métodos abstratos que uma classe deve implementar.

- Contrato entre Classes:

↳ Elas garantem que tenha um padrão de métodos entre as classes, mas, dão liberdade para cada classe implementar os métodos de forma personalizada.

- Diferencie uma interface de uma classe abstrata:

↳ Em uma interface todos os métodos são abstratos, possuem apenas atributos constantes e uma classe pode implementar várias interfaces.
Em uma classe abstrata pode ter métodos abstratos e concretos, possuem atributos normais com qualquer modificador e uma classe pode herdar apenas uma classe abstrata.

5. Classe abstrata e interface

▸ Faça uma comparação entre classe abstrata e interface:

↳ Classes abstratas e interfaces são usadas para estruturar sistemas e definir comportamentos. Ambas podem ter métodos Abstratos, mas as classes abstratas também permitem métodos concretos e atributos. As classes abstratas permitem herança simples e as interfaces herança multipla

▸ Quando utilizar classe abstrata ou interface:

↳ Classes abstratas são mais indicadas para hierarquias de Classes relacionadas, enquanto interfaces são ideais para definir contratos comuns em sistemas mais diversos e flexíveis.

▸ Vantagens e desvantagens:

↳ As abstratas compartilham lógicas em comum e usam atributos de instância, mas limitam a herança. As interfaces, por sua vez, oferecem flexibilidade da herança multipla, mas não permitem atributos de instância e exige a implementação de todos os métodos nas demais classes

## 6. CONCLUSÃO:

↳ Na Programaçaõ Orientada a Objetos (POO), o polimorfismo, as classes abstratas e as interfaces saõ essenciais para o desenvolvimento de sistemas eficientes.

O polimorfismo possibilita que objetos distintos respondam de forma especifica a métodos com a mesma assinatura, promovendo adaptabilidade e reduzindo o acoplamento.

As classes abstratas estabelecem estruturas base que garantem uniformidade e permitem personalizaçoẽs controladas em suas subclasses.

As interfaces, por sua vez, definem contratos que asseguram a integraçaõ entre diferentes componetes. Esses conceitos, aplicados em conjunto, promovem modularidade, escabilidade e facilidade na manutençaõ dos softwares.