1. O que é?
2. Vantagens e desvantagens
3. Diferença do POO e Programação Estruturada
4. Cite um exemplo de cada conceito (Código)

* Classe
* Objeto
* Propriedades
* Métodos
* Encapsulamentos
* Herança
* Polimorfismo
* Abstração
* Get e Set
* Classe Abstrata
* Associação
* Agregação
* Dependencia

**CLASSE:**

1. Uma classe em programação orientada a objetos é um modelo ou um "molde" para criar objetos. Ela define os atributos (variáveis) e métodos (funções) que todos os objetos de um tipo específico terão em comum. Pense em uma classe como um plano ou uma planta baixa para construir diferentes instâncias de objetos com características e comportamentos semelhantes.

Por exemplo, se você estiver criando um programa para gerenciar carros, você pode ter uma classe chamada "Carro". Essa classe pode ter atributos como modelo, cor e ano, e métodos como ligar(), desligar() e acelerar().

Uma vez que você tenha definido essa classe, você pode criar objetos individuais (também chamados de instâncias) dessa classe. Por exemplo, você pode criar um objeto carro1 que representa um Toyota Corolla azul de 2019 e um objeto carro2 que representa um Ford Fusion vermelho de 2020. Ambos os objetos teriam os mesmos atributos e métodos definidos na classe "Carro", mas seus valores podem ser diferentes para cada objeto.

1. Reutilização de código: Classes permitem a criação de objetos que encapsulam funcionalidades específicas, tornando mais fácil reutilizar esse código em diferentes partes do programa ou em projetos diferentes.

Abstração: Classes permitem modelar entidades do mundo real de forma mais abstrata, o que facilita a compreensão e manutenção do código. Isso também ajuda na identificação de padrões e na resolução de problemas complexos.

Encapsulamento: Classes encapsulam dados (atributos) e comportamentos (métodos) relacionados em uma única unidade, o que ajuda a proteger os dados sensíveis e a manter a integridade do objeto.

Modularidade: Classes promovem a modularidade do código, dividindo-o em unidades independentes e coesas. Isso facilita a manutenção, teste e extensão do software.

Herança: Classes podem herdar atributos e métodos de outras classes, permitindo a criação de hierarquias de classes que compartilham comportamentos comuns e evitando a duplicação de código.

Complexidade: Em projetos grandes, a utilização extensiva de classes pode aumentar a complexidade do código, tornando-o mais difícil de entender e dar manutenção.

Overhead: Classes adicionam um certo overhead de memória e processamento devido à alocação de objetos e chamadas de métodos. Em situações em que a eficiência é crucial, isso pode ser uma desvantagem.

Herança múltipla: Em algumas linguagens de programação, como C++, a herança múltipla pode levar a problemas de ambiguidade e complexidade. Embora seja uma vantagem em certos casos, pode ser difícil de gerenciar.

Acoplamento: Se não forem bem projetadas, as classes podem se tornar fortemente acopladas umas às outras, o que pode dificultar a reutilização e a manutenção do código.

Overengineering: Em alguns casos, a aplicação de conceitos de POO pode levar a uma estrutura excessivamente complexa e genérica, que é desnecessária para os requisitos do projeto, resultando em um código mais difícil de entender e manter.

**OBJETO:**

1. Um objeto, na programação orientada a objetos (POO), é uma instância de uma classe. Uma classe é um modelo que descreve um conjunto de características (atributos) e comportamentos (métodos) que os objetos criados a partir dela possuem.

Quando você cria um objeto a partir de uma classe, está essencialmente criando uma variável que pode armazenar dados e métodos relacionados. Cada objeto tem seu próprio conjunto de valores para os atributos da classe, permitindo que cada instância seja única.

Por exemplo, considere uma classe Carro. Esta classe pode ter atributos como cor, modelo, ano, etc., e métodos como ligar, acelerar, frear, etc. Quando você cria um objeto a partir da classe Carro, digamos um carro vermelho de modelo "X", ano "2022", você está criando uma instância única dessa classe com seus próprios valores para os atributos. Este é um objeto.

Em resumo, um objeto é uma instância específica de uma classe que tem seus próprios dados e comportamentos, mas segue o modelo definido pela classe.