Java

POO

O que é Programação Orientada a Objetos (POO)?

A Programação Orientada a Objetos (POO) é um paradigma de programação que se baseia na organização e estruturação do código em torno de objetos e suas interações.

Os objetos representam entidades do mundo real ou conceitos abstratos, e a POO visa modelar o comportamento e as características dessas entidades por meio de classes.

Dica: *Na explicação da POO, é importante enfatizar os conceitos fundamentais, como classes, objetos, encapsulamento, herança e polimorfismo. Explique como a POO promove a reutilização de código, a modularidade e a manutenção do software.*

Quais os conceitos de classe e objeto em POO?

***Classe**: Uma classe é um modelo, uma espécie de "planta baixa" que define o conjunto de atributos (características) e métodos (comportamentos) que os objetos desse tipo possuirão. É uma estrutura que encapsula os dados e as operações que podem ser realizadas com esses dados. Em outras palavras, uma classe é uma abstração que descreve o que um objeto pode fazer e quais informações ele pode armazenar.*

***Dica**: Quando explicar uma classe, destaque que ela é apenas uma definição, um molde, e não ocupa espaço na memória durante a execução do programa. É como um plano que define a estrutura e o comportamento dos objetos que serão criados a partir dele.*

***Objeto**: Um objeto é uma instância concreta de uma classe. Representa uma entidade específica com seus dados (atributos) e comportamentos (métodos) definidos pela classe a partir da qual foi criado. Os objetos são criados em tempo de execução, a partir da classe, e ocupam espaço na memória.*

***Dica**: Enfatize que os objetos são os "indivíduos" reais criados com base na classe. Assim como podemos ter várias casas construídas com base em um único projeto arquitetônico (classe), podemos ter múltiplos objetos criados a partir de uma única definição de classe.*

Quais são os quatro pilares da POO e o que eles representam?

Encapsulamento:

É o conceito de agrupar os dados (atributos) e comportamentos (métodos) relevantes de uma classe, tornando-os privados e controlados pelo próprio objeto. Isso significa que o acesso direto aos atributos é restrito, e o acesso a eles deve ser feito por meio de métodos públicos (getters e setters).

O encapsulamento permite a ocultação dos detalhes internos de implementação de uma classe, tornando o código mais seguro e facilitando a manutenção e evolução do sistema.

Herança:

Permite criar uma nova classe (subclasse ou classe derivada) com base em uma classe já existente (superclasse ou classe base). A subclasse herda os atributos e métodos da superclasse, podendo adicionar novos atributos e comportamentos específicos, ou mesmo sobrescrever os métodos herdados para personalizar o comportamento.

A herança promove a reutilização de código e a organização hierárquica de classes, facilitando a modelagem de objetos com características comuns.

Polimorfismo:

Permite que uma classe se comporte de diferentes formas, dependendo do contexto em que é usada. Isso pode ser alcançado por meio da sobrecarga de métodos (métodos com o mesmo nome, mas com parâmetros diferentes) e da sobreposição de métodos (métodos com a mesma assinatura, mas com implementações diferentes em subclasses).

O polimorfismo permite escrever código mais genérico e flexível, onde objetos de diferentes classes podem ser tratados de maneira uniforme.

Abstração:

É o conceito de representar as características essenciais de um objeto do mundo real, focando apenas nos detalhes relevantes para o contexto do problema. As classes são criadas como abstrações para modelar os objetos, e apenas os atributos e métodos relevantes para o objetivo da classe são definidos.

A abstração permite criar modelos simplificados e mais fáceis de entender, reduzindo a complexidade do sistema e facilitando sua compreensão.

Java

Forjando Devs