**ETAPA 1 - TEÓRICA**

1. **O que é orientação a objetos e quais são seus princípios fundamentais?**

A orientação a objetos é um paradigma de programação que organiza o código em unidades chamadas objetos, que representam entidades do mundo real. Seus princípios fundamentais são:

**Abstração**: Representa características essenciais de um objeto, ignorando detalhes irrelevantes.

**Encapsulamento**: Oculta os detalhes de implementação de um objeto, expondo apenas a interface necessária para interagir com ele.

**Herança**: Permite criar novas classes aproveitando características de classes existentes, promovendo reutilização de código.

**Polimorfismo**: Permite que objetos de diferentes classes sejam tratados de maneira uniforme, promovendo flexibilidade e extensibilidade no código.

1. **Quais são as vantagens da orientação a objetos em comparação com paradigmas de programação procedural?**

A orientação a objetos oferece vantagens significativas em comparação com paradigmas de programação procedural:

**Reutilização de código**: Facilita a reutilização através de herança e composição.

Encapsulamento: Protege os dados e comportamentos dentro dos objetos.

**Abstração**: Permite modelar entidades do mundo real de forma simplificada.

**Manutenção** **e** **evolução**: Alterações tendem a ser localizadas e a estrutura é flexível para mudanças.

**Modularidade**: Organiza o código em unidades autônomas, facilitando a compreensão e modificação.

**Desenvolvimento** colaborativo: Permite que equipes trabalhem em partes separadas do sistema de forma independente.

1. **Explique o conceito de encapsulamento e como ele é implementado em linguagens de programação orientadas a objetos**.

O encapsulamento é um princípio da programação orientada a objetos que oculta os detalhes internos de um objeto, expondo apenas uma interface controlada para interações externas. Em linguagens orientadas a objetos, isso é alcançado através do uso de modificadores de acesso, como público, protegido e privado. O encapsulamento protege os dados de um objeto contra acesso não autorizado e promove a manutenção e evolução do código, permitindo alterações na implementação interna do objeto sem afetar o restante do sistema, desde que a interface pública permaneça a mesma.

1. **Explique o conceito de classe e objeto em programação orientada a objetos, e como eles estão relacionados**

Em programação orientada a objetos, uma classe é um modelo ou plano para criar objetos. Ela define os atributos (dados) e métodos (ações) que os objetos de uma determinada classe podem ter. Um objeto é uma instância de uma classe - uma entidade concreta criada com base nesse modelo, com suas próprias características únicas. Em resumo, uma classe é como um projeto e um objeto é uma instância desse projeto em uso.

1. **Qual é a importância dos construtores e métodos em uma classe em programação orientada a objetos?**

Os construtores são métodos especiais usados para inicializar objetos quando são criados. Eles permitem que você defina como um objeto deve ser configurado inicialmente. Métodos em uma classe definem o comportamento ou ações que os objetos dessa classe podem executar. Eles são essenciais para interagir com e modificar os objetos de uma maneira controlada e consistente. Em resumo, construtores e métodos são cruciais para definir o estado inicial e o comportamento de objetos em programação orientada a objetos.

1. **Quais são as diferenças entre métodos estáticos e métodos de instância em uma classe?**

- Métodos de instância são associados a objetos específicos e podem acessar e modificar os dados desses objetos.

- Métodos estáticos não estão vinculados a instâncias específicas; eles pertencem à classe como um todo e podem ser invocados sem criar uma instância da classe. Eles geralmente são usados para operações que não dependem do estado de um objeto específico, como funções utilitárias ou operações de configuração.

1. **Como são tratados os conceitos de acoplamento e coesão na orientação a objetos e por que são importantes?**

Acoplamento refere-se à dependência entre partes do código, enquanto coesão trata da relação entre os elementos dentro de uma unidade de código. Ambos são importantes na orientação a objetos para promover flexibilidade e organização no software. Acoplamento baixo e alta coesão tornam o código mais modular, fácil de entender e manter.

1. **Qual é o objetivo do metodo main?**

O método main é o ponto de entrada de um programa Java. Ele é onde a execução do programa começa e é responsável por iniciar a aplicação, chamando outros métodos conforme necessário. Em resumo, o objetivo do método main é iniciar a execução do programa e servir como ponto de partida para a lógica principal do software.

1. **Como ocorre a passagem de parâmetros em métodos de uma classe em programação orientada a objetos?**

Em programação orientada a objetos, a passagem de parâmetros em métodos ocorre quando valores são transmitidos para os métodos durante sua invocação. Esses parâmetros são usados dentro do método para realizar operações específicas. Eles podem ser passados por valor ou por referência, dependendo da linguagem de programação e da definição do método. Em resumo, os parâmetros são meios de fornecer informações necessárias para que os métodos realizem suas tarefas.

1. **Qual é a diferença entre os modificadores de acesso "public", "private" e "protected" em uma classe?**

**Public**: Os membros (atributos e métodos) são acessíveis de qualquer lugar, dentro ou fora da classe.

**Private**: Os membros são acessíveis somente dentro da própria classe. Não podem ser acessados por classes externas.

**Protected**: Os membros são acessíveis dentro da própria classe e por subclasses (herança), mas não são acessíveis fora da hierarquia de herança.

1. **Qual é a importância da palavra-chave "this" em programação orientada a objetos e como ela é utilizada?**

Em programação orientada a objetos, a palavra-chave "this" é usada para se referir ao objeto atual dentro de uma classe. Ela é importante para distinguir entre variáveis locais e membros da classe que têm o mesmo nome, além de ser usada para chamar métodos ou acessar atributos do próprio objeto. Em resumo, "this" é usado para evitar ambiguidades e facilitar o acesso aos membros da classe dentro dos métodos.