**Revisão Programação orientada a objetos em Java**

**1- Explique o conceito de classe em programação orientada a objetos e sua importância na definição de objetos em Java.**

**Uma classe em programação orientada a objetos define a estrutura e o comportamento de um tipo de objeto.**

**Em Java, sua importância está na organização de atributos e métodos, garantindo reutilização de código, encapsulamento e modularidade, permitindo a criação de objetos bem estruturados.**

**2-** **Defina o que é um objeto em Java e como ele se relaciona com a definição de uma classe.**

**Em Java, um objeto se relaciona com uma classe ao ser sua instância, que define atributos e métodos.**

**O objeto concretiza esses conceitos na memória durante a execução do programa, funcionando como a construção final do projeto arquitetônico.**

**3- Descreva os principais membros de uma classe em Java e a função de cada um (campos e métodos).**

**Campos (Atributos): Variáveis dentro da classe que armazenam o estado dos objetos. Podem ser de instância (individuais) ou estáticos (compartilhados). Exemplo: nome, idade e CPF na classe Pessoa.**

**Métodos: Definem comportamentos e operações dos objetos. Manipulam atributos e permitem ações. Métodos de instância atuam nos atributos do objeto, enquanto estáticos pertencem à classe e não precisam de um objeto para serem chamados.**

**Juntos, possibilitam a criação de objetos organizados, garantindo encapsulamento e modularidade na programação orientada a objetos.**

**4-** **Explique o processo de compilação de uma classe Java para bytecode e sua importância na portabilidade da linguagem.**

**Em Java, o código-fonte é convertido em bytecode pelo compilador javac, tornando-o independente de hardware e sistema operacional.**

**Na execução, a JVM interpreta ou compila esse bytecode para código nativo, permitindo que o programa rode em qualquer plataforma com JVM, seguindo o princípio Write Once, Run Anywhere.**

**5- Qual é a diferença entre atributos públicos, privados e protegidos em Java? Dê exemplos de quando usar cada um.**

**Público (public): Pode ser acessado de qualquer lugar. Use para constantes ou atributos/métodos que devem ser amplamente acessíveis. Exemplo: public static final double PI = 3.14159;**

**Privado (private): Só pode ser acessado dentro da própria classe. É ideal para dados sensíveis, garantindo o encapsulamento. Exemplo: private String nome; (gerenciado via getters e setters)**

**Protegido (protected): Permite acesso no mesmo pacote e em subclasses, até de pacotes diferentes. Ideal para expor atributos apenas a classes derivadas sem exposição total. Exemplo: protected int saldo em uma classe de contas bancárias.**

**6-** **Explique o conceito de herança em Java e como ela é aplicada para promover a reutilização de código.**

**A herança em Java permite que uma classe (subclasse) reutilize atributos e métodos de outra classe (superclasse).**

**Isso evita duplicação de código e facilita a manutenção. Por exemplo, uma classe Animal pode ter métodos como comer() e dormir(), e uma classe Cachorro pode herdar esses métodos e adicionar latir().**

**7-** **Diferencie herança simples e herança múltipla em Java, e explique como a linguagem resolve limitações relacionadas à herança múltipla.**

**Herança simples ocorre quando uma classe herda de apenas uma outra classe, facilitando a estrutura e evitando ambiguidades.**

**Já a herança múltipla, que permitiria uma classe herdar de várias classes, não é suportada diretamente em Java devido a possíveis conflitos de implementação.**

**Java resolve a limitação da herança múltipla com interfaces, permitindo que uma classe implemente vários métodos sem conflitos. Isso garante flexibilidade e organização no código.**

**8- Explique o que são interfaces em Java, suas características principais e como elas permitem a implementação de múltiplas heranças.**

**Interfaces em Java definem um conjunto de métodos que uma classe deve implementar, funcionando como um contrato.**

**Suas características incluem a ausência de implementação concreta, a possibilidade de múltiplos comportamentos sem herança múltipla e, desde Java 8, o suporte a métodos padrão e estáticos.**

**Elas permitem a implementação de múltiplas heranças de forma segura, evitando conflitos.**

**9- Disserte sobre o encapsulamento em Java, detalhando como ele protege os dados e permite o acesso controlado aos atributos de uma classe.**

**O encapsulamento em Java oculta os detalhes internos de uma classe, restringindo o acesso direto a seus atributos.**

**Os atributos são privados e acessados por métodos públicos (getters e setters), protegendo os dados contra alterações indevidas e garantindo a integridade e manutenção do estado do objeto.**

**10-** **Defina polimorfismo em Java e exemplifique como ele pode ser aplicado na programação orientada a objetos.**

**O polimorfismo em Java permite que um mesmo método tenha diferentes comportamentos conforme a classe que o implementa.**

**Uma classe Animal pode definir o método fazerSom(), e subclasses como Cachorro e Gato podem sobrescrevê-lo para emitir um latido ou um miado. Assim, ao chamar fazerSom() em uma referência Animal, o método adequado será executado.**

**11- Explique o papel dos métodos em uma classe Java e como eles definem o comportamento dos objetos.**

**Em Java, os métodos são blocos de código que definem as ações dos objetos, encapsulando a lógica e manipulando atributos.**

**Eles determinam o comportamento da classe, permitindo que os objetos executem tarefas de forma organizada e modular.**

**12-** **Analise a seguinte afirmação: “Um objeto de uma classe é apenas uma definição, pois a ação só ocorre quando o objeto é invocado através de um método”. Você concorda? Justifique sua resposta.**

**Não, não concordo. Um objeto não é apenas uma definição, mas sim uma instância concreta de uma classe que possui estado (seus atributos) e comportamento (seus métodos).**

**Enquanto a classe serve como um molde, o objeto representa a materialização dessa definição, onde os métodos são usados para executar ações e manipular o estado. (pessoal).**

**13-** **Explique as diferenças entre instância e identidade no contexto de programação orientada a objetos.**

**Em programação orientada a objetos, instância é o objeto concreto criado a partir de uma classe.**

**Enquanto identidade refere-se à unicidade de cada objeto, ou seja, a sua referência única na memória. Mesmo se duas instâncias tiverem os mesmos valores, elas mantêm identidades distintas.**

**14- Descreva as características de um método público e um método privado e discuta como a herança afeta o acesso a esses métodos em subclasses.**

**Os métodos públicos podem ser acessados por qualquer classe, incluindo subclasses, e podem ser herdados e sobrescritos.**

**Os métodos privados são restritos à classe onde foram definidos, sem possibilidade de herança, protegendo detalhes internos.**

**Na herança, os públicos podem ser reutilizados e personalizados, enquanto os privados permanecem inacessíveis.**

**15- Explique a importância da abstração na programação orientada a objetos, citando exemplos de como uma classe pode representar uma abstração do mundo real.**

**A abstração simplifica modelos do mundo real, destacando características essenciais e ignorando detalhes desnecessários.**

**Uma classe Carro pode ter atributos como marca e modelo e métodos como ligar() ou acelerar(), representando sua essência sem toda a complexidade mecânica. Isso facilita a compreensão, manutenção e reuso do código.**

**16-O que é a biblioteca swing e quais suas características?**

**A biblioteca Swing é um conjunto de componentes gráficos para a construção de interfaces de usuário em Java. Ela faz parte da API Java Foundation Classes (JFC) e foi criada para substituir a AWT (Abstract Window Toolkit), oferecendo mais flexibilidade e funcionalidade.**

**As suas características incluem independência de plataforma, componentes personalizáveis, modelo de eventos aprimorado e suporte a Look-and-Feel. Também utiliza o modelo MVC para organizar a interface e é amplamente usado em aplicações desktop Java.**

**17-O que é JDBC e suas características?**

**JDBC (Java Database Connectivity) é uma API do Java que permite a conexão e interação com bancos de dados. Ele fornece métodos para executar consultas SQL, manipular dados e gerenciar conexões de forma eficiente.**

**As características do JDBC incluem independência de plataforma, suporte a diversos bancos de dados, API robusta para manipulação de consultas SQL e gerenciamento eficiente de conexões. Ele permite comunicação direta entre aplicações Java e bancos de dados, garantindo acesso e manipulação de dados de forma estruturada.**

**18-O que é o JPA?**

**JPA (Java Persistence API) é uma especificação do Java para o mapeamento objeto-relacional (ORM), permitindo que objetos Java sejam armazenados, manipulados e recuperados de bancos de dados.**

**Ele gerencia a persistência dos dados, facilitando operações como inserção, atualização, exclusão e busca. É amplamente usado em frameworks como Hibernate e Spring Data JPA.**

**19-O que é o THROWS?**

**THROWS em Java, é usado na assinatura de um método para indicar que ele pode lançar exceções, exigindo que o código chamador trate possíveis erros. Isso melhora a segurança do programa.**