EXERCÍCIOS DE LINGUAGEM TÉCNICA DE PROGRAMAÇÃO WEB

1. O que é a JVM e qual seu papel na execução de um programa Java?

Resposta: A JVM é a máquina virtual responsável por executar programas Java. Ela interpreta e/ou compila o bytecode para código de máquina específico do sistema operacional, garantindo portabilidade e segurança.

2. Explique a diferença entre JDK, JRE e JVM.

Respostas:

JDK: Pacote completo para desenvolvimento Java, incluindo JRE e ferramentas como o compilador (`javac`).

JRE: Ambiente necessário para rodar programas Java, incluindo JVM e bibliotecas padrão.

JVM: Responsável por executar bytecode, convertendo-o para código de máquina.

3. O que é um construtor em Java? Como ele funciona?

Resposta: Um construtor é um método especial chamado automaticamente ao criar um objeto. Ele inicializa atributos e pode ter sobrecarga (múltiplos construtores com diferentes parâmetros).

4. Qual a diferença entre tipos primitivos e objetos em Java?

Resposta: Tipos primitivos (int, double, boolean etc.) armazenam valores diretamente na memória. Objetos são instâncias de classes e armazenados na heap, com referências apontando para eles.

5. O que é sobrecarga de métodos em Java?

Resposta: A sobrecarga (overloading) ocorre quando métodos têm o mesmo nome, mas diferentes parâmetros (tipo e/ou quantidade).

6. Qual a diferença entre ArrayList e LinkedList?

Resposta: ***ArrayList:*** Usa um array dinâmico, acessando elementos rapidamente, mas inserções e remoções no meio são lentas. ***LinkedList:*** Usa uma lista duplamente encadeada, facilitando inserções e remoções, mas acessos diretos são mais lentos.

7. O que é herança em Java e como funciona?

Resposta: A herança permite que uma classe (`filha`) herde atributos e métodos de outra (`pai`), promovendo reuso de código. Usa-se `extends`.

8. Explique a diferença entre `==` e `.equals()` em Java.

Resposta: `==` compara referências de objetos na memória. ***`.equals()`*** compara valores de objetos (se sobrescrito corretamente).

9. O que é o tratamento de exceções e como ele é feito em Java?

Resposta: O tratamento de exceções previne falhas inesperadas no programa. Usa-se ***`try-catch-finally`*** para capturar e lidar com exceções (`Exception`, ***`RuntimeException`*** etc.).

10. O que são classes abstratas e interfaces em Java? Qual a diferença entre elas?

Resposta: ***Classe abstrata:*** Pode ter métodos concretos e abstratos usa-se `extends`. ***Interface:*** Apenas declara métodos (até Java 8), exigindo implementação. Usa `implements`.

11. O que é encapsulamento e como ele é implementado em Java?

Resposta: Encapsulamento protege atributos/métodos, acessando-os via ***`getters`*** e ***`setters`***. Usa modificadores como ***`private`*** e ***`public`***.

13. Qual a diferença entre sobrecarga e sobrescrita de métodos?

Resposta:

Sobrecarga: Mesmo nome, diferentes parâmetros.

Sobrescrita (overriding): Mesma assinatura na classe filha, alterando o comportamento.

14. O que é polimorfismo e como se manifesta em Java?

Resposta:

Polimorfismo permite que objetos de uma classe derivada sejam tratados como objetos da classe base. Se manifesta na sobrescrita de métodos e no uso de interfaces.

16. O que é associação, composição e agregação em POO?

Resposta:

Associação: Relacionamento entre classes (ex.: `Carro` tem `Motor`).

Composição: Forte dependência; se a classe pai for destruída, a dependente também (`Casa` e `Cômodos`).

Agregação: Dependência mais fraca; objetos podem existir independentemente (`Universidade` e `Aluno`).

17. O que acontece se uma classe implementar duas interfaces que possuem métodos com a mesma assinatura?

Resposta: Não há erro, pois interfaces não têm implementação padrão (até Java 8). A classe deve fornecer a implementação do método.

18. O que é a palavra-chave `super` em Java e quando deve ser usada?

Resposta: ***`super`*** referência a superclasse e é usada para acessar métodos/atributos da classe pai ou invocar seu construtor.

19. Como funciona o conceito de classes aninhadas (`inner classes`) em Java?

Resposta: Classes aninhadas são definidas dentro de outra classe. Tipos:

Inner Class: Precisa de uma instância da classe externa.

Static Inner Class: Pode ser instanciada sem uma instância da classe externa.

Local Inner Class: Definida dentro de um método.

Anonymous Inner Class: Criada sem nome, geralmente para sobrescrever métodos.

20. O que é um método `final` e uma classe `final` em Java?

Resposta: Em Java, quando a gente usa final, significa que algo não pode ser mudado. Mas depende de onde a gente coloca essa palavra.

Se a gente coloca *FINAL* antes de um método, isso quer dizer que ele *não pode ser sobrescrito* por uma classe filha. Ou seja, se outra classe tentar mudar o que ele faz, vai dar erro no console.

Agora, se a gente colocar *FINAL* antes de uma classe, significa que *ninguém pode herdar* dessa classe.

Podemos resumir que, se um método é final, não pode ser sobrescrito. Se uma classe é final, não pode ser herdada.