Questões Teóricas

1. O que é sobrescrita de método (override) em Java? Dê um exemplo simples.

Sobrescrita é quando uma subclasse redefine um método da superclasse mantendo a mesma assinatura (nome, parâmetros e tipo de retorno).

```
class Animal {
    void som() {
        System.out.println("Som Massa");
    }
}

class Cachorro extends Animal {
    @Override
    void som() {
        System.out.println("Latido");
    }
}
```

---

2. Quais são as regras para que um método possa ser sobrescrito corretamente em uma subclasse?

- Mesma assinatura (nome e parâmetros)
- Tipo de retorno igual ou covariante
- O método não pode ser `final`, `static` ou `private`

---

3. Diferencie sobrecarga (Overload) de sobrescrita (Override) de métodos em Java.

- **Override (sobrescrita):** redefinir método da superclasse na subclasse com mesma assinatura.
- **Overload (sobrecarga):** mesmo método, mas com parâmetros diferentes na mesma classe.

---

4. O que é polimorfismo? Dê um exemplo prático em que o polimorfismo é utilizado.

Polimorfismo é a capacidade de um objeto ser tratado como instância da sua superclasse, permitindo métodos que se comportam de formas diferentes conforme a subclasse.

```
Animal Rex = new Cachorro();
Rex.som(); // Saída: Latido
```

---

5. Explique como o polimorfismo de tempo de execução funciona em Java.

No tempo de execução, o método chamado é o da classe real do objeto (subclasse)**,** mesmo que a referência seja do tipo da superclasse (pai)**.** Isso permite comportamento dinâmico.

---

6. É possível invocar métodos específicos da subclasse por meio de uma variável do tipo da superclasse? Justifique.

Não**.** A variável do tipo da superclasse só conhece os métodos definidos nela.

---

7. O que significa um atributo ou método ser estático (estática) em Java?

Significa que o atributo ou método pertence à classe, não a uma instância específica. Pode ser acessado diretamente pela classe.

---

8. Qual a diferença entre atributos/métodos estáticos e de instância em Java? Dê exemplos.

- **Estáticos:** compartilhados por todas as instâncias
- **De instância:** cada objeto tem sua própria cópia

```
class Cont {
    static int total = 0; // atributo estático
    int inst;             // atributo de instância

    Cont() {
        total++;
        inst = total;
    }
}
```

---

9. Por que não é possível acessar um atributo não estático dentro de um método estático?

Porque métodos estáticos não têm acesso ao `this`, que é a referência ao objeto, e não podem acessar atributos que pertencem a instâncias específicas.

---

10. Explique o uso do operador `instanceof` em Java. Quando ele é útil?

`instanceof` verifica se um objeto é instância de uma classe ou interface, evitando erros de conversão (cast).

```
if (obj instanceof Cachorro) {
    Cachorro Rex = (Cachorro) obj;
    Rex.latir();
}
```

---

11. O que acontece se você utilizar `instanceof` com `null`?

O resultado é sempre `false`.

```
String bicho = null;
System.out.println(s instanceof String); // false
```

---

12. O que é uma classe abstrata? Quais as suas características principais?

Classe que não pode ser instanciada diretamente. Pode conter métodos abstratos (sem corpo) e métodos concretos (com implementação). Serve como base para outras classes.

---

13. É possível instanciar uma classe abstrata? Por quê?

Não**.** Porque ela pode conter métodos que não possuem implementação. Ela serve como modelo para subclasses.

---

14. O que é uma interface em Java? Quais as principais diferenças entre interface e classe abstrata?

Uma interface é como um contrato: quem implementa precisa definir os métodos que ela declara.

Aspecto	Interface	Classe Abstrata
Métodos	Apenas declaração (desde Java 8: default/estático)	Pode ter métodos com ou sem código
Atributos	Apenas constantes (públicos, estáticos e finais)	Pode ter atributos comuns
Herança	Pode implementar várias interfaces	Só pode herdar uma classe abstrata
Uso	Garantir que certas ações sejam implementadas	Compartilhar código entre classes relacionadas

---

15. Uma classe pode implementar várias interfaces em Java? Como isso é feito na prática?

Sim. Basta separar os nomes das interfaces com vírgula:

```
class MC implements Interface1, Interface2 {
    // implementação dos métodos das interfaces
```