Interface em Java

Qual a diferença entre Interface e Abstração

Conceito	O que é	Palavra-chave	Pode ter código implementado?
Abstração (classe abstrata)	Um modelo base que pode ter comportamento **parcialmente implementado**. Representa uma **ideia incompleta**, que precisa ser **especializada**.	abstract class	✅ Sim
Interface	Um **contrato** que define **o que** uma classe deve fazer, mas **não como**.	interface	❌ Não

Interface vs Classe Abstrata

Aspecto	Interface	Classe Abstrata
Implementação	Não possui estado (até Java 8, só métodos abstratos)	Pode ter estado e construtores
Herança múltipla	✅ Permite várias interfaces	❌ Apenas uma classe abstrata
Uso	Contratos	Base de comportamento comum
Métodos concretos	Desde Java 8 com default	Sempre possível

O que significa quando dizemos que a interface não tem estado?

Uma **interface não pode armazenar valores específicos de instância**, porque **ela não tem objetos concretos**.

Definição:

Interface é um **contrato** que define o **que** uma classe deve fazer, mas **não como**.

```
public interface Animal {
    void emitirSom();
}
```

VER EXEMPLO 1 NA IDE

🔷 Serve para **abstrair comportamentos comuns**.
🔷 Garante **padronização** entre classes diferentes.
🔷 Permite **polimorfismo** e **baixo acoplamento**.

Analogias:

- Interface = “lista de promessas” que uma classe deve cumprir.
- Exemplo do mundo real: uma “tomada elétrica” → define o formato, mas não o que está por trás.

Primeira Implementação

```
public class Cachorro implements Animal {
    public void emitirSom() {
        System.out.println("Au au!");
    }
}
```

VER EXEMPLO 2 NA IDE

Conceitos demonstrados:

- Implementação de interface (implements).
- **Polimorfismo**: Animal pode referenciar qualquer classe que implemente a interface.

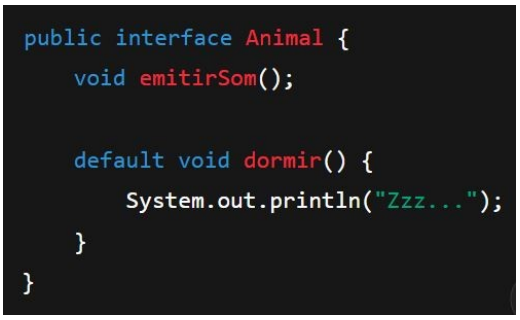
Herança de interface

```
java

interface SerVivo {
    void respirar();
}

interface Animal extends SerVivo {
    void emitirSom();
}
```

VER EXEMPLO 3

Criar interface Servivo e estender pelo Animal

Métodos default



```java
public interface Animal {
    void emitirSom();

    default void dormir() {
        System.out.println("Zzz...");
    }
}
```

Criar o nosso Zoologico usando as interfaces

Ver exemplo 4

Encerramento

- Interfaces garantem **contratos de comportamento**.
- Permitem **polimorfismo** e **baixo acoplamento**.
- São essenciais em **injeção de dependências (Spring)** e **arquiteturas limpas**.