**✅ Interfaces em Java podem ter:**

1. **Métodos abstratos** (sem corpo) → **obrigatórios** de implementar.
2. **Métodos concretos** (com corpo) usando default ou static.
3. **Atributos (variáveis)**, mas com **restrições muito específicas**.

**🔹 Sobre os métodos:**

Por padrão, os métodos em uma interface são:

* **public e abstract** — mesmo que você não escreva essas palavras.

java

CopiarEditar

interface Animal {

void comer(); // é automaticamente public e abstract

}

Desde o Java 8:

* Você pode ter métodos **default** com implementação:

java

CopiarEditar

default void respirar() {

System.out.println("Respirando...");

}

* E métodos **static** com implementação:

java

CopiarEditar

static void info() {

System.out.println("Interface Animal");

}

**🔸 Sobre os atributos:**

Interfaces **podem ter atributos**, MAS:

* Eles são **sempre**:
  + public
  + static
  + final  
    (ou seja, são **constantes**, e não variáveis de instância)

**Exemplo:**

java

CopiarEditar

interface Animal {

int NUM\_PERNAS = 4; // equivalente a: public static final int NUM\_PERNAS = 4;

}

🔒 Você **não pode** declarar atributos mutáveis (sem final) em interfaces.

**⚠️ Comparação com classes:**

| **Elemento** | **Interface** | **Classe abstrata** | **Classe comum** |
| --- | --- | --- | --- |
| Métodos abstratos | ✅ Sim | ✅ Sim | ❌ Não |
| Métodos concretos | ✅ Sim (com default ou static) | ✅ Sim | ✅ Sim |
| Atributos | ✅ Sim, **constantes só** | ✅ Sim (normais e constantes) | ✅ Sim |

**🧠 Resumindo:**

* Interface **tem métodos abstratos por padrão**.
* Pode ter métodos **concretos (default, static)** desde o Java 8.
* Pode ter **atributos**, mas **apenas constantes públicas (public static final)**.
* **Não pode ter atributos mutáveis** nem construtores.

**Qual a diferença de default e static?**

**🟦 default**

* **Pertence à instância da classe que implementa a interface.**
* **Pode ser sobrescrito (override) pela classe implementadora.**
* **Serve para dar uma implementação "padrão" que pode ser reutilizada por várias classes.**

**✅ Exemplo:**

**java**

**CopiarEditar**

**interface Animal {**

**default void respirar() {**

**System.out.println("Respirando...");**

**}**

**}**

**class Gato implements Animal {**

**// Não é obrigado a sobrescrever**

**// Mas pode fazer isso se quiser**

**}**

**Chamando:**

**java**

**CopiarEditar**

**Gato g = new Gato();**

**g.respirar(); // Usa o método default da interface**

**🟨 static**

* **Pertence à interface, não à instância.**
* **Não pode ser sobrescrito.**
* **Só pode ser chamado diretamente pela interface, como Interface.metodo().**

**✅ Exemplo:**

**java**

**CopiarEditar**

**interface Animal {**

**static void info() {**

**System.out.println("Sou um animal");**

**}**

**}**

**class Gato implements Animal {**

**// Não pode sobrescrever info()**

**}**

**Chamando:**

**java**

**CopiarEditar**

**Animal.info(); // Correto**

**// g.info(); ← Errado! Não pode chamar via instância**

**🔍 Diferença resumida:**

| **Característica** | **default** | **static** |
| --- | --- | --- |
| **Pertence a** | **Instância (objeto) da classe** | **À própria interface** |
| **Pode ser sobrescrito?** | **✅ Sim** | **❌ Não** |
| **Pode ser herdado pela classe?** | **✅ Sim** | **❌ Não (não "vai junto")** |
| **Como é chamado?** | **Via instância (obj.metodo())** | **Via interface (Interface.metodo())** |
| **Desde qual versão do Java?** | **Java 8** | **Java 8** |

**🧠 Quando usar cada um?**

* **Use default quando quer fornecer um comportamento padrão que as classes podem herdar ou sobrescrever.**
* **Use static quando quer um método utilitário relacionado à interface, que não depende da instância (como Math.max() na classe Math).**