🧠 Herança + Construtores + Sobrecarga + Encapsulamento

🧩Exercício 1 — Casa Inteligente (Smart Devices)

🎯Objetivo

Treinar os conceitos de **herança** (classe base e subclasses), **encapsulamento** (proteger atributos com private), e **sobrecarga** (usar o mesmo método com diferentes parâmetros).

O aluno vai criar um pequeno sistema que simula dispositivos inteligentes (lâmpadas, ventoinhas, termóstatos) controlados por uma classe "hub".

🧱 O que deves fazer

1️⃣ Criar uma classe **Dispositivo** (classe base):

```
// Atributos
private String nome;
private boolean ligado;

// Metodos
public Dispositivo(String nome);
public void ligar();
public void desligar();
public boolean isLigado();
public String getNome();
```

2️⃣ Criar as subclasses **Lampada**, **Ventoinha**, e **Termostato**:

```
// 💡 Lampada
private int intensidade; // 0–100
public Lampada(String nome);
```

```
public Lampada(String nome, int intensidade);
public void ajustarIntensidade(int valor);
public void ajustarIntensidade(String modo); // "baixo", "medio", "alto"

// 💨 Ventoinha
private int velocidade; // 0–3
public Ventoinha(String nome);
public Ventoinha(String nome, int velocidade);
public void setVelocidade(int valor);
public void setVelocidade(String modo); // "off", "baixo", "medio", "alto"

// 🌡️ Termostato
private double temperatura; // 10.0 – 30.0
public Termostato(String nome);
public Termostato(String nome, double valor);
public void setTemperatura(double valor);
public void setTemperatura(String texto); // ex: "21.5C" ou "70F"
```

3️⃣ Criar uma classe **RoomHub**:

```
public void aplicarCena(String nome);
public void aplicarCena(String nome, int intensidade);
```

🧪 Exemplo de utilização

```
Lampada l = new Lampada("Mesa", 120); // intensidade será 100 (limitada)
Ventoinha v = new Ventoinha("Teto", 2);
Termostato t = new Termostato("Parede");
t.setTemperatura("70F"); // converte para cerca de 21°C

RoomHub hub = new RoomHub(l, v, t);
hub.aplicarCena("Filme", 30);
```

💻 Output esperado

```
[Lâmpada: intensidade=30% | ligada=true]
[Ventoinha: velocidade=1 | ligada=true]
[Termostato: temperatura=21.0°C | ligada=true]
Cena “Filme” aplicada com intensidade 30
```

💰 Exercício 2 — Conta Bancária + Conta Poupança

🎯Objetivo

Compreender como a **herança** permite criar diferentes tipos de contas e como os **construtores sobrecarregados** e **métodos sobrecarregados** ajudam a inicializar e manipular objetos de formas flexíveis.

Também treina **encapsulamento**, impedindo o acesso direto ao saldo.

📥 O que deves fazer

💼 Classe Base: Conta

```
// Atributos
private String titular;
private double saldo;

// Construtores
public Conta(String titular);              // saldo = 0
public Conta(String titular, double valor); // saldo inicial

// Metodos
public void depositar(double valor);
public void depositar(int valor); // sobrecarga
public void levantar(double valor);
public double getSaldo();
public String getTitular();
```

💰 Subclasse: ContaPoupanca

```
// Atributos
private double taxaJuro;

// Construtores
public ContaPoupanca(String titular); // taxa = 0.02
public ContaPoupanca(String titular, double valor, double taxa);

// Metodos
public void renderJuros();       // aplica juro anual
public void renderJuros(int mes); // aplica juro proporcional (sobrecarregado)
```

🧪 Exemplo de utilização

```
ContaPoupanca c = new ContaPoupanca("Ana", 100, 0.03);
c.depositar(50);
c.levantar(30);
c.renderJuros(12);
System.out.println(c.getSaldo());
```

💻 Output esperado

```
Depósito de 50.0€ efetuado com sucesso.
Levantamento de 30.0€ efetuado com sucesso.
Juros aplicados com taxa 3.0%.
Saldo atual: 123.6€
```

🎬 Exercício 3 — Bilhetes de Cinema

🎯Objetivo

Compreender **sobrecarga de métodos** e **herança** criando diferentes formas de calcular o preço de um bilhete com base na idade, estatuto de estudante ou tipo de bilhete.

📥 O que deves fazer

🎟️ Classe Base: Bilhete

```
// Atributos
private double base; // preço base

// Construtores
public Bilhete(double base);

// Metodos
public double preco(int idade);
public double preco(boolean estudante);
public double preco(int idade, boolean estudante);
```

Regras:

- <12 anos → −50%
- ≥65 anos → −30%
- Estudante → −20%
- Preço mínimo: 3.00€

🎓 Subclasse: BilheteEstudante

Adiciona um **desconto fixo extra** de 0.5€ (mas nunca abaixo de 3€).

🧪 Exemplo de utilização

```
Bilhete b = new Bilhete(8.0);
System.out.println(b.preco(10));          // criança
System.out.println(b.preco(true));        // estudante
System.out.println(b.preco(70, true));    // sénior + estudante

BilheteEstudante be = new BilheteEstudante(8.0);
System.out.println(be.preco(20, true));   // desconto extra
```

💻 Output esperado

```
4.0
6.4
4.48
3.5
```