🧪 Lista de Exercícios – Classes Abstratas e Interfaces em Java

🧠 Objetivo:

- Compreender o papel de **classes abstratas** e **interfaces**.
- Praticar o uso de **herança**, **polimorfismo** e **contratos de implementação**.
- Estimular a modelagem orientada a objetos.

📝 Exercício 1 – Classe Abstrata: Animal

Descrição:
Crie uma classe abstrata chamada `Animal` com os seguintes elementos:

- Atributo: `String nome`
- Método abstrato: `void emitirSom()`
- Método concreto: `void dormir()` que imprime "O animal está dormindo"

Crie duas classes que herdem de `Animal`: `Cachorro` e `Gato`. Cada uma deve implementar o método `emitirSom()` com um som característico.

Teste:
No método `main`, crie uma lista de animais e percorra com um `for`, chamando `emitirSom()` e `dormir()`.

📝 Exercício 2 – Interface: Operações Matemáticas

Descrição:
Crie uma interface chamada `OperacaoMatematica` com o método:

```
double calcular(double a, double b);
```

Crie as classes `Soma`, `Subtracao`, `Multiplicacao` e `Divisao` que implementam essa interface.
Implemente o método `calcular` adequadamente em cada classe.

Teste:
No `main`, crie uma instância de cada operação e aplique `calcular(10, 2)`.

📝 Exercício 3 – Classe Abstrata com Interface: Funcionário

Descrição:
Crie uma classe abstrata `Funcionario` com:

- Atributos: `String nome`, `double salarioBase`
- Método abstrato: `double calcularSalario()`

Crie uma interface `Bonificavel` com o método `double calcularBonus()`

Crie duas classes:

- `Gerente` (implementa `Bonificavel` e herda `Funcionario`) → bônus de 20%
- `Programador` (implementa `Bonificavel` e herda `Funcionario`) → bônus de 10%

Teste:
Crie uma lista de funcionários e exiba o nome, salário com bônus e o valor do bônus.

📝 Exercício 4 – Interface com Herança Múltipla

Descrição:
Crie as interfaces:

- `Nadador`: método `void nadar()`
- `Corredor`: método `void correr()`

Crie uma classe `Triatleta` que implemente ambas as interfaces e adicione o método `competir()`, que chama `nadar()` e `correr()`.

Teste:
Instancie um `Triatleta` e chame `competir()`.

📝 Exercício 5 – Polimorfismo com Interface

Descrição:
Crie a interface `Imprimivel` com o método `void imprimir()`.

Crie as classes `Relatorio`, `Contrato` e `Curriculo`, todas implementando `Imprimivel`.

Teste:
Crie uma lista de objetos `Imprimivel` com instâncias das três classes. Use `for` para chamar `imprimir()` para todos.

📝 Exercício 6 – Simulação Bancária com Interface e Abstração

Descrição:
Crie a interface `Transacao` com:

```
void executar();
```

Crie uma classe abstrata `Conta` com:

- Atributos: `numero`, `saldo`
- Métodos: `depositar(double valor)`, `sacar(double valor)` (com verificação de saldo)

Crie duas classes que herdam de `Conta`:

- `ContaCorrente`
- `ContaPoupanca`

Crie a classe `Pagamento` que implementa `Transacao` e realiza um `sacar()` de uma conta.

Teste:
Simule 3 transações com contas diferentes.

📝 Exercício 7 – Jogo com Interface

Descrição:
Crie a interface `Jogavel` com os métodos:

- `void iniciarJogo()`
- `void encerrarJogo()`

Crie duas classes:

- `JogoRPG` e `JogoCorrida` que implementam `Jogavel`.

Teste:
Instancie os dois jogos, chame os métodos, e simule o início e o fim de uma partida.

🎯 Dicas para os Alunos:

- **Use `@Override`** sempre que implementar métodos de interface ou sobrescrever métodos de classe abstrata.
- **Não use `new` em classes abstratas.**
- Organize os arquivos em pacotes (`package`) como `model`, `interface`, `app`, etc.
- Experimente usar **listas polimórficas**, por exemplo, `List<Imprimivel>` ou `List<Animal>`.