Generics em Java

📘 TAREFA DE CASA — GENERICS EM JAVA (EXERCÍCIOS INÉDITOS)

🎯 Objetivo

Avaliar se o aluno **entendeu o conceito de Generics** e consegue **aplicar em problemas novos**, sem copiar modelos prontos.

📝 Exercício 1 — Logger Genérico de Eventos

📌 Descrição

Crie um **Logger genérico** capaz de registrar qualquer tipo de evento do sistema.

🔧 Requisitos

1. Criar a classe genérica:

```
public class Logger<T> {
    public void log(T event) {
        System.out.println("LOG: " + event);
    }
}
```

2. Criar duas classes de domínio:

- `LoginEvent` (usuario, data)

- `PagamentoEvent` (valor, metodo)

3. Demonstrar o uso:

```
Logger<LoginEvent>
Logger<PagamentoEvent>
```

🎯 Avaliação

- Entendimento de classe genérica
- Reutilização sem duplicação
- Clareza do código

📝 Exercício 2 — Serviço Genérico de Processamento

📌 Descrição

Implemente um **serviço genérico** que execute uma operação sobre qualquer tipo.

🔧 Requisitos

1. Criar a interface:

```
public interface Processor<T> {
    void process(T data);
}
```

2. Implementar:

- `EmailProcessor` → imprime envio de e-mail
- `RelatorioProcessor` → imprime geração de relatório

3. Criar um método genérico que receba qualquer `Processor<T>` e execute o processamento.

🎯 Avaliação

- Uso de Generics em interfaces

- Inversão de controle
- Código flexível

📑 Exercício 3 — Repositório Genérico com Validação

📌 Descrição

Crie um **repositório genérico** que só aceite entidades válidas.

🔧 Requisitos

1. Criar a interface:

```
public interface Validavel {
    boolean isValid();
}
```

2. Criar o repositório:

```
public class SecureRepository<T extends Validavel> {
    public void save(T entity) {
        if (!entity.isValid()) {
            throw new IllegalArgumentException("Entidade inválida");
        }
        System.out.println("Salvo com sucesso");
    }
}
```

3. Criar duas entidades que implementem `Validavel`:

- `Usuario`
- `Produto`

🎯 Avaliação

- Uso de bounded types (`extends`)
- Validação em tempo de compilação

- Design orientado a contratos

📝 Exercício 4 — Conversor Genérico Bidirecional

📌 Descrição

Implemente um **conversor genérico de ida e volta** entre dois tipos.

🔧 Requisitos

1. Criar a interface:

```
public interface BiConverter<A, B> {
    B toB(A value);
    A toA(B value);
}
```

2. Implementar:

- `StringIntegerConverter`
- `UserUserDTOConverter`

3. Demonstrar o uso em uma classe `Main`.

🎯 Avaliação

- Uso de múltiplos tipos genéricos
- Clareza na conversão
- Organização do código

📝 Exercício 5 — Serviço Genérico de Auditoria (Wildcard)

📌 Descrição

Crie um serviço que **audite listas de qualquer subtipo** de uma classe base.

🔧 Requisitos

1. Criar a classe base:

```
public abstract class Evento {
    private String descricao;
}
```

2. Criar subclasses:

- `EventoLogin`
- `EventoCompra`

3. Criar o serviço:

```
public class AuditoriaService {
    public void auditar(List<? extends Evento> eventos) {
        eventos.forEach(e ->
            System.out.println("Auditando: " + e)
        );
    }
}
```

🎯 Avaliação

- Uso correto de wildcards
- Polimorfismo + Generics
- Entendimento avançado

📌 Critérios Gerais de Correção

✔ Não repetir código do PDF
✔ Uso correto de `<T>`, `<K, V>` e bounds
✔ Separação em classes
✔ Classe `Main` para testes
✔ Sem casts manuais