Generics em Java

🎯 AULA COMPLETA – GENERICS EM JAVA (1 hora)

Nível: Iniciante → Intermediário

Estilo: Didático, direto, com exemplos reais de projeto

🧩 1. Introdução ao Tema

✅ O que são Generics?

Generics são um recurso do Java que permite criar **classes, interfaces e métodos parametrizados por tipo**.
Ou seja, você pode escrever código que funciona para qualquer tipo, mantendo **segurança de tipo (type-safety)**.

Exemplo simples:

java Copiar código

```java
List<String> nomes = new ArrayList<>();
```

✅ Qual problema os Generics resolvem?

Antes dos Generics, usávamos `List` sem tipo:

java Copiar código

```java
List lista = new ArrayList();
lista.add("Alex");
lista.add(10); // permitido!
```

Isso gerava **erros somente em runtime**, dificultando manutenção.

Generics resolvem isso garantindo que:

- A lista contenha **apenas um tipo**
- Os erros aparecem **em tempo de compilação**

✅ Quando usar

- Ao criar Listas, Maps, Sets
- Ao construir estruturas reutilizáveis
- Ao criar Repositories genéricos
- Ao escrever métodos utilitários flexíveis
- Ao criar abstrações de serviços

✅ Por que usar

- Segurança de tipos
- Reutilização de código
- Menos repetição
- Código mais limpo e mais fácil de manter

📌 Vantagens

- Type-safety
- Reuso
- Abstração mais poderosa
- Evita casts desnecessários
- Ajuda em projetos escaláveis

📌 Desvantagens

- Sintaxe pode parecer complexa para iniciantes
- Inferência limitada em algumas situações
- Pode gerar código mais verboso se mal utilizado

🎯 2. EXEMPLOS PRÁTICOS (10 EXEMPLOS)

Os **primeiros 4** serão sobre **CRUD Genérico** conforme solicitado.

🚀 EXEMPLO 1 — Interface Genérica de CRUD

🎯 Objetivo

Criar a base de um CRUD reutilizável usando Generics.

📌 Código Completo

java Copiar código

```java
public interface CrudRepository<T, ID> {

    void salvar(T entity);

    T buscarPorId(ID id);

    List<T> listarTodos();

    void excluir(ID id);
}
```

📌 Explicação

- `T` = tipo da entidade (Ex: Produto, Cliente, Usuário)
- `ID` = tipo do identificador (Long, UUID, Integer)
- Essa interface pode ser usada em qualquer entidade.

Essa é a base para criar **Repositories reais**, sem repetição de código.

🚀 EXEMPLO 2 — Implementação Genérica (em memória)

🎯 Objetivo

Criar uma implementação reutilizável de um CRUD sem banco de dados, usando `ArrayList`.

📌 Código Completo

java Copiar código

```java
public abstract class InMemoryCrudRepository<T, ID> implements CrudRepository<T

    protected List<T> banco = new ArrayList<>();

    protected abstract ID getId(T entity);

    @Override
    public void salvar(T entity) {
        banco.add(entity);
    }

    @Override
    public T buscarPorId(ID id) {
        return banco.stream()
                .filter(e -> getId(e).equals(id))
                .findFirst()
                .orElse(null);
    }

    @Override
    public List<T> listarTodos() {
```

```
        return banco;
    }

    @Override
    public void excluir(ID id) {
        banco.removeIf(e -> getId(e).equals(id));
    }
}
```

📌 Explicação

- Essa classe é **abstrata** porque cada entidade tem sua própria forma de obter o ID.
- `getId()` é o método que as subclasses devem implementar.
- CRUD completo usando apenas lista em memória.

🚀 EXEMPLO 3 — Criando uma Entidade e Repository Real (Produto)

🎯 Objetivo

Demonstrar como usar o CRUD genérico.

📌 Código Completo

Entidade

java Copiar código

```
public class Produto {
    private Long id;
    private String nome;
    private double preco;

    public Produto(Long id, String nome, double preco) {
        this.id = id;
        this.nome = nome;
        this.preco = preco;
    }
```

```
    public Long getId() { return id; }
    public String getNome() { return nome; }
    public double getPreco() { return preco; }
}
```

Repositório

java Copiar código

```
public class ProdutoRepository extends InMemoryCrudRepository<Produto, Long> {

    @Override
    protected Long getId(Produto entity) {
        return entity.getId();
    }
}
```

📌 Explicação

- Agora você tem um CRUD funcional para a entidade Produto.
- Sem repetir código.
- Sem necessidade de JPA.

🚀 EXEMPLO 4 — Simulando um Service com o Repository Genérico

🎯 Objetivo

Usar o Repository Genérico em um "service" real.

📌 Código Completo

java Copiar código

```
public class ProdutoService {

    private ProdutoRepository repository = new ProdutoRepository();
```

```
    public void cadastrarProduto(Long id, String nome, double preco) {
        Produto p = new Produto(id, nome, preco);
        repository.salvar(p);
    }

    public Produto buscar(Long id) {
        return repository.buscarPorId(id);
    }

    public void excluir(Long id) {
        repository.excluir(id);
    }

    public List<Produto> listar() {
        return repository.listarTodos();
    }
}
```

📌 Explicação

- Essa camada simula uma aplicação real.
- Recebe parâmetros, cria objeto e delega ao repositório.
- Usando Generics, o CRUD é 100% reaproveitado em qualquer entidade.

📌 Até aqui cobrimos:

✔ CRUD Genérico
✔ Interface genérica
✔ Abstração
✔ Repository
✔ Uso em cenário real

Agora vamos para mais 6 exemplos aplicados.

🚀 EXEMPLO 5 — Método Genérico Utilitário

🎯 Objetivo

Criar um método que retorna o último item de qualquer lista.

📌 Código

java Copiar código

```java
public class ListaUtils {

    public static <T> T ultimo(List<T> lista) {
        if (lista.isEmpty()) return null;
        return lista.get(lista.size() - 1);
    }
}
```

📌 Explicação

- `public static <T>` = define T como tipo genérico para o método.
- Funciona para qualquer lista: `List<String>`, `List<Integer>`, etc.

🚀 EXEMPLO 6 — Interface Genérica com Dois Tipos

java Copiar código

```java
public interface Pair<K, V> {
    K getKey();
    V getValue();
}
```

Explicação:

- Muito usado em Maps.
- Permite representar pares (chave/valor).

🚀 EXEMPLO 7 — Classe Genérica de Resposta de API

java Copiar código

```java
public class ApiResponse<T> {

    private boolean sucesso;
    private T dados;

    public ApiResponse(boolean sucesso, T dados) {
        this.sucesso = sucesso;
        this.dados = dados;
    }

    public boolean isSucesso() { return sucesso; }
    public T getDados() { return dados; }
}
```

Explicação:

- Evita duplicação de classes como `UserResponse`, `ProductResponse`, etc.

🚀 EXEMPLO 8 — Filtro Genérico

java Copiar código

```java
public class Filtro {

    public static <T> List<T> filtrar(List<T> lista, Predicate<T> condicao) {
        return lista.stream()
                .filter(condicao)
                .toList();
    }
}
```

Explicação:

- Permite filtrar qualquer tipo.
- Altamente reutilizável.

🚀 EXEMPLO 9 — Ordenação Genérica

java Copiar código

```java
public class Ordenador {

    public static <T extends Comparable<T>> List<T> ordenar(List<T> lista) {
        Collections.sort(lista);
        return lista;
    }
}
```

Explicação:

- `T extends Comparable<T>` garante que só entra tipo comparável.

🚀 EXEMPLO 10 — Builder Genérico para Objetos

java Copiar código

```java
public class ObjectBuilder<T> {

    private T instancia;

    public ObjectBuilder(T instancia) {
        this.instancia = instancia;
    }

    public T build() {
        return instancia;
    }
}
```

Explicação:

- Pode ser usado para criar objetos dinamicamente.
- Muito usado em testes automatizados.

🏁 3. Conclusão da Aula

Nesta aula você aprendeu:

✅ O que são generics
✅ Por que existem
✅ Quando usar

✅ Interfaces genéricas
✅ Métodos genéricos
✅ Abstração com classes genéricas
✅ Um CRUD COMPLETO e genérico
✅ Aplicações reais em projetos profissionais

📝 4. Tarefa de Casa

📌 Exercícios Práticos

1. Crie uma entidade `Cliente` e implemente um `ClienteRepository` usando o CRUD Genérico.
2. Implemente um método genérico chamado `primeiroElemento(List)`
3. Crie uma interface genérica de Paginação `<T>` com métodos
   - `List<T> paginar(int pagina, int tamanhoPagina)`
4. Crie uma classe `ResponseList<T>` contendo:
   - lista
   - total
   - página
   - tamanho

🚀 DESAFIO FINAL

Crie um CRUD COMPLETO de Funcionário usando os conceitos da aula:

- Interface genérica
- Repository genérico
- Implementação em memória
- Service
- Métodos de busca por nome
- Listagem com filtro

Simulando um sistema empresarial real.

FIM