# **2 - FlatMap**

O método flatMap é uma operação intermediária que é usada para transformar um Stream de coleções em um Stream de elementos. Aqui está um exemplo de como o flatMap é usado:

List<List<String>> list = **List.of**(

List.of("a", "b"),

List.of("c", "d")

);

Stream<String> stream = list.stream()

.flatMap(Collection::stream);

stream.forEach(System.out::println);

 **Imutabilidade: A lista retornada por List.of é imutável, o que significa que não é possível adicionar, remover ou modificar elementos após a criação.**

 **Tipo de Elemento**: List.of pode ser usado com qualquer tipo de elemento, incluindo tipos primitivos (como int, double, etc.) e objetos.

 **Tratamento de Elementos Nulos**: List.of não aceita elementos nulos. Se algum dos elementos passados for null, será lançada uma exceção NullPointerException.

Cria uma lista  
List<List<String>> list = **List.of**(

Que não pode ser alterada e não pode receber nulos.

Aqui estamos criando duas listas.

List<List<String>> list = **List.of**(

Passando para cada lista esses dados

Entao lista1 vai conter A, B

Lista 2 vai conter C D

List.of("a", "b"),

List.of("c", "d")

Agora vamos pegar os dados que recebemos e vamos usar o flatMap para transformar uma coleção de lista em coleção de elementos, no caso estamos transformando em uma coleção de string

Stream<String> stream = list.stream()

.flatMap(Collection::stream);

stream.forEach(System.out::println);

Neste exemplo, transformamos um Stream de List para um Stream de Strings.

# **3 - Redução de Streams**

Stream.reduce() é uma operação terminal que é utilizada para reduzir o conteúdo de um Stream para um único valor.

Criando uma lista que não; Adicionar, alterar ou remover

List<Integer> numbers = List.of(1, 2, 3, 4, 5);

// Passando essa lista para dentro de um Optional result

Optional<Integer> result = numbers.stream()

// Reduzindo essa lista para apenas 1 valor.

// E somando todos eles.

.reduce(Integer::sum);

result.ifPresent(System.out::println); //prints 15

No exemplo acima, somamos todos os números da lista usando o método reduce().

Lembre-se que o Java Streams é uma ferramenta poderosa que pode tornar seu código mais elegante e eficiente. Continue praticando e explorando todas as diferentes operações e métodos disponíveis para você com Java Streams para se tornar mais hábil em lidar com dados em suas aplicações.