

Default methods

Default methods são métodos de **interface** que possuem corpo. Eles são desenhados dessa maneira pra garantir que um recurso seja implementado sem quebrar implementações já existentes.

Um bom exemplo disso é o próprio método **forEach**, a gente o chama e usa normalmente, mas ele vem de uma Interface (**Iterable**) e a gente não precisou implementar ele, pois o **código já existe** na **própria Interface** como um **default method**:

```
public interface Iterable<T> {  
    Returns an iterator over elements of type T.  
    Returns: an Iterator.  
    @NotNull  
    Iterator<T> iterator();  
  
    default void forEach(Consumer<? super T> action) {  
        Objects.requireNonNull(action);  
        for (T t : this) {  
            action.accept(t);  
        }  
    }  
}
```

Interfaces funcionais só podem ter **UM único método abstrato**, mas elas podem ter **N métodos default**. A própria Interface funcional **Consumer** tem um método default, o **andThen()**:

```
@FunctionalInterface  
public interface Consumer<T> {  
    Performs this operation on the given argument.  
    Params: t – the input argument  
    void accept(T t);  
  
    @Contract(pure = true) @NotNull  
    default Consumer<T> andThen( @NotNull Consumer<? super T> after) {  
        Objects.requireNonNull(after);  
        return (T t) -> { accept(t); after.accept(t); };  
    }  
}
```

Esse método é uma composição de um **Consumer**, então ele **pode ser usado para executar em sequência vários consumidores** (é como se tivéssemos encadeando-os):

```
Consumer<Usuario> imprimeNome = u -> System.out.println(u.getNome());  
Consumer<Usuario> mostraPonto = u -> System.out.println(u.getPontos());  
  
usuarioList.forEach(imprimeNome.andThen(mostraPonto));
```

Predicatos

As collections também ganharam um **método default chamado de removeIf()**, o grande lance aqui é o **Predicate** na verdade, que é uma **interface funcional** que **faz validações**, então você pode criar um predicato e utilizar ele para executar remoção de coleções:

```
Predicate<Usuario> predicadoPontosMaiorQue = u -> u.getPontos() > 83;  
usuarioList.removeIf(predicadoPontosMaiorQue);
```