

 06

## Outros métodos de Collections.

Além do método `sort()` que vimos neste capítulo, a classe `Collections` também possui muitos outros métodos interessantes. Vamos dar uma olhada em alguns:

### `Collections.reverse()`

O método `reverse()` serve para inverter a ordem de uma lista. As vezes precisamos imprimir uma lista de nomes do último para o primeiro, ou uma lista de `ids` do maior para o menor e é nestas horas que utilizamos o `reverse` para inverter a ordem natural da lista para nós.

### `Collections.shuffle()`

O método `shuffle()` serve para embaralhar a ordem de uma lista. Por exemplo em um caso de um sistema de sorteio, em que precisamos de uma ordem aleatória na nossa lista, utilizamos o método `shuffle` para embaralhá-la.

### `Collections.singletonList()`

O método `singletonList()` nos devolve uma lista imutável que contém um único elemento especificado. Ele é útil quando precisamos passar um único elemento para uma API que só aceita uma `Collections` daquele elemento.

### `Collections.nCopies()`

O método `nCopies()` nos retorna uma lista imutável com a quantidade escolhida de um determinado elemento. Se temos uma lista específica e precisamos obter uma outra lista imutável, contendo diversas cópias de um destes objetos, utilizamos o método `nCopies()`. O bom deste método é que mesmo que nós solicitemos uma lista com um número grande, como 10000 objetos, ele na verdade se referencia a apenas um, ocupando assim um pequeno espaço.

Este método também é utilizado para inicializar Listas recém criadas com Null, já que ele pode rapidamente criar diversos objetos, deste modo:

```
List<Type> list = new ArrayList<Type>(Collections.nCopies(1000, (Type)null));
```

Estes são apenas alguns exemplos dos diversos métodos da classe `Collections`.