10

Para saber mais: Arrays.asList

Frequentemente surgem dúvidas quanto a diferença entre o `Arrays.asList(..)` e o `List.of(..)`.

Para ficar mais claro, análise e execute o código abaixo:

```
List<String> asList = Arrays.asList("nome1", "nome2");
asList.add("nome3"); // java.lang.UnsupportedOperationException
asList.set(1, "nome3"); // Ok
System.out.println(asList);

List<String> ofList = List.of("nome1", "nome2");
ofList.add("nome3"); // java.lang.UnsupportedOperationException
ofList.set(3, "nome3"); // java.lang.UnsupportedOperationException
System.out.println(ofList);
```

Repare que ambos os métodos `add` das listas geram uma exceção. No entanto, apenas o método `set` da segunda lista também gera uma exceção. Vamos às motivações das exceções acima:

No caso do `asList.add(...)`, a exceção ocorre porque no momento em que declaramos `Arrays.asList("nome1", "nome2")`, estamos informado que esse *array* tem duas posições fixas. Como já populamos com "nome1" e "nome2", não há como adicionar um terceiro nome, ocorrendo a exceção. O `set` irá funcionar, porque estamos querendo substituir o valor que está na segunda posição (índice 1) pelo valor "nome3". Então, ao executar essa ação, o valor "nome2" será substituído pelo "nome3". Isso mostra que o objeto pode ser alterado, ou seja, é um objeto **mutável**.

Já no caso do `List.of(..)`, a motivação em ambos os casos deve-se ao fato da mesma retornar **uma lista imutável** e um objeto imutável não pode sofrer alterações.

Portanto, objetos com `Arrays.asList(..)` trabalham com objetos com tamanhos fixos e mutáveis, enquanto o `List.of(...)` trabalha com objetos sem tamanho definidos e imutáveis.