02

Declare e use uma ArrayList

Nesta prova, veremos somente a `ArrayList`, uma lista que usa internamente um array. Rápida no método `get`, pois sua estrutura interna permite acesso aleatório (*random access*) em tempo constante.

Jamais se esqueça de importar a `ArrayList`:

```
import java.util.ArrayList;
```

COPIAR CÓDIGO

O primeiro passo é criar uma `ArrayList` vazia de `String`s:

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

COPIAR CÓDIGO

A `ArrayList` herda diversos métodos abstratos e concretos e veremos vários deles aqui, dentre esses, os principais para a certificação, vindos da interface `Collection`.

Por exemplo, para adicionar itens, fazemos:

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

```
nomes.add("certificação");
nomes.add("java");
```

COPIAR CÓDIGO

Para remover e verificar a existência do mesmo na lista:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
nomes.add("java");

System.out.println(nomes.contains("java")); // true
System.out.println(nomes.contains("c#")); // false

// true, encontrado e removido
boolean removido = nomes.remove("java");

System.out.println(nomes.contains("java")); // false
System.out.println(nomes.contains("c#")); // false
```

COPIAR CÓDIGO

Note que o `remove` remove somente a primeira ocorrência daquele objeto.

Podemos também verificar o tamanho de nossa `ArrayList` :

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
nomes.add("java");
System.out.println(nomes.size()); // imprime 2
```

COPIAR CÓDIGO

E convertê-la para um array:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
nomes.add("java");

Object[] nomesComoString = nomes.toArray();
```

COPIAR CÓDIGO

Caso desejarmos um array de String, devemos indicar isso ao método `toArray` de duas formas diferentes:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
nomes.add("java");

String[] nomes2 = nomes.toArray(new String[0]);
String[] nomes3 = nomes.toArray(new String[nomes.size()]);
```

COPIAR CÓDIGO

Ambas passam um array de String: o primeiro menor e o segundo com o tamanho suficiente para os elementos. Se ele possui o tamanho suficiente, ele mesmo será usado, enquanto que, se o tamanho não é suficiente, o `toArray` cria um novo array do mesmo tipo.

Além disso, podemos adicionar uma coleção inteira em outra:

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

```
nomes.add("certificação");
nomes.add("java");

ArrayList<String> paises = new ArrayList<String>();
paises.add("coreia");
paises.add("brasil");

ArrayList<String> tudo = new ArrayList<String>();
tudo.addAll(nomes);
tudo.addAll(paises);
System.out.println(tudo.size()); // imprime 4
```

COPIAR CÓDIGO

Outros métodos são específicos da interface `List` e recebem uma posição específica onde você quer colocar ou remover algo do array usado na `ArrayList`. O método `get` devolve o elemento na posição desejada, lembrando que começamos sempre com 0:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
System.out.println(nomes.get(0)); // imprime certificação
```

COPIAR CÓDIGO

Já o método `add` foi sobrecarregado para receber a posição de inclusão:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("certificação");
System.out.println(nomes.get(0)); // imprime certificação

nomes.add(0, "java");
```

```
System.out.println(nomes.get(0)); // imprime java
System.out.println(nomes.get(1)); // imprime certificação
```

COPIAR CÓDIGO

O mesmo acontece para o método `remove`:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("java");
nomes.add("certificação");

String removido = nomes.remove(0); // retorna java
System.out.println(nomes.get(0)); // imprime certificação
```

COPIAR CÓDIGO

E o método `set`, que serve para alterar o elemento em determinada posição:

```
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("java");
nomes.set(0, "certificação");

System.out.println(nomes.get(0)); // imprime certificação
System.out.println(nomes.size()); // imprime 1
```

COPIAR CÓDIGO

Os métodos `indexOf` e `lastIndexOf` retornam a primeira ou a última posição que possui o elemento desejado. Caso esse elemento não esteja na lista, ele retorna -1:

```java
ArrayList<String> nomes = new ArrayList<String>();
nomes.add("guilherme");
nomes.add("mario");
nomes.add("paulo");
nomes.add("mauricio");
nomes.add("adriano");
nomes.add("alberto");
nomes.add("mario");

System.out.println(nomes.indexOf("guilherme")); // 0
System.out.println(nomes.indexOf("mario")); // 1
System.out.println(nomes.indexOf("joao")); // -1
System.out.println(nomes.lastIndexOf("mario")); // 6
System.out.println(nomes.lastIndexOf("joao")); // -1
```

COPIAR CÓDIGO