Introdução

É uma tarefa muito comum ter que lidar com coleções de dados quando criamos um programa. Por isso o Java oferece uma série de classes que facilitam a criação de coleções. Vamos conhecer algumas delas?

Arrays

Arrays são as **estruturas de dados mais simples** usadas para **criar coleções de dados de um mesmo tipo**.
Nós podemos declarar o tipo dos dados da coleção e o tamanho que ela terá:

```
String nomes[] = new String[3];
nomes[0] = "Lucas";
nomes[1] = "Paulo";
nomes[2] = "Maria";
```

No exemplo anterior foi criado um **Array** de **tipo String** chamado "**nomes**", que conterá 3 elementos. Note que para adicionarmos itens ao Array "nomes" nós precisamos informar o índice da posição entre colchetes.

O primeiro elemento sempre estará posicionado na posição 0 e não é possível adicionar itens em índices fora do limite definido na inicialização do Array.

Características de Array

- Seu tamanho é fixo;
- Operações como ordenação, busca e iteração devem ser implementadas pelo programador;
- Requer menos menos memória e possui melhor performance na obtenção dos dados.

```
nomes[3] = "Joao";
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of
bounds for length 3
      at Main.main(Main.java:8)
```

No exemplo anterior, ao tentar adicionar um item no índice 3 uma exceção será lançada, informando que o índice está fora do limite.

Uma vez que o Array é inicializado seu tamanho não pode ser modificado.

Na inicialização de um Array não é necessário informar seu tamanho, desde que os itens sejam definidos na própria inicialização:

```
String nomes[] = {"Lucas", "Paulo", "Maria"};
```

Nesse caso, o Java irá inferir que o Array terá a quantidade de elementos que foi informada na inicialização.

Para acessar um item em um Array, basta informar a índice entre colchetes:

```
String nomes[] = {"Lucas", "Paulo", "Maria"};
System.out.println("Olá, " + nomes[1]); // Olá,
Paulo
```

Percorrendo Arrays

Para percorrer itens de um Array, basta usar um laço "for":

```
String nomes[] = {"Lucas", "Paulo", "Maria"};

for(int i = 0; i < nomes.length; i++) {
   System.out.println("Olá, " + nomes[i]);
}
```

É necessário limitar o laço ao tamanho do Array para evitar a exceção "ArrayIndexOutOfBoundsException";

Podemos percorrer itens do Array usando um "for" otimizado:

```
for (String nome : nomes) {
   System.out.println("Olá, " + nome);
}
```

ArrayList

ArrayList é um framework (conjunto de funcionalidades já prontas) construído em cima dos Arrays. É uma classe que oferece um conjunto de métodos que facilitam ainda mais a manipulação de coleções de dados:

```
ArrayList<String> emails =new ArrayList<>();
emails.add("lucas@gmail.com");
emails.add("paulo@gmail.com");
emails.add("maria@gmail.com");
```

Para criarmos um ArrayList também informamos o tipo dos dados e inicializamos um objeto de "ArrayList". Para adicionar elementos, usamos o método "add".

Características de ArrayList

- O tamanho do ArrayList é dinâmico e pode crescer à medida em que os dados são adicionados
- Requer mais memória e é menos performático do que Arrays
- Operações como busca, ordenação e outras já são implementadas pela classe ArrayList

```
ArrayList<String> emails =  new ArrayList<>();
emails.add("lucas@gmail.com");
emails.add("paulo@gmail.com");
emails.add("maria@gmail.com");

if (emails.contains("paulo@gmail.com")) {
   System.out.println("Email já cadastrado");
} else {
   System.out.println("Email não existe");
}
```

No exemplo acima usamos o método "contains" para verificar se um email está dentro do ArrayList "emails". A classe ArrayList implementa outros métodos úteis como "isEmpty", "equals", "remove", "sort", etc...

Para aprender mais sobre ArrayList, esse site é uma ótima referência: https://www.w3schools.com/java/java_arraylist.asp

HashMap

HashMap é uma estrutura de dados que usamos para **armazenar** e **gerenciar conjunto de chaves e valores**. Podemos pensar num cenário em que precisamos armazenar usuários e identificá-los pelos seus CPFs.

Nesse caso usamos um HashMap e definimos que o tipo do CPF será String e o tipo do usuário, uma String também:

```
HashMap<String, String> usuarios =new HashMap<>();
```

O primeiro (do CPF) tipo será o tipo da chave e o segundo (do nome do usuário), o valor.
O HashMap é análogo à tabelas, como essa:

CPF	Nome do Usuário

Para adicionarmos itens ao HashMap "usuarios", podemos usar o método "put":

```
usuarios.put("111.111.111-11", "Lucas Santos");
usuarios.put("222.222.222-11", "Paulo Correia");
```

CPF	Nome do Usuário
`111.111.111-11`	`Lucas Santos`
`222.222.222-11`	`Paulo Correia`

Para obter o nome do usuário, basta informar sua chave (o CPF no caso) no método "get":

```
String nome = usuarios.get("111.111.111-11");
System.out.println(nome);
```

Se for passado uma chave que não existe, será retornado "null":

```
String nome = usuarios.get("333.333.333-33");
System.out.println(nome); // null
```