ArrayList

A classe ArrayList fornece uma implementação de fácil uso para uma lista de dados. Ao contrário do que ocorre com um vetor (Array convencional) com um ArrayList temos a flexibilidade de aumentar ou diminuir o tamanho da lista (quantidade de dados).

ArrayList x Array

No código abaixo é possível ver um paralelo entre a criação de um ArrayList e um Vetor:

```
String[] nomesComoVetor = new String[10];
ArrayList<String> nomeComArrayList = new ArrayList<String>();
```

Note que ao criar um vetor convencional precisamos definir a quantidade de elementos que ele vai possuir, mesmo que nem todas as posições estejam preenchidas.

Já na criação de um ArrayList apenas definimos qual é o tipo de dado que vamos armazenas.

Além desta diferença um objeto criado da classe ArrayList fornece uma série de métodos que podemos utilizar para manipular os dados dentro da lista.

Criação

Para usar e criar um objeto da classe ArrayList precisamos importá-la do pacote java.util. Podemos fazer da seguinte forma:

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

Depois de importado podemos instanciar (criar) um objeto desta classe.

A classe ArrayList permite que usemos qualquer tipo de dado que precisarmos, isso acontece devido ao uso de generics (são estes símbolos <> após o nome da classe). Desta forma quando criamos a variável e instanciamos o objetos precisamos definir qual é o tipo de dado que vai ser usado ali:

```
ArrayList<String> listaDeStrings = new ArrayList<String>();
ArrayList<Integer> listaDeInteiros= new ArrayList<Integer>();
ArrayList<Double> listaDeNumerosReais = new ArrayList<Double>();
ArrayList<Animal> listaDeClassesCriadas = new ArrayList<Animal>();
```

Note que podemos usar como Tipo as classes que criamos para nosso projeto.

Métodos principais

Para inserir, remover, buscar e fazer as diversas operações de um array precisamos utilizar os métodos que a classe fornece, abaixo listo alguns métodos úteis para utilização:

- Adicionar Elementos

Para adicionar elementos a um ArrayList usamos o método add(). Ele recebe por parâmetro o tipo do dado que você definiu na instanciação do objeto ArrayList.

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNome.add("Marcos");

ArrayList<Integer> listaDeNumeros = new ArrayList<Integer>();
Integer numero = 10;
listaDeNumeros.add(numero);
```

- Acessar elementos

Para acessar os elementos da mesma forma que fazemos usando o Vetor, ou seja, pelo índice, utilizamos o método get. Ele recebe como parâmetro o índice do elemento que queremos acessar.

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");
```

```
System.out.println(listaDeNomes.get(0));
```

- Obter o índice de um elemento

Para descobrir qual o índice de um elemento específico precisamos utilizar o método indexOf. Ele recebe como parâmetro um objeto do mesmo tipo definido na instanciação da lista. Internamente ele fará uso do método equals do objeto para comparar e identificar qual estamos buscando.

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

System.out.println(listaDeNomes.indexOf("Marcos"));
```

- Remover elementos

Para remover um elemento de dentro da lista podemos usar o método remove(). Ele tem duas implementações: Uma que aceita o índice do elemento buscado e outra que recebe o próprio elemento, assim como no método indexOf.

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

listaDeNomes.remove("Marcos");
listaDeNomes.remove(0);
```

- Obter o tamanho da lista

Ao contrário do Vetor que possui um atributo para obtermos o tamanho no ArrayList temos o método size() que devolve a quantidade de elementos que estão inseridos na lista.

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

listaDeNomes.size();
```

- Verificar se a lista está vazia

O método isEmpty() pode ser usado para verificar se existe ou não algum elemento inserido na lista. Ele retorna um booleano (verdadeiro ou falso).

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

System.out.println(listaDeNomes.isEmpty() ? "Vazio" : "Possui elementos");
```

Percorrer por todos elementos

Para percorrer por todos elementos temos duas formas:

1. Fazer um for assim como com vetores

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

for(int i = 0; i < listaDeNomes.size(); i++){
  System.out.prinln(listaDeNomes.get(i));
}
```

2. Fazer um for simplificado

```
ArrayList<String> listaDeNomes = new ArrayList<String>();
listaDeNomes.add("Marcos");

for(String nome : listaDeNomes){
  System.out.println(nome);
}
```

Exemplo Prático de uso

```
import java.util.ArrayList;

public class Principal {
  public static void main(String[] args) {
    ArrayList<String> nomes = new ArrayList<String>();

        nomes.add("João");
        nomes.add("Maria");
        nomes.add("Pedro");

        System.out.println("Tamanho da lista: " + nomes.size());

        System.out.println("Primeiro nome: " + nomes.get(0));

        nomes.remove("Maria");

        System.out.println("Tamanho atualizado: " + nomes.size());
        System.out.println("Lista após a remoção: " + nomes);
  }
}
```