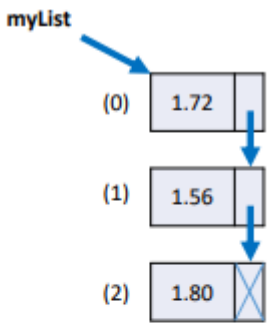
Dia 15 - Listas

Listas

Nessa aula falaremos sobre listas;

Mas antes de falar sobre listas é importante sempre ressaltar que Lista é uma interface e o ArrayList é a classe que a implementa, entao não devemos confundir um com o outro.

Lista é uma estrutura de dados:



• Homogênea (dados do mesmo tipo)

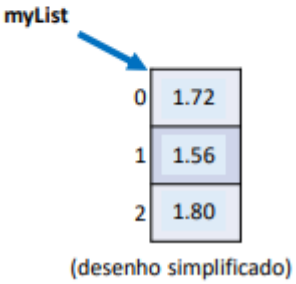
• Ordenada (elementos acessados por meio de posições)

• Inicia vazia, e seus elementos são alocados sob demanda

• Cada elemento ocupa um "nó" (ou nodo) da lista

• Tipo (interface): List

• Classes que implementam: ArrayList, LinkedList etc.



(desenho simplificado)

• Vantagens:

• Tamanho variável

• Facilidade para se realizar inserções e deleções

• Desvantagens:

• Acesso sequencial aos elementos *

• Tamanho da lista: size()

• Obter o elemento de uma posição: get(position)

• Inserir elemento na lista: add(obj), add(int, obj)

• Remover elementos da lista: remove(obj), remove(int), removeIf(Predicate)

• Encontrar posição de elemento: indexOf(obj), lastIndexOf(obj)

• Filtrar lista com base em predicado:

```
List result = list.stream().filter(x -> x > 4).collect(Collectors.toList());
```

• Encontrar primeira ocorrência com base em predicado:

```
Integer result = list.stream().filter(x -> x > 4).findFirst().orElse(null);
```

Assuntos pendentes:

Dia 15 - Listas

• interfaces

• generics

• predicados (lambda)

Exemplo de lista:

```
package application;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class Program {

public static void main(String[] args) {

	List<String> list = new ArrayList<>();

	list.add("Maria");
	list.add("Alex");
	list.add("Bob");
	list.add("Anna");
	list.add(2, "Marco");

	System.out.println(list.size());
	for (String x : list) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	list.removeIf(x -> x.charAt(0) == 'M');
	for (String x : list) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	System.out.println("Index of Bob: " + list.indexOf("Bob"));
	System.out.println("Index of Marco: " + list.indexOf("Marco"));
	System.out.println("---------------------");
	List<String> result = list.stream().filter(x -> x.charAt(0) == 'A').collect(Collectors.toList());
	for (String x : result) {
		System.out.println(x);
	}
	System.out.println("---------------------");
	String name = list.stream().filter(x -> x.charAt(0) == 'J').findFirst().orElse(null);
	System.out.println(name);
	}
}
```

Listas possuem as principais subclasses:

vetores – ideal para acesso randômico. Sincronizado.

ArrayList – ideal para acesso randômico. Não sincronizada.

LinkedList – ideal para acesso sequencial. Não sincronizada.

Principais métodos adicionais

void add(int index, Object o): adiciona objeto na posição indicada (empurra elementos existentes para a frente)

Object get(int index): recupera objeto pelo índice

Dia 15 - Listas

int indexOf(Object o): procura objeto e retorna índice da primeira ocorrência

Object set(int index, Object o): grava objeto na posição indicada (apaga qualquer outro que ocupava a posição).

Object remove(int index)

ListIterator listIterator(): retorna uma ListIterator

A implementação mais utilizada da interface List é ArrayList.

ArrayList é ideal para pesquisa LinkedList é ideal para inserção e remoção de itens nas pontas. A partir do Java 5 podemos usar o recurso de Generics para restringir as listas a um determinado tipo de objetos (e não qualquer Object):

```
List<ContaCorrente> contas = new ArrayList<ContaCorrente>();

ContaCorrente c1, c2, c3;

contas.add(c1);
contas.add(c3);
contas.add(c2);
```

O uso de Generics também elimina a necessidade de casting, já que seguramente todos os objetos inseridos na lista serão do tipo ContaCorrente:

```
for(int i = 0; i < contas.size(); i++) {

   ContaCorrente cc = contas.get(i); // sem casting!
   System.out.println(cc.getSaldo());

}
```