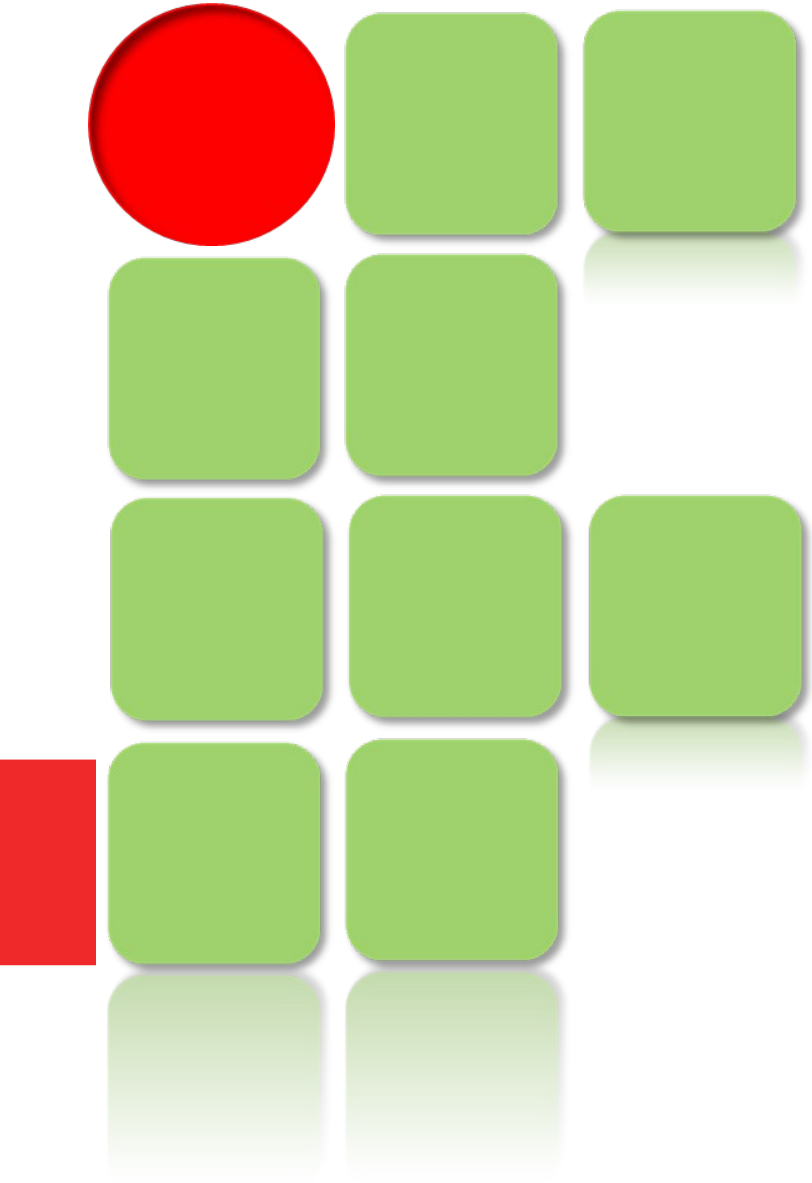
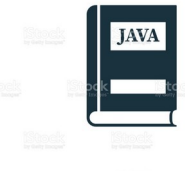


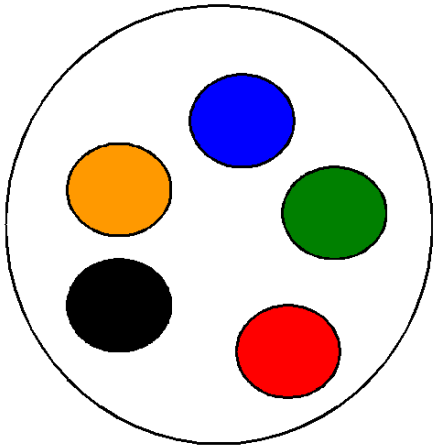
INSTITUTO FEDERAL

Prof. Dr. Bruno Queiroz Pinto

Conceito geral de coleções



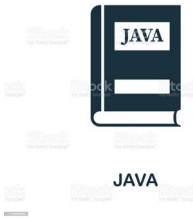
- As operações que podem ser feitas em coleções variam, mas normalmente incluem:
  - Adição de elementos;
  - Remoção de elementos;
  - Acesso aos elementos;
  - Pesquisa de elementos.



Possíveis ações em um conjunto:

- A camiseta Azul está no conjunto?
- Remova a camiseta Azul.
- Adicione a camiseta Vermelha.
- Limpe o conjunto.

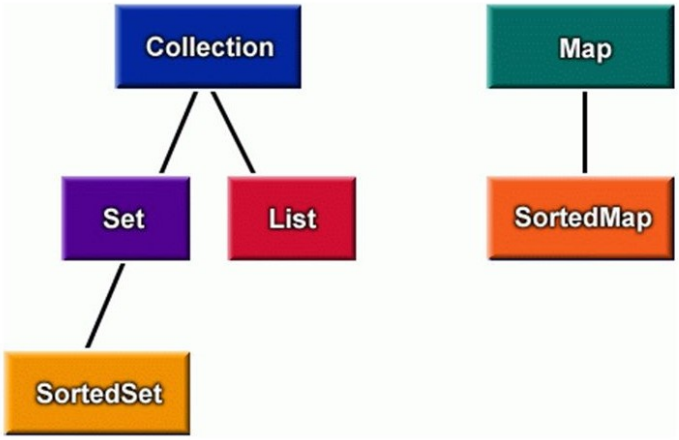
Tipo de coleções



- Dependendo da forma de fazer as 4 operações básicas (adição, remoção, acesso e pesquisa), teremos vários tipos de coleções
  - Certas operações poderão ter um desempenho melhor ou pior dependendo do tipo de coleção;
  - Certas operações poderão ter restrições ou funcionalidade especial dependendo do tipo de coleção.

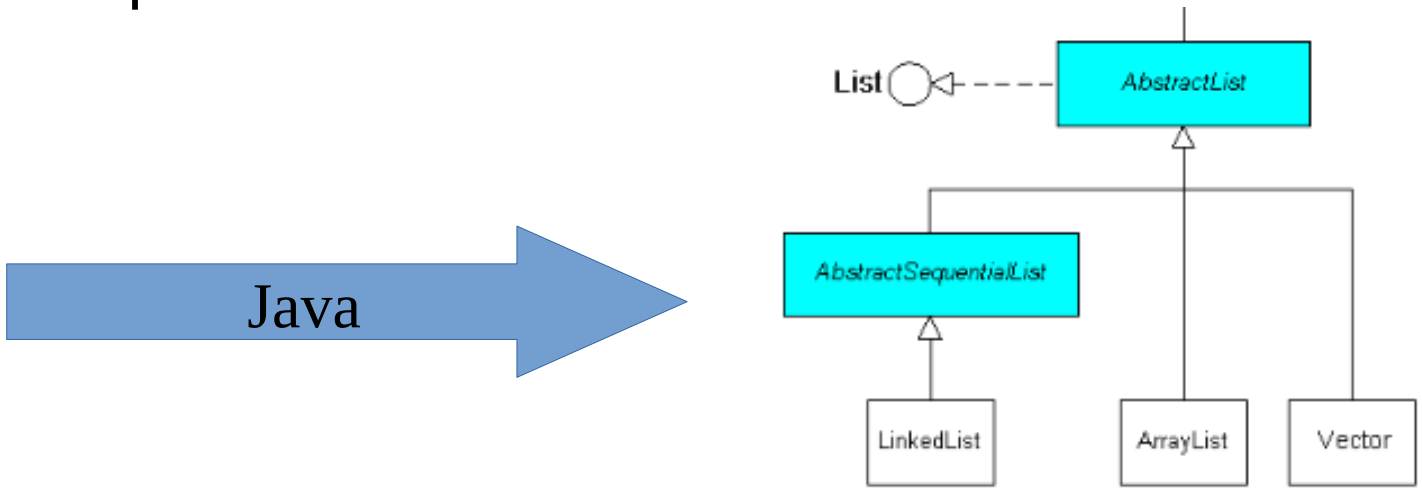
Tipo de coleções

- Os três grandes tipos de coleções são:
  - A lista
    - Também chamado "Sequência"
  - O conjunto
  - O mapa
    - Também chamado "Dicionário"

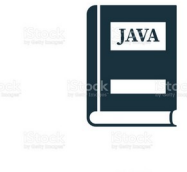


Listas

- Uma lista é uma coleção de elementos arrumados numa ordem linear, isto é, onde cada elemento tem um antecessor (exceto o primeiro) e um sucessor (exceto o último).
- Normalmente implementada como "Array" ou "Lista Encadeada".
- A Lista pode ser mantida ordenada ou não.



Resumo



➔Listas: Problemas de Vetores(Arrays)

➔Listas: java.util.List

➔ArrayList

Problemas de Vetores (Arrays)

- não podemos redimensionar um array em Java;

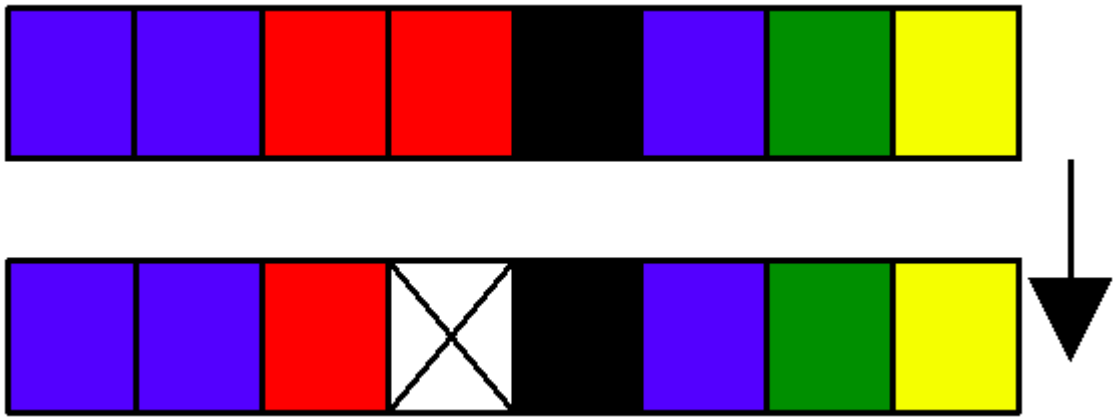
```
int v[] = new int[10];
   v[20] = 10;
```

- é impossível buscar diretamente por um determinado elemento cujo índice não se sabe;
  - ✗ Precisa fazer uma busca até encontrar.

- não conseguimos saber quantas posições do array já foram populadas sem criar, para isso, métodos auxiliares.

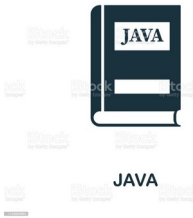
```
int v[] = new int[10];
 v[0] = 10;
 v[8] = 8;
 //:
```

Problemas de Vetores (Arrays)



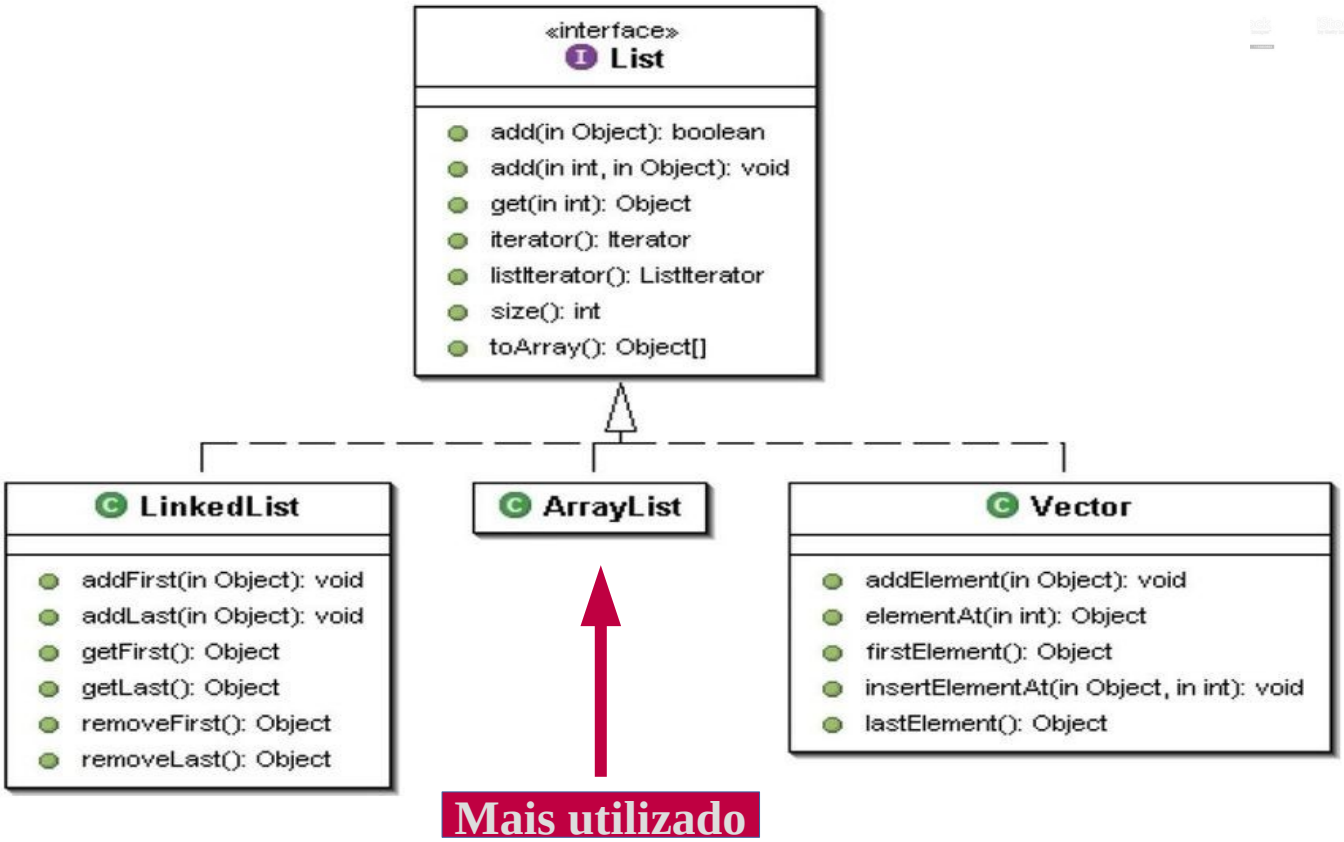
- Vetor completamente utilizado que teve um de seus elementos removidos.
- O que acontece quando precisarmos inserir um novo elemento?
  - Precisaremos procurar por um espaço vazio? Guardaremos em alguma estrutura de dados externa, as posições vazias?
  - E se não houver espaço vazio?
  - Teríamos de criar um array maior e copiar os dados do antigo para ele?

Problemas de Vetores (Arrays)



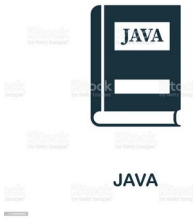
- Além dessas dificuldades que os vetores apresentavam, faltava um conjunto robusto de classes para suprir a necessidade de estruturas de dados básicas.

Listas: java.util.List



Ela resolve todos os problemas que levantamos em relação aos vetores (busca, remoção, tamanho "infinito",...)

ArrayList



➔A implementação mais utilizada da interface List é a ArrayList.

- ✔ ArrayList é mais rápida na pesquisa do que sua concorrente;

```
ArrayList lista1 = new ArrayList();
```

- ✔ LinkedList é mais rápida na inserção e remoção de itens nas pontas da lista.

```
LinkedList lista2 = new LinkedList();
```

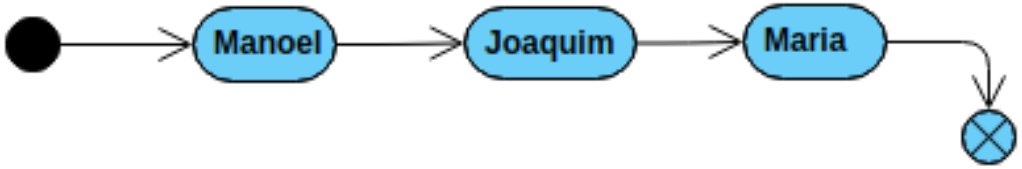
ArrayList

➔Para criar um ArrayList, basta chamar o construtor:

```
ArrayList lista1 = new ArrayList();
```

➔Para criar uma lista de nomes (String), podemos fazer:

```
ArrayList lista = new ArrayList();
lista.add("Manoel");
lista.add("Joaquim");
lista.add("Maria");
```



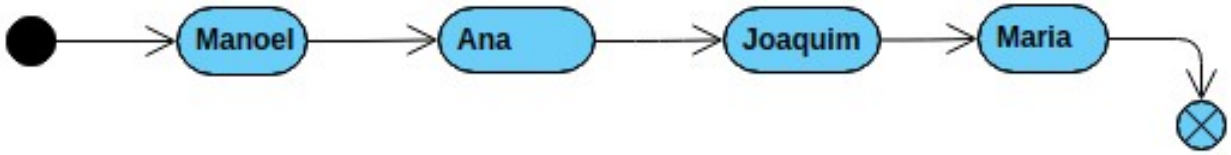
.add
Insere no fim

ArrayList

➔Outro método add :

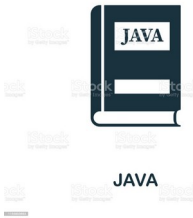
```
lista.add(1, "Ana");
```

.add
Insere na posição desejada



Não é necessário saber o tamanho da lista;
podemos acrescentar quantos elementos quisermos.

ArrayList



➔Toda lista (na verdade, toda Collection) trabalha do modo mais genérico possível.

```
ArrayList listaInteiros = new ArrayList();
        listaInteiros.add(1);
        listaInteiros.add(2);
```

```
ArrayList listaChar = new ArrayList();
        listaChar.add('c');
        listaChar.add('s');
```

```
ArrayList listaString = new ArrayList();
        listaString.add("Manoel");
        listaString.add("Joaquim");
```

```
ArrayList geral = new ArrayList();
        geral.add(1);
        geral.add(2.4);
        geral.add("oi");
        geral.add(new String("iftm"));
        geral.add(new Cliente("Jose"));
```

Não recomendado

ArrayList

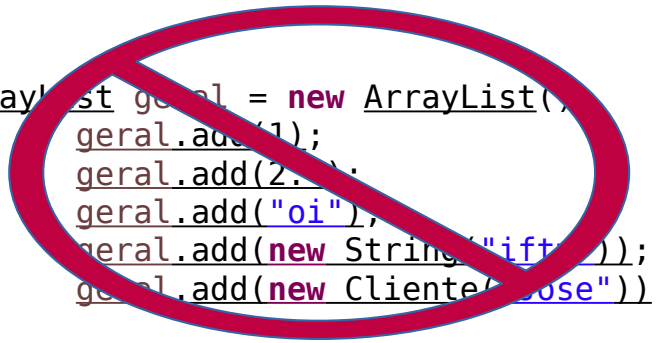
➔Uso de **Generics**, como especificar/especializar a lista :

```
ArrayList<String> listaString = new ArrayList<String>();
       listaString.add("Manoel");
       listaString.add("Joaquim");
       listaString.add("Maria");
```

```
ArrayList<Integer> listaInteiros = new ArrayList<Integer>();
       listaInteiros.add(1);
       listaInteiros.add(2);
```

```
ArrayList<Character> listaChar = new ArrayList<Character>();
      listaChar.add('c');
      listaChar.add('s');
```

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
      clientes.add(new Cliente("Jose"));
      clientes.add(new Cliente("Maria"));
```



```
ArrayList geral = new ArrayList();
      geral.add(1);
      geral.add(2.5);
      geral.add("oi");
      geral.add(new String("ifto"));
      geral.add(new Cliente("Jose"));
```

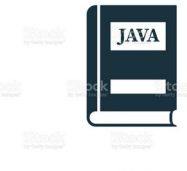
Classes wrappers:

Integer

Double

Character

ArrayList

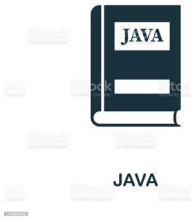


➔Quantos elementos há na lista?

✔ método size():

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
clientes.add(new Cliente("Jose"));
clientes.add(new Cliente("Maria"));
System.out.println(clientes.size());
```

ArrayList



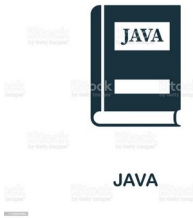
➔Há ainda um método get(int) que recebe como argumento o índice do elemento que se quer recuperar.

- ✔ Através dele, podemos fazer um for para iterar na lista:

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
        clientes.add(new Cliente("Jose"));
        clientes.add(new Cliente("Maria"));
        System.out.println(clientes.size());

        for (int i=0; i < clientes.size();i++) {
           Cliente cliente = clientes.get(i);
           System.out.println(cliente.getNome());
        }
```

Melhor utilizar Iterator

ArrayList



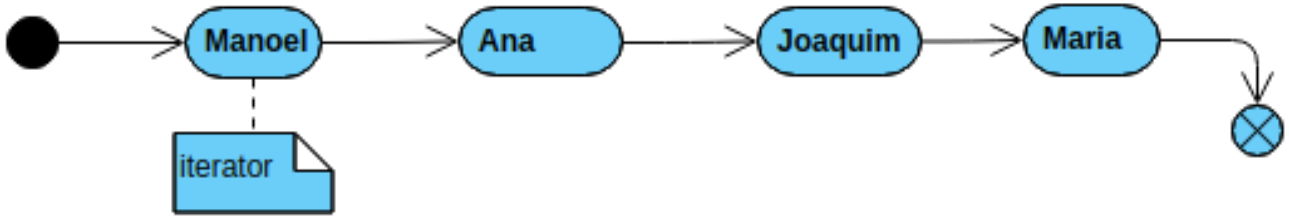
➔Caso a lista não seja especificada, será necessário realizar o cast para o tipo de dado retornado pelo get.

```
ArrayList clientes2 = new ArrayList();
clientes2.add(new Cliente("Jose"));
clientes2.add(new Cliente("Maria"));
System.out.println(clientes2.size());

for (int i=0; i < clientes2.size();i++) {
    Cliente cliente2 = (Cliente) clientes2.get(i);
    System.out.println(cliente2.getNome());
}
```

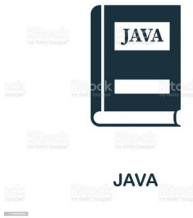
ArrayList

➔Utilizando o iterator:

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
clientes.add(new Cliente("Jose"));
clientes.add(new Cliente("Maria"));
System.out.println(clientes.size());


for (Iterator iterator = clientes.iterator(); iterator.hasNext();){
 Cliente cliente = (Cliente) iterator.next();
 System.out.println(cliente.getNome());
}
```



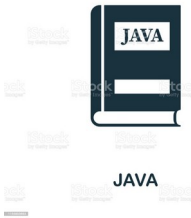
ArrayList



- remove() que recebe um elemento(ou índice) que se deseja remover da lista.
  - ✔ remove(objeto);
  - ✔ remove(índice);

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
Cliente cli1, cli2, cli3;
cli1 = new Cliente("Jose");
cli2 = new Cliente("Maria");
cli3 = new Cliente("Ana");
clientes.add(cli1);
clientes.add(cli2);
clientes.add(cli3);
clientes.remove(cli2);
clientes.remove(0);
for (Iterator iterator = clientes.iterator(); iterator.hasNext();){
    Cliente cliente = (Cliente) iterator.next();
    System.out.println(cliente.getNome());
}
```

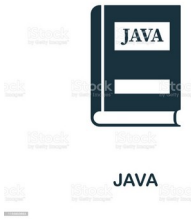
ArrayList



- contains() que recebe um objeto como argumento e devolve true ou false, indicando se o elemento está ou não na lista.
  - ✔ contains(objeto)

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
Cliente cli1, cli2, cli3;
cli1 = new Cliente("Jose");
cli2 = new Cliente("Maria");
clientes.add(cli1);
clientes.add(cli1);
clientes.add(cli2);
if (clientes.contains(cli2)==false)
    clientes.add(cli2);
for (Iterator iterator = clientes.iterator(); iterator.hasNext();){
    Cliente cliente = (Cliente) iterator.next();
    System.out.println(cliente.getNome());
}
```

ArrayList



- indexOf(objeto) : retorna a primeira posição de ocorrência do objeto. (existe o lastIndexOf)

```
System.out.println(clientes.indexOf(cli2));
```

- clear() : apaga todos os elementos da lista.

```
clientes.clear();
```

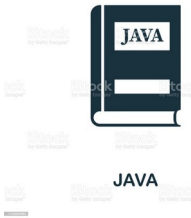
- isEmpty() : verifica se está vazio, retorna true ou false.

```
clientes.isEmpty()
```

- subList(indice1, indice2) : retorna uma sublista, essa lista não é um *ArrayList* e sim uma *List*.

```
ArrayList<Cliente> novos = new ArrayList<Cliente>(clientes.subList(0, 1));
```

ArrayList



➔set(indice, objeto) substitui um determinado elemento da lista.

```
ArrayList<Cliente> clientes = new ArrayList<Cliente>();
Cliente cli1, cli2, cli3;
cli1 = new Cliente("Jose");
cli2 = new Cliente("Maria");
clientes.add(cli2);
clientes.add(cli1);
clientes.add(cli1);
clientes.add(cli2);
clientes.set(clientes.indexOf(cli2), cli1);
for (Iterator iterator = clientes.iterator(); iterator.hasNext();){
    Cliente cliente = (Cliente) iterator.next();
    System.out.println(cliente.getNome());
}
```

Fim aula 03 - parte01.....

bruno.queiroz@iftm.edu.br