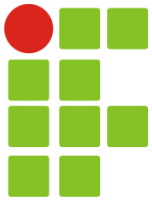


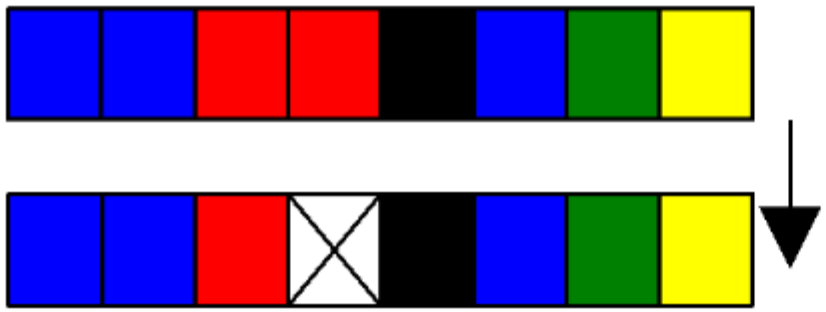
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE
CAMPUS JOÃO CÂMARA

COLEÇÕES DE OBJETOS - LISTAS

Nickerson Fonseca Ferreira
nickerson.ferreira@ifrn.edu.br

Introdução

- Já trabalhamos com uma estrutura de dados que armazena um conjunto de dados de um determinado tipo...
  - Como essa estrutura se chama ?
  - R: Vetores (ou Arrays)



Retire a quarta Conta

conta[3] = **null**;

Introdução

- Sua utilização é complexa e pouco flexível.
- Se for necessário adicionar uma posição ao vetor ?? É possível ??
- A utilização das Coleções de objetos (*Collections*) disponibiliza diversas classes que representam algumas estruturas de dados.
- As principais estruturas estudadas são:
  - Listas (*Lists*)
  - Conjuntos (*Sets*)
  - Mapas (*Maps*)

Listas

- É uma estrutura de dados semelhante ao vetor, porém, possui uma quantidade de elementos variável.
- Permite elementos duplicados.
- Todos as classes que implementam a interface java.util.List possuem todos os comportamentos que ela necessita para ser uma lista.
- A classe mais utilizada que implementa a interface List é a ArrayList.

Listas - ArrayList

- Bastante semelhante ao vetor, porém, NÃO É UM VETOR.
- Recebe um conjunto de **Objects**.
- Para criar um ArrayList:
  - ArrayList lista = new ArrayList();
- NOTE QUE NÃO INFORMAMOS A QUANTIDADE DE ELEMENTOS.
- A lista se adequa a medida em que vamos adicionando novos elementos.
- Para adicionar um novo elemento na lista utilizamos o método add(Object e);

Listas - ArrayList


```
ArrayList l = new ArrayList();
l.add("Teste");
l.add(1000);

Conta c = new Conta();
c.setSaldo(1000);

l.add(c);
```

Listas - ArrayList

- Podemos manipular os elementos da lista utilizando os seguintes métodos:
  - Get: para recuperar um Object de um determinado índice.
  - Remove: retira um elemento da lista e reorganizando ela.
  - Contains: verifica se um Object passado como parâmetro existe na lista.
  - Size: informa o tamanho da lista.

Listas - ArrayList


```
ArrayList l = new ArrayList();
l.add("Teste");
l.add(1000);

Conta c = new Conta();
c.setSaldo(1000);

l.add(c);

System.out.println("Primeiro elemento: " + l.get(0));

l.remove(0);
l.remove(c);

System.out.println("Quantos elementos ? " + l.size() );

System.out.println("O elemento existe ? " + l.contains(1000) );
```

Listas - ArrayList

- Para percorrer uma lista realizamos de forma semelhante ao array (vetor).

```
for (Object o : l){

    System.out.println("Elemento: " + o);

}
```

Listas - ArrayList

- Nesse caso podemos adicionar vários tipos de objetos.
- Mas como saberemos o tipo em tempo de execução ?? Como saber se vamos executar o método sacar de uma classe tipo conta ou trim de uma String ??
- Podemos informar qual o tipo de elemento iremos aceitar na nossa lista, para isso Java possui os Genéricos (*Generics*).
- Agora criaremos uma lista utilizando o Generics:
  - `ArrayList<String> ls = new ArrayList();`

Listas - ArrayList


```
ArrayList<String> l = new ArrayList<>();
l.add("Teste");
l.add("Teste2");
l.add(1000);

Conta c = new Conta();
c.setSaldo(1000);
l.add(c);
```

Listas - ArrayList

- E para percorrer uma lista...

```
for (String s : l){

    System.out.println("Elemento: " + s);

}
```

Exercício 01

1. Crie uma classe TesteArrayListNumero que possui um método **main**.
2. Dentro do main crie um ArrayList de Integer.
3. Adicione 10 números informados pelo usuário.
4. Se o usuário tiver digitado os números 10, 100 ou 1000 mostre uma mensagem informando que ele ganhou um bônus de R$ 50,00.

Exercício 02

1. Crie uma classe TesteArrayListString que possui um método **main**.
2. Dentro do main crie um ArrayList de String (lista1).
3. Adicione 10 Strings informadas pelo usuário.
4. Percorra a lista verificando se o usuário digitou alguma String com menos de 3 caracteres. Em caso positivo, adicione essa String em outra lista que você vai criar (lista2).
5. Utilizando o método removeAll, remova todos os elementos dessa segunda lista (lista2) da lista principal (lista1).
6. No final imprima a quantidade de Strings da lista.

Listas - ArrayList

- Podemos ordenar os elementos da lista.
- Para isso utilizamos o método static Collections.sort(List l).
- Essa ordenação só funcionará caso o objeto possua uma ordenação natural (Ex: inteiros, strings, etc.)

- Collections.sort(lista);

Listas - ArrayList


```
ArrayList<String> l = new ArrayList<>();
l.add("teste");
l.add("teste2");
l.add("aaTeste2");
l.add("ccTeste2");
l.add("bbTeste2");

Collections.sort(l);

for (String s : l){

    System.out.println("Elemento: " + s);

}
```

Exercício 01

1. Crie uma classe TesteArrayListNumero que possui um método **main**.
2. Dentro do main crie um ArrayList de Integer.
3. Adicione 10 números informados pelo usuário.
4. Percorra a lista imprimindo o valor do número inteiro de cada objeto.
5. Ordene os objetos contidos na lista.
6. Percorra a lista imprimindo o valor do número inteiro de cada objeto (NOTE QUE NESTE PONTO OS ELEMENTOS JÁ ENCONTRAM-SE ORDENADOS).

Exercício 02

1. Crie uma classe TesteArrayListString que possui um método **main**.
2. Dentro do main crie um ArrayList de String.
3. Adicione 10 Strings informadas pelo usuário.
4. Percorra a lista imprimindo o valor de cada String.
5. Ordene os objetos contidos na lista.
6. Percorra a lista imprimindo o valor de cada String (NOTE QUE NESTE PONTO OS ELEMENTOS JÁ ENCONTRAM-SE ORDENADOS).

Trabalho

- Implemente uma classe Conta que possui um saldo como atributo.
- Crie uma classe teste onde possui uma lista de vários objetos do tipo Conta.
- Ordene essa lista a partir do saldo da conta.
- **Vale 3 pontos extra.**
- **DICA: pesquisem sobre a interface Comparable.**