Coleções de Objetos
Interface Collection

Profa. Karen Selbach Borges

Última Atualização: Jun/2021

Interface Collection

- Define métodos para a manipulação de um conjunto não ordenado de elementos
  - Adição de elementos
  - Remoção de elementos
  - Teste de continência
  - Listagem de todos os elementos (em ordem arbitrária)

Adição de Elementos

- **boolean add (E e)** : adiciona à coleção o elemento passado como argumento.
- Se a coleção em questão não aceitar duplicatas, então é retornado *False* e o objeto não é inserido.
- Exemplo :

```
Collection<String> c = new ArrayList<String>(); // Uma coleção de Strings
c.add(“Jose”); // Adiciona José à lista e retorna true
c.add(“Maria”); // Adiciona Maria à lista e retorna true
booelan b = c.add(“Maria”); // Não adiciona Maria à lista e retorna false
```

Adição de Elementos

- **boolean addAll(Collection c)** : faz a união entre duas coleções.
- Retorna true se a coleção apresentar modificações após a adição de c.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“José”);
Collection<String> c2 = new ArrayList<String>();
c2.add(“João”);
c2.add(“Maria”);
boolean b = c.addAll(c2); //c = {“José, “Maria”, “João”}
```

Remoção de Elementos

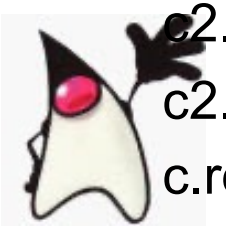
- **boolean remove(Object o)** : remove da coleção o objeto passado como argumento.
- Retorna f*alse* caso o objeto não exista na coleção.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.remove(“Jose”); // Retorna true
c.remove(“Pedro”); //Retorna false
```

Remoção de Elementos

- **boolean removeAll(Collection c)** : remove uma coleção interna de elementos.
- Retorna true se a coleção apresentar modificações após a remoção de c
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.add(“Joao”); // c = {“José, “Maria”, “João”};
Collection<String> c2 = new ArrayList<String>();
c2.add(“Jose”);
c2.add(“Maria”); //c2 = {“José”, “Maria”}
c.removeAll(c2); // c = {“João”}
```



Remoção de Elementos

- **boolean retainAll(Collection c)** : remove todos os elementos que não estejam contidos na coleção passada como argumento.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.add(“Joao”); // c = {“José, “Maria”, “João”}
Collection<String> c2 = new ArrayList<String>();
c2.add(“Jose”); // c2 = {“José”}
c.retainAll(c2); // c = {“José”}
```

Teste de Continência

- **boolean contains(Object o)** : verifica se o objeto passado como argumento existe na coleção.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
System.out.println(c.contains(“Jose”)); // True
```

Teste de Continência

- **boolean containsAll(Collection c)** : retorna verdadeiro se todos os elementos contidos na coleção passada como argumento estejam presentes na coleção.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.add(“Joao”);
Collection<String> c2 = new ArrayList<String>();
c2.add(“Jose”);
c2.add(“Maria”);
c.containsAll(c2); // True
```

Listagem dos Elementos

- É feita a partir do uso da interface Iterator, a qual:
  - Implementa métodos que permitem percorrer qualquer tipo de coleção de objetos
  - Não garante que os elementos serão visitados na ordem em que foram inseridos
- Métodos principais:
  - E next(): examina os elementos da coleção um a um
  - boolean hasNext(): retorna true se houver outro elemento a visitar
  - void remove(): remove e retorna o último objeto visitado

Listagem dos Elementos

- **Iterator iterator():** retorna o objeto que faz a iteração entre os elementos da coleção.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.add(“Joao”);
Iterator<String> it = c.iterator();
while(it.hasNext()) {
  String nome = it.next();
  System.out.println(nome);
}
```

Outros Métodos

- **int size()** : retorna a quantidade de elementos da coleção
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
System.out.println(c.size()); // 2
```

Outros Métodos

- **void clear()** : apaga todo o conteúdo da coleção.
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.clear();
System.out.println(c.size()); // o
```

Outros Métodos

- **boolean isEmpty():** retorna true se a coleção estiver vazia
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.clear();
System.out.println(c.isEmpty()); // true
```

Outros Métodos

- **Object[ ] toArray() :** converte os elementos da coleção em um array (rápidos acesso aos elementos).
- Exemplo :

```
Collection<String> c = new ArrayList<String>();
c.add(“Jose”);
c.add(“Maria”);
c.add(“Joao”);
Object[] elementos = c.toArray();
for(int i=0; i<elementos.length;i++) {
   String nome = (String) elementos[i];
   System.out.println(nome);
}
```