Curso de Java

Collections

Coleções e Mapas

- O pacote java.util oferece duas categorias de estruturas de dados 'genéricas'
  - **Coleções**: implementam 'coleções' de objetos sob a forma de *listas* e *conjuntos*.
  - **Mapas**: mantém pares de objetos na forma (chave,valor).

A interface *Collection*

- define os métodos disponíveis para operar com as 'coleções' de objetos:
  - `size()`
  - `isEmpty()`
  - `clear()`
  - `add(Object)`
  - `contains(Object)`
  - `remove(Object)`
  - `containsAll(Collection)`
  - `removeAll(Collection)`
  - `retainAll(Collection)`
  - `toArray()`

A interface List

- Define os métodos para operações com listas:
  - `get(int)`
  - `set(Object,int)`
  - `add(Object,int)`
  - `remove(int)`
  - `indexOf(Object)`

ArrayList e *LinkedList*

- Implementam a interface *List*:
  - *ArrayList*: implementa uma lista de objetos num vetor cujo tamanho pode variar dinamicamente (!).
  - *LinkedList*: implementa uma lista de objetos sob a forma de uma lista ligada.
  - *ArrayList* é mais adequada em situações onde o acesso aleatório aos elementos é mais freqüente. A implementação do vetor de 'tamanho variável' é cara.
  - *LinkedList* é mais adequada em casos onde o acesso aleatório não é freqüente e o tamanho da lista pode variar muito.

A interface Set

- Define os métodos para operações com conjuntos de objetos (um conjunto não pode conter elementos repetidos):
  - **`add(Object)`**
  - **`remove(Object)`**
  - **`contains(Object)`**
  - **`addAll(Set)`**
  - **`retainsAll(Set)`**
  - **`removeAll(Set)`**

A interface *Set*

- Implementações da interface Set
  - HashSet: baseada em tabela de *hashing*.
  - TreeSet: baseada em árvore binária de busca balanceada.

Ma*pas*

- A interface Map define os seguintes métodos:
  - `put(Object key, Object value)`
  - `get(Object key)`
  - `putAll(Map)`
  - `remove(Object key)`
  - `contains(Object key)`
  - `size()`
  - `isEmpty()`
  - `clear()`

Mapas

- Classes que implementam a interface Map
  - HashMap – baseada em tabela de *hashing*.
  - TreeMap – baseada em árvore binária de busca balanceada.
  - LinkedHashMap – combinação de tabela de *hashing* e lista ligada.

A interface *Iterator*

- A interface *Iterator* define métodos para percorrer *Collections*:
  - iterator(): As classes que implementam a interface Collection oferecem este método que deve retornar um objeto Iterator para a coleção.
  - hasNext(): devolve o valor true se ainda existir objeto a ser percorrido na coleção.
  - next(): devolve o próximo objeto da coleção.

Iterator: um exemplo

```
...
String[] names = {
                    "um","dois","tres",
                    "quatro","cinco","seis"
                  };
...
ArrayList list = new ArrayList();
...
for(int i = 0; i < names.length; i++) list.add(names[i]);
...
Iterator it = list.iterator();
while(it.hasNext()){
   System.out.println("==> "+(String)it.next());
}
```

Outro exemplo

```
...
HashMap map = new HashMap();
...
for(int i = 0; i < names.length; i++)
  map.put(names[i],new Xbluft(names,i,etc));
...
Iterator it = map.keySet().iterator();
while(it.hasNext()){
   Xbluft xb = (Xbluft)map.get(it.next());
   System.out.println("==>"+xb.toString());
}
```