Coleções em Java

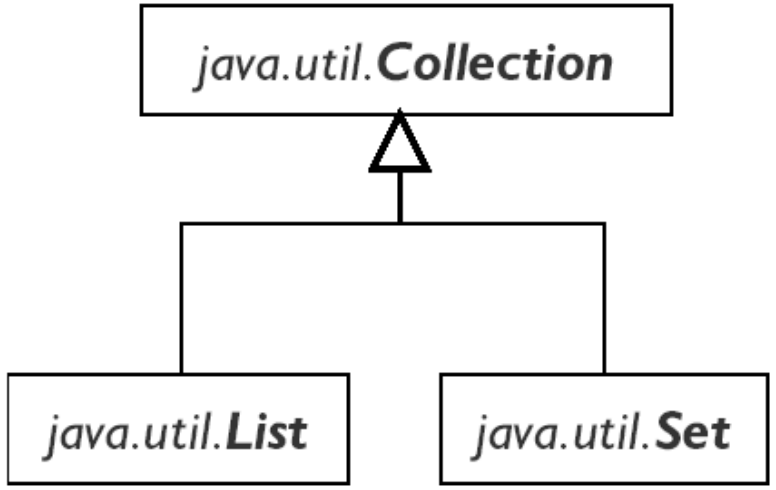
Técnicas de Programação

Jose Macedo

Coleções em Java

- Uma coleção é um objeto que agrupa outros tipos de objetos, conhecido pelo tipo dos seus elementos.
- A API de coleções contém interfaces para agrupar objetos como:
  - ***Collection*** ➔ ***grupo de objetos chamados de elementos.***
  - ***Set*** ➔ ***coleção não-ordenada. Duplicidade não é permitida.***
  - ***List*** ➔ ***coleção ordenada. Duplicidade é permitida.***
  - ***Vector*** ➔ ***implementa a interface List.***
  - ***Map*** ➔ ***implementa mapeamento chave-valor***

Coleções de elementos individuais



- *seqüência definida*
- *elementos indexados*

- *seqüência arbitrária*
- *elementos não repetem*

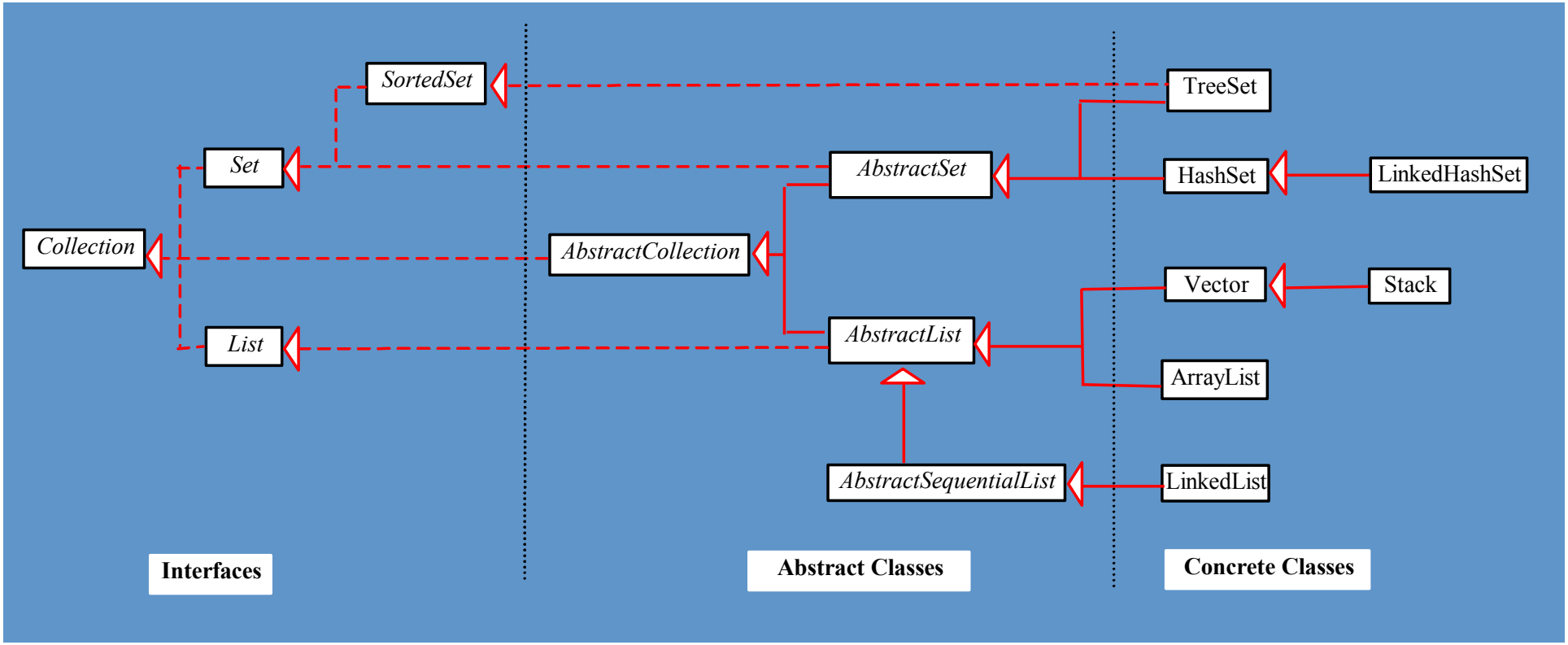
Coleções de pares de elementos

*java.util.**Map***

- *Pares chave/valor (vetor associativo)*
- ***Collection** de valores (podem repetir)*
- ***Set** de chaves (unívocas)*

Java Collection Framework

Set e List são subinterfaces de Collection.



Interface `Collection`

• `add(o)`	Add a new element
• `addAll(c)`	Add a collection
• `clear()`	Remove all elements
• `contains(o)`	Membership checking.
• `containsAll(c)`	Inclusion checking
• `isEmpty()`	Whether it is empty
• `iterator()`	Return an iterator
• `remove(o)`	Remove an element
• `removeAll(c)`	Remove a collection
• `retainAll(c)`	Keep the elements
• `size()`	The number of elements

Interface `List`

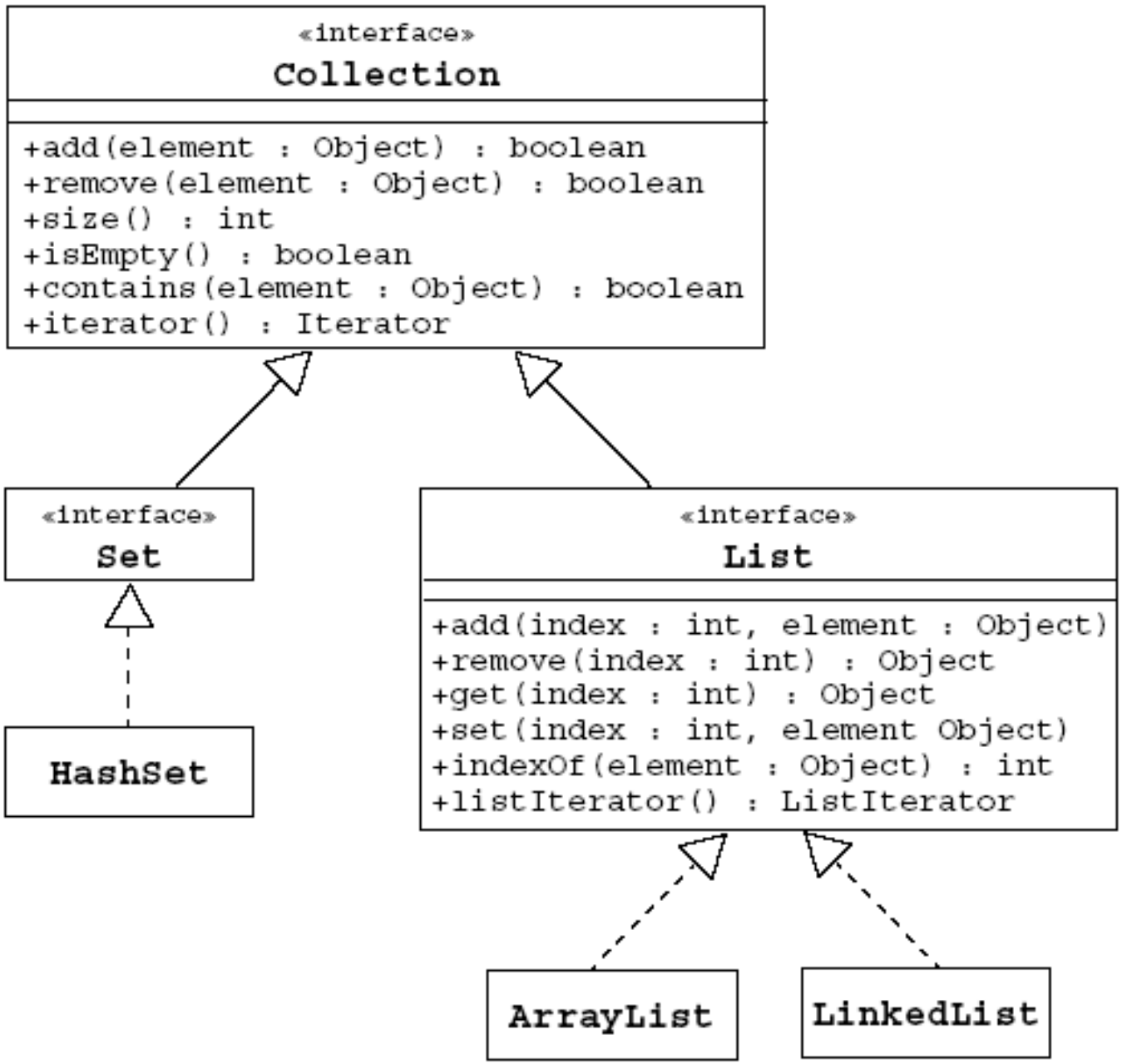
- `add(i,o)` Insert `o` at position `i`
- *`add(o)`* Append `o` to the end
- `get(i)` Return the `i`-th element
- `remove(i)` Remove the `i`-th element
- `set(i,o)` Replace the `i`-th element with `o`
- `indexOf(o)`
- `lastIndexOf(o)`
- `listIterator()`
- `sublist(i,j)`

Interface `Map`

• `clear()`	Remove all mappings
• `containsKey(k)`	Whether contains a mapping for `k`
• `containsValue(v)`	Whether contains a mapping to `v`
• `entrySet()`	Set of key-value pairs
• `get(k)`	The value associated with `k`
• `isEmpty()`	Whether it is empty
• `keySet()`	Set of keys
• `put(k,v)`	Associate `v` with `k`
• `remove(k)`	Remove the mapping for `k`
• `size()`	The number of pairs
• `values()`	The collection of values

Coleções Concretas

Concreta coleção	implementa	descricão
`HashSet`	`Set`	hash table
`TreeSet`	`SortedSet`	balanced binary tree
`ArrayList`	`List`	resizable-array
`LinkedList`	`List`	linked list
`Vector`	`List`	resizable-array
`HashMap`	`Map`	hash table
`TreeMap`	`SortedMap`	balanced binary tree
`Hashtable`	`Map`	hash table


«interface»
Collection
+add(element : Object) : boolean
+remove(element : Object) : boolean
+size() : int
+isEmpty() : boolean
+contains(element : Object) : boolean
+iterator() : Iterator
«interface»
Set
HashSet
«interface»
List
+add(index : int, element : Object)
+remove(index : int) : Object
+get(index : int) : Object
+set(index : int, element Object)
+indexOf(element : Object) : int
+listIterator() : ListIterator
ArrayList
LinkedList

Exemplo usando HashSet

```
1   import java.util.*;
2
3   public class SetExample {
4     public static void main(String[] args) {
5       Set set = new HashSet();
6       set.add("one");
7       set.add("second");
8       set.add("3rd");
9       set.add(new Integer(4));
10      set.add(new Float(5.0F));
11      set.add("second");          // duplicate, not added
12      set.add(new Integer(4));    // duplicate, not added
13      System.out.println(set);
14    }
15  }
```

Saída:

```
[one, second, 5.0, 3rd, 4]
```

Exemplo usando ArrayList

```
1   import java.util.*
2
3   public class ListExample {
4     public static void main(String[] args) {
5       List list = new ArrayList();
6       list.add("one");
7       list.add("second");
8       list.add("3rd");
9       list.add(new Integer(4));
10      list.add(new Float(5.0F));
11      list.add("second");          // duplicate, is added
12      list.add(new Integer(4));    // duplicate, is added
13      System.out.println(list);
14    }
15  }
```

Saída:

```
[one, second, 3rd, 4, 5.0, second, 4]
```

Iterar através de Coleções

- Interface `Iterator` :

```
interface Iterator {
  boolean hasNext();
  Object next();
  void remove();
}
```

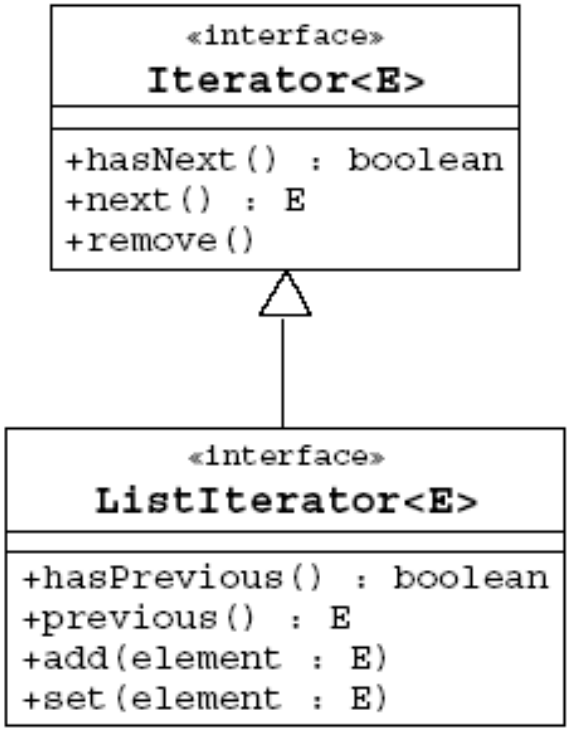
- Método `iterator()` método definido na interface `Collection`:

```
Iterator iterator()
```

Iteração

- Iteração é o processo de captura de cada elemento em uma coleção.
- Um ***Iterator*** de ***Set*** não é ordenado.
- Um ***ListIterator*** de uma ***List*** pode percorrer para frente(usando método **next()**) ou para trás(usando método **previous()**).

```
List list = new ArrayList();
// add some elements
Iterator elements = list.iterator();
while ( elements.hasNext() ) {
  System.out.println(elements.next());
}
```


«interface»
Iterator<E>
+hasNext() : boolean
+next() : E
+remove()
«interface»
ListIterator<E>
+hasPrevious() : boolean
+previous() : E
+add(element : E)
+set(element : E)

Vector Exemplo

```
Vector nomes = new Vector();
nomes.addElement("Joris");
nomes.addElement("Andre");
nomes.addElement("Mauricio");
nomes.addElement("Sheila");
System.out.println(nomes);
```

Saída:

[Joris,Andre,Mauricio,Sheila]

Vector

```
for (int i = 0; i < nomes.size(); i++)  {
//Object obj = (Object)nomes.elementAt(i);
String nome = (String)nomes.elementAt(i);
}
```

- Caso o elemento seja um array ou uma coleção, acessamos cada item através do método **`get()`**;

```
String[] nome = (String)nomes.elementAt(i);
System.ou.println(nome.get(0)+nome.get(1));
```

```
for (int i = 0; i < nomes.size(); i++)  {
 //Object obj =
(Object)nomes.elementAt(i);
 String nome = (String)nomes.elementAt(i);
 }
```

Iterando sobre um HashSet

```
Set set = new HashSet(); // instantiate a concrete set
// ...
set.add(obj); // insert an elements
// ...
int n = set.size(); // get size
// ...
if (set.contains(obj)) {...} // check membership

// iterate through the set
Iterator iter = set.iterator();
while (iter.hasNext()) {
  Object e = iter.next();
  // downcast e
  // ...
}
```

Map

- O Map tem uma combinação de chaves/ valores.
- Funciona da seguinte maneira: uma chave está associada a um valor.
- As chaves, claro, são únicas.
- Os valores podem ser duplicados.

Principais métodos

- put(Object key, Object value) – acrescenta um objeto na lista, associando-o a uma chave.
- get(Object key) - retorna um objeto através de sua chave.
- keySet() - retorna um Set com as chaves
- values() - retorna uma Collection com os valores.

Exemplo de Map

```
import java.util.*;

public class ExemploDeMap {
  public static void main(String[] args) {
    Map mapa = new HashMap();
    mapa.put("um", "Primeiro valor");
    mapa.put("dois", "Segundo valor");
    mapa.put("tres", "Terceiro valor");
    System.out.println("Valor recuperado: " + mapa.get("dois"));
    Set chaves = mapa.keySet();
    Collection valores = mapa.values();
    System.out.println(chaves);
    System.out.println(valores);
  }
}
```

Usando `Map`

```
Map map = new HashMap(); // instantiate a concrete map
// ...
map.put(key, val); // insert a key-value pair
// ...
// get the value associated with key
Object val = map.get(key);
map.remove(key); // remove a key-value pair
// ...
if (map.containsValue(val)) { ... }
if (map.containsKey(key)) { ... }
Set keys = map.keySet(); // get the set of keys
// iterate through the set of keys
Iterator iter = keys.iterator();
while (iter.hasNext()) {
  Key key = (Key) iter.next();
  // ...
}
```