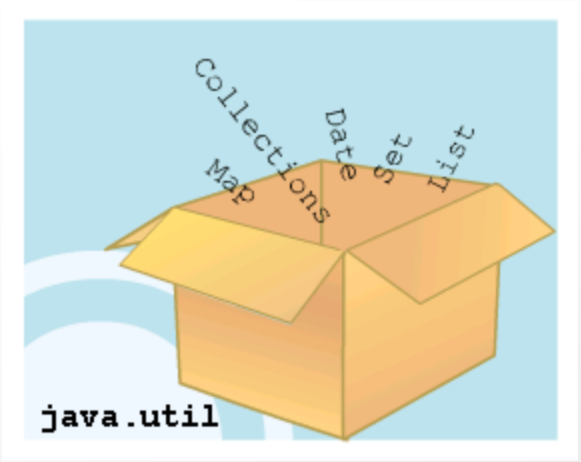
Hackademics Hanoi
Hack your passion, Code your life!

CHƯƠNG 12
COLLECTIONS

Gói java.util

- Bao gồm các lớp hỗ trợ:
  - Thao tác trên tập hợp
  - Mô hình sự kiện
  - Thao tác trên dữ liệu Date, Time
  - Toàn cầu hóa ứng dụng
  - Thao tác trên Chuỗi



Tập hợp(Collections)

- Tập hợp dùng lưu trữ, thao tác trên một nhóm các đối tượng.
- Các đối tượng của tập hợp có thể thuộc nhiều loại dữ liệu khác nhau
- Số phần tử trong tập hợp có thể thêm hoặc bớt

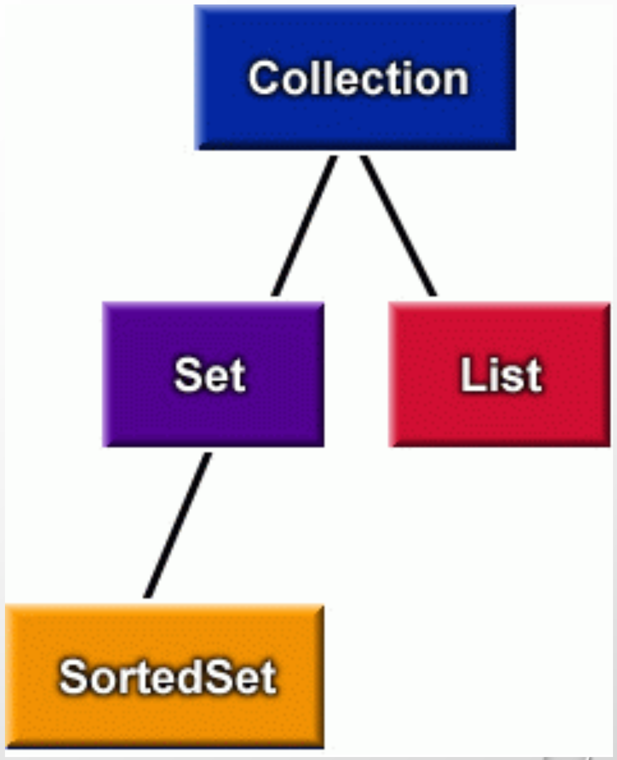
Các giao diện của Tập Hợp

1. List
   - Lưu trữ các phần tử theo thứ tự được thêm vào
   - Truy xuất các phần tử theo chỉ mục(index)
   - Các phần tử trong List có thể trùng nhau.

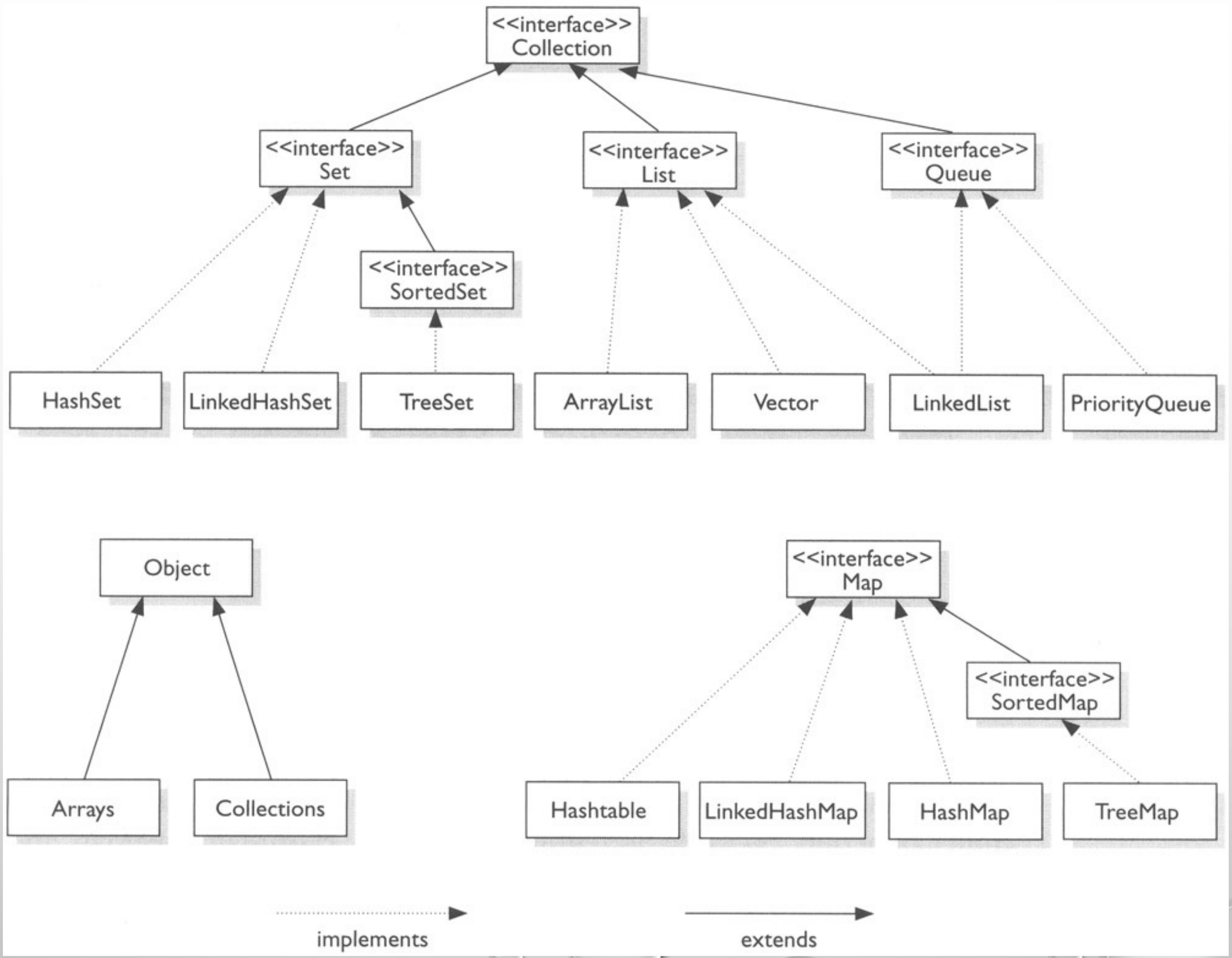
2. Set
   - Các phần tử trong Set lưu trữ không theo thứ tự đã thêm vào .
   - Không chấp nhận các phần tử trùng.

3. SortedSet
   - Thừa kế từ Set
   - Lưu trữ các phần tử th eo thứ tự tăng.
   - Không chấp nhận các phần tử trùng.

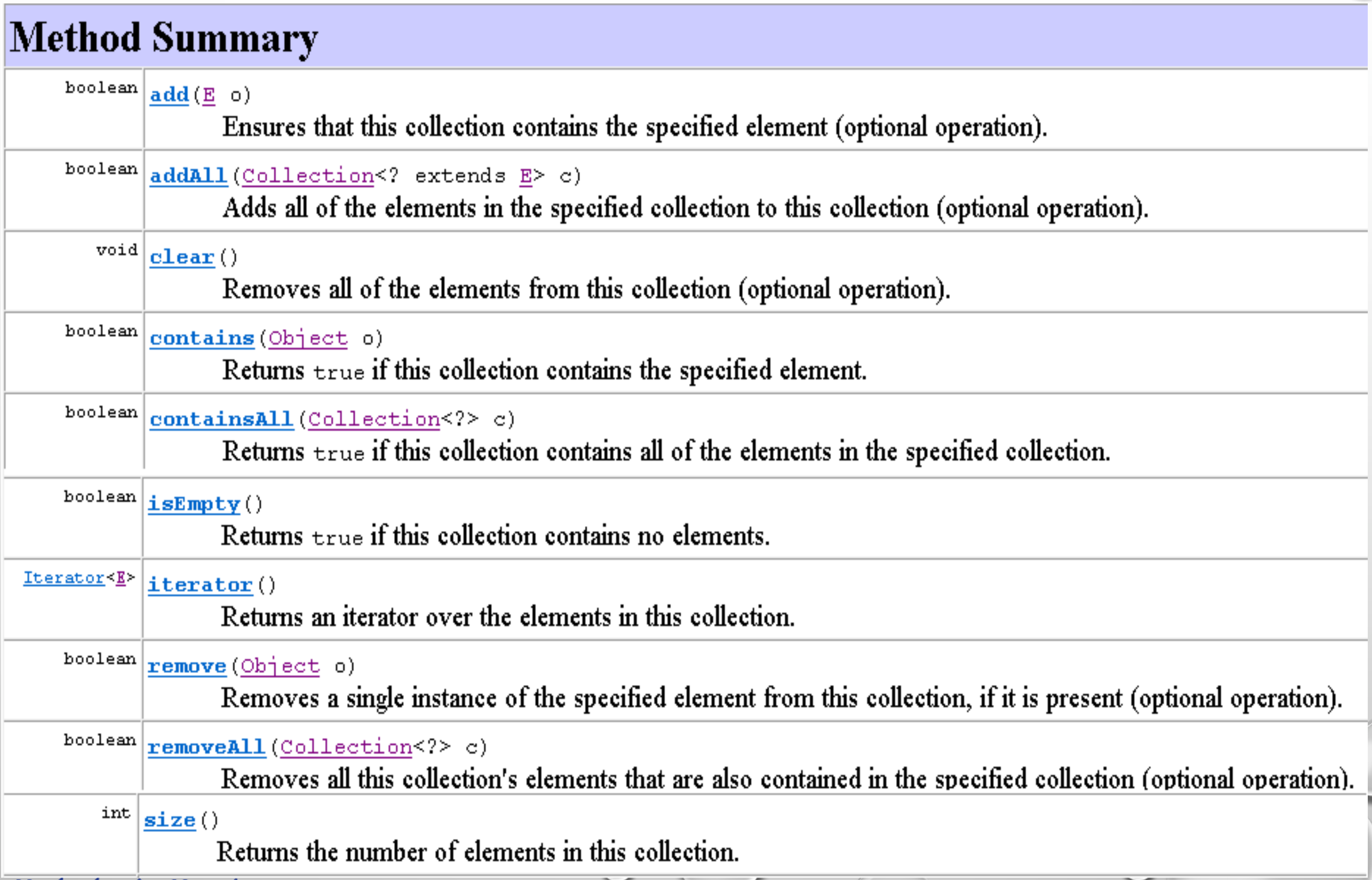
4. Queue



COLLECTION API



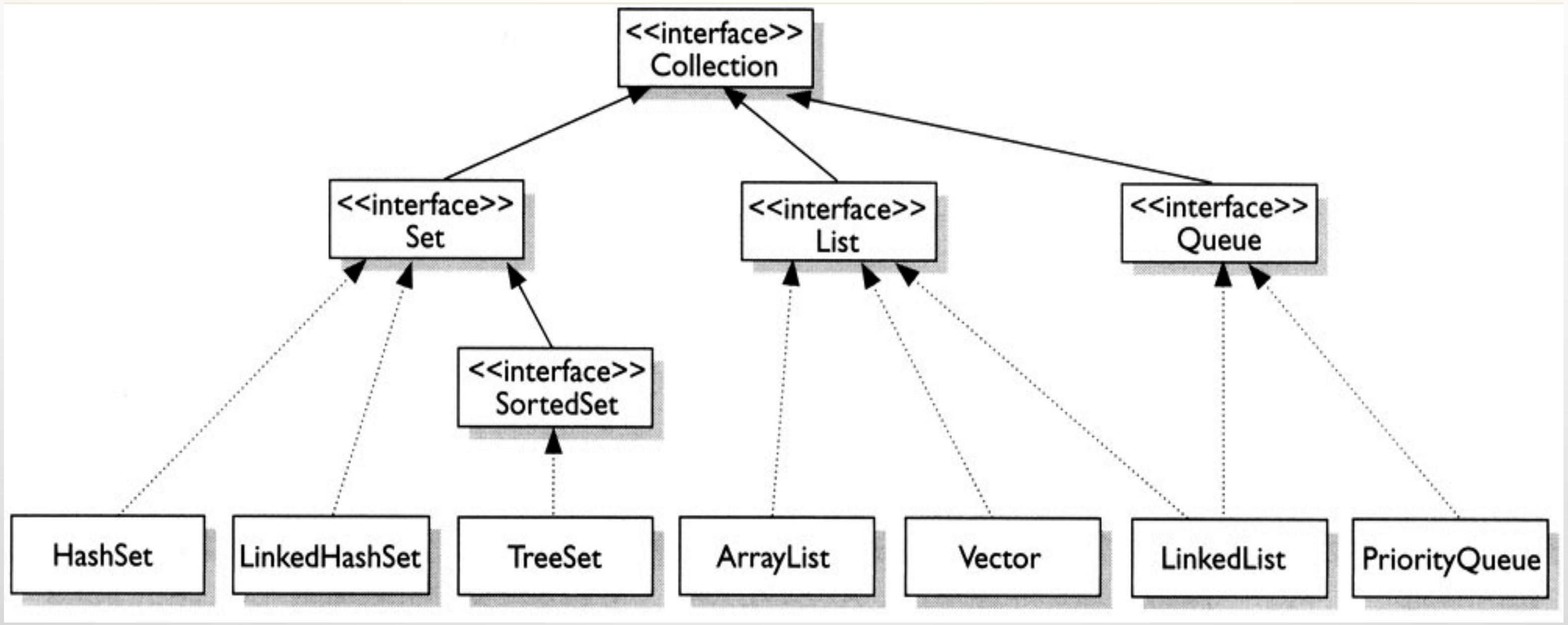
Các phương thức của các giao diện



Method Summary

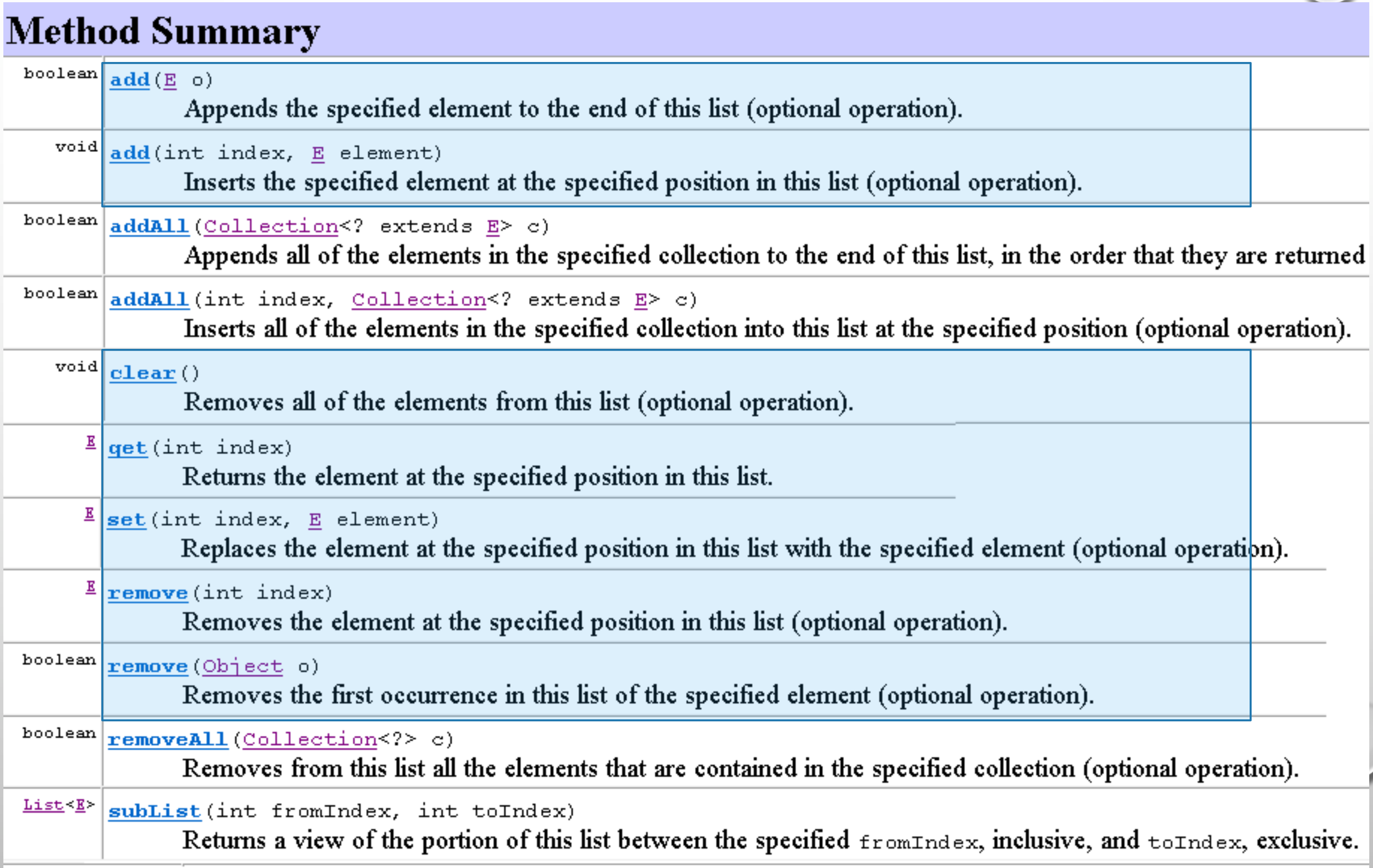
`boolean`	`add(E o)` Ensures that this collection contains the specified element (optional operation).
`boolean`	`addAll(Collection<? extends E> c)` Adds all of the elements in the specified collection to this collection (optional operation).
`void`	`clear()` Removes all of the elements from this collection (optional operation).
`boolean`	`contains(Object o)` Returns `true` if this collection contains the specified element.
`boolean`	`containsAll(Collection<?> c)` Returns `true` if this collection contains all of the elements in the specified collection.
`boolean`	`isEmpty()` Returns `true` if this collection contains no elements.
`Iterator<E>`	`iterator()` Returns an iterator over the elements in this collection.
`boolean`	`remove(Object o)` Removes a single instance of the specified element from this collection, if it is present (optional operation).
`boolean`	`removeAll(Collection<?> c)` Removes all this collection's elements that are also contained in the specified collection (optional operation).
`int`	`size()` Returns the number of elements in this collection.

LIST



<<interface>> Collection
<<interface>> Set
<<interface>> List
<<interface>> Queue
<<interface>> SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue

Các phương thức của List

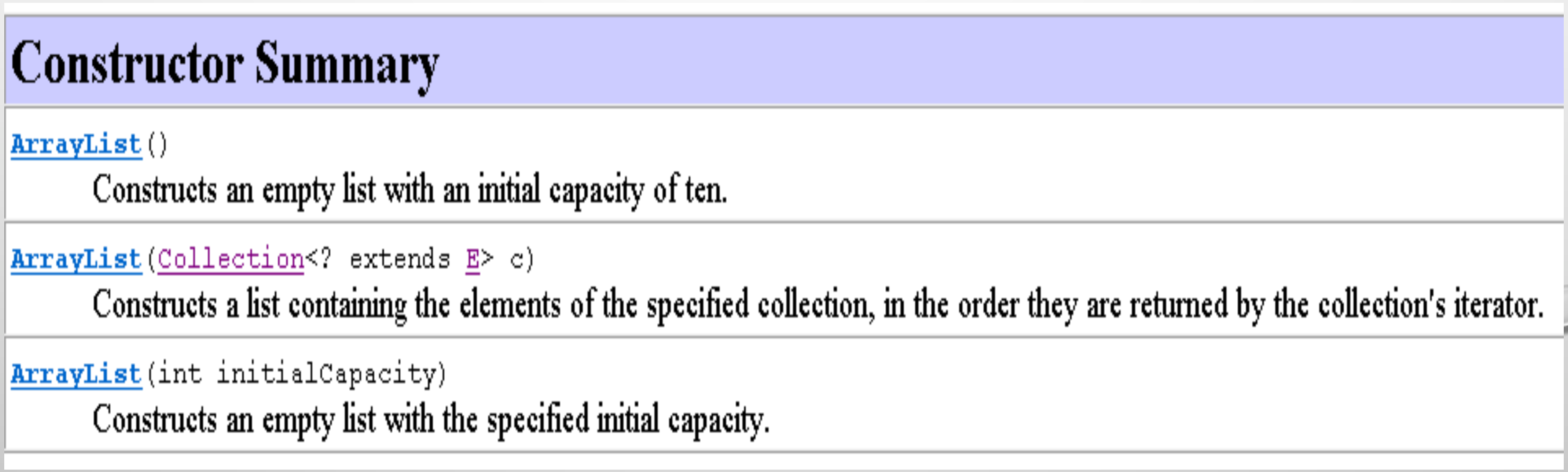


Method Summary

boolean	add(E o) Appends the specified element to the end of this list (optional operation).
void	add(int index, E element) Inserts the specified element at the specified position in this list (optional operation).
boolean	addAll(Collection<? extends E> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned
boolean	addAll(int index, Collection<? extends E> c) Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
void	clear() Removes all of the elements from this list (optional operation).
E	get(int index) Returns the element at the specified position in this list.
E	set(int index, E element) Replaces the element at the specified position in this list with the specified element (optional operation).
E	remove(int index) Removes the element at the specified position in this list (optional operation).
boolean	remove(Object o) Removes the first occurrence in this list of the specified element (optional operation).
boolean	removeAll(Collection<?> c) Removes from this list all the elements that are contained in the specified collection (optional operation).
List<E>	subList(int fromIndex, int toIndex) Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

ARRAYLIST

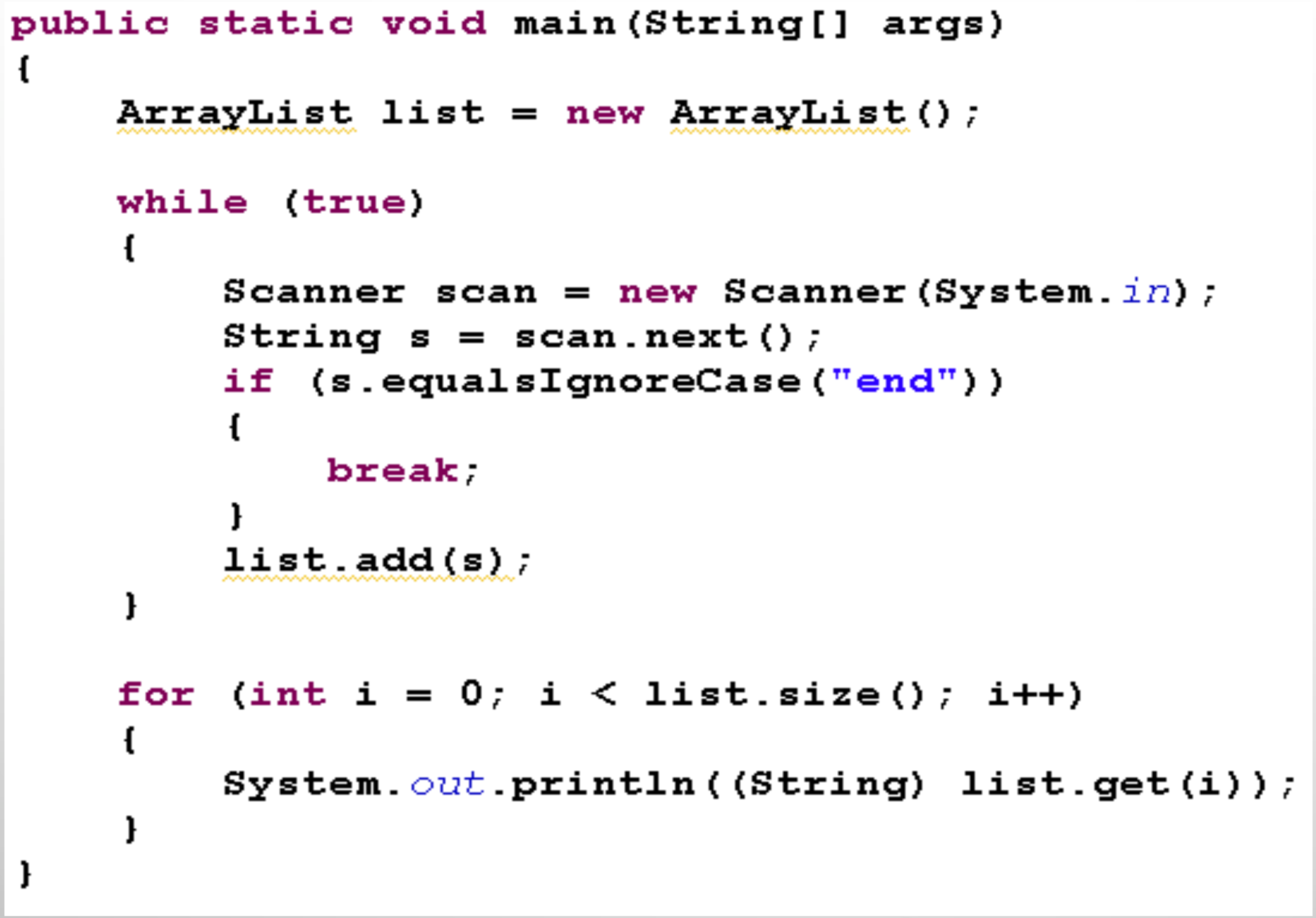
- Là một "thực thi" của giao diện List
- Phù hợp khi cần truy xuất ngẫu nhiên các phần tử trong tập hợp .



Constructor Summary
`ArrayList()` Constructs an empty list with an initial capacity of ten.
`ArrayList(Collection<? extends E> c)` Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.
`ArrayList(int initialCapacity)` Constructs an empty list with the specified initial capacity.

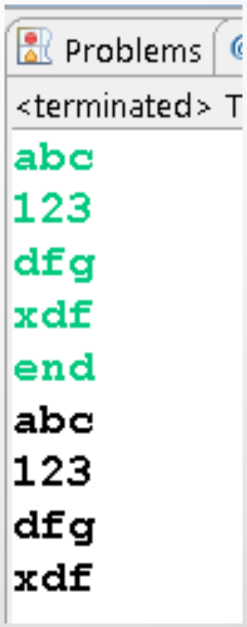
Hackademics Hanoi
Hack your passion, Code your life!

Ví dụ về ArrayList



```
public static void main(String[] args)
{
    ArrayList list = new ArrayList();

    while (true)
    {
        Scanner scan = new Scanner(System.in);
        String s = scan.next();
        if (s.equalsIgnoreCase("end"))
        {
            break;
        }
        list.add(s);
    }

    for (int i = 0; i < list.size(); i++)
    {
        System.out.println((String) list.get(i));
    }
}
```

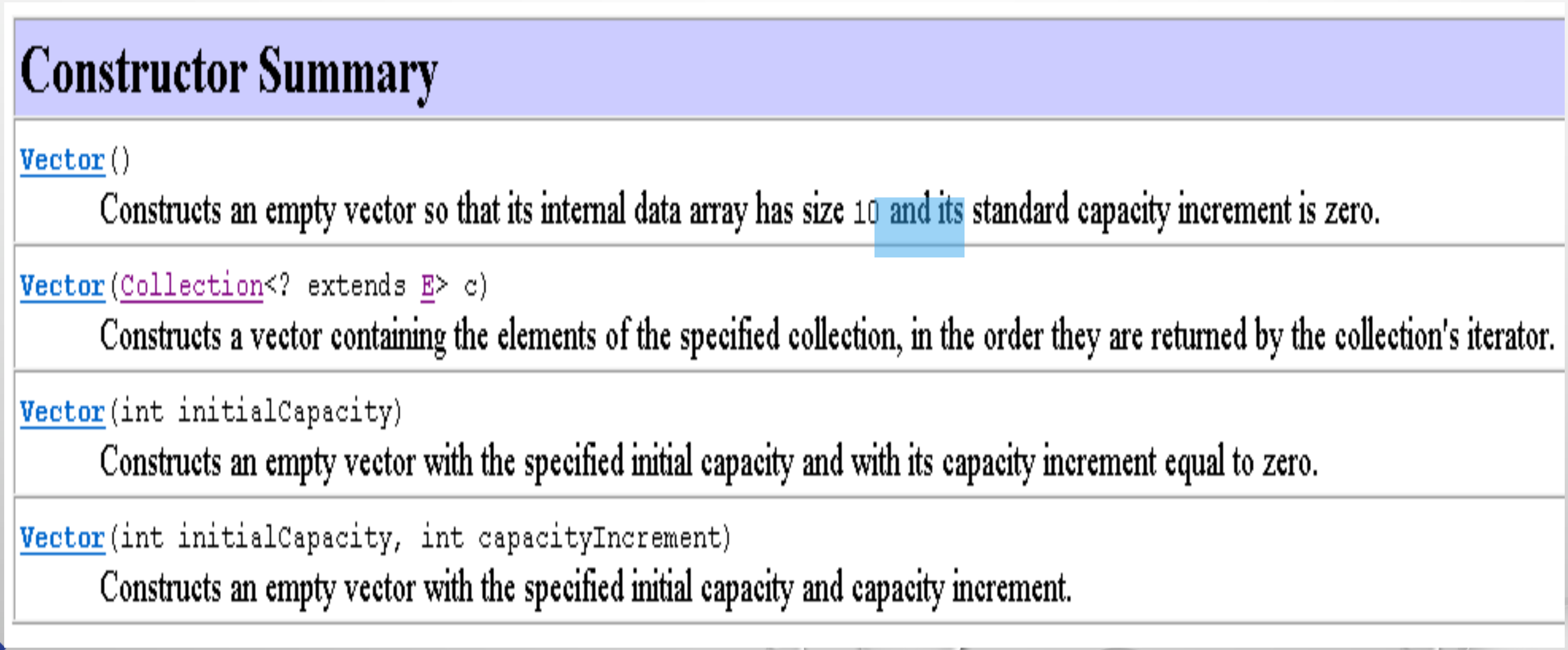
Output



```
<terminated> T
abc
123
dfg
xdf
end
abc
123
dfg
xdf
```

Lớp Vector

- Tương tự ArrayList
- Các phương thức của vector được đồng bộ → an toàn khi được sử dụng trong các Thread.

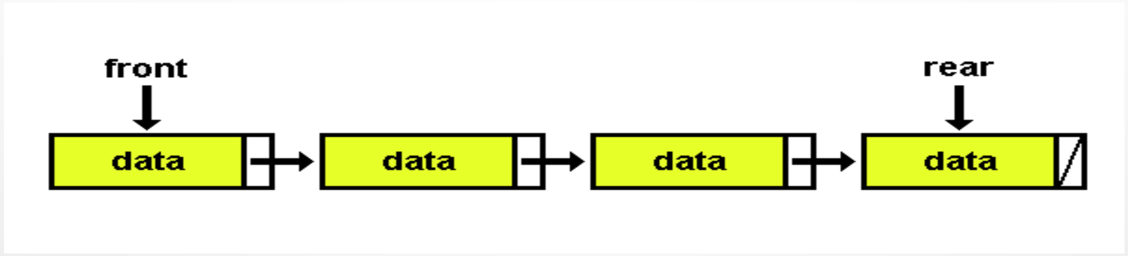


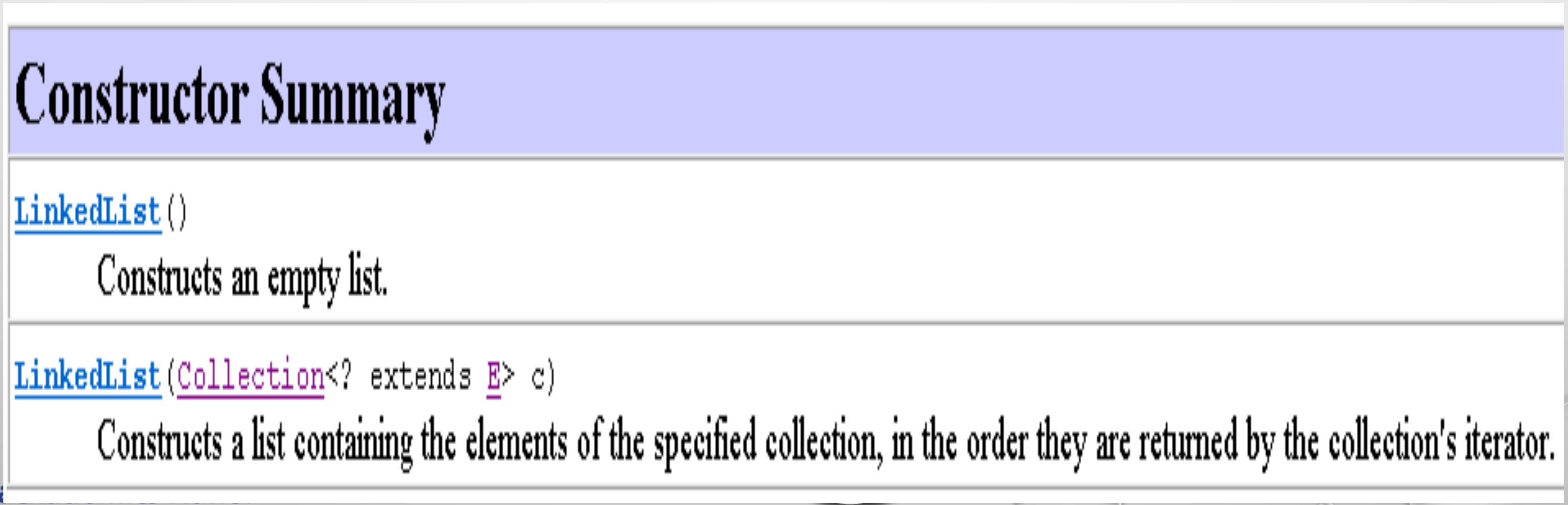
Constructor Summary
`Vector()` Constructs an empty vector so that its internal data array has size 10 and its standard capacity increment is zero.
`Vector(Collection<? extends E> c)` Constructs a vector containing the elements of the specified collection, in the order they are returned by the collection's iterator.
`Vector(int initialCapacity)` Constructs an empty vector with the specified initial capacity and with its capacity increment equal to zero.
`Vector(int initialCapacity, int capacityIncrement)` Constructs an empty vector with the specified initial capacity and capacity increment.

Hackademics Hanoi
Hack your passion, Code your life!

LinkedList

- Các phần tử được lưu trữ dạng một danh sách liên kết.





Constructor Summary

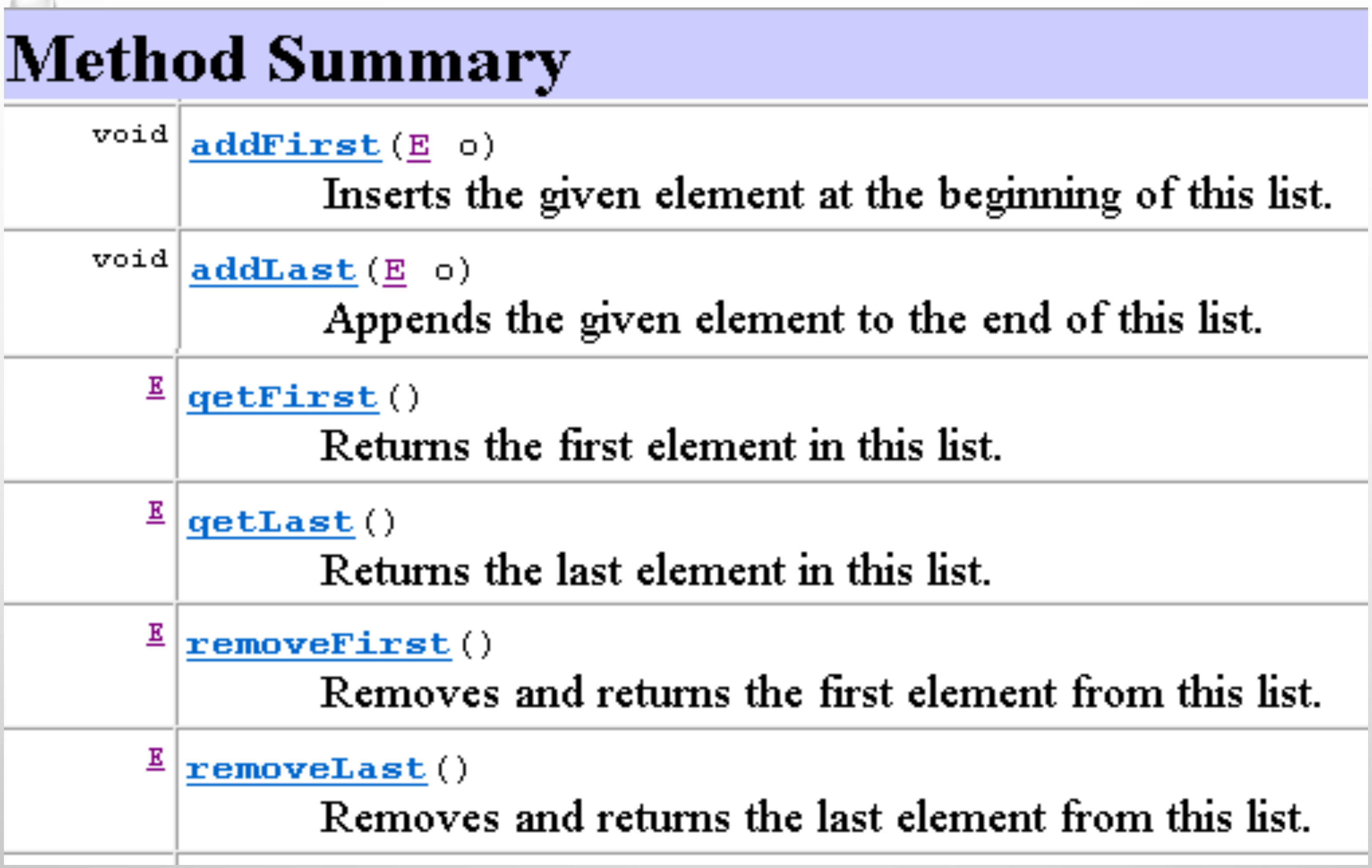
`LinkedList()`

Constructs an empty list.

`LinkedList(Collection<? extends E> c)`

Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator.

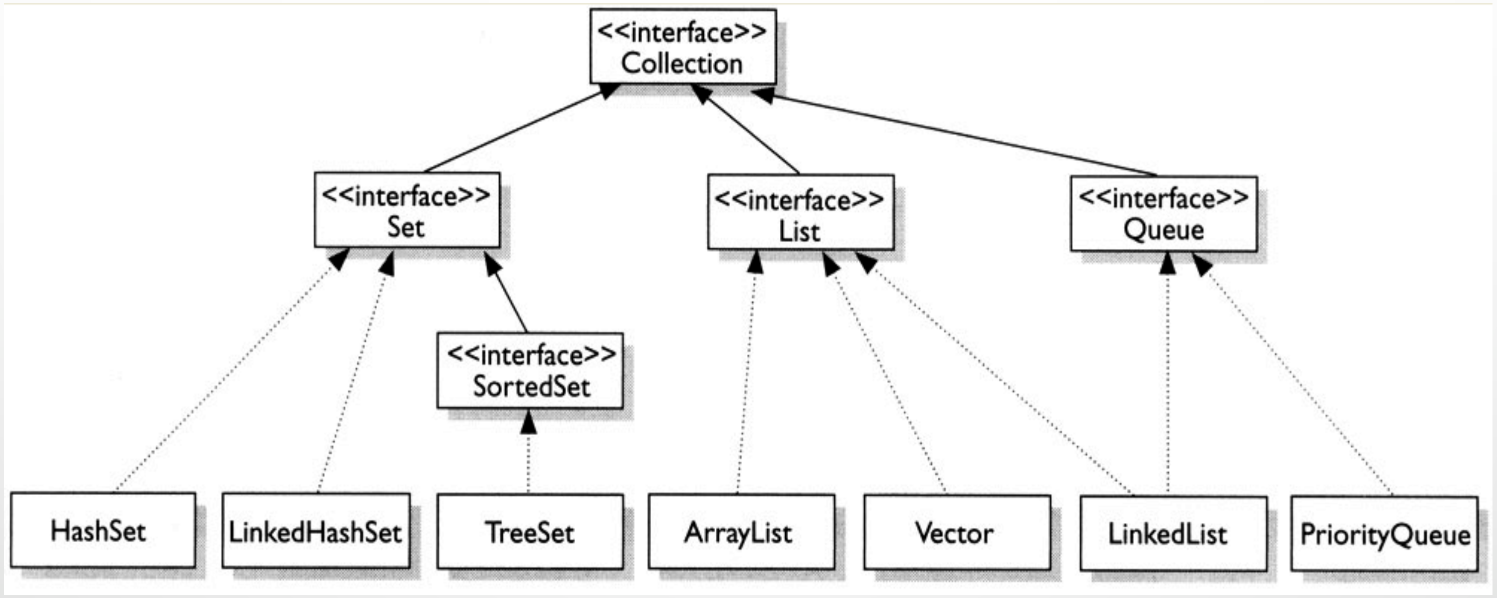
Các phương thức của lớp LinkedList



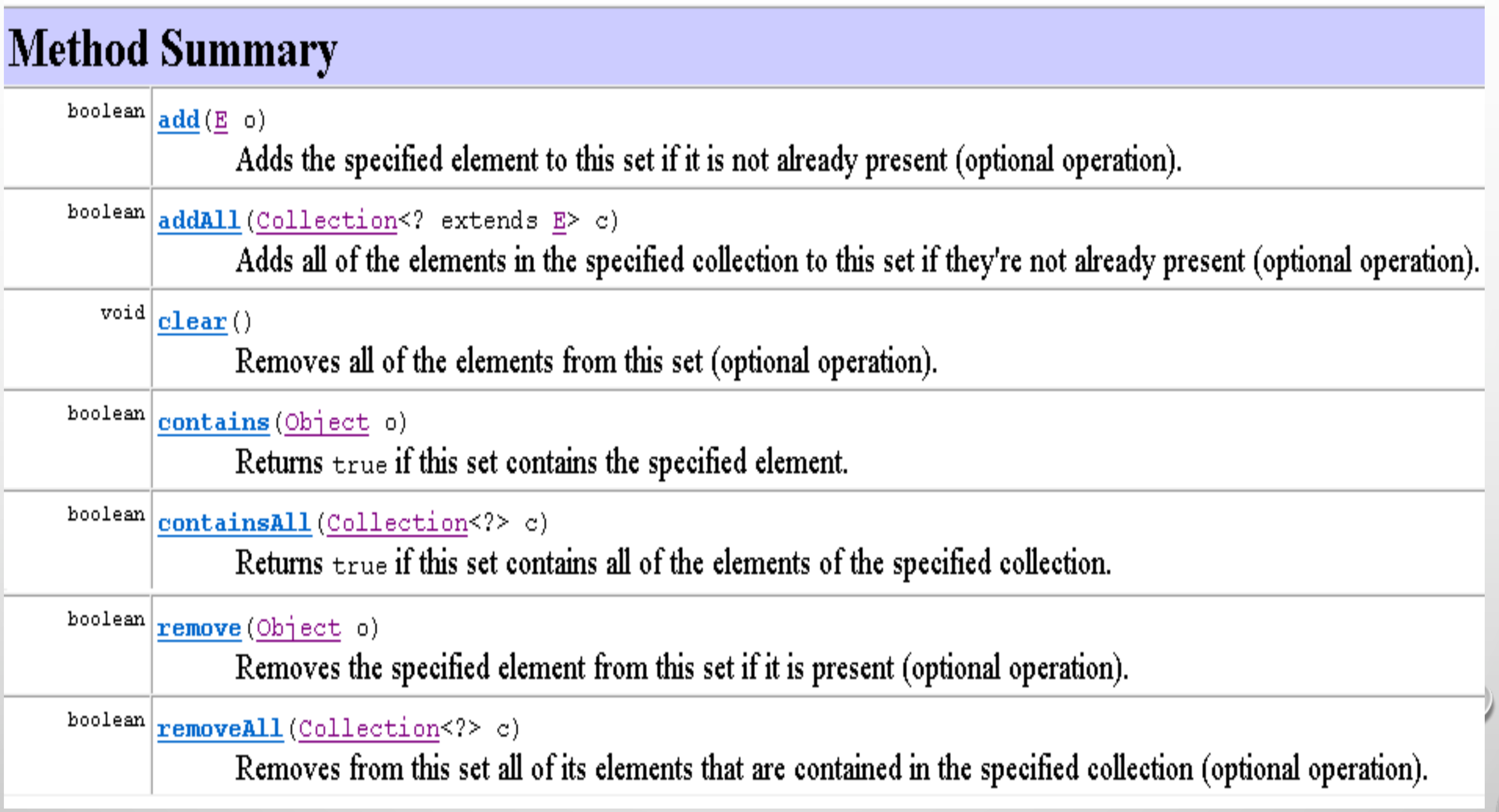
Method Summary

void	`addFirst(E o)` Inserts the given element at the beginning of this list.
void	`addLast(E o)` Appends the given element to the end of this list.
E	`getFirst()` Returns the first element in this list.
E	`getLast()` Returns the last element in this list.
E	`removeFirst()` Removes and returns the first element from this list.
E	`removeLast()` Removes and returns the last element from this list.

SET


<<interface>> Collection
<<interface>> Set
<<interface>> List
<<interface>> Queue
<<interface>> SortedSet
HashSet
LinkedHashSet
TreeSet
ArrayList
Vector
LinkedList
PriorityQueue

Các phương thức của Set

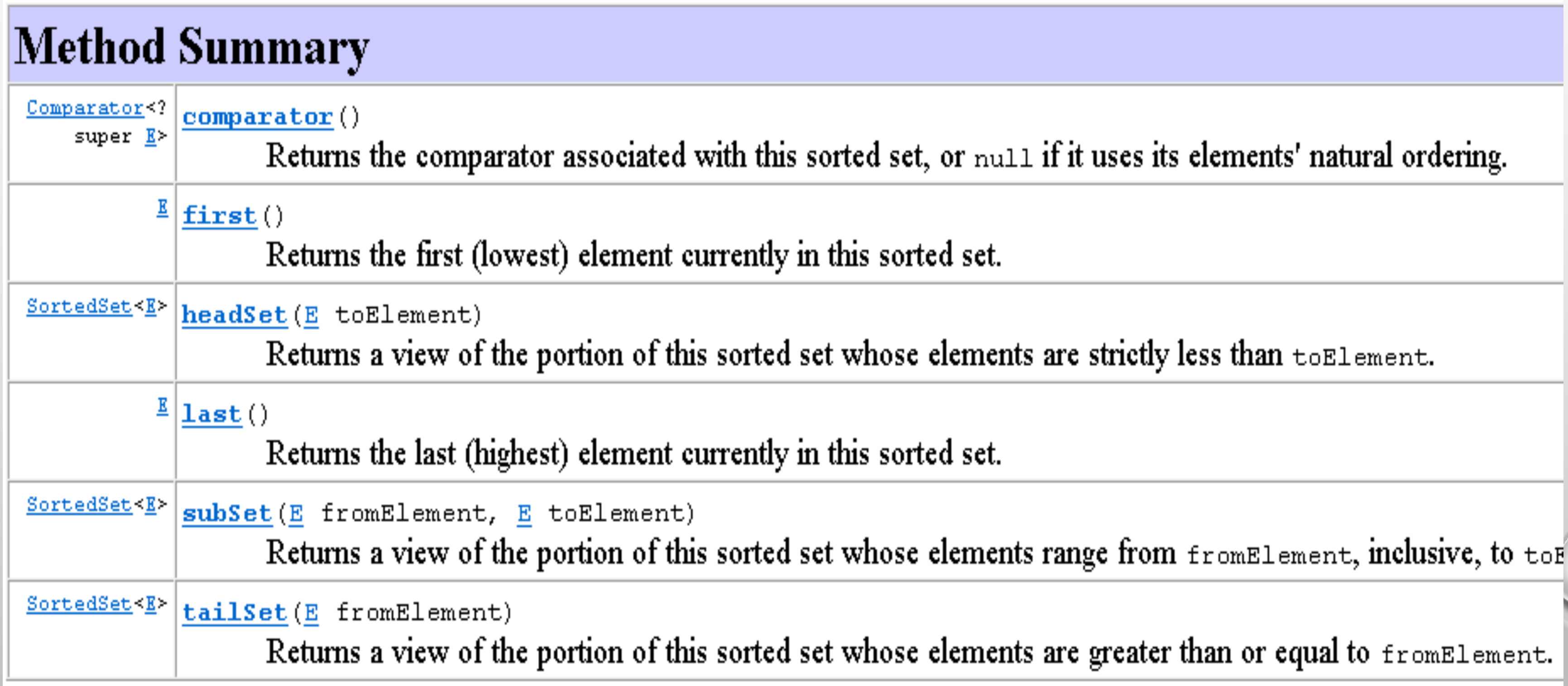


Method Summary

boolean	add(E o) Adds the specified element to this set if it is not already present (optional operation).
boolean	addAll(Collection<? extends E> c) Adds all of the elements in the specified collection to this set if they're not already present (optional operation).
void	clear() Removes all of the elements from this set (optional operation).
boolean	contains(Object o) Returns true if this set contains the specified element.
boolean	containsAll(Collection<?> c) Returns true if this set contains all of the elements of the specified collection.
boolean	remove(Object o) Removes the specified element from this set if it is present (optional operation).
boolean	removeAll(Collection<?> c) Removes from this set all of its elements that are contained in the specified collection (optional operation).

Giao diện SortedSet

- Thừa kế từ giao diên Set
- Không chấp nhận các đối tượng trùng nhau.



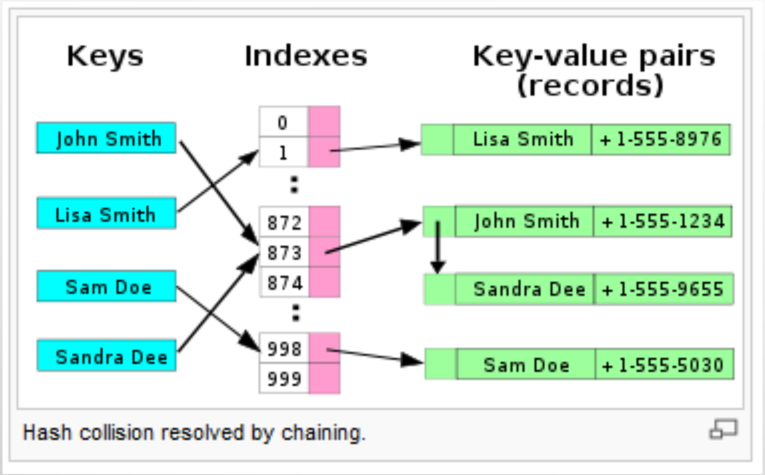
Method Summary	
Comparator<? super E>	comparator() Returns the comparator associated with this sorted set, or null if it uses its elements' natural ordering.
E	first() Returns the first (lowest) element currently in this sorted set.
SortedSet<E>	headSet(E toElement) Returns a view of the portion of this sorted set whose elements are strictly less than toElement.
E	last() Returns the last (highest) element currently in this sorted set.
SortedSet<E>	subSet(E fromElement, E toElement) Returns a view of the portion of this sorted set whose elements range from fromElement, inclusive, to toE
SortedSet<E>	tailSet(E fromElement) Returns a view of the portion of this sorted set whose elements are greater than or equal to fromElement.

Hackademics Hanoi
Hack your passion, Code your life!

Lớp HashSet

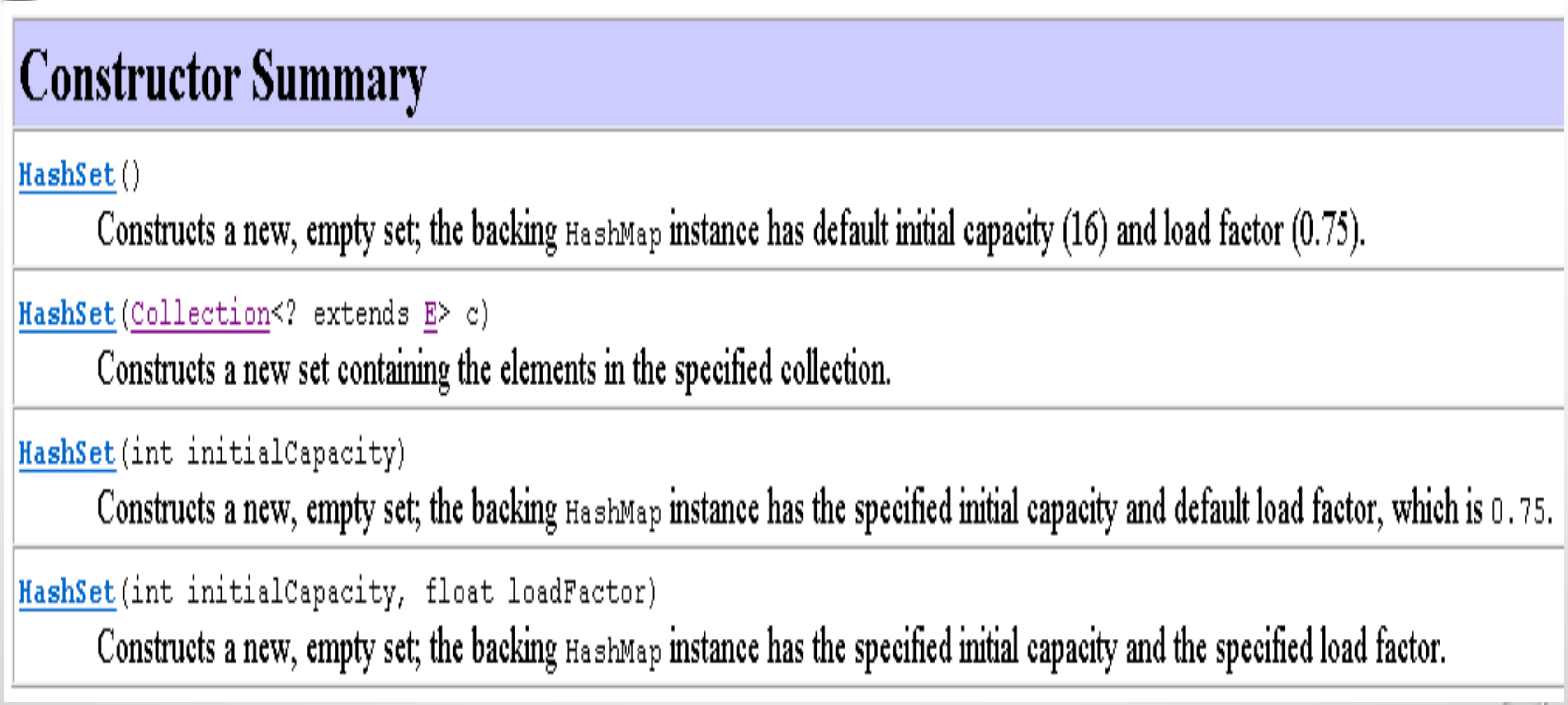
- Thực thi giao diện Set

- Sử dụng Hash Table để lưu dữ liệu.



Hash collision resolved by chaining.

Các constructor của HashSet



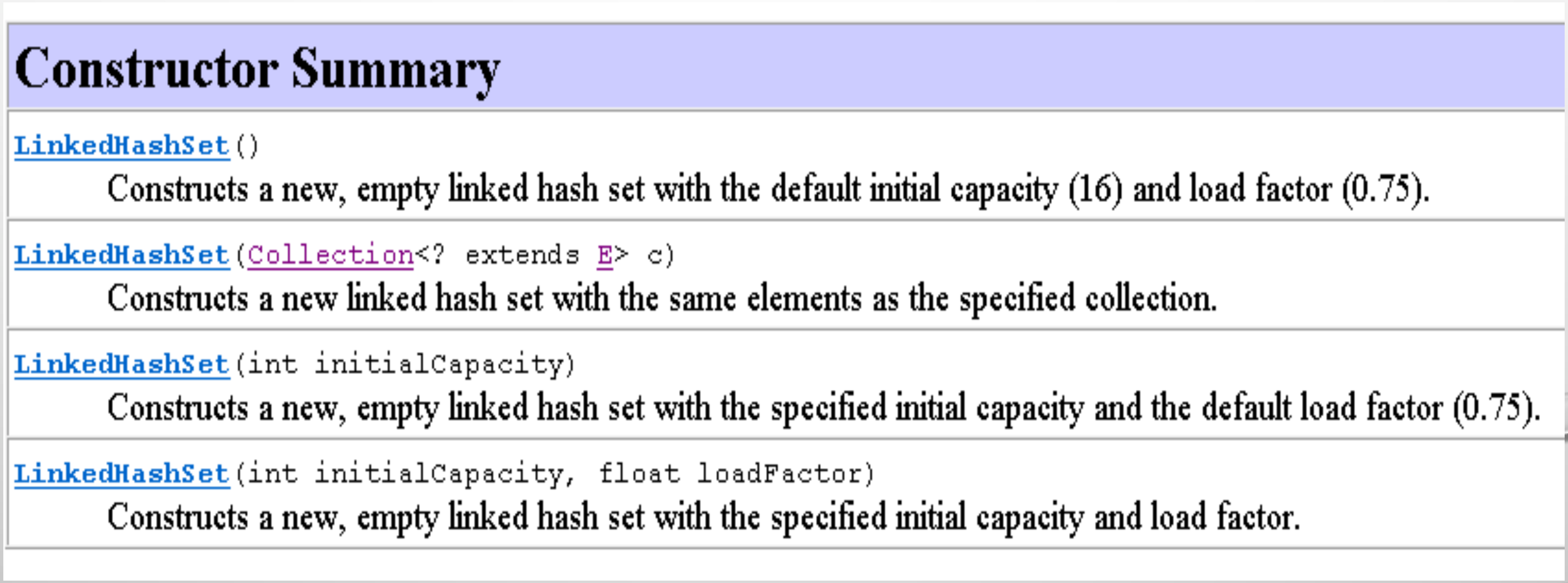
Constructor Summary
`HashSet()` Constructs a new, empty set; the backing `HashMap` instance has default initial capacity (16) and load factor (0.75).
`HashSet(Collection<? extends E> c)` Constructs a new set containing the elements in the specified collection.
`HashSet(int initialCapacity)` Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and default load factor, which is `0.75`.
`HashSet(int initialCapacity, float loadFactor)` Constructs a new, empty set; the backing `HashMap` instance has the specified initial capacity and the specified load factor.

Hackademics Hanoi
Hack your passion, Code your life!

Lớp LinkedHashSet

- Kết hợp giữa HashSet và LinkedList

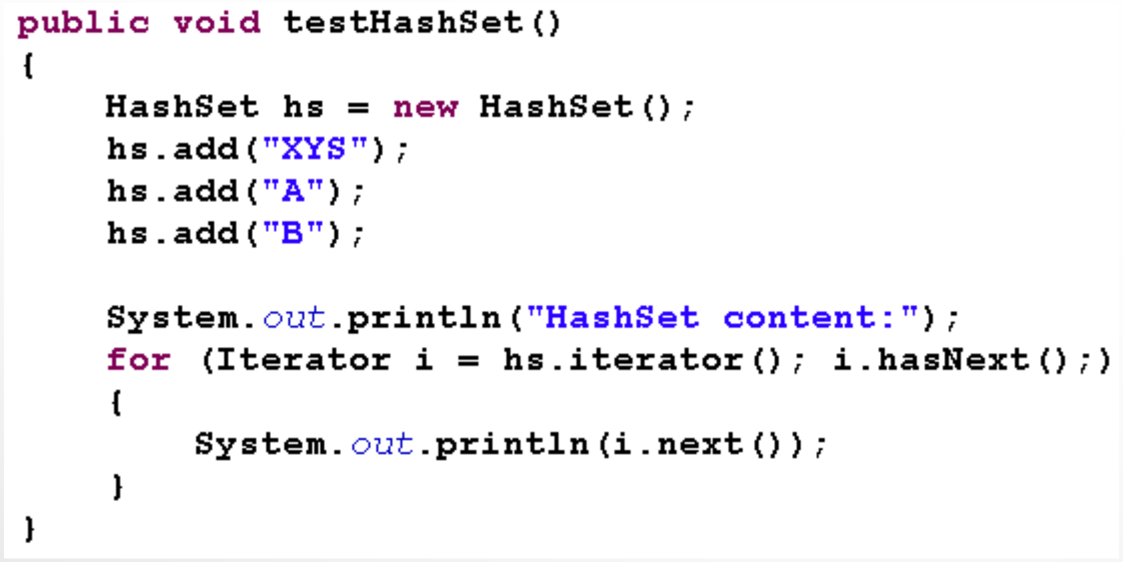
- Sử dụng một List để duy trì thứ tự của các phần tử như khi chúng được thêm vào



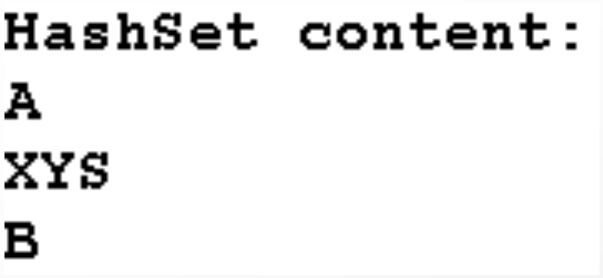
Constructor Summary

`LinkedHashSet()` Constructs a new, empty linked hash set with the default initial capacity (16) and load factor (0.75).
`LinkedHashSet(Collection<? extends E> c)` Constructs a new linked hash set with the same elements as the specified collection.
`LinkedHashSet(int initialCapacity)` Constructs a new, empty linked hash set with the specified initial capacity and the default load factor (0.75).
`LinkedHashSet(int initialCapacity, float loadFactor)` Constructs a new, empty linked hash set with the specified initial capacity and load factor.

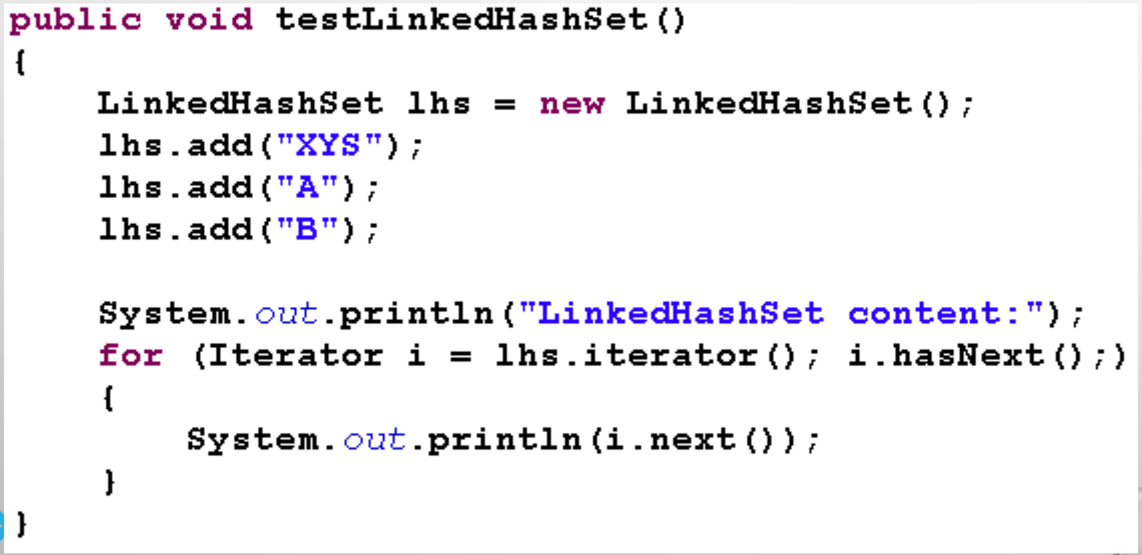
Ví dụ HashSet và LinkedHashSet



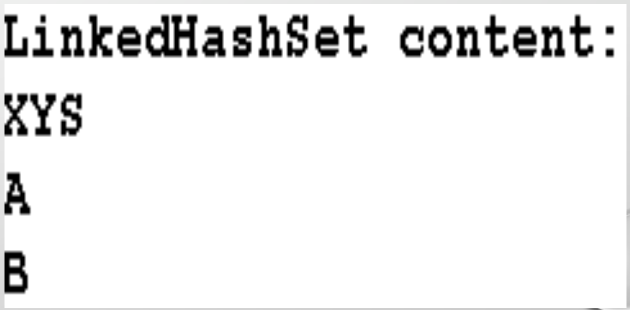
```java
public void testHashSet()
{
    HashSet hs = new HashSet();
    hs.add("XYS");
    hs.add("A");
    hs.add("B");

    System.out.println("HashSet content:");
    for (Iterator i = hs.iterator(); i.hasNext();)
    {
        System.out.println(i.next());
    }
}
```



```
HashSet content:
A
XYS
B
```



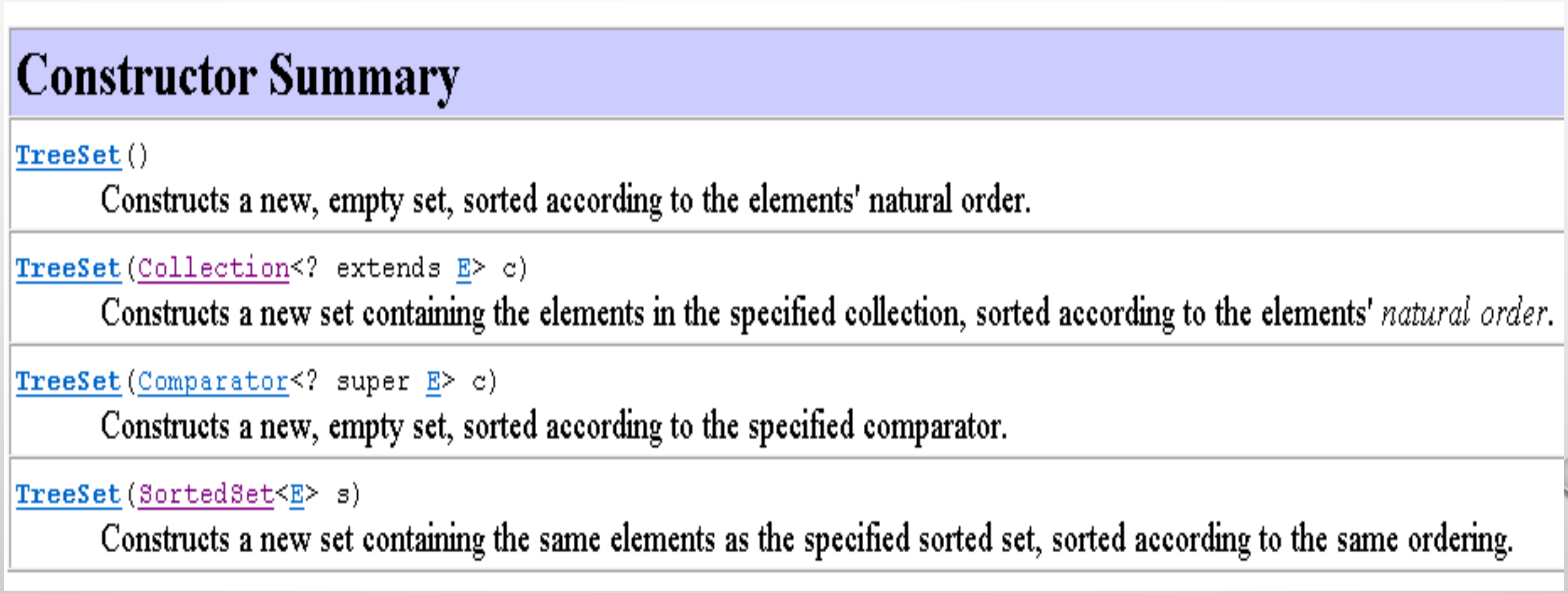
```java
public void testLinkedHashSet()
{
    LinkedHashSet lhs = new LinkedHashSet();
    lhs.add("XYS");
    lhs.add("A");
    lhs.add("B");

    System.out.println("LinkedHashSet content:");
    for (Iterator i = lhs.iterator(); i.hasNext();)
    {
        System.out.println(i.next());
    }
}
```



```
LinkedHashSet content:
XYS
A
B
```

Lớp TreeSet

- Lưu giữ liệu theo cấu trúc “cây”.
- Các phần tử được lưu trữ theo thứ tự tăng dần

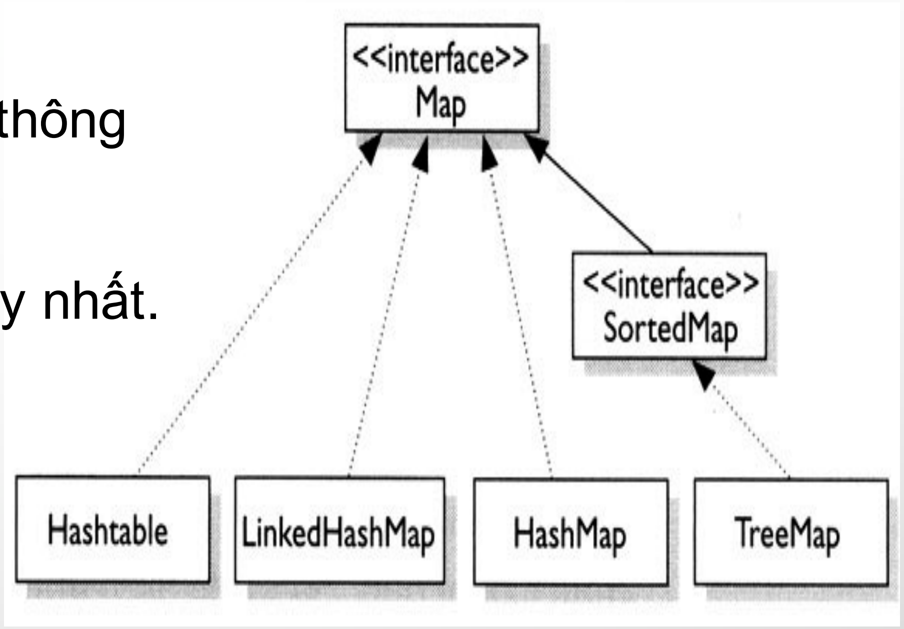


Constructor Summary
`TreeSet()` Constructs a new, empty set, sorted according to the elements' natural order.
`TreeSet(Collection<? extends E> c)` Constructs a new set containing the elements in the specified collection, sorted according to the elements' *natural order*.
`TreeSet(Comparator<? super E> c)` Constructs a new, empty set, sorted according to the specified comparator.
`TreeSet(SortedSet<E> s)` Constructs a new set containing the same elements as the specified sorted set, sorted according to the same ordering.

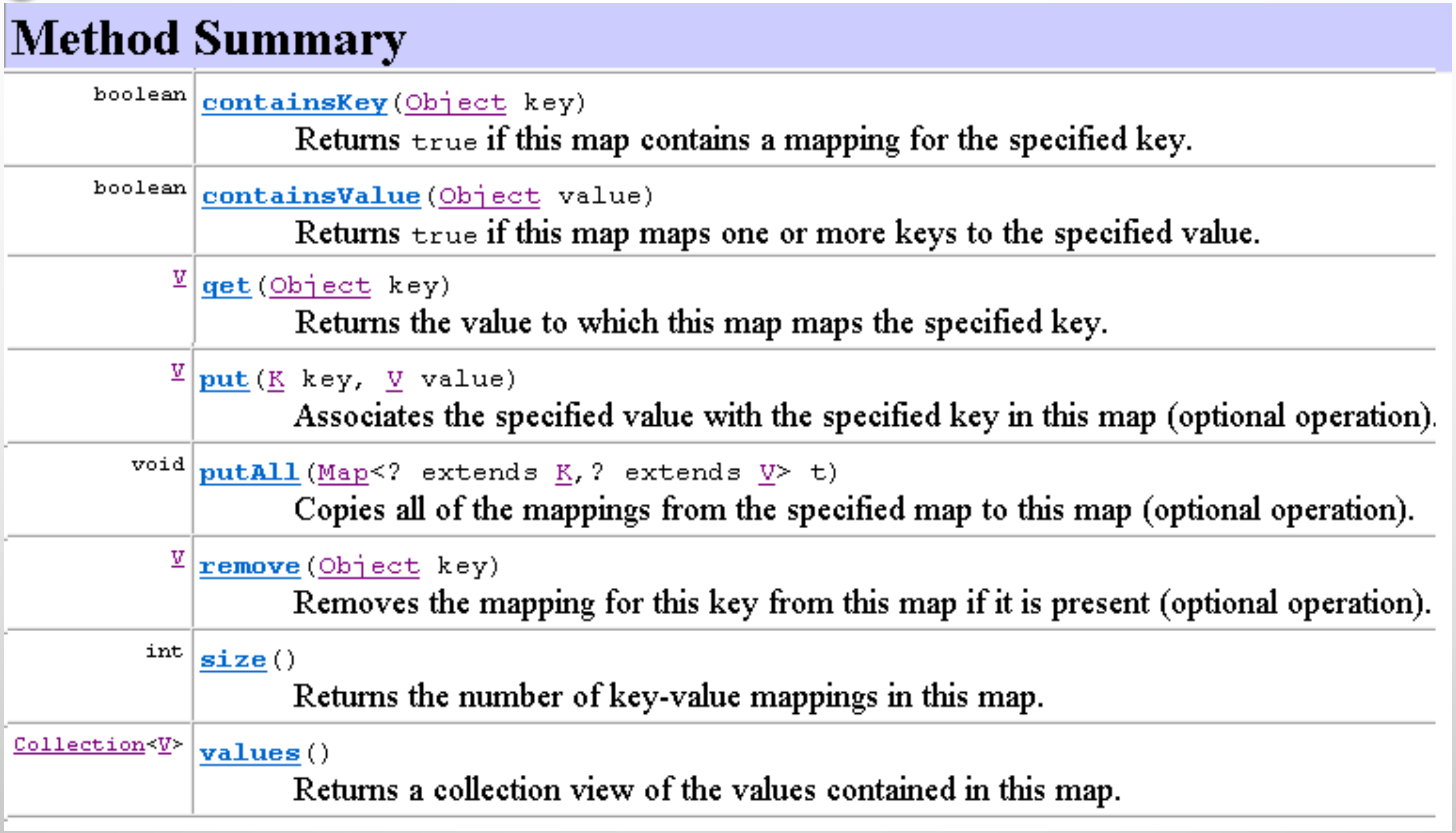
Hackademics Hanoi
Hack your passion, Code your life!

Map

- MAP lưu trữ dữ liệu theo từng cặp: khóa – giá trị (key-value)
- Các giá trị được lấy từ MAP thông qua khóa của nó.
- Các khóa trong MAP phải duy nhất.



Các phương thức của Map

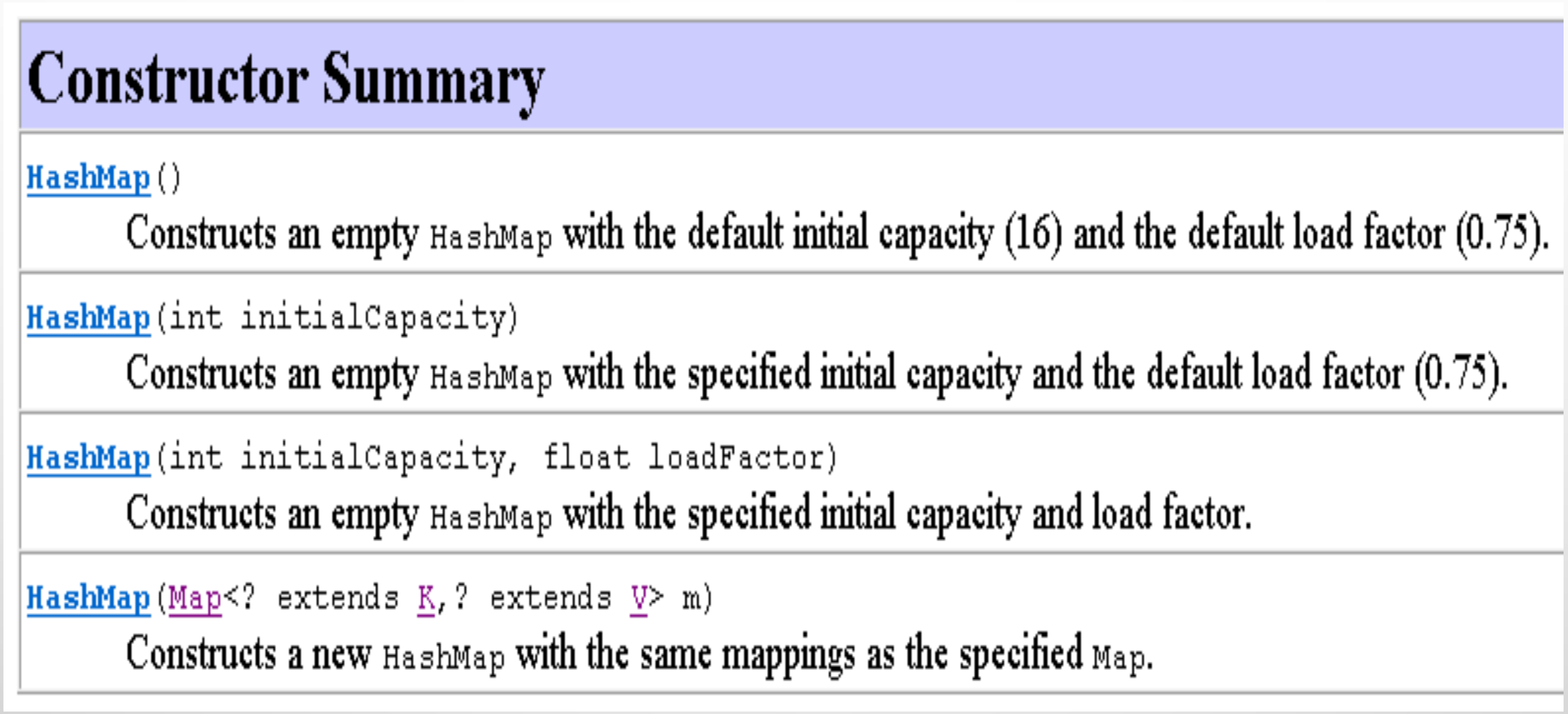


Method Summary

`boolean`	`containsKey(Object key)` Returns `true` if this map contains a mapping for the specified key.
`boolean`	`containsValue(Object value)` Returns `true` if this map maps one or more keys to the specified value.
`V`	`get(Object key)` Returns the value to which this map maps the specified key.
`V`	`put(K key, V value)` Associates the specified value with the specified key in this map (optional operation).
`void`	`putAll(Map<? extends K,? extends V> t)` Copies all of the mappings from the specified map to this map (optional operation).
`V`	`remove(Object key)` Removes the mapping for this key from this map if it is present (optional operation).
`int`	`size()` Returns the number of key-value mappings in this map.
`Collection<V>`	`values()` Returns a collection view of the values contained in this map.

Lớp HashMap

- Thực thi giao diện MAP



Constructor Summary
`HashMap()` Constructs an empty `HashMap` with the default initial capacity (16) and the default load factor (0.75).
`HashMap(int initialCapacity)` Constructs an empty `HashMap` with the specified initial capacity and the default load factor (0.75).
`HashMap(int initialCapacity, float loadFactor)` Constructs an empty `HashMap` with the specified initial capacity and load factor.
`HashMap(Map<? extends K,? extends V> m)` Constructs a new `HashMap` with the same mappings as the specified `Map`.

Hackademics Hanoi
Hack your passion, Code your life!

Ví dụ về HashMap

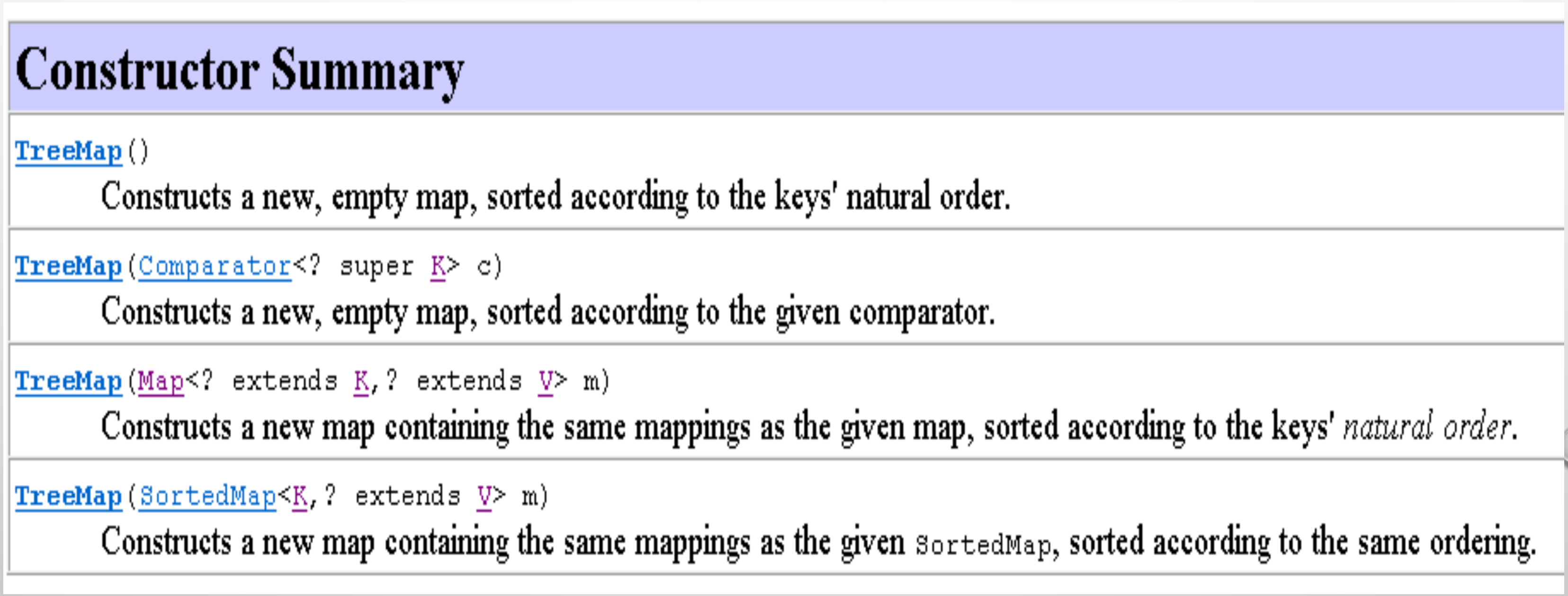
```
public void testHashMap()
{
    HashMap hMap = new HashMap();

    hMap.put("K1", "Hi");
    hMap.put("K2", "Hello");
    hMap.put("K3", "Morning");
    hMap.put("K3", "Bonjour");

    System.out.println("HashMap content:");

    Set keySet = hMap.keySet();
    for (Iterator i = keySet.iterator(); i.hasNext();)
    {
        Object key = i.next();
        System.out.println(key + "- " + hMap.get(key));
    }
}
```

```
HashMap content:
K3- Bonjour
K1- Hi
K2- Hello
```

Hackademics Hanoi
Hack your passion, Code your life!

Lớp TreeMap

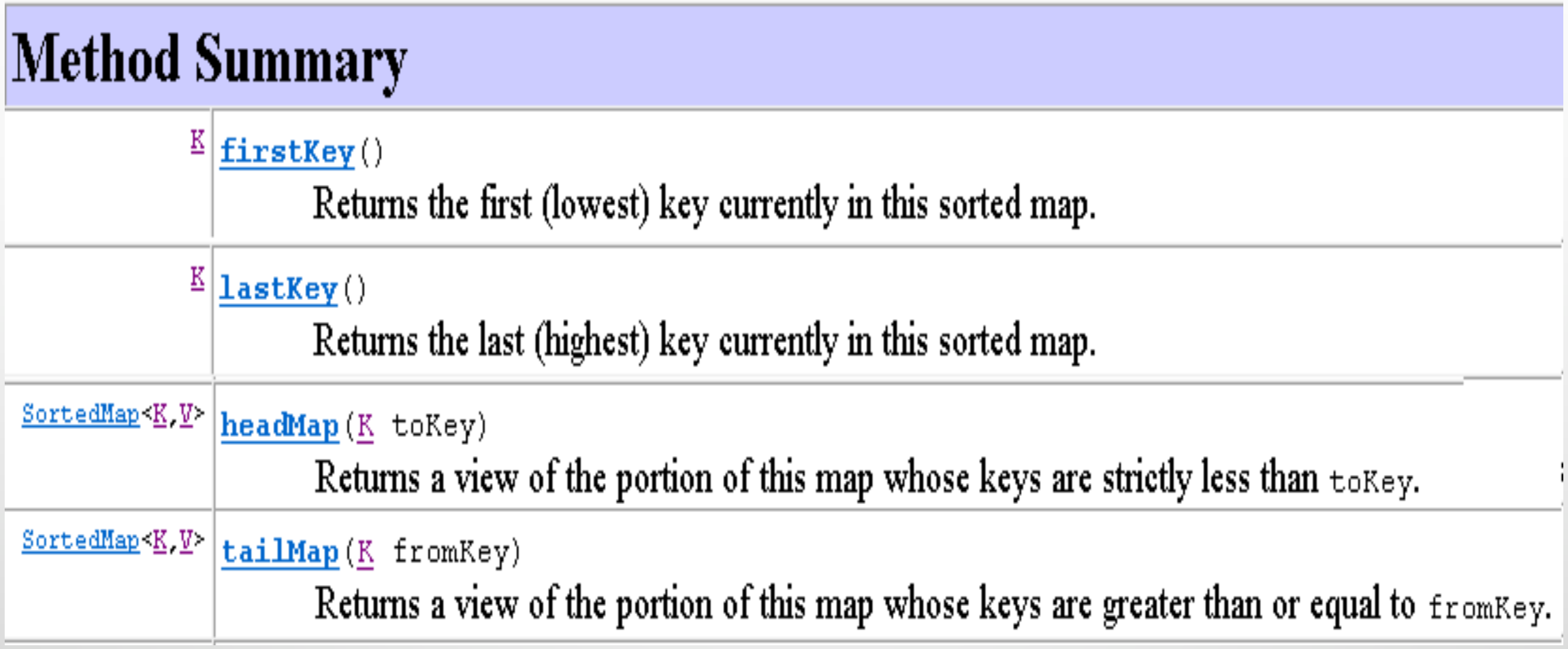
- Lưu trữ các phần tử theo cấu trúc cây
- Các phần tử sắp xếp dựa trên giá trị của khóa.



Constructor Summary
`TreeMap()` Constructs a new, empty map, sorted according to the keys' natural order.
`TreeMap(Comparator<? super K> c)` Constructs a new, empty map, sorted according to the given comparator.
`TreeMap(Map<? extends K,? extends V> m)` Constructs a new map containing the same mappings as the given map, sorted according to the keys' *natural order*.
`TreeMap(SortedMap<K,? extends V> m)` Constructs a new map containing the same mappings as the given `SortedMap`, sorted according to the same ordering.

Hackademics Hanoi
Hack your passion, Code your life!

Các phương thức của TreeMap



Method Summary	
K	firstKey() Returns the first (lowest) key currently in this sorted map.
K	lastKey() Returns the last (highest) key currently in this sorted map.
SortedMap<K,V>	headMap(K toKey) Returns a view of the portion of this map whose keys are strictly less than toKey.
SortedMap<K,V>	tailMap(K fromKey) Returns a view of the portion of this map whose keys are greater than or equal to fromKey.

Hackademics Hanoi
Hack your passion, Code your life!

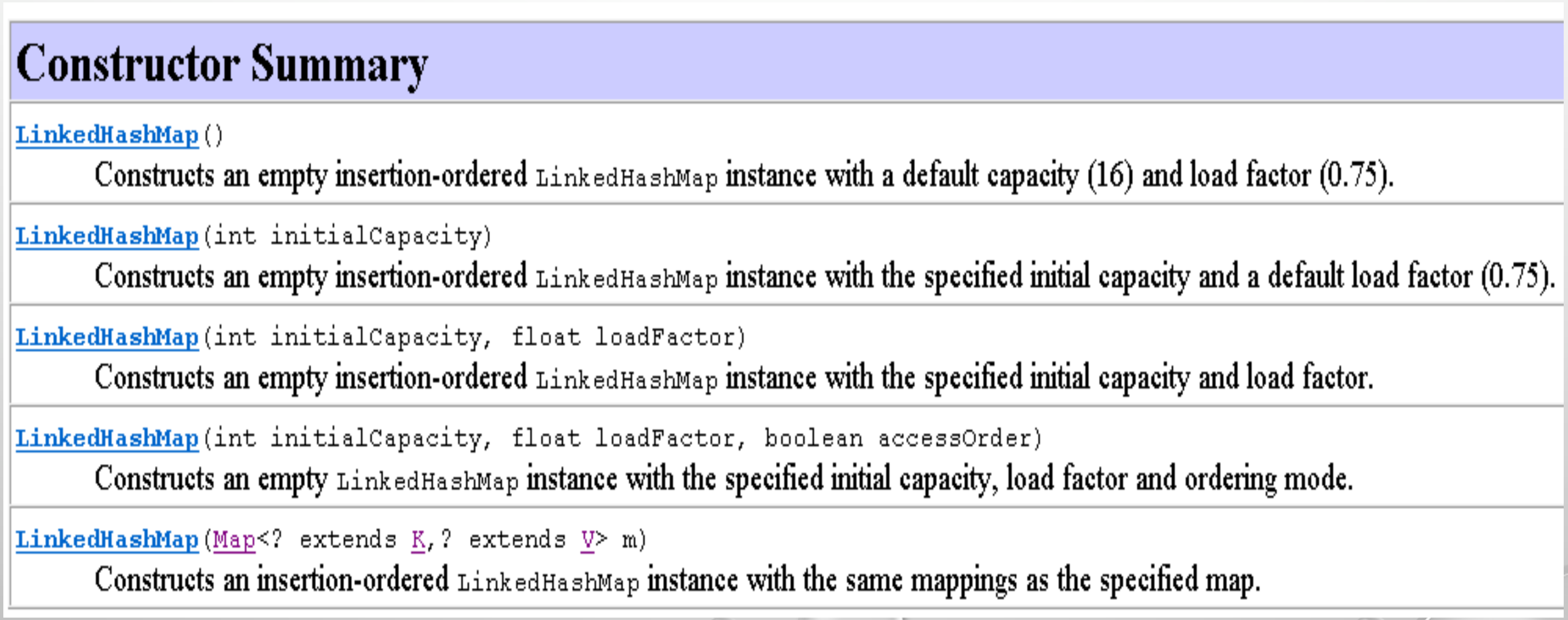
Ví dụ “TreeMap"

```
public void testTreeMap()
{
    TreeMap treeMap = new TreeMap();
    treeMap.put("101", "Hello");
    treeMap.put("102", "Hi");
    treeMap.put("103", "Morning");
    treeMap.put("104", "Bonjour");

    // Get first element
    Object fkey = treeMap.firstKey();
    System.out.println("First element: " + treeMap.get(fkey));

    // Get last element
    Object lkey = treeMap.lastKey();
    System.out.println("Last element: " + treeMap.get(lkey));

    System.out.println("Elements before key 103");
    SortedMap smap = treeMap.headMap("103");
    Set hMapKeys = smap.keySet();
    for (Iterator i = hMapKeys.iterator(); i.hasNext();)
    {
        Object key = (Object) i.next();
        System.out.println(smap.get(key));
    }
}
```

```
First element: Hello
Last element: Bonjour
Elements before key 103
Hello
Hi
```

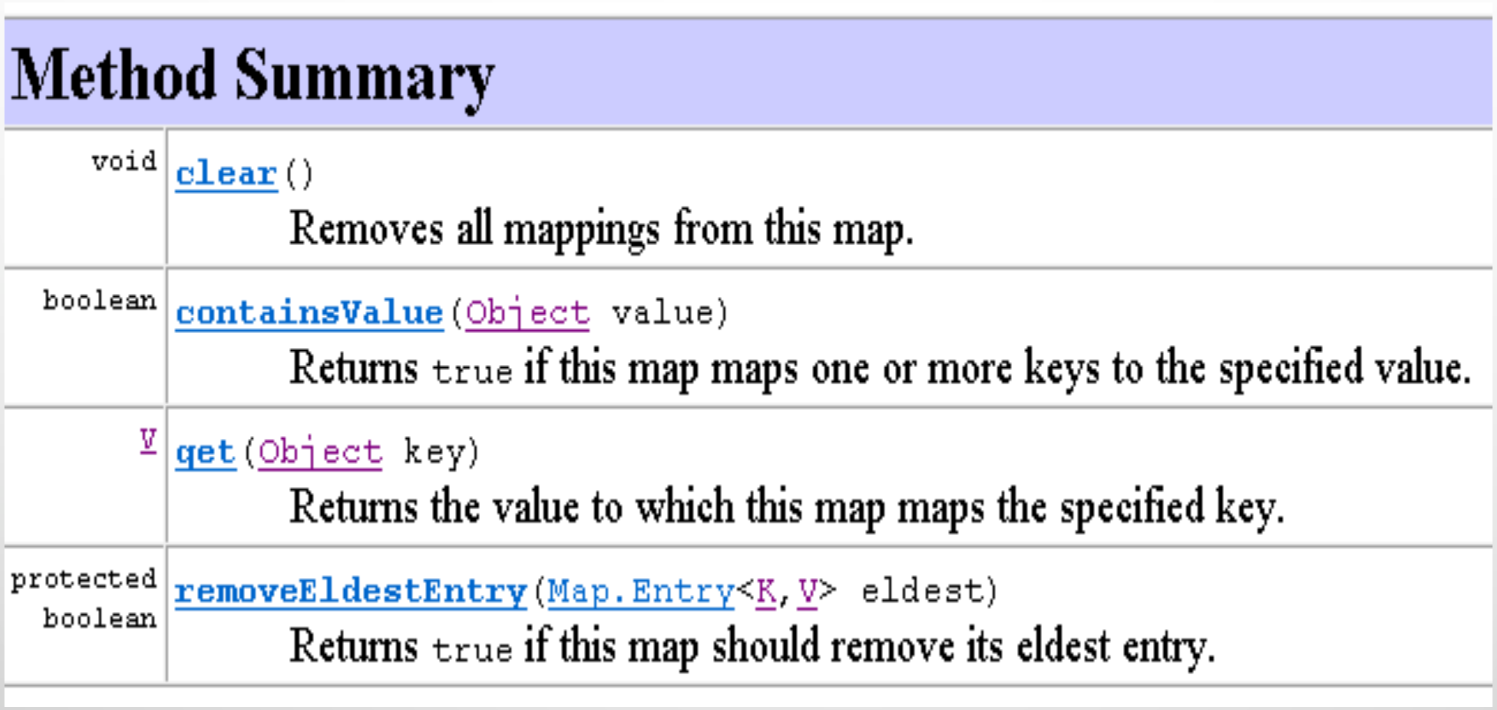
Lớp “LinkedHashMap"

- Các phần tử trong tập hợp được duy trì thứ tự như khi chúng được thêm vào



Constructor Summary
`LinkedHashMap()` Constructs an empty insertion-ordered `LinkedHashMap` instance with a default capacity (16) and load factor (0.75).
`LinkedHashMap(int initialCapacity)` Constructs an empty insertion-ordered `LinkedHashMap` instance with the specified initial capacity and a default load factor (0.75).
`LinkedHashMap(int initialCapacity, float loadFactor)` Constructs an empty insertion-ordered `LinkedHashMap` instance with the specified initial capacity and load factor.
`LinkedHashMap(int initialCapacity, float loadFactor, boolean accessOrder)` Constructs an empty `LinkedHashMap` instance with the specified initial capacity, load factor and ordering mode.
`LinkedHashMap(Map<? extends K,? extends V> m)` Constructs an insertion-ordered `LinkedHashMap` instance with the same mappings as the specified map.

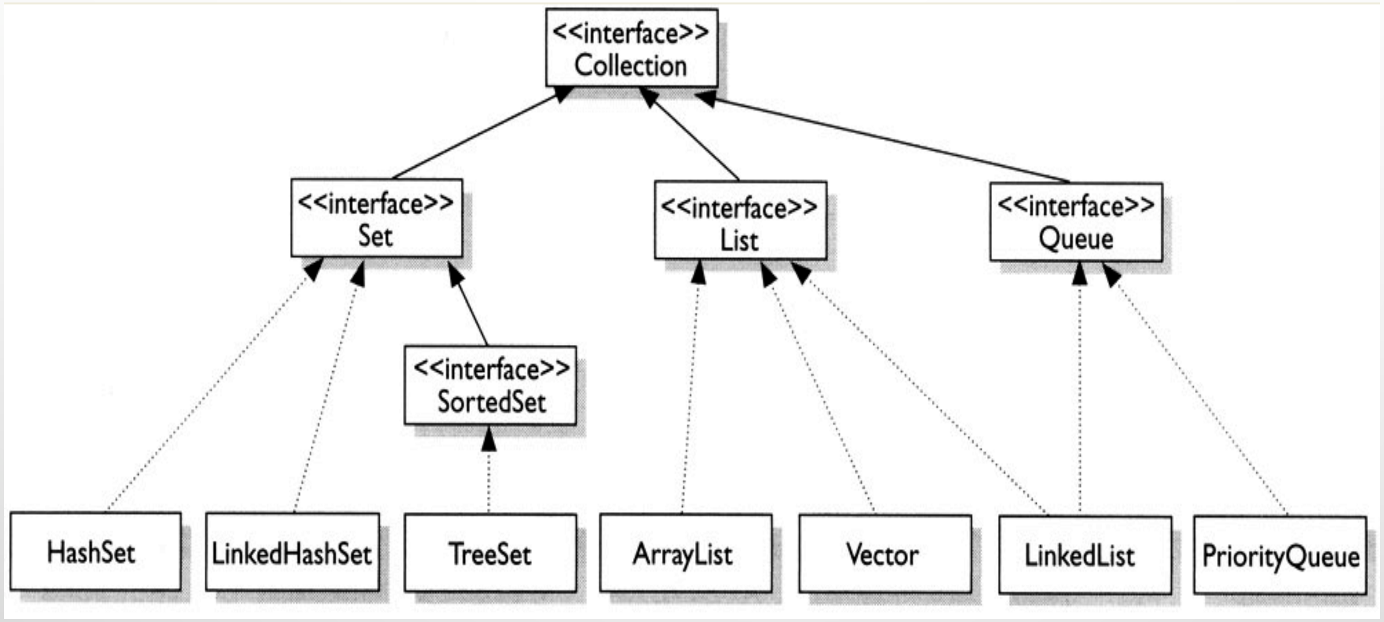
Các phương thức của LinkedHashMap



Method Summary

`void`	`clear()` Removes all mappings from this map.
`boolean`	`containsValue(Object value)` Returns `true` if this map maps one or more keys to the specified value.
`V`	`get(Object key)` Returns the value to which this map maps the specified key.
`protected boolean`	`removeEldestEntry(Map.Entry<K,V> eldest)` Returns `true` if this map should remove its eldest entry.

Hàng đợi (Queues) và Mảng(Arrays)

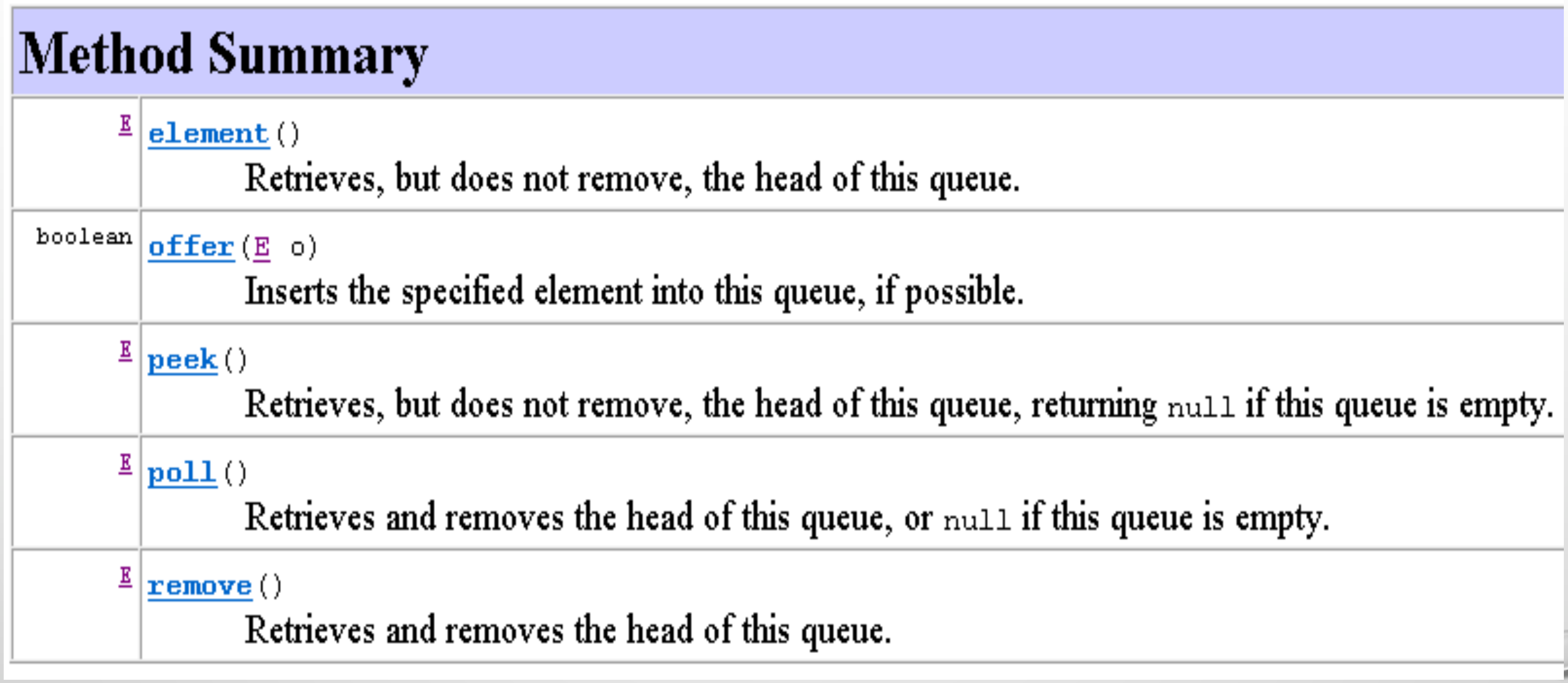


Giao diện QUEUE

- Queue: Các phần tử được truy xuất theo thứ tự First In First Out (FIFO).

- Priority queue(hàng đợi ưu tiên)Thứ tự truy xuất các phần tử phụ thuộc vào giá trị của chúng.

Các phương thức của Queue



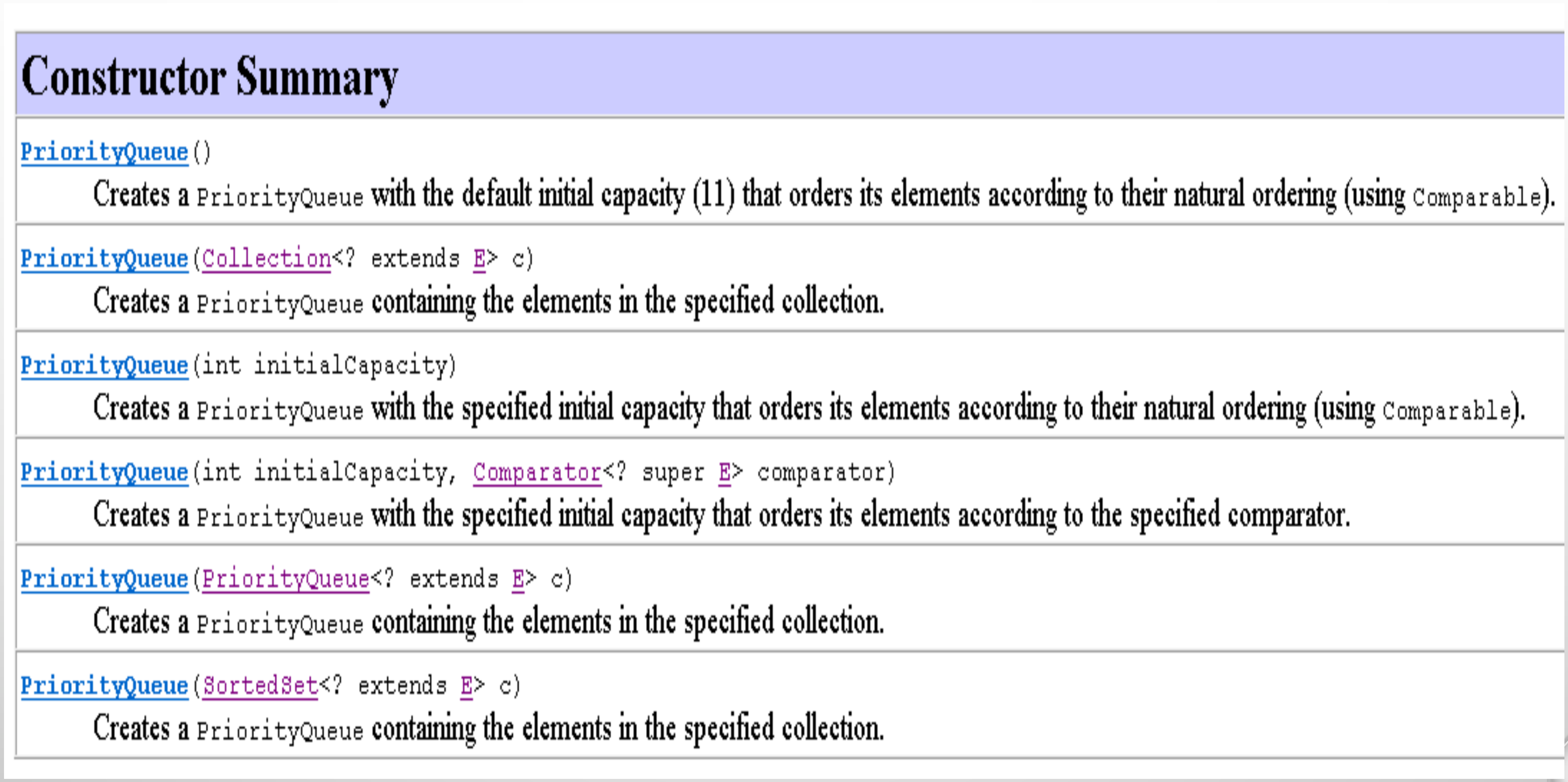
Method Summary	
E	element() Retrieves, but does not remove, the head of this queue.
boolean	offer(E o) Inserts the specified element into this queue, if possible.
E	peek() Retrieves, but does not remove, the head of this queue, returning null if this queue is empty.
E	poll() Retrieves and removes the head of this queue, or null if this queue is empty.
E	remove() Retrieves and removes the head of this queue.

Lớp PriorityQueue

- Các phần tử được sắp xếp theo thứ tự tự nhiên hoặc dựa vào một comparator.

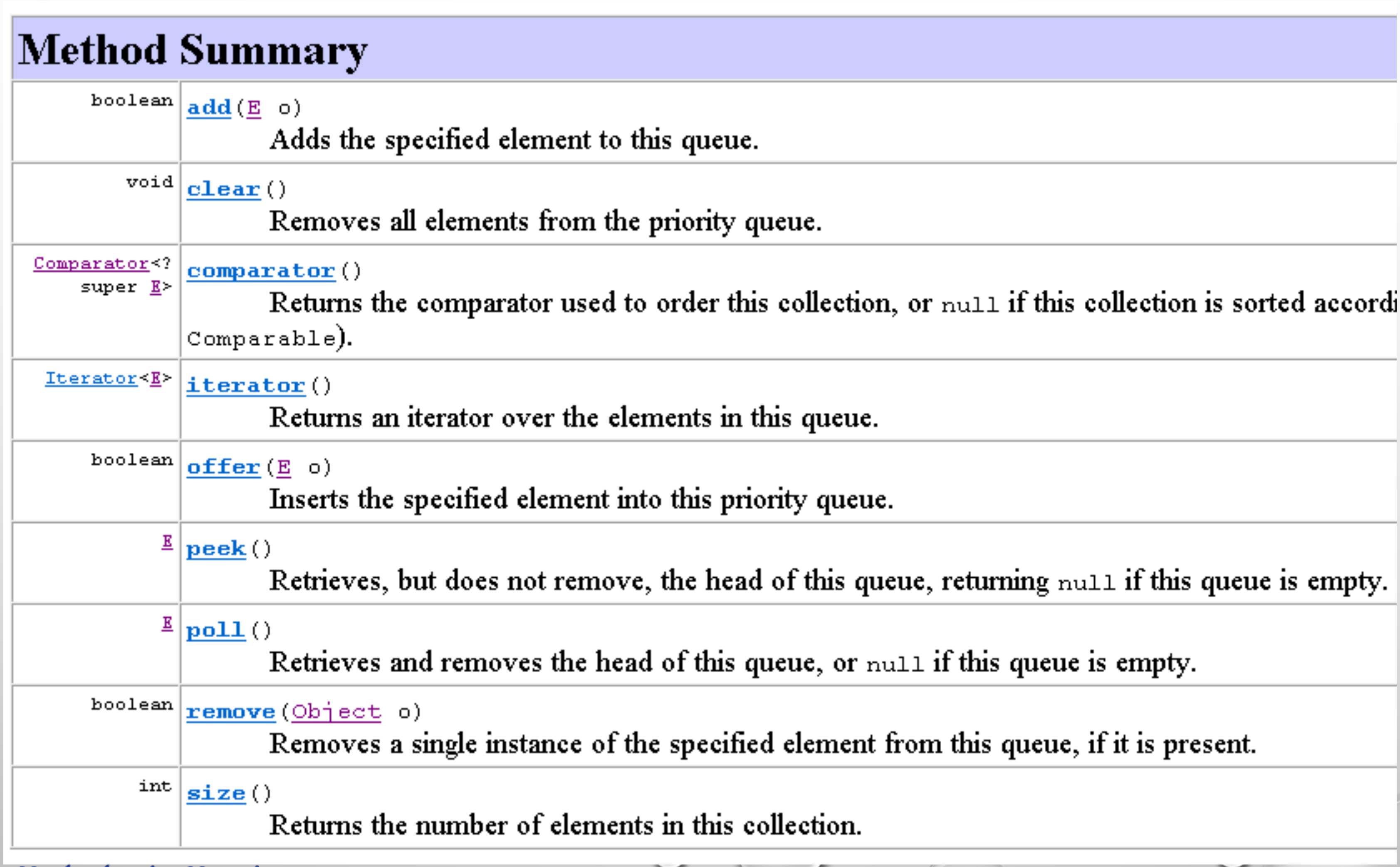
- Không chấp nhận phần tử có giá trị null.

Các Constructor của PriorityQueue



Constructor Summary
`PriorityQueue()` Creates a `PriorityQueue` with the default initial capacity (11) that orders its elements according to their natural ordering (using `Comparable`).
`PriorityQueue(Collection<? extends E> c)` Creates a `PriorityQueue` containing the elements in the specified collection.
`PriorityQueue(int initialCapacity)` Creates a `PriorityQueue` with the specified initial capacity that orders its elements according to their natural ordering (using `Comparable`).
`PriorityQueue(int initialCapacity, Comparator<? super E> comparator)` Creates a `PriorityQueue` with the specified initial capacity that orders its elements according to the specified comparator.
`PriorityQueue(PriorityQueue<? extends E> c)` Creates a `PriorityQueue` containing the elements in the specified collection.
`PriorityQueue(SortedSet<? extends E> c)` Creates a `PriorityQueue` containing the elements in the specified collection.

Các phương thức của PriorityQueue



Method Summary

boolean	add(E o) Adds the specified element to this queue.
void	clear() Removes all elements from the priority queue.
Comparator<? super E>	comparator() Returns the comparator used to order this collection, or null if this collection is sorted accordi Comparable).
Iterator<E>	iterator() Returns an iterator over the elements in this queue.
boolean	offer(E o) Inserts the specified element into this priority queue.
E	peek() Retrieves, but does not remove, the head of this queue, returning null if this queue is empty.
E	poll() Retrieves and removes the head of this queue, or null if this queue is empty.
boolean	remove(Object o) Removes a single instance of the specified element from this queue, if it is present.
int	size() Returns the number of elements in this collection.

Hackademics Hanoi
Hack your passion, Code your life!

Ví dụ về PriorityQueue

```java
public void testPriorityQueue()
{
    PriorityQueue pQueue = new PriorityQueue();

    pQueue.offer("Hello");
    pQueue.offer("Bonjour");
    pQueue.offer("Konichiowa");
    pQueue.offer("Abc");

    System.out.println("1. Comparator: " + pQueue.comparator());
    System.out.println("2. Content of Priority Queue");
    for (Iterator i = pQueue.iterator(); i.hasNext();)
    {
        System.out.print(i.next() + " - ");
    }
    System.out.println("");
    System.out.println("3. Retrieve and remove head element: " + pQueue.poll());

    System.out.println("4. Now, content of Priority Queue is:");
    for (Iterator i = pQueue.iterator(); i.hasNext();)
    {
        System.out.print(i.next() + " - ");
    }
}
```

Output

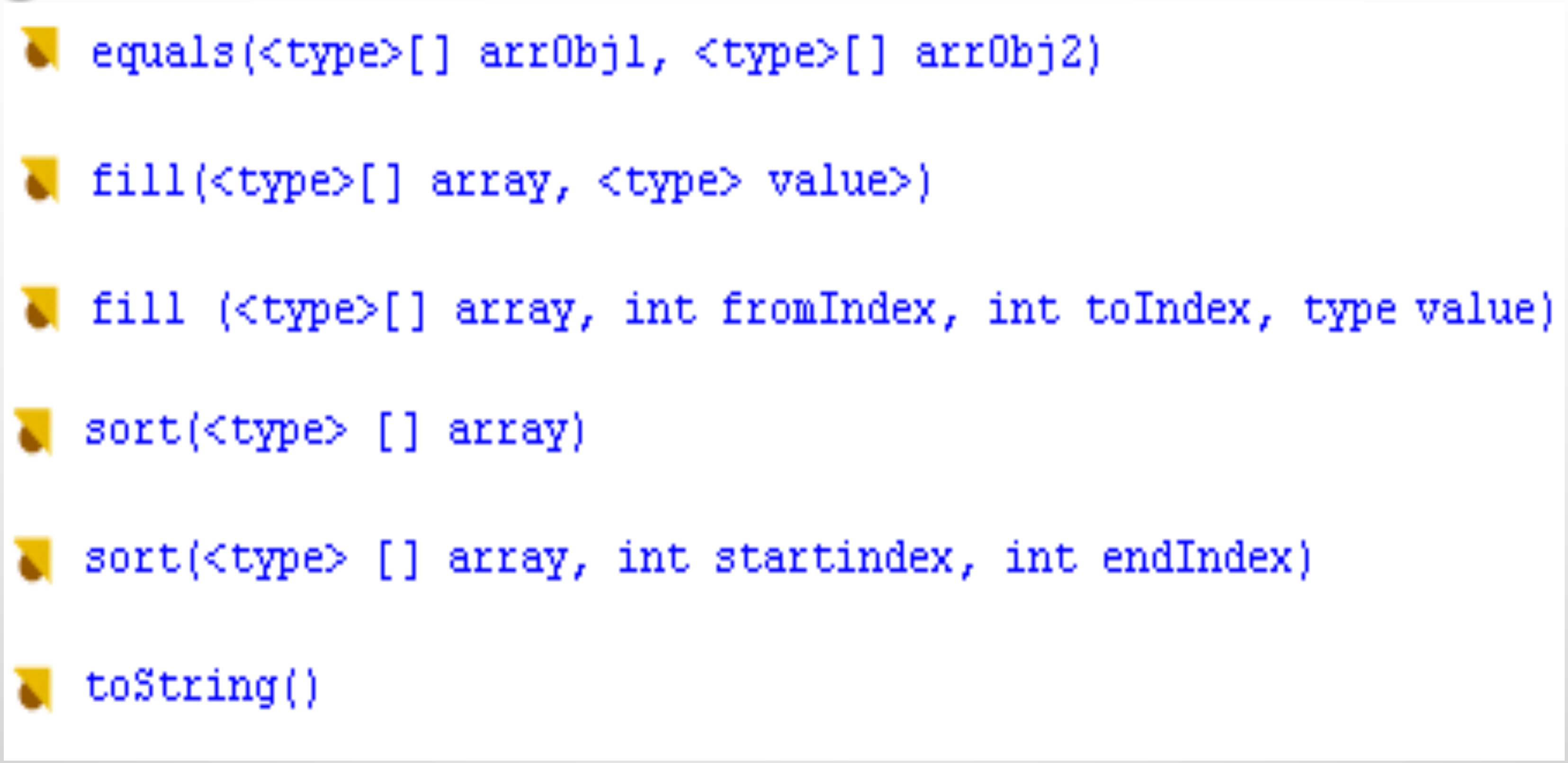
```
1. Comparator: null
2. Content of Priority Queue
Abc - Bonjour - Konichiowa - Hello -
3. Retrieve and remove head element: Abc
4. Now, content of Priority Queue is:
Bonjour - Hello - Konichiowa -
```

Hackademics Hanoi
Hack your passion, Code your life!

Lớp Arrays

- Chứa các phương thức cho phép thao tác trên mảng (sorting, searching)
-

Các phương thức của lớp Arrays



```
equals(<type>[] arrObj1, <type>[] arrObj2)

fill(<type>[] array, <type> value>)

fill (<type>[] array, int fromIndex, int toIndex, type value)

sort(<type> [] array)

sort(<type> [] array, int startindex, int endIndex)

toString()
```

Ví dụ Arrays

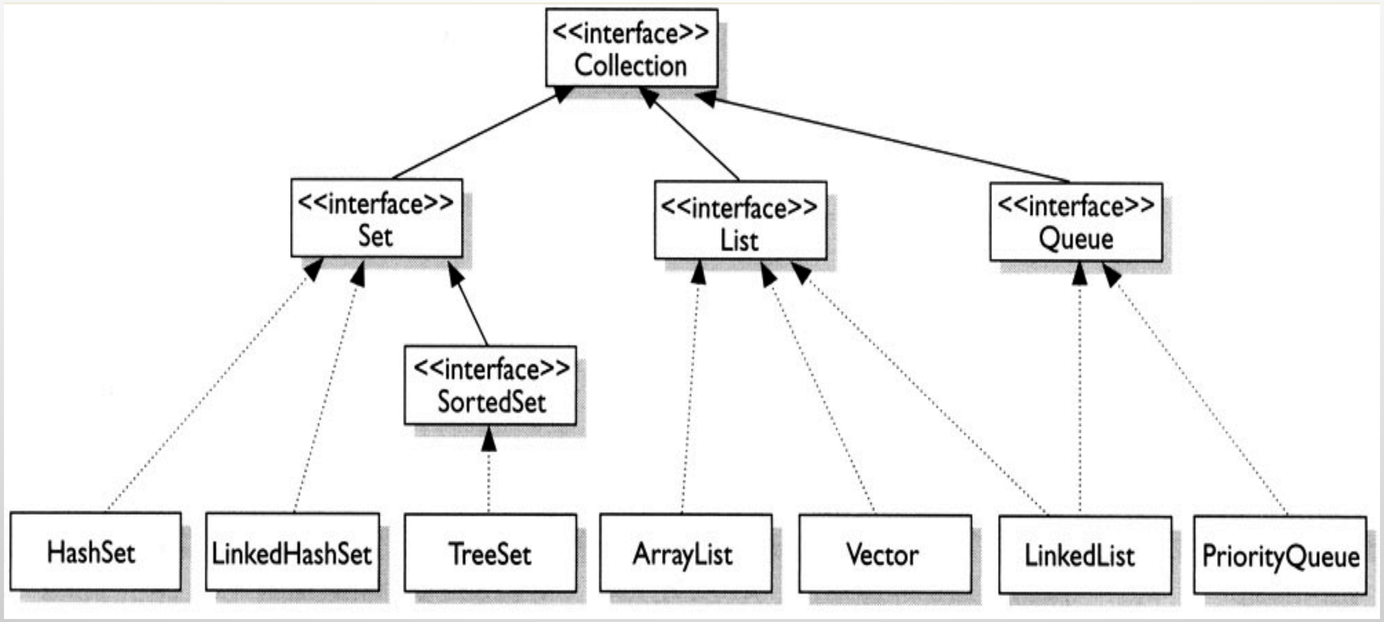
```java
public void testArrays()
{
    int a[] = new int[3];
    a[0] = 9;
    a[1] = 6;
    a[2] = 3;

    Arrays.sort(a);

    System.out.println("Array after sorted:");
    for (int i = 0; i < a.length; i++)
    {
        System.out.println(a[i]);
    }
}
```

Output

```
Array after sorted:
3
6
9
```

THAT'S ABOUT ALL FOR TODAY!

- "java.util" Package
- List Classes and Interfaces
- Set Classes and Interfaces
- **Map Classes and Interfaces**
- **Queues and Arrays**



Hackademics Hanoi
Hack your passion, Code your life!

Hackademics Hanoi
Hack your passion, Code your life!

HẾT

CHƯƠNG 12