

Chương 5

TẬP HỢP TRÊN JAVA

Mục tiêu



- Phân biệt tập hợp và mảng
- Phân biệt các đặc trưng của các Collection interface
- Chọn loại tập hợp thích hợp để giải quyết bài toán

Nội dung



- 5.1. Khái niệm về tập hợp
- 5.2. So sánh tập hợp và mảng
- 5.3. Các lớp tập hợp trong Java
- 5.4. Case study

5.1. Khái niệm về tập hợp



- Tập hợp là đối tượng có khả năng chứa các đối tượng khác
- Các thao tác thông thường trên tập hợp
 - Thêm đối tượng vào tập hợp
 - Xoá đối tượng khỏi tập hợp
 - Kiểm tra một đối tượng có tồn tại trong tập hợp hay không
 - Lấy một đối tượng từ tập hợp
 - Duyệt các đối tượng trong tập hợp
 - Xoá toàn bộ tập hợp
 - ...

5.1. Khái niệm về tập hợp

Collections Framework



- Collections Framework (từ Java 1.2)
 - Là một kiến trúc hợp nhất để biểu diễn và thao tác trên các loại tập hợp
 - Giúp cho việc xử lý tập hợp độc lập với biểu diễn chi tiết bên trong
- Một số lợi ích của Collections Framework
 - Giảm thời gian lập trình
 - Tăng cường hiệu năng chương trình
 - Dễ mở rộng các collection mới
 - Sử dụng lại mã chương trình

5.1. Khái niệm về tập hợp

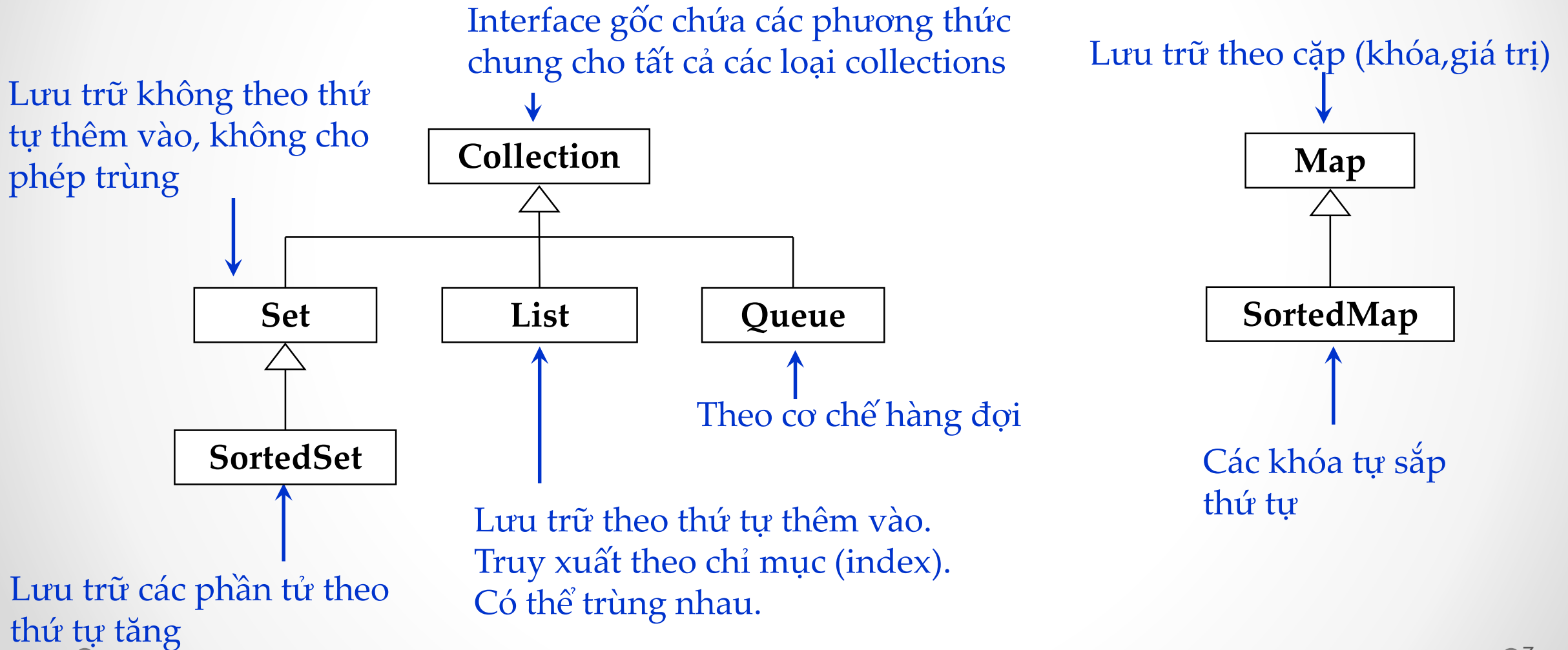
Collections Framework



- Collections Framework được cung cấp từ gói `java.util`, bao gồm:
 - **Interfaces**: Là các interface thể hiện tính chất của các kiểu collection khác nhau như **List**, **Set**, **Map**
 - **Implementations**: Là các lớp collection có sẵn được cài đặt các collection interfaces như **LinkedList**, **HashSet**,...
 - **Algorithms**: Là các phương thức tĩnh để xử lý collection, ví dụ: sắp xếp danh sách, tìm phần tử lớn nhất...

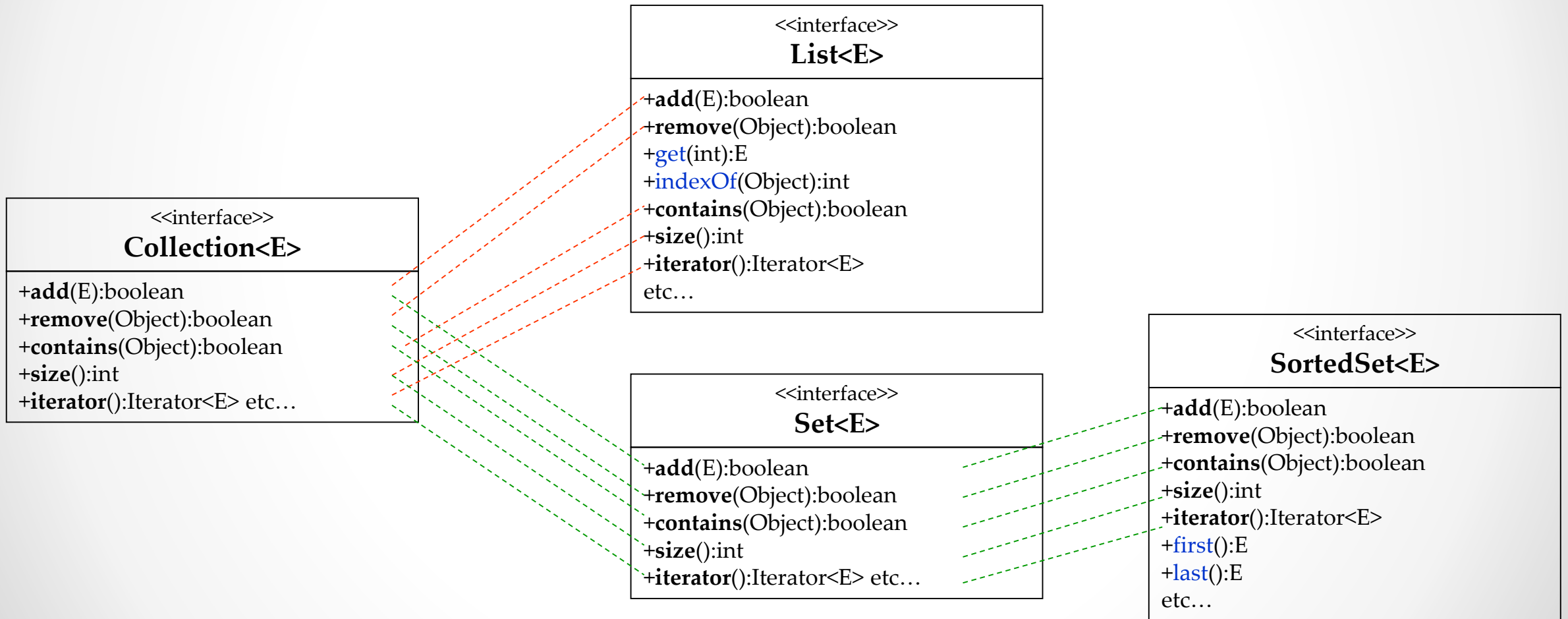
5.1. Khái niệm về tập hợp

Collection và Map interface



5.1. Khái niệm về tập hợp

So sánh một số interface



5.1. Khái niệm về tập hợp

Các phương thức của Collection interface



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

5.1. Khái niệm về tập hợp

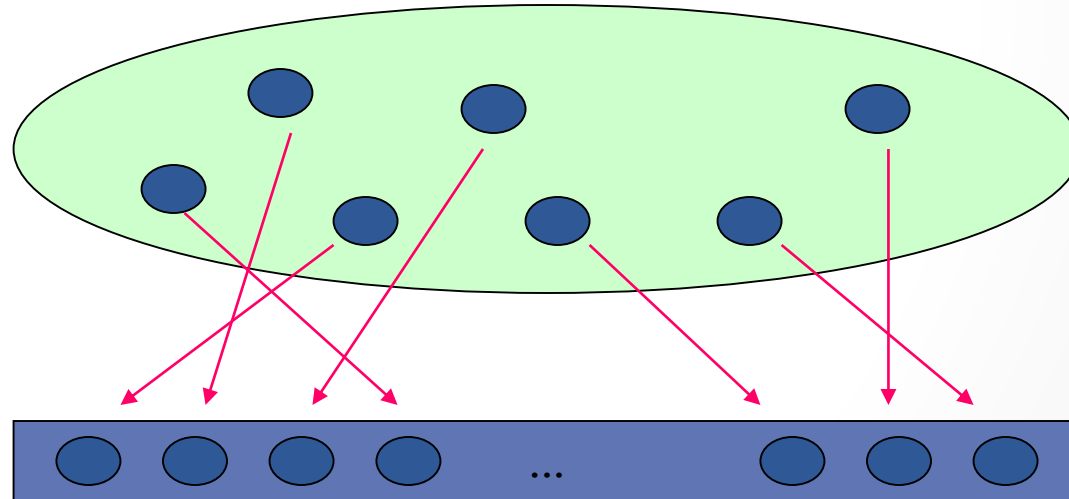
Duyệt collection



- Các phần tử trong collection có thể được duyệt tuần tự thông qua **Iterator** interface
- Các lớp cài đặt Collection cung cấp phương thức trả về Iterator trên các phần tử của chúng

Collection c;

Iterator it = c.**iterator**();



5.1. Khái niệm về tập hợp

Duyệt collection



- Các phương thức của Iterator:
 - boolean **hasNext()**: trả về *true* nếu còn phần tử chưa duyệt
 - Object **next()**: trả về phần tử hiện tại và di chuyển con trỏ tới phần tử tiếp theo
 - void **remove()**: xóa phần tử đang duyệt
- Cách sử dụng:

```
Iterator<Item> it = items.iterator();
while(it.hasNext()) {
    Item item = it.next();
    System.out.println(item);
}
```

=

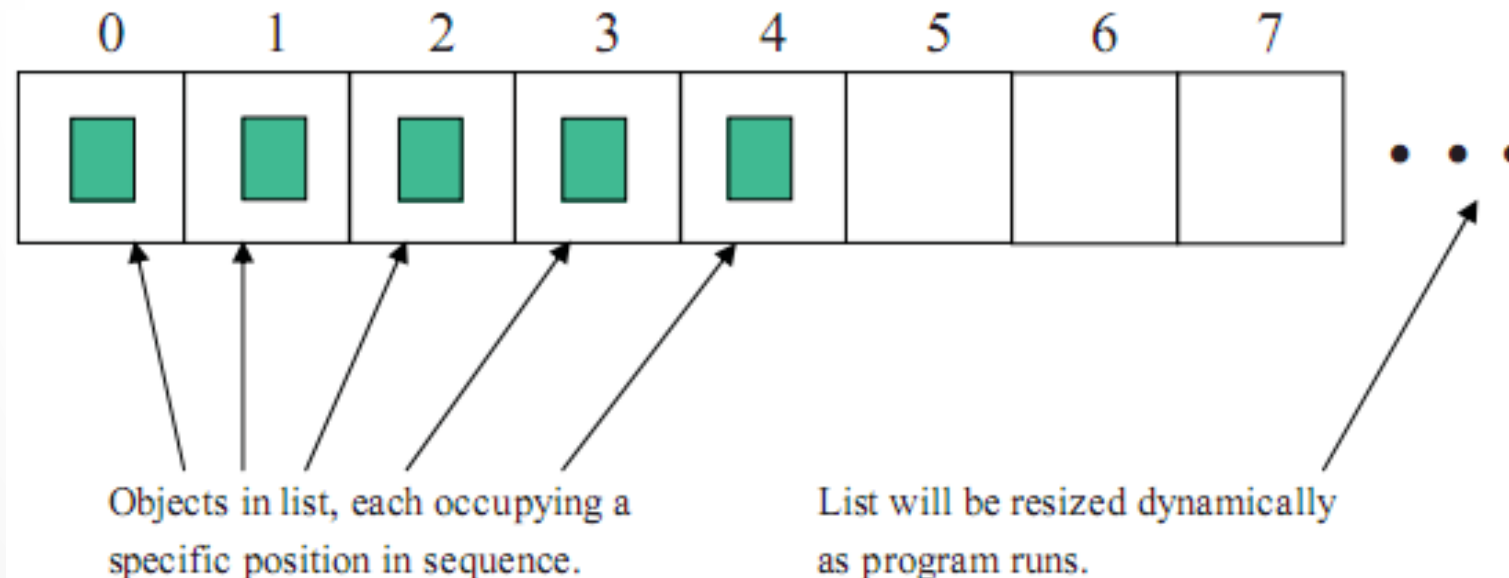
```
for (Item item : items) {
    System.out.println(item);
}
```

5.1. Khái niệm về tập hợp

List interface



- Giống như array nhưng linh hoạt hơn, vì có thể tự điều chỉnh kích thước khi thêm dữ liệu
- Lưu trữ dữ liệu theo thứ tự thêm vào, có thể truy xuất theo chỉ mục (index)

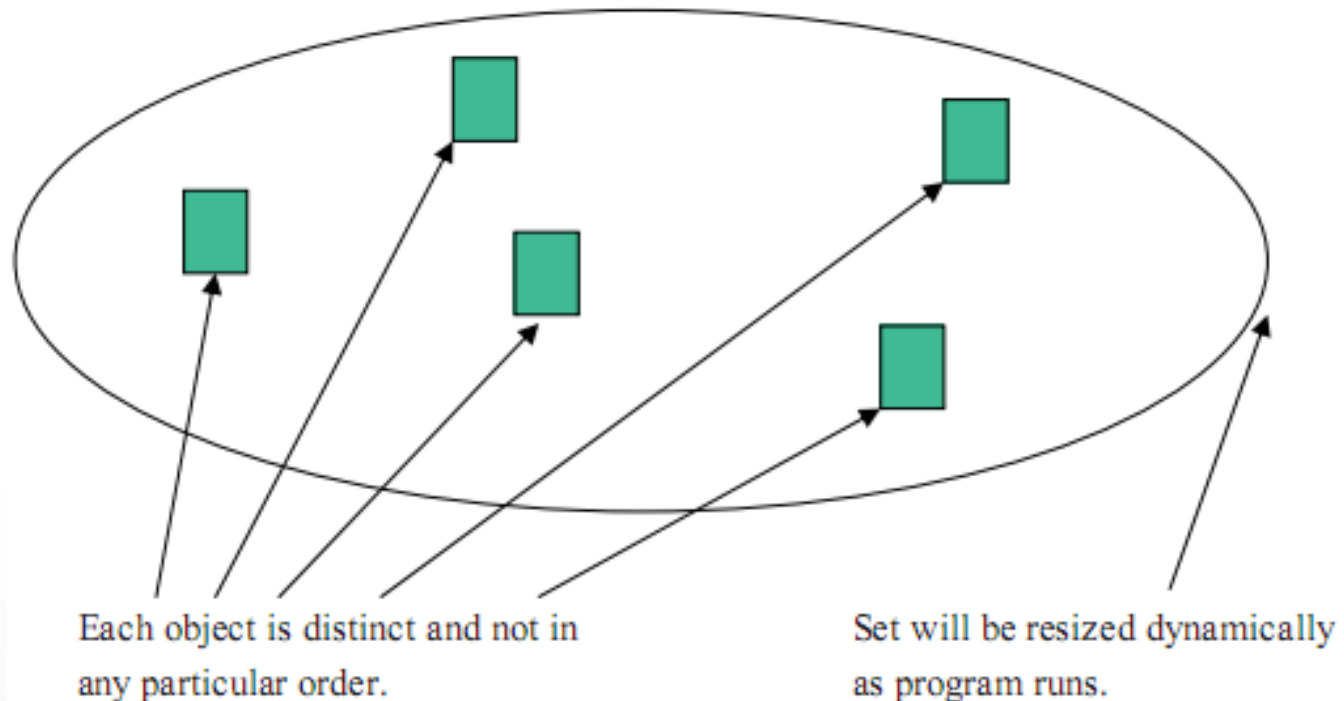


5.1. Khái niệm về tập hợp

Set interface



- Giống như “bag” hơn là “list”
- Các phần tử lưu trong Set không được trùng và không quan tâm thứ tự thêm vào



5.1. Khái niệm về tập hợp

SortedSet interface



- Hỗ trợ thao tác trên tập hợp các phần tử có thể so sánh được
- Các đối tượng đưa vào SortedSet phải có khả năng so sánh được, tức là phải implements **Comparable** interface HOẶC lớp cài đặt SortedSet phải nhận một **Comparator** trên đối tượng khoá

5.1. Khái niệm về tập hợp

Queue interface



- Các phần tử được truy xuất theo thứ tự FIFO
- Các phương thức của Queue

Method Summary

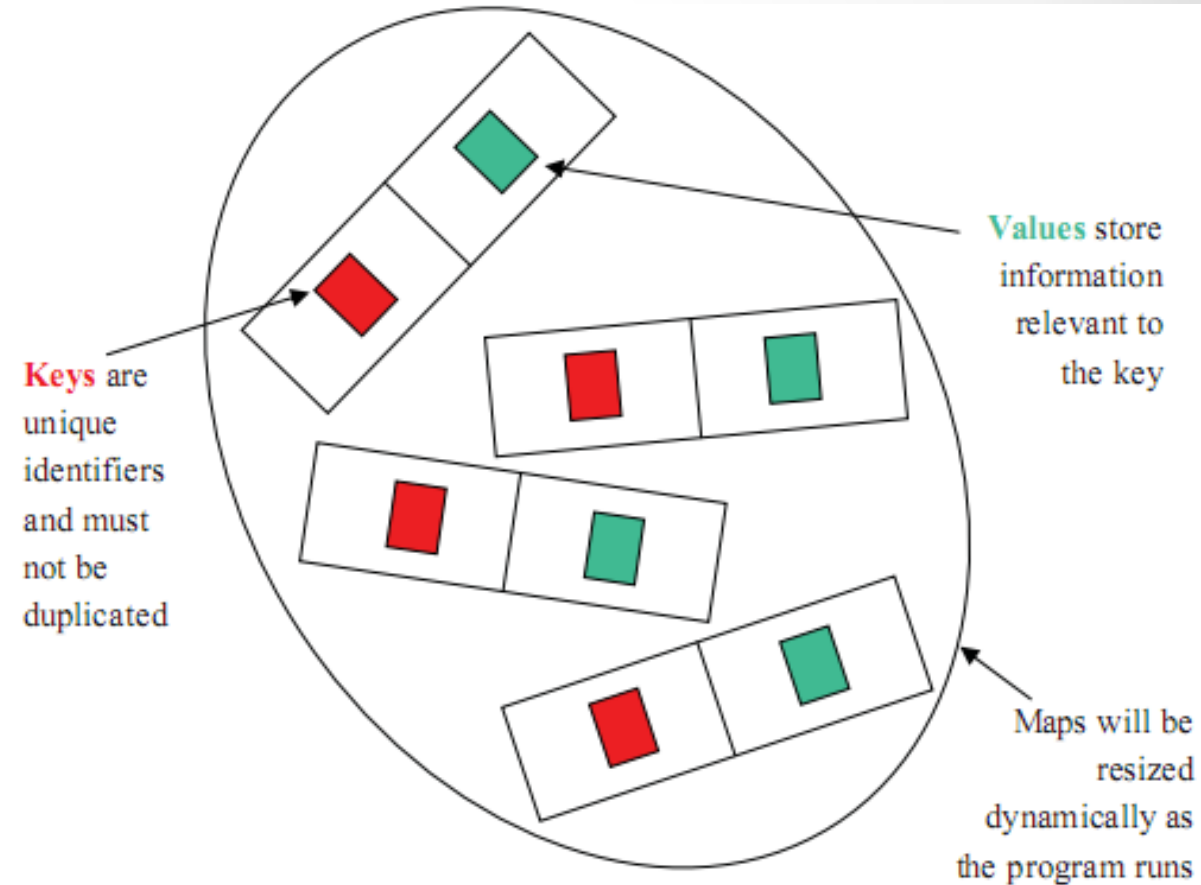
<u>E</u>	<u>element</u> () Retrieves, but does not remove, the head of this queue.
boolean	<u>offer</u> (<u>E</u> o) Inserts the specified element into this queue, if possible.
<u>E</u>	<u>peek</u> () Retrieves, but does not remove, the head of this queue, returning <code>null</code> if this queue is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

5.1. Khái niệm về tập hợp

Map interface



- 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
- Giống như Set, các phần tử trong Map không có thứ tự



5.1. Khái niệm về tập hợp

Map interface (tt)



- Hai phương thức truy cập:
 - V **put**(K key, V value) // Thêm cặp key-value vào map
 - V **get**(Object key) // Trả về value tương ứng với key cho trước
- Ba cách xem dữ liệu:
 - Lấy tất cả các khoá:
 - Set **keySet**(); // Trả về các khoá
 - Lấy tất cả các giá trị:
 - Collection **values**(); // Trả về các giá trị
 - Lấy tất cả các cặp khoá-giá trị
 - Set **entrySet**(); // Trả về các cặp khoá-giá trị

5.1. Khái niệm về tập hợp

SortedMap interface



- Kế thừa từ Map, cung cấp thao tác trên các bảng ánh xạ với khoá có thể so sánh được
- Giống như SortedSet, các đối tượng khoá đưa vào trong SortedMap phải implements interface **Comparable** HOẶC lớp cài đặt SortedMap phải nhận một **Comparator** trên đối tượng khoá



5.2. So sánh tập hợp và mảng

Mảng	Tập hợp
Có kích thước cố định	Kích thước có tính “tự giãn nở”
Duyệt các phần tử thông qua chỉ số mảng	Có thể duyệt các phần tử thông qua Iterator
Phải lập trình hoàn toàn	Gọi những phương thức đã được định nghĩa



Review questions 1 [1/2]

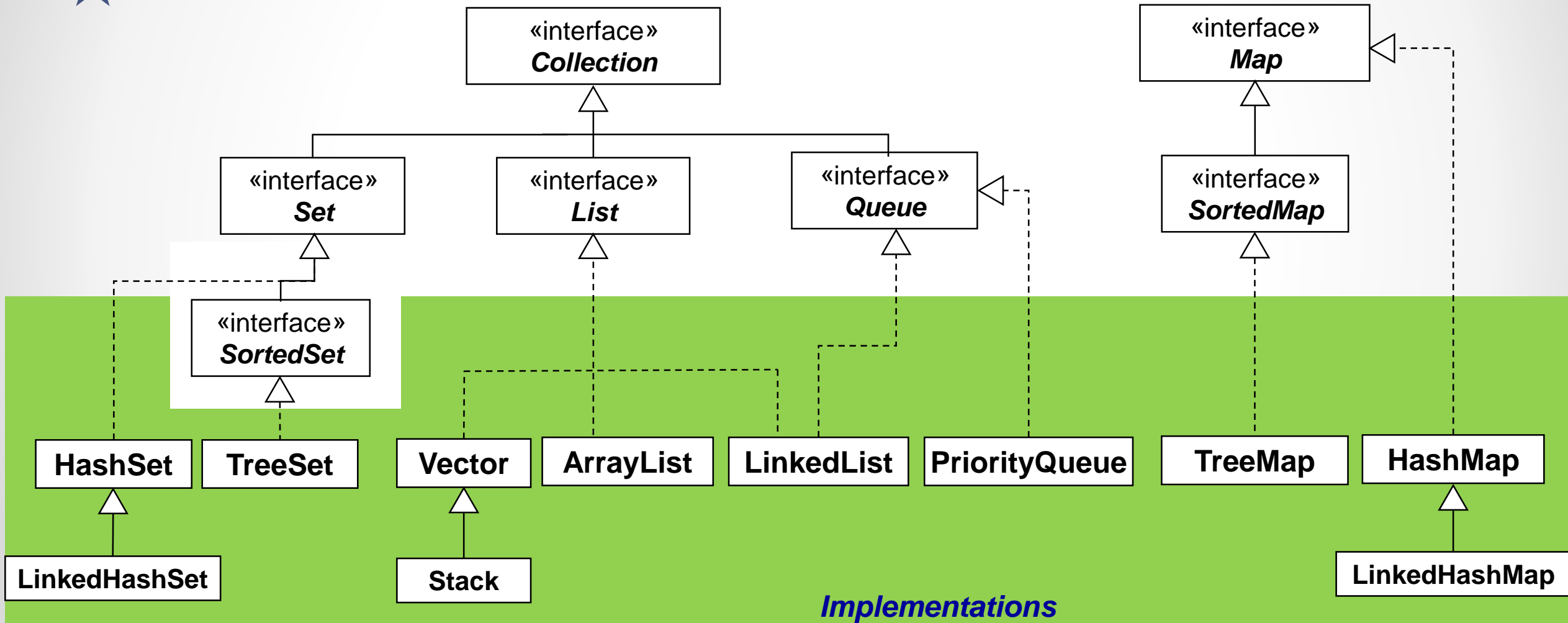
1. List và Set khác nhau ở điểm đặc trưng nào.
2. Cần chú ý gì khi sử dụng SortedSet, SortedMap.
3. Các phần tử trong SortedSet, SortedMap được tự động sắp theo thứ tự tăng, dựa vào tiêu chí nào.
4. Làm thế nào để duyệt các phần tử trong tập hợp.
5. Mỗi trường hợp sau đây nên dùng collection nào (List, Set, Map,...):
 - a) Ghi nhận các thành viên của câu lạc bộ
 - b) Ghi nhận các khoản tiền đã chi
 - c) Ghi nhận chi tiết thông tin tài khoản ngân hàng, với mỗi tài khoản có một số hiệu duy nhất



Review questions 1 [2/2]

5. (tt) Mỗi trường hợp sau đây nên dùng collection nào (List, Set, Map,...):
- d) Công ty XYZ cần lưu tên của các nhân viên của mình. Mỗi tháng một nhân viên sẽ được chọn ngẫu nhiên để nhận một quà tặng.
 - e) Công ty XYZ cần đặt tên cho sản phẩm mới, tên sản phẩm được chọn từ tên của nhân viên, và tên các sản phẩm không được trùng nhau. Chọn loại collection để lưu danh sách tên sản phẩm.
 - f) Công ty XYZ muốn cho nhân viên đi du lịch, chính sách được tạo ra là ưu tiên cho những người đăng ký trước. Chọn loại collection để lưu ds nhân viên đăng ký du lịch.
 - g) Công ty XYZ muốn tạo danh sách các khách hàng theo thứ tự tăng dần theo doanh số.

5.3. Các lớp tập hợp trong Java



5.3. Các lớp tập hợp trong Java

Lớp ArrayList



- Thực thi List interface
- Phù hợp khi cần truy xuất ngẫu nhiên các phần tử trong tập hợp
- Các hàm khởi tạo:

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.

5.3. Các lớp tập hợp trong Java

Lớp ArrayList: Ví dụ



```
import java.util.*;
public class ArrayListTest {
    public static void main(String[] args) {
        ArrayList<String> obj = new ArrayList<String>();
        obj.add("Ajeet");
        obj.add("Harry");
        obj.add("Steve");
        obj.add("Anuj");
        /* Displaying array list elements */
        System.out.println("Currently the array list has following elements:" + obj);
        /* Add element at the given index */
        obj.add(1, "Justin");
        /* Remove elements from array list like this */
        obj.remove("Harry");
        System.out.println("Current array list is:" + obj);
        /* Remove element from the given index */
        obj.remove(1);
        System.out.println("Current array list is:" + obj);
    }
}
```

Currently the array list has following elements:[Ajeet, Harry, Steve, Anuj]
Current array list is:[Ajeet, Justin, Steve, Anuj]
Current array list is:[Ajeet, Steve, Anuj]

5.3. Các lớp tập hợp trong Java

Lớp ArrayList: Quiz



- Cho đoạn mã sau:

```
ArrayList myArrayList = new ArrayList( );  
myArrayList.add("One");  
myArrayList.add("Two");  
myArrayList.add("Three");  
myArrayList.add("Four");
```

Đoạn lệnh nào sau đây làm thay đổi myArrayList thành: One; Two; Four

- a. myArrayList.remove (myArrayList.get(3));
- b. myArrayList.remove (myArrayList.indexOf("Three"));
- c. myArrayList.remove (Three);
- d. myArrayList.remove (myArrayList.get(2));

5.3. Các lớp tập hợp trong Java

Lớp ArrayList: Quiz



- Cho đoạn mã sau:

```
ArrayList myArrayList = new ArrayList( );  
myArrayList.add("One");  
myArrayList.add("Two");  
myArrayList.add("Three");  
myArrayList.add("Four");
```

Đoạn lệnh nào sau đây làm thay đổi myArrayList thành: One; Two; Three; Five

- a. myArrayList[3] = "Five";
- b. myArrayList[4] = "Five";
- c. myArrayList.set (myArrayList.indexOf("Four"), "Five");
- d. myArrayList.set (myArrayList.indexOf("Five"), "Four");

5.3. Các lớp tập hợp trong Java

Lớp ArrayList: Quiz



- Cho đoạn mã sau:

```
ArrayList myArrayList = new ArrayList();  
myArrayList.add(1);  
myArrayList.add(3);  
myArrayList.add(7);
```

Đoạn lệnh nào sau đây làm thay đổi myArrayList thành: 1 3 5 7

- a. myArrayList.add(5);
- b. myArrayList.add(2, 5);
- c. myArrayList.add(4, 5);
- d. myArrayList.add(3, 5);

5.3. Các lớp tập hợp trong Java

Lớp ArrayList: Case study



- Biết mỗi sinh viên gồm các thông tin *mã*, *họ tên*, *năm sinh*. Xây dựng lớp với các thao tác như sau với yêu cầu sử dụng ArrayList để lưu danh sách sinh viên:
 - Lấy thông tin toàn bộ sinh viên
 - Thêm một sinh viên mới, không được trùng mã
 - Xóa sinh viên khi biết mã; xóa các sv sinh năm nào đó,...
 - Sửa thông tin sinh viên (không sửa mã)
 - Tìm kiếm sinh viên theo mã; theo tên,...
 - Sắp xếp danh sách theo mã tăng dần,...
- .Thêm vào danh sách 5 sinh viên tùy ý và test các phương thức trên.

5.3. Các lớp tập hợp trong Java

Lớp Vector



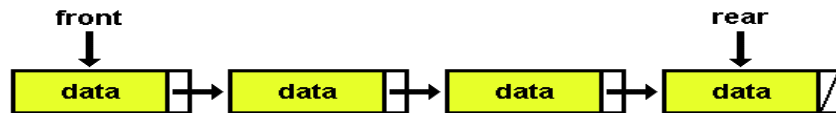
- Tương tự ArrayList
- Các phương thức của vector được đồng bộ → an toàn khi được sử dụng trong các Thread

5.3. Các lớp tập hợp trong Java

Lớp LinkedList



- Thực thi List và Queue interface
- Các phần tử được lưu trữ dạng một danh sách liên kết



- Các phương thức

Method Summary

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

5.3. Các lớp tập hợp trong Java

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 hàm:

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 according to its natural ordering (which is the 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.

5.3. Các lớp tập hợp trong Java

Lớp PriorityQueue: Ví dụ



```
public static void main(String[] args) {  
    PriorityQueue<String> pQueue = new PriorityQueue<String>();  
    pQueue.offer("Hello");  
    pQueue.offer("Bonjour");  
    pQueue.offer("Konichiowa");  
    pQueue.offer("Abc");  
    System.out.println("Content:" + pQueue);  
  
    System.out.println("Retrieve and remove head element: " + pQueue.poll());  
  
    System.out.println("Current content:" + pQueue);  
}
```

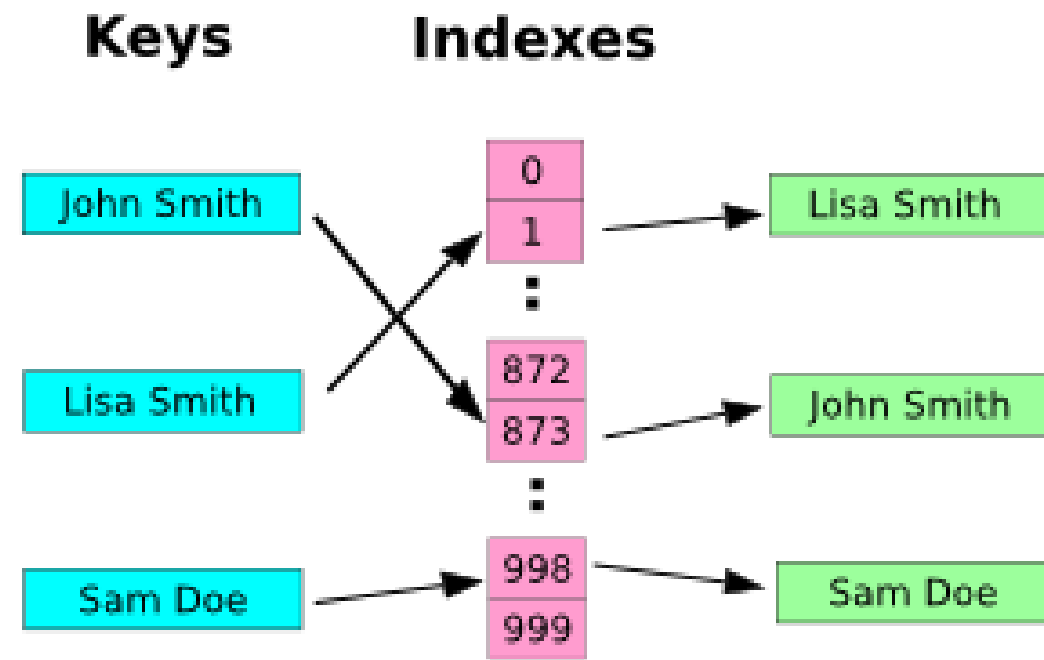
```
Content:[Abc, Bonjour, Konichiowa, Hello]  
Retrieve and remove head element: Abc  
Current content:[Bonjour, Hello, Konichiowa]
```


5.3. Các lớp tập hợp trong Java

Lớp HashSet



- Thực thi Set interface
- HashSet lưu trữ các phần tử bằng cách sử dụng một cơ chế gọi là băm (hashing)
- HashSet không cho phép lưu trùng lặp



5.3. Các lớp tập hợp trong Java

Lớp HashSet: Ví dụ



```
import java.util.*;
public class HashSetTest {
    public static void main(String[] args) {
        Set<String> words = new HashSet<String>();
        words.add("Bats");
        words.add("Ants");
        words.add("Crabs");
        words.add("Ants");
        System.out.println(words);
    }
}
```

5.3. Các lớp tập hợp trong Java

Lớp TreeSet



- Thực thi SortedSet interface
- Lưu giữ liệu theo cấu trúc “cây”
- Chậm hơn HashSet nhưng có khả năng tự sắp xếp: các phần tử được lưu trữ theo thứ tự tăng dần
- Có thể quy định thứ tự bằng:
 - Comparable (theo thứ tự tự nhiên: số, chuỗi)
 - Comparator

5.3. Các lớp tập hợp trong Java

Lớp TreeSet: Ví dụ 1



- String có thứ tự tự nhiên:

```
Set<String> words = new TreeSet<String>();  
words.add("Bats");  
words.add("Ants");  
words.add("Crabs");  
System.out.println(words);
```

What's the output?
[Ants, Bats, Crabs]

5.3. Các lớp tập hợp trong Java

Lớp TreeSet: Ví dụ 2



- Quy định thứ tự bằng Comparable:

```
public class Student implements Comparable<Student> {  
    private String name;  
    private int gpa;  
    //...  
    @Override  
    public int compareTo(Student o) {  
        return this.name.compareToIgnoreCase(o.name);  
    }  
}
```

Student
- name
- gpa

```
Set<Student> studentSet = new TreeSet<Student>();  
studentSet.add(new Student("Fred", 3));  
studentSet.add(new Student("Sam", 4));  
studentSet.add(new Student("Steve", 3));  
studentSet.add(new Student("Laura", 2));  
for (Student st: studentSet) {  
    System.out.println(st);  
}
```

What's the output?

Fred-3
Laura-2
Sam-4
Steve-3

5.3. Các lớp tập hợp trong Java

Lớp TreeSet: Ví dụ 3



- Quy định thứ tự bằng Comparator (truyền vào constructor của TreeSet)

```
public class StudentNameComparator implements Comparator<Student> {  
    @Override  
    public int compare(Student o1, Student o2) {  
        return o1.getName().compareToIgnoreCase(o2.getName());  
    }  
}
```

Student
- name
- gpa

```
Set<Student> studentSet = new TreeSet<Student>(new StudentNameComparator());  
studentSet.add(new Student("Fred", 3));  
studentSet.add(new Student("Sam", 4));  
studentSet.add(new Student("Steve", 3));  
studentSet.add(new Student("Laura", 2));  
for (Student st: studentSet) {  
    System.out.println(st);  
}
```

What's the output?

Fred-3

Laura-2

Sam-4

Steve-3

5.3. Các lớp tập hợp trong Java

Lớp HashMap



- Thực thi Map interface
- Hoạt động dựa trên nguyên lý của việc băm dữ liệu (hashing)
- Lưu trữ dữ liệu theo từng cặp: khóa-giá trị (key-value)
- Các khóa trong Map phải duy nhất
- Có thể có một khóa null và nhiều giá trị null
- Lưu trữ không theo trật tự thêm vào

5.3. Các lớp tập hợp trong Java

Lớp HashMap: Ví dụ



- Tạo và xuất HashMap:

```
HashMap<Integer, String> hmap = new HashMap<Integer, String>();  
hmap.put(24, "Hà Nội");  
hmap.put(236, "Đà Nẵng");  
hmap.put(258, "Khánh Hòa");  
hmap.put(28, "Thành Phố Hồ Chí Minh");  
System.out.println(hmap);
```

{258=Khánh Hòa, 24=Hà Nội, 236=Đà Nẵng, 28=Thành Phố Hồ Chí Minh}

- Lấy phần tử có khóa tùy ý:

```
String var = hmap.get(28);  
System.out.println("Value at index 28 is: " + var);
```

Value at index 2 is: Thành Phố Hồ Chí Minh.

- Lấy ra tập key rồi duyệt dựa trên nó:

```
for (int key : hmap.keySet()) {  
    System.out.println("Key = " + key + " & Value = " + hmap.get(key));  
}
```

Key = 258 & Value = Khánh Hòa
Key = 24 & Value = Hà Nội
Key = 236 & Value = Đà Nẵng
Key = 28 & Value = Thành Phố Hồ Chí Minh

5.3. Các lớp tập hợp trong Java

Lớp TreeMap



- Thực thi SortedMap interface
- Lưu trữ dữ liệu dưới dạng cặp key-value, theo cấu trúc cây
- Các phần tử sắp xếp dựa trên giá trị của khóa: các đối tượng khoá đưa vào trong TreeMap phải implements interface Comparable HOẶC lớp cài đặt TreeMap phải nhận một Comparator trên đối tượng khoá
- Không cho phép key là null nhưng có thể có nhiều giá trị null

5.3. Các lớp tập hợp trong Java

Lớp TreeMap: Ví dụ



- Tạo và xuất TreeMap:

```
TreeMap<Integer, String> tmap = new TreeMap<Integer, String>();  
tmap.put(24, "Hà Nội");  
tmap.put(236, "Đà Nẵng");  
tmap.put(258, "Khánh Hòa");  
tmap.put(28, "Thành Phố Hồ Chí Minh");  
System.out.println(tmap);
```

{24=Hà Nội, 28=Thành Phố Hồ Chí Minh, 236=Đà Nẵng, 258=Khánh Hòa}

- Lấy phần tử có khóa tùy ý:

```
String var = tmap.get(28);  
System.out.println("Value at index 28 is: " + var);
```

Value at index 2 is: Thành Phố Hồ Chí Minh.

- Lấy ra tập key rồi duyệt dựa trên nó:

```
for (int key : tmap.keySet()) {  
    System.out.println("Key = " + key + " & Value = " + tmap.get(key));  
}
```

Key = 24 & Value = Hà Nội
Key = 28 & Value = Thành Phố Hồ Chí Minh
Key = 236 & Value = Đà Nẵng
Key = 258 & Value = Khánh Hòa

5.3. Các lớp tập hợp trong Java

Lớp Arrays



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

```
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()
```

5.3. Các lớp tập hợp trong Java

Lớp Arrays: Ví dụ



```
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
```

5.3. Các lớp tập hợp trong Java

Các lớp bao (wrapper classes)



- Collection chỉ làm việc trên các Object. Những kiểu dữ liệu cơ bản như: byte, short, int, long, double, float, char, boolean không thể đưa được trực tiếp vào Collection mà phải thông qua các lớp bao tương ứng: Byte, Short, Integer, Long, Double, Float, Char, Boolean
- Ví dụ:

```
Integer intObject = new Integer(9);  
int value = intObject.intValue();
```



Review questions 2

- Trong ArrayList, làm thế nào để thêm, xóa, sửa, tìm kiếm, lấy phần tử bất kỳ.
- So sánh HashSet, TreeSet với các đặc điểm: ordering, performance, null value
- ArrayList, Vector, LinkedList có đặc điểm chung gì.
- TreeSet, TreeMap có đặc điểm chung gì.
- PriorityQueue và ArrayList giống nhau và khác nhau ở điểm nào.

