



Generics and Collections

Nguyễn Lê Hoàng Dũng
Hồ Tuấn Thanh

1. Generics

Why use Generics? (Example1)

```
18 public static void main(String[] args) {  
19     ArrayList list = new ArrayList();  
20     //one do  
21     list.add(new Integer(5));  
22     list.add(new Integer(10));  
23     //another do  
24     list.add(new String("3"));  
25     //get an item  
26     Integer first = (Integer)list.get(0); //true  
27     Integer third = (Integer)list.get(2); //false  
28 }  
29
```

```
Exception in thread "main" java.lang.ClassCastException: java.lang.String cannot be cast to java.lang.Integer  
    at javaapplication1.JavaApplication1.main(JavaApplication1.java:28)
```

Java Result: 1

BUILD SUCCESSFUL (total time: 0 seconds)

Why use Generics?

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    ArrayList myList = new ArrayList();  
    myList.add(1);  
    myList.add("hai");  
    myList.add("ba");  
    Integer _1Int = (Integer) myList.get(0); //true  
    Integer _2Int = (Integer) myList.get(1); //false  
  
    Integer _3Int;  
    Object obj = myList.get(2);  
    if(obj instanceof Integer)  
    {  
        _3Int = (Integer) obj; //true  
    }  
}
```

Why use Generics?

```
public void nonGenericCode()
```

```
{  
    ArrayList list = new ArrayList();  
    list.add(new Integer(10));
```

```
    Integer i = (Integer) list.get(0);  
}
```

required explicit cast

```
public void genericCode()
```

```
{  
    ArrayList<Integer> list = new ArrayList<Integer>();  
    list.add(new Integer(10));
```

```
    Integer i = list.get(0);  
}
```

Type of element

No required explicit cast

Why use Generics?

Phương pháp chỉ ra **kiểu** đối tượng mà một Lớp có thể chứa

- Phát hiện sớm các kiểu dữ liệu không phù hợp tại thời điểm biên dịch chương trình.
- Hạn chế việc ép kiểu các đối tượng.
- Cài đặt các thuật toán kết hợp kiểu generic và Collection

Generic Class

- Lớp Generic là một cơ chế để chỉ rõ mối quan hệ giữa Lớp và kiểu dữ liệu liên quan đến nó (type parameter).

- Lớp có thể có nhiều tham số. Ví dụ:

```
public class CBox<T1, T2, ..., Tn> {...}
```

- Quy ước về tên tham số kiểu (Type Parameter Naming Conventions)

- E – Element

- K – Key

- N – Number

- T – Type

- V – Value

- *The diamond operator <>*

Generic Class (Example2)

```
public class NumberList<T>
{
    private T obj;

    public void add(T value)
    {
        this.obj = value;
    }

    public T get()
    {
        return obj;
    }
}
```

```
public void testNumberList_String()
{
    NumberList<String> list = new NumberList<String>();
    list.add("Hello");

    System.out.println(list.get());
}

public void testNumberList_Integer()
{
    NumberList<Integer> list = new NumberList<Integer>();
    list.add(new Integer(10));
    // list.add("10"); //Error

    System.out.println(list.get().intValue());
}
```


Generic Methods

- Các “tham số kiểu” được khai báo, sử dụng trong phạm vi của phương thức.
- **Tham số kiểu** phải được chỉ rõ **trước** kiểu dữ liệu trả về của phương thức và **đặt trong cặp dấu <>**
- Có thể dùng tham số kiểu cho:
 - Dữ liệu trả về
 - Các tham số của phương thức
 - Biến cục bộ

Generic Methods (Example3)

```
public class GenericMethods
{
    public static <E> void printArray(E[] inputArray)
    {
        for (E element : inputArray)
        {
            System.out.printf("%s ", element);
        }
    }
}
```

```
public static void main(String args[])
{
    Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
    Character[] characterArray = { 'H', 'E', 'L', 'L', 'O' };

    System.out.println("\nArray doubleArray contains:");
    printArray(doubleArray);

    System.out.println("");

    System.out.println("\nArray characterArray contains:");
    printArray(characterArray);
}
```

Output

```
Array doubleArray contains:
1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array characterArray contains:
H E L L O
```

Bounded type parameters

- Giới hạn tham số kiểu:
 - Lớp chỉ định
 - Các lớp con của lớp chỉ định

cuu duong than cong . com

cuu duong than cong . com

Bounded type parameters

```
public class Box<T> {  
    private T t;  
    public void set(T t) {  
        this.t = t;  
    }  
    public T get() {  
        return t;  
    }  
    public <U extends Number> void inspect(U u){  
        System.out.println("T: " + t.getClass().getName());  
        System.out.println("U: " + u.getClass().getName());  
    }  
    public static void main(String[] args) {  
        Box<Integer> integerBox = new Box<Integer>();  
        integerBox.set(new Integer(10));  
        integerBox.inspect("some text"); // error: this is still String!  
    }  
}
```

Multiple Bounds

- Có thể giới hạn với nhiều tham số kiểu
- Sử dụng “&”
- Nếu kiểu giới hạn thuộc kiểu class thì phải đặt trước
- Ví dụ:

```
Class A { /* ... */ }
```

```
interface B { /* ... */ }
```

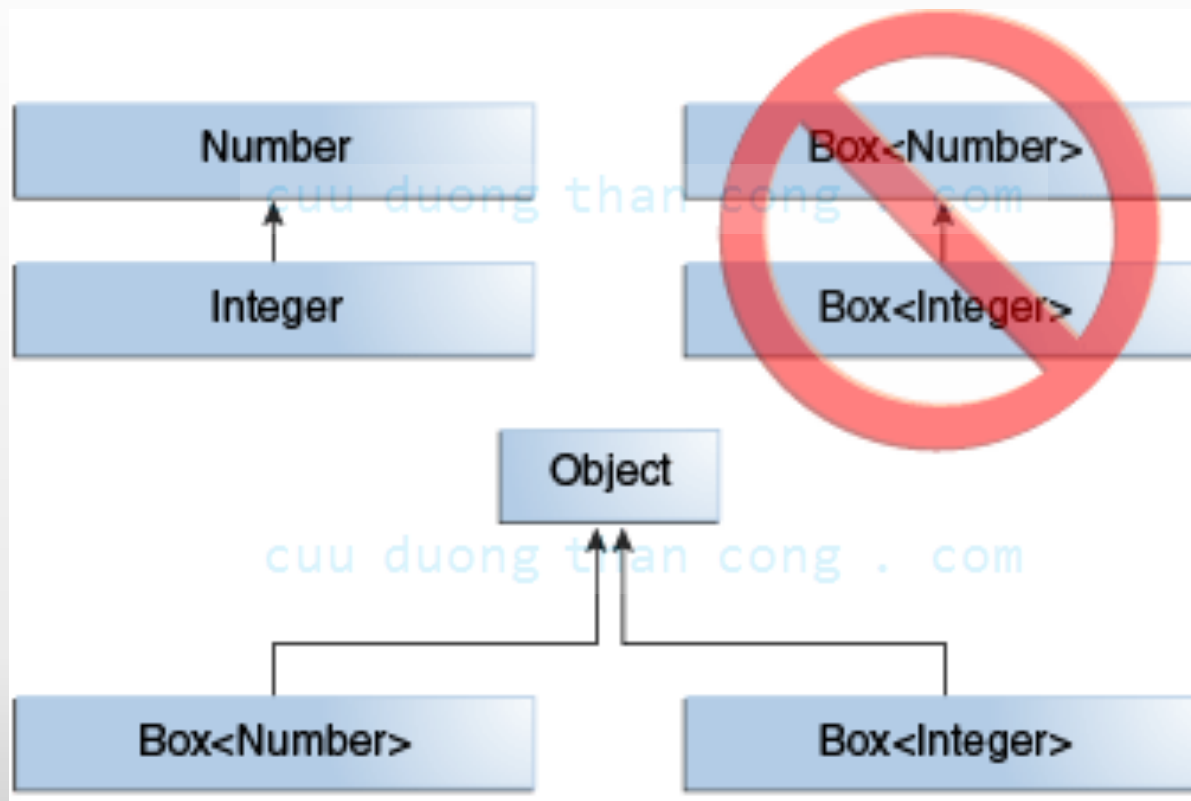
```
interface C { /* ... */ }
```

```
class D <T extends A & B & C> { /* ... */ }
```

Generics, Inheritance, and Subtypes

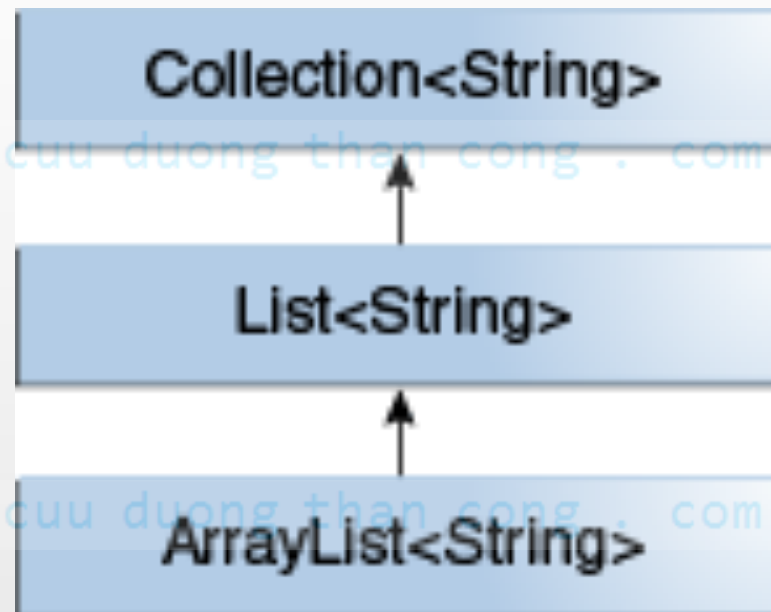
```
public void boxTest(Box<Number> n) { /* ... */ }
```

=> Box<Integer> or Box<Double>?



Generics, Inheritance, and Subtypes

`ArrayList<E>` implements `List<E>`, and `List<E>` extends `Collection<E>`



Wildcards

- “?” : đại diện cho một kiểu chưa xác định.
- “? extends Type”
 - Đại diện cho một kiểu là lớp con của lớp được chỉ ra hoặc chính nó. Ví dụ: `List <? extends Number>`
- “? super Type”
 - Đại diện cho một kiểu là lớp cha của lớp được chỉ ra hoặc chính nó. Ví dụ: `List <? super Number>`

Wildcards- Ví dụ

“?”

```
List<?> list = null;  
  
list = new ArrayList<Date>();  
list = new ArrayList<String>();
```

“? extends type”

```
List<? extends Date> dateList = null;  
  
dateList = new ArrayList<Date>();  
dateList = new ArrayList<MyDate>(); // class MyDate extends Date
```

“? super type”

```
List<? super MyDate> dateList1 = null;  
  
dateList1 = new ArrayList<Date>();  
dateList1 = new ArrayList<MyDate>(); // class MyDate extends Date
```

Wildcards and Subtyping

- Ví dụ:

```
class A { /* ... */ }
```

```
class B extends A { /* ... */ }
```

```
B b = new B();
```

```
A a = b; //OK
```

```
List<B> lb = new ArrayList<>();
```

```
List<A> la = lb; // compile-time error
```

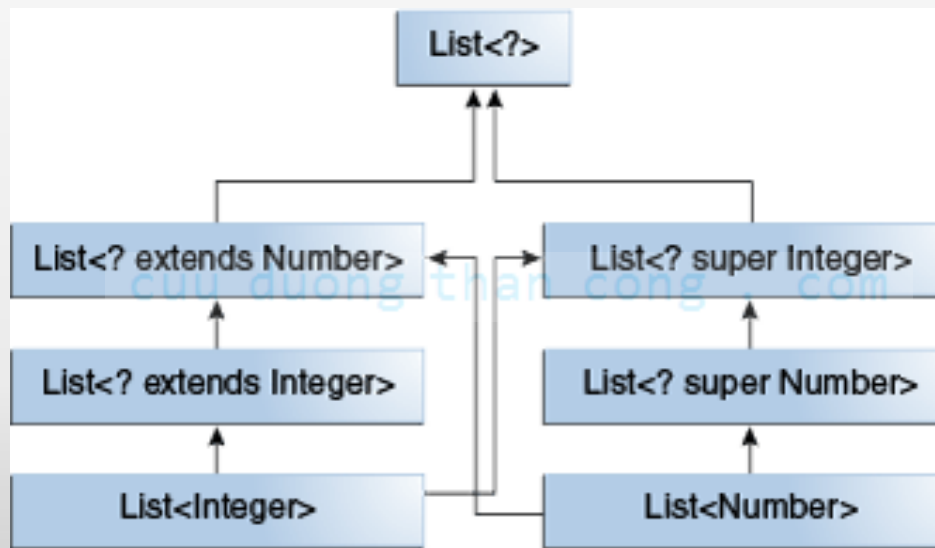
Wildcards and Subtyping

- Ví dụ:

```
List<? extends Integer> intList = new ArrayList<>();
```

```
List<? extends Number> numList = intList;
```

```
//OK. List<? extends Integer> lớp con List<? extends Number>
```



2. Collections

Mảng – Ví dụ

```
public class Car {  
  
}
```

```
Car[] cars1; null
```

```
Car[] cars2 = new Car[5];
```

null	null	null	null	null
------	------	------	------	------

```
for(int i=0; i<5;i++){  
    cars2[i]=new Car();  
    // Xử lý khác  
}
```

Mảng – Nhận xét

- Phải cho biết trước số lượng phần tử trong mảng
- Không thể thay đổi kích thước về sau (mở rộng)
 - Khai báo mảng mới cars3
 - Copy dữ liệu qua mảng mới cars3
 - Cấp vùng nhớ mới cho cars2
 - Copy dữ liệu từ cars3 qua cars2
- Thêm phần tử x vào vị trí k
- Xóa phần tử x

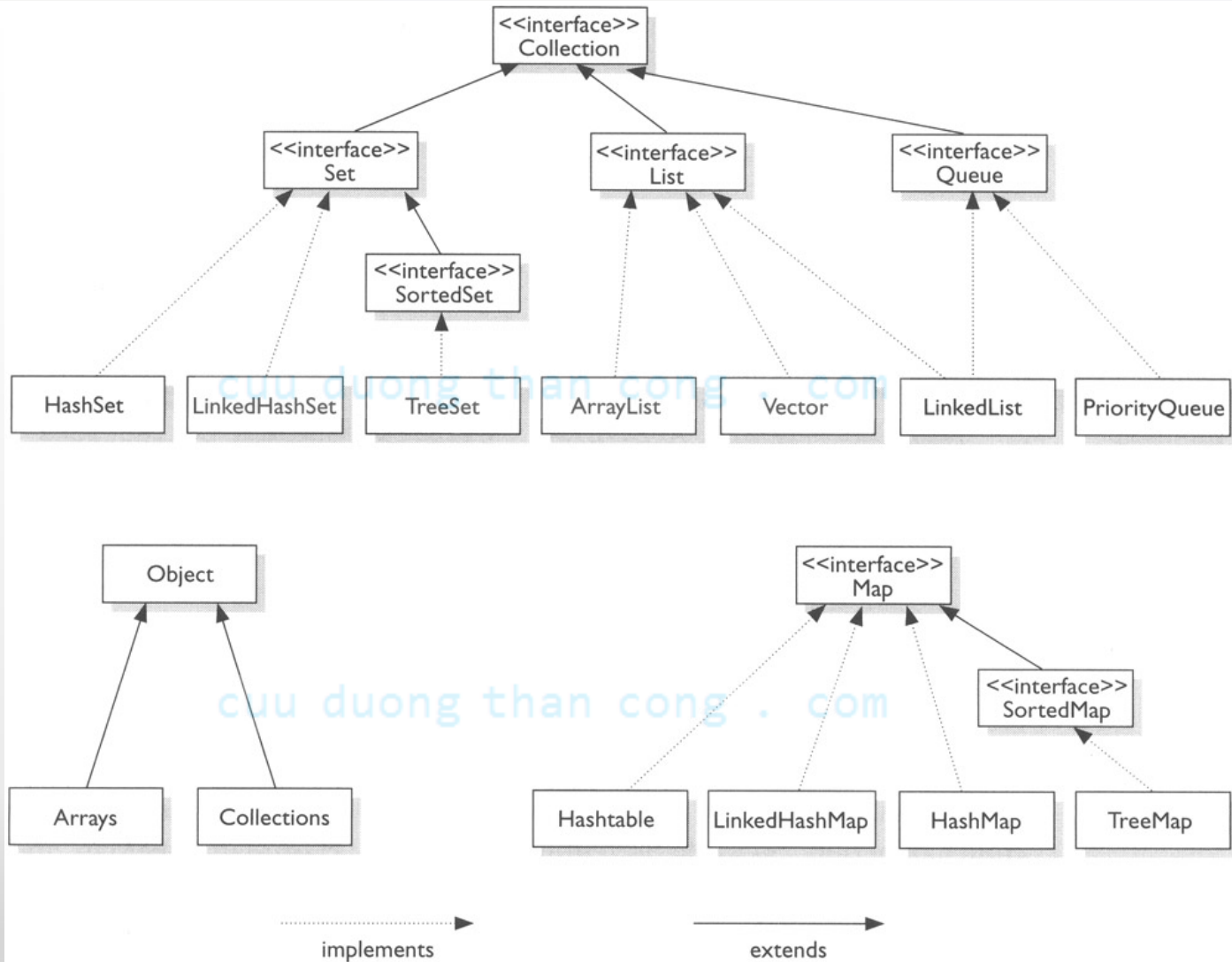
Collections

- Collection là đối tượng có khả năng chứa các đối tượng khác.
- Các thao tác thông thường trên collection
 - Khởi tạo collection
 - Thêm/Xoá đối tượng vào/khỏi collection
 - Kiểm tra một đối tượng có ở trong collection không
 - Lấy một đối tượng từ collection
 - Duyệt các đối tượng trong collection
 - Xoá toàn bộ collection

Collections Framework

- Một số lợi ích của Collections Framework:
 - Giảm thời gian lập trình
 - Tăng cường hiệu năng chương trình
 - Dễ mở rộng các collection mới
 - Khuyến khích việc sử dụng lại mã chương trình

Collections Framework



Collections Framework

- Collections Framework bao gồm:
 - Interfaces: Là các giao tiếp 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.
 - Algorithms: Là các phương thức tính để xử lý trên collection, ví dụ: sắp xếp danh sách, tìm phần tử lớn nhất...

Interface Collection

- Lớp cơ sở
- Không có lớp con được cài đặt sẵn, chỉ có lớp con thông qua Set, List, Queue
- Cung cấp các thao tác chính trên collection:
 - boolean add(Object element);
 - boolean remove(Object element);
 - boolean contains(Object element);
 - int size();
 - boolean isEmpty();

Interface List

- List kế thừa từ Collection
- Cung cấp thêm các phương thức để xử lý collection kiểu danh sách (chứa các phần tử được xếp theo chỉ số).
 - Phần tử có thể trùng
- Một số phương thức của List
 - Object get(int index);
 - Object set(int index, Object o);
 - void add(int index, Object o);
 - Object remove(int index);
 - int indexOf(Object o); - int lastIndexOf(Object o);

Interface Set

- Set kế thừa từ Collection
- Hỗ trợ các thao tác xử lý trên collection kiểu tập hợp (toán học)
 - Phần tử **KHÔNG THỂ** trùng
- Một số method riêng:
 - `set1.containsAll(set2)` // set2 is a **subset** of set1
 - `set1.addAll(set2)`; // phép **hội**
 - `set1.retainAll(set2)`; // phép **giao**
 - `set1.removeAll(set2)`; // phép **trừ**

Interface SortedSet

- SortedSet kế thừa từ Set
- Nó hỗ trợ thao tác trên tập hợp các phần tử có thể so sánh được.
- Các đối tượng đưa vào trong một SortedSet phải cài đặt giao tiếp Comparable hoặc phải truyền vào lớp cài đặt SortedSet một Comparator
- Một số phương thức của SortedSet:
 - Object first();
 - Object last();
 - SortedSet headSet(Object end); // <= end
 - SortedSet tailSet(Object start); // >=start
 - SortedSet subSet(Object start, Object end);

Interface Queue

- Kế thừa interface Collection
- Hoạt động theo cơ chế FIFO
- Một số method riêng:
 - `boolean offer(Object obj);` // Thêm vào queue
 - `Object poll();` // Xóa khỏi queue, trả về phần tử đầu
 - `Object peek();` // Lấy phần tử đầu, ko xóa

60	50	40	30	20	10
----	----	----	----	----	----

Interface Map

- Ko kế thừa interface Collection
- Quản lý các phần tử theo cơ chế key-value
 - Các Key không trùng nhau

cuu duong than cong . com

cuu duong than cong . com

Interface Map

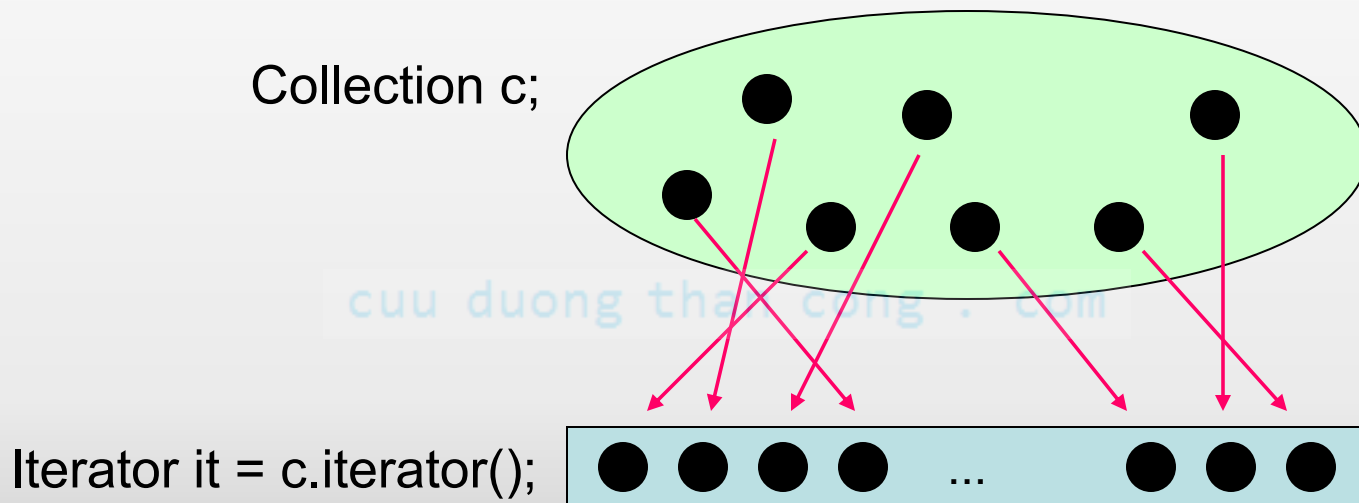
- Một số method thông dụng:
 - Object put(Object key, Object value);
 - Object get(Object key);
 - Object remove(Object key);
 - boolean containsKey(Object key);
 - boolean containsValue(Object value);
 - Set keySet(); // Trả về các key
 - Collection values(); // Trả về các value
 - Set entrySet(); // Trả về các cặp key-value

Interface SortedMap

- Kế thừa interface Map
- Giống như SortedSet, các đối tượng khoá đưa vào trong SortedMap phải cài đặt giao tiếp Comparable hoặc lớp cài đặt SortedMap phải nhận một Comparator trên đối tượng khoá.
- Một số method riêng:
 - Object firstKey();
 - Object lastKey();
 - SortedMap headMap(Object end);
 - SortedMap tailMap(Object start);
 - SortedMap subMap(Object start, Object end);

Iterator Interface

- Các phần tử trong collection có thể được duyệt thông qua **Iterator (bộ duyệt)**.
- Các lớp cài đặt Collection cung cấp phương thức trả về iterator trên các phần tử của chúng.

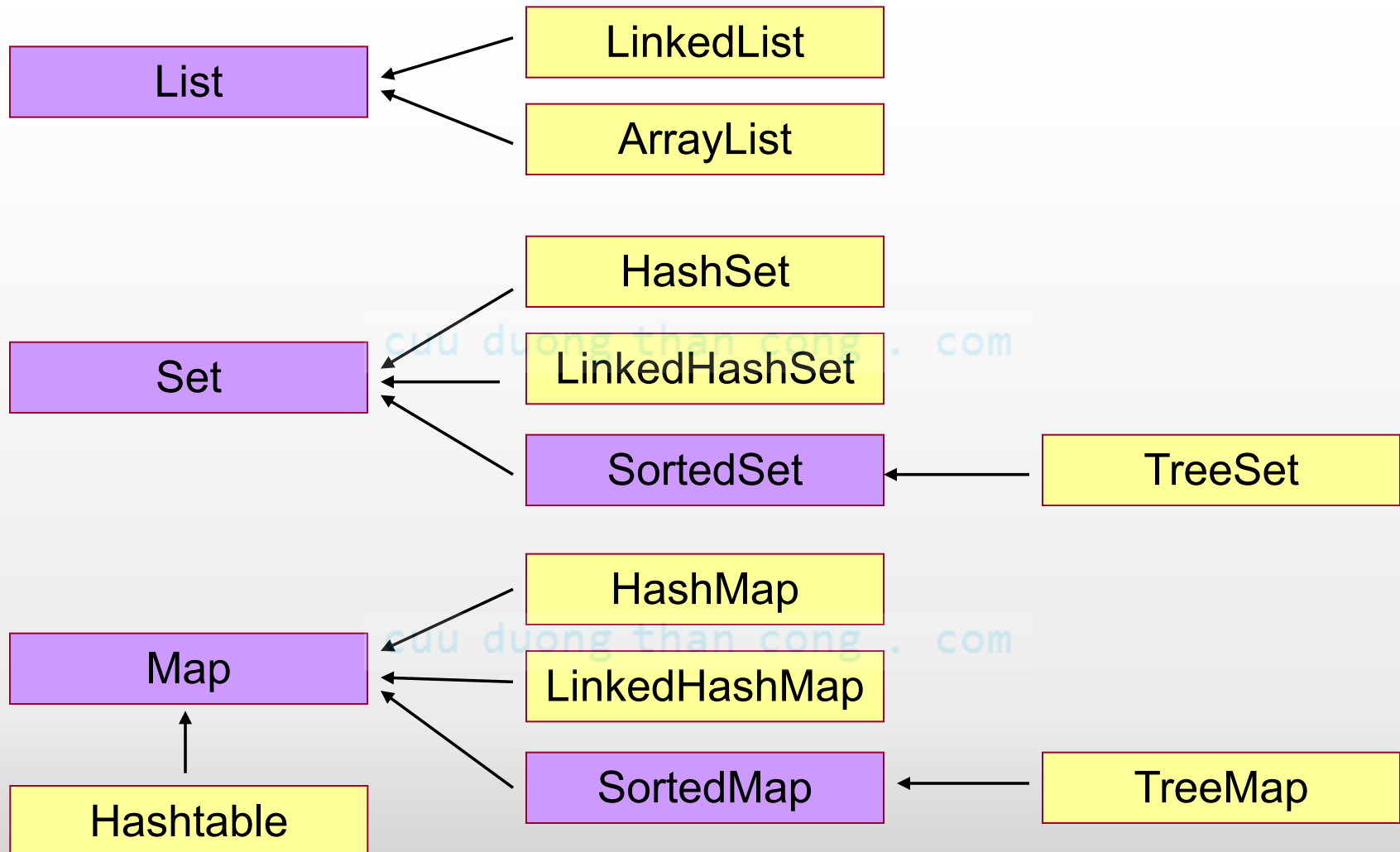


Duyệt Collection

- Iterator cho phép duyệt tuần tự một collection.
- Các phương thức của Iterator:
 - boolean hasNext();
 - Object next();
 - void remove();
- Ví dụ:

```
Iterator it = c.iterator();
while ( it.hasNext() ) {
    Point p = (Point) it.next();
    System.out.println( p.toString() );
}
```

Implementations



Ví dụ 5: ArrayList

```
// create an array list
List al = new ArrayList();
System.out.println("Initial size of al: " + al.size());
// add elements to the array list
al.add("C");
al.add("A");
al.add("E");
al.add("B");
al.add("D");
al.add("F");
al.add(1, "A2");
System.out.println("Size of al after additions: " + al.size());

// display the array list
System.out.println("Contents of al: " + al);
// Remove elements from the array list
al.remove("F");
al.remove(2);
System.out.println("Size of al after deletions: " + al.size());
System.out.println("Contents of al: " + al);
```

Ví dụ 6: LinkedList

```
// remove elements from the linked list
ll.remove("F");
ll.remove(2);
System.out.println("Contents of ll after deletion: "
    + ll);

// remove first and last elements
ll.removeFirst();
ll.removeLast();
System.out.println("ll after deleting first and last: "
    + ll);

// get and set a value
Object val = ll.get(2);
ll.set(2, (String) val + " Changed");
System.out.println("ll after change: " + ll);
```

Ví dụ 7: Queue

```
Queue queue = new LinkedList();

for (int i = 0; i < 100; i+=10)
    queue.offer(i);
System.out.println("Queue: " + queue);
while (!queue.isEmpty()) {
    System.out.println(queue.poll());
}
```

cuu duong than cong . com

Ví dụ 8: Vector

Khởi tạo:

Vector();

Vector(int size);

Vector(int size, int incr);

```
// initial size is 3, increment is 2
Vector v = new Vector(3, 2);
System.out.println("Initial size: " + v.size());
System.out.println("Initial capacity: " +
v.capacity());
v.addElement(new Integer(1));
v.addElement(new Integer(2));
v.addElement(new Integer(3));
v.addElement(new Integer(4));
System.out.println("Capacity after four additions: "
+ v.capacity());
System.out.println("Vector: "+ v);
```

Ví dụ 9: TreeSet

```
// This program sorts a set of names
```

```
import java.util.*;
```

```
public class TreeSetTest1
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        SortedSet names = new TreeSet();
```

```
        names.add(new String("Minh Tuan"));
```

```
        names.add(new String("Hai Nam"));
```

```
        names.add(new String("Anh Ngoc"));
```

```
        names.add(new String("Trung Kien"));
```

```
        names.add(new String("Quynh Chi"));
```

```
        names.add(new String("Thu Hang"));
```

```
        System.out.println(names);
```

```
    }
```

```
}
```

Ví dụ 10: Student Set

```
class Student implements Comparable
{
    private String code;
    private double score;

    public Student(String code, double score)
    {
        this.code = code;
        this.score = score;
    }

    public double getScore()
    {
        return score;
    }

    public String toString()
    {
        return "(" + code + "," + score + ")";
    }
}
```

Ví dụ 10: Student Set

```
public boolean equals(Object other)
{
    Student otherStu = (Student) other;
    return code.equals(otherStu.code);
}
```

cuu duong than cong . com

```
public int compareTo(Object other)
{
    Student otherStu = (Student) other;
    return code.compareTo(otherStu.code);
}
}
```

cuu duong than cong . com

Ví dụ 10: Student Set

// This programs sorts a set of students by name and then by score

```
import java.util.*;
```

```
public class TreeSetTest2
```

```
{
```

```
    public static void main(String[] args)
```

```
    {
```

```
        SortedSet stu = new TreeSet();
```

```
        stu.add(new Student("A05726", 8.5));
```

```
        stu.add(new Student("A06338", 7.0));
```

```
        stu.add(new Student("A05379", 7.5));
```

```
        stu.add(new Student("A06178", 9.5));
```

```
        System.out.println(stu);
```

```
    }
```

Ví dụ 11: HashMap

```
// This program stores a phone directory by hashing
import java.util.*;

public class MyMapTest
{
    public static void main(String[] args)
    {
        Map phoneDir = new HashMap();
        phoneDir.put("5581814", new String("Dept. Informatics"));
        phoneDir.put("8584490", new String("Defense Staff"));
        phoneDir.put("8587346", new String("Administrative Staff"));
        phoneDir.put("7290028", new String("Student Club"));

        // print all entries
        System.out.println(phoneDir);

        // remove an entry
        phoneDir.remove("8584490");
    }
}
```

Ví dụ 11: HashMap (cont)

// replace an entry

```
phoneDir.put("7290028", new String("International Relation"));
```

// look up a value

```
System.out.println(phoneDir.get("5581814"));
```

// iterate through all entries

```
Set entries = phoneDir.entrySet();
```

```
Iterator iter = entries.iterator();
```

```
while (iter.hasNext())
```

```
{
```

```
    Map.Entry entry = (Map.Entry) iter.next();
```

```
    String key = (String) entry.getKey();
```

```
    String value = (String) entry.getValue();
```

```
    System.out.println("key=" + key + ", value=" + value);
```

```
}
```

```
}
```

```
}
```

Tài liệu tham khảo

- Slide “Java Collections” – Professor Evan Korth
- <https://docs.oracle.com/javase/tutorial/java/generics/>
- http://www.tutorialspoint.com/java/java_collections.htm
- http://www.tutorialspoint.com/java/java_generics.htm

cuu duong than cong . com

Bài tập

Xem trong tập tin “Bai tap.pdf”

cuu duong than cong . com

cuu duong than cong . com