



# **LẬP TRÌNH JAVA 2**

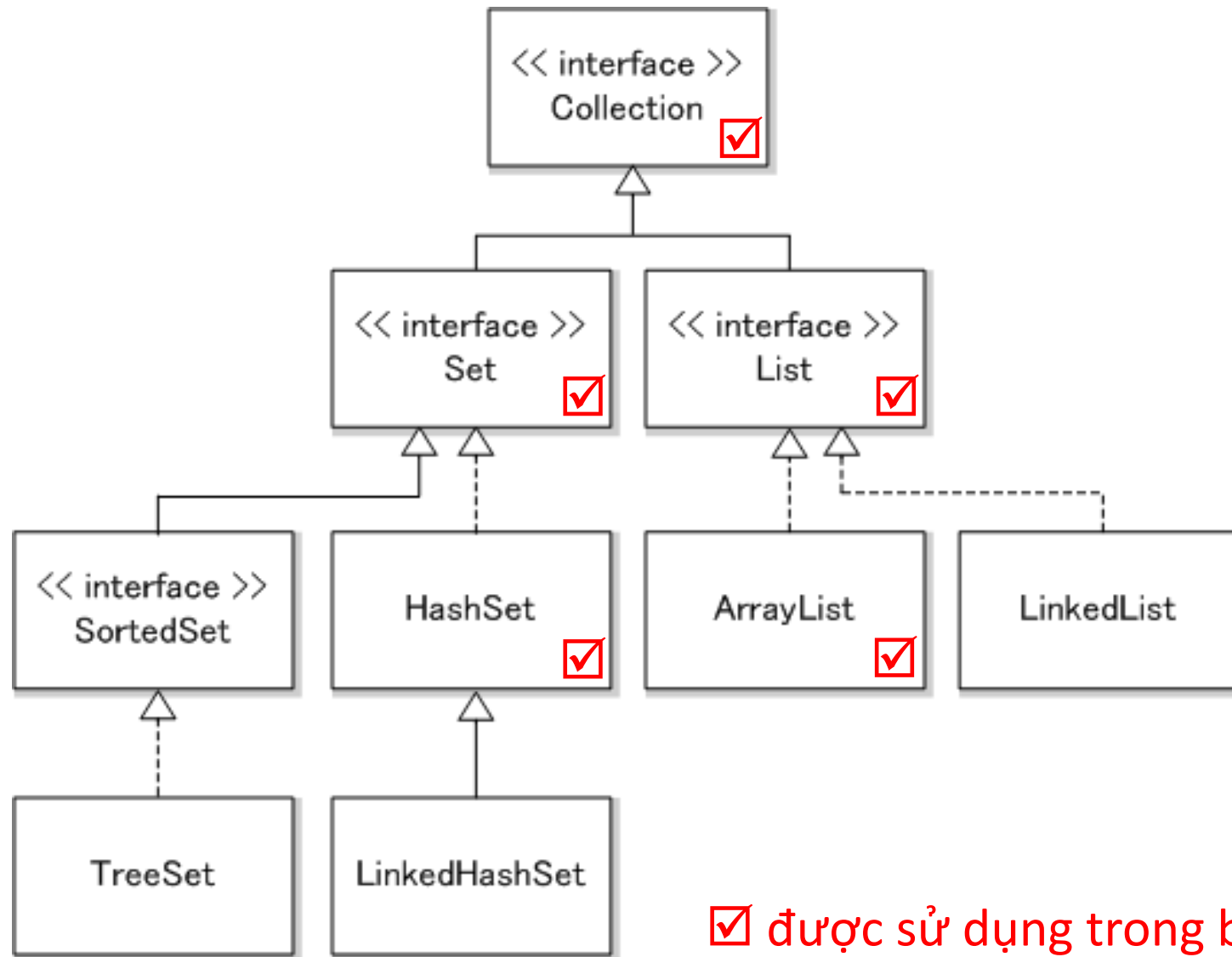
## **BÀI 3: COLLECTION & MAP**

### **PHẦN 1**

- ◎ Sử dụng Collection
  - ◎ Giải thích phân cấp thừa kế
  - ◎ Sử dụng List & ArrayList
  - ◎ Sử dụng Set & HashSet
  - ◎ Sử dụng lớp tiện ích Collections
- ◎ Sử dụng Map
  - ◎ Giải thích phân cấp thừa kế
  - ◎ Sử dụng Map & HashMap
  - ◎ Sử dụng Properties



- ❑ Collection là cấu trúc dữ liệu được sử dụng để nắm giữ nhiều phần tử.
  - ❖ Có thể thêm, xóa, cập nhật các phần tử.
  - ❖ Hợp, giao, trừ... các tập hợp
- ❑ Collection được chia làm 2 loại là List và Set
  - ❖ List là collection mà mỗi phần tử được phép xuất hiện nhiều lần và truy xuất bằng chỉ số
  - ❖ Set là collection mà mỗi phần tử chỉ được phép xuất hiện 1 lần và không được phép truy xuất theo chỉ số



✓ được sử dụng trong bài học

```
List<Integer> list = new ArrayList<Integer>();  
list.add(1);  
list.add(2);  
list.add(2);  
System.out.print(list.toString());
```

[1, 2, 2]

```
Set<Integer> set = new HashSet<Integer>();  
set.add(100);  
set.add(200);  
set.add(200);  
System.out.print(set.toString());
```

[100, 200]



# DEMO



Hiện thực hóa slide trước

Phương thức	Mô tả
<code>boolean add(Object)</code>	Thêm vào
<code>addAll(Collection)</code>	Hợp 2 tập hợp
<code>boolean remove(Object)</code>	Xóa phần tử chỉ định
<code>removeAll(Collection)</code>	Hiệu 2 tập hợp
<code>retainAll(Collection)</code>	Giao 2 tập hợp
<code>boolean contains(Object)</code>	Kiểm tra sự tồn tại một phần tử
<code>boolean containsAll(Collection)</code>	Kiểm tra sự tồn tại một tập con
<code>int size()</code>	Số phần tử
<code>boolean isEmpty()</code>	Kiểm tra rỗng hay không
<code>void clear()</code>	Xóa sạch
<code>toArray(T[])</code>	Chuyển đổi sang mảng

```
List<Integer> list = new ArrayList<Integer>();  
list.add(1);  
list.add(2);  
list.add(2);
```

```
Set<Integer> set = new HashSet<Integer>();  
set.add(100);  
set.add(200);  
set.add(200);
```

list.addAll(set)

?

```
set.addAll(list);  
System.out.print(set.toString());
```

[1,2,100,200]



- ❑ Bên cạnh các phương thức thao tác tập hợp, List được bổ sung các phương thức làm việc với chỉ số (**index**)

Phương thức	Mô tả
Object get(int index)	Truy xuất phần tử tại index
Object set(int index, Object elem)	Thay thế phần tử tại index
void add(int index, Object elem)	Chèn phần tử tại index
Object remove(int index)	Xóa phần tử tại index
int indexOf(Object elem)	Tìm vị trí phần tử từ đầu
int lastIndexOf(Object elem)	Tìm vị trí phần tử từ cuối

```
List<String> names = new ArrayList<>();  
  
names.add("Tuấn");  
names.add("Hạnh");  
names.add("Phương");  
names.add("Hằng");  
names.set(1, "Khanh");  
names.remove("Phương");  
  
System.out.println(names.toString());
```

[Tuấn, Khanh, Hằng]

```
List<String> names = new ArrayList<>();
```

```
for(int i=0;i<names.size();i++){  
    String name = names.get(i);  
    System.out.println(" >> Name: " + name);  
}
```

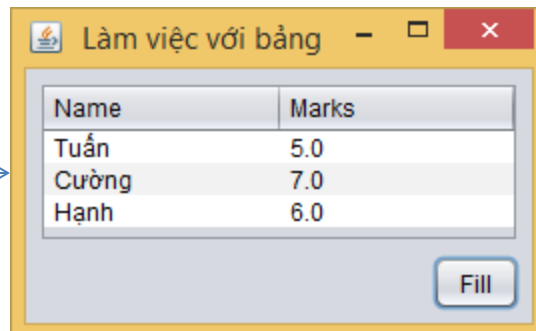
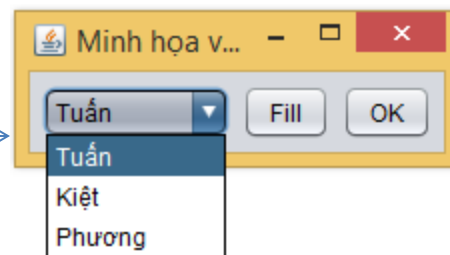
```
for(String name : names){  
    System.out.println(" >> Name: " + name);  
}
```

```
Iterator<String> iterator = names.iterator();  
while(iterator.hasNext()){  
    String name = iterator.next();  
    System.out.println(" >> Name: " + name);  
}
```

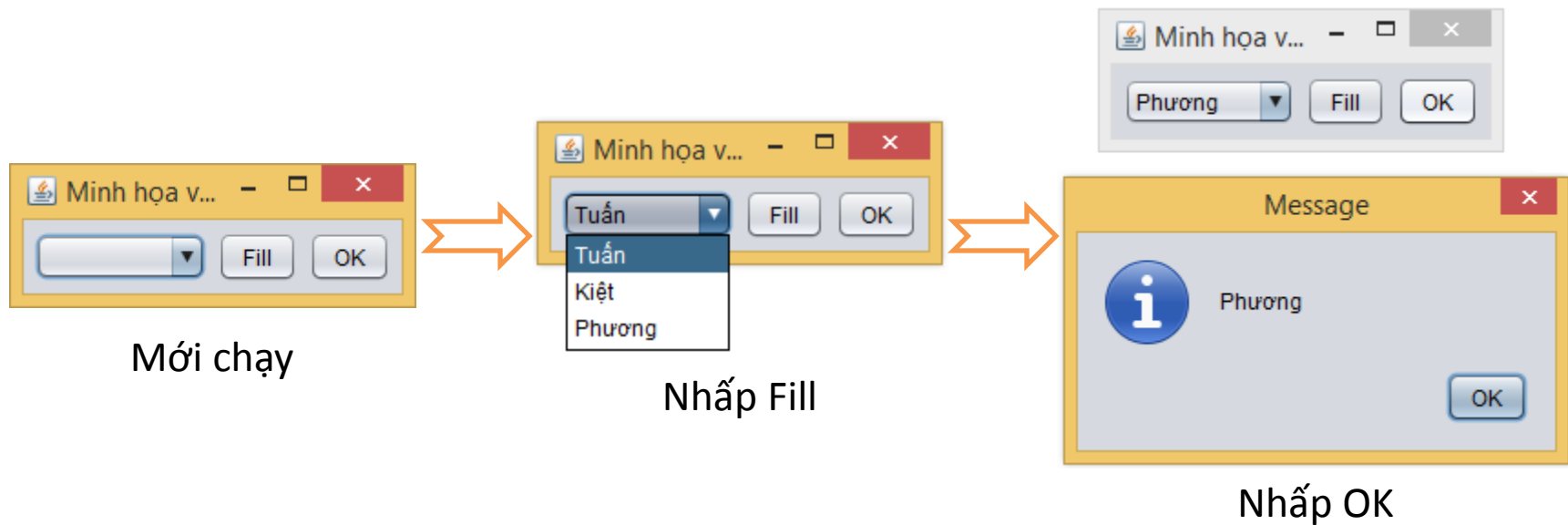
# LÀM VIỆC VỚI COMBOBOX VÀ TABLE

```
public class Student {  
    public String name;  
    public double marks;  
  
    public Student(String name, double marks) {  
        this.name = name;  
        this.marks = marks;  
    }  
}
```

```
List<Student> list = new ArrayList<>();  
list.add(new Student("Tuấn", 5.0));  
list.add(new Student("Kiệt", 7.0));  
list.add(new Student("Phương", 6.0));
```



- ❑ [Fill]: đổ dữ vào ComboBox
- ❑ [OK]: đọc lấy mục chọn

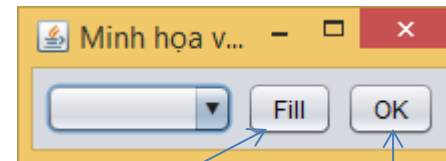


❑ Thiết kế giao diện như hình và đặt tên cho các thành phần giao diện:

❖ cboStudents

❖ btnFill

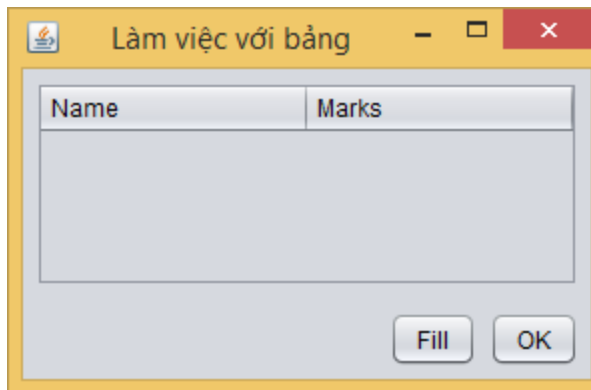
❖ btnOK



```
List<Student> list = new ArrayList<>();  
list.add(new Student("Tuấn", 5.0));  
list.add(new Student("Kiệt", 7.0));  
list.add(new Student("Phương", 6.0));  
  
cboStudents.removeAllItems();  
for(Student sv :list){  
    cboStudents.addItem(sv.name);  
}
```

```
String name = (String) cboStudents.getSelectedItem();  
JOptionPane.showMessageDialog(this, name);
```

- ❑ [Fill]: đổ dữ liệu vào bảng
- ❑ [OK]: đọc lấy giá trị cột đầu tiên của hàng được chọn

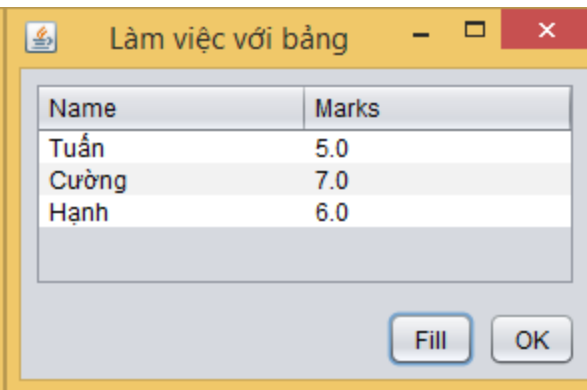


Làm việc với bảng

Name	Marks
------	-------

Fill OK

Chưa nhấn Fill

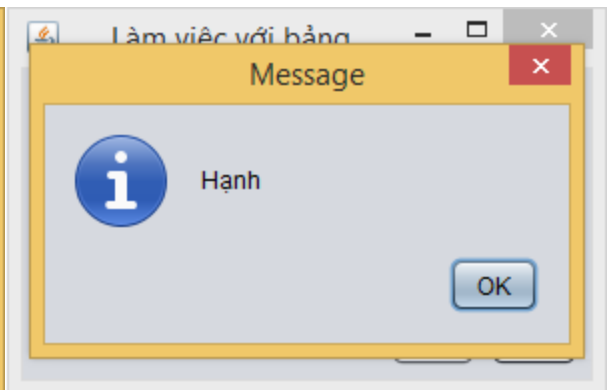


Làm việc với bảng

Name	Marks
Tuấn	5.0
Cường	7.0
Hạnh	6.0

Fill OK

Đã nhấn Fill



Message

**i** Hạnh

OK

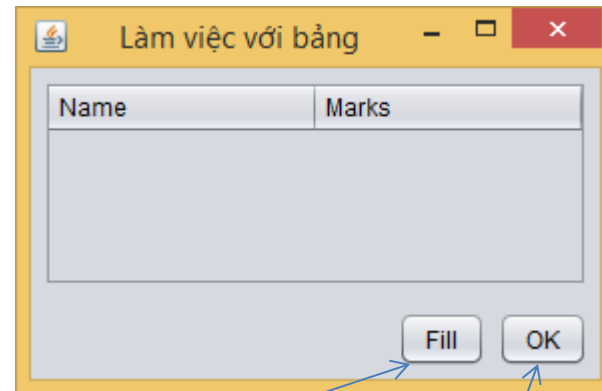
Đã nhấn OK

❑ Thiết kế giao diện như hình và đặt tên cho các thành phần giao diện:

❖ tblStudents

❖ btnFill

❖ btnLogin



```
List<Student> list = new ArrayList<>();  
list.add(new Student("Tuấn", 5.0));  
list.add(new Student("Cường", 7.0));  
list.add(new Student("Hạnh", 6.0));  
  
DefaultTableModel model =  
    (DefaultTableModel) tblStudents.getModel();  
model.setRowCount(0);  
for(Student s : list){  
    model.addRow(new Object[]{s.name, s.marks});  
}
```

```
int i = tblStudents.getSelectedRow();  
String name = (String) tblStudents.getValueAt(i, 0);  
JOptionPane.showMessageDialog(this, name);
```





**Java™**

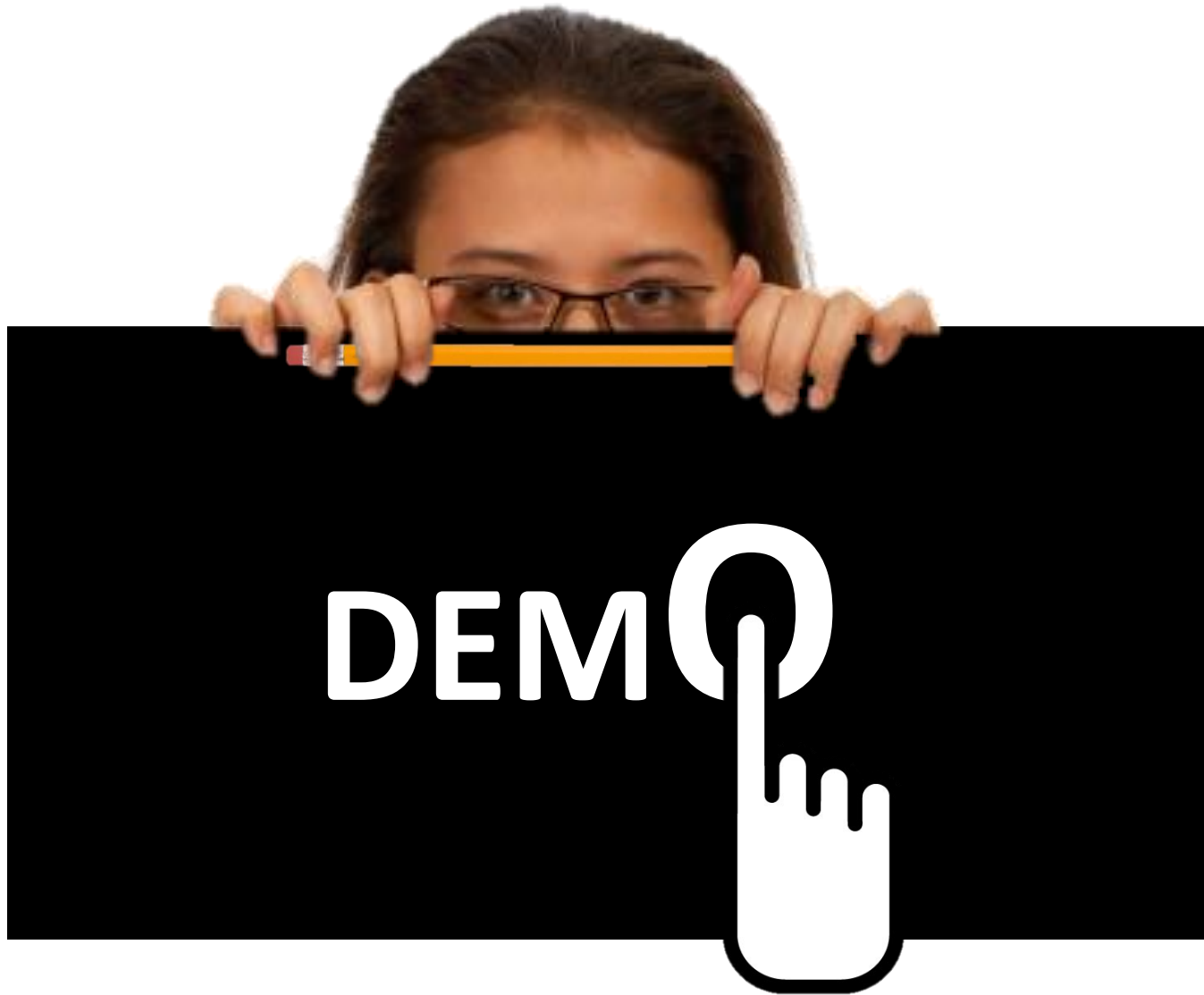
# **LẬP TRÌNH JAVA 2**

## **BÀI 2: COLLECTION & MAP**

### **PHẦN 2**

- ❑ Lớp Collections cung cấp tập các hàm tiện ích mạnh mẽ nhằm hỗ trợ xử lý List


Phương thức	Mô tả
int binarySearch (List list, Object key)	Tìm kiếm nhị phân
void fill (List list, Object obj)	Gán giá trị cho các phần tử
void shuffle (List list)	Hoán vị ngẫu nhiên
void sort (List list)	Sắp xếp tăng dần
void reverse (List list)	Đảo ngược
void rotate (List list, int distance)	Xoay vòng
void swap(List list, int i, int j)	Tráo đổi



- ❑ Để sắp xếp tập các đối tượng cần tiêu chí so sánh các đối tượng.
- ❑ Có 2 cách cung cấp tiêu chí so sánh các đối tượng
  - ❖ Cách 1: Định nghĩa tiêu chí so sánh trong class bằng cách implements interface Comparable sau đó viết mã so sánh 2 đối tượng trong phương thức compareTo(). Cách này ít được sử dụng vì khó thay đổi tiêu chí so sánh.
  - ❖ Cách 2: Tạo đối tượng từ interface Comparator sau đó cung cấp cho phương thức Collections.sort(). Cách này được sử dụng nhiều vì tính linh hoạt về tiêu chí so sánh.

# CÁCH 1: SẮP XẾP TẬP ĐỐI TƯỢNG

```
public class Student implements Comparable<Student>{  
    public String fullname;  
    public Double marks;  
  
    public Student(String fullname, Double marks) {  
        this.fullname = fullname;  
        this.marks = marks;  
    }  
    @Override  
    public int compareTo(Student other) {  
        return marks.compareTo(other.marks);  
    }  
}
```



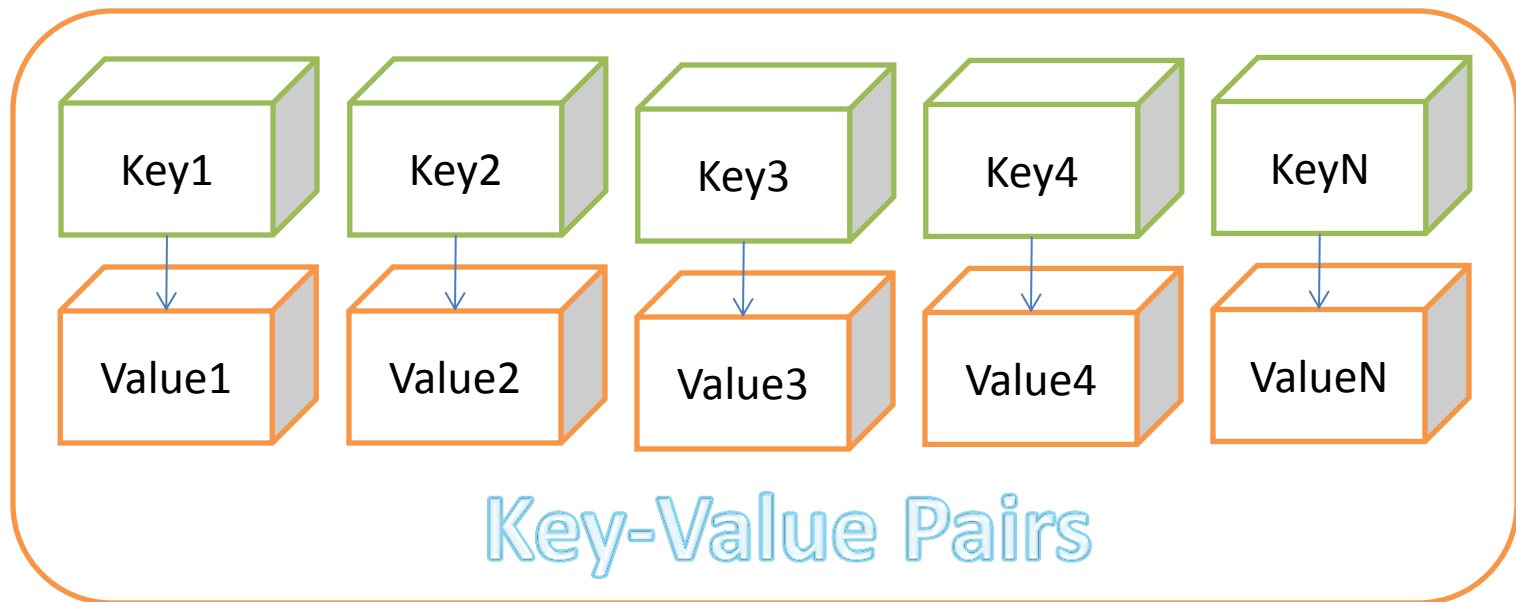
```
List<Student> list = new ArrayList<>();  
list.add(new Student("Tuấn", 5.0));  
list.add(new Student("Cường", 7.0));  
list.add(new Student("Phương", 6.0));  
  
Collections.sort(list);
```

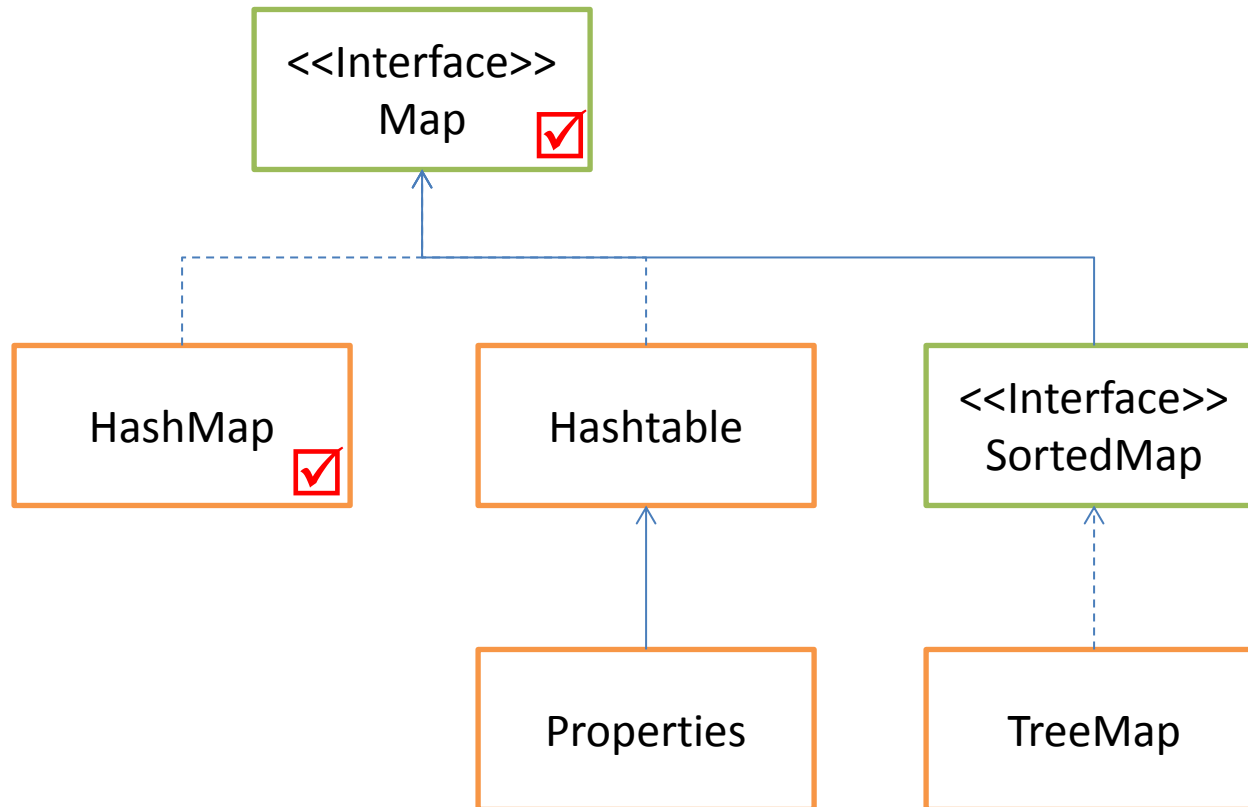
## CÁCH 2: SẮP XẾP TẬP ĐỐI TƯỢNG

```
public class Student {  
    public String fullname;  
    public Double marks;  
  
    public Student(String fullname, Double marks) {  
        this.fullname = fullname;  
        this.marks = marks;  
    }  
}
```

```
List<Student> list = new ArrayList<>();  
list.add(new Student("Tuấn", 5.0));  
list.add(new Student("Cường", 7.0));  
list.add(new Student("Phương", 6.0));  
  
Comparator<Student> com = new Comparator<Student>() {  
    @Override  
    public int compare(Student o1, Student o2) {  
        return o1.marks.compareTo(o2.marks);  
    }  
};  
Collections.sort(list, com);
```

- ❑ Map là tập hợp các entry.
- ❑ Mỗi entry gồm key và value
- ❑ Sử dụng key để truy xuất giá trị của mỗi phần tử







```
// Khai báo tập hợp các ánh xạ giữa chuỗi và số thực
Map<String, Double> map = new HashMap<String, Double>();
// bổ sung 4 cặp vào tập hợp
map.put("Nokia", 500.0);
map.put("Samsung", 600.99);
map.put("Motorola", 399.99);
map.put("Sony Ericson", 400.50);
// cập nhật giá trị của phần tử có khóa là Samsung
map.put("Samsung", 555.55);
// chuyển sang chuỗi và xuất ra
System.out.print(map.toString());
```

{Motorola=399.990, Nokia=500.000, Sony  
Ericson=400.500, Samsung=555.550}



# DEMO



Hiện thực hóa slide trước

Phương thức	Mô tả
Object <b>put</b> (Object key, Object value)	Bổ sung hoặc cập nhật một entry
Object <b>get</b> (Object key)	Lấy value theo key
Object <b>remove</b> (Object key)	Xóa một phần tử theo key
boolean <b>containsKey</b> (Object key)	Kiểm tra sự tồn tại entry theo key
int <b>size</b> ()	Lấy số lượng entry
boolean <b>isEmpty</b> ()	Kiểm tra có rỗng hay không
void <b>clear</b> ()	Xoá sạch các entry.
Set <b>keySet</b> ()	Lấy tập key
Collection <b>values</b> ()	Lấy tập value
Set <b>entrySet</b> ()	Lấy tập entry

```
Set<String> keys = map.keySet();  
for(String key: keys){  
    Double diem = map.get(key);  
}
```

```
for(Entry<String, Double> entry : map.entrySet()){  
    String ten = entry.getKey();  
    double diem = entry.getValue();  
}
```

## □ Collection

- ❖ Phân cấp thừa kế
- ❖ List & ArrayList
- ❖ Set & HashSet
- ❖ Lớp tiện ích Collections

## □ Map

- ❖ Phân cấp thừa kế
- ❖ Map & HashMap
- ❖ Properties

