

Chương 2: LẬP TRÌNH HĐT VỚI JAVA

Khoa CNTT

ĐH GTVT TP.HCM

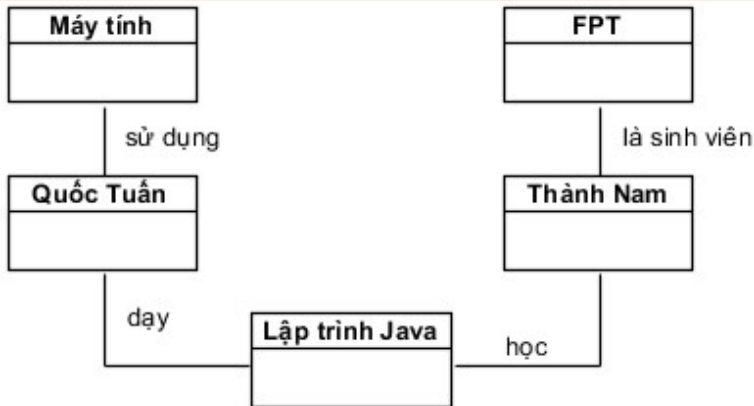
- ❶ Object oriented programming
- ❷ Class & Object
- ❸ Interface
- ❹ Collection & Map

Thế nào là OOP?

- * Thực chất lập trình là mô tả thế giới thực bằng ngôn ngữ của máy tính.
- * Thế giới thực bao gồm các sự vật, hiện tượng và sự tương tác giữa chúng.
- * Lập trình OOP hướng đến việc xây dựng phần mềm có mô hình như thế giới thực.
- * Bắt đầu từ việc xem xét các thành phần của một phần mềm như là các đối tượng (objects)
- * Và cho phép chúng (objects) tương tác với nhau để giải quyết bài toán

Object Oriented Programming (2/4)

Ví dụ OOP mô tả thế giới thực:



Các đặc điểm của OOP (1):

* Tính trừu tượng (Abstraction):

- ❶ Quốc Tuấn là một **Giảng viên**
- ❷ Thành Nam là một **Sinh viên**
- ❸ Lập trình Java là một **Học phần**

* Tính đóng gói (Encapsulation) & Che dấu dữ liệu (Data hiding)

- ❶ Thông tin sinh viên gồm: **Mã, Họ tên, Ngày sinh, ...**
- ❷ Thông tin học phần gồm: **Mã HP, Tên HP, Số tiết, ...**
- ❸ Sinh viên **không nên biết** hệ số lương của Giảng viên

Các đặc điểm của OOP (2):

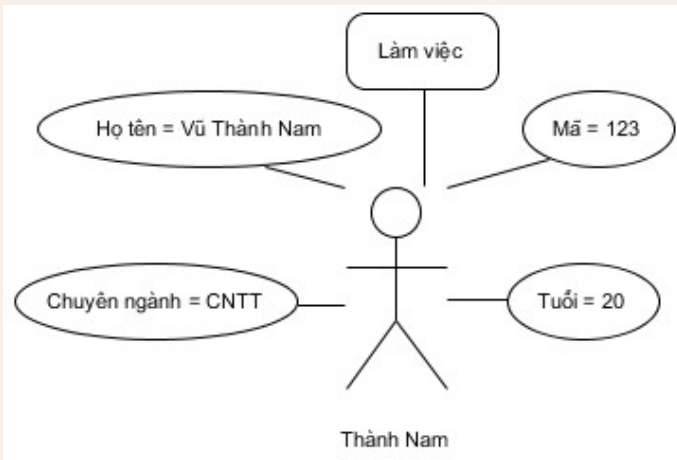
* Tính thừa kế (Inheritance)

- ❶ Giảng viên & Sinh viên đều là **Người**
- ❷ Máy tính là một loại **Thiết bị**

* Tính đa hình (Polymorphism)

- ❶ Giảng viên & Sinh viên đều **Làm việc**
- ❷ Nhưng công việc của GV là **Dạy**, trong khi với sinh viên là **Học**

Object là gì? (1)

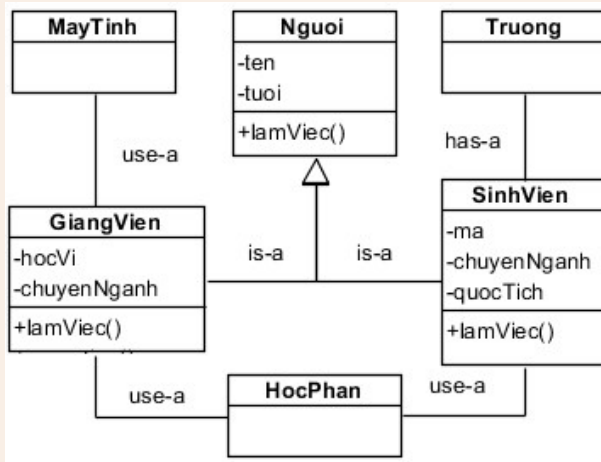


Object là gì? (2)

Object = Methods + Fields

Object & Class (3/5)

Class là khuôn mẫu để tạo ra các Objects



Object & Class (4/5)

Các vấn đề liên quan đến class & interface:

- ❶ Xây dựng class như thế nào?
- ❷ Tạo object và khởi tạo các giá trị cho các thành phần dữ liệu.
- ❸ Truy xuất đến thành phần của một object.
- ❹ Phạm vi truy xuất của các thành phần trong class
- ❺ Thành phần chung (static) cho tất cả các object của cùng một class.
- ❻ Mỗi quan hệ giữa các class & các interface (thừa kế và đa hình)

Tạo class như thế nào?

```
public class HangHoa implements Serializable {  
    public int maHH;  
    public int soLuong;  
    public double donGia;  
    public String tenHH;  
    public HangHoa() { /*initialization*/}  
    @Override  
    public String toString() {  
        return String.format("%d, %s, %f", maHH, tenHH, donGia);  
    }  
    static public ArrayList<HangHoa> dsHangHoa() {  
        ArrayList<HangHoa> lst = new ArrayList();  
        //add elements to lst  
        return lst;  
    }  
}
```

Interface (1/3)

Interface là gì?

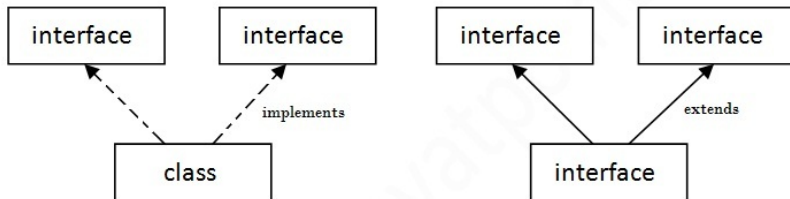
- * An interface in java is a blueprint of a class. It has static constants and abstract methods only.
- * There can be only abstract methods in the java interface not method body. It is used to achieve fully abstraction and multiple inheritance in Java.

Interface (2/3)

Tạo interface như thế nào?

```
public interface IGioHang {  
    double tinhTongTien();  
    HangHoa tim(int maHH);  
    void them(HangHoa h, int soLuong);  
    void hienThi();  
    int tinhTongHang();  
    int soMatHang();  
    void xoa(int maHH);  
}
```

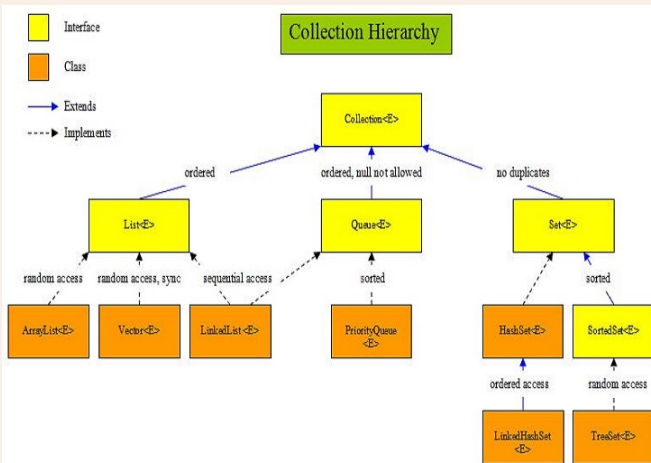
Multiple inheritance in Java by interface



Multiple Inheritance in Java

Collection in Java (1/4)

Hierarchy



List

```
public static void listDemo() {  
    List<String> lst = new ArrayList<String>();  
    lst.add("collection");  
    lst.add("in");  
    lst.add("java");  
    for (Iterator<String> i = lst.iterator(); i.hasNext();) {  
        System.out.println(i.next());  
    }  
    System.out.println(lst.indexOf("java"));  
}
```


Stack

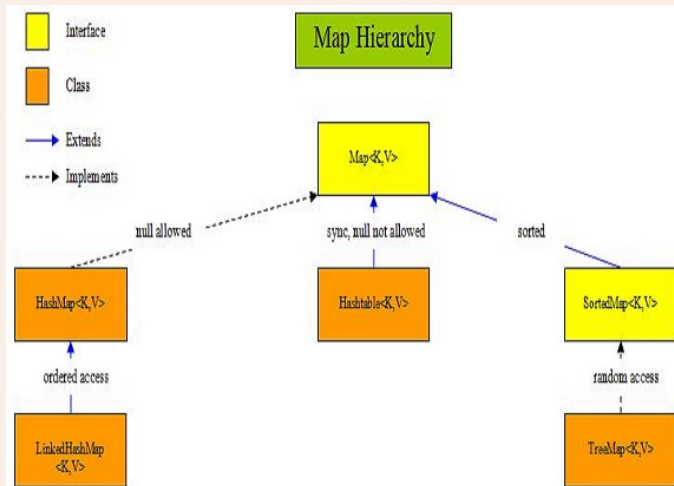
```
public static void stackDemo() {  
    Stack<String> st = new Stack<String>();  
    st.push("collection");  
    st.push("in");  
    st.push("java");  
    Iterator<String> i = st.iterator();  
    while (i.hasNext()) {  
        System.out.println(i.next());  
    }  
}
```

Set

```
public static void setDemo(){  
    Set s = new HashSet();  
    s.add("GTVT");  
    s.add("KHTN");  
    s.add("HUTECH");  
    for(Iterator i = s.iterator();i.hasNext();){  
        System.out.println(i.next());  
    }  
}
```

Map in Java (1/2)

Hierarchy



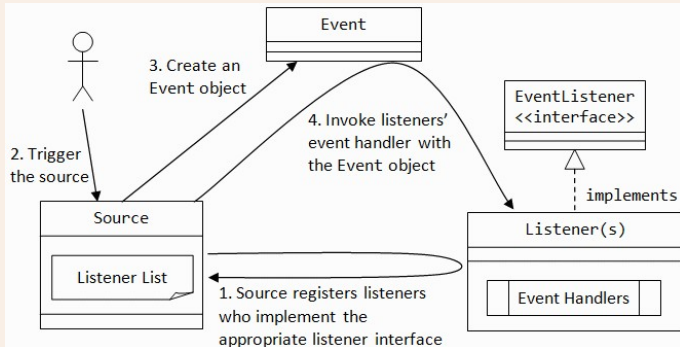
Map in Java (2/2)

Map

```
public static void mapDemo() {  
    Map m = new Hashtable<Integer, String>();  
    m.put(1, "tivi");  
    m.put(2, "laptop");  
    m.put(3, "iphone");  
    for (Object i : m.keySet()) {  
        System.out.println(m.get(i));  
    }  
}
```

Xử lý sự kiện trong Java (1/4)

Mô hình truyền sự kiện



Xử lý sự kiện trong Java (2/4)

Các thành phần trong mô hình xử lý sự kiện

- * Generator / Publisher: đối tượng phát sự kiện.
Ví dụ: rạp Galaxy phát sự kiện "có phim mới"
- * Listener / Subscriber: đối tượng lắng nghe / đăng ký sự kiện.
Ví dụ: sinh viên Tuấn thường xuyên theo dõi thông báo về phim mới trên website của Galaxy
- * Event: thông tin sự kiện được truyền từ Generator → Listener
Ví dụ: thông tin về bộ phim, diễn viên chính, xuất chiếu, giá vé, ...

Generator

```
public interface Generator {  
    void addListener(Listener listener);  
    void removeListener(Listener listener);  
}
```

Xử lý sự kiện trong Java (4/4)

Listener

```
public interface Listener {  
    void receive(Generator from, Event event);  
}
```

Event

```
public interface Event {  
    //anything you want.  
}
```


—Hết—