



Java™

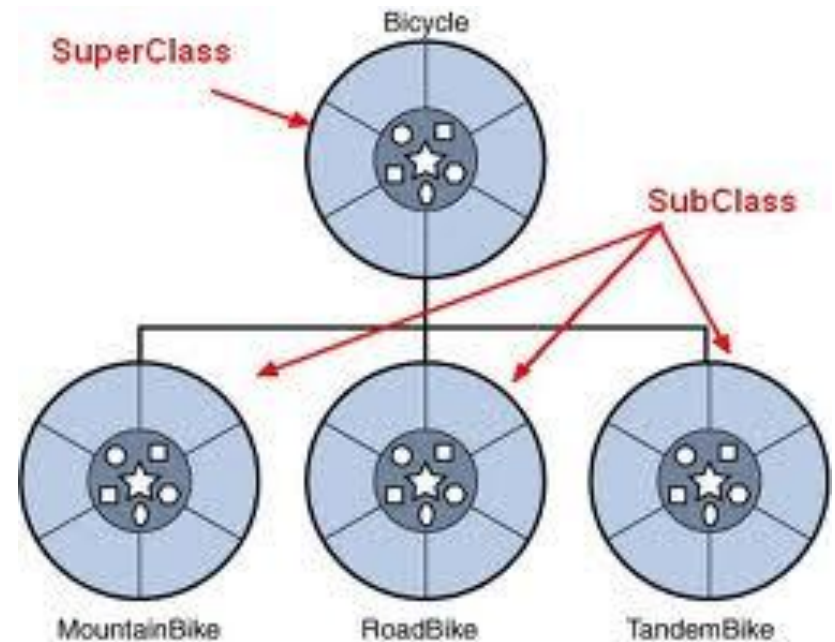
LẬP TRÌNH JAVA 2

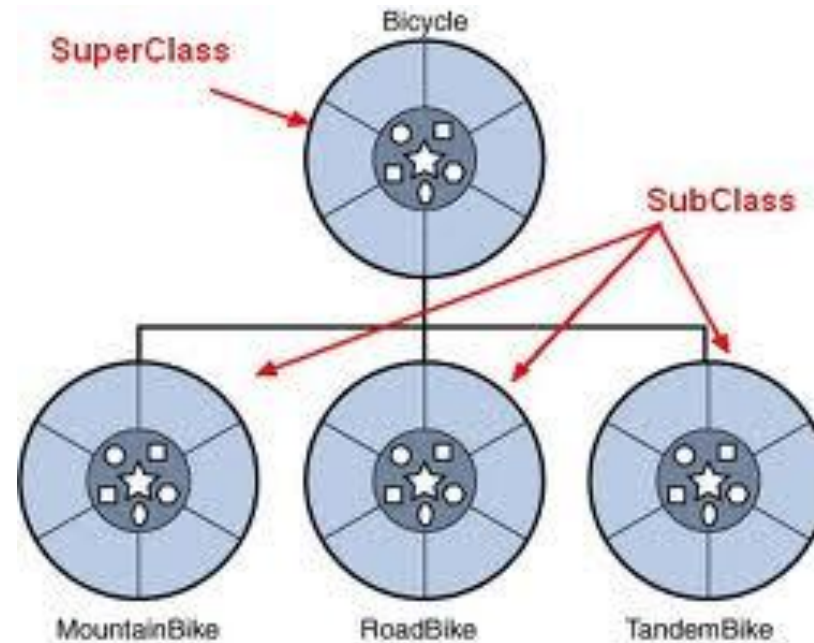
BÀI 1: KẾ THỪA

PHẦN 1

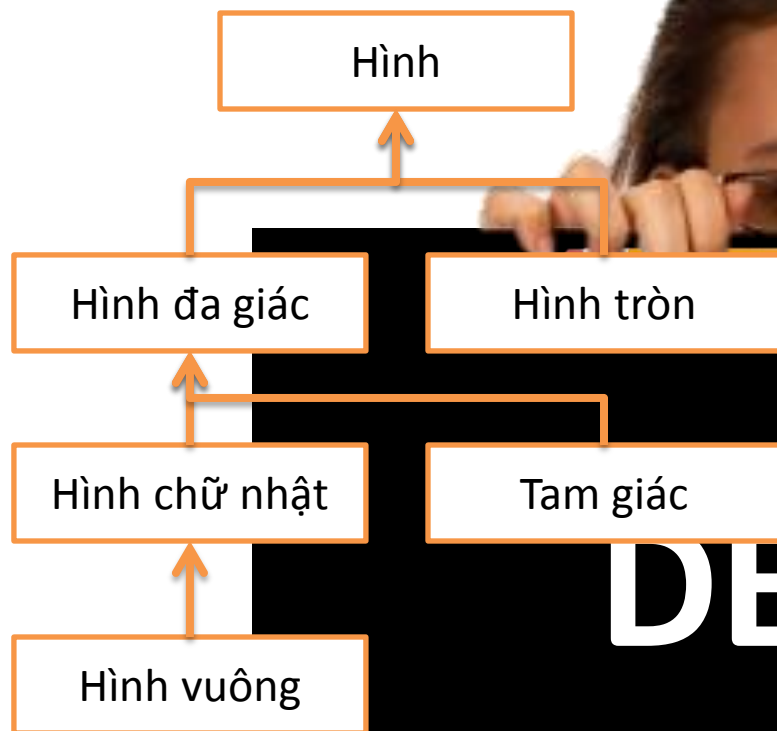
- ☐ Hiểu vững sự phân cấp thừa kế
- ☐ Tái sử dụng các lớp sẵn có
- ☐ Biết cách ghi đè phương thức
- ☐ Hiểu vững lớp và phương thức trừu tượng

- ❑ Các lớp trong Java tồn tại trong một hệ thống thứ bậc phân cấp, gọi là cây thừa kế
- ❑ Lớp bậc trên gọi là lớp cha (super class) trong khi các lớp bậc dưới gọi là lớp con (sub class)
- ❑ Trong Java một lớp chỉ có một lớp cha duy nhất (đơn thừa kế)





```
class Bicycle{...}  
class MountainBike extends Bicycle{...}  
class RoadBike extends Bicycle{...}  
class TandemBike extends Bicycle{...}
```



DEMO

Xây dựng các lớp theo cấu trúc
phân cấp kế thừa như sơ đồ

- ❑ Mục đích của thừa kế là tái sử dụng.
- ❑ Lớp con được phép sở hữu các tài sản (trường và phương thức) của lớp cha
 - ❖ Lớp con được phép sở hữu các tài sản public hoặc protected của lớp cha
 - ❖ Lớp con cũng được phép sở hữu các tài sản mặc định {default} của lớp cha nếu lớp con và lớp cha được định nghĩa cùng gói
 - ❖ Lớp con không thể truy cập thành viên private của lớp cha

```
package poly.ho;
public class NhanVien{
    public String hoTen;
    protected double luong;
    public NhanVien(String hoTen, double luong){...}
    void xuat(){...}
    private double thueThuNhap(){...}
}
```

- A. super.hoTen
- B. super.luong
- C. super.xuat()
- D. super.thueThuNhap()

```
package poly.hcm;
public class TruongPhong extends NhanVien{
    public double trachNhiem;
    public TruongPhong_(String hoTen, double luong, double trachNhiem){...}
    public void xuat(){
        // Mã ở đây có thể sử dụng những tài sản nào của lớp cha
    }
}
```



DEMO

Hiện thực hóa ví dụ của slide trước



- ❑ Truy cập đến các thành viên của lớp cha bằng cách sử dụng từ khóa `super`
- ❑ Có thể sử dụng `super` để gọi hàm tạo của lớp cha

```
public class Parent{  
    public String name;  
    public void method(){}  
}
```

```
public class Child extends Parent{  
    public String name;  
    public void method(){  
        this.name = super.name;  
        super.method()  
    }  
}
```

```
package poly.ho;
public class NhanVien{
    public NhanVien(String hoTen, double luong){...}
        public void xuat(){...}
}
```

```
package poly.hcm;
public class TruongPhong extends NhanVien{
    public double trachNhiem;
    public TruongPhong (String hoTen, double luong, double trachNhiem){
        super(hoTen, luong);
        this.trachNhiem = trachNhiem
    }
    public void xuat(){
        super.xuat()
        System.out.println(trachNhiem)
    }
}
```

- ❑ Overriding là trường hợp lớp con và lớp cha có phương thức cùng cú pháp.

```
public class Parent{  
    public void method() {...}  
}
```

```
public class Child{  
    public void method() {...}  
}
```

- ❑ Lớp Parent và Child đều có phương thức `method()` cùng cú pháp nên `method()` trong Child sẽ đè lên `method()` trong Parent

```
Parent o = new Child();  
o.method()
```

Mặc dù `o` có kiểu là Parent nhưng `o.method()` thì `method()` của lớp Child sẽ chạy do bị đè

- ☐ Lớp con ghi đề phương thức của lớp cha thì sẽ che dấu phương thức của lớp cha
- ☐ Mục đích của ghi đề là để sửa lại phương thức của lớp cha trong lớp con
- ☐ Sử dụng từ khóa super để truy cập đến phương thức đã bị ghi đề của lớp cha.
- ☐ Đặc tả truy xuất của phương thức lớp con phải có độ công khai **bằng hoặc hơn** đặc tả truy xuất của phương thức lớp cha.

NhanVien

+ hoTen
+ luong
+ getThuNhap()

TruongPhong

+ trachNhiem
+ getThuNhap()

LaoCong

+ soGioLamViec
+ getThuNhap()

DEMO

Lương của NhanVien, Trưởng phòng, Lao công... được tính theo công thức khác nhau. Ví dụ nhân viên là lương tháng, lao công thì lương giờ, trưởng phòng còn có lương trách nhiệm



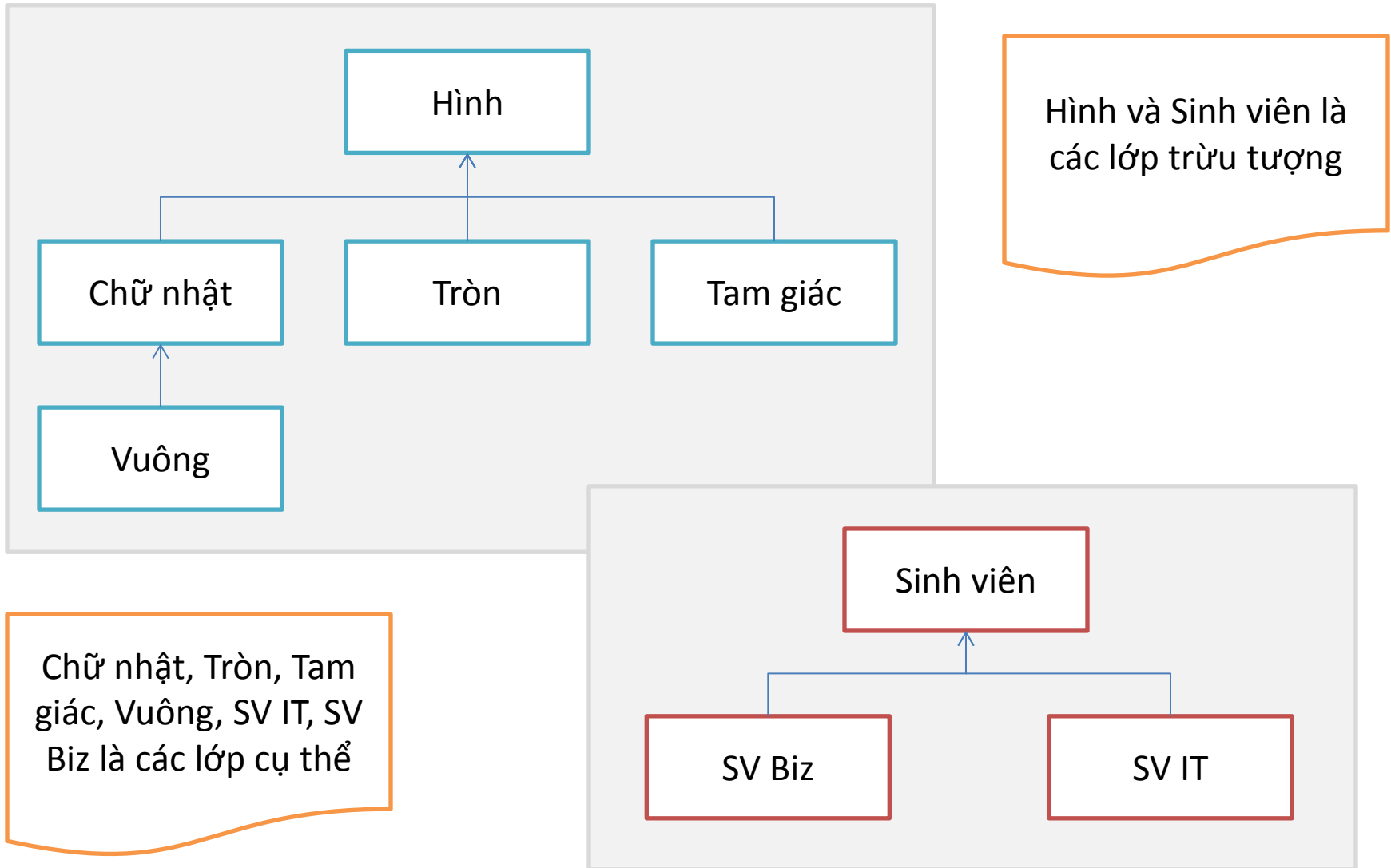
Java™

LẬP TRÌNH JAVA 2

BÀI 1: KẾ THỪA

PHẦN 2

- ❑ Lớp trừu tượng là lớp có các hành vi chưa được xác định rõ
 - ❖ Ví dụ 1: Đã là hình thì chắc chắn là có diện tích và chu vi nhưng chưa xác định được cách tính mà phải là một hình cụ thể như chữ nhật, tròn, tam giác... mới có thể xác định cách tính
 - ❖ Ví dụ 2: Sinh viên thì chắc chắn có điểm trung bình nhưng chưa xác định được cách tính như thế nào mà phải là sinh viên của ngành nào mới biết được môn học và công thức tính điểm cụ thể.
- ❑ Vậy lớp Hình và lớp SinhVien là các lớp trừu tượng vì phương thức tính chu vi, diện tích và tính điểm chưa thực hiện được.




```
abstract public class MyClass{  
    abstract public type MyMethod();  
}
```

Sử dụng từ khóa
abstract để định
nghĩa lớp và
phương thức trừu
tượng

```
abstract public class SinhVien{  
    abstract public double getDiemTB();  
}
```

```
abstract public class Hinh{  
    abstract public double getChuVi();  
    abstract public double getDienTich();  
}
```

```
abstract public class SinhVien{  
    public String hoTen;  
    abstract public double getDiemTB();  
}
```

```
public class SinhVienIT extends SinhVien{  
    public double diemJava;  
    public double diemCss;  
    @Override  
    public double getDiemTB(){  
        return (2 * diemJava + diemCss)/3;  
    }  
}
```

```
public class SinhVienBiz extends SinhVien {  
    public double keToan;  
    public double marketting;  
    public double banHang;  
    @Override  
    public double getDiemTB(){  
        return  
            (keToan + marketting + banHang)/3;  
    }  
}
```

- ❑ Từ khóa **abstract** được sử dụng để định nghĩa lớp và phương thức trừu tượng
- ❑ Phương thức trừu tượng là phương thức không có phần thân xử lý và được khai báo bằng từ khóa abstract.
- ❑ Lớp chứa phương thức trừu tượng thì lớp đó phải là lớp trừu tượng.
- ❑ Trong lớp trừu tượng có thể định nghĩa các phương thức cụ thể hoặc các trường



DEMO

Hiện thực hóa mô hình
thừa kế ở slide trước về Hình



- ❑ Overriding thực hiện tính đa hình trong lập trình hướng đối tượng (một hành vi được thể hiện với các hình thái khác nhau)
- ❑ Gọi phương thức bị ghi đè được quyết định lúc chạy chương trình (runtime) chứ không phải lúc biên dịch chương trình (compile time)



TÍNH ĐA HÌNH (POLYMORPHISM)

```
abstract public class DongVat{  
    abstract public void speak();  
}
```

```
public class Cho extends DongVat{  
    public void speak() {  
        System.out.println("Woof");  
    }  
}
```

```
public class Meo extends DongVat{  
    public void speak() {  
        System.out.println("Meo");  
    }  
}
```

```
public class Vit extends DongVat{  
    public void speak() {  
        System.out.println("Quack");  
    }  
}
```

```
DongVat cho = new Cho();  
DongVat meo = new Meo();  
DongVat vit = new Vit();
```

```
cho.speak();  
meo.speak();  
vit.speak();
```

- ❑ Trong interface chỉ có các method **abstract** và các biến **final**
- ❑ Khi một class **thực thi** một interface, nó phải viết lại (**override**) tất cả các method trong interface.
- ❑ Interface là public hoặc default
- ❑ Interface có thể được kế thừa
- ❑ Một interface có thể được thực hiện bởi **nhiều** class, và một class có thể thực thi **nhiều** interface. Đó chính là cách để dùng “kế thừa từ nhiều class”.

```
package mypackage;  
  
interface MyInterface {  
    void myMethod1 ();  
    void myMethod2 ();  
}
```



```
package mypackage;
```

```
public class MyClass implements MyInterface {  
    public void myMethod1() { //phải là public  
        System.out.println("Override my method 1");  
    }
```

```
    public void myMethod2() {  
        System.out.println("Override my method 2");  
    }
```

```
// method riêng của class
```

```
void mymethod3() {  
    System.out.println("My method 3");  
}
```

```
}
```

```
public static void main (String a[]) {  
  
    MyClass myObject1 = new MyClass ();  
    obj1.mymethod1 ();  
    obj1.mymethod2 ();  
    obj1.mymethod3 ();  
  
    MyInterface myObject2 = new MyClass ();  
    obj2.mymethod1 ();  
    obj2.mymethod2 ();  
    obj2.mymethod3 (); // error;  
  
}  
  
}
```

```
public interface MyInter1{
    void meth1();
    void meth2();
}

public interface MyInter2{
    void meth3();
}

public class MyClass implements MyInter1, MyInter2 {
    public void meth1() {
        System.out.println("Implements method 1");
    }
    public void meth2() {
        System.out.println("Implements method 2");
    }
    public void meth3() {
        System.out.println("Implements method 3");
    }
}
```

```
public interface MyInter1{
    void meth1();
}
public interface MyInter2{
    void meth2();
}
public interface MyInter3 extends MyInter1, MyInter2{
    void meth3();
}
public class MyClass implements MyInter3 {
    public void meth1() {
        System.out.println("Implements method 1");
    }
    public void meth2() {
        System.out.println("Implements method 2");
    }
    public void meth3() {
        System.out.println("Implements method 3");
    }
}
```

- ☐ Thừa kế
- ☐ Gọi hàm tạo của lớp cha
- ☐ Sử dụng super
- ☐ Ghi đè phương thức
- ☐ Lớp và phương thức trừu tượng

