



LẬP TRÌNH JAVA 1

BÀI 4: LỚP VÀ ĐỐI TƯỢNG

- ❑ Kết thúc bài học này bạn có khả năng
 - ❖ Hiểu rõ khái niệm đối tượng và lớp
 - ❖ Mô hình hóa lớp và đối tượng
 - ❖ Định nghĩa được lớp và tạo đối tượng
 - ❖ Định nghĩa các trường, phương thức
 - ❖ Định nghĩa và sử dụng hàm tạo
 - ❖ Hiểu và sử dụng package
 - ❖ Sử dụng thành thạo các đặc tả truy xuất
 - ❖ Hiểu được tính che dấu (encapsulation)

- ❑ Biểu diễn đối tượng trong thế giới thực
- ❑ Mỗi đối tượng được đặc trưng bởi các thuộc tính và các hành vi riêng của nó



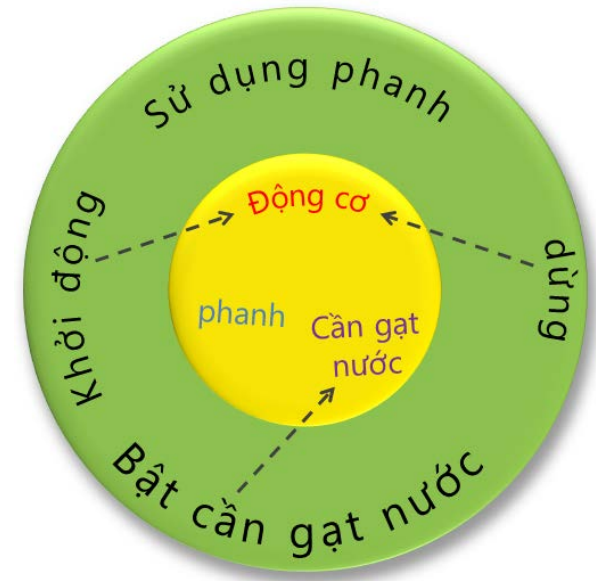
□ Đặc điểm

- Hãng sản xuất
- Model
- Năm
- Màu



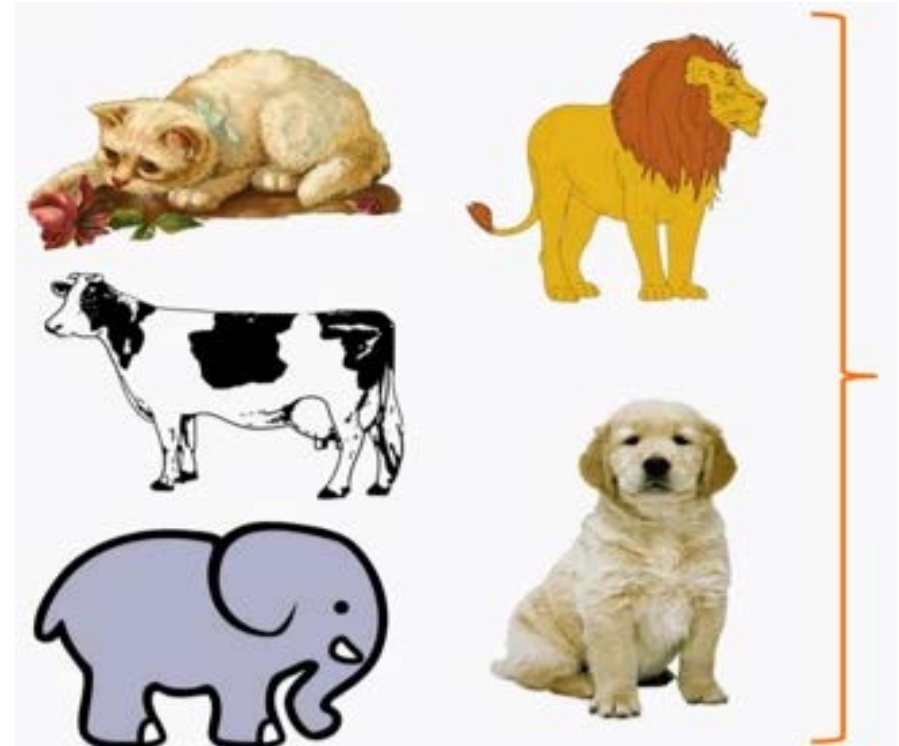
□ Hành vi (Ô tô có thể làm gì?)

- Khởi động
- Dừng
- Phanh
- Bật cần gạt nước



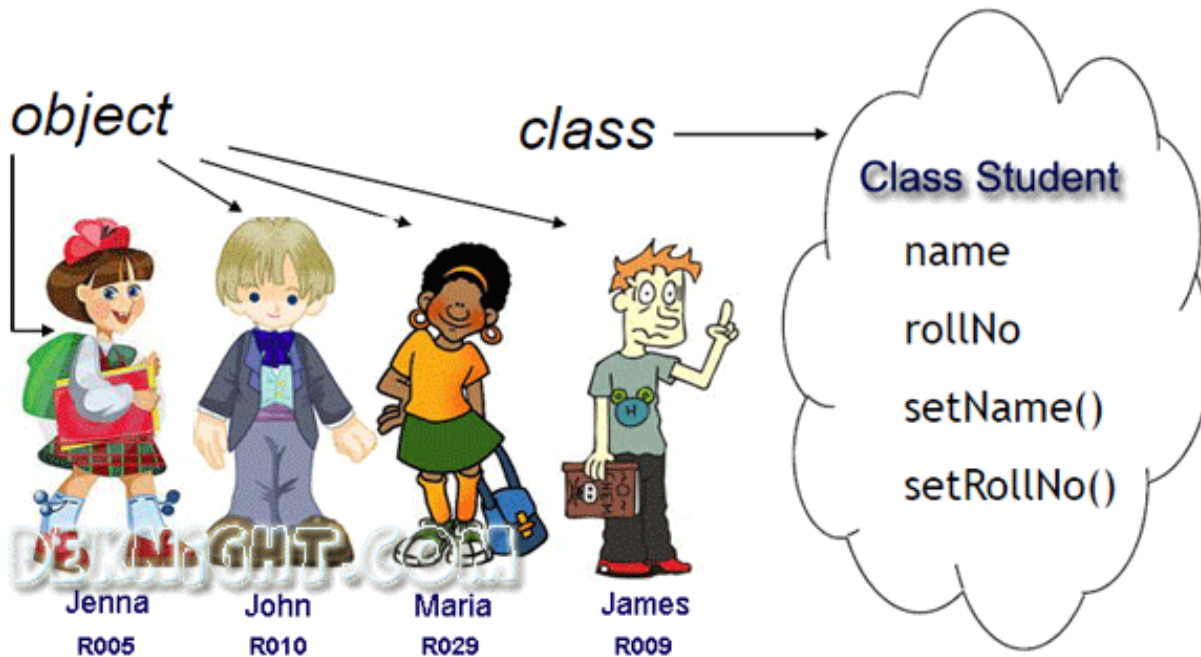


Nhóm các **Xe ô-tô**



Nhóm các **Động vật**

- ❑ Lớp là một khuôn mẫu được sử dụng để mô tả các đối tượng cùng loại.
- ❑ Lớp bao gồm các thuộc tính (trường dữ liệu) và các phương thức (hàm thành viên)



❑ Thuộc tính (field)

- ❖ Hãng sản xuất
- ❖ Model
- ❖ Năm
- ❖ Màu

Danh từ

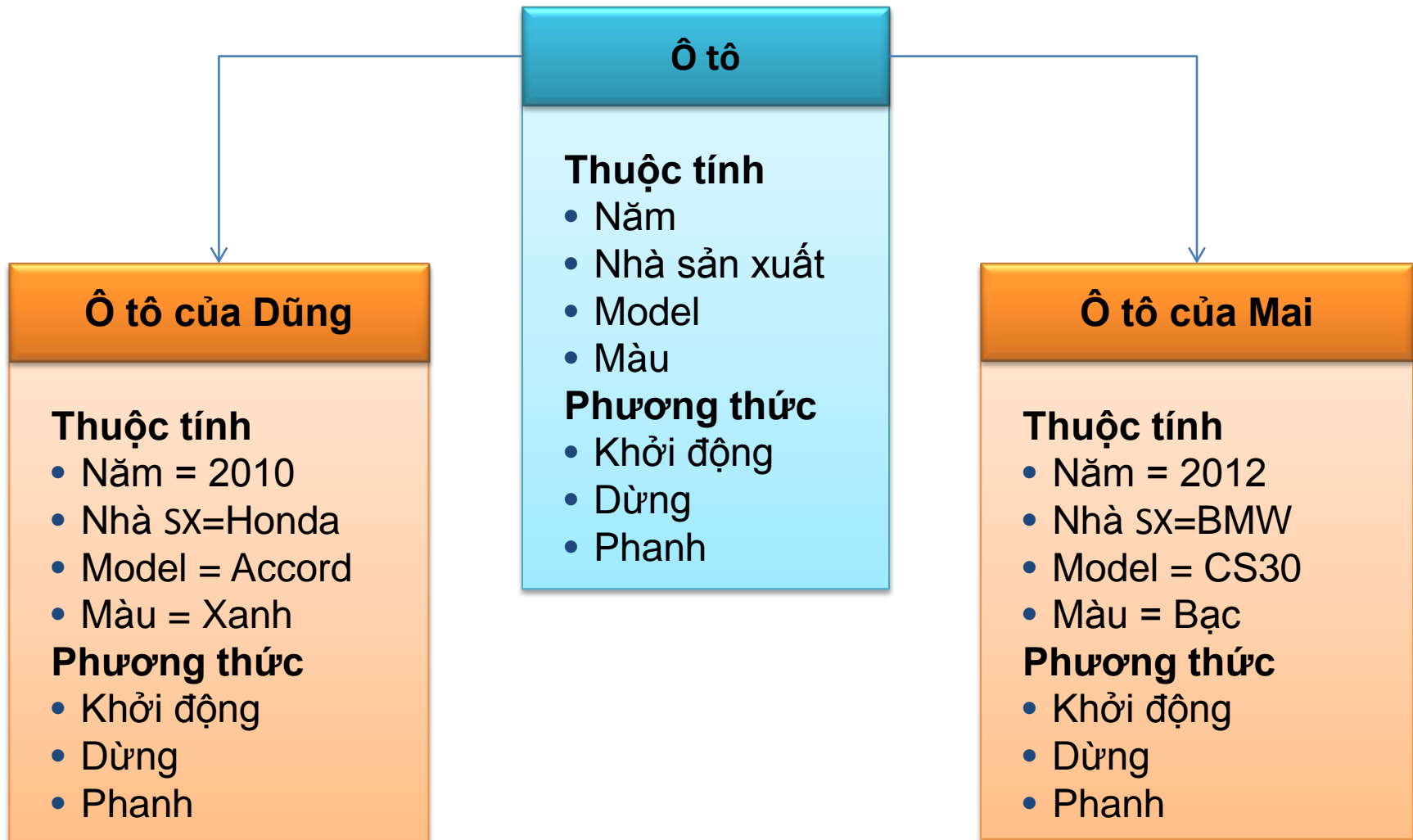


❑ Phương thức (method)

- ❖ Khởi động()
- ❖ Dừng()
- ❖ Phanh()
- ❖ Bật cần gạt nước()

Động từ





- ❑ Abstraction là công việc lựa chọn các thuộc tính và hành vi của thực thể vừa đủ để mô tả thực thể đó trong một bối cảnh cụ thể mà không phải liệt kê tất cả các thuộc tính, hành vi của thực thể có.
- ❑ Ví dụ: Mô tả một sinh viên ngành CNTT có rất nhiều thuộc tính và hành vi. Ở đây chúng ta chỉ sử dụng mã, họ và tên, điểm, ngành mà thôi, không cần thiết phải mô tả ~~cao, nặng, hát, cười, nhảy cò cò...~~

```
class <<ClassName>>
```

```
{
```

```
<<type>> <<field1>>;
```

Khai báo các trường

```
...
```

```
<<type>> <<fieldN>>;
```

Khai báo các phương thức

```
<<type>> <<method1>>([parameters]) {
```

```
// body of method
```

```
}
```

```
...
```

```
<<type>> <<methodN>>([parameters]) {
```

```
// body of method
```

```
}
```

```
}
```

```
public class Employee{  
    public String fullname;  
    public double salary;  
  
    public void input(){  
        Scanner scanner = new Scanner(System.in);  
        System.out.print(" >> Full Name: ");  
        this.fullname = scanner.nextLine();  
  
        System.out.print(" >> Salary: ");  
        this.salary = scanner.nextDouble();  
    }  
  
    public void output(){  
        System.out.println(this.fullname);  
        System.out.println(this.salary);  
    }  
}
```

Trường



Phương thức

Lớp Employee có 2 thuộc tính là fullname và salary và 2 phương thức là input() và output()

- ❑ Đoạn mã sau sử dụng lớp Employee để tạo một nhân viên sau đó gọi các phương thức của lớp.

```
public static void main(String[] args) {  
  
    Employee emp = new Employee();  
    emp.input();  
    emp.output();  
}
```



❑ Chú ý:

- ❖ Toán tử **new** được sử dụng để tạo đối tượng
- ❖ Biến emp chứa tham chiếu tới đối tượng
- ❖ Sử dụng dấu chấm (.) để truy xuất các thành viên của lớp (trường và phương thức).



DEMO

Tạo lớp mô tả sinh viên bao gồm họ
tên, điểm và các phương thức nhập,
xuất và xếp loại học lực



- ❑ Phương thức là một mô-đun mã thực hiện một công việc cụ thể nào đó
 - ❖ Trong lớp Employee có 2 phương thức là input() và output()
- ❑ Phương thức có thể có một hoặc nhiều tham số
- ❑ Phương thức có thể có kiểu trả về hoặc void (không trả về gì cả)
- ❑ Cú pháp

```
<<kiểu trả về>> <<tên phương thức>> ([danh sách tham số])  
{  
    // thân phương thức  
}
```

```
public class Employee{
    public String fullname;
    public double salary;
```

```
    public void input(){...}
    public void output(){...}
```

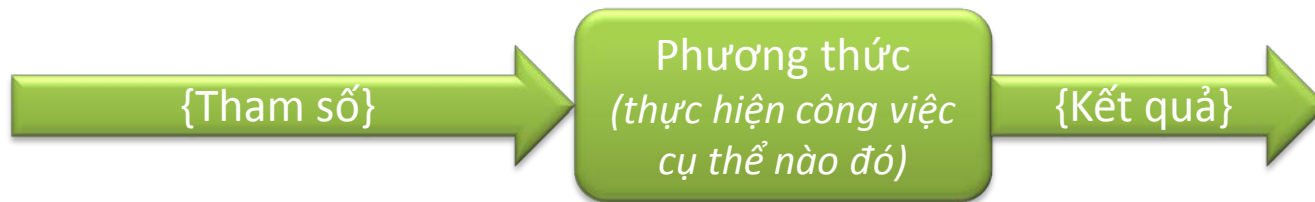
Kiểu trả về là **void** nên thân phương thức **không chứa lệnh return giá trị**

```
    public void setInfo(String fullname, double salary) {
        this.fullname = fullname;
        this.salary = salary;
    }
```

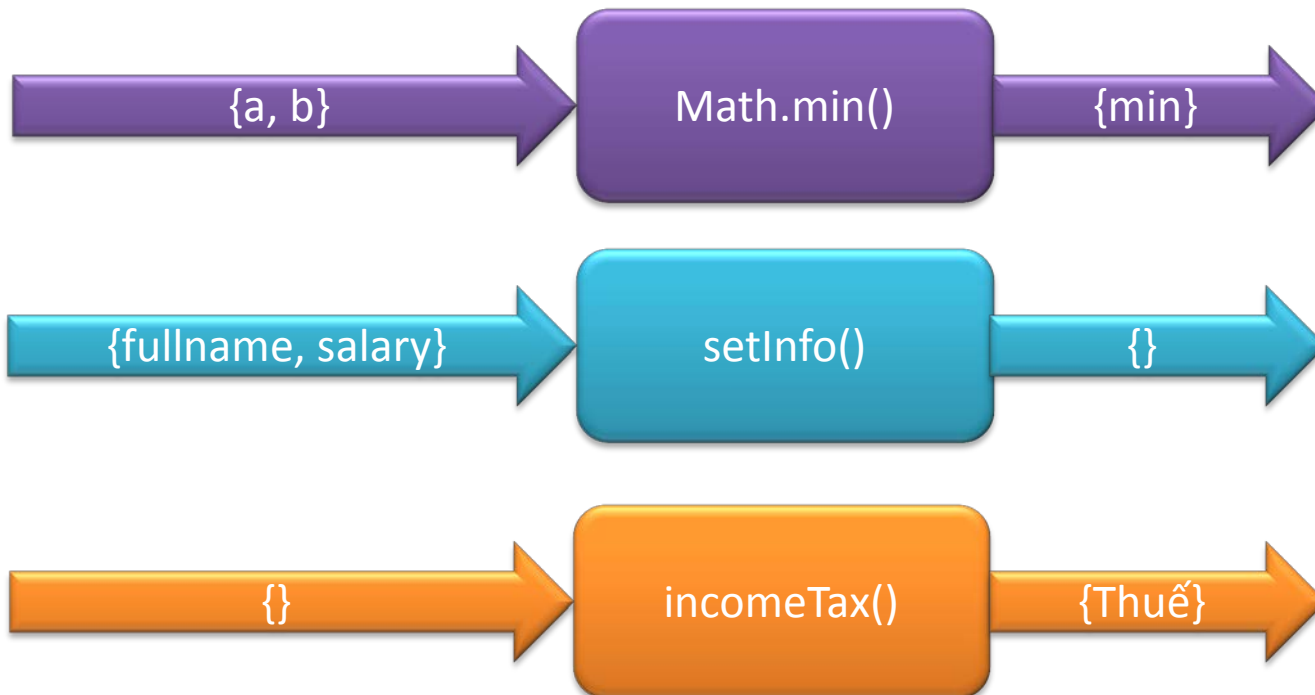
Kiểu trả về là **double** nên thân phương thức phải chứa lệnh **return số thực**

```
    public double incomeTax(){
        if(this.salary < 5000000){
            return 0;
        }
        double tax = (this.salary - 5000000) * 10/100;
        return tax;
    }
}
```

□ Mô hình



□ Ví dụ



NẠP CHỒNG PHƯƠNG THỨC (OVERLOADING)

- ❑ Trong một lớp có thể có nhiều phương thức cùng tên nhưng khác nhau về tham số (kiểu, số lượng và thứ tự)

```
public class MyClass{  
    void method()  
    void method(int x)  
    void method(float x)  
    void method(int x, double y)  
}
```

- ❑ Trong lớp MyClass có 4 phương thức cùng tên là method nhưng khác nhau về tham số

- ❑ Xét trường hợp overload sau

```
class MayTinh{  
    int tong(int a, int b){return a + b;}  
    int tong(int a, int b, int c){return a + b + c;}  
}
```

- ❑ Với lớp trên, bạn có thể sử dụng để tính tổng 2 hoặc 3 số nguyên.

```
MayTinh mt = new MayTinh();  
int t1 = mt.tong(5, 7);  
int t2 = mt.tong(5, 7, 9);
```

- ❑ Hàm tạo là một phương thức đặc biệt được sử dụng để tạo đối tượng.
- ❑ Đặc điểm của hàm tạo
 - ❖ Tên trùng với tên lớp
 - ❖ Không trả lại giá trị
- ❑ Ví dụ

Lớp

```
public class ChuNhat{  
    double dai, rong;  
    public ChuNhat(double dai, double rong){  
        this.dai = dai;  
        this.rong = rong;  
    }  
}
```

Đối tượng

```
ChuNhat cn1 = new ChuNhat(20, 15);  
ChuNhat cn2 = new ChuNhat(50, 25);
```


- ❑ Trong một lớp có thể định nghĩa nhiều hàm tạo khác tham số, mỗi hàm tạo cung cấp 1 cách tạo đối tượng.
- ❑ Nếu không khai báo hàm tạo thì Java tự động cung cấp hàm tạo mặc định (không tham số)

```
public class ChuNhat{  
    double dai, rong;  
    public ChuNhat(double dai, double rong){  
        this.dai = dai;  
        this.rong = rong;  
    }  
    public ChuNhat(double canh){  
        this.dai = canh;  
        this.rong = canh;  
    }  
}
```

```
ChuNhat cn = new ChuNhat(20, 15);  
ChuNhat vu= new ChuNhat(30);
```

- ❑ **this** được sử dụng để đại diện cho đối tượng hiện tại.
- ❑ **this** được sử dụng trong lớp để tham chiếu tới các thành viên của lớp (field và method)
- ❑ Sử dụng **this.field** để phân biệt field với các biến cục bộ hoặc tham số của phương thức

```
public class MyClass{  
    int field;  
    void method(int field){  
        this.field = field;  
    }  
}
```



The diagram consists of two orange-outlined boxes with speech bubble tails pointing to the **this.field** expression in the code. The box on the left points to **this** and contains the text 'Trường' (Field). The box on the right points to **field** and contains the text 'Tham số' (Parameter).



SinhVien

+ hoTen: String
+ diemTB: double

+ xepLoai(): String
+ xuat(): void
+ nhap(): void

+ SinhVien()
+ SinhVien(hoTen, diemTB)

DEMO



Xây dựng lớp mô tả sinh viên như mô hình trên.
Trong đó nhap() cho phép nhập họ tên và điểm từ bàn phím; xuat() cho phép xuất họ tên, điểm và học lực ra màn hình; xepLoai() dựa vào điểm để xếp loại học lực
Sử dụng 2 hàm tạo để tạo 2 đối tượng sinh viên

- ❑ Package được sử dụng để chia các class và interface thành từng gói khác nhau.
 - ❖ Việc làm này tương tự quản lý file trên ổ đĩa trong đó class (file) và package (folder)
- ❑ Ví dụ sau tạo lớp MyClass thuộc gói com.poly

```
package com.poly;  
public class MyClass{...}
```
- ❑ Trong Java có rất nhiều gói được phân theo chức năng
 - ❖ java.util: chứa các lớp tiện ích
 - ❖ java.io: chứa các lớp vào/ra dữ liệu
 - ❖ java.lang: chứa các lớp thường dùng...

- ❑ Lệnh import được sử dụng để chỉ ra lớp đã được định nghĩa trong một package
- ❑ Các lớp trong gói java.lang và các lớp cùng định nghĩa trong cùng một gói với lớp sử dụng sẽ được import ngầm định

```
package com.polyhcm;  
import com.poly.MyClass;  
import java.util.Scanner;  
public class HelloWorld{  
    public static void main(String[] args){  
        MyClass obj = new MyClass();  
        Scanner scanner = new Scanner(System.in);  
    }  
}
```


❑ Đặc tả truy xuất được sử dụng để định nghĩa khả năng cho phép truy xuất đến các thành viên của lớp. Trong java có 4 đặc tả khác nhau:

❖ **private**: chỉ được phép sử dụng nội bộ trong class

❖ **public**: công khai hoàn toàn

❖ **{default}**:

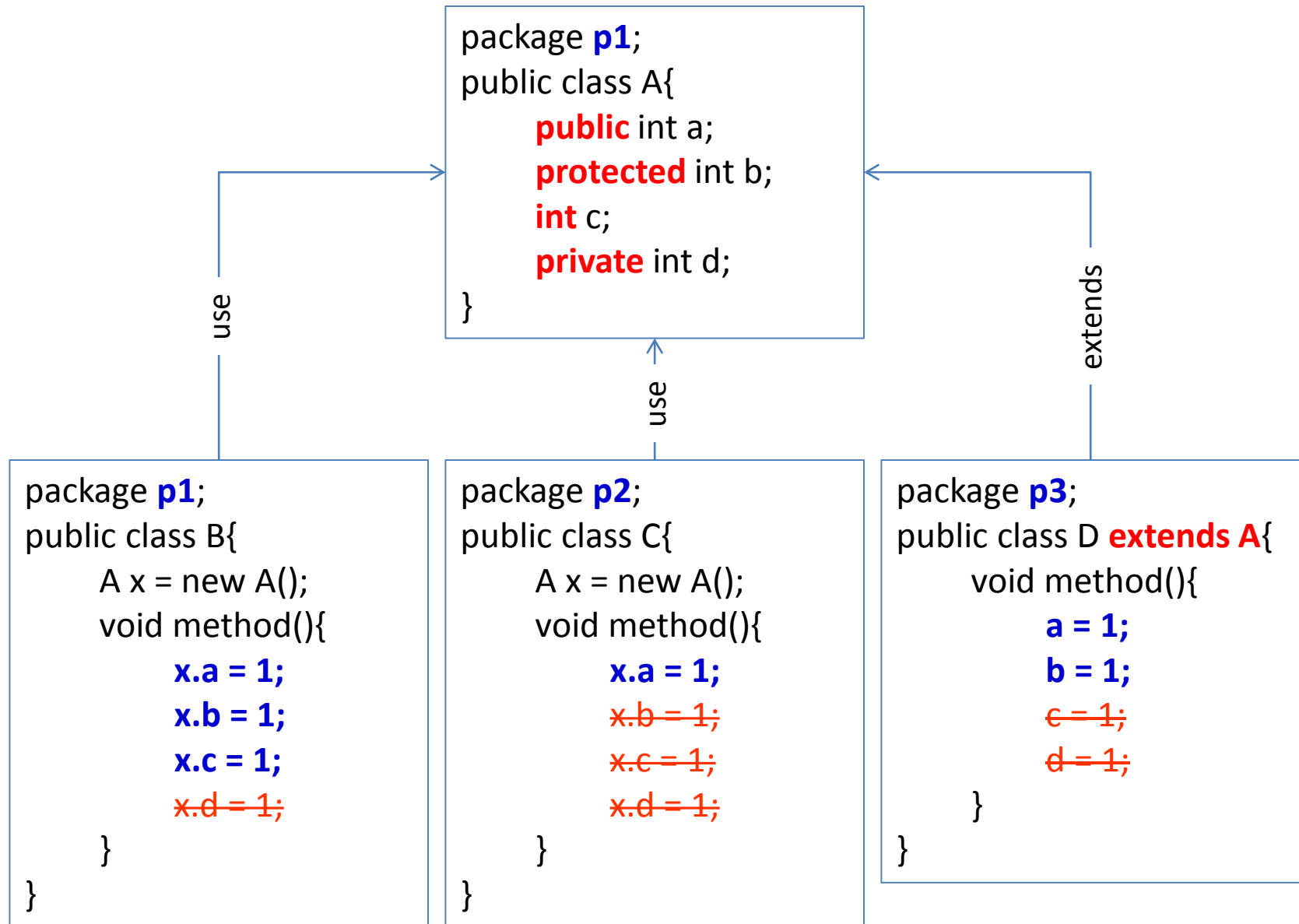
➤ Là public đối với các lớp truy xuất cùng gói

➤ Là private với các lớp truy xuất khác gói.

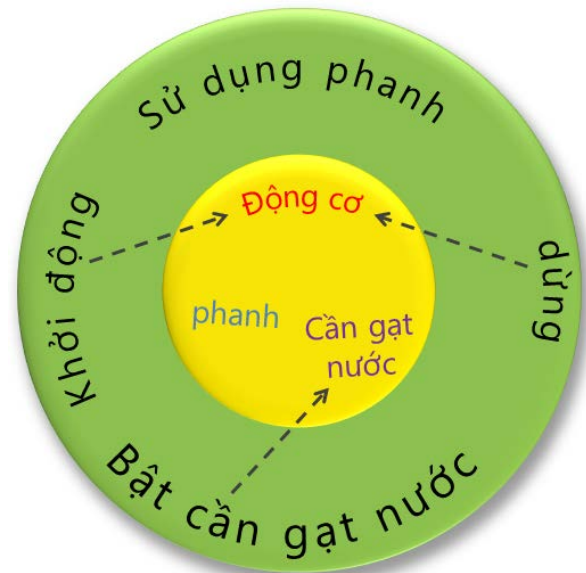
❖ **protected**: tương tự {default} nhưng cho phép kế thừa dù lớp con và cha khác gói.

❑ Mức độ che dấu tăng dần theo chiều mũi tên

public → **protected** → **{default}** → **private**



- ❑ Encapsulation là tính che dấu trong hướng đối tượng.
 - ❖ Nên che dấu các trường dữ liệu
 - ❖ Sử dụng phương thức để truy xuất các trường dữ liệu
- ❑ Mục đích của che dấu
 - ❖ Bảo vệ dữ liệu
 - ❖ Tăng cường khả năng mở rộng



- ❑ Giả sử định nghĩa lớp SinhVien và công khai hoTen và điểm như sau

```
public class SinhVien{  
    public String hoTen;  
    public double diem;  
}
```

```
public class MyClass{  
    public static void main(String[] args){  
        SinhVien sv = new SinhVien();  
        sv.hoTen = "Nguyễn Văn Tèo";  
        sv.diem = 20.5;  
    }  
}
```

- ❑ Khi sử dụng người dùng có thể gán dữ liệu cho các trường một cách tùy tiện
- ❑ Điều gì sẽ xảy ra nếu điểm hợp lệ chỉ từ 0 đến 10

- ❑ Để che dấu thông tin, sử dụng private cho các trường dữ liệu.

private double diem;

- ❑ Bổ sung các phương thức getter và setter để đọc ghi các trường đã che dấu

```
public void setDiem(double diem){  
    this.diem = diem;  
}  
  
public String getDiem() {  
    return this.diem;  
}
```

```
public class SinhVien{
    private String hoTen;
    private double diem;
    public void setHoTen(String hoTen){
        this.hoTen = hoTen;
    }
    public String getHoTen(){
        return this.hoTen;
    }
    public void setDiem(double diem){
        if(diem < 0 || > 10){
            System.out.println("Điểm không hợp lệ");
        }
        else{
            this.diem = diem;
        }
    }
    public String getDiem(){
        return this.diem;
    }
}
```

❑ Chỉ cần thêm mã vào phương thức setDiem() để có những xử lý khi dữ liệu không hợp lệ

```
public class MyClass{
    public static void main(String[] args){
        SinhVien sv = new SinhVien();
        sv.setHoTen("Nguyễn Văn Tèo");
        sv.setDiem(20);
    }
}
```

- ❑ Tên (class, field, method, package, interface, variable) được đặt theo qui ước (mềm) như sau:
 - ❖ Tên package: toàn bộ ký tự thường và dấu chấm
 - java.util, com.poly
 - ❖ Tên class, interface: Các từ phải viết hoa ký tự đầu
 - class **E**mployee{}, class **S**inh**V**ien{}, class **H**inh**C**hu**N**hat()
 - ❖ Tên field, method, variable: Các từ phải viết hoa ký tự đầu ngoại trừ từ đầu tiên phải viết thường
 - **h**o**T**en, diem, **f**ull**N**ame, **m**ark
 - **s**et**H**o**T**en(), **i**nput(), **s**et**D**iem()
- ❑ Tên class, field và variable sử dụng danh từ
- ❑ Tên phương thức sử dụng động từ

- ☐ Khái niệm về đối tượng
- ☐ Khái niệm lớp
- ☐ Mô hình đối tượng và lớp
- ☐ Định nghĩa lớp
- ☐ Tạo đối tượng
- ☐ Định nghĩa phương thức
- ☐ Nạp chồng phương thức
- ☐ Hàm tạo
- ☐ Package
- ☐ Đặc tả truy xuất
- ☐ Encapsulation
- ☐ Quy ước đặt tên

