

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**  
**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# **LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

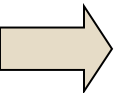
## **Bài 05. Kết tập và kế thừa**

Nguyễn Thị Thu Trang  
[trangntt-fit@mail.hut.edu.vn](mailto:trangntt-fit@mail.hut.edu.vn)

# Mục tiêu bài học

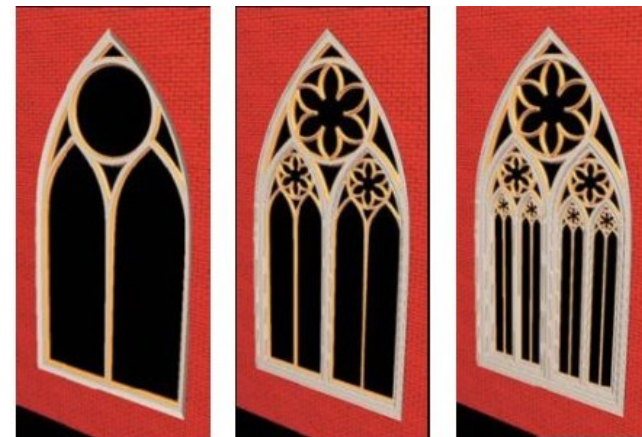
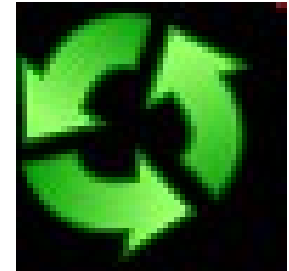
- Giải thích về khái niệm tái sử dụng mã nguồn
- Chỉ ra được bản chất, mô tả các khái niệm liên quan đến đến kết tập và kế thừa
- So sánh kết tập và kế thừa
- Biểu diễn được kết tập và kế thừa trên UML
- Giải thích nguyên lý kế thừa và thứ tự khởi tạo, hủy bỏ đối tượng trong kế thừa
- Áp dụng các kỹ thuật, nguyên lý về kết tập và kế thừa trên ngôn ngữ lập trình Java

# Nội dung

- 
1. Tái sử dụng mã nguồn
  2. Kết tập (Aggregation)
  3. Kế thừa (Inheritance)

# 1. Tái sử dụng mã nguồn (Re-usability)

- Tái sử dụng mã nguồn: Sử dụng lại các mã nguồn đã viết
  - Lập trình cấu trúc: Tái sử dụng hàm/chương trình con
  - OOP: Khi mô hình thế giới thực, tồn tại nhiều loại đối tượng có các thuộc tính và hành vi tương tự hoặc liên quan đến nhau  
→ *Làm thế nào để tái sử dụng lớp đã viết?*



# 1. Tái sử dụng mã nguồn (2)

- Các cách sử dụng lại lớp đã có:
  - *Sao chép* lớp cũ thành 1 lớp khác ◇ Dư thừa và khó quản lý khi có thay đổi
  - Tạo ra lớp mới là sự *tập hợp* hoặc *sử dụng các đối tượng* của lớp cũ đã có ◇ Kết tập (Aggregation)
  - Tạo ra lớp mới trên cơ sở *phát triển* từ lớp cũ đã có ◇ Kế thừa (Inheritance)

# Ưu điểm của tái sử dụng mã nguồn

- Giảm thiểu công sức, chi phí
- Nâng cao chất lượng phần mềm
- Nâng cao khả năng mô hình hóa thế giới thực
- Nâng cao khả năng bảo trì (maintainability)

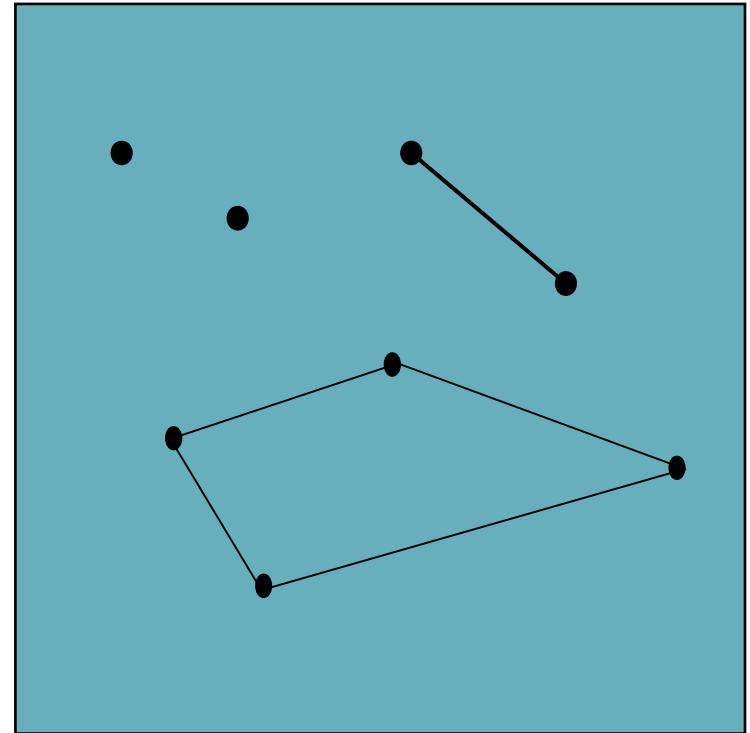


# Nội dung

1. Tái sử dụng mã nguồn
2. Kết tập (Aggregation)
3. Kế thừa (Inheritance)

## 2. Kết tập

- Ví dụ:
  - Điểm
    - Tứ giác gồm 4 điểm  
→ Kết tập
- Kết tập
  - Quan hệ chứa/có (“has-a”) hoặc là một phần (is-a-part-of)



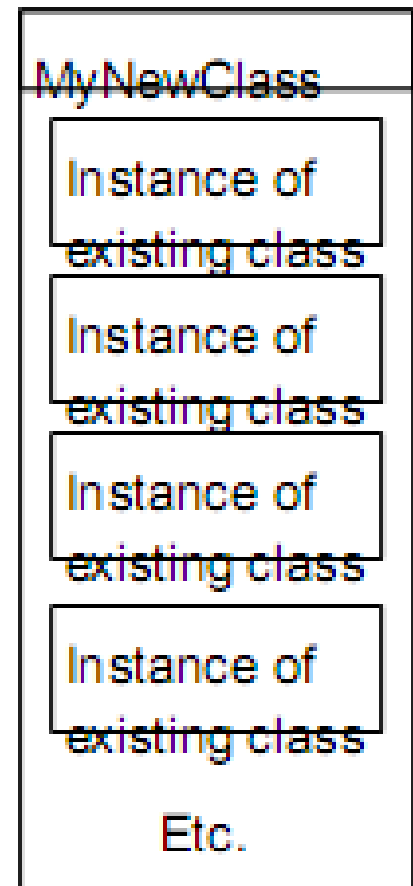


## 2.1. Bản chất của kết tập

- Kết tập (aggregation)
  - Tạo ra các đối tượng của các lớp có sẵn trong lớp mới → thành viên của lớp mới.
  - Kết tập tái sử dụng thông qua *đối tượng*
- Lớp mới
  - Lớp toàn thể (Aggregate/Whole),
- Lớp cũ
  - Lớp thành phần (Part).

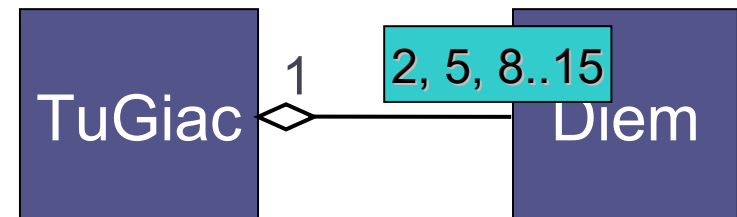
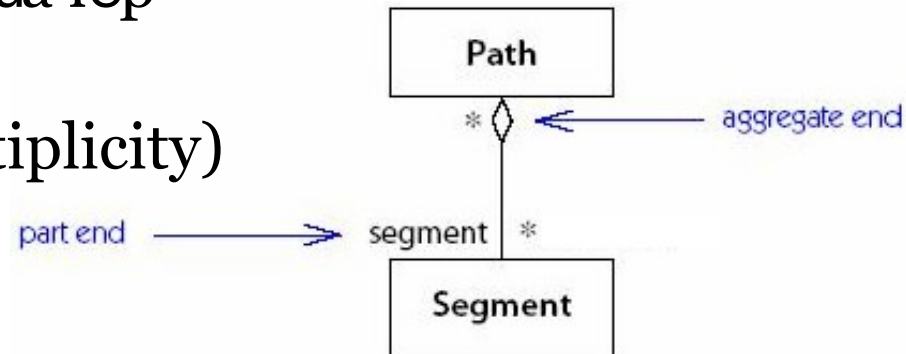
## 2.1. Bản chất của kết tập (2)

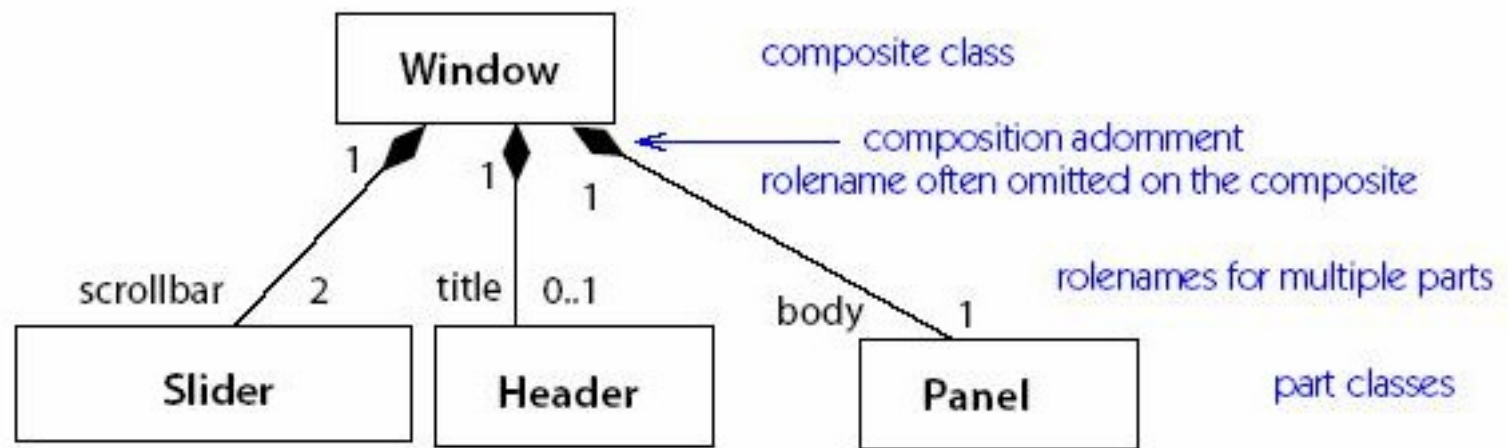
- Lớp toàn thể chứa đối tượng của lớp thành phần
  - Là một phần (is-a-part of) của lớp toàn thể
  - Tái sử dụng các thành phần dữ liệu và các hành vi của lớp thành phần thông qua đối tượng thành phần



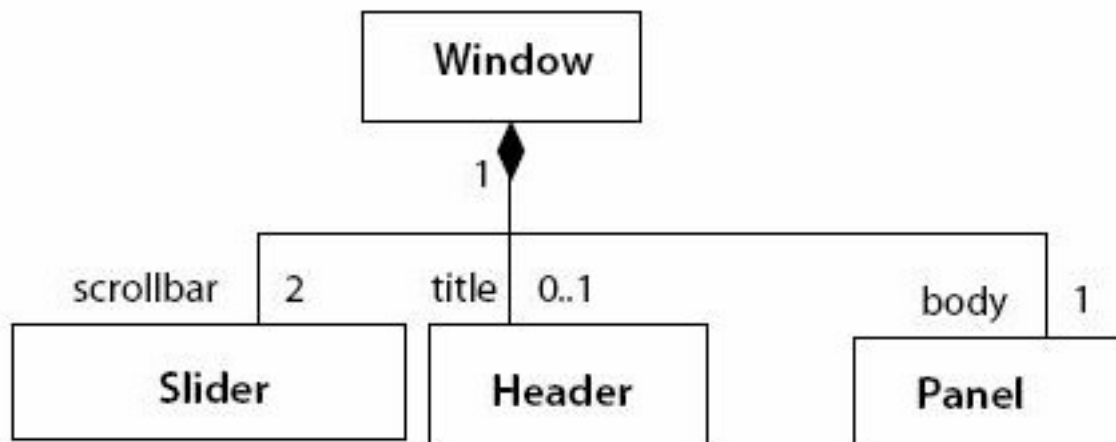
## 2.2. Biểu diễn kết tập bằng UML

- Sử dụng “hình thoi” tại đầu của lớp toàn thể
- Sử dụng bội số quan hệ (multiplicity) tại 2 đầu
  - 1 số nguyên dương: 1, 2,...
  - Dải số (0..1, 2..4)
  - \*: Bất kỳ số nào
  - Không có: Mặc định là 1
- Tên vai trò (rolename)
  - Nếu không có thì mặc định là tên của lớp (bỏ viết hoa chữ cái đầu)





composition notation: oblique paths



alternate notation: grouping paths as a tree

Ví dụ

## 2.3. Minh họa trên Java

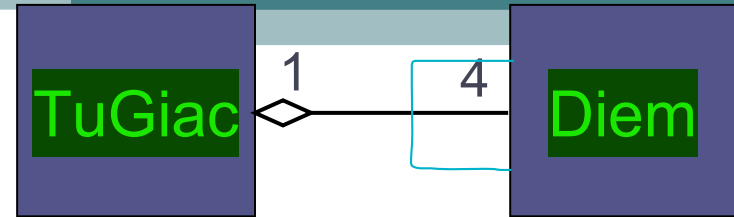
```
class Diem {  
    private int x, y;  
    public Diem() {}  
    public Diem(int x, int y) {  
        this.x = x; this.y = y;  
    }  
    public void setX(int x) { this.x = x; }  
    public int getX() { return x; }  
    public void hienThiDiem() {  
        System.out.print("(" + x + ", "  
                           + y + ")");  
    }  
}
```

```

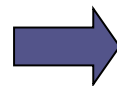
class TuGiac {
    private Diem d1, d2;
    private Diem d3, d4;
    public TuGiac(Diem p1, Diem p2,
                  Diem p3, Diem p4) {
        d1 = p1; d2 = p2; d3 = p3; d4 = p4;
    }
    public TuGiac() {
        d1 = new Diem();      d2 = new Diem(0,1);
        d3 = new Diem (1,1);  d4 = new Diem (1,0);
    }
    public void printTuGiac() {
        d1.printDiem();  d2.printDiem();
        d3.printDiem();  d4.printDiem();
        System.out.println();
    }
}

```

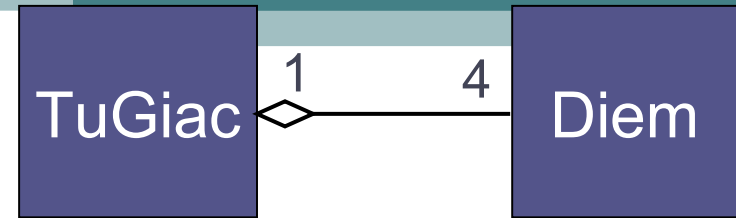
Sau



```
public class Test {  
    public static void main(String arg[])  
    {  
        Diem d1 = new Diem(2,3);  
        Diem d2 = new Diem(4,1);  
        Diem d3 = new Diem (5,1);  
        Diem d4 = new Diem (8,4);  
  
        TuGiac tg1 = new TuGiac(d1, d2, d3, d4);  
        TuGiac tg2 = new TuGiac();  
        tg1.printTuGiac();  
        tg2.printTuGiac();  
    }  
}
```

A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The window contains the following text:  
<2, 3><4, 1><5, 1><8, 4>  
<0, 0><0, 1><1, 1><1, 0>  
Press any key to continue . . .  
The window has standard Windows controls (minimize, maximize, close) and a scrollbar on the right.

# Cách cài đặt khác

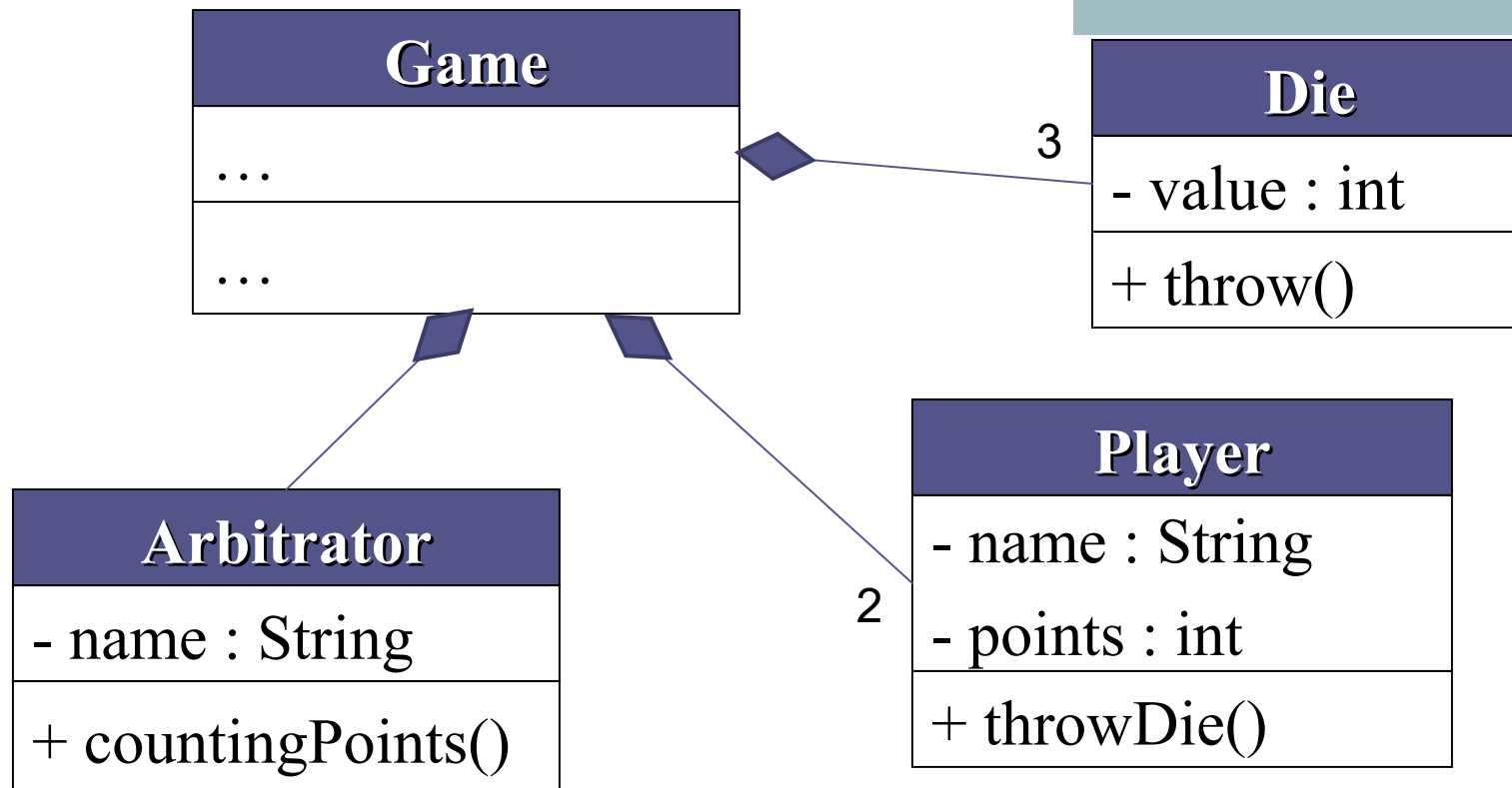


```
class TuGiac {
    private Diem[] diem = new Diem[4];
    public TuGiac(Diem p1, Diem p2,
                  Diem p3, Diem p4) {
        diem[0] = p1; diem[1] = p2;
        diem[2] = p3; diem[3] = p4;
    }
    public void printTuGiac() {
        diem[0].printDiem(); diem[1].printDiem();
        diem[2].printDiem(); diem[3].printDiem();
        System.out.println();
    }
}
```



## Ví dụ khác về Kết tập

- Một trò chơi gồm 2 đối thủ, 3 quân súc sắc và 1 trọng tài.
  - Cần 4 lớp:
    - Người chơi (Player)
    - Súc sắc (Die)
    - Trọng tài (Arbitrator)
    - Trò chơi (Game)
- ◇ Lớp Trò chơi là lớp kết tập của 3 lớp còn lại



```

class Game
{
    Die die1, die2, die3;
    Player player1, player2;
    Arbitrator arbitrator1;
    ...
}
  
```

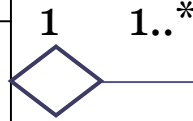
## 2.4. Thứ tự khởi tạo trong kết tập

- Khi một đối tượng được tạo mới, các thuộc tính của đối tượng đó đều phải được khởi tạo và gán những giá trị tương ứng.
  - Các đối tượng thành phần được khởi tạo trước
- ◇ Các phương thức khởi tạo của các lớp của các đối tượng thành phần được thực hiện trước

**PhongBan**

-tenPhongBan:String  
 -soNhanVien:byte  
+SO\_NV\_MAX:byte = 100

+themNV(NhanVien):boolean  
 +xoaNV():NhanVien  
 +tongLuong():double  
 +inTTin()

**NhanVien**

-tenNhanVien:String  
 -heSoLuong:double  
+LUONG\_CO\_BAN:double=750.000  
+LUONG\_MAX:double=20.000.000

+tangLuong(double):boolean  
 +tinhLuong():double  
 +inTTin()

- Viết mã nguồn cho lớp **PhongBan** với các thuộc tính và phương thức như biểu đồ trên cùng phương thức khởi tạo với số lượng tham số cần thiết, biết rằng:
  - Việc thêm/xóa nhân viên được thực hiện theo cơ chế của stack
  - tongLuong()** trả về tổng lương của các nhân viên trong phòng.
  - inTTin()** hiển thị thông tin của phòng và thông tin của các nhân viên trong phòng.

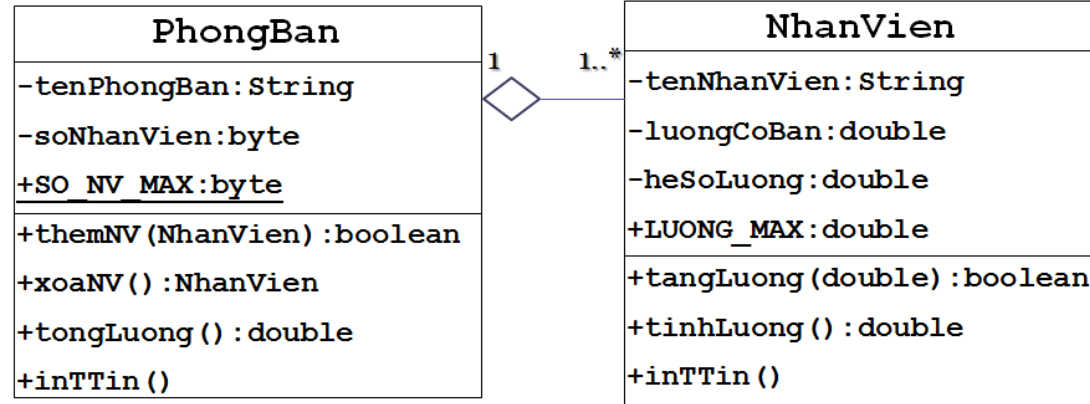
```
public class PhongBan {
    private String tenPhongBan; private byte soNhanVien;
    public static final SO_NV_MAX = 100;
    private NhanVien[] dsnv;
    public boolean themNhanVien(NhanVien nv) {
        if (soNhanVien < SO_NV_MAX) {
            dsnv[soNhanVien] = nv; soNhanVien++;
            return true;
        } else return false;
    }
    public boolean xoaNhanVien(NhanVien nv) {
        if (soNhanVien > 0) {
            NhanVien tmp = dsnv[soNhanVien-1];
            dsnv[soNhanVien-1] = null; soNhanVien--;
            return dsnv[soNhanVien];
        } else return null;
    }
    // (cont) ...
}
```

```
public PhongBan(String tenPB){
    dsnv = new NhanVien[SO_NV_MAX];
    tenPhongBan = tenPB; soNhanVien = 0;
}
public double tongLuong(){
    double tong = 0.0;
    for (int i=0;i<soNhanVien;i++)
        tong += dsnv[i].tinhLuong();
    return tong;
}
public void inTTin(){
    System.out.println("Ten phong: "+tenPhong);
    System.out.println("So NV: "+soNhanVien);
    System.out.println("Thong tin cac NV");
    for (int i=0;i<soNhanVien;i++)
        dsnv[i].inTTin();
}
}
```

# Thảo luận

Trong ví dụ trên

- Lớp cũ? Lớp mới?
  - Lớp cũ: NhanVien
  - Lớp mới: PhongBan
- Lớp mới tái sử dụng lớp cũ thông qua?
  - Mảng đối tượng của lớp NhanVien: dsNV
- Lớp mới tái sử dụng được những gì của lớp cũ?
  - tinhLuong() trong phương thức tongLuong()
  - inTTin() trong phương thức inTTin()



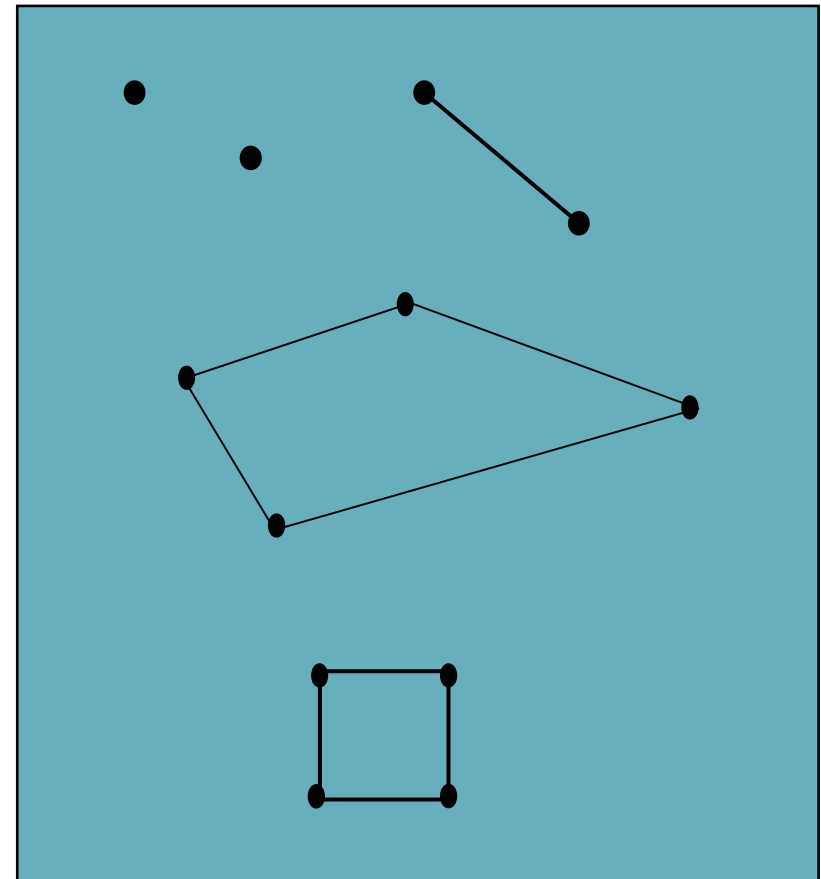
# Nội dung

1. Tái sử dụng mã nguồn
2. Kết tập (Aggregation)
3. Kế thừa (Inheritance)



## 3.1. Tổng quan về kế thừa

- Ví dụ:
  - Điểm
    - Tứ giác gồm 4 điểm
      - Kết tập
  - Tứ giác
    - Hình vuông
      - Kế thừa

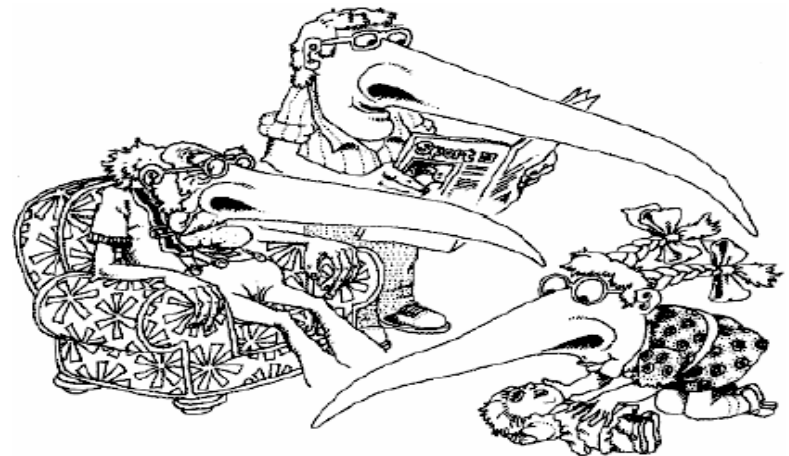


## 3.1.1. Bản chất kế thừa

- Kế thừa (Inherit, Derive)
  - Tạo lớp mới bằng cách phát triển lớp đã có.
  - Lớp mới kế thừa những gì đã có trong lớp cũ và phát triển những tính năng mới.
- Lớp cũ:
  - Lớp cha (parent, superclass), lớp cơ sở (base class)
- Lớp mới:
  - Lớp con (child, subclass), lớp dẫn xuất (derived class)

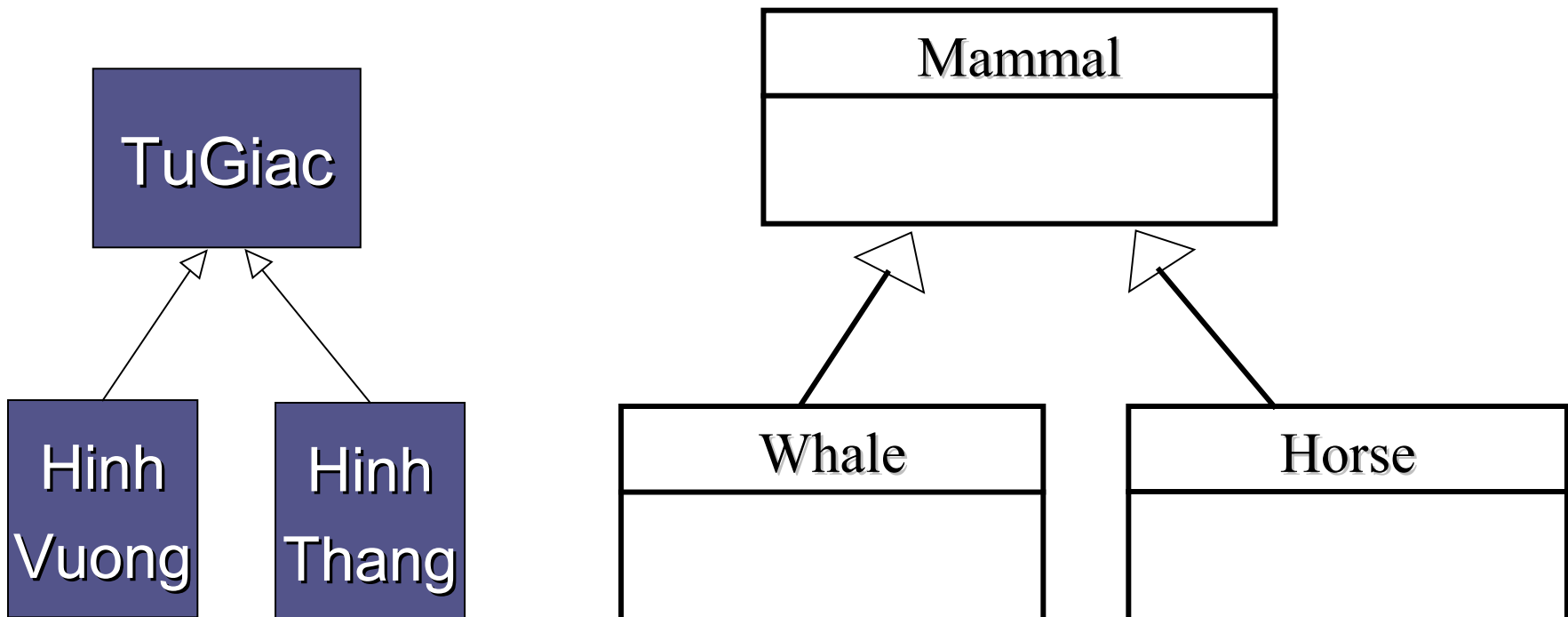
## 3.1.1. Bản chất kế thừa (2)

- Lớp con
  - Là một loại (is-a-kind-of) của lớp cha
  - Tái sử dụng bằng cách kế thừa các thành phần dữ liệu và các hành vi của lớp cha
  - Chi tiết hóa cho phù hợp với mục đích sử dụng mới
    - Extension: Thêm các thuộc tính/hành vi mới
    - Redefinition (Method Overriding): Chỉnh sửa lại các hành vi kế thừa từ lớp cha



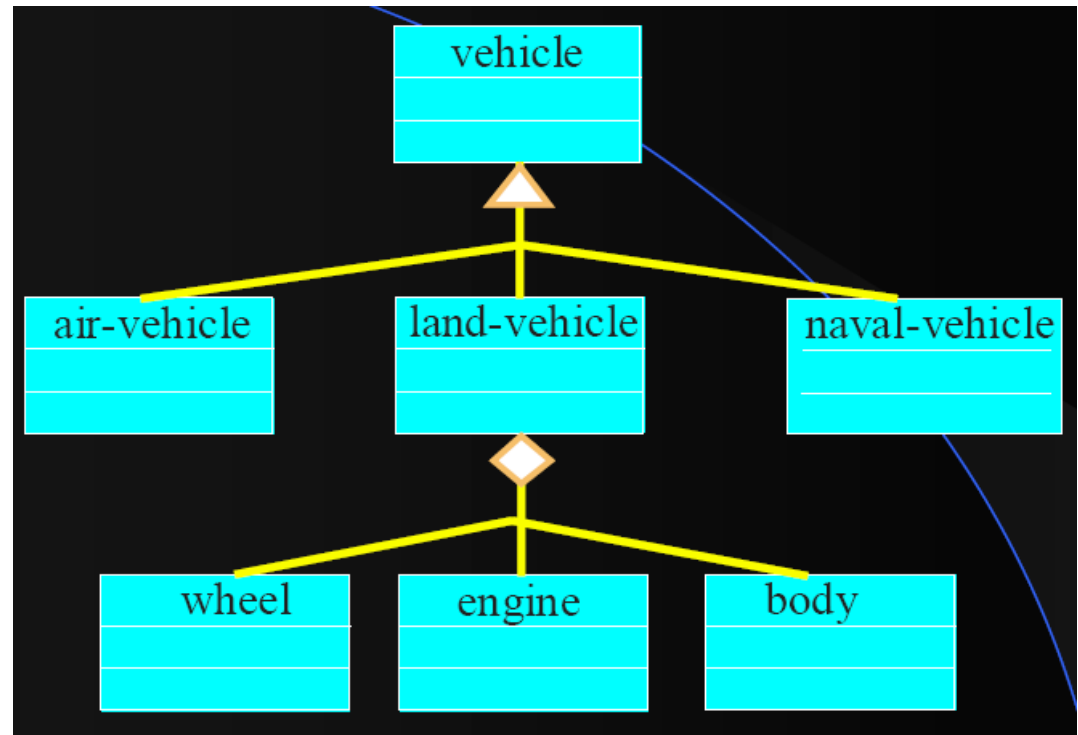
## 3.1.2. Biểu diễn kế thừa trong UML

- Sử dụng “tam giác rỗng” tại đầu Lớp cha



### 3.1.3. Kết tập và kế thừa

- So sánh kết tập và kế thừa?
  - Giống nhau
    - Đều là kỹ thuật trong OOP để tái sử dụng mã nguồn
  - Khác nhau?



# Phân biệt kế thừa và kết tập

## Kế thừa

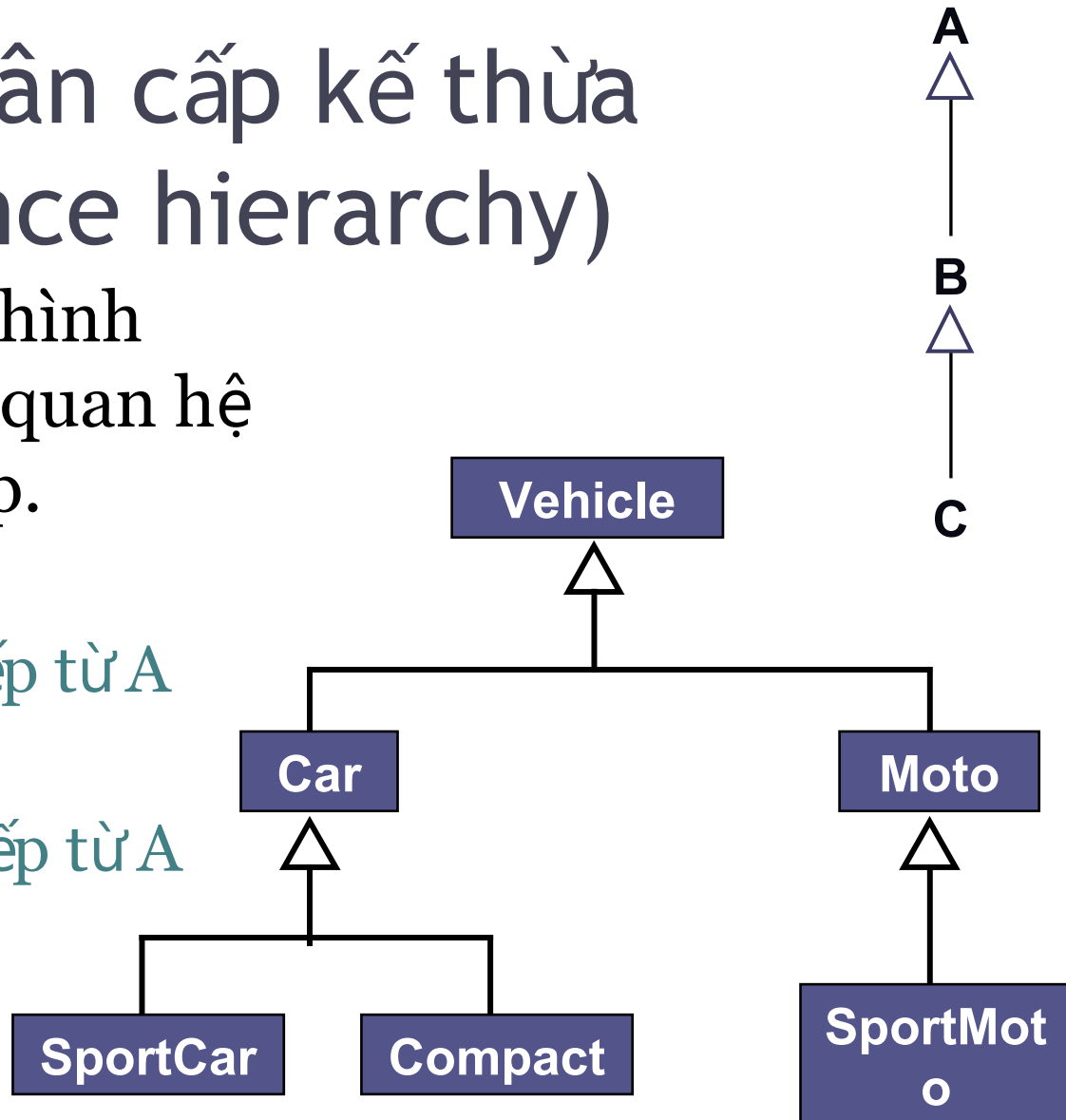
- Kế thừa **tái sử dụng** thông qua **lớp**.
  - Tạo lớp mới bằng cách phát triển lớp đã có
  - Lớp con kế thừa dữ liệu và hành vi của lớp cha
- Quan hệ “**là một loại**” (“is a kind of”)
- Ví dụ: Ô tô là một loại phương tiện vận tải

## Kết tập

- Kết tập **tái sử dụng** thông qua **đối tượng**.
  - Tạo ra lớp mới là tập hợp các đối tượng của các lớp đã có
  - Lớp toàn thể có thể sử dụng dữ liệu và hành vi thông qua các đối tượng thành phần
- Quan hệ “**là một phần**” (“is a part of”)
- Ví dụ: Bánh xe là một phần của Ô tô

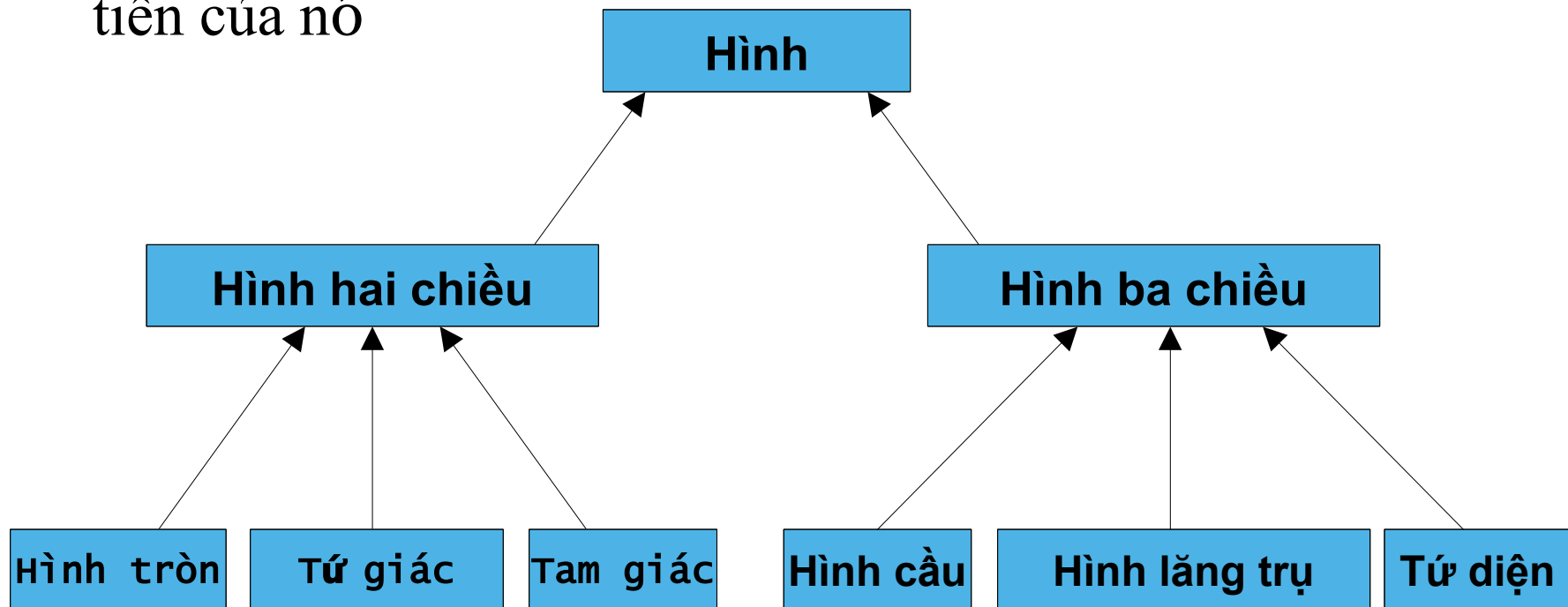
### 3.1.4. Cây phân cấp kế thừa (Inheritance hierarchy)

- Cấu trúc phân cấp hình cây, biểu diễn mối quan hệ kế thừa giữa các lớp.
- Dẫn xuất trực tiếp
  - B dẫn xuất trực tiếp từ A
- Dẫn xuất gián tiếp
  - C dẫn xuất gián tiếp từ A



## 3.1.4. Cây phân cấp kế thừa (2)

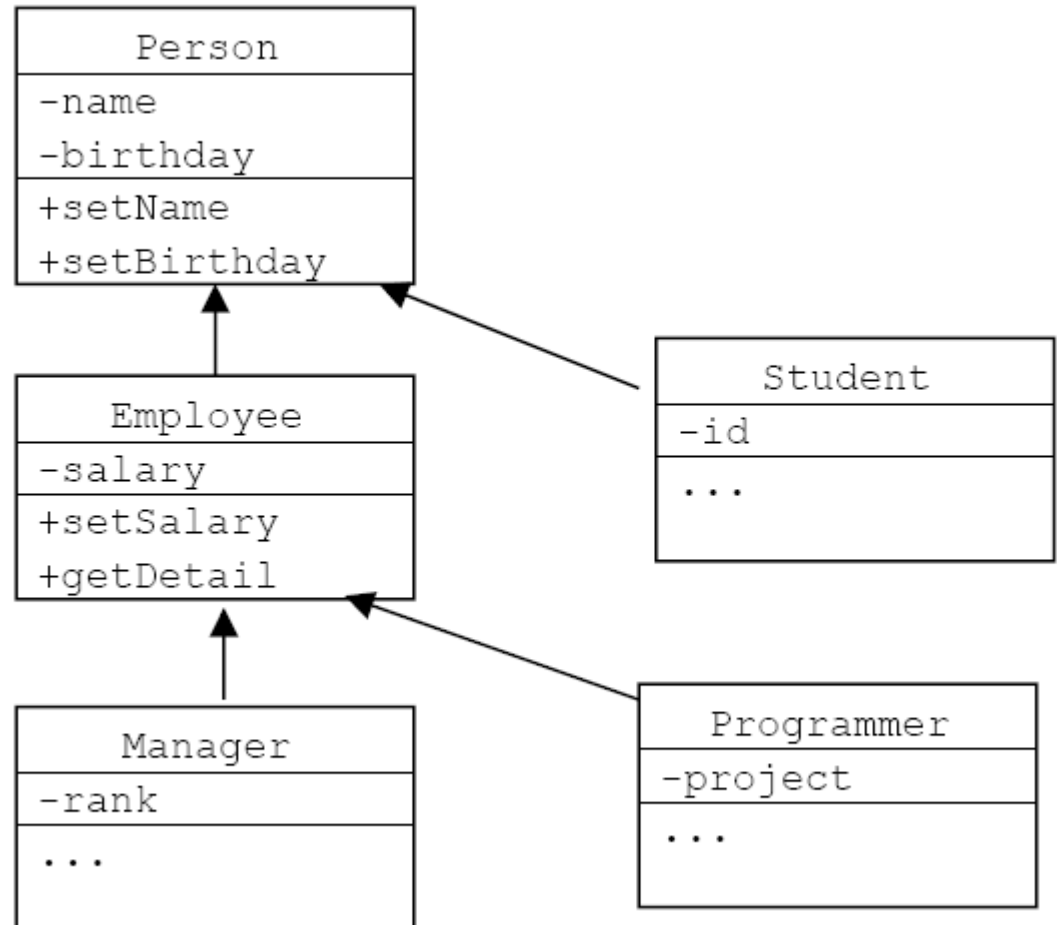
- Các lớp con có cùng lớp cha gọi là anh chị em (siblings)
- Thành viên được kế thừa sẽ được kế thừa xuống dưới trong cây phân cấp → Lớp con kế thừa tất cả các lớp tổ tiên của nó





## 3.1.4. Cây phân cấp kế thừa (2)

Mọi lớp  
đều kế thừa từ  
lớp gốc Object

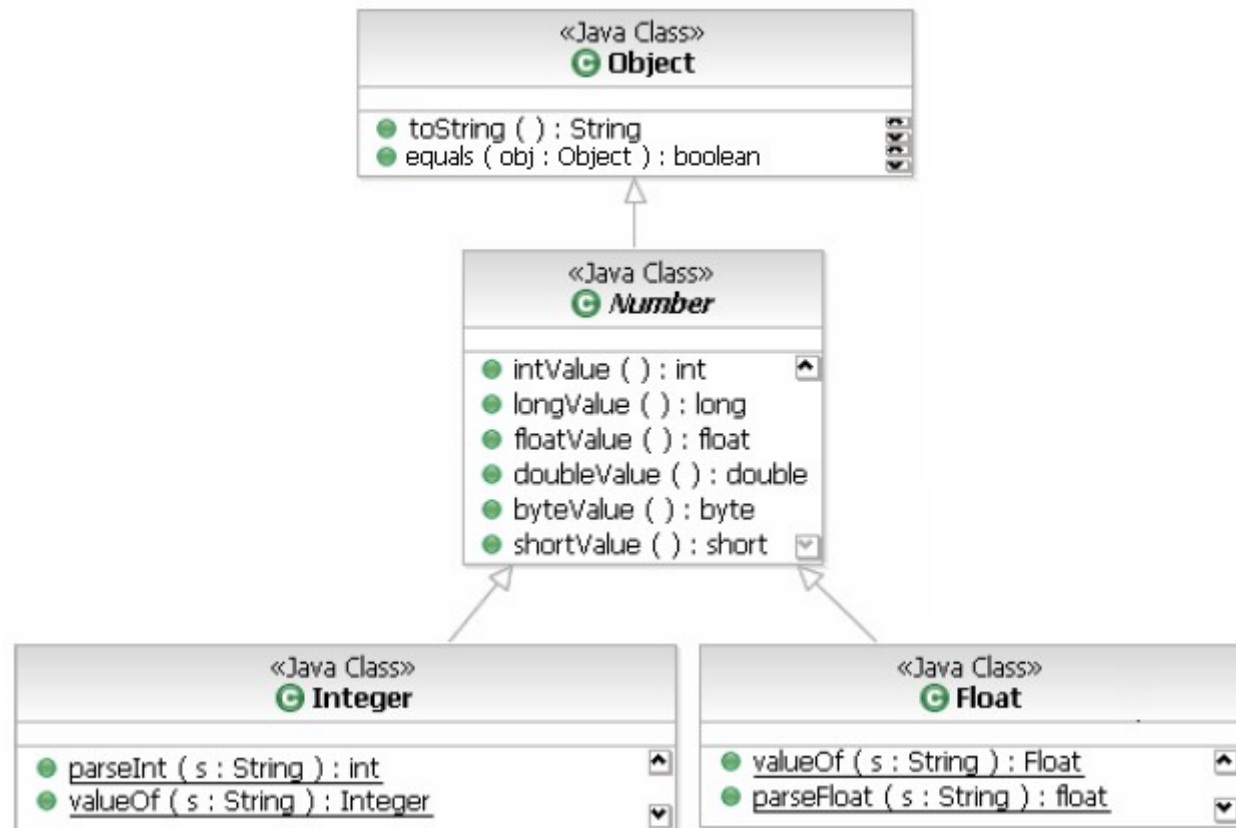


# Lớp Object

- Trong gói java.lang
- Nếu một lớp không được định nghĩa là lớp con của một lớp khác thì mặc định nó là lớp con trực tiếp của lớp Object.
  - Lớp Object là lớp gốc trên cùng của tất cả các cây phân cấp kế thừa

# Lớp Object (2)

- Chứa một số phương thức hữu ích kế thừa lại cho tất cả các lớp, ví dụ: `toString()`, `equals()`...



## 3.2. Nguyên lý kế thừa

- Chỉ định truy cập protected
- Thành viên protected trong lớp cha được truy cập trong:
  - Các thành viên lớp cha
  - Các thành viên lớp con
  - Các thành viên các lớp cùng thuộc 1 package với lớp cha
- Lớp con có thể kế thừa được gì?
  - Kế thừa được các thành viên được khai báo là public và protected của lớp cha.
  - Không kế thừa được các thành viên private.

## 3.2. Nguyên lý kế thừa (2)

	<b>public</b>	<b>Không có</b>	<b>protected</b>	<b>private</b>
Cùng lớp				
Lớp con cùng gói				
Lớp con khác gói				
Khác gói, non-inher				

## 3.2. Nguyên lý kế thừa (2)

	<b>public</b>	<b>Không có</b>	<b>protected</b>	<b>private</b>
Cùng lớp	Yes	Yes	Yes	Yes
Lớp con cùng gói	Yes	Yes	Yes	No
Lớp con khác gói	Yes	No	<b>NO</b>	No
Khác gói, non-inher	Yes	No	No	No

## 3.2. Nguyên lý kế thừa (3)

- Các trường hợp không được phép kế thừa:
  - Các phương thức khởi tạo và hủy
    - Làm nhiệm vụ khởi đầu và gỡ bỏ các đối tượng
    - Chúng chỉ biết cách làm việc với từng lớp cụ thể
  - Toán tử gán =
    - Làm nhiệm vụ giống như phương thức khởi tạo

## 3.3. Cú pháp kế thừa trên Java

- Cú pháp kế thừa trên Java:
  - `<Lớp con> extends <Lớp cha>`
- Lớp cha nếu được định nghĩa là **final** thì không thể có lớp dẫn xuất từ nó.
- Ví dụ:

```
class HìnhVuong extends TuGiac {  
    . . .  
}
```



# Ví dụ 1.1

```
public class TuGiac {
    protected Diem d1, d2, d3, d4;
    public void setD1(Diem _d1) {d1=_d1;}
    public Diem getD1(){return d1;}
    public void printTuGiac(){...}
    ...
}
```

Sử dụng các thuộc tính  
protected của lớp cha  
trong lớp con

```
public class HinhVuong extends TuGiac {
    public HinhVuong(){
        d1 = new Diem(0,0); d2 = new Diem(0,1);
        d3 = new Diem(1,0); d4 = new Diem(1,1);
    }
}

public class Test{
    public static void main(String args[]){
        HinhVuong hv = new HinhVuong();
        hv.printTuGiac();
    }
}
```

Gọi phương thức public  
lớp cha của đối tượng lớp con

## Ví dụ 1.2

```
public class TuGiac {
    protected Diem d1, d2, d3, d4;
    public void printTuGiac(){...}
    public TuGiac(){...}
    public TuGiac(Diem d1, Diem d2,
                  Diem d3, Diem d4) { ...}
}

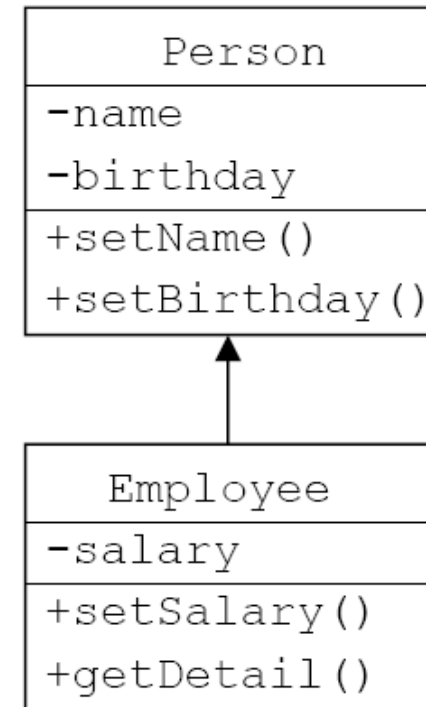
public class HìnhVuong extends TuGiac {
    public HìnhVuong(){ super(); }
    public HìnhVuong(Diem d1, Diem d2,
                     Diem d3, Diem d4){
        super(d1, d2, d3, d4);
    }
}

public class Test{
    public static void main(String args[]){
        HìnhVuong hv = new HìnhVuong();
        hv.printTuGiac();
    }
}
```

# Ví dụ 2

protected

```
class Person {
    private String name;
    private Date birthday;
    public String getName() {return name;}
    ...
}
class Employee extends Person {
    private double salary;
    public boolean setSalary(double sal) {
        salary = sal;
        return true;
    }
    public String getDetail(){
        String s = name+", "+birthday+", "+salary; //Loi
    }
}
```

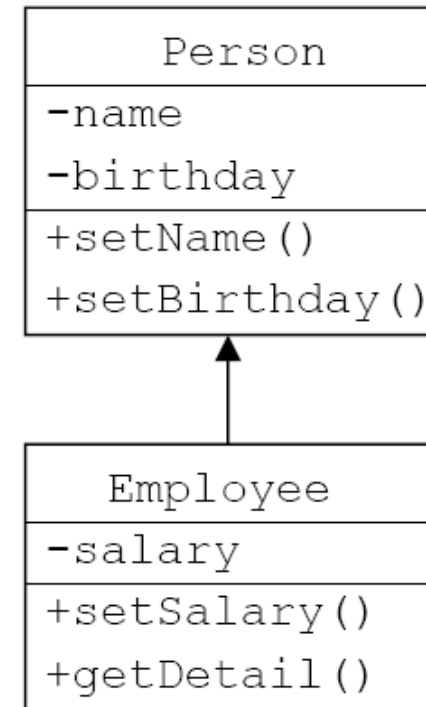


# Ví dụ 2

protected

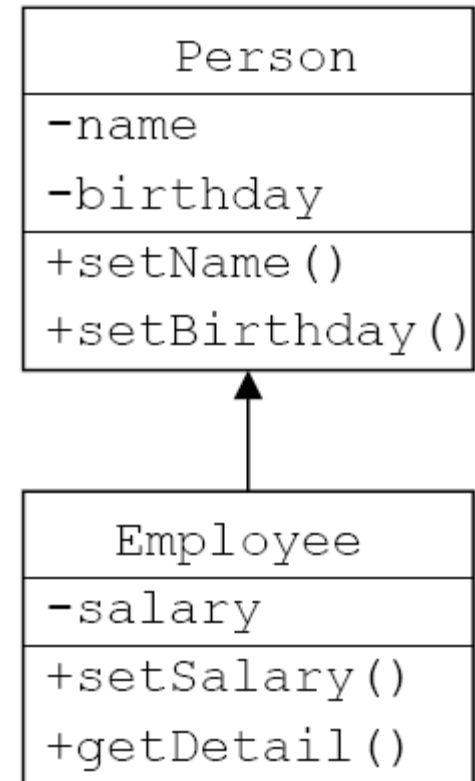
```
class Person {
    protected String name;
    protected Date birthday;
    public String getName() {return name;}
    ...
}

class Employee extends Person {
    private double salary;
    public boolean setSalary(double sal) {
        salary = sal;
        return true;
    }
    public String getDetail(){
        String s = name+", "+birthday+", "+salary;
    }
}
```



## Ví dụ 2 (tiếp)

```
public class Test {  
    public static void main(String args[]) {  
        Employee e = new Employee();  
        e.setName("John");  
        e.setSalary(3.0);  
    }  
}
```



## Ví dụ 3 - Cùng gói

```
public class Person {  
    Date birthday;  
    String name;  
    ...  
}  
public class Employee extends Person {  
    ...  
    public String getDetail() {  
        String s;  
        String s = name + "," + birthday;  
        s += ", " + salary;  
        return s;  
    }  
}
```

## Ví dụ 3 - Khác gói

```
package abc;
public class Person {
    protected Date birthday;
    protected String name;
    ...
}

import abc.Person;
public class Employee extends Person {
    ...
    public String getDetail() {
        String s;
        s = name + "," + birthday + "," + salary;
        return s;
    }
}
```

## 3.4. Khởi tạo và huỷ bỏ đối tượng

- Khởi tạo đối tượng:
  - Lớp cha được khởi tạo trước lớp con.
  - Các phương thức khởi tạo của lớp con luôn gọi phương thức khởi tạo của lớp cha ở câu lệnh đầu tiên
    - Tự động gọi (không tường minh - implicit): Khi lớp cha CÓ phương thức khởi tạo mặc định
    - Gọi trực tiếp (tường minh - explicit)
- Huỷ bỏ đối tượng:
  - Ngược lại so với khởi tạo đối tượng



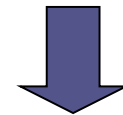
### 3.4.1. Tự động gọi constructor của lớp cha

```
public class TuGiac {
    protected Diem d1, d2;
    protected Diem d3, d4;
    public TuGiac() {
        System.out.println
            ("Lop cha TuGiac()");
    }
    //...
}

public class HinhVuong
    extends TuGiac {
    public HinhVuong() {
        //Tu dong goi TuGiac()

        System.out.println
            ("Lop con HinhVuong()");
    }
}
```

```
public class Test {
    public static void
        main(String arg[])
    {
        HinhVuong hv =
            new HinhVuong();
    }
}
```



```
C:\WINDOWS\system32\cmd...
Lop cha TuGiac()
Lop con HinhVuong()
Press any key to continue . . .
```

### 3.4.2. Gọi trực tiếp constructor của lớp cha

- Câu lệnh đầu tiên trong phương thức khởi tạo của lớp con có thể gọi phương thức khởi tạo của lớp cha
  - **`super(Danh_sach_tham_so) ;`**
  - Điều này là bắt buộc nếu lớp cha không có phương thức khởi tạo mặc định
    - Đã viết phương thức khởi tạo của lớp cha với một số tham số
    - Phương thức khởi tạo của lớp con không bắt buộc phải có tham số.

# Ví dụ

```
public class TuGiac {
    protected Diem d1, d2;
    protected Diem d3, d4;
    public TuGiac(Diem d1,
        Diem d2, Diem d3, Diem d4){
        System.out.println("Lop cha
            TuGiac(d1, d2, d3, d4)");
        this.d1 = d1; this.d2 = d2;
        this.d3 = d3; this.d4 = d4;
    }
}
```

```
public class HìnhVuong extends
    TuGiac {
    public HìnhVuong() {
        System.out.println
            ("Lop con HìnhVuong()");
    }
}
```

```
public class Test {
    public static
    void main(String
        arg[])
    {
        HìnhVuong hv =
            new
                HìnhVuong();
    }
}
```

**Lỗi** ↓

```
C:\ Command Prompt
D:\FIT-HUT\Lectures\OOP\Test>javac -cla
.\HìnhVuong.java:8: cannot find symbol
symbol  : constructor TuGiac()
location: class TuGiac
    public HìnhVuong() {
        ^
1 error
```

# Gọi trực tiếp constructor của lớp cha

## Phương thức khởi tạo lớp con **KHÔNG** tham số

```
public class TuGiac {
    protected Diem d1,d2,d3,d4;
    public TuGiac(Diem d1, Diem d2,
        Diem d3, Diem d4){
        System.out.println("Lop cha
            TuGiac(d1, d2, d3, d4)");
        this.d1 = d1; this.d2 = d2;
        this.d3 = d3; this.d4 = d4;
    }
}
```

```
public class HìnhVuong extends TuGiac {
    public HìnhVuong(){
        super(new Diem(0,0), new Diem(0,1),
            new Diem(1,1),new Diem(1,0));
        System.out.println("Lop con HìnhVuong()");
    }
}
```

. . .

```
HìnhVuong hv = new
    HìnhVuong();
```



```
C:\WINDOWS\system32\cmd.exe
Lop cha TuGiac(d1, d2, d3, d4)
Lop con HìnhVuong()
Press any key to continue . . .
```

# Gọi trực tiếp constructor của lớp cha

## Phương thức khởi tạo lớp con **CÓ** tham số

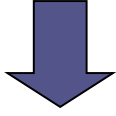
```

public class TuGiac {
    protected Diem d1,d2,d3,d4;
    public TuGiac(Diem d1,
Diem d2, Diem d3, Diem d4) {
        System.out.println
            ("Lop cha TuGiac(d1,d2,d3,d4)");
        this.d1 = d1; this.d2 = d2;
        this.d3 = d3; this.d4 = d4;
    }
}

public class HinhVuong extends TuGiac {
    public HinhVuong(Diem d1, Diem d2,
Diem d3, Diem d4) {
        super(d1, d2, d3, d4);
        System.out.println("Lop con
            HinhVuong(d1,d2,d3,d4)");
    }
}

```

. . .  
HinhVuong hv =  
 new HinhVuong(  
 new Diem(0,0),  
 new Diem(0,1),  
 new Diem(1,1),  
 new Diem(1,0));



Lop cha TuGiac(d1, d2, d3, d4)  
 Lop con HinhVuong(d1, d2, d3, d4)