

# Bài 8 Đa hình

---



# Nội dung

1. Upcasting và downcasting
2. Đa hình
3. Liên kết tĩnh và liên kết động

# 1

## Up-casting và down-casting

Chuyển đổi kiểu dữ liệu đối tượng



# Chuyển đổi kiểu dữ liệu nguyên thủy

- Java tự động chuyển đổi kiểu khi
  - Kiểu dữ liệu tương thích
  - Chuyển đổi từ kiểu hẹp hơn sang kiểu rộng hơn

```
int i;
```

```
double d = i;
```

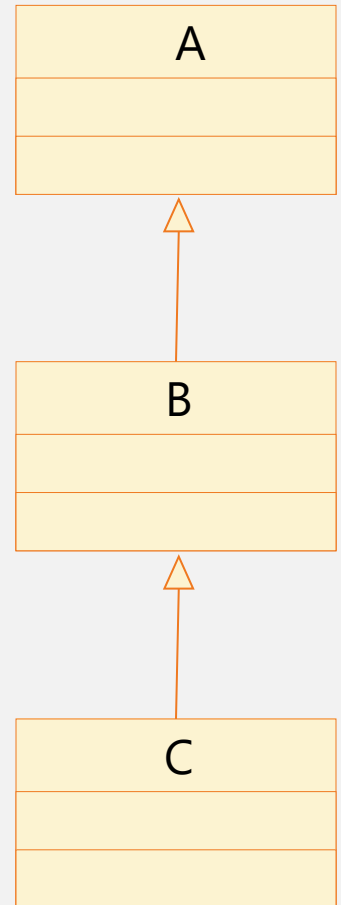
- Phải ép kiểu khi
  - Kiểu dữ liệu tương thích
  - Chuyển đổi từ kiểu rộng hơn sang kiểu hẹp hơn

```
int i;
```

```
byte b = i; byte b = (byte)i;
```

# Chuyển đổi kiểu dữ liệu tham chiếu

- Kiểu dữ liệu tham chiếu có thể được chuyển đổi kiểu khi
  - Kiểu dữ liệu tham chiếu (lớp) *tương thích*
    - *Nằm trên cùng một cây phân cấp kế thừa*
- Hai cách chuyển đổi
  - Up-casting
  - Down-casting

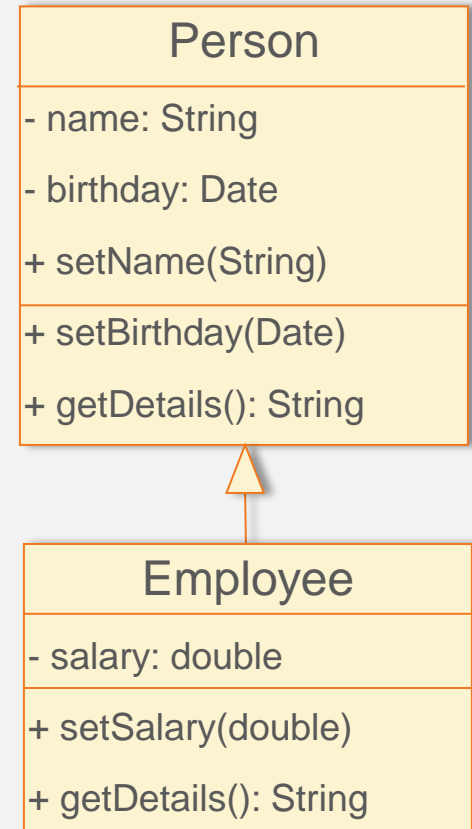


# Up-casting

- Up casting: đi lên trên cây phân cấp thừa kế (moving up the inheritance hierarchy)
- Up casting là khả năng nhìn nhận đối tượng thuộc lớp dẫn xuất như là một đối tượng thuộc lớp cơ sở.
- Tự động chuyển đổi kiểu

# Ví dụ

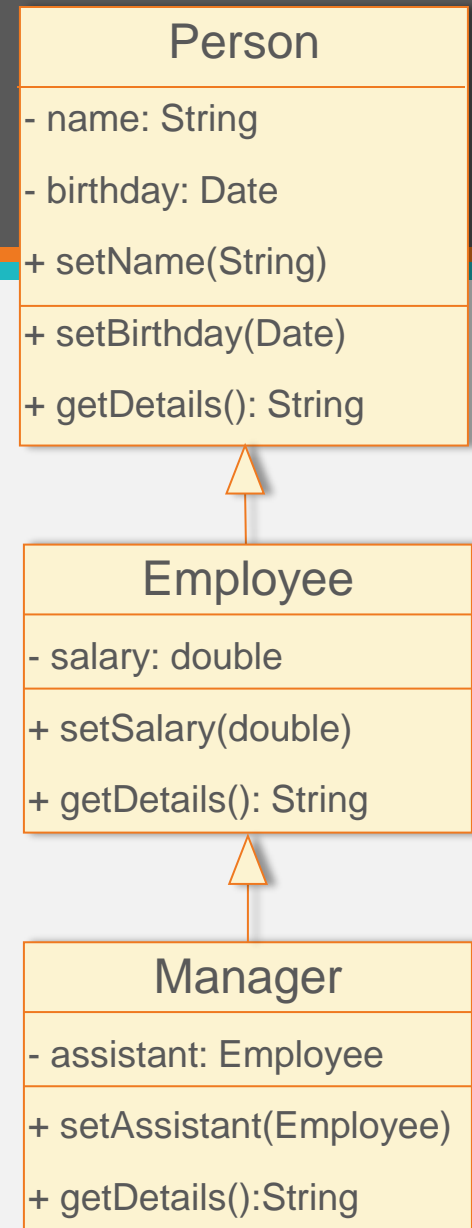
```
public class Test1 {  
    public static void main(String arg[]) {  
        Employee e = new Employee();  
        Person p;  
        p = e;  
        p.setName("Hoa");  
        p.setSalary(350000);  
        // compile error  
    }  
}
```



# Ví dụ

```
class Manager extends Employee {
    Employee assistant;
    // ...
    public void setAssistant(Employee e) {
        assistant = e;
    }
    // ...
}

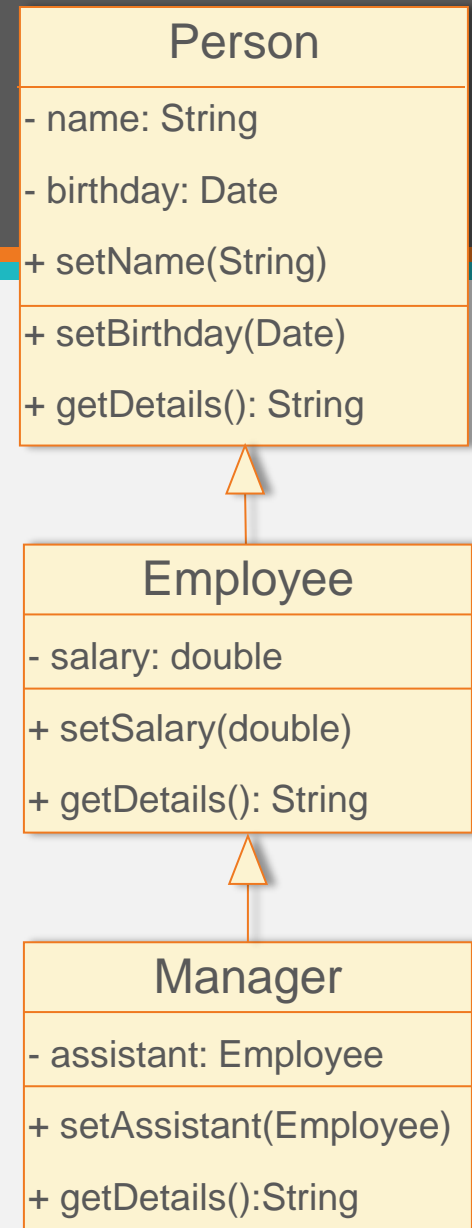
public class Test2 {
    public static void main(String arg[]) {
        Manager junior, senior;
        // ...
        senior.setAssistant(junior);
    }
}
```





# Ví dụ

```
public class Test3 {  
    String static teamInfo(Person p1, Person p2) {  
        return "Leader: " + p1.getName() +  
            ", member: " + p2.getName();  
    }  
  
    public static void main(String arg[]) {  
        Employee e1, e2;  
        Manager m1, m2;  
        // ...  
        System.out.println(teamInfo(e1, e2));  
        System.out.println(teamInfo(m1, m2));  
        System.out.println(teamInfo(m1, e2));  
    }  
}
```

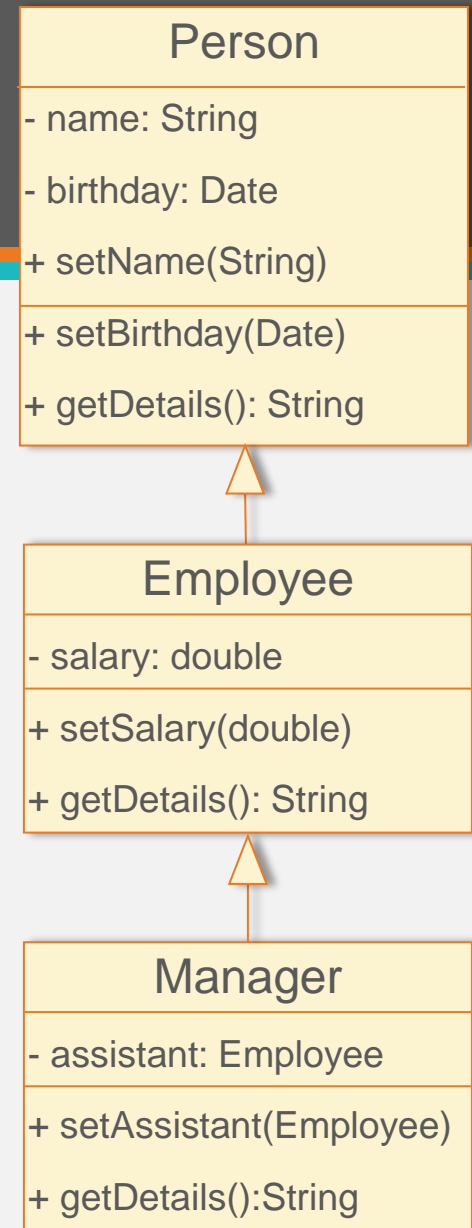


# Down-casting

- Down casting: đi xuống cây phân cấp thừa kế (move back down the inheritance hierarchy)
- Down casting là khả năng nhìn nhận một đối tượng thuộc lớp cơ sở như một đối tượng thuộc lớp dẫn xuất.
- Không tự động chuyển đổi kiểu  
→ Phải ép kiểu.

# Ví dụ

```
public class Test2 {  
    public static void main(String arg[]) {  
        Employee e = new Employee();  
        Person p = e; // up casting  
        Employee ee = (Employee) p;  
        // down casting  
        Manager m = (Manager) ee;  
        // run-time error  
  
        Person p2 = new Manager();  
        Employee e2 = (Employee) p2;  
    }  
}
```



# 2

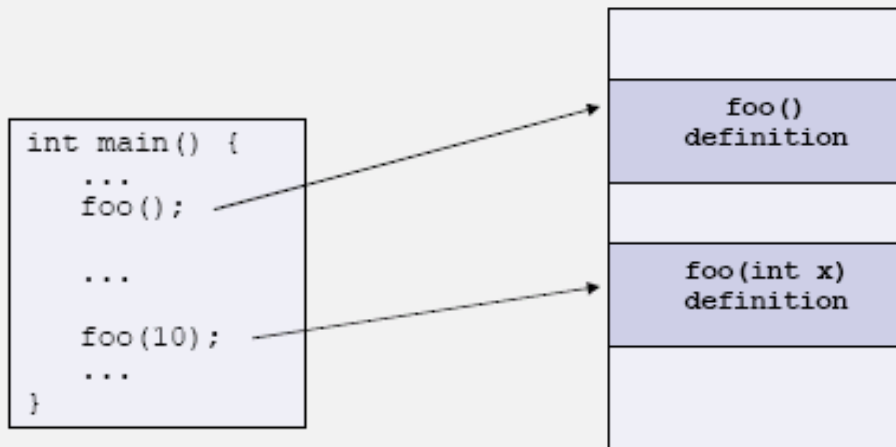
## Liên kết tĩnh và liên kết động

Static binding & dynamic binding



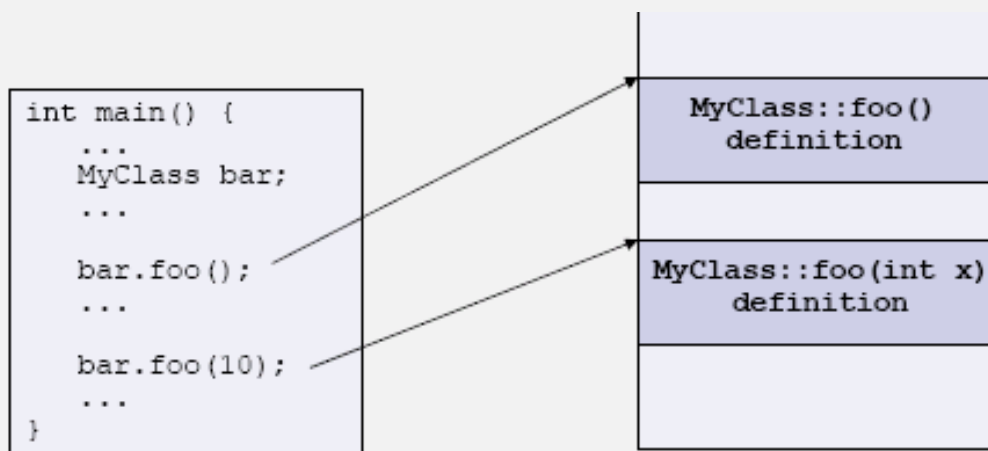
# Liên kết lời gọi hàm

- Liên kết lời gọi hàm (function call binding) là quy trình xác định khối mã hàm cần chạy khi một lời gọi hàm được thực hiện
- C: đơn giản vì mỗi hàm có duy nhất một tên
- C++: chồng hàm, phân tích chữ ký kiểm tra danh sách tham số.



# Trong ngôn ngữ HĐT

- Liên kết lời gọi phương thức
- Đối với các lớp độc lập (không thuộc cây thừa kế nào), quy trình này gần như không khác với function call binding
  - so sánh tên phương thức, danh sách tham số để tìm định nghĩa tương ứng
  - một trong số các tham số là tham số ẩn: con trỏ this



# Liên kết tĩnh

- Liên kết tại thời điểm biên dịch
  - Early Binding/Compile-time Binding
  - Lời gọi phương thức được quyết định khi biên dịch, do đó chỉ có một phiên bản của phương thức được thực hiện
  - Nếu có lỗi thì sẽ có lỗi biên dịch
  - Ưu điểm về tốc độ
- C/C++ function call binding, và C++ method binding cơ bản đều là ví dụ của liên kết tĩnh (static function call binding)

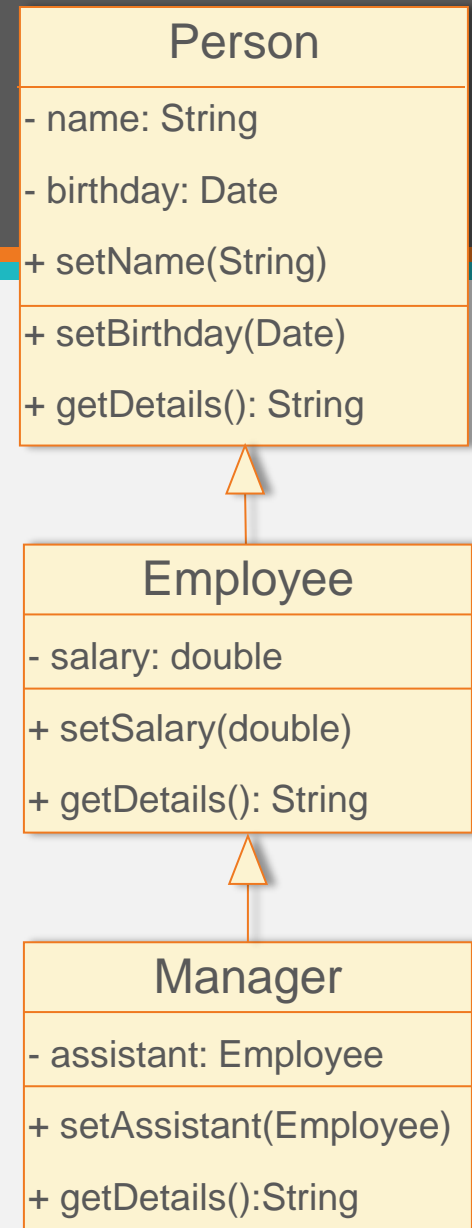
# Liên kết động

- Lời gọi phương thức được quyết định khi thực hiện (run-time)
  - Late binding/Run-time binding
  - Phiên bản của phương thức phù hợp với đối tượng được gọi.
  - Java mặc định sử dụng liên kết động



# Ví dụ

```
public class Test {  
    public static void main(String arg[]){  
        Person p = new Person();  
        // ...  
        Employee e = new Employee();  
        // ...  
        Manager m = new Manager();  
        // ...  
        Person pArr[] = {p, e, m};  
        for (int i=0; i< pArr.length; i++){  
            System.out.println(  
                pArr[i].getDetail());  
        }  
    }  
}
```



# 3

## Đa hình

Polymorphism



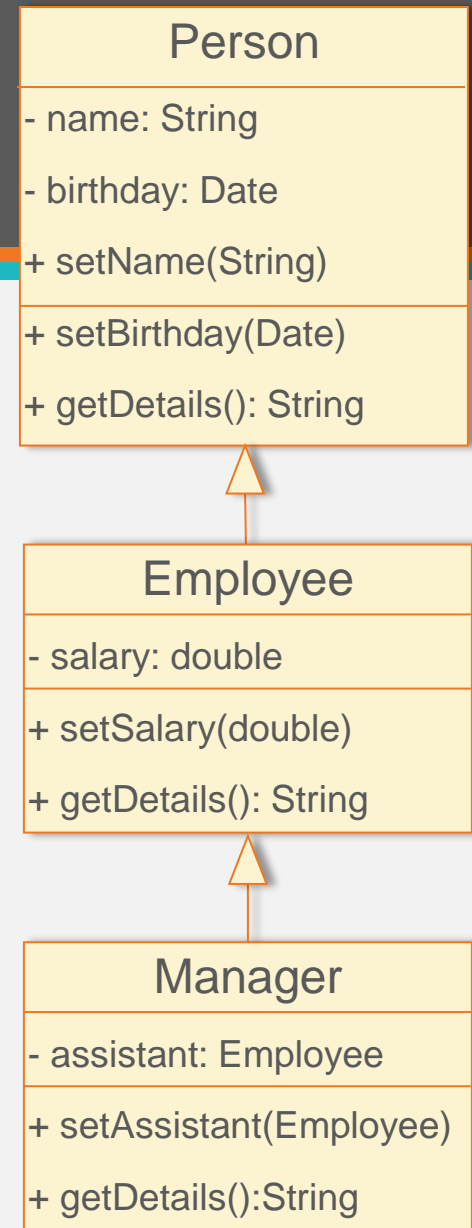
# Đa hình

- Polymorphism: Nhiều hình thức thực hiện, nhiều kiểu tồn tại
- Đa hình trong lập trình
  - Đa hình phương thức:
    - + Phương thức trùng tên, phân biệt bởi danh sách tham số.
  - Đa hình đối tượng
    - + Nhìn nhận đối tượng theo nhiều kiểu khác nhau
    - + Các đối tượng khác nhau cùng đáp ứng chung danh sách các thông điệp có giải nghĩa thông điệp theo cách thức khác nhau.

# Ví dụ

- Các đối tượng khác nhau giải nghĩa các thông điệp theo các cách thức khác nhau
  - Liên kết động (Java)

```
Person p1 = new Person();
Person p2 = new Employee();
Person p3 = new Manager();
// ...
System.out.println(p1.getDetail());
System.out.println(p2.getDetail());
System.out.println(p3.getDetail());
```

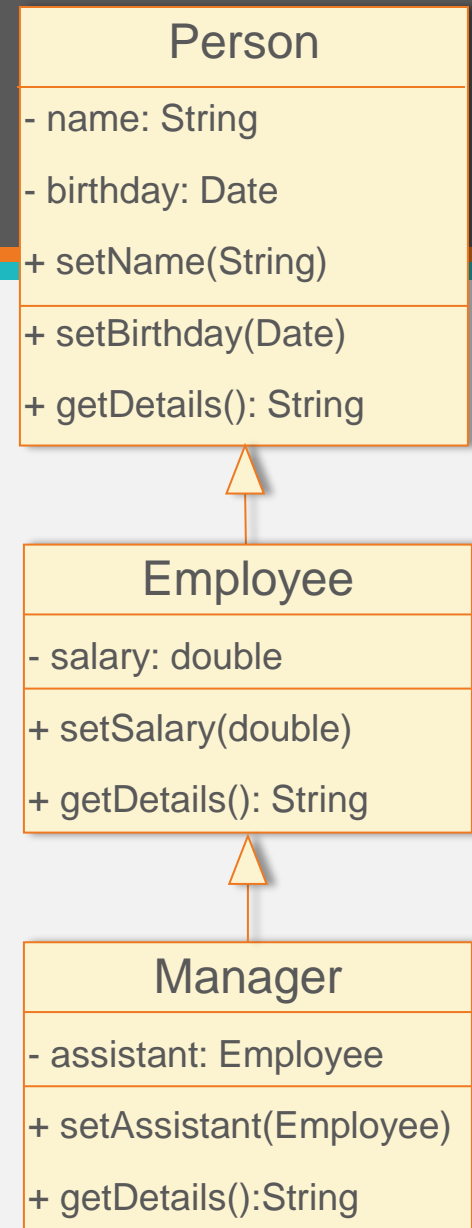


# Ví dụ

```
class EmployeeList {
    Employee list[];
    ...
    public void add(Employee e) {...}
    public void print() {
        for (int i=0; i<list.length; i++) {

            System.out.println(list[i].getDetail());
        }
    }
}

...
EmployeeList list = new EmployeeList();
Employee e1; Manager m1;
...
list.add(e1); list.add(m1);
list.print();
```



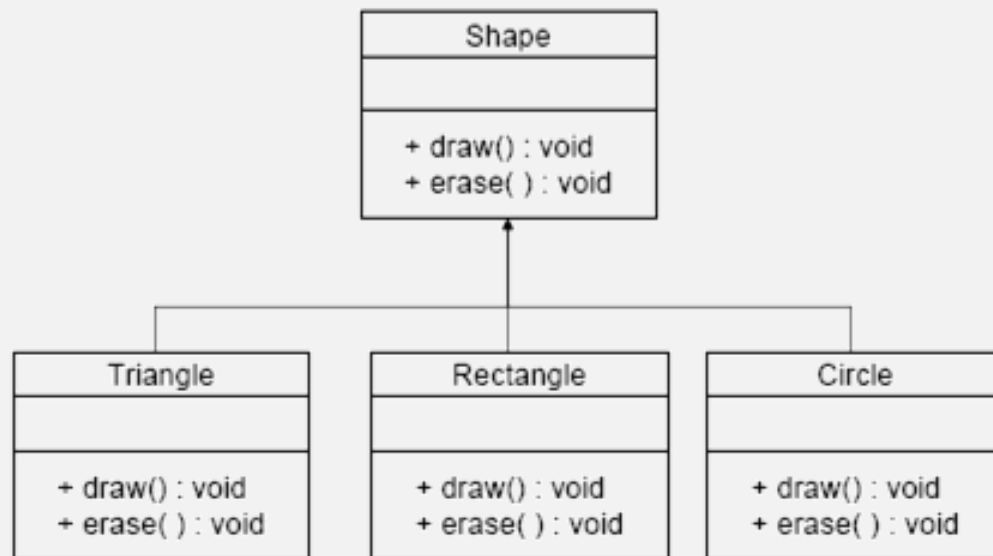
# Toán tử *instanceof*

- Kiểm tra xem một đối tượng có phải là thể hiện của một lớp nào đó không

```
public class Employee extends Person {}  
public class Student extends Person {}  
  
public class Test{  
    public doSomething(Person e) {  
        if (e instanceof Employee) {...  
        } else if (e instanceof Student) {...  
        } else {...  
        }  
    }  
}
```

# Ví dụ

- Các đối tượng Triangle, Rectangle, Circle đều là các đối tượng Shape



# Ví dụ

...

```
public static void handleShapes(Shape[] shapes){  
    // Vẽ các hình theo cách riêng của mỗi hình  
    for( int i = 0; i < shapes.length; ++i) {  
        shapes[i].draw();  
    }  
    ...  
    // Gọi đến phương thức xóa, không cần quan tâm  
    // đó là hình gì  
    for( int i = 0; i < shapes.length; ++i) {  
        shapes[i].erase();  
    }  
}  
...
```



# Tổng kết

- Upcasting và downcasting
  - Nhìn nhận các đối tượng thuộc lớp cơ sở như đối tượng thuộc lớp dẫn xuất (upcasting) và ngược lại (down-casting)
- Liên kết tĩnh và liên kết động
  - Liên kết lời gọi hàm lúc biên dịch (liên kết tĩnh) hay lúc chạy chương trình (liên kết động)
- Đa hình
  - Nhìn nhận một đối tượng dưới nhiều kiểu khác nhau

# Thank you!

Any questions?

