



ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**IT3100**

# **LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

## **Bài 07. Đa hình**

# Nội dung

1. Khái niệm Đa hình (Polymorphism)
2. Liên kết tĩnh và Liên kết động
3. Upcasting và Downcasting

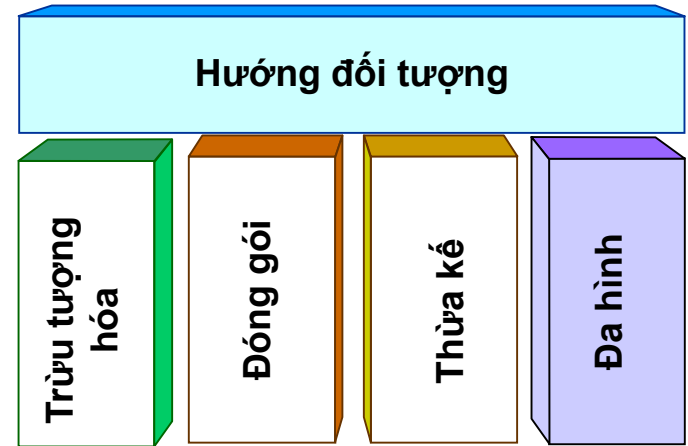
# 1/ ĐA HÌNH

# 1.1 Khái niệm đa hình

- ❖ Ví dụ: Nếu đi du lịch, bạn có thể chọn ô tô, thuyền, hoặc máy bay
  - Dù đi bằng phương tiện gì, kết quả cũng giống nhau là bạn đến được nơi cần đến
  - Cách thức đáp ứng các dịch vụ có thể khác nhau



# 1.1 Khái niệm đa hình



## ❖ Polymorphism: nhiều hình thể

- thực hiện một hành động bằng nhiều cách khác nhau

## ❖ 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: một đối tượng nào đó có khả năng nhập vai thành các đối tượng khác
  - => Một đối tượng có thể thực hiện một tác vụ theo nhiều cách khác nhau

# 1.1 Khái niệm đa hình

❖ Có hai kiểu của đa hình trong java:

- Nạp chồng phương thức
  - Khi được gọi, dựa vào tham số truyền vào, phương thức tương ứng sẽ được thực hiện
- Ghi đè phương thức
  - hệ thống xác định được đối tượng nào và phương thức nào thực sự đang hoạt động khi ứng dụng đang chạy

## 2/ LIÊN KẾT TĨNH – ĐỘNG

## 2.1. Liên kết tĩnh (Static Binding)

- ❖ **Binding:** Kết nối một lời gọi phương thức/hàm tới một thân phương thức/hàm gọi là binding
- ❖ **Static Binding:** Liên kết tại thời điểm biên dịch
  - Static Binding/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



## 2.2. Liên kết động (Dynamic binding)

❖ **Dynamic Binding:** Lời gọi phương thức được quyết định tại thời điểm chạy (run-time)

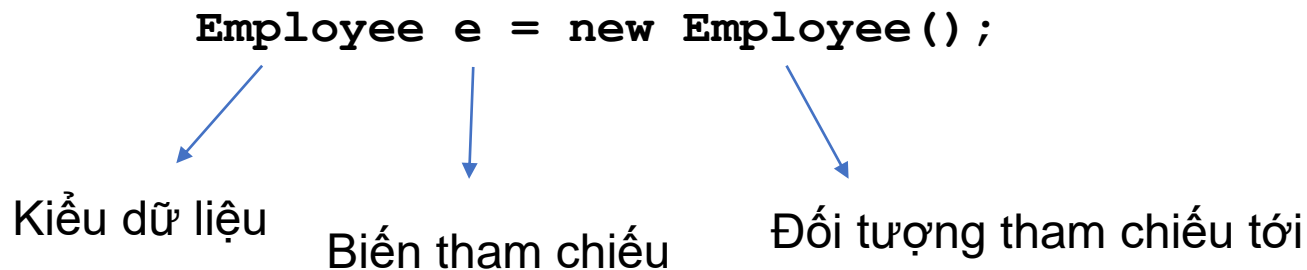
- Dynamic Binding/Late binding/Run-time binding
- Phiên bản của phương thức sẽ phù hợp với đối tượng được gọi.
- Java mặc định sử dụng liên kết động. trừ khi một phương thức được khai báo với từ khóa final

# 3/ UPCASTING VÀ DOWNCASTING

## 3.1. Biến tham chiếu

- ❖ Cách duy nhất để truy cập đến một đối tượng là thông qua biến tham chiếu
- ❖ Khi đã được khai báo, kiểu của một biến tham chiếu không thể thay đổi
- ❖ Kiểu của biến tham chiếu sẽ quyết định phương thức có thể gọi được từ đối tượng

`Employee e = new Employee();`



Kiểu dữ liệu      Biến tham chiếu      Đối tượng tham chiếu tới

## 3.2. Đa hình tại runtime

- ❖ Trong Java, một lời gọi tới một phương thức **được ghi đè** sẽ được xử lý tại runtime thay vì tại compile time.
- ❖ Việc quyết định phương thức được gọi dựa trên đối tượng được tham chiếu bởi biến tham chiếu chứ không phải theo kiểu dữ liệu như với các phương thức thông thường.

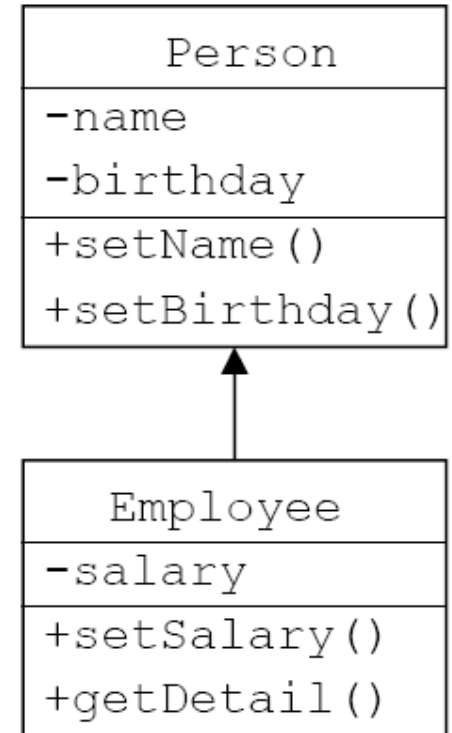
## 3.3. Upcasting

- ❖ Là khả năng nhìn nhận đối tượng thuộc lớp dẫn xuất như là một đối tượng thuộc lớp cơ sở.
- ❖ Chuyển 1 đối tượng là một thể hiện của lớp con lên thành đối tượng là thể hiện của lớp cha
- ❖ Khi biến tham chiếu của lớp cha tham chiếu tới đối tượng của lớp con
- ❖ Tự động chuyển đổi kiểu

```
class A{}  
class B extends A{}  
A a = new B(); //đây là upcasting
```

# Ví dụ

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



# Ví dụ

```
class Animal {  
    public void eat() { System.out.println("eating..."); }  
}  
  
public class Cat extends Animal {  
    public void meow(){ System.out.println("meowing..."); }  
}  
  
public class Upcasting {  
    public static void main(String[] args) {  
        Cat cat = new Cat();  
        Animal animal1 = cat; // Tự động chuyển kiểu  
        Animal animal2 = (Animal) cat; // Chuyển kiểu TM  
        cat.eat();  
        cat.meow();  
        animal1.eat();  
        animal2.eat();  
        animal2.meow(); // error compiler  
    }  
}
```

## 3.3.1. Đa hình tại runtime với upcasting

- ❖ Mọi phương thức của lớp *Animal* hoàn toàn có thể gọi qua 1 đối tượng thuộc lớp *Cat* do giữa *Animal* và *Cat* có quan hệ thừa kế
- ❖ Tuy nhiên, nếu bất kỳ phương thức nào của lớp *Animal* được ghi đè tại lớp *Cat* thì trong quá trình runtime, hàm được gọi sẽ là hàm của lớp *Cat*.



# Ví dụ

```
public class Cat extends Animal {  
    @Override phương thức eat của lớp Animal  
    public void eat() {  
        System.out.println("Eat meat");  
    }  
  
    public void meow() {  
        System.out.println("meowing...");  
    }  
}
```

Lời gọi từ hàm main()  
cat.eat();  
cat.meow();  
animal1.eat();  
animal2.eat();

## 3.3.1. Đa hình tại runtime với upcasting (2)

```
class Bike{
    void run(){
        System.out.println("dang chay");
    }
}
class Splender extends Bike{
    void run(){
        System.out.println("chay an toan voi 60km");
    }

    public static void main(String args[]){
        Bike b = new Splender();
        b.run();
    }
}
```

1. biến tham chiếu của lớp cha gọi phương thức run

2. biến tham chiếu đang tham chiếu tới đối tượng của lớp con

3. phương thức lớp con ghi đè phương thức của lớp cha

4. phương thức của lớp con được gọi tại runtime

chay an toan voi 60km

## 3.3.1. Đa hình tại runtime với upcasting (3)

- ❖ Đa hình tại runtime không hoạt động với thuộc tính (kể cả bị ghi đè)

```
class Bike{
    int speedlimit=90;
}
class Honda3 extends Bike{
    int speedlimit=150;

    public static void main(String args[]){
        Bike obj=new Honda3();
        System.out.println(obj.speedlimit);
    }
}
```

## 3.4. Downcasting

- ❖ Là khả năng nhìn nhận một đối tượng thuộc lớp cơ sở như một đối tượng thuộc lớp dẫn xuất.
- ❖ Chuyển 1 đối tượng là một thể hiện của lớp cha xuống thành đối tượng là thể hiện của lớp con
- ❖ Không tự động chuyển đổi kiểu

```
Cat cat = new Animal(); // Compilation error
```

```
Animal a = new Animal();
```

```
Cat cat2 = (Cat) a ; // Compilation error
```

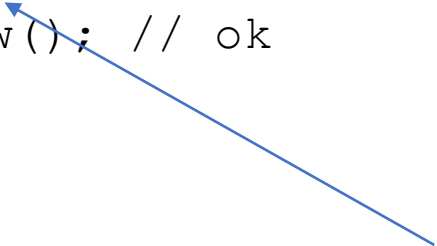
## 3.4. Downcasting (2)

- ❖ Trong upcasting, ta có thể gọi những phương thức đã được ghi đè của lớp cha tại lớp con qua đối tượng lớp cha.
- ❖ Nếu muốn gọi **mọi phương thức** của lớp con thông qua việc ép kiểu đối tượng thuộc lớp cha: sử dụng downcasting.
  - (Chuyển 1 đối tượng là một thể hiện của lớp cha xuống thành đối tượng là thể hiện của lớp con)

## 3.4. Downcasting (3)

```
public class Downcasting {  
    public static void main(String[] args) {  
        Animal animal = new Cat(); //upcasting  
        Cat cat = (Cat) animal; // downcasting  
        cat.meow(); // ok  
    }  
}
```

không cần new Cat()



## 3.4. Downcasting (4)

- ❖ T/h downcasting có thể sẽ gặp lỗi **ClassCastException**  
=> cần kiểm tra một đối tượng có phải là thể hiện của một kiểu dữ liệu cụ thể không !

```
public class Cat extends Animal {...}
```

```
Animal animal = new Cat();
```

```
Cat cat = (Cat) animal; // downcasting is ok
```

```
Object obj = new Rectangle();
```

```
Cat cat = (Cat) obj; // Error runtime: ClassCastException
```



```
if (obj instanceof Cat) {  
    Cat cat = (Cat) obj; // downcasting  
    System.out.println("ok downcasting performed");  
} else {  
    System.out.println("obj is not instance of Cat");  
}
```

