

# Thảo luận một số vấn đề về kế thừa

Nguyễn Khắc Huy

BMCNPM – ĐHKHTN TPHCM  
09/2018



# Nội dung

- Đa kế thừa**
- Hàm ảo
- Hàm thuần ảo
- Hàm hủy ảo
- Bài tập



# Thảo luận nhóm (4-5 người)

## □ Đơn kế thừa (Single Inheritance)

- Thuộc tính và phương thức tĩnh có được kế thừa ở lớp kế thừa (lớp dẫn xuất) không? Tại sao?

## □ Đa kế thừa (multiple inheritance)

- Những trường hợp nào thì đa kế thừa có vấn đề? Cho ví dụ minh họa.
- Những vấn đề đó là gì?
- Cách giải quyết?



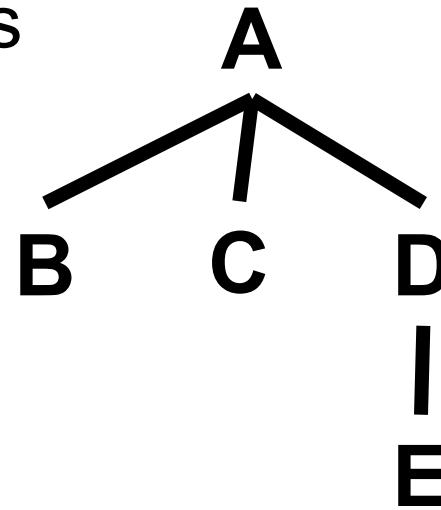
# Đa kế thừa

- Nếu đơn kế thừa có thể giải quyết tốt cho các bài toán khác nhau, tại sao không cho phép đa kế thừa?



# Tree, DAGs, và Diamonds

- Sơ đồ kế thừa có dạng như một đồ thị
  - Mỗi nút tương ứng với một lớp đối tượng
  - Cạnh biểu thị cho quan hệ giữa lớp cơ sở và lớp kế thừa
  - Đơn kế thừa: tạo ra cấu trúc cây
  - Đa kế thừa: tạo ra DAGs



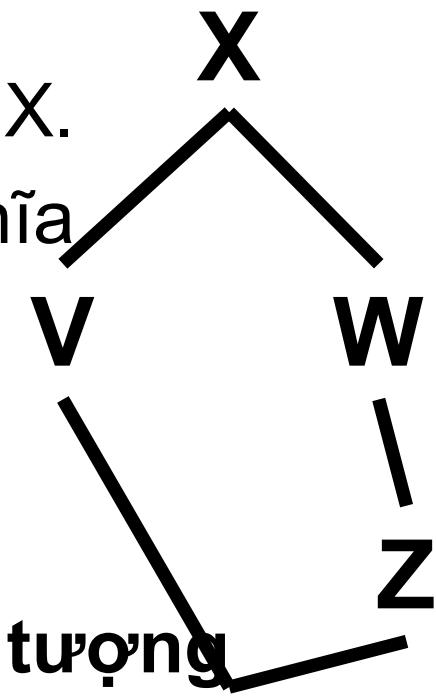
# Vấn đề hình thoi

## □ Diamonds

- Với đa kế thừa, dễ thấy rằng Y là lớp được kế thừa chuyển tiếp của lớp cơ sở X.
- Điều gì xảy ra khi các lớp V, Z định nghĩa cùng một hành vi m hay thuộc tính f?

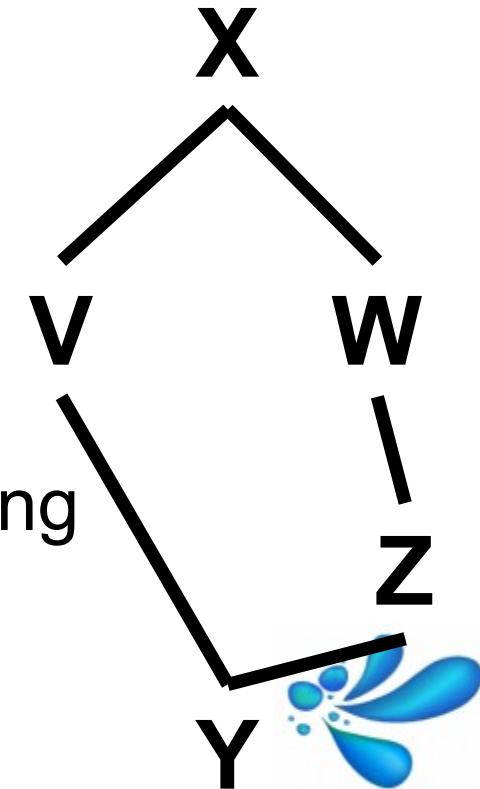
VD: Artists, Cowboys, ArtistCowboys

- Tất cả đều có phương thức **draw**
- **Phương thức draw nào sẽ được đối tượng gọi xử lý? Y sẽ kế thừa phương thức m từ lớp đối tượng nào?**



# Vấn đề hình thoi – Phương thức

- Sẽ như thế nào nếu phương thức m được Z cài đặt nạp chồng lại nhưng V thì không?
- Một vài cách giải quyết
  - Sử dụng chỉ dẫn Z::m
  - Không sử dụng các p.thức và thuộc tính gây nhập nhằng ở Z
  - Lớp kế thừa có thể cài đặt nạp chồng Phương thức m.



# Vấn đề hình thoi – Thuộc tính

- Tương tự cho thuộc tính trong đa kế thừa: có một hay nhiều thuộc tính f nếu vấn đề hình thoi xảy ra?

- Kế thừa thành nhiều thuộc tính f:

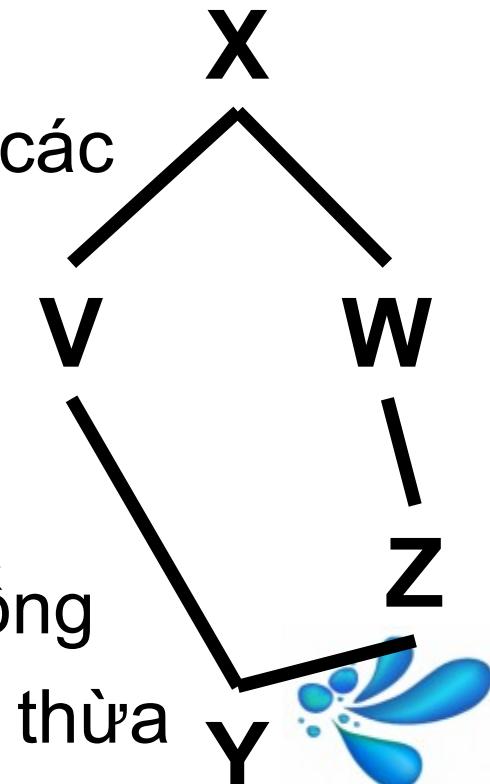
Artist::draw và Cowboy::draw sử dụng các thuộc tính f này một cách khác nhau.

- C++ cung cấp cả hai dạng kế thừa đa và đơn thuộc tính

- Hai thuộc tính luôn được thừa kế xuống

lớp con, để chỉ một thuộc tính được kế thừa

8 xuống lớp con. Hãy khai báo ở lớp cơ sở chung.



# Java-style Interfaces

- Trong Java có thể định nghĩa ra các interfaces và các lớp đối tượng sẽ hiện thực các interfaces này.
  - Interface mô tả các phương thức và các kiểu dữ liệu
  - Interface là kiểu dữ liệu – chương trình có thể sử dụng khai báo như biến hoặc tham số.
  - Giá sử lớp đối tượng A được định nghĩa bởi interface IObj, các thể hiện của A có thể sử dụng kiểu biến IObj để khai báo biến. Nhưng cần cài đặt mọi thứ đã khai báo trong IObj.



# No interface in C++

- C++ cho phép các phương thức và lớp đối tượng được khai báo ở mức trừu tượng (sử dụng từ khóa abstract)
  - Khai báo trong lớp và không phải cài đặt (giống Java)
  - Trong C++, được gọi là phương thức thuần ảo (pure virtual method)
- Lớp trừu tượng có thể được sử dụng kế thừa nhưng không được phép tạo lập thể hiện.
- Do đó, một lớp đối tượng có thể đa kế thừa từ nhiều lớp trừu tượng (giống Java)
- Có nên thực hiện đa kế thừa?



# Nội dung

- Đa kế thừa
- Hàm ảo
- Hàm thuần ảo
- Hàm hủy ảo
- Bài tập



# Hàm ảo

## □ Con trỏ đối tượng trong kế thừa:

- ✓ Truy xuất đối tượng bằng con trỏ => linh động.
- ✓ Truy xuất đối tượng kế thừa bằng con trỏ lớp cơ sở.
- ✓ Kiểu con trỏ quyết định phương thức được gọi  
→ **liên kết tĩnh.**
- ✓ Đối tượng kế thừa truyền vào hàm nhận tham số kiểu cơ sở.  
→ **Đối tượng kế thừa có thể đóng vai trò đối tượng cơ sở.**

```
A obj;  
A *p;  
p = &obj;  
p = new A;
```

```
// B kế thừa A.  
B obj;  
A *p = &obj;  
p->func();
```

```
// B kế thừa A.  
void func(A obj) { }  
B obj;  
func(obj);
```

# Hàm ảo

- Ví dụ:

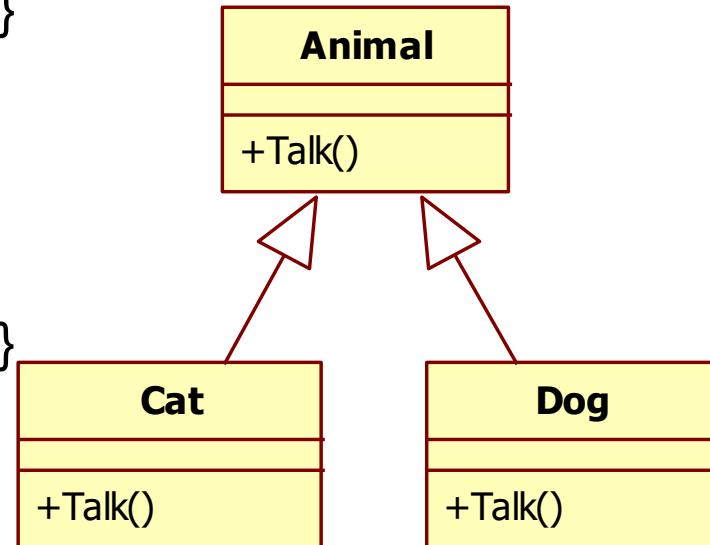
```
class Animal
{
public:
    void talk() { cout << "Don't talk!"; }
```

```
class Cat: public Animal
{
public:
```

```
    void talk() { cout << "Meo meo!"; }
```

```
class Dog: public Animal
{
public:
```

```
    void talk() { cout << "Gau gau!"; }
```



# Hàm ảo

- Ví dụ:

```
void giveATalk(Animal *p)
{
    p->talk();
}
```

```
void main()
```

```
{
    Cat    c;
    Dog    d;
```

```
giveATalk(&c);
giveATalk(&d);
```

```
}
```

```
void main()
```

```
{
```

```
Animal a;
Cat    c;
Dog    d;
```

```
Animal *p;
```

```
p = &a;
```

```
p->talk();
```

```
p = &c;
```

```
p->talk();
```

```
p->talk();
```

```
p->talk();
```

Animal talks!!

Animal talks!!

Animal talks!!

# Hàm ảo

## □ Khái niệm hàm ảo:

- ✓ Một phương thức của lớp.
- ✓ Mang tính ảo.
  - ➔ Chuyển lời gọi hàm cho đúng đối tượng con trỏ đang trỏ đến.
  - ➔ Liên kết động.
- ✓ Chỉ có ý nghĩa khi gọi thông qua con trỏ.
- ✓ Khai báo hàm ảo trong C++:  
**virtual <Chữ ký hàm>;**



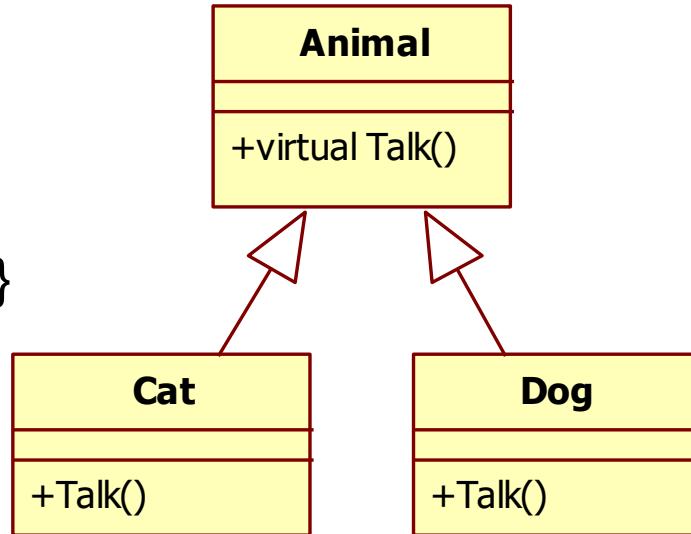
# Hàm ảo

- Ví dụ:

```
class Animal
{
public:
    virtual void talk() { cout << "Don't talk!"; }
};

class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo!"; }
};

class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau!"; }
};
```



# Hàm ảo

- Ví dụ:

```
void giveATalk(Animal *p)
{
    p->talk();
}
```

```
void main()
```

```
{
    Cat    c;
    Dog    d;
```

```
giveATalk(&c);
giveATalk(&d);
```

```
}
```

```
void main()
{
    Animal a;
    Cat    c;
    Dog    d;
    Animal *p;
    p = &a;
    p->Talk();
```

Animal talks!!

```
p = &c;
p->Talk();
```

Cat talks!!

```
p = &d;
p->Talk();
```

Dog talks!!

# Hàm ảo

## □ Sử dụng hàm ảo để làm gì?

- ✓ Gọn gàng, đơn giản, uyển chuyển, linh động.
- ➔ Chương trình có tính dễ mở rộng, nâng cấp.

```
void giveATalk(Animal *p)
{
    p->talk();
}
```

```
void giveATalk(Animal obj, int
iType)
{
    if (iType == 0)
    {
        Cat c = (Cat)obj;
        c.talk();
    }
    else if (iType == 1)
    {
        Dog d = (Dog)obj;
        d.talk();
    }
}
```



# Nội dung

- Đa kế thừa
- Hàm ảo
- Hàm thuần ảo
- Hàm hủy ảo
- Bài tập



# Hàm thuần ảo

- Có một số hàm ảo không thể cài đặt hoặc không có ý nghĩa khi cài đặt trong lớp cơ sở.

```
class Animal
{
public:
    virtual void talk() { cout << "Don't talk!"; }
};
```

**Biến thành hàm thuần ảo!!**



# Hàm thuần ảo

- Khái niệm hàm thuần ảo:
  - ✓ Hàm ảo chỉ có khai báo mà không có cài đặt.
  - ✓ Phần cài đặt do lớp kế thừa đảm nhận.
  - ✓ Khai báo trong C++:  
**virtual <Chữ ký hàm> = 0;**
- Lớp trừu tượng (abstract class):
  - ✓ Lớp chứa hàm thuần ảo.
  - ✓ Không thể tạo đối tượng từ lớp trừu tượng.
  - ✓ Chỉ dùng để kế thừa.



# Hàm thuận ảo

- Ví dụ:

```
class Animal
{
public:
    virtual void talk() = 0;
};

class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo!"; }
};

class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau!"; }
};
```

```
void main()
{
    Animal a;           // Sai.
    Animal *p = new Animal; // Sai.
    Animal *q = new Cat;   // Đúng.

    q->talk();
}
```

Cat talks!!



# Nội dung

- Đa kế thừa
- Hàm ảo
- Hàm thuần ảo
- Hàm hủy ảo
- Bài tập



# Hàm hủy ảo

- Ví dụ:

```
class GiaoVien
{
private:
    char    *m_strHoTen;
public:
    ~GiaoVien() { delete m_strHoTen; }
};

class GVCN : public GiaoVien
{
private:
    char    *m_strLopCN;
public:
    ~GVCN() { delete m_strLopCN; }
};
```

```
void main()
{
    GiaoVien  *p1 = new GiaoVien;
    delete p1;

    GVCN     *p2 = new GVCN;
    delete p2;

    GiaoVien *p3 = new GVCN;
    delete p3;
}
```



# Hàm hủy ảo

□ Dr. Guru khuyên:

✓ **Hàm hủy của lớp phải luôn là hàm ảo.**

➔ Chuyển lời gọi đến hàm hủy của lớp kế thừa.

```
class GiaoVien
{
private:
    char *m_strHoTen;
public:
    virtual ~GiaoVien() { delete m_strHoTen; }
};
```

```
GiaoVien *p3 = new GVCN;
delete p3;
```

~GVCN()  
~GiaoVien()



# Tóm tắt

## □ Đa kế thừa

## □ Hàm ảo:

- ✓ Chuyển lời gọi hàm đến đúng đối tượng.
- ✓ Chỉ có ý nghĩa khi gọi từ con trỏ.

## □ Hàm thuần ảo:

- ✓ Hàm ảo chỉ có khai báo mà không có cài đặt.
- ✓ Lớp kế thừa đảm nhận việc cài đặt.
- ✓ Lớp có chứa hàm thuần ảo → lớp trừu tượng
- ✓ Lớp trừu tượng chỉ dùng để kế thừa.

## □ Hàm hủy ảo:

- ✓ Hàm hủy phải luôn là hàm ảo.



# Nội dung

- Đa kế thừa
- Hàm ảo
- Hàm thuần ảo
- Hàm hủy ảo
- Bài tập



# Bài tập

## □ Bài tập 9.1:

class A

{ public:

[yyy] void f1() { cout << "Good morning.\n"; **f2();** }

[zzz] void f2() { cout << "Good afternoon.\n"; }

};

class B: public A

{ public:

void f1() { cout << "Good evening.\n"; **f2();** }

void f2() { cout << "Good night.\n"; }

};

void main()

{

A \*pObj = new B;

pObj->f1();

}

Cho biết những gì xuất hiện  
trên màn hình trong các

trường hợp:

a) [yyy] trống, [zzz] trống.

b) [yyy] trống, [zzz] virtual.

c) [yyy] virtual, [zzz] trống.

d) [yyy] virtual, [zzz] virtual.



# Bài tập

## □ Bài tập 9.2:

Có 2 loại hình: đường thẳng và hình chữ nhật.

- Đường thẳng: biểu diễn bởi hai điểm đầu cuối.

- Hình chữ nhật: biểu diễn bởi hai điểm trên trái và dưới phải.

Giả sử có danh sách các hình thuộc 2 loại trên. Viết chương trình xuất thông tin của từng hình trong danh sách đó.

Sau đó, giả sử có thêm loại hình mới là hình tròn.

- Hình tròn: biểu diễn bởi tâm và bán kính.

Khi đó, chương trình sẽ phải được chỉnh sửa như thế nào?



# Bài tập

## □ Bài tập 9.3:

Tốc độ chạy của các động vật cho bởi bảng sau:

Động vật	Tốc độ
Báo	100km/h
Linh dương	80km/h
Sư tử	70km/h
Chó	60km/h
Người	30km/h

Viết chương trình cho phép so sánh tốc độ chạy giữa một cặp động vật bất kỳ thuộc nhóm trên.

Thêm vào con ngựa chạy 60km/h, chương trình sẽ thay đổi thế nào?

