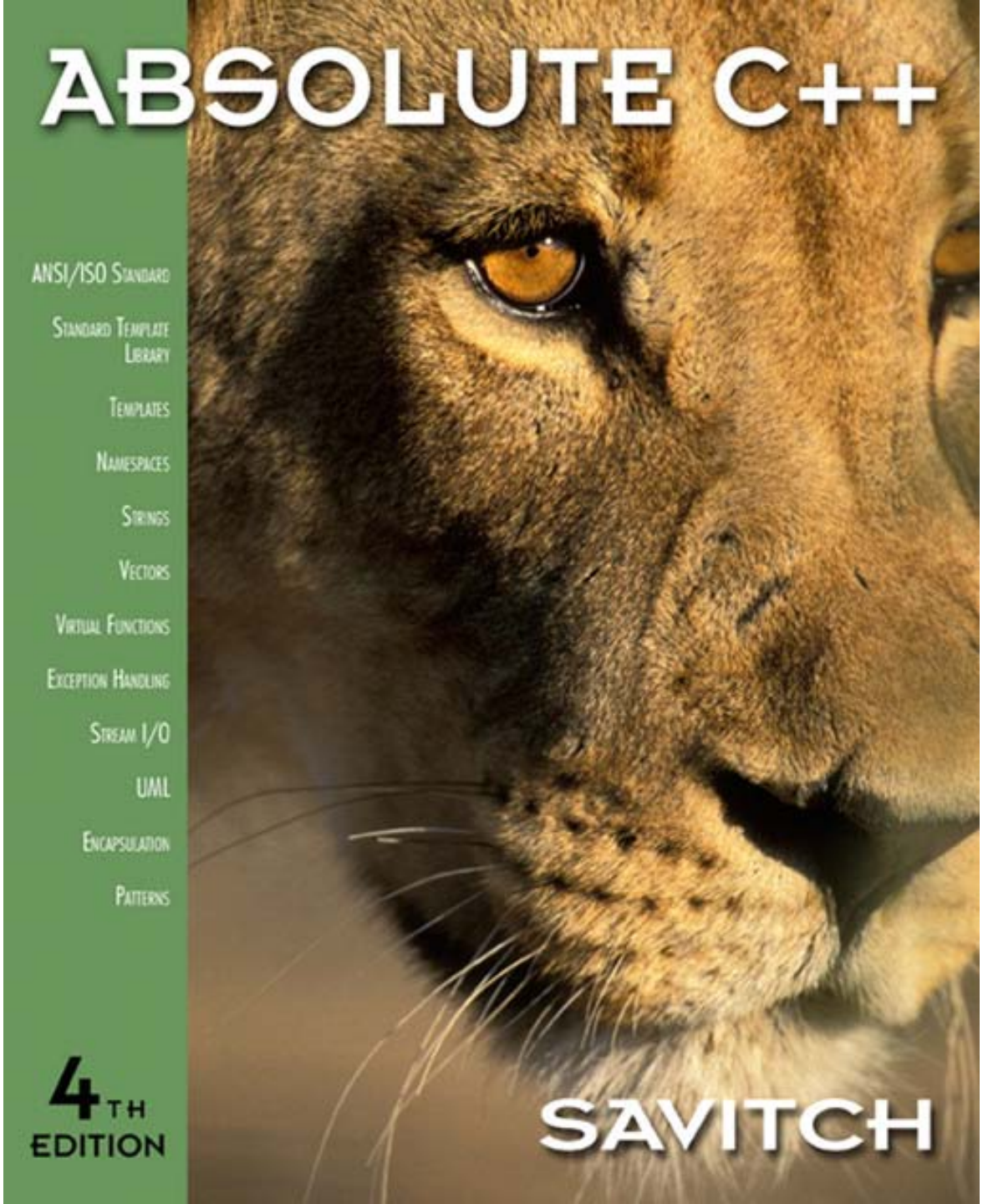
Bài 13: Tính thừa kế

Giảng viên: Hoàng Thị Điệp
Khoa Công nghệ Thông tin – ĐH Công Nghệ

Thuật ngữ

- inheritance: tính thừa kế
- derive: dẫn xuất / thừa kế
- base/parent class: lớp cơ sở / lớp cha
- derived/child class: lớp dẫn xuất / lớp con
- override: che khuất (khác overload)
  - function overriding khác function overloading
- multiple inheritance: đa thừa kế

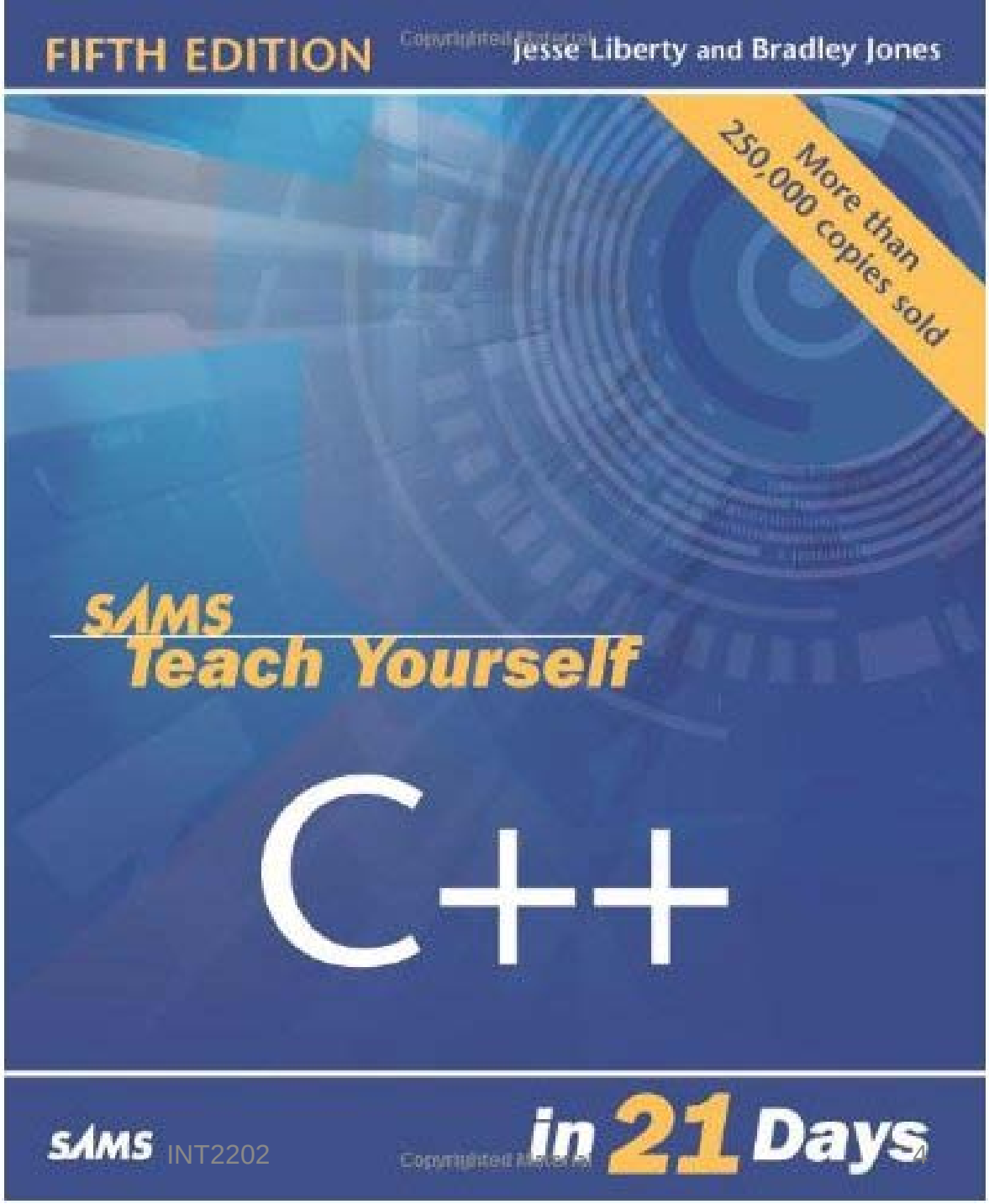


Chapter 14

Inheritance

PEARSON Addison Wesley

Day 12
Inheritance



Mục tiêu bài học

- Căn bản về tính thừa kế
  - Lớp dẫn xuất và hàm kiến tạo
  - Từ khóa protected
  - Định nghĩa lại hàm thành viên
  - Những hàm không được thừa kế
- Lập trình tính thừa kế
  - Toán tử gán và hàm kiến tạo sao chép
  - Hàm hủy trong lớp dẫn xuất
  - Đa thừa kế

Giới thiệu tính thừa kế

- Lập trình hướng đối tượng
  - Kĩ thuật lập trình mạnh
  - Cung cấp cơ chế trừu tượng gọi là thừa kế
- Trước tiên định nghĩa dạng tổng quát của lớp
  - Sau đó định nghĩa dạng cụ thể thừa kế các thuộc tính của dạng tổng quát
  - Và bổ sung/chỉnh sửa tính năng cho phù hợp với dạng cụ thể

Căn bản về tính thừa kế

- Một lớp mới thừa kế từ một lớp khác
- Lớp cơ sở
  - là lớp "tổng quát" để các lớp khác dẫn xuất
- Lớp dẫn xuất
  - là lớp mới
  - tự động sao từ lớp cơ sở:
    - các biến thành viên
    - các hàm thành viên
  - có thể thêm hàm và biến thành viên của riêng nó

Lớp dẫn xuất

- Xét ví dụ: lớp biểu diễn động vật Mammal
- Lớp này bao gồm: chó (Dog), mèo (Cat), ngựa (Horse), …
- Mỗi loài là một tập con của lớp động vật

Lớp cơ sở

- Một con vật phải thuộc một loài nào đó trong lớp động vật
- Động vật nói chung
  - có tuổi đời (itsAge)
  - có cân nặng (itsWeight)
  - có các khả năng cơ bản như kêu (speak), ngủ (sleep)
- Như vậy lớp tổng quát Mammal có thể chứa các thuộc tính chung cho tất cả động vật

Lớp Mammal

- Nhiều hàm/biến thành viên của lớp động vật Mammal có thể sử dụng cho tất cả các loài
  - Hàm truy cập biến thành viên
  - Hàm biến đổi biến thành viên
  - Các biến
    - tuổi đời
    - cân nặng
- Tuy nhiên ta sẽ không tạo đối tượng thuộc lớp Mammal

Lớp Mammal

- Xét hàm speak() :
  - Nó sẽ được “định nghĩa lại” trong các lớp dẫn xuất
  - Để những loài khác nhau có tiếng kêu khác nhau
    - Dog: “Ruff ruff woof woof”
    - Cat: “Meow meow meow”
    - Horse: “Winnie winnie”
    - Pig: “Oink oink”
  - Hàm speak() vô nghĩa khi ta chưa biết con vật thuộc loài nào
    - Chưa biết nó thuộc loài nào thì không biết tiếng kêu sẽ như thế nào
  - Do đó hàm speak() của Mammal chỉ in ra màn hình xâu “Mammal sound!”

Dẫn xuất từ lớp Mammal

- Các lớp dẫn xuất từ lớp Mammal:
  - sẽ tự động có tất cả các biến/hàm thành viên của Mammal
- Lớp dẫn xuất vì thế được hiểu là “thừa kế” các thành viên từ lớp cơ sở
- Sau đó có thể ĐỊNH NGHĨA LẠI các thành viên đã có sẵn hoặc THÊM thành viên mới

Giao diện của lớp Mammal

```
class Mammal{
public:
    // ham kien tao, ham huy
    Mammal(): itsAge(2), itsWeight(6){}
    ~Mammal(){}

    // ham truy cap bien thanh vien
    int getAge()const { return itsAge; }
    void setAge(int age) { itsAge = age; }
    int getWeight() const { return itsWeight; }
    void setWeight(int weight) { itsWeight = weight; }

    // ham khac
    void speak()const { cout << "Mammal sound!\n"; }
    void sleep()const { cout << "shhh. I'm sleeping.\n"; }

protected:
    int itsAge;
    int itsWeight;
};
```

Giao diện của lớp Dog

```
enum BREED {YORKIE, CAIRN, DANDIE, SHETLAND, DOBERMAN, LAB};

// Lop Dog mo phong cho
class Dog: public Mammal{
public:
    // ham kien tao, ham huy
    Dog(): itsBreed(YORKIE){}
    ~Dog(){}

    // ham truy cap bien thanh vien
    BREED getBreed() const { return itsBreed; }
    void setBreed(BREED breed) { itsBreed = breed; }

    // ham khac
    void speak()const { cout << "Ruff ruff woof woof!\n"; }
    void wagTail()const { cout << "Tail wagging...\n"; }
    void begForFood()const { cout << "Begging for food...\n"; }

private:
    BREED itsBreed;
};
```

Giao diện lớp Dog

- Nếu chia chương trình thành nhiều tệp
  - main.cpp, Mammal.h, Mammal.cpp, Dog.h, Dog.cpp
  - thì bạn cần bổ sung cấu trúc #ifndef
  - và khai báo các thư viện cần thiết
  - Trong main.cpp, khai báo “Dog.h”
- Dòng đầu của lớp Dog:
  class Dog: public Mammal
  - xác định nó thừa kế public từ lớp Mammal

Các thuộc tính bổ sung của lớp Dog

- Giao diện của lớp dẫn xuất chỉ liệt kê những thành viên mới hoặc “cần định nghĩa lại”
  - vì tất cả những thành viên khác thừa kế từ lớp cơ sở đã được định nghĩa rồi
  - nghĩa là, mọi con chó đều có itsAge và itsWeight...
- Lớp Dog bổ sung:
  - hàm kiến tạo
  - biến thành viên itsBreed
  - hàm thành viên getBreed(), setBreed(), wagTail(), begForFood()

Hàm thành viên được định nghĩa lại trong Dog

- Lớp Dog đã định nghĩa lại:
  - hàm thành viên speak()
  - phiên bản cài đặt nó trong Dog sẽ “che khuất” phiên bản cài đặt trong Mammal
- Do đó nó phải được khai báo lại trong giao diện lớp Dog
  - giống như những hàm thành viên mới bổ sung cho Dog
    - Giao diện gồm thành viên mới và thành viên “cần định nghĩa lại”

Thuật ngữ liên quan đến tính thừa kế

- Mô phỏng quan hệ gia đình
- Lớp cha
  - lớp cơ sở
- Lớp con
  - lớp dẫn xuất
- Lớp tổ tiên
  - cha của lớp cha …
- Lớp cháu
  - con của lớp con …

Hàm kiến tạo của lớp dẫn xuất

- Lớp dẫn xuất không thừa kế hàm kiến tạo của lớp cơ sở.
  - Nhưng chúng có thể được gọi từ hàm kiến tạo của lớp dẫn xuất
    - Đây là tất cả những gì ta cần.
- Hàm kiến tạo của lớp cơ sở phải khởi tạo tất cả các biến thành viên của lớp cơ sở
  - Lớp dẫn xuất sẽ thừa kế các biến này
  - Vì vậy hàm kiến tạo của lớp dẫn xuất sẽ gọi tới hàm này
    - Việc đầu tiên cần làm trong hàm kiến tạo của lớp cơ sở

Ví dụ hàm kiến tạo lớp dẫn xuất

- Ví dụ 2 hàm kiến tạo:

```
Mammal(int age, int weight): itsAge(age), itsWeight(weight){}

Dog(int age, int weight, BREED breed)
        : Mammal(age, weight), itsBreed(breed){}
```

- Phần sau dấu hai chấm là phần khởi tạo
  - chứa lời gọi tới hàm kiến tạo Mammal
- Nên luôn gọi đến hàm kiến tạo cơ sở nào đó

Nếu không gọi đến hàm kiến tạo cơ sở

- Hàm kiến tạo mặc định của lớp cơ sở sẽ được gọi tự động
- Các định nghĩa dưới đây là tương đương:

  Dog(): itsBreed(YORKIE){}

  Dog(): Mammal(), itsBreed(YORKIE){}

Lỗi thường gặp: Biến thành viên private của lớp cơ sở

- Lớp dẫn xuất thừa kế các biến thành viên
  - nhưng vẫn không có quyền truy cập trực tiếp đến chúng
  - kể cả thông qua các hàm thành viên của lớp dẫn xuất
- Biến thành viên private chỉ truy cập trực tiếp được trong các hàm thành viên của lớp định nghĩa nó

Lỗi thường gặp: Hàm thành viên private của lớp cơ sở

- Điều này cũng đúng cho hàm thành viên cơ sở
  - Không thể truy cập tới nó từ ngoài giao diện và cài đặt của lớp cơ sở
  - Kể cả trong định nghĩa hàm thành viên của lớp dẫn xuất
- Do đó
  - Hàm thành viên private chỉ nên là hàm làm các việc phụ trợ
  - Chỉ nên dùng trong lớp định nghĩa nó

Từ khóa chỉ định quyền truy cập: protected

- Phân loại mới cho thành viên của lớp
- Cho phép lớp dẫn xuất truy cập trực tiếp thành viên của lớp cơ sở “bằng tên”
  - Nhưng với những chỗ khác, chúng giống như private
- Trong lớp cơ sở → coi chúng là private
- Chúng được coi là "protected" trong lớp dẫn xuất
  - Để cho phép các dẫn xuất cháu chắt
- Nhiều người cho rằng cơ chế này vi phạm nguyên lý che giấu thông tin

Định nghĩa lại hàm thành viên

- Giao diện của lớp dẫn xuất:
  - chứa khai báo các hàm thành viên mới
  - và khai báo các hàm thành viên thừa kế nhưng cần thay đổi
  - Hàm thành viên thừa kế y nguyên thì không cần khai báo
- Cài đặt của lớp dẫn xuất:
  - sẽ định nghĩa các hàm thành viên mới
  - và định nghĩa lại các hàm thừa kế đã khai báo

Phân biệt: Định nghĩa lại và nạp chồng

- Rất khác nhau!
- Định nghĩa lại trong lớp dẫn xuất:
  - Còn gọi là che khuất
  - Danh sách tham số giống hệt hàm cơ sở
  - Về cơ bản là viết lại chính hàm đó
- Nạp chồng:
  - Danh sách tham số khác nhau
  - Định nghĩa hàm mới với tham số khác đi
  - Các hàm nạp chồng nhau phải có chữ kí khác nhau

Chữ kí của hàm

- Chữ kí gồm:
  - Tên hàm
  - Liệt kê các kiểu dữ liệu trong danh sách tham số
    - gồm thứ tự, số lượng, kiểu
- Chữ kí không bao gồm:
  - Kiểu trả về
  - Từ khóa const
  - &

Truy cập tới hàm cơ sở đã định nghĩa lại

- Hàm cơ sở khi bị định nghĩa lại sẽ không mất đi
- Ví dụ bên dưới minh họa 1 cách gọi tới nó:

```
Mammal tom;
Dog snoopy;
tom.speak();              → gọi tới hàm speak() của Mammal
snoopy.speak();           → gọi tới hàm speak() của Dog
snoopy.Mammal::speak();   → gọi tới hàm speak() của Mammal
```

Những hàm không được thừa kế

- Mọi hàm chuẩn của lớp cơ sở đều được lớp dẫn xuất thừa kế
- Ngoại trừ:
  - Hàm kiến tạo (ta đã tìm hiểu)
  - Hàm hủy
  - Hàm kiến tạo sao chép
    - Nếu không định nghĩa sẽ được sinh tự động 1 hàm mặc định
    - Cần định nghĩa nếu có biến thành viên private kiểu con trỏ
  - Toán tử gán
    - Nếu không định nghĩa → được sinh tự động 1 hàm mặc định

Toán tử gán và hàm kiến tạo sao chép

- Toán tử gán nạp chồng và hàm kiến tạo sao chép không được thừa kế
  - Nhưng có thể được gọi trong các định nghĩa của lớp dẫn xuất
  - Người ta thường làm cách này
  - Tương tự cách hàm kiến tạo của lớp dẫn xuất gọi tới hàm kiến tạo cơ sở

Ví dụ toán tử gán

- Dog được dẫn xuất từ Mammal:

```
Dog& Dog::operator =(const Dog& rightSide)
{
        Mammal::operator =(rightSide);
       …
}
```

- Chú ý dòng code gọi tới toán tử gán của lớp cơ sở
  - Nó đã xử lý tất cả công việc liên quan tới những biến thành viên được thừa kế
  - Sau lời gọi đó nên tiếp tục xử lý các biến thành viên của riêng lớp dẫn xuất

Ví dụ hàm kiến tạo sao chép

- Xét hàm:
  Dog::Dog(const Dog& object): Mammal(object), …
  {…}
- Sau dấu hai chấm là lời gọi tới hàm kiến tạo sao chép cơ sở
  - Lập giá trị các biến thành viên thừa kế từ lớp cơ sở cho đối tượng đang kiến tạo của lớp dẫn xuất
  - Chú ý là object có kiểu Dog nhưng nó cũng có kiểu Mammal nên đối số trong lời gọi là hợp lệ

Hàm hủy của lớp dẫn xuất

- Nếu hàm hủy của lớp cơ sở đã hoạt động chính xác
  - thì rất dễ viết hàm hủy cho lớp dẫn xuất
- Khi hàm hủy của lớp dẫn xuất được gọi:
  - chương trình sẽ **tự động** gọi hàm hủy của lớp cơ sở
  - Do đó không cần gọi tường minh
- Vậy hàm hủy dẫn xuất chỉ cần quan tâm tới các biến thành viên của riêng lớp dẫn xuất
  - và dữ liệu chúng trỏ tới (nếu có)
  - vì hàm hủy cơ sở đã tự đông xử lý các dữ liệu được thừa kế

Thứ tự gọi hàm hủy

- Xét:
  lớp B dẫn xuất từ lớp A
  lớp C dẫn xuất từ lớp B
  A ← B ← C
- Khi đối tượng thuộc lớp C ra ngoài phạm vi hoạt động:
  - Đầu tiên hàm hủy C được gọi
  - Sau đó hàm hủy B được gọi
  - Cuối cùng hàm hủy A được gọi
- Ngược với thứ tự gọi hàm kiến tạo

Quan hệ “IS A" và "HAS A”

- Thừa kế được xem là quan hệ "IS A”
  - Chó là động vật (Mammal ← Dog)
  - Ô tô là xe (Vehicle ← Car)
  - Xe tải là xe (Vehicle ← Truck)
- Một lớp chứa biến thành viên có kiểu định nghĩa bởi lớp khác được xem là một quan hệ “HAS A”
  - Trang trại có chó, mèo ngựa (Farm có biến thành viên kiểu Dog, Cat và Horse)

Thừa kế protected và private

- Là dạng thừa kế mới
  - hiếm khi sử dụng
- Thừa kế protected:
  class Cat: protected Mammal
  {…}
  - Các thành viên public của lớp cơ sở trở thành protected trong lớp dẫn xuất
- Thừa kế private:
  class Horse: private Mammal
  {…}
  - Tất cả các thành viên trong lớp cơ sở trở thành private trong lớp dẫn xuất

Đa thừa kế

- Lớp dẫn xuất có thể có hơn 1 lớp cơ sở
  - Cú pháp là liệt kê tên các lớp cơ sở (tách bằng dấu phẩy):
    class Mule: public Donkey, Horse
    {…}
- Nhiều khả năng nhập nhằng.
- Nguy hiểm khi sử dụng!
  - Một vài người cho rằng không nên dùng đa thừa kế
  - Chỉ nên dùng bởi lập trình viên kinh nghiệm.

Tóm tắt 1

- Thừa kế cho phép dùng lại mã
  - Cho phép một lớp dẫn xuất từ lớp khác và bổ sung các tính năng riêng
- Đối tượng của lớp dẫn xuất thừa kế các thành viên của lớp cơ sở
  - và có thể bổ sung thành viên
- Trong lớp dẫn xuất, không thể truy cập bằng tên tới biến thành viên private của lớp cơ sở
- Hàm thành viên private không được thừa kế

Tóm tắt 2

- Có thể định nghĩa lại hàm thành viên được thừa kế
  - Để nó hoạt động khác hàm cơ sở
- Thành viên protected của lớp cơ sở:
  - Có thể được truy cập bằng tên trong các hàm thành viên của lớp dẫn xuất
- Toán tử gán nạp chồng không được thừa kế
  - Nhưng có thể được gọi trong lớp dẫn xuất
- Hàm kiến tạo không được thừa kế
  - Được gọi từ hàm kiến tạo của lớp dẫn xuất