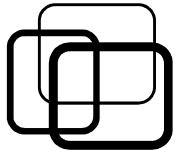
Kế thừa

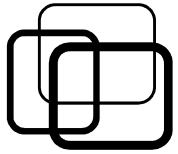
GV. Nguyễn Minh Huy

Nội dung



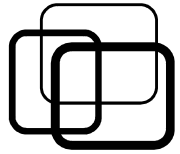
- Khái niệm kế thừa.
- Tầm vực trong kế thừa.
- Định nghĩa lại phương thức.
- Quan hệ IS-A và HAS-A.

Nội dung



- **Khái niệm kế thừa.**
- Tầm vực trong kế thừa.
- Định nghĩa lại phương thức.
- Quan hệ IS-A và HAS-A.

Khái niệm kế thừa



- Vấn đề trùng lắp thông tin:
  - Xét nhiều lớp có thông tin trùng nhau.
  - Có 2 dạng:
    - Dạng chia sẻ: $A \cap B \neq \emptyset$.
    - Dạng mở rộng: $B = A + \varepsilon$.
  - Nhược điểm:
    - Tốn thời gian, công sức.
    - Dung lượng lưu trữ lớn.
    - Thay đổi phần chung khó khăn.

A
x, y, z
a, b, c

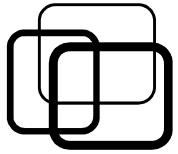
B
x, y, z
u, v, w

A
x, y, z

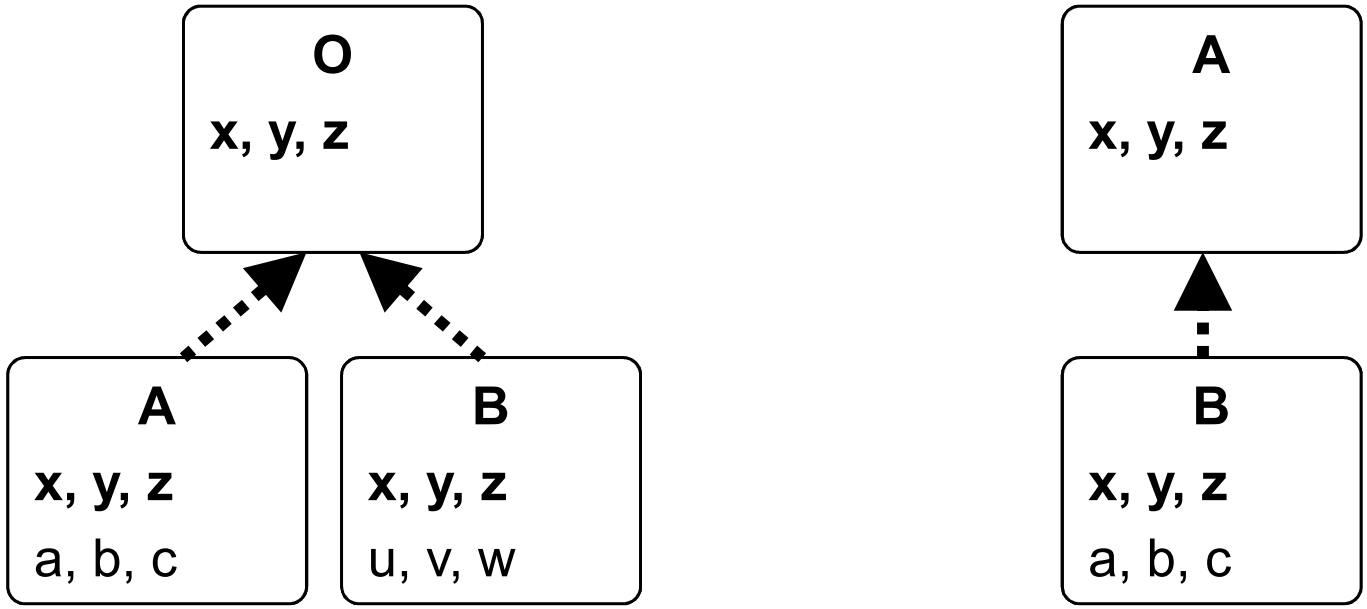
B
x, y, z
a, b, c

Giải quyết: tái sử dụng!!

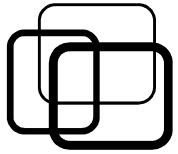
Khái niệm kế thừa



- Khái niệm kế thừa:
  - Định nghĩa lớp mới dựa trên những lớp đã có.
  - **Lớp kế thừa:** lớp được định nghĩa từ lớp đã có.
  - **Lớp cơ sở:** lớp dùng để định nghĩa lớp mới.
  - Lớp kế thừa thừa hưởng **TẤT CẢ** từ lớp cơ sở.



Khái niệm kế thừa



- Khai báo trong C++:

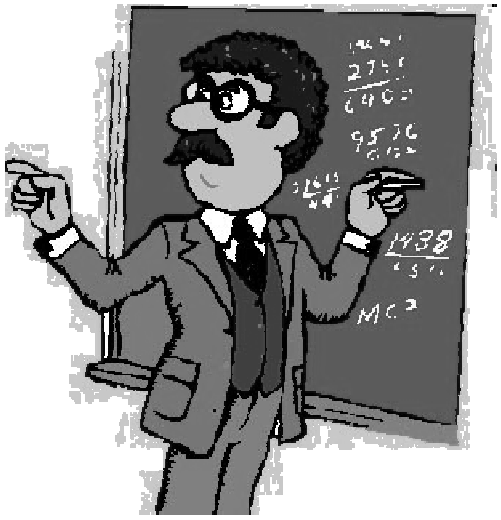
  class **<Lớp kế thừa> :** <*Loại kế thừa*> **<Lớp cơ sở>**

- Loại kế thừa:
  - public, private, protected.
- Ví dụ:

```
class A : public O
{
private:
        // Khai báo thuộc tính mới của A.
public:
        // Khai báo phương thức mới của A.
};
```

Khái niệm kế thừa

- Ví dụ:



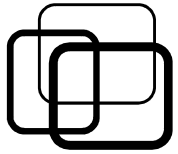
Giáo viên

- Thông tin:
  - Họ tên.
  - Mức lương.
  - Số ngày nghỉ.
- Công việc:
  - Giảng dạy.
  - Tính lương.

- Thông tin:
  - Họ tên.
  - Mức lương.
  - Số ngày nghỉ.
  - **Lớp chủ nhiệm.**
- Công việc:
  - Giảng dạy.
  - Tính lương.
  - **Sinh hoạt chủ nhiệm.**

GVCN

Khái niệm kế thừa



- Ví dụ:

```
class GiaoVien
{
private:
        char      *m_hoTen;
        float     m_mucLuong;
        int       m_ngayNghi;
public:
        GiaoVien(char  *hoTen,
                 float  mucLuong,
                 int  ngayNghi);
        void giangDay();
        float tinhLuong();
};
```

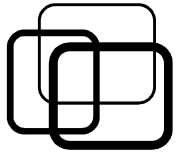
Lớp kế thừa

Lớp cơ sở

```
class GVCN : public GiaoVien
{
private:
        char      *m_lopCN;
public:
        GVCN(char  *hoTen,
                 float  mucLuong,
                 int  ngayNghi,
                 char  *lopCN);
        void sinhHoatCN();
};
```

GVCN thừa hưởng TẤT CẢ thuộc tính và phương thức của GiaoVien

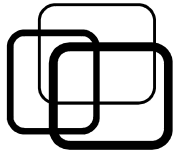
Khái niệm kế thừa



- Ví dụ:

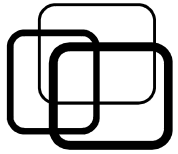
```
void main()
{
        GiaoVien   gv1("Minh", 500000, 5);
        gv1.giangDay();
        float  luong1 = gv1.tinhLuong();

        GVCN   gv2("Hanh", 700000, 3, "10A5");
        gv2.giangDay();
        gv2.sinhHoatCN();
        float  luong2 = gv2.tinhLuong();
}
```

Nội dung

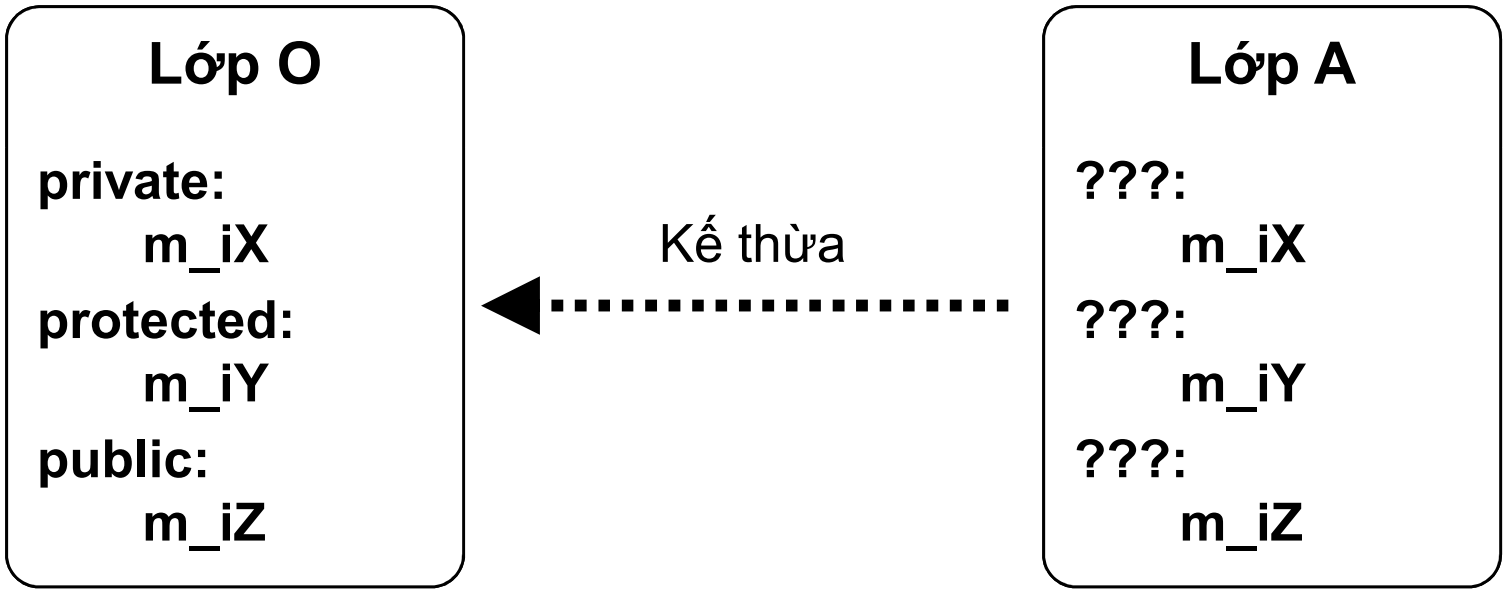


- Khái niệm kế thừa.
- **Tầm vực trong kế thừa.**
- Định nghĩa lại phương thức.
- Quan hệ IS-A và HAS-A.

Tầm vực trong kế thừa

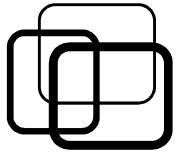


- Xét lớp A kế thừa lớp O:
  - A kế thừa toàn bộ thuộc tính và phương thức của O.
  - Tầm vực của những thành phần này thế nào?



➔ **Do loại kế thừa quyết định!!**

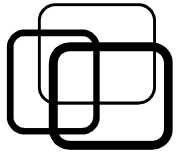
Tầm vực trong kế thừa



- Bảng tầm vực trong kế thừa:

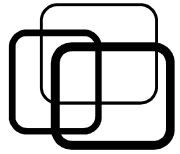
Tầm vực	**Kế thừa public**	**Kế thừa protected**	**Kế thừa private**
public	public	protected	private
protected	protected	protected	private
private	***Không thể truy xuất***	***Không thể truy xuất***	***Không thể truy xuất***

Nội dung



- Khái niệm kế thừa.
- Tầm vực trong kế thừa.
- **Định nghĩa lại phương thức.**
- Quan hệ IS-A và HAS-A.

Định nghĩa lại phương thức

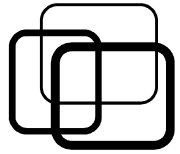


- Kế thừa một phần:
  - Có thể thay đổi những phần đã kế thừa??
  - Không kế thừa "máy móc" tất cả.
  - **Lớp kế thừa có thể thay đổi những gì đã kế thừa!!**

    ➔ **Định nghĩa lại phương thức đã kế thừa.**

Lớp kế thừa thừa hưởng TẤT CẢ thuộc tính và phương thức của lớp cơ sở TRỪ những phương thức được định nghĩa lại!!

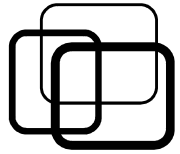
Định nghĩa lại phương thức



- Ví dụ:
  - GVCN kế thừa từ GiaoVien.
  - GVCN tính lương khác GiaoVien.
    - Lương GV = Mức lương – Số ngày nghỉ * 100,000.
    - Lương GVCN = Lương GV + Phụ cấp 1,000,000.

  ➔ Viết lại phương thức tinhLuong() cho lớp GVCN.

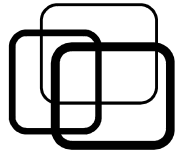
Định nghĩa lại phương thức



- Ví dụ:

```
class GiaoVien
{
private:
        char      *m_hoTen;
        float     m_mucLuong;
        int       m_ngayNghi;
public:
        GiaoVien(char  *hoTen, float  mucLuong, int   ngayNghi);
        void giangDay();
        float tinhLuong()
        {
                return m_mucLuong – m_ngayNghi * 100000;
        }
};
```

Định nghĩa lại phương thức

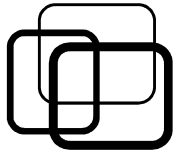


- Ví dụ:

```
class GVCN : public GiaoVien
{
private:
        char      *m_lopCN;
public:
        GVCN(char  *hoTen,
                  float  mucLuong,
                  int  ngayNghi,
                  char  *lopCN);
        void sinhHoatCN();
        float tinhLuong()
        {
                  return GiaoVien::tinhLuong() + 1000000;
        }
};
```

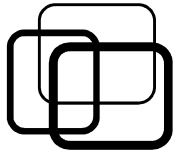
```
void main()
{
        GiaoVien   gv1("Minh", 500000, 5);
        gv1.giangDay();
        float  luong1 = gv1.tinhLuong();

        GVCN   gv2("Hanh", 700000, 3);
        gv2.giangDay();
        float  luong2 = gv2.tinhLuong();
}
```

Nội dung



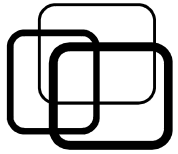
- Khái niệm kế thừa.
- Tầm vực trong kế thừa.
- Định nghĩa lại phương thức.
- **Quan hệ IS-A và HAS-A.**

Quan hệ IS-A và HAS-A



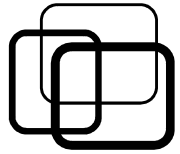
- Quan hệ IS-A:
  - Lớp A quan hệ IS-A với lớp B
    - A là một trường hợp đặc biệt của B.
    - A cùng loại với B.
- Ví dụ:
  - GVCN là một GiaoVien đặc biệt.
  - HinhVuong là một HinhChuNhat đặc biệt.
  - ConMeo là một ConVat đặc biệt.

Quan hệ IS-A và HAS-A



- Quan hệ HAS-A:
  - Lớp A quan hệ HAS-A với lớp B
    - A bao hàm B.
    - A chứa B.
    - B là một bộ phận của A.
- Ví dụ:
  - ChiecXe chứa BanhXe.
  - QuyenSach chứa TrangSach.

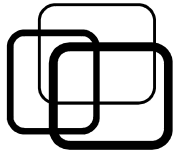
Quan hệ IS-A và HAS-A



- Dr. Guru khuyên: **luật xây dựng lớp.**
  - A có quan hệ IS-A với B.
    - ➔ **Cho A kế thừa B.**
  - A có quan hệ HAS-A với B.
    - ➔ **Cho B là một thuộc tính của A.**
- Ví dụ:

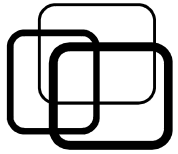
```
class ConMeo : public ConVat { };
class ChiecXe
{
private:
        BanhXe   *m_banhXe;
};
```

Tóm tắt



- Khái niệm kế thừa:
  - Định nghĩa lớp mới dựa trên những lớp đã có.
  - Lớp kế thừa thừa hưởng TẤT CẢ từ lớp cơ sở.
- Tầm vực trong kế thừa:
  - Tầm vực thay đổi tùy theo loại kế thừa.
- Định nghĩa lại phương thức:
  - "Cải biên" những phương thức kế thừa.
- Quan hệ IS-A và HAS-A:
  - IS-A: A là trường hợp đặc biệt của B.
  - HAS-A: A bao hàm B.

Bài tập

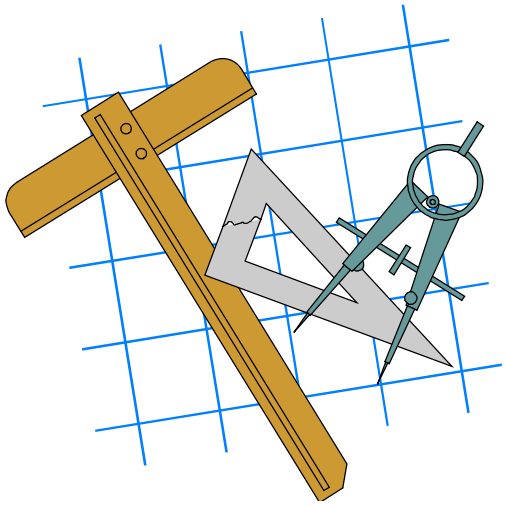


Bài tập 6.1:

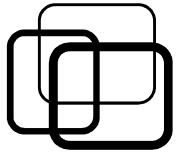
Những cặp đối tượng sau có quan hệ IS-A hay HAS-A?

Khai báo lớp cho từng cặp thể hiện quan hệ giữa chúng.

- Hình vuông / Hình chữ nhật.
- Đa giác / Cạnh.
- Giám đốc / Nhân viên.
- Hình tròn / Hình Ellipse.
- Máy bay / Động cơ.
- Câu / Từ.
- Mỹ phẩm / Hàng hóa.
- Cây lúa / Cây lương thực.
- Thư viện / Sách.
- Phim hoạt hình / Phim ảnh.



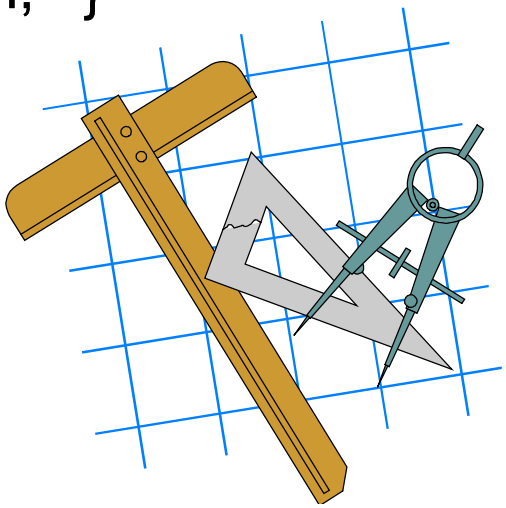
Bài tập



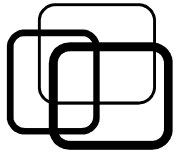
- **Bài tập 6.2:**

  Cho lớp TaiKhoan như sau:

```
class TaiKhoan
{
private:
        float      m_soDu;
public:
        float baoSoDu() {   return m_soDu;   }
        void napTien(float  soTien) {   m_soDu += soTien;   }
        bool rutTien(float soTien)
        {
                if (soTien <= m_soDu) {
                        m_soDu -= soTien;
                        return true; }
                return false;
        }
};
```



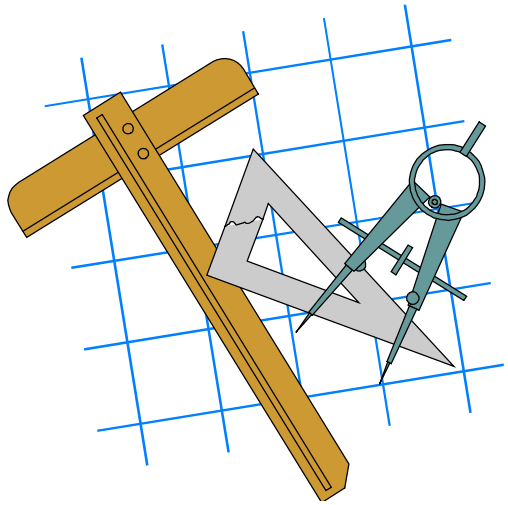
Bài tập



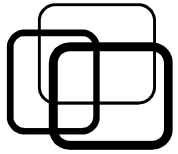
Bài tập 6.2:

Dựa trên lớp TaiKhoan, xây dựng lớp TaiKhoanTietKiem như sau:

- Có thêm thông tin:
  - Kỳ hạn gửi.
  - Lãi suất.
  - Số tháng đã gửi.
- Khi nạp tiền, tính lãi và số tháng gửi tính lại từ đầu.
- Khi rút tiền, tính lãi và số tháng gửi tính lại từ đầu.
- Cho phép tăng số tháng đã gửi.



Bài tập



Bài tập 6.3:

Một chiếc xe máy chạy không tải 100km tốn 2 lit xăng, cứ chở thêm 10kg hàng xe tốn thêm 0.1lit xăng (mỗi 100km).

Một chiếc xe tải chạy không tải 100km tốn 20lit xăng, cứ chở thêm 1000kg hàng xe tốn thêm 1lit xăng (mỗi 100km).

Dùng kế thừa để xây dựng lớp XeMay và XeTai cho phép:

- Chất một lượng hàng lên xe.
- Bỏ bớt một lượng hàng xuống xe.
- Đổ một lượng xăng vào xe.
- Cho xe chạy một đoạn đường.
- Kiểm tra xem xe đã hết xăng chưa.
- Cho biết lượng xăng còn trong xe.