Thiết kế chương trình theo hướng đối tượng

Nội dung

- Các nguyên tắc xây dựng lớp
- Các giai đoạn phát triển hệ thống
- Cách tìm lớp
- Các bước cần thiết để thiết kế chương trình
- Ví dụ minh họa
- Giới thiệu Rational Rose.

Các nguyên tắc xây dựng lớp

- Khi ta có thể nghĩ đến “nó” như một khái niệm riêng rẽ, xây dựng lớp biểu diễn khái niệm đó. Ví dụ lớp SinhVien.
- Khi ta nghĩ đến “nó” như một thực thể riêng rẽ, tạo đối tượng thuộc lớp. Ví dụ đối tượng Sinh viên “Nguyen Van A” (và các thuộc tính khác như mã số, năm sinh…).
- Lớp là biểu diễn cụ thể của một khái niệm, vì vậy lớp luôn luôn là DANH TỪ.
- Các thuộc tính của lớp là các thành phần dữ liệu, nên chúng luôn luôn là DANH TỪ.
- Các hàm thành phần là các thao tác chỉ rõ hoạt động của lớp nên các hàm này là ĐỘNG TỪ.
- Các thuộc tính dữ liệu phải vừa đủ để mô tả khái niệm, không dư, không thiếu.

Các nguyên tắc xây dựng lớp

- Các thuộc tính có thể được suy diễn từ những thuộc tính khác thì dùng hàm thành phần để thực hiện tính toán. Chu vi, diện tích tam giác là thuộc tính suy diễn.

```
// SAI
class TamGiac
{
   Diem A,B,C;
   double    ChuVi,
             DienTich;
public:
//...
};
```

```
// DUNG
class TamGiac
{
   Diem A,B,C;
public:
//...
   double ChuVi()const;
   double DienTich() const;
};
```

Các nguyên tắc xây dựng lớp

- Cá biệt có thể có một số thuộc tính suy diễn đòi hỏi nhiều tài nguyên hoặc thời gian để thực hiện tính toán, ta có thể khai báo là dữ liệu thành phần. Ví dụ tuổi trung bình của dân Việt Nam.

```
class QuocGia
{
   long DanSo;
   double DienTich;
   double TuoiTrungBinh;
//...
public:
   double TinhTuoiTB() const;
//...
};
```

Các nguyên tắc xây dựng lớp

- Chi tiết cài đặt, bao gồm dữ liệu và phần mã hoá các hàm thành phần có thể thay đổi uyển chuyển nhưng phần giao diện, nghĩa là phần khai báo các hàm thành phần cần phải cố định để không ảnh hưởng đến người sử dụng. Tuy nhiên nên cố gắng cài đặt dữ liệu một cách tự nhiên theo đúng khái niệm.

```
// NEN
class PhanSo
{
   int tu, mau;
public:
//...
};
```

```
// KHONG NEN
class PhanSo
{
   long tu_mau;
public:
//...
};
```

Các nguyên tắc xây dựng lớp

- Dữ liệu thành phần nên được kết hợp thay vì phân rã

```
// NEN
class TamGiac
{
   Diem A,B,C;
public:
//...
};
class HinhTron
{
   Diem Tam;
   double BanKinh;
public:
//...
};
```

```
// KHONG NEN
class TamGiac
{
   double xA, yA, xB, yB,
          xC, yC;
public:
//...
   };
class HinhTron
{
   double tx, ty,
          BanKinh;
public:
//...
};
```

Các nguyên tắc xây dựng lớp

- Trong mọi trường hợp, nên có phương thức thiết lập để khởi động đối tượng.
- Nên có phương thức thiết lập có khả năng tự khởi động không cần tham số.
- Nếu đối tượng có nhu cầu cấp phát tài nguyên thì phải có phương thức thiết lập, phương thức thiết lập sao chép để khởi động đối tượng bằng đối tượng cùng kiểu và có phương thức huỷ bỏ để dọn dẹp. Ngoài ra còn phải có phép gán (chương tiếp theo).
- Ngược lại, đối tượng đơn giản không cần tài nguyên riêng thì không cần phương thức thiết lập sao chép và cũng không cần phương thức huỷ bỏ.

Các giai đoạn phát triển hệ thống

- Phân tích yêu cầu (requirement analysis)
- Phân tích (Analysis)
- Thiết kế (Design)
- Lập trình (Programming)
- Kiểm tra (Testing)

Cách tìm lớp

- Trả lời các câu hỏi sau đây:
  - Có thông tin nào cần được lưu trữ hay phân tích không?
  - Có hệ thống bên ngoài hay không?
  - Có các mẫu thiết kế, thư viện lớp, component không?
  - Có thiết bị nào kết nối với hệ thống không?
  - Tác nhân đóng vai trò như thế nào trong hệ thống?

Các bước cần thiết để thiết kế chương trình

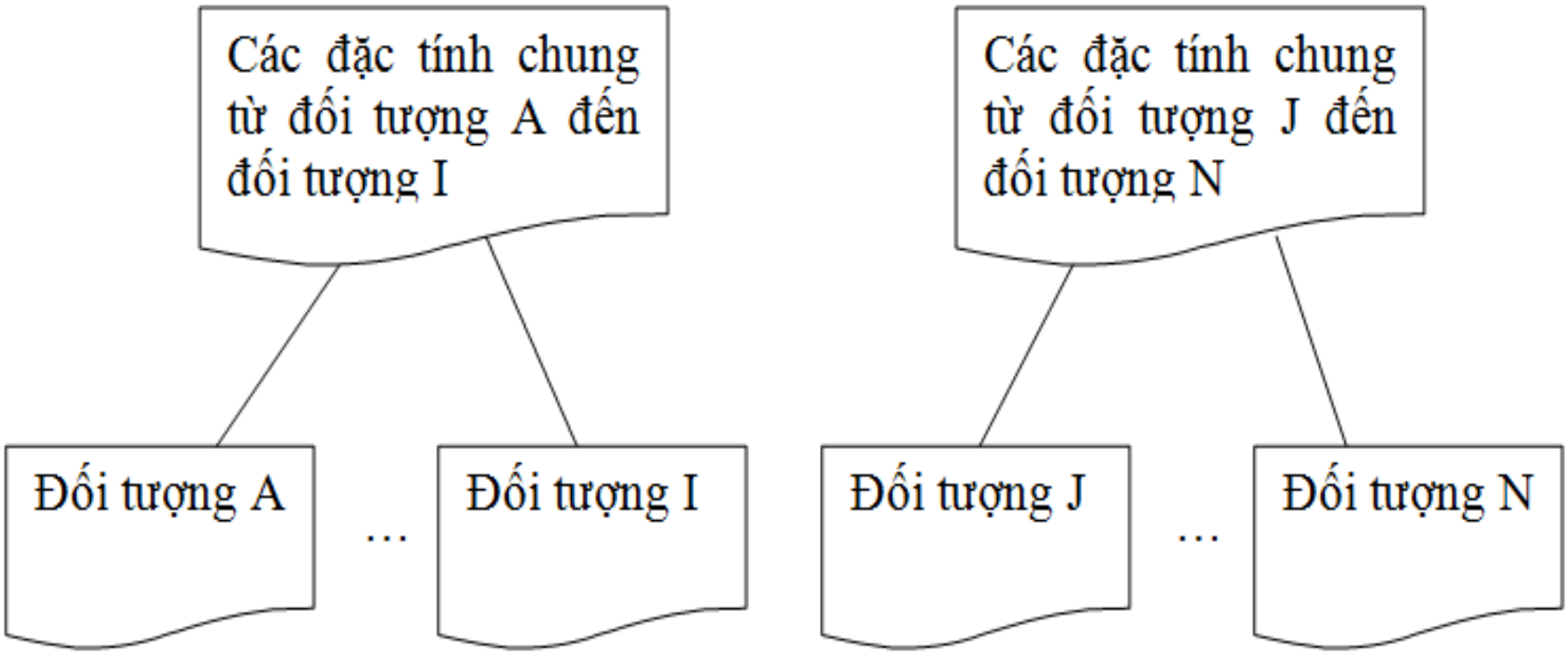
- Xây dựng các dạng đối tượng của bài toán
- Tìm kiếm các đặc tính chung của các đối tượng này
- Xác định được lớp cơ sở (Based-class)
- Xác định được lớp dẫn xuất (Derived-class)

→ Với 4 bước trên, ta có thể xây dựng được mối quan hệ giữa các lớp đối tượng với nhau

Các bước cần thiết để thiết kế chương trình

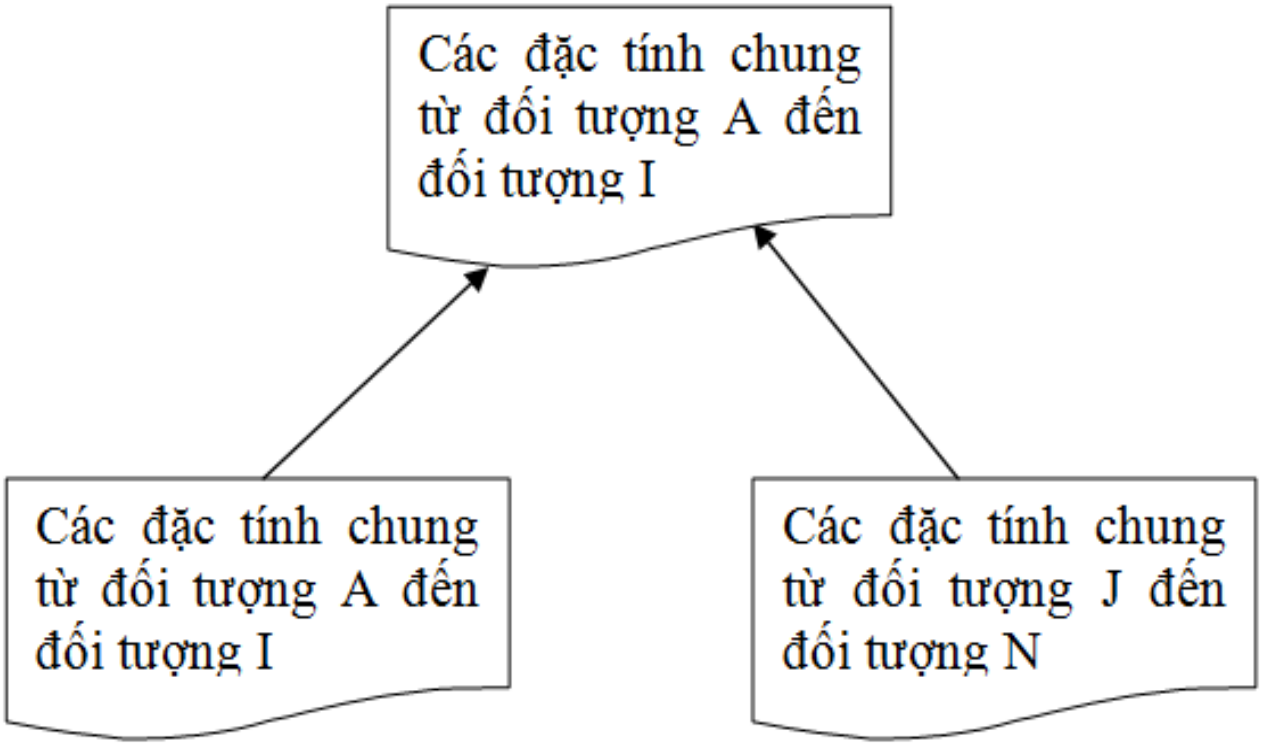
- Đối với loại bài toán phức tạp hơn, chúng ta cần phân tích theo thứ tự sau:
  - Phân tích bài toán từ dưới lên (Bottom up)
  - Tìm ra đặc tính chung giữa các lớp đối tượng

Các bước cần thiết để thiết kế chương trình



Các bước cần thiết để thiết kế chương trình

- Tiếp tục theo hướng từ dưới lên cho tất cả đối tượng từ A đến N



Ví dụ minh họa

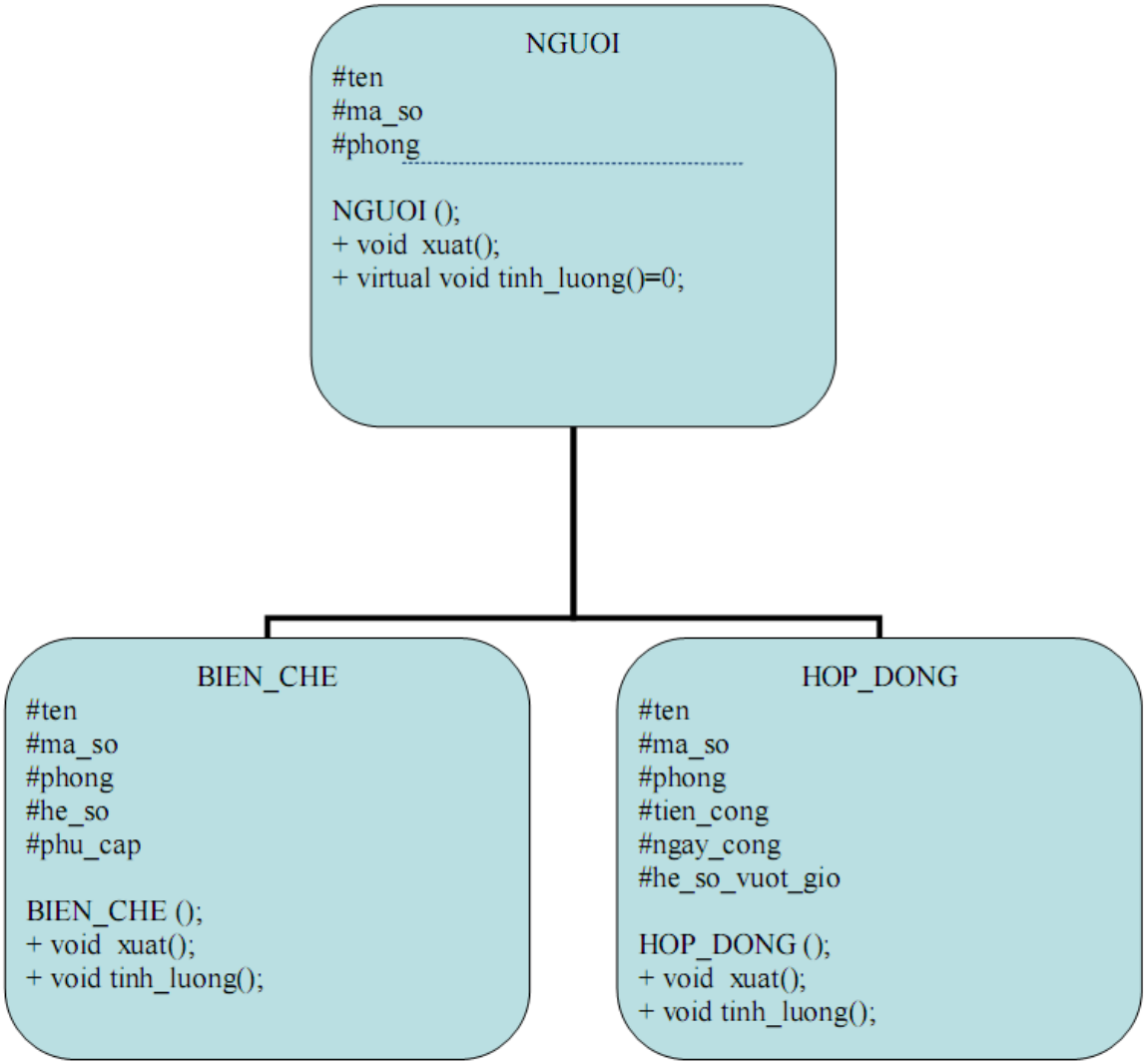
- Thiết kế chương trình tính tiền lương theo các dạng khác nhau

→Có 2 loại đối tượng chính: Biên chế và Hợp đồng

- Xác định các thuộc tính chung của 2 đối tượng

- Các thuộc tính còn lại nằm trong lớp dẫn xuất và kế thừa từ lớp cơ sở

Ví dụ minh họa



Ví dụ minh họa

```
#include <iostream.h>

#define MAX_TEN 50
#define MAX_MASO 5
 #define MUC_CO_BAN 120000

 class Nguoi
 {
 protected:
 char HoTen[MAX_TEN];
 char MaSo[MAX_MASO];
 float Luong;
 public:
 Nguoi();
 virtual void TinhLuong()=0;
 void Xuat() const;
 virtual void Nhap();
 };
```

Ví dụ minh họa

```
BienChe::BienChe()
{
HeSoLuong=HeSoPhuCap=0;
}

 void BienChe::Nhap()
 {
 Nguoi::Nhap();
 cout<<"He so luong:";
 cin>>HeSoLuong;
 cout<<"He so phu cap chu vu:";
 cin>>HeSoPhuCap;
 }

 void BienChe::TinhLuong()
 {
 Luong=MUC_CO_BAN*(1.0+HeSoLuong+HeSoPhuCap);
 }
```

```
class BienChe: public Nguoi
{
protected:
 float HeSoLuong;
 float HeSoPhuCap;
 public:
 BienChe();
 virtual void TinhLuong();
 virtual void Nhap();
 };
```

Ví dụ minh họa

```
class HopDong : public Nguoi
{
 protected:
 float TienCong;
 float NgayCong;
 float HeSoVuotGio;
 public:
 HopDong();
 virtual void TinhLuong();
 virtual void Nhap();
 };
```

```
HopDong::HopDong()
{
TienCong=NgayCong=HeSoVuotGio=0;
}

 void HopDong::Nhap()
 {
 Nguoi::Nhap();
 cout<<"Tien cong:";
 cin>>TienCong;
 cout<<"Ngay cong:";
 cin>>NgayCong;
 cout<<"He so vuot gio:";
 cin>>HeSoVuotGio;
 }

 void HopDong::TinhLuong()
 {
 Luong=TienCong*NgayCong*(1+HeSoVuotGio);
 }
```