Đại Học Quốc Gia Tp. Hồ Chí Minh
Trường ĐH Khoa Học Tự Nhiên
Khoa Công Nghệ Thông Tin

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

CHƯƠNG 6
ĐA XẠ

ThS: Phạm Nguyễn Sơn Tùng
Email: pnstung@fit.hcmus.edu.vn

NỘI DUNG BÀI HỌC

1. Định nghĩa Đa Xạ
2. Từ khóa Virtual
3. Abstract Class
4. Overloading & Overriding
5. Demo bài tập ứng dụng tại lớp

GIỚI THIỆU ĐA XẠ

- **Tính đóng gói** (*encapsulation*): Một đối tượng luôn luôn được bảo mật thông tin với người sử dụng. Tác động của bên ngoài không ảnh hưởng đến đối tượng.
- **Tính kế thừa** (*inheritance*): Giúp cho việc tái sử dụng lại một đối tượng sẵn có mà không cần phải định nghĩa lại.

GIỚI THIỆU ĐA XẠ

- **Định nghĩa:** Đa xạ hay còn gọi là đa hình tên tiếng anh **Polymorphism**, với một phương thức cùng tên ở nhiều lớp con khác nhau, lớp cơ sở sẽ hiểu được là lớp dẫn xuất nào đang gọi để thi hành phương thức của lớp dẫn xuất đó.

GIỚI THIỆU ĐA XẠ

```
#pragma once
#include "NVCongNhat.h"
#include "NVQuanLy.h"
#include "NVSanXuat.h"
class CCongTy
{
private:
    int sl_nvcn;
    CNVCongNhat ds_nvcn[100];
    int sl_nvsx;
    CNVSanXuat ds_nvsx[100];
    int sl_nvql;
    CNVQuanLy ds_nvql[100];
public:
    void Nhap();
    void Xuat();
    float tinhLuong();
    CCongTy();
    ~CCongTy();
};
```

```
#pragma once
#include "NhanVien.h"

class CCongTy
{
private:
    int slnv;
    CNhanVien dsns[100];

public:
    void Nhap();
    void Xuat();
    float tinhLuong();
    CCongTy();
    ~CCongTy();
};
```

GIỚI THIỆU ĐA XẠ

➢ Vấn đề gặp phải khi thay đổi.

- Các phương thức của lớp CCongTy sẽ thay đổi (void Nhap(), void Xuat(), float tinhLuong() → Làm sao để hiểu nhân viên đó là loại nào để truy cập đúng lớp nhân viên đó để thao tác.

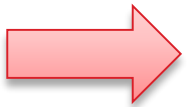
```
float CCongTy::tinhLuong()
{
    float tongluong = 0;

    for (int i = 0; i < sl_nvcn; i++)
        tongluong += ds_nvcn[i].tinhLuong();

    for (int i = 0; i < sl_nvsx; i++)
        tongluong += ds_nvsx[i].tinhLuong();

    for (int i = 0; i < sl_nvql; i++)
        tongluong += ds_nvql[i].tinhLuong();

    return tongluong;
}
```



```
float CCongTy::tinhLuong()
{
    float tongluong = 0;

    for (int i = 0; i < slnv; i++)
        tongluong += dsnv[i].tinhLuong();

    return tongluong;
}
```

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- **Virtual:** Là từ khóa nằm trước phương thức, khi một phương thức có từ khóa **virtual** nằm trước thì ta gọi là "**phương thức ảo**".
- **Phương thức ảo**: là phương thức chỉ có ở **lớp cơ sở** các **lớp dẫn xuất** có thể có hoặc không, nếu có thì khi khai báo sử dụng, nó sẽ sử dụng của chính nó. Nếu không có phương thức ảo của lớp cha sẽ được sử dụng.

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- **Phương thức ảo dùng để làm gì:** Dùng để lớp cơ sở (cha) điều hướng phương thức đó về đúng lớp dẫn xuất (con) mà phương thức đó thuộc về.

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

➢ Cú pháp:

virtual <kiểu trả về> <tên phương thức >(<d/s tham số>)

➢ Ví dụ:

```
#pragma once
#include "Date.h"
class CNhanVien
{
protected:
    string HoTen;
    CDate NgaySinh;
    string DiaChi;
public:
    virtual void Nhap();
    void Xuat();
    virtual float TinhLuong();
    CNhanVien(void);
    ~CNhanVien(void);
};
```

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- Một phương thức khi có từ khóa Virtual Method()=0 thì ta gọi là “phương thức thuần ảo”.
- Phương thức thuần ảo (pure Virtual): là phương thức chỉ có ở lớp cở sở, **không bắt buộc** lớp cơ sở phải định nghĩa. Nhưng ở lớp dẫn xuất bắt buộc phải có phương thức cùng tên với phương thức ảo của lớp cơ sở.

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- **Phương thức thuần ảo dùng để làm gì:** Có tác dụng tương tự như phương thức ảo, tiết kiệm công sức khi không cần định nghĩa trong lớp cha.

- **Cú pháp:**

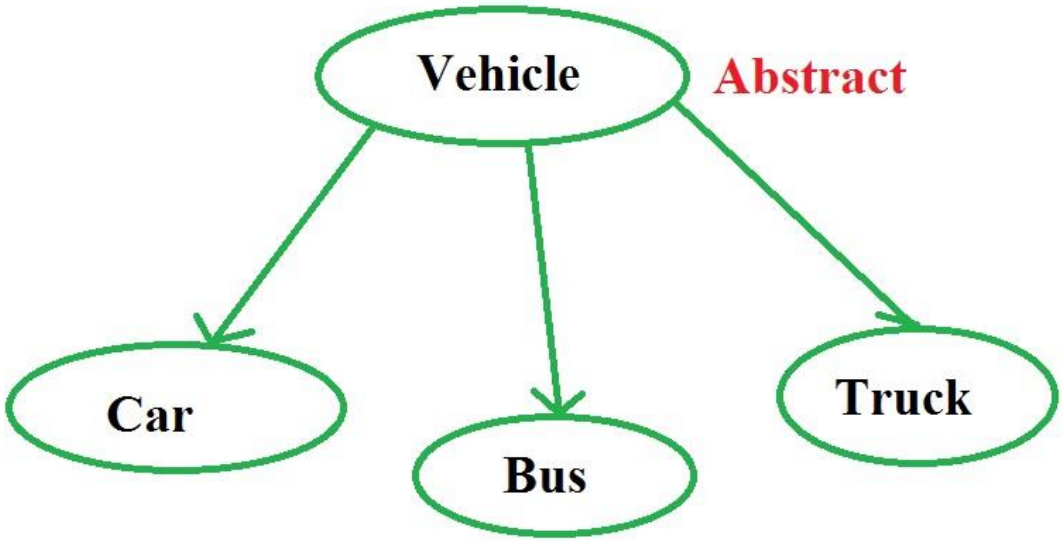
virtual <kiểu trả về> <tên phương thức >(<d/s tham số>)=0

- **Ví dụ:**

```
virtual float TinhLuong()=0;
```

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- **Lưu ý:** Khi một lớp đã có một phương thức thuần ảo, thì lớp đó sẽ trở thành lớp trừu tượng (Abstract class)



https://toidammeit.wordpress.com/2016/06/21/abstract-class-trong-oop/

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

- **Lưu ý:** Một lớp trừu tượng không được khai báo bình thường mà phải khai báo dạng con trỏ.

```
#include "CongTy.h"
#include "NhanVien.h"
#include "NVQuanLy.h"


void main()
{
    CNhanVien a;

    CNhanVien
```

Error: object of abstract class type "CNhanVien" is not allowed:
function "CNhanVien::TinhLuong" is a pure virtual function

TỪ KHÓA VIRTUAL TRONG ĐA XẠ

```
#include "CongTy.h"
#include "NhanVien.h"
#include "NVQuanLy.h"


void main()
{
    CNhanVien a;

    CNhanVien *b;

    CCongTy ABC;
    ABC.Nhap();
    float kq = ABC.TinhLuong();
    cout<<kq;
}
```

OVERLOADING VÀ OVERRIDING

- **Overloading** tức là tạo ra nhiều phương thức ở lớp cơ sở, có cùng tên nhưng khác nhau về danh sách tham số đầu vào (argument).

- **Tác dụng:** Tăng khả năng sử dụng phương thức của lớp. Trình biên dịch sẽ dựa vào tham số được truyền vào phương thức mà quyết định xem sẽ gọi phương thức nào trong danh sách phương thức overloading.

OVERLOADING VÀ OVERRIDING

Một phương thức có thể có nhiều tham số đầu vào:

```
#include <iostream>
using namespace std;
int main()
{
    string s;
    s.erase();
    return 0;
}
```

▲ 3 of 4 ▼ std::string & erase(**size_t _Off**, size_t _Count)

OVERLOADING VÀ OVERRIDING

➢ **Ví dụ:**

```
#pragma once
#include "Ngay.h"
class CNhanVien
{
protected:
    string hoTen;
    CNgay ngaySinh;
    string diaChi;
public:
    virtual void Nhap();
    void Xuat();
    virtual float tinhLuong()=0;
    virtual float tinhLuong(int);
    virtual float tinhLuong(float);
    CNhanVien();
    ~CNhanVien();
};
```

OVERLOADING VÀ OVERRIDING

- **Overriding** là tạo ra một phương thức trong lớp dẫn xuất có cùng loại với một phương thức trong lớp cơ sở. Và lớp dẫn xuất sẽ định nghĩa lại phương thức đó cho riêng mình.
- **Tác dụng:** Lớp dẫn xuất mong muốn ghi đè lại phương thức của lớp cha để sử dụng riêng cho mình.

OVERLOADING VÀ OVERRIDING

➢ Ví dụ:

CNhanVien

```
#pragma once
#include "Ngay.h"
class CNhanVien
{
protected:
    string hoTen;
    CNgay ngaySinh;
    string diaChi;
public:
    virtual void Nhap();
    void Xuat();
    virtual float tinhLuong()=0;
    virtual float tinhLuong(int);
    virtual float tinhLuong(float);
    CNhanVien();
    ~CNhanVien();
};
```

CNVCongNhat

```
#pragma once
#include "NhanVien.h"
class CNVCongNhat: public CNhanVien
{
private:
    int soNgayCong;
public:
    void Nhap();
    void Xuat();
    float tinhLuong();
    float tinhLuong(int);
    float tinhLuong(float);
    CNVCongNhat();
    ~CNVCongNhat();
};
```

OVERLOADING VÀ OVERRIDING

➢ **Đặc điểm:**

- Phương thức overriding và được overriding phải có chung kiểu trả về, tên phương thức và danh sách tham số.
- Overriding chỉ xảy ra giữa các lớp có quan hệ kế thừa.

BÀI TOÁN MINH HỌA

➢ **Bài tập 1:** Một công ty ABC có 3 loại nhân viên.

Nhân viên quản lý

Nhân viên công nhật

Nhân viên sản xuất

BÀI TOÁN MINH HỌA



- Yêu cầu: Hãy tính lương cho các nhân viên của công ty biết:
  - Lương NV quản lý: Lương cơ bản x Hệ số lương
  - Lương NV công nhật: Số ngày công x 120.000
  - Lương NV sản xuất: Số sản phẩm x 50.000

BÀI TẬP ÁP DỤNG

Bài tập 2: Quản lý siêu thị (Tập tin word).

TÀI LIỆU THAM KHẢO

- 1. Lập trình hướng đối tượng, Trần Đan Thư, Đinh Bá Tiến, Nguyễn Tấn Trần Minh Khang, NXB Khoa Học Kỹ Thuật, 2010.
- 2. Lập trình hướng đối tượng, Trần Văn Lăng, NXB Thống Kê, 2004.
- 3. Object-oriented Programming in c++, 4th Edition, Robert Lafore, SAMS, 1997.
- 4. C++ Primer, Fifth Edition, Stephen Prata, SAMS, 2004.
- 5. Slide bài giảng của: Thầy Nguyễn Minh Huy, Thầy Hồ Tuấn Thanh, Thầy Đinh Bá Tiến, Thầy Trần Văn Lăng, Thầy Đặng Bình Phương, Cô Đặng Thị Thanh Nguyên.
- 6. Các website về lập trình:
  - http://www.cplusplus.com/
  - http://stackoverflow.com/
  - http://www.codeproject.com/