ĐỀ CƯƠNG BÀI GIẢNG
BÀI 1
ĐỊNH NGHĨA VÀ SỬ DỤNG LỚP

1. Khái niệm

Đối tượng: thế giới hữu hình được tạo nên từ các đối tượng. Đây là khái niệm dùng để chỉ một thực thể trong thế giới thực.

Trong lập trình, đối tượng là một thành phần của chương trình. Một chương trình được tạo ra bằng cách sinh ra các đối tượng và cho chúng tương tác với nhau hoặc thực hiện các phương thức hoạt động của chúng. Mỗi đối tượng có một tập các thuộc tính và một tập các phương thức.

Thuộc tính: là các đặc điểm của đối tượng. Trong lập trình, thuộc tính được mô tả bằng các biên, dùng để lưu trữ dữ liệu của đối tượng.

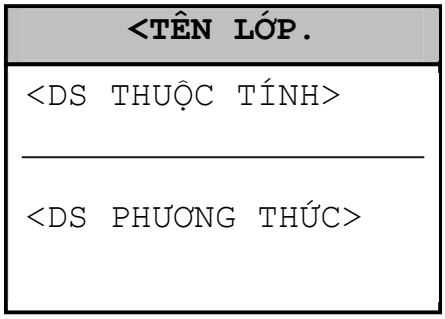
Phương thức: hay phương thức hoạt động của đối tượng là các thủ tục mô tả các hoạt động, thao tác của đối tượng. Trong lập trình, phương thức được định nghĩa bởi các hàm.

Lớp: là khái niệm trừu tượng dùng để chỉ tập tất cả các đối tượng có cùng tập thuộc tính và phương thức.

2. Đọc - hiểu sơ đồ lớp

Đa số các bài tập LTHĐT thường cho dưới dạng sơ đồ lớp. Chúng ta cần cài đặt một bài tập theo một sơ đồ lớp cho trước và đáp ứng được các đòi hỏi kèm theo.

Theo chuẩn UML, một lớp được biểu diễn bằng một hình chữ nhập, trong đó ghi tên lớp, danh sách các thuộc tính, danh sách các phương thức như hình dưới đây:



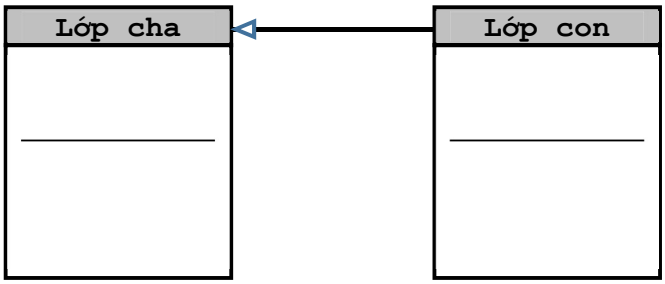
Hình 1. Biểu diễn một lớp

Như vậy, bằng cách nhìn vào sơ đồ lớp, trước hết cần phải xác định: có bao nhiêu lớp cần cài đặt trong bài; với mỗi lớp, cần xác định sơ qua các thuộc tính và phương thức của chúng.

Một sơ đồ lớp không chỉ có duy nhất 1 lớp mà thường chứa nhiều lớp. Các lớp trong sơ đồ không độc lập với nhau mà thường có những mối liên hệ nhất định. Vì vậy, để đọc chính xác một sơ đồ lớp, cần phải nắm vững các mối liên hệ giữa các lớp.

Theo chuẩn UML, hai lớp bất kỳ có thể có nhiều kiểu liên hệ với nhau. Tuy nhiên, để đơn giản, tài liệu này sẽ trình bày 2 kiểu liên hệ chính:

- **Liên hệ cha con:** Nếu hai lớp A và B có liên hệ cha con thì mối liên hệ này được biểu diễn bằng một mũi tên rỗng, nối giữa A và B. Chiều của mũi tên chỉ từ lớp con về lớp cha.

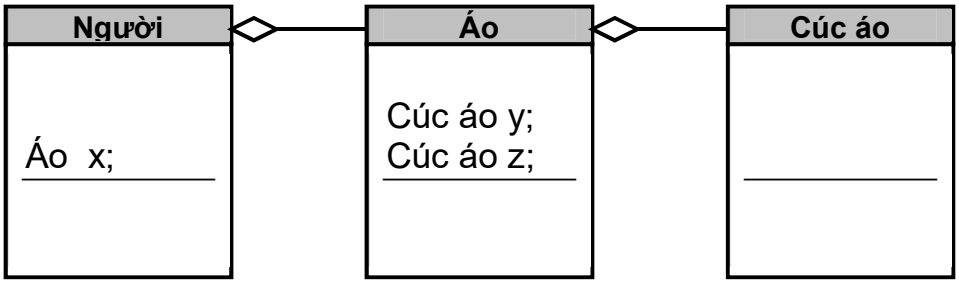


Hình 2. Mũi tên chỉ mối liên hệ cha – con

- **Liên hệ kết tập (hàm chứa):** Bạn là một đối tượng thuộc lớp "Người". Chiếc áo bạn đang mặc chỉ là một thuộc tính của bạn; tuy nhiên nó lại là một đối tượng thuộc lớp áo. Vậy giữa lớp "Áo" và lớp "Người" có quan hệ với nhau như thế nào?

Trên thực tế có rất nhiều mối quan hệ kiểu như vậy. Chẳng hạn với lớp áo, những chiếc cúc gắn trên đó chỉ là các thuộc tính của áo, nhưng nó lại là các đối tượng thuộc lớp "Cúc áo".

Khi đó, ta nói: lớp "Cúc áo" kết tập vào lớp "Áo" và lớp "Áo" kết tập vào lớp "Người". Để biểu diễn mối quan hệ giữa hai lớp A và B trong đó B kết tập vào A, người ta dùng một đường nối giữa A và B, đầu phía A, người ta vẽ một hình thoi rỗng. Vì vậy, có thể biểu diễn mối quan hệ giữa 3 lớp "Người", "Áo", "Cúc áo" theo hình sau:

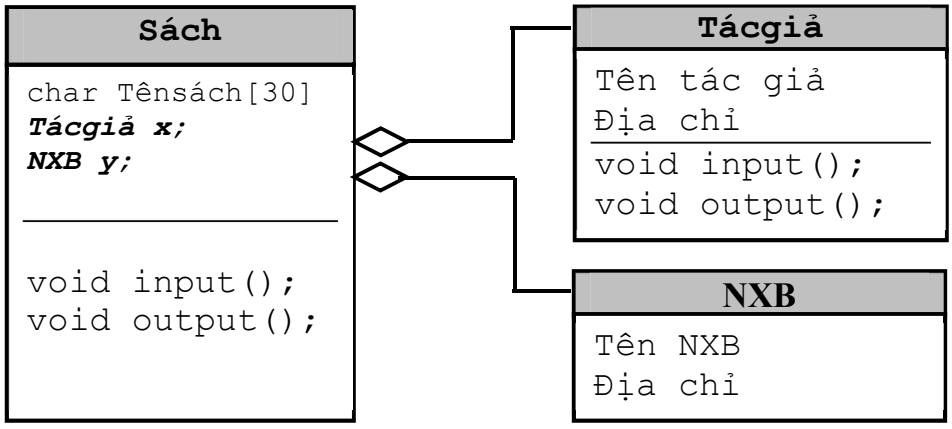


Hình 3. Mối quan hệ kết tập.

Dễ thấy x là một thuộc tính của lớp "Người", tuy nhiên nó lại là một đối tượng thuộc lớp "Áo" (hay còn gọi là áo x); và thuộc tính này đã thể hiện mối quan hệ kết tập của lớp "Áo" vào lớp "Người". Khi đọc sơ đồ lớp, hãy quan tâm tới những thuộc tính kiểu như vậy. Tương tự thuộc tính y, z của lớp "Áo" thể hiện sự kết tập của lớp "Cúc áo" vào lớp "Áo". Những thuộc tính như vậy sẽ không có kiểu nguyên thủy (int, float, char...) mà có kiểu là một tên lớp (x có kiểu Áo; y, z có kiểu Cúc áo).

Ví dụ: cho sơ đồ gồm 3 lớp: Sách, Tác giả, NXB và hai lớp Tác giả, NXB kết tập vào lớp Sách. Tuy nhiên, trong lớp Sách, các bạn hãy chú ý tới 2 thuộc tính thể hiện sự kết tập này: "Tác giả" (1) , "Nhà xuất bản" (2). Tác giả của một cuốn sách phải là một đối tượng thuộc lớp "Tác giả"; vì vậy ta cần hiểu (1) chính là một đối tượng thuộc lớp "Tác giả"; cách hiểu chính xác phải là: Tác giả x; Tương tự, với thuộc tính (2), ta cần hiểu đó chính là NXB y; với x, y là hai đối tượng mà ta tự đặt tên, thuộc lớp "Tác giả" và lớp "NXB".

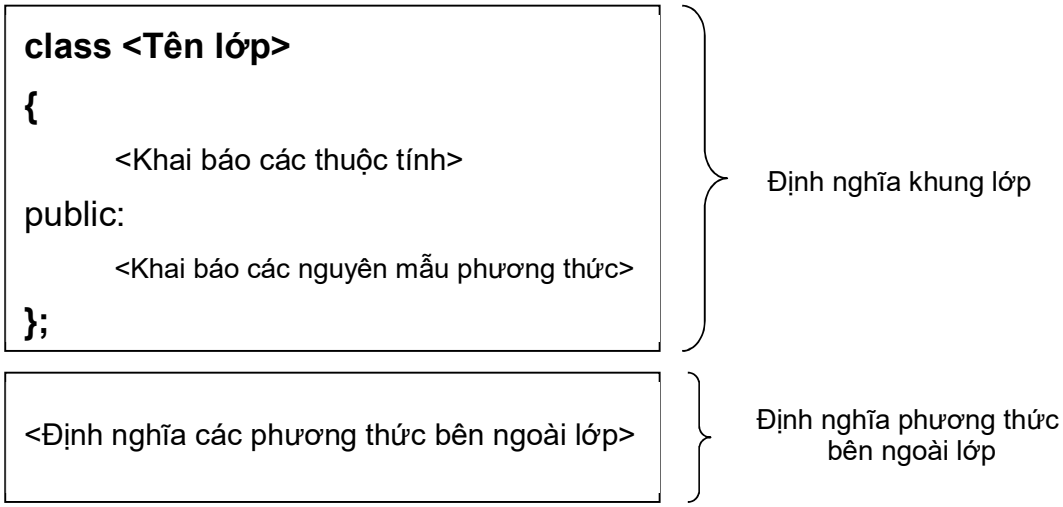
Vậy cụ thể, sơ đồ lớp trên được hiểu như sau:



Hình 4. Cách hiểu rõ hơn các thuộc tính kết tập.

3. Định nghĩa một lớp đơn giản

Một lớp thường được cài đặt theo cách sau (để không mất tập trung, ta chỉ xét 1 cách phổ biến).



Hình 5. Định nghĩa một lớp

Như vậy ta chia công việc này thành 2 phần: định nghĩa khung lớp và định nghĩa các phương thức ngoài lớp. Với khung lớp, mỗi thuộc tính là một biến. Biến này có thể có kiểu nguyên thủy (int, float, char...) hoặc kiểu lớp (nếu nó thể hiện quan hệ kết tập – xin xem phần trên).

Trước phần <Khai báo các nguyên mẫu phương thức> ta đặt từ khóa "public:". Từ khóa này nhằm xác định phạm vi truy cập "công cộng" cho các phương thức của lớp. Mỗi thuộc tính hay phương thức ta đều có thể chỉ định 1 phạm vi truy cập cho chúng với 3 mức độ truy cập là **private** (riêng tư – chỉ được truy cập trong nội bộ lớp), **public** (công cộng – được truy cập ở mọi nơi) và **protected** (được bảo vệ - chỉ được truy cập trong nội bộ lớp và các lớp con kế thừa lớp này).

Cách đặt phạm vi truy cập: ta viết

`<Tên phạm vi truy cập> :`

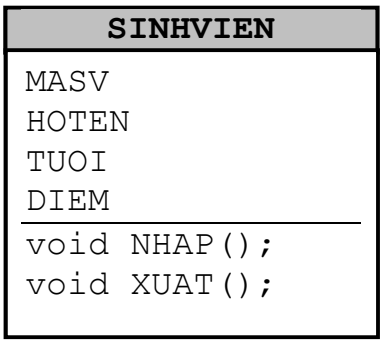
Phạm vi này sẽ ảnh hưởng tới tất cả các thuộc tính, phương thức đứng sau nó cho tới khi gặp một phạm vi truy cập khác. Các thuộc tính không được chỉ định phạm vi truy cập thì mặc định có phạm vi truy cập **private** (riêng tư).

Mỗi phương thức là một hàm được viết theo quy tắc viết hàm của C++ như sau:

```
<Kiểu trả về> <Tên phương thức> ([Danh sách các đối])
{
     //Thân phương thức
}
```

Chú ý: Khi định nghĩa các phương thức ở ngoài lớp, người ta thêm TÊN_LỚP:: vào trước tên phương thức nhằm xác định xem phương thức đó thuộc về lớp nào.

Ví dụ: Cài đặt lớp theo sơ đồ sau:



- Cài đặt khung lớp:

```
class SINHVIEN
{
     char MASV[30];
     char HOTEN[50];
     int TUOI
     float DIEM
public:
     void NHAP();
     void XUAT();
};
```

- Cài đặt phương thức ngoài lớp:

```
void SINHVIEN::NHAP()
{
     //Thân phương thức NHAP()
}
```

```
void SINHVIEN::XUAT()
{
     //Thân phương thức XUAT()
}
```

Về nội dung của các phương thức (thân phương thức) thường rất đơn giản. Với ví dụ trên ta chỉ việc tiến hành nhập, xuất các thuộc tính của lớp tương ứng. Với thuộc tính kiểu xâu, ta dùng lệnh gets(Thuộc tính); để nhập và chú ý làm sạch bộ đệm bàn phím sau khi nhập bằng lệnh fflush(stdin); để xuất, ta dùng lệnh cout<< "nội dung cần xuất"; và dùng endl để xuống dòng. Bây giờ, hãy bắt đầu với phương thức NHAP()/ XUAT() của lớp "SINHVIEN":

```
void SINHVIEN::NHAP()
{
     cout<<"Ma sv  : "; fflush(stdin); gets(MASV);
     cout<<"Ho ten : "; fflush(stdin); gets(HOTEN);
     cout<<"Tuoi    : "; cin>>TUOI;
     cout<<"Diem  : "; cin>>DIEM;
}
```

```
void SINHVIEN::XUAT()
{
     cout<<"Ma sv: "<<MASV<<endl;
     cout<<"Ho ten: "<<HOTEN<<endl;
     cout<<"Tuoi: "<<TUOI<<endl;
     cout<<"Diem :"<<DIEM<<endl;
}
```

Phương thức XUAT() có thể được viết theo cách 2 để xuất dữ liệu của mỗi sinh viên trên 1 dòng như sau:

```
void SINHVIEN::XUAT()
{
     cout<<setw(10)<<MASV<<setw(20)<<HOTEN
     <<setw(20)<<TUOI<<setw(20)<<DIEM<<endl;
}
```

4. Sử dụng các lớp

Sau khi định nghĩa lớp, ta có thể sinh các đối tượng thuộc lớp.

- Sinh 1 đối tượng:

Cú pháp	Ví dụ
Tên_lớp Tên_đối_tượng;	SINHVIEN a;

- Sinh một mảng đối tượng:

Cú pháp	Ví dụ
Tên_lớp * Tên_Mảng;	SINHVIEN* a;
Tên_Mảng = new Tên_Lớp[n];	a = new SINHVIEN[10];

- Truy cập các thành phần của đối tượng:

Cú pháp	Ví dụ
Tên_ĐT.Tên_thuộc_tính;	**a.MASV:** truy cập tới thuộc tính mã sinh viên của đối tượng a.
Tên_ĐT.Tên_phương_thức();	**a.NHAP();** nhập thông tin cho đối tượng a.

- Truy cập các thành phần của mảng đối tượng:

Cú pháp	Ví dụ
Tên_mảng[i].Tên_thuộc_tính;	**a[i].MASV:** truy cập tới thuộc tính mã sinh viên của đối tượng thứ i trong mảng a.
Tên_mảng[i].Tên_phương_thức();	**a[i].NHAP();** nhập thông tin cho đối tượng thứ I trong mảng a.