**Giải Thích Về Đối Tượng Class**



Class(Đối Tượng) có:

Private: thuộc tính

Public: phương thức / hành động

VD: con gà:

private: chân, đầu, đuôi, mắt,… **=>những đặc tính có thể mô tả ra nó**

public: ăn, chạy, đẻ trứng,…. **=>hành động**

Sinh viên:

private: mã số, họ tên, …

public: nhập, xuất….

**private: bảo vệ dữ liệu. còn public muốn làm gì làm.**

VD: con gà: mắt(private) được bảo vệ tức nó còn 2 con mắt. Không ai làm đuôi 1 con mắt của nó được.

Tương tự sinhvien: diemThi(private) được bảo vệ tức diemThi=7 thì không ai sửa điểm thi lại bằng 10 được. Nếu cố gắng sửa bằng cách gán diemThi=10 ngoài class(vd: hàm main, … các hàm không thuộc class này) sẽ báo lỗi.

**Vậy main… không thể đụng tới các private nên cần get, set trong class**(get, và set nằm bên trong nên đụng đến private được) **để có thể đụng được.**

+ hàm main… muốn gán giá trị diemThi=10 thì sử dụng set

+ hàm main… muốn biết giá trị diemThi bao nhiêu thì sử dụng get

* Xong một đối tượng bây giờ qua nhiều đối tượng

VD: **con gà** nó chỉ có thể ăn cho nó thôi không thể ăn dùm con khác được

Nên hành động ăn chỉ ở trong 1 con gà.

Tương tự **sinhvien** phương thức nhập chỉ nhập được cho chính nó không thể nhập cho những đứa khác được.

**Vậy muốn nhập nhiều sinh viên phải làm sao?**

Mình sẽ xài mảng tạo mảng SinhVien sv[20];

Có nghĩa là: mảng có tên là: sv gồm 20 phần tử

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | … | 14 | 15 | 16 | 17 | 18 | 19 |

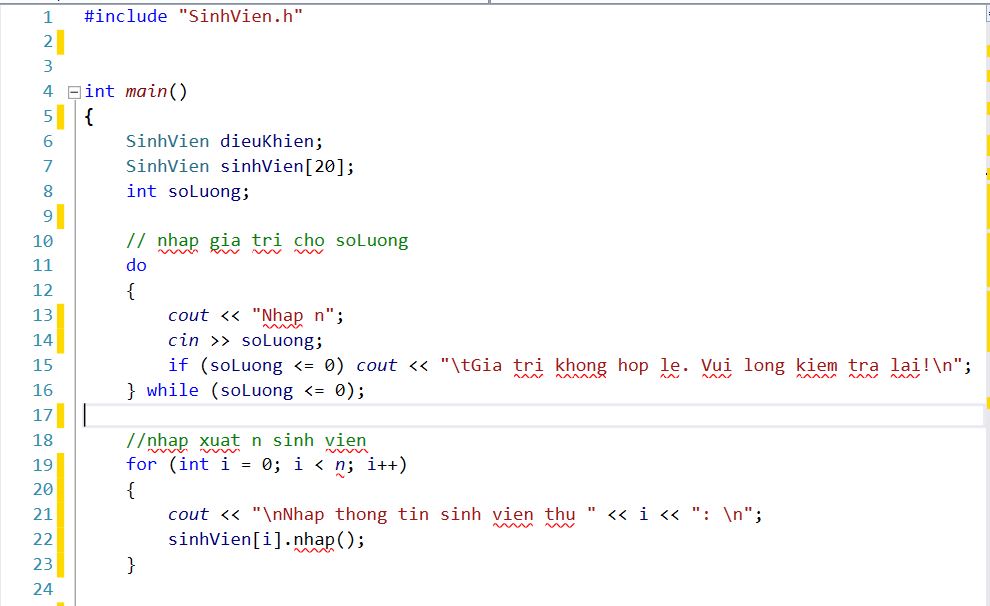
Mỗi phần tử như vậy là một thằng SinhVien có đầy đủ thuộc tính(private) và hành động(public).

Tạo ra sẵn như vậy nhưng không biết người dùng sử dụng bao nhiêu nên tạo biến lưu số lượng người ta sử dụng.

Vd: người ta muốn xài 5 thì người ta nhập vô 5.

Int soLuong; nhập vào số lượng 5 => soLuong có giá trị là 5

Tương tự như dưới đây!



**Mình cần làm trong main. Muốn biết tại sao thì gần phía dưới có câu trả lời nha!**

Muốn nhập nhiều sinh viên mà mình biết được số lượng mà người ta cần rồi. Nên chỉ cần gọi bao nhiêu thằng đó ra là được. VD: nhập 5 thì gọi 5 thằng sinh viên ra xài.

Mảng của mình từ 0 - > 19 mỗi thằng như vậy là 1 sinh viên nên 5 thằng là: 0 1 2 3 4 => đủ 5 thằng. Thấy thằng cuối cùng là 4 nhỏ hơn 1 so với thằng người ta nhập nên có thể viết: for(int i=0; i<n;i++). i=5 nhỏ hơn 5 là 4 3 2 1 0.

Còn nếu xài từ 1 thì tương tự: 1 -> 19 =>5 thằng là 1 2 3 4 5. Thằng cuối bằng 5 un nên viết: for(int i=1; i<=5; i++) i<=5 => 5,4,3,2,1

**Vậy mình gọi được 5 thằng rồi!**

**Vd: Bây giờ có 5 con gà muốn cho 5 con gà này ăn thì chỉ cần mỗi con đều ăn là được.**

**Tương tự. Bây giờ muốn nhập thông tin cho nhiều thằng sinh viên thì mỗi thằng nó gọi hành động(public) nhập của nó.**

For(int i=0; i < n; i++)

{

SV[i].nhap**(); Nếu i=0 thì là thằng sinh vien 0, 1 là sinh vien 1…**

}

Trước khi nhập phải cho người ta biết nhập vô đâu; nên thêm **cout**.

For(int i=0; i < n; i++)

{

cout << “Nhap sinh vien thu ” << i << “: ”;

SV[i].nhap**(); Nếu i=0 thì là thằng sinh vien 0, 1 là sinh vien 1…**

}

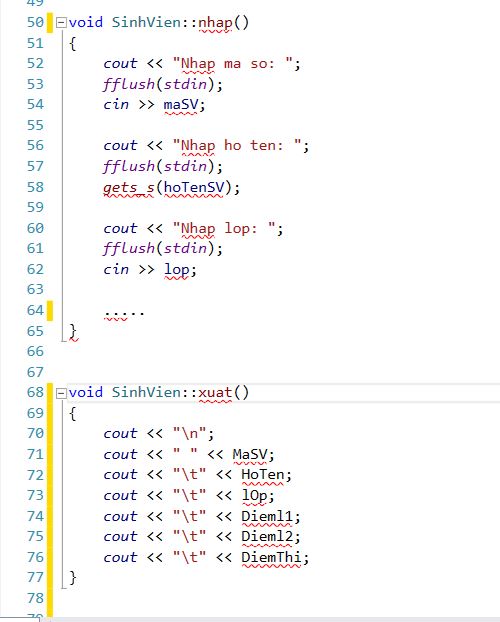
Tương tự với xuất cũng gọi nó ra rồi cho nó gọi hành động(public) xuất

For(int i=0; i < n; i++)

{

SV[i].xuat**(); Nếu i= thì là thằng sinh vien 0,…**

}



**Vậy nếu làm hàm nhập nhiều thằng trong file Header.h được không?**

=>Vậy mình cứ nghĩ 1 con gà có thể ăn(phương thức/ public) dùm những con gà khác được không thì sẽ trả lời được câu hỏi này!

Nếu đến đây hiểu rồi có thể không cần xem phần này!

T cho 1 vd nữa nha!

In ra thằng có điểm KTM >=5

**Vậy trước tiên cần gọi nó ra.**

For(int i=0; i<n; i++)

{

SV[i]

}

**Gọi phương thức(public) tính điểm kết thúc môn**

For(int i=0; i<n; i++)

{

SV[i].diemKTM();

}

**Có được điểm kết thúc môn nhưng đề có điều kiện >=5 vậy thêm vào**

For(int i=0; i<n; i++)

{

**if** ( SV[i].diemKTM() >=5)

{

}

}

**Có được điều kiện rồi mình làm gì thì làm. ví dụ cho thằng sinh viên đó xuất ra thì gọi phương thức(public) xuất**

For(int i=0; i<n; i++)

{

**if** ( SV[i].diemKTM() >=5)

{

SV[i].xuat();

}

}