Phương pháp Lập trình Hướng đối tượng

Thiết kế Lớp Đối tượng

"An toàn là trên hết !"

GV: Lê Xuân Định

Nhắc lại khái niệm "biến"

"Cuộc đời" của một biến "thụ động" (trước HĐT):

- "**Ra đời**": Khai báo biến. `int x...`
  - Có khi được *trang bị đầy đủ*: **Khởi tạo** giá trị mặc định.
    `int x = 0;`
  - Có khi bị "*đem con bỏ chợ*": Không có giá trị xác định.
    `int x; //x = ?`
- Bị đem ra **sử dụng** (thường lặp lại nhiều lần)
  - Bị "**đọc**": `int y=x; cout<<(x+1); f(x);`
  - Bị "**ghi**": `x=y-1; cin>>x; x=g(123);`
- "**Qua đời**": Hết phạm vi sử dụng (tầm vực, scope).
  `for(int i;i<10;i++){cout<<i;}` //ở đây i đã "chết"

Bài toán Mẫu: "SV trong nhóm"

- Hãy viết chương trình cho 1 SV **làm bài tập**, **làm việc nhóm**, **đi thi**, và **tính điểm tổng kết**.
  - Mỗi SV có một **MSSV** & **tên** cố định trong suốt quãng đời SV.
  - Mỗi SV được GV gán vào một **nhóm** nào đó (có thể thay đổi).
  - **đTK** = (đLT*6 + đTH*4) / 10 + đCộng
  - **Điểm LT** và **điểm TH** của SV chỉ có được thông qua hành động "***thi LT***", "***thi TH***". (Muốn thay đổi thì phải "thi lại", tức thực hiện hành động "thi" một lần nữa.)
  - **Điểm cộng** chỉ được tích luỹ thông qua hành động "***làm việc nhóm***". (Mỗi lần làm làm việc nhóm thì điểm cộng tăng thêm một ít.)

Cuộc đời của biến đối tượng

- "**Ra đời**": Khai báo biến. `SinhVien sv...`
  - Được *trang bị đầy đủ*: **Khởi tạo** giá trị mặc định.
    `SinhVien sv = SinhVien("0964123","Cam");`
  - Nếu muốn "*đem con bỏ chợ*" cũng **không được**.
    ~~`SinhVien sv;`~~
- Thực hiện các **hành động**
  - Cho phép "**đọc**":
    `char nhom =` ~~`sv.nhom;`~~
  - Cho phép "**ghi**":
    ~~`sv.nhom = 'A';`~~
  - Các hành động khác: `sv.lamNhom();`
    `sv.thiLT(); sv.thiTH(); float dtk=sv.tinhDTK();`
- "**Qua đời**": Hết phạm vi sử dụng (tầm vực, scope).
  - Phương thức phá huỷ (nếu có) được gọi.

VD: Hàm main() của “SV trong nhóm”

```
void main()
{
  SinhVien a("001","An"); //tương đương v  ới dòng dưới
  //SinhVien a = SinhVien("001","An")
  cout<<"Nhap diem cho SV "
      <<a.layMSSV()<<" ("
      <<a.layTen()<<"): "<<endl;
  a.ganNhom('A'); a.lamNhom();
  a.thiLT(); a.thiTH();
  cout<<"diem tong ket = "<<a.tinhDTK()<<endl;
}
```

PThức Khởi tạo (Constructor): *"Sinh con thì phải đặt tên!"*

- Quy tắc *an toàn* với *Biến*: **Khai báo** biến phải gắn liền với **khởi tạo** giá trị mặc định!
  - Được tích hợp vào đối tượng qua *pthức khởi tạo*.
- PThức khởi tạo: Đặt **giá trị xác định** cho **tất cả các thuộc tính** của đối tượng ngay ***từ lúc mới "ra đời"***.
  - Nguyên mẫu hàm: Tên pthức trùng với tên lớp & không khai báo kiểu trả về (không khai báo **`void`**).
  - Sử dụng (khai báo & khởi tạo biến đối tượng):
    - Kiểu rút gọn (phổ thông): `Lớp biến(các đối số);`
    - Kiểu tường minh: `Lớp biến = Lớp(các đối số);`

Các loại PThức khởi tạo

- Mỗi **lớp** có thể cài đặt **nhiều pt khởi tạo** khác nhau, nhưng mỗi **đối tượng** chỉ được khởi tạo **1 lần** duy nhất bởi **1 pt khởi tạo** nào đó.
- PT khởi tạo Mặc định (default constructor): Ko có tham số: Đặt tất cả các thuộc tính bằng giá trị **mặc định**.
  - VD:
```
PhanSo::PhanSo() {this->tu=0; this->mau=0;}
PhanSo p; //không có ngoặc!!! Tương đương với dòng dưới
//PhanSo p = PhanSo();
```
- PT khởi tạo có tham số: Gán các tham số cho các thuộc tính tương ứng; Những thuộc tính còn lại đặt bằng giá trị **mặc định**.

Hàm Tạo khác P.Thức Đặt G.Trị

```
C::C(int ij):i(1),j(ij),k(0){}
```

- Constructor có tham số: đặt giá trị cho các thuộc tính bằng giá trị đối số;
- Mỗi constructor phải đảm bảo *giá trị xác định cho **mọi thuộc tính***;
- Constructor chỉ ***được gọi 1 lần*** trong đời của mỗi đối tượng; "*Sinh ra chỉ có 1 lần!*"
- Constructor thường & nên ***dùng danh sách khởi tạo để đặt giá trị*** *cho thuộc tính*;

```
C::setJ(int ij){this->j = ij;}
```

- Setter cũng đặt giá trị cho các thuộc tính bằng giá trị đối số; Nhưng...
- Mỗi setter thường chỉ đặt giá trị cho **1 thuộc tính**;
- Setter thường *được gọi đi **gọi lại nhiều lần*** trong đời đối tượng để cập nhật giá trị;
- Setter chỉ có thể ***dùng phép gán để cập nhật giá trị*** *cho thuộc tính*;

Thay Hàm tạo bằng P.Thức Đặt Giá trị?

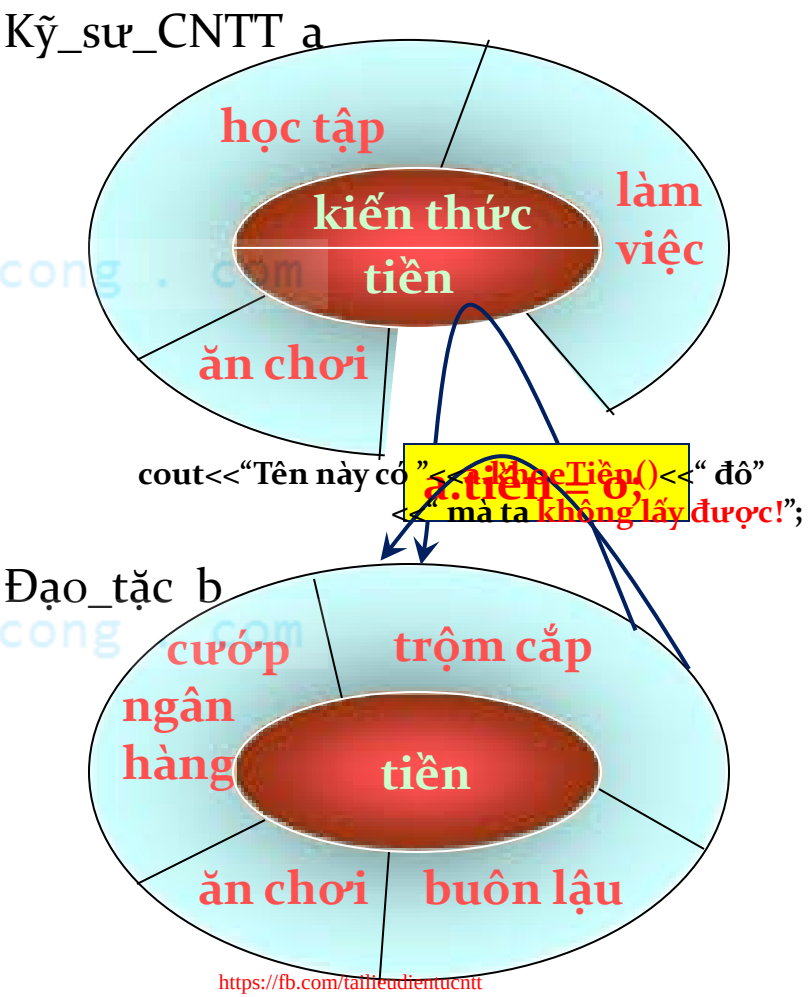
- Khi đã có các setter thì các constructor có tham số là dư thừa??? (Chỉ cần constructor mặc định.)
- Constructor có tham số không dư, vì "*Sinh con thì phải đặt tên!*"
  - Có những ***thuộc tính đặc trưng*** của đối tượng, bắt buộc phải có giá trị riêng ➔ Không thể cùng 1 giá trị mặc định. VD: Họ tên, MSSV, v.v.
  - Tạo thói quen an toàn cho người dùng: ***khởi tạo*** giá trị ngay khi tạo ra đối tượng.

Các loại PThức khởi tạo

- PT khởi tạo có tham số: Gán các tham số cho các thuộc tính tương ứng; Những thuộc tính còn lại đặt bằng giá trị **mặc định**.
  - Đầy đủ thông tin:
    VD:
    ```
    PhanSo::PhanSo(int tu, int mau)
    {this->tu = tu; this->mau = mau;}
    PhanSo haiPhanBa(2,3);
    ```
  - Thông tin cần thiết:
    VD:
    ```
    PhanSo::PhanSo(int n)
    {this->tu = n; this->mau = 1;}
    PhanSo bon(4);
    ```
  - Thông tin cơ bản (đặc trưng cho từng đối tượng và không thể thiếu): VD: `SinhVien a("001","An");`
    - ***Nếu có loại này thì không có pt khởi tạo mặc định!***

Quy tắc Đóng gói Kín

- Đóng gói hở: Mở một số thuộc tính ra "public" cho mọi người sử dụng trực tiếp → nguy hiểm!
- Đóng gói kín: Mọi thuộc tính đều "private", muốn đọc giá trị của nó cũng phải thông qua phương thức! → Đối tượng kiểm soát được mọi tác động đến thuộc tính của mình.



PThức Lấy/Đặt (Getter/Setter)

- Quy tắc đóng kín → mọi thuộc tính phải được...
  - Truy xuất thông qua getter: int ĐồngHồ::hiệnGiờ()
  - Truy nhập thông qua setter: void ĐồngHồ::đặtGiờ(int h)
- Các loại thuộc tính:
  - Thuộc tính nội bộ không có get/setter.
  - Thuộc tính chỉ đọc chỉ có getter
  - Thuộc tính ảo có getter (setter), nhưng không có thực trong bộ nhớ. Thường là công thức tương đương với những thuộc tính thực khác. VD: (giờ, phút, giây) ~ i_giây
  - Thuộc tính công cộng có cả getter và setter nhưng mọi truy cập đến thuộc tính đều được kiểm soát.
    VD: PhânSố::đặtMẫu(float ≠ 0)

Phương thức đặt (Setter)

- Là phương thức cho phép bên sử dụng đặt một giá trị mới cho thuộc tính
  - Giống như phép gán bằng: x.setA(3) ⇔ x.A = 3
  - Khác các phương thức thay đổi thuộc tính khác, như x.increaseA(3) ⇔ x.A += 3
  - Khác hàm tạo có tham số:
    - Constructor chỉ được gọi 1 lần
    - Setter có thể được gọi nhiều lần (đổi rồi đổi lại)

Phân tích các Thuộc tính

- Các thuộc tính của lớp SinhVien

Thuộc tính	Get	Set	Khởi tạo	PThức khác	Ghi chú
MSSV	x		Tham số	thiLT(), thiTH(), làmNhóm()	Định danh
Tên	x		Tham số	thiLT(), thiTH(), làmNhóm()	Định danh
MS Nhóm	x	x	'Z'	làmNhóm()	
điểm LT	(x)		-1	thiLT()	
điểm TH	(x)		-1	thiTH()	
điểm Cộng	(x)		0	làmNhóm()	
điểm TK	x				Xác định bởi đ. LT, đ. TH, đ. Cộng

Phân tích các Thuộc tính

- Các loại thuộc tính của một đối tượng nói chung

Thuộc tính	Get	Set	Khởi tạo	PThức khác	Ghi chú
Nội bộ			GT Mặc định	Các pthức sử dụng và **xử lý** thuộc tính này	
Chỉ đọc	x		Tham số / GT Mặc định	(Các pthức sử dụng và **xử lý** thuộc tính này)	
Chỉ ghi		x	Tham số / GT Mặc định	(Các pthức sử dụng và **xử lý** thuộc tính này)	
Công cộng	x	x	Tham số / GT Mặc định	(Các pthức sử dụng và **xử lý** thuộc tính này)	
Định danh	x		Tham số / Sinh tự động	Các pthức **sử dụng** thuộc tính này	
Ảo	x	(x)		(Các pthức **sử dụng** thuộc tính này)	Được xác định bởi các thuộc tính khác

BT1: Thiết kế giao diện Xe - Xăng

- Ta cần lập trình cho đối tượng "xe" như sau:
  - Mỗi chiếc xe có một **số xe** không đổi để xác định xe.
  - **Lượng xăng** trong bình xăng chỉ *tăng* khi ***đổ xăng*** và *giảm* khi xe ***chạy*** (không có thất thoát).
    - Đổ xăng không được vượt quá **dung tích bình xăng**.
    - Tương ứng với mỗi km xe chạy, xe tiêu thụ một lượng xăng bằng **độ hao xăng** (lít/km).
    - Dung tích bình xăng và độ hao xăng của mỗi chiếc xe *không thay đổi tuỳ tiện*, *có thể* được chỉ định (khác nhau) khi sản xuất.
    - Vì người dùng không thể biết độ hao xăng, nên phải dựa vào vạch *báo* ***sắp hết xăng*** trên đồng hồ của xe.
      - "Chưa sắp hết xăng" := "Còn chạy được ít nhất 5km"

Tư duy Hướng đối tượng

"Đối tượng tự quản lý dữ liệu của mình qua các phương thức"

Tư duy Hướng đối tượng

- Hàm xử lý dữ liệu → Đối tượng ***thực hiện hành động***
  - Hàm "tính độ dài chuỗi" (trong thư viện "string.h")

    Khai báo: `int strlen(char str[]);`

    Sử dụng:
    ```
    char s[100] = "abc";
    int l = strlen(s);
    ```
    Ý nghĩa: "*Tính độ dài của chuỗi **s**.*"

    Dữ liệu ***bị xử lý***
  - Phương thức "tính độ dài" của lớp "string"

    Khai báo:

    Sử dụng:
    ```
    string s("abc");
    int l =
    ```
    Ý nghĩa: "*Chuỗi **s** tính độ dài (của chính nó).*"

    Chủ thể của hành động ("khổ chủ")

```
class string
{...
  int length();
  ...
};
```

Tư duy Hướng đối tượng

- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diên := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO:

```
s.assign(t);
```

Chủ thể (S) thực hiện **Hành động (V)** **Khách thể (O)**

Sử dụng → **Giao diện** → **Cài đặt**

Tư duy Hướng đối tượng

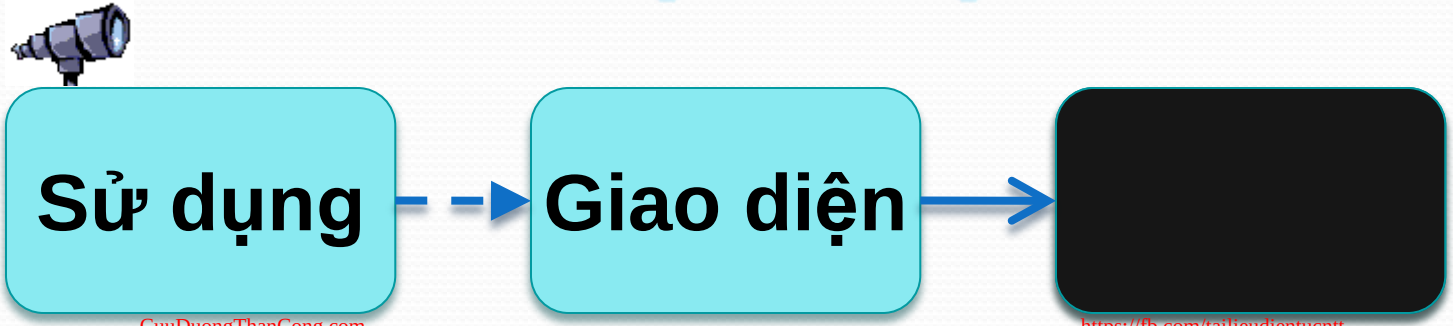
- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diên := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO:

```
s.assign(t);
```

Sử dụng phương thức assign() của s:
- *Không thấy cách xử lý của assign();*
- *Không thấy cấu trúc dữ liệu của đối tượng chủ thể s.*

Sử dụng đối tượng t:
- *Không thấy cấu trúc dữ liệu của đối tượng t.*

- Với người sử dụng đối tượng, *phần cài đặt là **hộp đen**!*



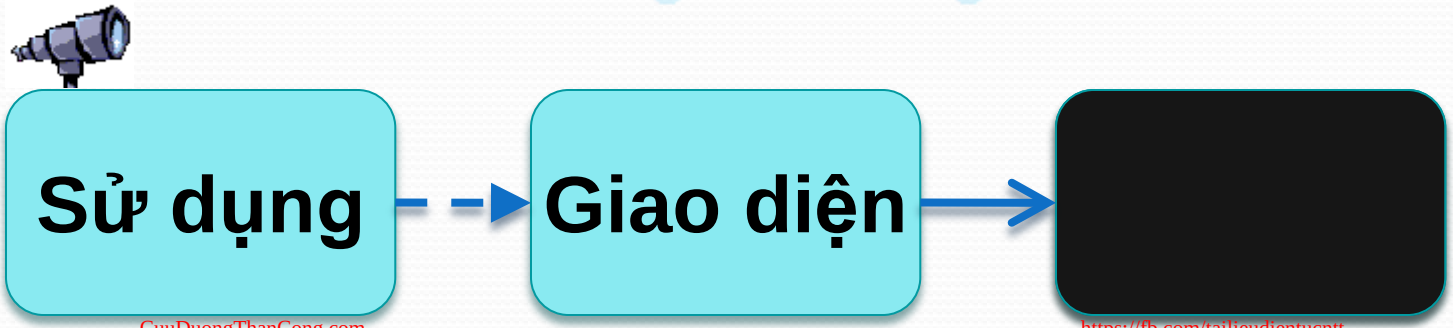
Tư duy Hướng đối tượng

- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diên := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO:

```
s.assign(t);
```

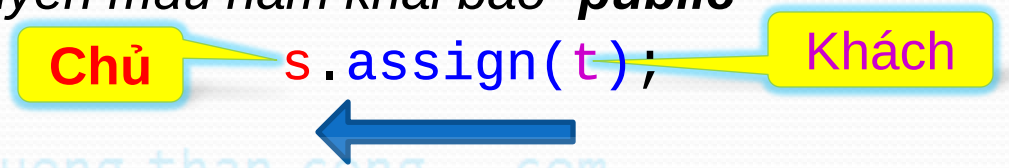
Chủ — Khách

- *Chỉ có 2 cách sử dụng đối tượng an toàn nhất là **gọi phương thức** và **truyền tham số**!*
  - Không truy cập trực tiếp đến các thuộc tính;
  - Không sử dụng toán tử, kể cả gán (=) và so sánh bằng (==)

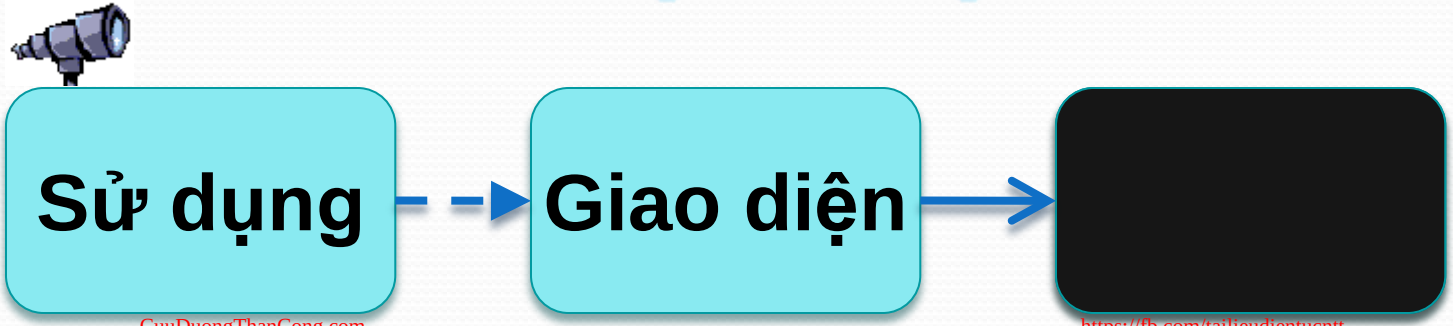


Tư duy Hướng đối tượng

- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diên := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO:



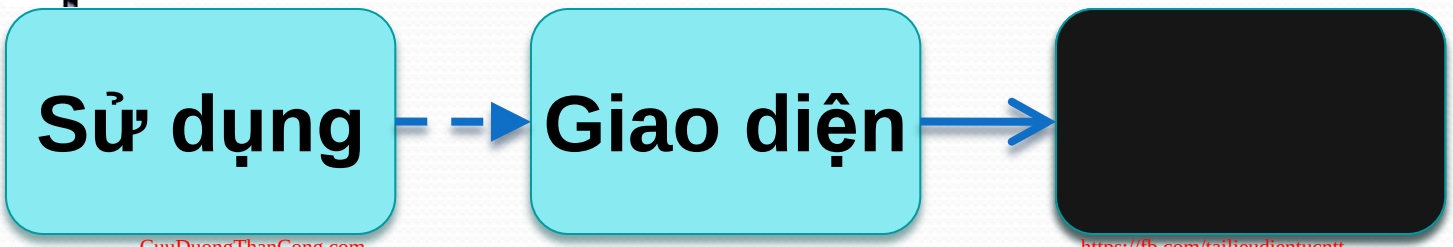
- *Đối tượng chịu trách nhiệm quản lý dữ liệu của mình & **không cho** người khác **can thiệp trực tiếp** vô!*
  - Mọi tác động đến đối tượng đều *thông qua phương thức* (gọi phương thức);
  - Thông thường, chiều cập nhật dữ liệu là: chủ ← khách



Tư duy Hướng đối tượng

- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diên := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO: `s.assign(t);` — **Chủ** — Khách

- *Đối tượng chịu trách nhiệm quản lý dữ liệu của mình, hạn chế cho bên ngoài **~~truy cập tới từng thuộc tính~~**!*
  - Hãy cung cấp *những **hành động liên quan** đến các thuộc tính* thay vì từng thuộc tính. VD: `PhanSo::nhan(PhanSo)`
  - Nếu cần, chỉ cho truy cập thuộc tính qua phương thức get/set.



Tư duy Hướng đối tượng (tóm)

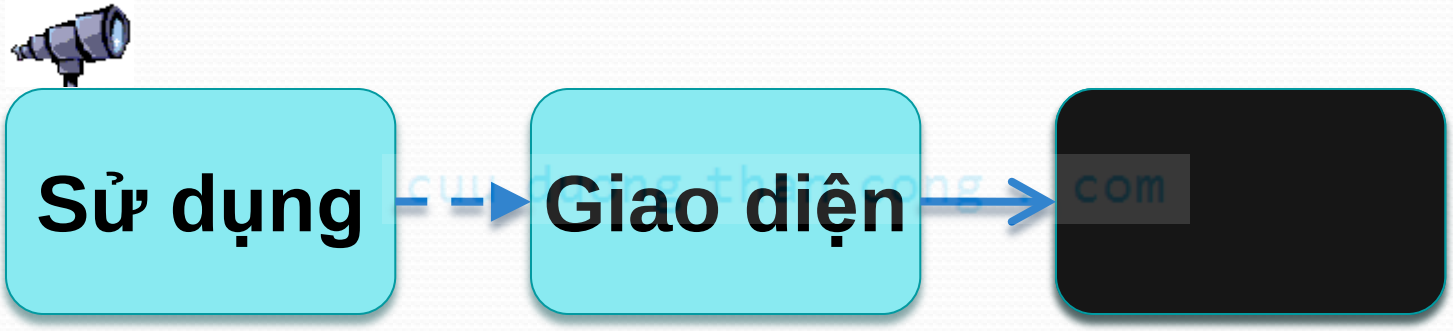
- Mọi tương tác với đối tượng đều thông qua **giao diện**:
  - *Giao diện := các nguyên mẫu hàm khai báo "**public**"*
  - Quy tắc vàng SVO:

```
s.assign(t);
```

Chủ — Khách

*"Chỉ có 2 cách sử dụng đối tượng an toàn nhất là **gọi phương thức** và **truyền tham số**!"*

- Với người sử dụng đối tượng, *phần cài đặt là **hộp đen**!*



BT1: Thiết kế giao diện Xe - Xăng

- Câu 1: Hãy khai báo lớp Xe như mô tả trên.
  - Gợi ý: Lập bảng phân tích thuộc tính trước.

- Câu 2: Hãy định nghĩa giao diện ***IXe*** tổng quát để thực hiện nghiệp vụ Điều phối xe của **Nhà Xe**:
  - Nhà Xe có một danh sách các Xe (qua giao diện IXe)
  - Điều phối xe có biển số X đi quãng đường d, biết rằng
    - Từ bất kỳ nơi đâu trong TP cũng tìm được cây xăng trong bán kính không quá 1km.
    - Chi phí đổ xăng luôn đảm bảo đủ cho cả quãng đường.
    - Phải điều phối cho xe đi hết quãng đường d.

BT2: Cài đặt lớp Cụ thể

- Cài đặt 2 lớp sau đều có giao diện IXe như đặc tả ở BT1, cộng thêm:
  - Xe Máy
    - Độ hao xăng = constant = 1lit/50km
    - Có 3 **loại**: Tay ga, Xe số, Tay côn
  - Xe Hơi
    - Có **số chỗ** ngồi, **số người** đã lên xe
    - 1/(Độ hao xăng) = 35*(số chỗ - số người + 1)/(số chỗ) km/lít
    - Có 2 **loại**: Số tự động (AT), Số sàn (MT)
    - Phương thức: khách lên xe, khách xuống xe
- Quản lý cả Xe Máy lẫn Xe Hơi bằng Nhà Xe.