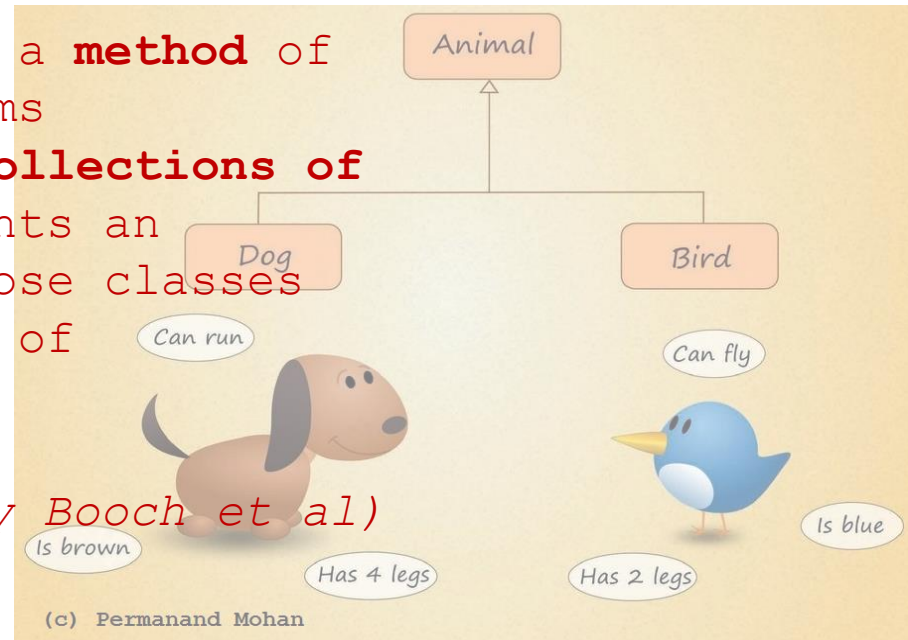


Object-oriented programming is a **method** of implementation in which programs are organized as **cooperative collections of objects**, each of which represents an **instance of some class**, and whose classes are all members of a hierarchy of classes united via inheritance relationships.

(Grady Booch et al)



Chương 1

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

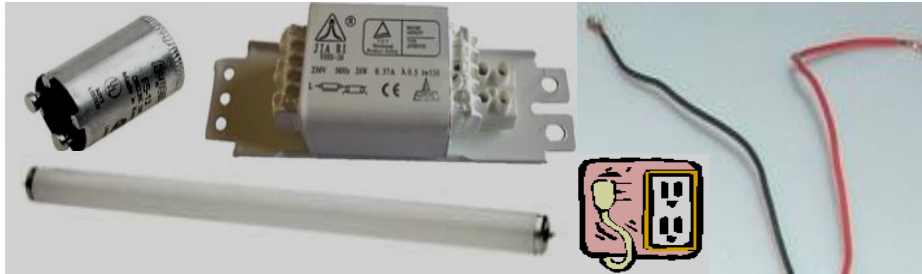
CT176 – LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Nội dung

- ◆ 1. Lịch sử của Ngôn ngữ lập trình.....●
- ◆ 2. Lập trình hướng đối tượng (OOP).....●
- ◆ 3. Các khái niệm quan trọng.....●
- ◆ 4. Các đặc điểm của OOP.....●

Mở đầu

- Lập trình Hướng đối tượng (Object-Oriented Programming)



**Cổ điện
Hướng thủ tục**



**Hướng đối
tượng**

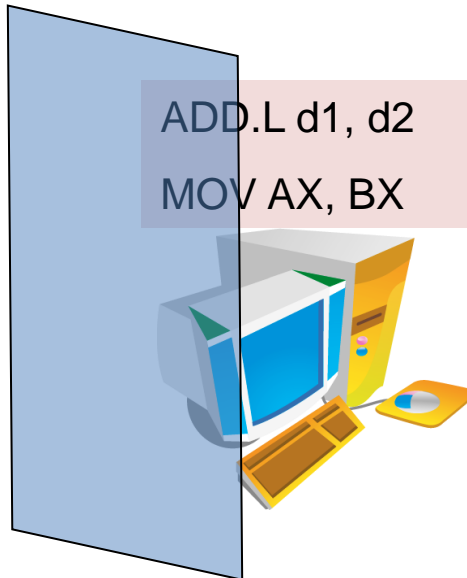
Máy tính & Ngôn ngữ lập trình

- **Máy tính:** Trung gian diễn đạt tư duy con người.
⇒ Kém giống máy và giống nhiều hơn tư duy của con người.
- **Ngôn ngữ lập trình:** Trừu tượng hóa (abstraction).

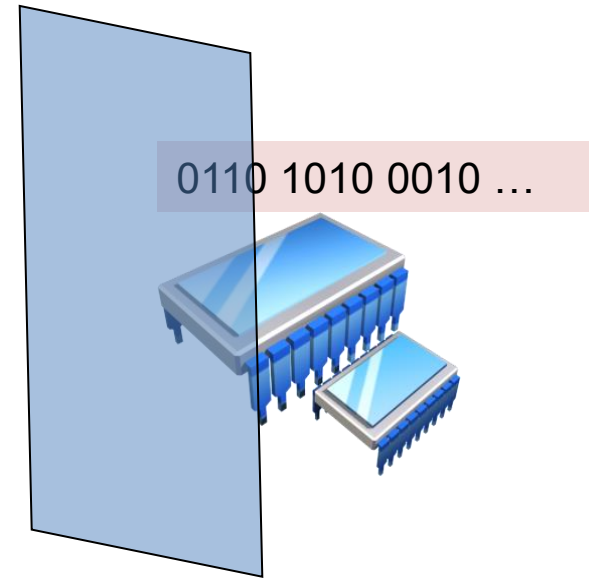
```
SORT(StudentList);  
WINDOW.SHOW();
```



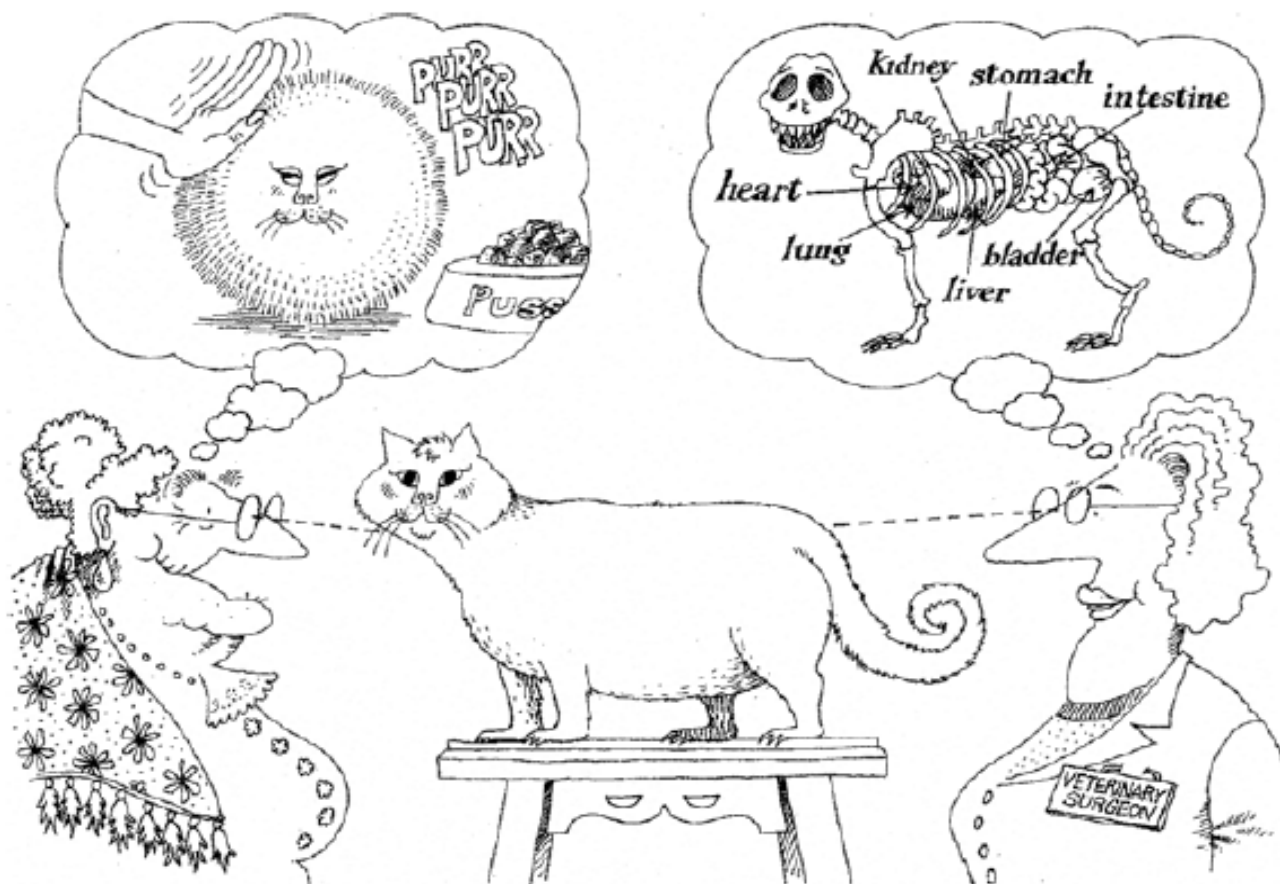
```
ADD.L d1, d2  
MOV AX, BX
```



```
0110 1010 0010 ...
```

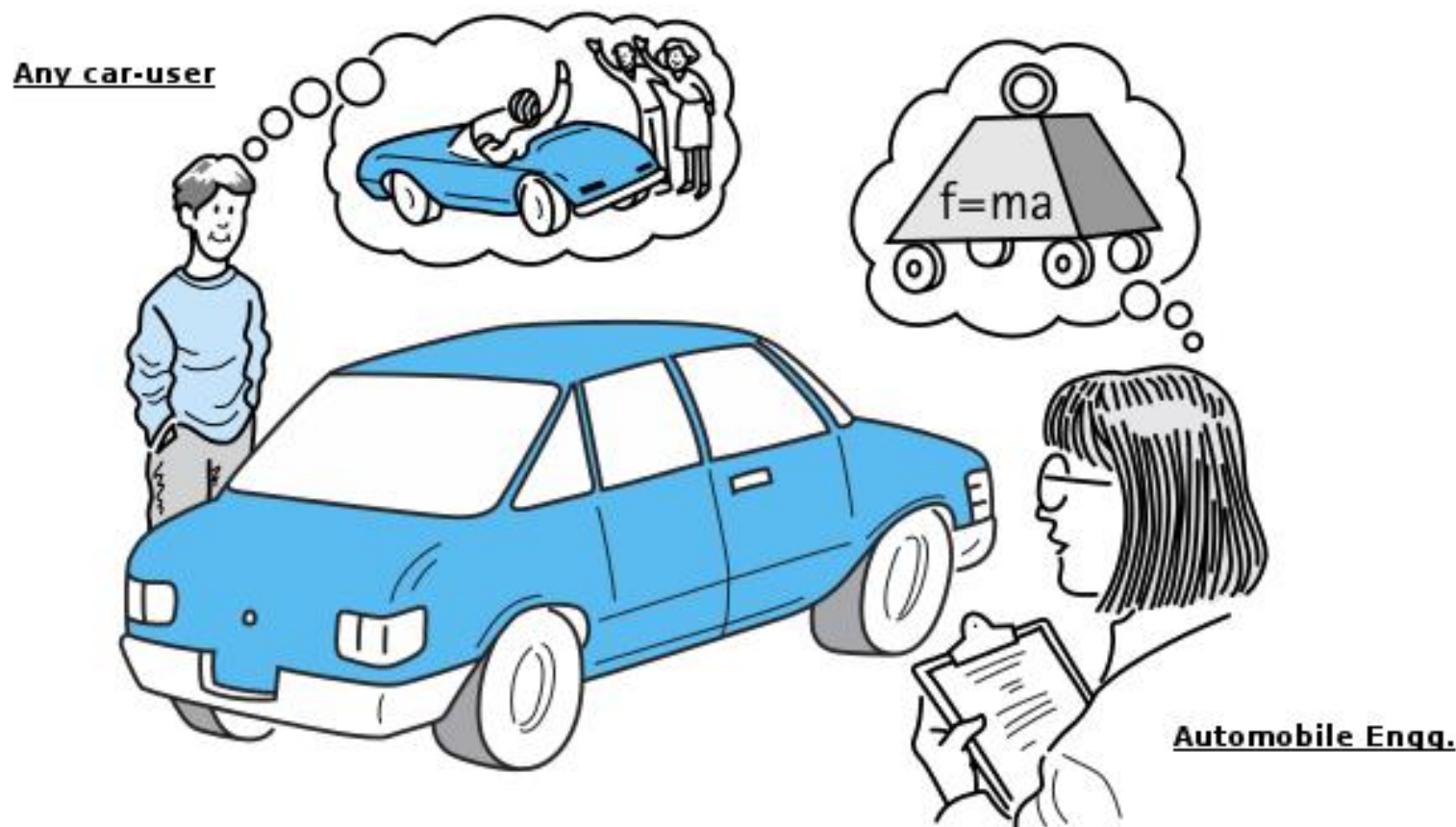


Trừu tượng hóa (Abstraction)



Abstraction focuses on the essential characteristics of some object, relative to the perspective of the viewer.

Trừu tượng hóa (Abstraction)



An abstraction includes the essential details relative to the perspective of the viewer

Ngôn ngữ máy & hợp ngữ

- Ngôn ngữ máy:

- Là các lệnh/chỉ thị của các bộ xử lý của máy tính.
- Là dãy các giá trị nhị phân 0, 1
- Không gần với ngôn ngữ của con người
⇒ khó hiểu, khó nhớ!

- Hợp ngữ:

- Trừu tượng hóa cho ngôn ngữ máy nền tảng.
- Các lệnh máy dưới dạng các dãy số 0, 1 được ký hiệu bằng các chỉ thị gần với ngôn ngữ con người.

001 110 001 011 010
001 010 001 011 010

subl %eax, %ecx
cmpl %eax, %edx
jl .L7
addl %ecx, %esi

Trong giai đoạn này, máy tính được sử dụng chủ yếu để tính toán.

Ngôn ngữ lập trình cấp cao

- Còn được gọi là ngôn ngữ ra lệnh:
 - Trừu tượng hóa cho hợp ngữ.
 - Vẫn đòi hỏi người lập trình suy nghĩ dưới dạng cấu trúc máy tính (do chưa đủ công cụ khái niệm để biểu diễn “thế giới thật” một cách gần gũi).
 - ⇒ Người lập trình phải thiết lập mối quan hệ giữa mô hình máy (trong không gian giải quyết vấn đề - máy tính) và mô hình của vấn đề (không gian của vấn đề - thế giới thật).

Trong giai đoạn này, máy tính bắt đầu được sử dụng để giải quyết nhiều vấn đề trong thế giới thật

Ngôn ngữ lập trình HDT

- Lập trình Hướng đối tượng:



- Cung cấp các công cụ (khái niệm) cho phép người lập trình mô hình hóa thế giới thật trong không gian giải quyết vấn đề một cách dễ dàng.



- Mô hình mà Lập trình hướng đối tượng chọn lựa là biểu diễn vấn đề trong không gian giải pháp như các “sự vật” hay “đối tượng” (object).



- Đây là một sự trừu tượng hóa mạnh mẽ và linh hoạt vì bản chất của thế giới là sự tương tác giữa các “sự vật”.



⇒ Nó cho phép mô tả vấn đề dưới dạng vấn đề, thay vì dưới dạng máy tính (nơi giải pháp sẽ chạy)

Ngôn ngữ lập trình HĐT

- Ý tưởng chủ đạo của OOP là các **sự vật**:
 - Chương trình là một tập các sự vật **tương tác lẫn nhau**.
 - Sự vật trong OOP là **sự tái hiện của các sự vật trong thế giới thật**: Mỗi sự vật có những đặc tính (properties/ characteristics) và khả năng (capacities) riêng.



Lịch sử của OOP

- OOP là phương pháp lập trình chính hiện nay:
 - Simula 1967, Smalltalk 1972
 - Giữa thập niên 90's, OOP mới bắt đầu được chú ý và sử dụng rộng rãi.
 - Hầu hết các ngôn ngữ lập trình hiện đại đều hướng đối tượng: C++, Java, .NET, ...

*(Dr. Alan Kay, cha đẻ của Smalltalk đã nhận được ACM **Turing Award** năm 2003)*

Lập trình cổ điển vs. OOP

- Lập trình cổ điển:

Chương trình

```
typedef struct {  
    char *mssv;  
    char *hoten;  
    float diemTB;  
    ...  
} Sinhvien;
```

```
Sinhvien *dssv1, *dssv2;
```

```
void sapxep(Sinhvien *) {...}  
void luu(Sinhvien *, char *) {...}  
void nhap(Sinhvien *) {...}
```

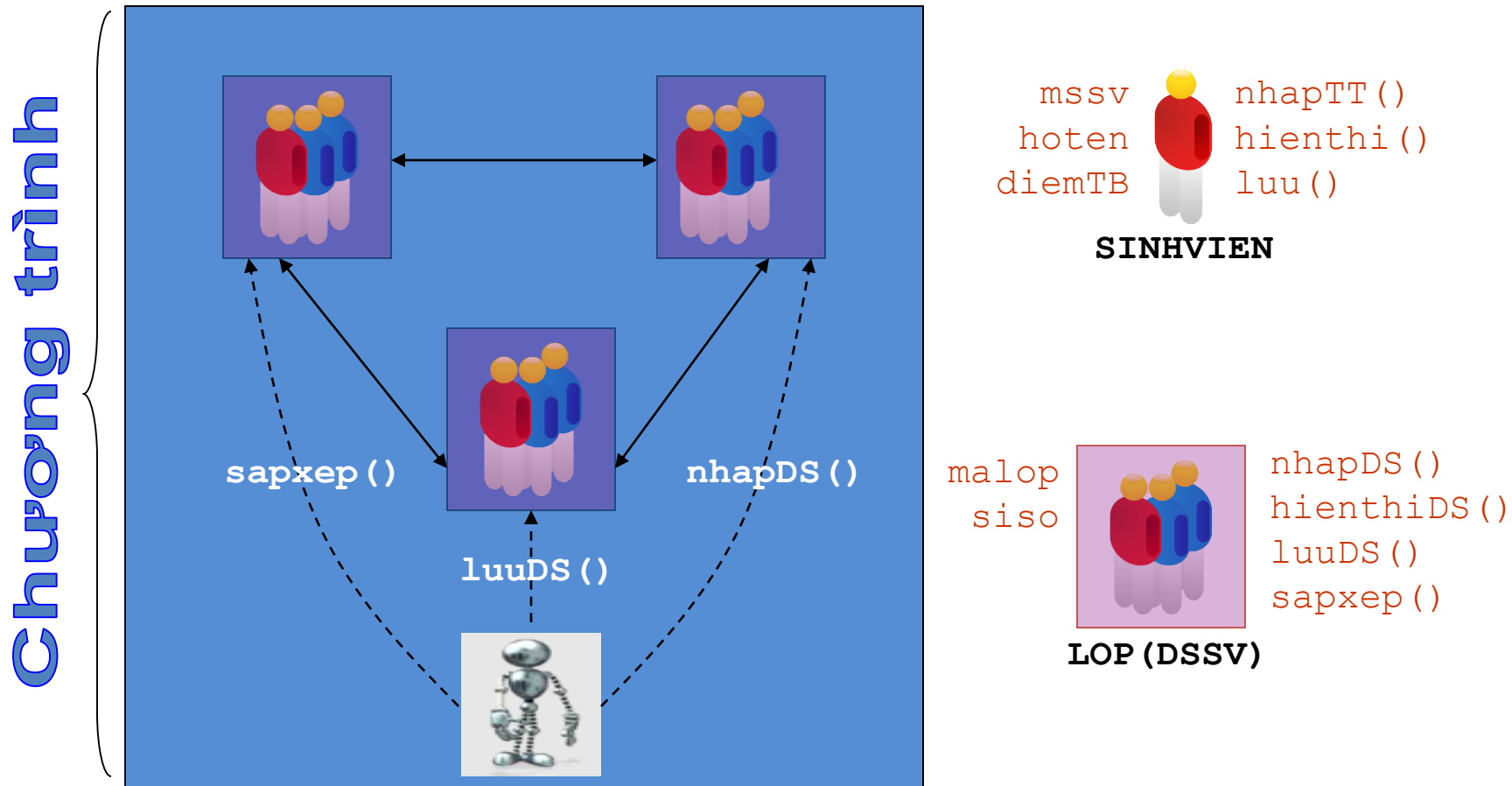
```
void main() {    nhap(dssv1);  
                sapxep(dssv2);...}
```

Cấu trúc dữ liệu

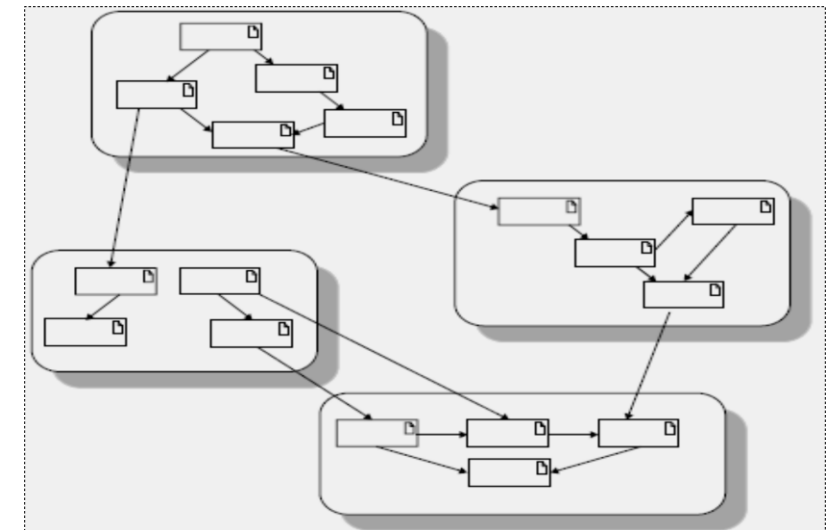
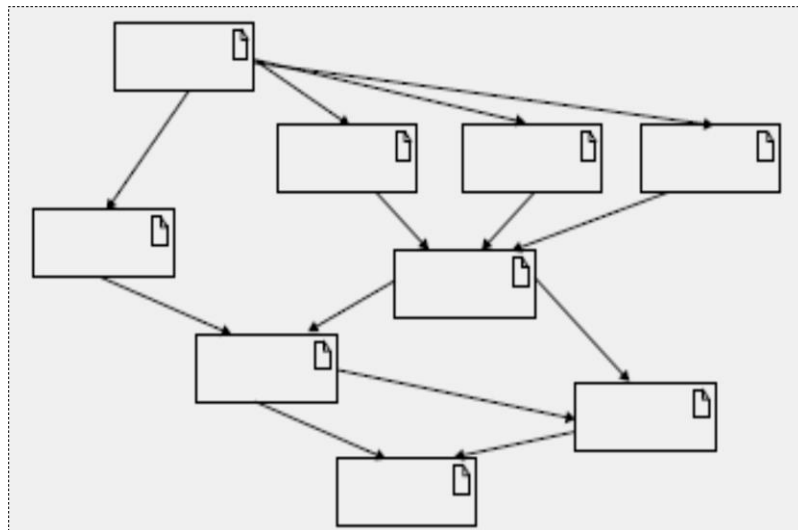
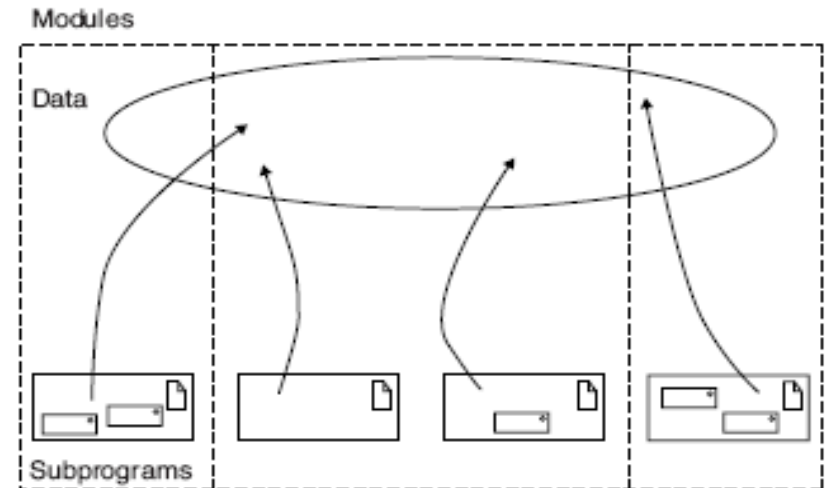
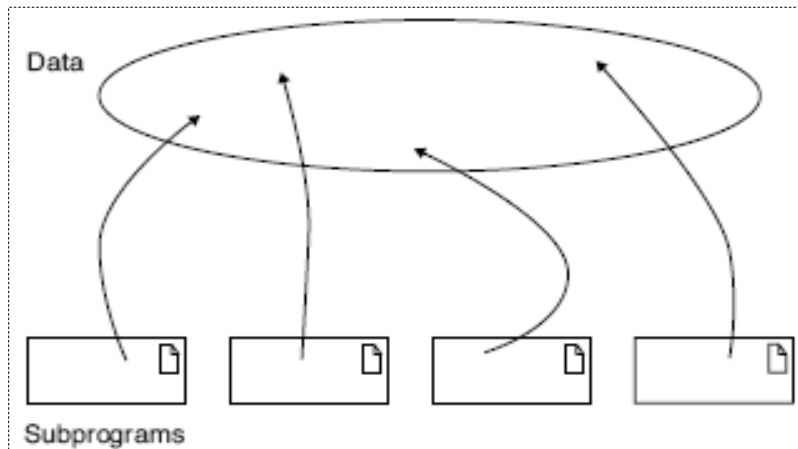
Giải thuật

Lập trình cổ điển vs. OOP

- Lập trình Hướng đối tượng:

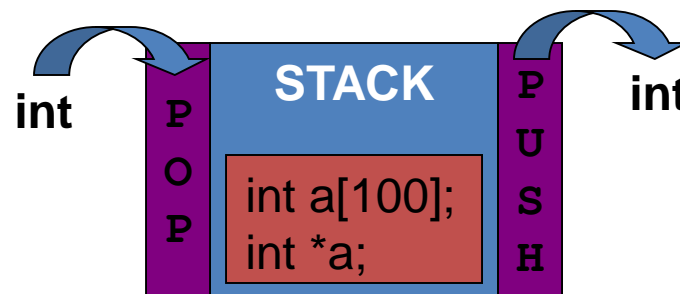


Lập trình cổ điển vs. OOP



Tại sao phải OOP?

- Liên kết chặt chẽ giữa dữ liệu và thao tác (hàm) của một đối tượng.
⇒ Cho phép ta **tập trung vào bản chất của vấn đề** hơn là các chi tiết bên trong vấn đề.
- Các dữ liệu và hàm được “bao gói” trong một đối tượng
⇒ Có thể **che dấu được những dữ liệu cần thiết**, hoặc chỉ cho phép truy xuất thông qua các hàm.



Đặc trưng của OOP

- Trong OOP, vấn đề (chương trình) được mô hình hóa như là **tập hợp của các đối tượng** hoạt động cộng tác với nhau:
 - Chương trình: tương tác của nhóm các đối tượng
 - Tương tác giữa các đối tượng: là sự hoạt động của từng đối riêng rẽ, thông qua việc gửi các thông điệp/yêu cầu cho nhau.

Đặc trưng của OOP

- Trong OOP, các đối tượng có thể được “nhân hóa”.

Ví dụ, một “cái cửa” có thể *tự* mở, một “menu” có thể *tự* “hiển thị”,...

- Tuy nhiên, các đối tượng phải được “yêu cầu” khi nào cần thực hiện thao tác gì bởi các đối tượng khác (thụ động)

⇒ Các đối tượng trong chương trình phải “hợp tác” với nhau để giải quyết một vấn đề.



Đặc trưng của OOP

1. Mọi thứ đều là **sự vật**.
2. Chương trình là **một nhóm các sự vật** “nói chuyện” với nhau bằng việc gửi các thông điệp cho nhau.
3. Mỗi sự vật đều có **bộ nhớ riêng**, được tạo nên từ các sự vật khác.
4. Mọi sự vật đều **có kiểu** (lớp).
5. Tất cả các sự vật cùng kiểu (lớp) đều có thể **nhận cùng thông báo**.

(Thinking in Java)

Đối tượng (object)

- Đây là khái niệm quan trọng nhất trong OOP: Trong OOP, **mọi thứ đều là đối tượng**.
- Là một thực thể được sử dụng bởi máy tính, là “**cái mà ứng dụng muốn đề cập đến**”.
- Mô tả cho **một sự vật hoặc khái niệm trong thực tế**.
Một đối tượng có thể là:
 - Một đối tượng thật (real object).
 - Một đối tượng khái niệm (conceptual object).
 - Một đối tượng phần mềm (software object).

Đối tượng

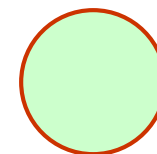
- Đối tượng thế giới thật: Là một đối tượng cụ thể mà ta có thể sờ, nhìn thấy hay cảm nhận được.
 - Ví dụ: cái đồng hồ, chiếc xe, con chó,...



- Đối tượng khái niệm: Đây thực sự là những khái niệm, những quá trình (process) trong thực tế được trừu tượng hóa thành các đối tượng.
 - Ví dụ: Ngày tháng (Date), chuỗi, hình tròn, ...

02-01-2008

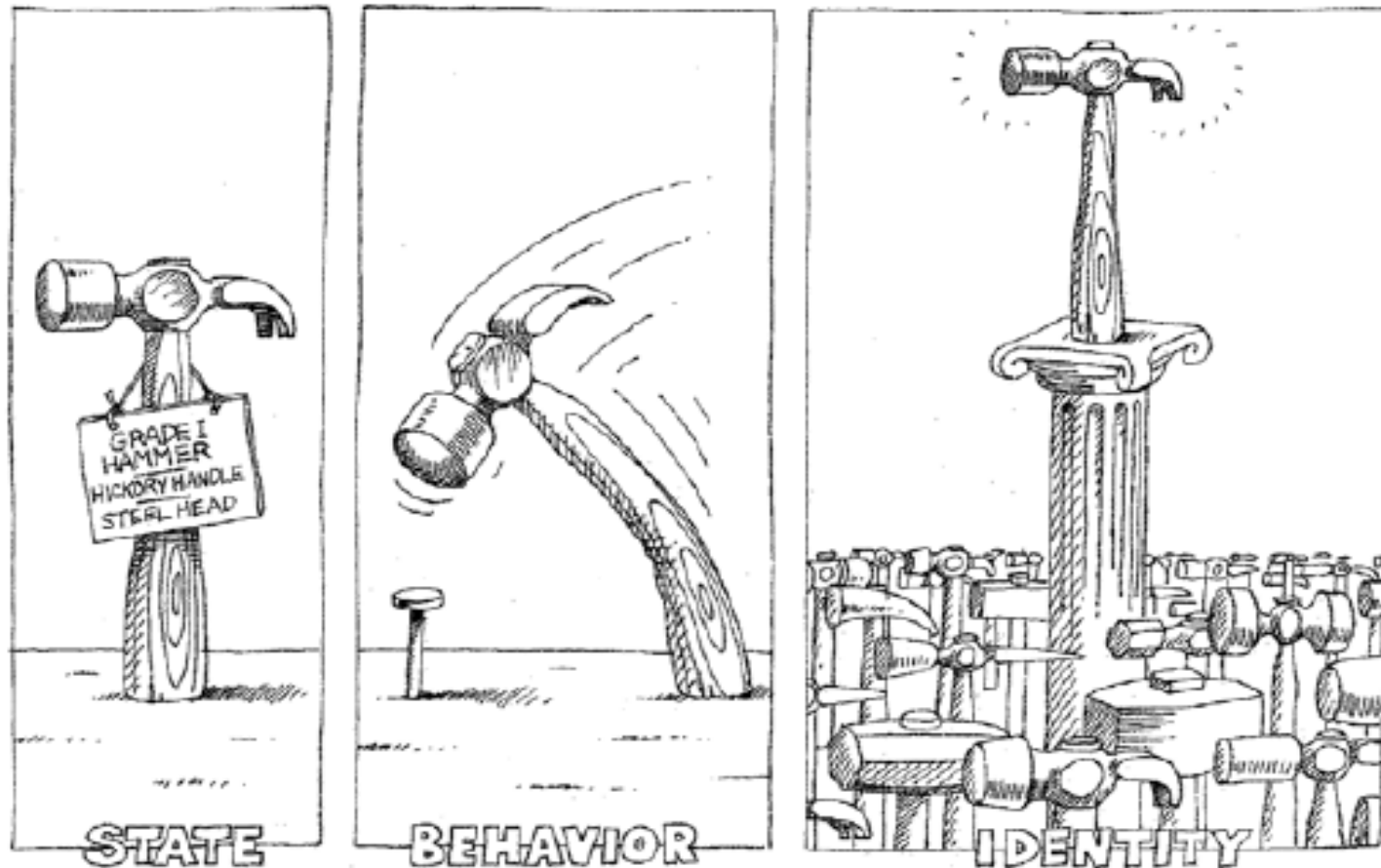
Welcome to OOP



Đối tượng (object)


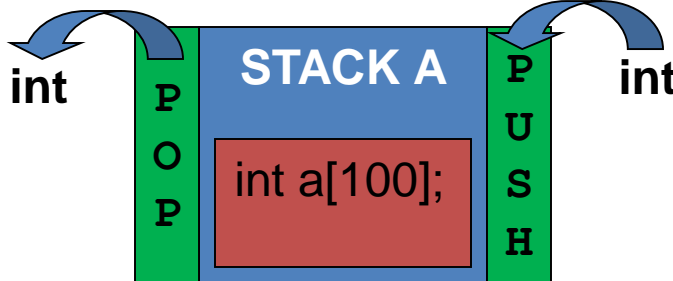

- Mỗi đối tượng có hai thành phần:
 - Thuộc tính (property, attribute): mô tả các đặc điểm, trạng thái của đối tượng.
 - Hành vi (behavior, method): mô tả các thao tác, các hoạt động mà đối tượng có thể thực hiện (thể hiện “khả năng”, “chức năng” của một đối tượng)
- Ngoài ra, một đối tượng còn có một **định danh** (object identifier) dùng để phân biệt giữa các đối tượng.

Đối tượng (object)



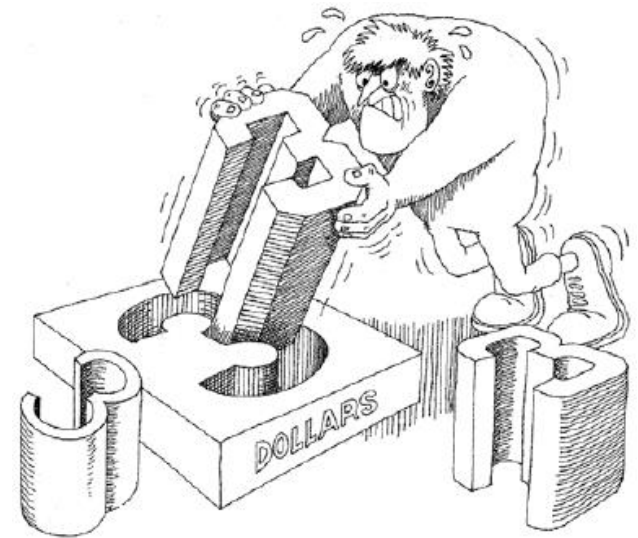
An object has state, exhibits some well-defined behavior, and has a unique identity.

Đối tượng (object)

Sự vật	Thuộc tính	Hành vi	Ví dụ			
Con chó	Tên: Mino Màu sắc: Xám Giống: Nhật Trạng thái: Vui vẻ	Sủa Ăn Chạy Cắn				
Stack A	List <table data-bbox="560 762 755 812"><tr><td>2</td><td>5</td><td>1</td></tr></table> Empty: NO.	2	5	1	PUSH POP	
2	5	1				
Bóng đèn	Nhãn hiệu: ABC Màu: Xanh Trạng thái: Mở Loại: Đèn bàn	Bật Tắt Sáng Mờ				

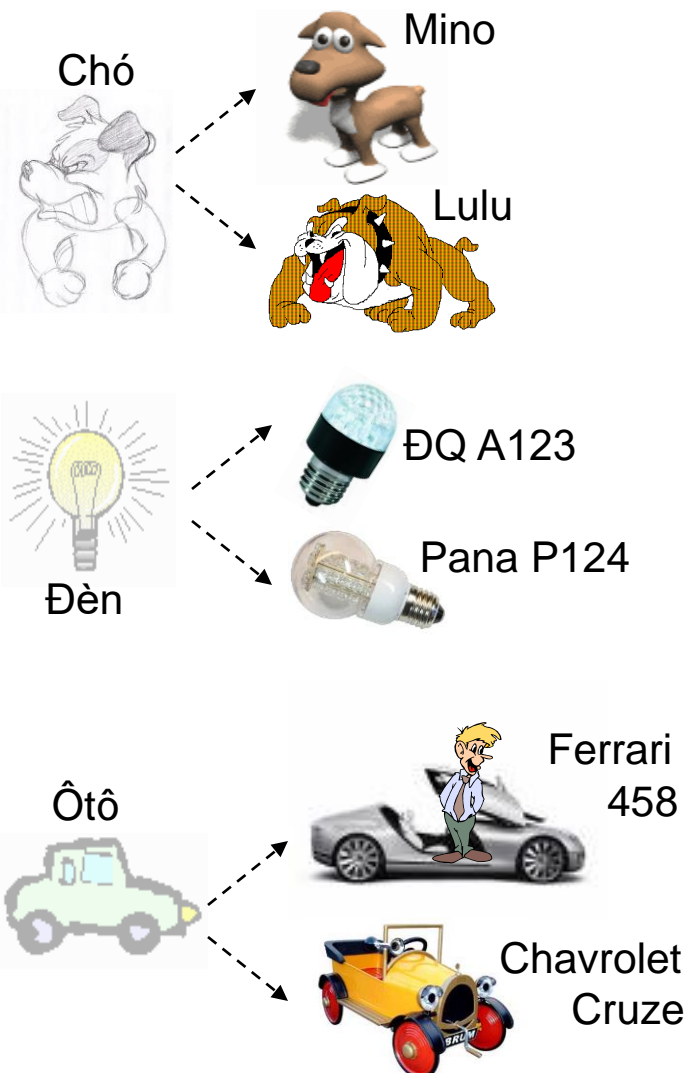
Lớp (class)

- Còn được gọi là **loại/kiểu** của đối tượng.
- Lớp là một **khuôn mẫu** để tạo ra các đối tượng cùng kiểu.
- Lớp định nghĩa các **thuộc tính** và **phương thức** (hành vi) chung cho tất cả các đối tượng cùng lớp.
- Một đối tượng được gọi là một **thể hiện** (instance) của một lớp và giá trị thuộc tính của chúng có thể khác nhau.

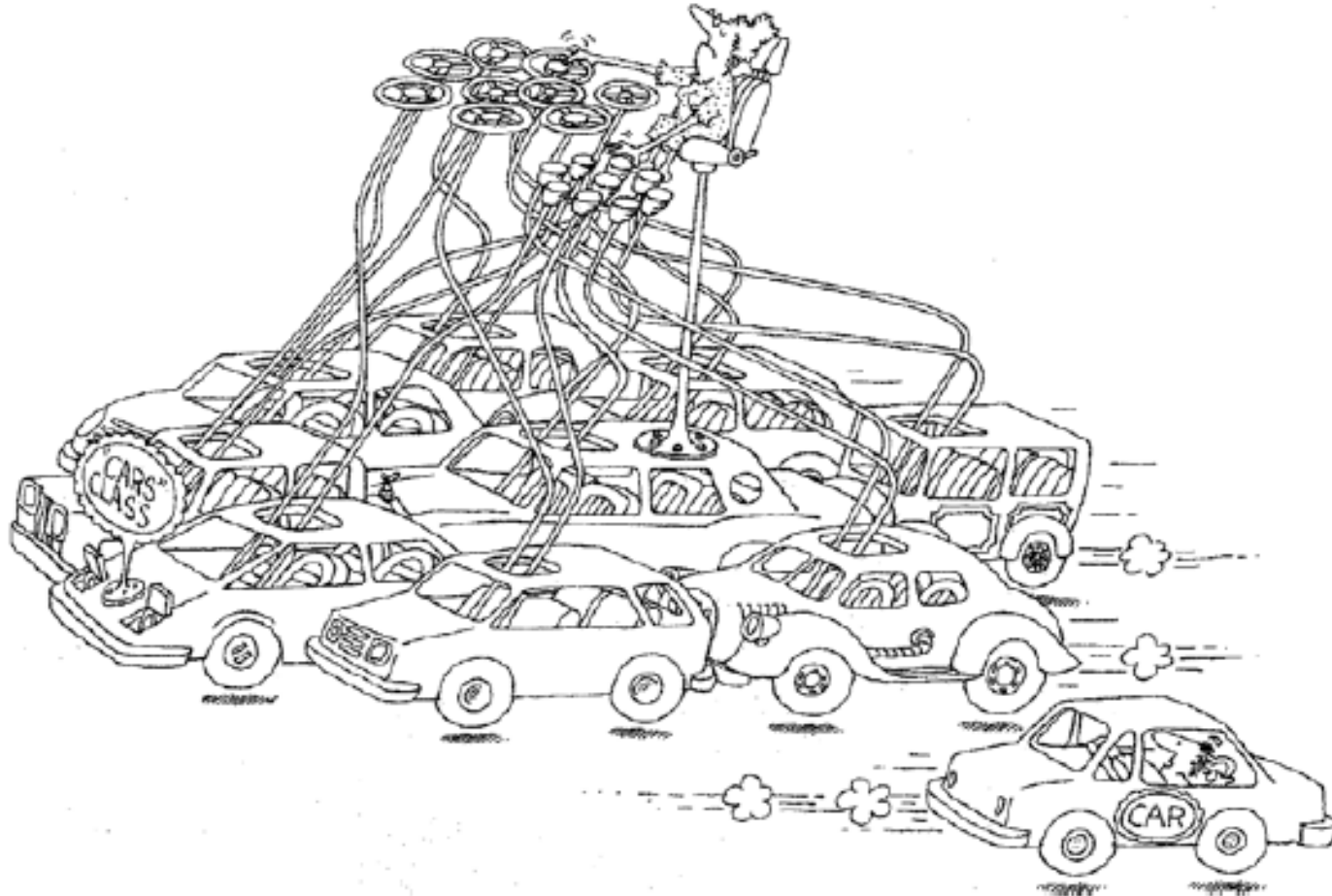


Lớp (class)

Lớp	Thuộc tính	Hành vi
Chó	Tên Màu sắc Giống Trạng thái	Sủa Ăn Chạy Cắn
Bóng đèn	Nhãn hiệu Màu Trạng thái Loại	Bật Tắt Sáng Mờ
Xe ô tô	Nhãn hiệu Màu sắc Giá Hộp số Tốc độ ...	Chạy Dừng Tăng tốc Giảm tốc ...



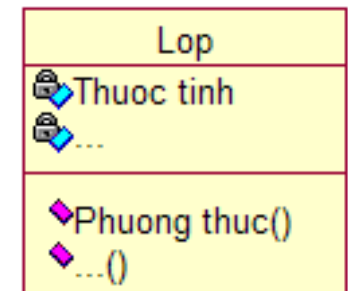
Lớp (class)



A class represents a set of objects that share a common structure and a common behavior.

Thuộc tính và phương thức

- Thuộc tính: Mô tả **trạng thái** của một đối tượng
⇒ Bao gồm: tên và kiểu
- Phương thức: Thể hiện cho **khả năng** của một đối tượng có thể **thực hiện được những hành vi gì?**
⇒ Bao gồm: tên, đối số và nội dung.



Lớp	Thuộc tính	Hành vi
Xe ô tô	Nhãn hiệu: <i>chuỗi</i> . Màu sắc: <i>chuỗi</i> . Giá: <i>số thực</i> . Hộp số: <i>số nguyên</i> . Tốc độ: <i>số nguyên</i>	Chạy {...} Dừng { <i>tốc độ</i> = 0; ...} Tăng tốc(km/h) { <i>tốc độ</i> += km/h; ...} Giảm tốc(km/h) { <i>tốc độ</i> -= km/h; ...} ...



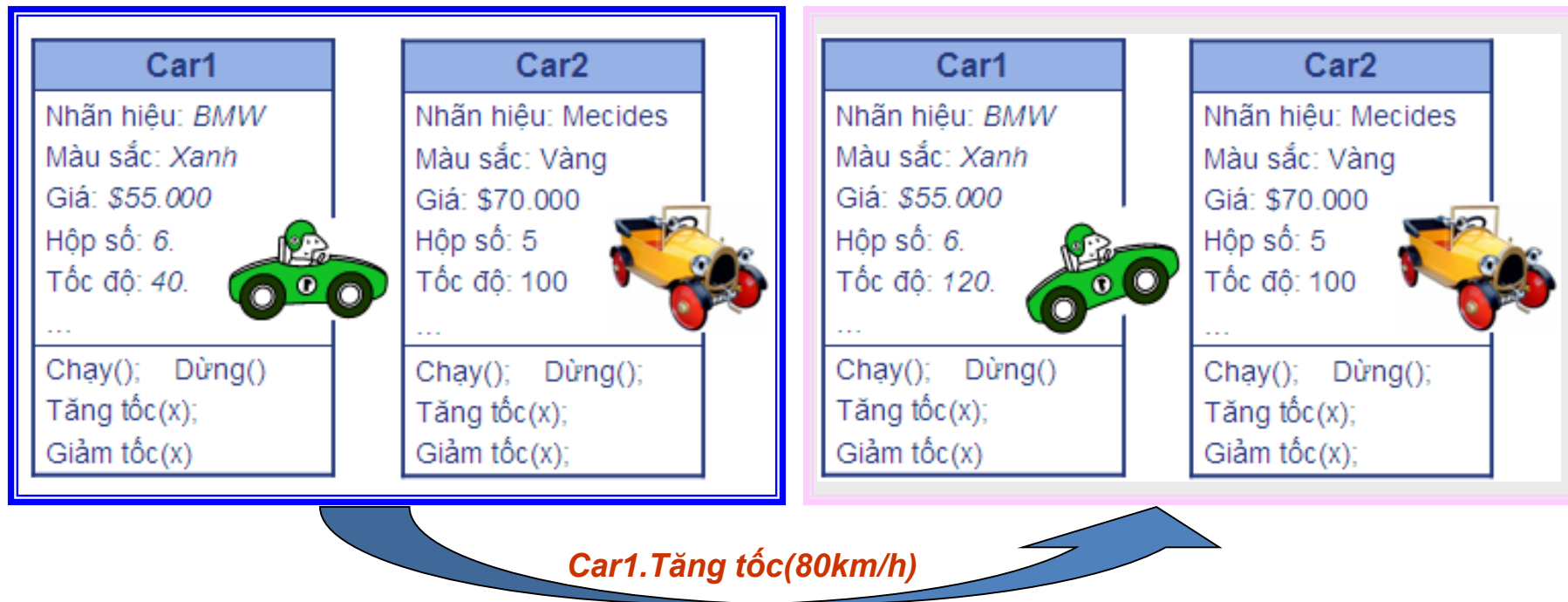
Hàm và việc truyền thông điệp

- Trong một chương trình OOP, các đối tượng hoạt động **cộng tác** với nhau thông qua việc **truyền thông điệp** cho nhau.
- Thông điệp (message): là một **yêu cầu** thực hiện một thao tác, hoạt động. Gồm có:
 - Tên thông điệp (tên của phương thức).
 - Các tham số của thông điệp (tham số của phương thức)
- Truyền thông điệp: gửi thông điệp đến đối tượng được yêu cầu. Bao gồm:
 - Đối tượng cần nhận thông điệp.
 - Thông điệp



Hàm và việc truyền thông điệp

- Việc thực hiện phương thức của một đối tượng **chỉ ảnh hưởng đến dữ liệu của chính đối tượng đó** chứ không ảnh hưởng đến các đối tượng khác trong cùng lớp



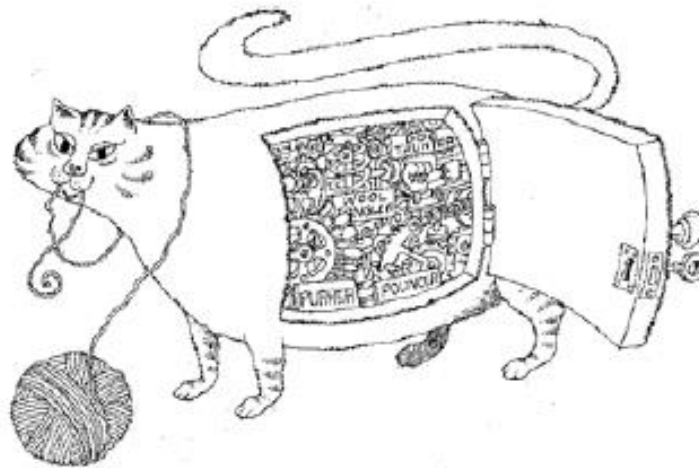
Các đặc điểm của OOP

- Tính bao gói (encapsulation): Là một trong những đặc điểm quan trọng của OOP.
- Thể hiện ở:
 - Sự **kết hợp chặt chẽ giữa dữ liệu và thao tác** của cùng một đối tượng \Rightarrow Chi tiết của một đối tượng được che dấu đi, giảm sự phức tạp.
 - Việc **nhóm chung dữ liệu và phương thức** của một đối tượng vào chung một nhóm.
 - **Xác định và giới hạn đường truy cập** đến các thành phần của một đối tượng.

Tính bao gói

- Thuộc tính truy cập (access modifier):

- **Public (dùng chung)**: các thành phần có thể được truy cập từ “bên ngoài”.
- **Private (dùng riêng)**: những thành phần chỉ được truy xuất “bên trong” lớp.
- **Protected (được bảo vệ)**: những thành phần chỉ được truy xuất từ bên trong lớp hoặc các lớp con (thừa kế) của nó.



Encapsulation hides the details of the implementation of an object.

<u>Class</u>	
◆	publicAttribute
◆	protectedAttribute
◆	privateAttribute
<hr/>	
◆	publicMethod()
◆	protectedMethod()
◆	privateMethod()

Tính bao gói

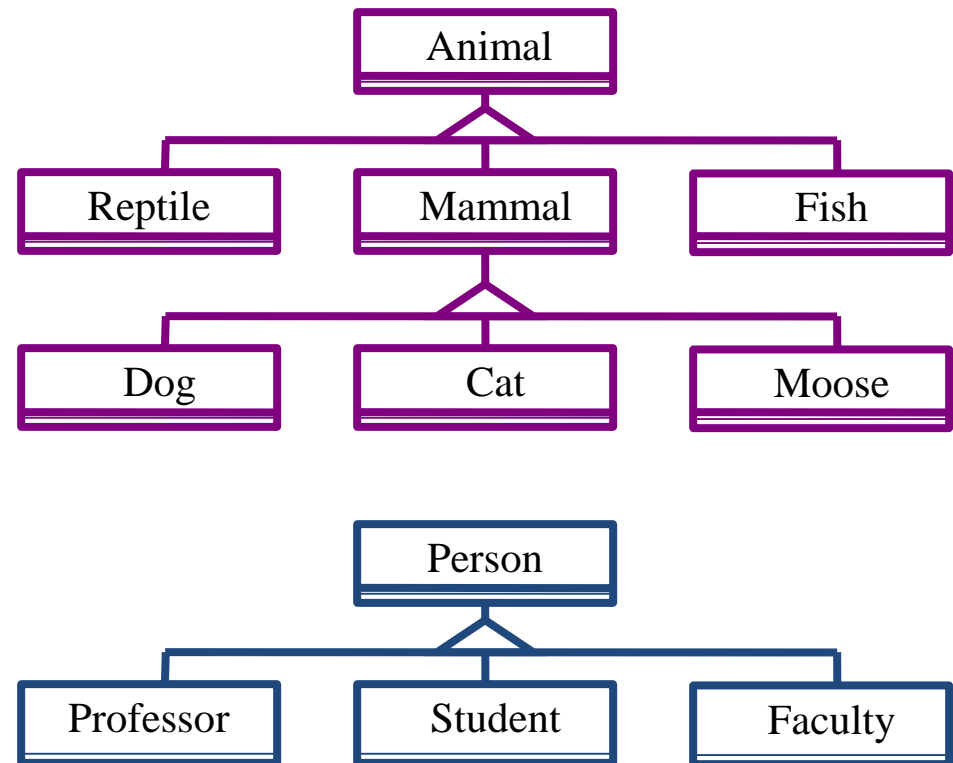
- Tính bao gói và sự trừu tượng hóa có mối liên hệ mật thiết, hỗ trợ lẫn nhau:
 - Trừu tượng hóa tập trung vào những đặc điểm thuộc về bản chất của đối tượng trong khi sự bao gói tập trung vào sự cài đặt những tính chất đó.
 - Sự bao gói thường đạt được nhờ sự che dấu thông tin (information hiding), che đi các chi tiết không cần thiết hoặc không phải là các thuộc tính thuộc về bản chất của đối tượng.

Tính thừa kế (inheritance)

- Là một trong những đặc điểm quan trọng của OOP, **cho phép dùng lại mã** (reusability).
- Dùng để mô hình hóa mối quan hệ “là” (“is a”) giữa các đối lớp/đối tượng với nhau:
 - Đối tượng “thừa kế” là một đối tượng đã có sẵn khác, với những thuộc tính và phương thức “tương tự” nhau.
 - Thừa kế sử dụng “sự tương tự” (similarities) và “sự khác nhau” (differences) để mô hình một nhóm các đối tượng có liên quan với nhau.

Tính thừa kế (inheritance)

- Lớp được thừa kế: **Lớp cha** (superclass), lớp cơ sở (based class).
- Lớp thừa kế: **lớp con** (subclass), lớp dẫn xuất (derived class).
- **EX:**
 - Loài bò sát, loài động vật có vú, loài cá đều **là** động vật (animal)
 - Thầy giáo, sinh viên, cán bộ đều **là** người (person).



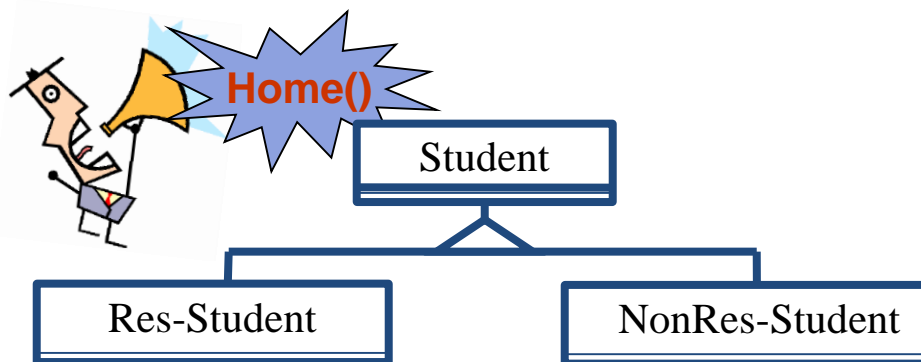
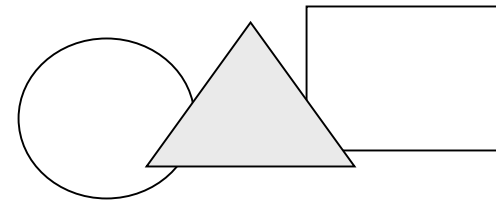
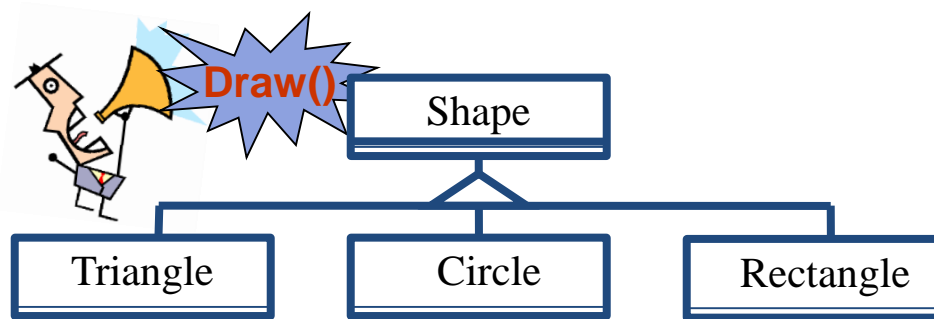
Tính thừa kế (inheritance)

- Trong thừa kế, lớp con “là” lớp cha \Rightarrow lớp con có tất cả thuộc tính và phương thức của lớp cha (“thừa kế” từ lớp cha!). Tuy nhiên, nó chỉ được truy xuất vào các thành phần nào mà lớp cha cho phép.
- Một lớp con được tạo ra bằng cách:
 - Thêm vào một số thuộc tính, phương thức mới
 - Tái định nghĩa các phương thức của lớp cha.
- Phân loại: thừa kế đơn, thừa kế bội (đa thừa kế).

Chú ý: Cần phân biệt giữa quan hệ “is a” và quan hệ “part of” giữa các đối tượng!

Tính đa hình (Polymorphism)

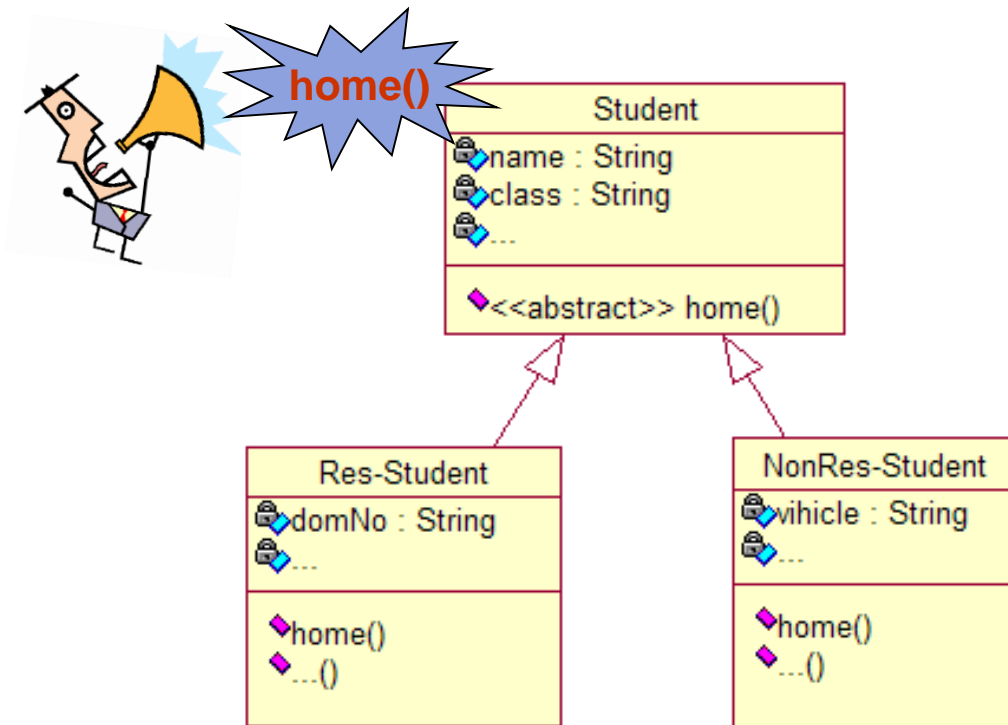
- Các loại đối tượng khác nhau có thể có cách **ứng xử khác nhau** cho cùng một thông điệp.



Tính đa hình (Polymorphism)

- Các kỹ thuật thể hiện tính đa hình:
 - Dùng hàm ảo (virtual function) kết hợp với chồng hàm / nạp đè (overriding): Khai báo phương thức ảo trong lớp cha chung. Mỗi lớp con sẽ cài đặt theo những cách khác nhau.
 - Dùng tái định nghĩa hàm (overloading): Định nghĩa các hàm trùng tên với nhau. Các hàm trùng tên phải khác nhau về:
 - Số đối số, hoặc
 - Thứ tự các đối số, hoặc
 - Kiểu của các đối số.

Tính đa hình (Polymorphism)





Question?

Phụ lục – UML

- UML: Unified Modeling Language.
- Là một ngôn ngữ dùng để **mô hình hóa** các hệ thống thông tin theo Hướng đối tượng.
- Bao gồm một hệ thống các **ký hiệu** (symbol), **sơ đồ** (diagram).
- Được sử dụng trong nhiều giai đoạn của quy trình phát triển phần mềm.
- Một số công cụ: Rational Rose, Visio, StartUML, ArgoUML, ...

Phụ lục – UML

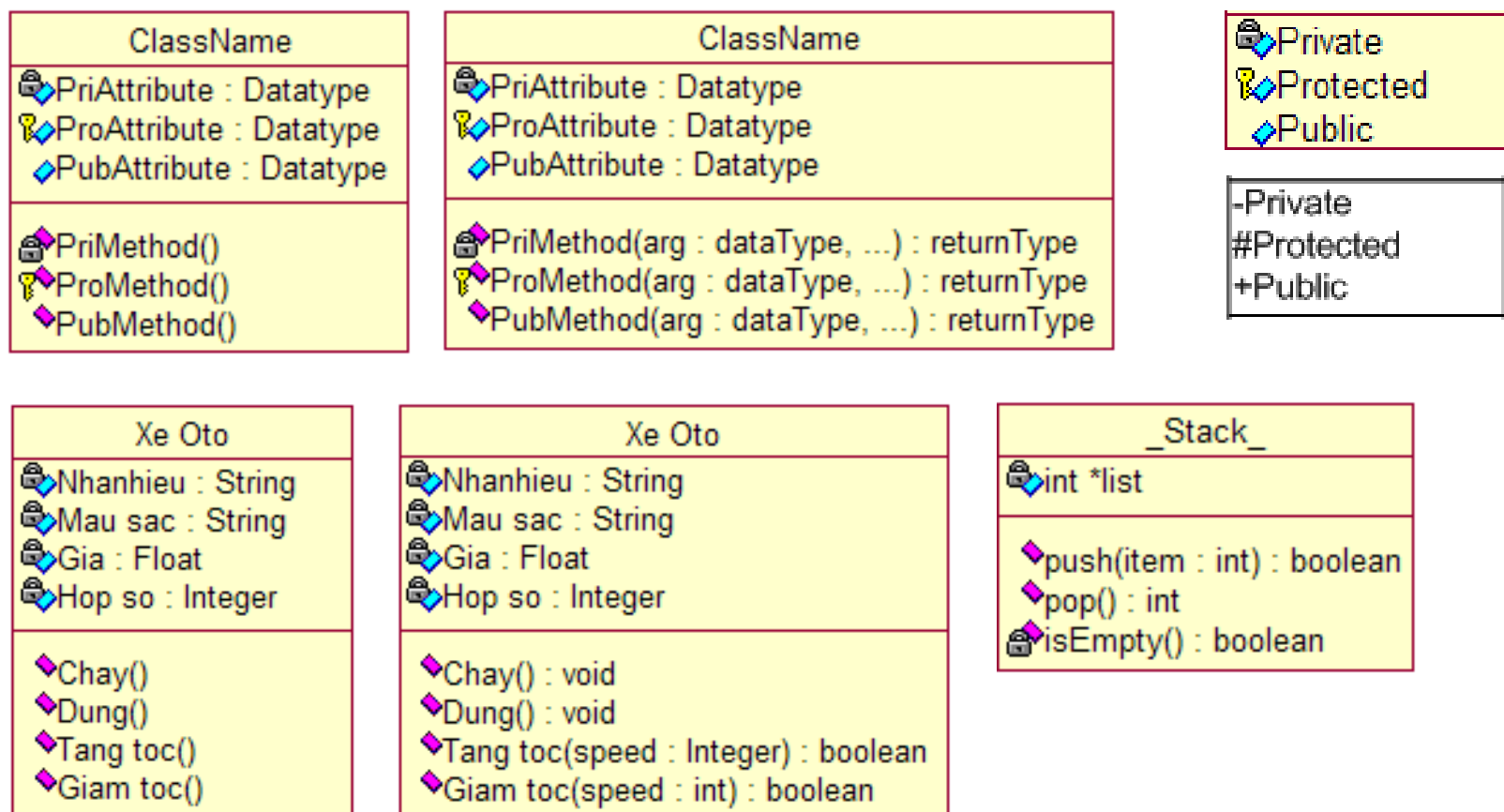
- UML bao gồm 9 loại sơ đồ:
 1. Use case diagram: Sơ đồ use case.
 2. **Class diagram: Sơ đồ lớp.**
 3. Object diagram: Sơ đồ đối tượng.
 4. Sequence diagram: Sơ đồ trình tự.
 5. State diagram: Sơ đồ trạng thái.
 6. Collaboration diagram: Sơ đồ cộng tác.
 7. Component diagram: Sơ đồ thành phần.
 8. Deployment diagram: Sơ đồ triển khai.
 9. Activity diagram: Sơ đồ hoạt động.

Phụ lục – UML

- Class diagram (sơ đồ lớp):
 - Mô tả cấu trúc của hệ thống theo hướng đối tượng.
 - Cấu trúc của một hệ thống được xây dựng từ các lớp và mối quan hệ giữa chúng.
- Ký hiệu (notation):
 - Lớp.
 - Giao diện.
 - Quan hệ:
 - Kết hợp (Association), tập hợp (Aggregation) và tổng hợp (Composition).
 - Hiện thực hóa (Realization).
 - Thừa kế/Tổng quát hóa (Generalization).

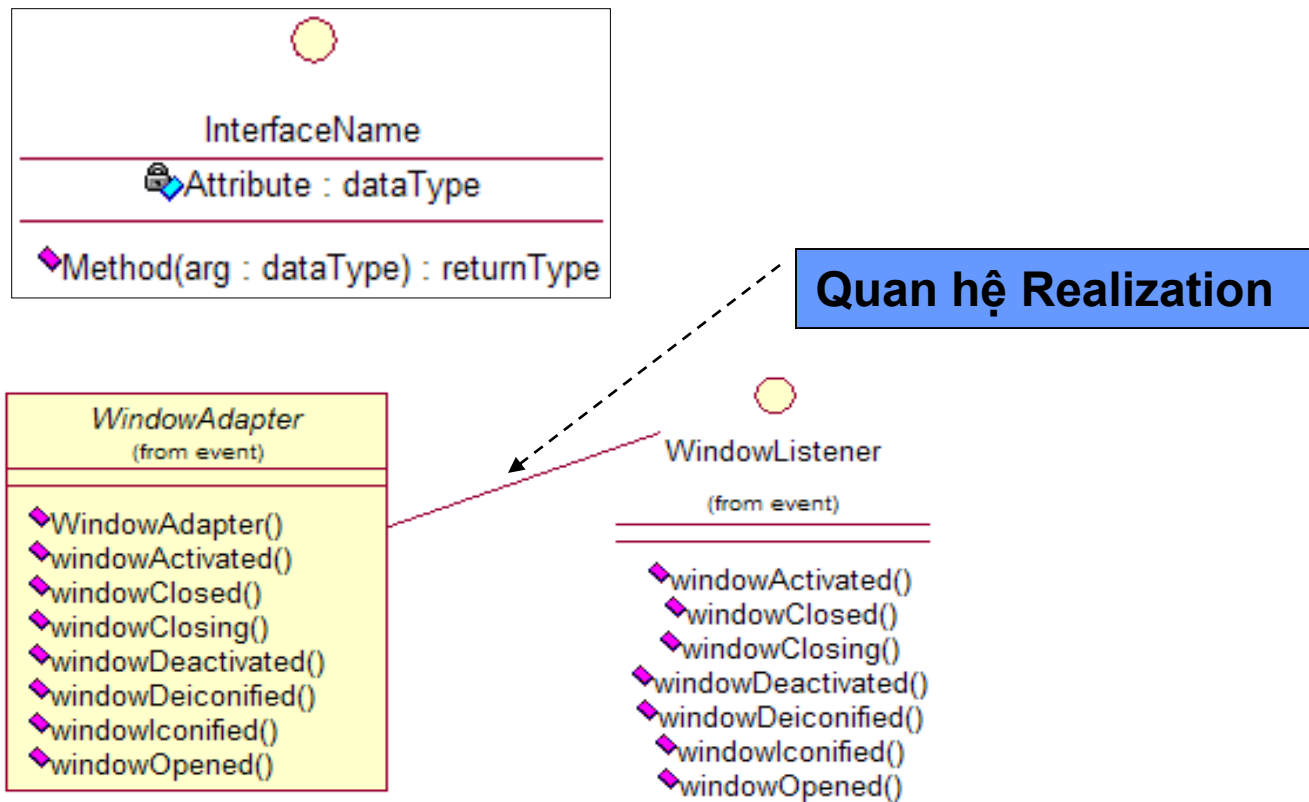
Phụ lục – UML

• Lớp:



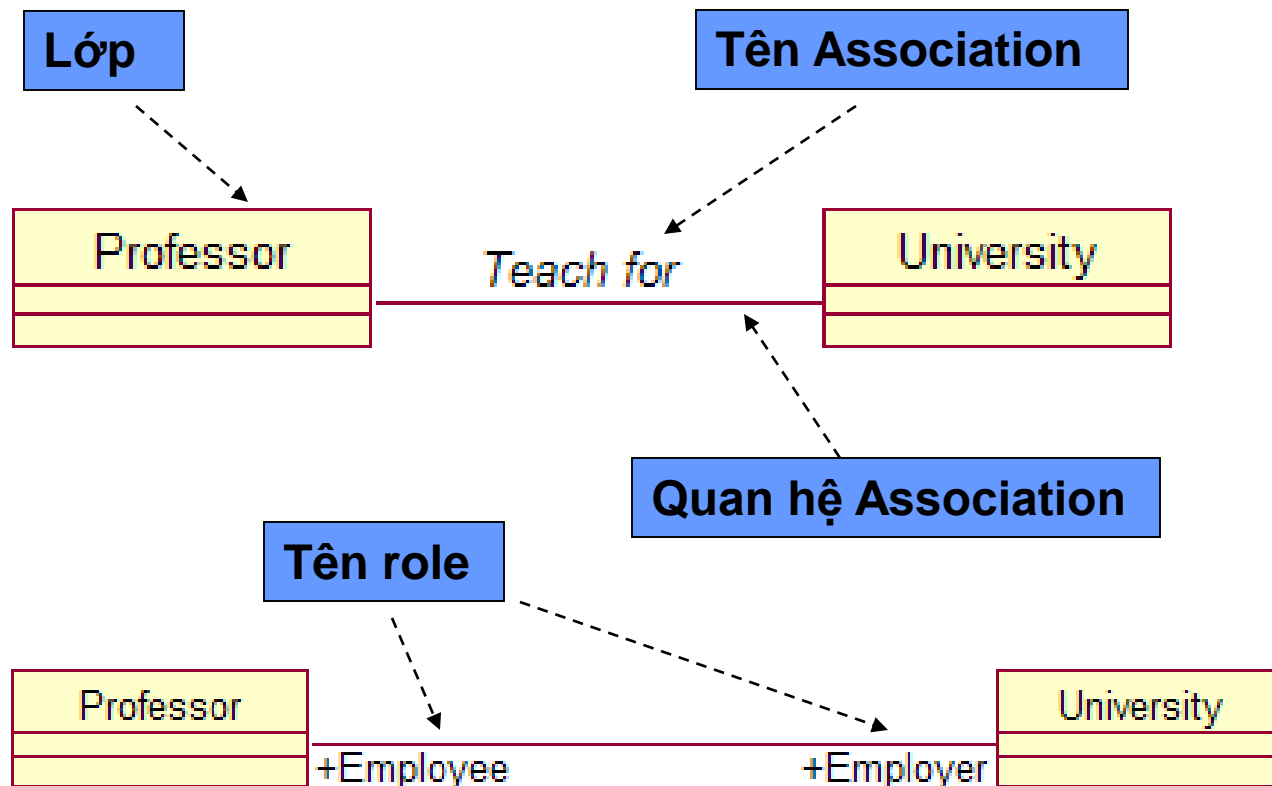
Phụ lục – UML

- Giao diện và quan hệ Realization: Quan hệ realization là quan hệ giữa giao diện và lớp cài đặt nó.



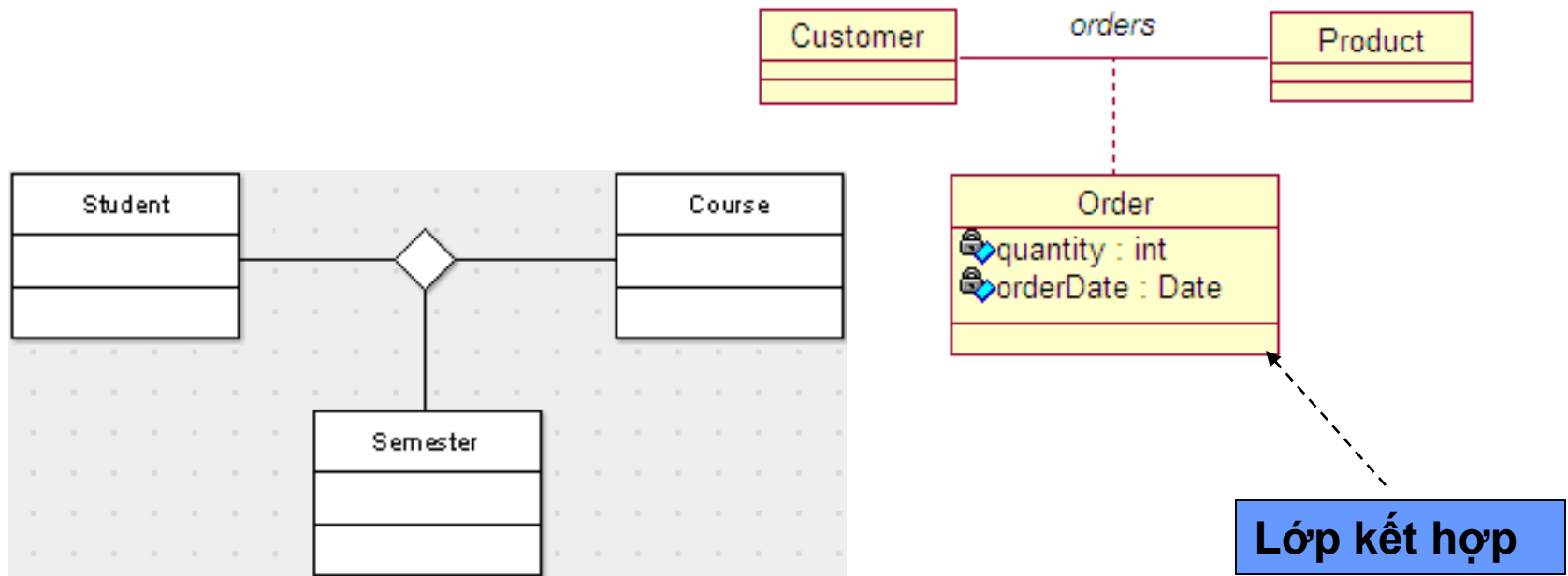
Phụ lục – UML

- Quan hệ kết hợp (Association): Mô tả mối liên hệ ngữ nghĩa giữa các lớp.



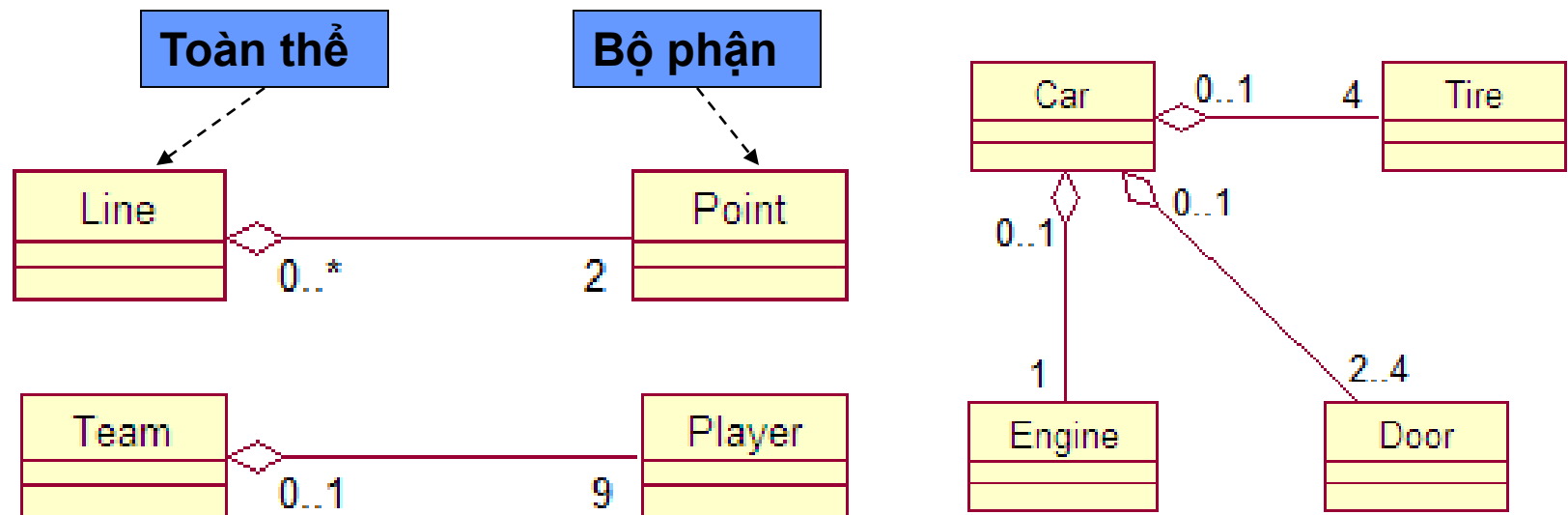
Phụ lục – UML

- Quan hệ kết hợp nhiều chiều (n-ary)
- Lớp kết hợp (association class): Được xem là thuộc tính của một quan hệ kết hợp.



Phụ lục – UML

- Quan hệ tập hợp (Aggregation): Là một dạng đặc biệt của quan hệ kết hợp, mô tả mối quan hệ toàn thể-bộ phận giữa một thực thể và các bộ phận của một thực thể.



Phụ lục – UML

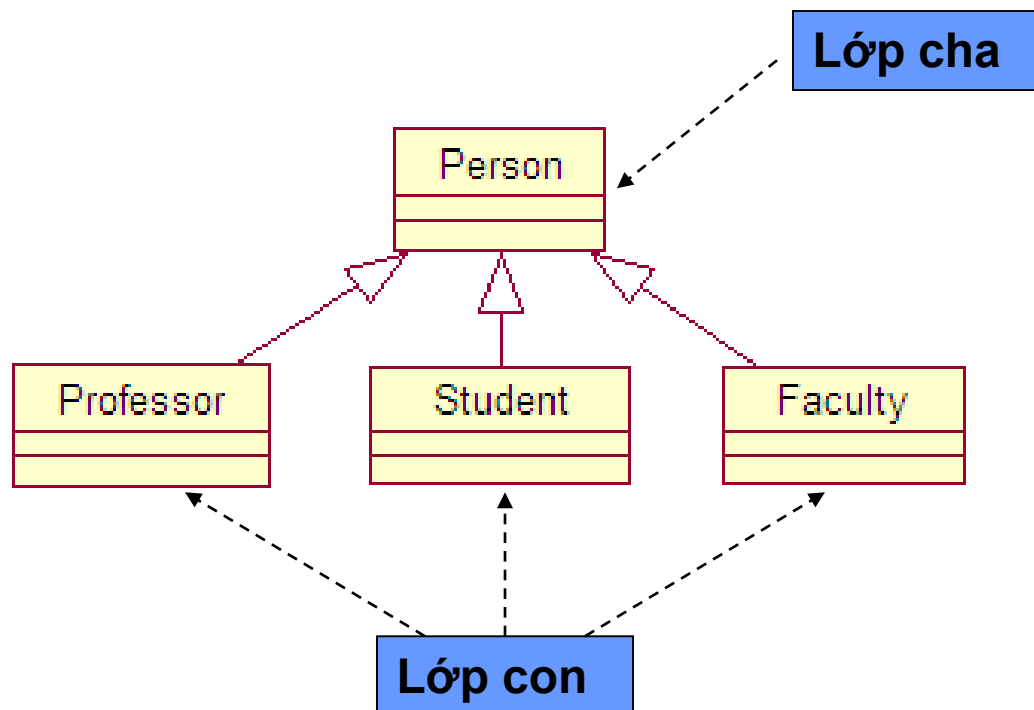
- Quan hệ tổng hợp (Composition): Là một dạng đặc biệt của quan hệ tập hợp, có tính sở hữu cao. Trong đó, các bộ phận của một thực thể **không thể sống lâu hơn** thực thể.



Chú ý: Nếu không có sinh viên → không có schedule. Hoặc, nếu không tồn tại quyển sách (book) thì không có các chương sách (chapter)

Phụ lục – UML

- Quan hệ thừa kế (Inheritance):



Assignment

1. Hãy cho biết trong trò chơi trên, bao gồm những đối tượng/sự vật nào?
2. Mỗi đối tượng có thể thực hiện các thao tác/hành động nào?
3. Liệt kê thuộc tính của từng đối tượng.

