Tổng quan
Lập trình Hướng đối tượng

v 2.1 - 02/2014

Nội dung

1. Tổng quan Lập trình Hướng đối tượng
2. Thiết kế hướng đối tượng với UML
   - 2.1. UML
   - 2.2. Class Diagram - Biểu đồ lớp
   - 2.3. Xác định lớp

Vì sao phải là đối tượng ?

Cho đến tận hôm nay...

máy tính vẫn chỉ thao tác trên các số 0 và 1

Nhưng số nhị phân là khó (cho con người) để làm việc

Hướng đến mức cao hơn của việc trừu tượng hóa

?

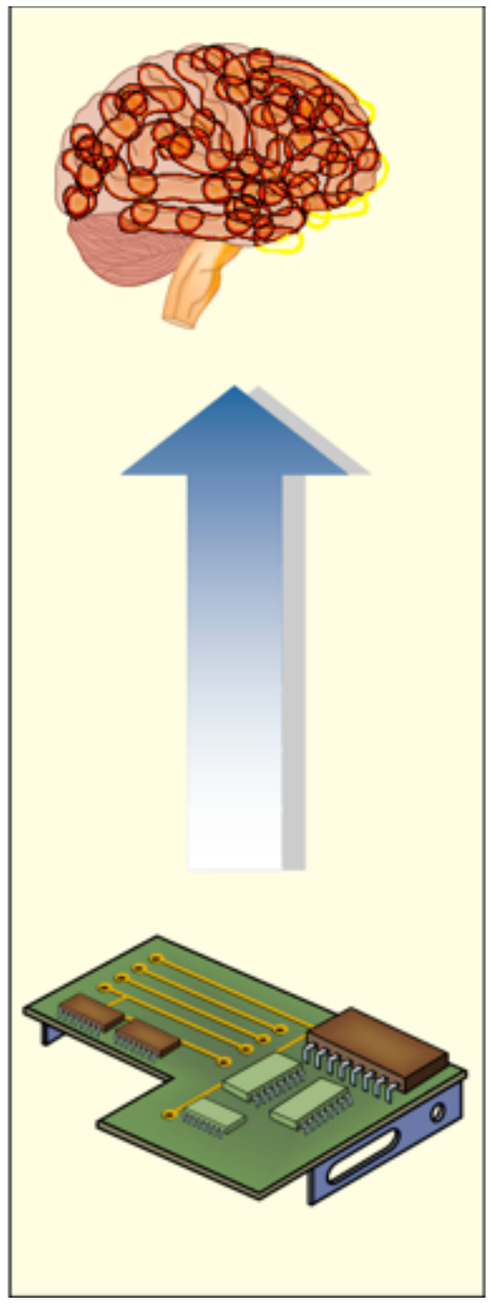
declarative languages (Haskell, ML, Prolog...)

OO languages (C++, Java, Python...)

procedural languages (C, Fortran, COBOL...)

assembly languages

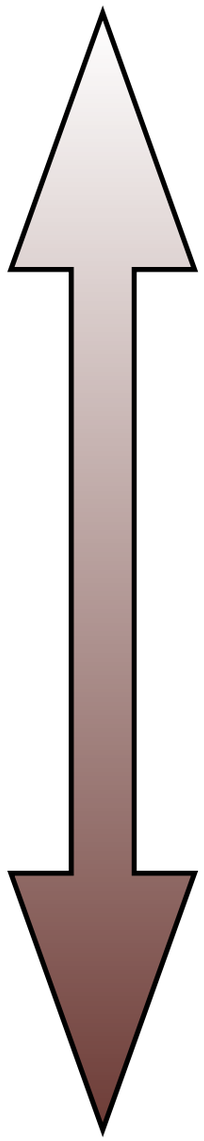
binary code



Luôn có những đánh đổi

Ngôn ngữ cấp cao

- gần với bài toán
- phụ thuộc vào hệ thống



Java, C#, Objective C

Fortran, COBOL, C++

C/C++

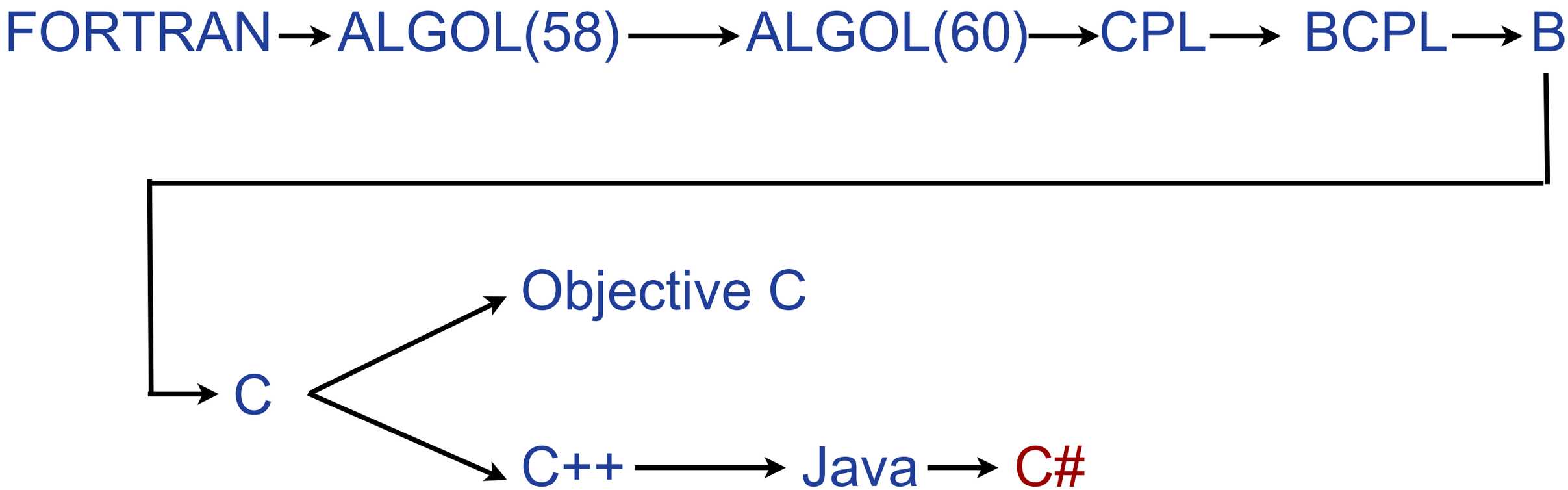
Ngôn ngữ cấp thấp

- gần với hệ thống
- không ánh xạ đến bài toán

Assembler

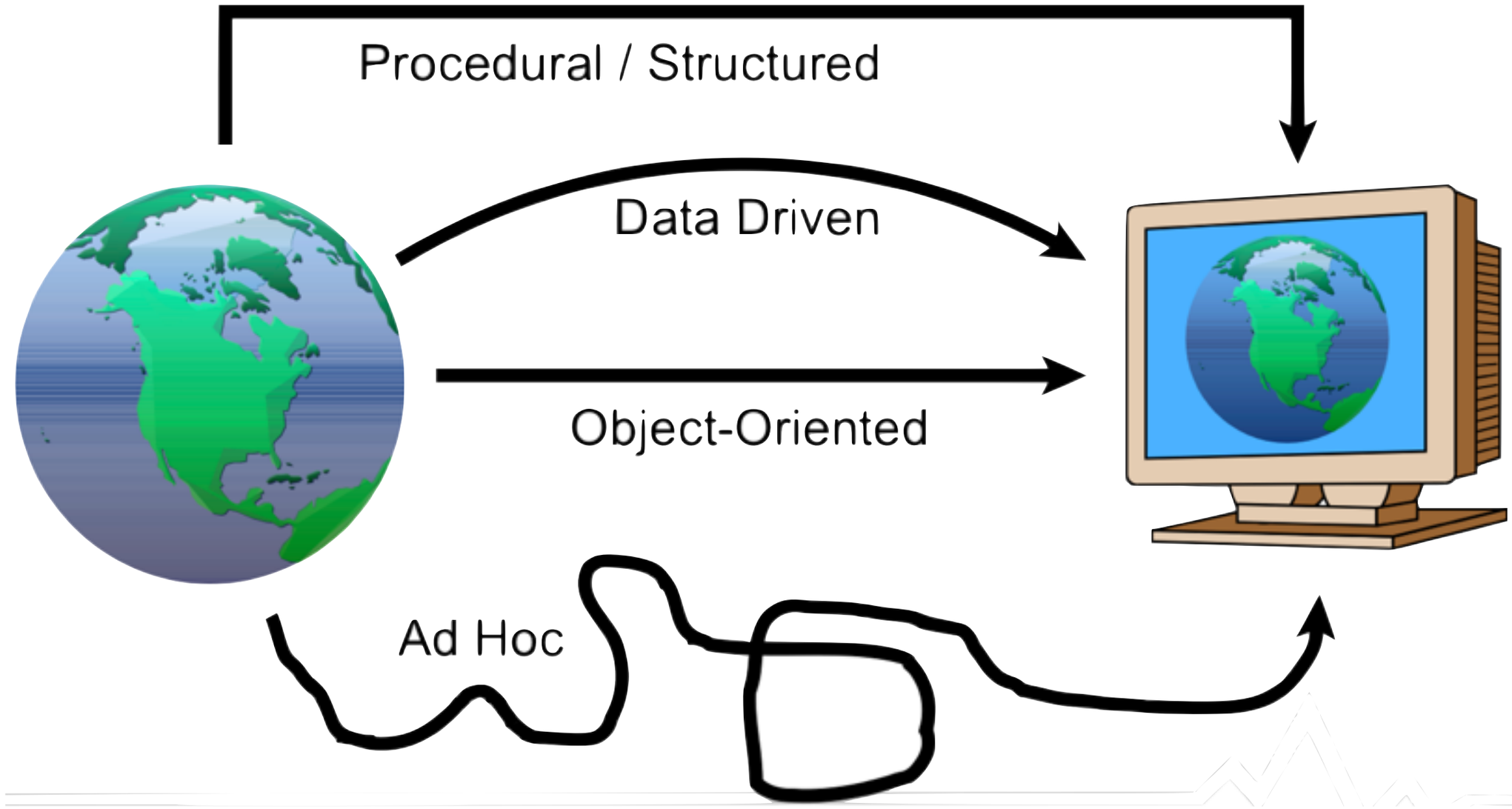
Machine

Các ngôn ngữ lập trình



Tham khảo *Lịch sử ngôn ngữ lập trình*, http://www.levenez.com/lang/

Phương pháp lập trình

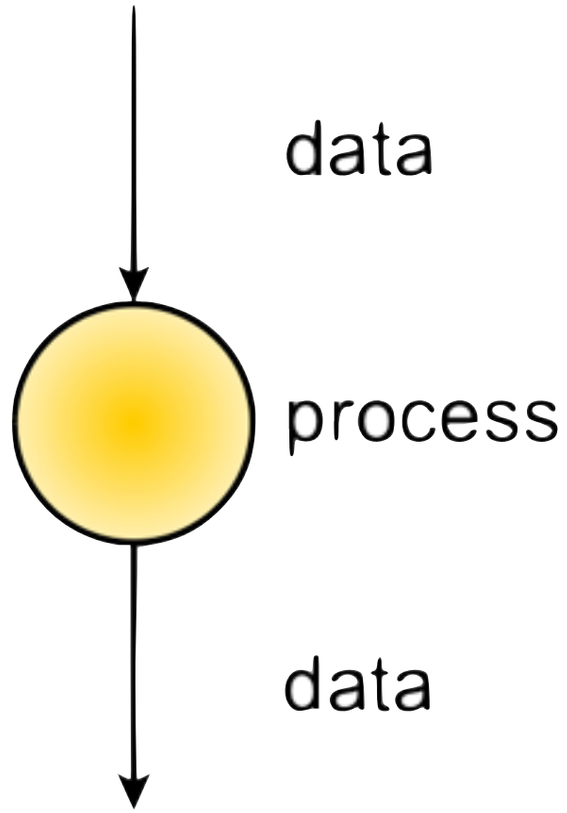


Hướng thủ tục

- Tập trung vào cách giải quyết vấn đề (ví dụ : thuật toán)
- Chia một bài toán lớn ra làm nhiều bài toán nhỏ
  - Các thủ tục hoặc chương trình con
- Ghép nối các đoạn chương trình xử lý các bài toán nhỏ thành một chương trình
- Hai kiểu dữ liệu (dữ liệu được định nghĩa hai vùng khác nhau)
  - Dữ liệu địa phương được định nghĩa ở bên trong và chỉ có thể truy xuất bên trong một thủ tục
  - Dữ liệu toàn cục được định nghĩa bên ngoài và có thể truy xuất ở bất kỳ đâu trong chương trình
- Dữ liệu toàn cục dẫn đến hiện tượng ghép nối thủ tục

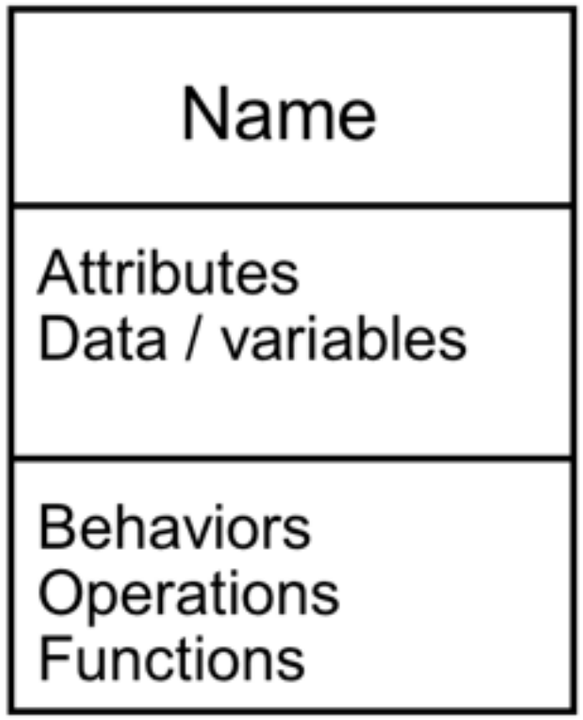
Hướng dữ liệu

- Cố gắng đầu tiên để cải tiến mô hình hướng thủ tục
- Luồng dữ liệu
  - Nối dữ liệu đầu vào với dữ liệu đầu ra
  - Thiết kế cấu trúc dữ liệu trước
  - Thiết kế tiến trình / hàm sau
- Che dấu dữ liệu
  - Đóng gói dữ liệu và các thủ tục xử lý dữ liệu cùng trong một module
  - Dữ liệu vẫn trong miền toàn cục nhưng chỉ cho phép truy xuất thông qua các hàm của module
- Abstract Data Type (ADT)
  - Người lập trình tạo ra kiểu dữ liệu
  - struct trong ngôn ngữ C#



Hướng đối tượng

- "Object-oriented modeling and design is a new way of thinking about problems using models organized around real-world concepts. The fundamental construct is the object, which combines both data structure and behavior in a single entity." James Rumbaugh, Object-Oriented Modeling and Design
- Các đặc tính của mô hình thủ tục và dữ liệu
- Sự tổ chức tự nhiên cho dữ liệu và chức năng
  - Các đối tượng đóng gói dữ liệu và chức năng với nhau
  - Hỗ trợ ADT : nhiều đối tượng của một kiểu có thể được tạo ra (class là một kiểu đặc biệt hay ADT)
  - Hỗ trợ che dấu dữ liệu : truy xuất dữ liệu được kiểm soát thông qua các từ khóa



UML class symbol

Mô hình

Lập trình Hướng đối tượng

Object & class

Document
name type state nameBorrower dateBorrow dateReminder
calculDateReminder

Tập trung những dữ liệu có cùng kiểu và những xử lý liên quan trong cùng một đơn vị vật lý để làm cho đơn giản việc duy trì hệ thống và việc truy xuất thông tin khi cải tiến hệ thống về sau.

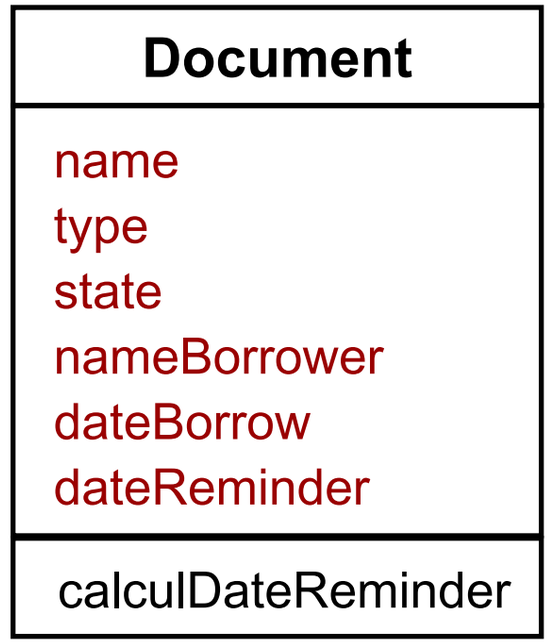
Object & class

- Một *đối tượng* là một thực thể trong miền xác định có một định danh riêng (*tên*)
- Một tập những *đặc tính* (attribute) mô tả tình trạng của đối tượng
- Một tập các *thao tác* (phương thức - methods) định nghĩa các hành vi của đối tượng
- Một đối tượng là một *thể hiện* (instance) của một *lớp*
- *Lớp* là kiểu dữ liệu trừu tượng, được mô tả bởi những thuộc tính (đặc tính và phương thức) chung của các đối tượng và cho phép tạo ra các đối tượng có những thuộc tính đó

Nhận xét - Object & class

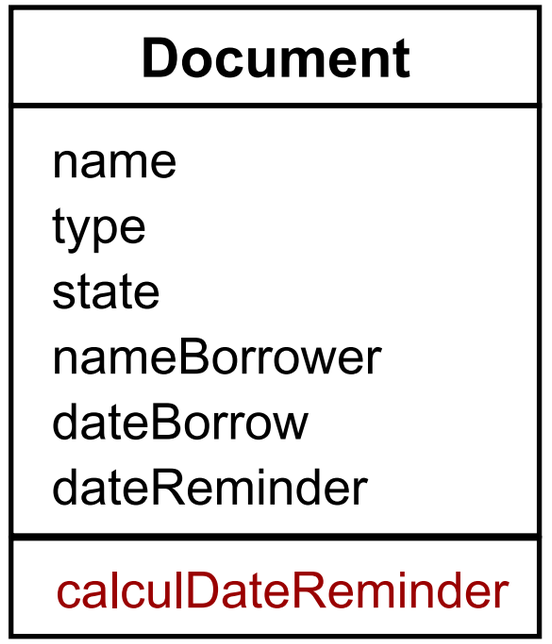
- Đối tượng là tác nhân trung tâm của mô hình đối tượng
  - Các thực thể có ý nghĩa trong ngữ cảnh ứng dụng
- Một thể hiện cụ thể của một lớp (class)
  - Các đối tượng với cùng thuộc tính và hành vi được mô tả bởi cùng một lớp
  - Dữ liệu trong mỗi đối tượng là phân biệt so với dữ liệu trong tất cả các đối tượng khác được khởi tạo từ cùng lớp
- Lớp là sự trừu tượng hóa của một hay nhiều đối tượng
  - Mô tả “những thứ” có cùng thuộc tính và hành vi
  - Cung cấp sự che dấu dữ liệu
    - Dữ liệu đặt trong một vùng duy nhất và việc truy xuất là bị kiểm soát
    - Việc truy xuất thông qua các public interface (method hay member function)
  - Kiểu dữ liệu mới

Attribute - đặc tính



- Mô tả một đối tượng
- Đặc tả hay phân biệt các đối tượng với nhau
- Là các giá trị dữ liệu (biến) lưu trữ trong đối tượng
- Là dữ liệu mà một đối tượng phải có trách nhiệm bảo quản
- Nên được đặt ở cấp cao nhất trong cây phân cấp thừa kế
- Đặc tính tốt phụ thuộc vào việc mô hình hóa

Behavior - hành vi



- Còn được gọi là `member function` hay `method`
- Một chức năng có thể được áp dụng cho hay bởi một đối tượng
- Các thao tác, hành vi là logic (người sử dụng có thể hình dung được); `member function` là các hàm vật lý mà nó cài đặt các hành vi hay thao tác
- Được gọi thông qua một đối tượng

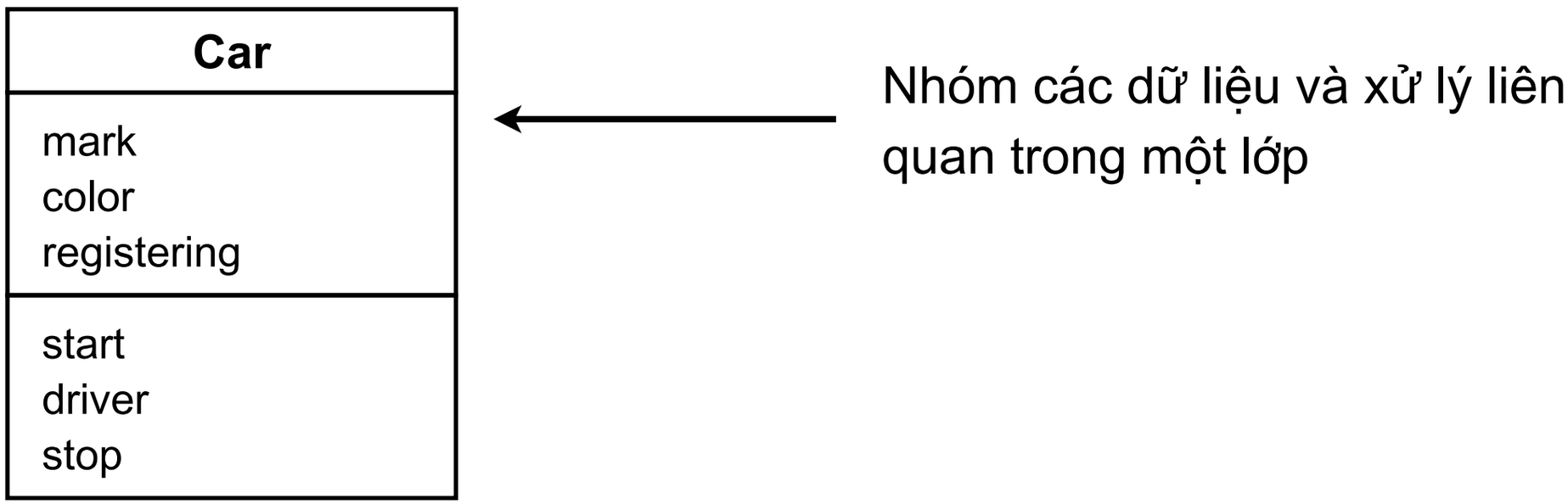
```
Bitmap bm = new Bitmap(20, 20);
bm.Save("bitmap.png");
```

- Một số thao tác có thể được áp dụng cho nhiều lớp và đó là đa hình (cài đặt thông qua nhiều `method`)

Encapsulation - bao gói

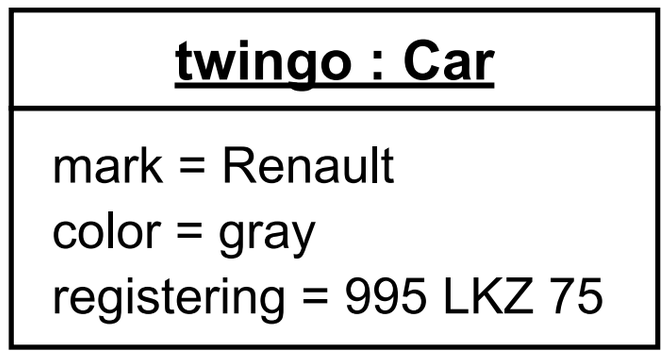
- Là việc che dấu những chi tiết cài đặt của đối tượng bằng cách định nghĩa một giao diện
- *Giao diện* là bề ngoài của một đối tượng, nó định nghĩa những khả năng truy xuất cho người sử dụng của đối tượng đó
- *Bao gói* làm đơn giản quy trình cải tiến ứng dụng vì nó cố định việc sử dụng các đối tượng : có thể thay đổi cài đặt của các đặc tính mà không làm thay đổi đến giao diện
- *Bao gói* đảm bảo sự toàn vẹn dữ liệu bởi vì nó cấm truy xuất trực tiếp vào các đặc tính của đối tượng (sử dụng các *từ khóa truy xuất*)

Ví dụ



Nhóm các dữ liệu và xử lý liên quan trong một lớp

Object : một thể hiện của lớp



Ví dụ

Bài toán Thiết kế và xây dựng một trò chơi hockey trên máy tính

Đối tượng Cầu thủ hockey

Thuộc tính Vị trí, chiều cao, cân nặng, lương, số bàn thắng

Hành vi Chuyền bóng, sút, trượt về phía trước, trượt về phía sau, húc cầu thủ khác, vân vân

Ví dụ khác

Bài toán Mô hình hóa việc tính toán trong sinh vật học

Viết chương trình mô phỏng sự sinh sôi của quần thể virus trong con người theo thời gian. Mỗi tế bào virus tự sinh sôi sau một khoảng thời gian nhất định. Bệnh nhân có thể uống thuốc để kiềm chế quá trình sinh sôi này, và loại bỏ các tế bào virus ra khỏi cơ thể. Tuy nhiên, một số tế bào chống lại thuốc và có thể tiếp tục tồn tại.

Viết chương trình mô phỏng sự sinh sôi của quần thể virus trong con người theo thời gian. Mỗi tế bào virus tự sinh sôi sau một khoảng thời gian nhất định. Bệnh nhân có thể uống thuốc để kiềm chế quá trình sinh sôi này, và loại bỏ các tế bào virus ra khỏi cơ thể. Tuy nhiên, một số tế bào chống lại thuốc và có thể tiếp tục tồn tại.

Đâu là đối tượng ?

Thuộc tính ?

Hành vi ?

Viết chương trình mô phỏng sự sinh sôi của quần thể virus trong con người theo thời gian. Mỗi tế bào virus tự sinh sôi sau một khoảng thời gian nhất định. Bệnh nhân có thể uống thuốc để kiềm chế quá trình sinh sôi này, và loại bỏ các tế bào virus ra khỏi cơ thể. Tuy nhiên, một số tế bào chống lại thuốc và có thể tiếp tục tồn tại.

Đâu là đối tượng ?

Thuộc tính ?

Hành vi ?

Bệnh nhân

Thuộc tính

- số lượng virus
- sự miễn dịch (%)

Hành vi

- uống thuốc

Virus

Thuộc tính

- tốc độ sinh sản (%)
- sự kháng thuốc (%)

Hành vi

- sinh sôi
- sống sót

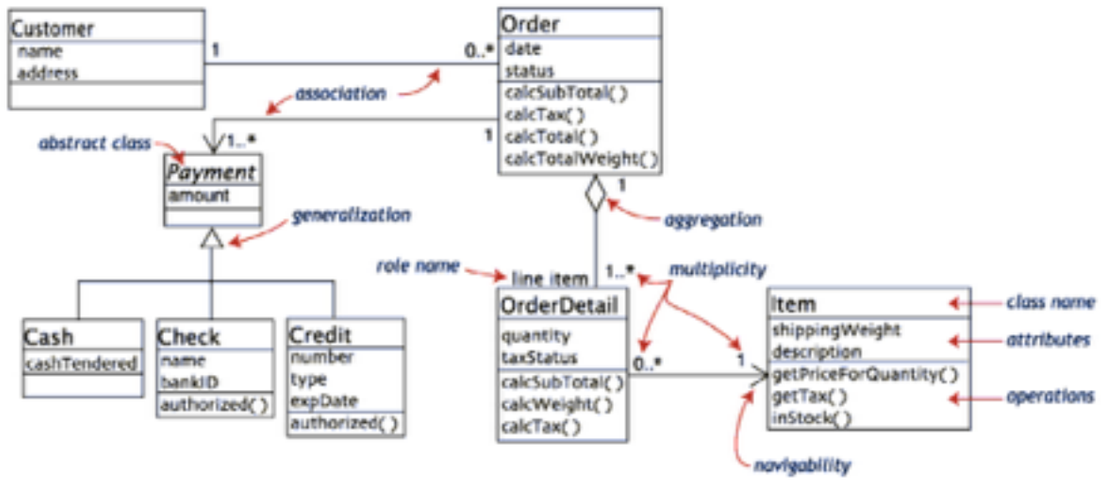
Hỏi

Tại sao chúng ta không mô hình một đối tượng tên là Bác sĩ ? Rõ ràng là bệnh viện nào cũng có bác sĩ ?

Bệnh nhân nào cũng có tuổi ? Giới tính ? Bệnh ? Triệu chứng ? Vì sao chúng ta không mô hình chúng như các thuộc tính ?

Class Relationships

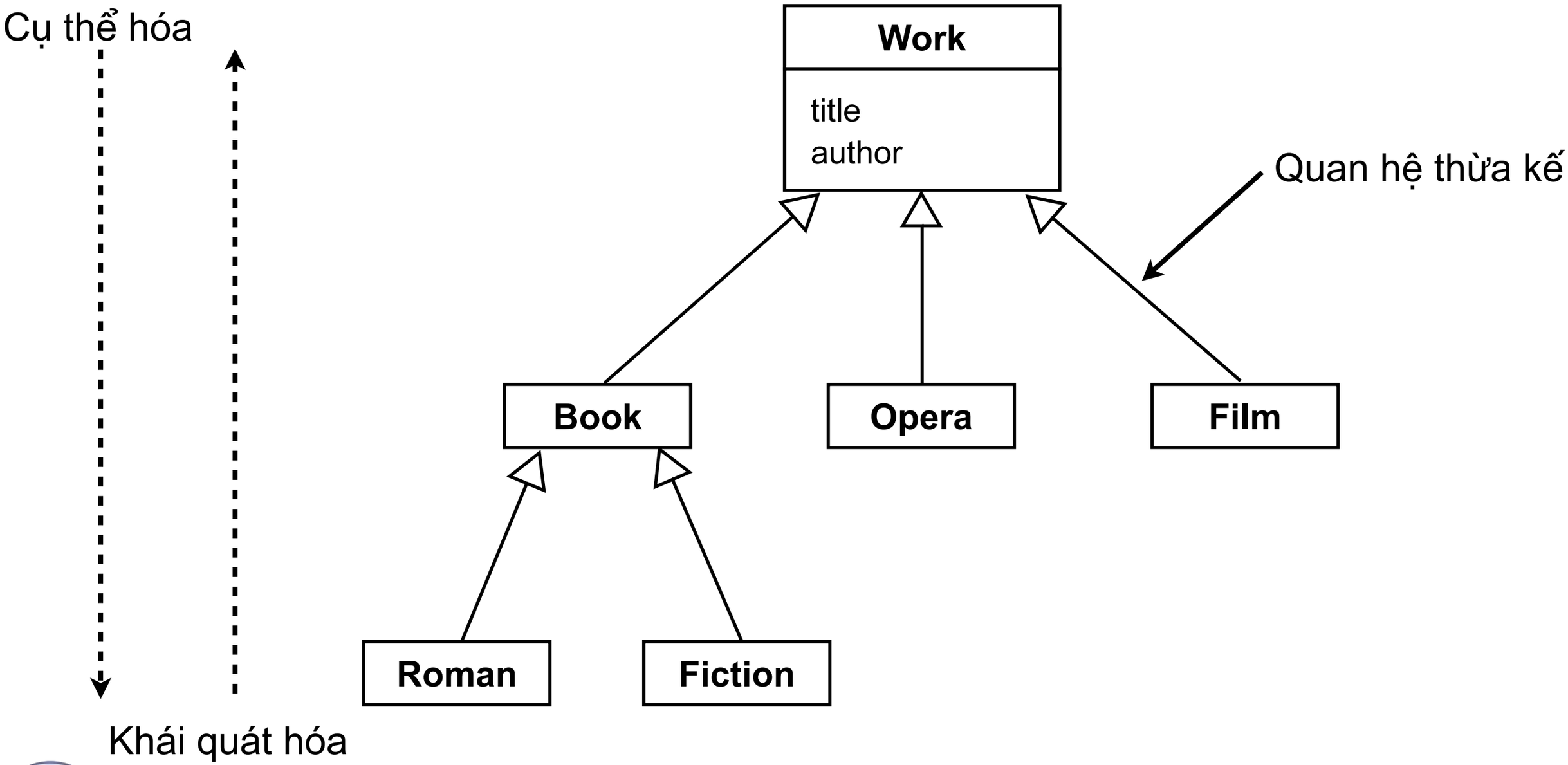
- Thể hiện hệ thống như một tập các mối quan hệ
  - Cố gắng bắt chước các mối quan hệ giữa các đối tượng trong thể giới thực
  - Được vẽ dưới dạng biểu đồ hoặc đồ thị liên kết
    - Các nút hay các đỉnh là các lớp
    - Các cạnh, cung, đường là các mối quan hệ
- Cho phép các đối tượng tương tác trong giải pháp toàn cục của vấn đề
- Được hỗ trợ bởi cú pháp ngôn ngữ máy đặc biệt



Inheritance và Polymorphism

- *Thừa kế* là cơ chế truyền các thuộc tính của một lớp (đặc tính và phương thức) cho lớp con của nó
- Một lớp có thể được *cụ thể hóa* bằng các lớp khác để thêm những đặc điểm riêng hoặc làm cho phù hợp
- Nhiều lớp có thể được *tổng quát hóa* trong một lớp, nhóm các đặc điểm chung của một tập các lớp
- Cụ thể hóa hay tổng quát hóa cho phép tạo ra sự phân cấp của các lớp
- Thừa kế xóa bỏ sự trùng lặp và khuyến khích sự dùng lại
- *Đa hình* thể hiện khả năng của *một hàm* có thể tương thích với các đối tượng khác nhau của những lớp khác nhau

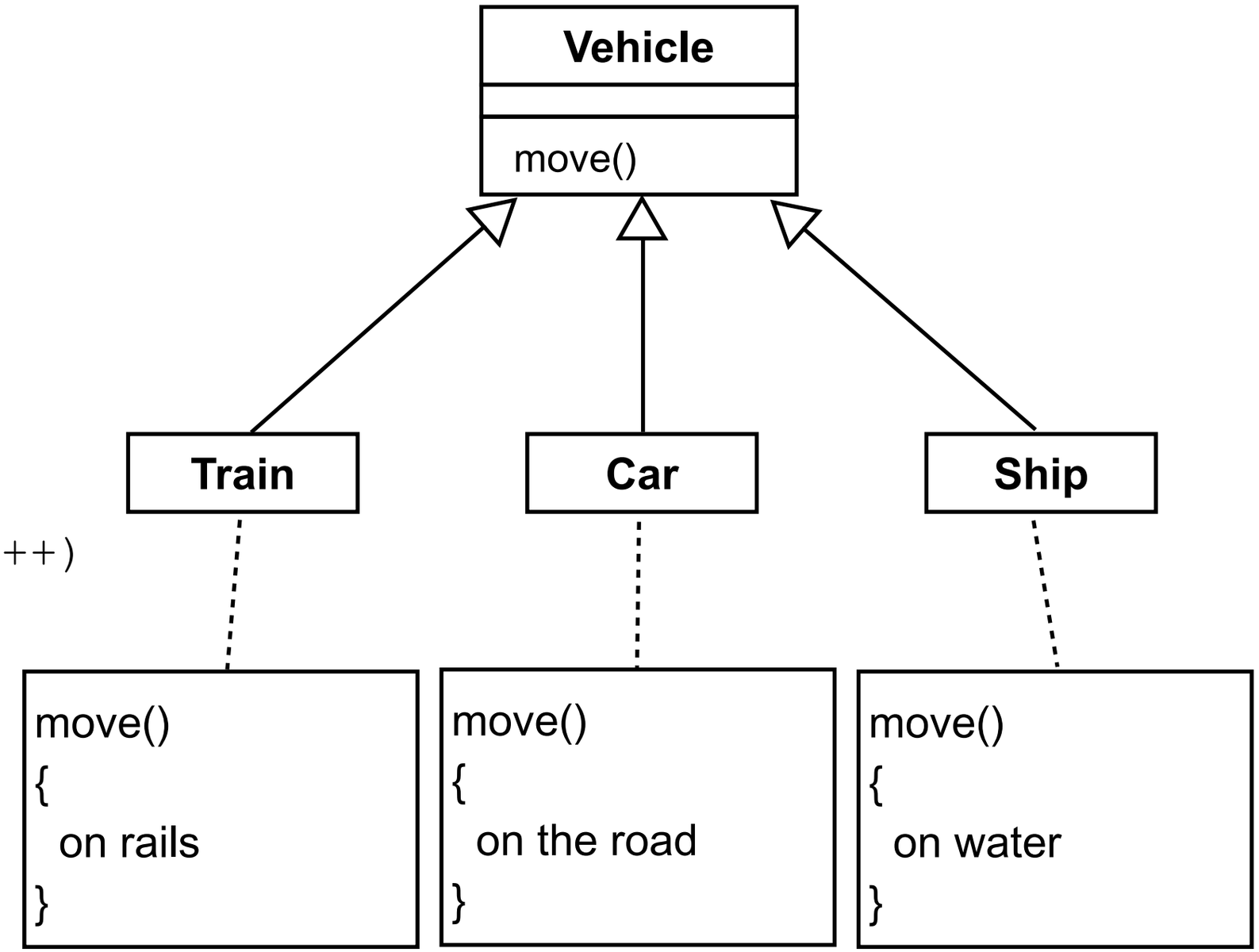
Ví dụ về sự phân cấp lớp



Ví dụ về đa hình

```
Vehicle veh [ 3 ] = {
    Train(“TGV”),
    Car(“twingo”),
    Ship(“Titanic”)
};

for (int i = 0; i < 3; i++)
{
   veh[ i ].move();
}
```



UML

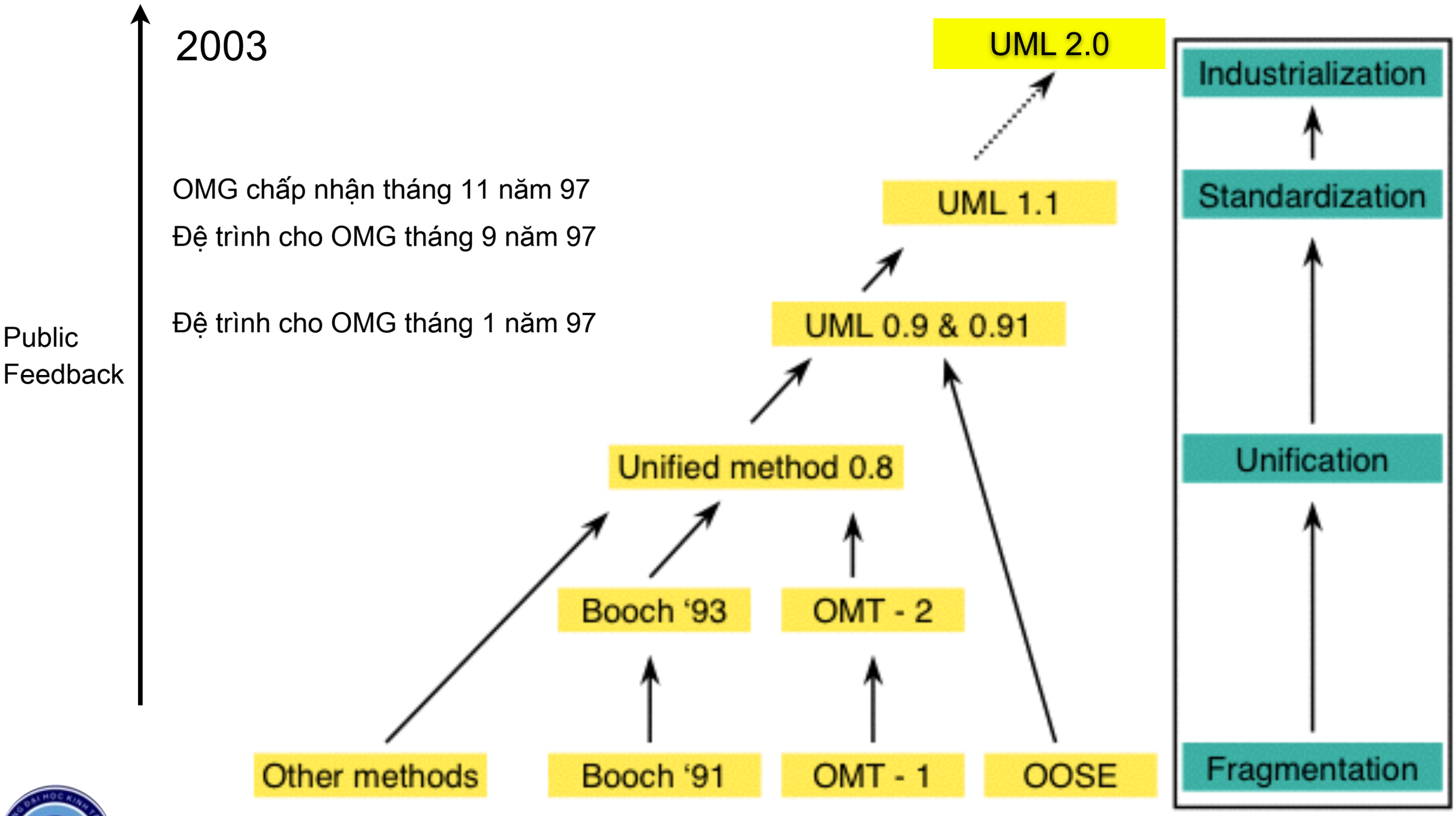
UML

- UML là viết tắt của Unified Modeling Language
  - UML là một ngôn ngữ
- Ngôn ngữ là công cụ để giao tiếp và trao đổi ý tưởng
- UML là một khái niệm chứ không phải là một phương pháp
  - Nó có thể được sử dụng kết hợp với bất kỳ phương pháp nào khác
- Một chuẩn của OMG và là chuẩn công nghiệp hiện nay
- UML sử dụng hầu hết các khái niệm hình vẽ để diễn đạt cho việc phân tích và thiết kế hướng đối tượng
- Nó đơn giản hóa các tiến trình phức tạp của thiết kế hệ thống

Vì sao dùng UML cho mô hình hóa ?

Việc sử dụng các khái niệm hình vẽ giúp giao tiếp rõ ràng, dễ hiểu hơn với *ngôn ngữ tự nhiên* (mập mờ) cũng như với *mã nguồn* (quá chi tiết)

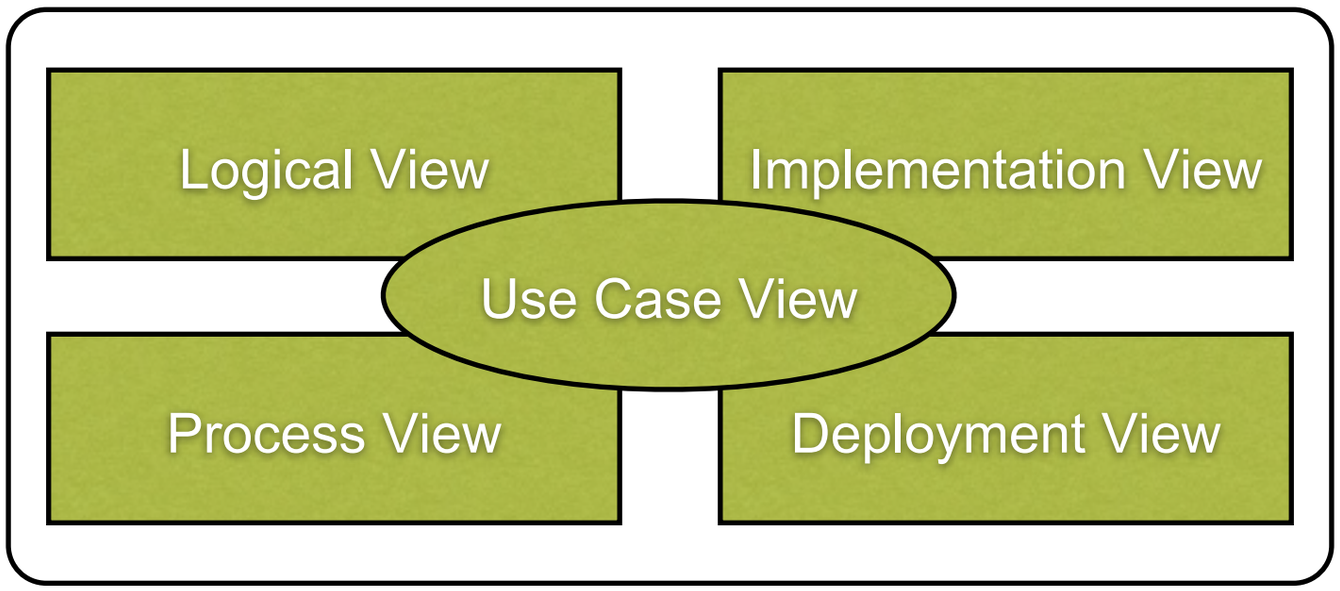
Lịch sử UML


2003
OMG chấp nhận tháng 11 năm 97
Đệ trình cho OMG tháng 9 năm 97
Đệ trình cho OMG tháng 1 năm 97
Public
Feedback
UML 2.0
UML 1.1
UML 0.9 & 0.91
Unified method 0.8
Booch '93
OMT - 2
Other methods
Booch '91
OMT - 1
OOSE
Industrialization
Standardization
Unification
Fragmentation

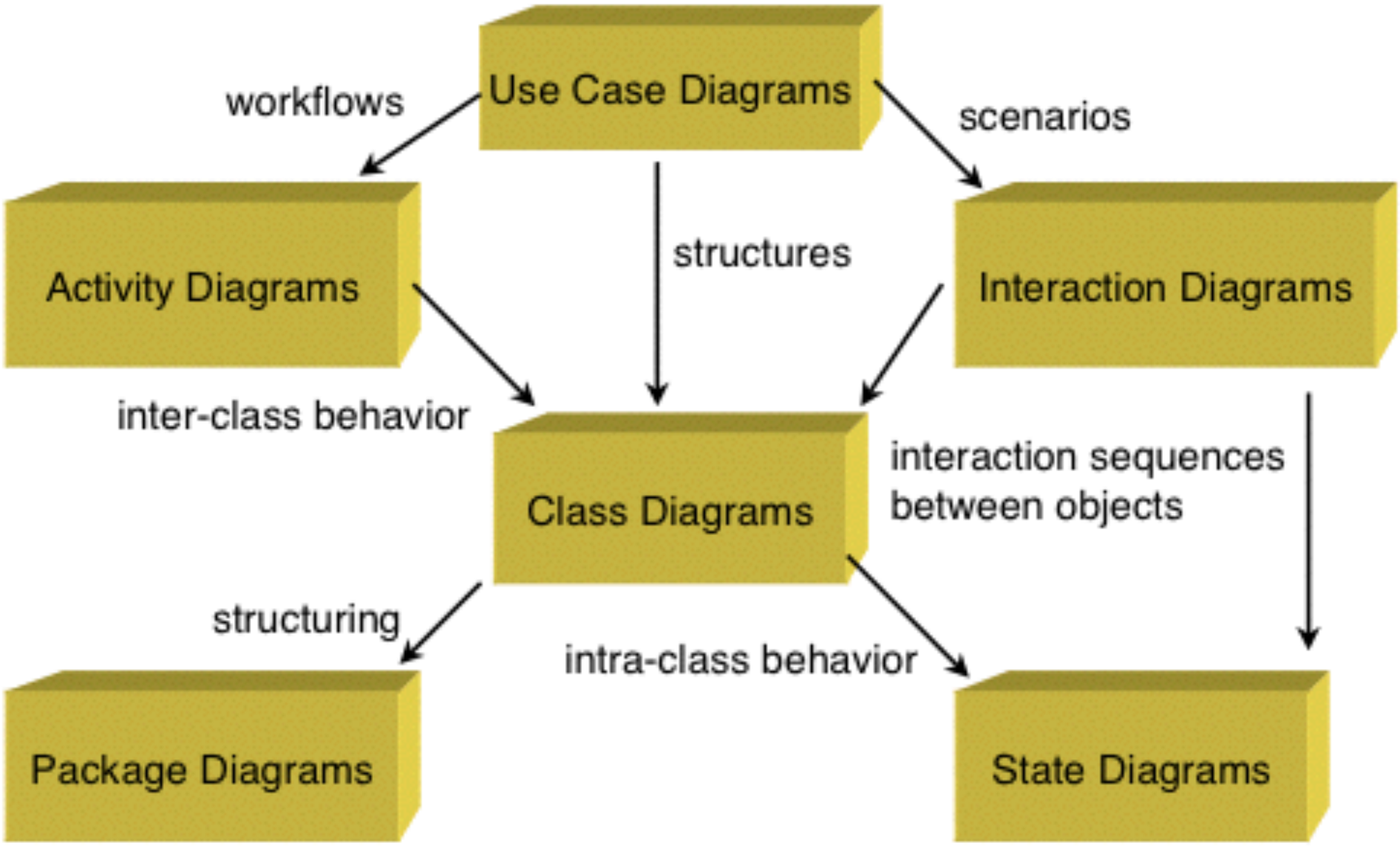
Góc nhìn trong mô hình hóa

Luôn có nhiều cách để nhìn một cái gì đó : mô tả một đối tượng là phụ thuộc vào góc nhìn

4+1 góc nhìn của UML



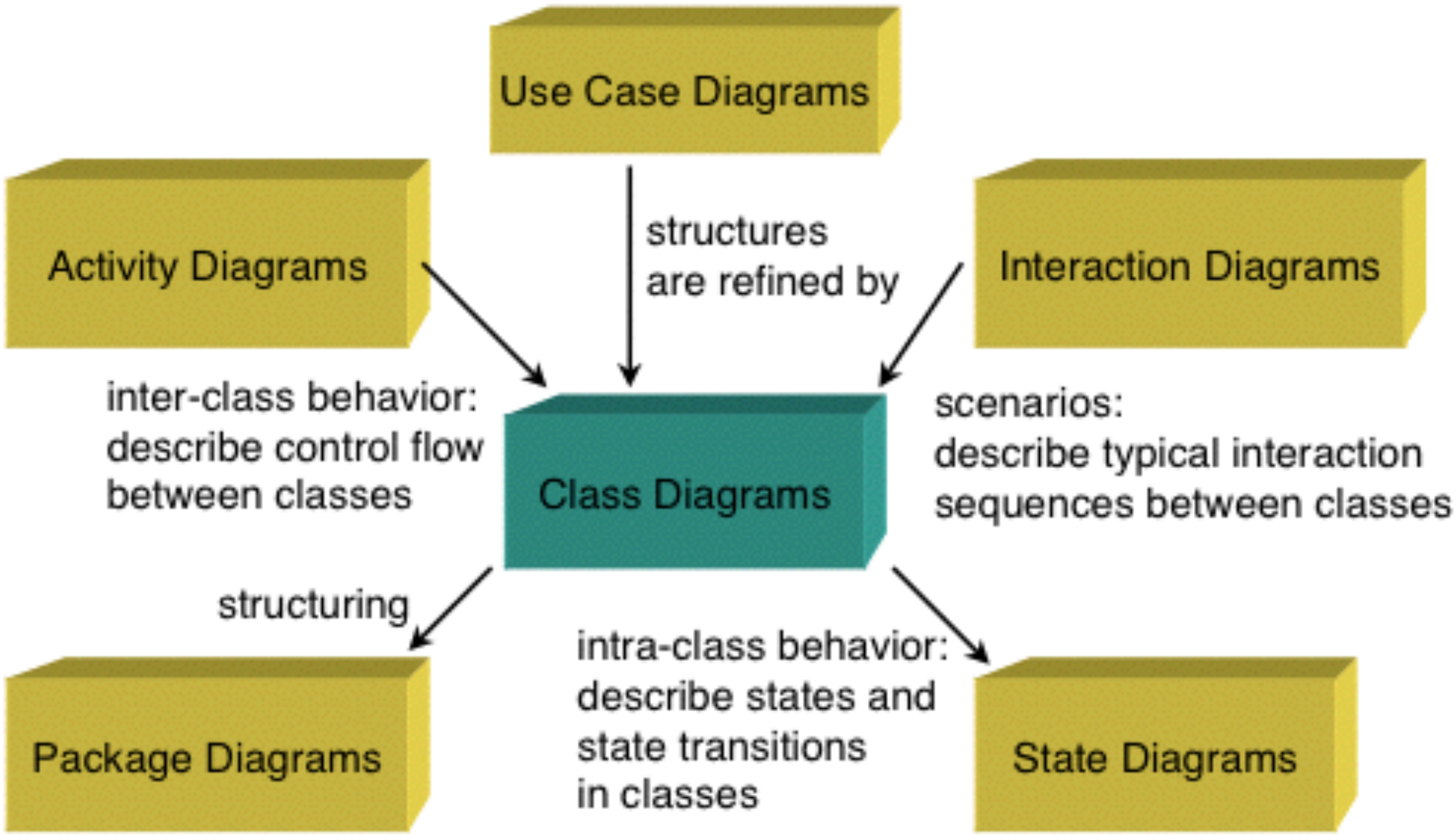
Các biểu đồ của UML



Class Diagram

Biểu đồ lớp

Vai trò của biểu đồ lớp



Biểu đồ lớp

- Các biểu đồ lớp mô tả các **kiểu** của các **đối tượng** trong hệ thống và các **quan hệ tĩnh** tồn tại giữa chúng
- Có hai kiểu quan hệ tĩnh chính :
  - kết hợp (association)
    - khách hàng có thể mượn một số đĩa DVD
  - kiểu con (subtype)
    - sinh viên là một kiểu của con người
- Các biểu đồ lớp cũng thể hiện các **đặc tính** và **hành vi** của một lớp và **các ràng buộc** áp dụng cho các mối liên hệ giữa các đối tượng

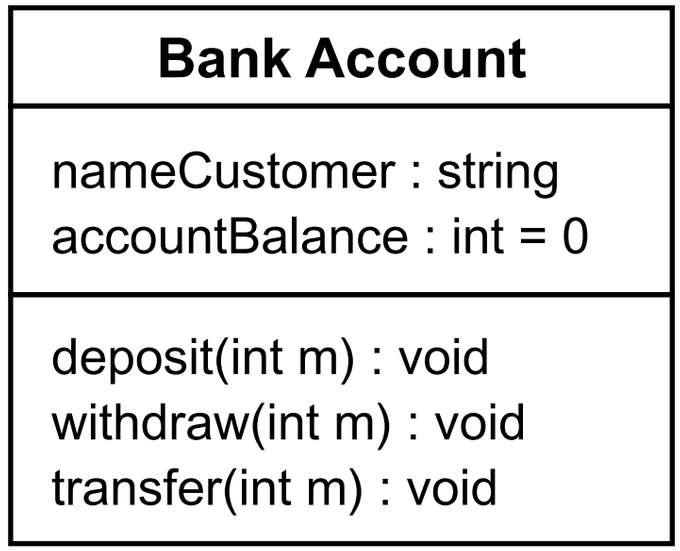
Thể hiện lớp

- Mỗi lớp được thể hiện bằng một hình chữ nhật được chia làm 3 phần
- Các đặc tính nằm ở giữa
- Các hành vi ở dưới
  - Có thể bỏ đi
- Cú pháp của đặc tính

```
name : type = default
```

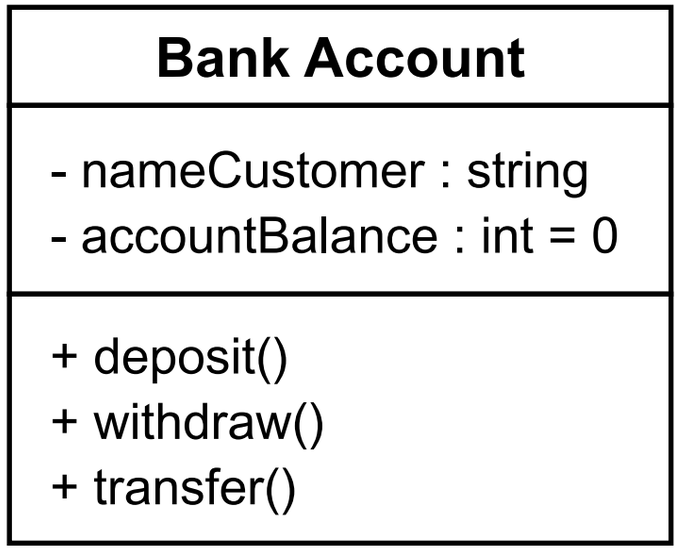
- Cú pháp của hành vi

```
name ( params ) : return type
```



Tiền tố phạm vi

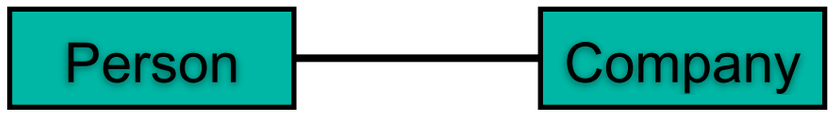
- Tiền tố phạm vi trong UML được sử dụng để che dấu thông tin



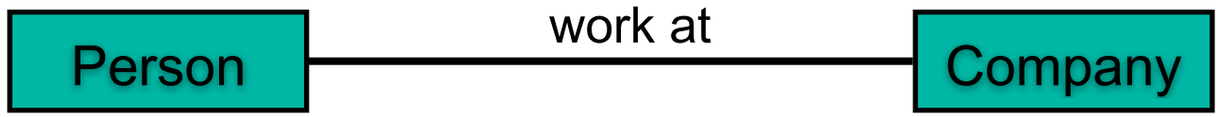
- Tiền tố + chỉ rằng một đặc tính hay hành vi là *public*
  - có thể truy xuất bởi tất cả các lớp
- Tiền tố - chỉ rằng một đặc tính hay hành vi là private
  - chỉ có thể truy xuất bởi lớp mà tại đó nó được định nghĩa
- Tiền tố # chỉ rằng một đặc tính hay hành vi là protected
  - có thể truy xuất bởi lớp mà nó được định nghĩa và cả các lớp con của lớp đó

Association - q.h. kết hợp

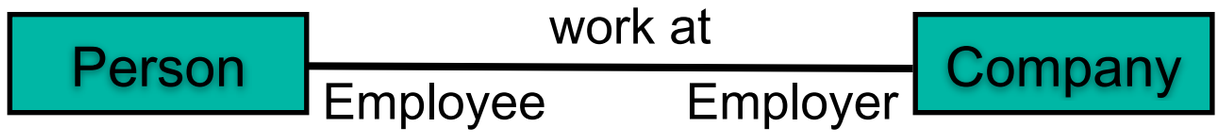
- Mối quan hệ kết hợp thể hiện các mối quan hệ giữa các lớp
  - Con người *làm việc tại* công ty



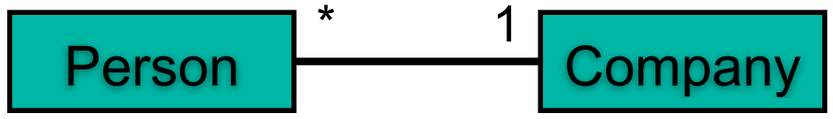
- Mỗi mối quan hệ có **tên quan hệ**



- Mỗi mối quan hệ có hai **vai trò**



- **Bản số (Multiplicites)** chỉ ra bao nhiêu đối tượng có thể tham gia vào quan hệ này
  - Một người được thuê bởi chỉ một công ty
  - Một công ty có thể thuê nhiều người



Multiplicity - bản số

- * thể hiện miền từ 0 đến vô cùng
- 1 thể hiện từ 1 đến 1
  - Một đơn hàng phải thuộc về chính xác một khách hàng
- Những bản số khác
  - một số - 11 cầu thủ bóng đá
  - một miền - 2..4 đấu thủ cho một ván bài
  - một tập hữu hạn - 2,4 cửa trong xe ôtô

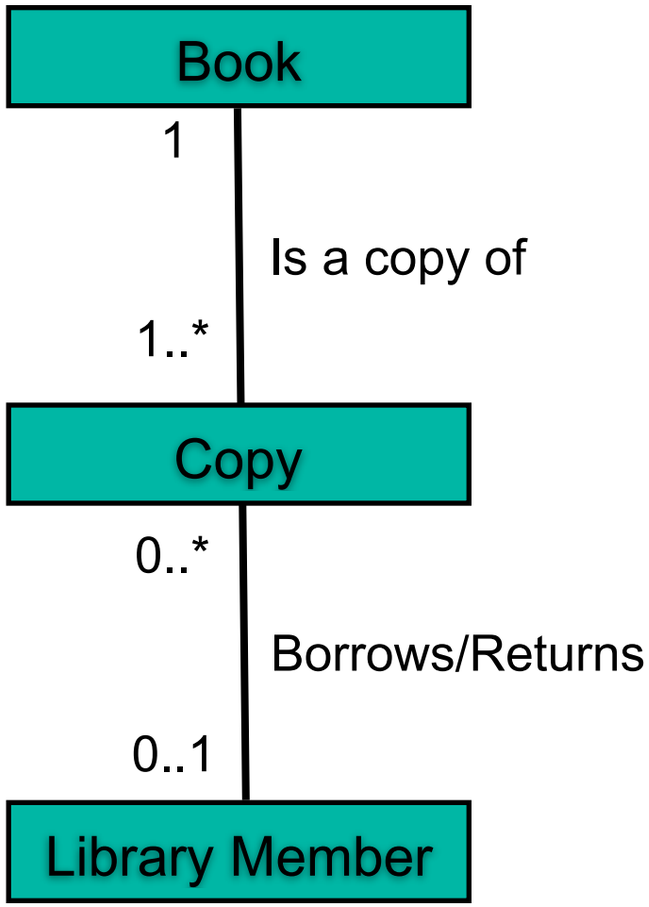
Navigability - tính khả điều hướng

- Để thể hiện hướng của các mối quan hệ kết hợp, các dấu mũi tên được thêm vào

  - Đơn hàng **biết** nó thuộc về khách hàng nào, nhưng khách hàng không biết nó có đơn hàng nào
  - Theo góc nhìn **cài đặt**, đơn hàng chứa một **con trỏ** trỏ đến khách hàng, nhưng khách hàng không trỏ đến đơn hàng
- Có ba loại điều hướng :
  - Không có hướng
  - Một chiều
  - Hai chiều

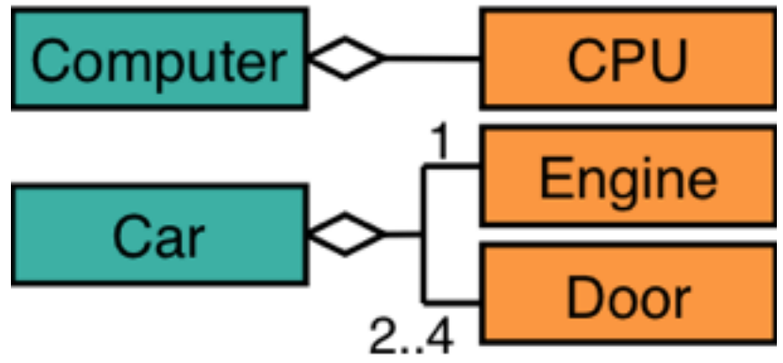
Ví dụ: Association

- Quản lý thư viện
- Một quyển sách có 1 hoặc nhiều bản sao
- Một bản sao chỉ thuộc về chính xác một quyển sách
- Một bản sao chỉ được mượn bởi 0 hoặc 1 thành viên của thư viện
- Có nên hình thành hai quan hệ kết hợp riêng cho mượn và trả sách ?

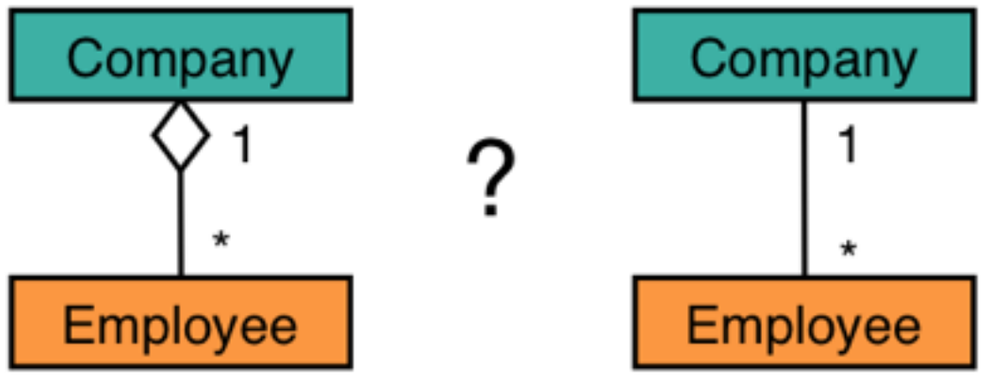


Aggregation - q.h. thu nạp

- Quan hệ thu nạp là quan hệ **part-of**
  - CPU là một phần của máy tính
  - Xe ôtô có một động cơ và các cửa

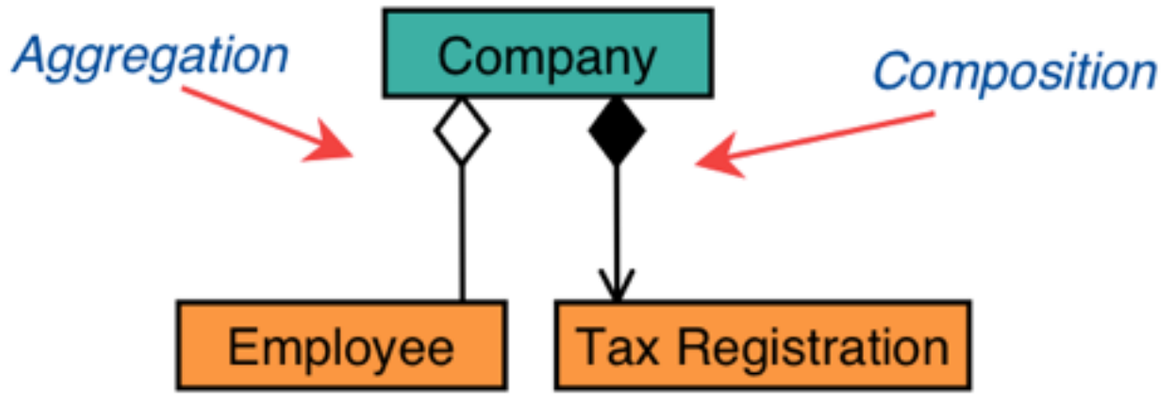


- Quan hệ thu nạp và các đặc tính :
  - Đặc tính mô tả thuộc tính của đối tượng, như tốc độ, giá, chiều dài
  - Quan hệ thu nạp mô tả kết cấu của đối tượng
- Quan hệ thu nạp và quan hệ kết hợp :



Composition - q.h. thành phần

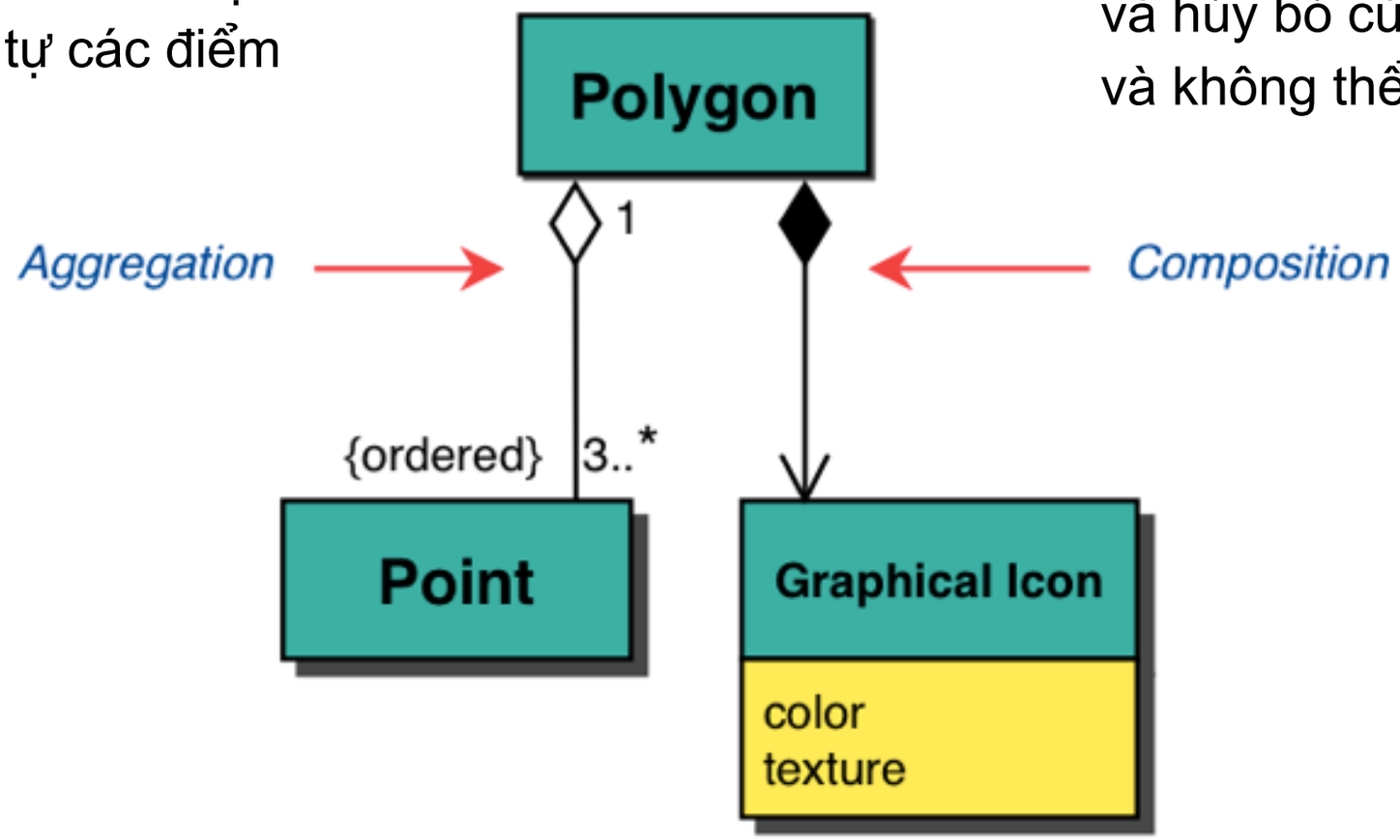
- Quan hệ thành phần là biến thể **mạnh hơn** của quan hệ thu nạp
  - Một thành phần **chỉ thuộc về** một toàn thể
  - Các thành phần thường sống và **chết** theo toàn thể
- Ví dụ :
  - Qh thu nạp : một công ty có các nhân viên. Nhân viên có thể thay đổi công ty
  - Qh thành phần : một công ty có một mã số thuế. Một mã số thuế thuộc về duy nhất một công ty



Ví dụ: Aggregation & Compostion

Một đa giác chứa một tập có thứ tự các điểm

Một biểu tượng được tạo ra và hủy bỏ cùng với đa giác và không thể thay đổi



Những điểm này có thể thay đổi khi đa giác sửa đổi

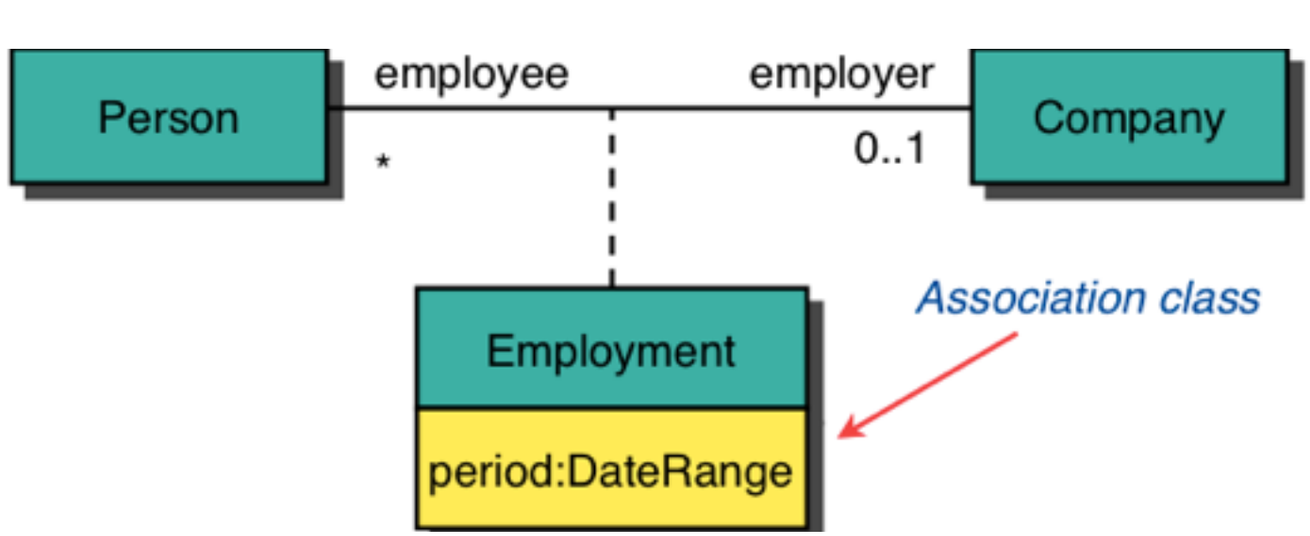
Các đặc tính của biểu tượng có thể thay đổi, nhưng nó không thể được thay thế bằng đối tượng khác

Association class

Ví dụ : Con người được thuê bởi công ty trong một khoảng thời gian

Hỏi : Đặc tính khoảng thời gian được đặt ở đâu ?

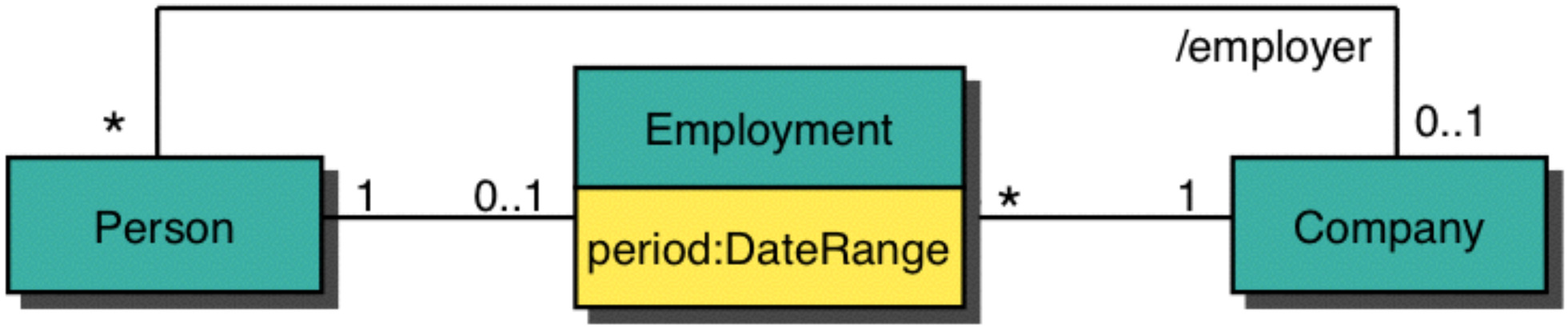
- Các lớp kết hợp cho phép mô hình các quan hệ kết hợp này bằng các lớp : **thêm các đặc tính**, **thao tác** mới cho các quan hệ này



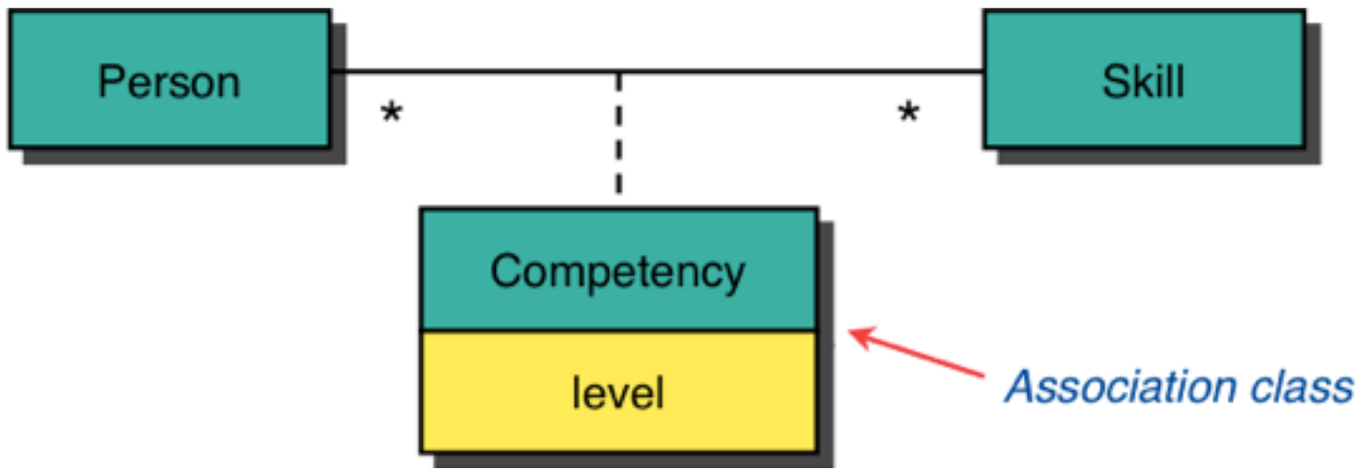
- Chú ý : một con người và một công ty chỉ kết hợp với nhau bằng **một** khoảng thời gian làm công

Association class & class

- Nếu con người có thể rời bỏ công ty, bạn có thể sử dụng lớp thay cho lớp kết hợp :



- Nhưng một người chỉ có một trình độ cho mỗi kỹ năng :

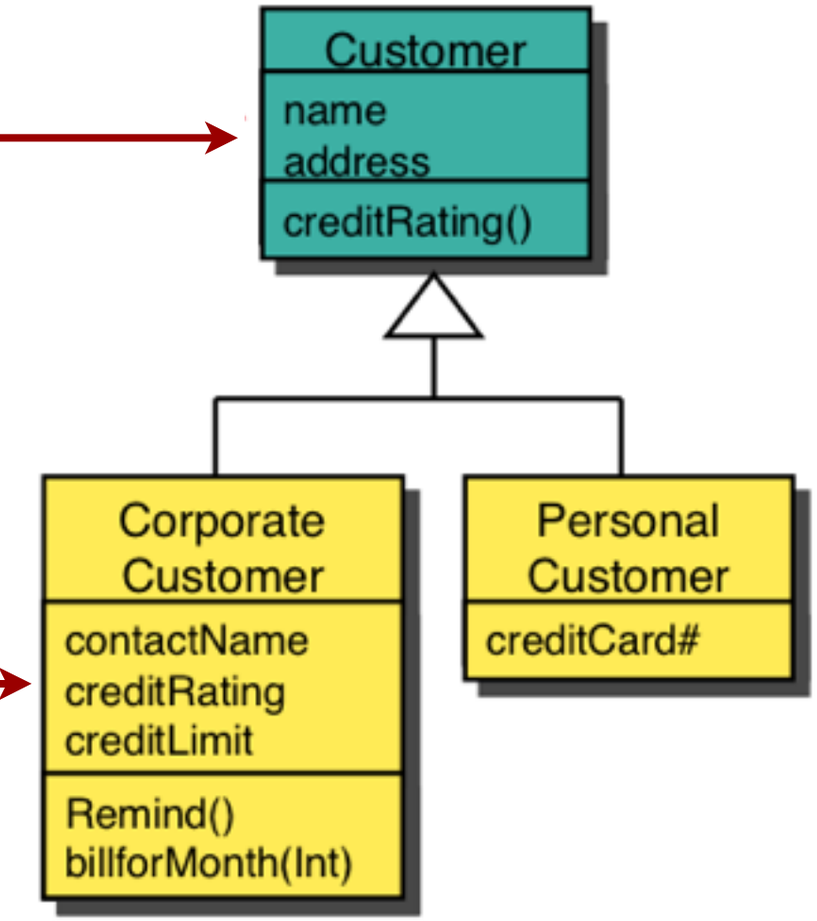


Generalization - tổng quát hóa

- **Tổng quát hóa** gom **những thứ giống nhau** giữa vài lớp trong một *lớp cha* (superclass)
- **Cụ thể hóa** (specialization) thêm **những thứ khác nhau** vào trong *lớp con*

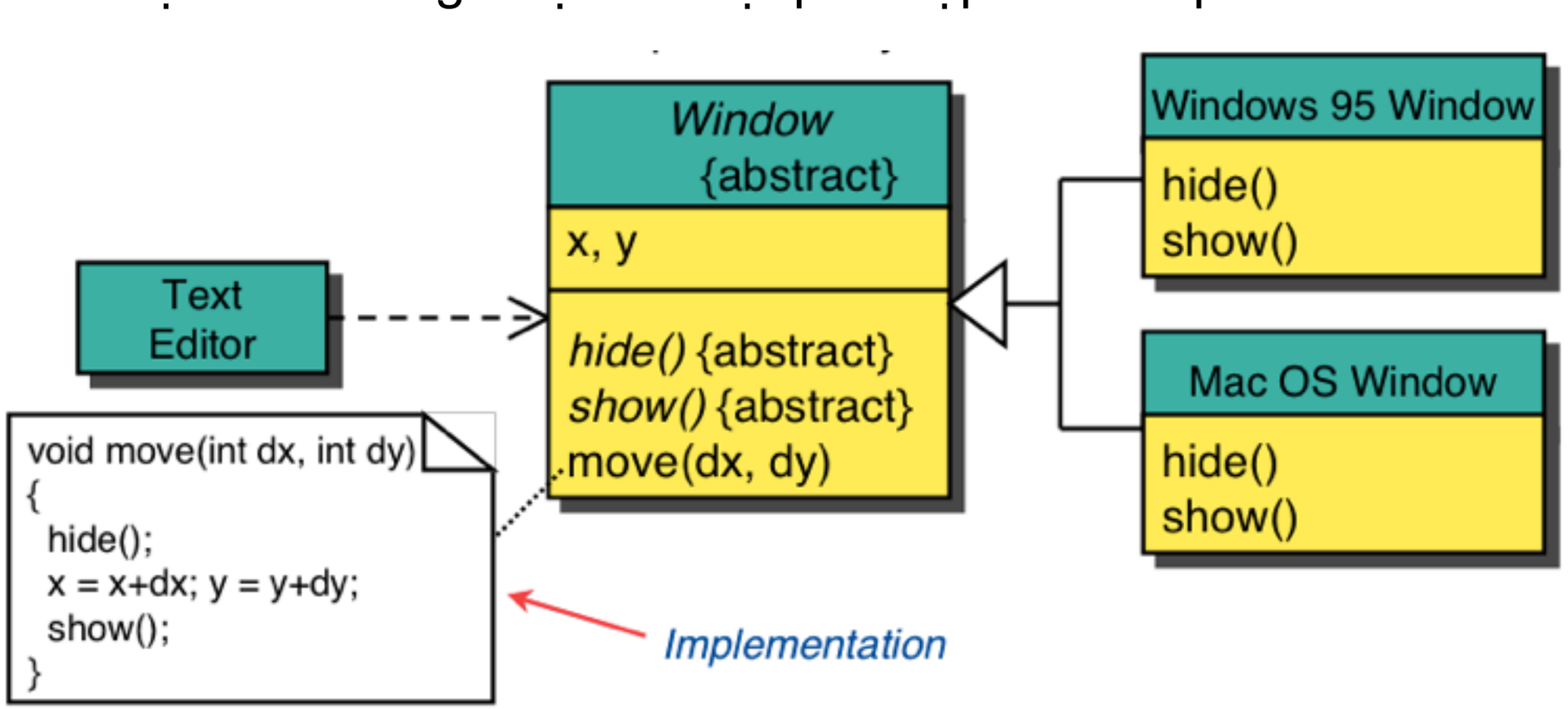
Những đặc tính giống nhau được đặt ở lớp cha

Những đặc tính khác nhau được tách ra đặt ở các lớp con



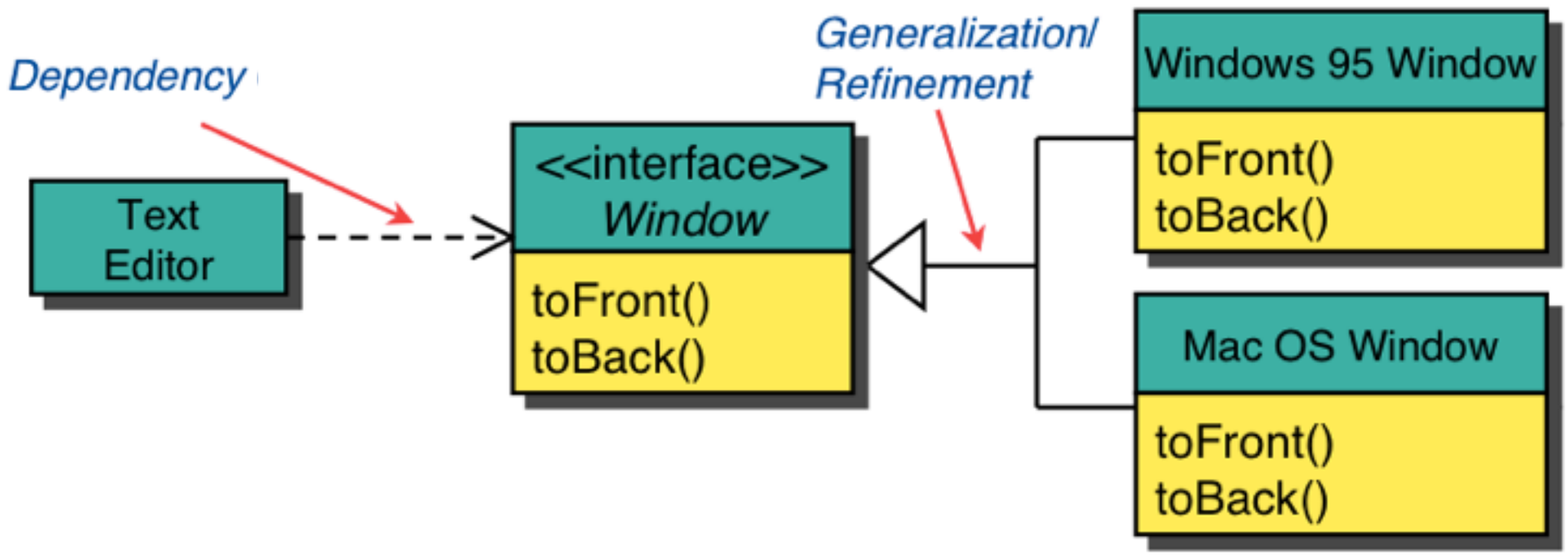
Abstract class - lớp trừu tượng

- **Lớp trừu tượng** là một lớp
  - **Phần cài đặt của một số phương thức** bị **bỏ qua**
  - Những phương thức bị bỏ qua chỉ được cài đặt tại các lớp con
- Ví dụ : Thao tác di chuyển cửa sổ được cài đặt sử dụng hai phương thức ẩn và hiện mà chúng được cài đặt phù hợp ở các lớp con



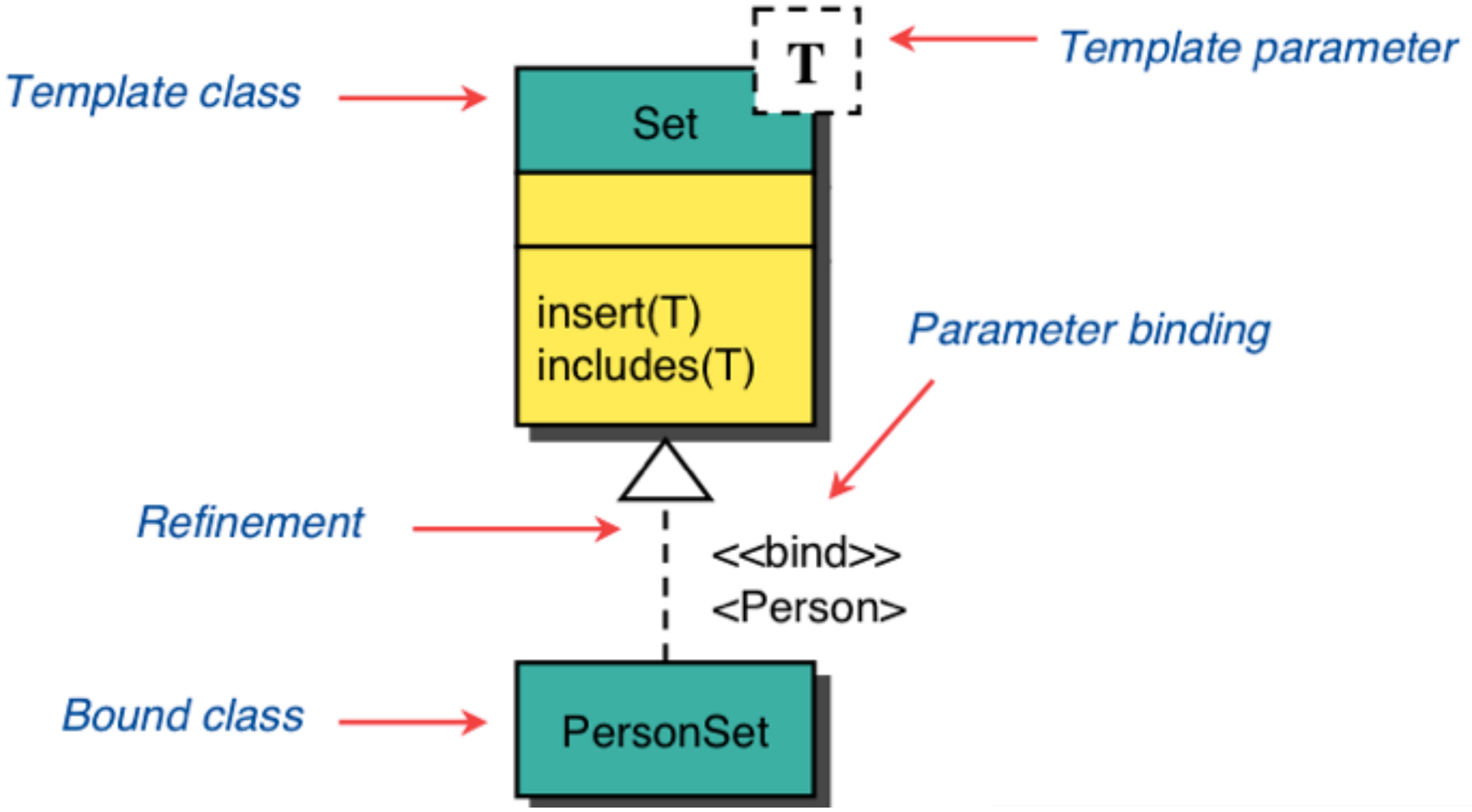
Interface - giao diện

- **Giao diện** là một lớp trừu tượng không **có cài đặt**
  - Giao diện được cài đặt thông qua các lớp khác
  - Các cài đặt có thể được thay đổi mà không cần những thay đổi của đối tượng sử dụng
- Ví dụ : Một chương trình xử lý văn bản hiển thị cửa sổ của nó sử dụng một giao diện cửa sổ mà nó được cài đặt khác nhau cho Windows 95 và Mac OS



Template class - khuôn hình

- Đôi khi bạn cần các lớp **chung** cho các thành phần của một kiểu nào đó, đặc biệt cho các **tập hợp**, vd : list, set,...
- Các lớp chung này ảo hóa từ các thành phần mà chúng làm việc, vd : tập các số nguyên, tập con người



Xác định lớp tiếp cận theo cụm danh từ

Xác định lớp

- Tách những đoạn ngắn súc tích từ yêu cầu
- Gạch chân các danh từ và cụm danh từ mà nó thể hiện một điều gì đó
  - Đây có thể là các *lớp ứng viên*
- Đối tượng hay không ?
  - Bên trong phạm vi hệ thống ?
  - Một sự kiện, tình trạng, khoảng thời gian ?
  - Một thuộc tính của đối tượng khác ?
  - Đồng nghĩa ?
- Lặp lại, nhìn lại tất cả một lần nữa

Ví dụ - hệ thống ATM

Tài khoản

Số dư tài khoản

Số tiền

Tiến trình đăng nhập

Thẻ ATM

Máy ATM

Ngân hàng

Khách hàng ngân hàng

Thẻ

Tiền mặt

Khách hàng

Tài khoản ngân hàng

VND

Bao thư

Bốn ký số

Ngân quỹ

Tiền

PIN

PIN không hợp lệ

Thông điệp

Mật khẩu

Mã PIN

Mẫu tin

Bước

Hệ thống

Giao dịch

Lịch sử giao dịch

Ví dụ - loại bỏ lớp giả

Tài khoản

Số dư tài khoản

Số tiền

Tiến trình đăng nhập

Thẻ ATM

Máy ATM

Ngân hàng

Khách hàng ngân hàng

Thẻ

Tiền mặt

Khách hàng

Tài khoản ngân hàng

VND

~~Bao thư~~

~~Bốn ký số~~

Ngân quỹ

Tiền

PIN

PIN không hợp lệ

Thông điệp

Mật khẩu

Mã PIN

Mẫu tin

~~Bước~~

Hệ thống

Giao dịch

Lịch sử giao dịch

Ví dụ - gộp các lớp trùng

Tài khoản

Số dư tài khoản

Số tiền

Tiến trình đăng nhập

Thẻ ATM

Máy ATM

Ngân hàng

~~Khách hàng ngân hàng~~

~~Thẻ~~

Tiền mặt

Khách hàng

~~Tài khoản ngân hàng~~

VND

Ngân quỹ

~~Tiền~~

PIN

PIN không hợp lệ

Thông điệp

Mật khẩu

~~Mã PIN~~

Mẫu tin

Hệ thống

Giao dịch

Lịch sử giao dịch

Ví dụ - loại bỏ đặc tính

Tài khoản

~~Số dư tài khoản~~

~~Số tiền~~

Tiến trình đăng nhập

Thẻ ATM

Máy ATM

Ngân hàng

Tiền mặt

Khách hàng

VND

Ngân quỹ

~~PIN~~

~~PIN không hợp lệ~~

Thông điệp

~~Mật khẩu~~

Mẫu tin

Hệ thống

Giao dịch

~~Lịch sử giao dịch~~

Ví dụ - thuộc phạm vi hệ thống

Tài khoản

~~Ngân quỹ~~

~~Tiến trình đăng nhập~~

Thẻ ATM

Máy ATM

Ngân hàng

~~Thông điệp~~

~~Tiền mặt~~

~~Mẫu tin~~

Khách hàng

~~Hệ thống~~

~~VND~~

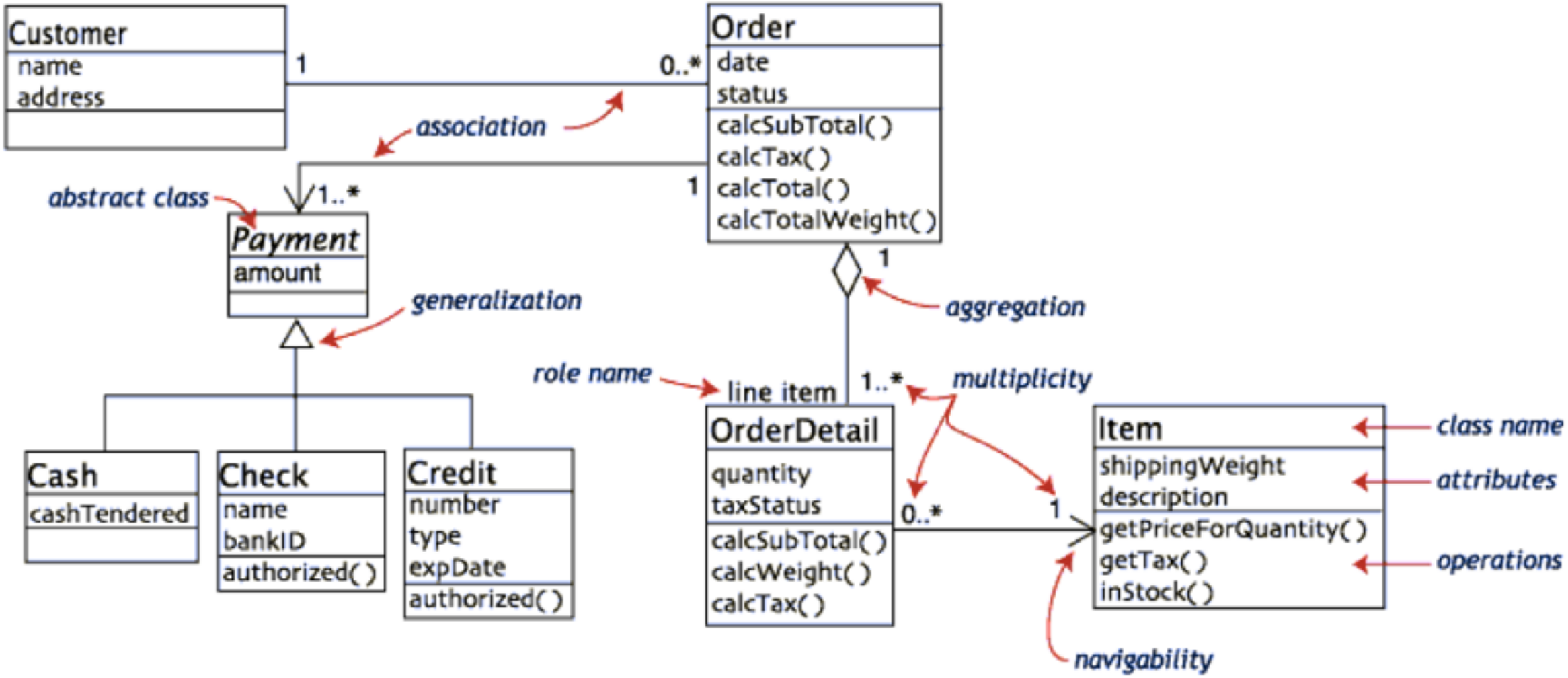
Giao dịch

Ví dụ - kết quả

Máy ATM	Thẻ ATM	Khách hàng
Ngân hàng	Tài khoản	Giao dịch

Ví dụ

Ví dụ - đơn hàng

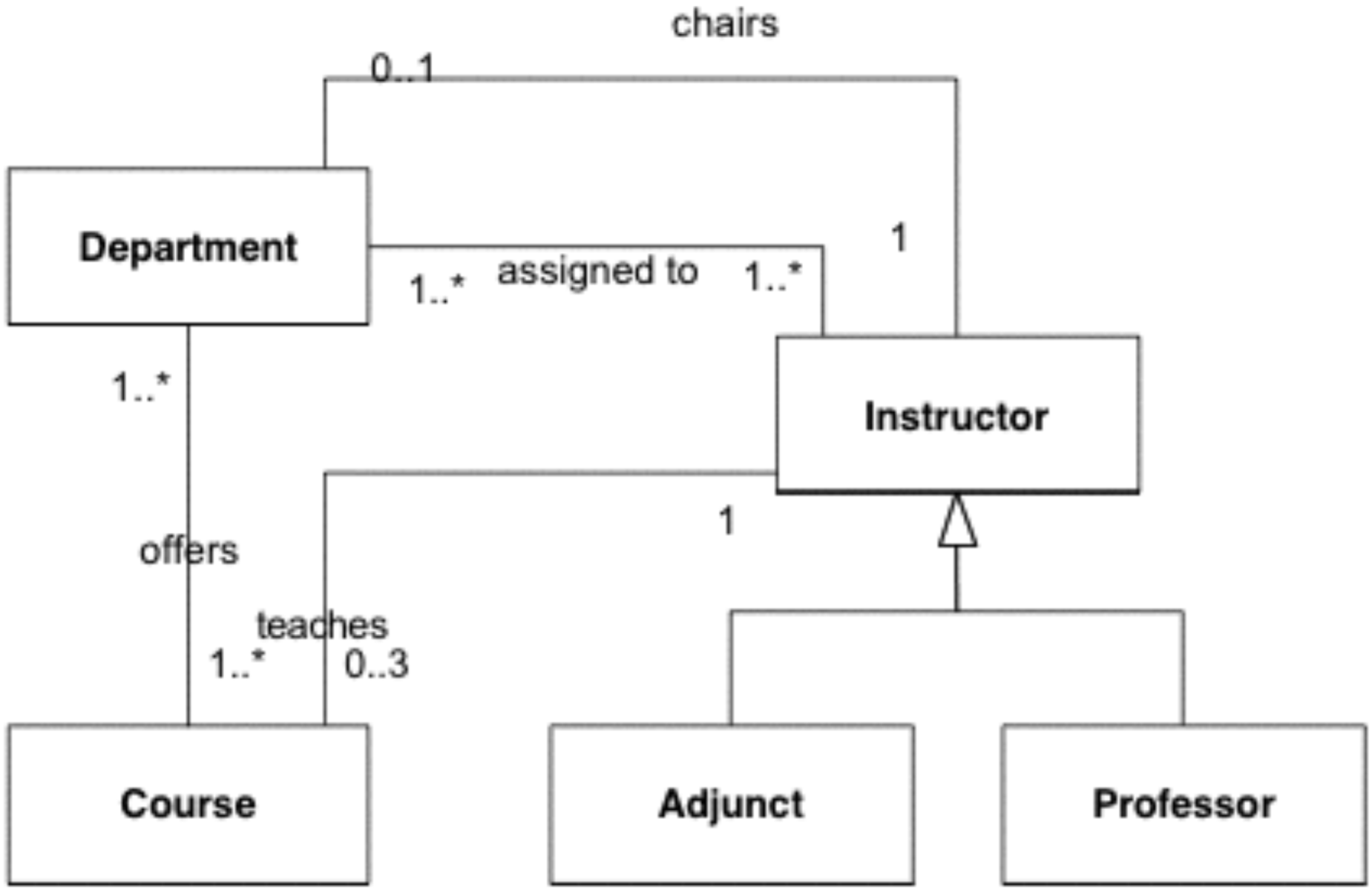


Mô hình lớp trên thể hiện một đơn hàng của khách hàng từ cửa hàng. Lớp trung tâm là **Order**. Kết hợp với nó là lớp **Customer**, thực hiện mua hàng và **Payment**. Một **Payment** có ba kiểu : **Cash**, **Check** hay **Credit**. Đơn hàng này chứa OrderDetails mà mỗi cái tương ứng với một Item.

Ví dụ - môn học

- Một số instructor là professor, số còn lại là adjunct
- Các khoa đưa ra nhiều môn học, nhưng mỗi môn học có thể được đưa ra do >1 khoa
- Các môn học được dạy bởi instructor và chỉ được dạy nhiều nhất 3 môn
- Một instructor có thể thuộc về một hoặc nhiều khoa
- Một instructor nào đó có thể là trưởng khoa

Ví dụ


chairs
0..1
Department
1..*
assigned to
1..*
1
1..*
Instructor
offers
1
teaches
1..*
0..3
Course
Adjunct
Professor

Cảm ơn sự chú ý

Câu hỏi ?