

# Ngôn ngữ mô hình hóa UML

## Chương 3

---

# Sơ đồ lớp (Class Diagram)

TS. Nguyễn Thanh Hải

---

# Nội dung

- Giới thiệu
- Đối tượng và sơ đồ đối tượng
- Lớp, thuộc tính và thao tác/phương thức
- Quan hệ giữa các lớp
- Ví dụ

# Giới thiệu

- Class Diagram thể hiện hệ thống một cách tổng quan ở khía cạnh tĩnh.
- Class Diagram biểu diễn các class và các mối quan hệ giữa chúng, và các thuộc tính của lớp, các phương thức
- Class Diagram là sơ đồ **quan trọng** trong số các sơ đồ trong phương pháp hướng đối tượng.

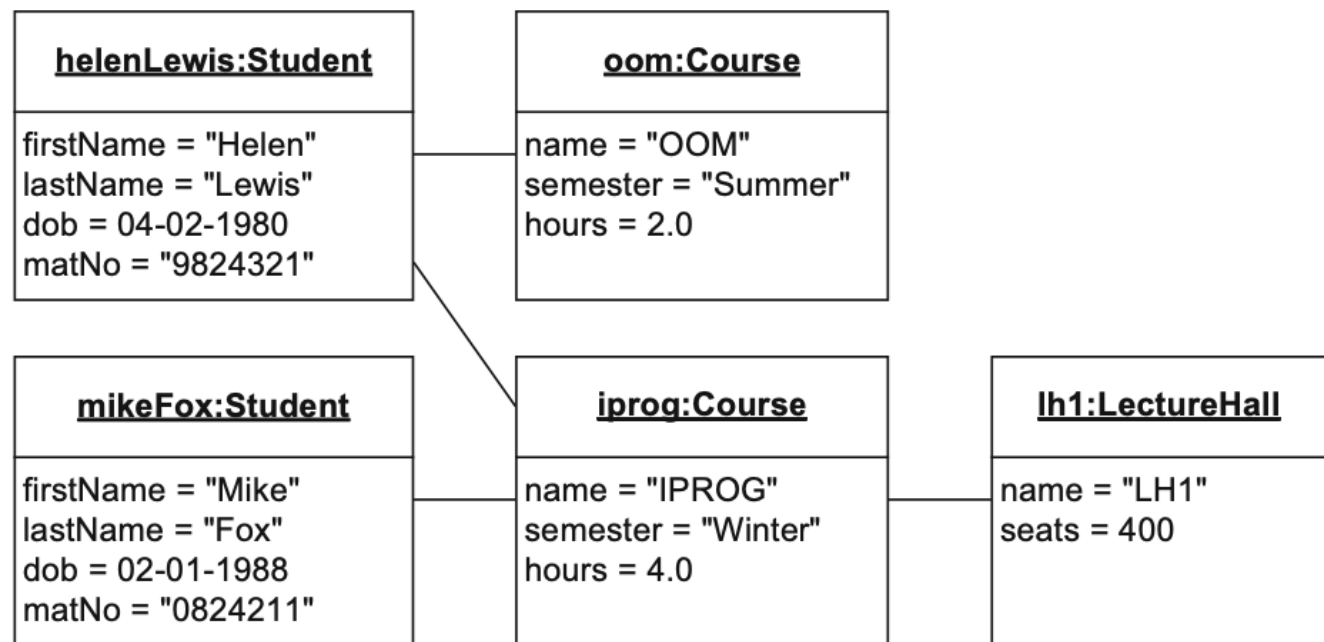
# Đối tượng và Sơ đồ đối tượng

Một hệ thống có thể chứa nhiều phần tử đơn lẻ khác nhau (có thể là người, động vật, cây trồng, đồ vật, đối tượng...) được xác định duy nhất.

VD: helenLewis dự lớp Object-Oriented Modeling (OOM) ở trường.  
helenLewis, OOM là các đối tượng có mối quan hệ với nhau

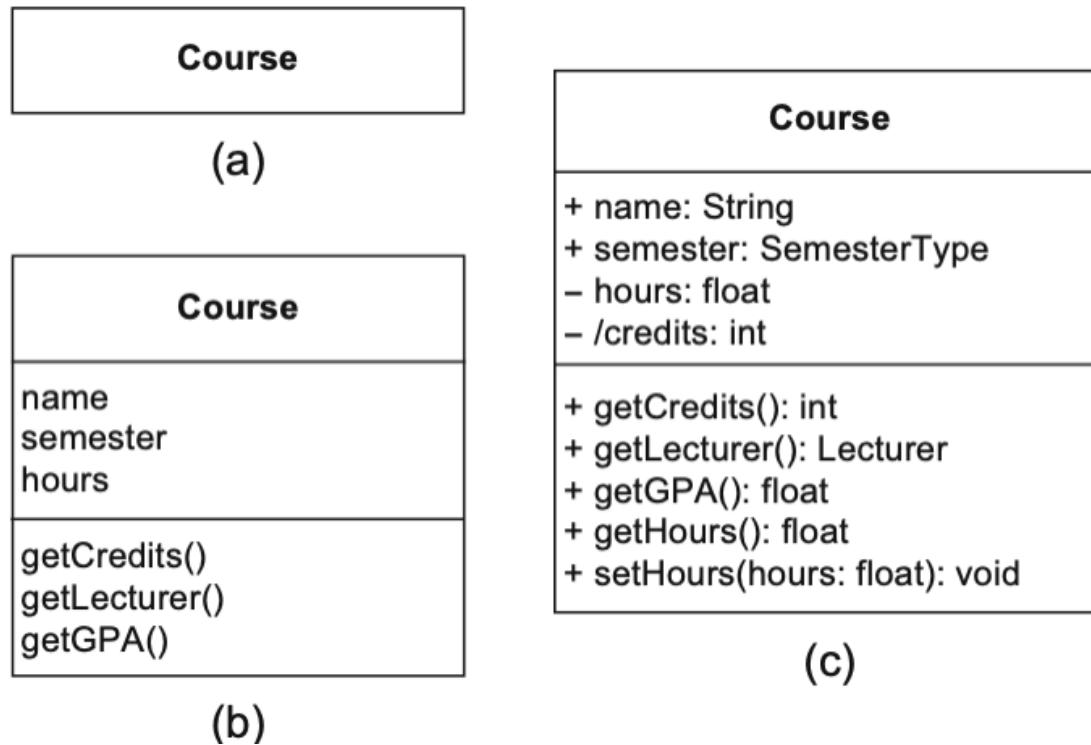
**Figure 4.1**

Example of an object diagram



# Lớp, thuộc tính và thao tác/Phương thức

**Lớp (class)** : Một lớp biểu diễn cho sự mô tả trừu tượng một tập các đối tượng có cùng tính chất. Một đối tượng (**object**) là một thể hiện (**instance**) của một lớp (**class**).

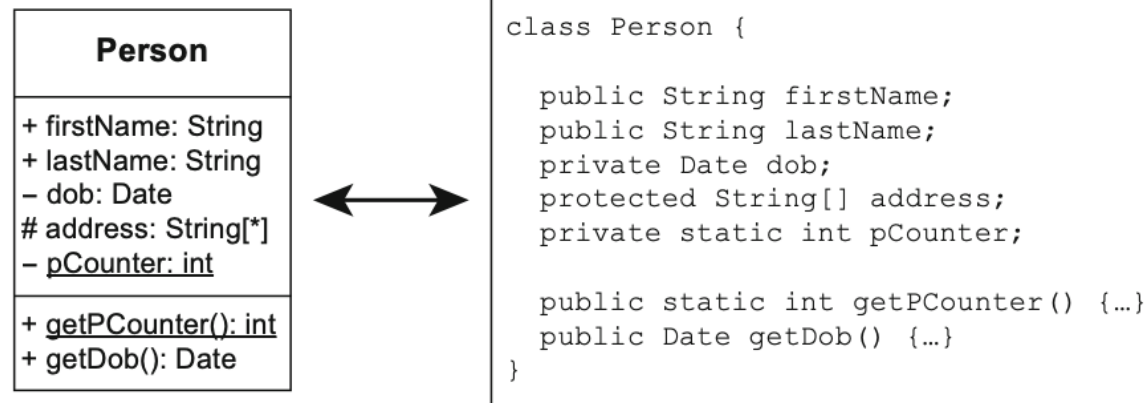


**Figure 4.4**  
Representation of a class  
and its characteristics

# Lớp, thuộc tính và thao tác/Phương thức

- ✦ **Thuộc tính (attribute):** một thuộc tính biểu diễn cho một kiểu thông tin bên trong một lớp.
  - ✦ Cú pháp như sau:  
*visibility name: type=default Value*

**Figure 4.10**  
Translation of a class from  
UML to Java



Name	Symbol	Description
public	+	Access by objects of any classes permitted
private	-	Access only within the object itself permitted
protected	#	Access by objects of the same class and its subclasses permitted
package	~	Access by objects whose classes are in the same package permitted

# Lớp, thuộc tính và thao tác/Phương thức

## ✦ Thao tác/phương thức (Operation):

- ✦ Là chức năng hoặc hành vi của một lớp.
- ✦ Cú pháp của các thao tác :

*visibility name (parameter-list): return-type-expression*

Trong đó:

✦ *visibility*

☞ + : public

☞ - : private

☞ # : protected

✦ *name*: tên thao tác

✦ *parameter-list* : các tham số, khai báo như thuộc tính

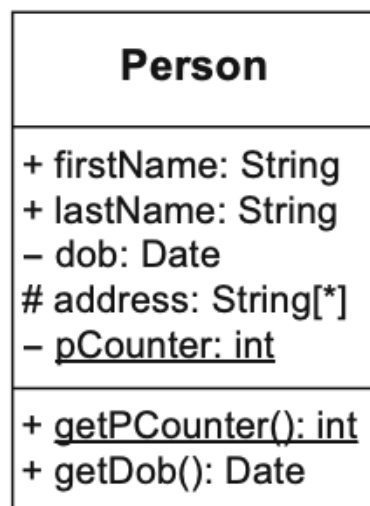
✦ *return-type-expression* : kiểu trả về

Document
+ open() + new()

# Lớp, thuộc tính và thao tác/Phương thức

## ✦ Class Variables và Class Operations:

- ✦ Thuộc tính trong 1 lớp gồm:
  - Thuộc tính thể hiện (**instance attributes**)
  - Thuộc tính lớp (**class attributes**) hay thuộc tính tĩnh (**static attributes**) là những thuộc tính được gạch dưới (pCounter: đếm số người, số thể hiện trong Class).
- ✦ Phương thức lớp/Phương thức tĩnh (**Class Operation/Static Operation**): được gọi thông qua Class, không thông qua các thể hiện

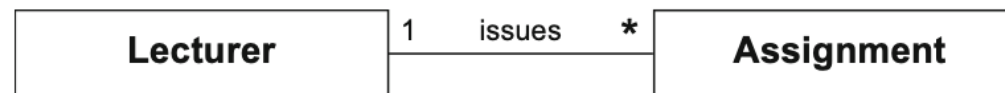


```
class Person {  
  
    public String firstName;  
    public String lastName;  
    private Date dob;  
    protected String[] address;  
    private static int pCounter;  
  
    public static int getPCounter() {...}  
    public Date getDob() {...}  
}
```



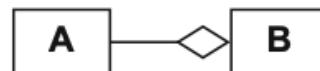
# Quan hệ giữa các lớp

- **Quan hệ kết hợp (association) giữa các lớp:** Một mối kết hợp biểu diễn một quan hệ ngữ nghĩa giữa các lớp.



- **Quan hệ kết tập- tụ hợp (aggregation):** biểu diễn mối quan hệ toàn thể (whole)-bộ phận (part) giữa 2 lớp.
  - Là một trường hợp đặc biệt của association.
  - Được dùng để cho biết một thể hiện của một lớp có thể chứa các thể hiện của một lớp khác.
  - Gồm: **shared aggregation** và **composition**

*Shared aggregation*



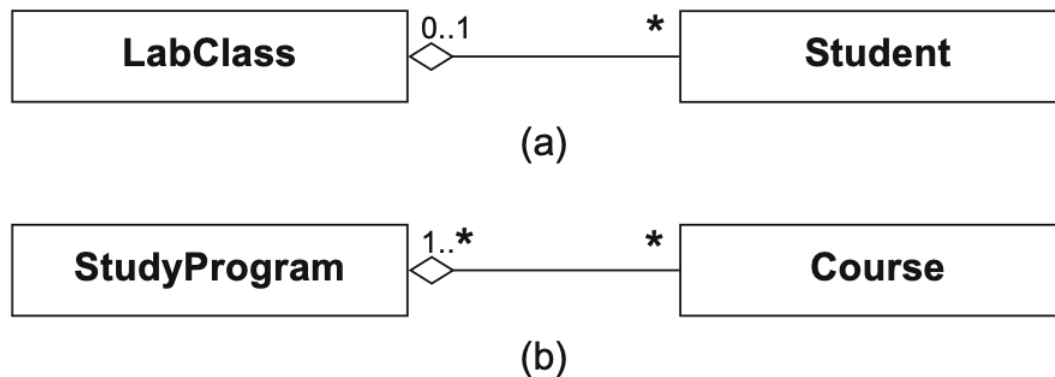
*Composition*



# Quan hệ giữa các lớp

- **Shared aggregation (kết tập chia sẻ):** thể hiện sự phụ thuộc “yếu” của bộ phận (part) đến các toàn thể (whole), nghĩa là các part (vd: Course) có thể **tồn tại độc lập** không phụ thuộc vào whole (StudyProgram)

**Figure 4.21**  
Examples of shared aggregations



## Quan hệ giữa các lớp

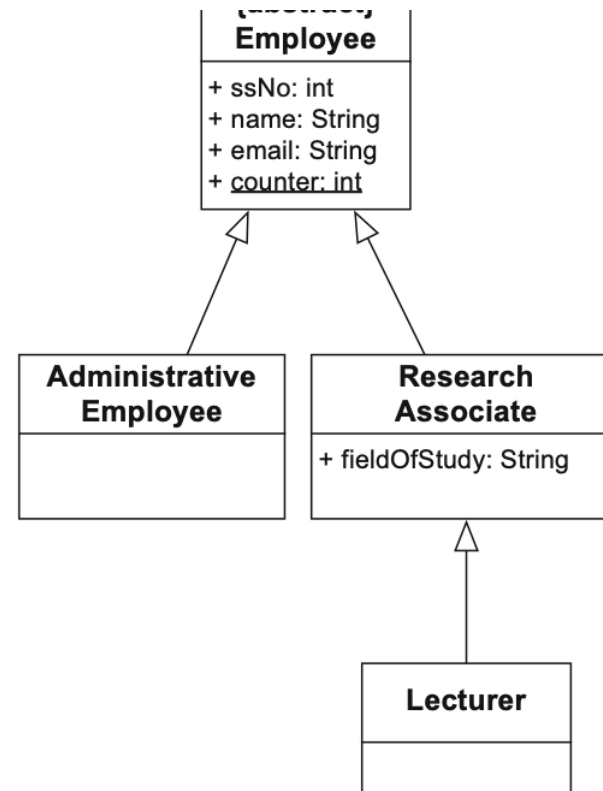
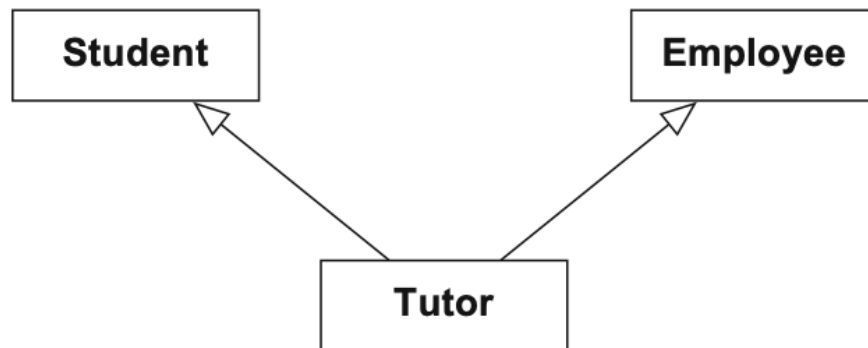
- ✦ **Quan hệ hợp thành (composition):** là một hình thức mạnh hơn của aggregation, nghĩa là: part (LectureHall) là tồn tại **phụ thuộc** vào whole, nếu whole bị xóa thì các part cũng sẽ xóa theo



# Quan hệ giữa các lớp

## ✦ Quan hệ tổng quát hóa (generalization), lớp cha (superclass), lớp con (subclass)

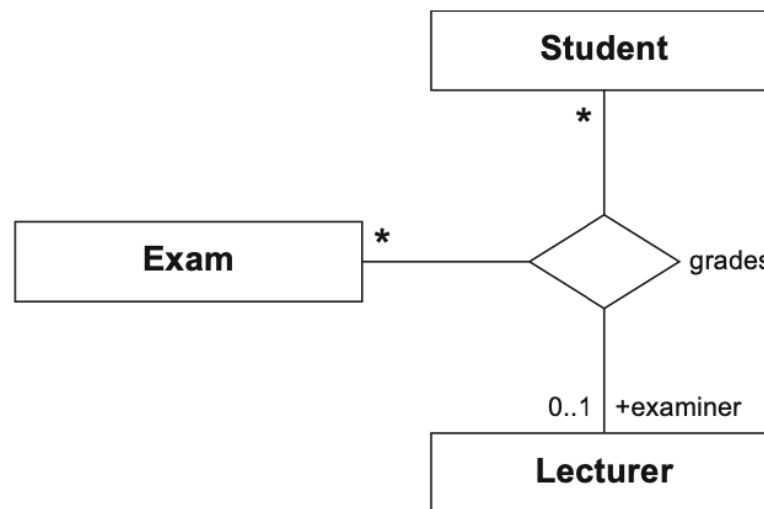
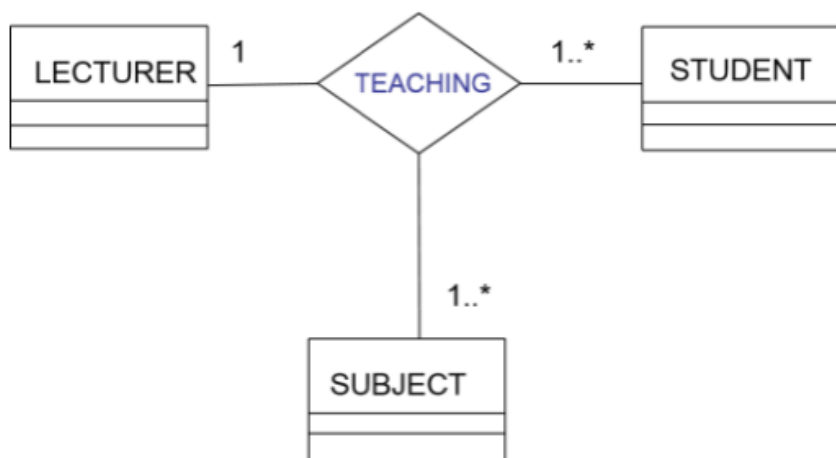
- ✦ Một lớp cha là một lớp tổng quát hơn kết nối với một hoặc nhiều lớp ở mức cụ thể hơn (lớp con) bởi một quan hệ tổng quát hóa.
- ✦ Lớp con kế thừa các tính chất của lớp cha của nó và có thể có những tính chất riêng của nó.



# Quan hệ giữa các lớp

## Kết hợp n-ngôi (n-ary association)

Biểu diễn cho các mối kết hợp có nhiều hơn hai lớp.

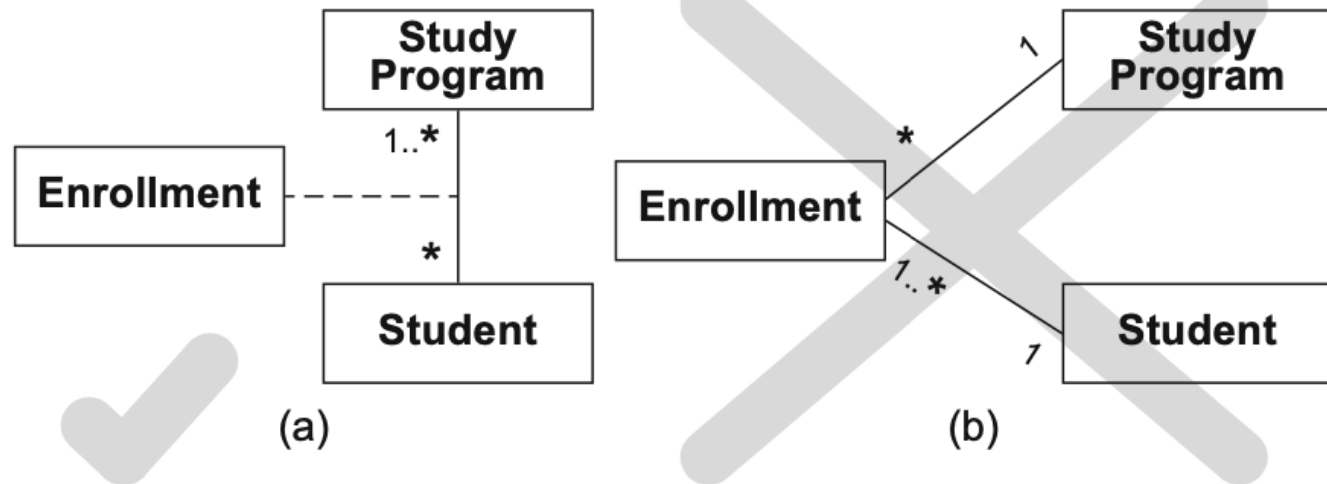


# Lớp kết hợp (association class)

**Lớp liên kết- lớp kết hợp:** Biểu diễn sự kết hợp như một lớp, nghĩa là sự kết hợp cũng có thuộc tính và thao tác

**Figure 4.19**

Attempt to model an association class with a “normal” class and corresponding relationships



# Abstract Class: Lớp trừu tượng

Lớp không thể có được thể hiện của chính nó gọi là Lớp trừu tượng (**abstract class**). Bản thân lớp không có các đối tượng, chỉ lớp con (**subclass**) của nó có thể có. Lớp trừu tượng để nhấn mạnh các đặc tính của lớp con trong ngữ cảnh “tổng quát hóa” (generalization)

***Person***

**{abstract}  
Person**

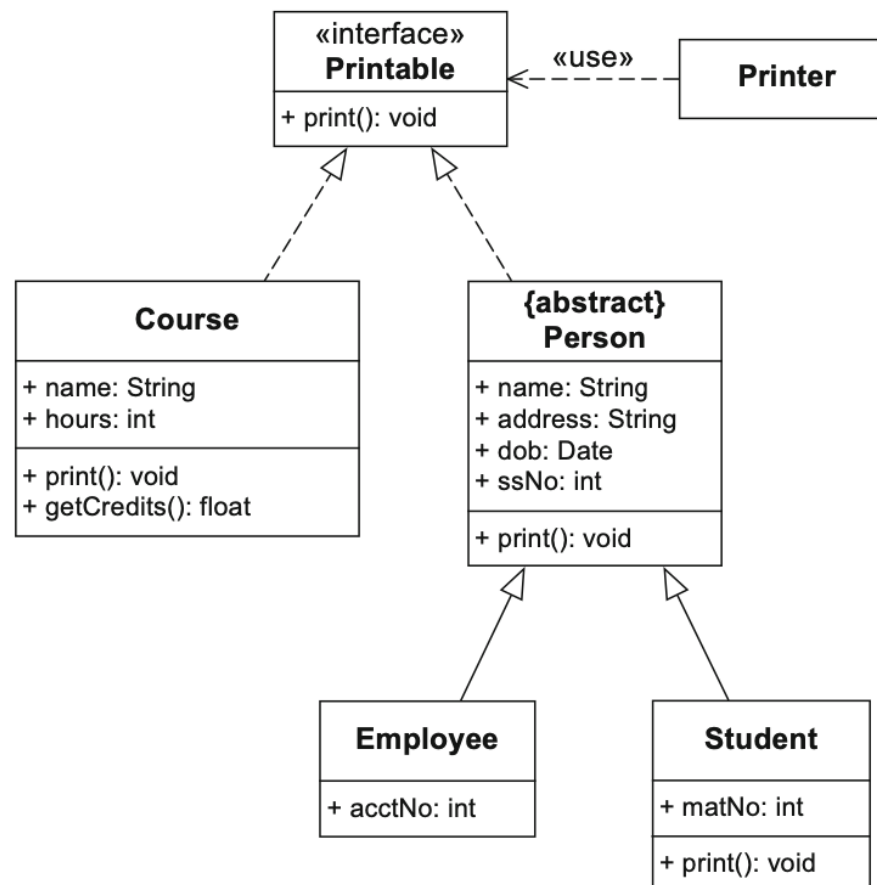
**Figure 4.27**  
Notation for abstract  
classes

# Interface

Như abstract Class, Interface không có các thể hiện trực tiếp. Các lớp cài đặt Interface thì bắt buộc chúng có những hành vi (phương thức) cụ thể cho Interface

Interface thường biểu diễn như một Class, thêm từ “interface” hoặc dùng ký hiệu riêng của phần mềm

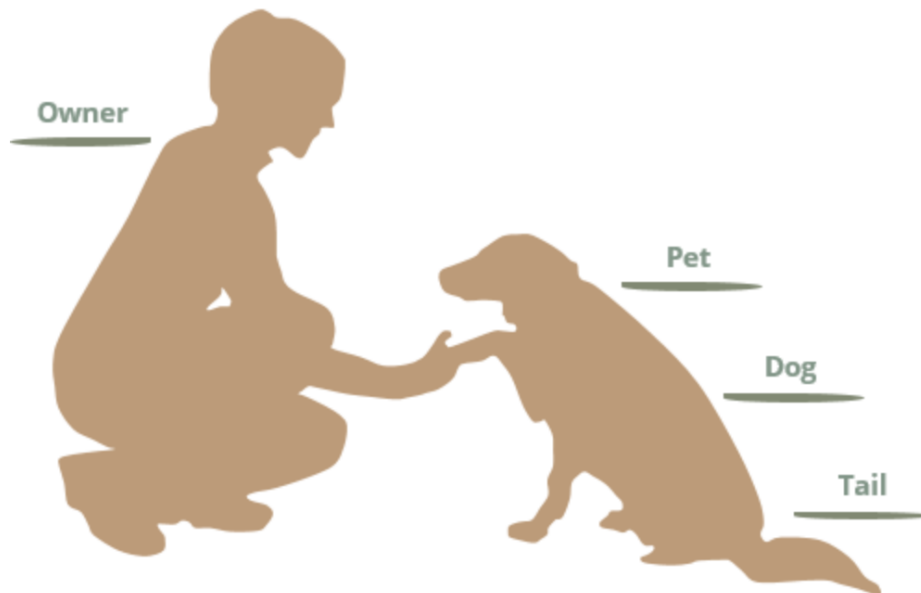
**Figure 4.28**  
Example of an interface





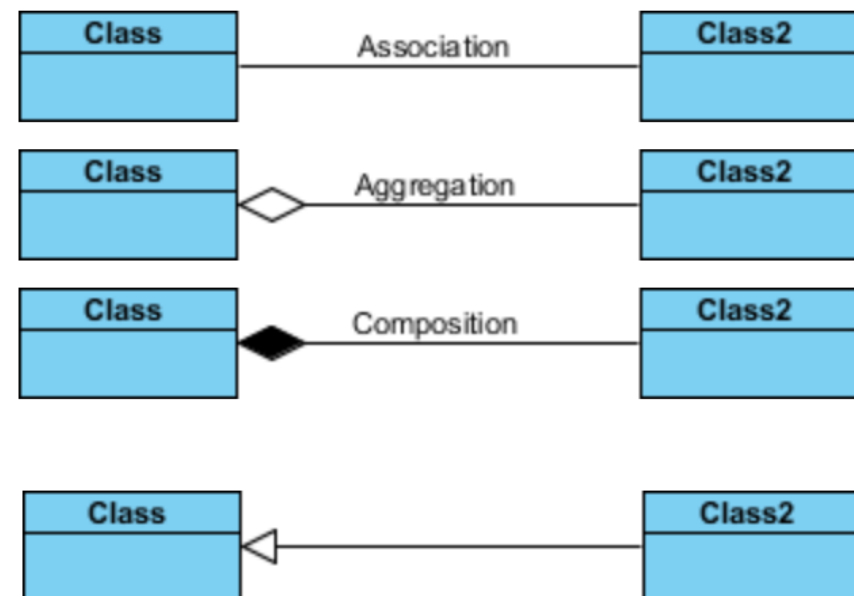
# UML Association vs Aggregation vs Composition

## Association • Aggregation • Composition

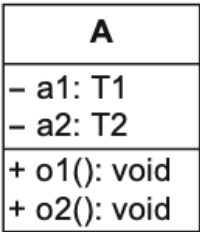

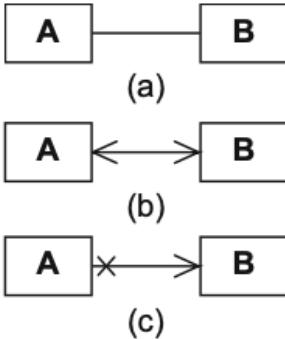
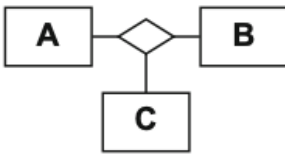
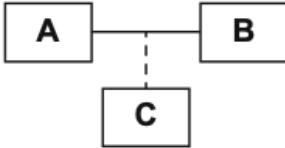


We see the following relationships:


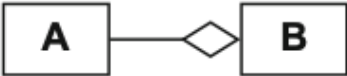
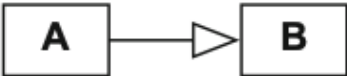
- owners feed pets, pets please owners (association)
- a tail is a part of both dogs and cats (aggregation / composition)
- a cat is a kind of pet (inheritance / generalization)



# Tổng hợp các ký hiệu

Name	Notation	Description
Class		Description of the structure and behavior of a set of objects
Abstract class		Class that cannot be instantiated
Association		Relationship between classes: navigability unspecified (a), navigable in both directions (b), not navigable in one direction (c)
N-ary association		Relationship between $N$ (in this case 3) classes
Association class		More detailed description of an association

# Tổng hợp các ký hiệu

Strong aggregation = composition		Existence-dependent parts-whole relationship (A is part of B; if B is deleted, related instances of A are also deleted)
Shared aggregation		Parts-whole relationship (A is part of B; if B is deleted, related instances of A need not be deleted)
Generalization		Inheritance relationship (A inherits from B)

# So sánh với Entity-Relationship diagram

Sơ đồ Class có nhiều nét tương đồng với E-R diagram:

- Đều thể hiện các thành phần của hệ thống (Classes và Entities) và mối quan hệ giữa chúng
- Mỗi thành phần được đặc tả bởi các thuộc tính

Chúng cũng có sự khác biệt:

- E-R diagram mô tả các thành phần của một database
- Class diagram thể hiện sự cài đặt hệ thống đã được mô hình hóa bằng ngôn ngữ lập trình hướng đối tượng
- E-R diagram yêu cầu xác định khóa thực thể, nhưng Class thì không cần thiết
- Hành vi của đối tượng được định nghĩa qua các phương thức, E-R diagram không hỗ trợ.

# Xây dựng sơ đồ lớp

- Danh từ (như student, lecturer, course,...) thường chỉ các lớp
- Các thuộc tính thường biểu diễn bằng tính từ hoặc danh từ
- Phương thức thường mô tả bằng động từ

(tr. 93 giáo trình tiếng Việt)

- Tìm lớp/thuộc tính: Xác định những thành phần cần lưu trữ, và tương tác với hệ thống (cần thiết bị ngoại vi nào, tương tác với hệ thống nào,...), tìm trong các tác nhân, tìm danh từ trong các luồng sự kiện của các Use Case- UC
- Tìm phương thức: Tìm các động từ trong luồng sự kiện của UC
- Chưa có quy tắc chung cho việc tách lớp và các thuộc tính!

# Xây dựng sơ đồ lớp

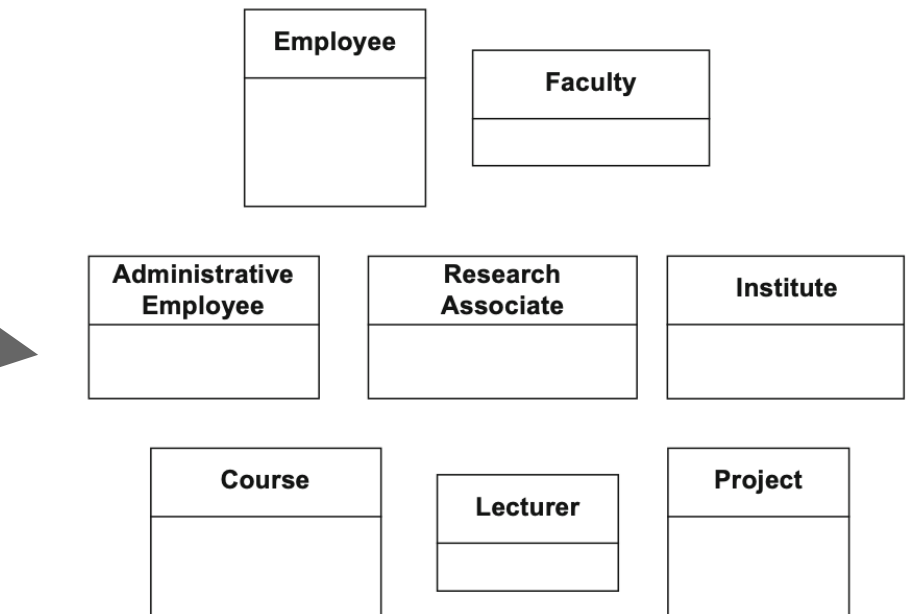
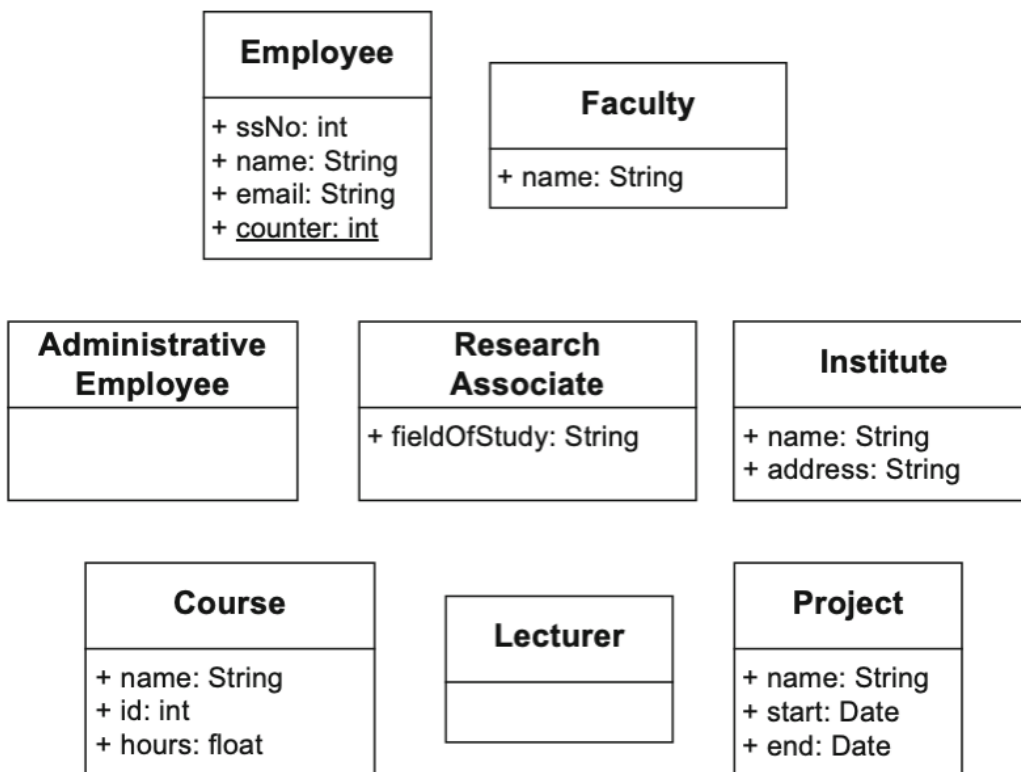
Vd: Bài tập tạo sơ đồ class cơ bản cho hệ thống quản lý một trường đại học:

*Information system of a university*

- A university consists of multiple faculties which are composed of various institutes. Each faculty and each institute has a name. An address is known for each institute.
- Each faculty is led by a dean, who is an employee of the university.
- The total number of employees is known. Employees have a social security number, a name, and an e-mail address. There is a distinction between research and administrative personnel.
- Research associates are assigned to at least one institute. The field of study of each research associate is known. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. Some research associates teach courses. They are called lecturers.
- Courses have a unique number (ID), a name, and a weekly duration in hours.

# Xây dựng sơ đồ lớp

## Bước 1: Xác định các lớp

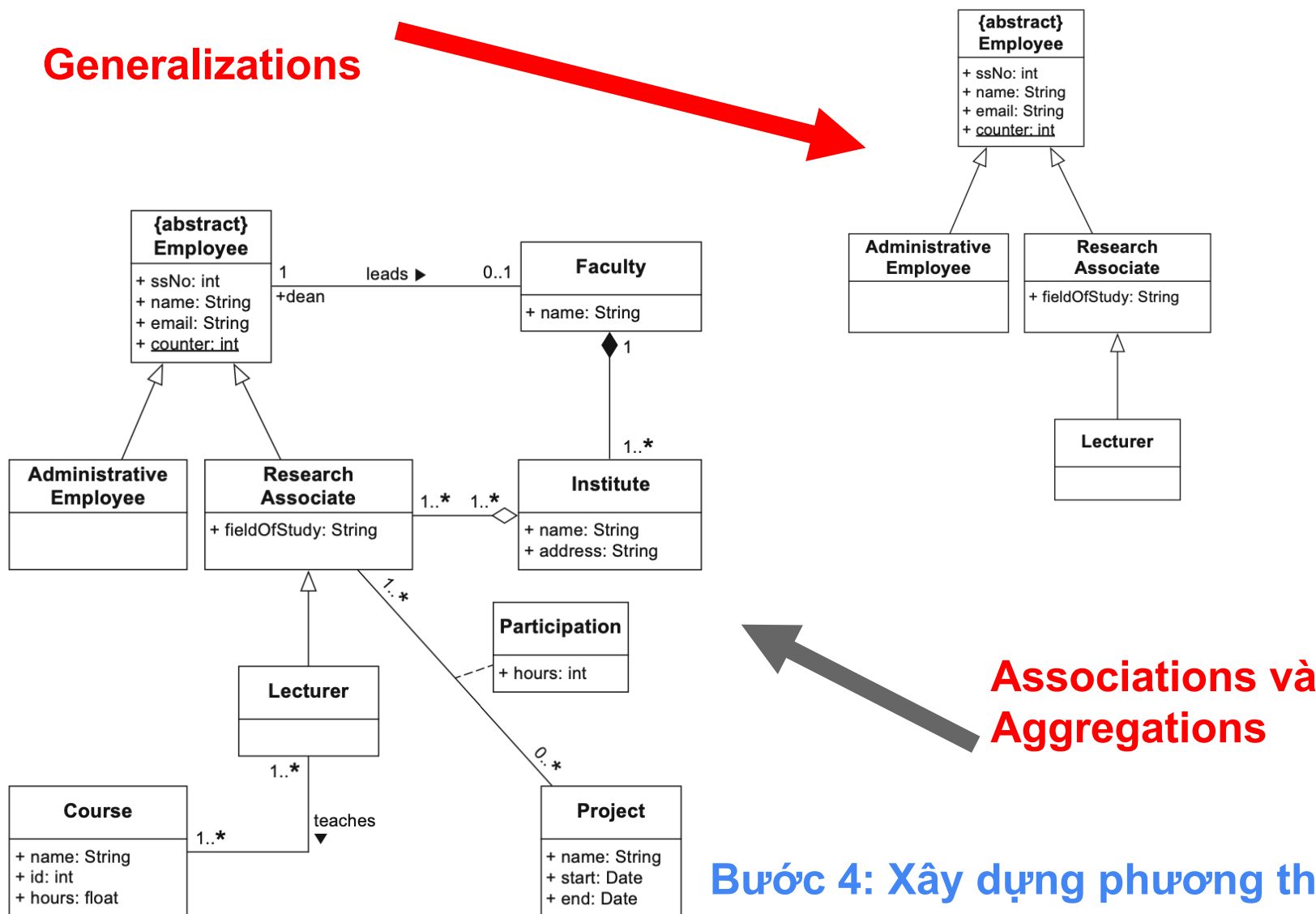


## Bước 2: Xác định các thuộc tính

# Xây dựng sơ đồ lớp

## Bước 3: Xác định các quan hệ các mối kết hợp

### Generalizations



**Figure 4.32**  
Identified generalization  
relationships

## Bước 4: Xây dựng phương thức cho lớp



# Tài liệu

**Martina Seidl et al. UML @ Classroom: An Introduction to Object-Oriented Modeling. ISBN:978-3-319-12741-5.**

**Đặng Văn Đức. Phân tích và thiết kế hệ thống hướng đối tượng.**