

**BỘ MÔN CÔNG NGHỆ PHẦN MỀM**  
**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**  
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# **LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**


## **Bài 12. Biểu đồ lớp**

Nguyễn Thị Thu Trang  
[trangntt-fit@mail.hut.edu.vn](mailto:trangntt-fit@mail.hut.edu.vn)

# Mục đích

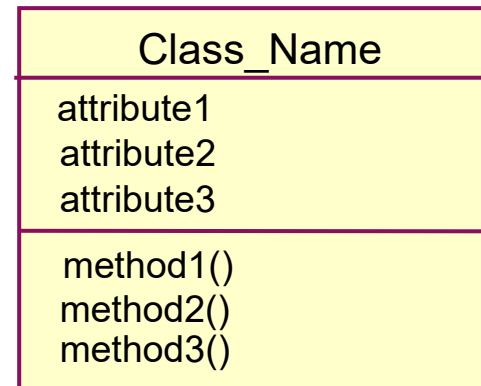
- Mô tả khung nhìn tĩnh của hệ thống và cách đưa nó vào trong một mô hình.
- Minh họa cách đọc và hiểu một biểu đồ lớp.
- Mô hình hóa mối liên kết (association) và kết tập (aggregation) và chỉ ra cách mô hình chúng vào biểu đồ lớp.
- Mô hình tổng quát hóa (generalization) trên một biểu đồ lớp.

# Nội dung

- 
- ⇒ 1. Biểu đồ lớp (Class diagram)
  - 2. Liên kết (Association)
  - 3. Kết tập (Aggregation)
  - 4. Tổng quát hóa (Generalization)

# 1.1. Lớp (Class)

- Sử dụng hình chữ nhật gồm 3 thành phần
  - Tên lớp
  - Các thuộc tính
  - Các phương thức



# Biểu diễn thuộc tính

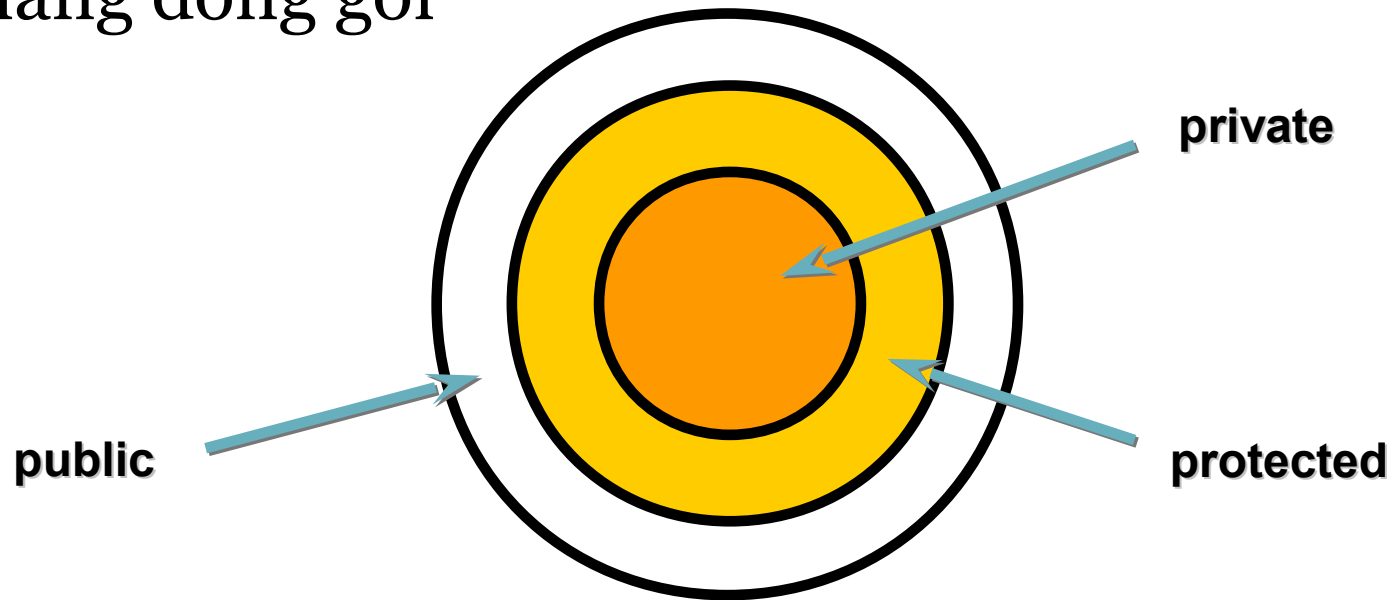
- Chỉ ra tên, kiểu và giá trị mặc định nếu có
  - `attributeName : Type = Default`
- Tuân theo quy ước đặt tên của ngôn ngữ cài đặt và của dự án.
- Kiểu (type) nên là kiểu dữ liệu cơ bản trong ngôn ngữ thực thi
  - Kiểu dữ liệu có sẵn, kiểu dữ liệu người dùng định nghĩa, hoặc lớp tự định nghĩa.

# Mô tả phương thức

- Tên phương thức:
  - Mô tả kết quả
  - Sử dụng góc nhìn của đối tượng khách (client – đối tượng gọi)
  - Nhất quán giữa các lớp
- Chữ ký của phương thức:  
**operationName([direction] parameter:class,...):returnType**
  - Direction: **in** (mặc định), **out** hoặc **inout**

# Phạm vi truy cập (Visibility)

- Phạm vi truy cập được sử dụng để thực hiện khả năng đóng gói



# Phạm vi truy cập được biểu diễn như thế nào?

- Các ký hiệu sau được sử dụng:

- + Public access
- # Protected access
- - Private access

Class1
- privateAttribute + publicAttribute # protectedAttribute
- privateOperation () + publicOperation () # protecteOperation ()



# Phạm vi (Scope)

- Xác định số lượng thể hiện của thuộc tính/thao tác:
  - Instance: Một thể hiện cho mỗi thể hiện của mỗi lớp
  - Classifier: Một thể hiện cho tất cả các thể hiện của lớp
- Phạm vi Classifier được ký hiệu bằng cách gạch dưới tên thuộc tính/thao tác.

Class1
<u>- classifierScopeAttr</u> - instanceScopeAttr
+ <u>classifierScopeOp ()</u> + instanceScopeOp ()

# Ví dụ: Scope

## Student

- name
- address
- studentID
- nextAvailID : int

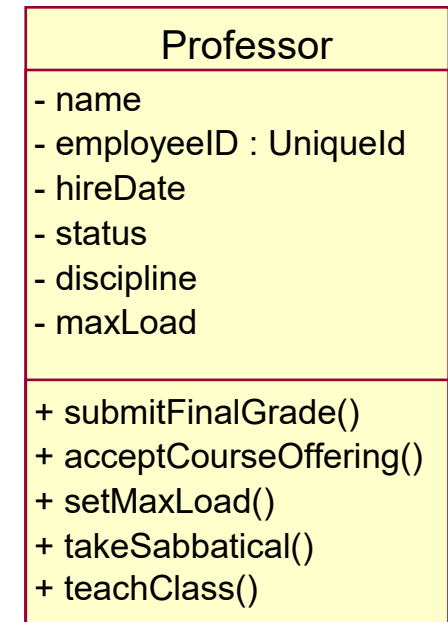
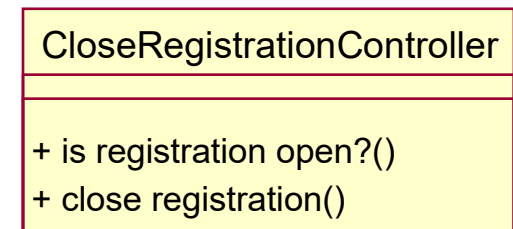
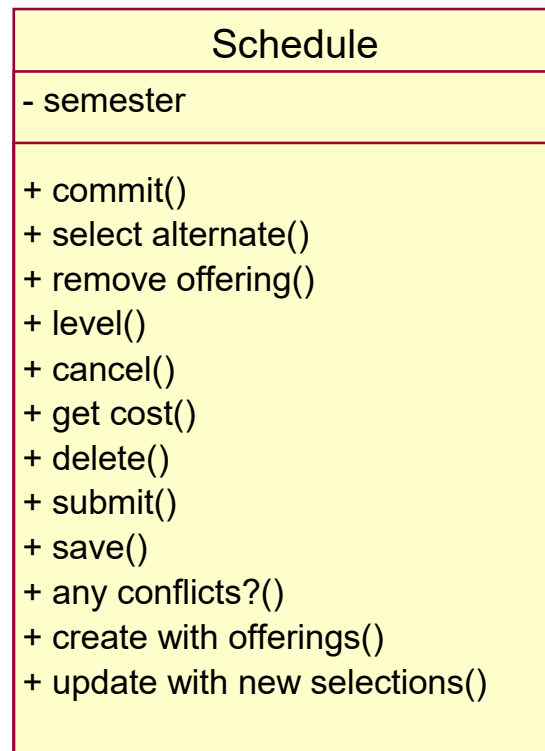
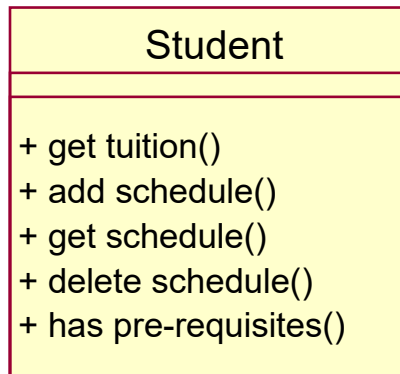
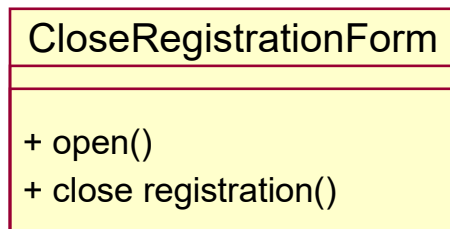
- + addSchedule ([in] theSchedule : Schedule, [in] forSemester : Semester)
- + getSchedule ([in] forSemester : Semester) : Schedule
- + hasPrerequisites ([in] forCourseOffering : CourseOffering) : boolean
- # passed ([in] theCourseOffering : CourseOffering) : boolean
- + getNextAvailID () : int

## 1.2. Biểu đồ lớp là gì?

- Biểu đồ lớp chỉ ra sự tồn tại của các lớp và mối quan hệ giữa chúng trong bản thiết kế logic của một hệ thống
  - Chỉ ra cấu trúc tĩnh của mô hình như lớp, cấu trúc bên trong của chúng và mối quan hệ với các lớp khác.
  - Chỉ ra tất cả hoặc một phần cấu trúc lớp của một hệ thống.
  - Không đưa ra các thông tin tạm thời.
- Khung nhìn tĩnh của một hệ thống chủ yếu hỗ trợ các yêu cầu chức năng của hệ thống.

# Biểu đồ lớp (Class Diagram – CD)

- Khung nhìn tĩnh của hệ thống

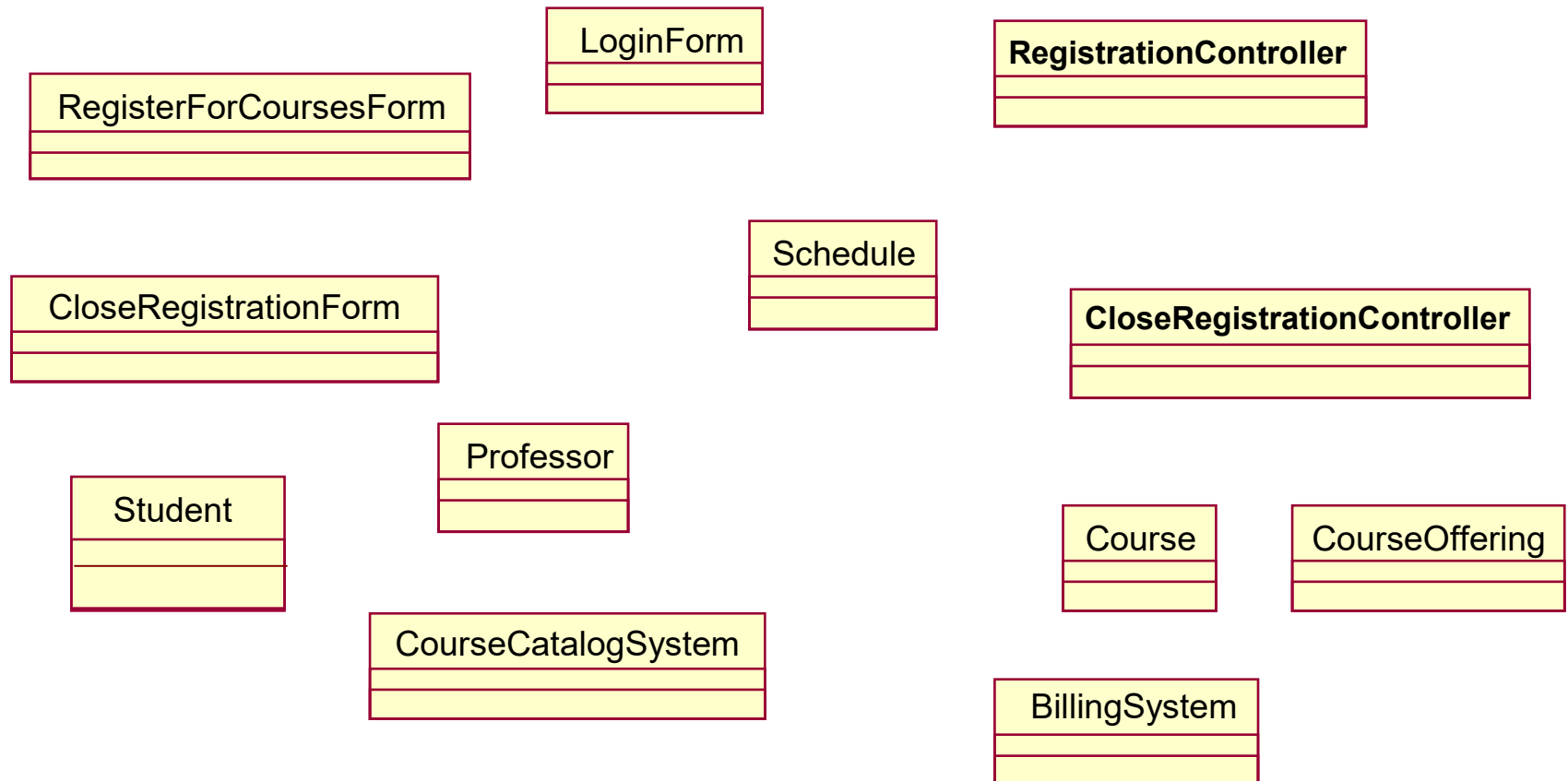


# Khi nào sử dụng biểu đồ lớp?

- Từ vựng của hệ thống (Vocabulary)
  - Khi trừu tượng hóa một phần hoặc bên ngoài hoặc biên của hệ thống.
  - Chỉ ra kết quả trừu tượng hóa và trách nhiệm của chúng
- Cộng tác (Collaboration)
  - Nhóm các lớp và các thành phần khác làm việc cùng nhau để thực hiện một công việc nào đó.
- Lược đồ CSDL logic (Logical database schema)
  - Tương tự như bản thiết kế khái niệm cho CSDL
  - Chứa các đối tượng cần lưu trữ lâu dài tức là cần lưu trong CSDL

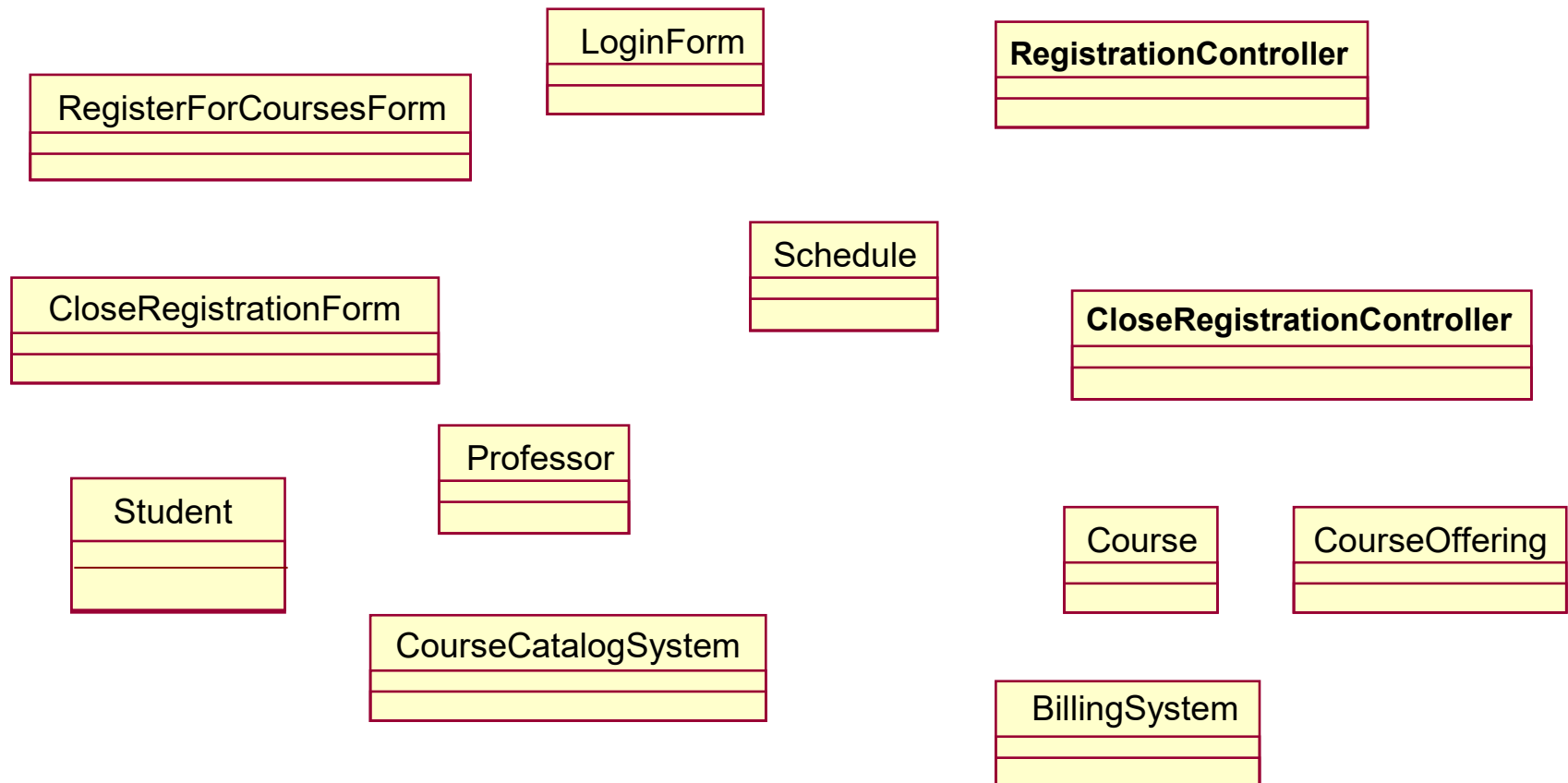
# Ví dụ Biểu đồ lớp

- Có cách nào tốt hơn để tổ chức biểu đồ lớp?



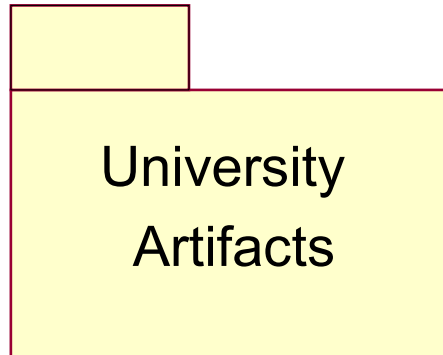
# Ví dụ Biểu đồ lớp

- Có cách nào tốt hơn để tổ chức biểu đồ lớp?



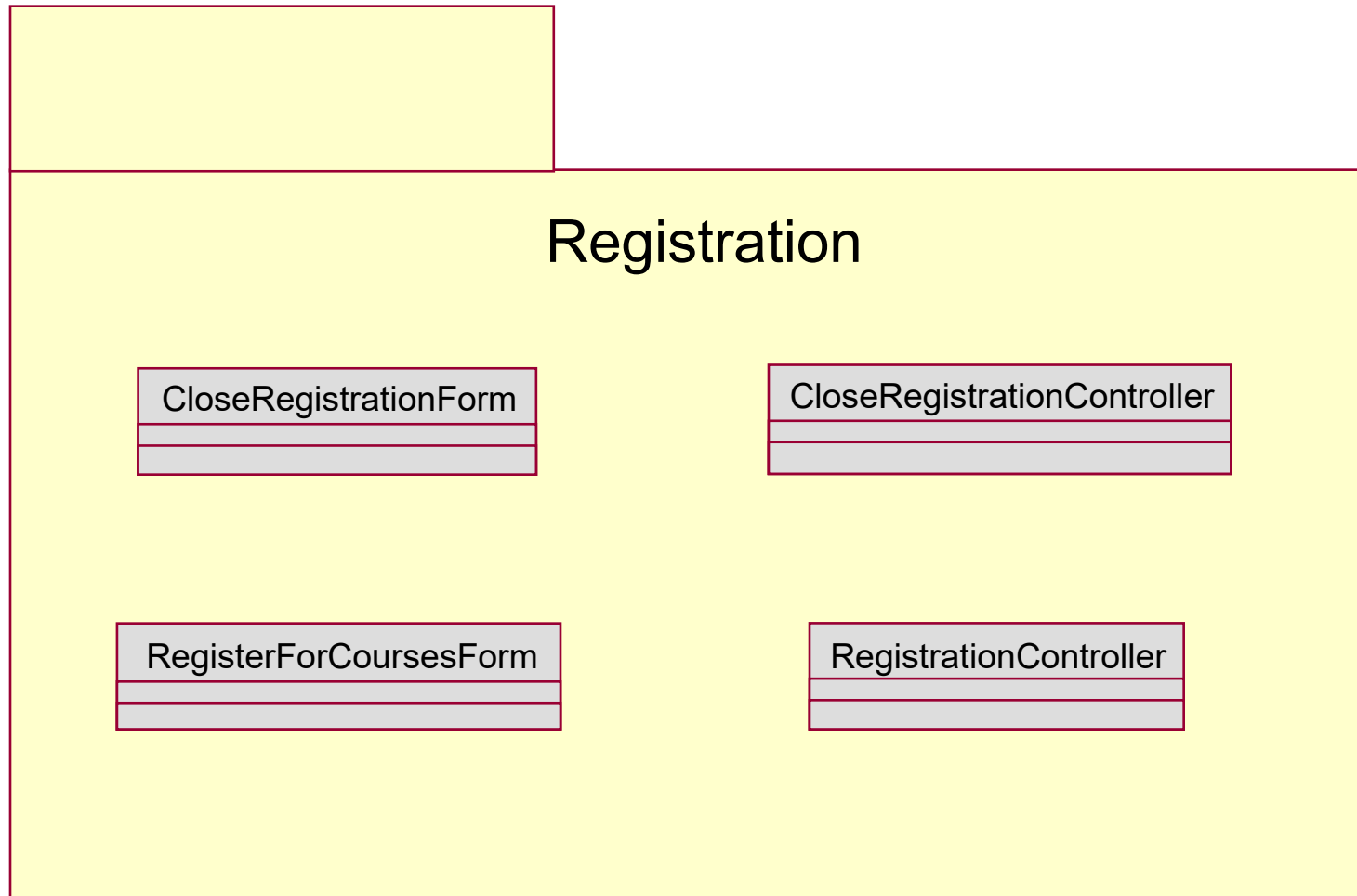
# Gói (package)

- Một cơ chế chung để tổ chức các phần tử thành nhóm.
- Một phần tử trong mô hình có thể chứa các phần tử khác.





# Ví dụ: Registration Package

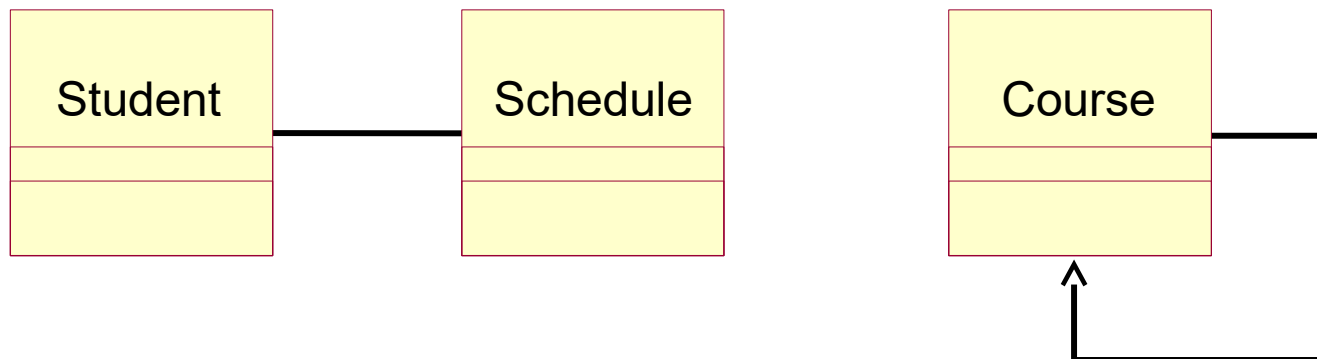


# Nội dung

1. Biểu đồ lớp (Class diagram)
- ⇒ 2. Liên kết (Association)
3. Kết tập (Aggregation)
4. Tổng quát hóa (Generalization)

# Liên kết (association) là gì?

- Mỗi liên hệ ngữ nghĩa giữa hai hay nhiều lớp chỉ ra sự liên kết giữa các thể hiện của chúng
- Mỗi quan hệ về mặt cấu trúc chỉ ra các đối tượng của lớp này có kết nối với các đối tượng của lớp khác.



# Bội số quan hệ (Multiplicity)

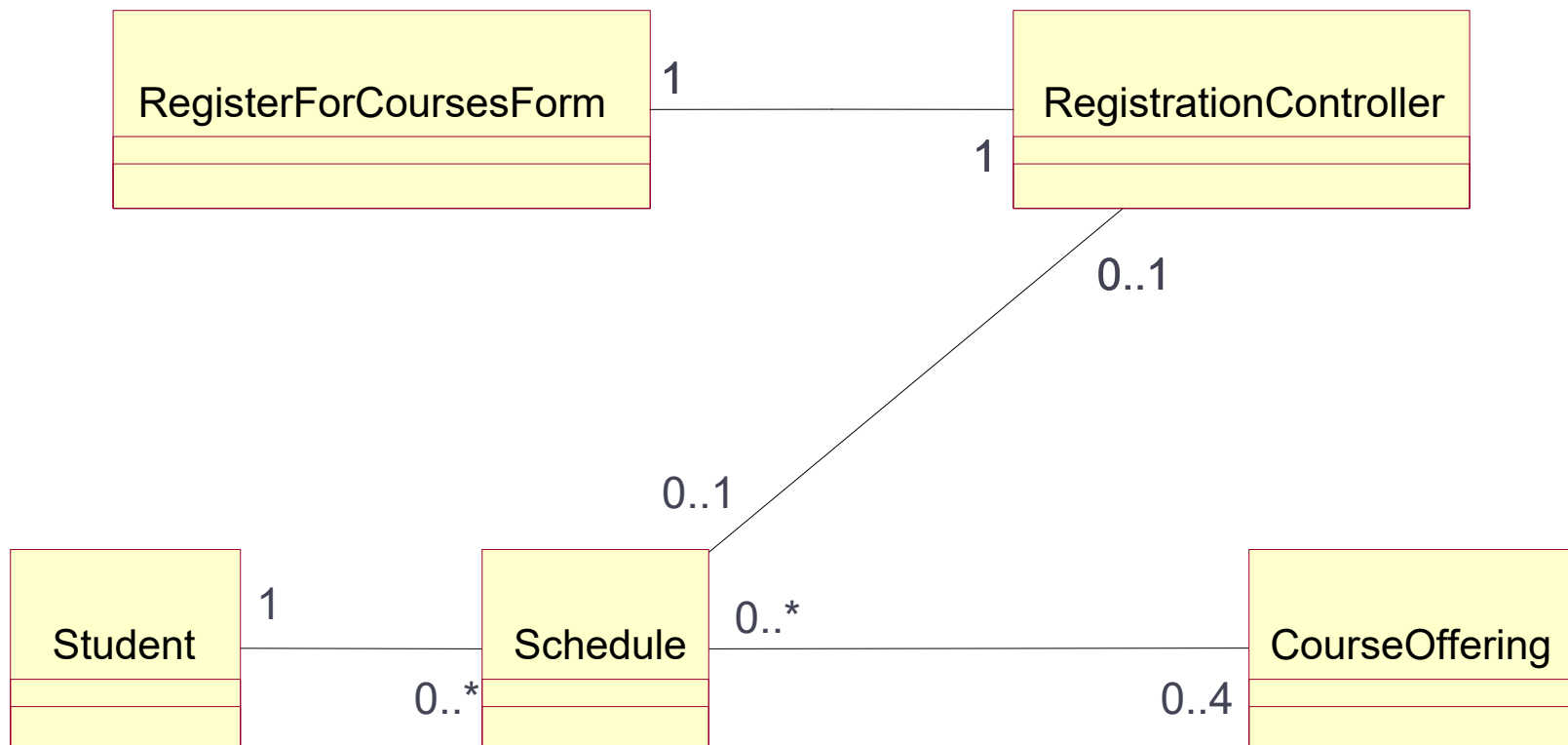
- Bội số quan hệ là số lượng thể hiện của một lớp liên quan tới MỘT thể hiện của lớp khác.
- Với mỗi liên kết, có hai bội số quan hệ cho hai đầu của liên kết.
  - Với mỗi đối tượng của Professor, có nhiều Course Offerings có thể được dạy.
  - Với mỗi đối tượng của Course Offering, có thể có 1 hoặc 0 Professor giảng dạy.



# Biểu diễn bội số quan hệ

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

# Ví dụ về bội số quan hệ



# Nội dung

1. Biểu đồ lớp (Class diagram)
2. Liên kết (Association)
- ⇒ 3. Kết tập (Aggregation)
4. Tổng quát hóa (Generalization)

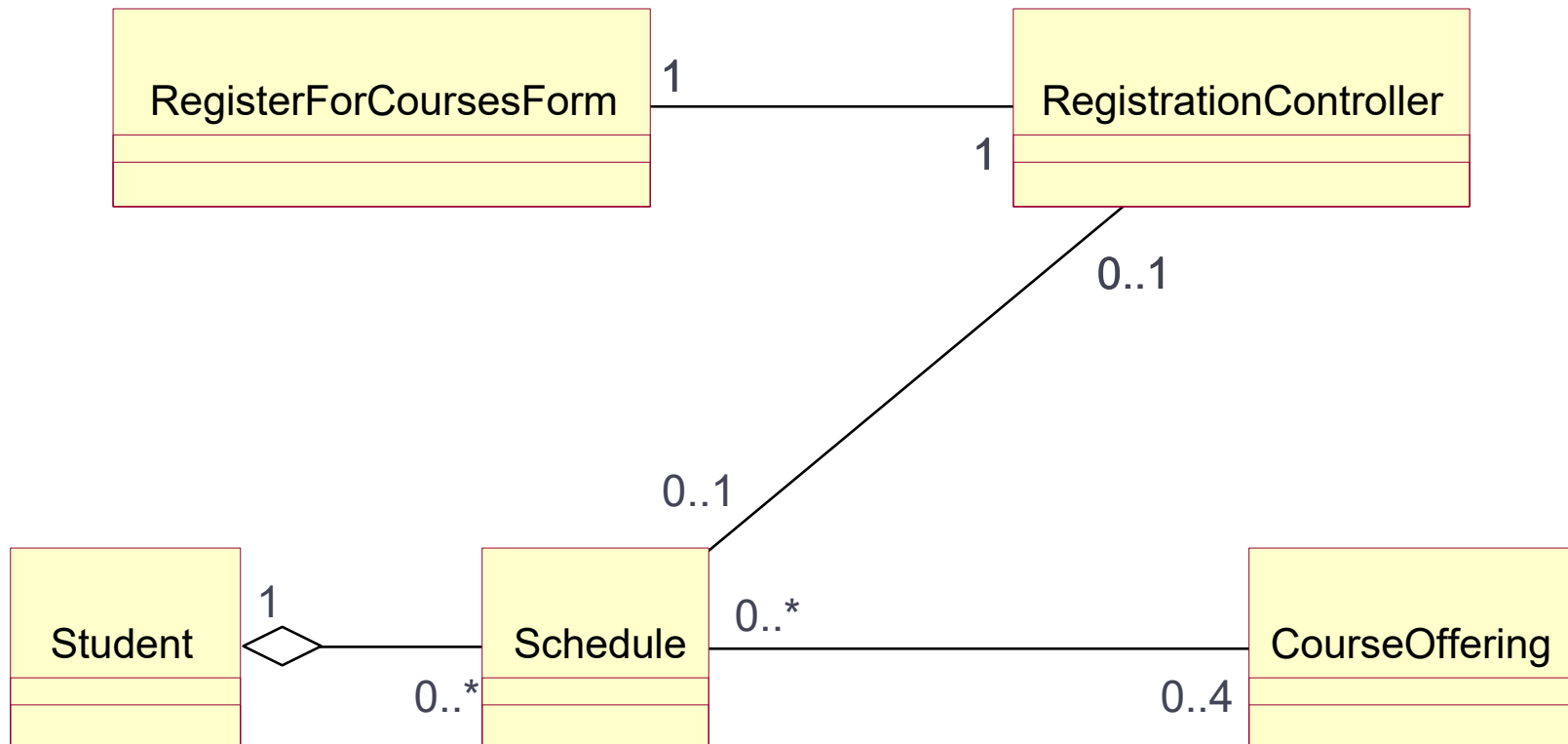
# Kết tập (aggregation) là gì?

- Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.
  - Kết tập là mối quan hệ “là một phần” (“is a part-of”).
- Bội số quan hệ được biểu diễn giống như các liên kết khác



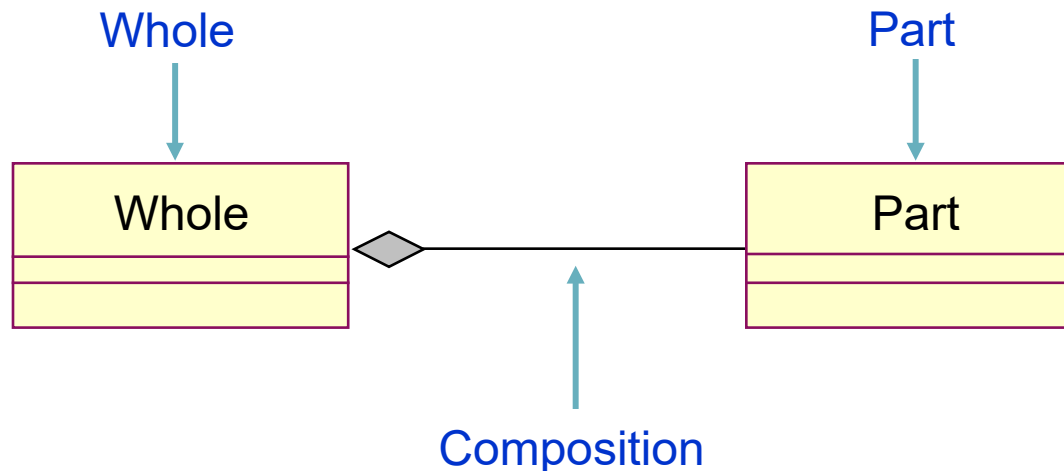


# Ví dụ về kết tập



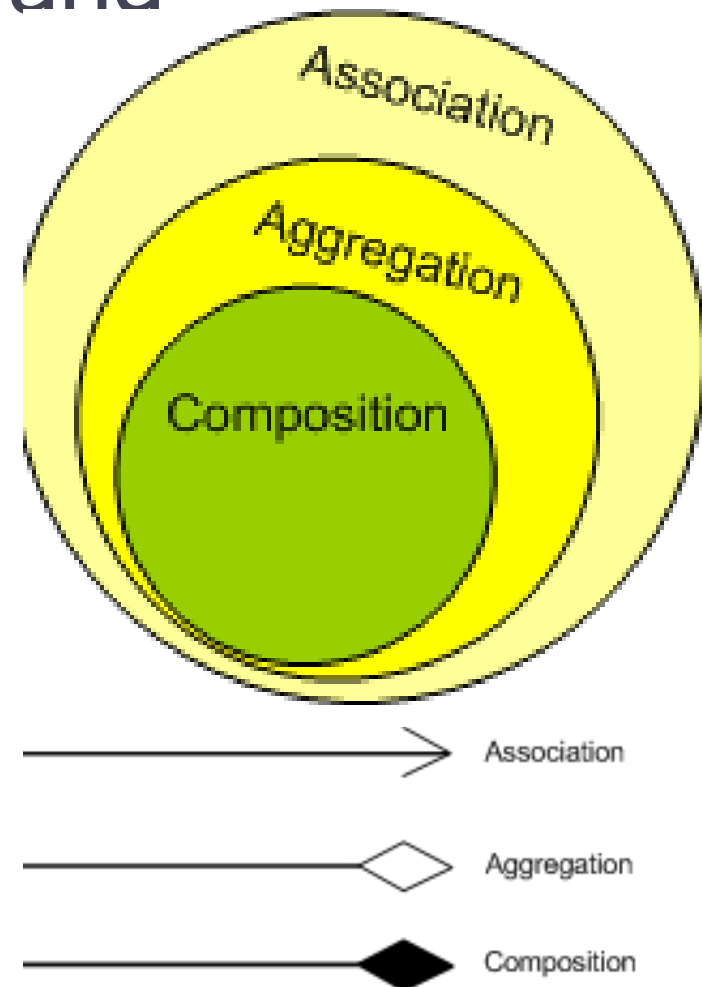
# Cấu thành (Composition) là gì?

- Một dạng của kết tập với quyền sở hữu mạnh và các vòng đời trùng khớp giữa hai lớp
  - Whole sở hữu Part, tạo và hủy Part.
  - Part bị bỏ đi khi Whole bị bỏ, Part không thể tồn tại nếu Whole không tồn tại.



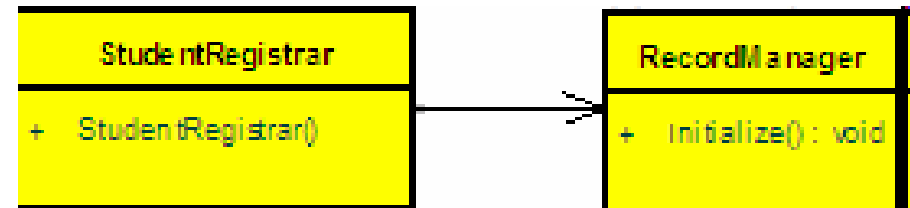
# Association, Aggregation and Composition

- Mỗi quan hệ giữa các lớp (relationship)
  - Liên kết (Association)
    - Sử dụng (use-a)
  - Kết tập (Aggregation)
    - Strong association
    - has-a/is-a-part
  - Hợp thành (Composition)
    - Strong aggregation
    - Share life-time



## Ví dụ – Association

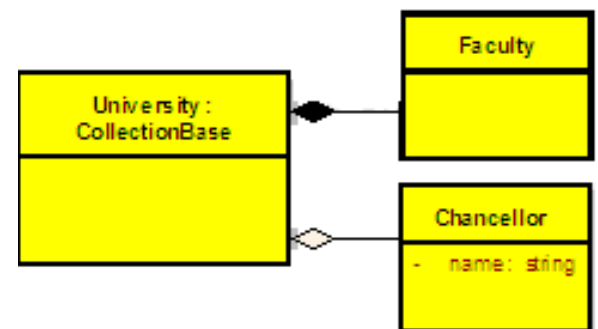
```
public class StudentRegistrar {  
    public StudentRegistrar () {  
        (new RecordManager()).initialize();  
    }  
}
```



- Một lớp sử dụng lớp khác
- Lời gọi phương thức của đối tượng thuộc lớp này trong lớp kia
- Thường được cài đặt bằng tham chiếu (nhưng không bắt buộc).

# Ví dụ – Aggregation vs. Composition

- Aggregation – *University and Chancellor*
  - Nếu không có trường Đại học (*University*), hiệu trưởng (*Chancellor*) không thể tồn tại.
  - Nếu không có *Chancellor*, *University* vẫn có thể tồn tại
- Composition – *University and Faculty*
  - *University* không thể tồn tại nếu không có các giảng viên (*Faculty*) và ngược lại (share time-life)
    - Thời gian sống của *University* gắn chặt với thời gian sống của *Faculty*
    - Nếu *Faculties* được giải phóng thì *University* không thể tồn tại và ngược lại



## Nội dung

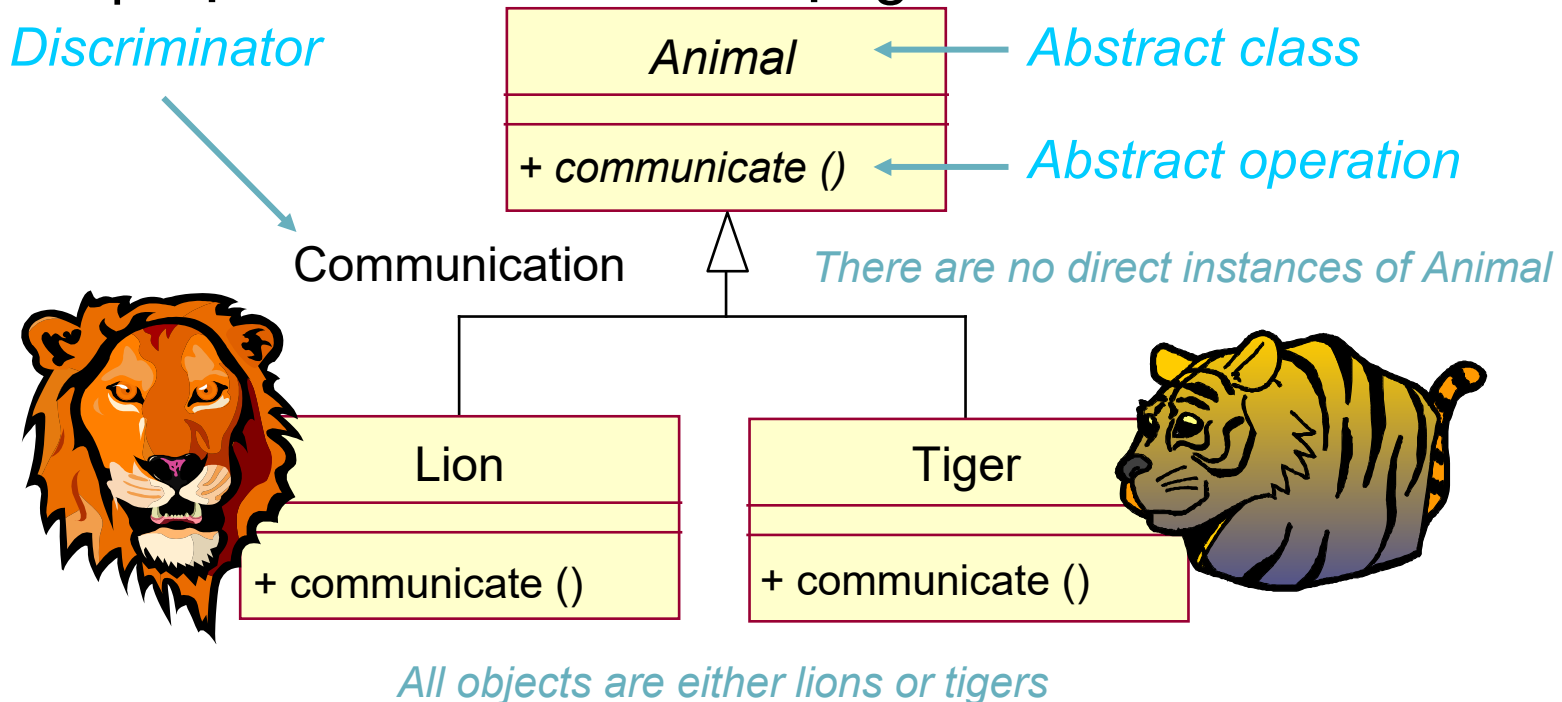
1. Biểu đồ lớp (Class diagram)
2. Liên kết (Association)
3. Kết tập (Aggregation)
- ⇒ 4. Tổng quát hóa (Generalization)

# Tổng quát hóa (Generalization)

- Mỗi quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hoặc nhiều lớp khác
- Xác định sự phân cấp về mức độ trừu tượng hóa trong đó lớp con kế thừa từ một hoặc nhiều lớp cha
  - Đơn kế thừa (Single inheritance)
  - Đa kế thừa (Multiple inheritance)
- Là mối liên hệ “là một loại” (“is a kind of”)

# Lớp trừu tượng và lớp cụ thể (Abstract and Concrete Class)

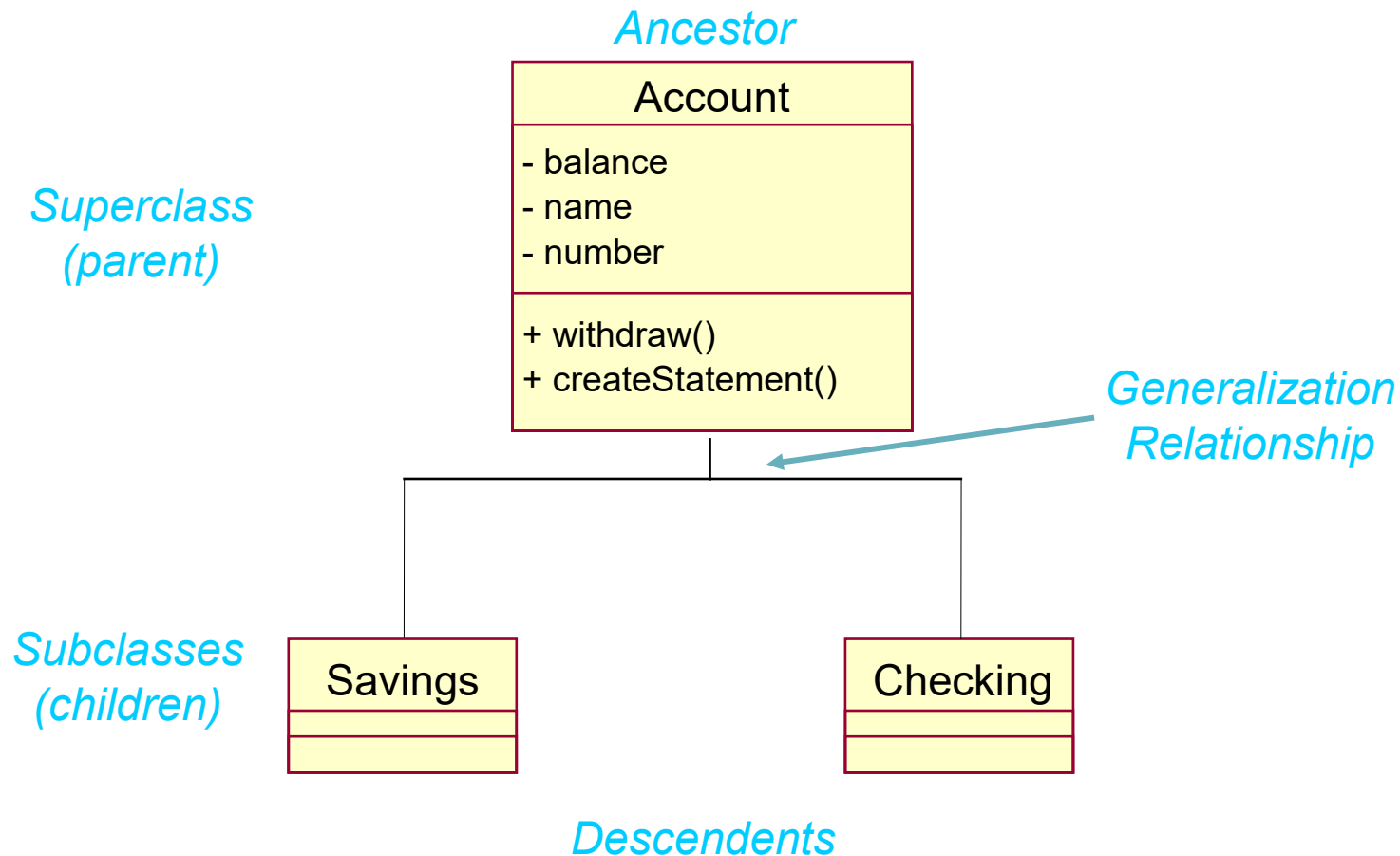
- Lớp trừu tượng không thể có đối tượng
  - Chứa phương thức trừu tượng
  - Chữ nghiêng
- Lớp cụ thể có thể có đối tượng





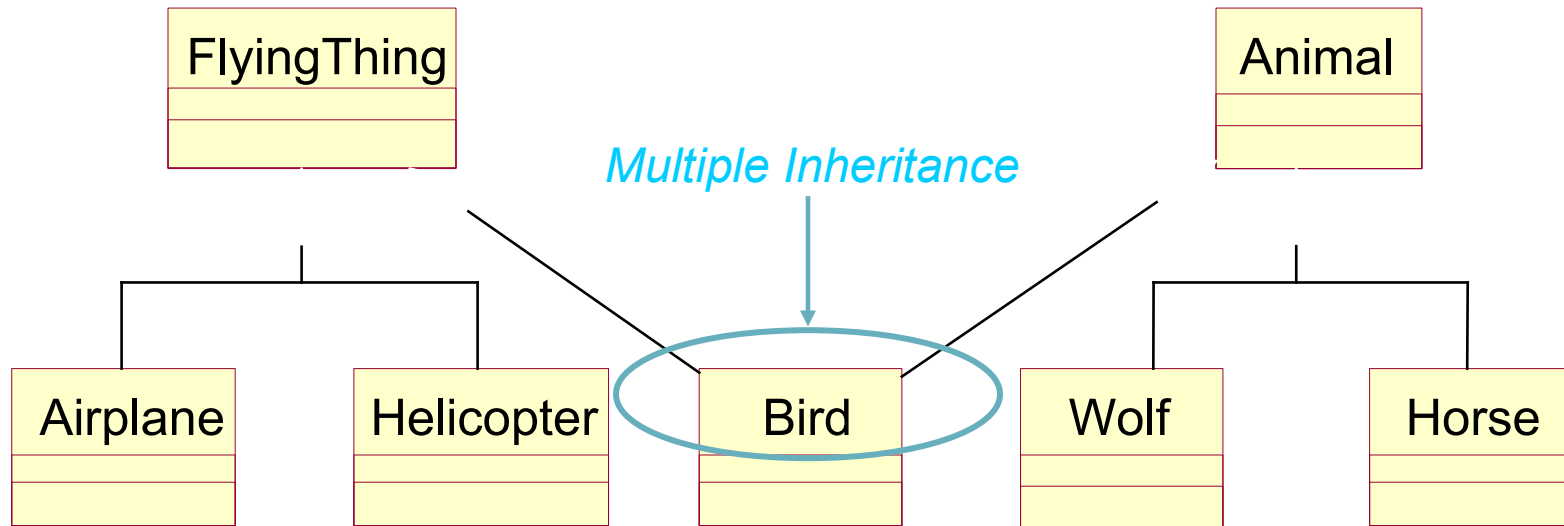
# Ví dụ về Đơn kế thừa

- Một lớp kế thừa từ MỘT lớp khác



# Ví dụ về Đa kế thừa

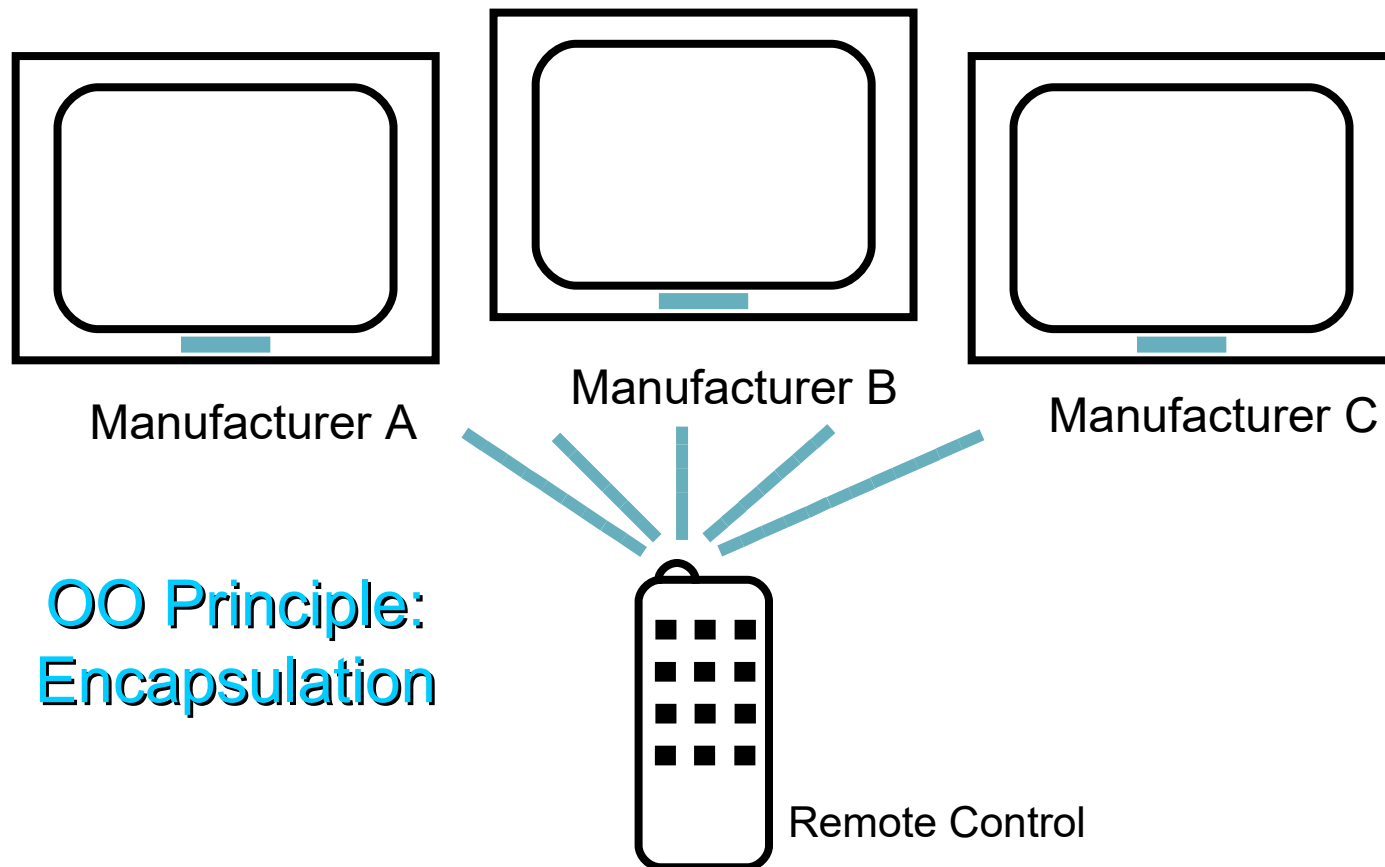
- Một lớp có thể kế thừa từ nhiều lớp khác



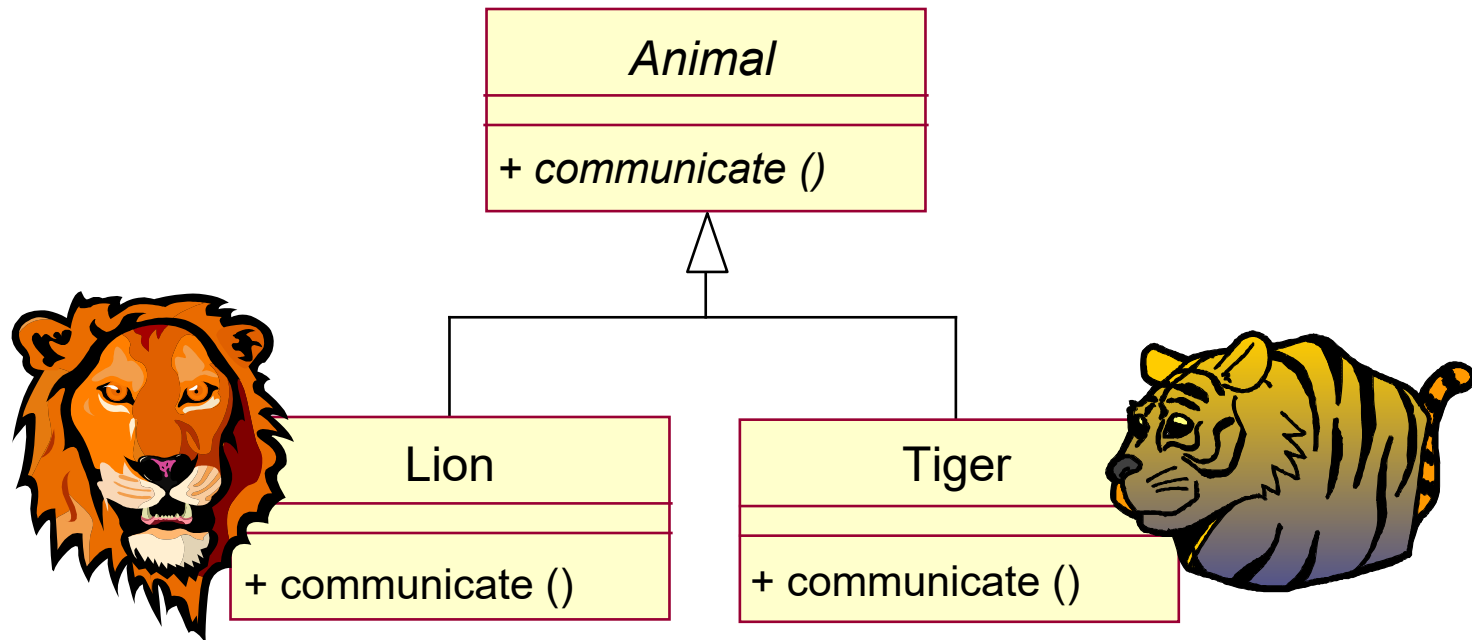
***Sử dụng đa kế thừa chỉ khi cần thiết và luôn luôn phải cẩn thận!***

# Đa hình (Polymorphism) là gì?

- Khả năng che giấu các thực thi khác nhau dưới một giao diện duy nhất.



# Tổng quát hóa: Thực thi đa hình



## *Without Polymorphism*

```
if animal = "Lion" then
    Lion communicate
else if animal = "Tiger" then
    Tiger communicate
end
```

## *With Polymorphism*

```
Animal communicate
```

# Bài tập

- Given:
  - A set of classes and their relationships
- Draw:
  - A class diagram

