Chương 3: Sơ đồ lớp (Class Diagram)

Nội dung

* Mục đích của sơ đồ lớp
* Định nghĩa và ký hiệu
* Giao diện
* Quan hệ giữa các lớp
* Ràng buộc
* Xây dựng một sơ đồ lớp

Mục đích của sơ đồ lớp

* Sơ đồ lớp được xem là mô hình quan trọng nhất trong phân tích hệ thống hướng đối tượng
* Dùng để mô tả cấu trúc bên trong, cấu trúc tĩnh của hệ thống
* Không dùng để chỉ ra cách thức làm thế nào sử dụng các tác tử (operation)

Định nghĩa và ký hiệu

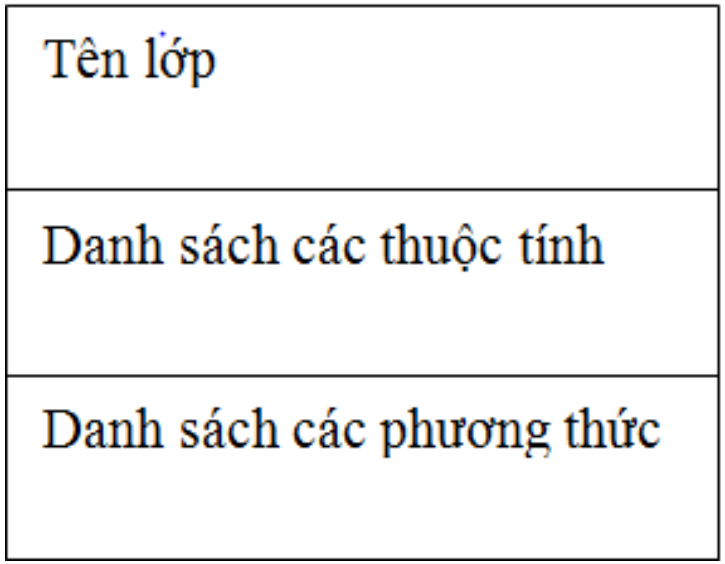
* Lớp
* Phương thức trừu tượng và lớp trừu tượng
* Sự bao gói (encapsulation) và mức độ hiển thị (visibility)
* Thuộc tính (attribute)
* Phương thức (method)

Lớp

* Lớp là một sự mô tả một tập hợp các đối tượng có cùng các đặc tính: cùng một ngữ nghĩa, có chung các thuộc tính, các phương thức và các quan hệ.
* Một đối tượng là một thể hiện của lớp.

Lớp

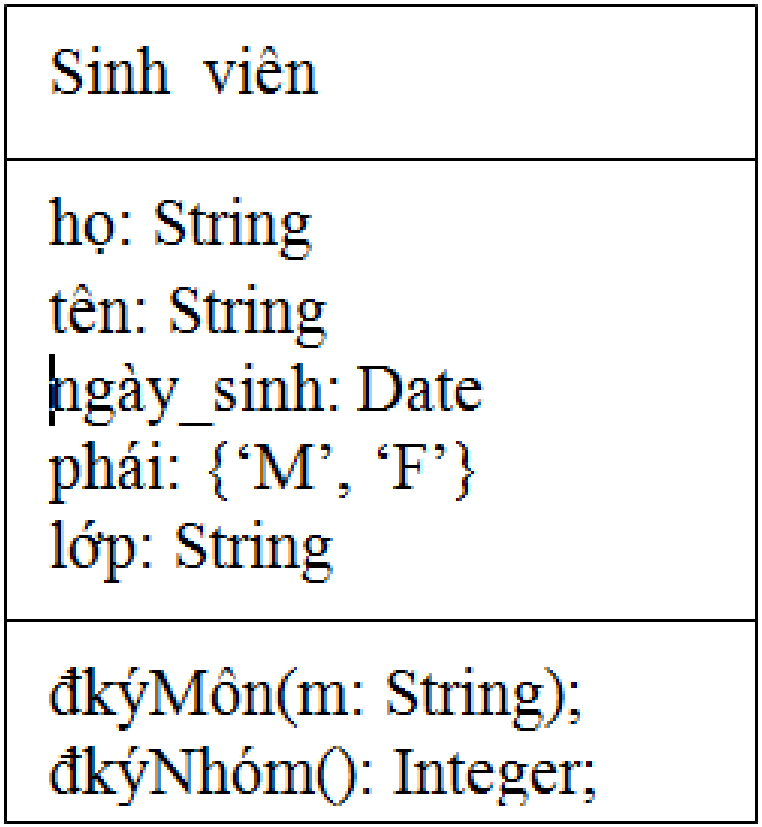
* **Ký hiệu:**



Lớp

* Tên lớp:
  * Phải có nghĩa và bắt đầu bằng chữ hoa.
  * Nếu nó được đóng gói, cần đặc tả tất cả các gói chứa nó theo thứ tự từ lớn đến nhỏ và cách bởi 2 dấu hai chấm (::).
  * Ví dụ : java ::lang ::Object
* Mỗi thuộc tính (attribute) được mô tả bằng tên và kiểu, tên của thuộc tính phải duy nhất trong lớp.
* Mỗi phương thức (method) được mô tả kiểu trả về (return type), danh sách các đối số (parameter) và kiểu tương ứng của mỗi đối số (parameter type).

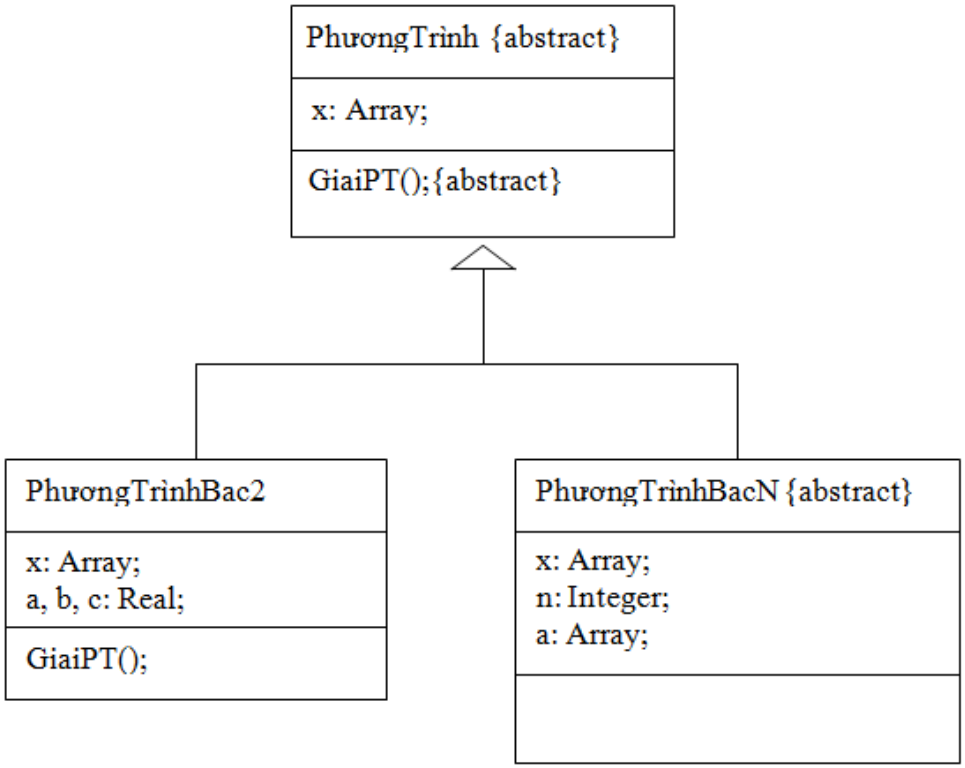
Ví dụ


Sinh viên
họ: String
tên: String
ngày_sinh: Date
phái: {'M', 'F'}
lớp: String
đkýMôn(m: String);
đkýNhóm(): Integer;

Phương thức trừu tượng và lớp trừu tượng

* Một phương thức được gọi là trừu tượng nếu người ta biết được phần mô tả đầu của nó (header / signature / entête) nhưng không biết cách thức nó có thể được thực hiện như thế nào
* Một lớp được gọi là trừu tượng nếu nó định nghĩa ít nhất một phương thức trừu tượng hoặc khi một lớp cha chứa một phương thức trừu tượng chưa được thực hiện
* Tên của một lớp trừu tượng sẽ có thêm từ khóa «abstract» đặt giữa dấu móc {}

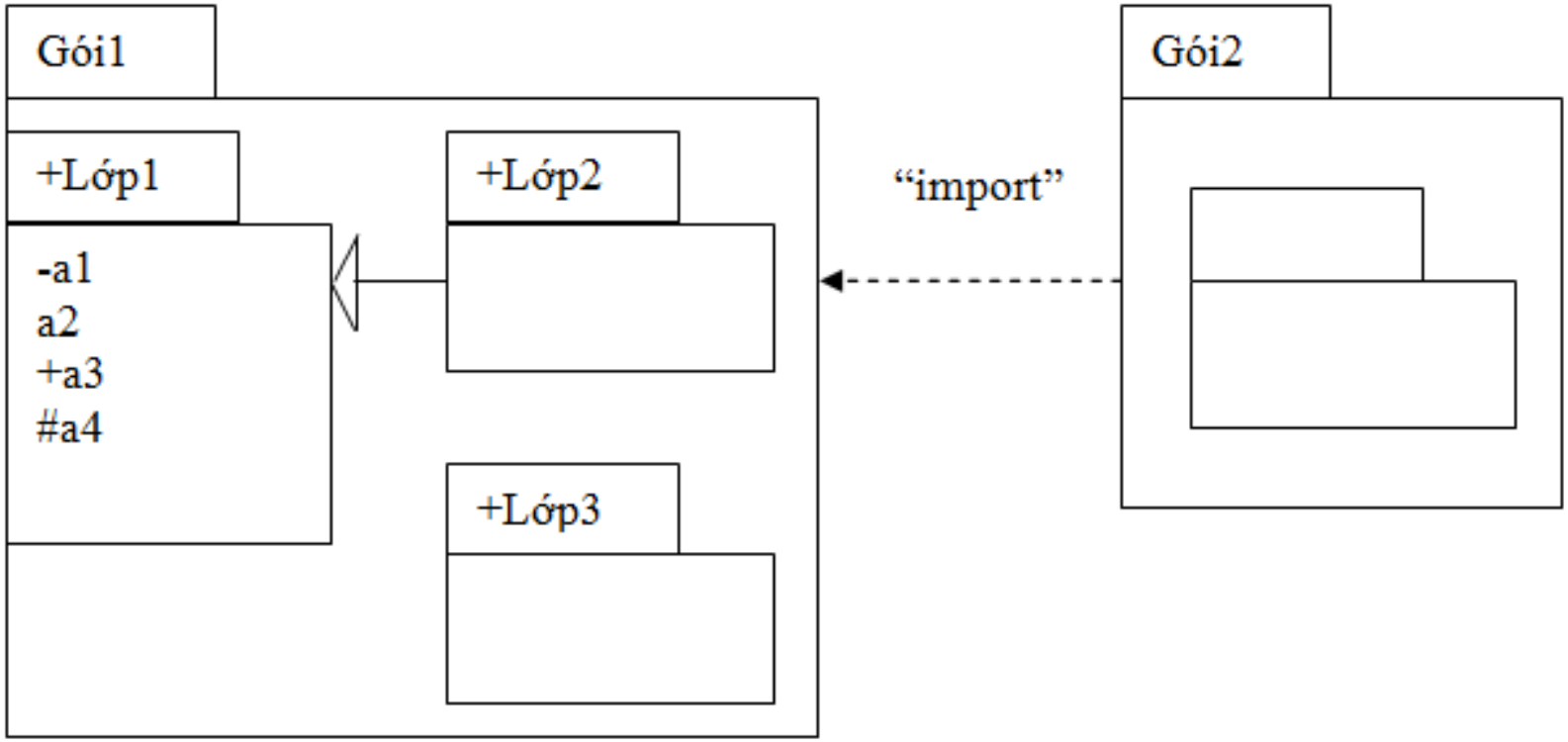
Ví dụ


PhươngTrình {abstract}
x: Array;
GiaiPT();{abstract}
PhươngTrìnhBac2
x: Array;
a, b, c: Real;
GiaiPT();
PhươngTrìnhBacN {abstract}
x: Array;
n: Integer;
a: Array;

Sự bao gói và mức độ hiển thị

* Một lớp hoặc thành phần của lớp (chẳng hạn thuộc tính) có nhiều mức độ được nhìn thấy tùy theo ký tự đứng trước nếu:
  * Có từ khóa *public* hoặc dấu + đứng trước: được nhìn thấy từ bên ngoài
  * Không có ký tự nào đứng trước: chỉ được nhìn thấy từ trong gói chứa lớp đang xét
  * Có từ khóa *protected* hoặc dấu # đứng trước: được nhìn thấy từ trong lớp đang xét hoặc các lớp con cháu của lớp đó
  * Có từ khóa *private* hoặc dấu - đứng trước: chỉ được nhìn thấy từ trong lớp đang xét.

Ví dụ



Thuộc tính

* Định nghĩa
* Thuộc tính của lớp (static attribute / class attribute)
* Thuộc tính do suy diễn (derived attribute)

Định nghĩa

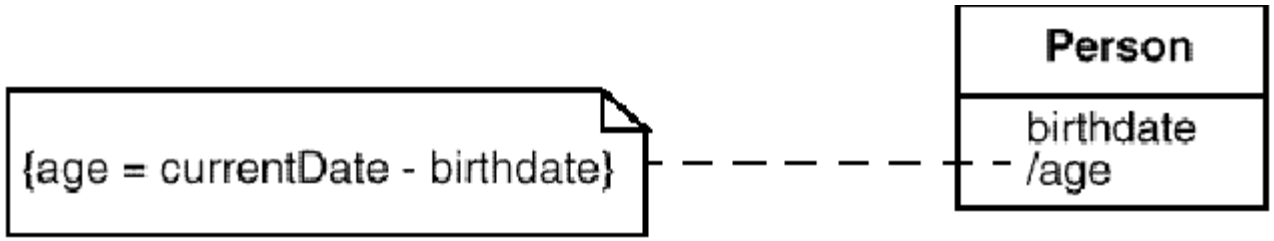
* Các thuộc tính biểu diễn các dữ liệu được bao gói trong các đối tượng của lớp đang xét
* Một thuộc tính có thể được khởi tạo lúc khai báo
* Ngữ pháp đầy đủ của thuộc tính như sau:
  * <mức độ hiển thị> [/] <tên thuộc tính>: <kiểu> [‘[’<bản số>] ’]’ [=<trị khởi tạo>]
    * <mức độ hiển thị>: dùng các từ khóa *public*, *private*, *protected* hoặc các dấu tương ứng
    * <kiểu>: kiểu dữ liệu cơ sở hoặc tên lớp
    * <bản số>: các chỉ số tối thiểu và tối đa cho một mảng của kiểu nói trên
    * <trị khởi tạo>: phải có kiểu tương ứng với kiểu nói trên

Thuộc tính của lớp

* Thông thường, một thuộc tính sẽ có các trị khác nhau ở các đối tượng khác nhau của lớp đó
* Tuy nhiên, tồn tại những thuộc tính có trị duy nhất cho tất cả các đối tượng của lớp đó, đó là thuộc tính tĩnh
  * Trong Java và C++ dùng từ khóa *static* đứng trước tên của thuộc tính
  * Trong UML, thuộc tính của lớp được gạch dưới
* **Ví dụ:** giá trị PI=3.14 của lớp Math trong Java luôn không đổi đối với bất kỳ đối tượng nào của lớp Math.

Thuộc tính do suy diễn

* Là thuộc tính có được do sự suy diễn, tính toán từ các thuộc tính khác
* Nó được sử dụng như một thuộc tính thực thụ, nhưng được tính toán qua một phương thức.
* Thuộc tính do suy diễn được đánh dấu bởi dấu "/" (slash) đứng trước
* **Ví dụ:**



Phương thức

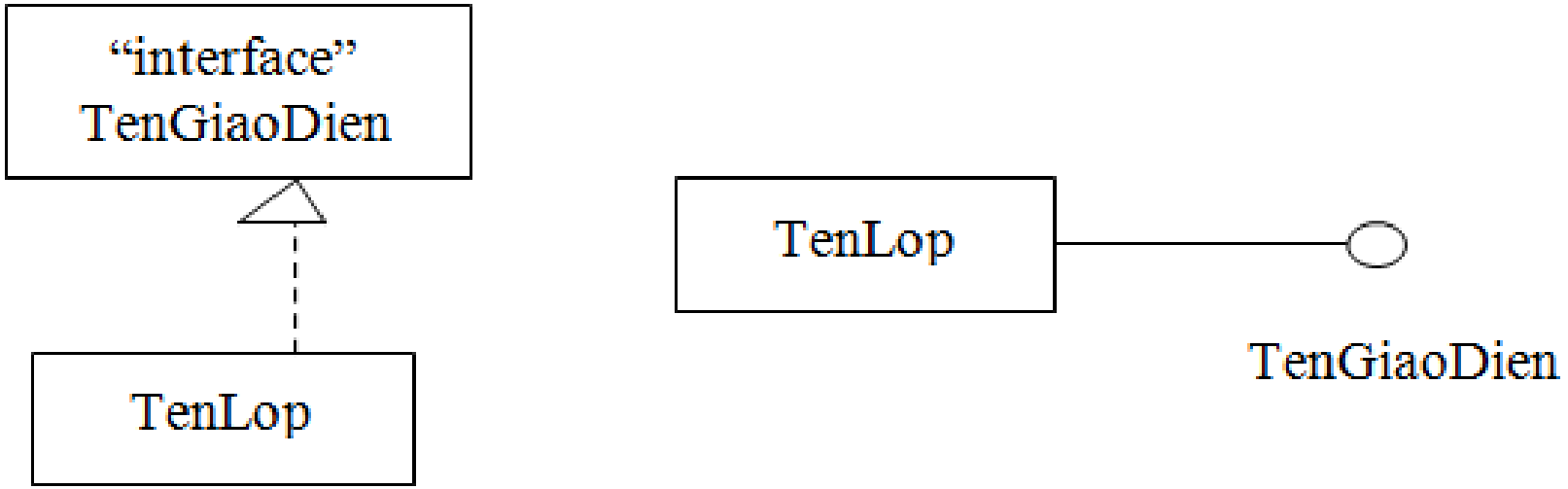
* Hành vi của một đối tượng được mô hình hóa bằng một tập các phương thức
* Người ta phân biệt đặc tả (specification / header / signature) của phương thức với cài đặt (implementation) của nó
* Tương ứng với mỗi đặc tả, có thể có nhiều cài đặt, nhưng với mỗi cài đặt chỉ tương ứng với một đặc tả duy nhất
* Đặc tả còn được gọi là “operation”, còn cài đặt cụ thể được gọi là “method”

Giao diện

* Giao diện cũng giống như lớp, nhưng không có đặc tả cho một cấu trúc bên trong, cũng như các giải thuật thực hiện cho các phương thức
* Giao diện có thể mô tả chính xác điều kiện và kết quả việc kích hoạt nó
* Ngược với lớp, giao diện không có thể hiện; để được sử dụng, nó thường phải được thực hiện bởi một lớp
* Một giao diện có thể được chuyên biệt hóa hoặc tổng quát hóa bởi một giao diện khác

Giao diện

* **Cách biểu diễn giao diện và mối liên hệ với lớp:**

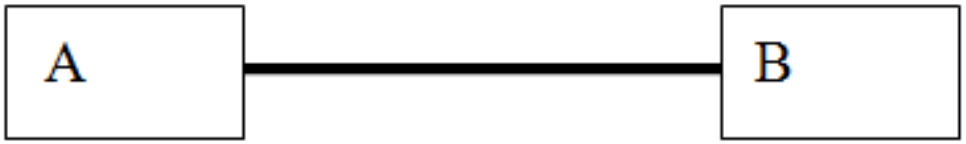


Quan hệ giữa các lớp

* Liên kết (association)
* Tính bội (multiplicity)
* Liên kết có ràng buộc (association with constraint)
* Lớp-liên kết (association class)
* Liên kết do suy diễn (derived association)
* Liên kết có thẩm định (qualified association)
* Liên kết nhiều chiều (multidimensional association)
* Quan hệ kết tập (aggregation)
* Quan hệ cấu thành (composition)
* Quan hệ phụ thuộc (dependancy)
* Quan hệ thừa kế (generalization)

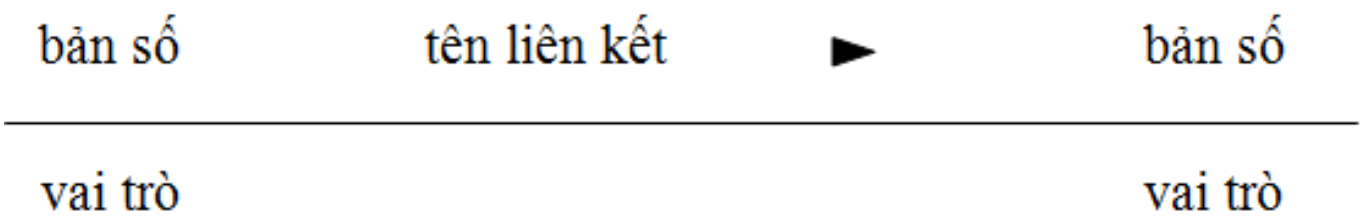
Liên kết

* Một liên kết biểu diễn mối liên hệ ngữ nghĩa bền vững giữa 2 lớp
* **Ký hiệu:**



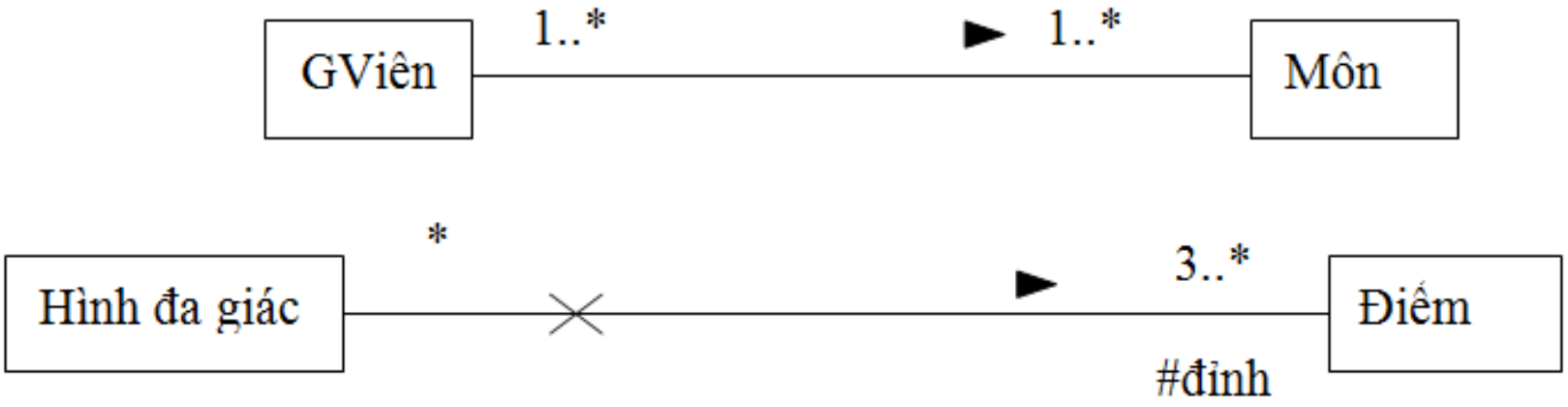
Liên kết

* Các thành phần đầy đủ của một liên kết gồm có:



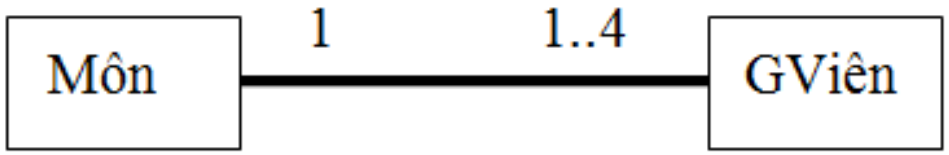
* Mũi tên để chỉ ý nghĩa chính xác của tên liên kết từ lớp bên trái sang lớp bên phải
* "Vai trò" phía một lớp dùng để chỉ vai trò của lớp đó trong liên kết đối với lớp kia.
* Nếu muốn biểu diễn sự thông thương từ một lớp sang lớp khác bị cấm ta dùng dấu tréo ở phía lớp đó trên đường nối.

Ví dụ



Tính bội

* Ở 2 đầu mối liên kết, phải có chỉ số để biểu diễn tính bội của liên kết, chính xác hơn là bản số (cardinality) của mỗi lớp tham gia vào liên kết
* Chú ý: bản số của một lớp ở đầu này của liên kết được ký hiệu ở lớp đầu kia của liên kết



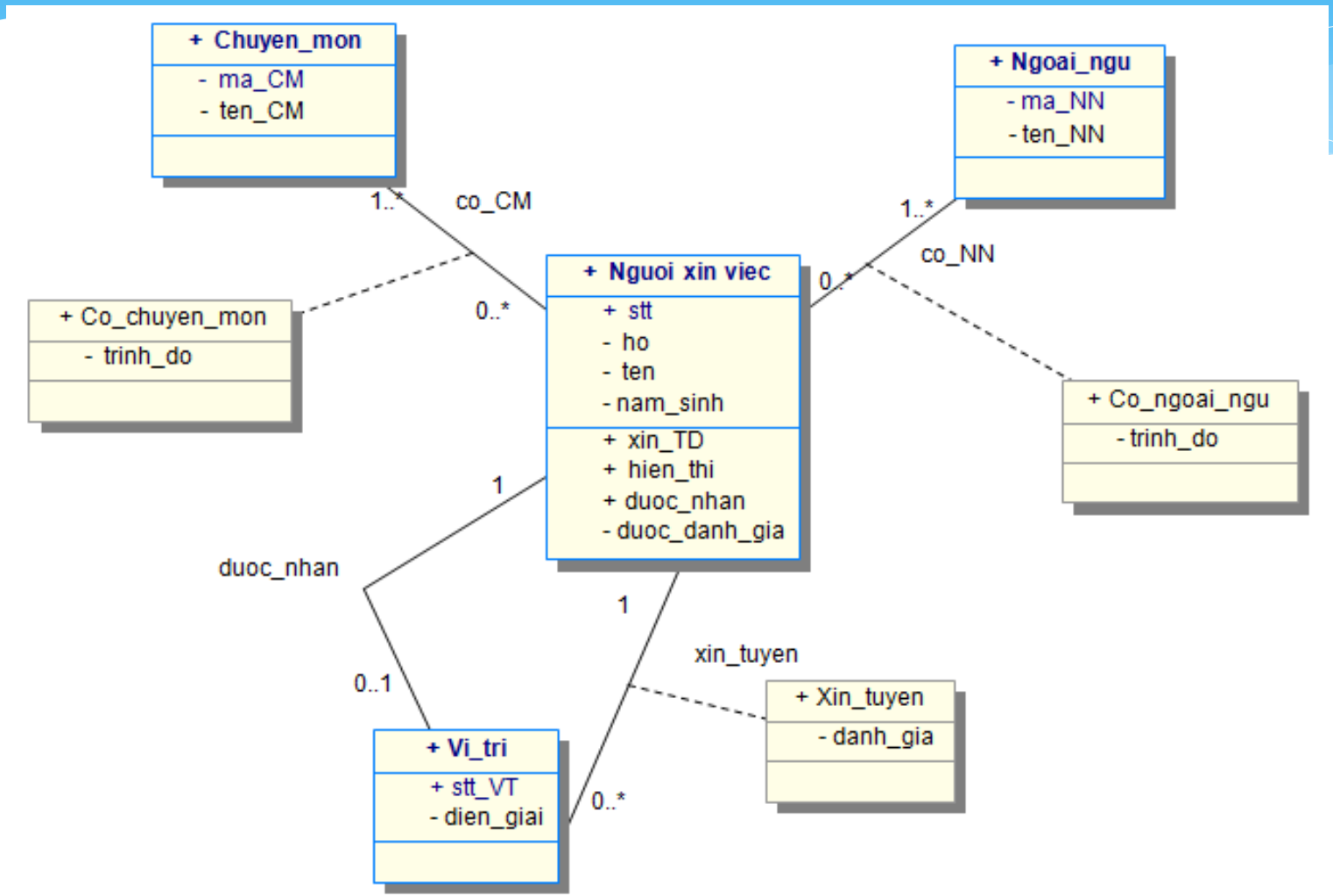
Liên kết có ràng buộc

* Việc thêm vào ràng buộc cho một hoặc nhiều liên kết mang lại nhiều thông tin hơn vì nó cho phép mô tả chính xác hơn tầm ảnh hưởng và chiều của liên kết
* Các ràng buộc được đặt trong dấu móc "{}" và có thể biểu diễn bằng bất kỳ ngôn ngữ nào; trong đó, ngôn ngữ OCL (Object Constraint Language) được ưa chuộng hơn
* **Ví dụ:**

Lớp-liên kết

* Khi phân tích, ta thấy có những thuộc tính không thể đặt vào được trong lớp thuần túy nào, mà phụ thuộc đồng thời vào nhiều lớp nối nhau qua một liên kết
* Vì trong phân tích hệ thống hướng đối tượng, chỉ có lớp mới có thể chứa được thuộc tính nên liên kết này trở thành một lớp, gọi là lớp-liên kết

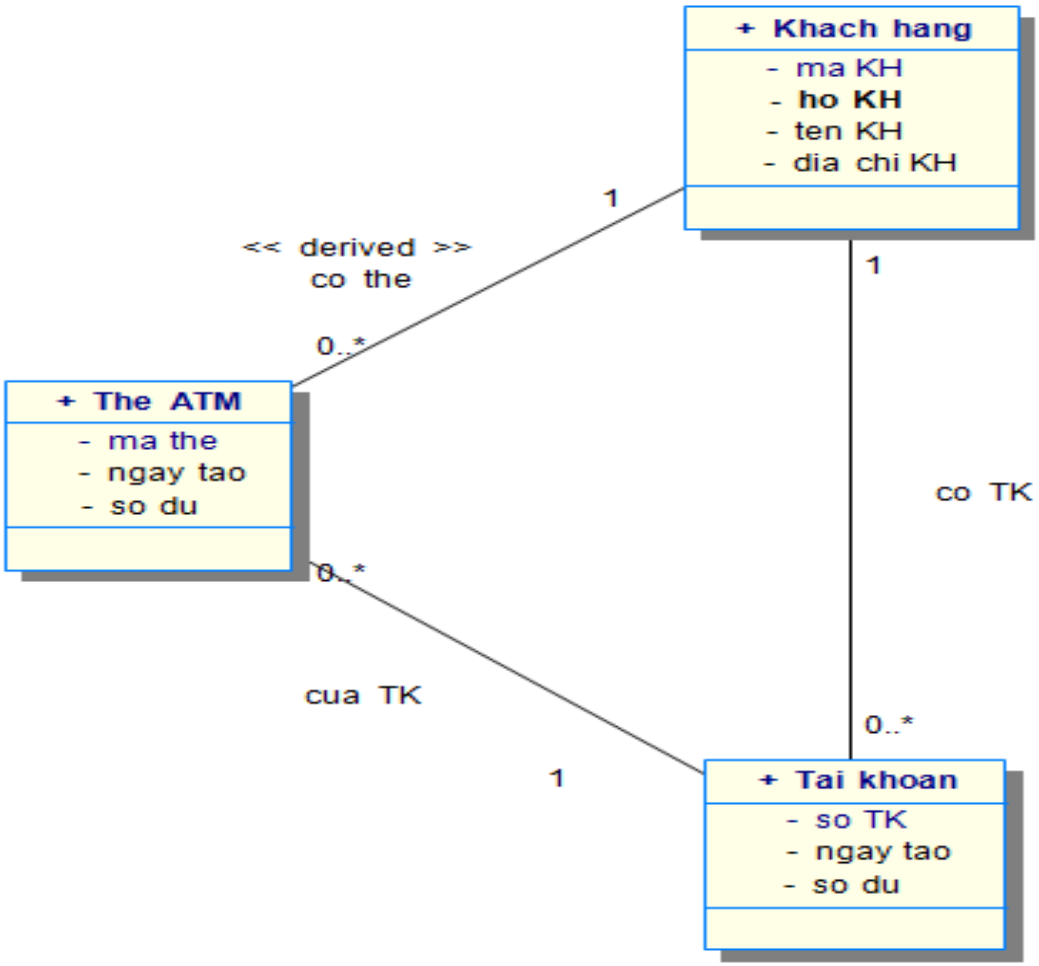
Ví dụ


+ Chuyen_mon
- ma_CM
- ten_CM
+ Ngoai_ngu
- ma_NN
- ten_NN
1..*
co_CM
1..*
co_NN
+ Co_chuyen_mon
- trinh_do
0..*
+ Nguoi xin viec
+ stt
- ho
- ten
- nam_sinh
+ xin_TD
+ hien_thi
+ duoc_nhan
- duoc_danh_gia
0..*
+ Co_ngoai_ngu
- trinh_do
1
duoc_nhan
1
xin_tuyen
0..1
+ Xin_tuyen
- danh_gia
+ Vi_tri
+ stt_VT
- dien_giai
0..*

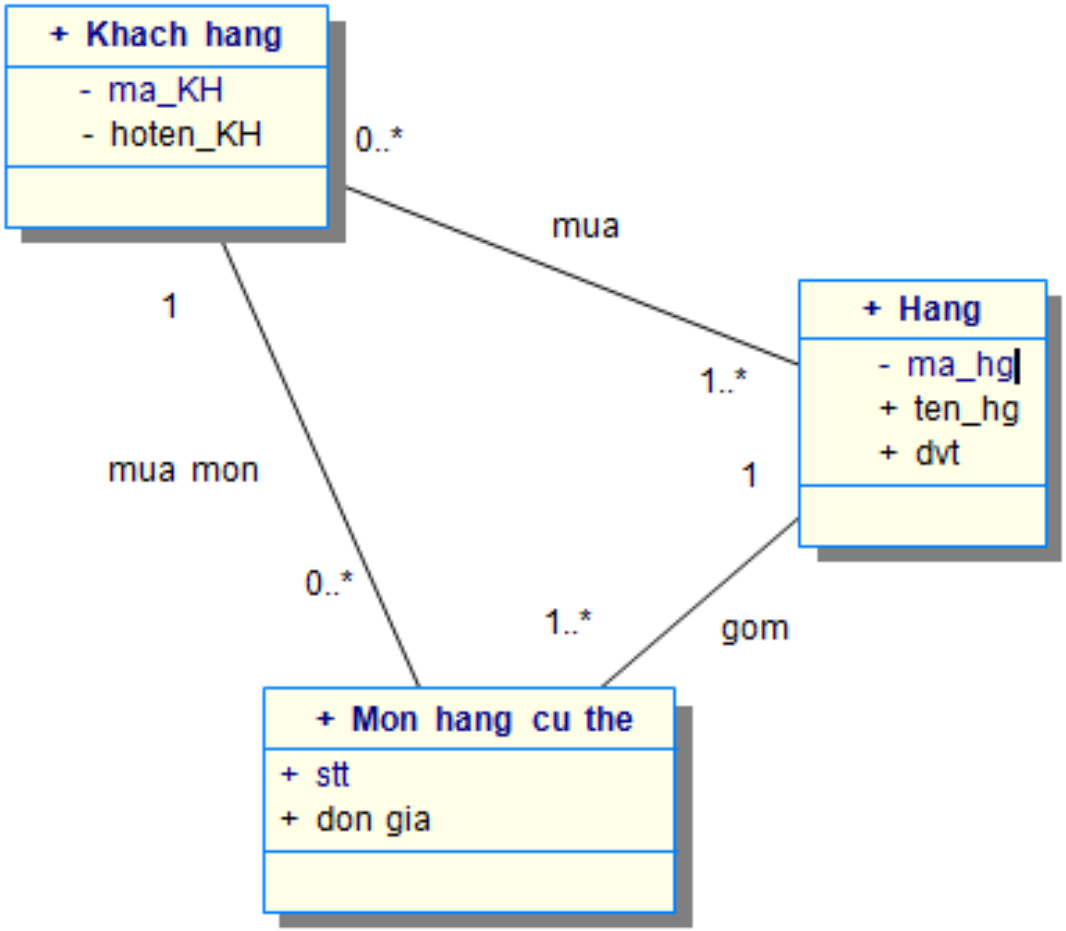
Liên kết do suy diễn

* Liên kết do suy diễn là liên kết được đặt điều kiện hoặc được suy diễn từ ít nhất một liên kết khác
* Tuy nó tạo ra sự dư thừa, nhưng thực tế lại có ích nếu phải sử dụng nhiều liên kết mới có được kết quả như mong muốn
* **Ký hiệu:** có dấu slash (“/”) trước tên liên kết

Ví dụ


+ Khach hang
- ma KH
- ho KH
- ten KH
- dia chi KH
1
<< derived >>
co the
0..*
1
+ The ATM
- ma the
- ngay tao
- so du
co TK
0..*
cua TK
0..*
1
+ Tai khoan
- so TK
- ngay tao
- so du

Ví dụ

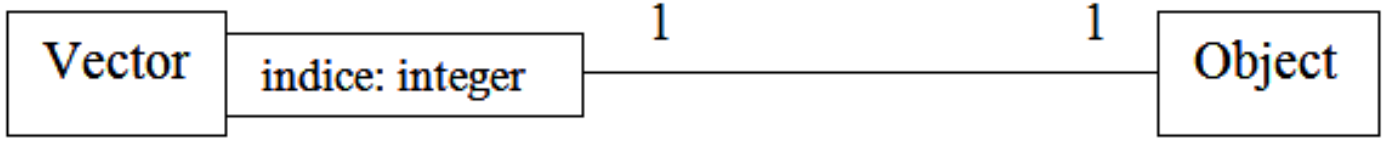

+ Khach hang
- ma_KH
- hoten_KH
0..*
mua
1
1..*
+ Hang
- ma_hg
+ ten_hg
+ dvt
mua mon
1
0..*
1..*
gom
+ Mon hang cu the
+ stt
+ don gia

Liên kết có thẩm định

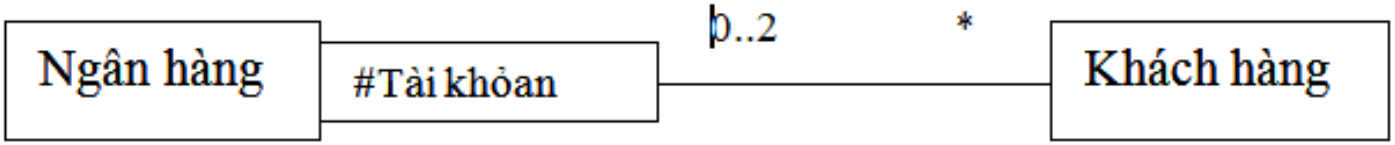
* Đôi khi một lớp này có tác động đến lớp khác liên kết với nó, đòi hỏi liên kết phải có một sự thẩm định (qualification) để tránh mô hình hóa không chính xác
* Sự thẩm định này biểu hiện bằng cách cho thêm:
  * Một chỉ số vào liên kết
  * Hoặc một lớp vào lớp ban đầu

Ví dụ

* **Ví dụ 1:**



* **Ví dụ 2:**

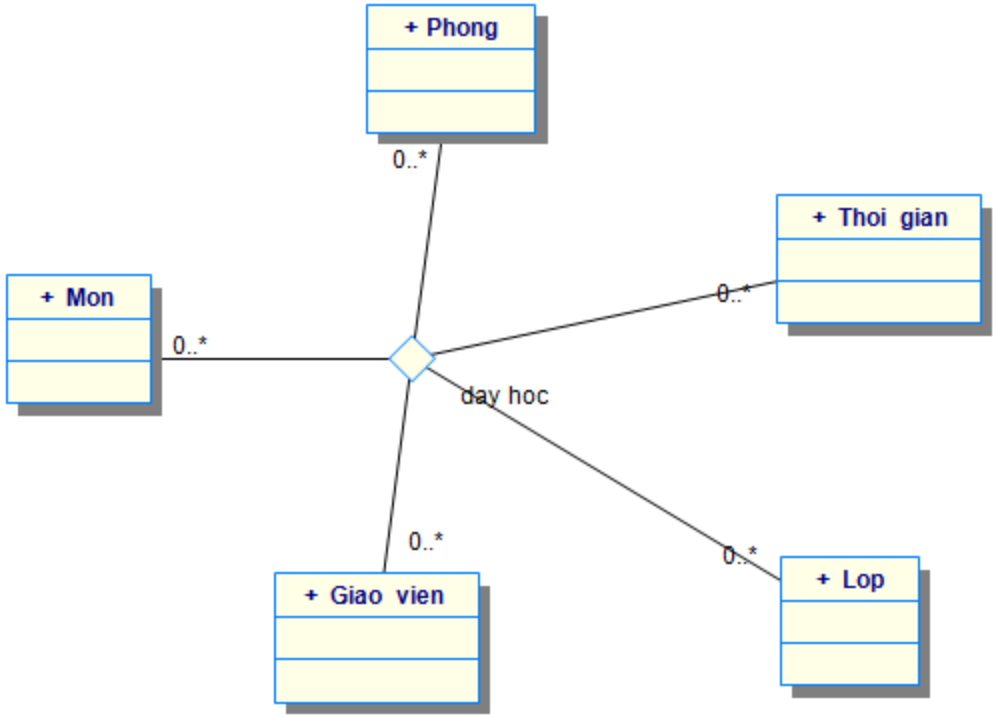


Liên kết nhiều chiều

* Trong UML, ít khi dùng liên kết nhiều chiều, mà thường giới hạn ở 2 chiều, đôi khi 3 chiều
* Vì để xác định bản số của mỗi lớp khá khó khăn
* Thường thì các liên kết nhiều chiều được chuyển sang dùng lớp-liên kết, hoặc nhiều liên kết 2 chiều

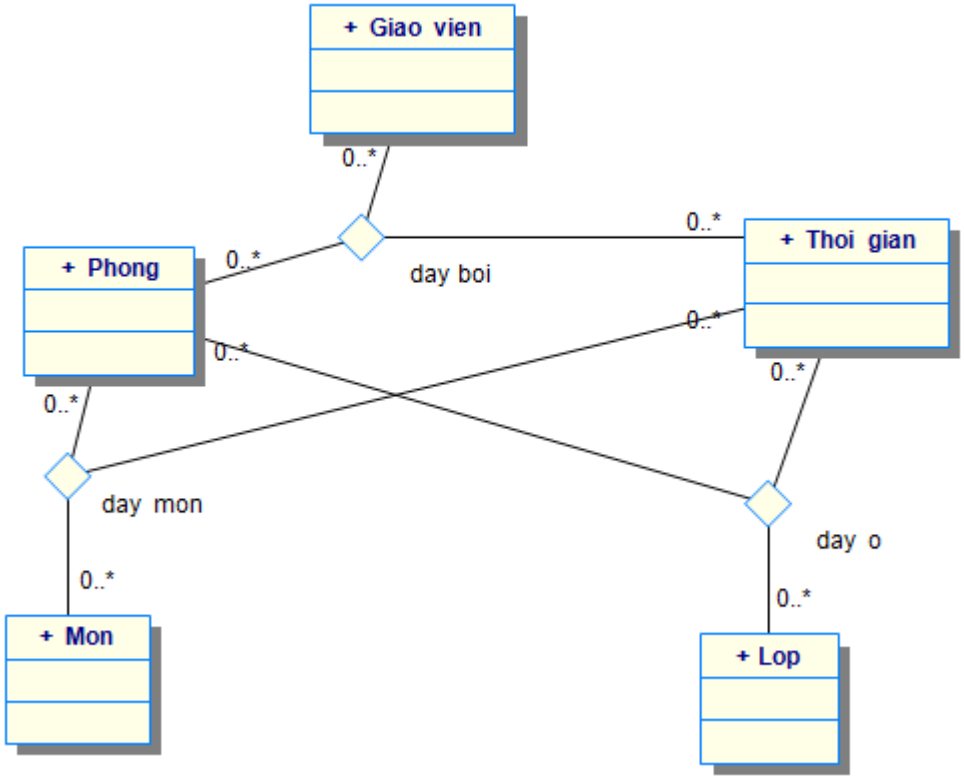
Liên kết nhiều chiều

* **Ví dụ:** Về thời khóa biểu dạy học, có nhiều cách mô hình hóa:
* ***Cách 1:*** Dùng liên kết nhiều chiều nối với tất cả các lớp liên quan:



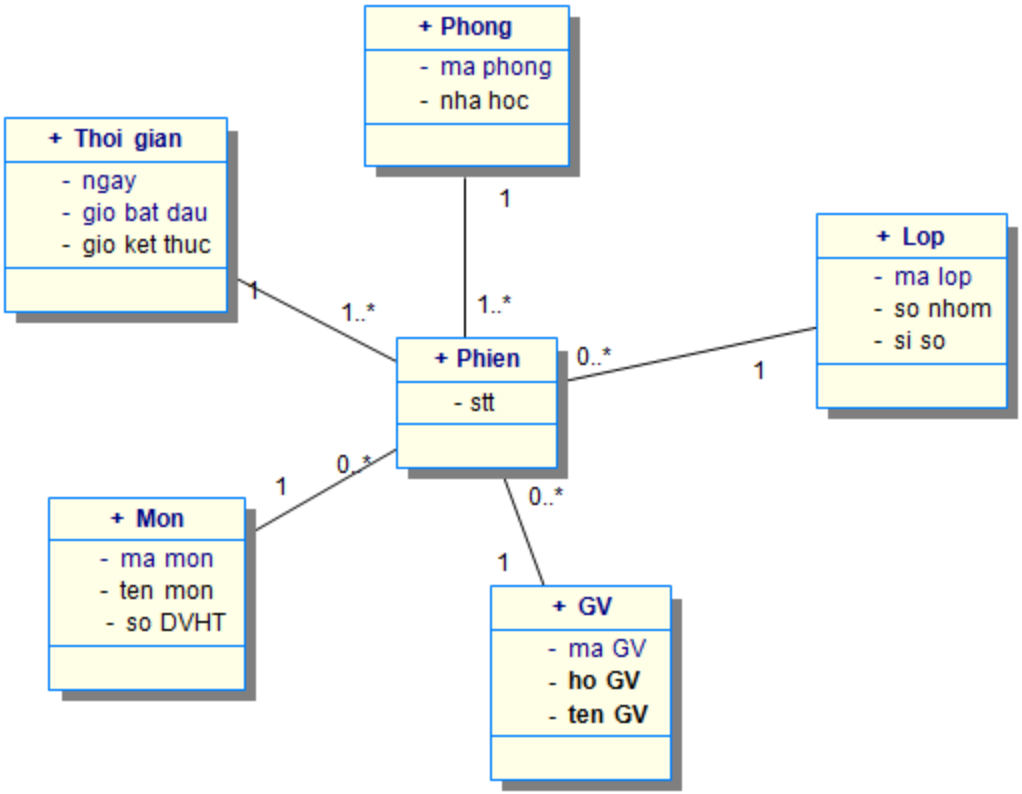
Liên kết nhiều chiều

* ***Cách 2:*** Khuynh hướng hiện nay là phá liên kết nhiều chiều ra thành các liên kết ít chiều hơn



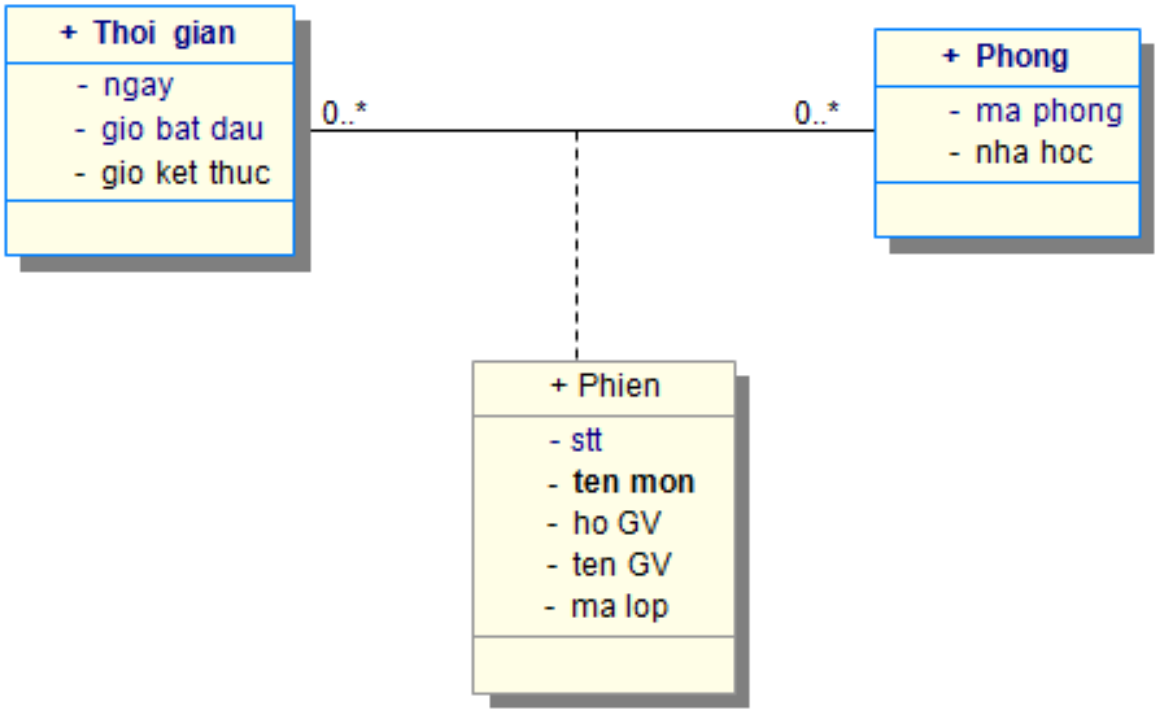
Liên kết nhiều chiều

* ***Cách 3*: Chuyển nút ở liên kết nhiều chiều thành lớp mới**



Liên kết nhiều chiều

* ***Cách 4:*** Khi lớp, giáo viên và môn không quan trọng lớp trong ngữ cảnh đang xét thì chuyển chúng chúng từ lớp thành thuộc tính trong lớp liên kết:

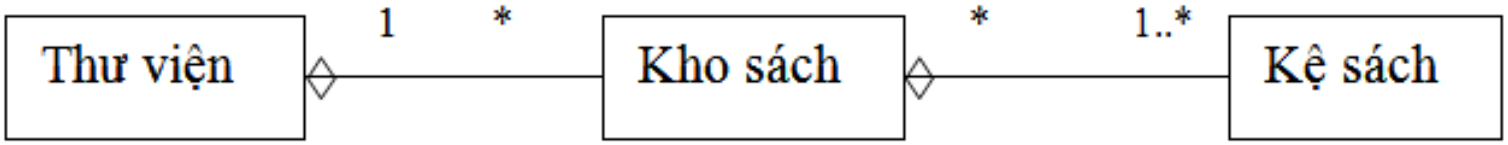


Quan hệ kết tập

* Một kết tập là một trường hợp đặc biệt của liên kết không đối xứng biểu diễn một mối quan hệ « chứa đựng » về cấu trúc hoặc hành vi của một phần tử trong một tập hợp
* Không như liên kết, quan hệ kết tập có tính truyền
* Quan hệ kết tập cũng cho phép việc ủy thác về tác tử: một tác tử có thể được thực hiện trên một lớp kết tập (thực tế được thực hiện trên các lớp thành phần của nó)

Quan hệ kết tập

* Chu kỳ sống của lớp kết tập là độc lập với các lớp thành phần của nó
* Một thể hiện của lớp thành phần có thể xuất hiện trong nhiều thể hiện của lớp kết tập
* **Ký hiệu:** có hình thoi rỗng trên liên kết về phía lớp biểu diễn tập hợp chứa đựng

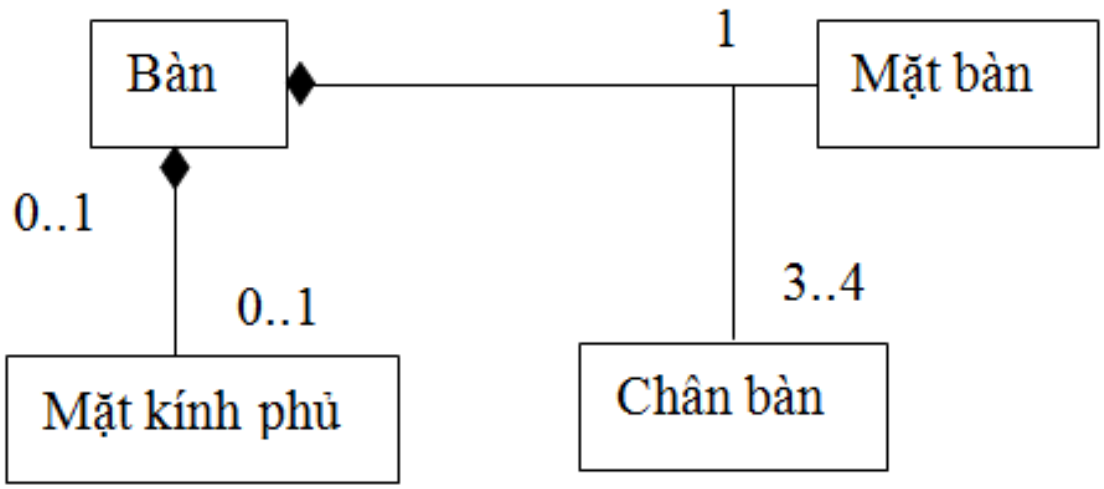


Quan hệ cấu thành

* Quan hệ cấu thành (còn được gọi là quan hệ kết tập phức hợp)
  * Là một quan hệ kết tập đặc biệt
  * Nó mô tả một sự chứa đựng về cấu trúc giữa các thể hiện
* Lớp chứa sẽ chịu trách nhiệm tạo ra, sao chép và xóa các lớp thành phần của nó
* Việc sao chép hoặc xóa đi lớp chứa sẽ kéo theo sao chép hoặc xóa các lớp thành phần của nó

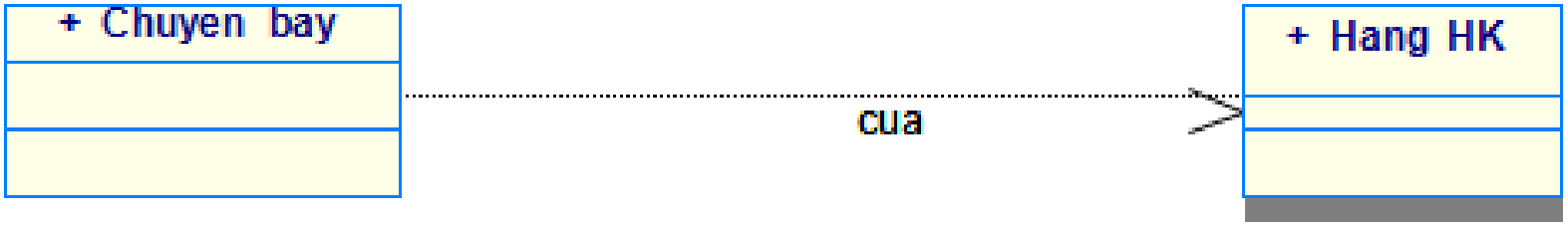
Quan hệ cấu thành

* Một thể hiện của lớp thành phần chỉ thuộc về duy nhất một thể hiện của lớp chứa nó
* **Ký hiệu:** hình thoi đặc trên liên kết ở phía lớp chứa
* **Ví dụ:**



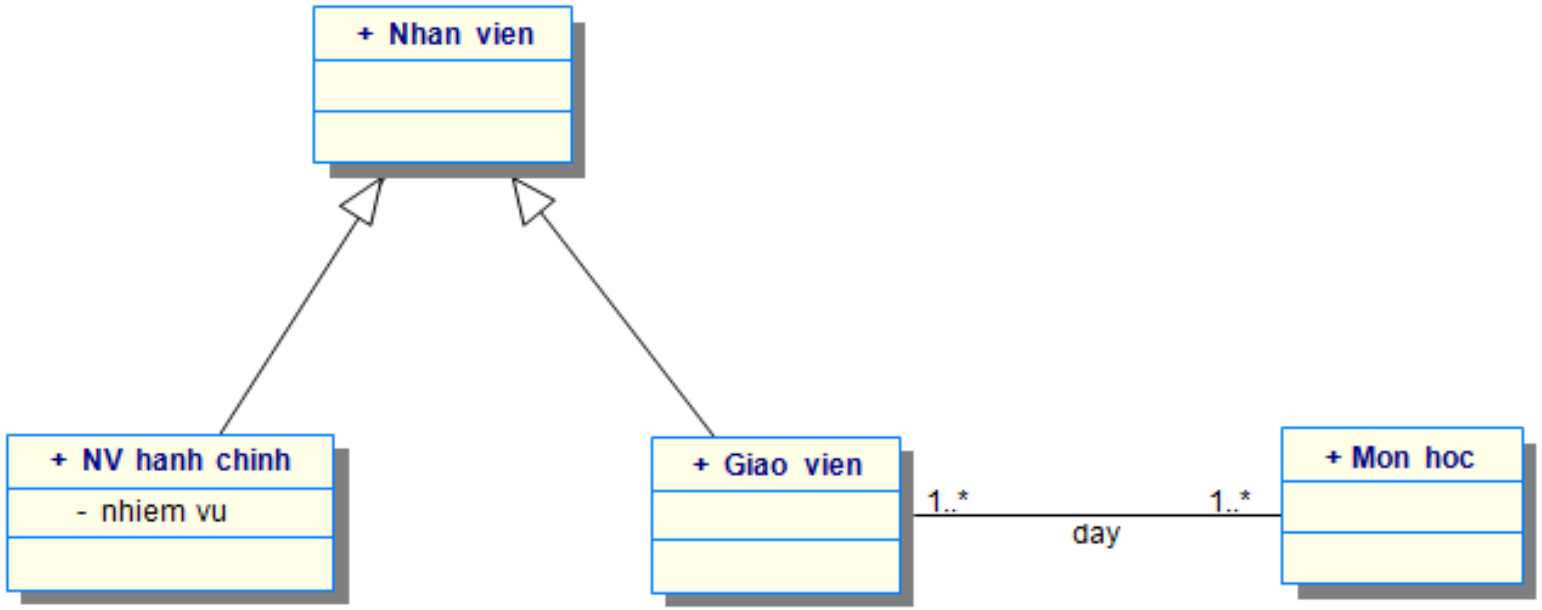
Quan hệ phụ thuộc

* Biểu biễn cho tình huống trong đó: một sự thay đổi của thành phần đích(target) có thể đòi hỏi sự thay đổi của thành phần nguồn(source)
* **Ký hiệu:** đường gạch đứt nét có mũi tên từ lớp chịu phụ thuộc
* **Ví dụ:**



Quan hệ thừa kế

* **Ký hiệu:** đường gạch có mũi tên rỗng hướng về lớp cha
* **Ví dụ:**

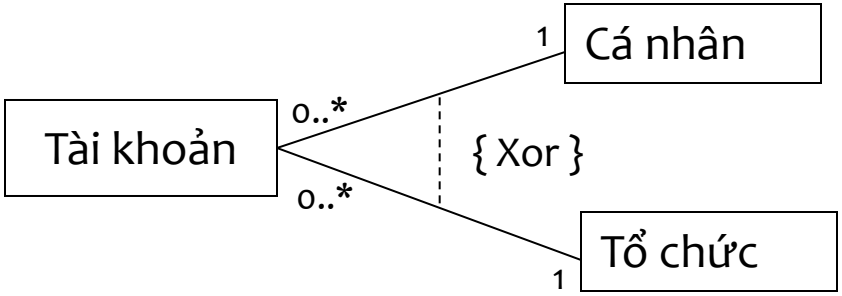


Ràng buộc

* Các ràng buộc có thể biểu diễn bằng ngôn ngữ tự nhiên, một ngôn ngữ lập trình, biểu thức toán học … hoặc ngôn ngữ OCL đi kèm theo UML
* Cụ thể có các dạng ràng buộc sau được biểu diễn bằng OCL:
  * Các qui tắc thừa kế: {complete}, {incomplete}, {overlaps}, {distinct} …
  * Hạn chế tầm vực của một liên kết: {subset}, {xor} …
  * Cách thức phát triển các đối tượng: {frozen}, {addOnly} …
  * Tổ chức các đối tượng:{ordered}, {frozen}, …

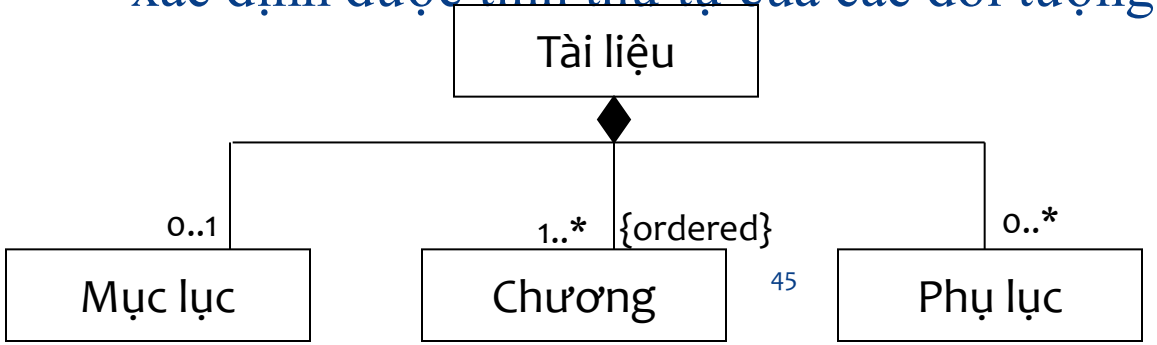
Ràng buộc

- **Ràng buộc Xor:**
  - Biểu diễn cho tình huống trong đó chỉ có một trong các kết hợp có giá trị tại một thời điểm



Mỗi tài khoản hoặc chỉ của cá nhân hoặc của tổ chức chứ không thể của cả hai

- **Ràng buộc ordered:**
  - Khi thành phần đích của kết hợp có bản số lớn hơn một, có thể xác định được tính thứ tự của các đối tượng được kết hợp



Các chương của tài liệu là có thứ tự

Xây dựng một sơ đồ lớp

* Các quan điểm mô hình hóa
* Các bước xây dựng

Các quan điểm mô hình hóa

* **Top-down:** Phân giải dần từ tổng quát xuống chi tiết
* **Bottom-up:** Sau khi có sơ đồ chi tiết ở tất cả các khóa cạnh, mới nhóm lại dần thành các phân hệ riêng dựa trên mối tương quan chặt chẽ giữa các lớp

Các bước xây dựng

* **1. Tìm các lớp của lĩnh vực chức năng :**
  * Tìm các đối tượng và lớp trong thế giới thực
    * Lớp trong thế giới thực
    * Lớp con trong thế giới thực
  * Chuyển đổi từ các đối tượng trong thế giới thực sang đối tượng dữ liệu
    * Lớp trong thế giới dữ liệu
    * Quản trị sự phức tạp
    * Ánh xạ cho các lớp
  * Chọn lựa giữa lớp và thuộc tính
  * Lớp dữ liệu bổ sung
  * Tìm các cấu trúc kết tập và lớp con

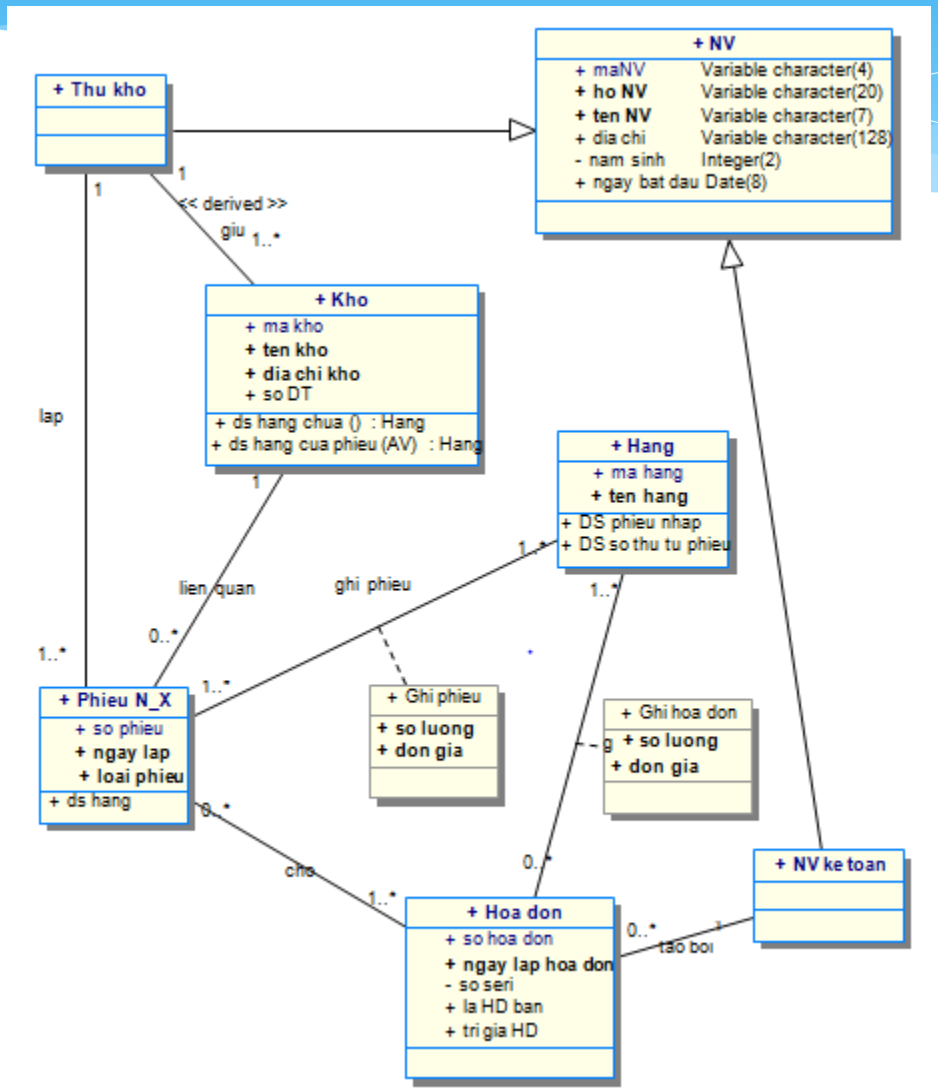
Các bước xây dựng

* **2. Tìm các mối liên kết giữa các lớp**
* **3.Tìm các thuộc tính của mỗi lớp**
* **4. Tổ chức lại và đơn giản hóa sơ đồ**
  * Bằng cách sử dụng sự tổng quát hóa
* **5. Thử các đường truy xuất đến các lớp**

Các bước xây dựng

* **Ví dụ:**

Quản lý hàng tồn

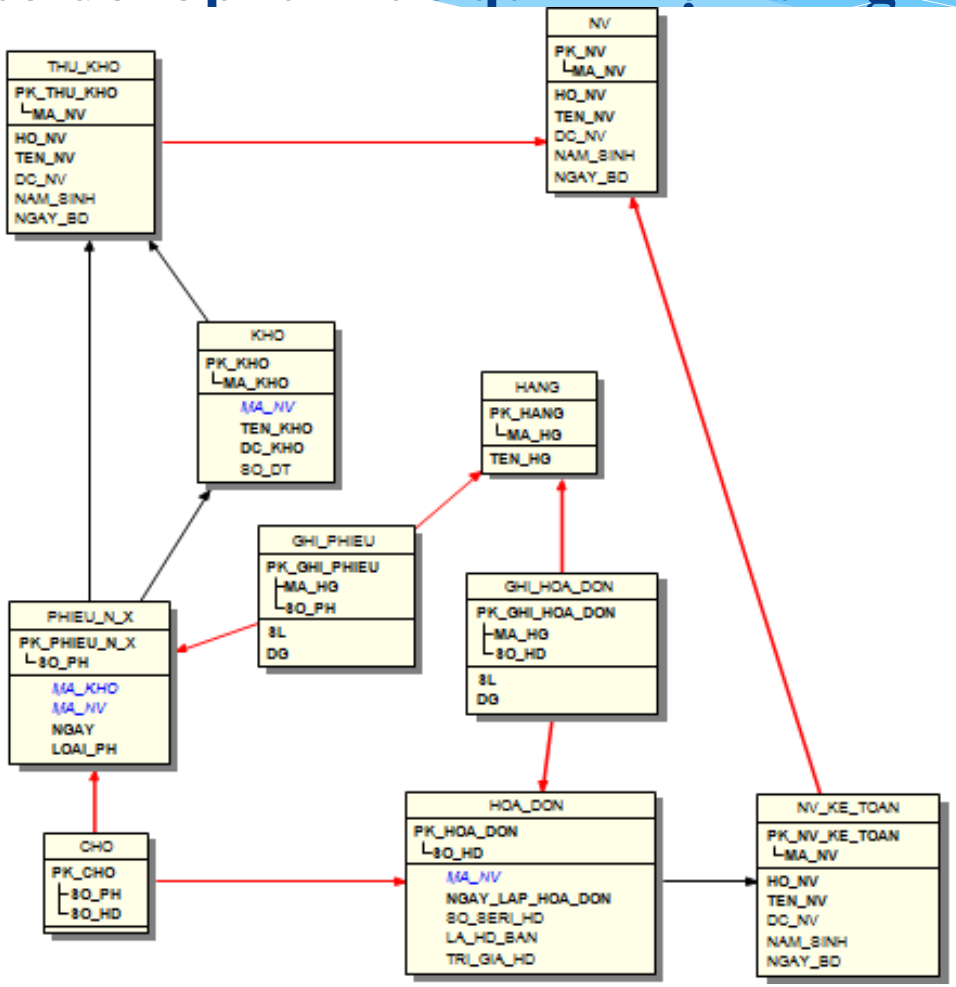


Các bước xây dựng

* **6. Tinh lọc hóa sơ đồ**
* Khi so khớp lại với người sử dụng, hoặc với sơ đồ hoạt vụ, ta có thể :
  * Bớt đi liên kết giữa các lớp, nếu đó là liên kết suy diễn, hoặc không tương ứng với nhu cầu trong thế giới thực ...
  * Chuyển lớp sang thuộc tính nếu nó không đóng vai trò thực sự quan trọng trong thế giới thực, hoặc không cần có dạng bảng mã…
  * Ngược lại, cũng có thể thêm liên kết
  * Chuyển một thuộc tính sang thành một lớp, chẳng hạn khi muốn đưa vào bảng mã tương ứng để hỗ trợ nhập liệu, hạn chế dữ liệu sai

Các bước xây dựng

* **7. Chuyển sơ đồ lớp từ mức quan niệm sang mức luận lý**
  * **Ví dụ :**

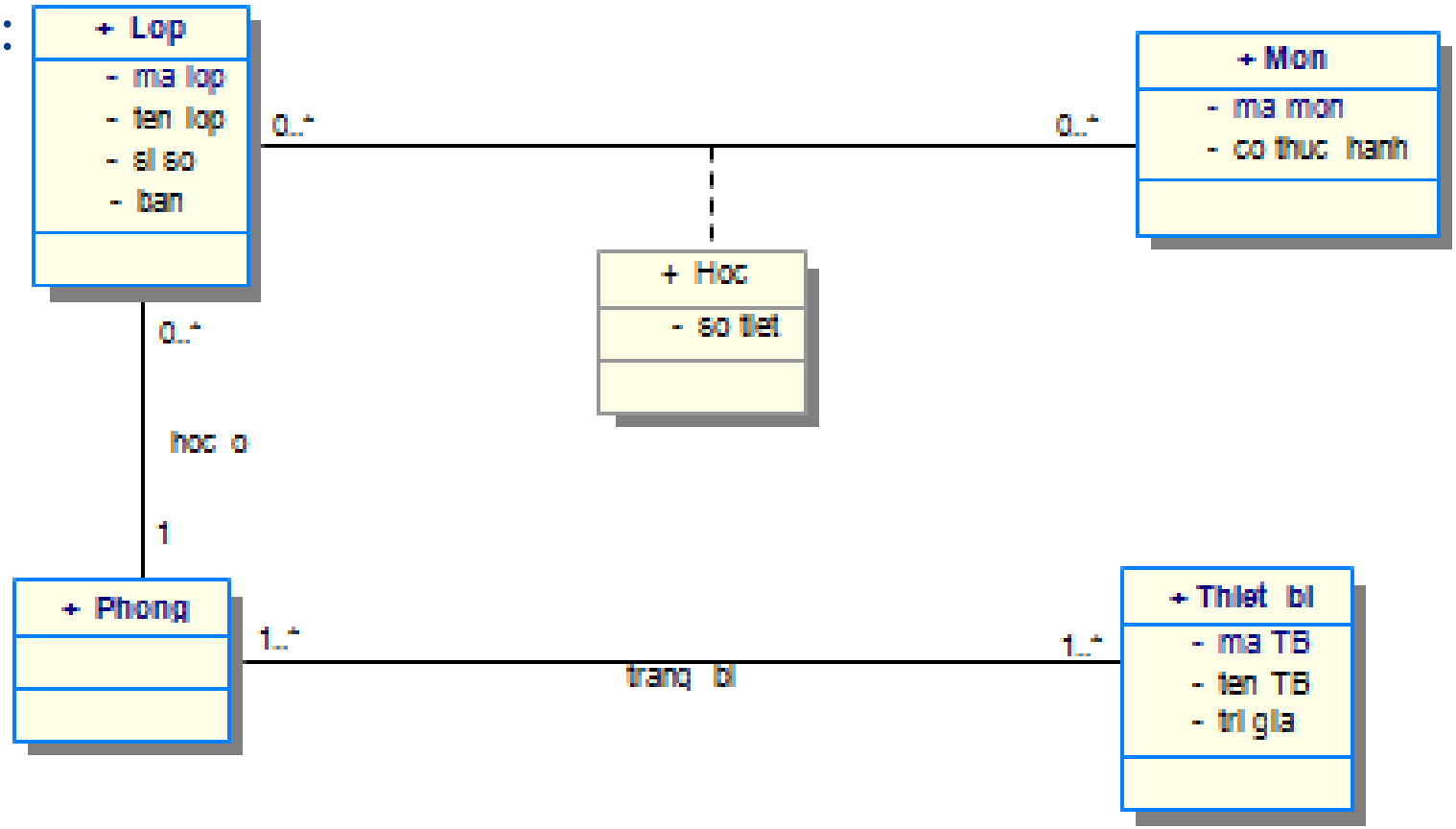


Các bước xây dựng

* **8. Thiết lập các phương thức cho mỗi lớp**
  * Các phương thức sẽ được bổ sung đầy đủ và chính xác sau khi so khớp với các sơ đồ khác như sơ đồ hoạt vụ, sơ đồ tuần tự hoặc sơ đồ cộng tác, sơ đồ hoạt động…
  * Ngược lại, sơ đồ hoạt vụ cũng sẽ được bổ sung sau khi lập các phương thức
  * Phương thức được thiết lập có thể bao gồm các phương thức lớp (class method, static method) hoặc không

Các bước xây dựng

* **Ví dụ:** Thêm các phương thức cho lớp "Lop" trong sơ đồ lớp sau:



Các bước xây dựng

* **9. Đóng gói**
  * Như ở sơ đồ hoạt vụ