

## Chương 3

# SƠ ĐỒ LỚP

( CLASS DIAGRAM )

### 3.1 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. Nó được dùng để mô tả cấu trúc bên trong, tính của hệ thống.

Sơ đồ lớp không dùng để chỉ ra cách thức làm thế nào sử dụng các tác từ (operation). Trách nhiệm đó sẽ do sơ đồ tương tác.

### 3.2 LỚP VÀ CÁC KHÁI NIỆM LIÊN QUAN

Có nhiều loại lớp: lớp thực thể (entity class), lớp điều khiển (control class), lớp biên (boundary class). Loại lớp phổ biến nhất và hay được coi mặc nhiên, là lớp thực thể.

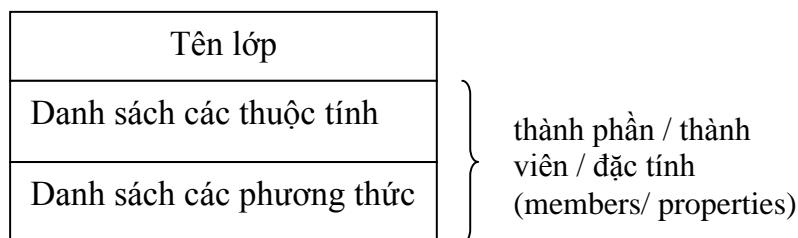
Thông thường, khi đề cập đến sơ đồ lớp, nếu không nói rõ ở mức nào của phân tích thiết kế, tức theo qui ước nói ở mức quan niệm (conceptual level).

Sơ đồ lớp ở mức quan niệm sau đó sẽ được chuyển sang mức luận lý (logical level), rồi vật lý (physical level) và được cài đặt trong một hệ quản trị cơ sở dữ liệu hướng đối tượng cụ thể, hoặc được tạo ra bởi một hệ thống các trình ứng dụng qua một ngôn ngữ lập trình.

#### 3.2.1 Lớp (class/ classe)

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 với các lớp khác.

Ký hiệu:



Hình 3.1 Ký hiệu lớp

- Tên lớp phải có nghĩa và ký tự đầu tiên phải viết hoa, phải là danh từ. 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, và phải là danh từ.

- Mỗi phương thức (method) được mô tả kiểu trả về (return type) nếu có, danh sách các đối số (parameter) và kiểu tương ứng của mỗi đối số (parameter type).

Các khái niệm thuộc tính và phương thức sẽ được trình bày kỹ hơn ở các mục tiếp theo.

Ví dụ :

Sinh_viện
họ: VC (18)
tên: VC (7)
ngày_sinh: Date
phái: {'M', 'F'}
lớp: VC(8)
đkýMôn(m: C);
đkýNhóm(): I;

**Hình 3.2 Ví dụ về lớp**

*Chú thích:* VC = Variable character: chuỗi ký tự có chiều dài biến đổi

C = Character : chuỗi ký tự có chiều dài cố định

D = Date: kiểu ngày (gồm ngày/ tháng/ năm)

I = Integer : số nguyên

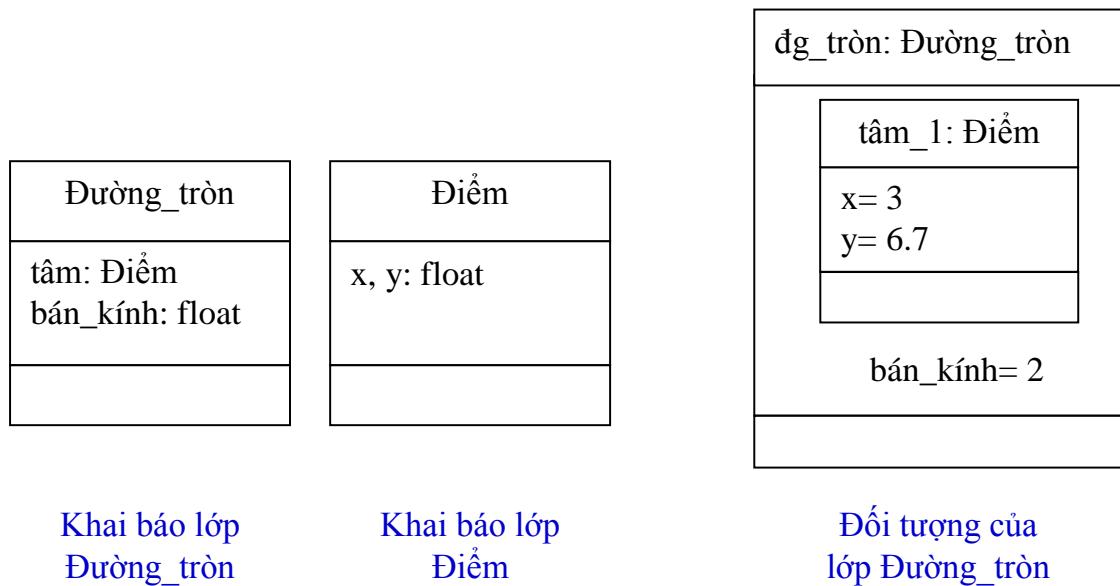
Lớp ban đầu chỉ gồm các thuộc tính có kiểu căn bản (primitive type) như số nguyên, số thực, ký tự, chuỗi. Tuy nhiên, về sau, do thiết kế ở mức luận lý, lớp đó có thể có thuộc tính mang kiểu là một lớp khác, lúc đó nó được gọi là lớp phức (composite class).

### 3.2.2 Đối tượng (object/ objet):

Một đối tượng là một thể hiện (instance) của lớp. Các đối tượng được mô tả theo cấu trúc của lớp. Mỗi đối tượng sẽ chiếm vùng nhớ như nhau theo đúng kích thước qui định bởi cấu trúc định sẵn đó.

Mỗi đối tượng có một định danh (OID: object identifier). OID cũng có tính chất duy nhất như khóa, nhưng trên toàn CSDL chứ không chỉ trên lớp đang xét. OID do hệ quản trị cơ sở dữ liệu gán tự động, thường rất dài và rất khó nhớ.

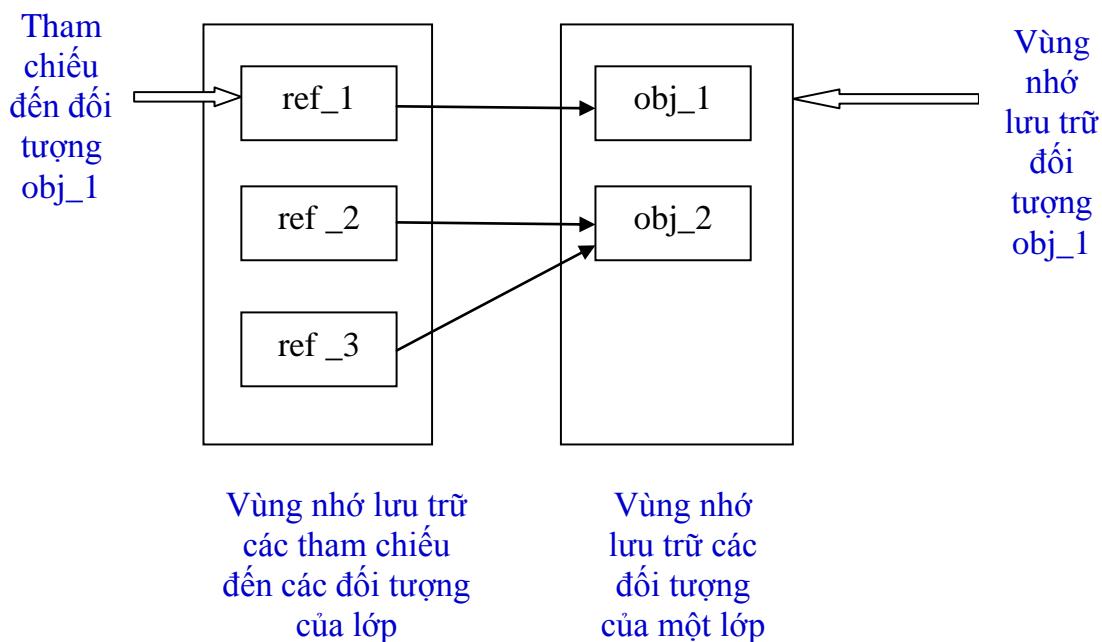
Đối tượng ban đầu chỉ gồm các giá trị có kiểu căn bản (primitive type) như số nguyên, số thực, ký tự, chuỗi theo đúng kiểu của thuộc tính tương ứng. Đối tượng phức (composite object) của lớp phức có chứa đối tượng của lớp khác.

**Hình 3.3 Ví dụ về đối tượng phức**

### 3.2.3 Tham chiếu (reference/ référent) đến một đối tượng của lớp

Tham chiếu đến một đối tượng của một lớp có thể được xem tương tự như con trỏ, sẽ trỏ đến vùng nhớ chứa một đối tượng của lớp đó.

Các tham chiếu được lưu trong một vùng nhớ khác. Mỗi tham chiếu chứa địa chỉ của đối tượng mà nó trỏ đến. Tên tham chiếu là duy nhất, nhưng trị của tham chiếu có thể trùng nhau.

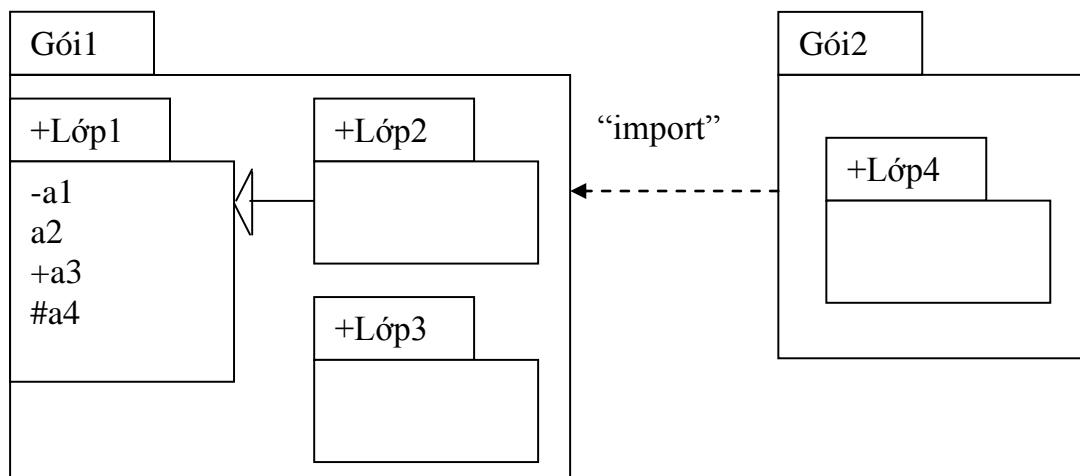
**Hình 3.4 Tham chiếu đến đối tượng của lớp**

### 3.2.4 Sự bao gói (encapsulation) và mức độ hiển thị (visibility) vision

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:

- 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 :



Hình 3.5 Ví dụ về sự bao gói và mức độ hiển thị

Trong ví dụ trên :

- a1 chỉ được Lớp1 nhìn thấy,
- a2 được các lớp cùng gói nhìn thấy, đó là các lớp Lớp1, Lớp2 và Lớp3.
- a3 được tất cả các lớp nhìn thấy,
- a4 chỉ được Lớp1 và Lớp2 nhìn thấy.

### 3.2.5 Thuộc tính (attribute)

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>]

Trong đó:

*<mức độ hiển thị>*: dùng các từ khóa *public*, *private*, *protected* hoặc các dấu tương ứng

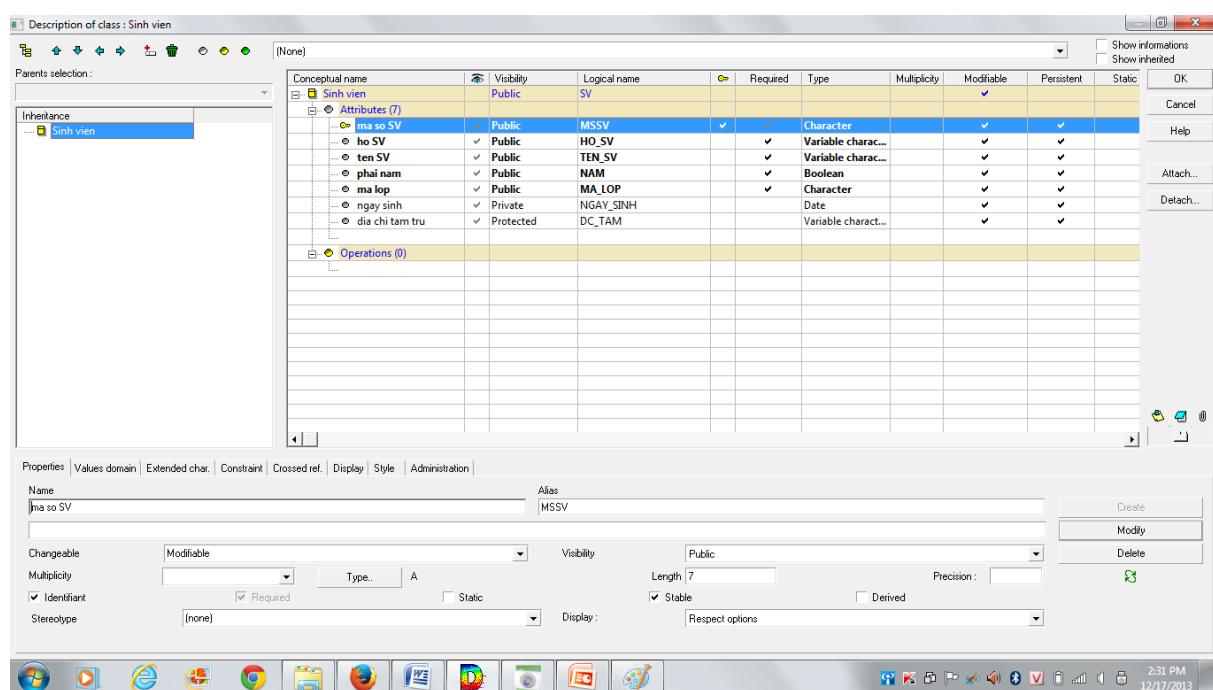
*<kiểu>*: kiểu dữ liệu cơ sở hoặc tên lớp. Ở mức quan niệm, nên chọn kiểu cơ bản. Khi sang mức luận lý, tùy theo mối liên kết giữa các lớp, có thể cân nhắc chọn tên lớp cho kiểu của thuộc tính.

*<bản số>*: các chỉ số tối thiểu và tối đa cho một mảng của kiểu vừa nói ở trên

*<trị khởi tạo>*: phải có kiểu tương ứng với kiểu nói trên.

+ Sinh viên	
⌚ + ma so SV	Character(7)
⌚ + ho SV	Variable character(18)
⌚ + ten SV	Variable character(7)
⌚ + phai nam	Boolean(1)
⌚ + ma lop	Character(8)
⌚ - ngay sinh	Date(8)
⌚ # dia chi tam tru	Variable character(128)

Hình 3.6 Ví dụ về thuộc tính trong lớp



Hình 3.7 Ví dụ về mô tả chi tiết cho thuộc tính trong lớp bằng công cụ

### 3.2.5.1 Thuộc tính của lớp (static attribute / class attribute)

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. Các đối tượng truy xuất được thuộc tính đó, nhưng không được sở hữu một bản sao của nó.

Đó chính là thuộc tính tĩnh mà trong Java và C++ dùng từ khóa *static* đứng trước tên của nó. 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.

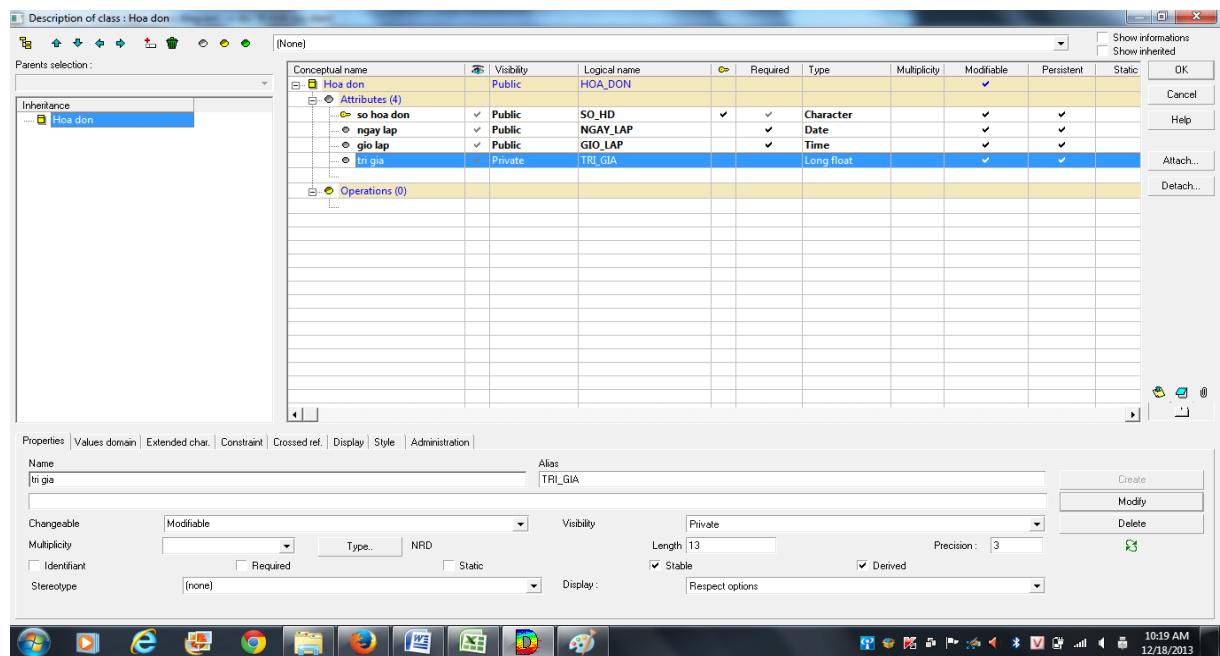
### 3.2.5.2 Thuộc tính do suy diễn (derived attribute)

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.

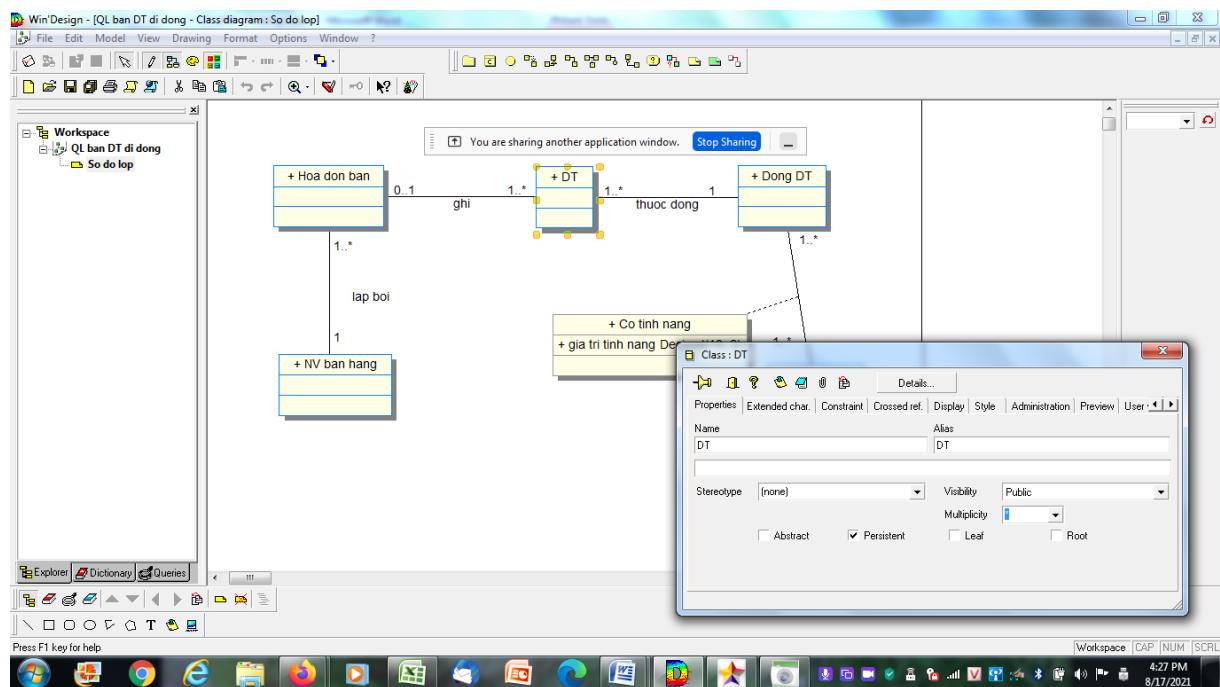
Thuộc tính do suy diễn được đánh dấu bởi dấu “/” (slash) đứng trước.

Ví dụ:

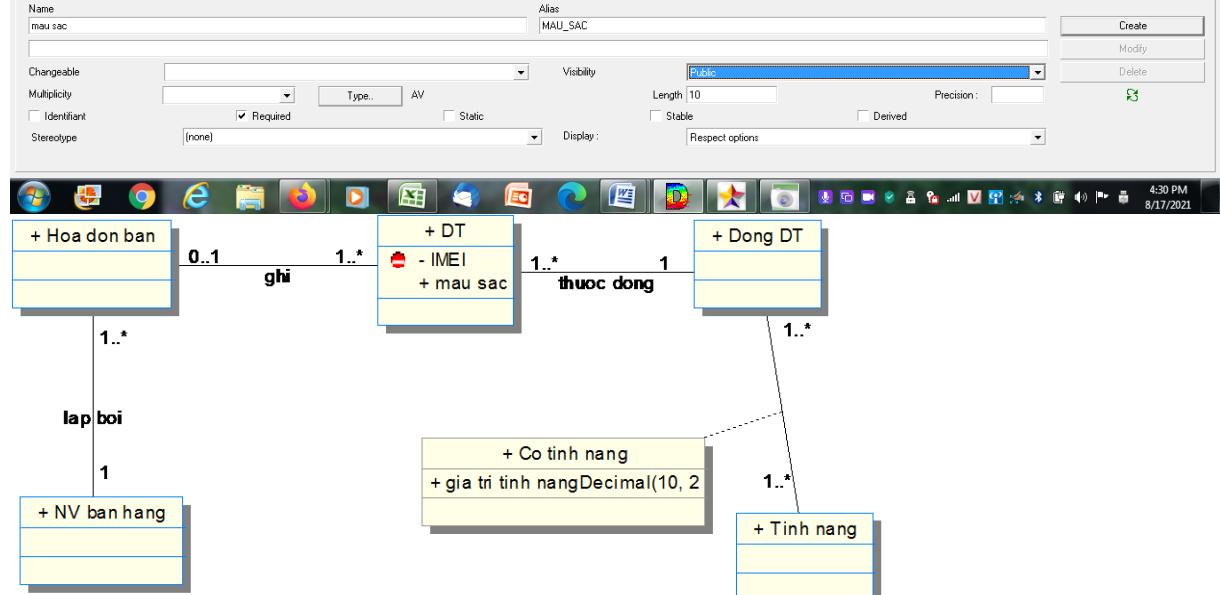
/tri\_gia: Integer;



**Hình 3.8 Thuộc tính suy diễn “trị giá” trong lớp “Hóa đơn”**  
(được check vào ô “”Derived”)



Nhấn vào lớp rồi vào “Details” trên cửa sổ vừa hiện ra.



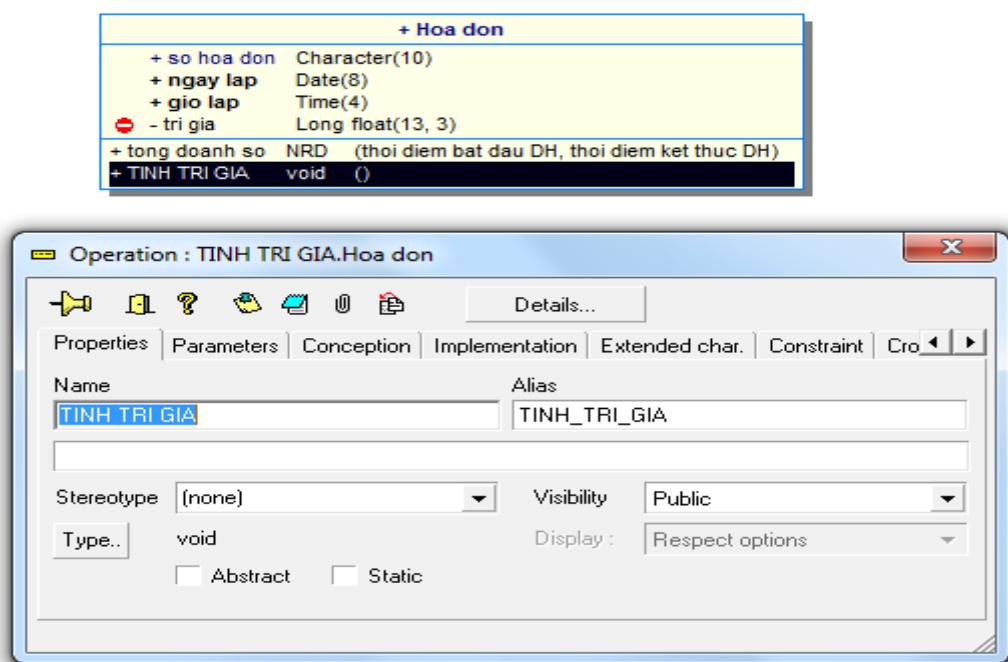
### 3.2.6 Phương thức (method)

Hành vi của một đối tượng được mô hình hóa bằng một tập các phương thức.

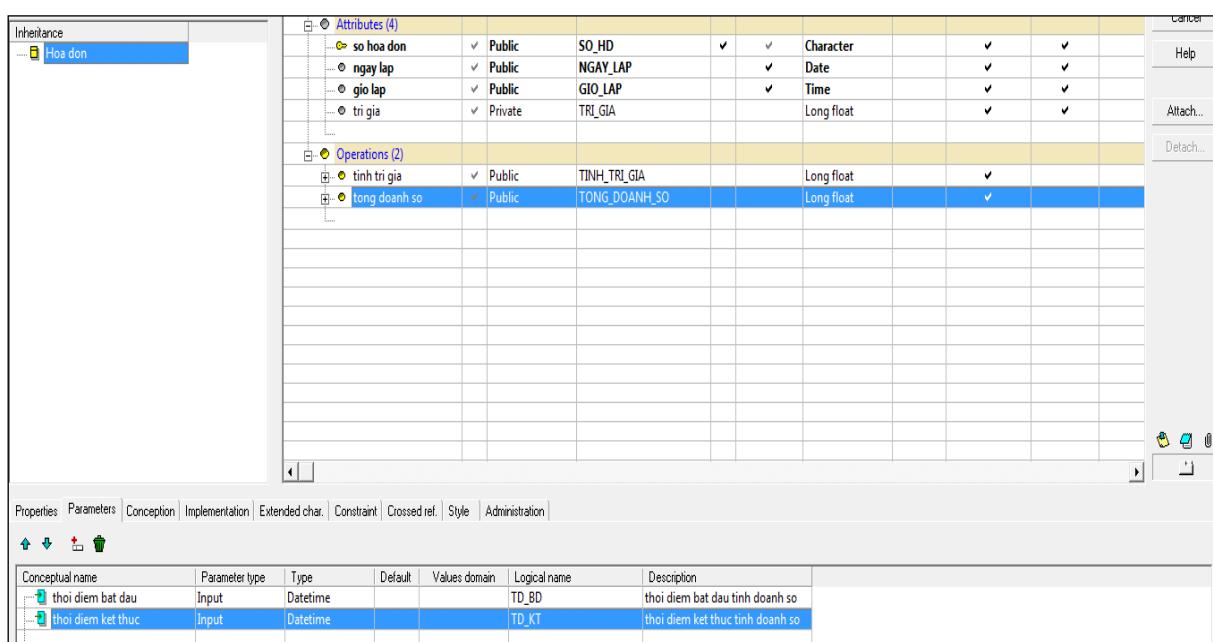
**Ví dụ:** ở lớp Hóa đơn, có phương thức

+ tính\_trí\_giá(): longint;

Phương thức có kiểu trả về thuộc loại hàm (function/ fonction), ngược lại được gọi là thủ tục (procedure/ procédure). Phương thức có thể có tham số (parameter/ paramètre), tham số phải có tên và kiểu.



Hình 3.9 Thủ tục “tính trị giá” trong lớp “Hóa đơn”



Hình 3.10 Hàm “tổng doanh số” có tham số trong lớp “Hóa đơn”

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”.

Dưới đây là ví dụ minh họa các cài đặt ban đầu cho các phương thức lớp “Hóa đơn”.

Ví dụ:

```
*****
 * Source File : Hoa_don.java
 * Author       : Xuan Loc
 * Project name : vi du cho UML* Created      : 18/12/2013
 * Modified    : 18/12/2013
 * Description : Definition of the class Hoa_don
*****/




import java.util.*;


public class Hoa_don
{
    //Inners Classifiers


    //Attributes

    public java.lang.Character so_hoa_don;
    public java.util.Date ngay_lap;
    public java.util.Date gio_lap;

    private java.lang.Double tri_gia;

    //Attributes Association


    //Operations

    public java.lang.Double tong_doanh_so
        (java.util.Date thoi_diem_bat_dau,
         java.util.Date thoi_diem_ket_thuc)
    {
        // TODO: implement

        return null;
    }

    public void TINH_TRI_GIA()
```

```

{
    // TODO: implement
}

} //End Class Hoa_don
```

### 3.2.7 Phương thức lớp, phương thức thành viên (class method, member method)

Phương thức lớp là phương thức phải được thực hiện trên nhiều đối tượng của lớp đang xét. Ngược lại, phương thức thành viên chỉ thực hiện trên đối tượng đang xét.

Ví dụ:

- Các hàm tạo (constructor, create), hàm hủy (destructor, destroy) là phương thức thành viên.
- Hàm tính tổng số đối tượng của một lớp là phương thức lớp.

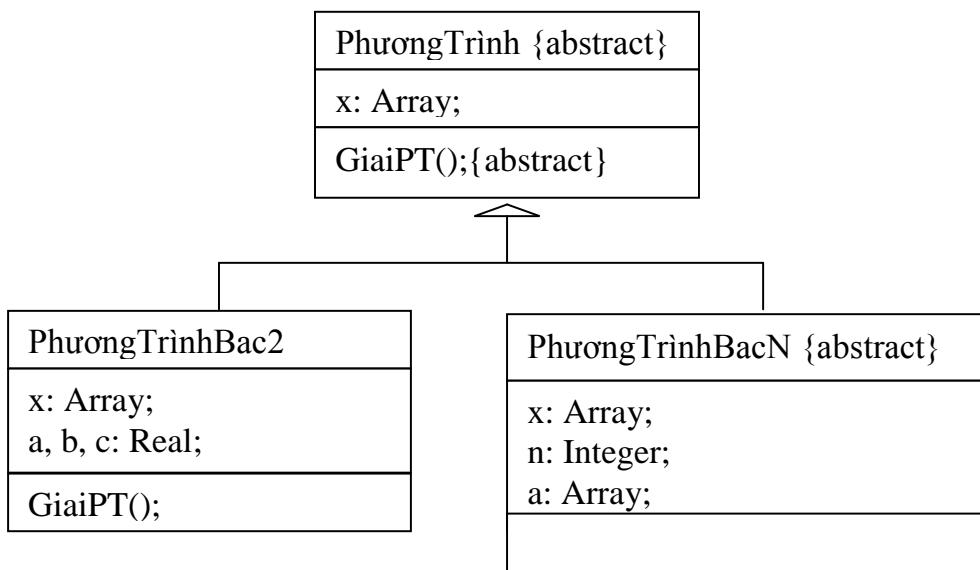
### 3.2.8 Phương thức trừu tượng và lớp trừu tượng (abstract method, abstract class)

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 cài đặt.

Tên của một lớp trừu tượng sẽ có thêm từ khóa « abstract » đặt giữa dấu móc {} như trong ví dụ *Hình 3.11*.

Ví dụ :



**Hình 3.11 Ví dụ về lớp trừu tượng và phương thức trừu tượng**

### 3.3 GIAO DIỆN (INTERFACE)

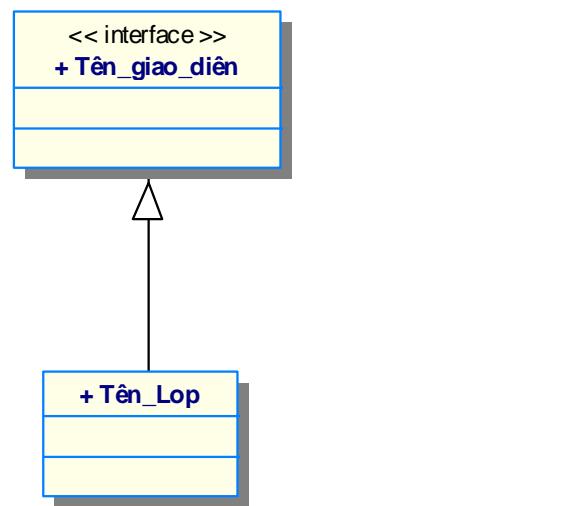
Giao diện là một phương tiện để phân ly giữa người dùng và cài đặt thực sự của lớp. Người ta trích ra từ một lớp các dịch vụ mà lớp đó có thể mang lại, khai báo

chúng trong giao diện. Các lớp khác và người dùng có thể gọi các dịch vụ của lớp thông qua giao diện như một trung gian làm “cò”. Giao diện sẽ gọi đến lớp để thực hiện thực sự dịch vụ đó.

Như vậy, 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. Ngược với lớp, giao diện không có thể hiện, tức không có đối tượng nào cả. Giao diện có thể mô tả chính xác điều kiện và kết quả việc kích hoạt nó.

Trong thực tế, điều này rất hữu dụng. Rất nhiều ngôn ngữ lập trình có dành phần khá lớn trong thư viện của mình cho giao diện.

Một giao diện có thể được chuyên biệt hóa hoặc tổng quát hóa bởi một giao diện khác. Hình 3.12 trình bày cách biểu diễn giao diện và mối liên hệ với lớp.



Hoặc:



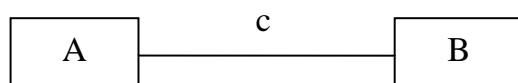
**Hình 3.12 Ký hiệu giao diện**

## 3.4 QUAN HỆ GIỮA CÁC LỚP

### 3.4.1 Liên kết (association)

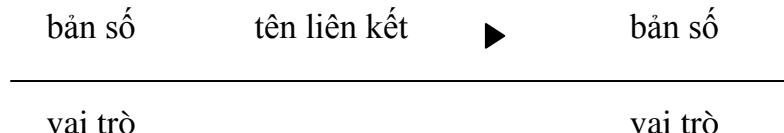
Một liên kết biểu diễn mối quan hệ ngữ nghĩa bền vững giữa 2 lớp.

Ký hiệu:



**Hình 3.13 Ký hiệu đơn giản của liên kết**

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

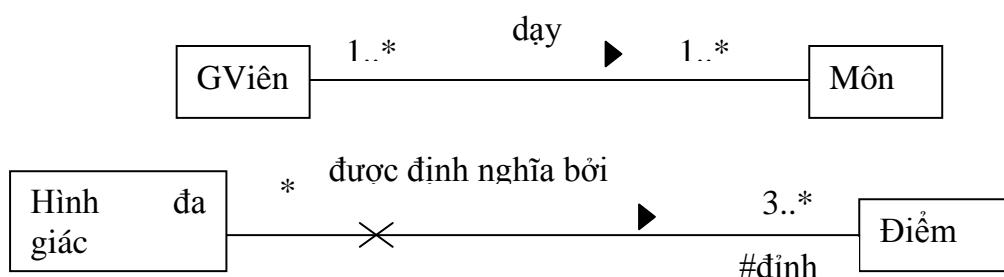


**Hình 3.14 Ký hiệu đầy đủ của liên kết**

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. Vai trò và mũi tên có thể được bỏ qua.

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:



**Hình 3.15 Các ví dụ về liên kết**

Trong ví dụ thứ hai, sự thông thương chỉ có chiều duy nhất từ lớp “Hình đa giác” sang lớp “Điểm”.

### 3.4.2 Tính bội (multiplicity)

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.

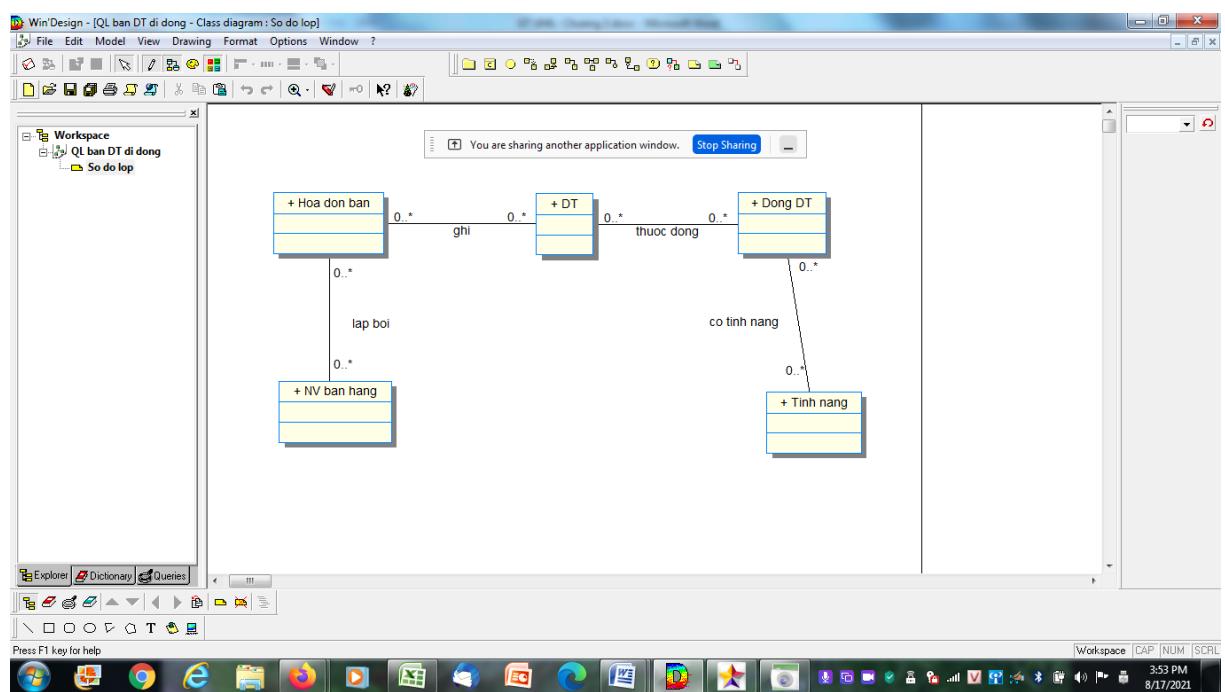
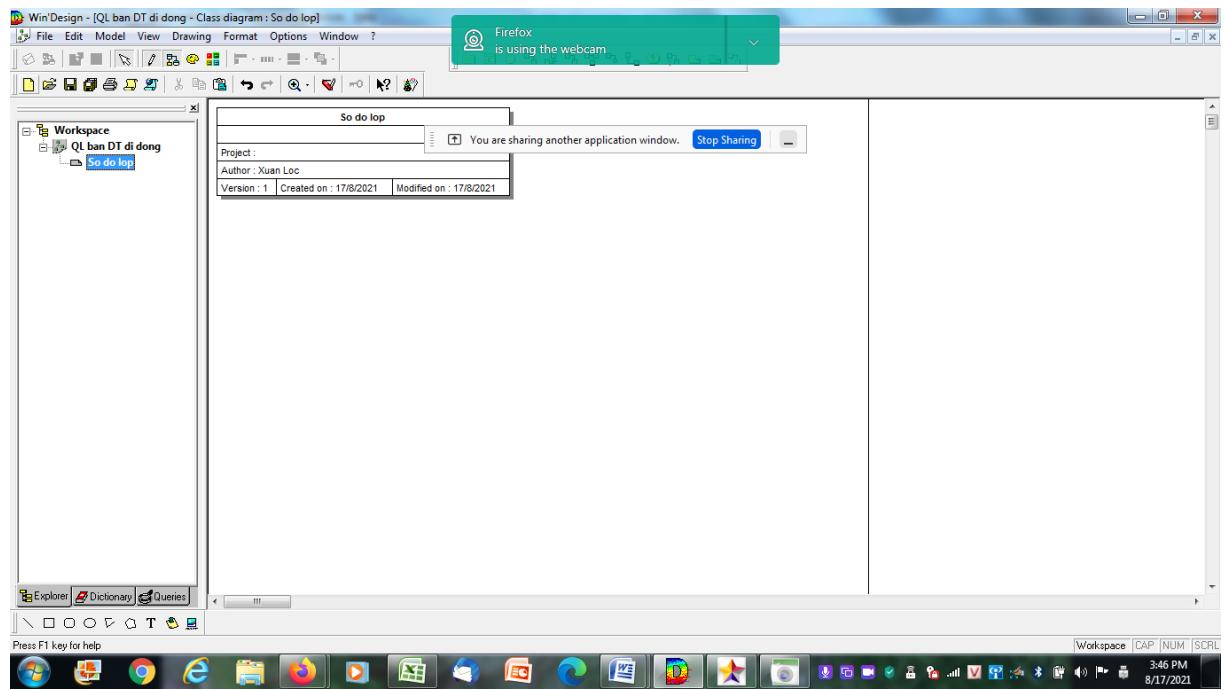
Chú ý: bản số của một lớp ở đầu này của liên kết được ký hiệu ở lớp đầu kia của liên kết.

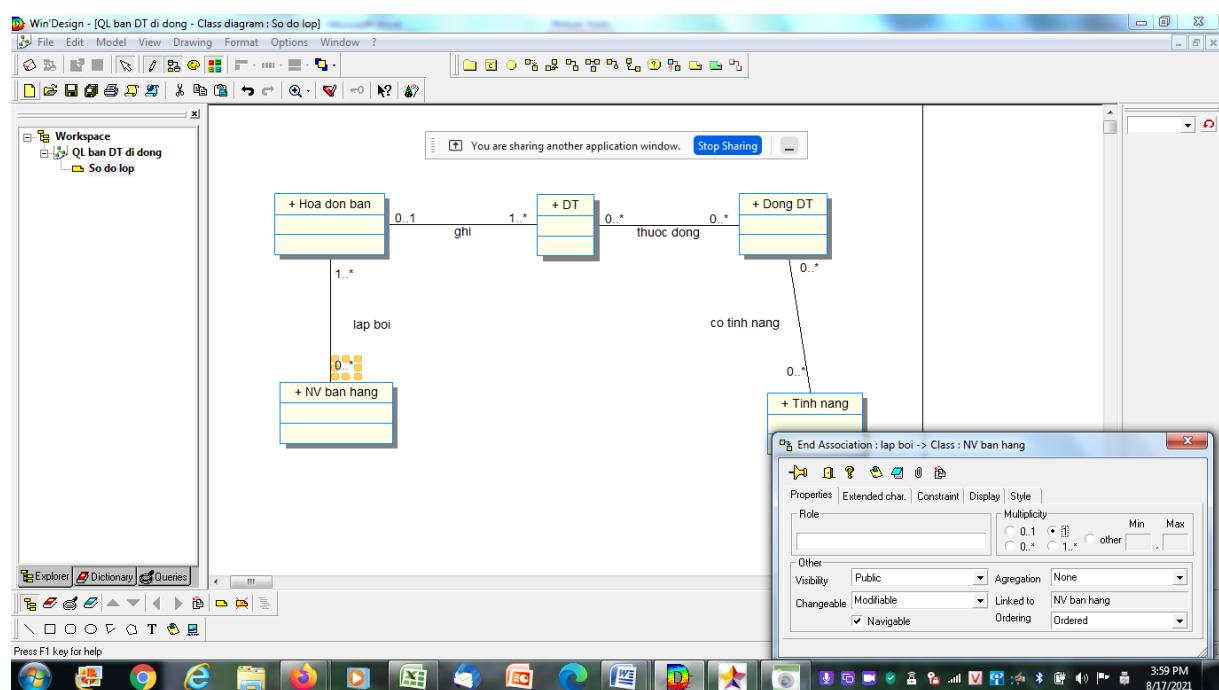
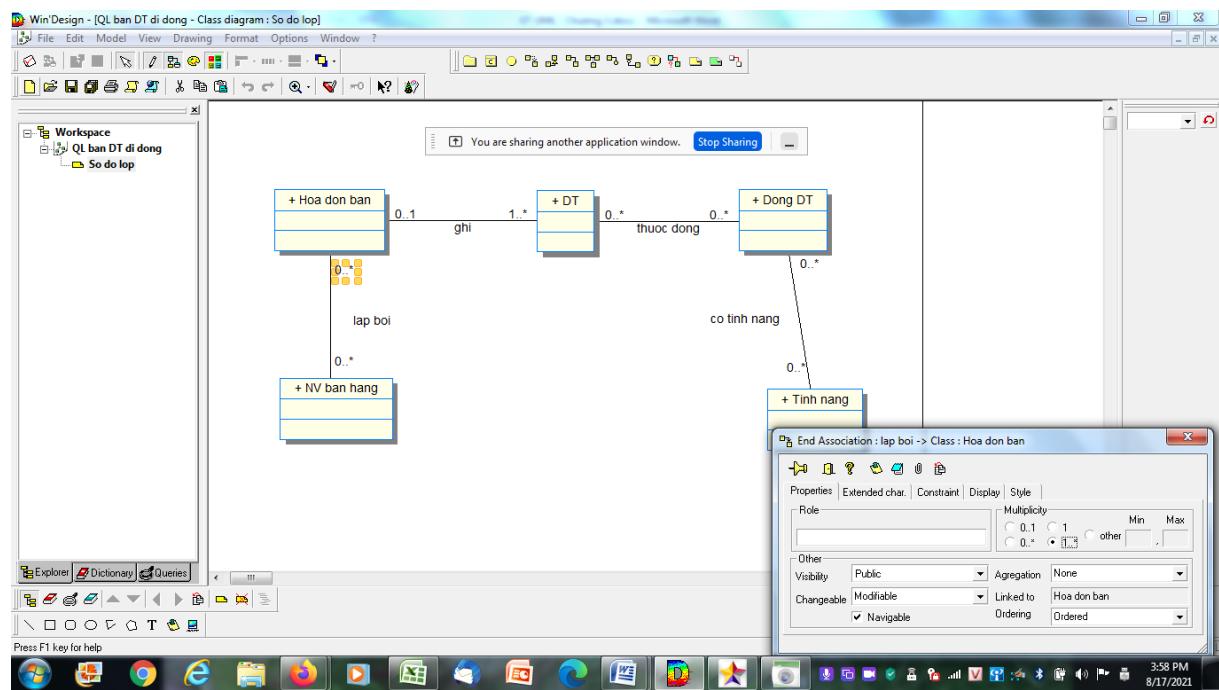
## Ví dụ:

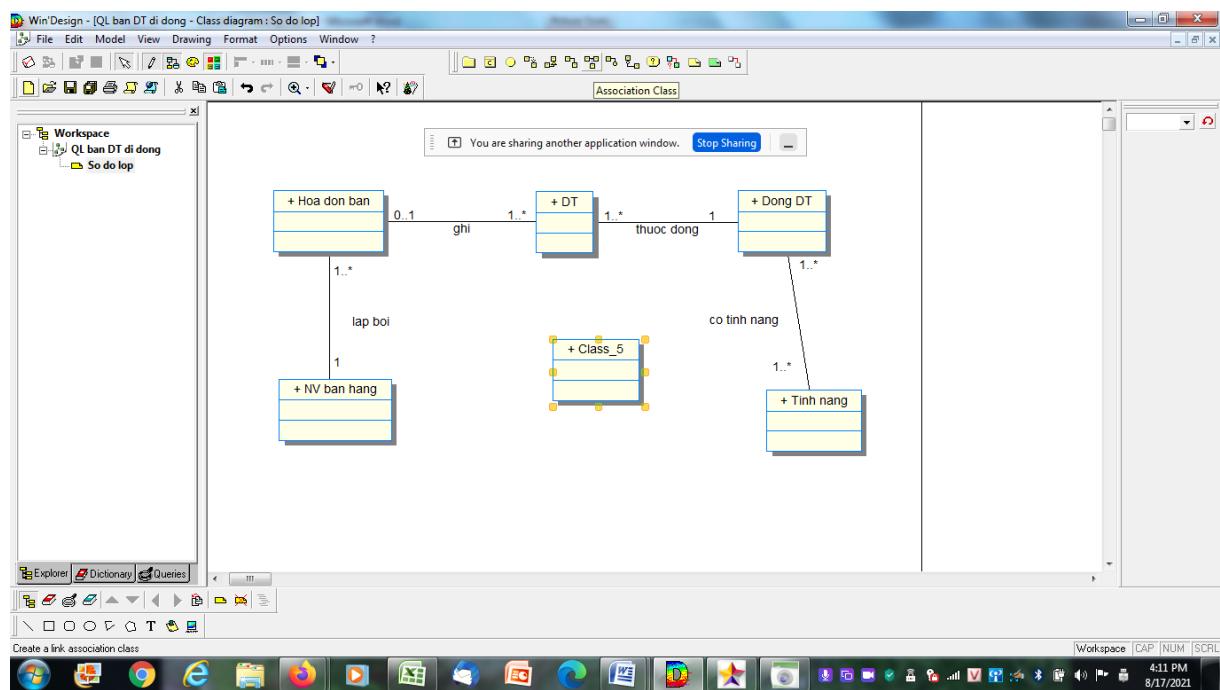
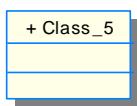
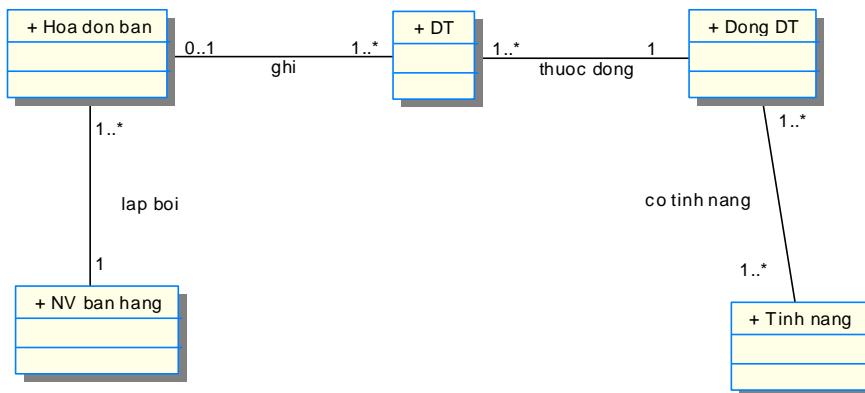


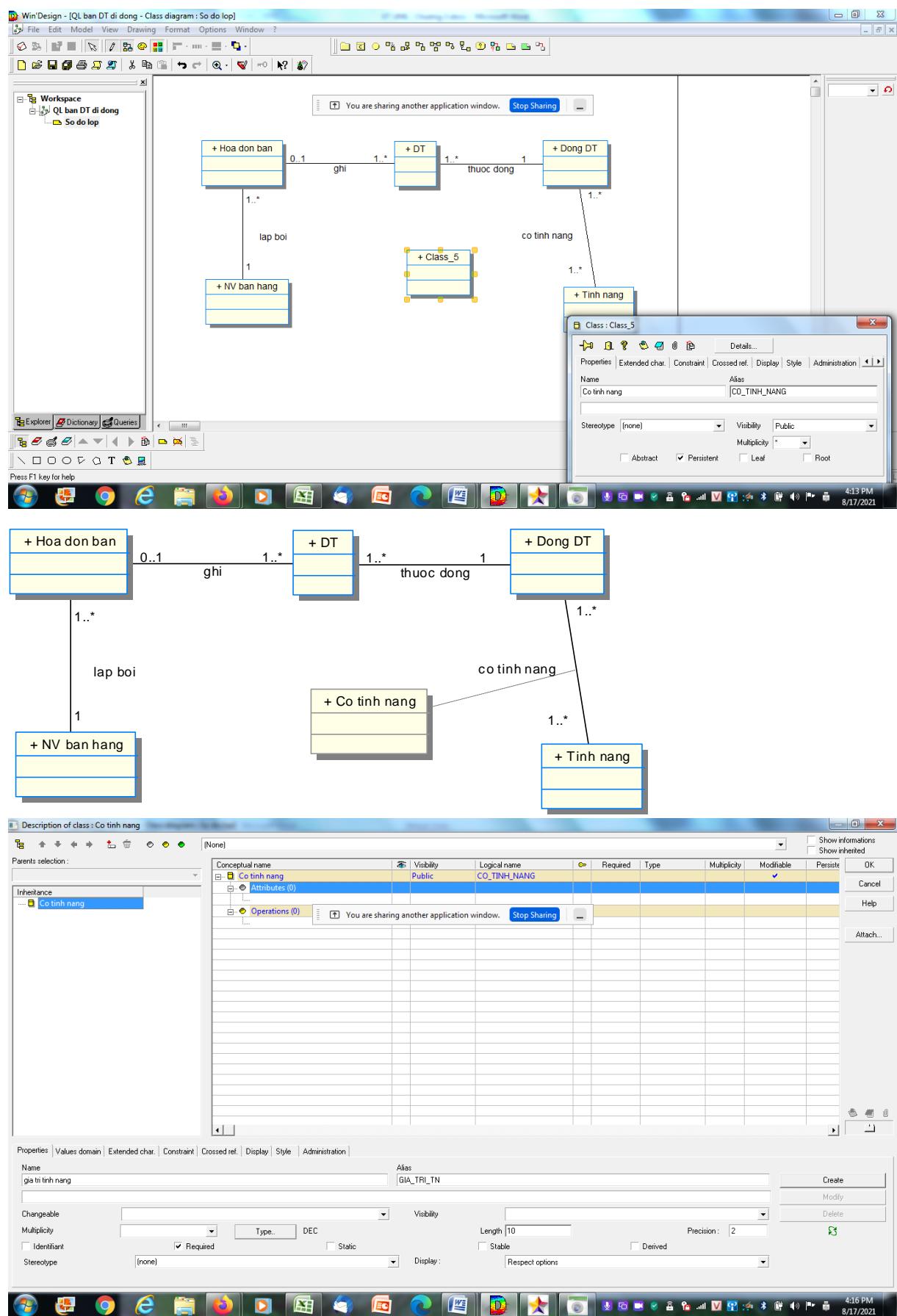
**Hình 3.16** Ví dụ về bản số

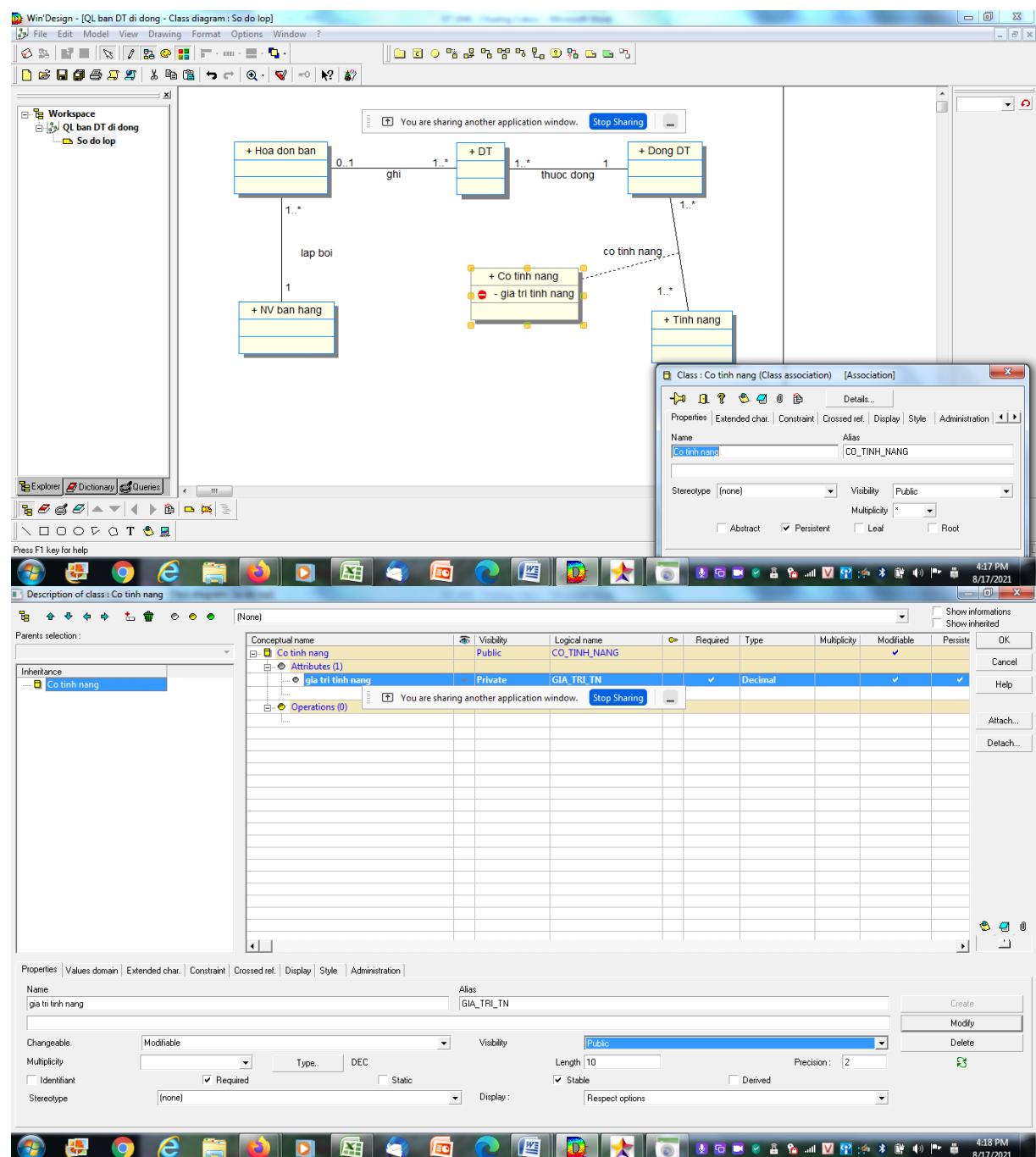
Ngữ nghĩa: 1 môn được dạy bởi ít nhất 1 giáo viên và nhiều nhất 4 giáo viên, trong khi 1 giáo viên chỉ dạy 1 môn duy nhất.

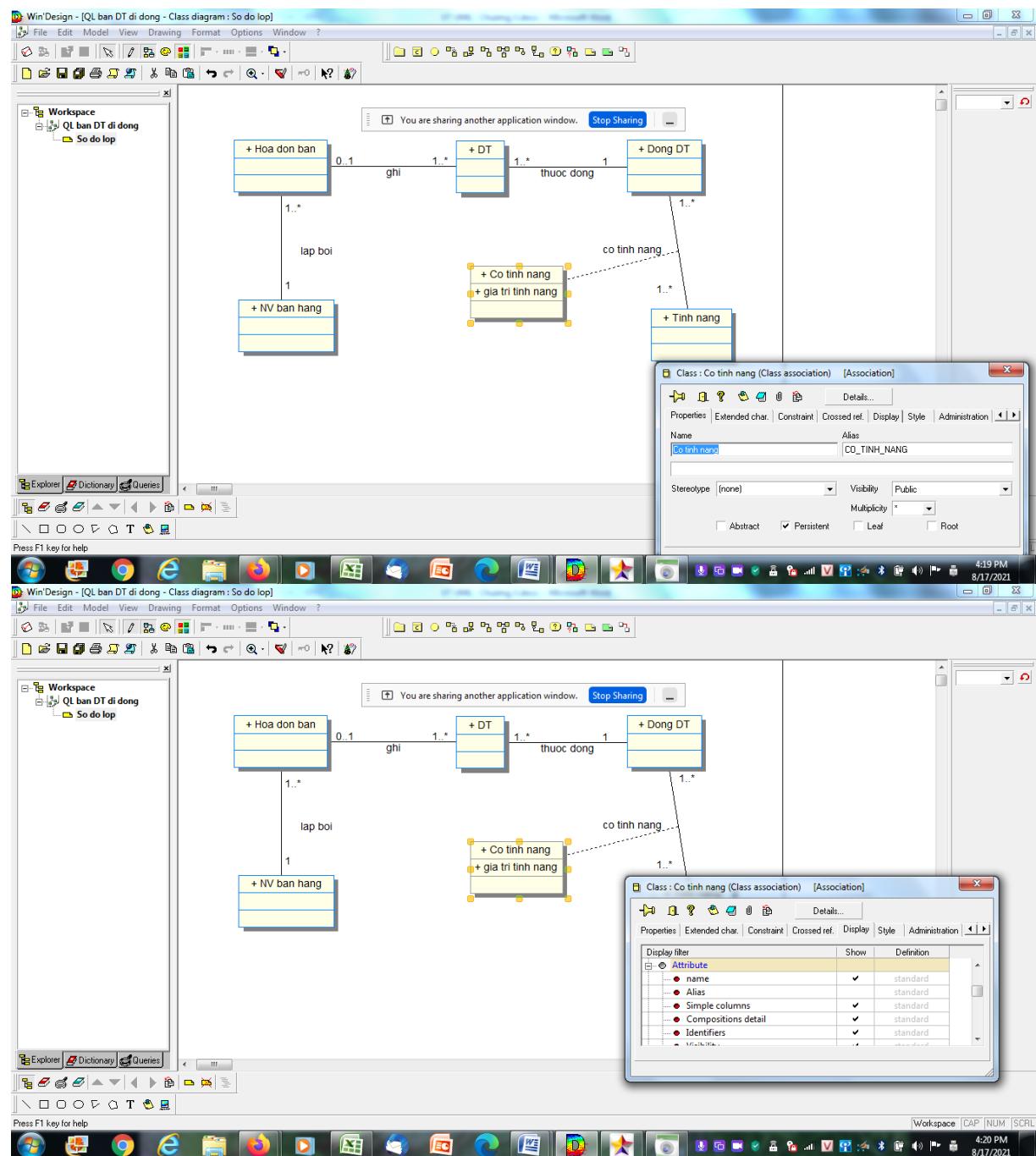


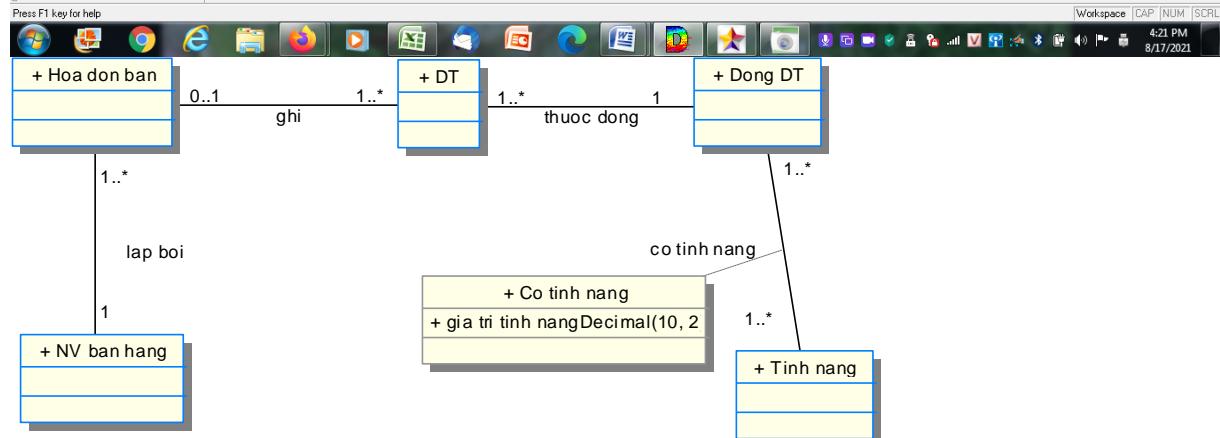
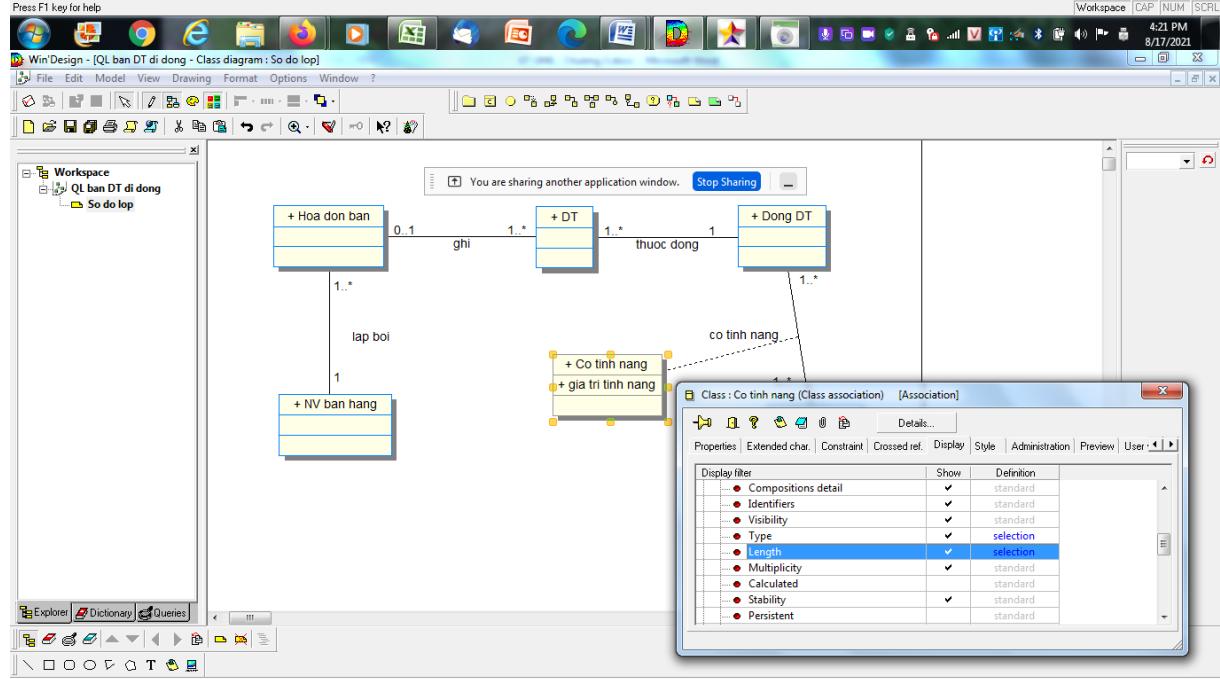
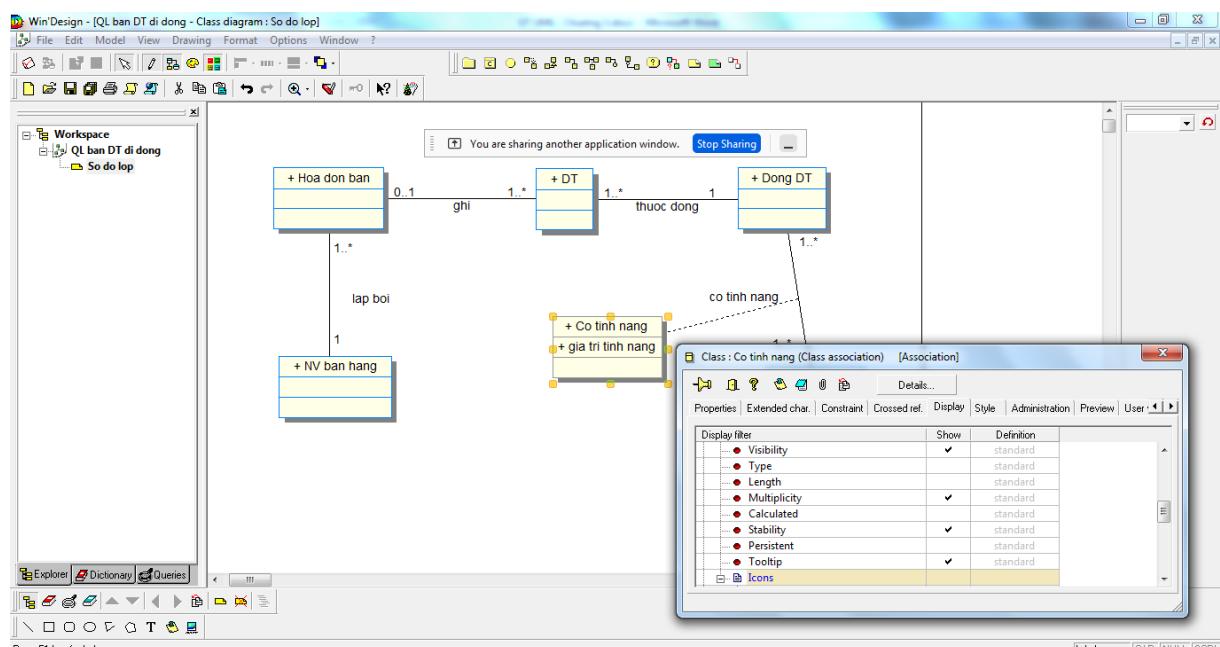


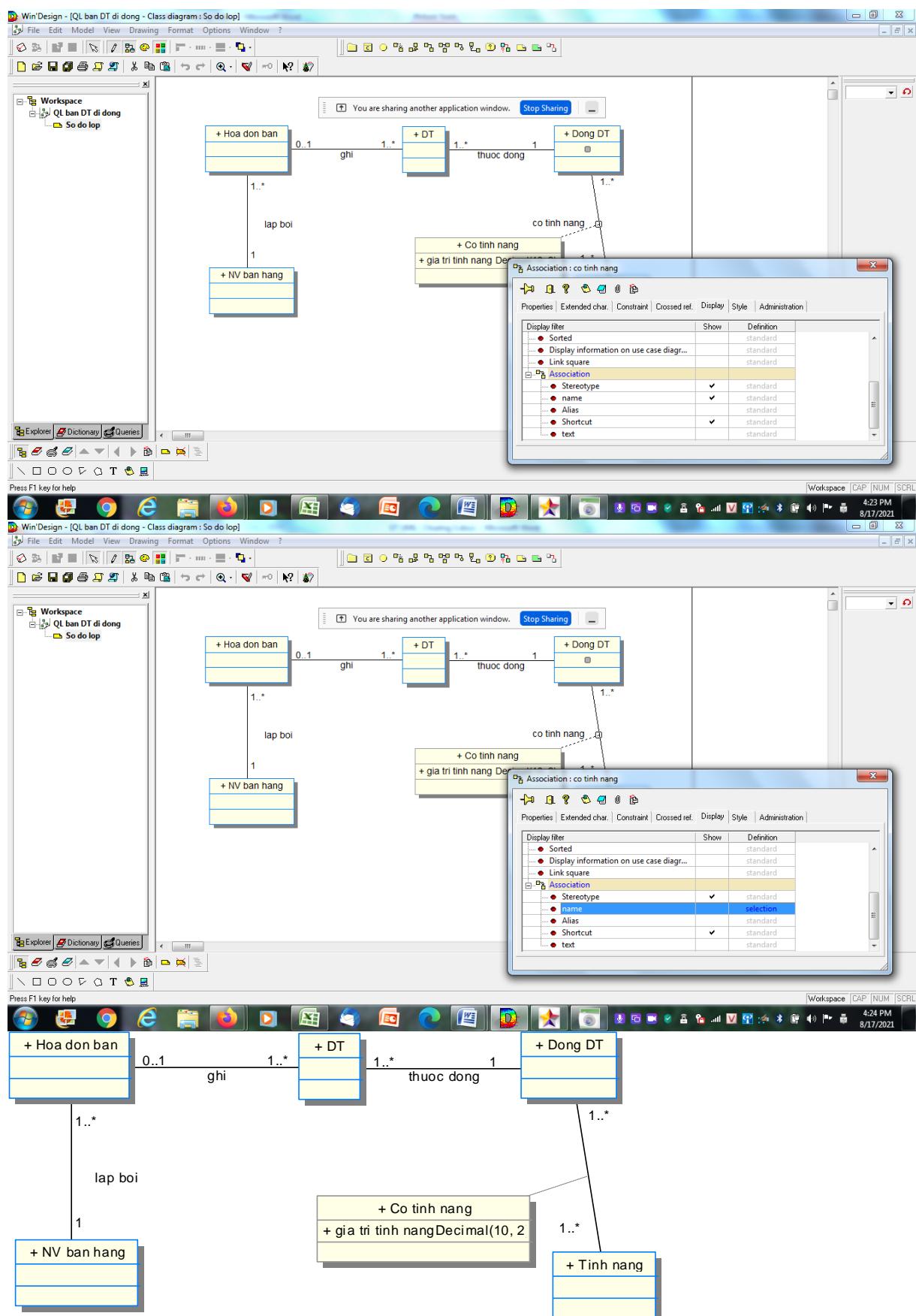








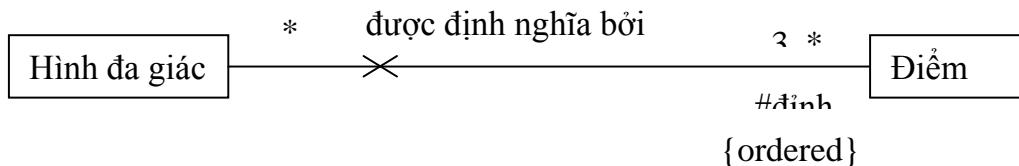




### 3.4.3 Liên kết có ràng buộc (association with constraint)

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. Tuy nhiên, ngôn ngữ OCL (Object Constraint Language) được ưa chuộng hơn.

Ví dụ:

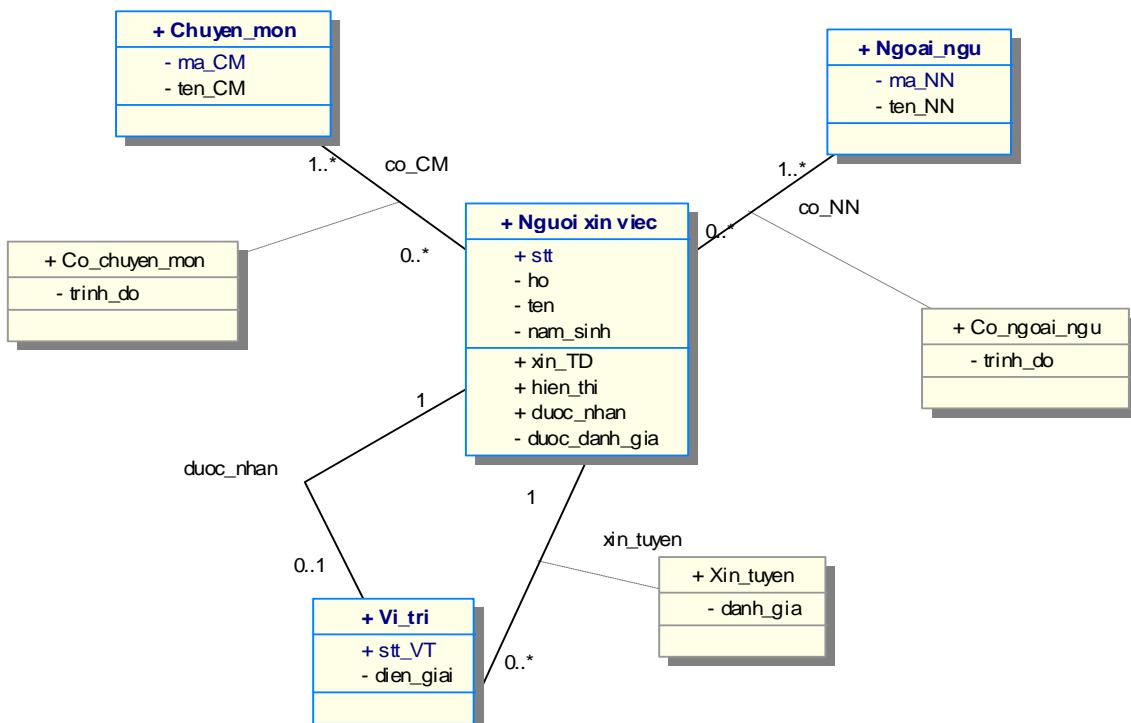


Hình 3.17 Ví dụ về liên kết có ràng buộc

### 3.4.4 Lớp-liên kết (association class)

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 nói 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: trong *Hình 3.18*, các lớp “Co\_chuyen\_mon”, “Co Ngoai\_ngu” và “Xin\_tuyen” là các lớp liên kết.



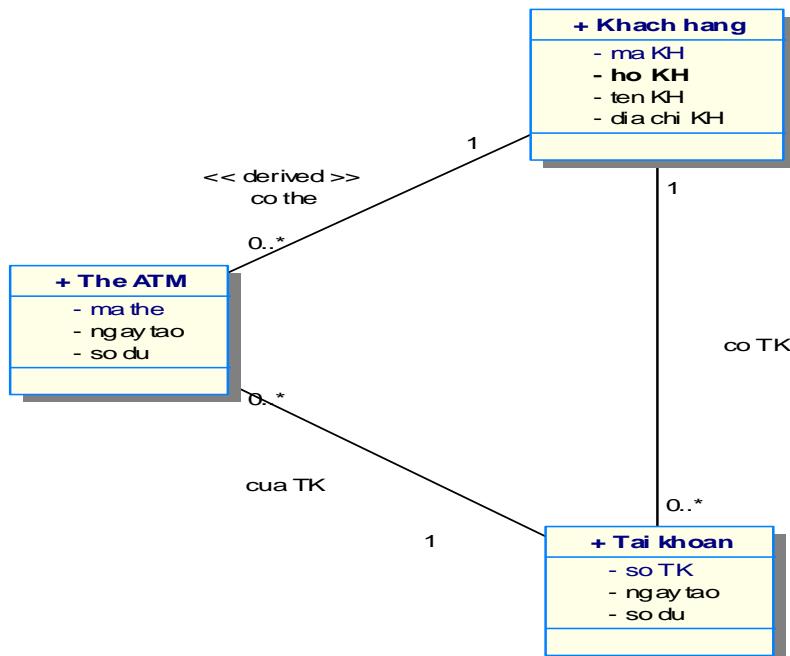
Hình 3.18 Ví dụ về lớp-liên kết

### 3.4.5 Liên kết do suy diễn (derived association)

Liên kết do suy diễn là liên kết được đặt điều kiện hoặc được suy diễn từ ít nhất một liên kết khác. Mặc dù 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.

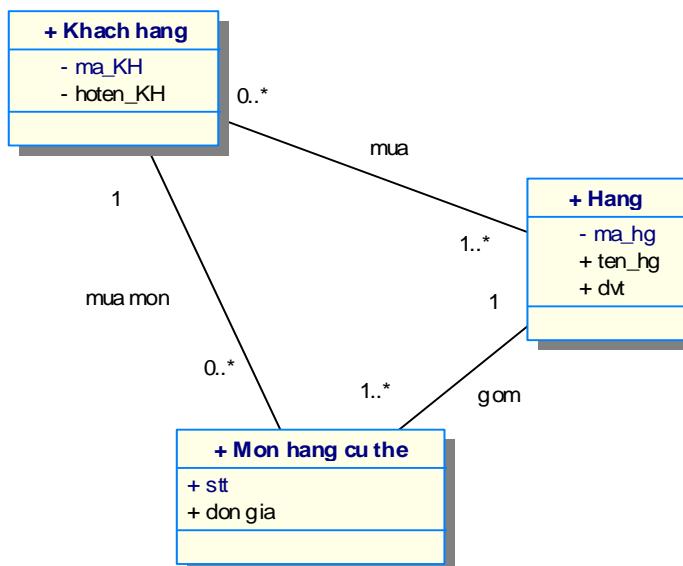
Ký hiệu: có dấu slash (“/”) trước tên liên kết.

Ví dụ 1: Liên kết “có thẻ” trong sơ đồ về thẻ ATM trong Hình 3.19 là một liên kết suy diễn (dấu “/” được thay bởi từ khóa “derived” do phần mềm không cho phép)



Hình 3.19 Ví dụ 1 về liên kết do suy diễn

Ví dụ 2: Liên kết “mua” trong Hình 3.20 là liên kết suy diễn và cần thiết duy trì khi cần biết thường xuyên khách hàng mua hàng gì bằng mỗi liên kết trực tiếp thay vì phải qua 2 liên kết “mua món” và “gồm”.



### Hình 3.20 Ví dụ 2 về liên kết do suy diễn

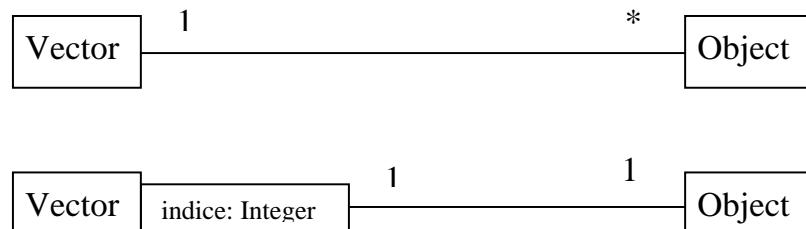
#### 3.4.6 Liên kết có thẩm định (qualified association)

Có 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:

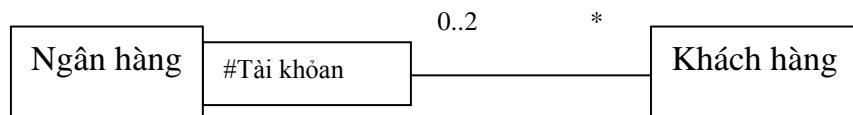
- một chỉ số vào liên kết
- hoặc một lớp vào lớp ban đầu

Ví dụ 1: một đối tượng của lớp Object được tham chiếu bởi một chỉ số duy nhất trong một đối tượng của lớp Vector.



### Hình 3.21 Ví dụ 1 về liên kết có thẩm định- thêm chỉ số vào liên kết

Ví dụ 2: Do ngân hàng có nhiều hoạt động khác không liên quan đến khách hàng trong ngữ cảnh đang xét, và ta muốn giới hạn mối quan hệ giữa khách hàng và ngân hàng chỉ là thông qua các tài khoản: ta thêm vào một lớp “Tài khoản” trong lớp “Ngân hàng”:



### Hình 3.22 Ví dụ 2 về liên kết có thẩm định- thêm lớp vào lớp ban đầu

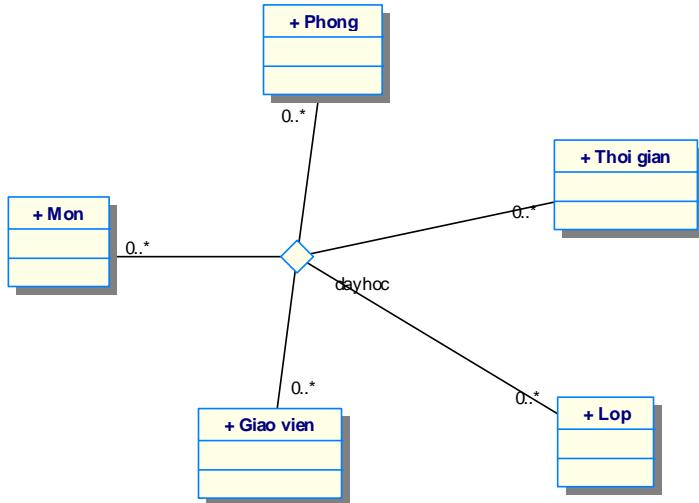
#### 3.4.7 Liên kết nhiều chiều

Trong UML, ít khi dùng liên kết nhiều chiều, mà thường giới hạn ở 2 chiều, đôi khi 3 chiều. Lý do là vì để xác định bản số của mỗi lớp khá khó khăn.

Thường thì nếu có thể được, các liên kết nhiều chiều được chuyển sang dùng lớp-liên kết, hoặc nhiều liên kết 2 chiều. Tuy nhiên, nếu không đảm bảo được ngữ nghĩa thực tế, thì vẫn phải dùng liên kết  $n > 2$  chiều.

Ví dụ: Về thời khóa biểu dạy học, có nhiều cách mô hình hóa:

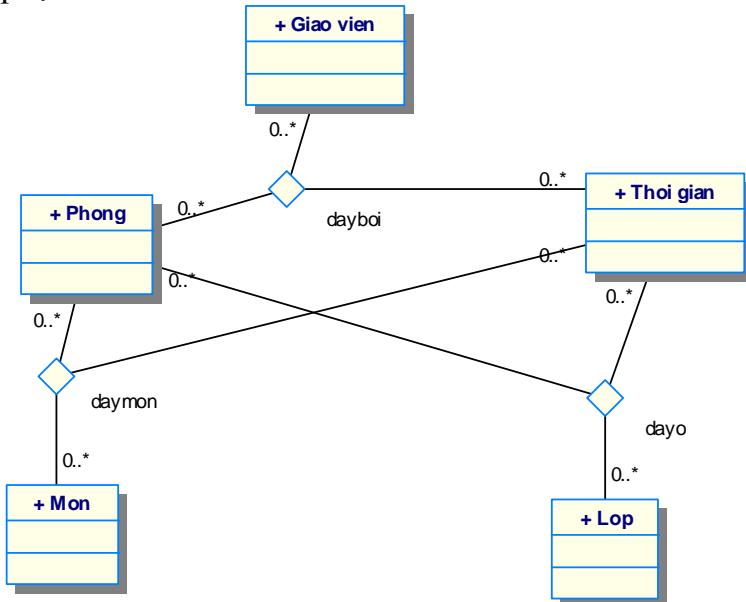
Cách 1: Dùng liên kết nhiều chiều nối với tất cả các lớp liên quan:

**Hình 3.23 Ví dụ 1 về liên kết nhiều chiều**

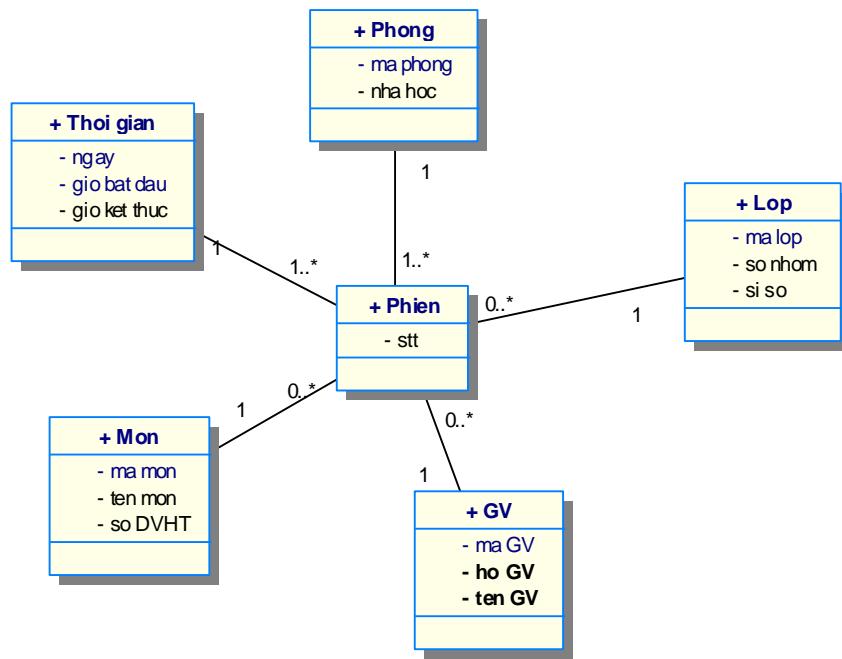
*Cách 2:* Khuynh hướng hiện nay là phá liên kết nhiều chiều ra thành các liên kết ít chiều hơn, nếu không làm mất ngữ nghĩa.

Trong ví dụ dưới đây, do có tồn tại phụ thuộc hàm:

Phòng, Thời gian -> Môn, Giáo viên, Lớp  
nên các liên kết 3 chiều thực sự có khóa chỉ gồm khóa của Phòng và khóa của Thời gian kết hợp lại:

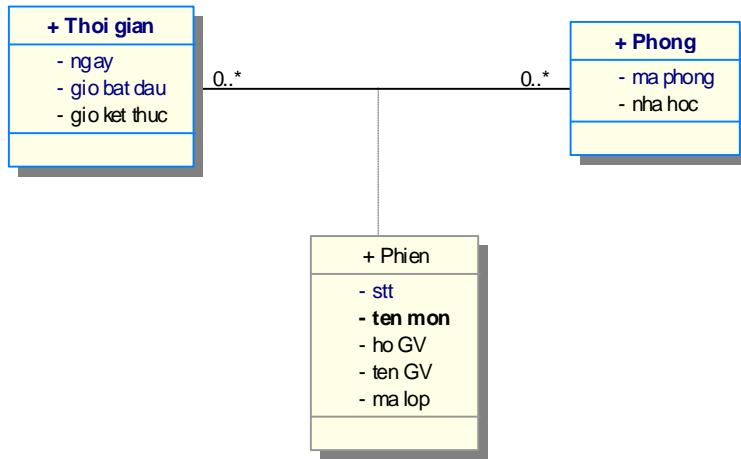
**Hình 3.24 Phá liên kết nhiều chiều ra thành các LK ít chiều hơn**

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



**Hình 3.25** Phá liên kết nhiều chiều bằng cách đưa nút LK thành lớp mới

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



**Hình 3.26** Phá liên kết nhiều chiều bằng cách đưa một số lớp thành thuộc tính

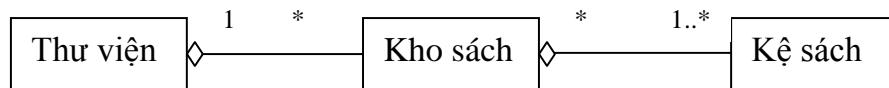
### 3.4.8 Quan hệ kết tập (aggregation [relation])

Một kết tập là một trường hợp đặc biệt của liên kết không đổi 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.

Chu kỳ sống của lớp kết tập là độc lập với các lớp thành phần của nó. Mặt khác, 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.



Hình 3.27 Ví dụ về quan hệ kết tập

### 3.4.9 Quan hệ cấu thành (composition)

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. Một khác, việc sao chép hoặc xóa đi lớp chứa sẽ kéo theo sao chép hoặc xóa các lớp thành phần của nó.

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:



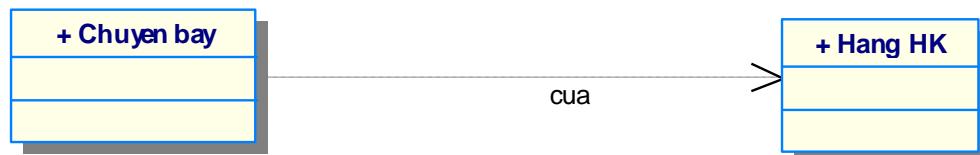
Hình 3.28 Ví dụ về quan hệ cấu thành

### 3.4.10 Quan hệ phụ thuộc (dependancy)

Khi một lớp A phải có lớp B khác mới tồn tại được, ta nói có quan hệ phụ thuộc giữa A và B, hay chính xác hơn, A phụ thuộc vào B.

Ký hiệu: đường gạch đứt nét có mũi tên từ lớp chịu phụ thuộc.

Ví dụ:



Hình 3.29 Ví dụ về quan hệ phụ thuộc

### 3.4.11 Quan hệ thừa kế

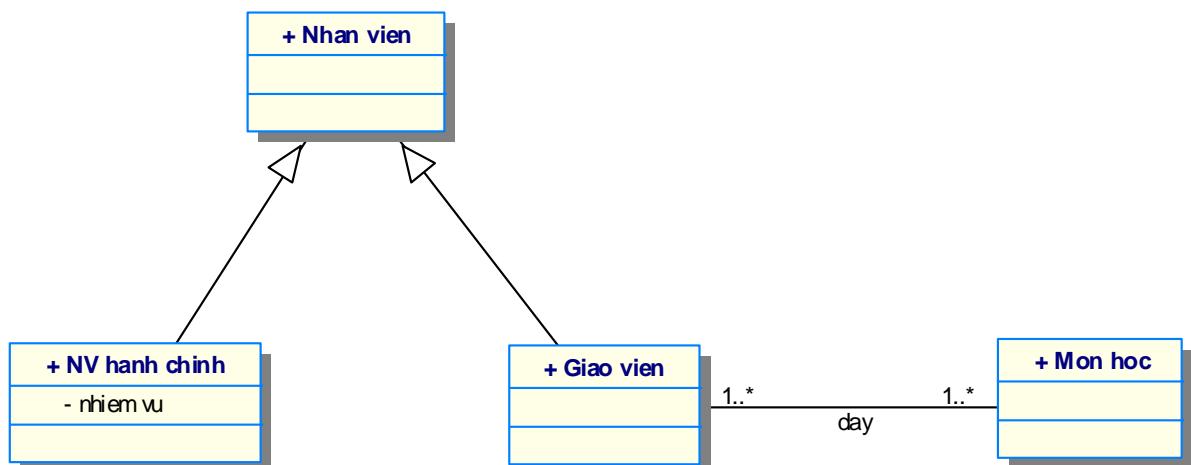
Khái niệm về quan hệ thừa kế tương tự như trong phân tích cỗ điển, cũng có thừa kế đơn và thừa kế bội.

Lớp con sẽ thừa kế hết những thuộc tính và phương thức của lớp cha. Giữa lớp con và lớp cha, hoặc giữa các lớp con với nhau phải có ít nhất một sự khác biệt nào đó :

- Hoặc về thuộc tính
- Hoặc về phương thức
- Hoặc về liên kết với lớp khác

Ký hiệu : đường gạch có mũi tên rỗng hướng về lớp cha.

Hình 3.30 là một ví dụ về quan hệ thừa kế. Trong ví dụ này lớp « Giao viên » và « NV hành chính » cùng thừa kế lớp cha « Nhân viên »



Hình 3.30 Ví dụ về quan hệ thừa kế giữa các lớp

### 3.5 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 (Object Constraint Language : ngôn ngữ mô tả ràng buộc trên đối tượng) 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}, ...

## 3.6 XÂY DỰNG MỘT SƠ ĐỒ LỚP

### 3.6.1 Các quan điểm mô hình hóa

Có hai quan điểm mô hình hóa sơ đồ lớp :

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

### 3.6.2 Các bước xây dựng

#### 3.6.2.1 Tìm các lớp của lĩnh vực chức năng

- Tìm các đối tượng và lớp trong thế giới thực :
  - o Lớp trong thế giới thực
  - o Lớp con trong thế giới thực

Có nhiều nguồn từ thế giới thực để quan sát và phân tích: người thật trong việc thật, tài liệu bằng văn bản, bảng biểu, phim, ảnh, lời thuật chuyện, ...

Chú ý các danh từ trong tên đề tài và trong các nguồn nói trên. Danh từ nào được lặp đi lặp lại, có tầm quan trọng trong thế giới thực thì có nhiều khả năng tương ứng với lớp. Có thể tìm các tập hợp lọc ra được từ thế giới thực, khi đó, mỗi tập hợp sẽ thường là một lớp, và mỗi phần tử của nó sẽ là đối tượng của lớp tương ứng.

Chẳng hạn như trong các biểu bảng, có thể lọc ra được một số lớp từ các mục trong dòng tiêu đề. Mỗi dòng trong thân bảng sẽ tương ứng với một hoặc nhiều đối tượng của các lớp đó.

Nếu có nhiều dạng khác nhau của một tập hợp tạo thành các tập con của nó, ta có thể tìm ra các lớp con của lớp ban đầu để ghi nhận các điểm khác biệt của mỗi tập con.

- Chuyển đổi từ các đối tượng trong thế giới thực sang đối tượng dữ liệu:
  - o *Lớp trong thế giới dữ liệu và ánh xạ cho các lớp*: Thông thường, mỗi lớp trong thế giới thực tương ứng với một lớp trong thế giới dữ liệu.

Các sơ đồ hoạt vụ cũng giúp ích cho việc tìm ra các lớp:

- Lớp tương ứng tên tác nhân kích hoạt trường hợp sử dụng
- Và các lớp xác định bởi các danh từ làm tíc từ của động từ định danh các trường hợp sử dụng.

- o *Quản trị sự phức tạp*: Khi yêu cầu người dùng là đơn giản và phạm vi của ngữ cảnh không quá rộng, ta có thể thiết kế một sơ đồ lớp nhỏ gọn. Nhưng nếu ngược lại, cần phải tăng thêm độ phức tạp cho sơ đồ của mình bằng cách thêm lớp, thêm thuộc tính cho lớp, thêm chiều cho liên kết, cân nhắc bẩn số sao cho thích ứng với nhiều trường hợp có thể xảy ra.

- Chọn lựa giữa lớp và thuộc tính:

Lớp sẽ được các danh từ mô tả đặc tính, tính chất của nó, các danh từ này sẽ là thuộc tính của lớp. Lưu ý, một danh từ chỉ số đo (chiều dài, trọng lượng, thể tích, đơn giá, ...) không bao giờ định danh cho một lớp, mà chỉ có thể là một thuộc tính cho lớp hoặc cho lớp liên kết.

Sơ đồ càng nhiều lớp thì thoát nhìn có vẻ phức tạp, nhưng lại giúp khâu kiểm soát các ràng buộc toàn vẹn trên dữ liệu tốt hơn và sau đó giúp các giao diện thân thiện hơn, người dùng thao tác ít bị lỗi hơn. Ví dụ, thay vì dùng các thuộc tính mang những

giá trị trong một tập cho trước (như thẻ loại sách, đơn vị tính, ...), nên chuyển chúng thành các lớp dưới dạng các bảng mã gồm mã và diễn giải.

- Lớp dữ liệu bổ sung:

Có khi xuất hiện thêm các lớp mới, xuất phát từ các danh từ chỉ đặc tính, sự vật, ... không thể là thuộc tính của một lớp được vì mang nhiều giá trị, hoặc có khi không có trị, ứng với một đối tượng trong thế giới thực.

- Tìm các cấu trúc kết tập và lớp con:

Nếu hai lớp có mối quan hệ với nhau mang tính chất giữa một phần tử và một tập hợp, ta có thể nghĩ đến quan hệ kết tập hoặc câu thành giữa chúng. Nếu hai lớp có chung nhau một số thuộc tính, hoặc phương thức, hoặc các liên kết với lớp khác, có thể dùng quan hệ tổng quát hóa.

### **3.6.2.2 Tìm các mối liên kết giữa các lớp**

Tên các liên kết là các động từ, thường là các động từ gắn liền các danh từ chỉ các sự vật tương ứng các lớp trong các câu mô tả thế giới thực.

Các động từ chỉ trường hợp sử dụng trong các sơ đồ hoạt vụ thường tương ứng tên các liên kết nối liền các lớp.

### **3.6.2.3 Tìm các thuộc tính và định khóa cho mỗi lớp**

Các thuộc tính được xác định bằng các danh từ, đóng vai trò mô tả định tính hoặc định lượng cho một sự kiện, sự vật ngoài thế giới thực đã được mô hình hóa bằng một lớp.

### **3.6.2.4 Tổ chức lại và đơn giản hóa sơ đồ**

Bằng cách sử dụng sự tổng quát hóa, ta có thể giảm bớt được các lớp có cùng một số thuộc tính, hoặc phương thức, hoặc liên kết.

### **3.6.2.5 Chuẩn hóa sơ đồ**

Nếu theo cách thiết kế truyền thống, áp dụng các qui tắc chuẩn 1NF đến 3NF, nếu thỏa mãn BCNF càng tốt.

### **3.6.2.6 Thủ các đường truy xuất đến các lớp**

Ví dụ: Quản lý hàng tồn, xem *Hình 3.31*

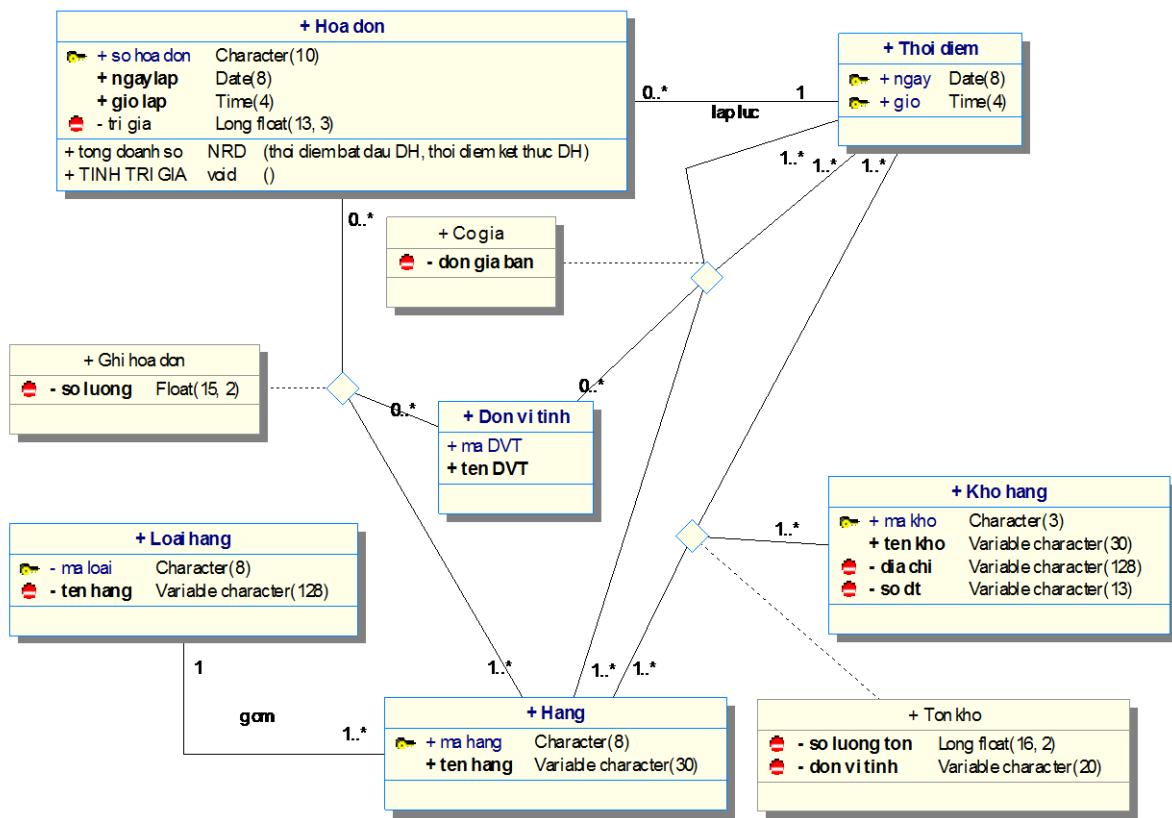
### **3.6.2.7 Tinh lọc hóa sơ đồ**

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. v.v...
- 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.

### **3.6.2.8 Chuyển sơ đồ lớp từ mức quan niệm sang mức luận lý**

Mức luận lý sẽ là bước trung gian để sang mức vật lý. Sơ đồ lớp lúc này tạo nên khung sườn cho cơ sở dữ liệu, với đầy đủ các ràng buộc toàn vẹn trên dữ liệu và ngữ pháp cho các thuộc tính, phương thức, tham số đã phải ở mức chính xác và đầy đủ.



Hình 3.31 Ví dụ về sơ đồ lớp ở mức quan niệm

### 3.6.2.9 Thiết lập các phương thức cho mỗi lớp

Các phương thức sẽ được bổ sung đầy đủ và chính xác (cách thiết lập ở mục 3.8).

### 3.6.2.10 Mô tả các thuộc tính của lớp

Các thuộc tính của mỗi lớp được mô tả trong một bảng có dạng như trong *Bảng 3.1*.

**Bảng 3.1 Mô tả thuộc tính của lớp "Hàng"**

HANG

Sđt	Tên thuộc tính	Kiểu	Kích thước	Số chữ số thập phân	Bản số	Khóa	Duy nhất	Bắt buộc	Trị mặc nhiên	Min	Max	Miền giá trị	RBTV luận lý	RBTV Khóa ngoài	Lớp được tham chiếu	Diễn giải
1	MA_HANG	C	8			x										mã hàng
2	TEN_HANG	VC	30				x	x								tên hàng
5	REF_LOAI	REF						x							LOAI_HG	loại hàng

### 3.6.2.11 Mô tả các phương thức của lớp

Các phương thức của mỗi lớp được mô tả trong một bảng có dạng như trong *Bảng 3.2*. Các cột “bản số” được dùng khi kiểu trả về hoặc kiểu tham số tương ứng với một mảng. Nếu phương thức là phương thức lớp, ở cột “Là PT lớp” sẽ được đánh dấu X. Do tính đa hình nên các phương thức có thể trùng tên với nhau. Tuy nhiên, khi sử dụng một công cụ vẽ thiết kế, công cụ đó có thể không cho phép dùng trùng tên thì tên phương thức có thể thay đổi một chút.

**Bảng 3.2 Mô tả phương thức của lớp "Hàng"**



### 3.6.2.12 So khớp lại với các sơ đồ khác

Sơ đồ lớp sẽ được 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.

### 3.6.2.13 Đóng gói

Cách làm đã được trình bày ở chương 2- Sơ đồ hoạt vụ.

## 3.7 CHUYỂN SƠ ĐỒ LỚP TỪ MỨC QUAN NIỆM SANG MỨC LUẬN LÝ

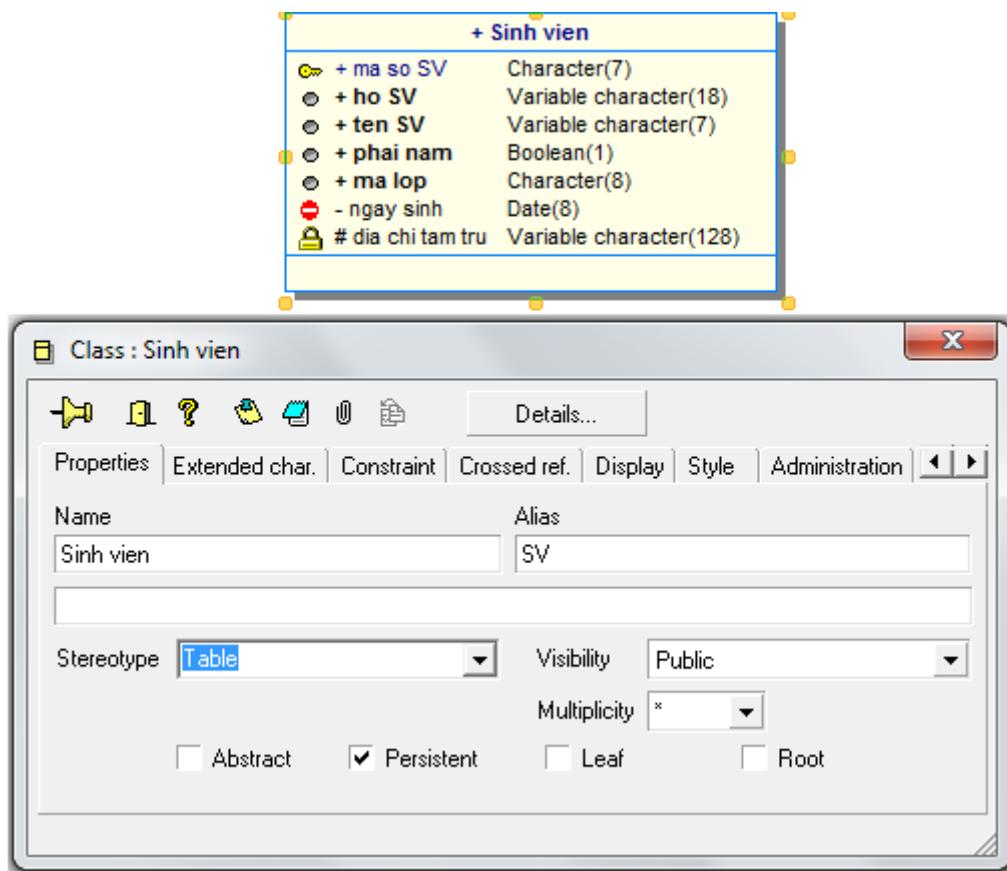
Có 2 cách chuyển từ sơ đồ lớp từ mức quan niệm sang mức luận lý: chuyển theo cách truyền thống như ở CSDL quan hệ và chuyển theo hướng đối tượng. Cách thứ nhất đơn giản, quen thuộc và tuân thủ các dạng chuẩn bắt buộc 1NF – 3NF (NF: normal form- dạng chuẩn), nhưng không tận dụng lợi thế và đặc điểm của CSDL hướng đối tượng.

CSDL hướng đối tượng theo khuynh hướng không hoàn toàn tôn trọng các qui tắc chuẩn, miễn sao đạt tối ưu, do đó có thể chấp nhận dữ thừa dữ liệu, có dữ liệu null, không có khóa chính ... Như vậy, theo cách chuyển thứ hai, một lớp có thể chứa thuộc tính là cả một đối tượng hoặc một mảng đối tượng của lớp khác, hoặc đơn giản là chứa tham chiếu (ref: reference) hoặc định danh đối tượng (OID: object identifier) của lớp khác.

Tham chiếu đến một lớp có thể được xem tương tự như con trỏ, sẽ trỏ đến một đối tượng của lớp đó.

### 3.7.1 Đối với lớp

Tên lớp ở mức luận lý thường được giữ như ở mức quan niệm, nhưng phải theo qui cách của các hệ quản trị cơ sở dữ liệu. Thông thường, các bí danh (alias) của lớp đã đặc tả ở mức quan niệm (không chứa khoảng trắng) được dùng làm tên lớp ở mức luận lý.



Hình 3.32 Ví dụ về dùng tên lớp ở mức quan niệm cho mức luận lý

### 3.7.2 Đối với thuộc tính của các lớp chính

Lớp chính là khái niệm dùng để chỉ các lớp đã có từ mức quan niệm. Tương tự như đối với lớp, tên thuộc tính ở mức luận lý thường được giữ như ở mức quan niệm, nhưng phải theo qui cách của các hệ quản trị cơ sở dữ liệu. Thông thường, các bí danh (alias) của thuộc tính đã đặc tả ở mức quan niệm (không chứa khoảng trắng) được dùng làm tên thuộc tính ở mức luận lý.

Ngoài ra, trong lớp chính có thể sẽ có thêm các thuộc tính từ các lớp liên kết, hoặc các thuộc tính có kiểu là tên lớp do mối liên kết giữa lớp đang xét với lớp khác (xem các mục tiếp theo).

Nếu ở mức quan niệm, lớp đã có khóa thì ở mức luận lý, khóa sẽ được giữ nguyên.

SV	
PK_SV	
MSSV	
MSSV	CHAR(7)
HO_SV	VARCHAR2(18)
TEN_SV	VARCHAR2(7)
NAM	NUMBER(1)
MA_LOP	CHAR(8)
NGAY SINH	DATE(8)
DC_TAM	VARCHAR2(128)

Hình 3.33 Ví dụ về dùng thuộc tính của lớp chính ở mức luận lý

### 3.7.3 Đổi với thuộc tính của các lớp liên kết

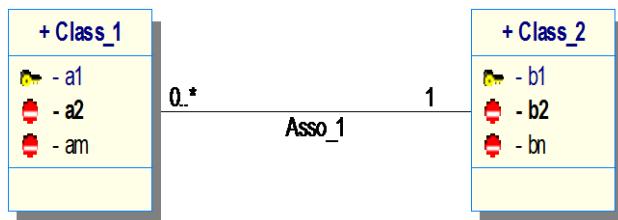
Tùy vào các cách biến đổi sẽ áp dụng cho các liên kết, các thuộc tính của các lớp liên kết sẽ đưa về các lớp chính, hay nằm trong lớp mới.

### 3.7.4 Đổi với liên kết phụ thuộc hàm mạnh

#### 3.7.4.1 Cách chuyển truyền thống như ở CSDL quan hệ

Phụ thuộc hàm (PTH) sẽ biến mất. Trong lớp nguồn, sẽ sao chép khóa chính của lớp đích về làm khóa ngoài, và như vậy các thuộc tính này phải có cùng kiểu và kích thước với nhau.

Trong hình minh họa dưới đây, Class\_1 được gọi là lớp nguồn, và Class\_2 là lớp đích.



Trở thành:

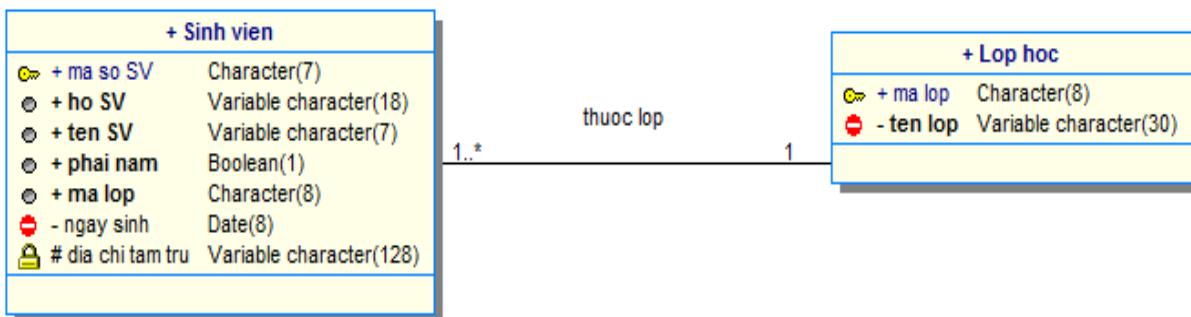


Khi đó, B1 là khóa ngoài trong CLASS\_1, có ràng buộc toàn vẹn “not null” hoặc “required” trong CLASS\_1.

Thực tế trong một số công cụ (như Win Design chẳng hạn), có vẽ một mũi tên chỉ để minh họa mối ràng buộc khóa ngoài giữa 2 lớp.

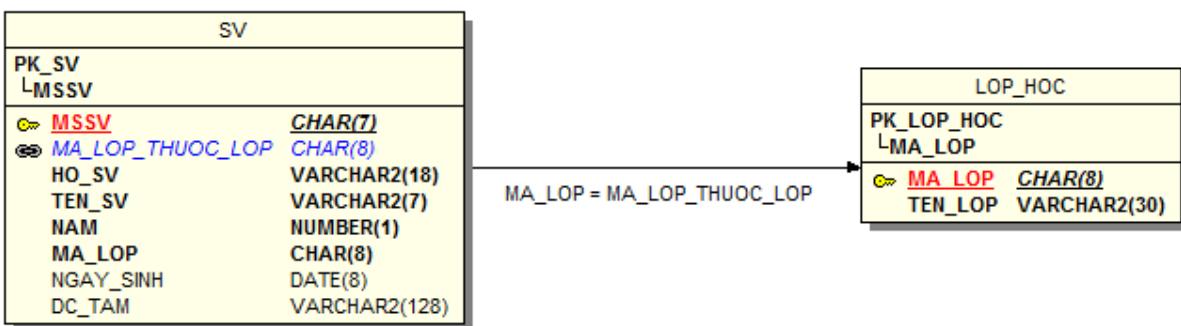


Ví dụ:



Hình 3.34 Ví dụ về phụ thuộc hàm mạnh ở mức quan niệm

Trở thành:



Hình 3.35 Ví dụ về chuyển sang mức luận lý theo cách truyền thống

Trong *Hình 3.35*, thuộc tính màu xanh *MA\_LOP\_THUOC\_LOP* là khóa ngoài (FK: foreign key) trong lớp *SV*, lấy từ khóa chính *MA\_LOP* của lớp *LOP\_HOC*.

Nếu có lớp liên kết (nhưng bình thường đối với phụ thuộc hàm, ít khi có lớp liên kết), các thuộc tính của nó sẽ được chuyển về lớp nguồn.

### 3.7.4.2 Cách chuyển theo hướng đối tượng

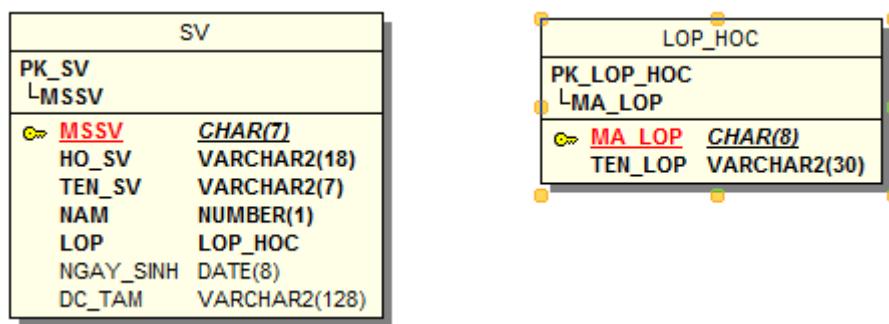
#### \* Cách 1 : thêm thuộc tính kiểu lớp

Cách 1 được dùng khi có các điều kiện sau:

- Ta muốn ở một lớp đồng thời tất cả các giá trị thuộc tính của lớp đối diện,
- Và lớp đối diện có kích thước khai báo không quá lớn.

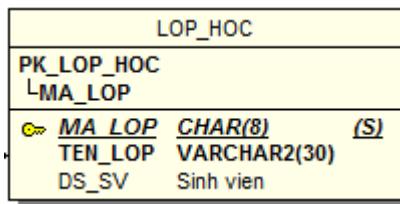
Khi đó, lớp có thêm thuộc tính mới sẽ trở thành lớp phức, các đối tượng của nó là các đối tượng phức.

Trong lớp **nguồn**, thuộc tính được thêm sẽ có kiểu là lớp đích. Trong ví dụ ở hình bên dưới, thuộc tính *LOP* ở lớp *SV* có kiểu là *LOP\_HOC* để chỉ lớp học quản lý sinh viên đang xét.



Hình 3.36 Ví dụ về chuyển sang mức luận lý theo cách 1 dùng kiểu lớp

**Lớp đích** cũng có thể thêm thuộc tính là một mảng các đối tượng của lớp nguồn. Ví dụ, ở *Hình 3.36*, nếu có nhu cầu thường xuyên muốn truy cập danh sách sinh viên với đầy đủ thông tin, lớp LOP\_HOC trong *Hình 3.37* có thể thêm một mảng các đối tượng của lớp SV.



Hình 3.37 Lớp có thuộc tính

Trong *Hình 3.37*, công cụ Win Design 6.5 chưa hiển thị bản số của thuộc tính để cho biết DS\_SV là một mảng các đối tượng)

#### \* Cách 2: thêm thuộc tính kiểu ref

Cách 2 và cách 3 được dùng khi có các điều kiện sau:

- Ta muốn khi cần, một lớp có thể truy cập được đồng thời tất cả các thuộc tính của lớp đối diện,
- Và lớp đối diện có kích thước khai báo hơi còng kềnh,

Cách 2 được chọn nếu người dùng quen với khái niệm tham chiếu được dùng phổ biến trong các hệ quản trị CSDL (như Oracle). Thuộc tính được thêm sẽ có kiểu là tham chiếu (ref: reference) đến lớp đối diện.

Trong ví dụ ở hình bên dưới, thuộc tính LOP ở lớp SV có kiểu là ref(LOP\_HOC) để chỉ tham chiếu trả đến lớp học quản lý sinh viên đang xét.

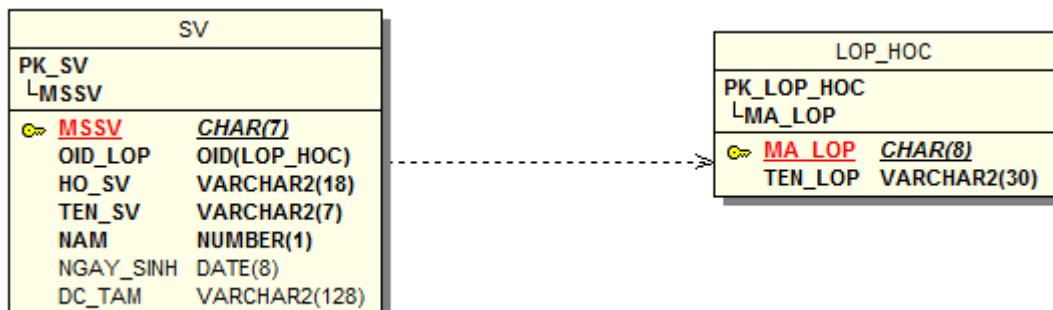


Hình 3.38 Ví dụ về chuyển sang mức luận lý theo cách 2 dùng kiểu ref

### \* Cách 3: thêm thuộc tính kiểu OID

Thuộc tính được thêm sẽ có kiểu là định danh đối tượng (OID: object identifier) của lớp đích.

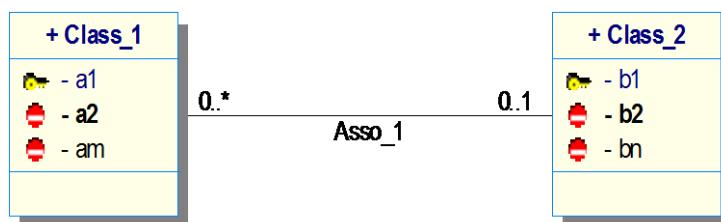
Trong ví dụ ở hình bên dưới, thuộc tính LOP ở lớp SV có kiểu là OID(LOP\_HOC) để chỉ tham chiếu trả đến lớp học quản lý sinh viên đang xét.



Hình 3.39 Ví dụ về chuyển PTH sang mức luận lý theo cách 2 dùng kiểu OID

### 3.7.5 Đối với liên kết phụ thuộc hàm yếu

#### 3.7.5.1 Cách chuyển truyền thông như ở CSDL quan hệ

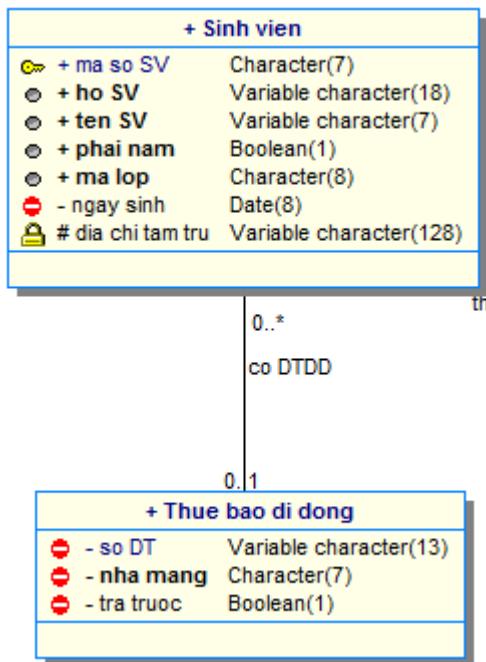


Hình 3.40 Sơ đồ tổng quát về phụ thuộc hàm yếu

Có 2 giải pháp:

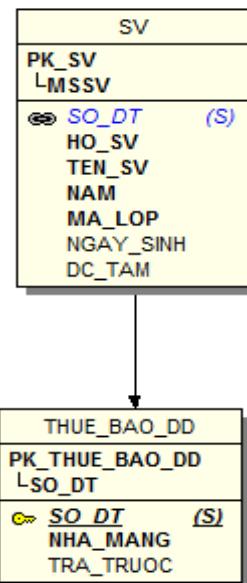
- *Chấp nhận vi phạm INF, nếu số đối tượng không tham gia liên kết chiếm tỷ lệ không nhiều (PTH gần mạnh): chuyển như ở trường hợp PTH mạnh, có một số đối tượng của CLASS\_1 sẽ có khóa ngoài B1 mang trị null (B1 sẽ không có ràng buộc toàn vẹn “not null” hoặc “required” trong CALSS\_1).*

Ví dụ: Hiện nay, đa số SV có điện thoại di động nên ta có mô hình bên dưới:



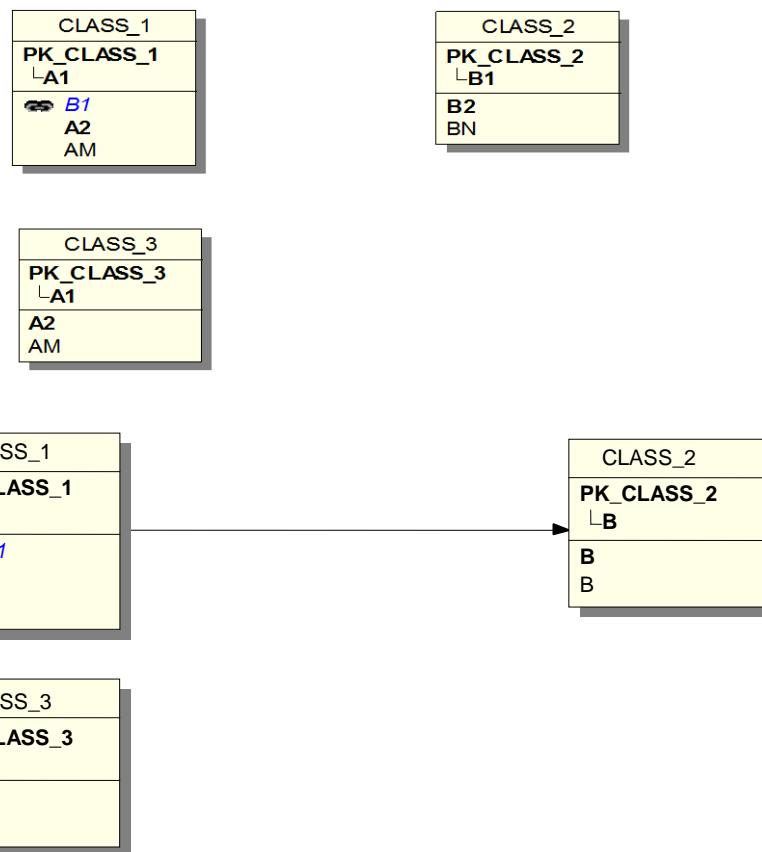
**Hình 3.41** Ví dụ về phụ thuộc hàm yếu có đa số đối tượng tham gia PTH yếu

Sơ đồ *Hình 3.41* được biến đổi ở mức luận lý như sau:



**Hình 3.42** Mức luận lý của sơ đồ ở *Hình 3.41*

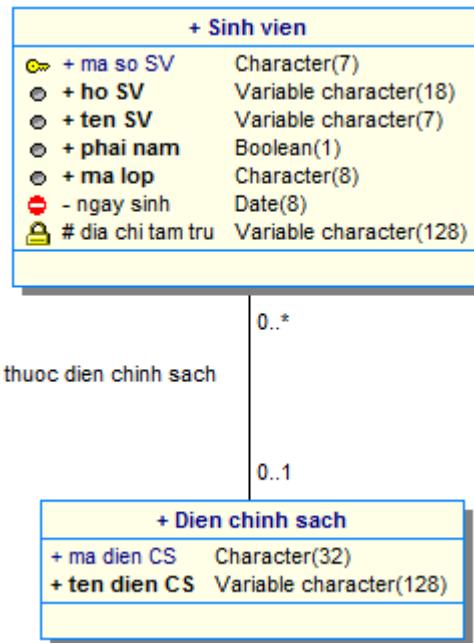
- *Chuyển lớp Class\_1 thành 2 lớp*: lớp CLASS\_1 sẽ có khóa ngoài B1 mang trị khác null và lớp CLASS\_3 chỉ có các thuộc tính như ban đầu, không chứa khóa ngoài nào.

**Hình 3.43 Sơ đồ biến đổi phụ thuộc hàm yếu ở mức luận lý**

Nếu có lớp liên kết cho PTH này, các thuộc tính của nó sẽ được chuyển về cho lớp nguồn.

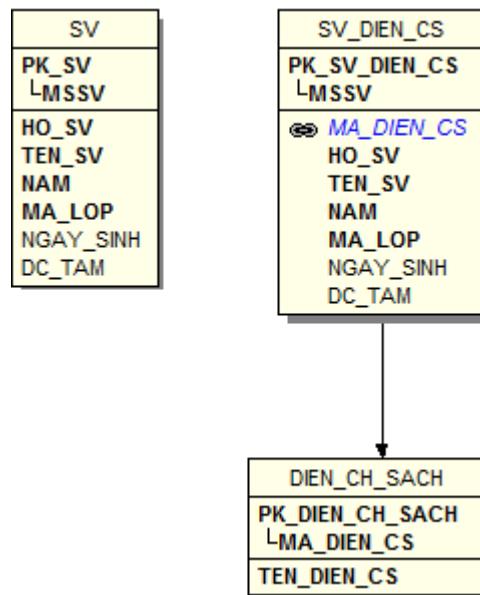
Ví dụ:

Giả sử mỗi sinh viên chỉ có thể thuộc một diện chính sách ưu tiên (nếu đồng thời thuộc nhiều diện thì sẽ được chọn diện có ưu tiên cao nhất), ta có phụ thuộc hàm yếu “thuộc diện chính sách” như dưới đây:



**Hình 3.44** Ví dụ về phụ thuộc hàm yếu ở mức quan niệm

Do chỉ có một số khá ít sinh viên thuộc dạng chính sách nên ở mức luận lý, ta sẽ có 2 lớp sinh viên tương ứng với 2 loại SV thuộc/ không thuộc diện chính sách như sau:



**Hình 3.45** Sơ đồ lớp ở mức luận lý của Hình 3.44

### 3.7.5.2 Cách chuyển theo hướng đối tượng

Tương tự như trên, nhưng thay vì thêm vào một lớp khóa ngoài từ khóa chính của lớp đối diện, sẽ chọn thêm thuộc tính có một trong 3 kiểu: toàn bộ lớp đối diện, tham chiếu đến lớp đối diện, hoặc OID của lớp đối diện.

### 3.7.6 Đối với liên kết các dạng khác

#### 3.7.6.1 Cách chuyển theo truyền thống

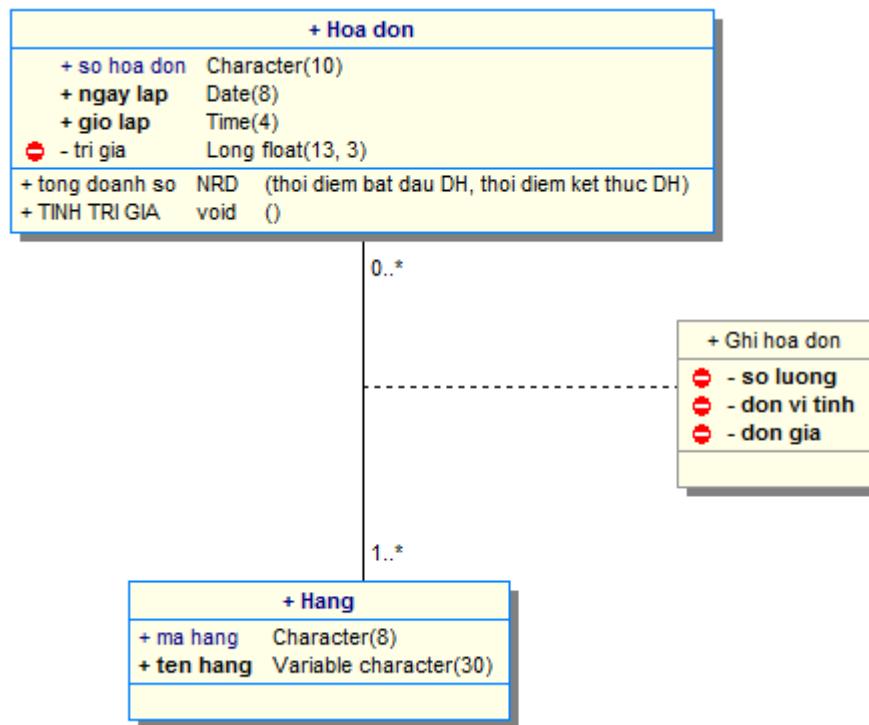
Liên kết trở thành một lớp mới, có khóa được tạo thành từ tổ hợp các khóa của các lớp tham gia liên kết ban đầu. Như vậy, mỗi thuộc tính tham gia khóa của lớp mới là một khóa ngoài tham chiếu đến một lớp ban đầu.

- *Liên kết 2 chiều nhiều - nhiều (bản số tối đa đều là \*)*: có thể có lớp liên kết



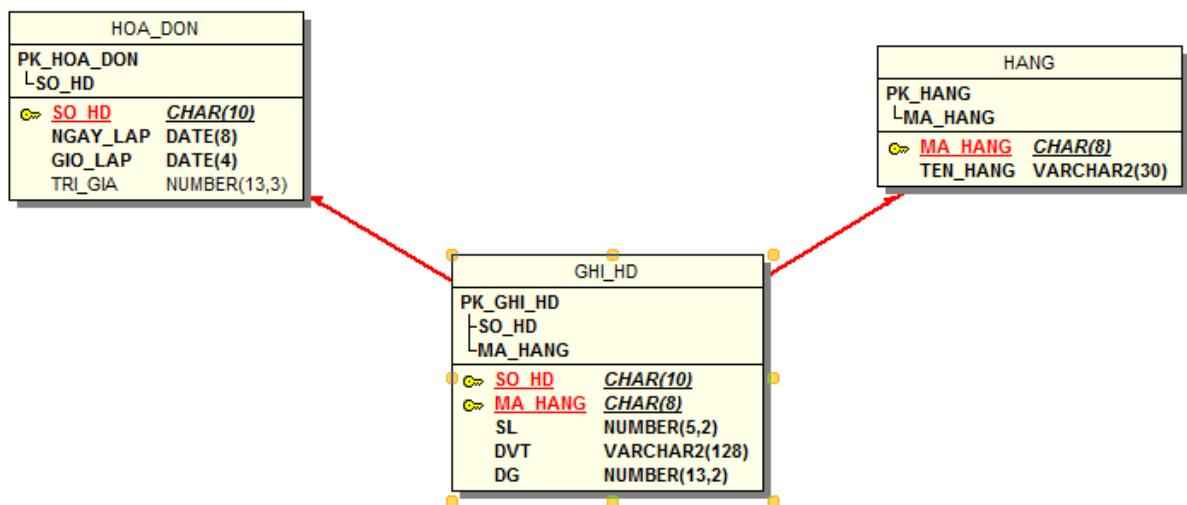
Hình 3.46 Sơ đồ tổng quát về liên kết 2 chiều nhiều - nhiều

Ví dụ:

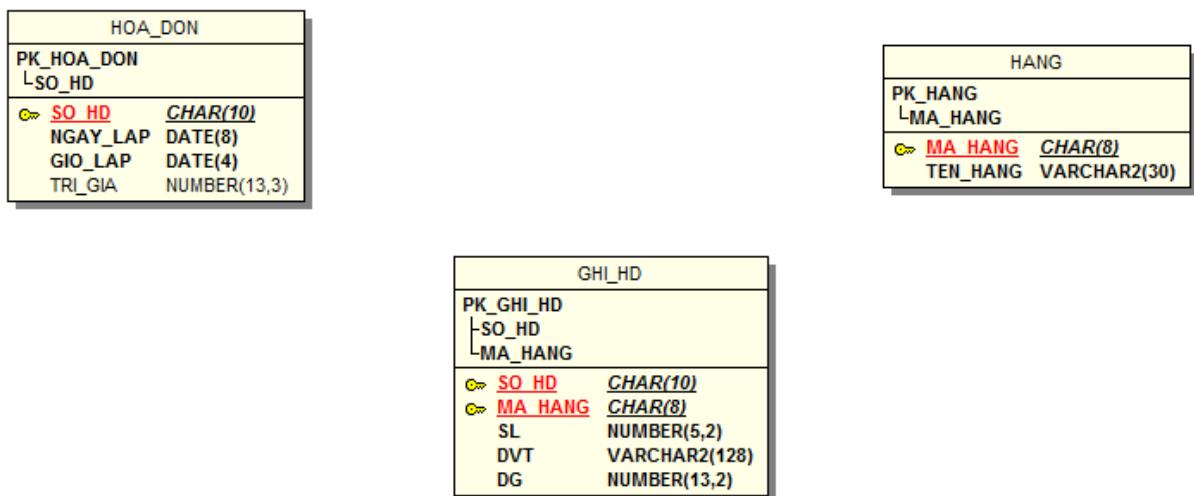


Hình 3.47 Ví dụ về liên kết 2 chiều nhiều - nhiều có lớp liên kết

Trở thành:



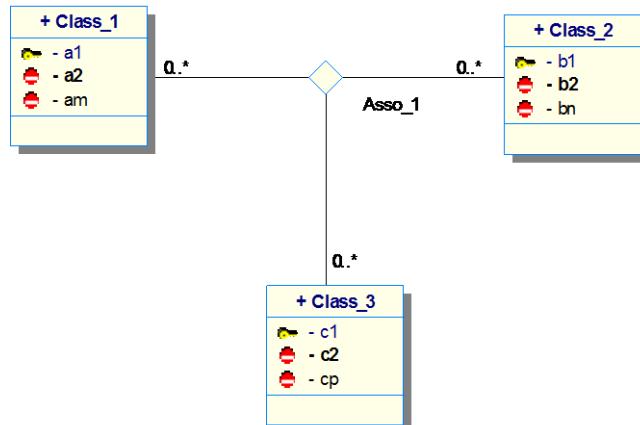
Hoặc:



**Hình 3.48** Mức luận lý của sơ đồ ở hình **Hình 3.47**

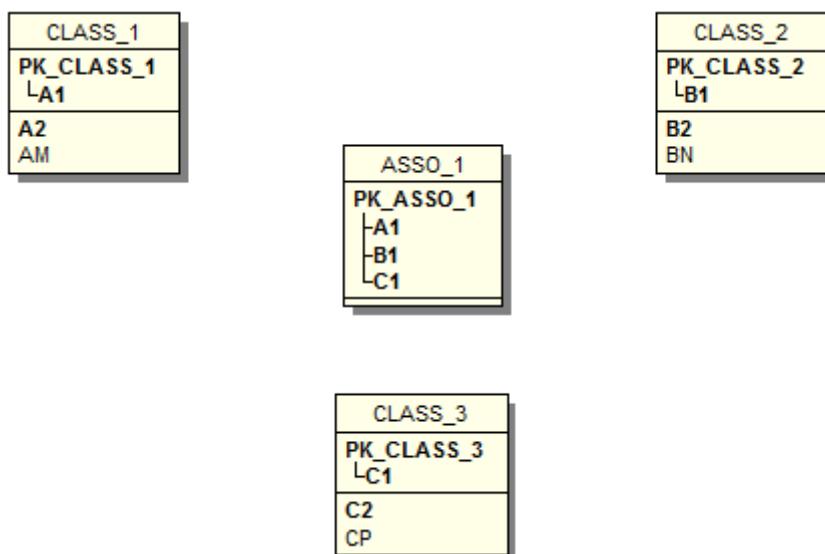
- *Liên kết nhiều hơn 2 chiều:*

*Hình 3.49* mô tả sơ đồ tổng quát về liên kết nhiều chiều ở mức quan niệm. *Hình 3.50* mô tả sơ đồ này ở mức luận lý.

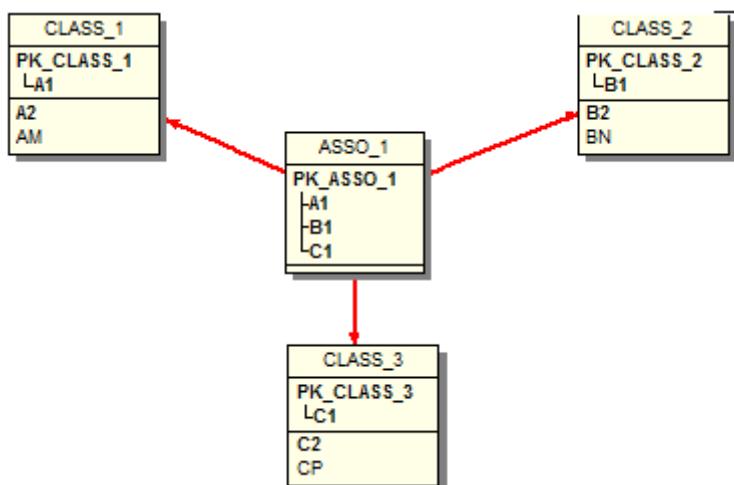


Hình 3.49 Sơ đồ tổng quát về liên kết n chiều mức quan niệm

Trở thành:

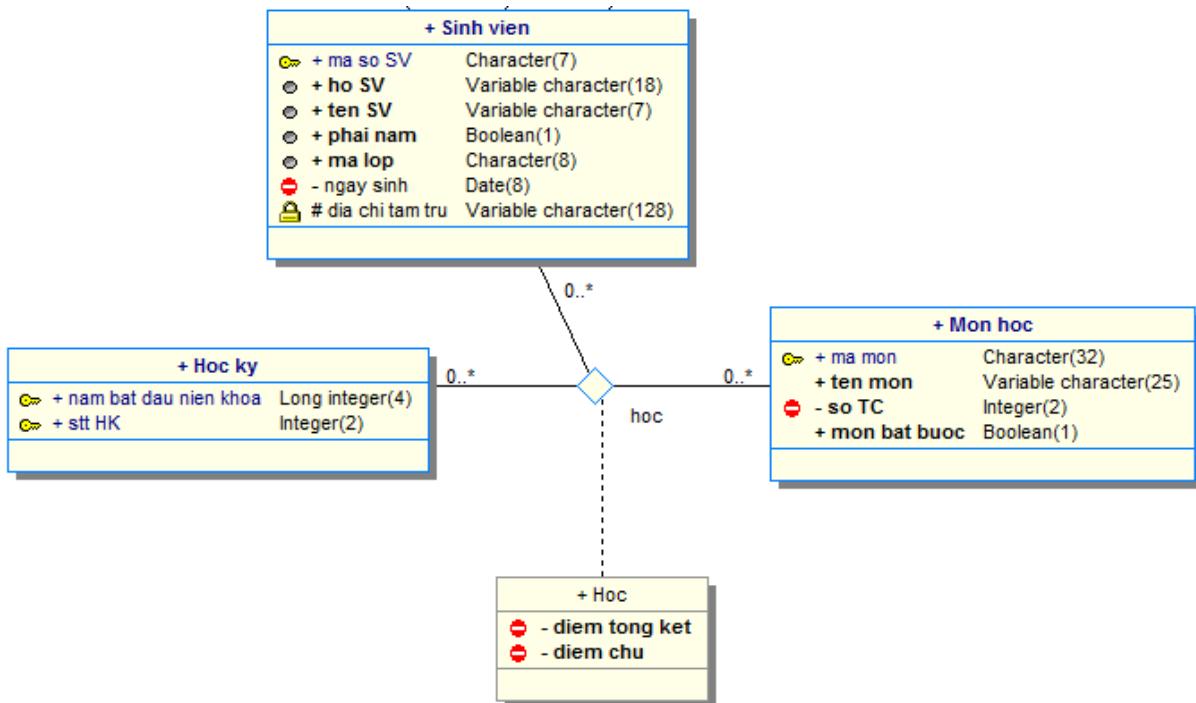


Hoặc:



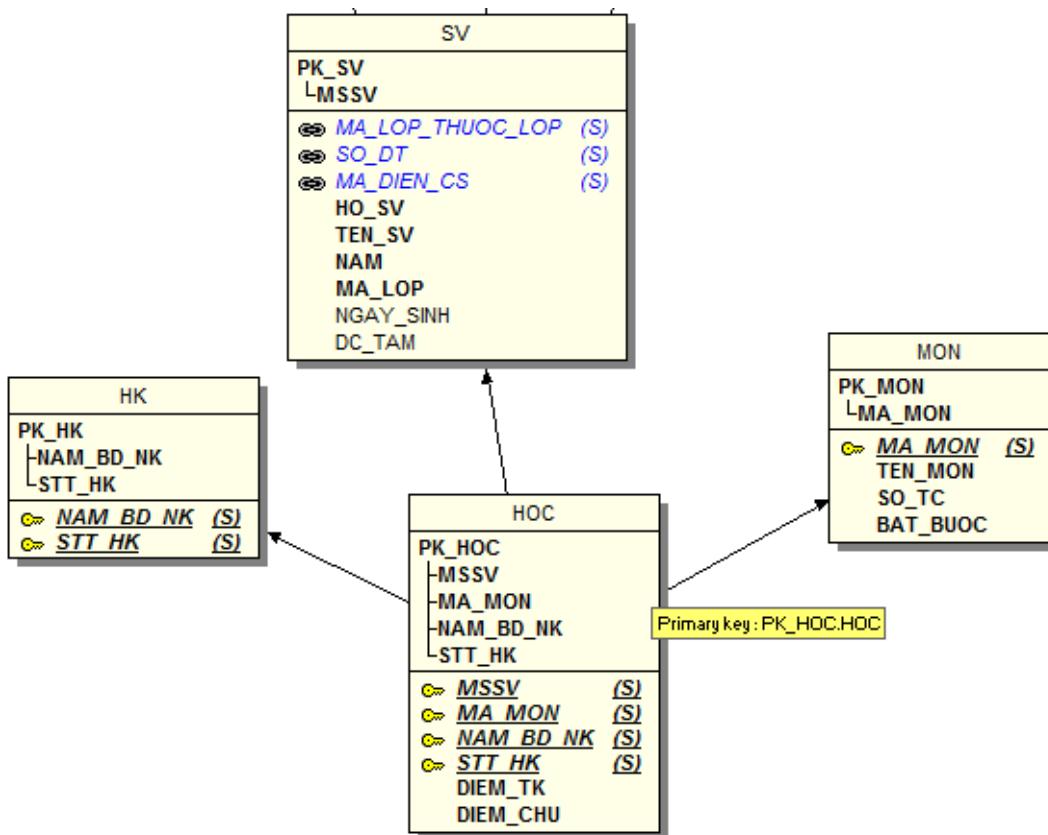
Hình 3.50 Sơ đồ tổng quát về liên kết n chiều mức luận lý

Ví dụ:



Hình 3.51 Ví dụ về liên kết n chiều mức quan niệm

Trở thành :



Hình 3.52 Mức luận lý của sơ đồ ở Hình 3.51

Mỗi lớp ở sơ đồ lớp mức luận lý sẽ được mô tả bằng một bảng với đầy đủ các ràng buộc toàn vẹn khóa chính, duy nhất, trị mặc nhiên, bắt buộc, khóa ngoài, tham chiếu và ý nghĩa của mỗi thuộc tính.

### 3.7.6.2 Cách chuyển theo hướng đối tượng

Theo cách này, mỗi thuộc tính được thêm vào một lớp A sẽ theo một trong những 4 kiểu : khóa ngoài, lớp, tham chiếu hoặc OID tương ứng lớp « đối diện » B, tùy vào nhu cầu sử dụng hết tất cả các thuộc tính hay không, kích thước mỗi đối tượng của lớp « đối diện » có lớn hay không :

- Nếu cần hầu hết các thuộc tính của lớp B và đối tượng của B chiếm kích thước chấp nhận được, A sẽ thêm thuộc tính có kiểu B;

- Ngược lại, có thể chọn một trong 3 cách còn lại :

- + Nếu khóa chính của B thường xuyên được sử dụng trực tiếp, chọn cách thêm khóa này như khóa ngoài trong A.

- + Nếu không cần một khóa có tính duy nhất trong toàn CSDL và ngại chuỗi quá dài của OID, chọn cách thêm tham chiếu đến B (ref(B)).

Từ quy tắc chung này, ta áp dụng cho các lớp liên quan đến liên kết nhiều chiều :

- Liên kết có thể trở thành lớp mới, vẫn chứa các thuộc tính liên kết nếu có, nhưng gắn với từng lớp tham gia liên kết trước đây bằng cách thêm một thuộc tính. Tuy nhiên, không phải tất cả các thuộc tính thêm vào đều là khóa ngoài. Vì thế, lớp mới này có thể không có khóa chính, các đối tượng của nó có thể được truy xuất qua các tham chiếu hoặc OID.

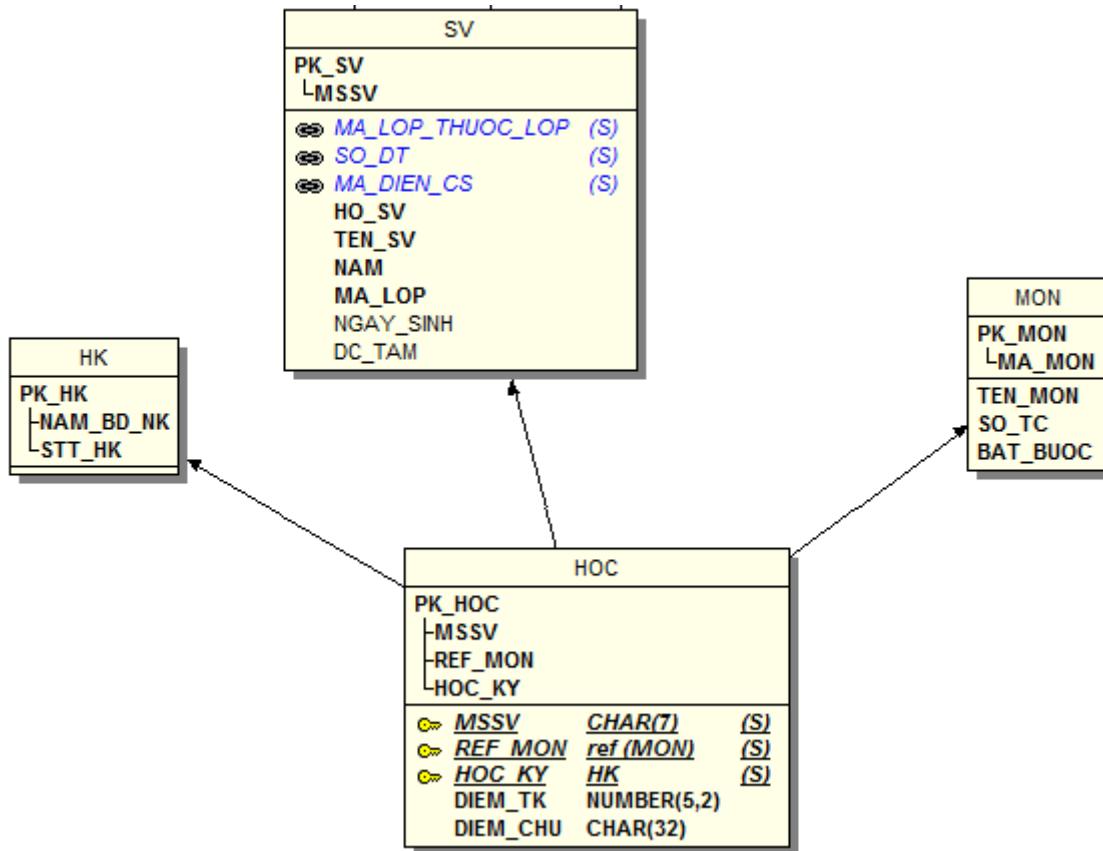
- Nếu không có thuộc tính liên kết, cũng có thể không phát sinh lớp mới như trên.

- Mỗi lớp tham gia liên kết có thể giữ nguyên như cũ nếu có lớp mới phát sinh từ liên kết.

- Nhưng nếu không có lớp mới, bắt buộc có ít nhất một lớp cũ có thêm thuộc tính liên quan một đối tượng hoặc mang các đối tượng của các lớp còn lại.

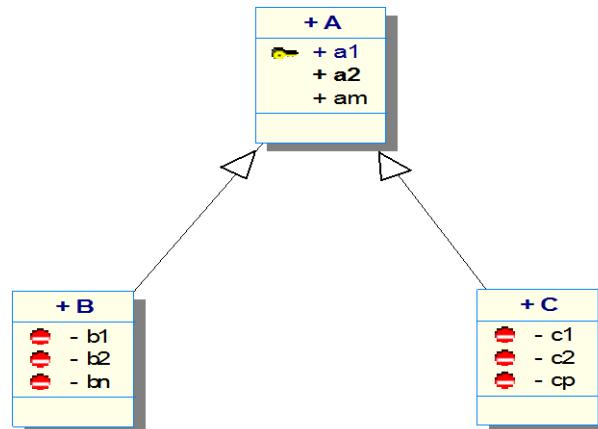
#### Ví dụ:

Liên kết « học » thành lớp mới « HỌC », nhưng có khác biệt so với cách truyền thống: thuộc tính REF\_MÔN có kiểu tham chiếu đến lớp MÔN, vì cần hết các thuộc tính của MÔN, nhưng không muốn lưu cả đối tượng của nó. Trong khi đó, thuộc tính HỌC\_KỲ cần dùng hết các thuộc tính của lớp HK cùng lúc nên sẽ được gán kiểu lớp HK.



Hình 3.53 Mức luận lý theo hướng đối tượng của sơ đồ ở Hình 3.51

### 3.7.7 Đối với liên kết dạng tổng quát hóa

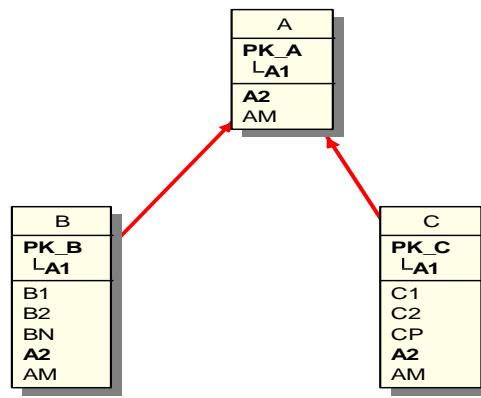


Hình 3.54 Sơ đồ tổng quát hóa về tổng quát hóa ở mức quan niệm

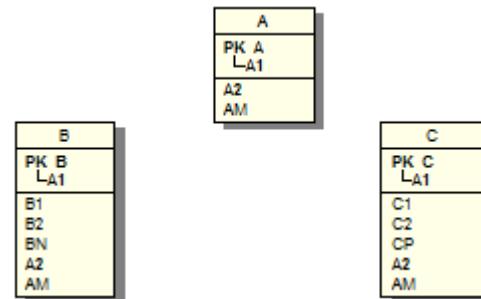
Có 3 trường hợp:

- Trường hợp 1: Số thuộc tính m của lớp cha không nhiều lăm và lớp cha thường được truy cập độc lập. Trong trường hợp này các lớp con sẽ bao gồm các thuộc tính của lớp cha.

Áp dụng trường hợp này cho sơ đồ *Hình 3.54* ta có sơ đồ mức luận lý ở *Hình 3.55*.



Hoặc:



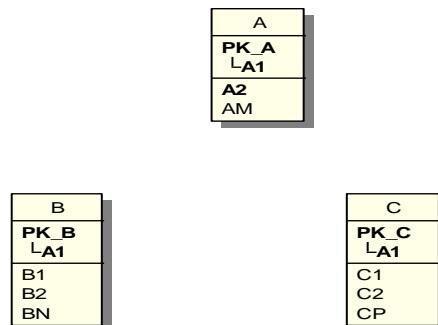
**Hình 3.55** Mức luận lý theo trường hợp 1 của sơ đồ ở **Hình 3.54**

- Trường hợp 2: Số thuộc tính m của lớp cha không nhiều lăm và lớp cha không còn được truy cập độc lập: lớp cha được bỏ đi luôn.  
Với trường hợp này sơ đồ *Hình 3.54* sẽ có sơ đồ luận lý như *Hình 3.56*



**Hình 3.56** Mức luận lý theo trường hợp 2 của sơ đồ ở **Hình 3.54**

- Trường hợp 3: Số thuộc tính m của lớp cha khá lớn và lớp con không thường xuyên dùng các thuộc tính của lớp cha. Với trường hợp này sơ đồ luận lý của lớp con không bao gồm các thuộc tính của lớp cha.



**Hình 3.57** Mức luận lý theo trường hợp 3 của sơ đồ ở **Hình 3.54**

### 3.8 THIẾT LẬP CÁC PHƯƠNG THỨC CHO MỘT LỚP

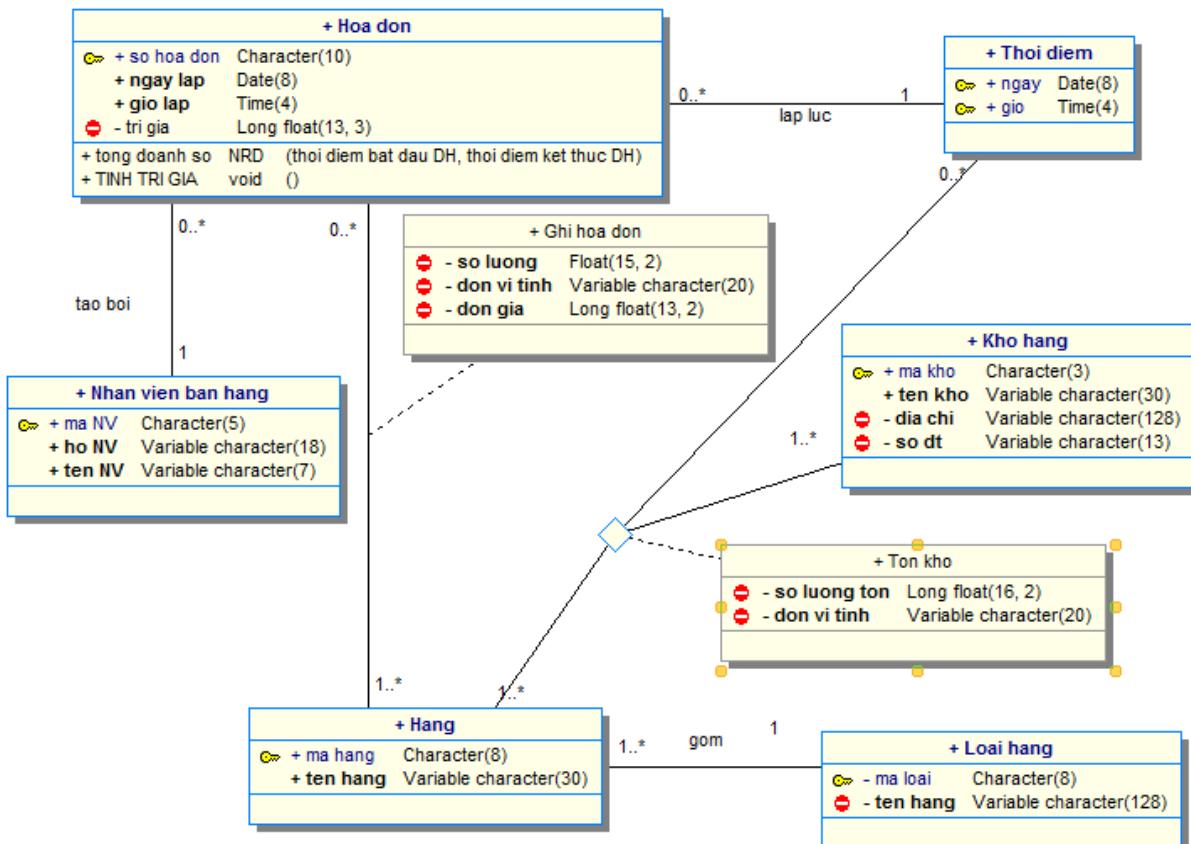
Phương thức được thiết lập có thể bao gồm các *phương thức lớp* (class method/static method, được ký hiệu là « \* ») hoặc *phương thức thành viên* (member method).

Thực tế, sử dụng sơ đồ lớp ở mức quan niệm sẽ dễ tạo phương thức hơn, căn cứ vào các liên kết, bản số tối thiểu và tối đa, chứ không chỉ ở các thuộc tính trong sơ đồ lớp mức luận lý.

- *Liên kết*: giúp tạo các loại phương thức từ loại i đến loại n.
- *Bản số tối đa*: để biết một hàm sẽ trả về chỉ tối đa một đối tượng/trị, hay cả một mảng nhiều đối tượng/trị.
- *Bản số tối thiểu*: để biết có khả năng tìm được kết quả trả về cho hàm hay không.

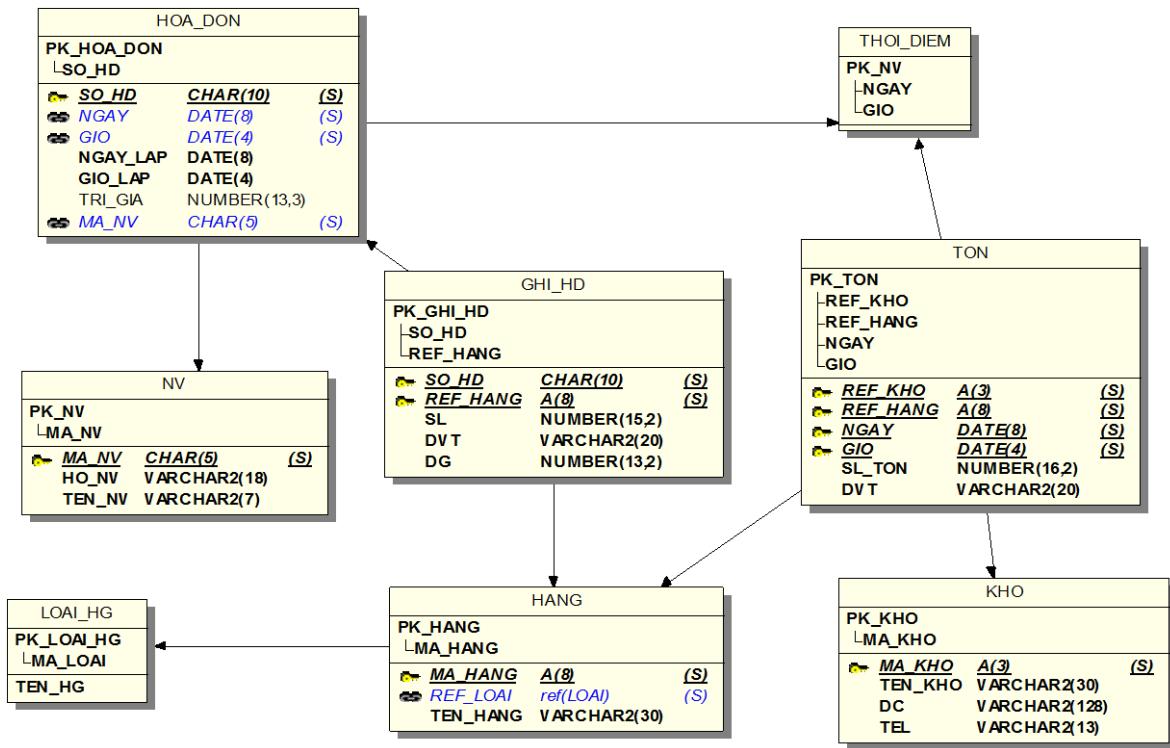
Ví dụ : Có sơ đồ lớp ở mức quan niệm như *Hình 3.58* và ta muốn thêm phương thức cho lớp « Hàng ». Để đơn giản cho ví dụ ở phần này, ta đã trích sơ đồ này ra từ sơ đồ lớp lớn hơn và tạm lưu đơn giá bán vào lớp liên kết « Ghi hóa đơn », sau khi truy cập đơn giá bán từ lớp liên kết giữa lớp Hàng và lớp Thời điểm (xem *hình 3.58*).

Ví dụ này được chọn vì từ lớp Hàng có các liên kết phụ thuộc hàm, liên kết 2 chiều nhiều - nhiều, liên kết n>2 chiều, có lớp liên kết.



**Hình 3.58** Sơ đồ lớp ở mức quan niệm để làm ví dụ về lập phương thức cho lớp « Hàng»

Sơ đồ trên được biến đổi ở mức luận lý như sau :



**Hình 3.59** Sơ đồ lớp ở mức luận lý của sơ đồ ở hình 3.58

Các phương thức đặc trưng cho nghiệp vụ, chức năng của ngữ cảnh từng đề tài sẽ phải thêm cụ thể. Ở đây, chỉ nêu lên các phương thức gợi ý nên có; chúng cần được so lại với thực tế xem có phù hợp không (hoặc vì người dùng không cần tới, hoặc thêm bớt tham số, ...).

Các phương thức được chia thành 3 nhóm: nhóm bắt buộc phải có (loại a-e), nhóm căn bản (loại f- p) và nhóm gồm các phương thức suy diễn từ các phương thức khác (các loại còn lại). Ngoài ra, còn có các phương thức phục vụ việc in ấn.

### 3.8.1 Loại a - Hàm tạo (constructor)

Hàm tạo để thêm một đối tượng cho lớp, hàm này trả về đối tượng mới vừa tạo. Hàm tạo có thể được đặt tên theo cách tùy ý, miễn là giữ nguyên ngữ nghĩa, hoặc tuân theo ngữ pháp qui ước của ngôn ngữ lập trình hướng đối tượng :

< tên lớp > ([ <ds các tham số> ]) : < tên lớp > ;

Danh sách các tham số gồm các phần tử cách nhau bởi dấu phẩy và theo dạng sau:

< tên tham số > : < kiểu tham số >

Những tham số còn thiếu ở các hàm tạo sẽ được gán trị mặc định theo khai báo của thuộc tính tương ứng. Nếu thuộc tính không có trị mặc định, tham số tương ứng thuộc tính đó sẽ không được thiếu trong hàm tạo.

Ví dụ :

~~Hang () : Hang;~~ // do mã và tên hàng không có trị mặc định

~~Hang (m : Character(8)) : Hang;~~ // do tên hàng không có trị mặc định

~~Hang (t : Variable character(30)) : Hang;~~

// do mã hàng không có trị mặc định  
 Hang (m : Character(8), t : Variable character(30)): Hang;

Trong đó m là mã hàng và t là tên hàng, được nhập vào để tạo một đối tượng mới cho lớp « Hàng ».

### 3.8.2 Loại b - Hàm hủy (destructor)

Để hủy một đối tượng của lớp, thực chất đây là một phương thức thuộc loại thủ tục. Tương tự hàm tạo, hàm hủy cũng có thể đặt tên theo ý riêng, hoặc theo qui ước như sau :

$\sim <\text{tên lớp}>()$  ;

Ví dụ :

$\sim\text{Hang}()$  ;

### 3.8.3 Loại c - Lấy ra trị của một thuộc tính (get method)

Đây là một hàm không tham số, trả về trị của một thuộc tính, do đó bắt buộc phải có kiểu trả về trùng với kiểu của thuộc tính đó.

Ngữ pháp :

$<\text{tên hàm}>() : <\text{kiểu của thuộc tính}>;$

Ví dụ :

$\text{get_ma()} : \text{Character}(8);$   
 $\text{get_ten}() : \text{Variable character}(30);$   
 $\text{get_ref_Loai(): ref(Loai)};$

### 3.8.4 Loại d - Lấy ra trị một thuộc tính của lớp liên kết

Ta cũng có thể lấy ra trị một thuộc tính của lớp liên kết có gắn với lớp đang xét. Loại hàm này vẫn phải có kiểu trả về trùng với kiểu của thuộc tính. Ngoài ra, ta cần cung cấp đủ các tham số để có thể cùng với đối tượng đang xét, xác định chính xác chỉ một trị cho thuộc tính cần tìm.

Ví dụ, lớp Hàng gắn với các lớp liên kết Ghi hóa đơn và Tồn hàng. Ta có thể có các hàm sau để lấy ra các thuộc tính của các lớp này:

$\text{get_sl_ban (hd : Hoa_don) : Long float (15, 2);}$   
 $\text{get_dg_ban (hd : Hoa_don) : Long float (15, 2);}$   
 $\text{get_sl_ton (kh : Kho, td : Thoi_diem) : Long float (16, 2);}$

hoặc :

$\text{get_sl_ban (so_hd : C(10)) : Long float (15, 2);}$   
 $\text{get_dg_ban (so_hd : C(10)) : Long float (15, 2);}$   
 $\text{get_sl_ton (ma_kh : C(8), ngay : Date, gio : Time) : Long float (16, 2);}$

### 3.8.5 Loại e - Thay đổi trị của một thuộc tính (set method)

Ngược với get method, set method thuộc loại thủ tục dùng để thay đổi trị của một thuộc tính nên bắt buộc phải có tham số có cùng kiểu với thuộc tính đó, hoặc có kiểu liên quan kiểu thuộc tính đó.

Ngữ pháp :

$<\text{tên thủ tục}>(<\text{tên tham số}> : <\text{kiểu của thuộc tính}>);$

Ví dụ :

```
set_ma (m : Character(8)) ;
set_ten (t : Variable character(30));
set_loai(loai: Loai_hg ) ;
```

**3.8.6 Loại f - Hiển thị đối tượng đang xét**

Đối tượng đang xét được hiển thị với đầy đủ các giá trị thuộc tính.

Ví dụ :

```
hien_Hang () ;
```

**3.8.7 Loại g - Hiển thị tất cả các đối tượng của lớp đang xét (\*)**

Mặc nhiên, các đối tượng được sắp theo thứ tự tăng của khóa chính khi hiển thị.

Ví dụ :

```
hien_DS_Hang() ;
```

**3.8.8 Loại h - Hiển thị tất cả các đối tượng của lớp đang xét, có sắp theo nhóm (\*)**

Hiển thị có sắp xếp các đối tượng của lớp đang xét, với tiêu chí để sắp là:

- + một thuộc tính của lớp đang xét,
- + hoặc một lớp nào đó có liên kết với lớp đang xét.

Ví dụ :

```
Hien_DS_Hang_theo_ten();
Hien_DS_Hang_theo_loai();
Hien_DS_Hang_theo_hoa_don();
Hien_DS_Hang_theo_kho();
Hien_DS_Hang_theo_NV_ban();
```

**3.8.9 Loại i - Tìm kiếm đối tượng (\*)**

Loại hàm này tìm kiếm đối tượng theo khóa đối tượng hoặc OID của chính lớp đang xét. Khóa ở đây có thể không phải khóa chính.

Ví dụ :

```
tim_Hang (m : Character (8)) : Hang ; // tìm theo mã hàng
tim_Hang (t: Variable character(30) ): Hang ; // tìm theo tên hàng
```

**3.8.10 Loại j - Tìm kiếm tất cả các đối tượng của một lớp (\*)**

Loại hàm này tìm kiếm tất cả các đối tượng sử dụng khóa đối tượng (hoặc OID) của chính lớp đang xét Class\_1 (\*) dựa trên Class\_1

- o Không tham số, hoặc
- o Có tham số, dựa trên các thuộc tính (khác khóa) của Class\_1

Lưu ý : Khi tham số tương ứng với khóa của lớp, kết quả trả về chỉ 1 đối tượng. Ngược lại, trả về một mảng nhiều đối tượng, hoặc mảng nhiều khóa đối tượng.

Ví dụ :

```
ds_Hang () : Hang [ ] ;
ds_Hang (loai: Loai_hg) : Hang [ ]; // tìm theo loại hàng
ds_ma_Hang () : Character (8) [ ] ;
```

ds\_ma\_Hang (loai: Loai\_hg) : Character (8) []; // tìm theo loại hàng

### 3.8.11 Loại k - Tìm kiếm tất cả các đối tượng dựa trên lớp liên kết hoặc tất cả các khóa đối tượng (hoặc OID) của chính lớp đang xét Class\_1 (\*) dựa trên lớp C khác có liên kết với Class\_1

- **Class\_1** có thể là lớp liên kết
- Có tham số
- Dựa trên một đối tượng của C, hoặc
- Dựa trên các thuộc tính của C

#### Lưu ý :

- Bản số của kết quả trả về phải tương xứng chính xác với bản số của lớp C ở liên kết với lớp Class\_1.
- Đây là phương thức lớp nên sẽ không hiệu quả bằng phương thức loại 1 - phương thức thành viên- tiếp theo sau.

#### Ví dụ :

Mỗi đối tượng của lớp Loai\_hg gắn với nhiều đối tượng của lớp Hàng đang xét nên ta có :

ds\_Hang (loai: Loai\_hg) : Hang []; // tìm theo loại hàng  
 ds\_Hang (ma\_loai: C(8)) : Hang []; // tìm theo mã loại hàng

Tương tự, một hóa đơn có ghi nhiều mặt hàng và một kho chứa nhiều mặt hàng nên :

ds\_Hang (hd: Hoa\_don) : Hang []; // tìm theo hóa đơn  
 ds\_Hang (kh: Kho) : Hang []; // tìm theo kho chứa hàng

ds\_Hang (so\_hd: C(10)) : Hang []; // tìm theo hóa đơn  
 ds\_Hang (ma\_kho: C(8)) : Hang []; // tìm theo kho chứa hàng  
 ds\_Hang (ma\_NV: C(8)) : Hang []; // tìm theo NV bán hàng

Ngoài ra, nếu cần tìm mã hàng thôi theo một loại hàng, ta dùng các hàm sau :

ds\_ma\_Hang (loai: Loai\_hg) : Character (8) []; // tìm theo loại hàng

Tương tự cho việc tìm danh sách mã hàng theo các tiêu chí khác.

### 3.8.12 Loại l - Tìm kiếm các đối tượng/ OID của ít nhất một lớp khác C có liên kết với đối tượng đang xét c1 của Class\_1

Đây là loại phương thức rất hữu ích vì là phương thức thành viên gắn với chỉ một đối tượng đang xét, nên sẽ giới hạn lại khá nhiều phạm vi tìm kiếm, do đó sẽ nhanh hơn rất nhiều so với phương thức lớp.

Có thể tìm các giá trị khóa thay vì tìm các đối tượng. Trong các ví dụ dưới đây, sẽ chỉ đưa kết quả là danh sách các đối tượng. Các phương thức tìm các giá trị khóa sẽ được suy ra theo cách tương tự.

- Khởi đầu từ tìm kiếm qua 1 liên kết :

- Không tham số
- Có tham số, dựa trên các thuộc tính (không là khóa) của C
- Có tham số, dựa trên các thuộc tính của lớp liên kết (nếu có)

Lưu ý : Kết quả trả về của phương thức tùy thuộc vào bản số của Class\_1 tham gia vào liên kết :

- + sẽ là mảng (array) nếu bản số tối đa là \* (bản số là 1..\* hoặc 0..\*),
- + và chỉ là một đối tượng hoặc một dữ liệu có kiểu dữ liệu sơ cấp nếu bản số tối đa là 1 (bản số 0..1 hoặc 1).

Ví dụ :

// Các hàm tìm danh sách các hóa đơn có ghi mặt hàng đang xét:

```
DS_Hoa_don () : Hoa_don [] ;
DS_Hoa_don (td: Thoi_diem) : Hoa_don [] ;
DS_Hoa_don (ngay: Date, gio: Time) : Hoa_don [] ;
DS_Hoa_don (ngay_bd: Date, gio_bd: Time, ngay_kt: Date, gio_kt: Time):
    Hoa_don [] ;
DS_Hoa_don (don_gia: Long float (13,2)) : Hoa_don [] ;
DS_Hoa_don (so_luong: Long float (15,2)) : Hoa_don [] ;
DS_Hoa_don (tri_gia: Long float (13,2)) : Hoa_don [] ;
```

// tìm loại hàng của mặt hàng đang xét

loai\_hang(): Loai\_hg;

// Các hàm tìm danh sách các kho có chứa mặt hàng đang xét:

```
DS_Kho (): Kho [];
DS_Kho (so_luong_ton: Long float (16, 2)): Kho [];
DS_Kho (td: Thoi_diem): Kho [];
DS_Kho (so_luong_ton: Long float (16, 2), td: Thoi_diem): Kho [];
```

// Các hàm tìm danh sách các thời điểm có bán hoặc chứa mặt hàng đang xét:

```
DS_Thoi_diem_ban () : Thoi_diem [] ;
DS_Thoi_diem_co_ton_hang () : Thoi_diem [] ;
```

- Sau đó, có thể mở rộng tìm kiếm các đối tượng có liên kết với đối tượng đang xét qua n>1 liên kết.

Ví dụ :

```
DS_NV_ban (): NV [];
DS_NV_ban
(ngay_bd: Date, gio_bd: Time, ngay_kt: Date, gio_kt: Time): NV [];
```

### 3.8.13 Loại m - Thông kê dùng hàm kết tập

Dùng các hàm kết tập (sum, count, max, min, avg) để tính ra kết quả tổng số, trung bình, tối đa, tối thiểu. Các phương thức này cũng có dạng không/ có tham số như trên.

#### 3.8.13.1 Loại m1- Thông kê dùng hàm kết tập, chỉ tính trên toàn bộ lớp đang xét (\*)

Các phương thức loại này dựa vào các mối liên kết với các lớp khác.

Ví dụ :

TS\_Hang () : Long integer ; // tổng số mặt hàng

```

TS_Hang (m : C(8)) : Integer ;
    // tìm theo mã hàng, cho ra kết quả là 1 hoặc 0
TS_Hang (t : VC(30)) : Integer ;
    // tìm theo tên hàng, cho ra kết quả là 1 hoặc 0

TS_Hang (loai : Loai_hg) : Long integer ;
TS_Hang (ma_loai: C(8)) : Long integer ;
TS_Hang (kh: Kho) : Long integer ;
TS_Hang (ma_kho: C(8)) : Long integer ;
TS_Hang (hd: Hoa_don) : Long integer ;
TS_Hang (so_hd: C(10)) : Integer ;
TS_Hang (td: Thoi_diem) : Long Integer;
TS_Hang (ngay: Date, gio: Time) : Long Integer ;
// các phương thức này là k nêu, vì sẽ trùng với phương thức TS_Hang ()
// tại từng lớp Loai_hg, Hoa_don, Kho, Thoi_diem,
// do đó chúng phức tạp hơn vì chứa tham số và lại gây dư thừa
TB_so_Hang_tren_moi_Loai (): float (15,2);
TB_so_Hang_tren_moi_HD (): float (15,2);
TB_so_Hang_tren_moi_Kho (): float (15,2);

min_so_Hang_tren_moi_Loai (): Integer;
min_so_Hang_tren_moi_HD (): Integer;
min_so_Hang_tren_moi_Kho (): Integer;

max_so_Hang_tren_moi_Loai (): Integer;
max_so_Hang_tren_moi_HD (): Integer;
max_so_Hang_tren_moi_Kho (): Integer;

```

### 3.8.13.2 Loại m2 - Thông kê dùng hàm kết tập trên lớp khác có liên kết với đối tượng đang xét

Đây là phương thức thành viên, dùng các hàm tính hàm kết tập trên các đối tượng của các lớp C có liên kết với đối tượng đang xét. Lưu ý không tính tổng đối với các lớp mà lớp đang xét đang phụ thuộc hàm mạnh (vì luôn là 1).

Ví dụ : Ta đang ở một đối tượng mặt hàng của lớp Hàng.

- **Tính trên các đối tượng lớp khác**

```

TS_Hoa_don () : Integer ; // tổng số hóa đơn có ghi mặt hàng đang xét
TS_Hoa_don (td_bd, td_kt: Thoi_diem) : Integer ;
    // tổng số hóa đơn trong một khoảng thời gian có ghi mặt hàng đang xét
TS_Hoa_don (don_gia: Long Float (15,2)) : Integer ;
    // tổng số hóa đơn có ghi mặt hàng đang xét với một đơn giá đã cho
TS_Hoa_don (sl_ban: Long Float (15,2)) : Integer ;
    // tổng số hóa đơn có ghi mặt hàng đang xét với một số lượng bán cho
    // trước

```

TS\_Kho () : Integer ;

```

        // tổng số kho có chứa mặt hàng đang xét
TS_Kho (td_bd, td_kt: Thoi_diem) : Integer ;
        // tổng số kho có chứa mặt hàng đang xét trong một khoảng thời gian
TS_Kho (sl_ton: Long Float (15,2)) : Integer ;
        // tổng số kho có chứa một lượng hàng tồn cho trước
        // đối với mặt hàng đang xét
TS_Kho (td: Thoi_diem, sl_ton: Long Float (15,2)) : Integer ;
        // tổng số kho có chứa một lượng hàng tồn cho trước
        // đối với mặt hàng đang xét tại một thời điểm

TS_Thoi_diem (): Integer ;
        // tổng số lần có kiểm kê mặt hàng đang xét
TS_Thoi_diem (kh: Kho): Integer ;
        // tổng số lần có kiểm kê mặt hàng đang xét tại một kho cho trước

max_Thoi_diem_kiem_kho (): Integer ;
        // thời điểm gần nhất có kiểm kê mặt hàng đang xét
max_Thoi_diem_kiem_kho (kh: Kho): Integer ;
        // thời điểm gần nhất có kiểm kê mặt hàng đang xét tại kho cho trước

```

- **Tính trên các thuộc tính của lớp liên kết**

```

max_so_Hoa_don_theo_TG (dinh_ky: VC(5)) : Integer ;
        // số hóa đơn tối đa có ghi mặt hàng đang xét
        // theo định kỳ (tuần, tháng, quý, năm ...)

min_Thoi_diem_kiem_kho (): Integer ;
        // thời điểm xa nhất có kiểm kê mặt hàng đang xét
min_Thoi_diem_kiem_kho (kh: Kho): Integer ;
        // thời điểm xa nhất có kiểm kê mặt hàng đang xét tại kho cho trước

TB_so_Hoa_don_theo_TG (dinh_ky: VC(5)) : Integer ;
        // trung bình số hóa đơn có ghi mặt hàng đang xét
        // theo định kỳ (tuần, tháng, quý, năm ...)

TB_so_lg_ban_tren_HD (): Long Float (15,2)) ;
        // trung bình số lượng bán mặt hàng đang xét tính trên tất cả các hóa đơn

TB_so_lg_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
        // trung bình số lượng bán mặt hàng đang xét tính trên tất cả các hóa đơn
        // trong một khoảng thời gian
TB_don_gia_ban_tren_HD (): Long Float (15,2)) ;
        // trung bình đơn giá bán mặt hàng đang xét tính trên tất cả các hóa đơn
TB_don_gia_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
        // trung bình đơn giá bán mặt hàng đang xét tính trên tất cả các hóa đơn
        // trong một khoảng thời gian
TB_so_lg_ton_tren_Kho (): Long Float (15,2)) ;

```

```

    // trung bình số lượng tồn mặt hàng đang xét tính trên tất cả các kho
TB_so_lg_ton_tren_Kho (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // trung bình số lượng tồn mặt hàng đang xét tính trên tất cả các kho
    // trong một khoảng thời gian

max_so_lg_ban_tren_HD (): Long Float (15,2)) ;
    // số lượng bán tối đa mặt hàng đang xét tính trên tất cả các hóa đơn

max_so_lg_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng bán tối đa mặt hàng đang xét tính trên tất cả các hóa đơn
    // trong một khoảng thời gian

max_don_gia_ban_tren_HD (): Long Float (15,2)) ;
    // đơn giá bán đắt nhất của mặt hàng đang xét tính trên tất cả các hóa đơn

max_don_gia_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // đơn giá bán đắt nhất của mặt hàng đang xét tính trên tất cả các hóa đơn
    // trong một khoảng thời gian

max_so_lg_ton_tren_Kho (): Long Float (15,2)) ;
    // số lượng tồn nhiều nhất của mặt hàng đang xét tính trên tất cả các kho

max_so_lg_ton_tren_Kho (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng tồn nhiều nhất của mặt hàng đang xét tính trên tất cả các kho
    // trong một khoảng thời gian

max_so_lg_ton_tai_Kho (kho: Kho): Long Float (15,2)) ;
    // số lượng tồn nhiều nhất của mặt hàng đang xét tại một kho cho trước
    // tính trên toàn thời gian

max_so_lg_ton_tren_Kho
(kho: Kho, td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng tồn nhiều nhất của mặt hàng đang xét tại một kho cho trước
    // trong một khoảng thời gian

max_so_lg_ban_tren_HD (): Long Float (15,2)) ;
    // số lượng bán tối đa mặt hàng đang xét tính trên tất cả các hóa đơn

min_so_lg_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng bán tối thiểu mặt hàng đang xét tính trên tất cả các hóa đơn
    // trong một khoảng thời gian

min_don_gia_ban_tren_HD (): Long Float (15,2)) ;
    // đơn giá bán rẻ nhất của mặt hàng đang xét tính trên tất cả các hóa đơn

min_don_gia_ban_tren_HD (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // đơn giá bán rẻ nhất của mặt hàng đang xét tính trên tất cả các hóa đơn
    // trong một khoảng thời gian

min_so_lg_ton_tren_Kho (): Long Float (15,2)) ;
    // số lượng tồn ít nhất của mặt hàng đang xét tính trên tất cả các kho

min_so_lg_ton_tren_Kho (td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng tồn ít nhất của mặt hàng đang xét tính trên tất cả các kho
    // trong một khoảng thời gian

min_so_lg_ton_tai_Kho (kho: Kho): Long Float (15,2)) ;
    // số lượng tồn ít nhất của mặt hàng đang xét tại một kho cho trước

```

```

min_so_lg_ton_tren_Kho
(kho: Kho, td_bd, td_kt: Thoi_diem): Long Float (15,2)) ;
    // số lượng tồn ít nhất của mặt hàng đang xét tại một kho cho trước
    // trong một khoảng thời gian

```

### 3.8.14 Loại n - Tính và hiển thị kết quả thống kê dùng hàm kết tập trên nhóm (\*)

Giống như loại m, dùng các hàm kết tập (sum, count, max, min, avg) để tính ra kết quả, nhưng được thực hiện trên từng nhóm. Tiêu chí chia nhóm là trên thuộc tính, hoặc trên lớp có liên kết với lớp đang xét. Phương thức thường là thủ tục.

#### 3.8.14.1 Loại n1- Thông kê dùng hàm kết tập trên nhóm, chỉ tính trên toàn bộ lớp đang xét (\*)

Các phương thức loại này:

- Dựa trên các thuộc tính (không là khóa) của lớp đang xét
- Có thể dựa vào các mối liên kết với các lớp khác

Ví dụ :

```

TS_Hang_theo_loai () ;
    // tính và hiển thị tổng số mặt hàng ở từng loại
TS_Hang_theo_muc_don_gia_ban (loai: Loai_hg; buoc_gia: Long Integer) ;
    // tính và hiển thị tổng số mặt hàng ở từng mức đơn giá bán
    // với loại hàng và bước nhảy giá cho trước
TS_Hang_theo_muc_don_gia_ban
(loai: Loai_hg; buoc_gia: Long Integer; td_bd, td_kt: Thoi_diem) ;
    // tính và hiển thị tổng số mặt hàng ở từng mức đơn giá bán
    // với loại hàng và bước nhảy giá cho trước
    // trong khoảng thời gian cho trước
TS_Hang_theo_muc_so_luong_ban (loai: Loai_hg; buoc_so_lg: Integer) ;
    // tính và hiển thị tổng số mặt hàng ở từng mức số lượng bán
    // với loại hàng và bước nhảy số lượng cho trước
TS_Hang_theo_muc_don_gia_ban
(loai: Loai_hg; buoc_gia: Long Integer; td_bd, td_kt: Thoi_diem) ;
    // tính và hiển thị tổng số mặt hàng ở từng mức đơn giá bán
    // với loại hàng và bước nhảy giá cho trước
    // trong khoảng thời gian cho trước
TS_Hang_theo_kho () ;
    // tính và hiển thị tổng số mặt hàng chúa ở từng kho
TS_Hang_theo_kho (td_bd, td_kt: Thoi_diem) ;
    // tính và hiển thị tổng số mặt hàng chúa ở từng kho
    // trong khoảng thời gian cho trước
TS_Hang_theo_quy (n: Integer) ;
    // tính và hiển thị tổng số mặt hàng bán ở từng quý của năm cho trước
TS_Hang_theo_thang (n: Integer) ;
    // tính và hiển thị tổng số mặt hàng bán ở từng tháng của năm cho trước
TS_Hang_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem) ;

```

```
// tính và hiển thị tổng số mặt hàng bán
// ở từng định kỳ (tháng, quý, tuần, ngày)
// của khoảng thời gian cho trước
```

### 3.8.14.2 Loại n2- Thống kê dùng hàm kết tập theo nhóm trên lớp khác có liên kết với đối tượng đang xét

Đây là phương thức thành viên, dùng các hàm tính hàm kết tập theo nhóm trên các đối tượng của các lớp C có liên kết với đối tượng đang xét. Lưu ý không tính tổng đối với các lớp mà lớp đang xét đang phụ thuộc hàm mạnh (vì luôn là 1).

Ví dụ : Ta đang ở một đối tượng mặt hàng của lớp Hàng.

- **Tính trên các đối tượng lớp khác**

```
TS_Hoa_don_theo_quy (n: Integer) ;
    // tính và hiển thị tổng số hóa đơn theo quý của năm cho trước
    // có ghi mặt hàng đang xét
TS_Hoa_don_theo_thang (n: Integer) ;
    // tính và hiển thị tổng số hóa đơn theo tháng của năm cho trước
    // có ghi mặt hàng đang xét
TS_Hoa_don_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem) ;
    // tính và hiển thị tổng số hóa đơn theo định kỳ (tháng, quý, tuần, ngày)
    // của khoảng thời gian cho trước
    // có ghi mặt hàng đang xét
TS_Hoa_don_theo_NV (n: Integer) ;
    // tính và hiển thị tổng số hóa đơn theo nhân viên bán hàng
    // của năm cho trước
    // có ghi mặt hàng đang xét
```

- **Tính trên các thuộc tính của lớp liên kết**

```
TS_doanh_so_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị tổng doanh số của mặt hàng đang xét trên mỗi định kỳ
    // trong khoảng thời gian cho trước
TB_doanh_so_tren_Hoa_don
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị trung bình doanh số trên hóa đơn
    // của mặt hàng đang xét
    // mỗi định kỳ trong khoảng thời gian cho trước
max_doanh_so_tren_Hoa_don
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị doanh số tối đa trên hóa đơn
    // của mặt hàng đang xét
    // mỗi định kỳ trong khoảng thời gian cho trước
min_doanh_so_tren_Hoa_don
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị doanh số tối thiểu trên hóa đơn
    // của mặt hàng đang xét
    // mỗi định kỳ trong khoảng thời gian cho trước
```

```

TB_don_gia_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị trung bình đơn giá bán trong mỗi định kỳ
    // của mặt hàng đang xét
    // trong khoảng thời gian cho trước
max_don_gia_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị đơn giá bán đắt nhất
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
min_don_gia_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị đơn giá bán rẻ nhất
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
TB_so_luong_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị trung bình số lượng bán
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
max_so_luong_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng bán nhiều nhất trên hóa đơn
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
min_so_luong_ban_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng bán ít nhất trên hóa đơn
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
TB_so_luong_ton_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị trung bình số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
max_so_luong_ton_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
min_so_luong_ton_theo_dinh_ky
(dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
TB_so_luong_ton_theo_dinh_ky_tai_mot_kho

```

```

(kho: Kho; dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị trung bình số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
max_so_luong_ban_theo_dinh_ky_tai_mot_kho
(kho: Kho; dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước
min_so_luong_ban_theo_dinh_ky_tai_mot_kho
(kho: Kho; dinh_ky: Variable Character; td_bd, td_kt: Thoi_diem);
    // tính và hiển thị số lượng số lượng tồn trong mỗi kho
    // trong mỗi định kỳ của mặt hàng đang xét
    // trong khoảng thời gian cho trước

```

### 3.8.15 Loại o - Tìm kiếm đối tượng của lớp đang xét có một thuộc tính lớp/ thuộc tính liên kết đạt trị min/ max

Ví dụ :

```

hang_dat_gia_nhat(): Hang [ ];
    // tìm các mặt hàng đắt giá nhất
hang_re_nhat(): Hang [ ];
    // tìm các mặt hàng rẻ nhất
hang_dat_gia_nhat(td_bd, td_kt: Thoi_diem): Hang [ ];
    // tìm các mặt hàng đắt giá nhất
    // trong khoảng thời gian cho trước
hang_re_nhat(td_bd, td_kt: Thoi_diem): Hang [ ];
    // tìm các mặt hàng đắt giá nhất
    // trong khoảng thời gian cho trước
hang_dat_gia_nhat(loai: Loai_hg; td_bd, td_kt: Thoi_diem): Hang [ ];
    // tìm các mặt hàng đắt giá nhất
    // trong số các mặt hàng của loại hàng cho trước
    // trong khoảng thời gian cho trước
hang_re_nhat(loai: Loai_hg; td_bd, td_kt: Thoi_diem): Hang [ ];
    // tìm các mặt hàng đắt giá nhất
    // trong số các mặt hàng của loại hàng cho trước
    // trong khoảng thời gian cho trước

```

### 3.8.16 Loại p - Tìm kiếm đối tượng của các lớp có kết nối với lớp đang xét có một thuộc tính liên kết đạt trị min/ max

Ví dụ :

```

kho_chua_nhieu_nhat(): Kho [ ];
    // tìm các kho hiện chứa mặt hàng đang xét với số lượng tồn nhiều nhất
kho_chua_it_nhat(): Kho [ ];
    // tìm các kho hiện chứa mặt hàng đang xét với số lượng tồn ít nhất
hoa_don_ban_sl_nhieu_nhat(): Hoa_don [ ];

```

```

        // tìm hóa đơn bán mặt hàng đang xét với số lượng nhiều nhất
        hoa_don_ban_gia_dat_nhat(): Hoa_don [ ];
            // tìm các hóa đơn bán mặt hàng đang xét với đơn giá đắt nhất
            hoa_don_ban_gia_re_nhat(): Hoa_don [ ];
                // tìm các hóa đơn bán mặt hàng đang xét với đơn giá rẻ nhất
                hoa_don_ban_tri_gia_nchieu_nhat(): Hoa_don [ ];
                    // tìm hóa đơn bán mặt hàng đang xét với trị giá bán nhiều nhất

        kho_chua_nchieu_nhat(td_bd, td_kt: Thoi_diem): Kho [ ];
            // tìm các kho hiện chứa mặt hàng đang xét với số lượng tồn nhiều nhất
            // trong khoảng thời gian cho trước
        kho_chua_it_nhat(td_bd, td_kt: Thoi_diem): Kho [ ];
            // tìm các kho hiện chứa mặt hàng đang xét với số lượng tồn ít nhất
            // trong khoảng thời gian cho trước
        hoa_don_ban_sl_nchieu_nhat(td_bd, td_kt: Thoi_diem): Hoa_don [ ];
            // tìm hóa đơn bán mặt hàng đang xét với số lượng nhiều nhất
            // trong khoảng thời gian cho trước
        hoa_don_ban_gia_dat_nhat(td_bd, td_kt: Thoi_diem): Hoa_don [ ];
            // tìm các hóa đơn bán mặt hàng đang xét với đơn giá đắt nhất
            // trong khoảng thời gian cho trước
        hoa_don_ban_gia_re_nhat(td_bd, td_kt: Thoi_diem): Hoa_don [ ];
            // tìm các hóa đơn bán mặt hàng đang xét với đơn giá rẻ nhất
            // trong khoảng thời gian cho trước
        hoa_don_ban_tri_gia_nchieu_nhat(td_bd, td_kt: Thoi_diem): Hoa_don [ ];
            // tìm hóa đơn bán mặt hàng đang xét với trị giá bán nhiều nhất
            // trong khoảng thời gian cho trước

        Thoi_diem_ban_gia_dat_nhat(): Thoi_diem [ ];
            // tìm các thời điểm bán mặt hàng đang xét với đơn giá đắt nhất
        Thoi_diem_ban_gia_re_nhat(): Thoi_diem [ ];
            // tìm các thời điểm bán mặt hàng đang xét với đơn giá rẻ nhất
    
```

### 3.8.17 Loại q - Tìm kiếm các đối tượng có một kết quả tổng theo nhóm đạt trị min/ max

Ở loại này chỉ dùng một loại hàm kết tập là tính tổng.

Ví dụ :

```

hang_chua_o_nchieu_kho_nhat(): Hang [];
// ds các mặt hàng chứa trong nhiều kho nhất
hang_ban_o_nchieu_hoa_don_nhat(): Hang [];
// ds các mặt hàng bán được trong nhiều hóa đơn nhất
    
```

### 3.8.18 Loại r- Kiểm tra một trị có là khóa của lớp đang xét không

Đây là một hàm trả về trị kiểu luận lý (đúng/ sai). Hàm này chỉ được dùng nếu lớp đang xét có khóa. Tham số phải có cùng kiểu với thuộc tính khóa.

Ví dụ :

```
ktra_la_khoa (m: Character(8)): Boolean;
ktra_la_khoa (m: Character(8)): Bit;
// trị 1 là đúng, trị 0 là sai
```

**3.8.19 Loại t - Tính và hiển thị kết quả thống kê dùng hàm kết tập (\*)**

Giống như loại m, nhưng là thủ tục, dùng các hàm kết tập (sum, count, max, min, avg) để tính ra kết quả, và hiển thị kết quả luôn trên màn hình. Cũng có dạng không/ có tham số như trên. Loại này được dùng khi không cần kết quả cho một tác vụ nào đó về sau.

Các ví dụ như ở các ví dụ của mục 3.8.13, ở đây chỉ trích dẫn một ít vì hoàn toàn có thể suy ra số ví dụ còn lại.

**3.8.19.1 Loại t1- Tính và hiển thị kết quả thống kê dùng hàm kết tập, chỉ tính trên toàn bộ lớp đang xét (\*)**

Các phương thức loại này dựa vào các mối liên kết với các lớp khác.

Ví dụ :

```
Hien_TS_Hang () ; // tính và hiển thị tổng số mặt hàng
```

**3.8.19.2 Loại t2- Tính và hiển thị kết quả thống kê dùng hàm kết tập trên lớp khác có liên kết với đối tượng đang xét**

Đây là phương thức thành viên, dùng các hàm tính hàm kết tập trên các đối tượng của các lớp C có liên kết với đối tượng đang xét. Lưu ý không tính tổng đối với các lớp mà lớp đang xét đang phụ thuộc hàm mạnh (vì luôn là 1).

Ví dụ : Ta đang ở một đối tượng mặt hàng của lớp Hàng.

- **Tính trên các đối tượng lớp khác**

```
Hien_TS_Hoa_don () : Integer ;
// tính và hiển thị tổng số hóa đơn có ghi mặt hàng đang xét
```

```
Hien_TS_Hoa_don (td_bd, td_kt: Thoi_diem) : Integer ;
// tính và hiển thị tổng số hóa đơn trong một khoảng thời gian có ghi mặt hàng đang xét
```

- **Tính trên các thuộc tính của lớp liên kết**

```
Hien_max_so_Hoa_don_theo_TG (dinh_ky: VC(5)) : Integer ;
// tính và hiển thị số hóa đơn tối đa có ghi mặt hàng đang xét
// theo định kỳ (tuần, tháng, quý, năm ...)
```

```
Hien_min_Thoi_diem_kiem_kho (): Integer ;
// tính và hiển thị thời điểm xa nhất có kiểm kê mặt hàng đang xét
```

```
Hien_min_Thoi_diem_kiem_kho (kh: Kho): Integer ;
// tính và hiển thị thời điểm xa nhất có kiểm kê mặt hàng đang xét tại kho
cho trước
```

**3.8.20 Loại u - In đối tượng đang xét**Ví dụ :

```
inLop();
```

### **3.8.21 Loại v - In danh sách tất cả các đối tượng**

Loại phương thức này in danh sách các đối tượng là kết quả tìm kiếm hoặc thống kê. Loại phương thức này là phương thức lớp. Đây là loại suy diễn từ tất cả các loại phương thức có liên quan tìm kiếm hoặc thống kê.

Ví dụ :

```
in_Ds_Hang();  
in_Ds_Hang_theo_kho();
```

