

Chương 4

SƠ ĐỒ TƯƠNG TÁC (INTERACTION DIAGRAM)

4.1 GIỚI THIỆU

Sơ đồ hoạt vụ chỉ ra các tác nhân tương tác với các chức năng lớn của một hệ thống. Đó là một cách nhìn thiên về chức năng và từ bên ngoài hệ thống.

Sơ đồ lớp mô tả nhân của hệ thống với các lớp và cách thức chúng kết hợp lại với nhau. Đó là một cách nhìn tĩnh và thiên về cấu trúc.

Các sơ đồ tương tác, về phần mình, cho phép đặt câu nối giữa hai cách tiếp cận trên. Chúng cho thấy làm thế nào các thể hiện ở bên trong hệ thống có thể tương tác với nhau để thực hiện một chức năng nào đó.

Các tương tác khá nhiều và rất đa dạng, cần có một ngôn ngữ phong phú để biểu diễn chúng. UML cung cấp nhiều sơ đồ cho mục đích này: sơ đồ tuần tự (sequence diagram), sơ đồ cộng tác (collaboration diagram, hoặc còn gọi là sơ đồ liên lạc, sơ đồ truyền thông- communication diagram), sơ đồ về thời gian (timing diagram). Các sơ đồ này mang lại phong cách động cho việc mô hình hóa hệ thống.

4.2 LỢI ÍCH CỦA TƯƠNG TÁC (INTERACTION)

4.2.1 Định nghĩa

4.2.1.1 Tương tác (interaction)

Một tương tác định nghĩa hành vi của một hệ có cấu trúc (một hệ thống con, một trường hợp sử dụng, một lớp, một thành tố, ...) bằng cách đặc biệt chú trọng đến việc trao đổi các thông tin qua các thông điệp (message) giữa các thể hiện (instances) của nó.

Ví dụ:

- Khi nhân viên bán hàng chọn chức năng lập hóa đơn, giao diện lập hóa đơn hiện ra, có hiển thị sẵn mã số, họ tên nhân viên
⇒ Tương tác giữa một thể hiện *gd* của lớp giao diện lập hóa đơn *GD_HD* và đối tượng *nv* của lớp Nhân viên tương ứng nhân viên đó.
- Khi nhân viên bán hàng quét thẻ khách hàng, thì mã số và họ tên khách hàng, ngày giờ, số hóa đơn được gán tự động trên giao diện. Một bảng gồm tiêu đề sẽ được in sẵn
⇒ Tương tác giữa *nv* và *gd*, giữa *gd* và đối tượng *th* của lớp Thẻ KH tương ứng thẻ được quét, giữa *gd* và đối tượng mới *hđ* của lớp Hóa đơn.

- Mỗi lần nhân viên bán hàng quét mặt hàng qua máy quét, một dòng trắng xuất hiện và dưới cột “Mã hàng”, “Tên hàng” và “Đơn giá” sẽ hiện lên các giá trị tương ứng lấy từ các lớp Hàng” và Bảng giá đang có hiệu lực
⇒ Tương tác giữa *hđ* và đối tượng *h* của lớp Hàng tương ứng mặt hàng được quét, giữa *h* và đối tượng *bg* tương ứng của lớp Bảng giá.
- Khi màn hình yêu cầu nhân viên bán hàng nhập số lượng một mặt hàng được mua, cột “Số lượng” hiện lên một hướng dẫn qui cách nhập số lượng, người này nhập một con số ở dòng hiện hành tại ô dành sẵn
⇒ Tương tác giữa *gd* và *nv*.
- Nhưng nếu nhân viên lập hóa đơn nhập vào một con số không nằm trong giới hạn cho phép (âm hoặc một con số dương nhưng quá lớn), trên màn hình xuất hiện thông điệp lỗi
⇒ Tương tác giữa *gd* và *nv*.

4.2.1.2 Định nghĩa sơ đồ tương tác

Các sơ đồ tuần tự và sơ đồ liên lạc biểu diễn các tương tác giữa các sinh tuyến. Một sơ đồ tuần tự chỉ ra các tương tác dưới góc độ thời gian, đặc biệt là các chuỗi thông điệp (messages) trao đổi nhau giữa các sinh tuyến, trong khi một sơ đồ liên lạc biểu diễn về mặt không gian của các sinh tuyến.

Còn có dạng thứ ba của sơ đồ tương tác là sơ đồ thời gian nhằm mô hình hóa các hệ thống được khai thác dưới những điều kiện nghiêm ngặt về thời gian, chẳng hạn các hệ thống thời thực. Tuy nhiên, các sơ đồ tuần tự hoàn toàn có thể thực hiện chức năng này.

4.2.1.3 Định nghĩa sinh tuyến (lifeline)

Một sinh tuyến biểu diễn một thành phần duy nhất (có thể tương ứng với một người ngoài thế giới thực, hoặc một đối tượng trong lớp dữ liệu) tham gia vào một tương tác.

Sinh tuyến hay được dùng cho sơ đồ tuần tự.

4.2.1.4 Liên kết (link)

Trong sơ đồ cộng tác, mỗi liên kết biểu diễn tương tác giữa một tác nhân với một lớp, hoặc một lớp với một lớp.

4.2.2 Ký hiệu

4.2.2.1 Ký hiệu sơ đồ tương tác

Biểu diễn bằng một hình chữ nhật, góc trái có tên của sơ đồ, đứng sau từ khóa:

- *sd*: nếu là sơ đồ tuần tự
- *com*: nếu là sơ đồ liên lạc/ sơ đồ cộng tác

Có thể có :

- Danh sách các sinh tuyến đứng tiếp theo với từ khóa lifelines và dấu hai chấm.
- Ràng buộc trên ít nhất một thuộc tính nào đó có dùng trong sơ đồ

Thông thường, các sơ đồ tuần tự giản lược đi phân liệt kê sinh tuyến và các ràng buộc.

4.2.2.2 Ký hiệu sinh tuyến

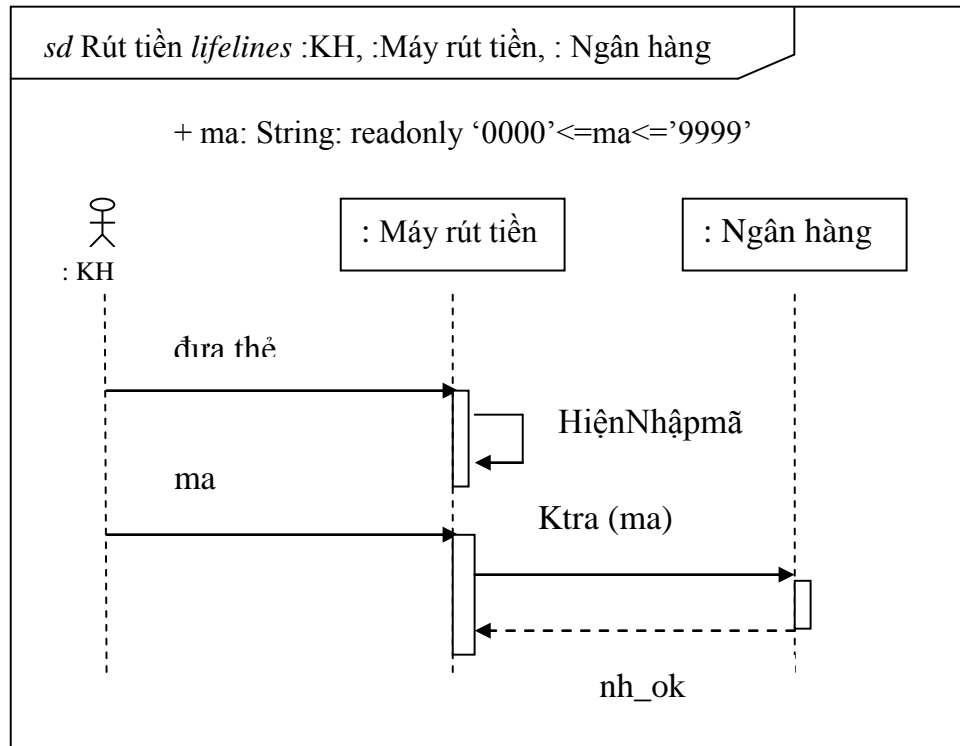
Trong sơ đồ tuần tự, mỗi sinh tuyến biểu diễn bằng một đường thẳng đứng chấm dứt đoạn, chiều đi xuống, tượng trưng cho trục thời gian. Các sinh tuyến được dàn hàng ngang từ trái sang phải, trong khi các thông điệp sẽ thực thi theo tiến độ thời gian theo chiều từ trên cao xuống thấp, từ sớm nhất cho đến trễ nhất. Như vậy, sơ đồ tuần tự có 2 chiều.

Sinh tuyến gắn với một thể hiện của một tác nhân (biểu diễn bằng hình người) hoặc gắn với một đối tượng của một lớp dữ liệu (biểu diễn bằng hình chữ nhật). Sinh tuyến giao diện ngăn không cho các sinh tuyến tác nhân và lớp dữ liệu trao đổi trực tiếp thông điệp với nhau. Theo qui ước, sinh tuyến tượng trưng tác nhân, nếu có, nằm bên trái của sinh tuyến giao diện, và các sinh tuyến tương ứng các lớp dữ liệu nằm bên phải sinh tuyến giao diện. Thông thường, sinh tuyến lớp nào trao đổi thông điệp với sinh tuyến giao diện sớm hơn sẽ nằm gần bên trái hơn.

Trong hình chữ nhật hoặc phía dưới hình người chứa một định danh có ngữ pháp như sau :

[<tên sinh tuyến> [<chỉ số>]] : <tên lớp>

Ví dụ 1:

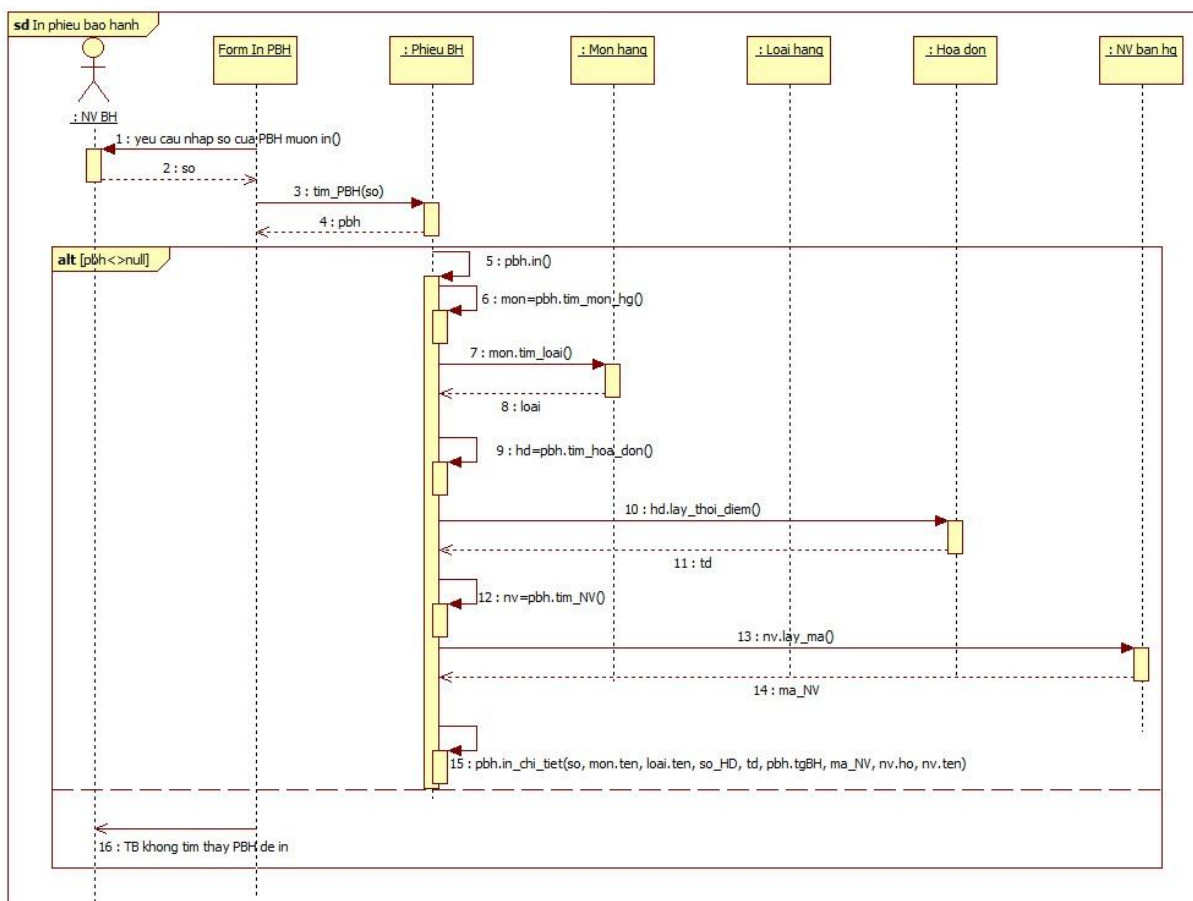


Hình 4.1 Ví dụ về sơ đồ tuần tự có ràng buộc và liệt kê sinh tuyến

Hình 4.1 biểu diễn một sơ đồ rất đơn giản, mục đích chỉ nhằm giới thiệu bước đầu các khái niệm. Trong hình, có một sinh tuyến tượng trưng cho một đối tượng của lớp « Ngân hàng », một sinh tuyến có ý nghĩa biểu tượng cho một khách hàng cụ thể, tức một thể hiện của tác nhân « KH », và một sinh tuyến tương ứng một thể hiện của giao diện « Máy rút tiền ».

Ràng buộc trong sơ đồ nêu lên điều kiện phải thỏa cho biến « ma » : phải là chuỗi, chỉ được đọc, chứ không được sửa, phải có trị nằm trong khoảng trị từ '0000' đến '9999'.

Ví dụ 2:

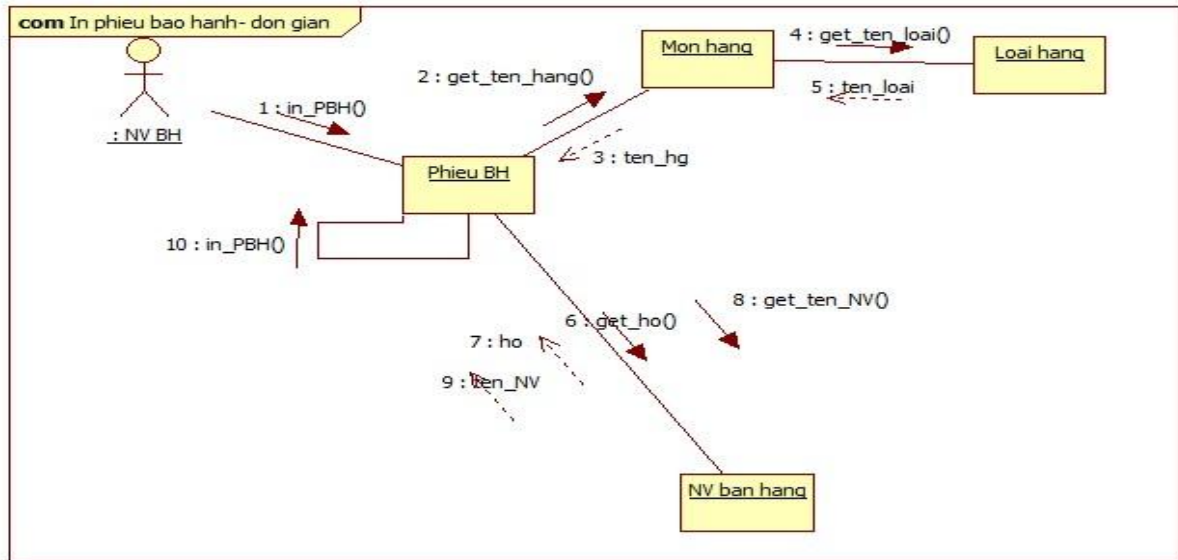


Hình 4.2 Ví dụ về sơ đồ tuần tự thông thường

4.2.2.3 Liên kết (link)

Trong sơ đồ cộng tác, mỗi liên kết được vẽ bằng một đoạn thẳng nối liền một hình người với một hình chữ nhật chứa tên lớp, hoặc giữa hai hình chữ nhật chứa tên lớp với nhau. Trên trục đoạn thẳng đó, có một số các thông điệp trao đổi giữa các đối tượng.

Ví dụ:



Hình 4.3 Ví dụ về sơ đồ cộng tác đơn giản

4.2.3 Ứng dụng của sơ đồ tương tác

Các sơ đồ tương tác được sử dụng trong suốt một dự án, từ khi thu thập các nhu cầu cho đến khi thiết kế. Chúng dùng để :

- Mô tả các trường hợp sử dụng
- Mô hình hóa việc sử dụng một lớp hoặc một phương thức của lớp
- Thêm vào khía cạnh động cho việc mô hình hóa một hệ thống.

Sơ đồ lớp trước tiên mô hình hóa lĩnh vực nào mà hệ thống sẽ được sử dụng trong đó, và lĩnh vực này tương đối độc lập với các CT ứng dụng (application). Việc cài đặt một sơ đồ lớp chỉ là một cơ sở để phát triển trình ứng dụng.

Đặc trưng của một trình ứng dụng là cách thức nó sử dụng sơ đồ lớp ra sao, nghĩa là cách thức các thể hiện của lớp tương tác nhau để thực hiện các chức năng của trình ứng dụng đó. Không có khía cạnh động này, một hệ thống chỉ là thuần túy tĩnh và không mang lại chức năng gì.

Ta không nên mô tả tất cả các tương tác chỉ trên một sơ đồ. Cần phải nhắm vào một hệ thống con và nghiên cứu cách thức các phần tử của nó tương tác với nhau để mô tả một ứng xử (behavior) đặc thù của hệ thống.

Sơ đồ tuần tự thường dùng để mô tả các tương tác trong một hệ thống con, một trường hợp sử dụng, một lớp, một thành tố, ..., nghĩa là trong một hệ có cấu trúc. Còn sơ đồ liên lạc dùng trong một ngữ cảnh đặc thù nào đó nhằm sử dụng một chức năng nào đó của hệ thống. Ở đây, ta chủ yếu nghiên cứu về sơ đồ tuần tự.

4.3 THÔNG điệp (MESSAGE)

4.3.1 Các dạng thông điệp

Các thông tin chủ yếu chứa trong các sơ đồ tuần tự là các thông điệp. Một thông điệp định nghĩa một liên lạc giữa các sinh tuyến. Có nhiều dạng thông điệp, trong đó phổ biến nhất là dạng:

(1) Gửi một tín hiệu (send). Thông điệp khởi động một phản ứng nơi đối tượng nhận một cách không đồng bộ (asynchrone) và không cần đợi trả lời. Lệnh gán trị cho biến có thể được xếp vào loại này.

(2) Kích hoạt thực hiện một thao tác trong một phương thức (call). Đây là dạng thông điệp được sử dụng nhiều nhất trong lập trình hướng đối tượng. Ví dụ : *dt.pt()*, trong đó *dt* là đối tượng nhận thông điệp và *pt* là một phương thức của *dt*. Trong thực tế, đa số thông điệp loại này là đồng bộ. Cũng có thể thực hiện thông điệp dạng này một cách không đồng bộ qua các thread.

Ta chia dạng này thành 2 loại nhỏ hơn là gọi hàm và gọi thủ tục. Hàm sẽ trả về một giá trị nên sẽ có thông điệp trả về (return), còn thủ tục thì không.

(3) : Tạo (create) hoặc hủy (destroy) một thể hiện (đối tượng)

Lưu ý :

Khi dùng biến trong lời gọi phương thức, biến phải được khai báo, khởi tạo trước đó, bằng cách: (để sinh ra lỗi “variable not found” khi chạy chương trình)

- Được gán trị bởi hệ thống
- Được nhập bởi tác nhân
- Được tính từ một hàm

4.3.2 Ký hiệu

4.3.2.1 Tổng quát

Một thông điệp từ đối tượng A sang đối tượng B được biểu diễn bằng một mũi tên.

Trong sơ đồ tuần tự, thông điệp đi từ sinh tuyến của A đến sinh tuyến của B (có khi được gọi là stimulus). Một đối tượng cũng có thể gửi thông điệp đến chính nó (tự kích hoạt : self-stimulus).

Trong sơ đồ cộng tác, thông điệp đi dọc theo đường liên kết nối 2 đối tượng và hướng về đối tượng nhận.




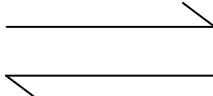
4.3.2.2 Thông điệp đồng bộ và thông điệp không đồng bộ

UML phân biệt 2 loại thông điệp:

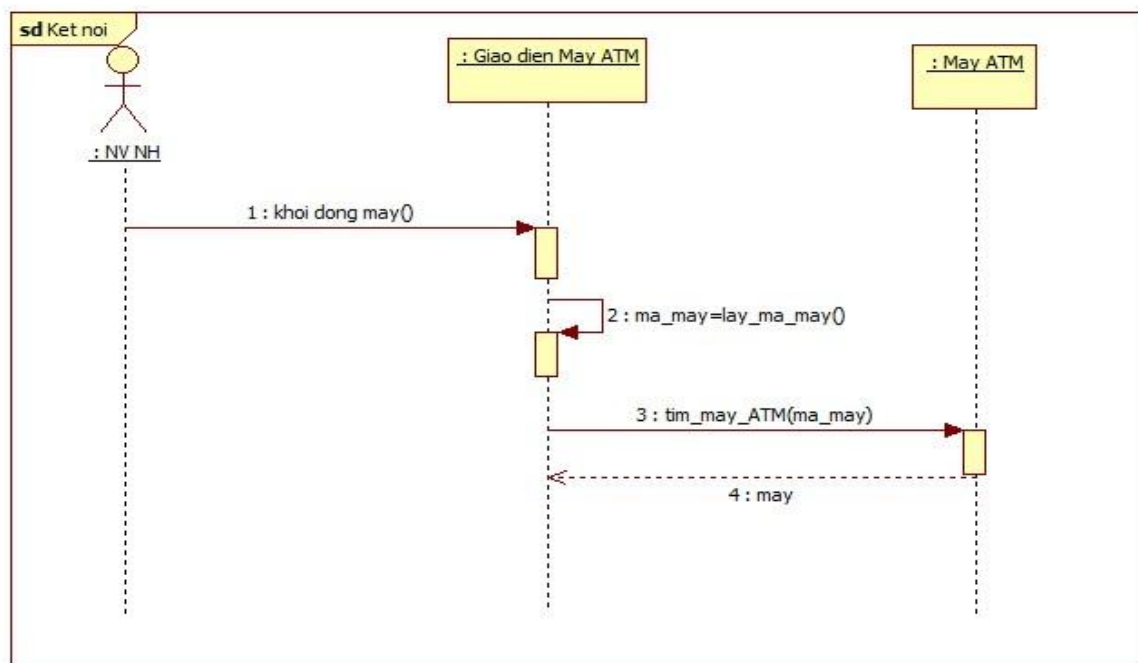
- Thông điệp đồng bộ (sychrone) : Cuộc gọi đồng bộ, được liên kết với một hoạt động, có một tin nhắn gửi và một tin nhắn nhận được. Một thông điệp được gửi từ sinh tuyến nguồn tới sinh tuyến đích. Sinh tuyến nguồn bị chặn từ các hoạt động khác cho đến khi nó nhận được một trả lời từ sinh tuyến đích. Ở cuối mũi tên, ở đối tượng nhận, có thể có trả lời phản hồi bằng mũi tên đứt đoạn, dưới hình chữ nhật biểu diễn khoảng thời gian xử lý (duration).

- Thông điệp không đồng bộ (asynchrone) : Các cuộc gọi không đồng bộ, được kết hợp với các hoạt động, thường chỉ có một tin nhắn gửi, nhưng cũng có thể có một tin nhắn trả lời. Ngược lại với một thông điệp đồng bộ, sinh tuyến nguồn không bị chặn từ việc nhận hoặc gửi tin nhắn khác.

Bảng 4.1 Ký hiệu thông điệp đồng bộ và bất đồng bộ

	Thông điệp đồng bộ (sychrone)	Thông điệp không đồng bộ (asynchrone)
Cách 1		
Cách 2		

Ví dụ :

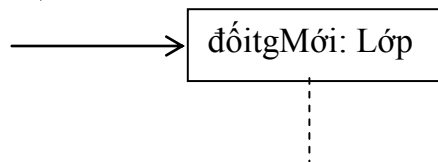


Hình 4.4 Ví dụ về các thông điệp dạng hàm và thủ tục, trong đó có loại tự kích hoạt

Trong sơ đồ *Hình 4.4*, các thông điệp số 2 và 3 là các lời gọi hàm, trong khi thông điệp số 1 là lời gọi thủ tục. Thông điệp số 4 để gửi trả về là một đối tượng tên « *may* » của lớp « *Máy_ATM* ». Thông điệp số 2 là tự kích hoạt (self-stimulus), tức được gọi và được nhận đều trên cùng một sinh tuyến, các thông điệp còn lại đều là stimulus.

4.3.2.3 Tạo và hủy đối tượng

- Tạo đối tượng (create):



Lưu ý :

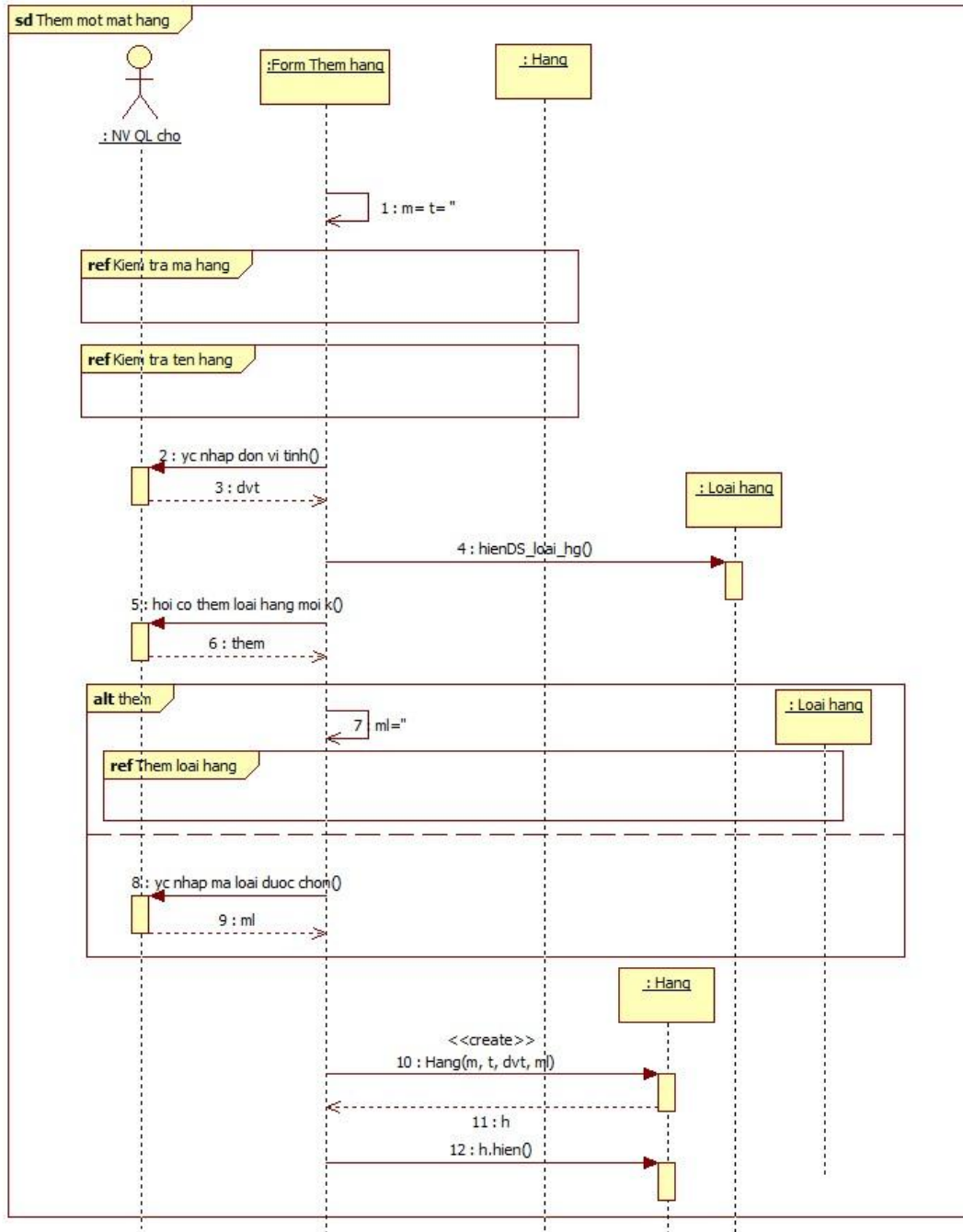
1. Khi tạo đối tượng mới, phải có sinh tuyến mới của lớp liên quan, cho dù trước đó đã có một sinh tuyến của lớp này cho các thông điệp khác.
2. Khi nhập tri vào cho từng thuộc tính, phải kiểm tra các ràng buộc toàn vẹn (RBTV) trên thuộc tính này ở cả 2 mức client (tại sinh tuyến biểu diễn các giao diện ngay tại người sử dụng) và server (gửi thông điệp đến các lớp liên quan). Cụ thể là :
 - a) *RBTV về trị null*: Hoàn toàn có thể ở mức client.
 - b) *RBTV về tính duy nhất*: Ở mức server, vì phải gọi phương thức đến lớp tương ứng để xem có đối tượng nào có trị trùng ở thuộc tính đang nhập hay không.
 - c) *RBTV về khóa*: kết hợp 2 kiểm tra RBTV trên.
 - d) *RBTV về luận lý*: Chẳng hạn, nếu kiểm tra xem trị phải thỏa các bất đẳng thức liên quan đến các biến hoặc hằng, chứ không liên quan đến đối tượng nào trong cơ sở dữ liệu, thì có thể kiểm tra tại client, tức tại sinh tuyến biểu diễn các giao diện ngay tại người sử dụng. Ngược lại, thì phải kiểm tra ở mức server để gọi phương thức lập sẵn, hoặc lấy trị thuộc tính của đối tượng liên quan ra mà so sánh theo điều kiện phải thỏa.
 - e) *RBTV về khóa ngoài (foreign key)/ tham chiếu (reference)* : Nếu lớp được tạo đối tượng là một lớp A tham chiếu (referencing class) đến một lớp B (referenced class), có thể B cũng được thêm đối tượng. Việc thêm này phải tiến hành trước khi lấy khóa ngoài/ tham chiếu cho đối tượng mới.

Ví dụ :

Lớp Hàng trong sơ đồ tuần tự *Hình 4.5* được thêm một đối tượng :

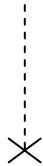
- Lớp Hàng có 2 thuộc tính « mã hàng » và « tên hàng » đều mang tính chất khóa sẽ được kiểm tra ở 2 mức client và server, do đó khung chữ nhật diễn tả hành vi « Kiểm tra mã hàng » và « Kiểm tra tên hàng » đều phải bao sinh tuyến của lớp Hàng.

- Lớp Hàng có tham chiếu đến lớp Loại hàng, nên nếu khi thêm một mặt hàng mới có loại hàng cũng chưa có trong danh mục thì cần thêm loại hàng đó trước khi lấy mã loại mới ra làm một tham số cho hàm tạo mặt hàng mới và sẽ đóng vai trò trị khóa ngoài cho đối tượng mới đó. scope



Hình 4.5 Ví dụ về tạo một đối tượng

- Hủy đối tượng (destroy):



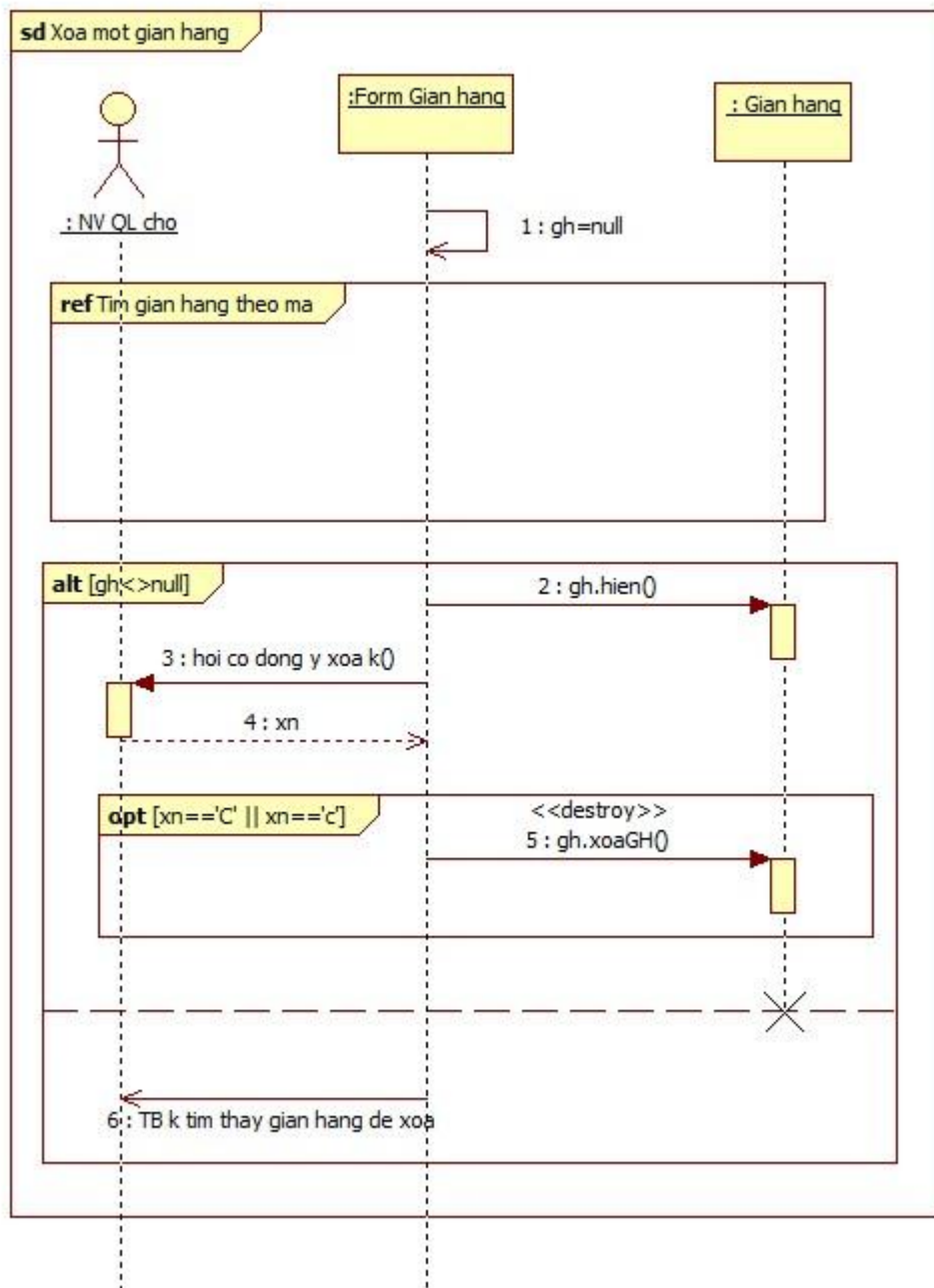
Lưu ý :

1. Khi hủy một đối tượng, phải kiểm tra xem có nó có tồn tại hay chưa.
2. Khi đã chắc chắn có đối tượng cần hủy, phải hiển thị các trị thuộc tính ra và phải yêu cầu xác nhận hủy của tác nhân. Nếu lớp B của đối tượng bị hủy là lớp được tham chiếu bởi một lớp A, phải xem lại ràng buộc toàn vẹn tham chiếu giữa hai lớp là khi xóa một đối tượng b của B có xóa kéo theo các đối tượng a liên quan trong A hay không. Do điều này cực kỳ quan trọng nên phải xác định cẩn thận từ khi lập sơ đồ lớp và lập các RBTV. (liên quan “On delete cascade, On update cascade” trong CSDL)
- 3.
4. Đôi khi, phải hiển thị ra hết các đối tượng a trước khi yêu cầu xác nhận hủy đối tượng b.

Ví dụ :

Trong Hình 4.6, đối tượng gh của lớp Gian hàng bị hủy.

- Trước tiên, phải tìm gh.
- Nếu tìm được gh, thì hiển thị đối tượng gh ra.
- Trong sơ đồ lớp ở mức quan niệm, có liên kết giữa lớp Gian hàng và lớp Hàng, cho biết một gian hàng có thể bán nhiều mặt hàng. Như vậy, ở mức luận lý, sẽ phát sinh lớp Bán hàng chứa khóa ngoài/ tham chiếu đến lớp Gian hàng. Tuy nhiên, do không đưa ra ràng buộc xóa một đối tượng của Gian hàng thì xóa kéo theo các đối tượng của lớp Bán hàng, nên chỉ hủy gh khi được người dùng xác nhận.
- Nếu không tìm được gh, thông điệp cho người dùng biết.

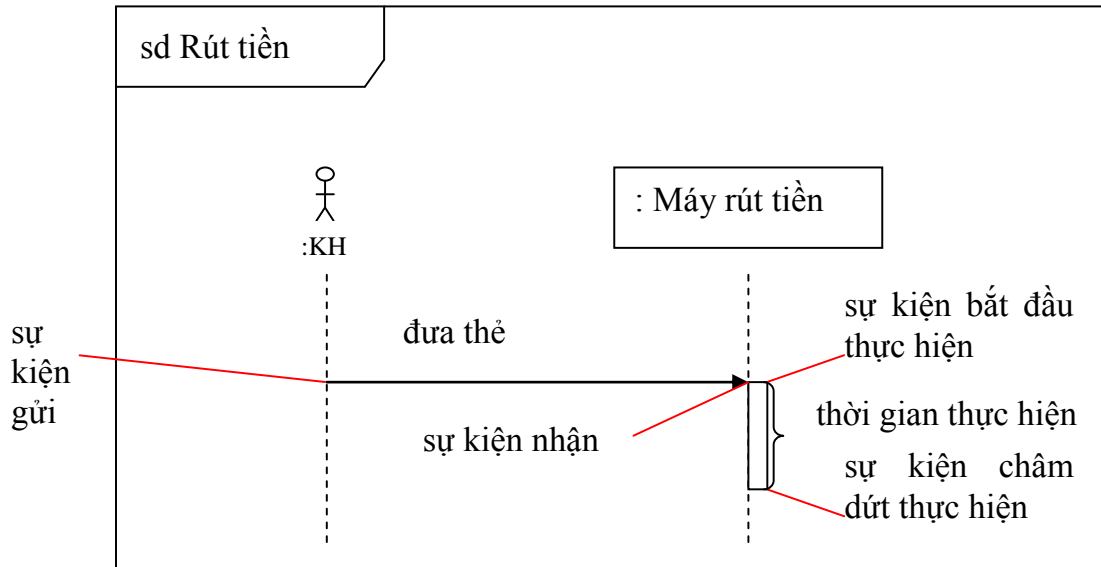


Hình 4.6 Ví dụ về hủy một đối tượng

4.3.3 Thông điệp và sự kiện

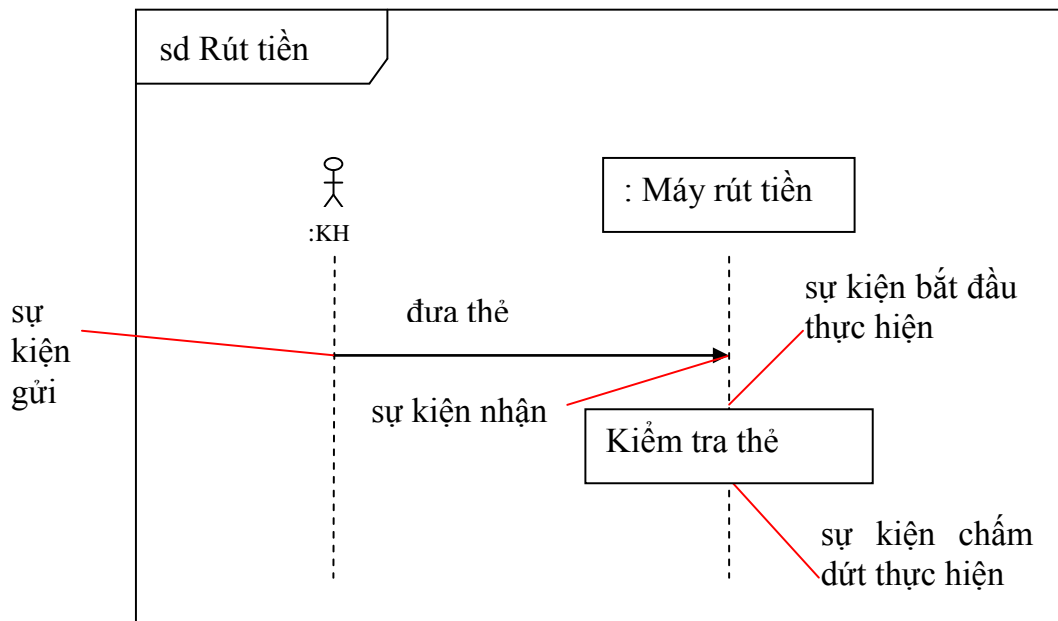
UML phân biệt việc gửi với nhận một thông điệp, cũng như phân biệt việc bắt đầu với việc kết thúc thực hiện một phản ứng. Các sự kiện (event) được dùng để đánh dấu từng giai đoạn.

Ví dụ :



Hình 4.7 Ví dụ về sự kiện bắt đầu thực hiện ngay khi có sự kiện nhận

Hoặc :



Hình 4.8 Ví dụ về sự kiện bắt đầu thực hiện trễ hơn sự kiện nhận

Dựa vào các sự kiện gửi và nhận, UML định nghĩa 3 kiểu thông điệp :

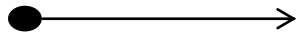
- Thông điệp đầy đủ : các sự kiện gửi và nhận đều được biết.
- Thông điệp bị lạc (lost message): sự kiện gửi có được biết, nhưng sự kiện nhận thì không.

Ký hiệu :



- Thông điệp tìm thấy (found message): ngược lại, sự kiện nhận có được biết, nhưng sự kiện gửi thì không.

Ký hiệu :



4.3.4 Ngữ pháp của thông điệp

4.3.4.1 Thông điệp yêu cầu, hoặc kích hoạt phương thức

Thông điệp các loại này có ngữ pháp dưới đây :

<tên tín hiệu hoặc tên phương thức> [(<danh sách tham số>)]

Trong danh sách, các tham số cách nhau bởi dấu phẩy, và mỗi tham số có ngữ pháp như sau :

[<tên tham số> [<dấu>] <trị của tham số>]

với :

<dấu> là dấu '=' nếu đó là tham số chỉ đọc,

dấu ':' nếu đó là tham số có thể sửa được (đọc/ ghi)

Lưu ý, tham số khi dùng phải có trị trước đó :

- Hoặc được nhập vào từ tác nhân
- Hoặc được gán
- Hoặc được tính toán

Phạm vi của biến có liên quan rất nhiều đến tham số và đối tượng thực hiện thông điệp.

Ví dụ :

nhậpMã(« 1234 »)

nhậpMã(m)

ghiNhận(maSV, maSach)

sửaSolg(sl : 10)

4.3.4.2 Thông điệp trả lời (return)

Thông điệp trả lời có ngữ pháp dưới đây :

[<tên biến> =] <thông điệp yêu cầu hoặc kích hoạt> [: <trị trả về>]

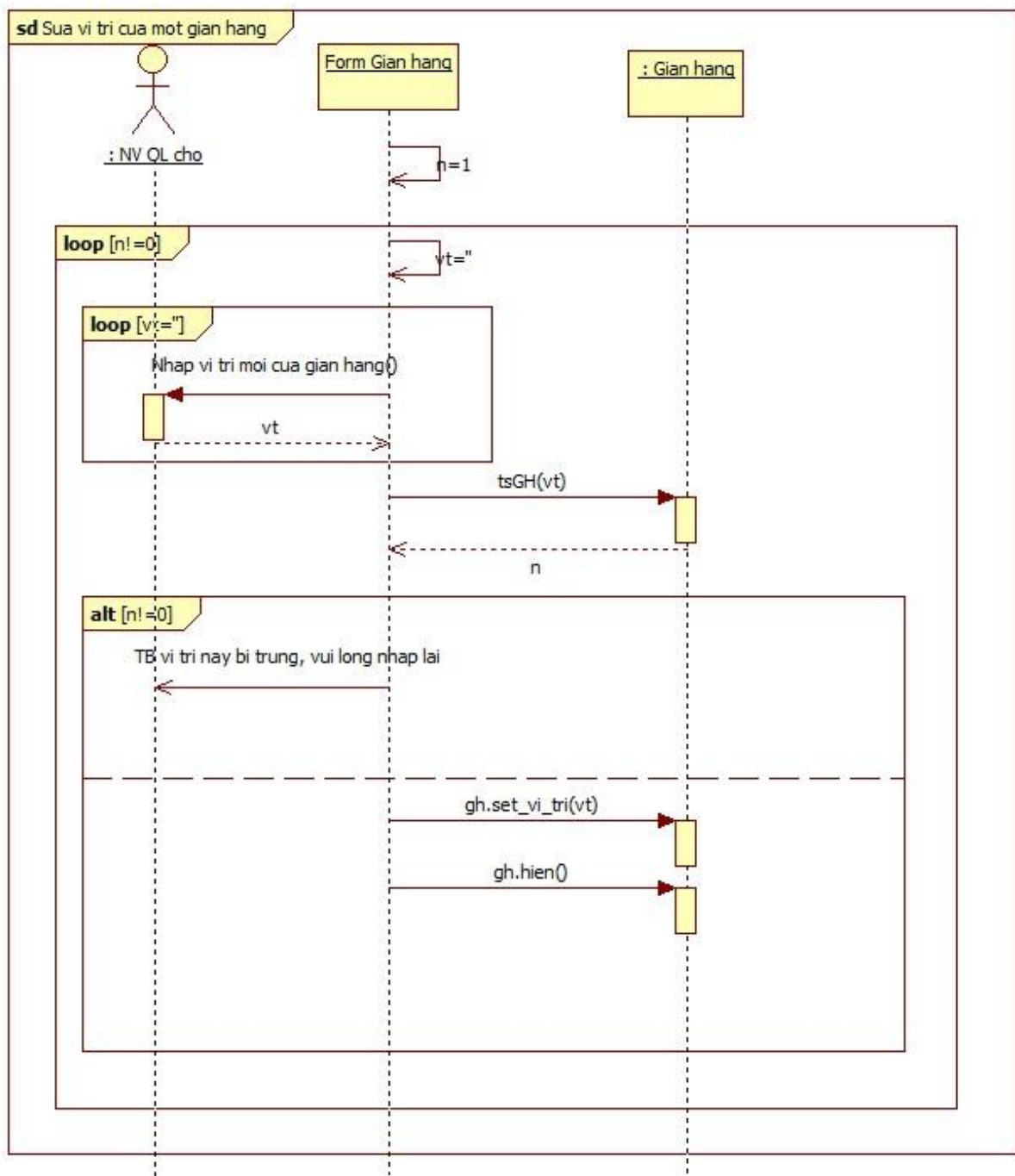
Ví dụ :

sốlgTồn= ktraHgTồn(« A001 »)

maOK=ktraMa(ma) : true

Tuy nhiên, cũng có thể đặt tên biến nhận trị trả về ngay trên thông điệp gửi (khi đó mũi tên trả về không cần có tên biến), hoặc chỉ đặt tên biến mà không cần tên thông điệp gửi.

Ví dụ :



Hình 4.9 Ví dụ về ngữ pháp của thông điệp

Trong hình 4.9, có :

- các lệnh gán $n=1$, $vt=""$
- thông điệp yêu cầu nhập vị trí mới của gian hàng
- thông điệp trả lời bằng trị của biến vt

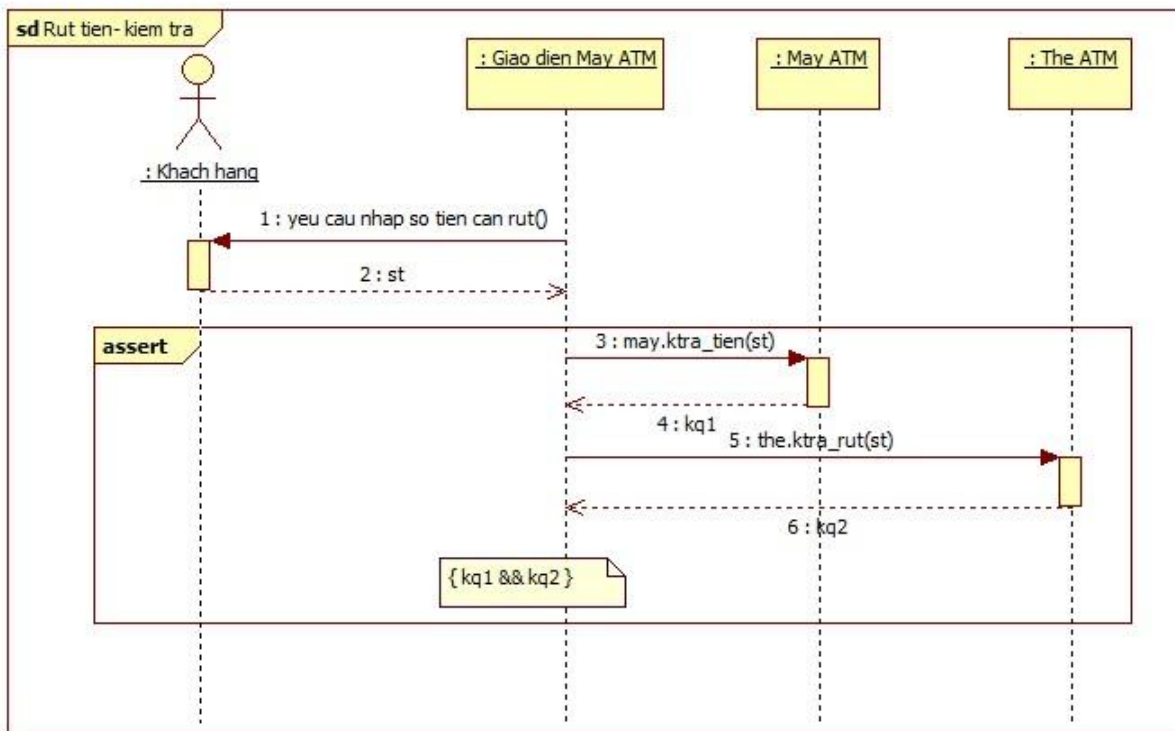
- thông điệp gửi cho biết vị trí bị trùng, yêu cầu nhập lại
- thông điệp gọi hàm tsGH để tính tổng số gian hàng với tham số vt và nhận thông điệp trả lời qua biến n
- các thông điệp gọi các set method « set_vi_tri » và « hien » từ đối tượng gh.

4.3.5 Ràng buộc trên các sinh tuyến

Các sinh tuyến của một tương tác có thể mang ràng buộc.

Ký hiệu:

- Một ràng buộc được biểu diễn trên một sinh tuyến bằng một văn bản giữa dấu móc ({ }).
- Một ràng buộc cũng có thể được chứa trong một ghi chú gắn với thể hiện của sự kiện liên quan.



Hình 4.10 Ví dụ về ràng buộc trên sinh tuyến và cách dùng « assert »

Phép *assert* khẳng định sự cần thiết của việc gửi thông điệp tiếp theo.

Như vậy, để rút được tiền ở một máy rút tiền, bắt buộc phải kiểm tra số tiền mặt hiện có trong máy và trong thẻ; sau khi kiểm tra, các biến logic kq1 và kq2 phải mang trị true, tức là máy và thẻ ATM của khách hàng đều có đủ số tiền cho rút (Hình 4.10). Biến

đối tượng « thể » đã được tìm ra ở sd trước đó (đã gọi sd đang xét), có phạm vi biến (scope) phủ luôn sd này, nhận trị từ ngoài truyền vào.

Ghi chú :

Một ràng buộc được đánh giá khi khai thác tương tác. Nếu ràng buộc không được thỏa mãn, các thể hiện của sự kiện đi theo sau ràng buộc này sẽ được xem là không hợp lệ, ngược lại với khi ràng buộc được thỏa mãn. Như vậy, một sơ đồ tương tác có thể mô tả các thông điệp không hợp lệ, nghĩa là không bao giờ được gửi đi.

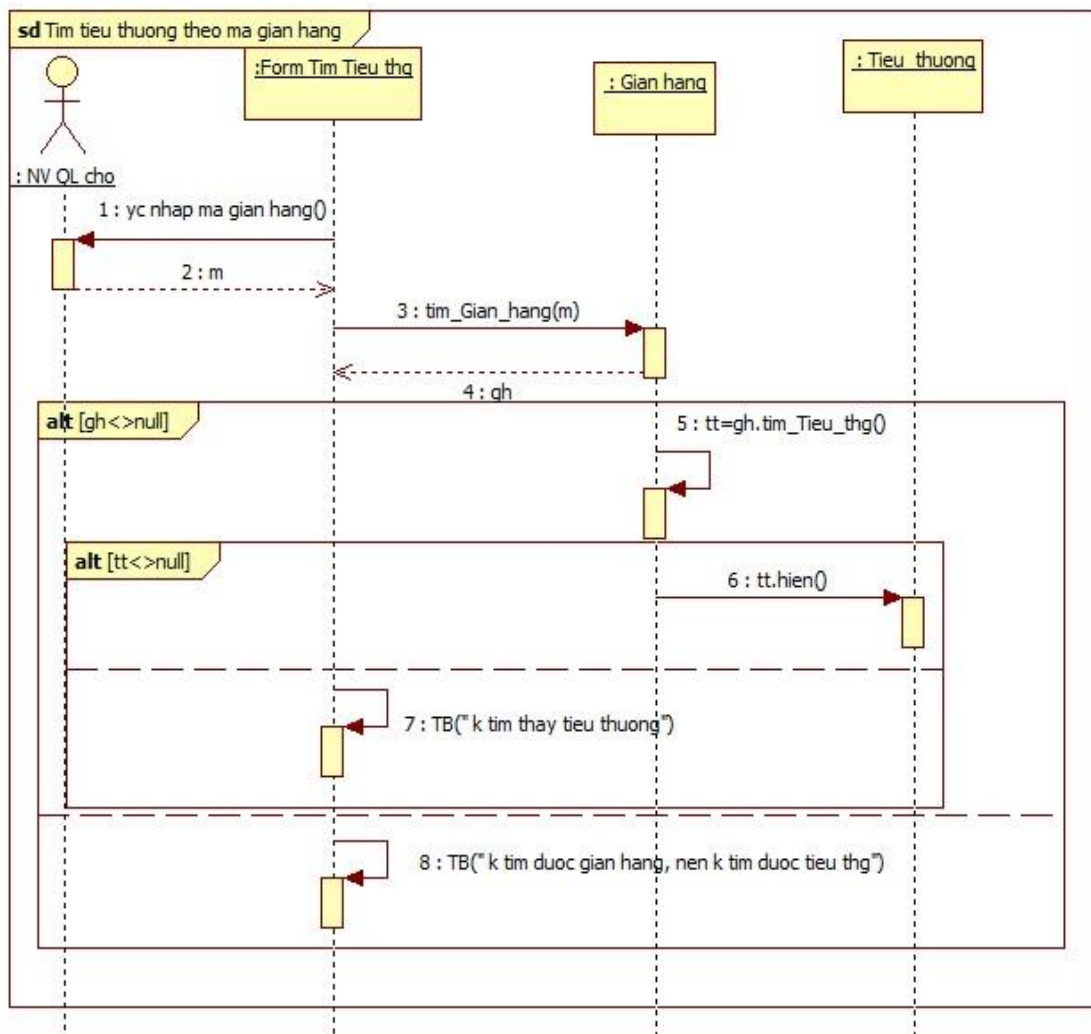
4.3.6 Các kiểu phân đoạn của tương tác

4.3.6.1 Rẽ nhánh

Có 2 từ khóa là :

- « alt » (alternative) và « else »: rẽ nhánh 1 nếu điều kiện đúng và rẽ nhánh 2 nếu ngược lại.
- « opt » (optional) : thực hiện hành vi tiếp theo nếu điều kiện đúng.

Ví dụ 1: Sử dụng « alt »



Hình 4.11 Ví dụ về rẽ nhánh dùng « alt »

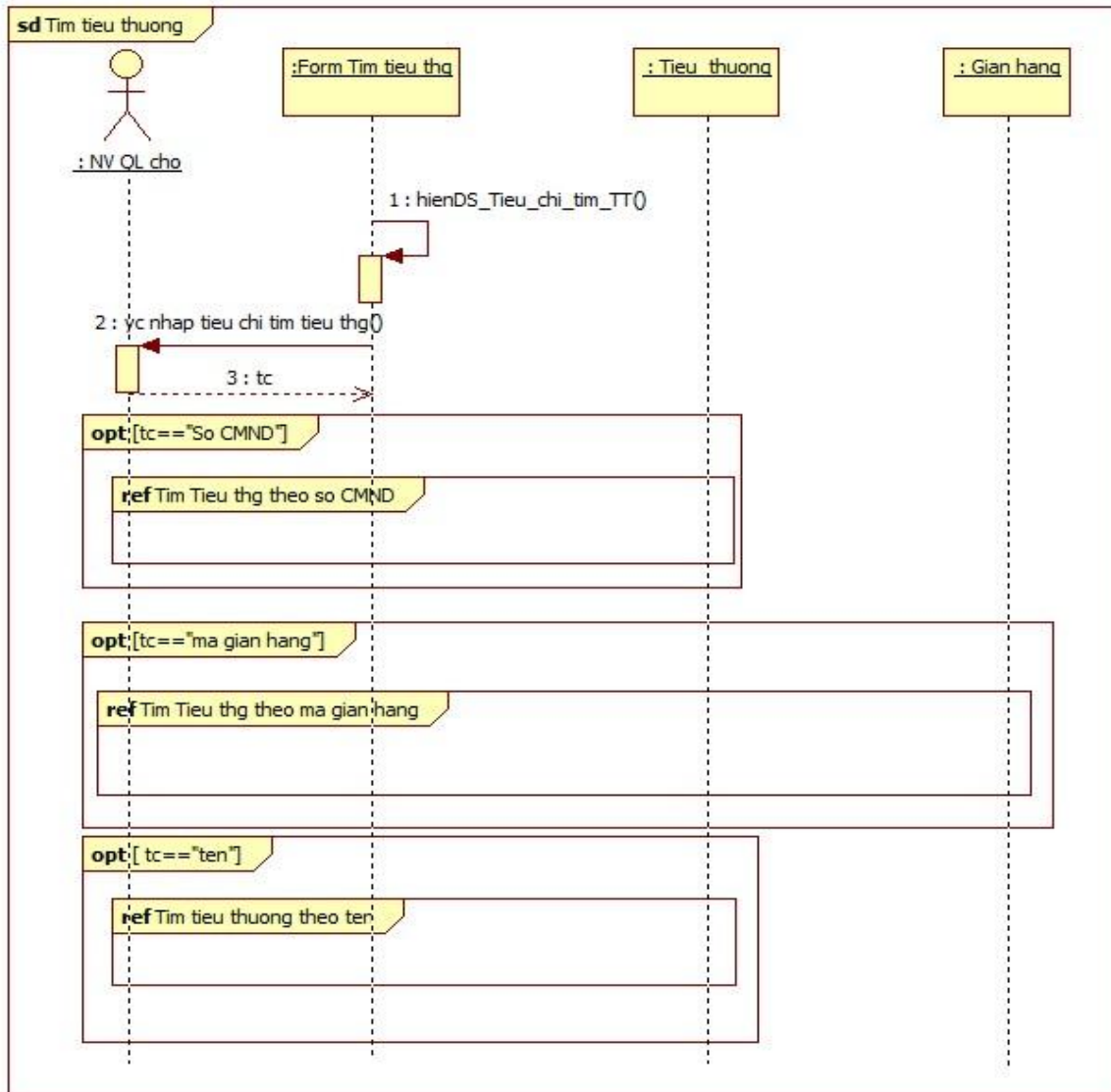
Hình 4.11 giới thiệu 2 phân đoạn rẽ nhánh lồng nhau. Trước tiên, giao diện «Form Tìm Tiêu thg» yêu cầu nhân viên quản lý chợ nhập mã gian hàng. Với biến m được nhập vào như trị của mã, giao diện gọi hàm « tìm_Gian_hàng(m) » để tìm đối tượng gh của lớp « Gian hàng ».

Phân đoạn rẽ nhánh đầu tiên dùng điều kiện [gh < > null] : nếu đúng, tức có tồn tại đối tượng gh thực sự, sẽ dùng hàm tự kích hoạt «tim_Tieu_thuong» trên sinh tuyến của lớp « Gian hàng », nhận về trị của biến tt.

Phân đoạn rẽ nhánh thứ hai dùng điều kiện [tt < > null] : Nếu đúng, thực sự có đối tượng tt của lớp «Tiêu thương», sẽ gọi thủ tục « hiện » nhờ tt thực thi để hiển thị thông tin của chính đối tượng tt. Nếu sai, sẽ thông báo không tìm thấy tiêu thương nào đang thuê gian hàng đã nói qua thông điệp « TB » dạng thủ tục có tham số dạng chuỗi.

Nếu điều kiện [gh < > null] sai, sẽ thông báo không tìm được gian hàng nên không tìm được tiêu thương, cũng dùng « TB ».

Ví dụ 2: Sử dụng « opt »



Hình 4.12 Ví dụ về rẽ nhánh dùng « opt »

Trong hình 4.12, « opt » được sử dụng để rẽ nhiều nhánh công việc (« ref » để gọi thực hiện sd khác) tùy theo giá trị nhập vào biến tc từ nhân viên quản lý chợ. Biến tc cho biết tiêu chí để tìm kiếm tiểu thương.

4.3.6.2 Vòng lặp (while [< đk >], repeat ... until, for)

Sử dụng 2 từ khóa :

- « loop » : có 3 trường hợp :
 - Loop : lặp mãi đến khi có lệnh « break ». Vòng lặp sẽ có ít nhất một bước lặp. Điều quan trọng với dạng lặp này là phải tránh lặp vô tận, nên ở mỗi bước lặp, phải có biến nào đó thay đổi trị được, và biến này sẽ nằm trong

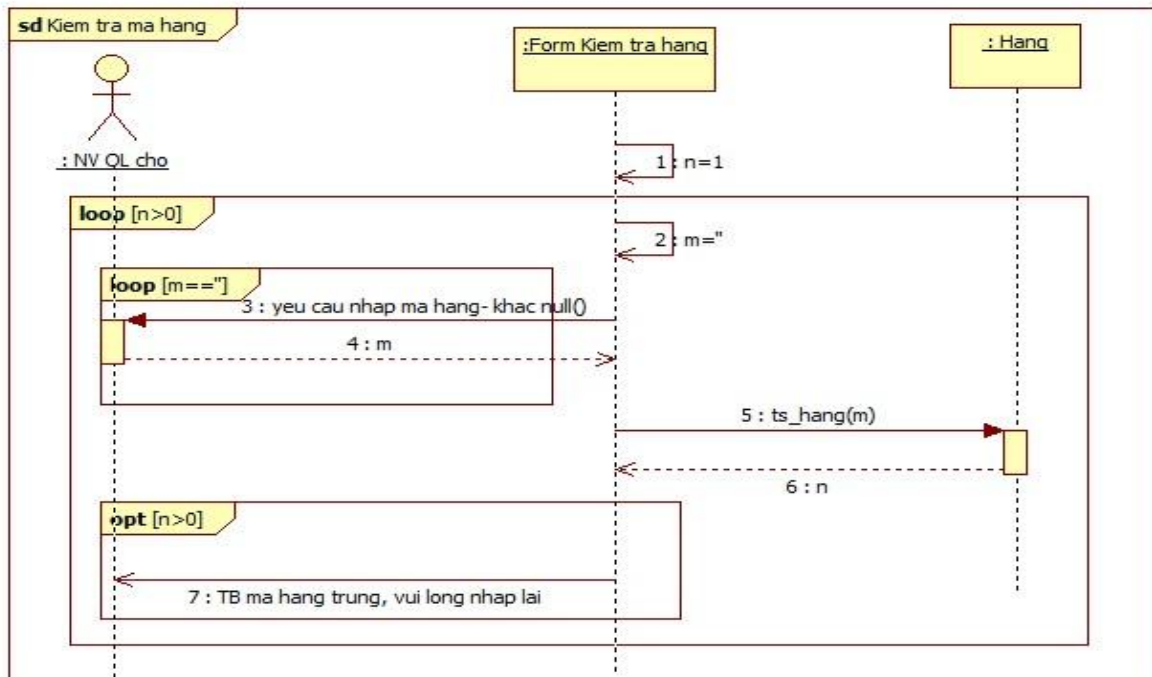
một biểu thức điều kiện để nếu điều kiện này thỏa, sẽ thực hiện lệnh « break ».

- Loop <điều kiện> : còn lặp lại khi điều kiện còn thỏa. Điều kiện được kiểm tra trước tiên nên có thể không qua được bước lặp nào. Cũng như dạng lặp trên, ở mỗi bước lặp, điều kiện có thể được thay đổi trị để có lúc vòng lặp phải được dừng.
- Loop [<chỉ số min>, <chỉ số max>] : thường có dạng loop [1, n], lặp lại các bước lặp theo chỉ số nhận trị từ <chỉ số min> đến <chỉ số max>.

Lưu ý : các biến trong điều kiện và các chỉ số phải được gán trị trước khi đặc tả loop.

- « break » : để thoát ra khỏi vòng lặp.

Ví dụ 1 : Sử dụng vòng lặp dùng « loop » (**Hình 4.13**)



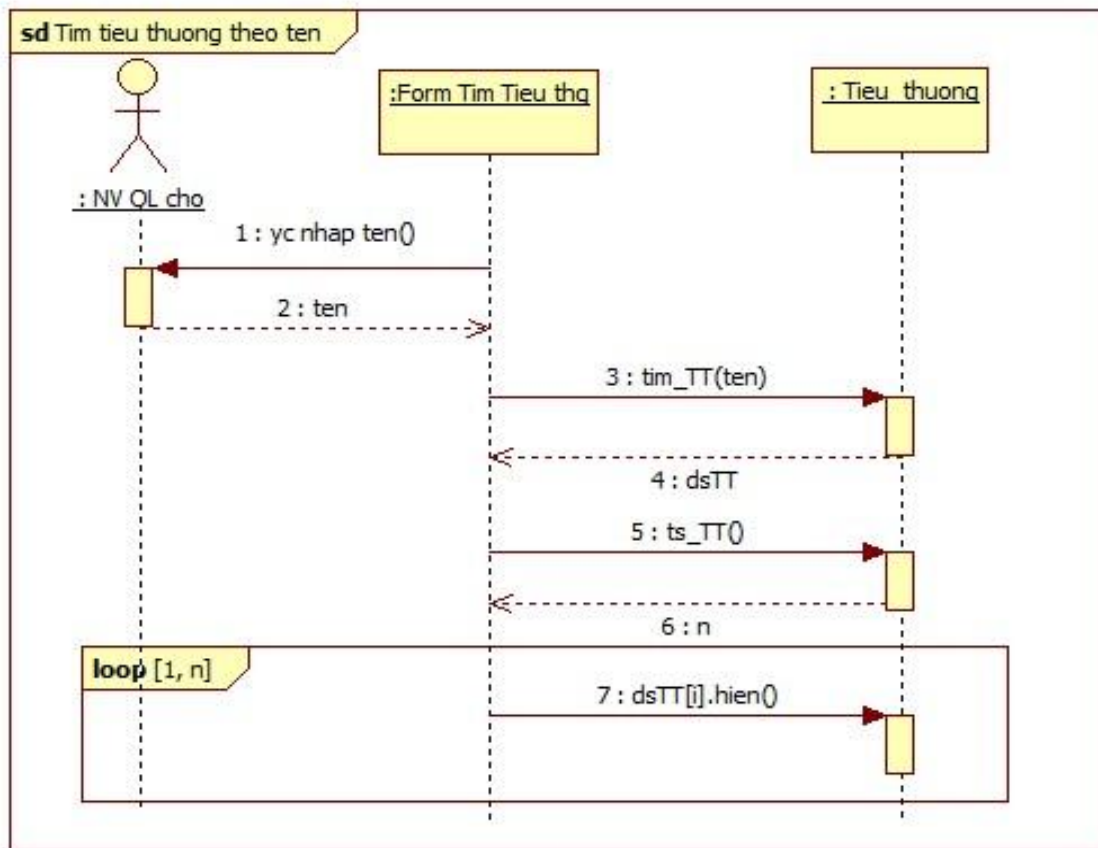
Hình 4.13 Ví dụ về vòng lặp dùng « loop » có điều kiện

Lớp Hàng có mã hàng làm khóa chính nên khi muốn chấp nhận một giá trị m cho mã để tạo một đối tượng mới cho lớp này, phải kiểm tra tính khác rỗng và tính duy nhất của m. Trong **Hình 4.13**, 2 vòng lặp được dùng cho việc kiểm tra này.

Vòng lặp ngoài với điều kiện [n>0] để đảm bảo tính duy nhất. n có ý nghĩa tổng số mặt hàng đã có mã m. Để nhập được trị đầu tiên cho m, phải thỏa điều kiện « n>0 », nên n trước đó phải được gán một trị dương, ở đây là 1. Nếu n>0, tức đã có mặt hàng dùng mã m này rồi, sẽ phải thông báo đến tác nhân và yêu cầu lặp lại động tác nhập trị cho mã m để họ nhập lại ở bước lặp tiếp theo của vòng lặp ngoài.

Vòng lặp trong với điều kiện [m==''] để đảm bảo tính khác rỗng. Bên trong vòng lặp ngoài, cứ trước khi vào vòng lặp trong này, m lại được gán lại để thỏa được điều kiện đó.

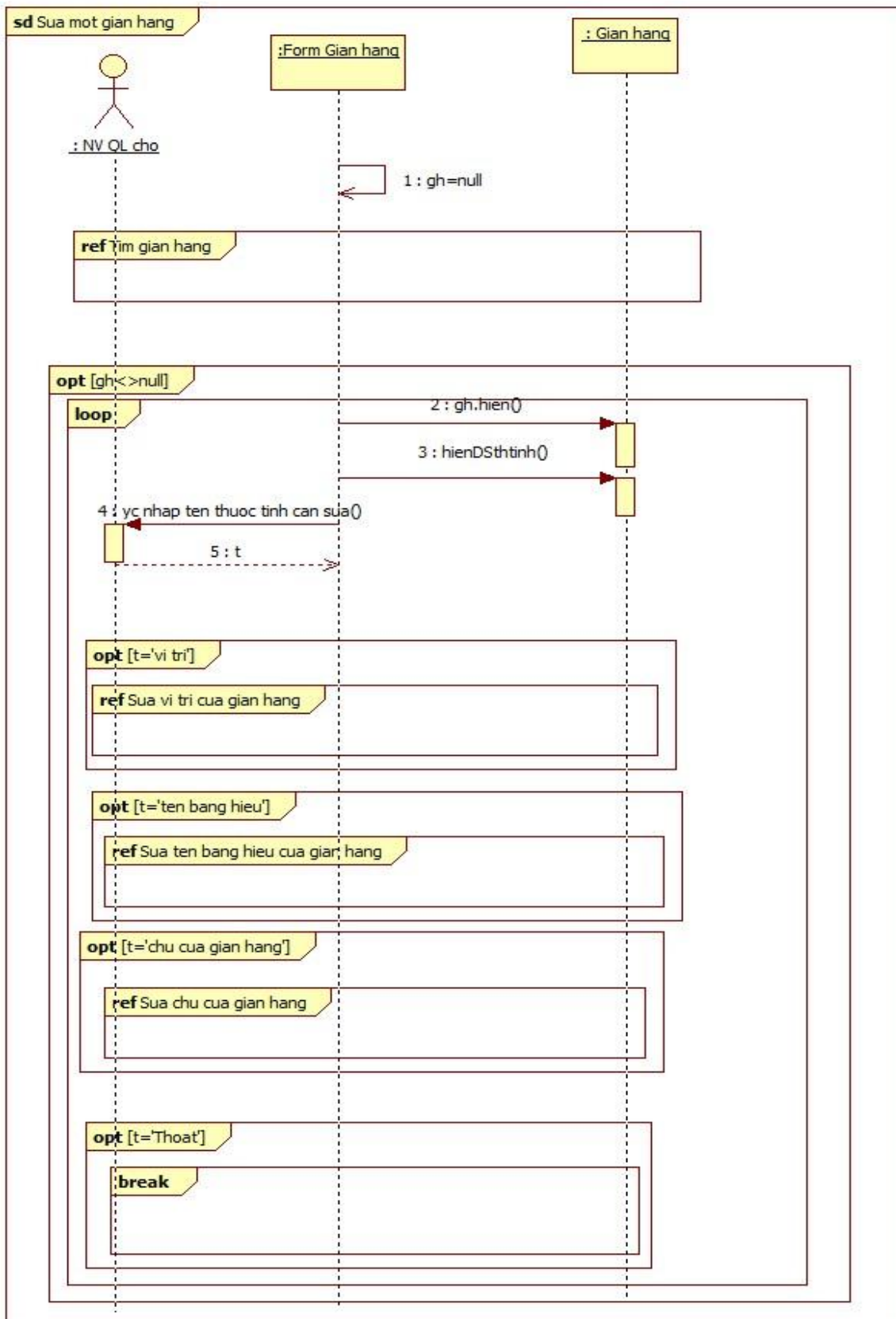
Ví dụ 2 : Sử dụng « loop » có chỉ số lặp (*Hình 4.14*)



Hình 4.14 Ví dụ về vòng lặp dùng « loop » có chỉ số lặp

Do nhiều tiểu thương có thể cùng tên nên khi có một biến « tên » để tìm tiểu thương, ta có thể có một danh sách chứ không chỉ một tiểu thương, nên phải dùng kiểu mảng cho kết quả dsTT của hàm « tìm_TT(ten) ». Muốn xem được chi tiết từng đối tượng Tiểu_thương trong dsTT, ta sẽ gọi thủ tục « hiện » của lớp Tiểu_thương đó từ mỗi phần tử thứ i của mảng dsTT trong vòng lặp có chỉ số lặp i.

Ví dụ 3 : Sử dụng « break » (*Hình 4.15*)



Hình 4.15 Ví dụ về vòng lặp dùng « loop » không điều kiện với « break »

Khi sửa một đối tượng, trước tiên, ta phải tìm chính xác đối tượng đó; nếu tìm được, sẽ phải cho đối tượng đó hiển thị ra. Hơn nữa, khi sd sửa được gọi, ta đã muốn sửa ở ít nhất một thuộc tính. Vì vậy, ta dùng vòng lặp dạng không điều kiện để chắc chắn rằng thực hiện trước các thao tác này.

Cứ mỗi lần sửa xong một thuộc tính, người dùng có thể tiếp tục sửa mà cũng có thể dừng. Bắt buộc phải có một điều kiện [t = 'Thoát'] để lệnh break được dùng để dừng vòng lặp.

4.3.7 Phân rã một sinh tuyến

Đôi khi, một tương tác quá phức tạp để có thể được mô tả trong một sơ đồ, nên ta có thể cắt một sinh tuyến ra trên nhiều sơ đồ. Một phần của sinh tuyến được phân rã ra sẽ được thay thế bằng một hình chữ nhật với từ khóa *ref* (reference), và phải được trình bày ở sơ đồ khác.

Dùng ref có các lợi ích sau:

- Dễ trình bày ý tưởng thiết kế tổng thể
- Các chi tiết được trình bày ở sơ đồ có qui mô nhỏ hơn, do đó dễ kiểm soát lỗi hơn
- Một sơ đồ được thiết kế chỉ một lần và có thể dùng nhiều lần mà không cần lặp lại chi tiết của nó ở nhiều sơ đồ khác nhau.
- Mỗi sơ đồ được tham chiếu có thể xem như tương ứng với một chương trình con ở mức vật lý.

Lưu ý:

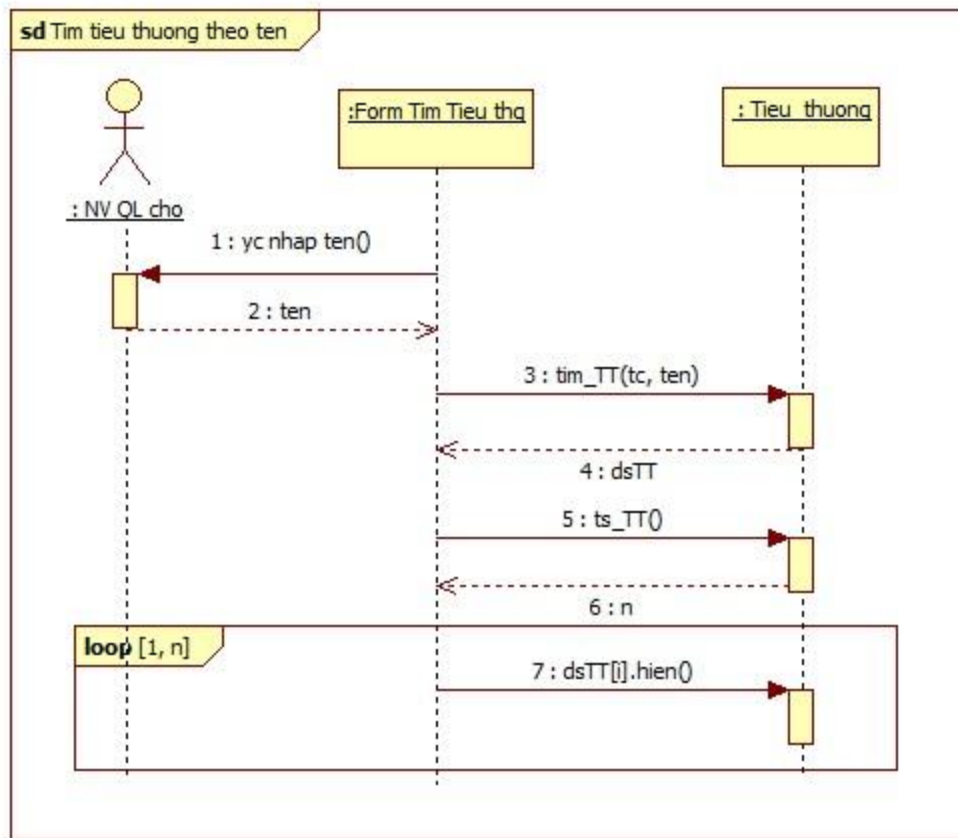
Khi dùng ref nhớ lưu ý đến phạm vi của các biến trong sơ đồ A chứa ref và sơ đồ B được tham chiếu đến :

- a) Để một biến V của B có thể sử dụng ở A sau khi tham chiếu đến B, V phải được khai báo/ khởi tạo/ dùng trong A trước khi tham chiếu đến B.
- b) Biến cục bộ trong B không thể dùng ở A sau khi A tham chiếu đến B.

Ví dụ:

Ở Hình 4.12, biến « tc » được dùng trong sơ đồ « Tìm tiểu thương » và sơ đồ này tham chiếu đến sơ đồ « Tìm tiểu thương theo tên ». Ở Hình 4.16 bên dưới, biến « tc » còn được tiếp tục dùng trong sơ đồ « Tìm tiểu thương theo tên ».

Ngược lại, biến « tên » dùng trong sơ đồ « Tìm tiểu thương theo tên » ở Hình 4.16 nếu dùng ra bên ngoài ở Hình 4.12 sau khi tham chiếu « Tìm tiểu thương theo tên » sẽ được xem là không hợp lệ.

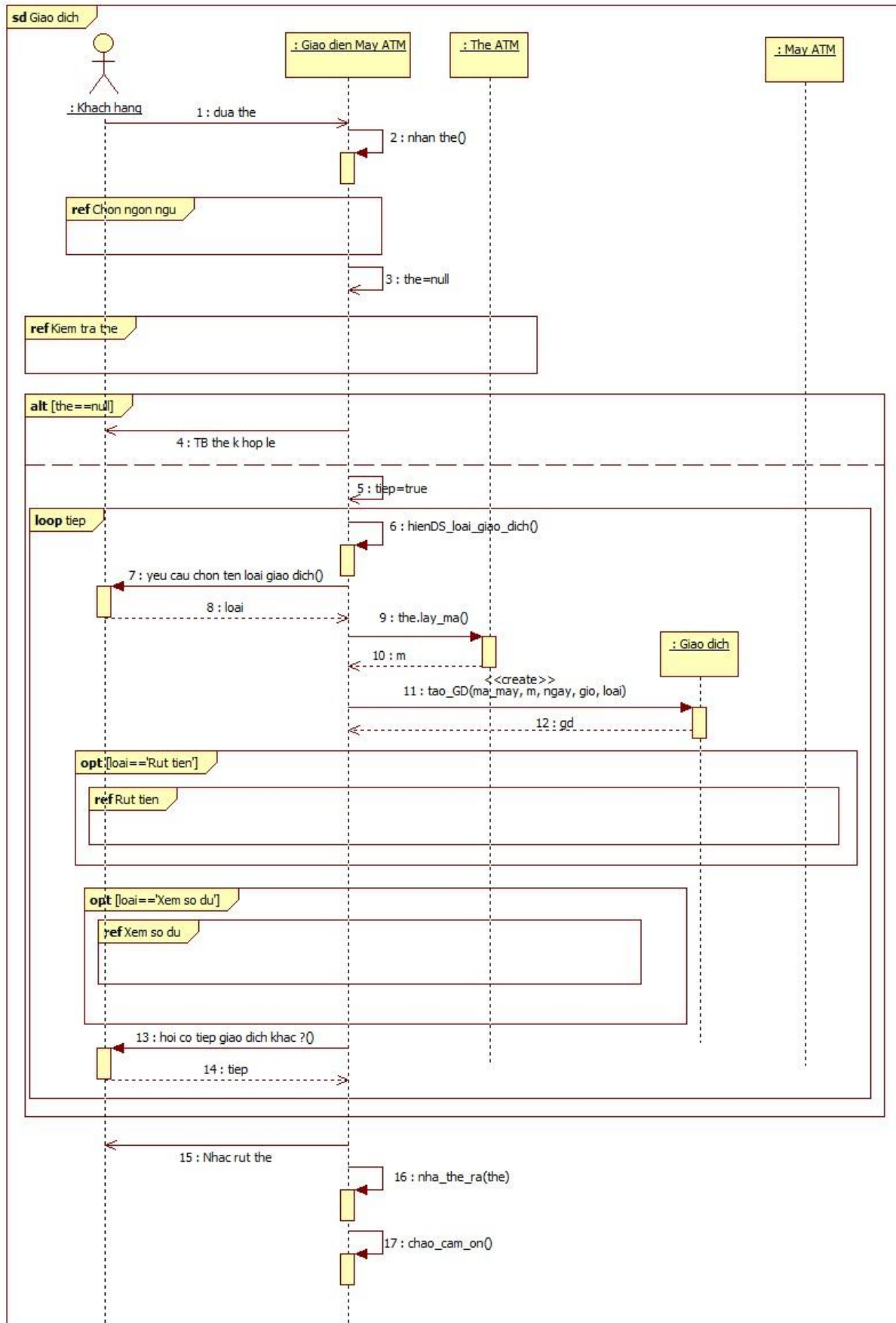


Hình 4.16 Ví dụ về phạm vi của biến

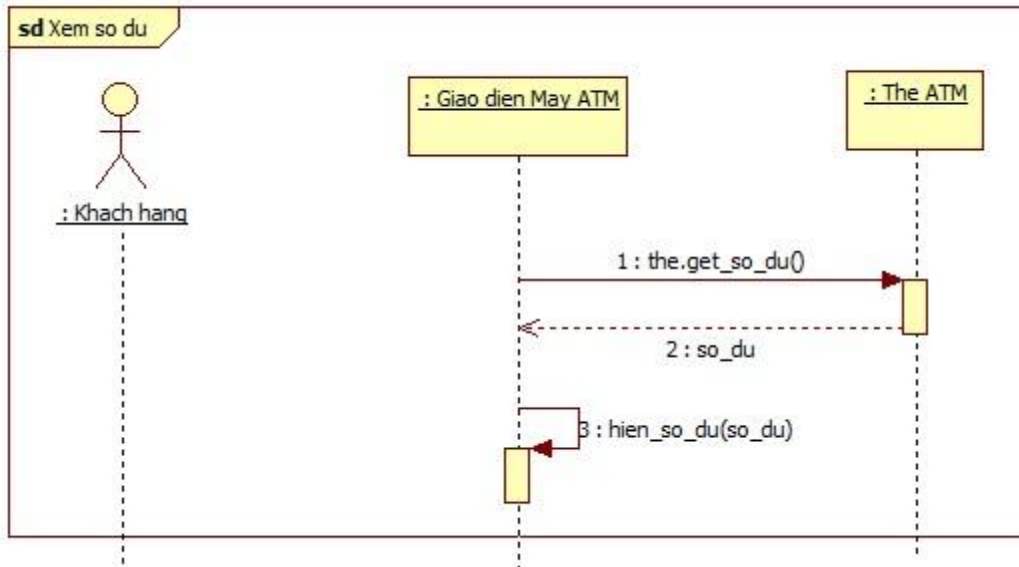
Hình 4.17 có khá nhiều sd được tham chiếu (ref) từ sd Giao dịch (Trong đó, có sd « Xem số dư » được mô tả trong hình 4.18). Các sd này chia làm 2 loại :

- Một số bắt buộc phải làm, không cần điều kiện : « Chọn ngôn ngữ », « Kiểm tra thẻ »,
- Số còn lại được thực hiện tùy theo loại giao dịch được chọn từ danh sách hiển thị ra trên giao diện. Ở đây, để đơn giản, chỉ cho ví dụ các sd « Rút tiền », « Xem số dư ».

Lưu ý về phạm vi các biến đối tượng « thẻ » và « gd ». Biến « thẻ » phải được khởi tạo trước khi gọi sd «Kiểm tra thẻ » để sau khi quay lại từ sd này thì nó mới hợp lệ cho việc sử dụng tiếp theo. Tương tự cho biến « gd » phải được khởi tạo trước khi gọi các sd « Rút tiền » hoặc « Xem số dư ». Các sd này và phương thức « nhả_thẻ_ra(thẻ) » ở phần còn lại của « Giao dịch » dùng được giá trị của « gd » và « thẻ » do phạm vi các biến đó phủ trên toàn sd « Giao dịch ».



Hình 4.17 Ví dụ về một sơ đồ tuần tự dùng « ref » đến sơ đồ khác

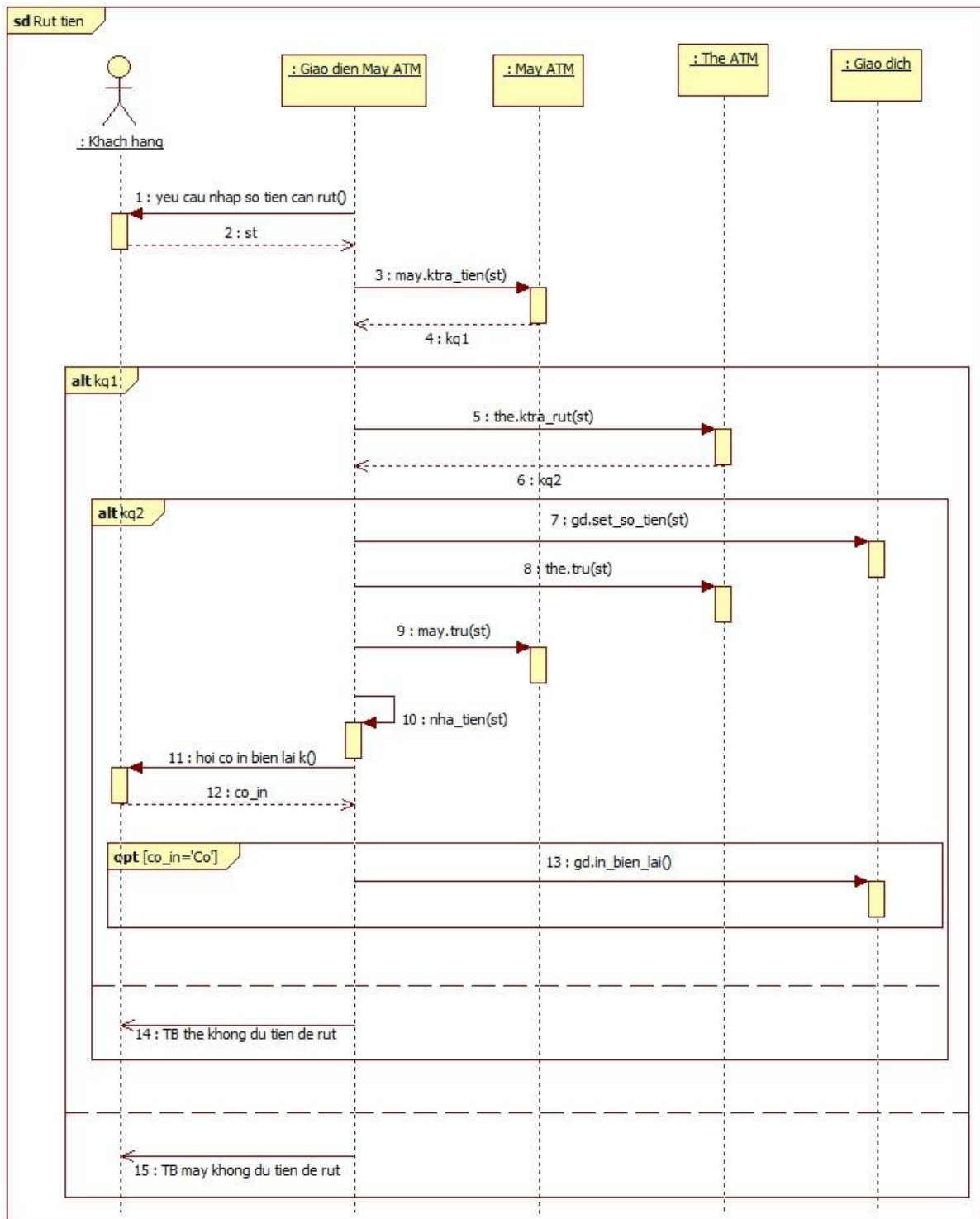


Hình 4.18 Ví dụ về một sơ đồ đơn giản được tham chiếu đến bởi « ref »

Hình 4.18 trình bày sd « Xem số dư », trong đó tiếp nhận đối tượng « thẻ » của lớp « Thẻ ATM » đã được kiểm tra trong sd « Kiểm tra thẻ ».

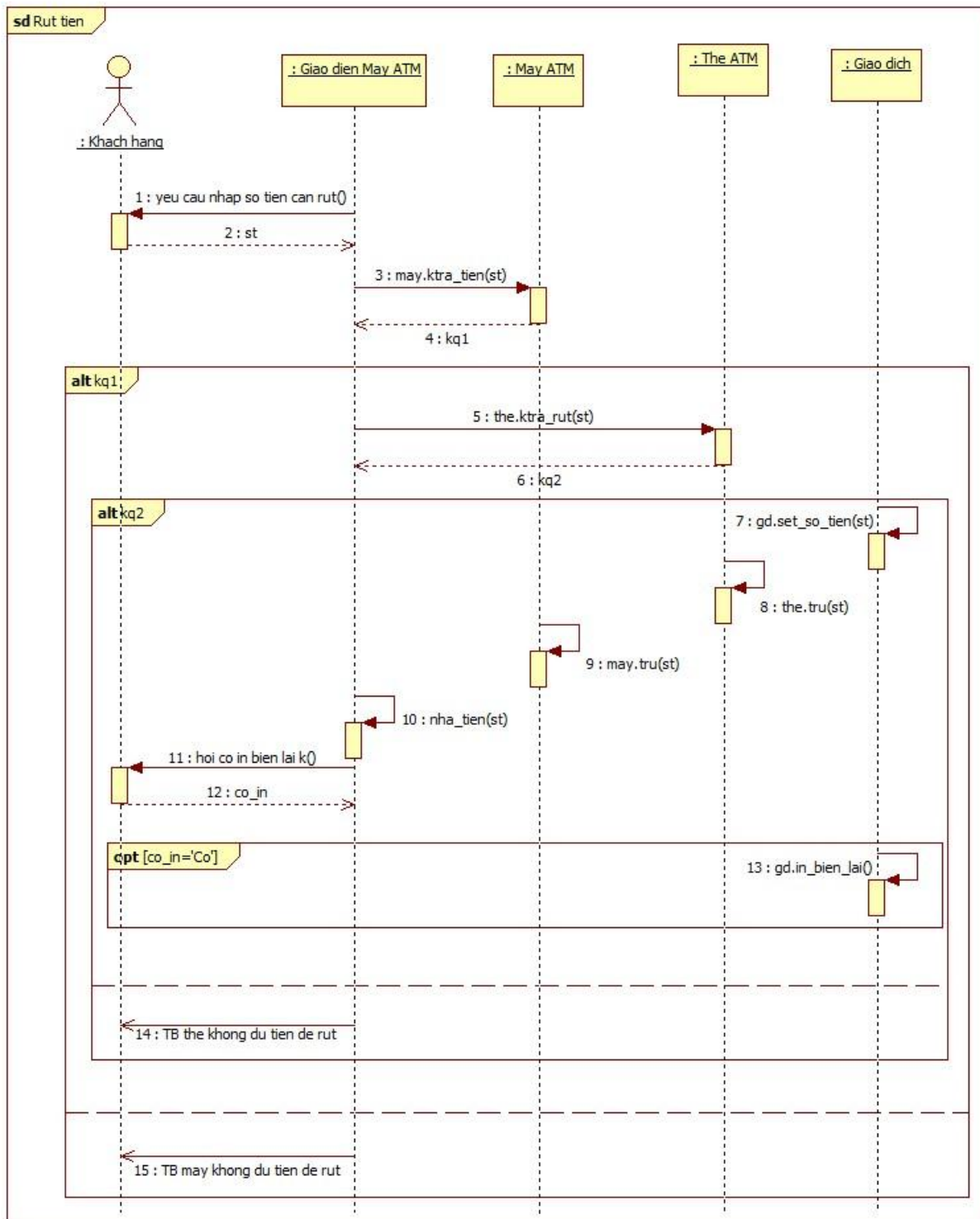
Biến « so_du » chỉ phát sinh bên trong sd « Xem số dư » nên có tính chất cục bộ, sau khi được dùng làm tham số của phương thức « hien_so_du » của lớp « Giao diện Máy ATM » thì khi thoát ra khỏi « Xem số dư » sẽ không còn được nhận biết nữa.

Hình 4.19 cho thấy rõ hơn phạm vi các biến « gd » và « thẻ ».



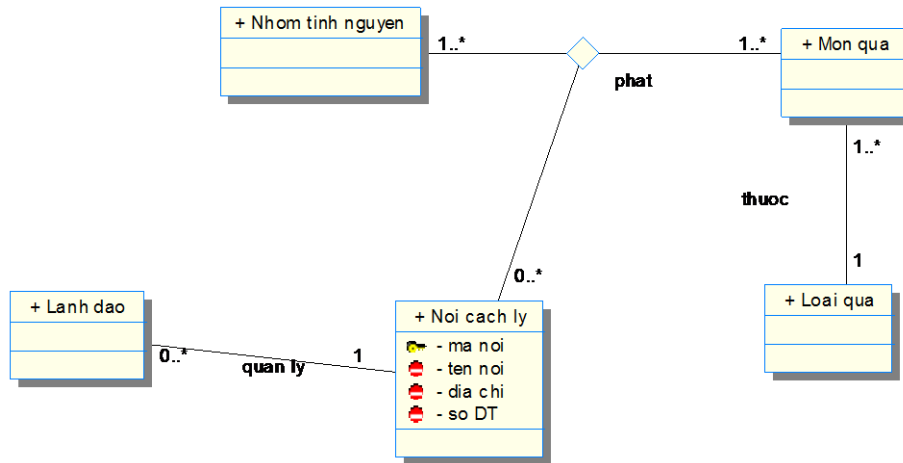
Hình 4.19 Ví dụ về một sơ đồ phức tạp được tham chiếu đến bởi « ref »

Sơ đồ trên *Hình 4.19* có thể được tinh chế hơn để giảm bớt thời gian truyền dữ liệu giữa client và server và giúp giao diện tại máy ATM trở thành “thin client” nhờ ít chứa lời gọi phương thức hơn, như hình 4.20 dưới đây:

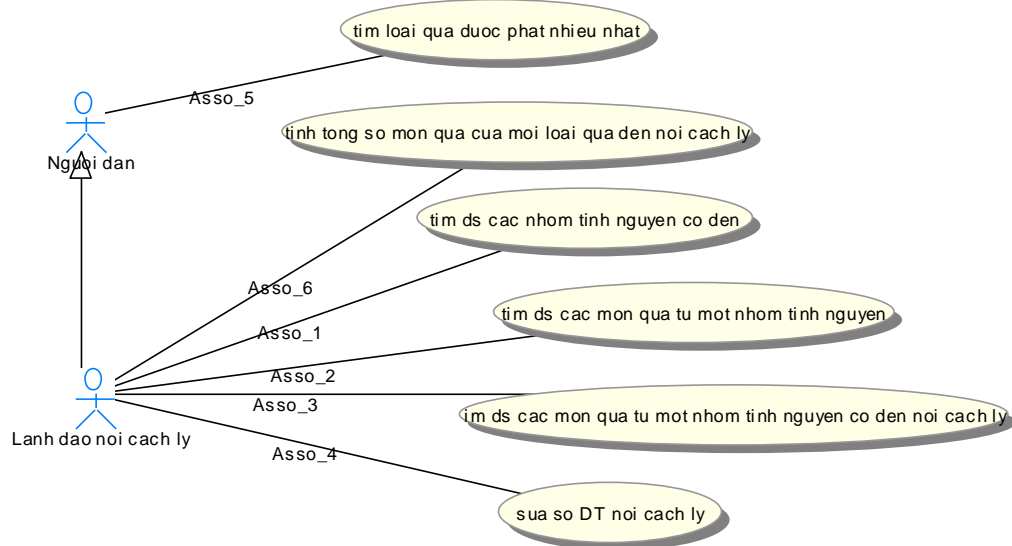


Hình 4.20 Ví dụ ở hình 4.19 được tinh chế

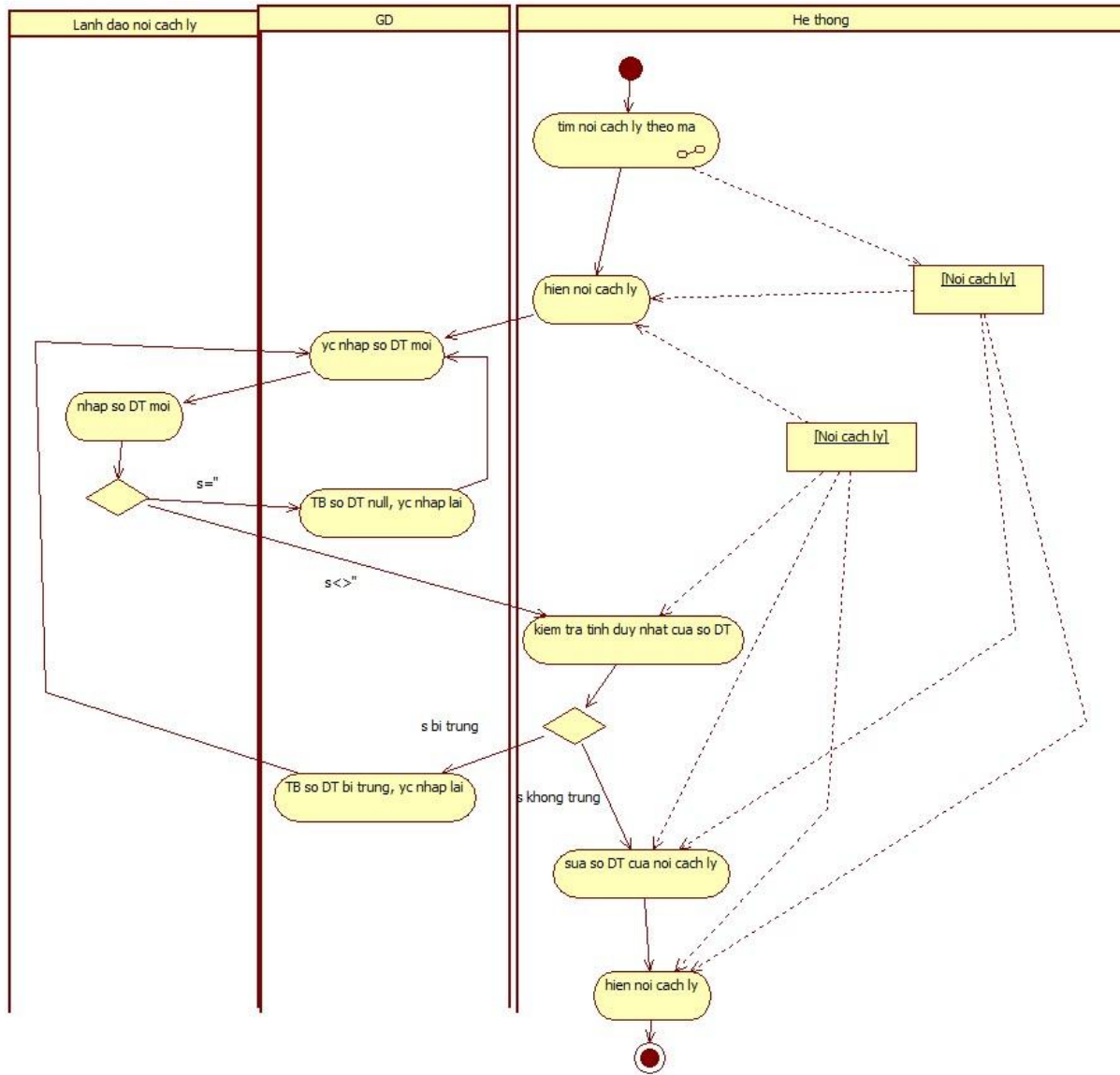
Vd Phát quà Trung tru cho nơi cách ly
Sơ đồ lớp:



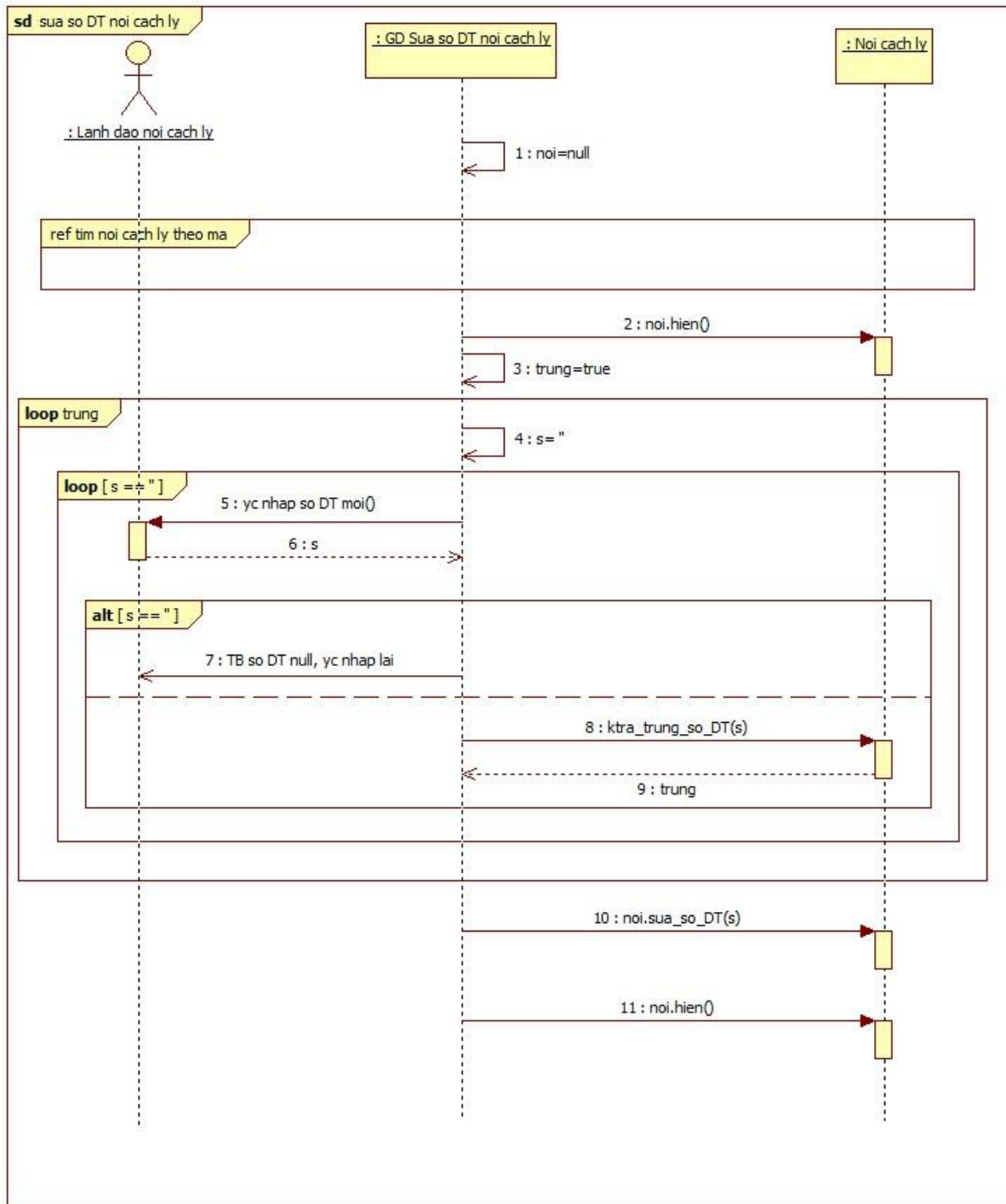
SĐHV cho tác nhân Lãnh đạo nơi cách ly:



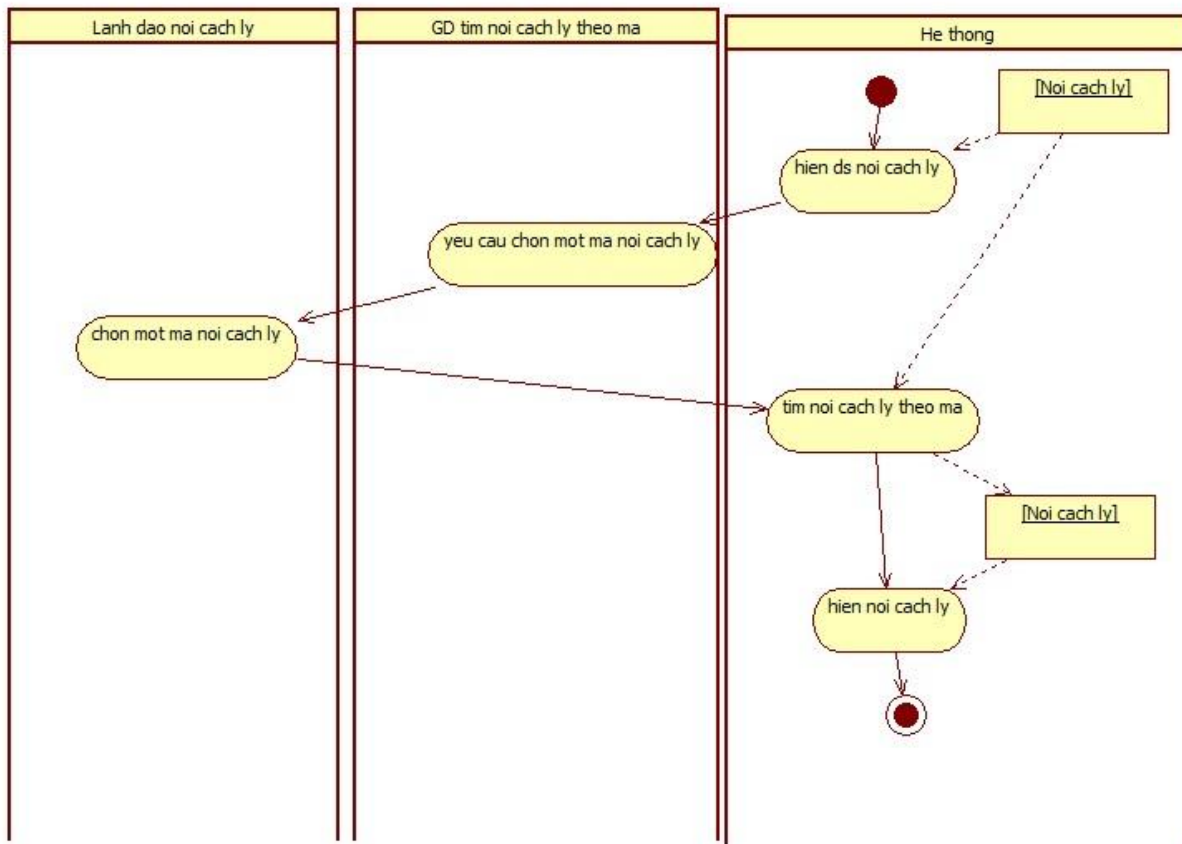
SĐHD Sửa số ĐT nơi cách ly:



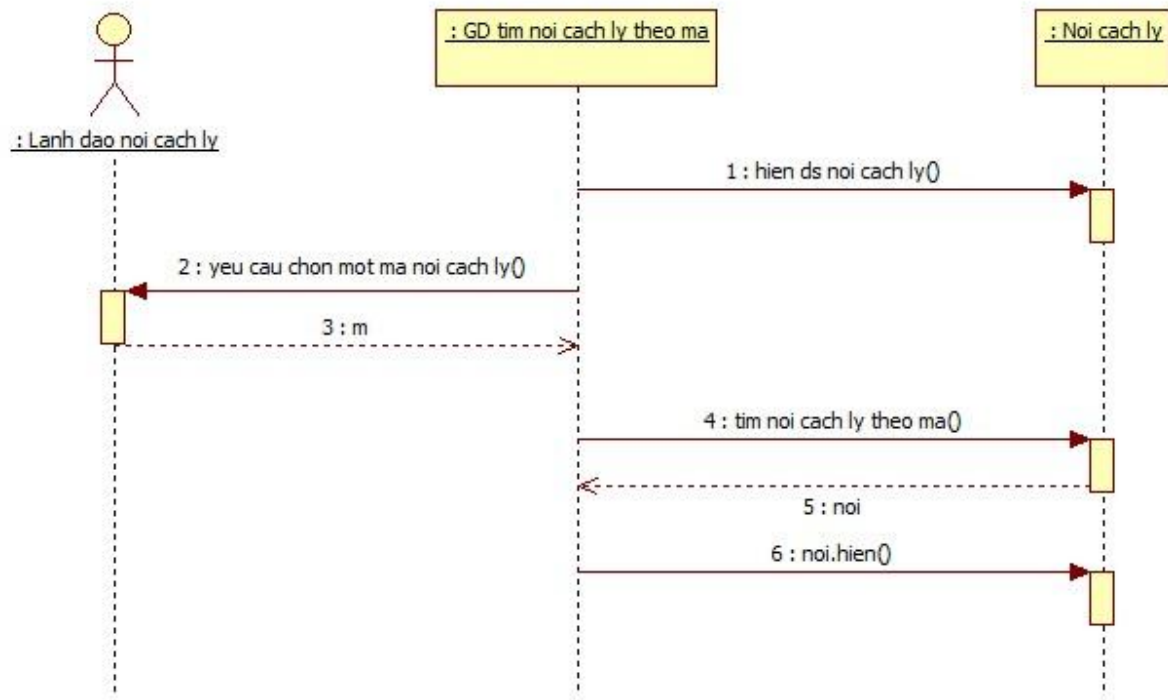
SĐTT Sửa số ĐT nơi cách ly:



SDHD Tìm nơi cách ly theo mã:



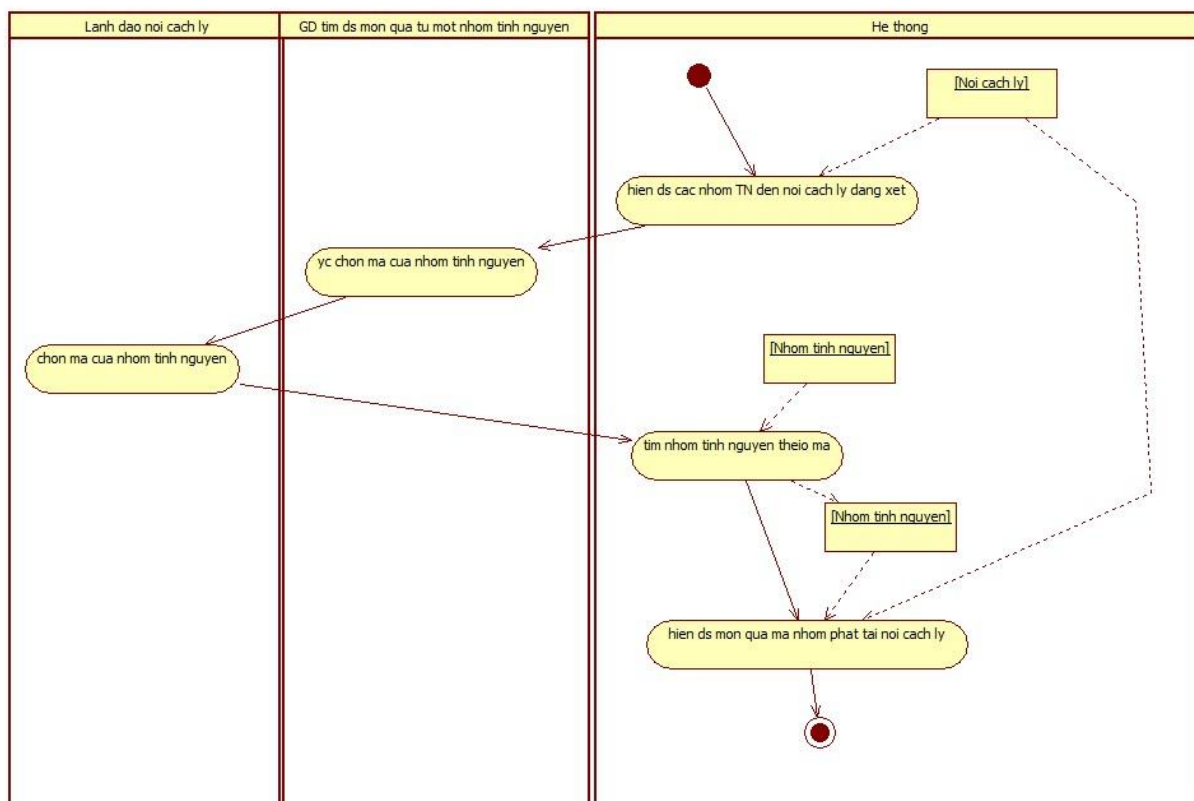
SĐTT Tìm nơi cách ly theo mã:



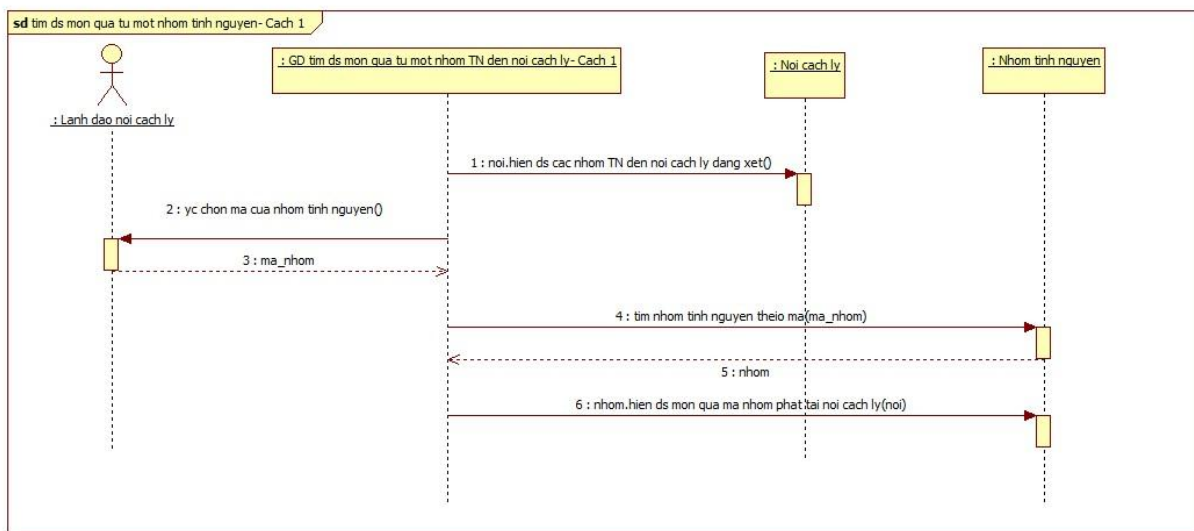
Bây giờ, ta muốn tìm danh sách các món quà từ một nhóm tình nguyện có đến nơi cách ly. Có 2 cách để làm.

- Cách 1: tìm nhóm tình nguyện theo mã và dùng phương thức thành viên từ đối tượng *nhom* vừa tìm được, và sử dụng đối tượng *noi* như tham số.
- Cách 2: Không cần tìm đối tượng của nhóm tình nguyện, mà chỉ dùng mã nhóm như tham số của phương thức thành viên từ đối tượng *noi* của lớp *Nơi cách ly* (đã tìm được trước đó).

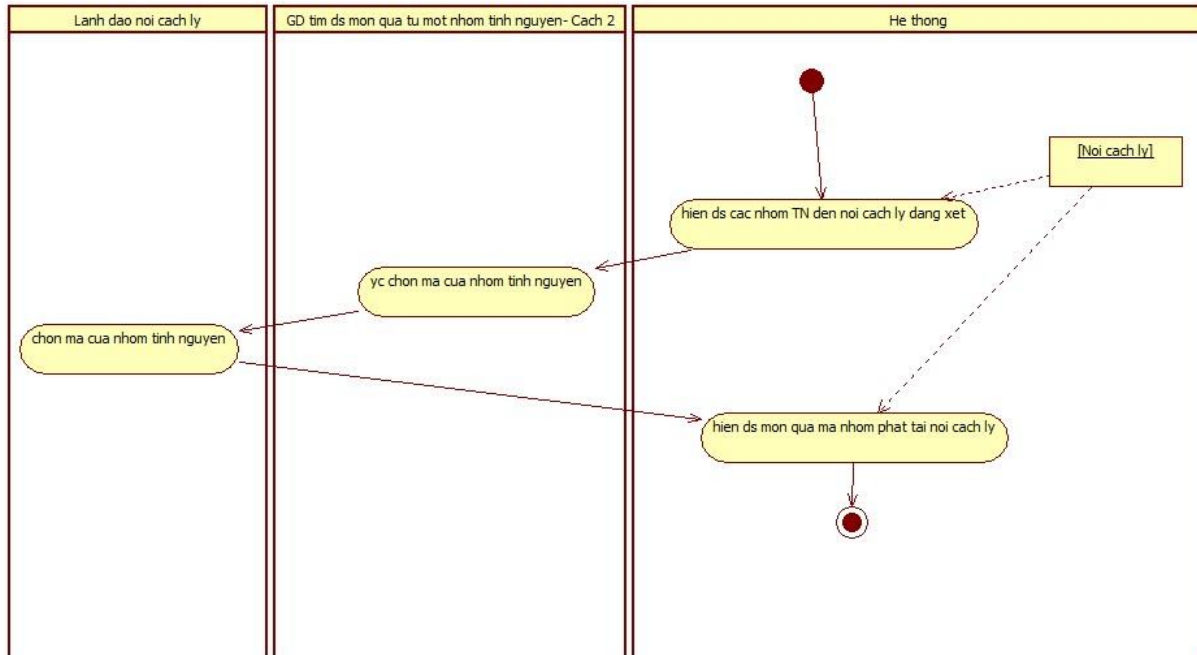
SĐHD theo cách 1:



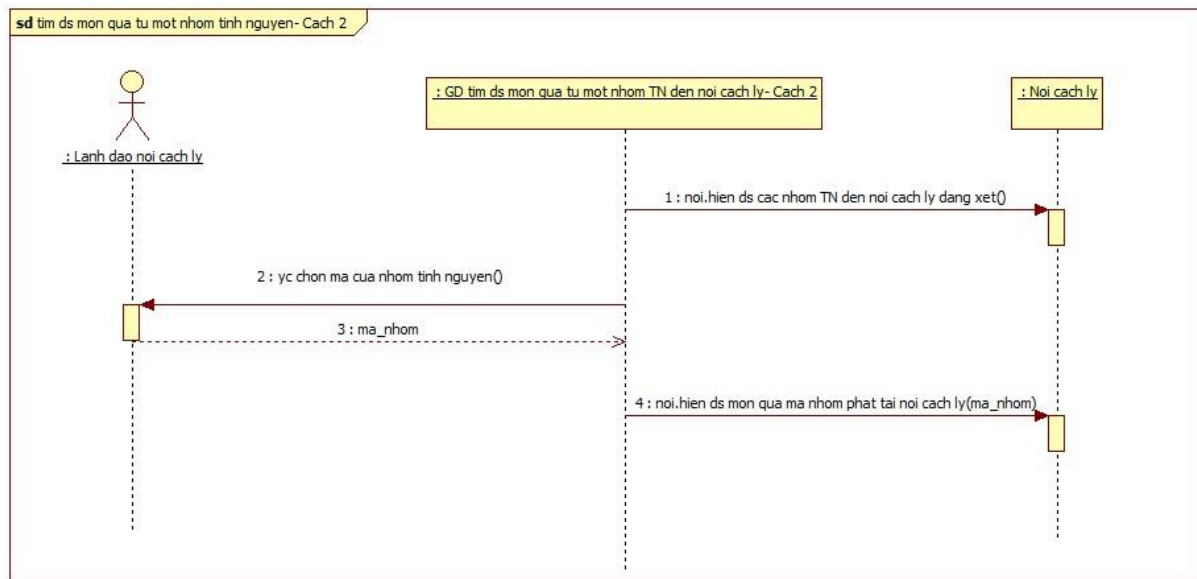
SĐTT theo cách 1:



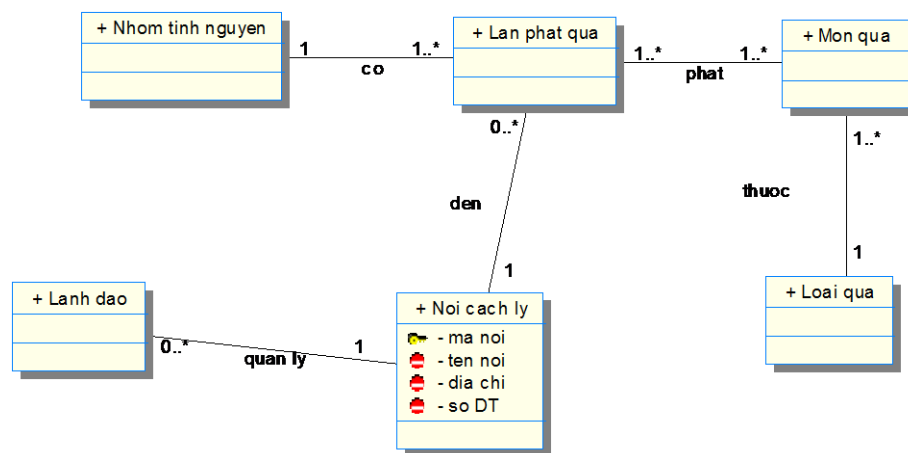
SĐHĐ theo cách 2:



SDTT theo cách 2:



Nếu sơ đồ lớp vẽ theo cách khác:

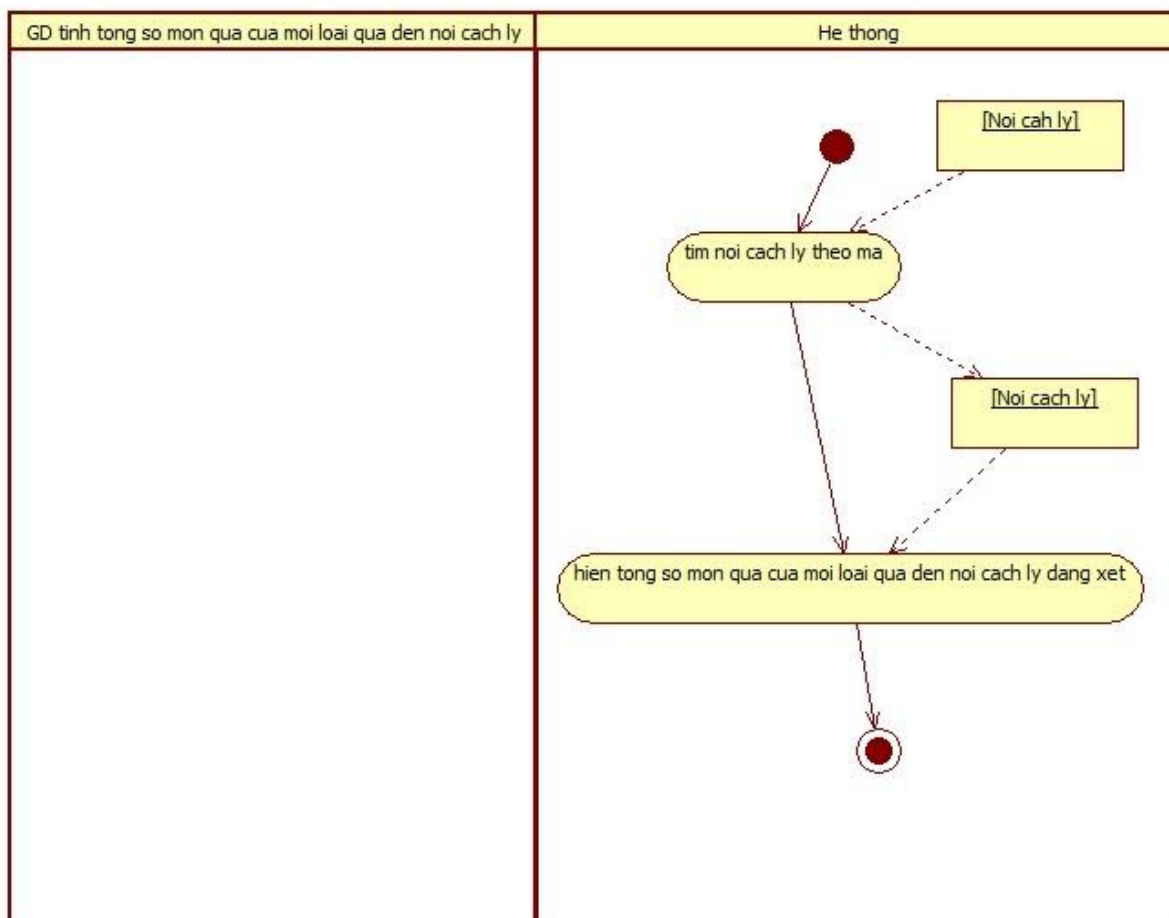


Thì sẽ có các SĐHĐ và SĐTT khác.

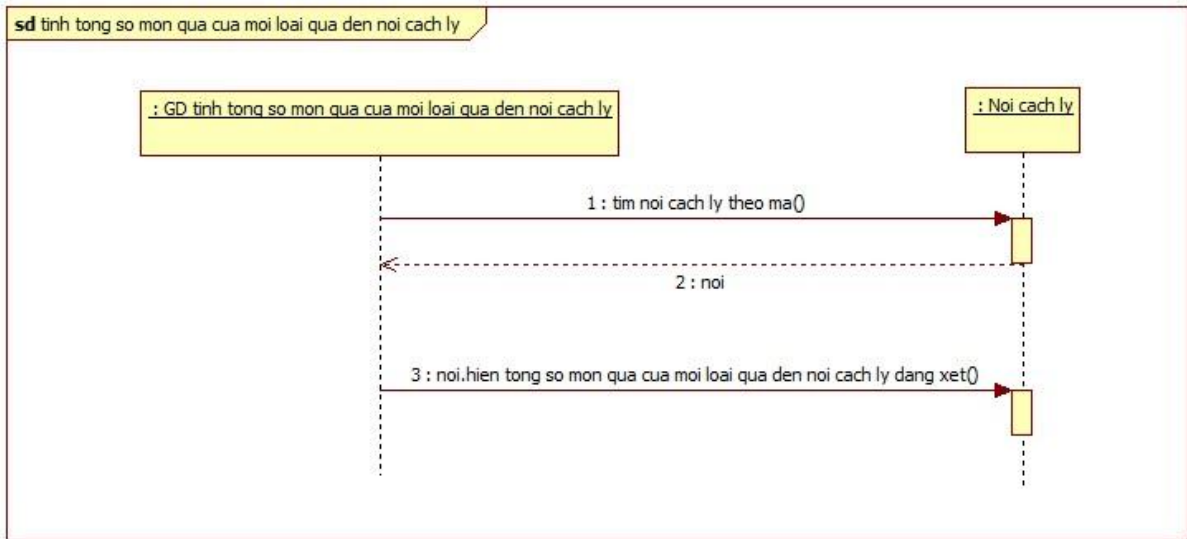
Giờ trở lại sơ đồ lớp như trước.

Ta cần tính **tổng số** món quà của mỗi loại quà đến nơi cách ly.

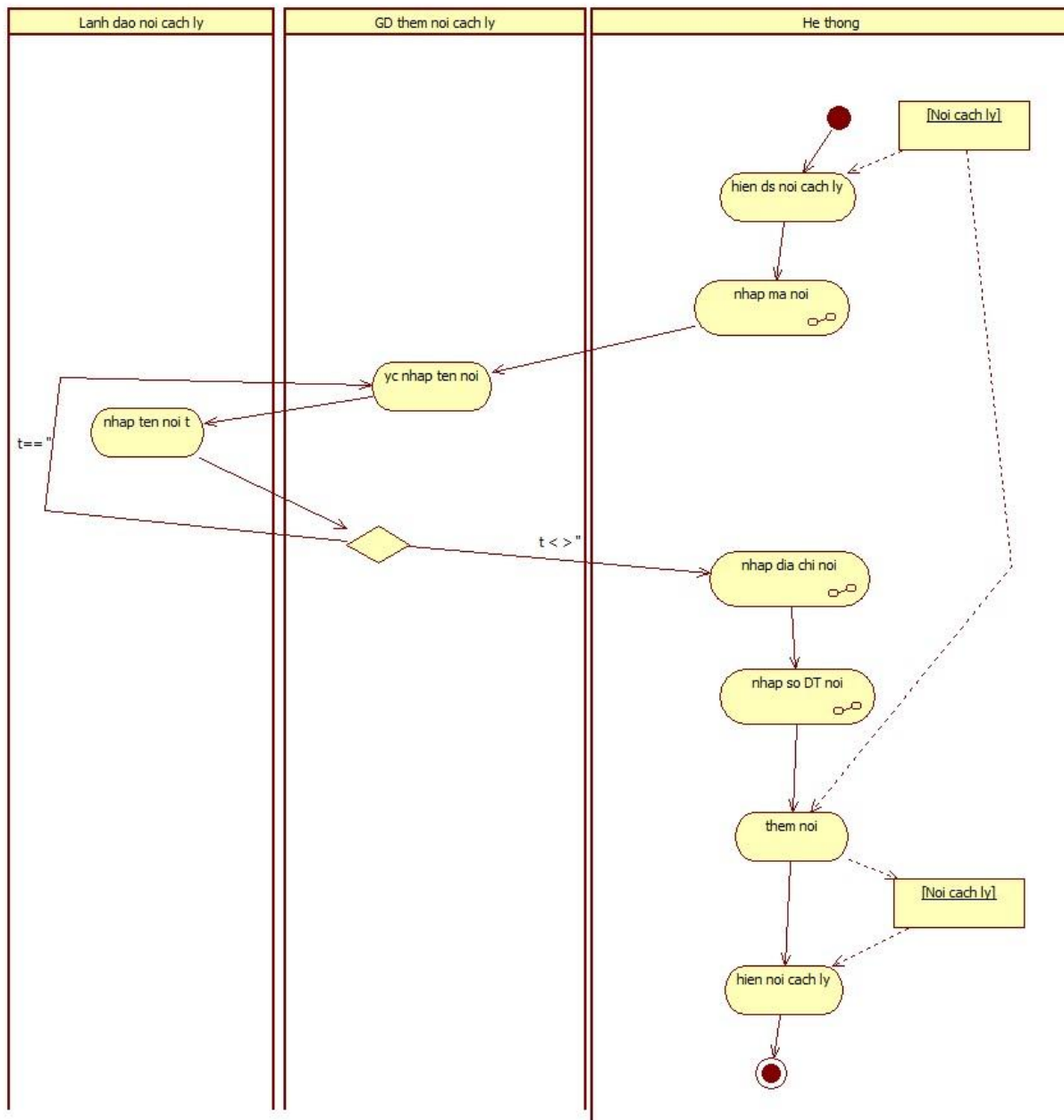
SĐHĐ:



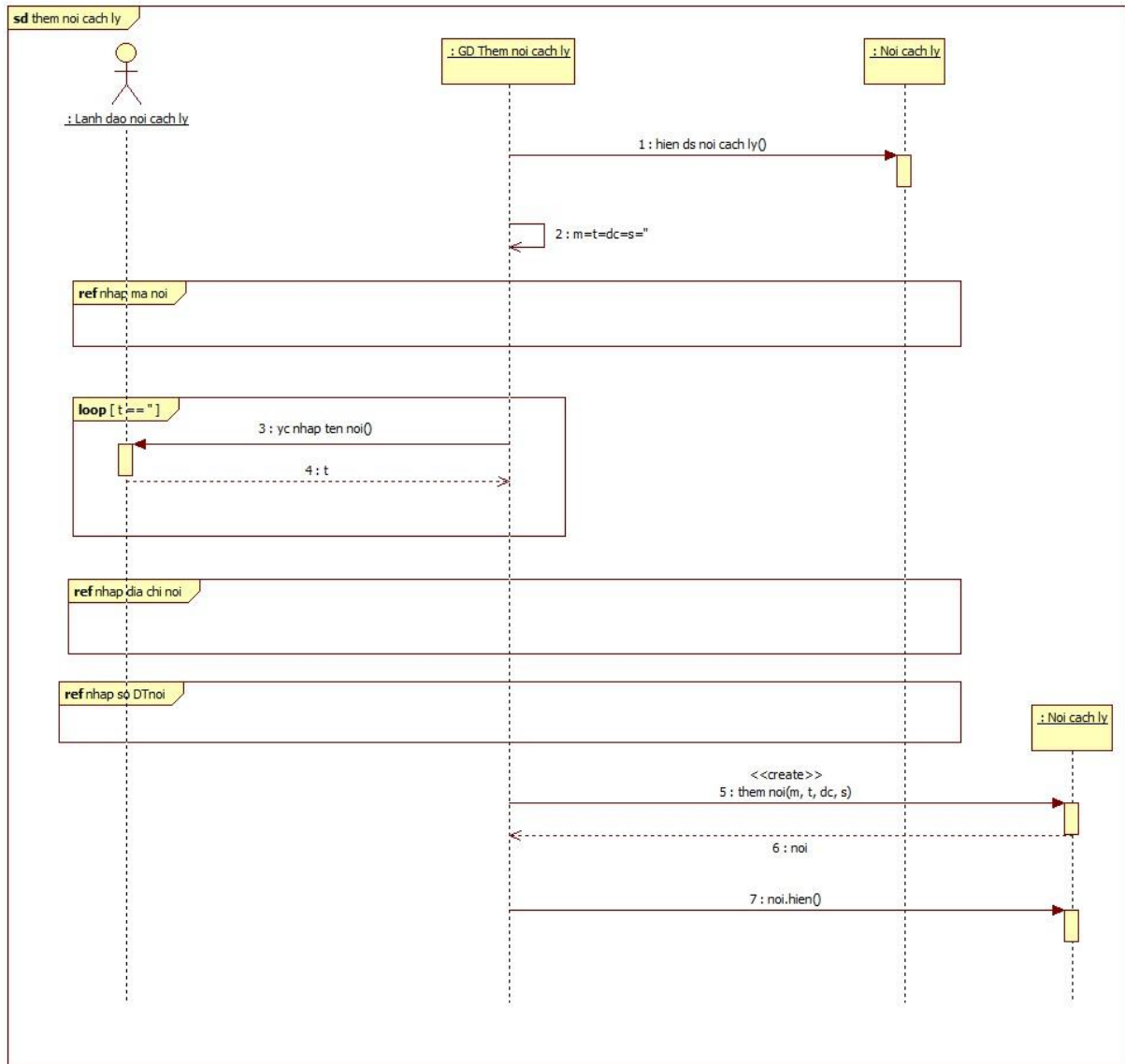
SĐTT”



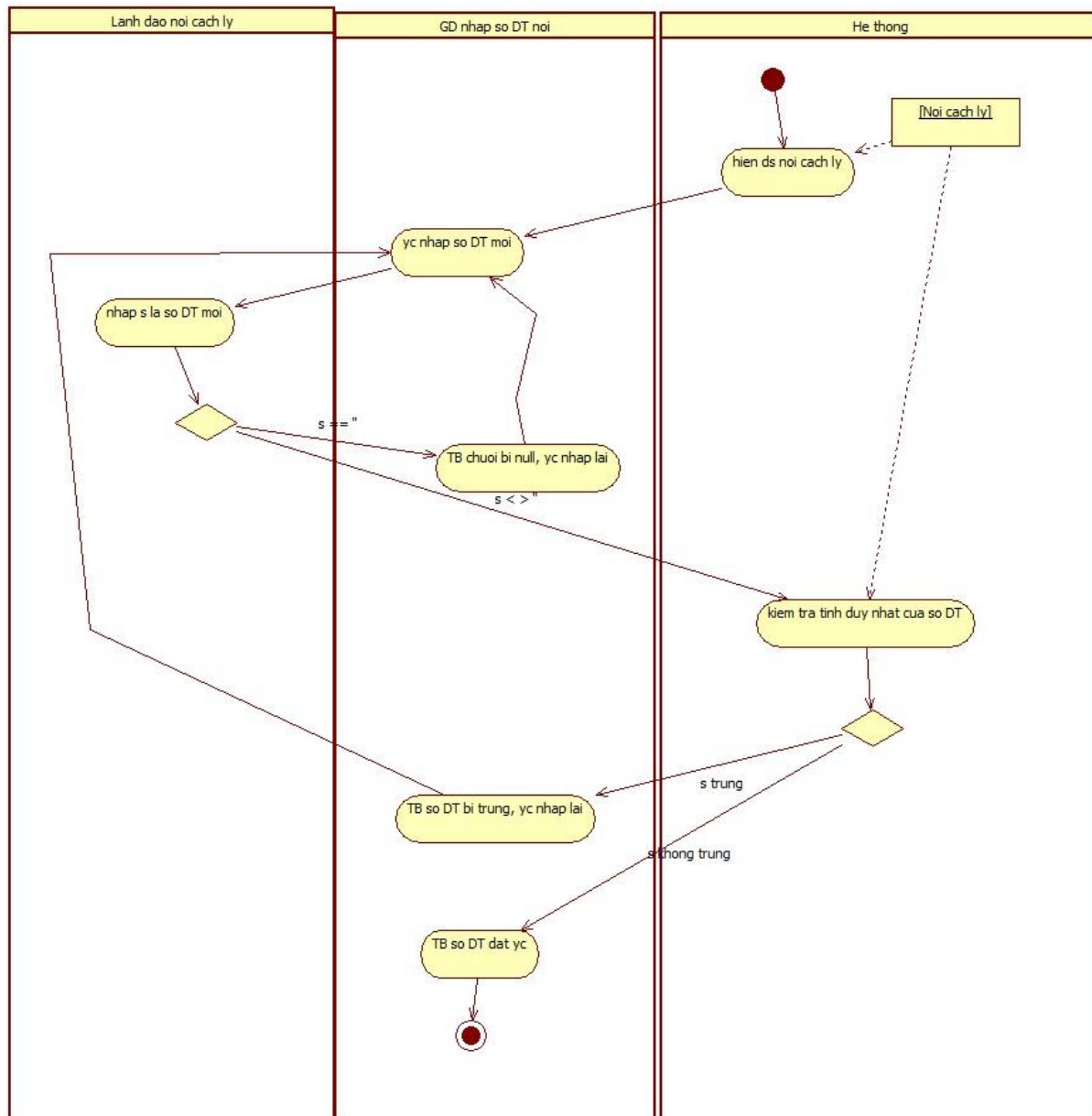
Giờ đến chức năng phức tạp nhất là thêm. Vd thêm nơi cách ly.
 SDHD:



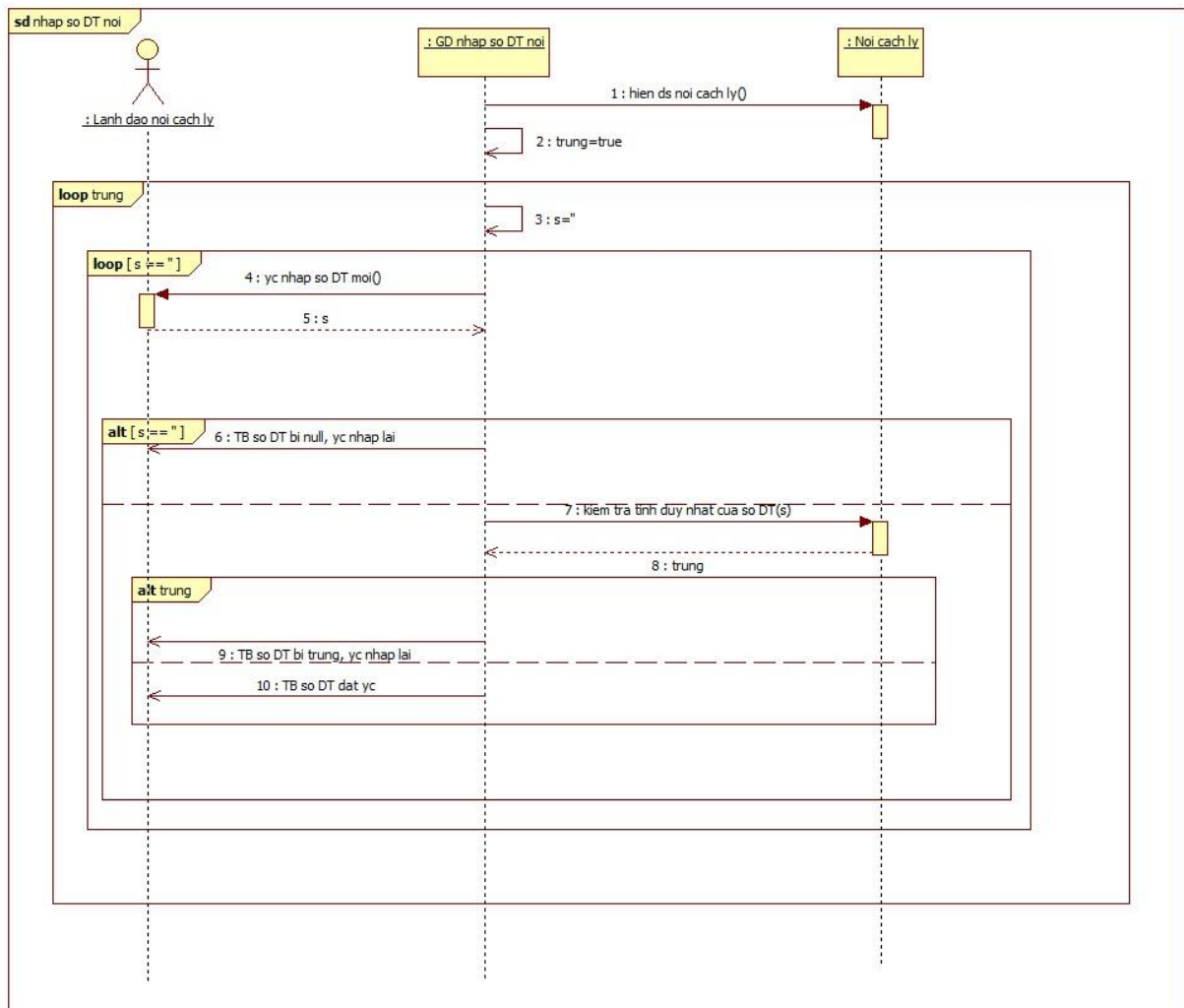
SĐTT:



Nhưng nhớ phải mô tả các SĐHĐ con, các SĐTT được tham chiếu tới !
Vd SĐHĐ nhập số ĐT:



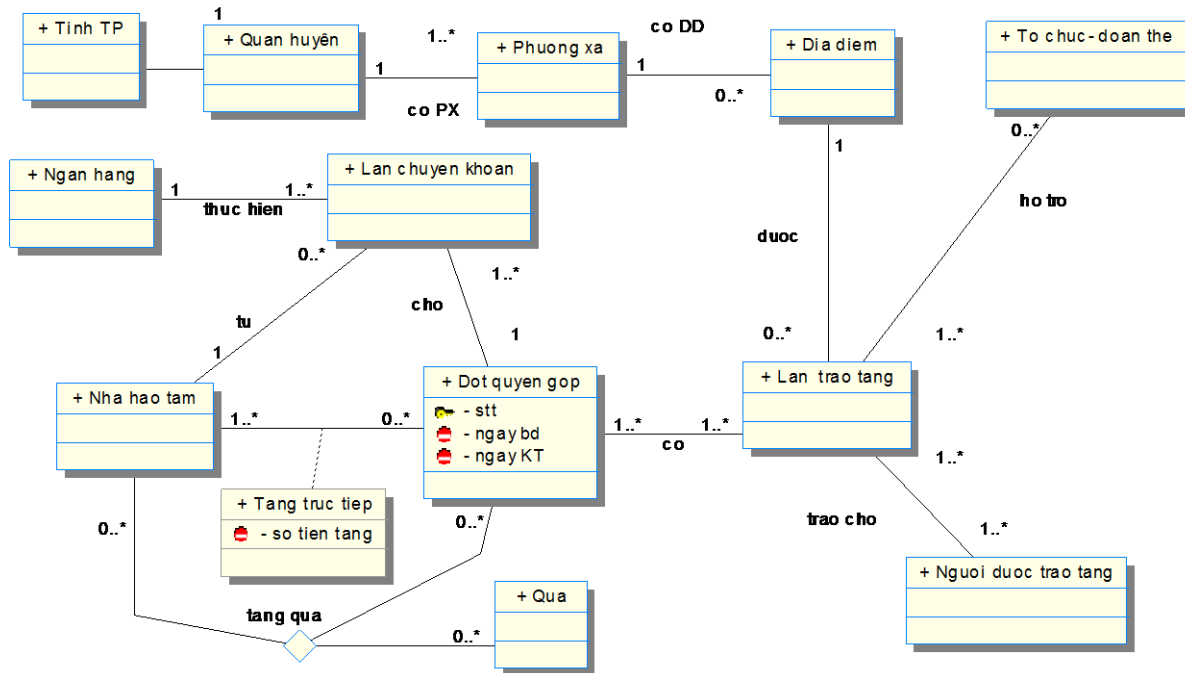
SĐTT nhập số ĐT nơi cách ly:



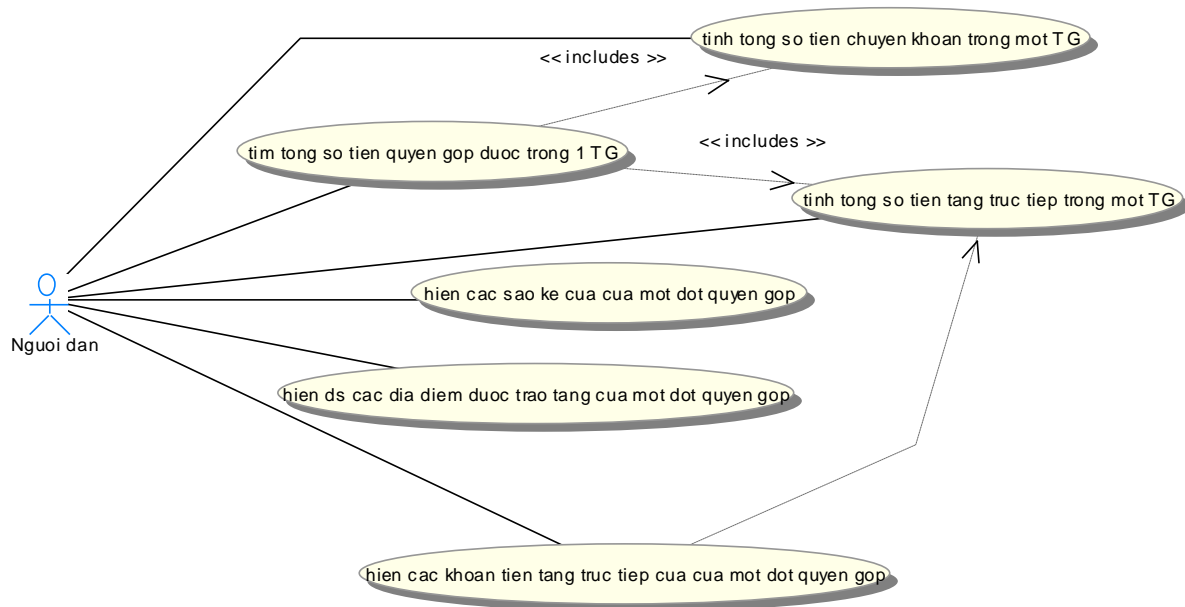
Ta giả sử ở mức luận lý có lớp Phát quà sinh ra từ liên kết “Phát” ở mức quan niệm.

Thêm một lần phát quà thì có thể có trường hợp gặp nhóm tình nguyện mới, hoặc nhóm cách ly mới, hoặc phát món quà mới chưa có trong danh mục (trong CSDL).

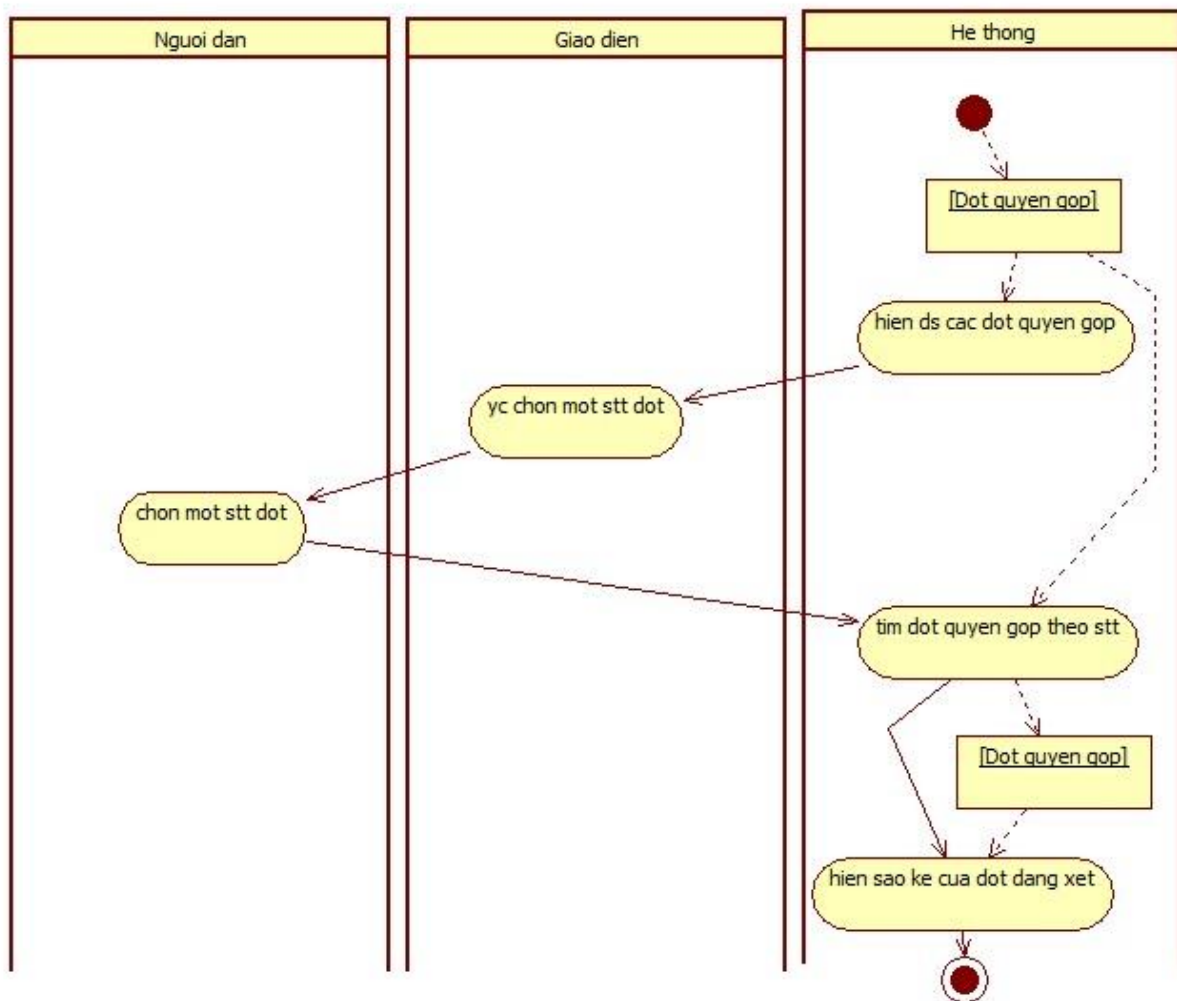
Ví dụ về Quyên góp từ thiện qua một nghệ sĩ.
Ta có sơ đồ lớp:



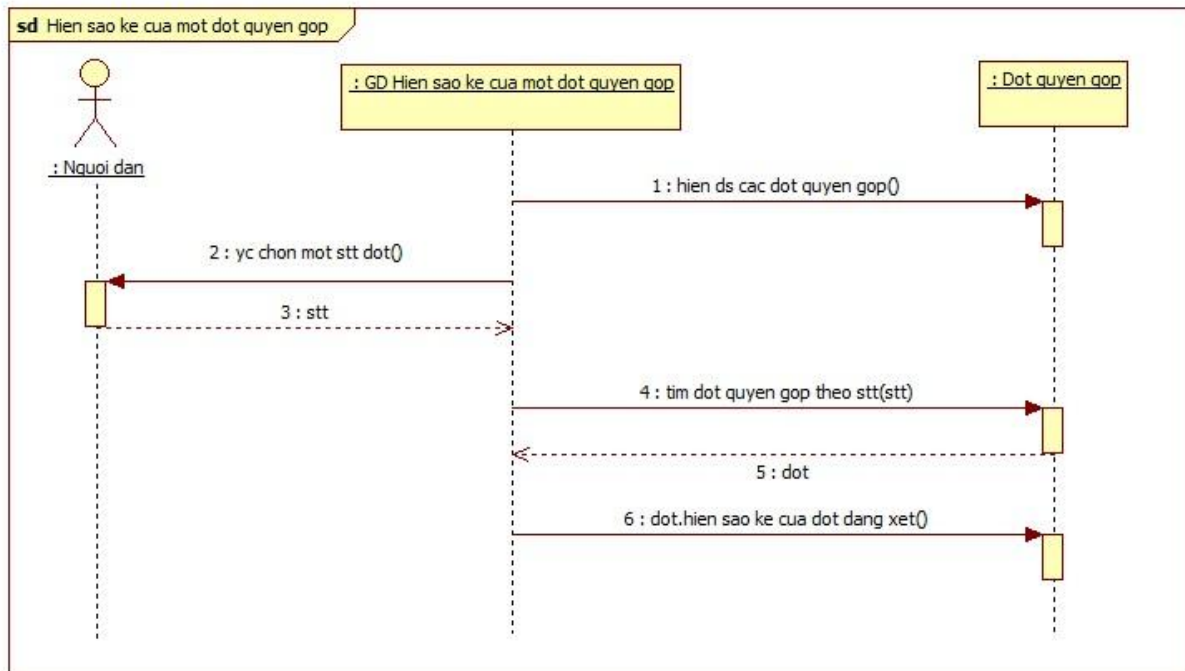
Đối với Người dân, ta có SĐHV sau:



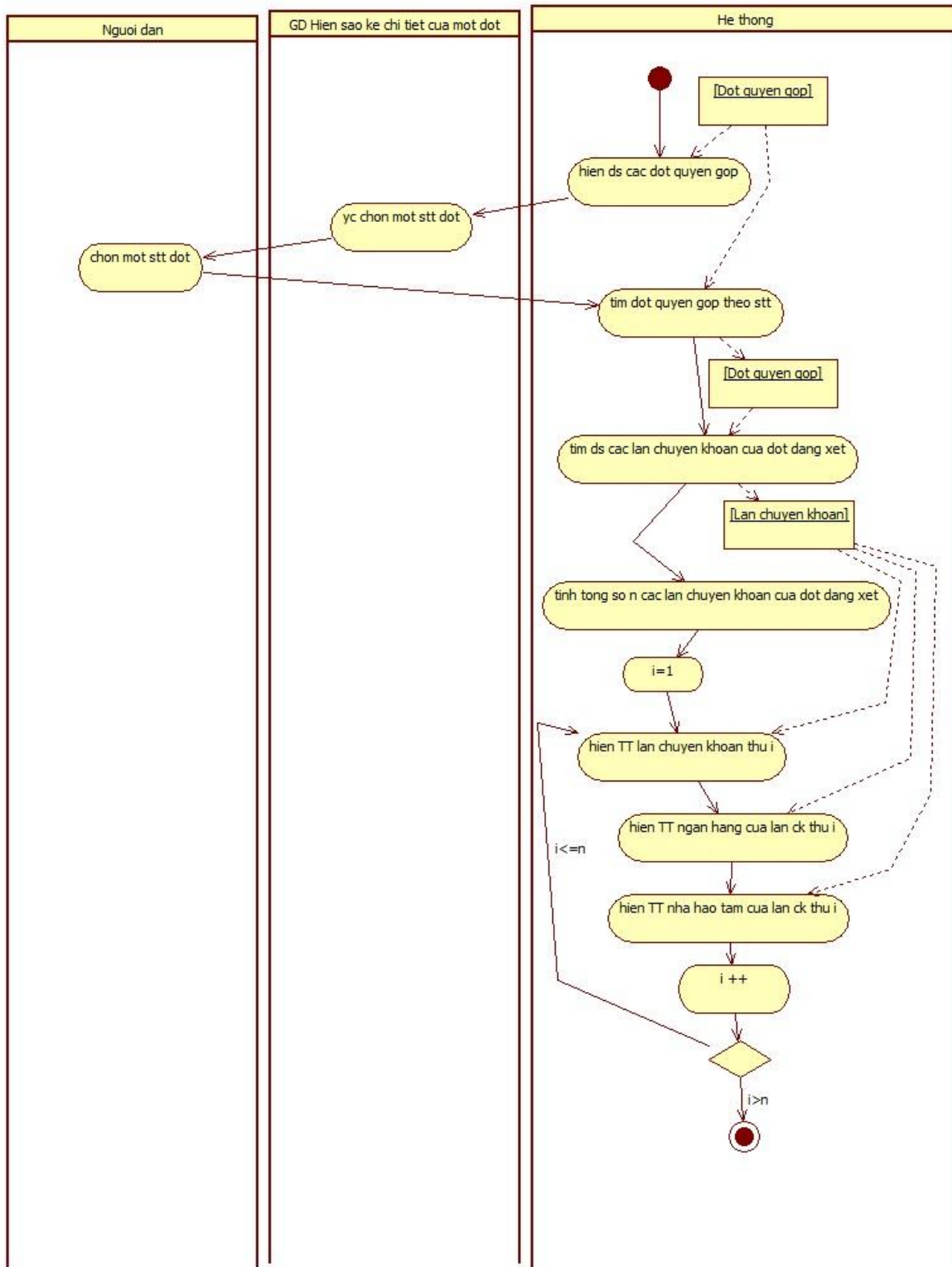
Bây giờ, ta muốn thiết kế SĐHD cho chức năng
“hiện các sao kê của một đot quyên góp”:
(LAN_CK) (DOT QUYEN GOP)

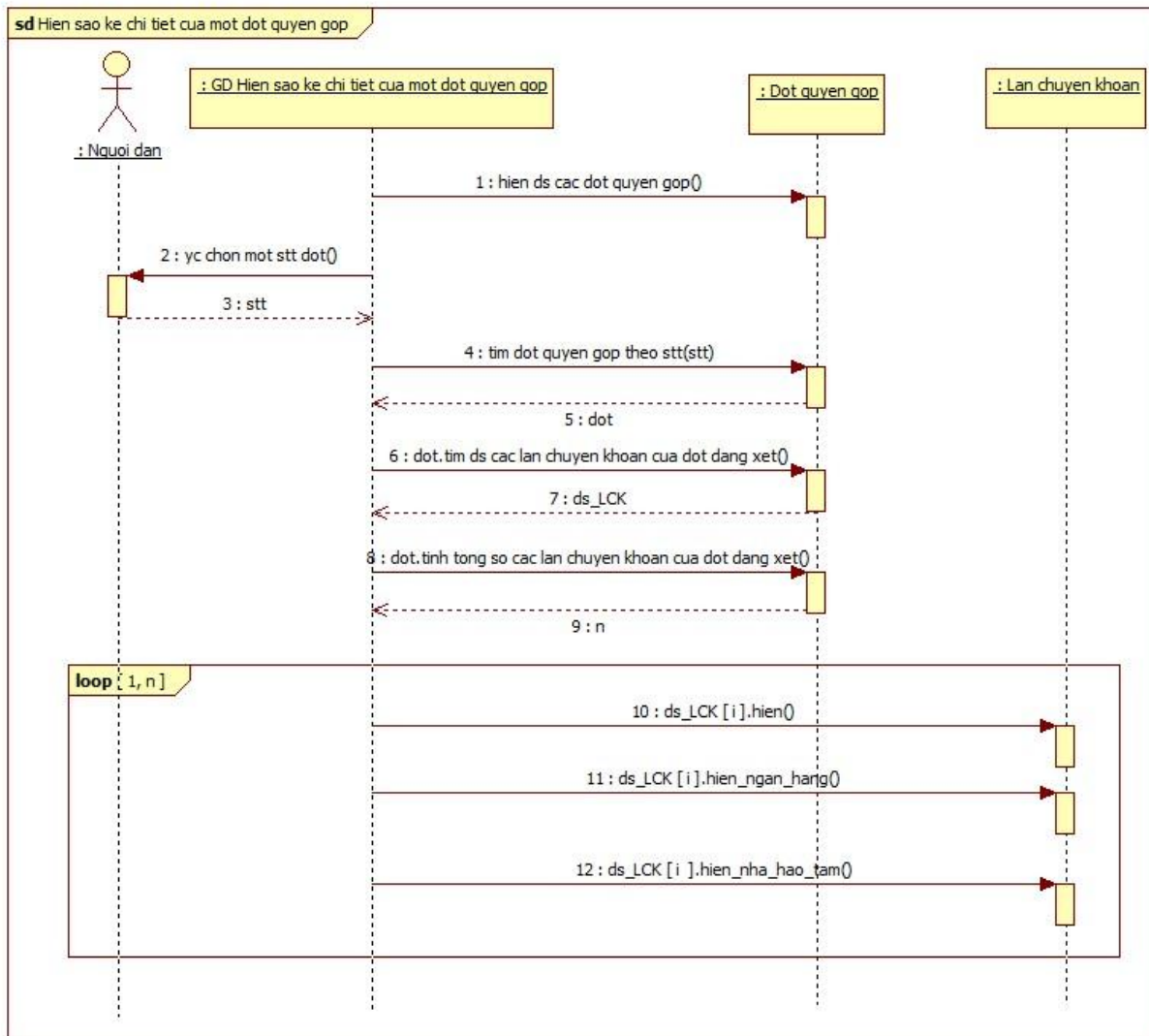


Chuyển sang SĐTT:



Tuy nhiên, vì muốn chắc chắn biết thông tin chi tiết của lần chuyển khoản về ngân hàng và nhà hảo tâm, nên ta có thêm SĐHD và SĐTT dưới đây:





SUM SO_TIEN (LAN_CK),

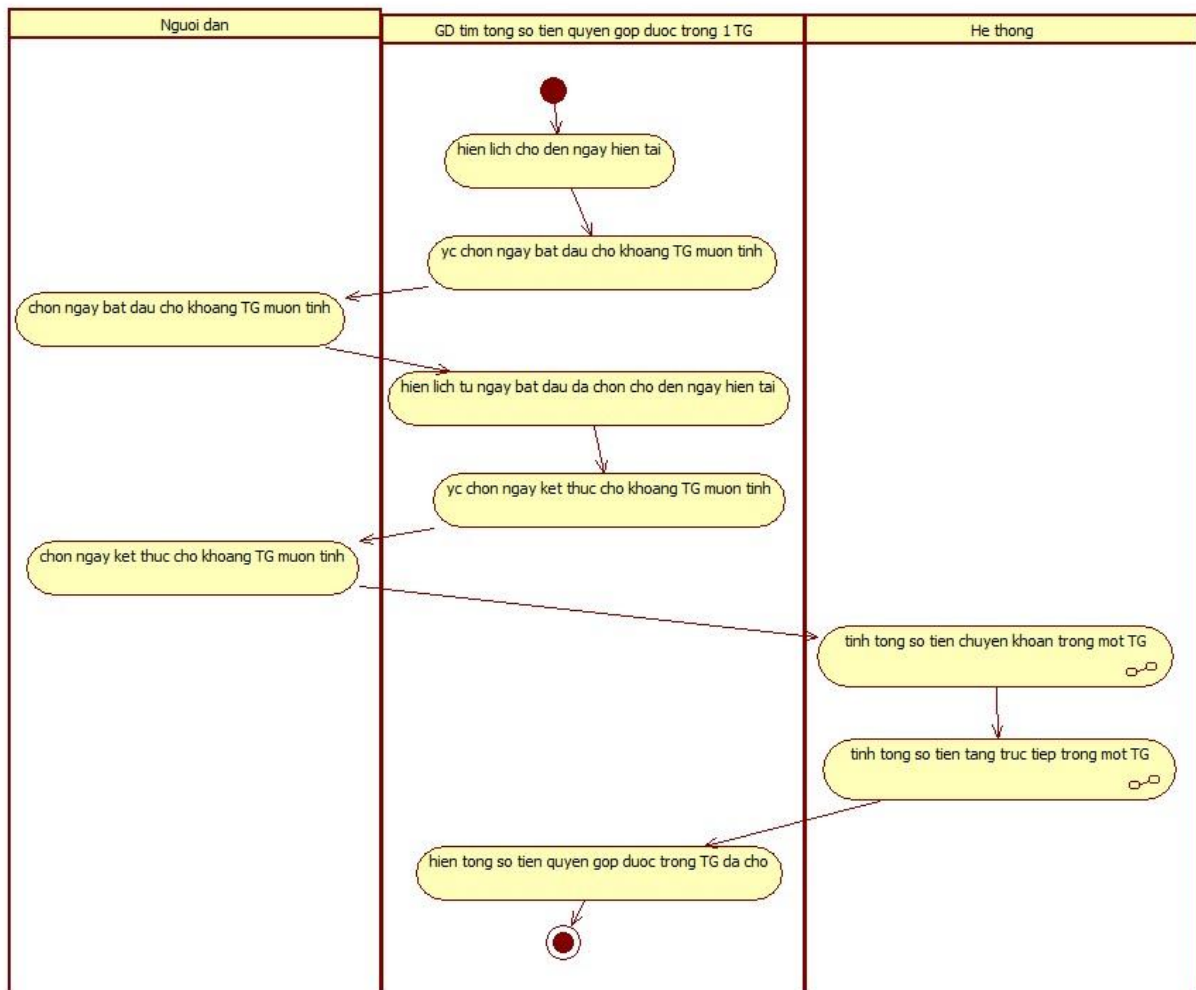
ST_TANG(TANG_TR_TIEP)

THOI_DIEM (LAN_CK),

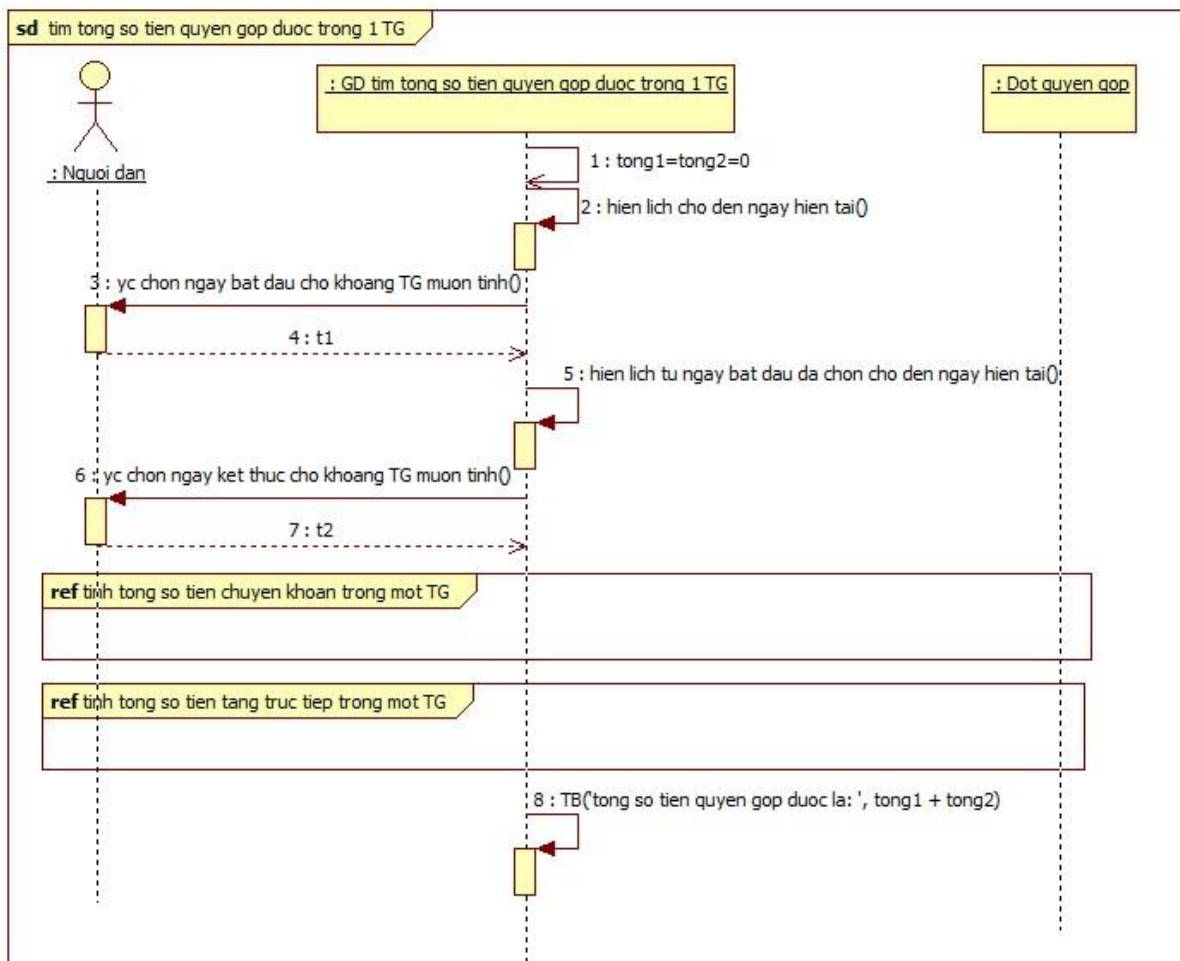
NGAY_BD,NGAY_KT (DOT_Q_GOP)

Giờ ta muốn tính **tổng số tiền quyên góp** được trong **một thời gian**.

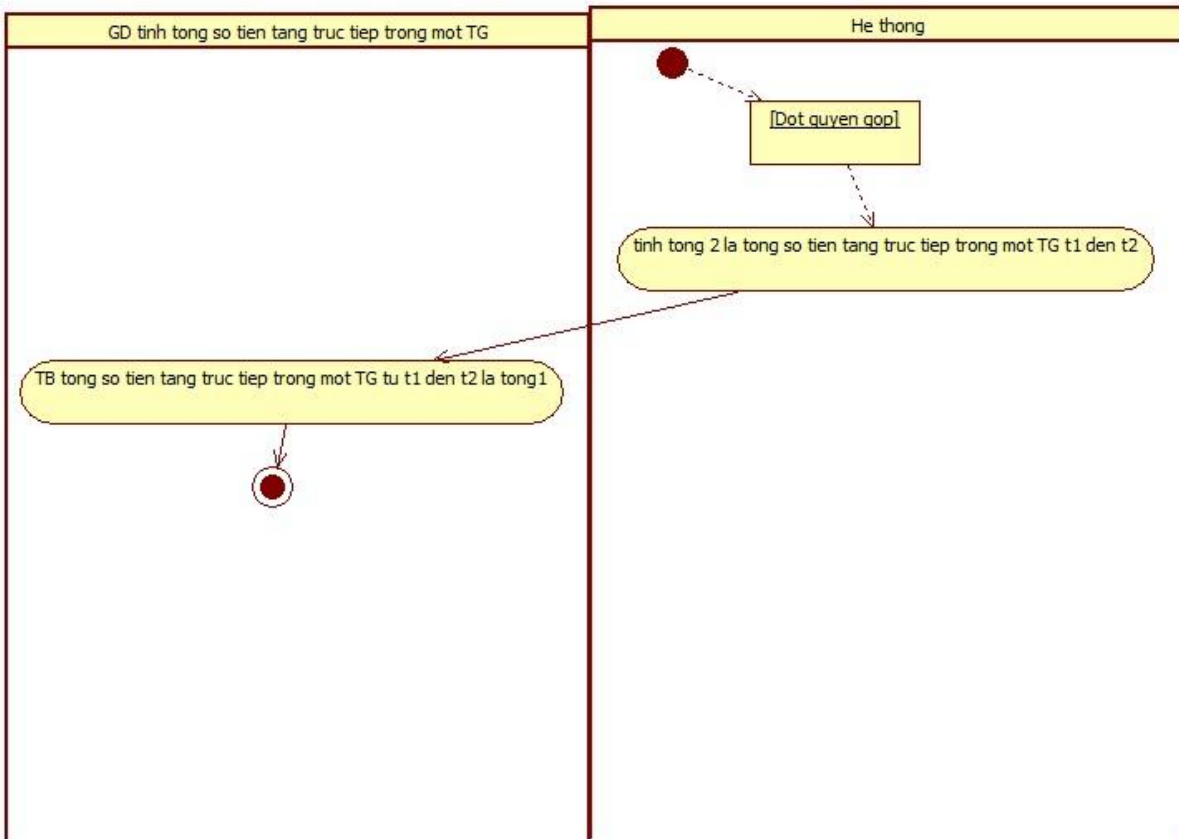
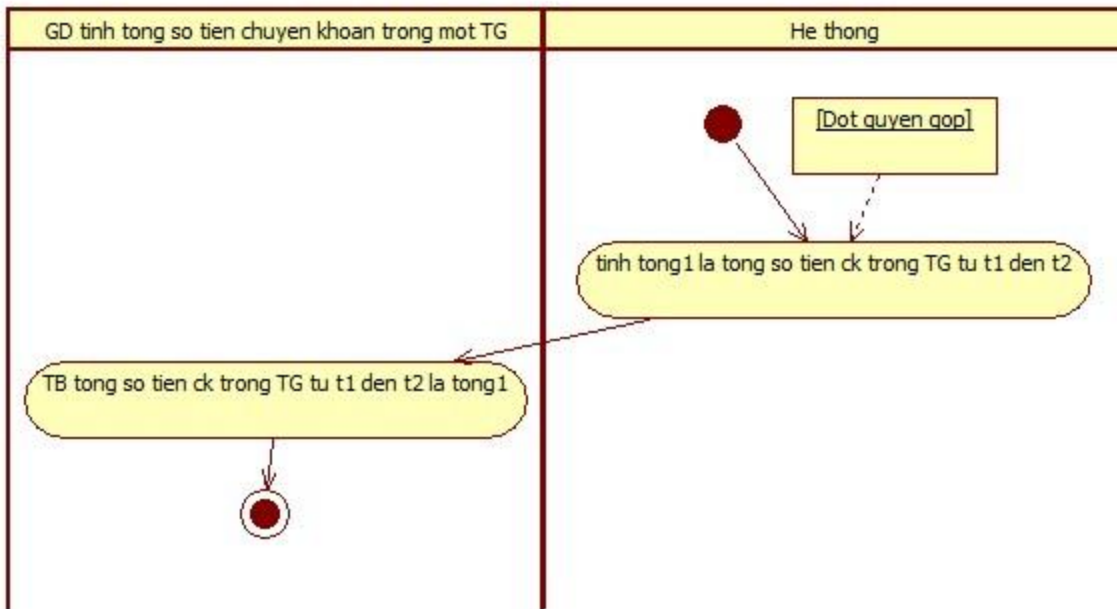
SDHD:



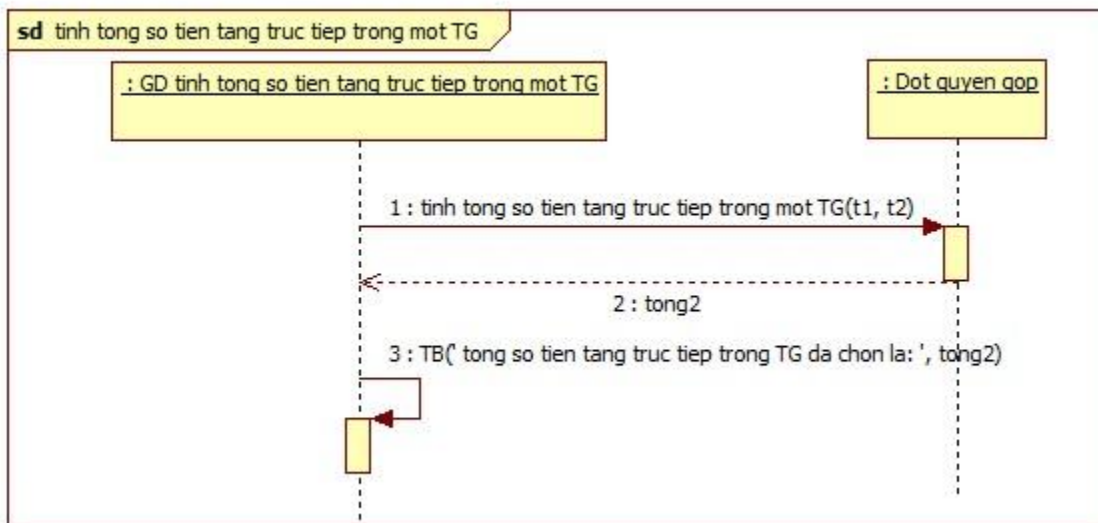
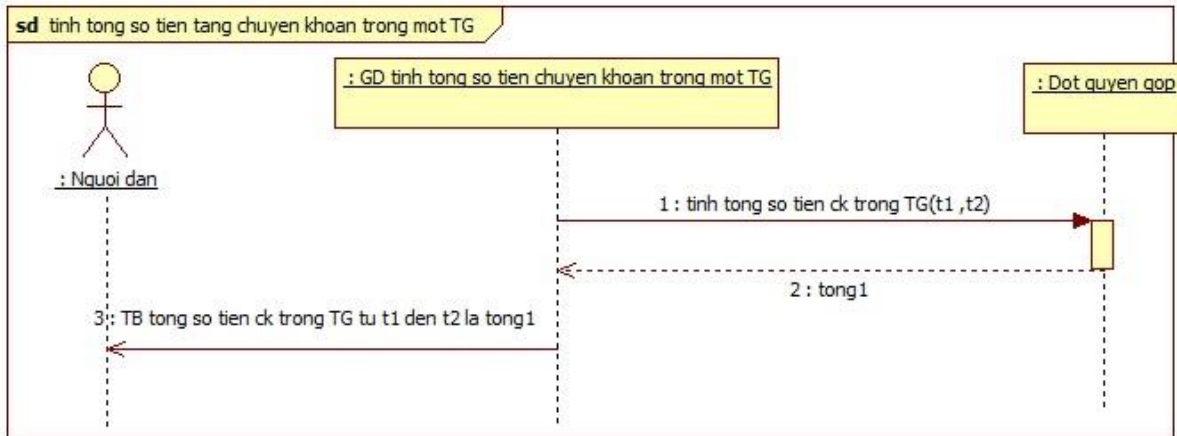
SDTT:



Giờ ta phải mô tả các SDHD con (sub-activity):

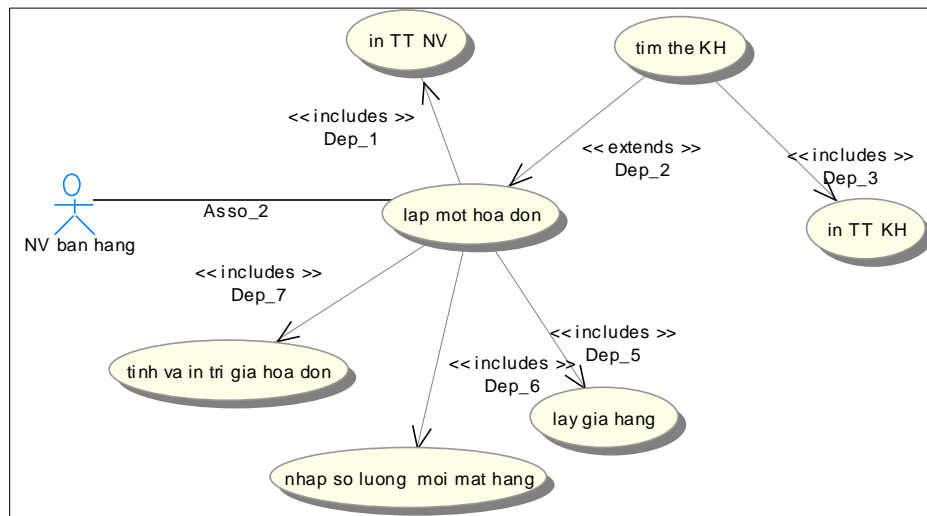


Sau đó, chuyển sang thành các SĐTT:

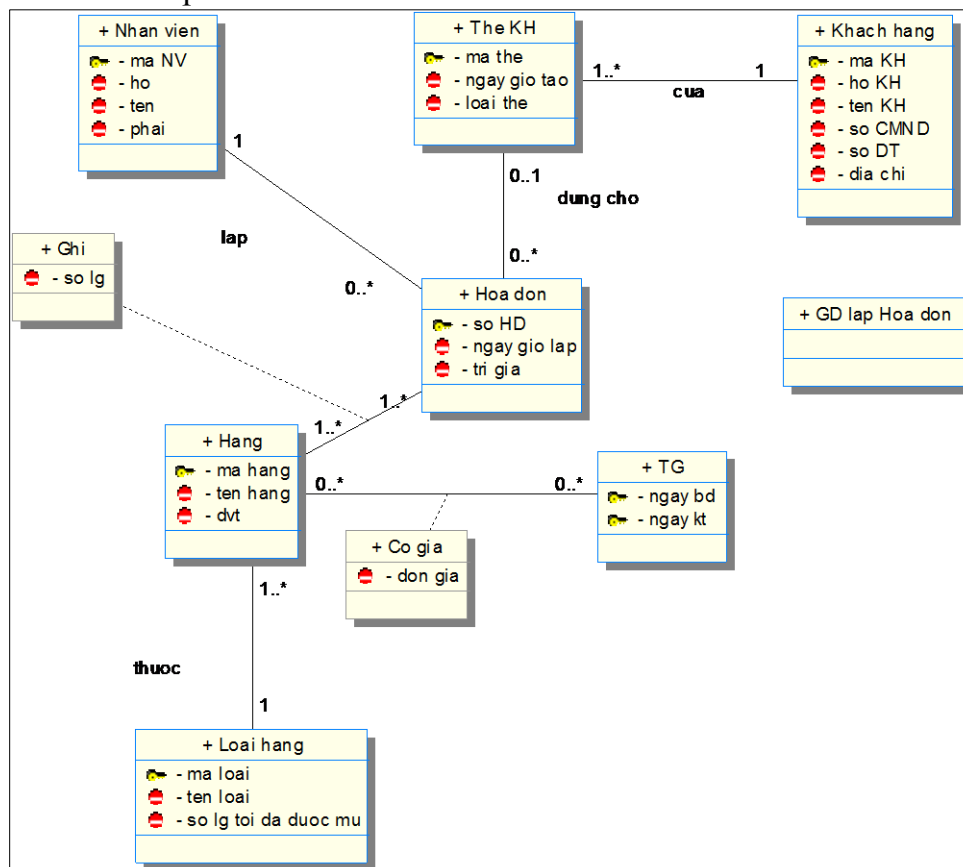


Ví dụ về lập hóa đơn bán hàng trong siêu thị có kéo theo thêm các đối tượng cho các dòng hóa đơn:

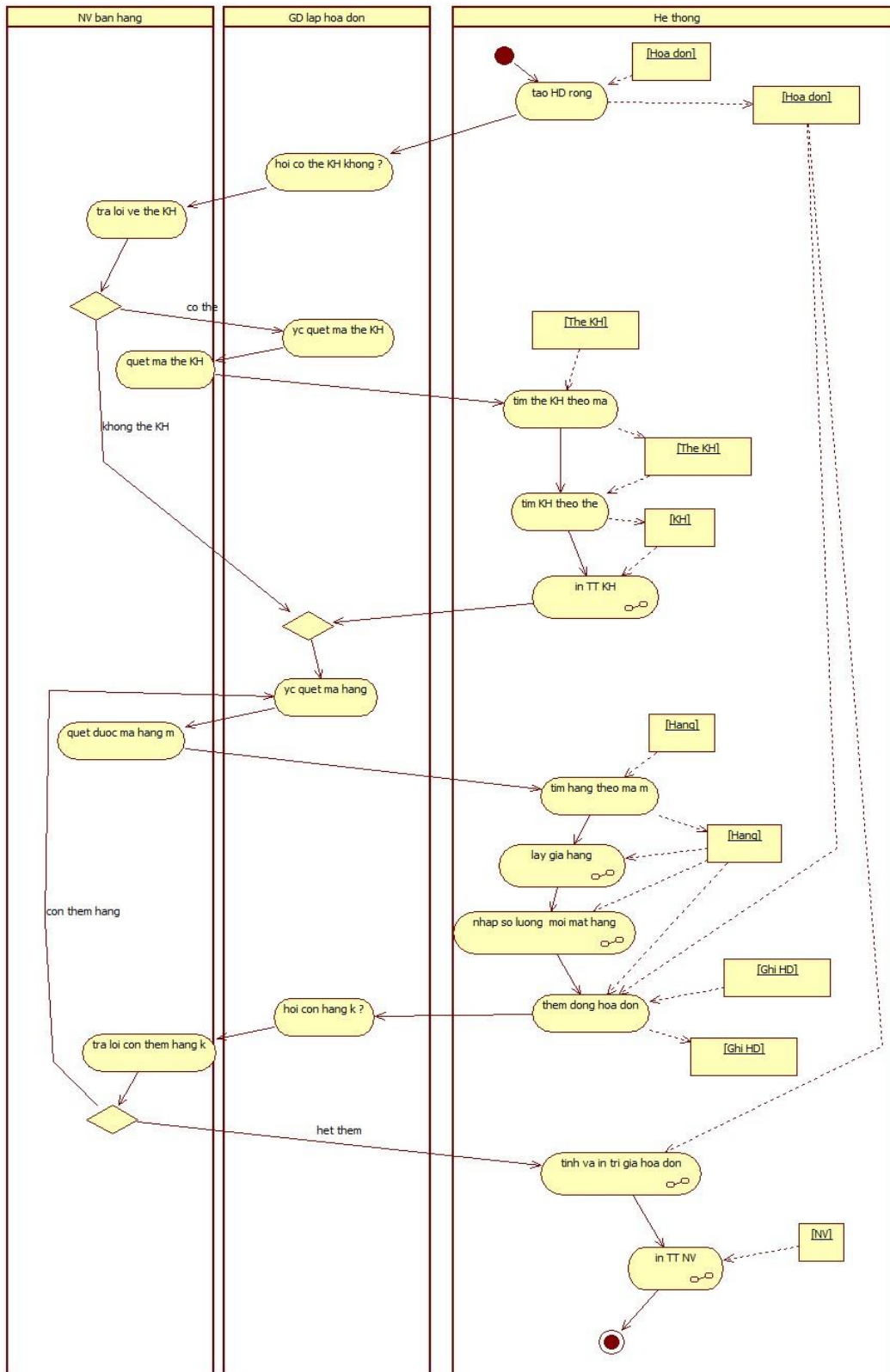
1. SDHV cho NV bán hàng: trình bày một phần



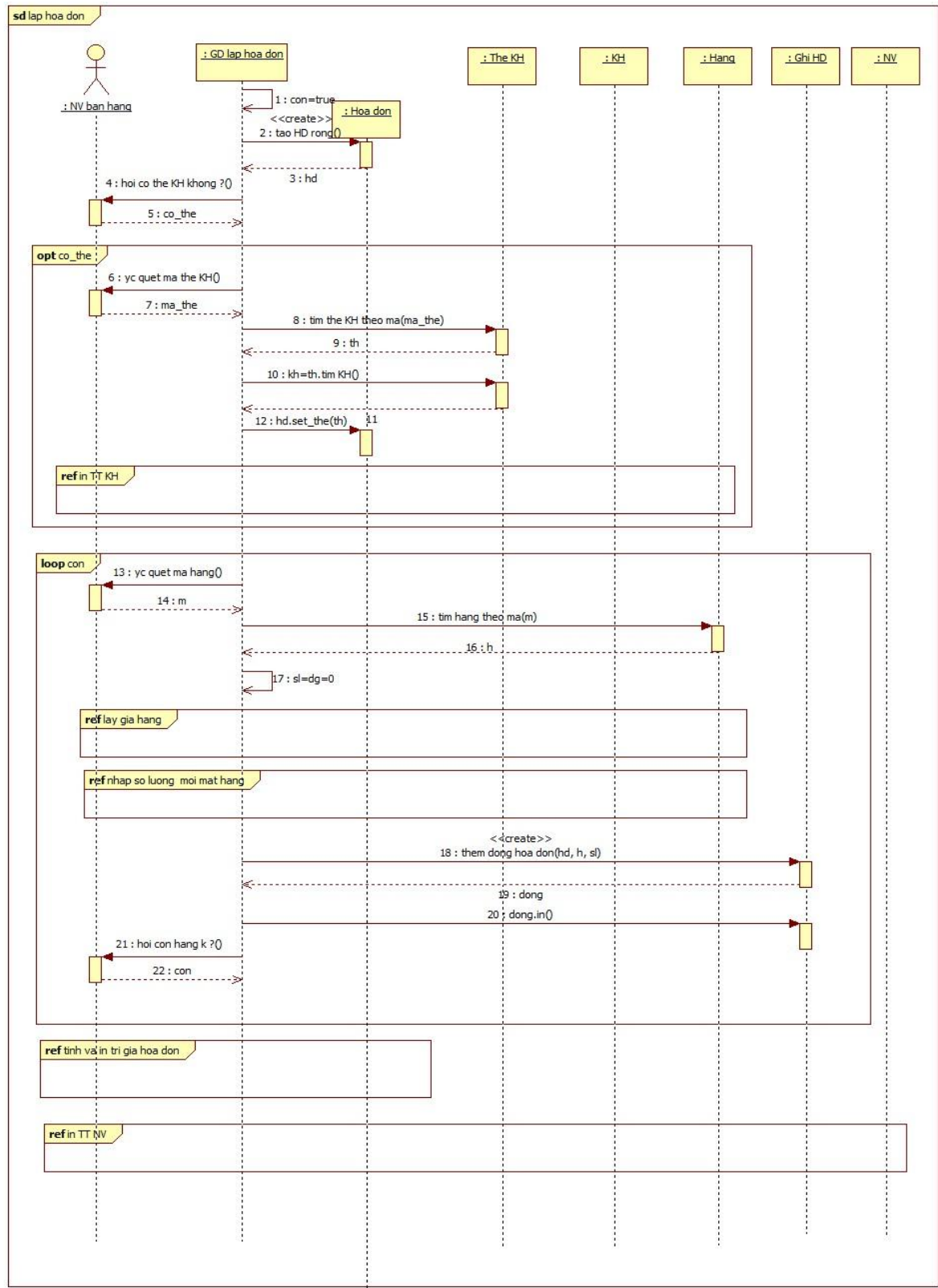
2. SD lớp:



3. SDHD lập hóa đơn:



4. SĐTT lập hóa đơn:

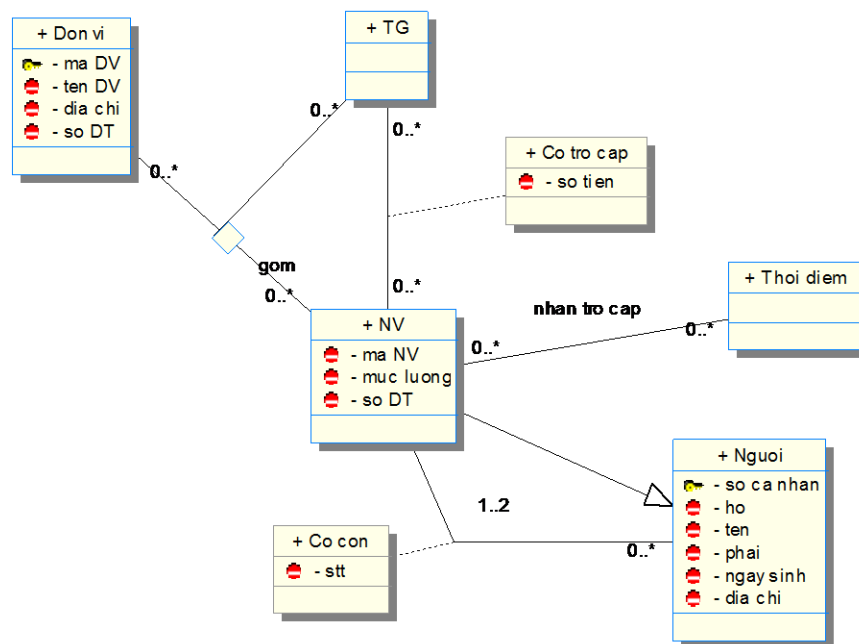


Chương 4- Sơ đồ tuần tự

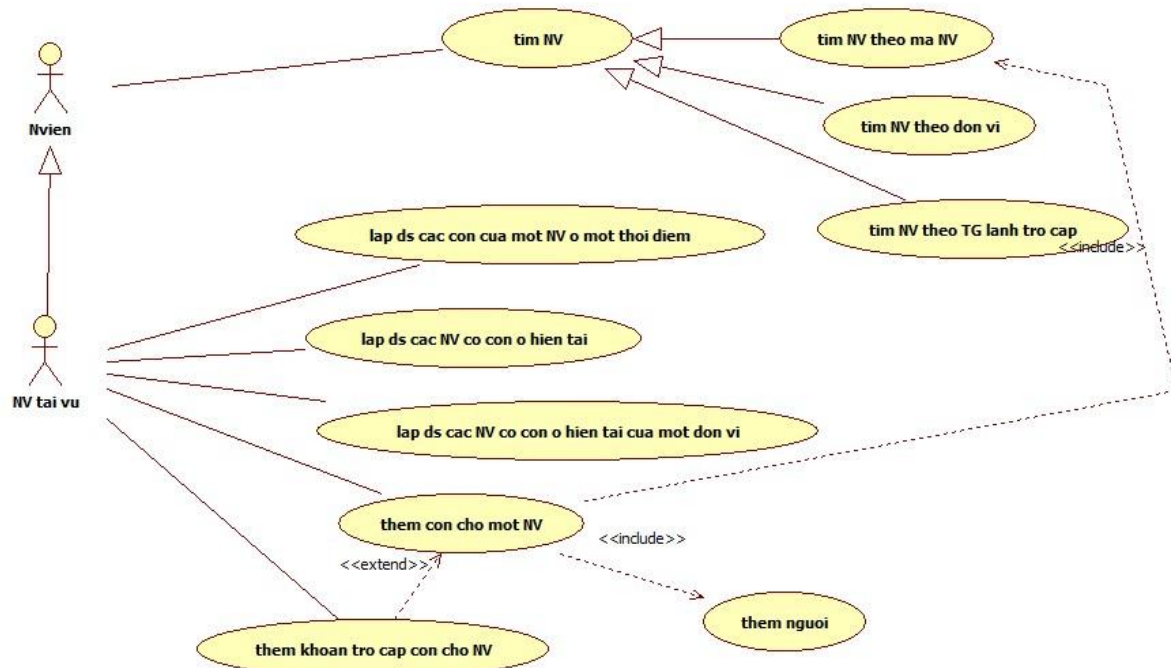
55



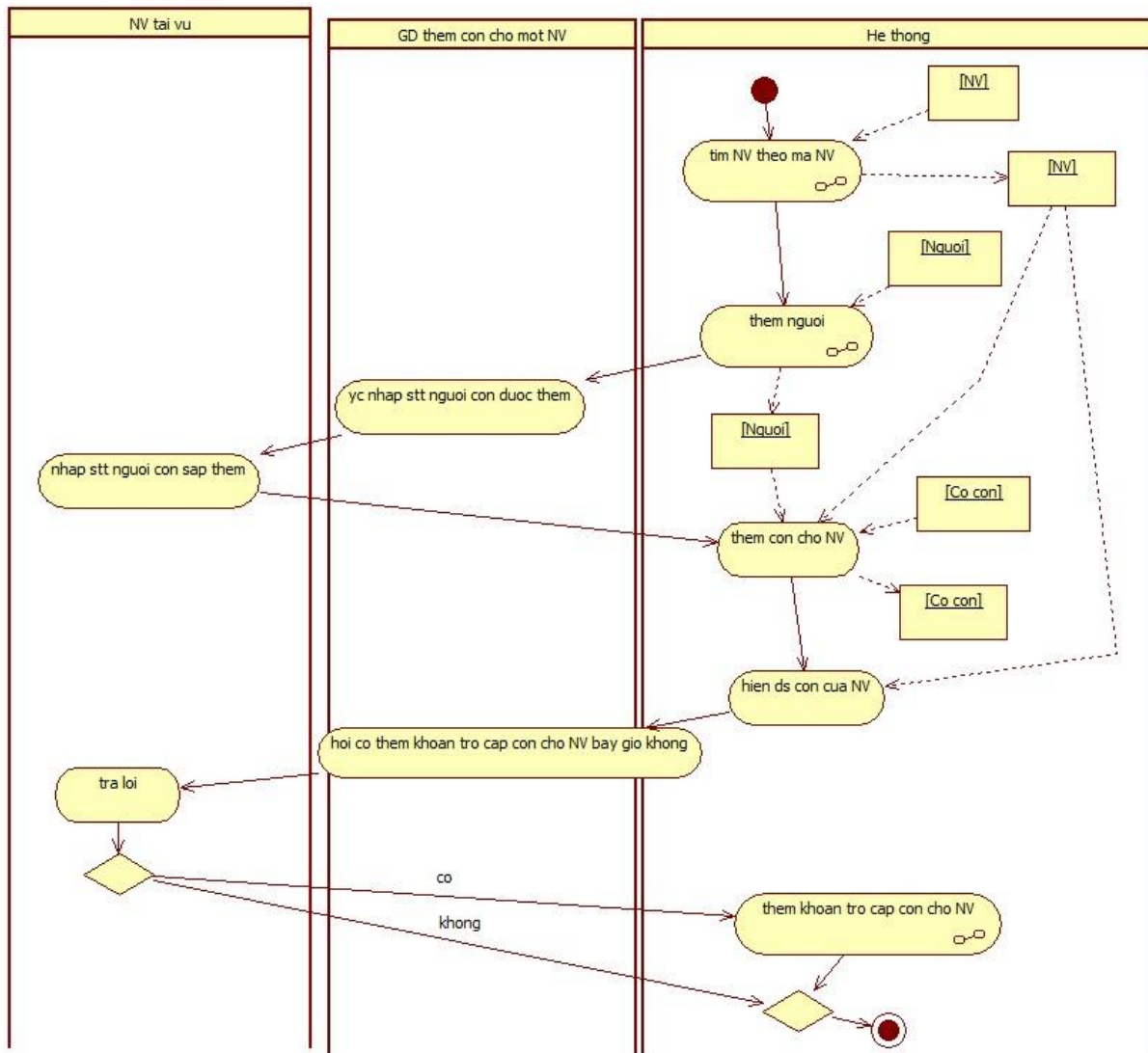
55



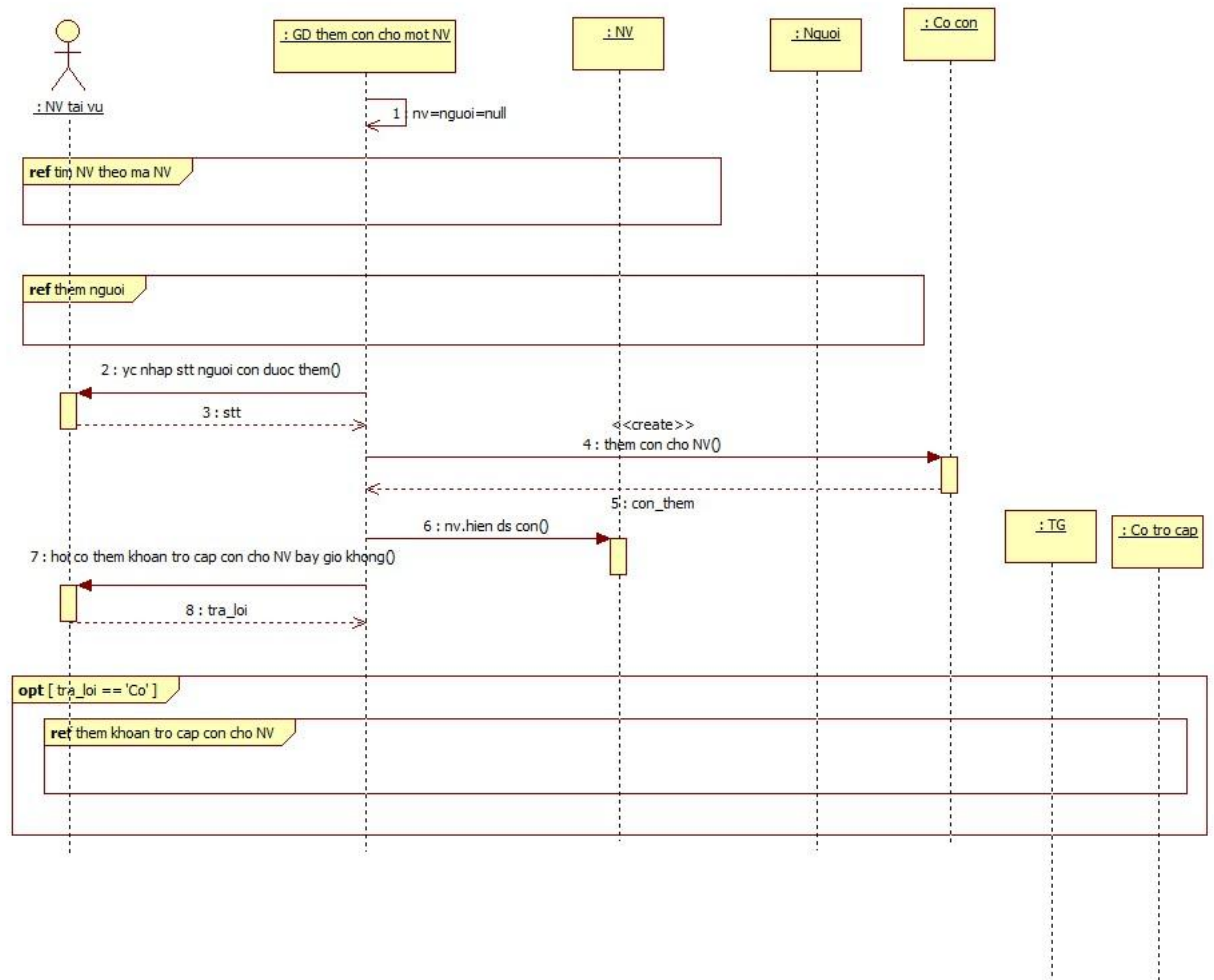
2. SDHV cho NV Tài vụ:



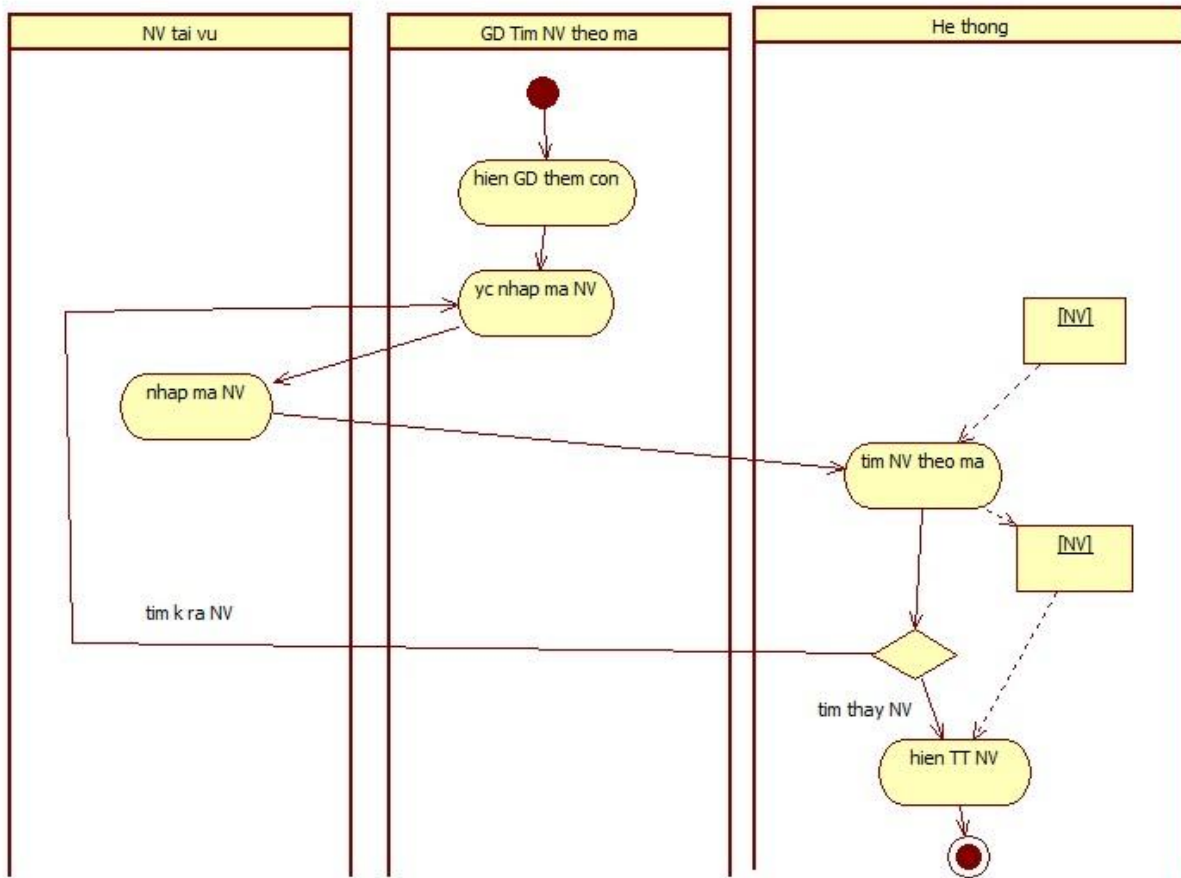
3. SDHD Thêm con cho một NV:



4. SĐTT Thêm con cho một NV:



5. SDHD Tìm NV theo mã NV:



Nếu k chú ý phạm vi biến (variable scope), thì ta sẽ bị error “variable not found” !

Ta có thể tìm NV theo mã nhưng thông qua đơn vị để đỡ phải kiểm tra lỗi gấp phải khi nhập mã sai:

6. SĐHD tìm NV theo mã thông qua đơn vị:

