

# Phân tích thiết kế hệ thống

Soạn bởi: Nguyễn Bá Ngọc

## Chương 1

Hà Nội-2022

# Chương 1.

## Tổng quan về phân tích và thiết kế hệ thống

# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu

# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu

# Chương trình máy tính và hệ thống thông tin

- Chương trình máy tính - là ứng dụng phần mềm chạy trên 1 hệ thống máy tính để thực hiện 1 tập chức năng cụ thể.
  - Phạm vi nhỏ hơn hệ thống thông tin
- Hệ thống thông tin - là sự hợp nhất nhiều thành phần được kết nối và cộng tác để thu thập, xử lý, lưu trữ, và cung cấp thông tin cần thiết để hoàn thành các hoạt động nghiệp vụ.
  - Phạm vi bao phủ chương trình máy tính
  - Bao gồm cả cơ sở dữ liệu và các quy trình thủ công liên quan

# Dự án

- Dự án - Bao gồm chuỗi hoạt động có thời hạn được lập kế hoạch có bắt đầu, có kết thúc và tạo ra các thành phẩm xác định
  - Dự án có thể được tạo lập nhằm phát triển hệ thống thông tin.
  - Các hoạt động cần được thực hiện được xác định, lập kế hoạch, tổ chức và giám sát.
    - Cần kiến thức về phân tích và thiết kế hệ thống, các công cụ và các kỹ thuật

# Quy trình

- Vì sao chúng ta cần quy trình hình thức?
  - Trong thực tế có nhiều dự án bị chậm tiến độ, vượt kinh phí hoặc được bàn giao với ít tính năng hơn kế hoạch ban đầu.
  - Thất bại xảy ra quá thường xuyên.
  - Phát triển hệ thống thông tin là công việc phức tạp.
- Người phân tích hệ thống có vai trò quan trọng
  - Tạo các đặc tả hệ thống đem lại giá trị.
  - Công việc nhiều triển vọng nhưng cũng nhiều thách thức.
  - Phải hiểu các quy trình nghiệp vụ và có tập kỹ năng riêng.

# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu

# Vòng đời phát triển hệ thống (SDLC)

SDLC bao quát toàn bộ tiến trình phát triển hệ thống.

- Tất cả các hoạt động cần cho các giai đoạn lập kế hoạch, phân tích, thiết kế, lập trình, kiểm thử, đào tạo người dùng, v.v.. trong tiến trình phát triển hệ thống
- Tất cả các hoạt động quản lý dự án cần thiết để triển khai thành công dự án.

*SDLC từng bước cho thấy cách một hệ thống thông tin được hình thành bắt đầu từ ý tưởng sơ khai cho tới khi được đưa vào ứng dụng, đáp ứng các nhu cầu nghiệp vụ, từ các đặc tả tới sản phẩm thực tế.*

# SDLC

Có nhiều cách thiết lập SDLC và các mô hình SDLC đều bao gồm nhiều pha, tuy nhiên thường có thể thiết lập ánh xạ giữa các pha của các mô hình SDLC khác nhau.

- Các pha của 1 mô hình SDLC điển hình:
  - Lập kế hoạch;
  - Phân tích;
  - Thiết kế;
  - Thực thi.
- Mỗi pha bao gồm 1 chuỗi các bước
- Mỗi pha đều tạo ra sản phẩm (có thể chuyển giao)
- Các pha có thể được thực hiện tuần tự, tăng dần, lặp hoặc theo 1 số quy luật khác.

# Tập câu hỏi cần được làm rõ trong các pha cơ bản

- Pha lập kế hoạch
  - Vì sao chúng ta nên xây dựng hệ thống này?
  - Nó sẽ đem lại lợi ích gì?
  - Xây dựng nó như thế nào? và ai sẽ xây dựng nó?
  - Cần bao nhiêu thời gian và kinh phí để xây dựng nó?
- Pha phân tích
  - Ai sẽ sử dụng nó? ở đâu? và trong điều kiện nào?
  - Hệ thống cần làm những gì?
- Pha thiết kế
  - Hệ thống hoạt động như thế nào?
  - Hệ thống được xây dựng như thế nào?

# Một số hoạt động tiêu biểu

## Pha lập kế hoạch

### 1. Khởi tạo dự án

- Tạo/tiếp nhận các yêu cầu hệ thống
- Thực hiện phân tích tính khả thi

### 2. Tổ chức quản lý dự án

- Lập kế hoạch làm việc
- Bố trí nhân sự dự án
- Lên kế hoạch theo dõi và điều hành dự án

# Một số hoạt động tiêu biểu<sub>(2)</sub>

## Pha phân tích

### 1. Lập chiến lược phân tích

- Mô hình hóa hệ thống đang có
- Định hình hệ thống mới

### 2. Thu thập thông tin

- Mô tả ý tưởng hệ thống
- Tạo các mô hình

### 3. Tạo bản đề xuất hệ thống

# Một số hoạt động tiêu biểu<sub>(3)</sub>

## Pha thiết kế

1. Xây dựng chiến lược thiết kế
2. Thiết kế kiến trúc hệ thống
3. Thiết kế giao diện người dùng
4. Thiết kế lưu trữ cố định
5. Thiết kế chi tiết lớp

# Một số hoạt động tiêu biểu<sub>(4)</sub>

Pha thực thi

1. Xây dựng hệ thống
  - Lập trình
  - Kiểm thử
2. Chuyển giao hệ thống
  - Cài đặt
  - Đào tạo & hướng dẫn sử dụng
3. Hỗ trợ (bảo trì) hệ thống

# Sản phẩm đầu ra tiêu biểu cho các pha

<b>Pha</b>	<b>Sản phẩm tiêu biểu</b>
Lập kế hoạch	⇒ Kế hoạch dự án
Phân tích	⇒ Bản đề xuất hệ thống
Thiết kế	⇒ Tài liệu đặc tả hệ thống
Thực thi	⇒ Hệ thống

# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu

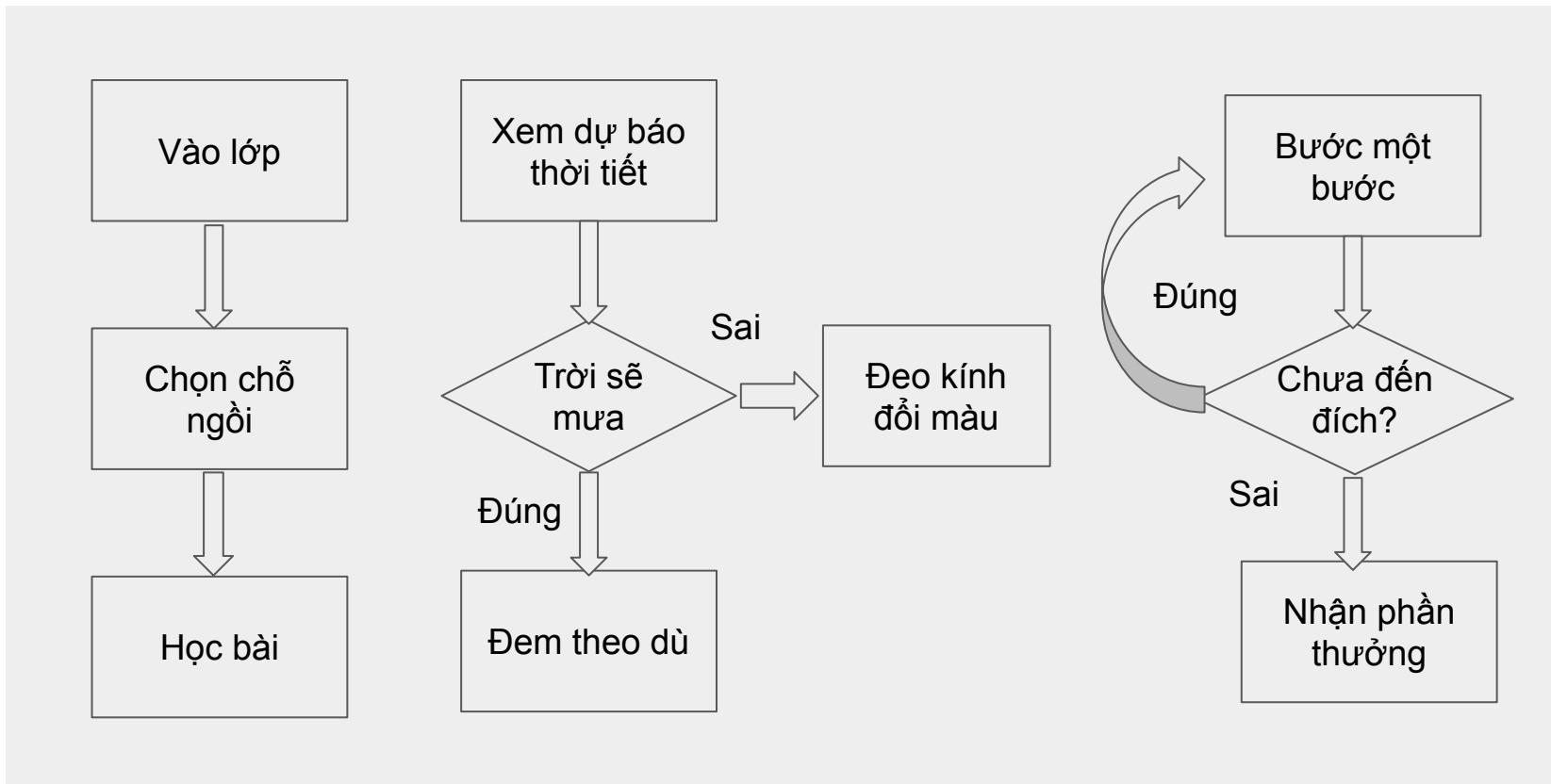


# Hai cách tiếp cận phát triển hệ thống

- Cách tiếp cận hướng cấu trúc
  - Cách tiếp cận ra đời sớm hơn. Biểu diễn hệ thống như một tập tiến trình tương tác và xử lý dữ liệu
  - Tập trung vào tiến trình.
  - Bao gồm: Phân tích hướng cấu trúc, thiết kế hướng cấu trúc, và lập trình hướng cấu trúc
- Cách tiếp cận hướng đối tượng
  - Cách tiếp cận mới hơn. Biểu diễn hệ thống như một tập đối tượng tương tác với nhau để hoàn thành các nhiệm vụ.
  - Nỗ lực cân bằng sự tập trung vào tiến trình và dữ liệu
  - Phân tích hướng đối tượng, thiết kế hướng đối tượng, và lập trình hướng đối tượng

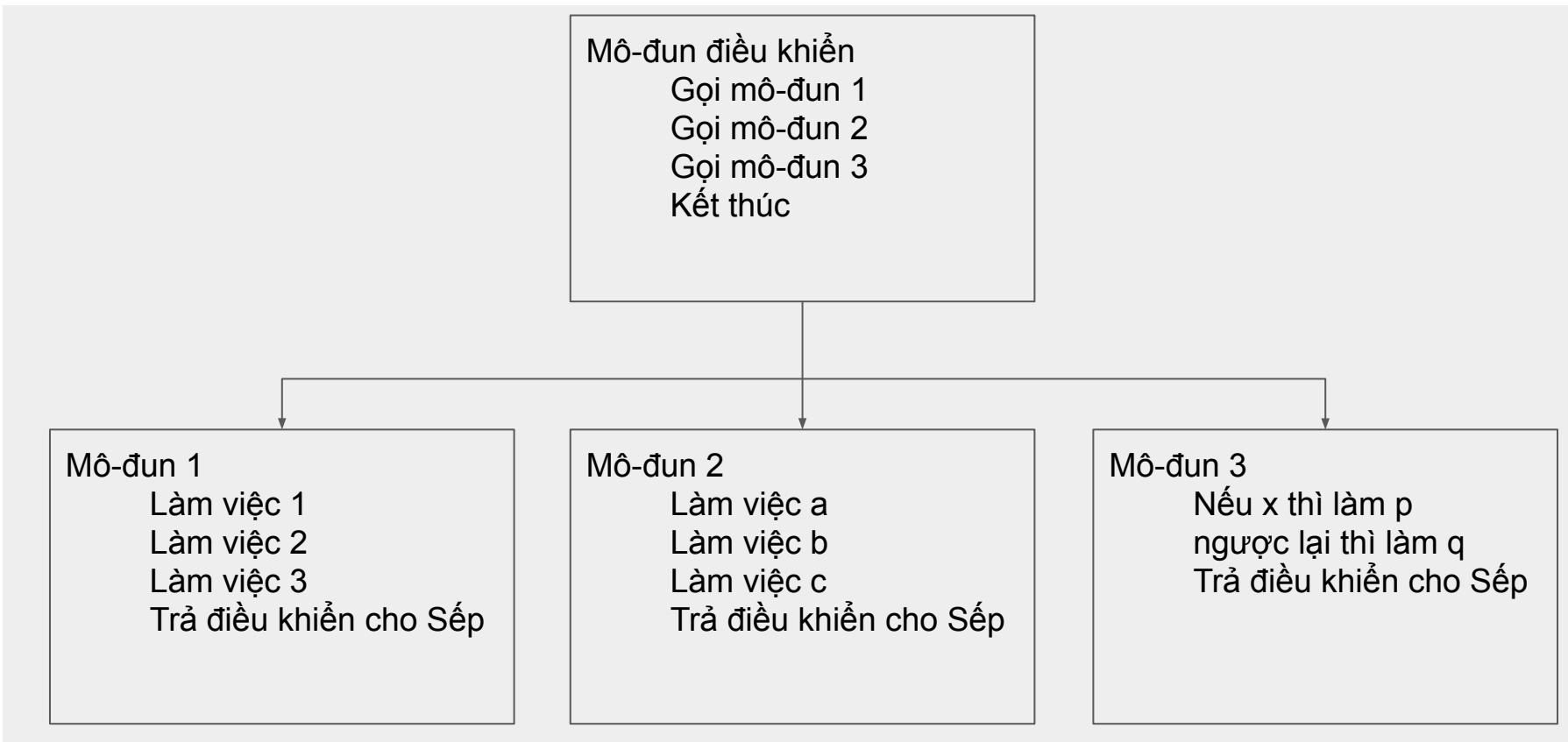
# Cách tiếp cận hướng cấu trúc

- Lập trình hướng cấu trúc
  - Biến
  - Xử lý tuần tự, rẽ nhánh, và lặp



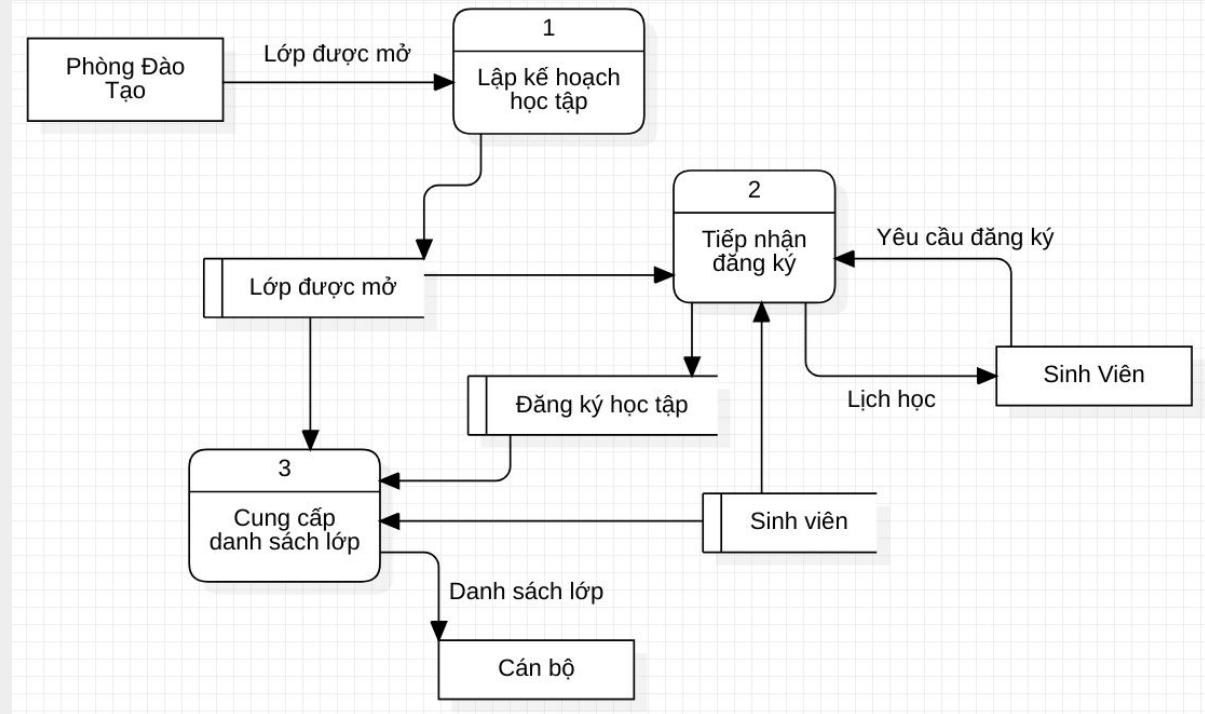
# Cách tiếp cận hướng cấu trúc<sub>(2)</sub>

- Chia nhỏ từ trên xuống



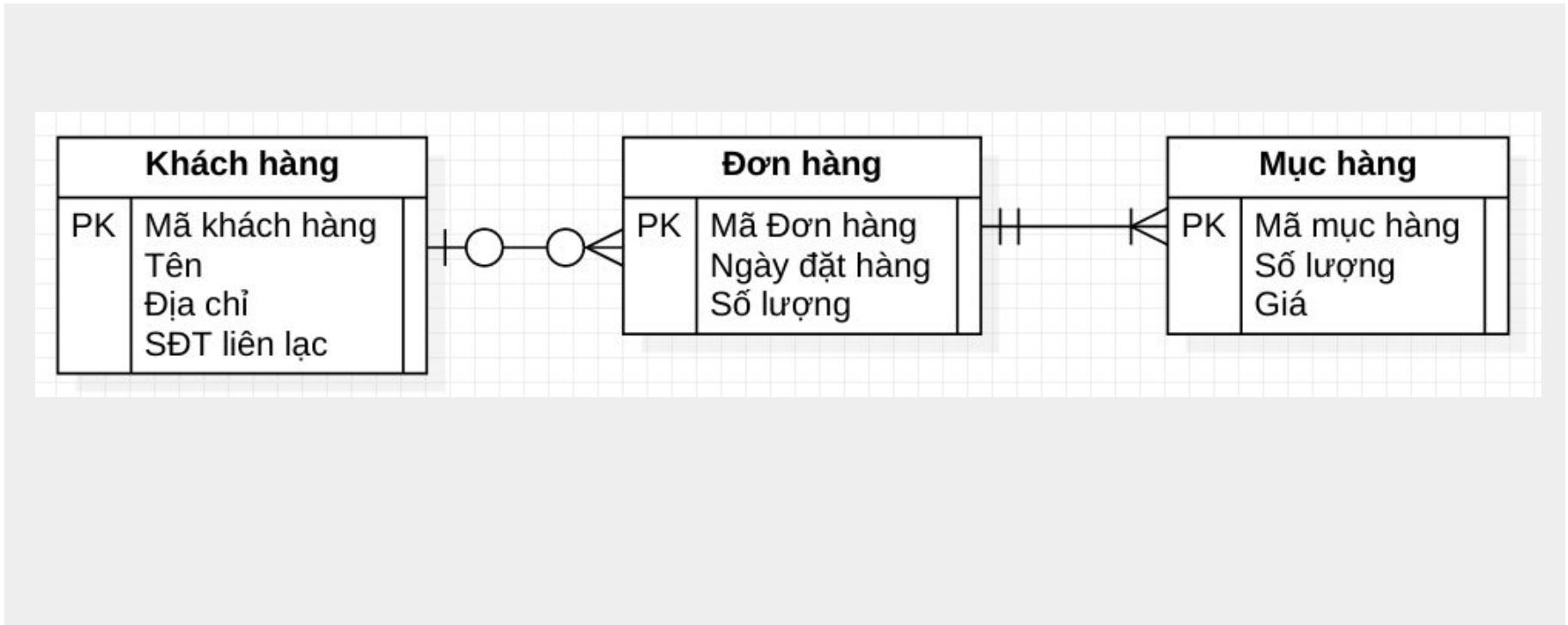
# Cách tiếp cận hướng cấu trúc<sub>(3)</sub>

- Phân tích hướng cấu trúc
  - Sơ đồ luồng dữ liệu (DFD)



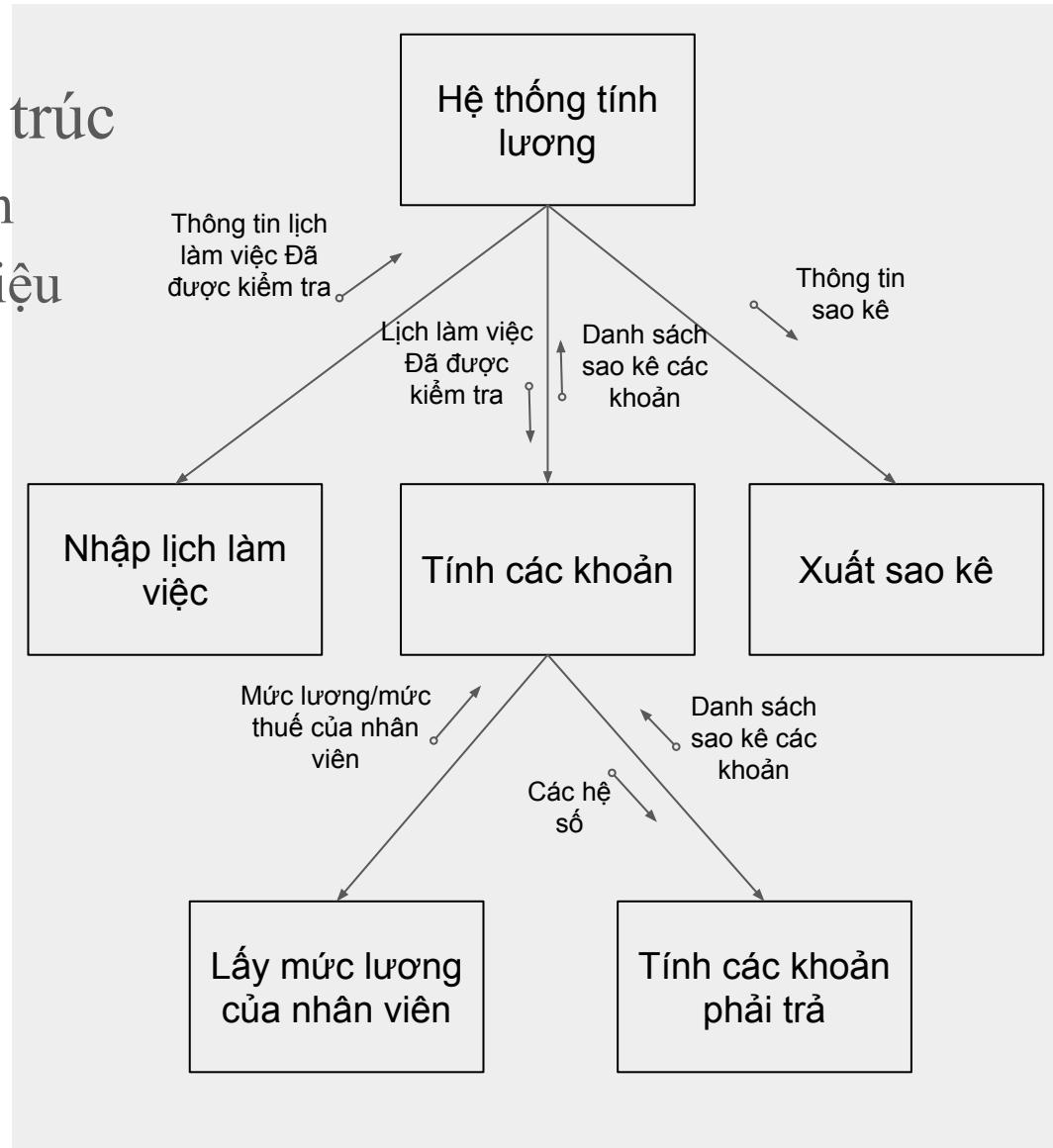
# Cách tiếp cận hướng cấu trúc<sub>(4)</sub>

- Mô hình hóa dữ liệu với sơ đồ thực thể-liên kết



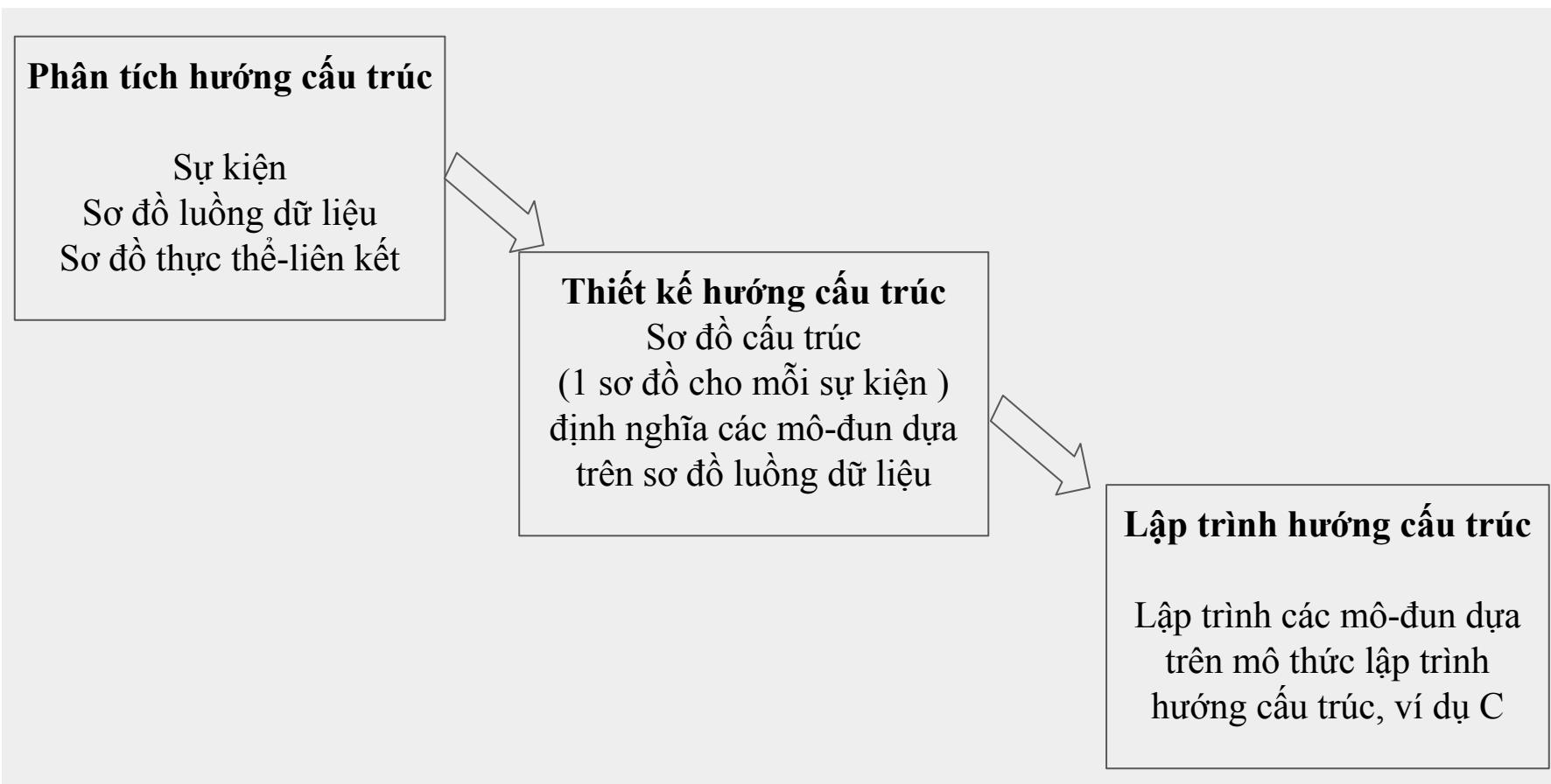
# Cách tiếp cận hướng cấu trúc<sub>(5)</sub>

- Thiết kế hướng cấu trúc
  - Biểu diễn các thành phần cùng với dữ liệu liên quan.



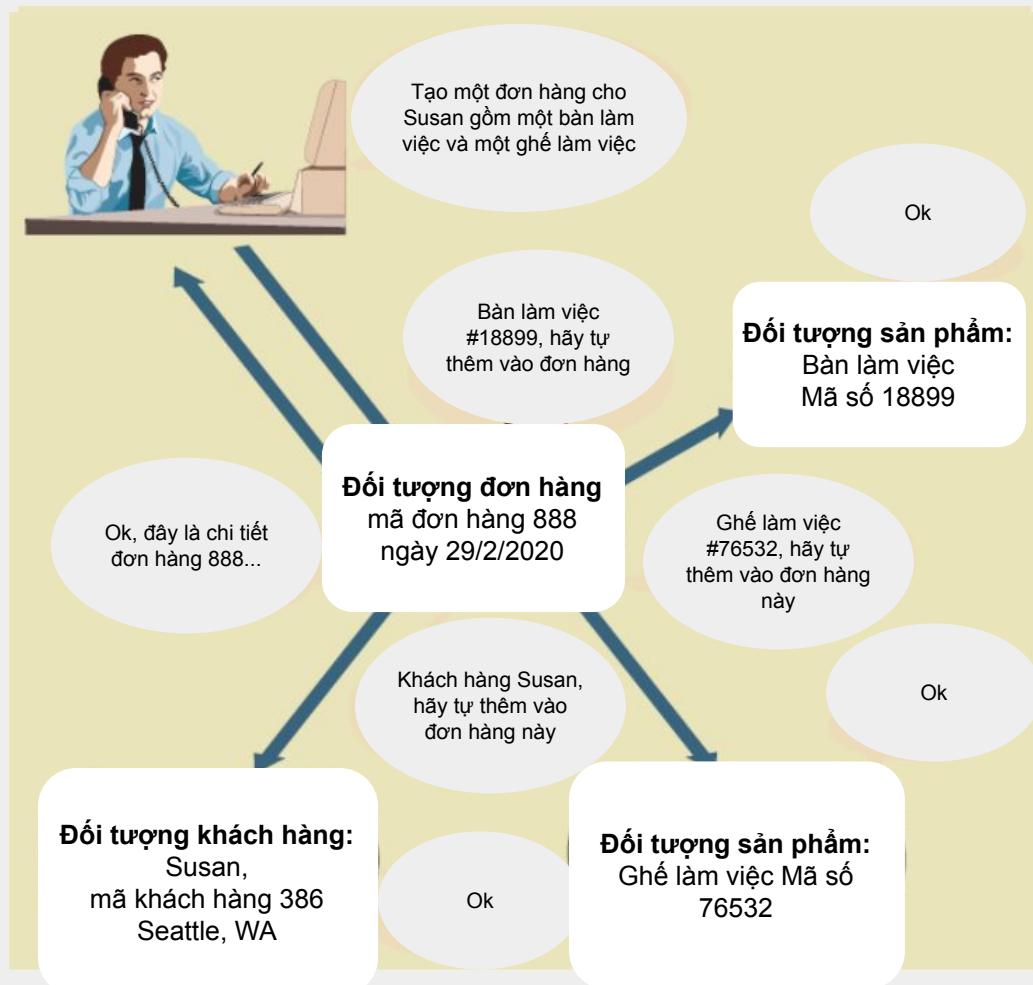
# Cách tiếp cận hướng cấu trúc<sub>(6)</sub>

- Kết hợp các thành phần

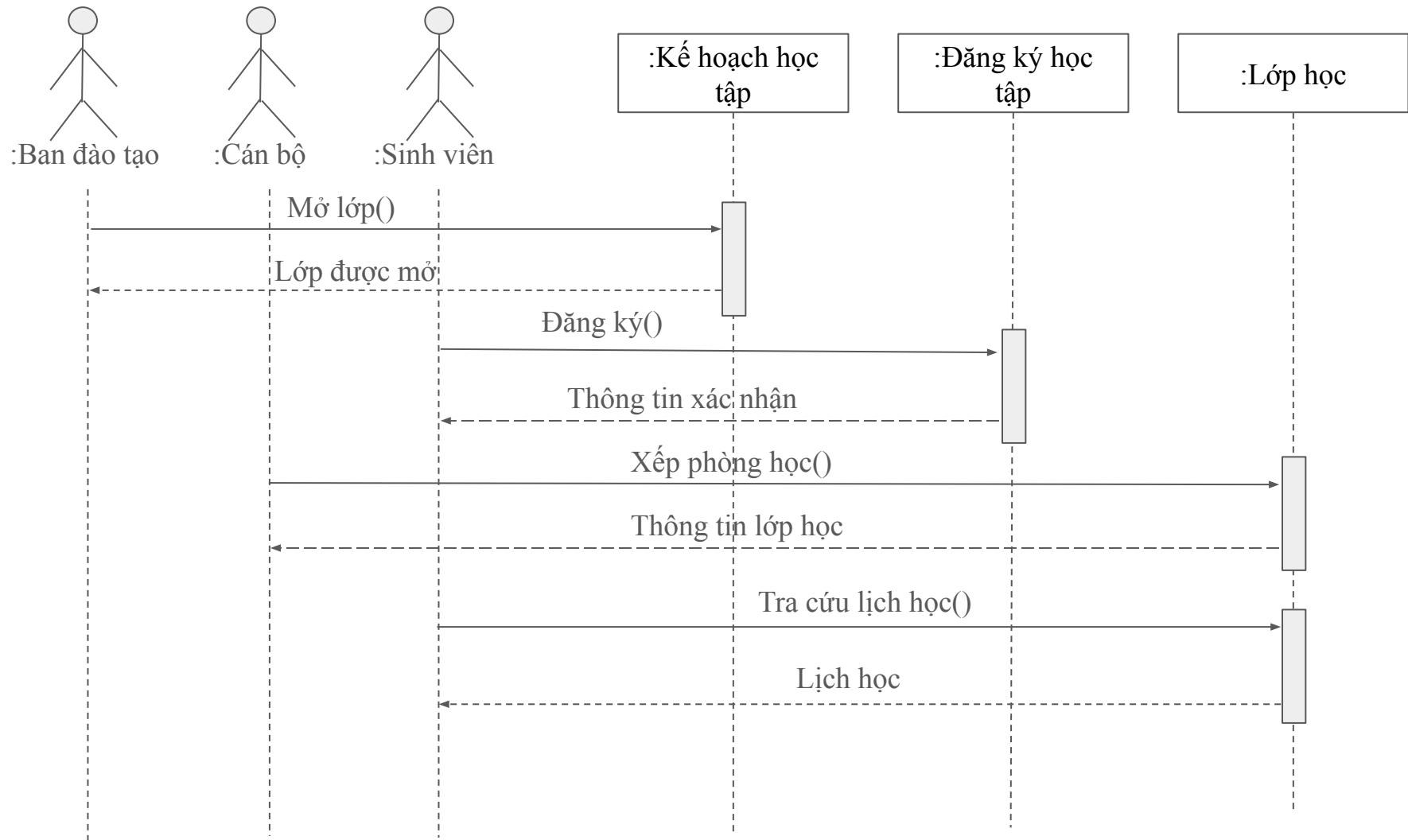


# Cách tiếp cận hướng đối tượng

- Gần với tư duy hợp tác và phân chia công việc
- Các đối tượng tương tác để hoàn thành công việc



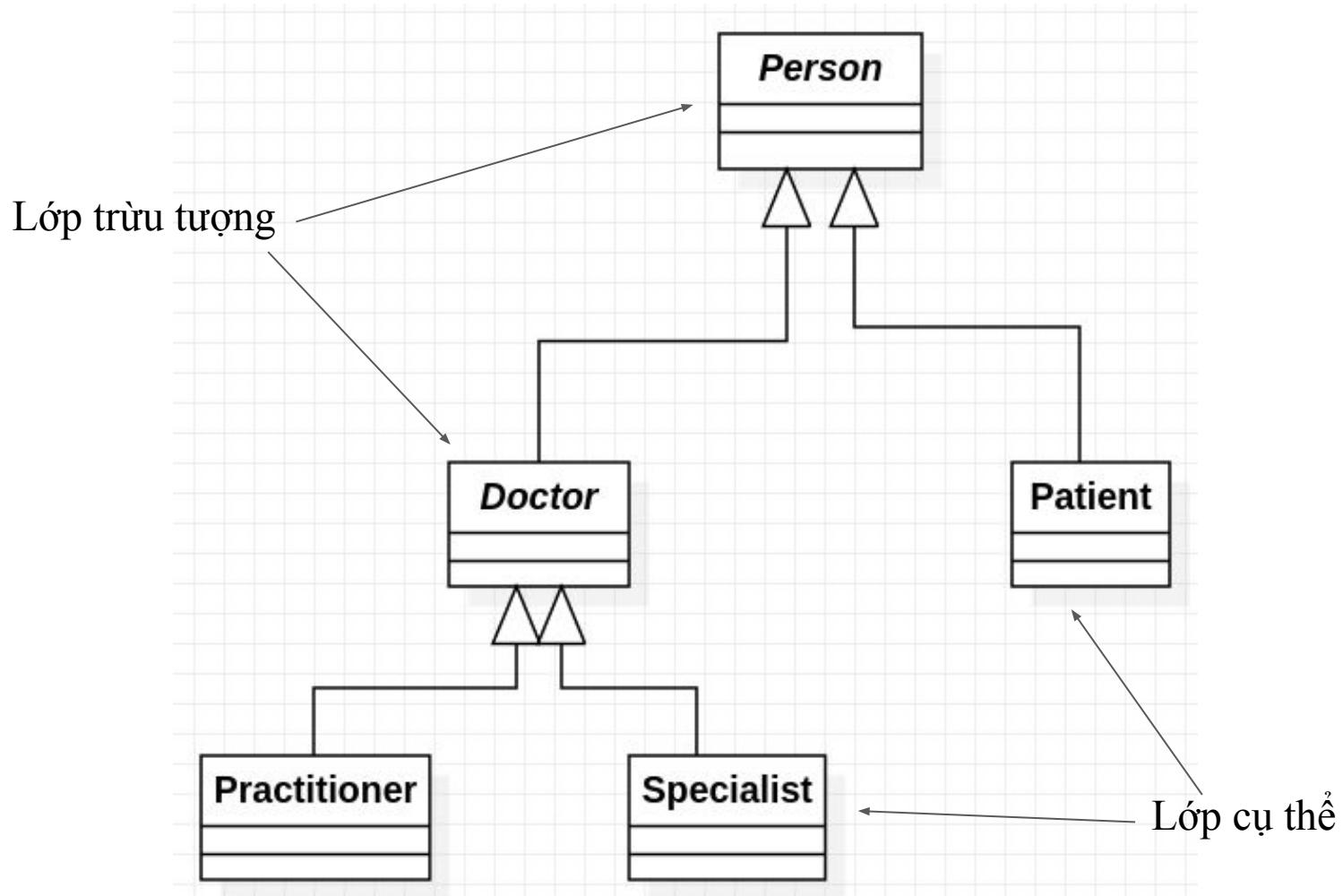
# Ví dụ mô hình hướng đối tượng



# Một số thành phần và tính năng cơ bản

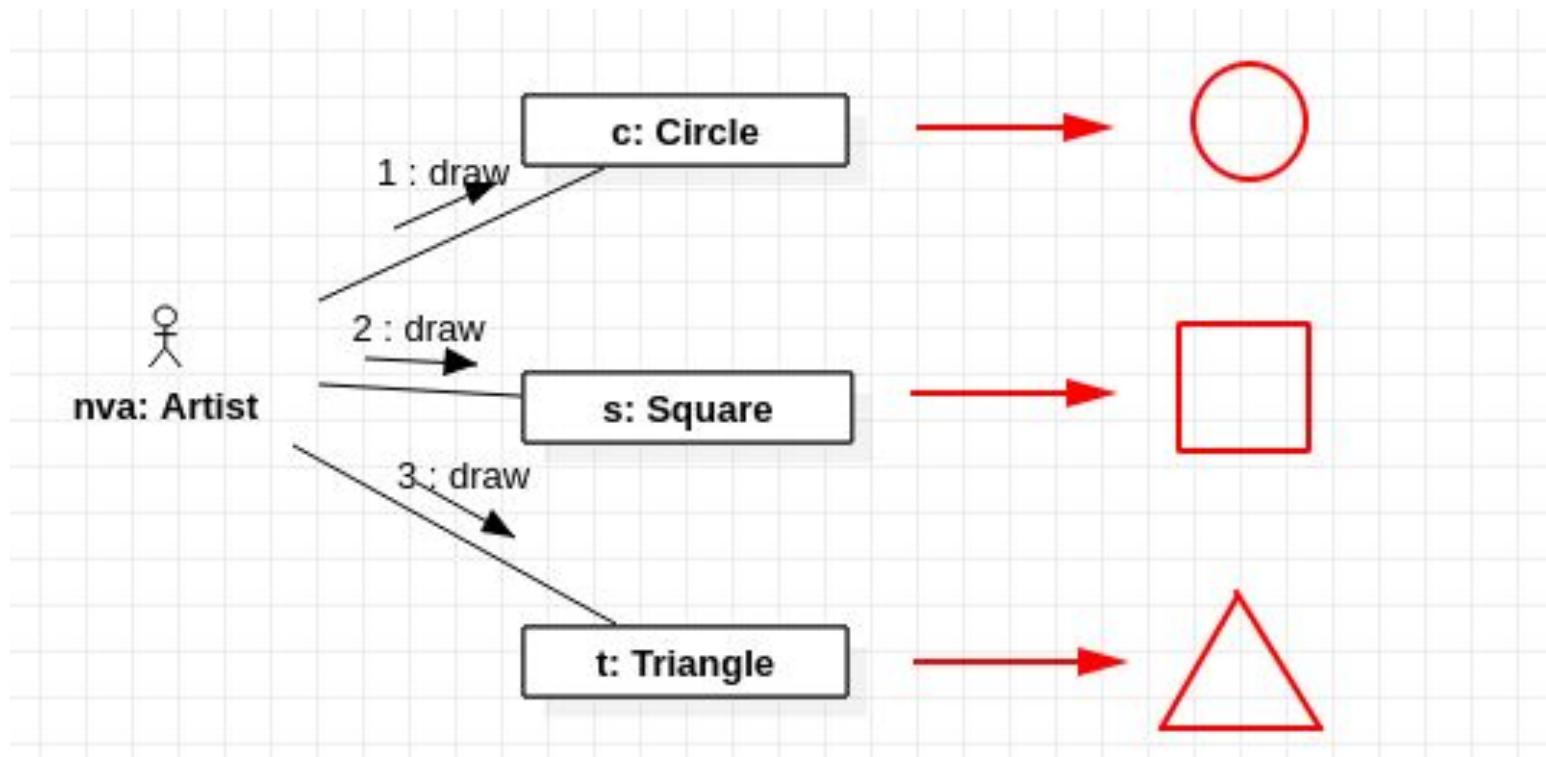
- Đối tượng và lớp
- Thông điệp và phương thức
- Đóng gói và ẩn dữ liệu
- Ké thừa
- Đa hình và phân giải động

# Kế thừa



*Lớp dưới có các phần tử của lớp trên*

# Đa hình



*Một thông điệp có thể được xử lý theo nhiều cách*

# Phân tích & Thiết kế Hệ thống theo phương pháp Hướng Đối Tượng

Object-Oriented Systems Analysis and Design (OOSAD)

- Thường sử dụng :
  - Ngôn ngữ mô hình hóa hướng đối tượng
    - UML - Unified Modeling Language
    - Thiết kế tương tác: IFML
    - Mô tả ràng buộc: OCL
    - v.v..
  - và tiến trình hợp nhất (Unified Process)
- Các đặc điểm cơ bản của OOSAD
  - Dẫn dắt bởi ca sử dụng
  - Kiến trúc khung
  - Lặp và tăng dần

[Grady Booch, Ivar Jacobson, and James Rumbaugh] <sub>30</sub>

# Dẫn dắt bởi ca sử dụng

## *Use-Case Driven*

- Ca sử dụng là phương tiện chính để mô hình hóa hành vi của hệ thống
  - Đóng gói các tương tác giữa người dùng và hệ thống.
  - *Mỗi ca sử dụng thường tập trung vào 1 quy trình nghiệp vụ cơ bản.*
- Có thể tập trung vào 1 hoạt động ở 1 thời điểm.

# Kiến trúc khung

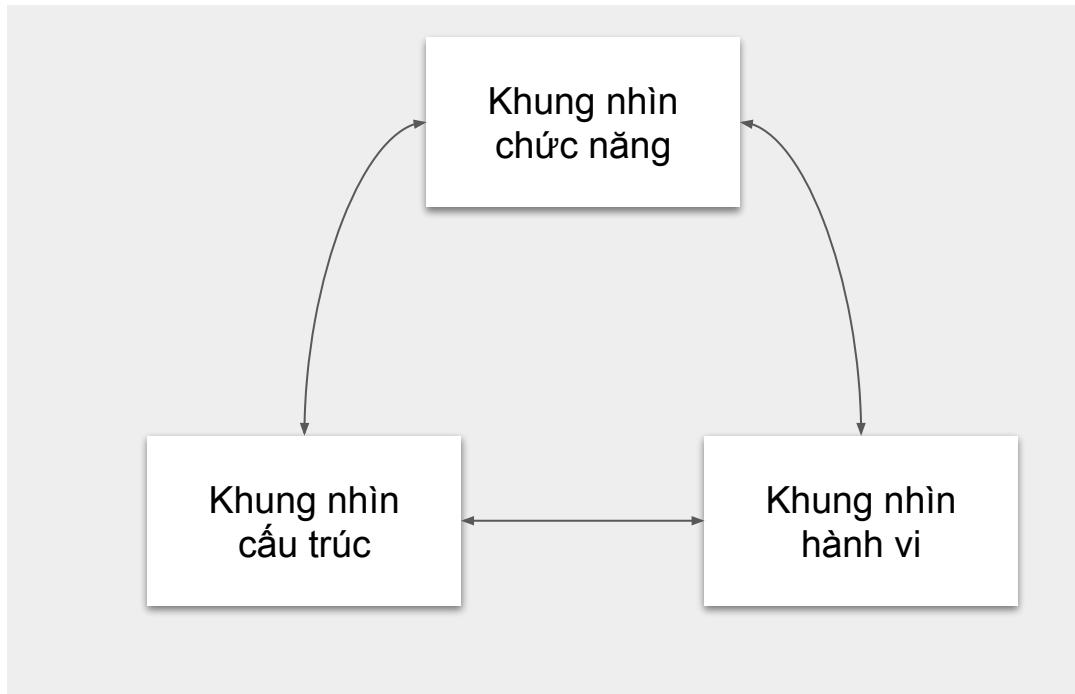
## *Architecture Centric*

- Khung kiến trúc được áp dụng cho đặc tả, xây dựng và tài liệu của hệ thống.
- Cách tiếp cận hướng đối tượng nên hỗ trợ tối thiểu 3 khung nhìn kiến trúc riêng biệt nhưng có quan hệ mật thiết:
  - Khung nhìn chức năng (từ bên ngoài): Tập trung vào góc nhìn người dùng
  - Khung nhìn cấu trúc (tĩnh): Tập trung vào các thuộc tính, phương thức, lớp và các mối quan hệ
  - Khung nhìn hành vi (động): Tập trung vào các thông điệp giữa các lớp và các hành vi phát sinh

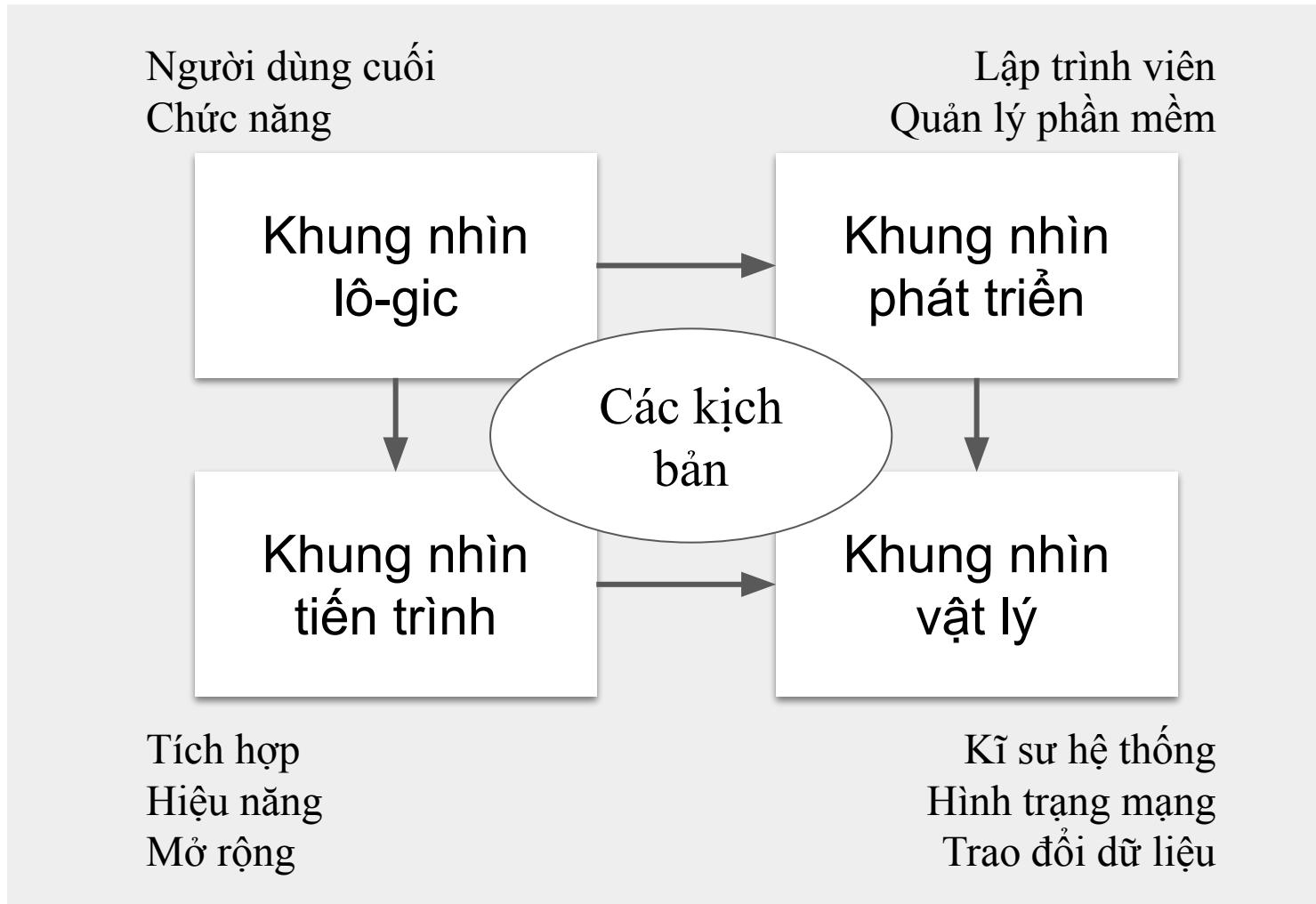
# Lắp và tăng dần

## *Iterative and Incremental*

- Trải qua kiểm thử và tinh chỉnh liên tục
- Sau mỗi bước người phân tích càng hiểu rõ hơn về hệ thống



# Mô hình kiến trúc 4 + 1 khung nhìn



[Philippe Kruchten]

# Các khung nhìn

- Các khung nhìn đáp ứng các nhu cầu thông tin khác nhau của các đối tượng người dùng khác nhau.
- Liên hệ với mô hình 4 + 1:
  - Các kịch bản sử dụng hệ thống được mô tả trong các ca sử dụng và được sử dụng trong các giai đoạn sau đó.
    - Mô hình hóa chức năng.
  - Nhiều mô hình được xây dựng trong giai đoạn Phân tích được coi là các khung nhìn lô-gic.
  - Nhiều mô hình được xây dựng trong thiết kế được coi là khung nhìn tiến trình, khung nhìn phát triển, hoặc khung nhìn vật lý.
    - *Đối với khung nhìn tiến trình - Tuy chúng ta ít sử dụng các khái niệm tiến trình (process) và luồng (thread), tuy nhiên hệ thống được chia nhỏ thành các thành phần, các gói, các cấu phần.*

# Các lợi ích của OOSAD

- Chia 1 hệ thống phức tạp thành nhiều cấu phần => dễ quản lý hơn
- Làm việc với từng cấu phần => tập trung hơn
- Sử dụng các vòng lặp => kịp thời thực hiện các thay đổi

# Nội dung

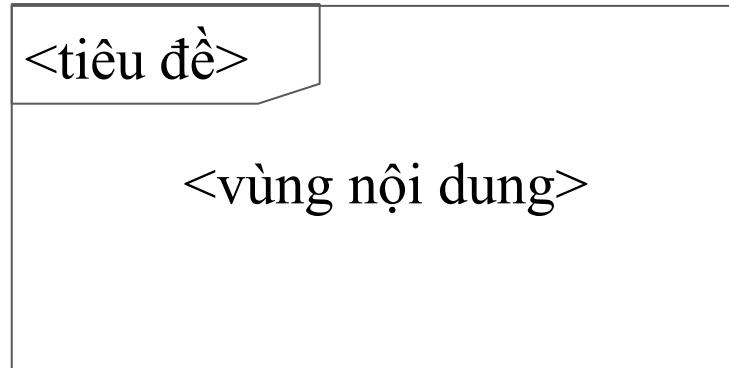
- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu



# Ngôn ngữ mô hình hóa hợp nhất

- Unified Modeling Language (UML)
- Ngôn ngữ trực quan, cung cấp nhiều sơ đồ hữu ích cho các hoạt động phân tích & thiết kế hệ thống
  - Nhưng không phải tất cả,
- Quy chuẩn OMG (từ 1997) và ISO (từ 2005)
  - [omg.org](http://omg.org)
  - [iso.org](http://iso.org)
  - [uml.org](http://uml.org)
- Có nhiều phiên bản

# Khung và tiêu đề sơ đồ



- Mỗi sơ đồ có thể được đặt trong 1 khung có tiêu đề
  - *(Không bắt buộc phải đặt trong khung)*
- Tiêu đề trong trường hợp sử dụng khung có định dạng:
  - [<phân loại>][<tên>][<tham số>]
  - Có thể mô tả không gian tên của vùng nội dung
  - ... hoặc thành phần của mô hình sở hữu các thành phần được biểu diễn trong vùng nội dung.

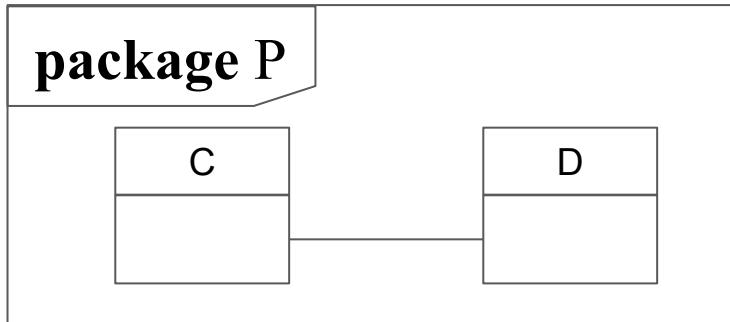
# Phân loại sơ đồ

Các loại sơ đồ sau được sử dụng trong tiêu đề khung:

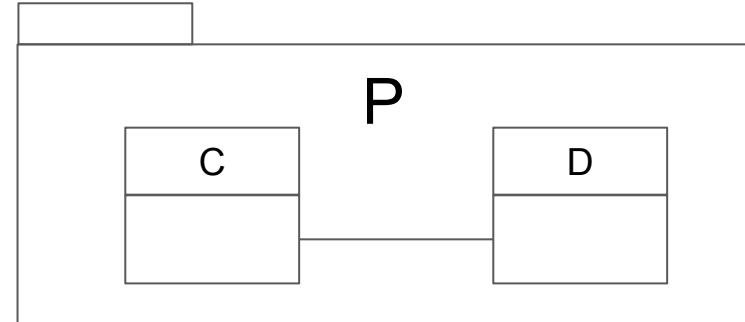
Tên tiếng Anh	Tên rút gọn	Tên tiếng Việt
activity	act	hoạt động
class	--	lớp
component	cmp	thành phần
deployment	dep	triển khai
interaction	sd	tương tác
package	pkg	gói
state machine	stm	máy trạng thái
use case	uc	ca sử dụng

# Ví dụ 1.1. Khung và tiêu đề

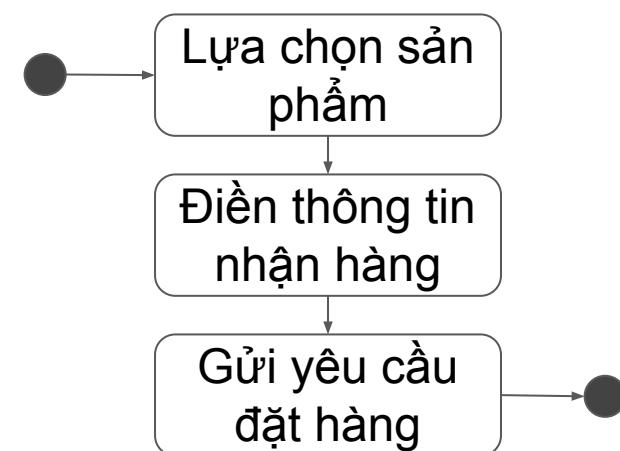
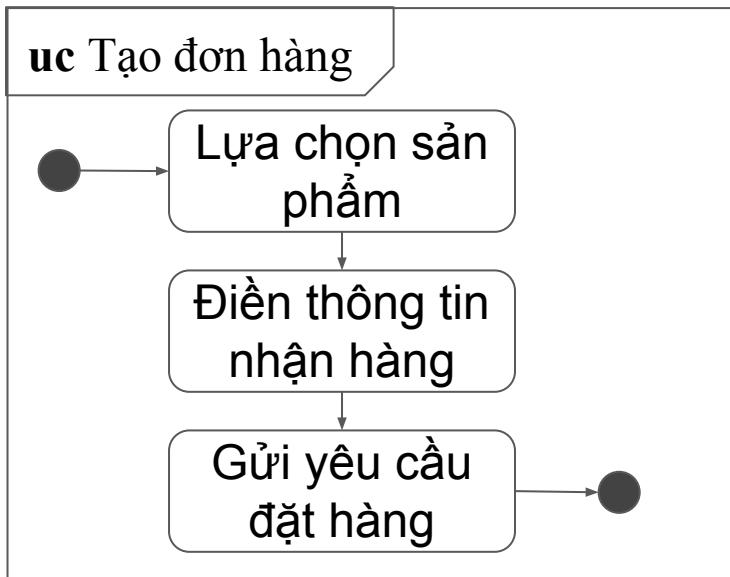
## Sử dụng khung



## Không sử dụng khung



*Sơ đồ lớp với các lớp trong gói P*



*Sơ đồ hoạt động của 1 kịch bản tương tác trong CSD Tạo đơn hàng*

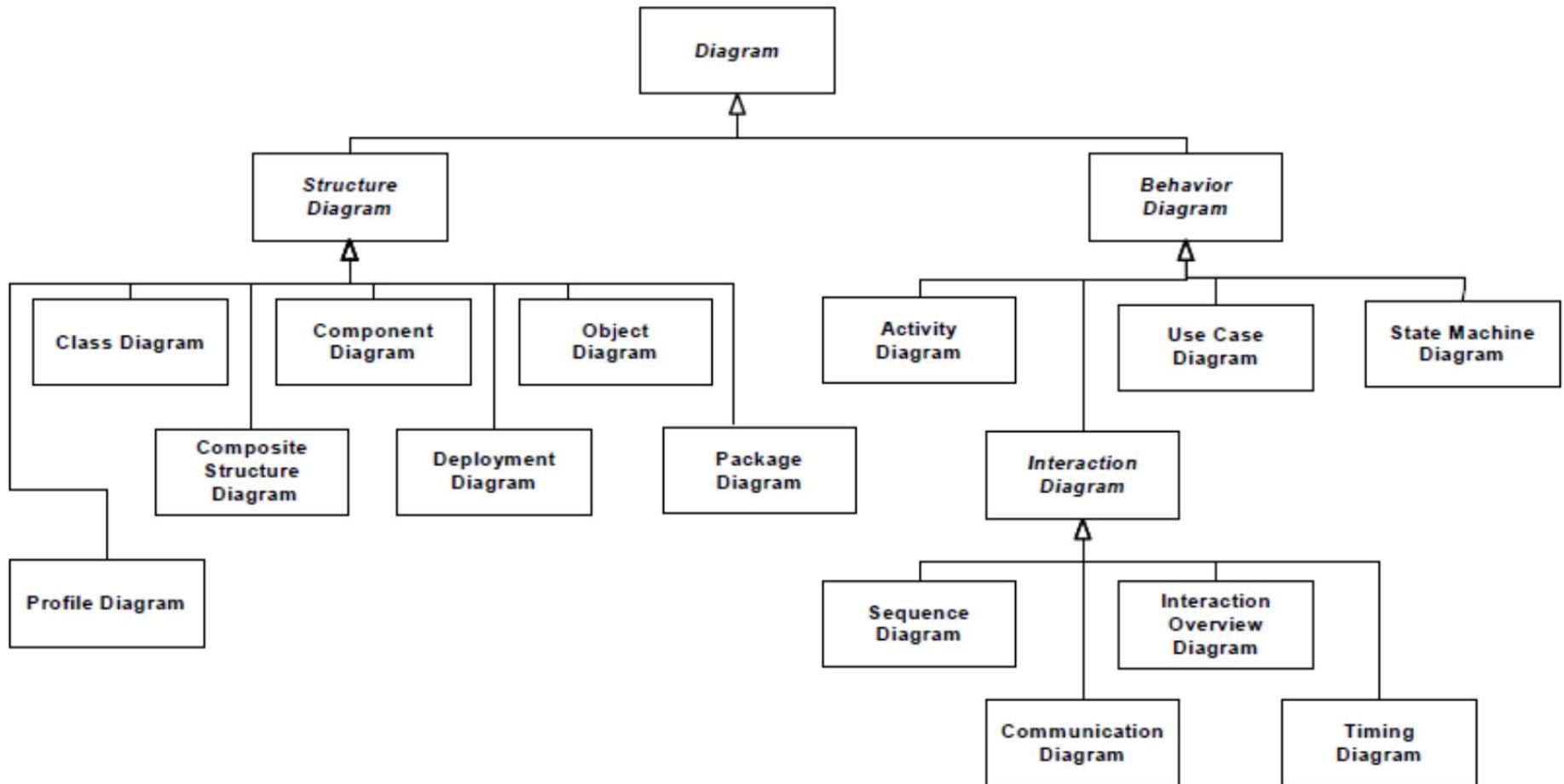
# Các loại sơ đồ UML

Loại sơ đồ	Biểu diễn	Tùy phiên bản
Lớp/Class	Lớp, kiểu, giao diện, và các mối quan hệ giữa các thành phần này.	UML 1.x
Đối tượng/Object	Các đối tượng/thực bản của các lớp trong sơ đồ lớp	UML 1.x
Gói/Package	Cấu trúc phân cấp của các nhóm lớp và các thành phần	UML 2.x
Kết cấu/Composite structure	Biểu diễn bên trong của 1 lớp hoặc 1 thành phần, và có thể mô tả các mối quan hệ của lớp trong 1 ngữ cảnh xác định	UML 2.x
Thành phần/Component	Các thành phần quan trọng trong hệ thống và các giao diện được sử dụng để tương tác với các thành phần khác	UML 1.x, được cập nhật trong UML 2.0
Triển khai/Deployment	Cách hệ thống sẽ được triển khai trong môi trường thực tế	UML 1.x
Hồ sơ/Profile	Mở rộng và tùy chỉnh UML với các thành phần và thuộc tính mới	UML 2.x

# Các loại sơ đồ UML<sub>(2)</sub>

Loại sơ đồ	Biểu diễn	Tùy phiên bản
Cơ sở dụng/Use Case	Tương tác giữa hệ thống với người dùng và các hệ thống ngoại.	UML 1.x
Hoạt động/Activity	Các hoạt động tuần tự và song song diễn ra trong hệ thống	UML 1.x
Máy trạng thái/State Machine	Trạng thái của đối tượng trong suốt thời gian tồn tại và các sự kiện làm thay đổi trạng thái của đối tượng	UML 1.x
Tuần tự/Sequence	Tương tác giữa các đối tượng trong trường hợp thứ tự tương tác có ý nghĩa quan trọng.	UML 1.x
Giao tiếp/Communication	Cách các đối tượng tương tác và các liên kết cần thiết để thực hiện các tương tác	Đổi tên sơ đồ cộng tác/collaboration, UML 1.x
Tính giờ/Timing	Tương tác giữa các đối tượng trong trường hợp thời gian là yếu tố quan trọng	UML 2.x
Tương tác tổng quan/Interaction Overview	Được sử dụng để tổng hợp các sơ đồ tuần tự, giao tiếp, và tính giờ để tìm tương tác quan trọng diễn ra trong hệ thống	UML 2.x

# Phân loại sơ đồ UML 2.5



# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu

# Tạo lập dự án

Dự án thường được phát sinh từ các nhu cầu nghiệp vụ

- Có thể được xác định bởi người làm nghiệp vụ
- Có thể được xác định bởi người làm CNTT
- Hoặc (tốt hơn nữa) được xác định đồng thời bởi cả 2 bên.

# Tạo lập dự án<sub>(2)</sub>

Các dự án phát triển Hệ thống thông tin thường:

- Khai thác 1 tiềm năng
  - Bước đi chiến lược
  - Đem lại lợi thế cạnh tranh
- Giải quyết 1 vấn đề:
  - Hệ thống hiện tại liên tục phát sinh sự cố vận hành
  - Nhu cầu người dùng không được đáp ứng
- Đáp ứng 1 chỉ thị từ bên ngoài
  - Tạo lập các biểu mẫu báo cáo theo quy định của pháp luật

# Tạo lập dự án<sub>(3)</sub>

Có thể chia các giá trị mà dự án sẽ tạo ra thành 2 nhóm:

- Giá trị hữu hình
  - Có thể định lượng và đo đếm trực tiếp được
  - Ví dụ: giảm 2% trong chi phí vận hành
- Giá trị vô hình
  - Chúng ta biết nó sẽ tạo ra giá trị và tiết kiệm thời gian, nhưng chúng ta có thể không định lượng được hoặc đo được những lợi ích của nó
  - Ví dụ: Làm khách hàng hài lòng hơn; nâng cao trình độ chuyên môn trong bộ phận IT; v.v.

# Đánh giá tính khả thi

- Dự án này có khả thi hay không?
  - Các rủi ro là gì?
  - Có tránh được những rủi ro đó hay không?
- Các thành phần chính:
  - Tính khả thi kỹ thuật (Có đủ khả năng xây dựng nó?)
  - Tính khả thi kinh tế (Có lợi khi xây dựng nó?)
  - Tính khả thi tổ chức (Mọi người sẽ sử dụng nó?)

# Tính khả thi kỹ thuật

Xác định rủi ro trong những mảng sau:

- Mảng chức năng: Những người phân tích có kinh nghiệm trong lĩnh vực nghiệp vụ này không?
- Công nghệ: Càng ít kinh nghiệm với công nghệ được lựa chọn thì rủi ro càng cao
- Quy mô dự án: Dự án càng lớn thì càng có nhiều rủi ro
- Tính tương thích: Càng khó tích hợp thì rủi ro càng cao

# Tính khả thi kinh tế (Phân tích Chi phí-Lợi ích)

- Xác định các khoản chi phí và các lợi ích
- Gán giá trị cho các khoản chi phí và các lợi ích
- Xác định dòng tiền
- Xác định giá trị bằng 1 hoặc nhiều phương pháp
  - Giá trị thuần hiện tại (NPV)
  - Tỷ lệ hoàn vốn đầu tư (ROI)
  - Điểm hòa vốn

# Tính khả thi tổ chức

- Người dùng sẽ đón nhận hệ thống?
- Dự án có thuận theo chiến lược chung?
- Phân tích các bên liên quan:
  - Người ủng hộ dự án
  - Ban quản lý
  - Người dùng hệ thống
  - v.v..

# Lựa chọn dự án

- Các dự án được phê duyệt, từ chối hoặc tạm hoãn dựa trên các kết quả đánh giá thực trạng
- Các dự án được lựa chọn được đưa vào quy trình quản lý dự án

# Nội dung

- Một số khái niệm cơ bản
- Vòng đời phát triển hệ thống (SDLC)
- Hai cách tiếp cận phát triển hệ thống
- Ngôn ngữ mô hình hóa hợp nhất (UML)
- Môi trường & Nhu cầu
- Xác định yêu cầu



# Tổng quan về xác định yêu cầu

- Mục đích: Chuyển đổi các nhu cầu nghiệp vụ thành các yêu cầu chi tiết hơn.
- Một bước rất quan trọng có tầm ảnh hưởng trên toàn bộ SDLC
- Thay đổi liên quan đến yêu cầu có thể dễ dàng được thực hiện ở giai đoạn bắt đầu dự án nhưng càng về sau càng khó thực hiện
- Hầu hết ( $>50\%$ ) dự án thất bại có nguyên nhân liên quan đến yêu cầu
- Tính lặp và tăng dần hiệu quả bởi vì:
  - Mỗi vòng lặp xác định và triển khai từng gói yêu cầu nhỏ
  - Hệ thống phát triển tăng dần theo thời gian

# Khái niệm yêu cầu

- Yêu cầu là gì?
  - Yêu cầu cho biết những gì hệ thống phải làm hoặc đặc điểm mà nó phải có,
  - cung cấp thông tin cần thiết cho các chuỗi công việc tiếp theo,
  - là cơ sở của những đặc tả kỹ thuật về cách triển khai hệ thống,
  - xác định phạm vi hệ thống.
- Các loại yêu cầu:
  - Chức năng: Liên quan đến xử lý hoặc dữ liệu
  - Phi chức năng: Liên quan đến hiệu năng hoặc đặc điểm sử dụng
- Các yêu cầu thường được liệt kê theo danh mục
- Có thể được đánh giá mức ưu tiên

# Khái niệm yêu cầu<sub>(2)</sub>

- Yêu cầu chức năng có liên quan chặt chẽ với các quy trình nghiệp vụ và các quy định của tổ chức, và được tập trung nghiên cứu ở pha phân tích
  - Được mô tả chủ yếu bằng các ca sử dụng: Đặc tả chi tiết bằng lời và sơ đồ.
- Yêu cầu phi chức năng được tập trung nghiên cứu ở pha thiết kế và có thể tiếp tục được phân loại thành:
  - (Các yêu cầu) vận hành: Các đặc điểm liên quan đến cách sử dụng
  - Độ tin cậy: Các lỗi có thể phát sinh và khả năng khắc phục
  - Hiệu năng: Tải và thời gian phản hồi
  - Bảo mật: Kiểm soát truy cập và bảo vệ dữ liệu.

# Các thuật ngữ

- Yêu cầu chức năng: Functional requirements
- Yêu cầu phi chức năng: Non-functional
  - Vận hành/sử dụng: Usability
  - Độ tin cậy: Reliability
  - Hiệu năng: Performance
  - Bảo mật: Security
- Phân loại FURPS (Functions, Usability, Reliability, Performance, Security Requirements).

# Ví dụ 2.1. Các yêu cầu hệ thống

- Yêu cầu chức năng:
  - Tạo phiếu mượn sách (trong thư viện)
  - Tra cứu ghế trống (trong rạp chiếu phim)
- Yêu cầu phi chức năng:
  - Vận hành:
    - Nhân viên sử dụng máy tính bảng để phục vụ gọi món tại bàn.
  - Độ tin cậy:
    - Thông tin được bảo toàn trong trường hợp phát sinh sự cố đột ngột.
  - Hiệu năng:
    - Có khả năng đáp ứng yêu cầu của 1000 người dùng sử dụng đồng thời
    - Thời gian phản hồi cho mỗi thao tác không quá 0.5 s.
  - Bảo mật
    - Dữ liệu được gửi từ người dùng phải được mã hóa.
    - Sử dụng giao thức HTTPS cho kết nối giữa người dùng và hệ thống.

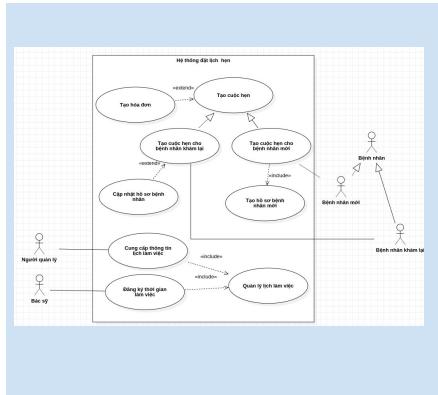
...

# Các mô hình

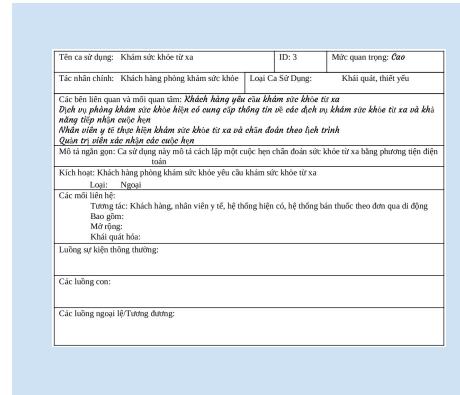
- Mỗi mô hình biểu diễn 1 khía cạnh của hệ thống
  - Người phân tích tạo các mô hình và sử dụng chúng như phương tiện trao đổi hiểu biết hiện tại về hệ thống với khách hàng và những người cùng phát triển hệ thống.
    - Để xác nhận tính đúng đắn và sử dụng cho những hoạt động tiếp theo trong SDLC
    - Có thể được tạo bằng nhiều vòng lặp
- Các mô hình trong phân tích & thiết kế có thể được phân loại thành:
  - Văn bản: Ví dụ, các đặc tả ca sử dụng.
  - Hình vẽ: Ví dụ, các sơ đồ UML
  - Toán học: Ví dụ, công thức tính giá khuyến mãi cho đơn hàng

## Ví dụ 1.2. Các mô hình

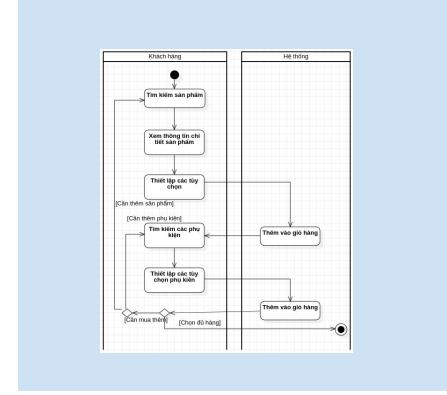
1. Mua ô-tô mới
  2. Bán ô-tô
  3. Bảo dưỡng ô-tô
  4. Thanh toán
  5. Mượn ô-tô



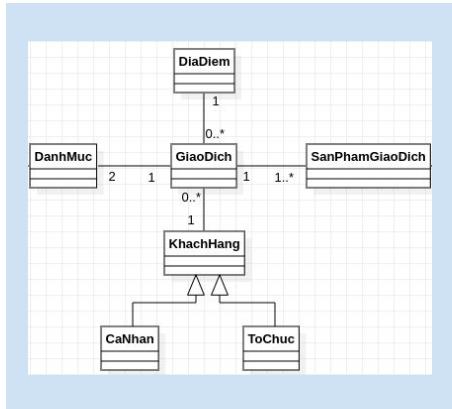
## Danh sách sự kiện



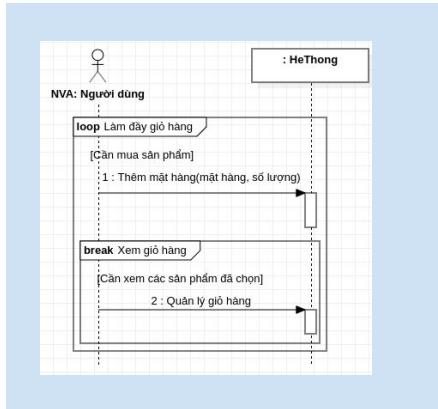
## Sơ đồ ca sử dụng



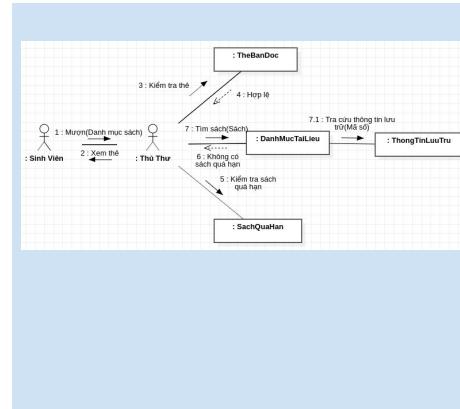
## Đặc tả ca sử dụng



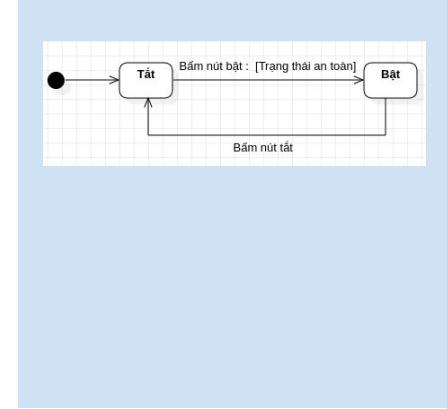
## Sơ đồ lớp



## Sơ đồ tuần tự



## Sơ đồ giao tiếp



# Sơ đồ máy trạng thái

# Kỹ năng xác định yêu cầu

- Kết hợp hiểu biết về CNTT và nghiệp vụ
- Các yêu cầu được xác định tốt nhất khi có người phân tích hệ thống và người làm nghiệp vụ làm việc cùng nhau
- Cần sử dụng các kỹ năng phân tích vấn đề và thu thập thông tin.

# Các khó khăn trong xác định yêu cầu

- Người phân tích có thể không tiếp cận được đúng người dùng
- Các đặc tả yêu cầu có thể không đầy đủ
- Yêu cầu có thể không được phát hiện từ ban đầu
  - Cần cố gắng phát hiện tất cả các yêu cầu (các yêu cầu được phát hiện sau trong tiến trình phát triển thường khó kết hợp)
- Có thể gặp khó khăn trong kiểm tra và đánh giá các yêu cầu.

# Tổng quan về các chiến lược phân tích vấn đề

- Phân tích nguyên nhân gốc - Tập trung vào nguyên nhân của 1 vấn đề - không phải cách giải quyết nó
  - Tìm hiểu nguyên nhân của các vấn đề và xây dựng giải pháp sau khi các nguyên nhân đã được làm rõ
- Phân tích thời lượng - Xác định thời gian cần thiết để hoàn thành mỗi bước trong 1 quy trình nghiệp vụ
  - So sánh với tổng thời gian cần cho toàn bộ tiến trình
  - Khác biệt lớn cho thấy rằng các vấn đề có thể được giải quyết bằng cách:
    - Kết hợp nhiều bước
    - Thực hiện nhiều bước đồng thời (song song)

# Tổng quan về các chiến lược phân tích vấn đề<sub>(2)</sub>

- Chi phí dựa trên hoạt động - Tương tự như thời lượng nhưng áp dụng cho các chi phí
- Đánh giá gián tiếp - Tham khảo quy trình tương tự ở môi trường khác.
- Phân tích đầu ra - Kết quả cuối cùng mà khách hàng muốn đạt được.
- Phân tích công nghệ - Giả định áp dụng công nghệ mới vào quy trình nghiệp vụ và xác định các lợi ích thu được.
- Loại bỏ hoạt động - Thủ loại bỏ từng hoạt động trong quy trình nghiệp vụ, và quan sát tiến trình trong điều kiện đó, tìm hiểu các hiệu ứng mà nó tạo ra.

# Tổng quan về các kỹ thuật thu thập thông tin

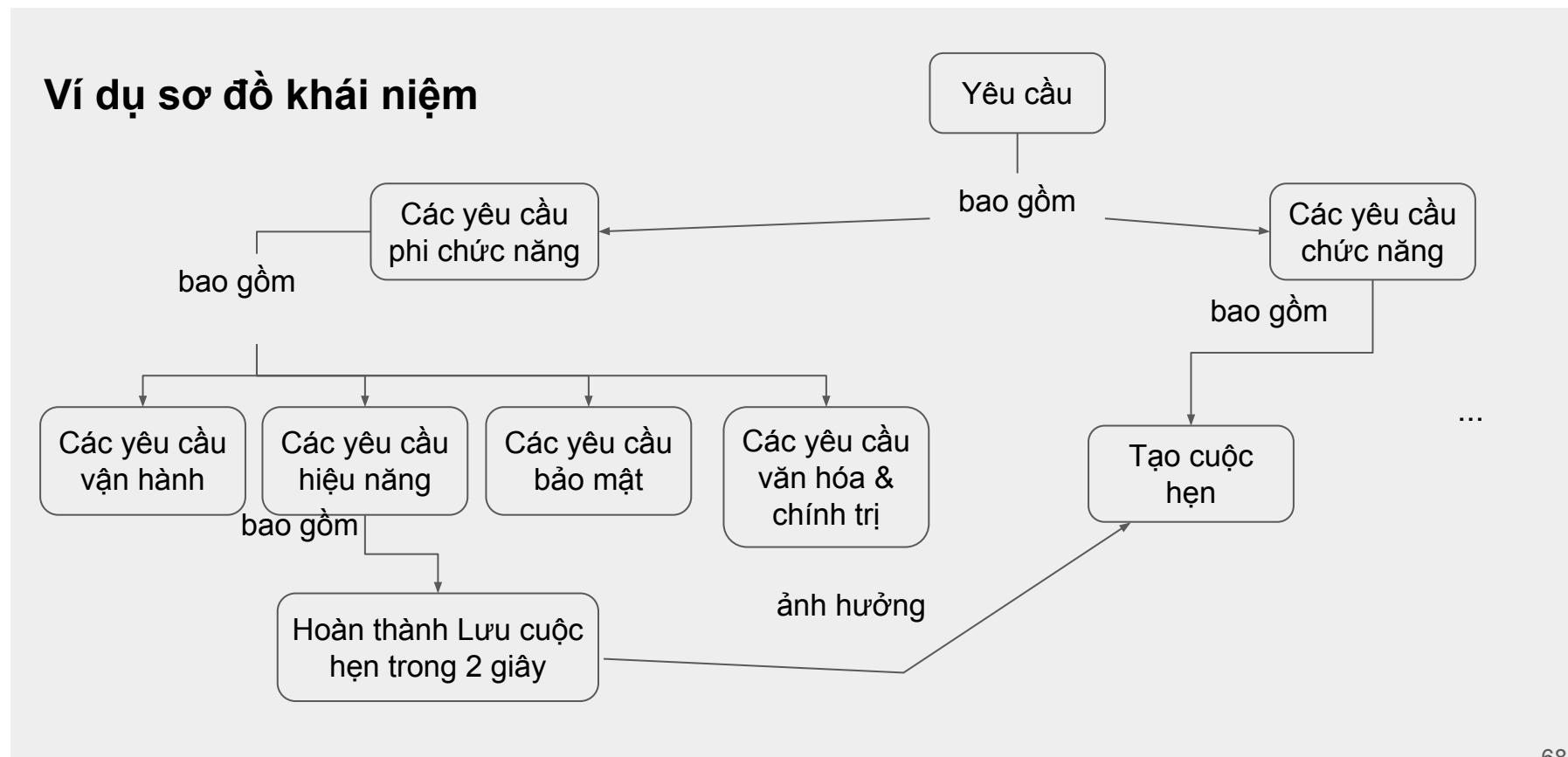
- Phỏng vấn - Kỹ thuật thông dụng nhất - hỏi ai đó khi cần biết về điều gì đó
- Hội thảo liên kết (JAD) - Cuộc họp bao gồm cả người dùng và người phân tích để thảo luận về các vấn đề phát triển hệ thống.
- Khảo sát - Sử dụng 1 tập câu hỏi để thu thập thông tin từ nhiều người.
- Phân tích tài liệu - Tìm hiểu các tài liệu kỹ thuật, các biểu mẫu, báo cáo, các quy định và điều khoản. v.v..
- Quan sát - Quan sát diễn biến thực tế của các hoạt động.

# So sánh các kỹ thuật thu thập thông tin

	Phỏng vấn	Hội thảo liên kết	Khảo sát	Phân tích tài liệu	Quan sát
Loại thông tin	Đang có, cải tiến, sẽ có	Đang có, cải tiến, sẽ có	Đang có, cải tiến	Đang có	Đang có
Độ sâu thông tin	Cao	Cao	Trung bình	Thấp	Thấp
Độ rộng thông tin	Thấp	Trung bình	Cao	Cao	Thấp
Tích hợp thông tin	Thấp	Cao	Thấp	Thấp	Thấp
Người dùng tham gia	Trung bình	Cao	Thấp	Thấp	Thấp
Chi phí	Trung bình	Thấp-Trung bình	Thấp	Thấp	Thấp tới Trung bình

# Các kỹ thuật khác

- Sơ đồ khái niệm
  - Biểu diễn trực quan mối quan hệ giữa các khái niệm
  - Tập trung nguồn lực vào một lượng nhỏ những ý tưởng chính



## Các kỹ thuật khác<sub>(2)</sub>

- Các thẻ câu chuyện & các danh sách nhiệm vụ
  - Mô tả yêu cầu người dùng theo cách tự nhiên nhất

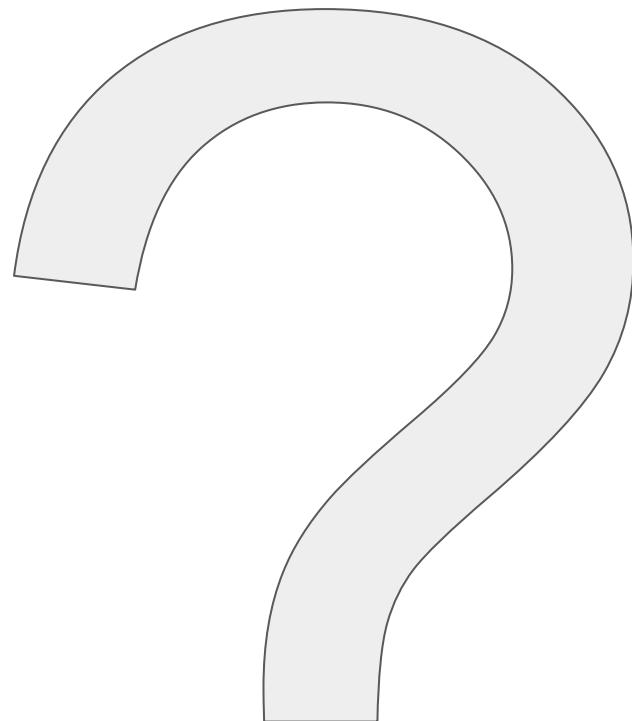
1. Là nhân viên tiếp nhận bệnh nhân, tôi mong muốn có phương tiện lập lịch hẹn để có thể đáp ứng tốt nhất các nhu cầu khám chữa bệnh của bệnh nhân.

2. Tôi mong muốn hệ thống có kết nối không dây với máy in để nhiều máy tính có thể dùng chung 1 máy in mà không cần đến những đường cáp kết nối phức tạp.

V.V...

# Tham khảo

- [1] Satzinger, Jackson, Burd. Systems Analysis and Design in a changing world, 7th edition.
- [2] Dennis, Wixon, Tegarden. Systems Analysis & Design: An object-oriented approach with UML, 5th edition.



# Phân tích thiết kế hệ thống

Soạn bởi: Nguyễn Bá Ngọc

## Chương 2

Hà Nội-2022

# Chương 2.

## Mô hình hóa chức năng

# Nội dung

- Mô hình hóa nghiệp vụ
- Sơ đồ ca sử dụng
- Đặc tả ca sử dụng
- Phương pháp đơn vị ca sử dụng

# Nội dung

- Mô hình hóa nghiệp vụ
- Sơ đồ ca sử dụng
- Đặc tả ca sử dụng
- Phương pháp đơn vị ca sử dụng



# Khái niệm quy trình nghiệp vụ

- Quy trình nghiệp vụ có thể được hiểu như sự kết hợp của các sự kiện, hoạt động, và các quyết định có liên quan lẫn nhau với sự tham gia của nhiều tác nhân và đối tượng, tất cả cùng góp phần vào kết quả đầu ra có giá trị cho ít nhất một người dùng.
- Quy trình nghiệp vụ thường được mô hình hóa như các luồng công việc được thực hiện theo quy tắc để hoàn thành giao dịch hoặc đáp ứng nhu cầu nghiệp vụ.
  - Thường được biểu diễn bằng sơ đồ BPMN hoặc sơ đồ hoạt động.
- Quy trình nghiệp vụ cơ bản (EBP - Elementary Business Process): Nhiệm vụ được thực hiện bởi 1 người ở 1 địa điểm để đáp ứng 1 sự kiện nghiệp vụ, tạo ra giá trị có thể đo được, dẫn đến các giao dịch làm thay đổi dữ liệu của hệ thống.

# Mô hình hóa quy trình nghiệp vụ

- Sơ đồ hoạt động là 1 sơ đồ UML thường được sử dụng để mô hình hóa quy trình nghiệp vụ:
  - Biểu diễn các hoạt động của người dùng và hệ thống, chủ thể thực hiện hoạt động, thông tin được trao đổi giữa các hoạt động và lô-gic thực hiện các hoạt động.

# Các thành phần của sơ đồ hoạt động

- Hành động & Hoạt động

- Công việc được thực hiện trong tiến trình nghiệp vụ
- Được đặt tên bằng động từ và danh từ (ví dụ, tra cứu thông tin khách hàng)
- Hoạt động có thể tiếp tục được chia nhỏ, còn hành động thì không

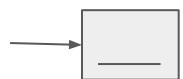
- Luồng điều khiển: Mô hình hóa trình tự thực hiện

- Các nút đối tượng: Có thể được sử dụng để biểu diễn dữ liệu được tạo ra hoặc được sử dụng bởi các hoạt động

- Luồng đối tượng: Mô hình hóa đường đi của các đối tượng

- Đường bơi: Biểu diễn chủ thể thực hiện hoạt động

- ... Các nút điều khiển: Có 7 loại



# Các nút điều khiển

- • **Nút khởi đầu:** Điểm bắt đầu luồng hoạt động
- **Nút kết thúc:** Điểm kết thúc luồng, dừng tất cả các tiến trình
- **Nút kết thúc nhánh:** Kết thúc một nhánh, các nhánh khác vẫn có thể tiếp tục thực hiện
- **Nút quyết định:** Biểu diễn một phép thử để xác định sẽ tiếp tục theo đường dẫn nào dựa trên một điều kiện bảo vệ, các nhánh loại trừ lẫn nhau.
- **Nút hợp nhất:** Kết hợp các nhánh của nút quyết định (các nhánh loại trừ).
- **Thanh chia đồng bộ:** Tách một tiến trình thành nhiều đường dẫn/tiến trình được thực hiện song song
- **Thanh hợp nhất đồng bộ:** Tái hợp các tiến trình của thanh chia đồng bộ (được thực hiện song song).

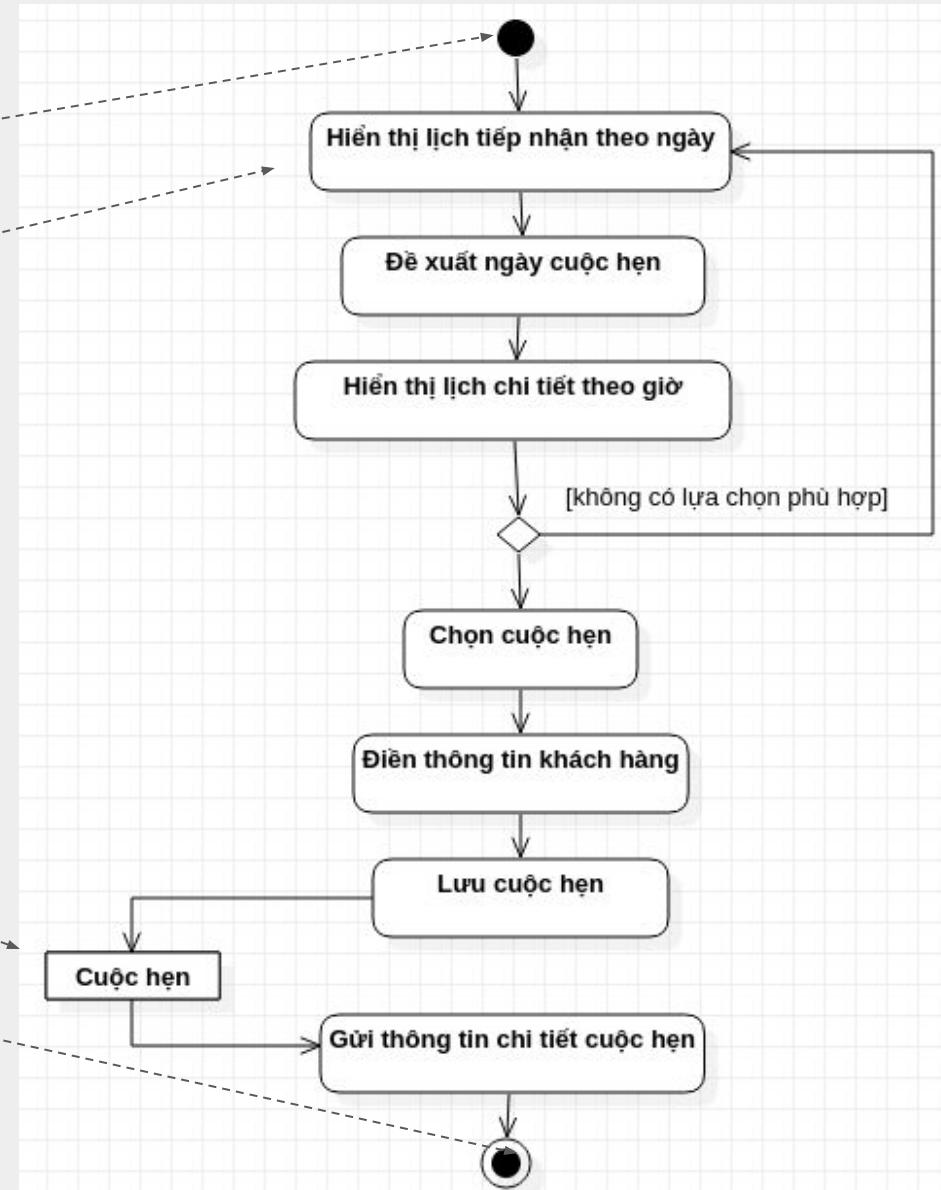
# Ví dụ 2.1. Sơ đồ hoạt động

Bắt đầu

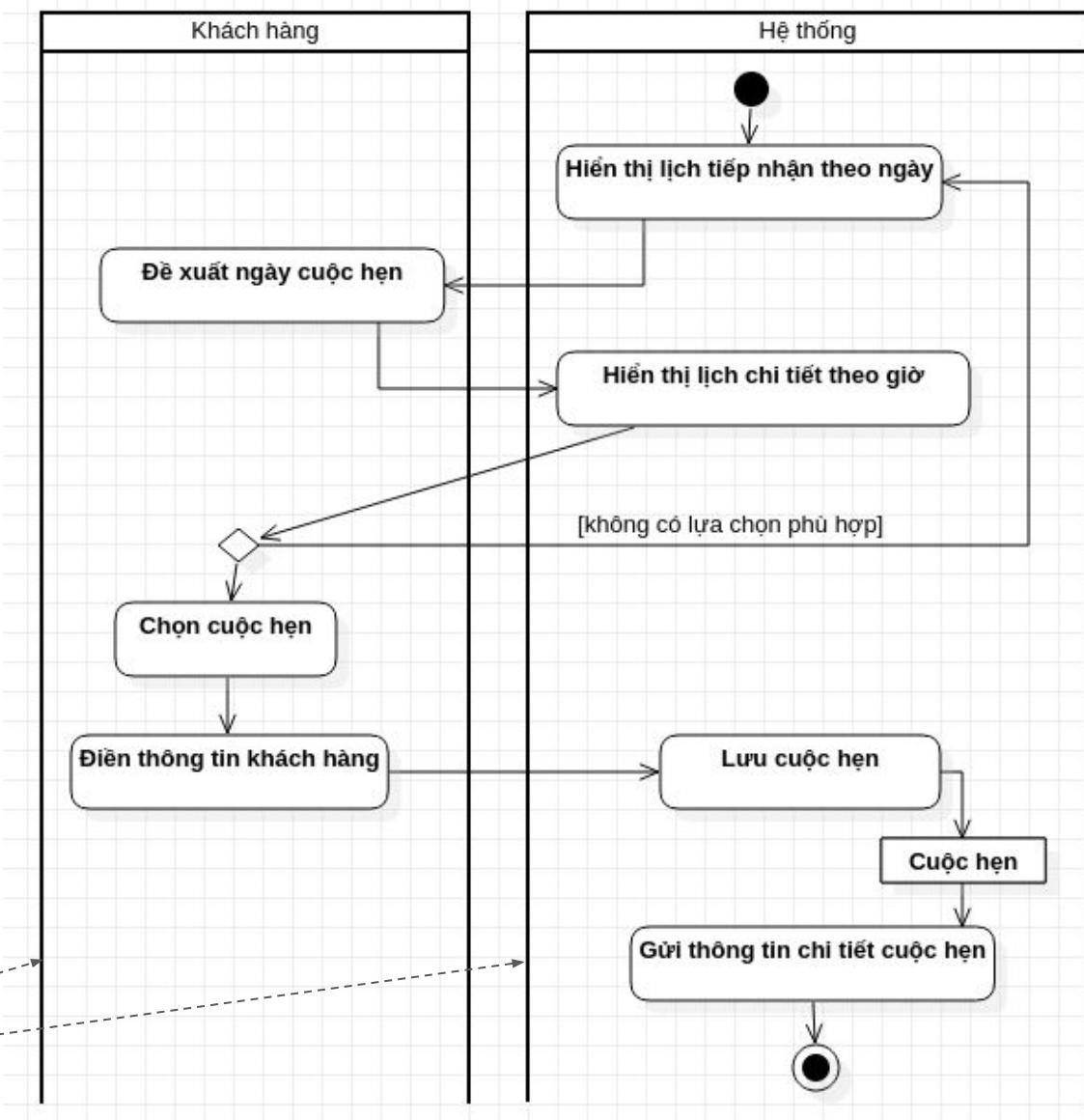
Hoạt động

Đối tượng

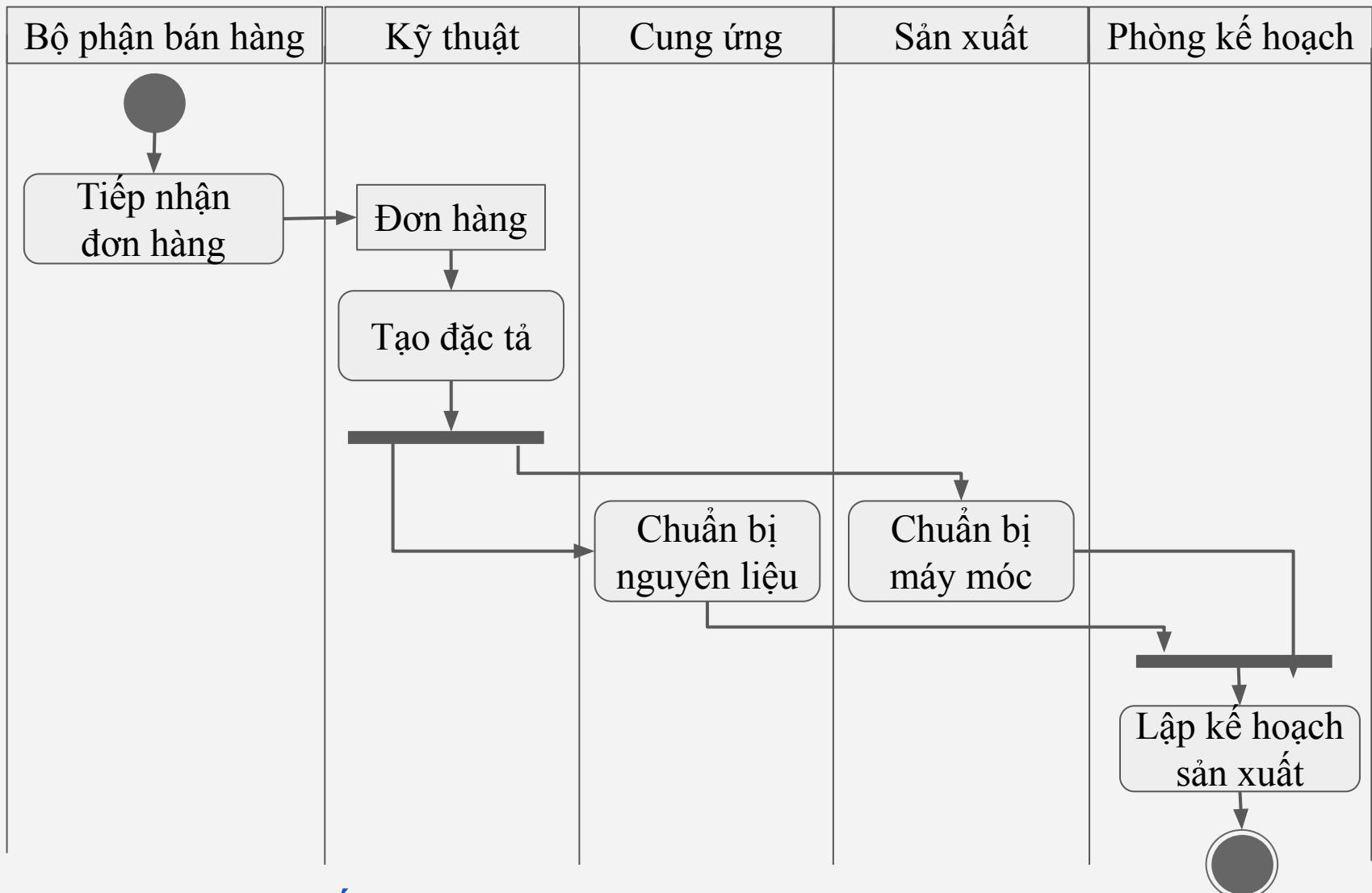
Kết thúc



# Ví dụ 2.2. Biểu diễn chủ thể của hoạt động



# Ví dụ 2.3. Biểu diễn các luồng song song

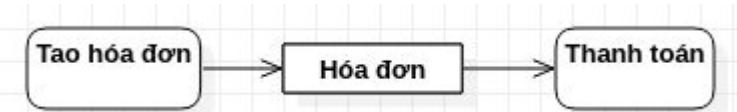


*... Thủ tiếp tục câu chuyện theo cách của bạn!*

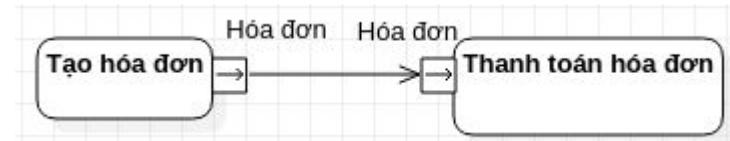
# Nút đối tượng

- Biểu diễn dữ liệu được sử dụng trong các hoạt động
- Có thể được định kiểu
  - Sử dụng các lớp trong mô hình cấu trúc
- Có thể hàm chứa nhiều đối tượng.
- Các loại nút đối tượng:
  - Tham số hoạt động: Tham số đầu vào và đầu ra của hoạt động/hành động.
  - Bộ đệm trung tâm: Bộ đệm cho các luồng đối tượng
  - Lưu trữ dữ liệu: Lưu trữ cố định các đối tượng
  - Cháu tham số: Tham số đầu vào và đầu ra được biểu diễn như các cháu trên các nút hành động/hoạt động.

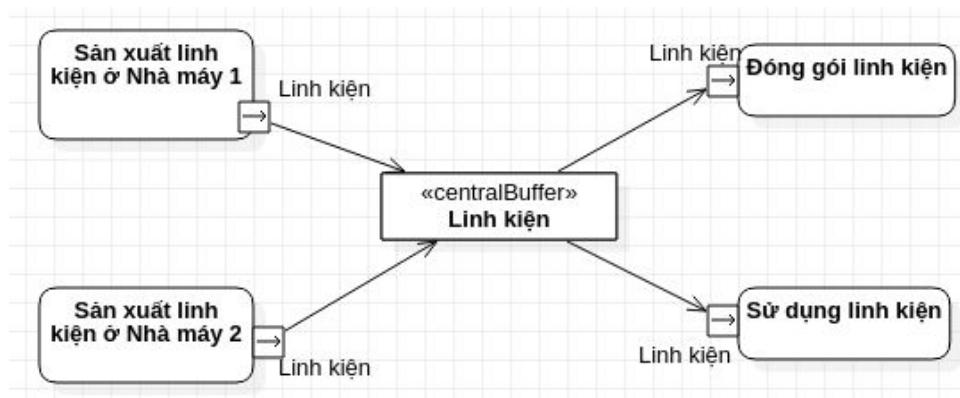
# Ví dụ 2.4. Các nút đối tượng



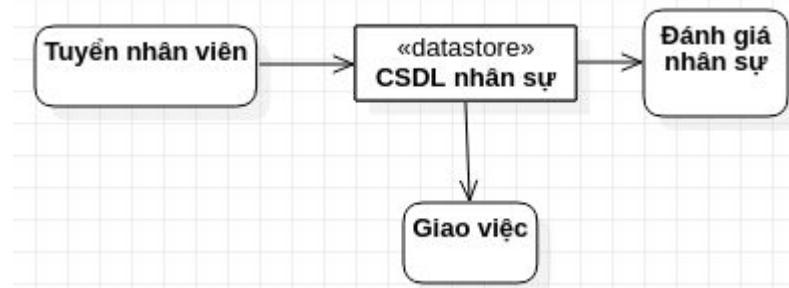
Tham số hoạt động



Biểu diễn tham số như các chấu



Bộ đệm trung tâm



Lưu trữ dữ liệu

# Nội dung

- Mô hình hóa nghiệp vụ
- Sơ đồ ca sử dụng
- Đặc tả ca sử dụng
- Phương pháp đơn vị ca sử dụng



# Ca sử dụng

- Khái niệm: Ca sử dụng là hoạt động sử dụng hệ thống để đáp ứng nhu cầu nghiệp vụ.
  - Mô tả các yêu cầu chức năng từ góc nhìn người dùng.
  - Không biểu diễn lô-gic hoạt động bên trong hệ thống.
  - Có thể bao gồm nhiều tương tác giữa người dùng và hệ thống để hoàn thành công việc.
- Xác định phạm vi ca sử dụng:
  - Tạo ra những lợi ích cho người dùng: Ở bước phân tích chúng ta chủ yếu quan tâm đến những tiến trình nghiệp vụ
  - Thường tương ứng với các tiến trình nghiệp vụ cơ bản (EBPs).
- Ca sử dụng được đặt tên theo quy cách Động từ + Danh từ:
  - Ngắn gọn và có tính gợi mở về hoạt động nghiệp vụ

# Xác định các ca sử dụng

- Các cách tiếp cận tiêu biểu:
  - Dựa trên mục đích của người dùng
  - Dựa trên sự kiện
- Có thể được thực hiện qua nhiều vòng lặp
  - Kiểm tra kỹ lưỡng tập ca sử dụng hiện có
  - Chia nhỏ hoặc hợp nhất ca sử dụng để có được kích thước hợp lý
  - Bổ xung các ca sử dụng mới được phát hiện.

# Tác nhân

- Biểu diễn người dùng hoặc hệ thống ngoại có tác động lên hệ thống và có trao đổi dữ liệu với hệ thống.
  - Tác nhân là con người được đặt tên theo vai trò/nhóm người dùng
  - Một người có thể giữ nhiều vai trò và nhiều người dùng có thể có cùng vai trò.
- Xác định nhóm người dùng:
  - Theo chức năng nghiệp vụ (ví dụ, giao hàng, bán hàng, tư vấn khách hàng)
  - Theo cơ cấu tổ chức (ví dụ, nhân viên, quản lý, giám đốc)

# Xác định ca sử dụng theo nhu cầu người dùng

- Thu thập thông tin và tìm hiểu mục đích sử dụng hệ thống của những người dùng cụ thể thuộc các nhóm
  - Mong chờ gì từ hệ thống?
  - Các yêu cầu cần được đáp ứng?
- Sắp xếp các yêu cầu theo nhóm người dùng
- Tạo danh mục ca sử dụng theo nhóm người dùng
- Xác định các ca sử dụng chung của nhiều nhóm người dùng

## Ví dụ 2.5. Nhu cầu người dùng và ca sử dụng

Nhóm người dùng	Nhu cầu và ca sử dụng
Khách hàng tiềm năng	Tìm kiếm sản phẩm Thêm hàng vào giỏ Xem phản hồi về sản phẩm
Người quản lý marketing	Thêm và cập nhật thông tin sản phẩm Thêm và cập nhật chương trình khuyến mãi Tạo báo cáo bán hàng
Nhân viên giao hàng	Giao sản phẩm Theo dõi trạng thái giao hàng Tạo yêu cầu trả hàng

# Xác định ca sử dụng theo sự kiện

- Sự kiện - Những diễn biến phát sinh ở một thời điểm và địa điểm cụ thể, có thể mô tả được, cần được ghi nhớ bởi hệ thống, kích hoạt một tiến trình của hệ thống
- Ví dụ: Khách hàng thanh toán tiền mua sản phẩm

*Các sự kiện nào cần phản hồi của hệ thống?*

# Phân loại các sự kiện cần được quan tâm

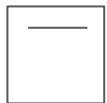
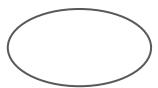
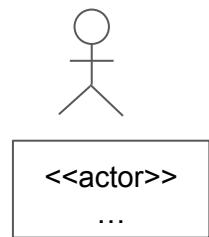
- Sự kiện ngoại được khởi tạo bởi một tác nhân của hệ thống
  - Tác nhân muốn hoàn thành 1 giao dịch
  - Tác nhân muốn tra cứu thông tin
  - Dữ liệu đã thay đổi và cần được cập nhật
- Sự kiện thời gian phát sinh ở 1 thời điểm hoặc sau 1 khoảng thời gian.
  - Cần xuất thông tin:
    - Các báo cáo phục vụ quản lý, vận hành
    - Thông báo nhắc nhở
- Sự kiện trạng thái phát sinh ở một trạng thái của hệ thống
  - Ví dụ, gần hết pin -> chuyển sang chế độ tiết kiệm pin, hiển thị thông báo
  - Số lượng tồn kho giảm xuống thấp hơn ngưỡng -> Gửi yêu cầu nhập hàng.

...

# Sự kiện và ca sử dụng



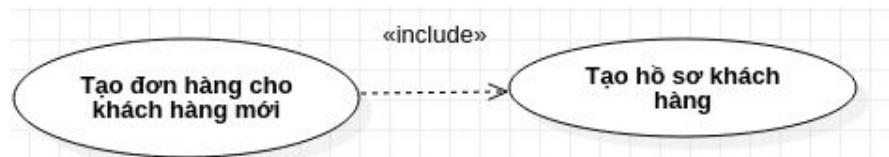
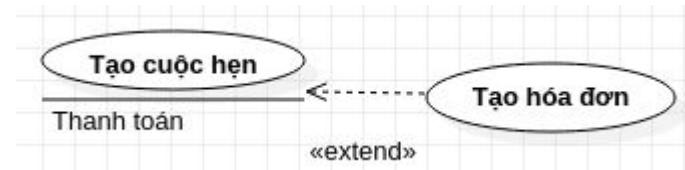
# Sơ đồ Ca sử dụng: Các thành phần thường gặp



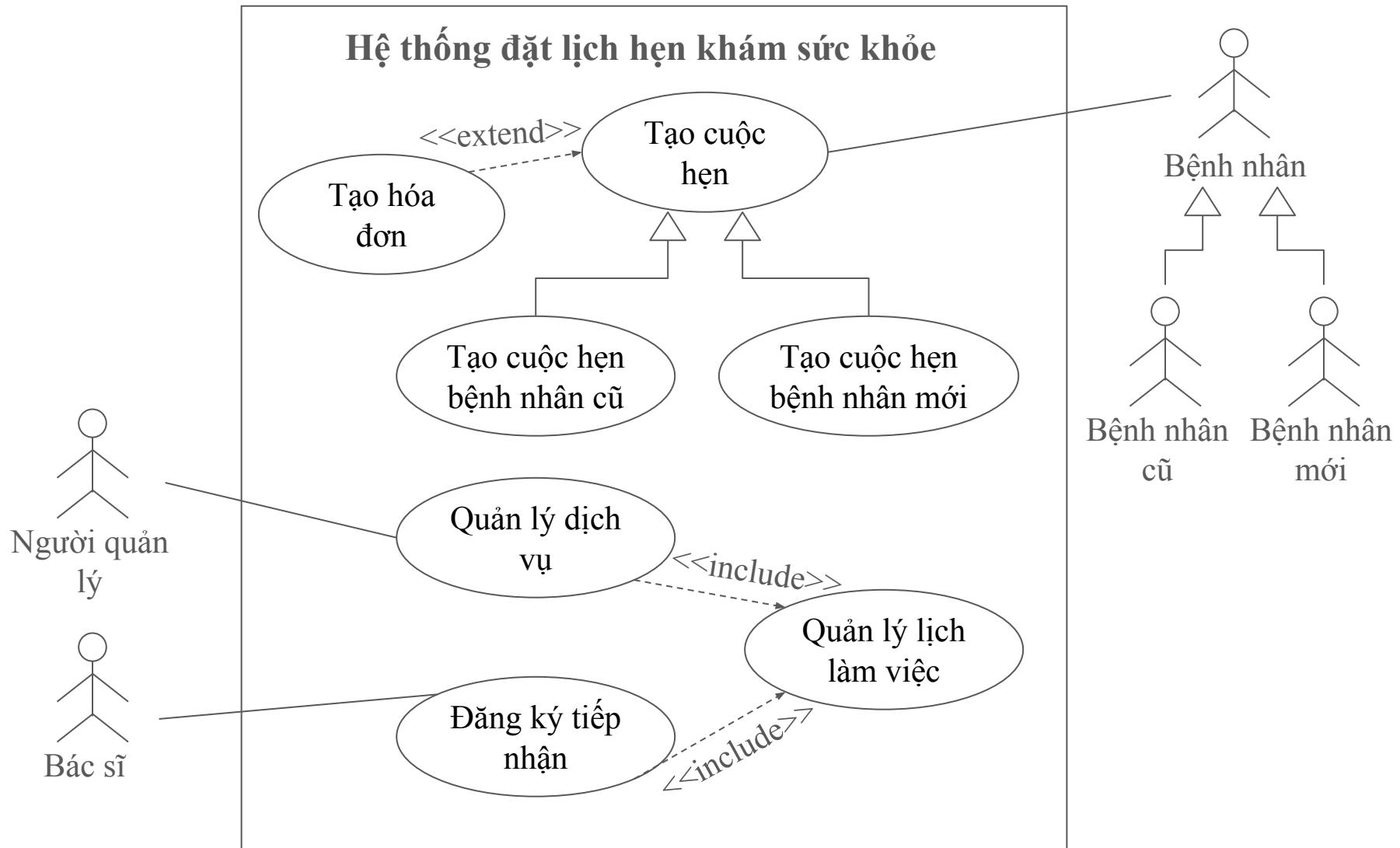
- **Tác nhân:**
  - Người dùng hoặc
  - Hệ thống ngoại
- **Ca sử dụng:** Hoạt động sử dụng hệ thống để đáp ứng nhu cầu nghiệp vụ
- **Chủ thể ca sử dụng:** Thực hiện các hoạt động trong ca sử dụng.
- **Quan hệ tương tác:** Kết nối tác nhân và ca sử dụng
- **Quan hệ bao gồm:** Các tương tác trong 1 ca sử dụng bao gồm 1 ca sử dụng khác.
- **Quan hệ nói rộng:** Các tương tác trong 1 ca sử dụng có thể được thêm vào 1 ca sử dụng khác.
- **Quan hệ khái quát hóa:** A là B hoặc A kế thừa B.

# Quan hệ nối rộng và bao gồm

- Các cơ chế mở rộng ca sử dụng: Nối rộng - mở rộng có điều kiện; bao gồm - luôn thêm vào.
  - Quan hệ nối rộng/extend:
    - Trong thời gian thực hiện ca sử dụng được mở rộng có thể kéo theo thực hiện 1 ca sử dụng khác (ca sử dụng được thêm vào).
    - Được biểu diễn bằng mũi tên nét đứt từ ca sử dụng được thêm vào tới ca sử dụng được mở rộng.
    - Vị trí mở rộng có thể được mô tả bằng các điểm mở rộng:
- Quan hệ bao gồm/include:
  - Phạm vi tương tác của ca sử dụng bao gồm việc thực hiện 1 ca sử dụng khác (ca sử dụng được thêm vào)
  - Được biểu diễn bằng mũi tên nét đứt từ ca sử dụng được mở rộng (ca cơ sở) tới ca sử dụng được thêm vào.



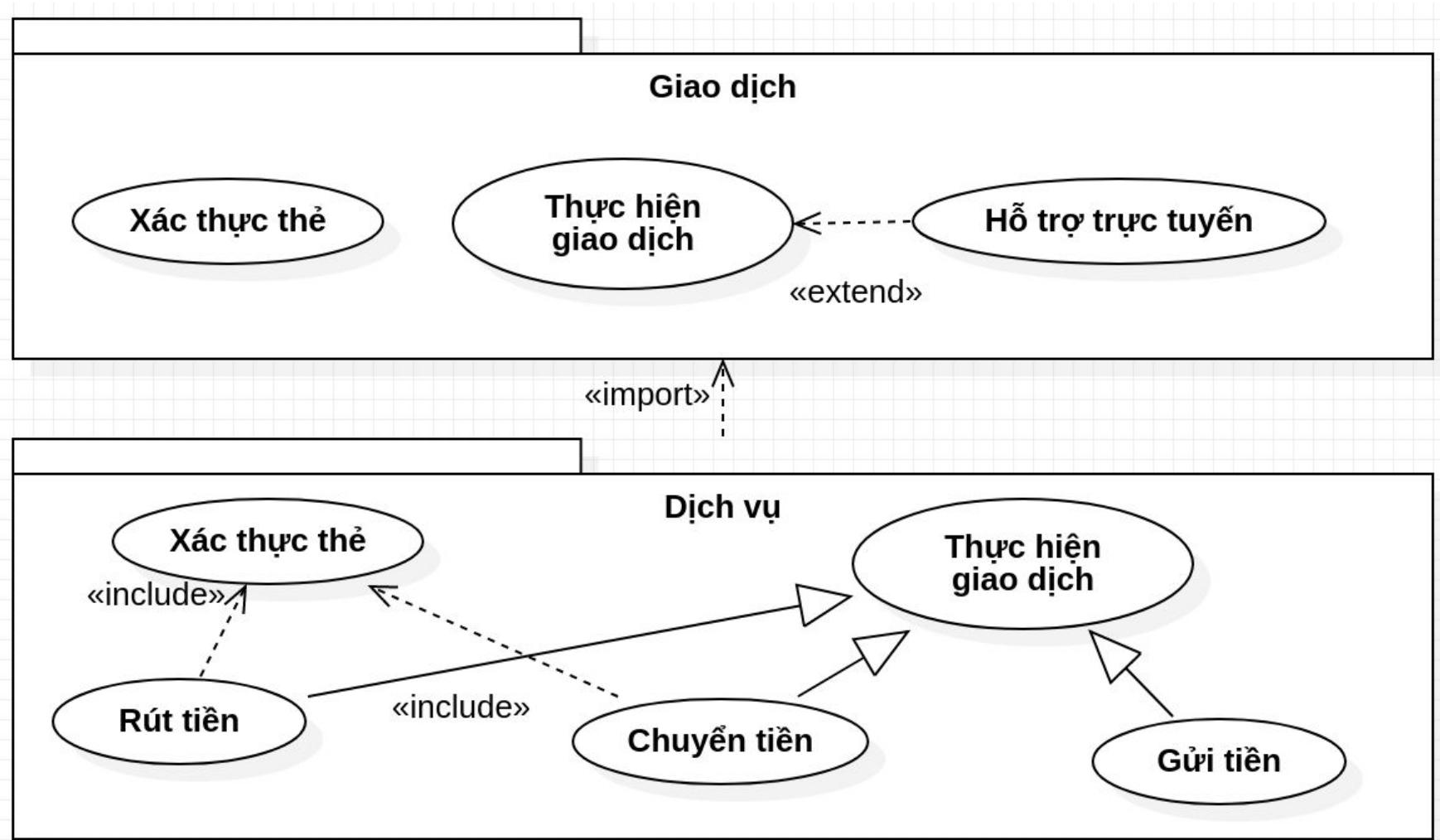
# Ví dụ 2.6. Sơ đồ ca sử dụng tổng quan



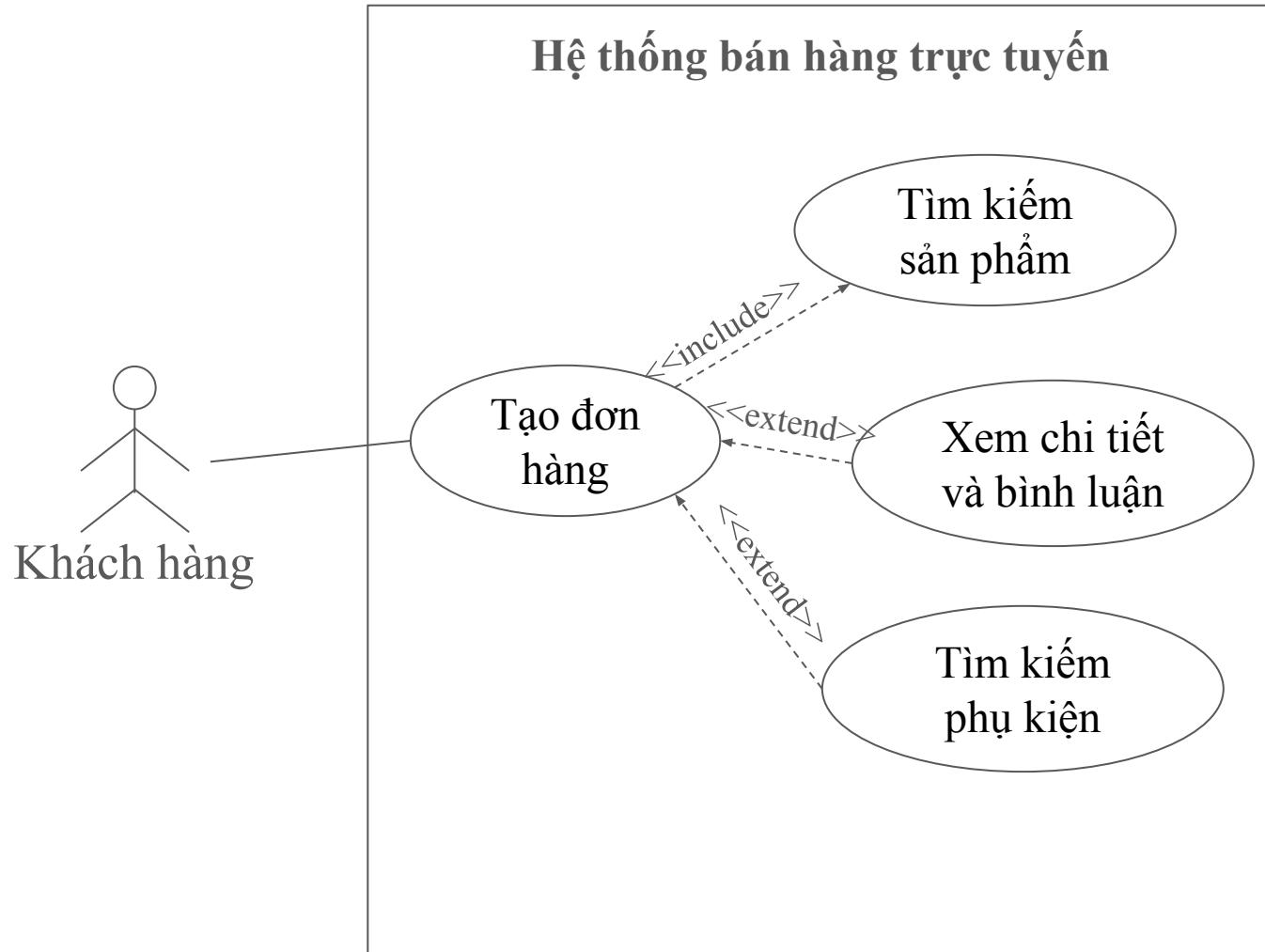
# Chủ thể và nhóm ca sử dụng

- Ca sử dụng có thể được áp dụng cho nhiều chủ thể
- Chủ thể của ca sử dụng có thể là hệ thống hoặc thành phần khác có hành vi: Thành phần (Component), Lớp (Class)
- Ca sử dụng có thể là phần tử của gói (package).
  - Gói với thành phần là ca sử dụng được gọi là nhóm ca sử dụng.

## Ví dụ 2.7. Nhóm ca sử dụng

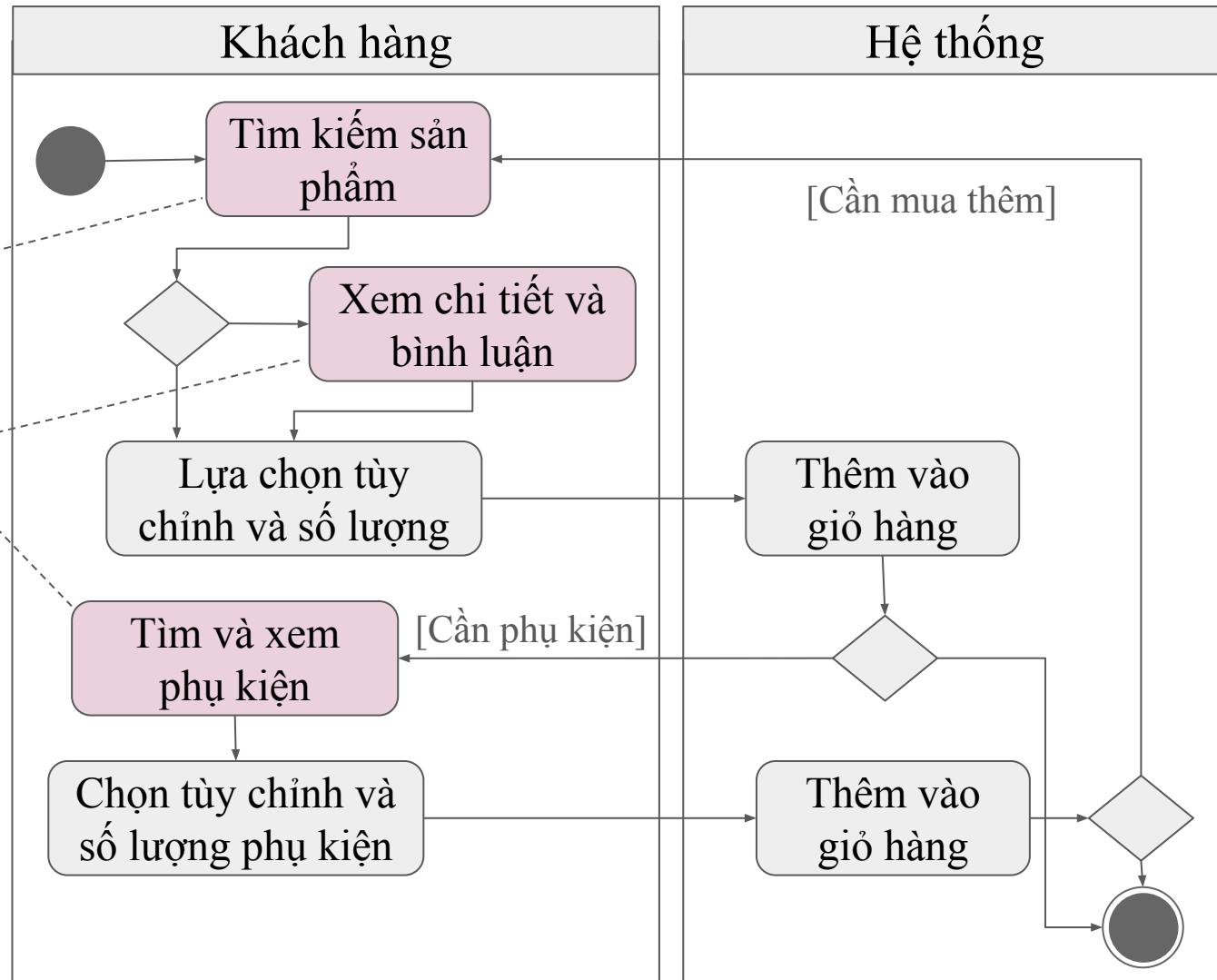


# Các điểm mở rộng



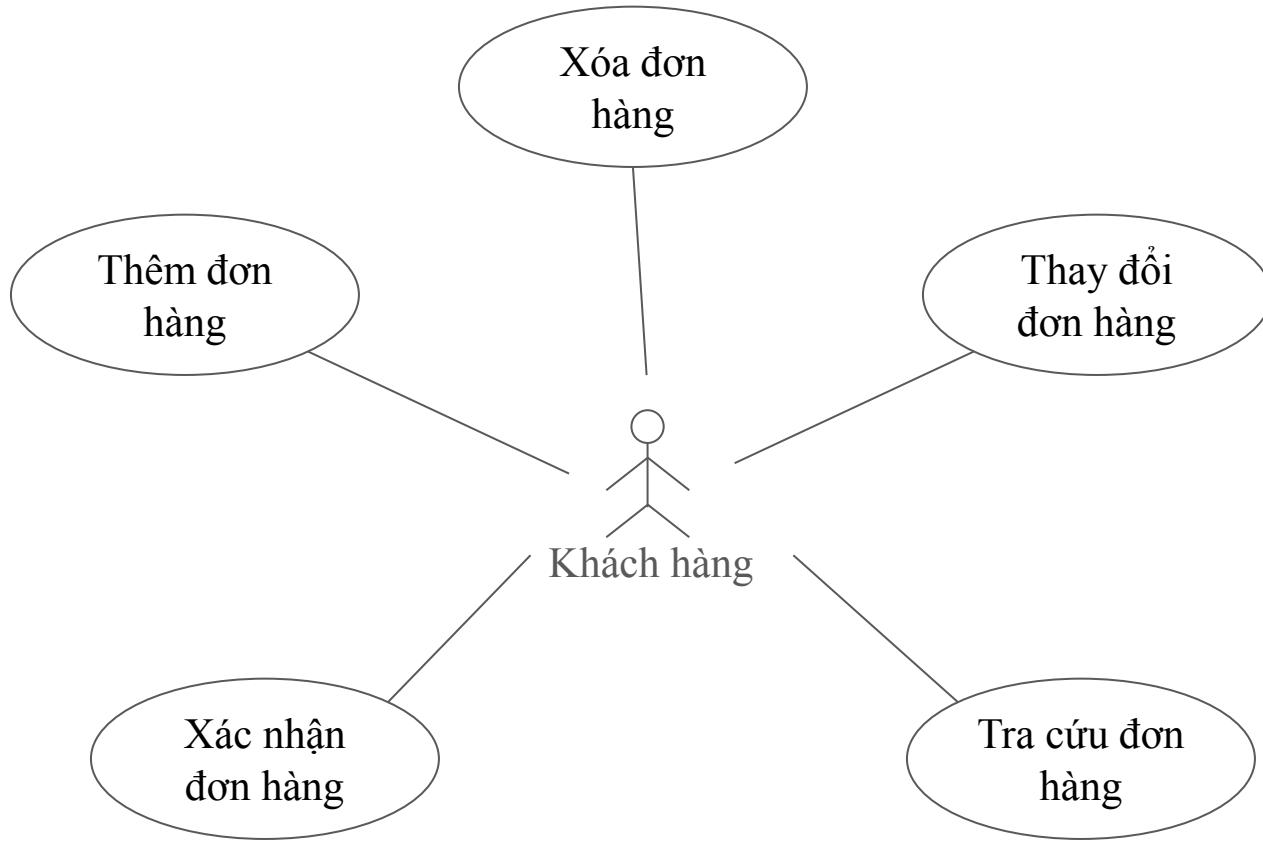
*Không nên lạm dụng việc phân rã ca sử dụng thành quá nhiều hoạt động nhỏ.*

# Các điểm mở rộng<sub>(2)</sub>



*Kịch bản thường của ca sử dụng tạo đơn hàng*

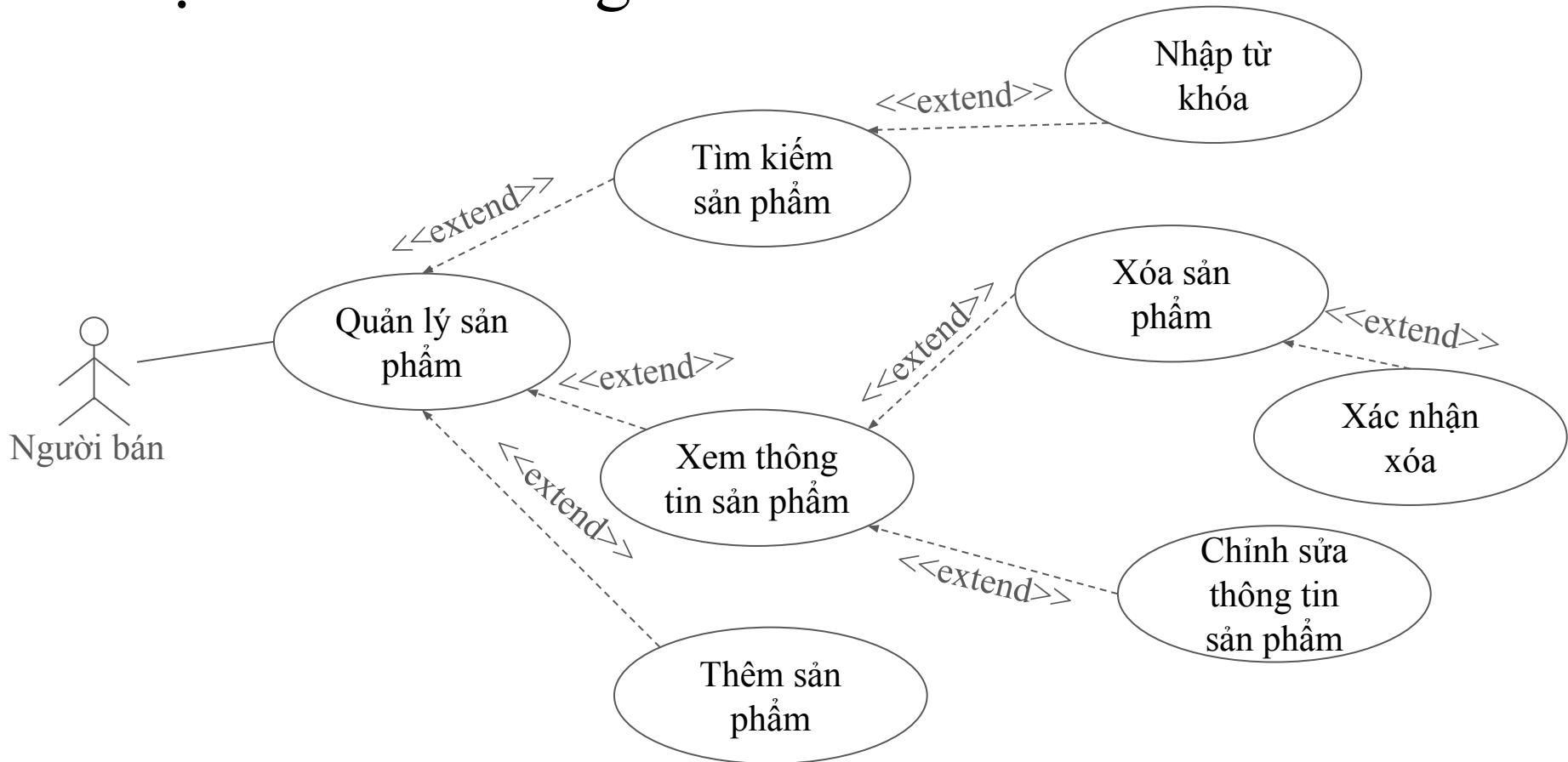
# Mô hình nào tốt hơn?



[*Why Use Cases Are Not "Functions"*, Kurt Bittner], [bản chuyển ngữ tiếng Việt](#)

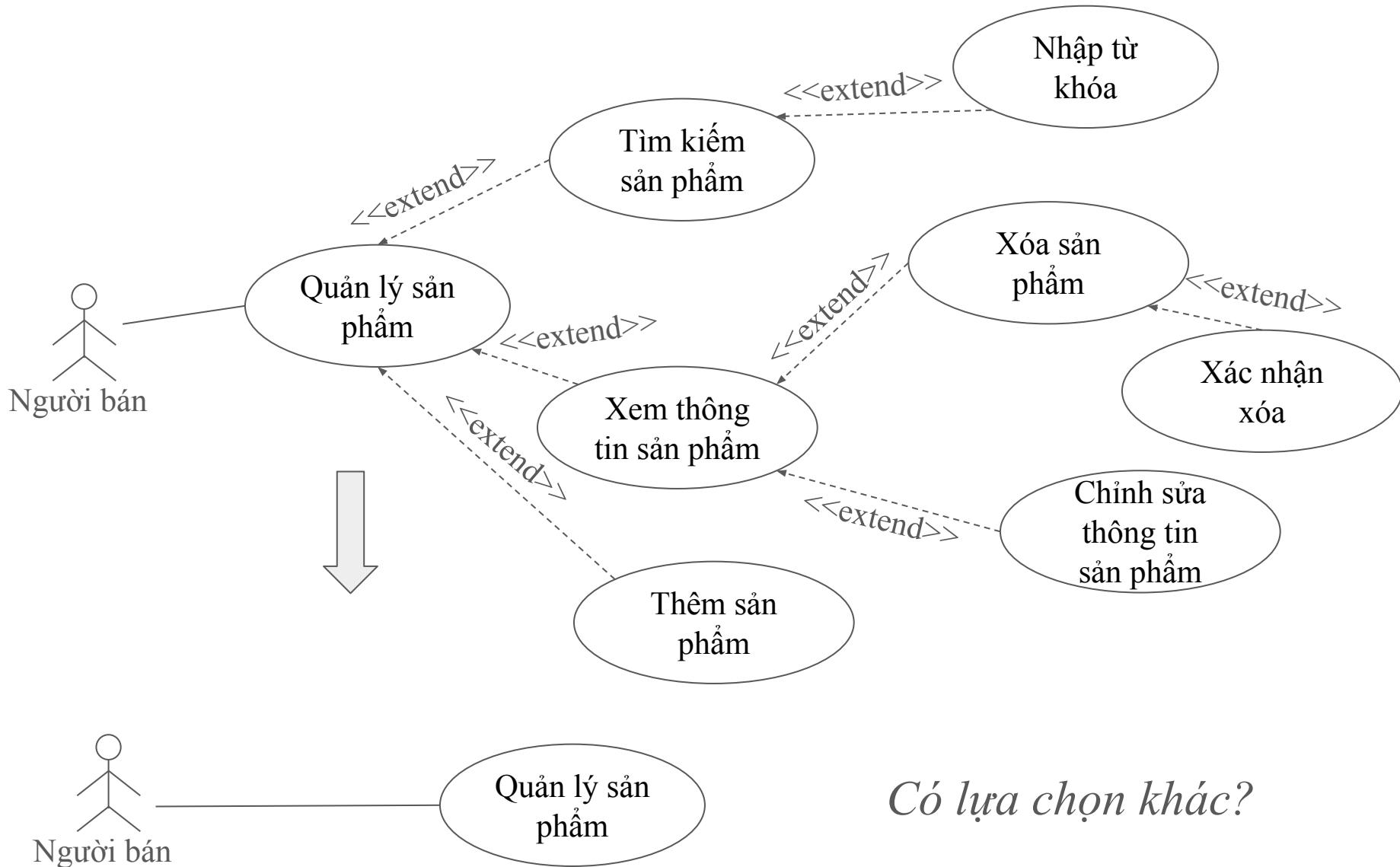


# Ví dụ về điều không nên làm



*Nhiều mức, vụn vỡ, không rõ giá trị!*

# Điều chỉnh



# Nội dung

- Mô hình hóa nghiệp vụ
- Sơ đồ ca sử dụng
- Đặc tả ca sử dụng
- Phương pháp đơn vị ca sử dụng



# Các thành phần trong đặc tả ca sử dụng

- Tổng quan:
  - Tên; ID; **Loại**; Tác nhân chính; Mô tả ngắn gọn; **Mức quan trọng**; (Các) *Cổ động* và mối quan tâm; (Các) Kích hoạt
- Các mối quan hệ:
  - Tương tác: Các giao tiếp giữa tác nhân và ca sử dụng
  - Bao gồm: Bao gồm một ca sử dụng khác
  - Nới rộng: Thêm vào nếu điều kiện được đáp ứng
  - Khái quát hóa: Từ trường hợp cụ thể đến trường hợp khái quát hơn
- Các luồng sự kiện/kịch bản thực hiện ca sử dụng:
  - Luồng sự kiện chính: Các hoạt động đặc trưng, thường diễn ra
  - Luồng sự kiện con: Chia nhỏ luồng sự kiện chính để đơn giản hóa mô tả ca sử dụng
  - Các luồng ngoại lệ hoặc tương đương: Các trường hợp chưa được tính đến trong luồng sự kiện chính

# Phân loại ca sử dụng

Lượng thông tin		
Mục đích	Khái quát	Chi tiết
Thiết yếu	Mô tả <b>khái quát</b> bậc cao của các vấn đề <b>thiết yếu</b> cần để hiểu chức năng đang được yêu cầu	Mô tả <b>chi tiết</b> các vấn đề <b>thiết yếu</b> cần để hiểu chức năng được yêu cầu
Thực tế	Mô tả <b>khái quát</b> bậc cao của một tập các bước cụ thể cần được thực hiện trên hệ thống <b>thực tế</b> sau khi triển khai	Mô tả <b>chi tiết</b> của một tập các bước được thực hiện trên hệ thống <b>thực tế</b> sau khi triển khai

# Mức quan trọng

- Thông thường được chia thành
  - Thấp
  - Trung bình
  - Cao
  - (So sánh tương đối giữa các ca sử dụng dựa trên lợi ích đem lại hoặc thiệt hại trong trường hợp sự cố)
- Được sử dụng cho mục đích lập kế hoạch & quản lý

# Biểu mẫu đặc tả ca sử dụng

Tên ca sử dụng:	ID:	Mức quan trọng:
Tác nhân chính:	Loại Ca Sử Dụng:	
Các cổ động và mối quan tâm:		
Mô tả ngắn gọn:		
Sự kiện kích hoạt: Loại sự kiện:		
Các mối liên hệ: Phối hợp: Bao gồm: Nới rộng: Khái quát hóa:	<i>Có thể lược bỏ nếu có sơ đồ CSD</i>	
Luồng sự kiện thông thường: 1.		
Các luồng con: <b>S-1:</b>	<i>Có thể để ngoài bảng</i>	
Các luồng ngoại lệ/Tương đương:		

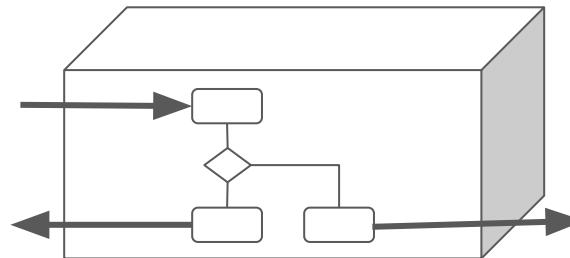
# Ví dụ 2.8. Đặc tả ca sử dụng *Không phải tài liệu mẫu.*

Tên ca sử dụng: Tạo cuộc hẹn	ID: 1-1	Mức quan trọng: Cao		
Tác nhân chính: Khách hàng phòng khám sức khỏe	Loại Ca Sử Dụng: Chi tiết, thiết yếu			
<p>Các cổ đông và mối quan tâm: <i>Khách hàng muốn đặt lịch hẹn gấp bác sĩ.</i>  <i>Đội ngũ quản lý cung cấp thông tin thời gian chờ, cập nhật lịch trình, và gửi xác nhận</i>  <i>Dịch vụ hệ thống phòng khám sức khỏe hiện tại cung cấp thông tin về khả năng tiếp nhận cuộc hẹn</i></p>				
Mô tả ngắn gọn: Ca sử dụng này mô tả cách khách hàng chọn một cuộc hẹn				
<p>Sự kiện kích hoạt: Khách hàng gặp vấn đề sức khỏe.          Loại sự kiện: Ngoại</p> <p>Luồng sự kiện thông thường:</p> <ol style="list-style-type: none"> <li>1. Hệ thống hiển thị khả năng tiếp nhận cuộc hẹn cùng với thời gian chờ cho khách hàng.</li> <li>2. Khách hàng để xuất thời gian cuộc hẹn.</li> <li>3. Yêu cầu cuộc hẹn của khách hàng được so sánh với khả năng tiếp nhận cuộc hẹn của phòng khám.</li> <li>4. Hiển thị kết quả kiểm tra khả năng tiếp nhận cuộc hẹn.</li> <li>5. Khách hàng chọn phương án phù hợp.</li> <li>6. Hệ thống cập nhật lịch làm việc.</li> <li>7. Hệ thống gửi thông tin cuộc hẹn đã được xác nhận đến khách hàng.</li> </ol>				
Các luồng con:				
<p>Các luồng ngoại lệ/Tương đương:</p> <p>5a. Nếu không chọn được, khách hàng có thể lặp các bước 2 -&gt; 5 cho tới khi tìm được lựa chọn phù hợp.</p>				

# Quản lý mức độ chi tiết



Hộp đen



Hộp trắng

Cố gắng:

- Hiểu người nghe
- Coi hệ thống như 1 hộp đen
- Sử dụng các thành phần hộp trắng, các chi tiết ở mức độ hợp lý làm ca sử dụng cụ thể hơn và dễ hiểu hơn.

# Mức độ chi tiết hợp lý

- Không chứa các chi tiết thiết kế giao diện người dùng - Tập trung vào thông tin và sự kiện, không phải định dạng và điều khiển.
- Tập trung vào yêu cầu thay vì giải pháp.
- Tập trung vào các hành vi mong đợi, không phải cách triển khai các hành vi.

# Kiểm tra và xác nhận ca sử dụng

- Ca sử dụng phải được kiểm tra và xác nhận trước khi bắt đầu mô hình hóa cấu trúc và mô hình hóa hành vi
  - 1-3) Đảm bảo tính nhất quán giữa luồng sự kiện trong đặc tả ca sử dụng và sơ đồ hoạt động: Sự kiện - Hành động/Hoạt động, các nút đối tượng; cách thực hiện
  - 4) Đảm bảo mỗi ca sử dụng đều có một và chỉ một đặc tả.
  - 5-7) Đảm bảo tính nhất quán giữa các đặc tả ca sử dụng và sơ đồ ca sử dụng: Danh sách tác nhân; Các cỗ động được liệt kê trong đặc tả ca sử dụng có thể được biểu diễn trên sơ đồ ca sử dụng; Tất cả các mối quan hệ.
  - 8) Đảm bảo đúng cú pháp sơ đồ

# Nội dung

- Mô hình hóa nghiệp vụ
- Biểu đồ ca sử dụng
- Đặc tả ca sử dụng
- Phương pháp đơn vị ca sử dụng



# Ước lượng chi phí dự án

- Những yếu tố tiêu biểu của dự án: Chức năng, Thời hạn và Kinh phí
- Thời gian và kinh phí có thể được ước lượng dựa trên các mô tả chức năng
- Những ước lượng hợp lý nhất được thực hiện dựa trên kinh nghiệm
- Phương pháp đơn vị ca sử dụng được dựa trên:
  - Tác nhân và các đặc tả ca sử dụng
  - Các hệ số phức tạp kỹ thuật (13)
  - Các hệ số môi trường (8)

# UCP: Phân loại tác nhân

**Bảng đánh giá trọng số tác nhân chưa hiệu chỉnh**

Loại tác nhân	Mô tả	Điểm	Số lượng	Tổng điểm
Đơn giản	Hệ thống ngoại với API được định nghĩa rõ ràng	1	0	0
Trung bình	Hệ thống ngoại sử dụng một giao diện dựa trên giao thức, ví dụ, HTTP, TCP/IP, hoặc một cơ sở dữ liệu	2	0	0
Phức tạp	Người	3	4	12
<b>Tổng trọng số tác nhân chưa hiệu chỉnh (UAW)</b>				<b>12</b>

# UCP: Phân loại ca sử dụng

<b>Bảng đánh giá trọng số ca sử dụng chưa hiệu chỉnh</b>				
<b>Loại CSD</b>	<b>Mô tả</b>	<b>Điểm</b>	<b>Số lượng</b>	<b>Tổng điểm</b>
Đơn giản	1-3 giao dịch	5	3	15
Trung bình	4-7 giao dịch	10	4	40
Phức tạp	>7 giao dịch	15	1	15
<b>Tổng trọng số ca sử dụng chưa hiệu chỉnh (UUCW)</b>				<b>70</b>

**Số lượng đơn vị ca sử dụng chưa hiệu chỉnh:**

$$\text{UUCP} = \text{UAW} + \text{UUCW} = 12 + 70 = 82$$

# UCP: Đánh giá các chỉ số kỹ thuật

TFactor ∈ [0..5]

- 0: Không ảnh hưởng gì.
- 3: Ảnh hưởng trung bình.
- 5: Ảnh hưởng cao.

Các hệ số phức tạp kỹ thuật					
Mã số	Mô tả	Hệ số	Giá trị	Giá trị thực	Ghi chú
T1	Hệ phân tán	2	0	0	
T2	Thời gian phản hồi hoặc thông lượng	1	5	5	
T3	Sử dụng thuận tiện và hiệu quả	1	3	3	
T4	Xử lý bên trong phức tạp	1	1	1	
T5	Khả năng tái sử dụng mã nguồn	1	1	1	
T6	Dễ cài đặt	0.5	2	1	
T7	Dễ vận hành	0.5	4	2	
T8	Tính khả chuyển	2	0	0	
T9	Dễ bảo trì và cập nhật	1	2	2	
T10	Xử lý tính toán song song/dồng thời	1	0	0	
T11	Bảo mật	1	0	0	
T12	Liên kết với đối tác, sử dụng/cung cấp	1	0	0	
T13	Đào tạo đặc biệt cho người dùng	1	0	0	
Tổng giá trị hệ số kỹ thuật (TFactor)					15

Hệ số phức tạp kỹ thuật: TCF = 0.6 + (0.01 \* TFactor) = 0.6 + (0.01 \* 15) = 0.75

# UCP: Đánh giá các chỉ số môi trường

EFactor ∈ [0..5]

- 0: Không có động lực.
- 3: Động lực trung bình.
- 5: Động lực cao.

Các hệ số môi trường					
Mã số	Mô tả	Trọng số	Giá trị	Giá trị thực	Ghi chú
E1	Có kinh nghiệm với quy trình phát triển hệ thống	1.5	4	6	
E2	Có kinh nghiệm về ứng dụng tương tự	0.5	4	2	
E3	Kinh nghiệm về hướng đối tượng	1	4	4	
E4	Khả năng lãnh đạo nhóm	0.5	5	2.5	
E5	Động lực làm việc	1	5	5	
E6	Sự ổn định của yêu cầu	2	5	10	
E7	Nhân sự bán thời gian	-1	0	0	
E8	Sự phức tạp của ngôn ngữ lập trình	-1	4	-4.0	
Tổng giá trị hệ số môi trường (EFactor)					25.5

$$\text{Hệ số môi trường: } EF = 1.4 + (-0.03 * \text{EFactor}) = 1.4 + (-0.03 * 25.5) = 0.635$$

# Đánh giá các chỉ số phức tạp

Tham khảo:

Object-Oriented Analysis and Design for Information Systems,  
Modeling with UML, OCL, and IFML. [Raul Sidnei Wazlawick].

Xem [Bản dịch tiếng Việt](#)

# UCP: Kết hợp các thành phần

Số lượng đơn vị ca sử dụng sau hiệu chỉnh:

$$\text{UCP} = \text{UUCP} * \text{TCF} * \text{EF} = 82 * 0.75 * 0.635 = 39.0525$$

Đặt số lượng đặc điểm môi trường không thuận lợi = (#đặc điểm trong khoảng E1...E6 được gán giá trị < 3) + (# đặc điểm trong khoảng E7...E8 được gán giá trị > 3)

Nếu số lượng đặc điểm môi trường không thuận lợi  $\leq 2$

thì PHM = 20

Ngược lại, nếu số lượng đặc điểm môi trường không thuận lợi = 3 hoặc 4

thì PHM = 28

Ngược lại

thì suy nghĩ lại về dự án; rủi ro thất bại quá cao.

**Chi phí tính bằng giờ nhân lực E = UCP \* PHM = 39.1 \* 20 = 782**

# UCP: Kích thước nhóm tối ưu

Giả sử số giờ làm việc trong 1 tháng = 158

**Chi phí theo tháng nhân lực E = UCP \* PHM / 158 = 4.9**

Thời gian tối ưu theo McConnel (1996)

Kích thước nhóm trung bình

$$T = 2.5 * \sqrt[3]{E} = 2.5 * \sqrt[3]{4.9} = 4.2$$

Với nhóm N thành viên. Nếu N > P thì thời gian thực hiện dự án được đánh giá > E / N, nếu ngược lại N < P thì thời gian thực hiện dự án được đánh giá khả thi trong giới hạn E / N.



# **Phân tích và thiết kế Hệ thống**

Giảng viên: Nguyễn Bá Ngọc

Chương 3

Hà Nội-2021

# **Chương 3**

## **Mô hình hóa cấu trúc**

# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập

# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập

# Lĩnh vực vấn đề

- Lĩnh vực vấn đề (*Problem domain*) - Vùng nhu cầu nghiệp vụ của người dùng trong phạm vi hệ thống
  - Bao gồm tất cả thông tin, các thứ như hóa đơn, sản phẩm v.v..
  - .. người dùng gặp khi tác nghiệp.

# Lớp lĩnh vực

- Lớp lĩnh vực (*domain class*) mô tả các đối tượng của lĩnh vực vấn đề, các thứ mà người dùng gặp khi thực hiện các hoạt động nghiệp vụ
  - Chứa các thông tin cần được ghi nhớ và xử lý bởi hệ thống
  - Ví dụ, sản phẩm, hóa đơn, thông tin khách hàng, khoản thanh toán, v.v.
- (*Lớp lĩnh vực là 1 tầng kiến trúc hệ thống*)

*Làm sao để xác định các lớp lĩnh vực?*

# Các bài toán

- Xác định các thành phần dữ liệu cần lưu trữ?
- Kết hợp các thành phần và xác định các đối tượng?
- Xác định các thuộc tính và các lớp lĩnh vực?
- Xác định các mối quan hệ giữa các đối tượng?
- Xây dựng mô hình cấu trúc?

# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập



# Các kỹ thuật

- Kỹ thuật danh từ (Noun): Tìm, phân loại, và thiết lập danh sách danh từ có trong các nguồn như thảo luận hoặc tài liệu. Sau đó phân tích và xác định các lớp lĩnh vực và thuộc tính, loại bỏ những thứ không cần ghi nhớ/chỉ giữ lại những thứ cần ghi nhớ.
- Tổng hợp kiến thức (BrainStorming): Tất cả mọi người cùng tham gia đóng góp hiểu biết, biên soạn danh sách các mục dữ liệu quan trọng.
  - Thông tin có thể được tổng hợp và phân loại theo danh mục và được sử dụng làm cơ sở xác định các lớp lĩnh vực.

# Kỹ thuật danh từ

- Thường dẫn đến 1 danh sách dài với nhiều danh từ, bao gồm cả những thứ không cần lưu trong hệ thống.
  - Danh từ có thể là tên của thuộc tính, đối tượng, lớp và cả những thứ khác
- Có thể đem lại kết quả ban đầu đáng kể, kết quả thu được có thể tiếp tục được mở rộng, ví dụ dựa trên kinh nghiệm, hiểu biết về lĩnh vực vấn đề.

# Các bước áp dụng kỹ thuật danh từ

1. Sử dụng các mô hình chức năng (đặc tả ca sử dụng, sơ đồ ca sử dụng v.v.. ), và các tài liệu khác về hệ thống (bao gồm các thông tin nhập và xuất), xác định tất cả các danh từ và lập danh mục danh từ.
2. Sử dụng các nguồn thông tin khác hiện có: Các biểu mẫu, các báo cáo hiện có v.v.., để mở rộng danh mục.
3. Kiểm tra các danh từ đã tìm được.
4. Kiểm tra danh mục và ghi chú phân loại từng danh từ cần được thêm vào, có thể giữ, loại bỏ hoặc cân nhắc thêm v.v..
5. Cùng kiểm tra danh sách với người dùng, các bên liên quan, và các thành viên thực hiện dự án và sau đó lập danh sách các thứ trong lĩnh vực nghiệp vụ.

# Một số câu hỏi hỗ trợ phân loại danh từ

Các câu hỏi có thể được sử dụng để thực hiện bước 3

- Để quyết định giữ 1 danh từ
  - Nó có nằm trong phạm vi của hệ thống đang được phát triển?
  - Hệ thống có cần ghi nhớ nhiều hơn 1 thứ tương tự?
- Để quyết định loại 1 danh từ
  - Nó có phải 1 từ đồng nghĩa với 1 thứ khác đã được xác định?
  - Chỉ là đầu ra do hệ thống cung cấp từ những thông tin khác đã được xác định?
  - Chỉ là đầu vào dẫn tới việc lưu thông tin khác đã được xác định?
- Để xác định thuộc tính và đối tượng
  - Nó có phải thành phần của thứ khác đã được xác định?
  - Nó có phải thứ người dùng có thể cần nếu các giả thuyết thay đổi?

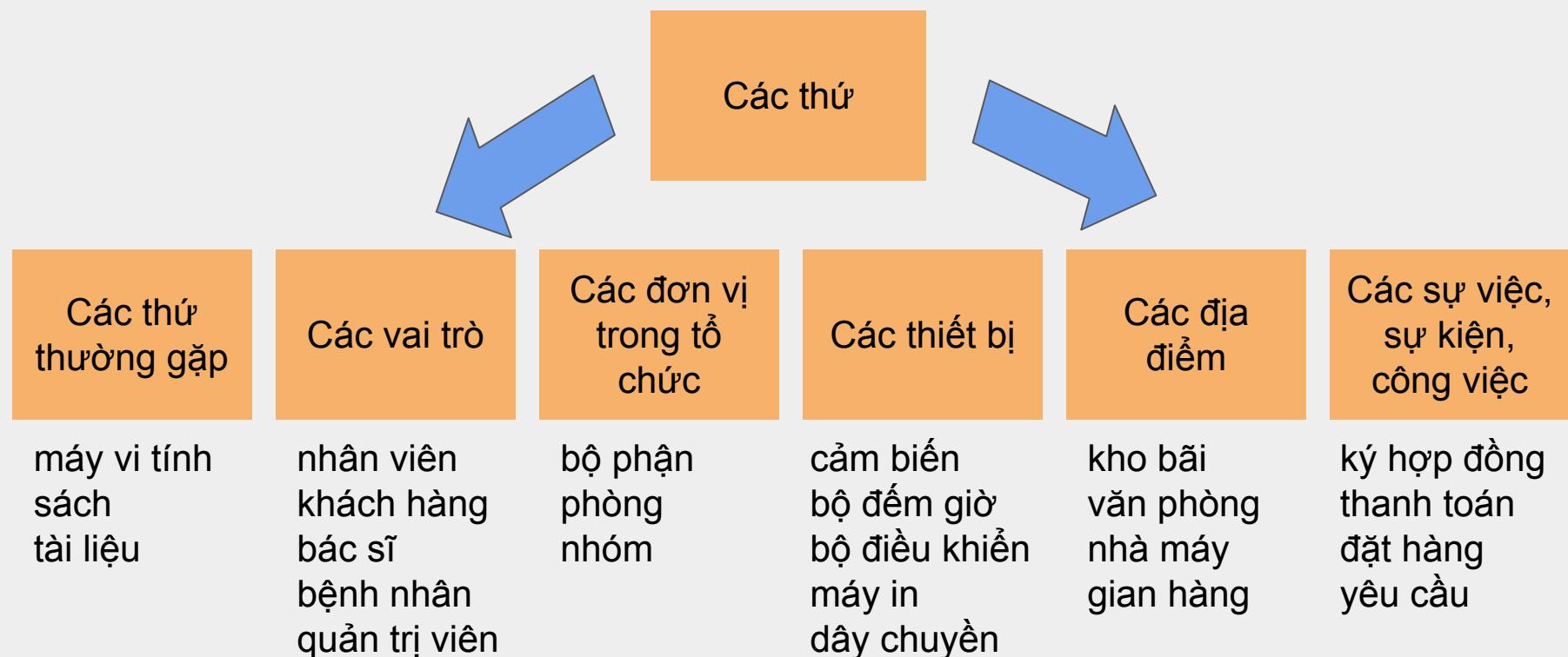
## Ví dụ 3.1. Các danh từ

Các danh từ đã được phát hiện	Ghi chú
Khoản thanh toán	
Đơn hàng	
Ngân hàng	
Danh mục	
Sản phẩm	
Yêu cầu hoàn trả	
Giá bán	
Thông tin xác nhận	
Thông tin thẻ tín dụng	
Khách hàng	
Thông tin quản lý	
Thông tin quảng cáo	
Bản báo cáo doanh thu	

# Các bước tổng hợp kiến thức

1. Lựa chọn người dùng và 1 tập ca sử dụng liên quan
2. Phỏng vấn người dùng và xác định các thứ liên quan đến ca sử dụng và thông tin về chúng cần được quản lý bởi hệ thống.
3. Sử dụng các danh mục để quản lý và đưa ra các câu hỏi: Bạn có lưu thông tin về những thứ quan trọng nào? Các hoạt động có được gắn với địa điểm nào không? Vai trò nào của người dùng cần được ghi nhớ?
4. Tiếp tục tổng hợp kiến thức từ tất cả các nhóm người dùng và những người liên quan để mở rộng kết quả.
5. Hợp nhất các kết quả, loại trùng lặp, và lập danh sách các thứ.

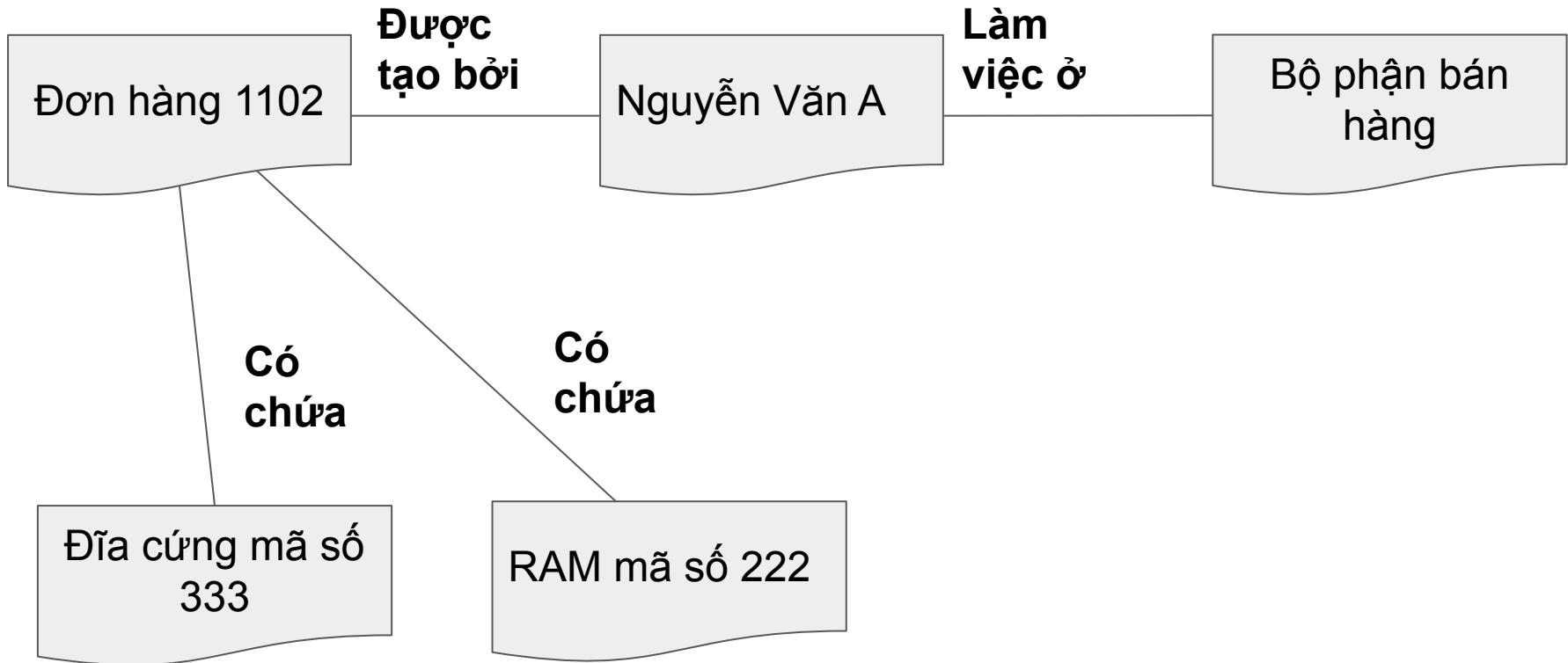
## Ví dụ 3.2. Danh mục phân loại



### Ví dụ 3.3. Các thuộc tính và giá trị

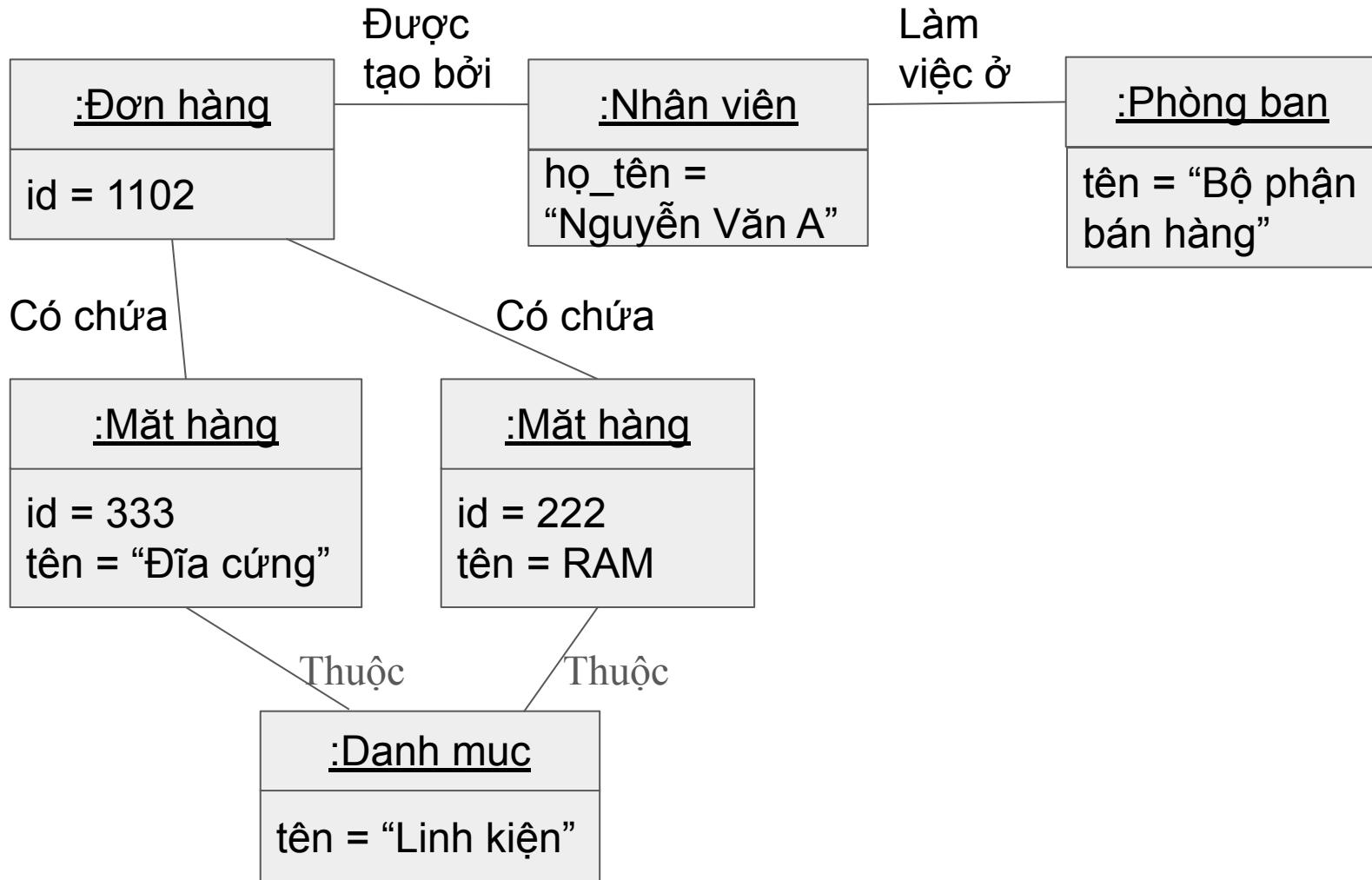
Tất cả khách hàng đều có các thuộc tính	Giá trị của thuộc tính		
ID	101	102	103
Họ-đệm	Nguyễn Văn	Trần Thị	Vũ Văn
Tên	A	C	D
SĐT	098888888	096777777	097333333

# Ví dụ 3.4. Liên kết các đối tượng



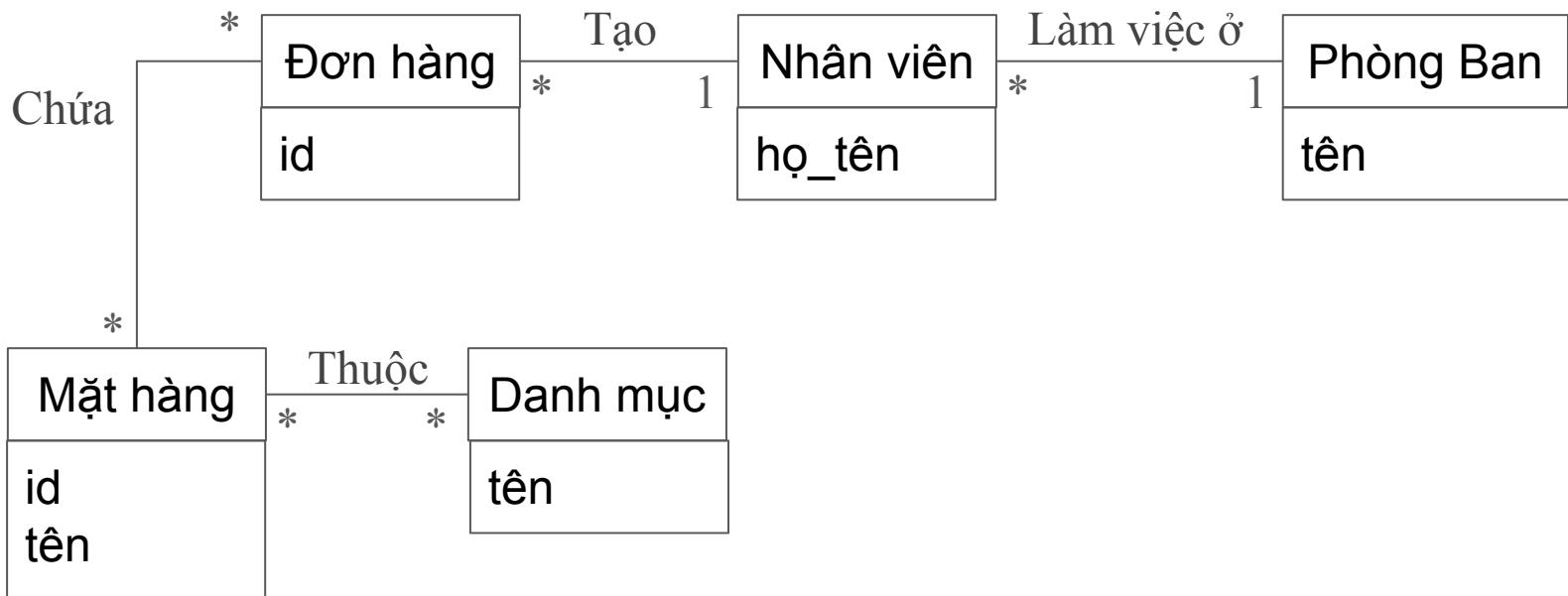
*Lưu ý:* Ví dụ này là 1 phác thảo tự do cho ý tưởng, không phải 1 sơ đồ UML

# Ví dụ 3.4. Liên kết các đối tượng<sub>(2)</sub>



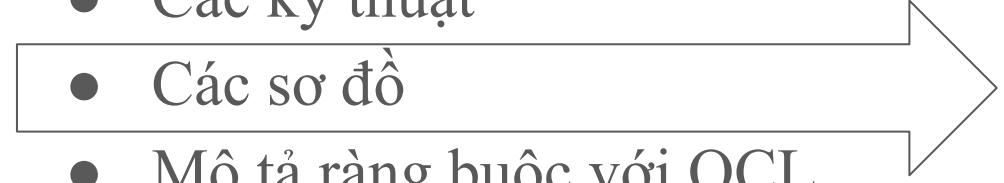
*Sơ đồ đối tượng*

## Ví dụ 3.4. Các mối liên hệ<sub>(3)</sub>



*Sơ đồ lớp*

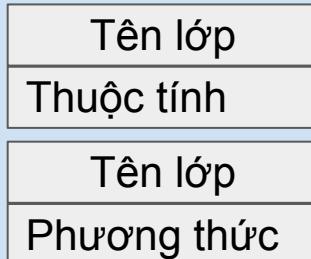
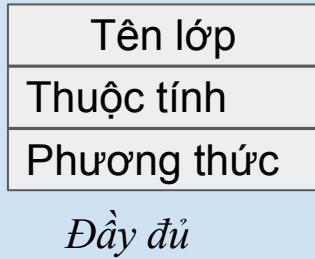
# Nội dung

- Các khái niệm
  - Các kỹ thuật
  - Các sơ đồ
  - Mô tả ràng buộc với OCL
  - Đặc tả lớp với thẻ CRC
  - Mẫu phân tích
  - Bài tập
- 

# Sơ đồ lớp: Các thành phần thường gặp

## Lớp

Các thành phần của lớp: Tên, thuộc tính, phương thức



Tên lớp  
*Tối giản*

Giới hạn truy cập:

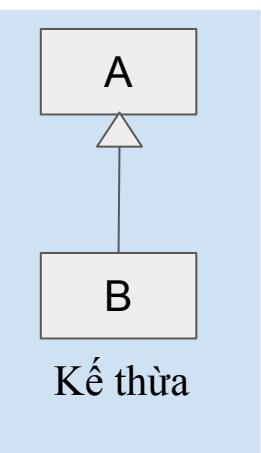
- + : Công khai
- : Riêng tư
- # : Bảo vệ

(Có thể lược bỏ thuộc tính và/hoặc phương thức)

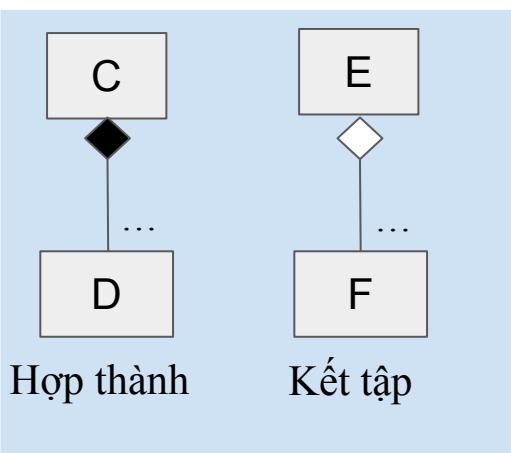
Ví dụ lớp

C
+ a1
- a2
# a3
+ op1 ()
- op2 ()
# op3()

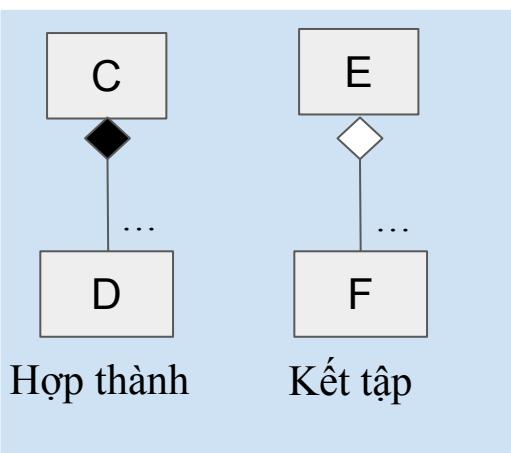
## Quan hệ



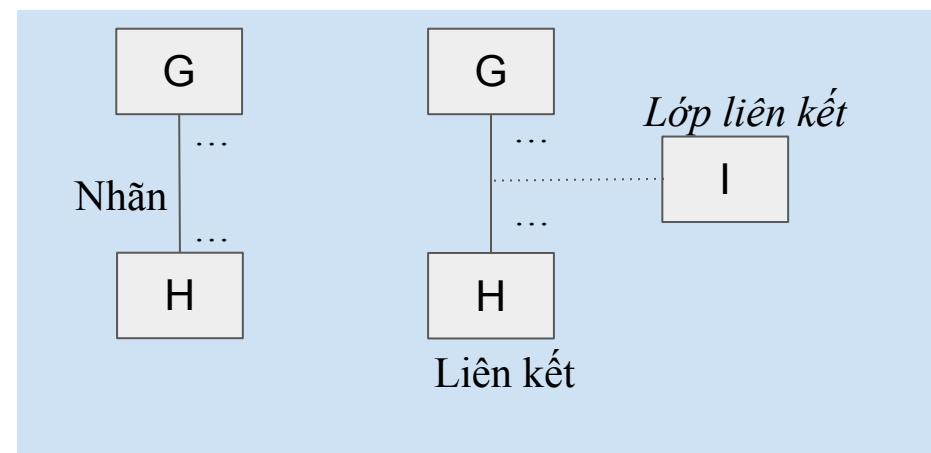
Kế thừa



Hợp thành

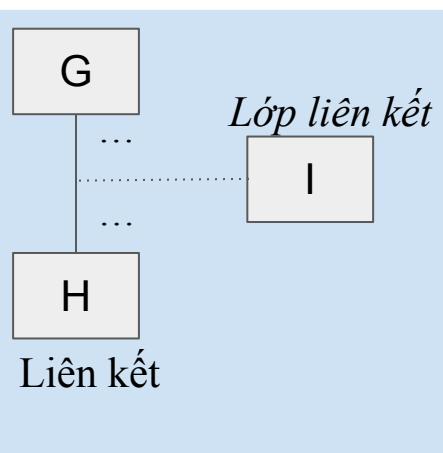


Kết tập



Nhãn

H



H

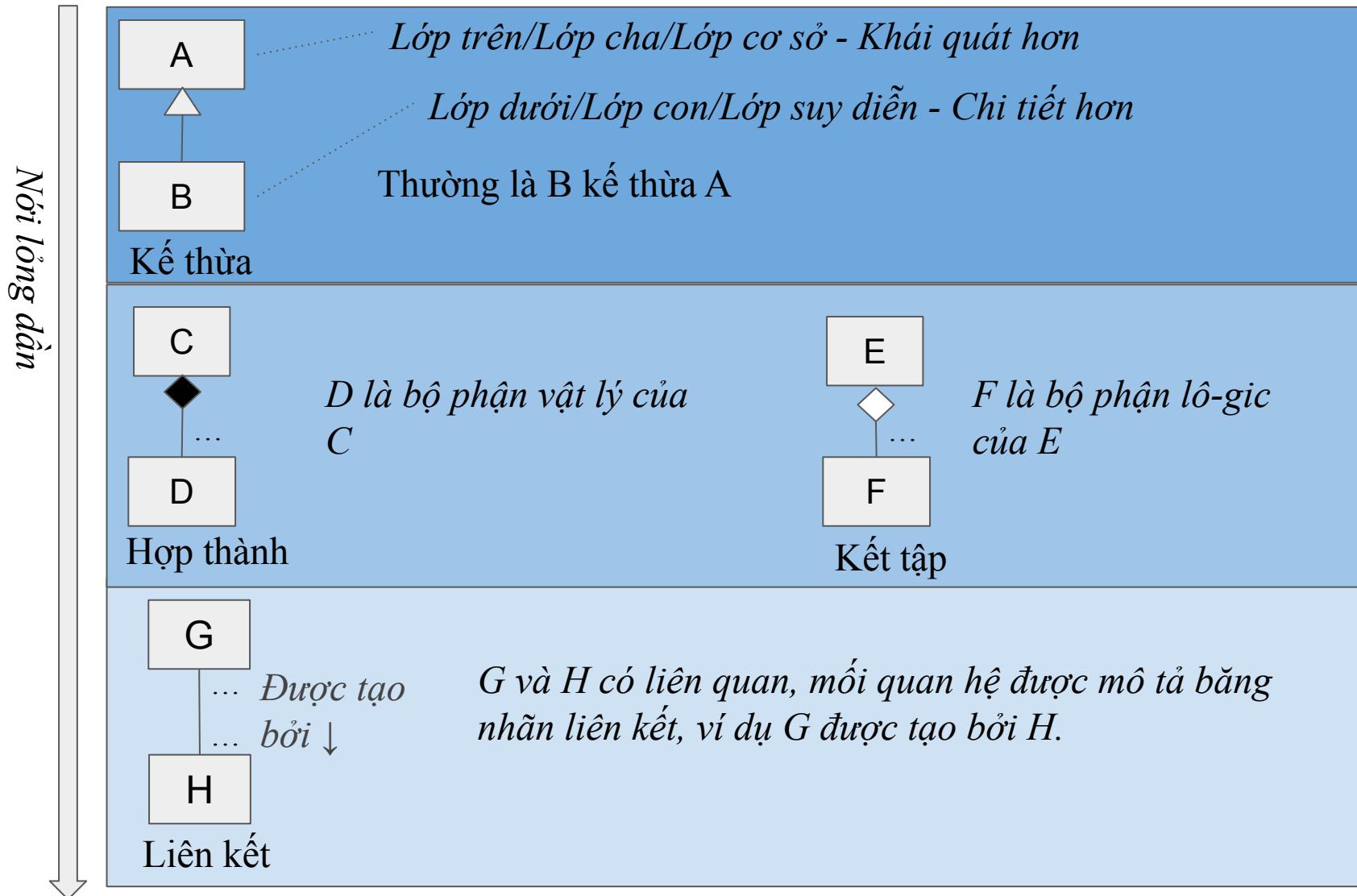
Lớp liên kết  
I

(ở vị trí ... là cơ sở - ràng buộc số lượng đối tượng ở mỗi đầu của mỗi quan hệ)

# Phân loại thuộc tính

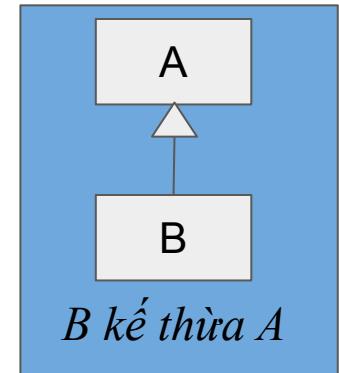
- **Khóa**
  - Một/bộ giá trị của 1/nhóm thuộc tính khóa xác định duy nhất 1 đối tượng.
  - Ví dụ mã khách hàng xác định đúng 1 khách hàng.
- **Thuộc tính đơn trị**
  - Giá trị của thuộc tính không được chia nhỏ thành các thành phần dữ liệu nhỏ hơn nữa.
  - Ví dụ, số lượng sản phẩm là 1 số nguyên.
- **Thuộc tính đa trị**
  - Bao gồm nhiều thành phần dữ liệu nhỏ hơn.
  - Ví dụ địa chỉ có thể bao gồm số nhà, đường phố, quận v.v..
- **Thuộc tính suy diễn**
  - Có thể tính được từ những giá trị của các thuộc tính khác.
  - Ví dụ tuổi của khách hàng có thể được tính từ ngày sinh.

# Các mối quan hệ



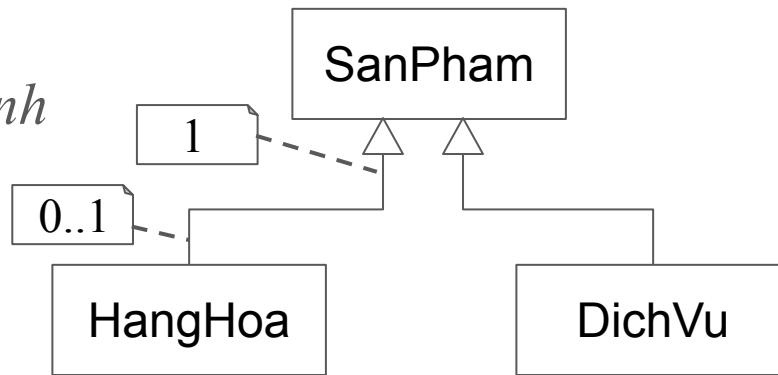
# Quan hệ kế thừa

- Hướng mũi tên đi từ lớp dưới đến lớp trên.
- Cho phép tái sử dụng lớp trên, bổ xung thành phần mới và có thể định nghĩa lại phương thức.
  - Trong các triển khai, B thường sở hữu 1 đối tượng kiểu A.
  - B có các giao diện như A, nhưng hành vi có thể không tương thích (do có thể định nghĩa lại, dù không mong đợi).



## Ví dụ 3.5. Quan hệ kế thừa

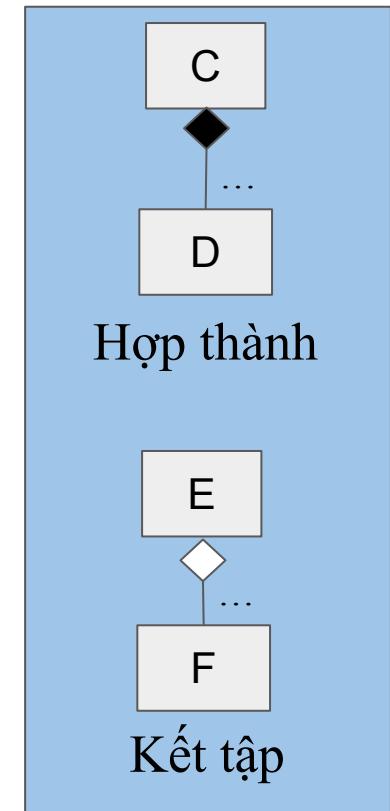
Cơ số mặc định



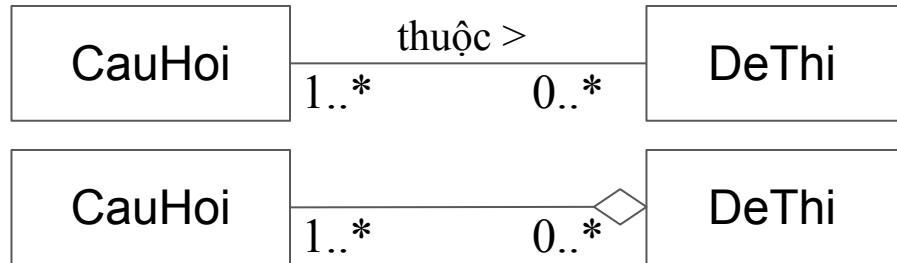
*Hàng hóa và Dịch Vụ là các loại Sản Phẩm / HangHoa và DichVu kế thừa SanPham*

# Các quan hệ bộ phận-tổng thể

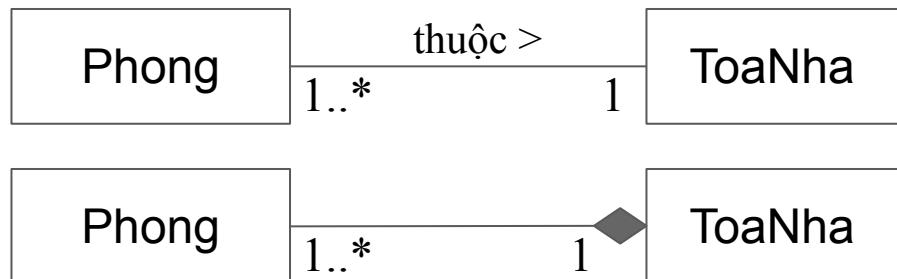
- Mũi tên đi từ bộ phận tới tổng thể.
- Gắn kết cái bộ phận với cái tổng thể.
- Được chia thành 2 mức
  - Hợp thành - chặt hơn:
    - Tổng thể sở hữu bộ phận
    - Đối tượng bộ phận bị hủy khi hủy đối tượng tổng thể.
    - Mỗi bộ phận chỉ có thể thuộc 1 tổng thể.
    - Ý nghĩa: D là bộ phận vật lý của C.
  - Kết tập - lỏng hơn:
    - Tổng thể không sở hữu bộ phận.
    - Bộ phận vẫn có thể tồn tại sau khi hủy cái tổng thể.
    - Mỗi bộ phận có thể thuộc nhiều tổng thể - chia sẻ
    - Ý nghĩa: F là bộ phận lô-gic của E.



## Ví dụ 3.6. Quan hệ bộ phận-tổng thể



*Mỗi câu hỏi có thể thuộc 1 hoặc nhiều đề thi, và cũng có thể không thuộc đề thi nào.*

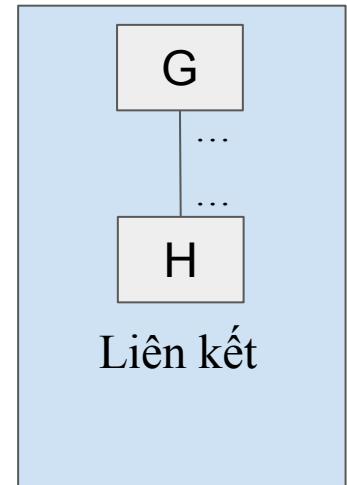


*Mỗi phòng phải thuộc đúng 1 tòa nhà.*

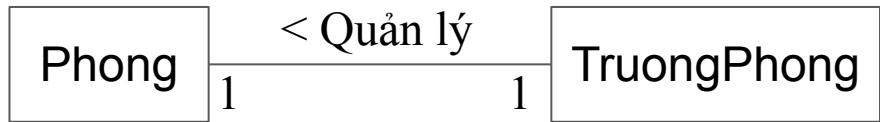
*Các đầu mũi tên kim cương (cú pháp hoa mĩ) được ưa chuộng hơn các nhãn để biểu diễn quan hệ bộ phận-tổng thể.*

# Quan hệ liên kết

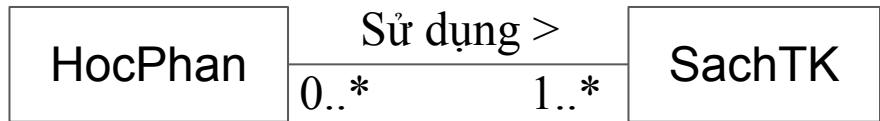
- Loại quan hệ lỏng nhất, ý nghĩa phụ thuộc vào trường hợp cụ thể.
- Được biểu diễn bằng đường nét liền kết nối 2 lớp cùng với tên quan hệ và các tô điểm đầu liên kết:
  - Tên quan hệ - Có thể khác nhau theo mỗi chiều - Có thể bổ xung ký tự ghi chú hướng đọc
  - Các tô điểm, mô tả đầu liên kết:
    - Vai trò của đối tượng ở mỗi đầu
    - Giới hạn hướng duyệt
    - Cơ số - số lượng đối tượng tham gia liên kết
    - Ràng buộc lưu trữ - các đối tượng được lưu như tập hợp, hay danh sách, v.v..



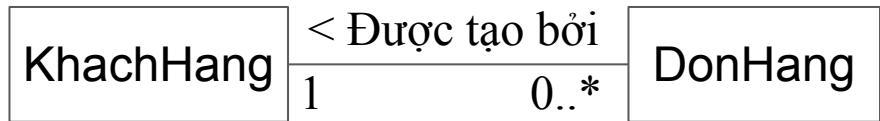
# Ví dụ 3.7. Quan hệ liên kết



Mỗi trường phòng quản lý đúng 1 phòng.

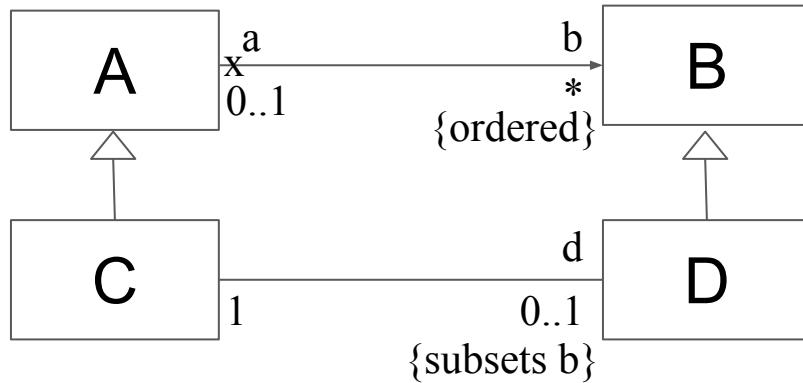


Mỗi học phần sử dụng ít nhất 1 sách tham khảo.



Mỗi đơn hàng được tạo bởi đúng 1 khách hàng.

## Ví dụ 3.8. Các tô điểm cho đầu liên kết



- Ý nghĩa của các thành phần tô điểm liên kết:
  - a, b, d là các tên/vai trò của các đầu liên kết.
  - Các cơ số 0..1 đối với a, \* đối với b, 1 ở đầu lớp C (không đặt tên), và 0..1 đối với d cho biết số lượng đối tượng có thể tham gia vào mối quan hệ với đối tượng ở đầu bên kia.
  - Đầu b là 1 tập có thứ tự.
  - Đầu d là tập con của đầu b, tương đương với:  
**context C inv:** b->includesAll(d)
  - x - đối tượng lớp B không truy cập được đối tác của nó ở phía A; mũi tên - đối tượng lớp A có thể truy cập các đối tác của nó ở phía B; đường nét liền giữa C-D - hướng duyệt chưa xác định.

## Ví dụ 3.9. Khả năng truy cập



Liên kết có thể truy cập cả 2 đầu



Liên kết không truy cập được đầu nào



Liên kết không mô tả truy cập



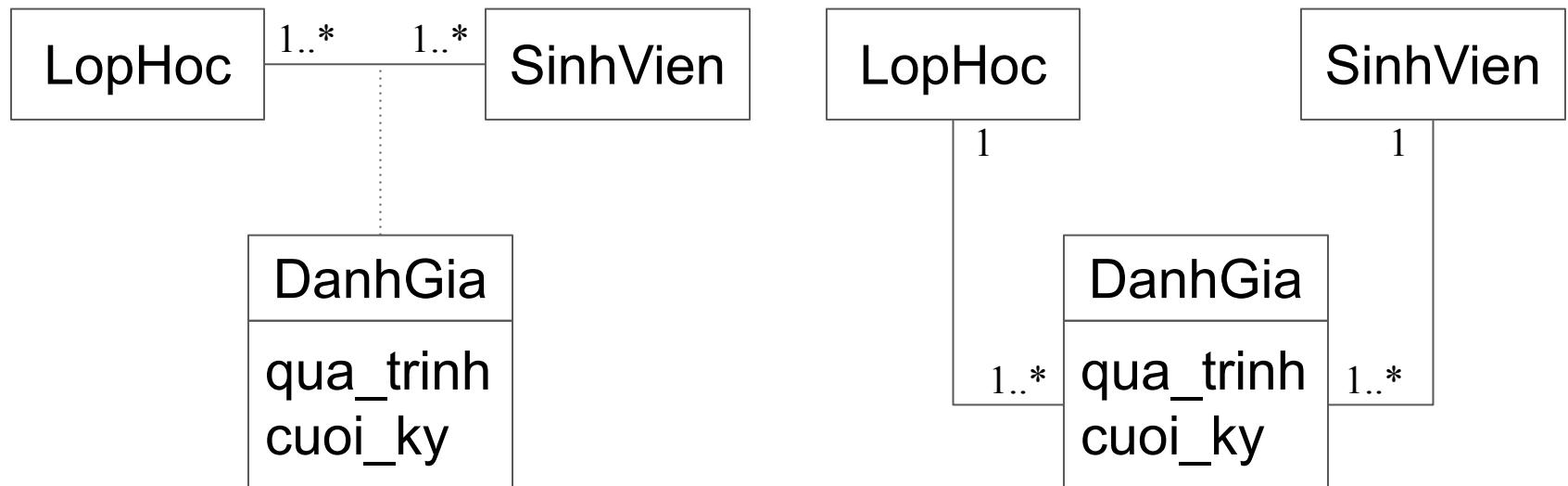
Liên kết có thể truy cập đầu h nhưng không truy cập được đầu g



Liên kết có thể truy cập đầu j, chưa xác định đầu i.

# Ví dụ 3.10. Lớp liên kết

*Biểu diễn các thuộc tính không phải thành phần của 2 đầu liên kết và đồng thời chỉ tồn tại trong liên kết.*



# Các phương thức

- Biểu diễn hành vi/trách nhiệm của đối tượng, những việc mà đối tượng có thể/cần thực hiện, những dịch vụ mà nó cung cấp.
- Để giản lược, giữ tính đơn giản của sơ đồ các phương thức có vai trò bổ trợ thường không được biểu diễn
  - Tạo hoặc hủy đối tượng
    - Hàm tạo
    - Hàm Hủy
  - Đọc hoặc thiết lập giá trị (get/set)
    - Lấy giá trị của thuộc tính
    - Thiết lập giá trị thuộc tính

# Giới hạn truy cập

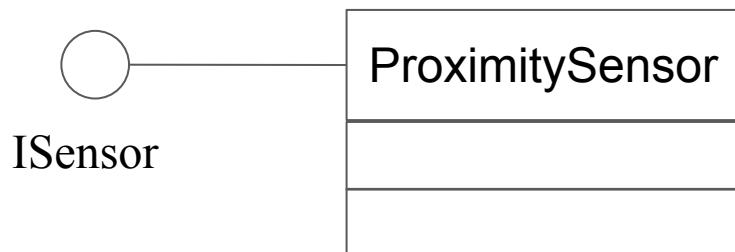
Thường được sử dụng để hỗ trợ đảm bảo tính nhất quán của các biểu diễn

- Công khai (+: public): Không giới hạn quyền truy cập
  - Thường được sử dụng cho phương thức
- Riêng tư (-: private): Giới hạn truy cập trong phạm vi lớp
  - Thường được sử dụng cho thuộc tính.
- Bảo vệ (#: protected): Giới hạn trong phạm vi cây kế thừa
  - (Có thể có khác biệt giữa các môi trường, ví dụ Java vs. C++).
  - Phạm vi nằm giữa riêng tư và công khai.

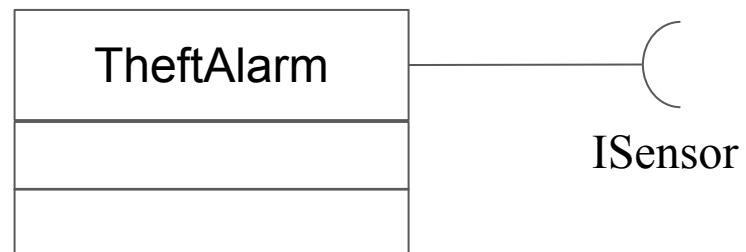
# Lớp giao diện

- (Lớp) Giao diện là 1 kiểu lớp đặc biệt:
  - Các phương thức phải được thiết lập truy cập công khai.
  - Không có đối tượng cụ thể.
- Giao diện mô tả 1 tập tính năng nhưng không cung cấp triển khai của các tính năng đó.
  - Sau đó được triển khai trong các lớp cụ thể.
- Trong UML giao diện có thể được biểu diễn bằng hình chữ nhật với nhãn loài hoặc bằng biểu tượng riêng.

# Ví dụ 3.11. Giao diện

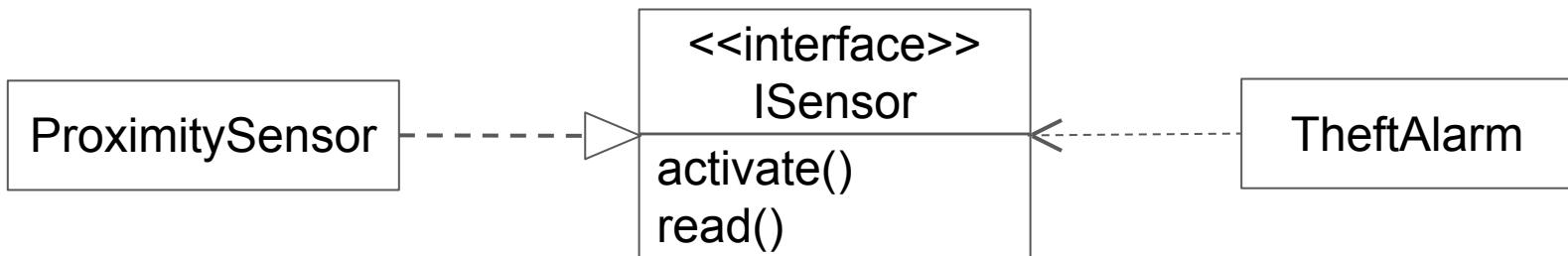


*ProximitySensor triển khai/cung cấp giao diện `ISensor`*

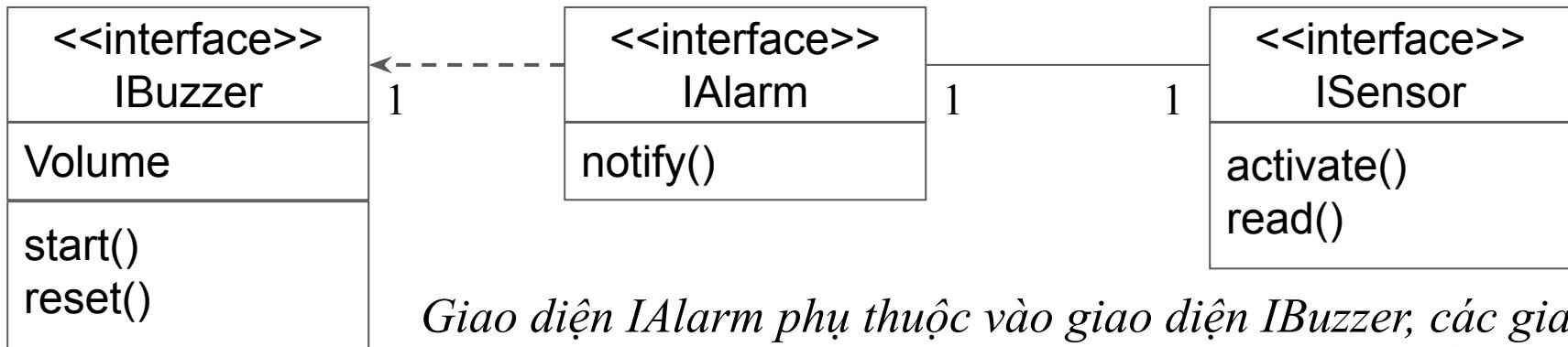


*TheftAlarm phụ thuộc vào/sử dụng giao diện `ISensor`*

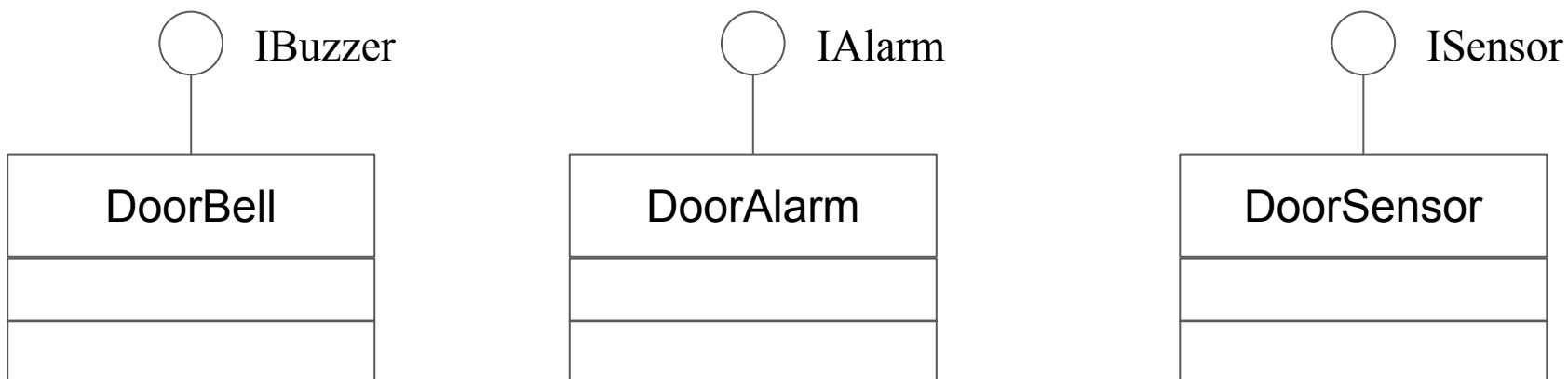
VS.



## Ví dụ 3.12. Kết hợp nhiều giao diện



Giao diện **IAlarm** phụ thuộc vào giao diện **IBuzzer**, các giao diện **IAlarm** và **ISensor** hình thành 1 giao thức 2 chiều.



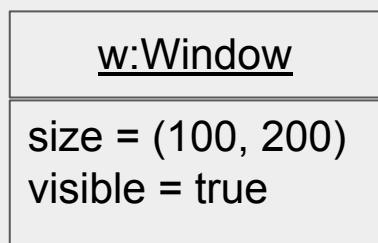
Các lớp **DoorBell**, **DoorAlarm** và **DoorSensor** tuy được biểu diễn độc lập, nhưng các đối tượng của chúng vẫn có thể tương tác thông qua các giao diện.

# Sơ đồ đối tượng

- Biểu diễn các đối tượng với các giá trị cụ thể của các thuộc tính và quan hệ cụ thể giữa các đối tượng
  - Đối tượng = Trường hợp cụ thể (phần tử) của lớp
- Có thể hỗ trợ tiến trình xây dựng sơ đồ lớp: Phân tích tương tác giữa các đối tượng thường dễ hơn phân tích mối quan hệ giữa các lớp.

# Ví dụ 3.13. Biểu diễn đối tượng

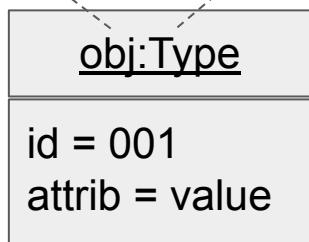
Đối tượng w thuộc  
lớp Window



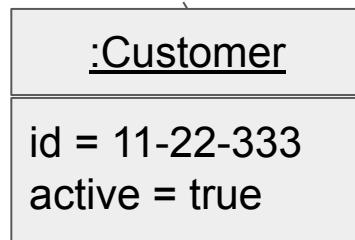
Lớp Window



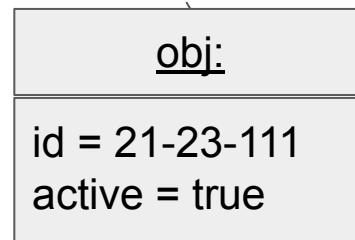
Tên đối tượng      Tên lớp



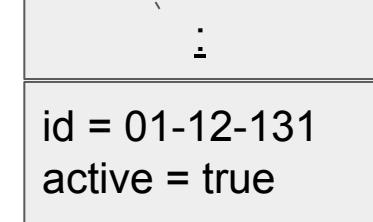
Đối tượng không tên  
(thuộc lớp Customer)



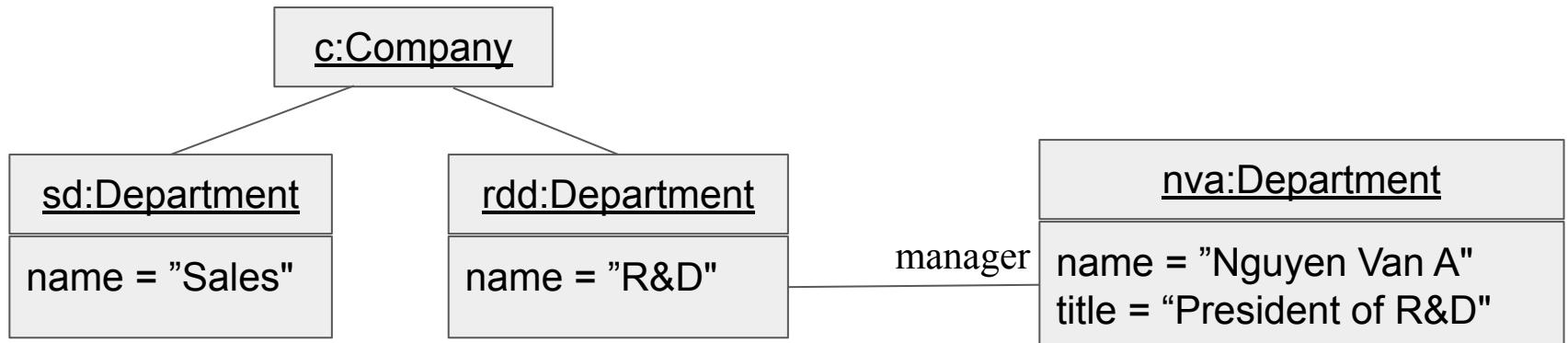
Đối tượng không lớp  
(chưa xác định lớp)



Đối tượng không tên  
không lớp



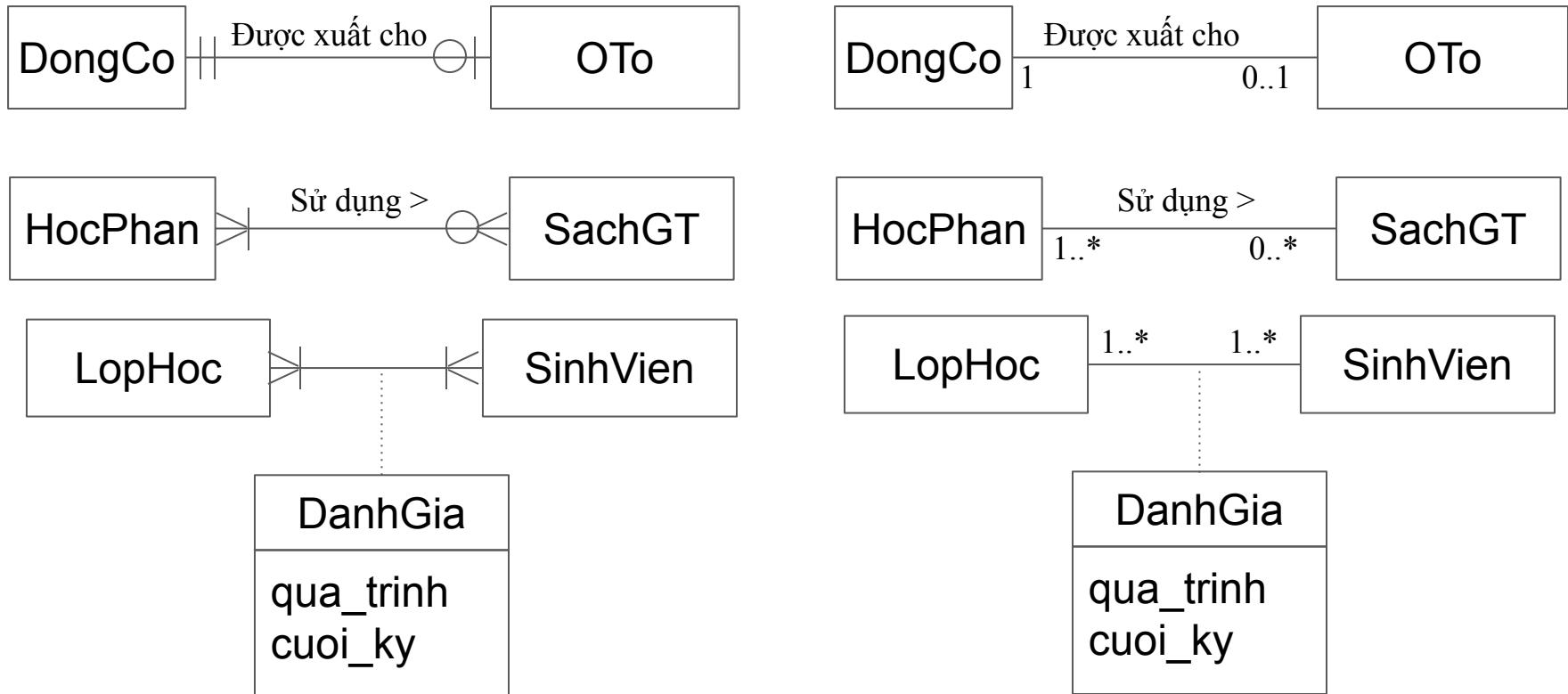
# Ví dụ 3.14. Sơ đồ đối tượng



# Sơ đồ thực thể liên kết vs. Sơ đồ lớp

- Tổng quan:
  - Sơ đồ thực thể-liên kết (ERD) mô tả các kiểu thực thể cùng với quan hệ giữa các thực thể.
    - Có nhiều hệ ký hiệu khác nhau;
    - Thường được sử dụng trong thiết kế CSDL quan hệ.
  - Sơ đồ lớp mô tả các lớp cùng với quan hệ giữa các đối tượng.
    - Là 1 sơ đồ UML, hệ ký hiệu trong quy chuẩn quốc tế.
    - Được sử dụng cho nhiều mục đích như mô hình hóa lĩnh vực, biểu diễn mô hình dữ liệu hướng đối tượng.
- Phạm vi diễn đạt:
  - Sơ đồ lớp có khả năng diễn đạt mạnh hơn. Có thể bảo toàn ý nghĩa khi chuyển đổi sơ đồ thực thể-liên kết sang sơ đồ lớp.
    - Kiểu thực thể => Lớp lĩnh vực chỉ chứa thuộc tính, không có phương thức.
    - Liên kết => Liên kết
    - Thực thể liên kết => Lớp liên kết.
    - v.v..

# Ví dụ 3.15. ERD vs. sơ đồ lớp



(Cơ số trong ERD chỉ có thể là 1 trong 4 trường hợp  $\{0..1, 1, 0..*, 1..*\}$ )

# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập



# OCL là gì?

- Ngôn ngữ ràng buộc đối tượng - Object Constraint Language
  - Được phát triển từ năm 1995 ở IBM.
  - Sau đó IBM đề xuất và OMG thông qua quy chuẩn.
  - Được sử dụng để đặc tả UML
  - Phiên bản mới nhất song hành với UML:
    - OCL 2.4: <https://www.omg.org/spec/OCL/2.4/PDF>
    - UML 2.4.1: <https://www.omg.org/spec/UML/2.4.1>
- Được sử dụng để bổ xung các chi tiết mô tả cho sơ đồ lớp
  - Các biến đổi với các thuộc tính, tiền điều kiện và hậu điều kiện đối với các phương thức, v.v..

# Các ràng buộc và ngũ cảnh

- Ràng buộc là 1 giới hạn đối với 1 hoặc nhiều thành phần của 1 mô hình hướng đối tượng hoặc hệ thống
  - Tiêu biểu như: Bất biến, tiền điều kiện, hậu điều kiện.
- Ngũ cảnh là cái (lớp, phương thức, v.v..) được thiết lập ràng buộc
  - Ràng buộc được thiết lập ở mức lớp sẽ có hiệu lực cho các đối tượng thuộc lớp đó.
- Ràng buộc có thể được thêm vào như ghi chú trên sơ đồ UML hoặc tách biệt trong biểu mẫu đặc tả riêng.

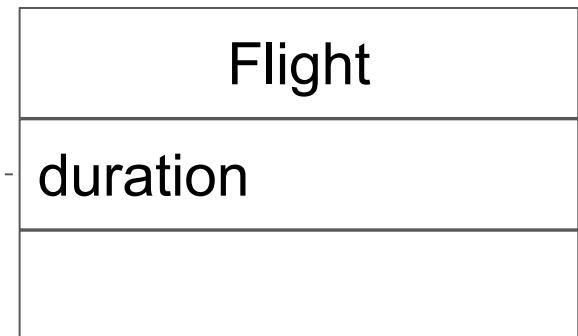
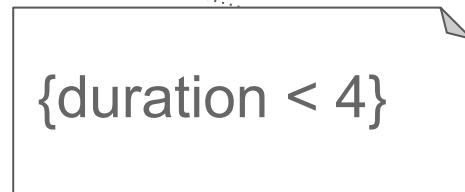
# Bất biến

- Bất biến đối với đối tượng là ràng buộc phải đúng trong suốt thời gian tồn tại của đối tượng đó.
- Cú pháp:  
**context <lớp>**  
**inv [<tên ràng buộc>]:**  
**<Biểu thức ràng buộc OCL>**

# Các mô tả bất biến tương đương

- **context Flight inv:** self.duration < 4
  - self - Đối tượng đang được kiểm tra ràng buộc.
- **context Flight inv:** duration < 4
- **context Flight inv flightDuration:** duration < 4
  - flightDuration - Tên được đặt cho ràng buộc

Ràng buộc OCL như  
ghi chú trong sơ đồ  
UML



# Tiền điều kiện và hậu điều kiện

- Được áp dụng cho các phương thức
- Tiền điều kiện phải đúng trước khi thực hiện phương thức.
- Hậu điều kiện phải đúng sau khi thực hiện phương thức.
- Cú pháp:

**context** <lớp>::<phương thức>(<các tham số>)

**pre|post** [<tên ràng buộc>]:

<Biểu thức ràng buộc OCL>

## Ví dụ 3.16. Mô tả điều kiện

**context** Flight::shiftDeparture( $t$ : Integer)

**pre:**  $t > 0$

**post:** result = departureTime@pre +  $t$  + duration

-- Chúng ta muốn trả về thời gian đến đã cập nhật

-- @pre - chỉ được sử dụng trong hậu điều kiện, để chỉ định trạng thái ban đầu.

Flight
departTime /arrivalTime duration
shiftDeparture( $t$ :Integer)

# Kế thừa ràng buộc

Các hệ quả của nguyên lý khả thay Liskov

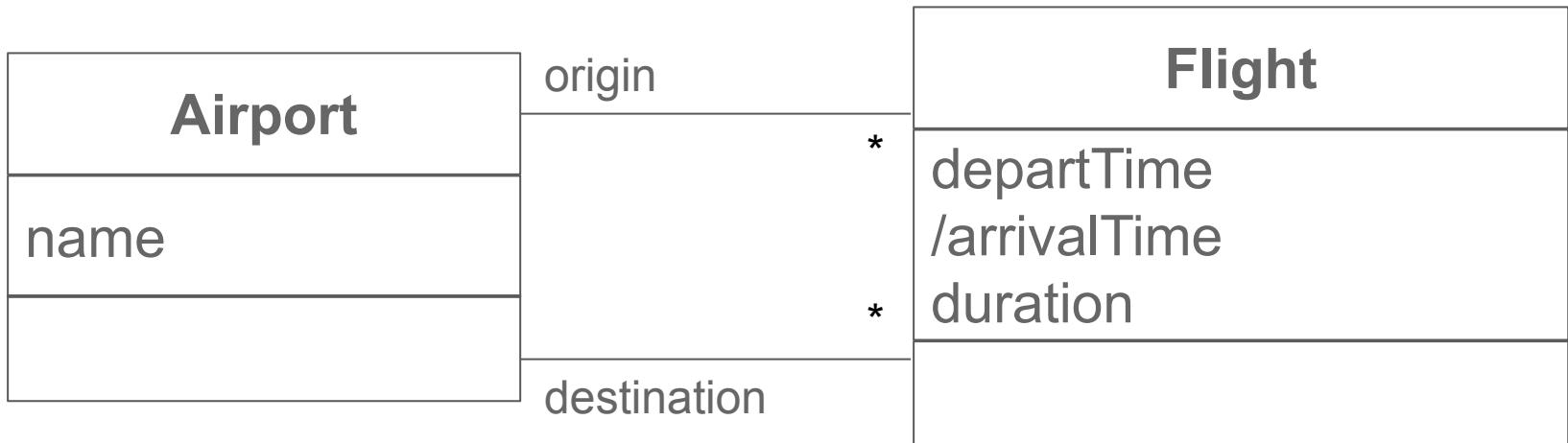
- Các lớp dưới kế thừa các bất biến của lớp trên
  - Lớp dưới có thể tăng cường các bất biến nhưng không được nói lỏng chung.
- Lớp dưới có thể nới lỏng tiền điều kiện trong lớp trên
  - **context** SuperClass::foo(i:Integer) **pre:** i > 100
  - **context** SubClass::foo(i:Integer) **pre:** i > 0
  - => Lớp dưới vẫn có thể xử lý dữ liệu đầu vào như lớp trên.
- Lớp dưới có thể tăng cường hậu điều kiện trong lớp trên
  - **context** SuperClass::foo(): Integer **post:** result > 0
  - **context** SuperClass::foo(): Integer **post:** result > 10
  - => Biểu thức gọi phương thức bằng giao của diện lớp trên vẫn thu được kết quả > 0 dù đối tượng thuộc lớp dưới.

# Các thành phần của biểu thức ràng buộc

- Các kiểu cơ sở:
  - Boolean
  - Integer
  - Real
  - String
- Các lớp từ mô hình UML và các thành phần của chúng
  - Thuộc tính
  - Thao tác truy xuất
- Liên kết từ mô hình UML
  - Bao gồm tên vai trò ở mỗi đầu liên kết

# Các biểu thức duyệt

- Duyệt theo liên kết - Được sử dụng để truy cập các đối tượng liên quan, bắt đầu từ đối tượng ngũ cảnh



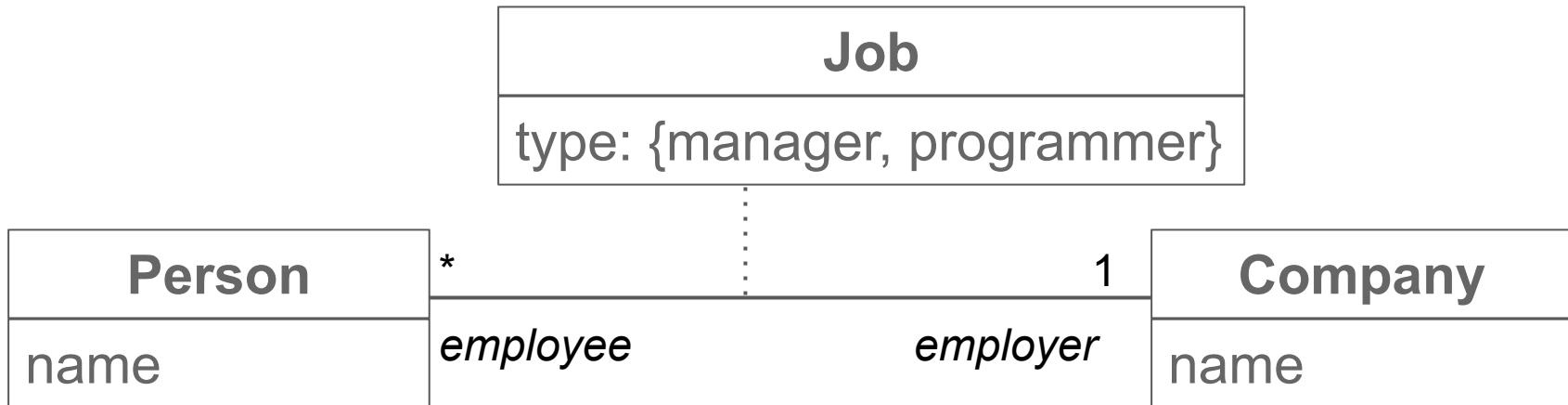
**context** Flight

**inv:** origin != destination

**inv:** origin.name == "Hanoi"

# Duyệt lớp liên kết

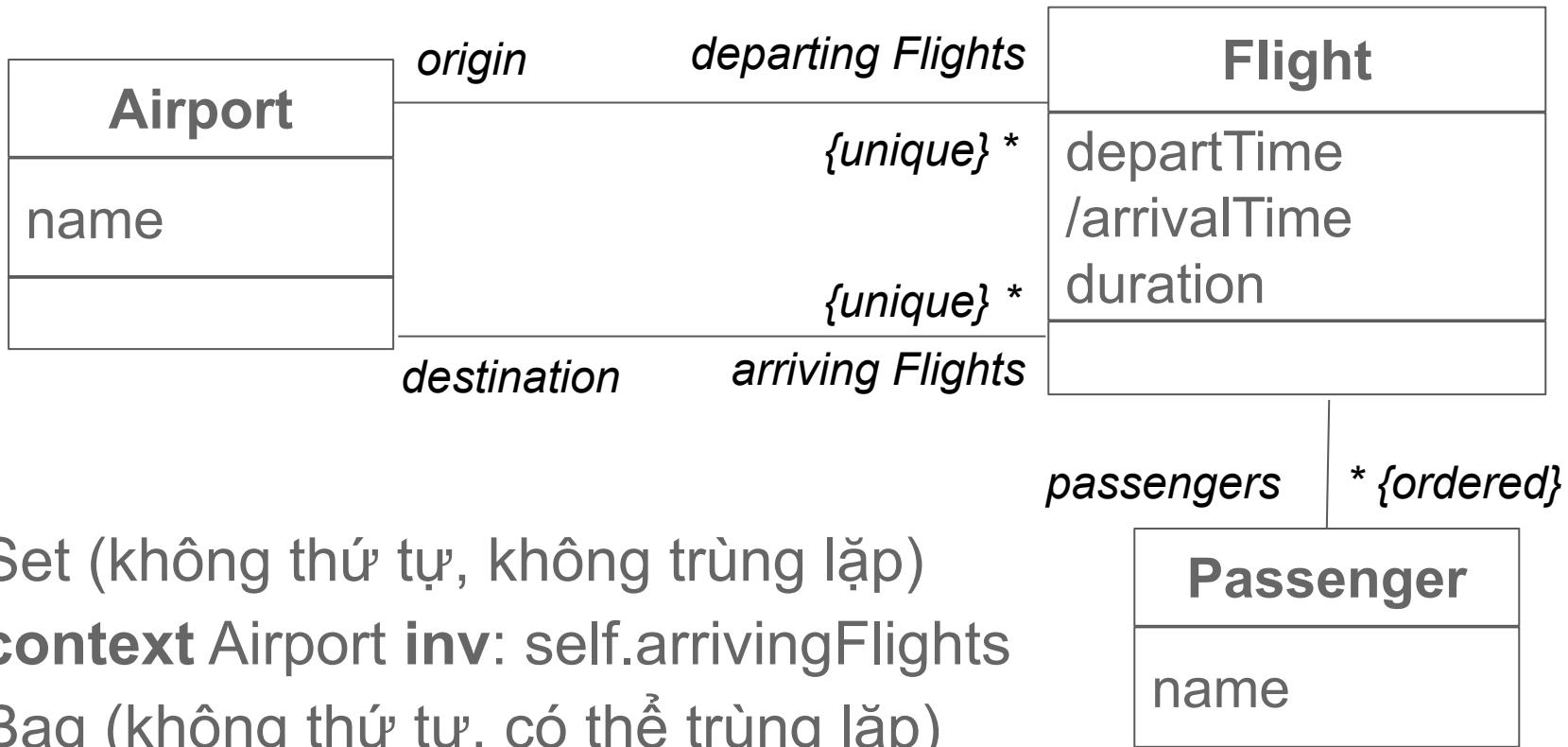
- Lớp liên kết không có tên vai trò, vì vậy biểu thức OCL phải sử dụng tên lớp, bắt đầu với chữ thường.



**context Person inv:**

```
if self.name = "Ivan Ivanov" then
    job.type = #manager
else
    job.type = #programmer
endif
```

# Các cấu trúc lưu trữ trong OCL



- Set (không thứ tự, không trùng lặp)  
**context** `Airport` **inv**: `self.arrivingFlights`
- Bag (không thứ tự, có thể trùng lặp)  
**context** `Airport` **inv**: `self.arrivingFlights.passengers.name`
- Sequence (có thứ tự, có thể trùng lặp)  
**context** `Flight` **inv**: `self.passengers`

# Các thao tác với cấu trúc lưu trữ

- collect: collection->collect(expr) hoặc collection.expr
  - Trả về 1 túi (bag) giá trị của biểu thức expr trên các phần tử của collection.
- select: collection->select(expr)
  - Trả về tập con của collection bao gồm các phần tử có expr đúng.
- forAll: collection->forAll(expr)
  - Thao tác forAll đúng nếu expr đúng với tất cả các phần tử của collection
- exists: collection->exists(expr)
  - Đúng nếu expr đúng với ít nhất 1 phần tử trong collection

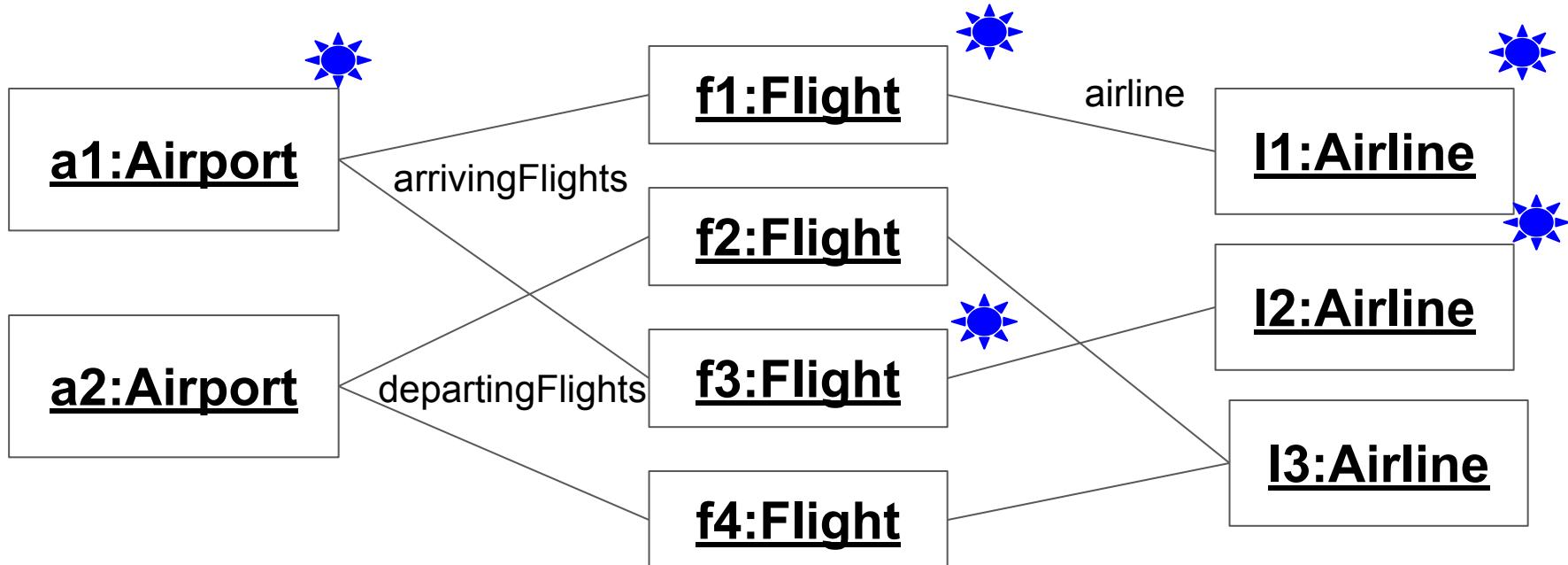
...

## Ví dụ 3.17. Thao tác *collect*



**context** Airport inv:

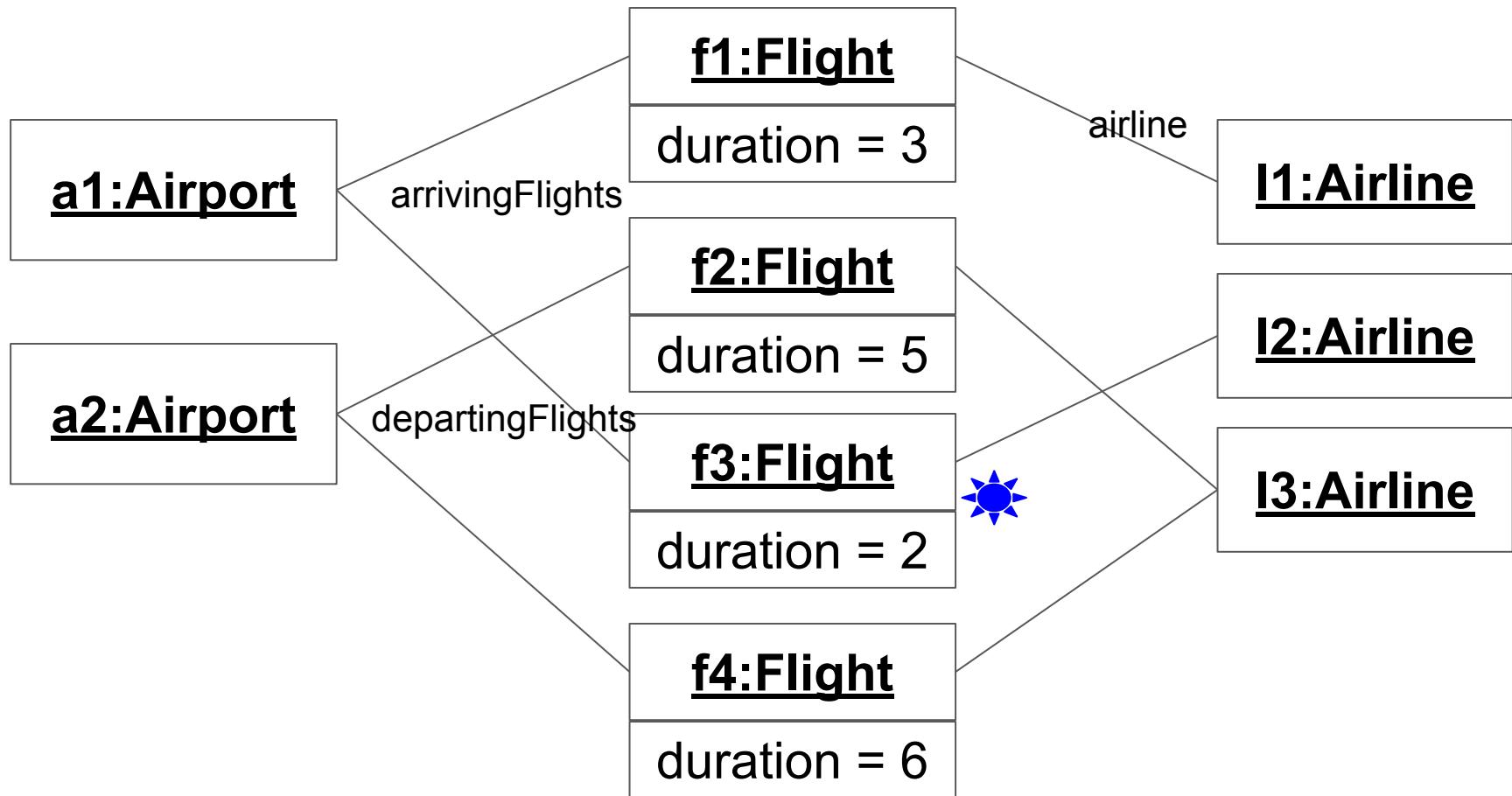
self.arrivingFlights->collect(airline)->notEmpty()



# Ví dụ 3.18. Thao tác select

context Airport **inv:**

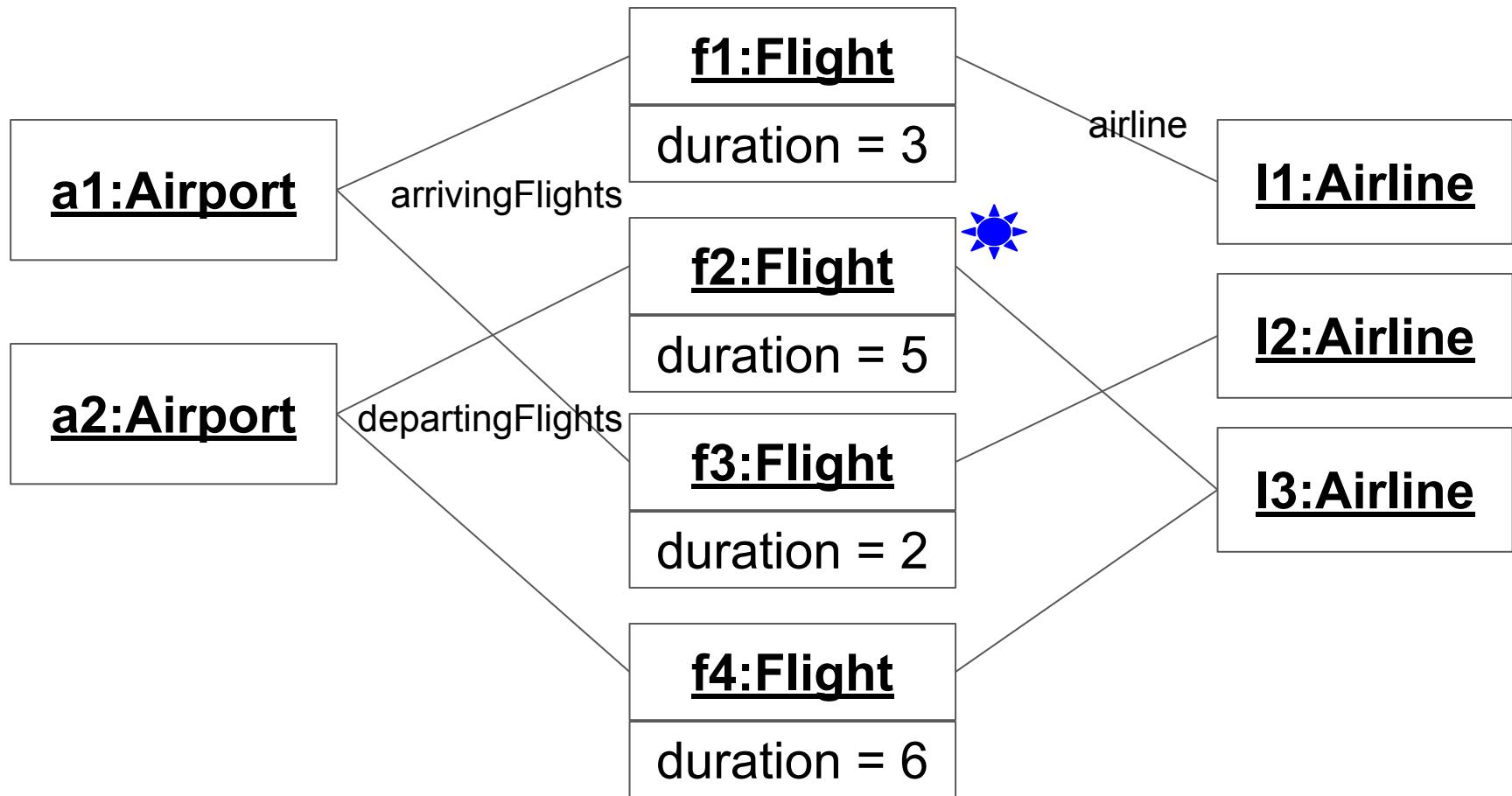
self.departingFlights->select(duration < 4)->notEmpty()



# Ví dụ 3.19. Thao tác *exists*

context Airport **inv:**

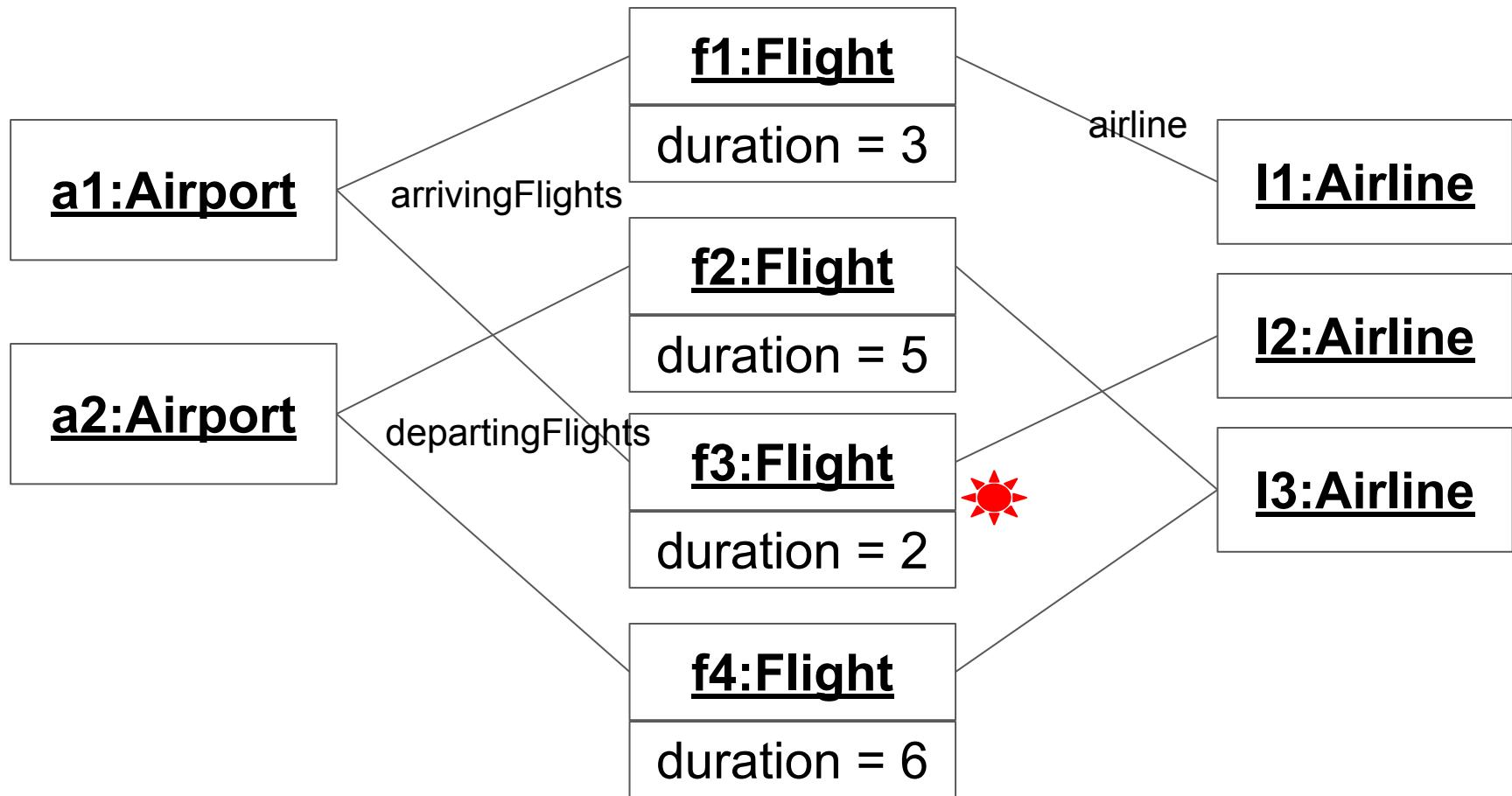
self.departingFlights->exists(duration = 5)



# Ví dụ 3.20. Thao tác *forAll*

context Airport **inv:**

self.arrivingFlights->forAll(duration > 2)



# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập

# Thẻ CRC

## Class Responsibilities and Collaborations

- Được sử dụng để đặc tả lớp
  - Còn có thể được sử dụng để đóng vai đối tượng trong kiểm tra các kịch bản ca sử dụng
- Các trách nhiệm:
  - Được biểu diễn như các phương thức
  - Biết gì? - các thao tác tra cứu thông tin
  - Làm gì? - những thao tác xử lý
- Lưu dữ liệu gì:
  - Danh sách thuộc tính

# Thẻ CRC<sub>(2)</sub>

- Đối tác:
  - Các đối tượng làm việc cùng nhau để đáp ứng 1 yêu cầu
    - Đối tượng yêu cầu (khách)
    - Đối tượng phản hồi (chủ)
  - Các tương tác được quy định theo hình thức tương tự hợp đồng
    - Nội dung chi tiết được trình bày trong phần thiết kế

# Mẫu thẻ CRC

Tên lớp:	ID:	Kiểu:
Mô tả:		CSD:
Các trách nhiệm		Các đối tác
Các thuộc tính		
Các mối quan hệ		
Kế thừa:		
Bộ phận-tổng thể:		
Liên kết:		

# Ví dụ 3.21. Đặc tả lớp với thẻ CRC

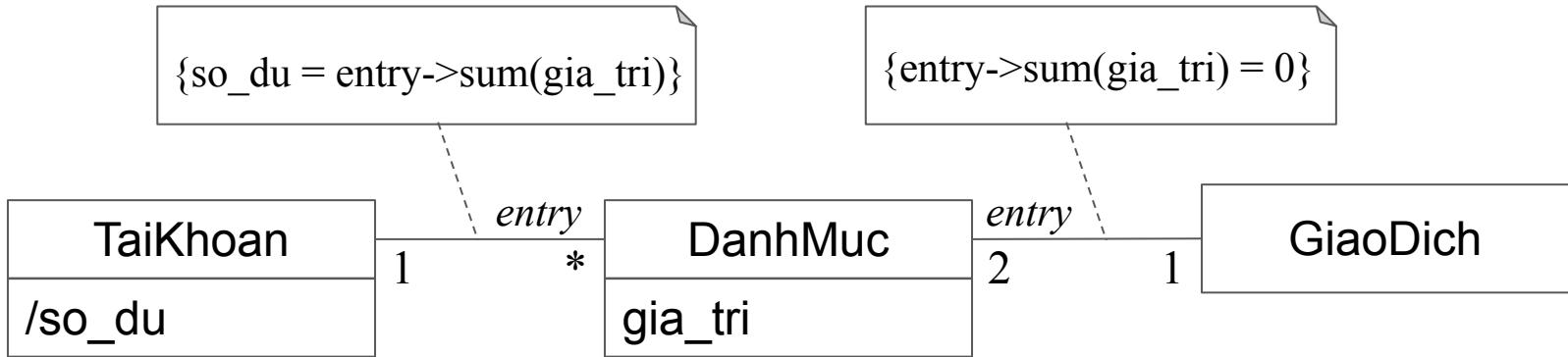
Tên lớp: Bản khảo sát	ID: 1	Kiểu: Lĩnh vực
Mô tả: Danh sách câu hỏi và câu trả lời được sử dụng để xác định dịch vụ.	CSD: UC01	
<b>Các trách nhiệm</b>		<b>Các đối tác</b>
Hiển thị câu hỏi Lưu câu trả lời		Câu hỏi khảo sát --
<b>Các thuộc tính</b>		
Mã số Danh sách câu hỏi Danh sách câu trả lời		
<b>Các mối quan hệ</b>		
<b>Kế thừa:</b> -- <b>Bộ phận-tổng thể:</b> Bao gồm các câu hỏi khảo sát <b>Liên kết:</b> Khách hàng, nhu cầu dịch vụ		

# Nội dung

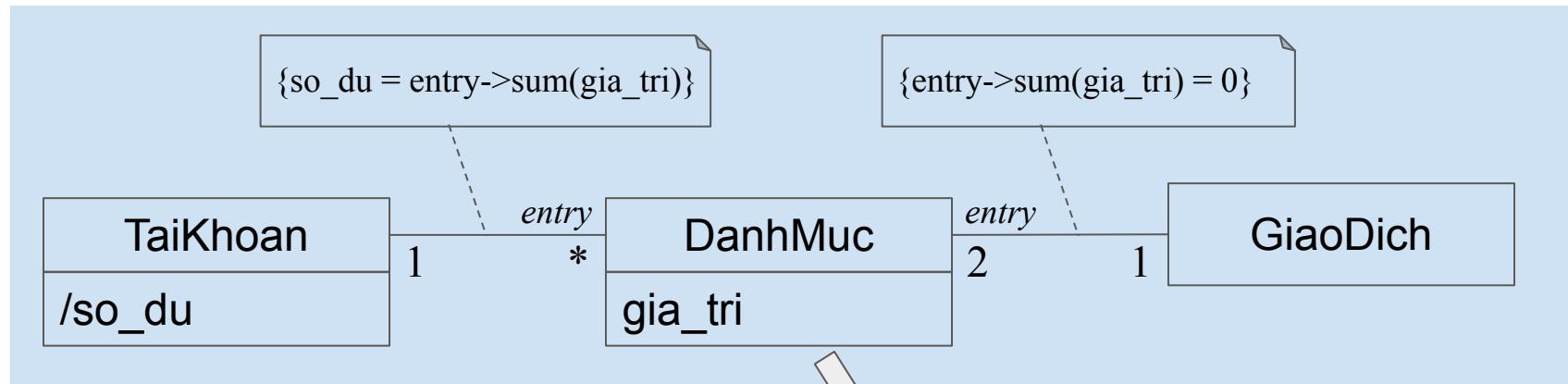
- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập



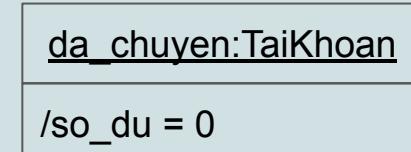
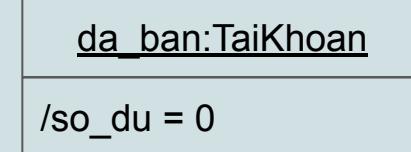
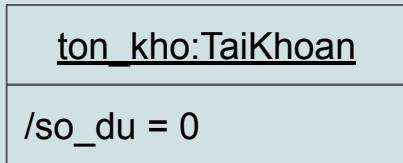
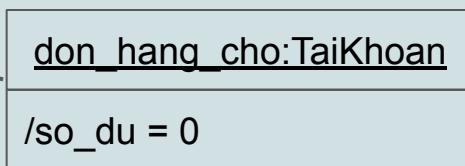
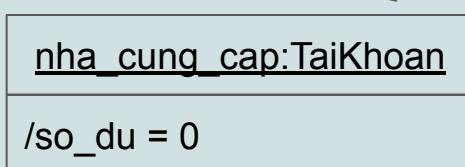
# Mẫu tài khoản/giao dịch



# Ví dụ 3.22. Các đối tượng tài khoản/giao dịch

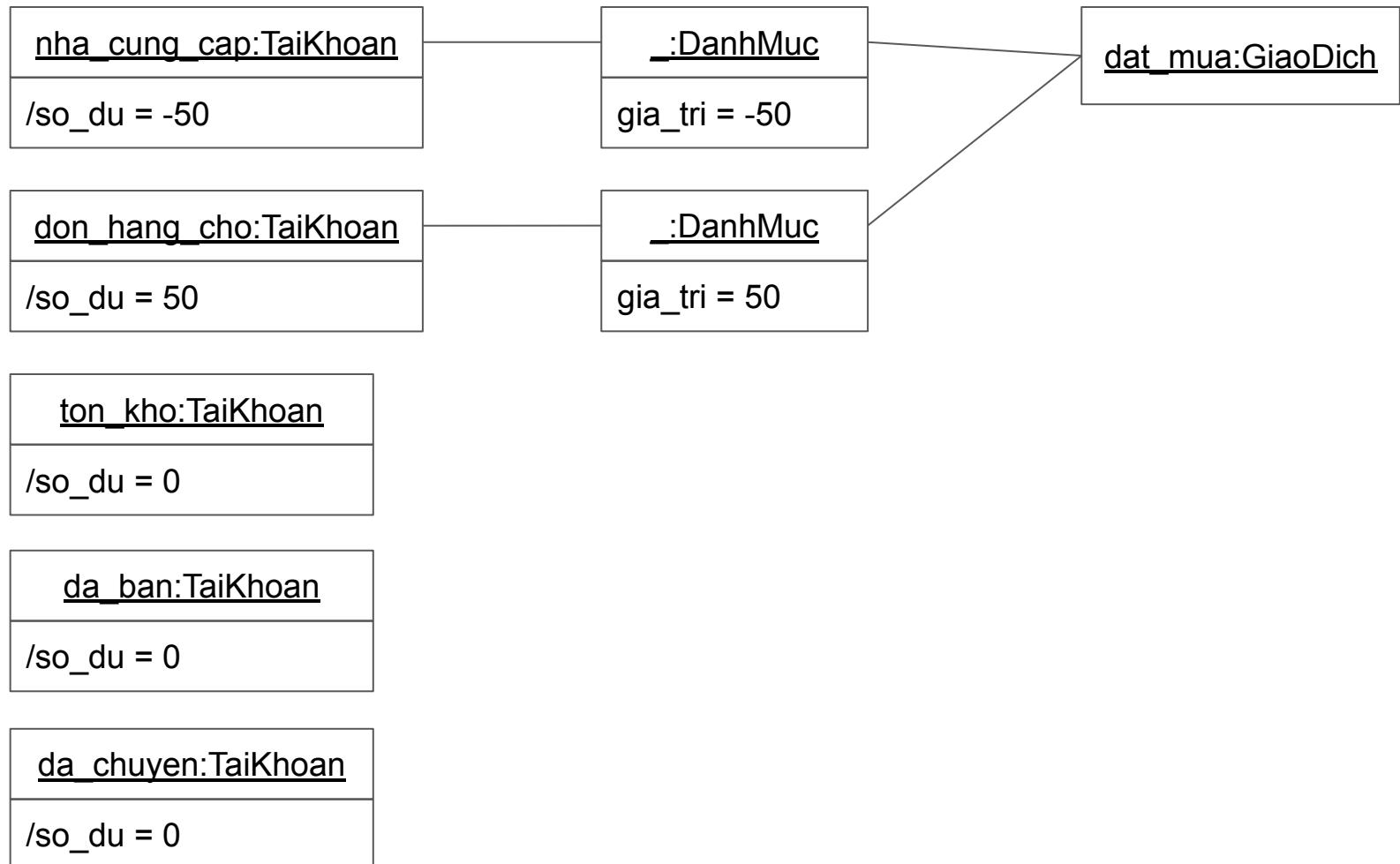


Trạng thái ban đầu



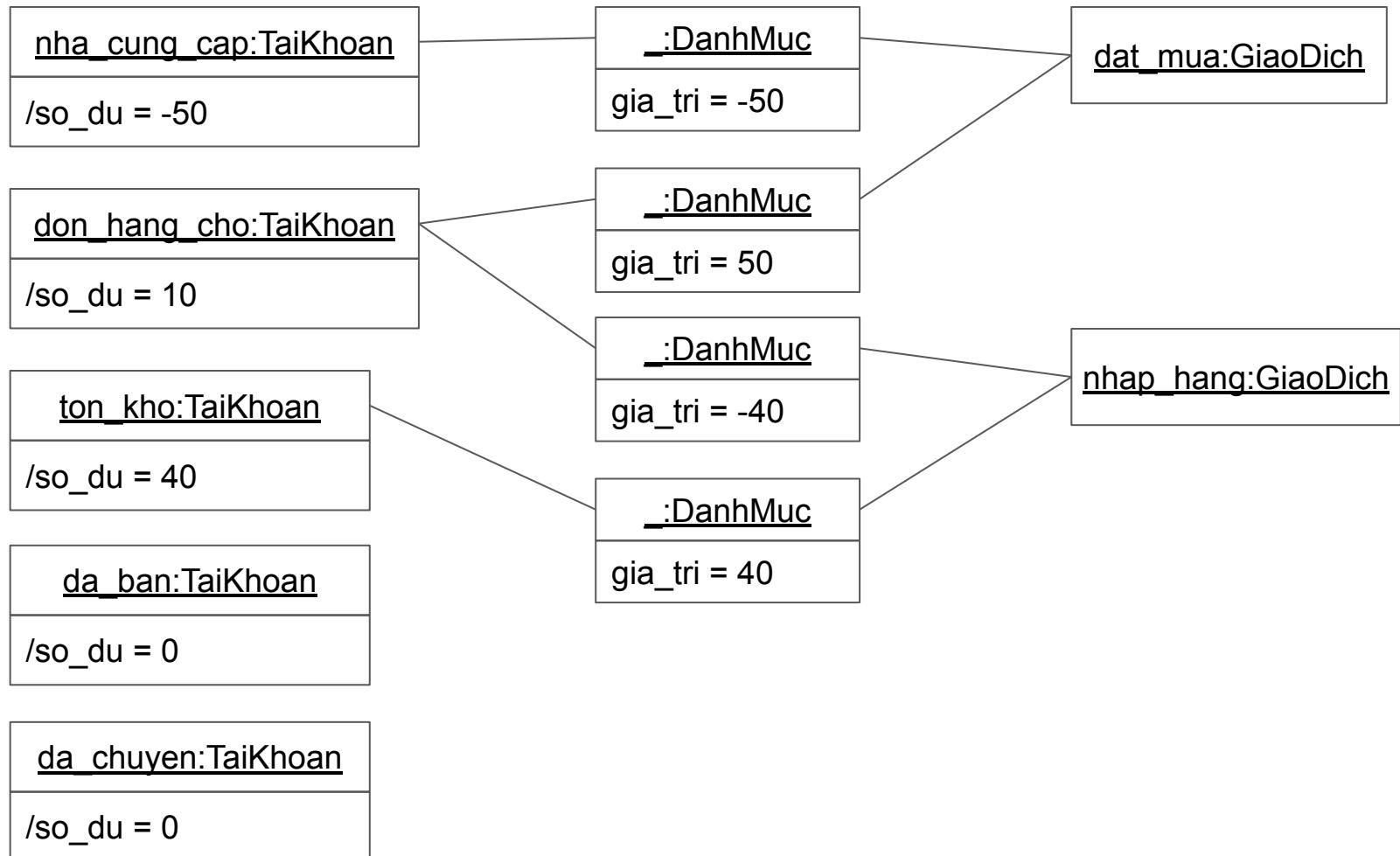
Sơ đồ đối tượng

## Ví dụ 3.22. Các đối tượng tài khoản/giao dịch<sub>(2)</sub>



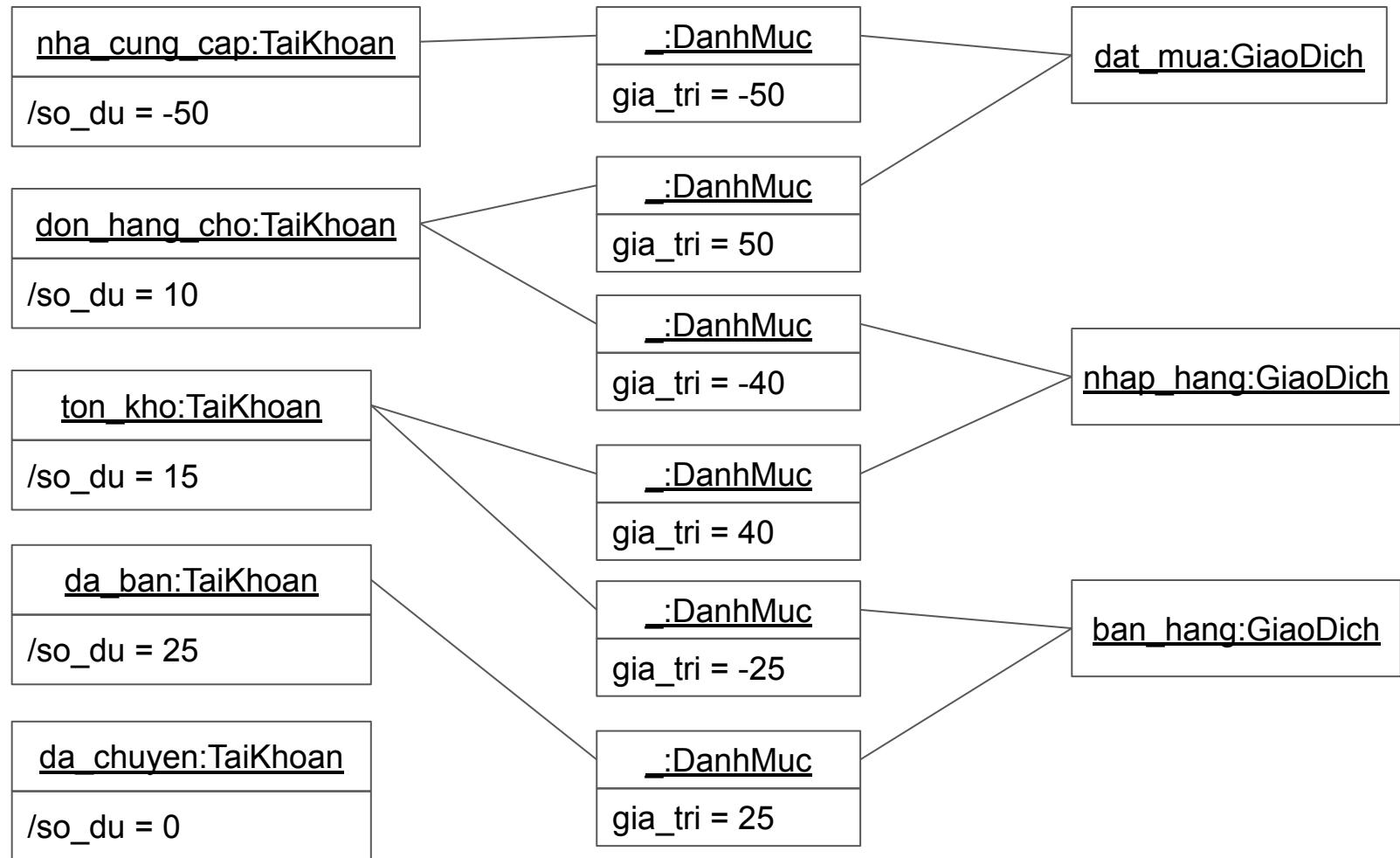
*Trạng thái sau khi đặt mua từ nhà cung cấp*

# Ví dụ 3.22. Các đối tượng tài khoản/giao dịch<sub>(3)</sub>



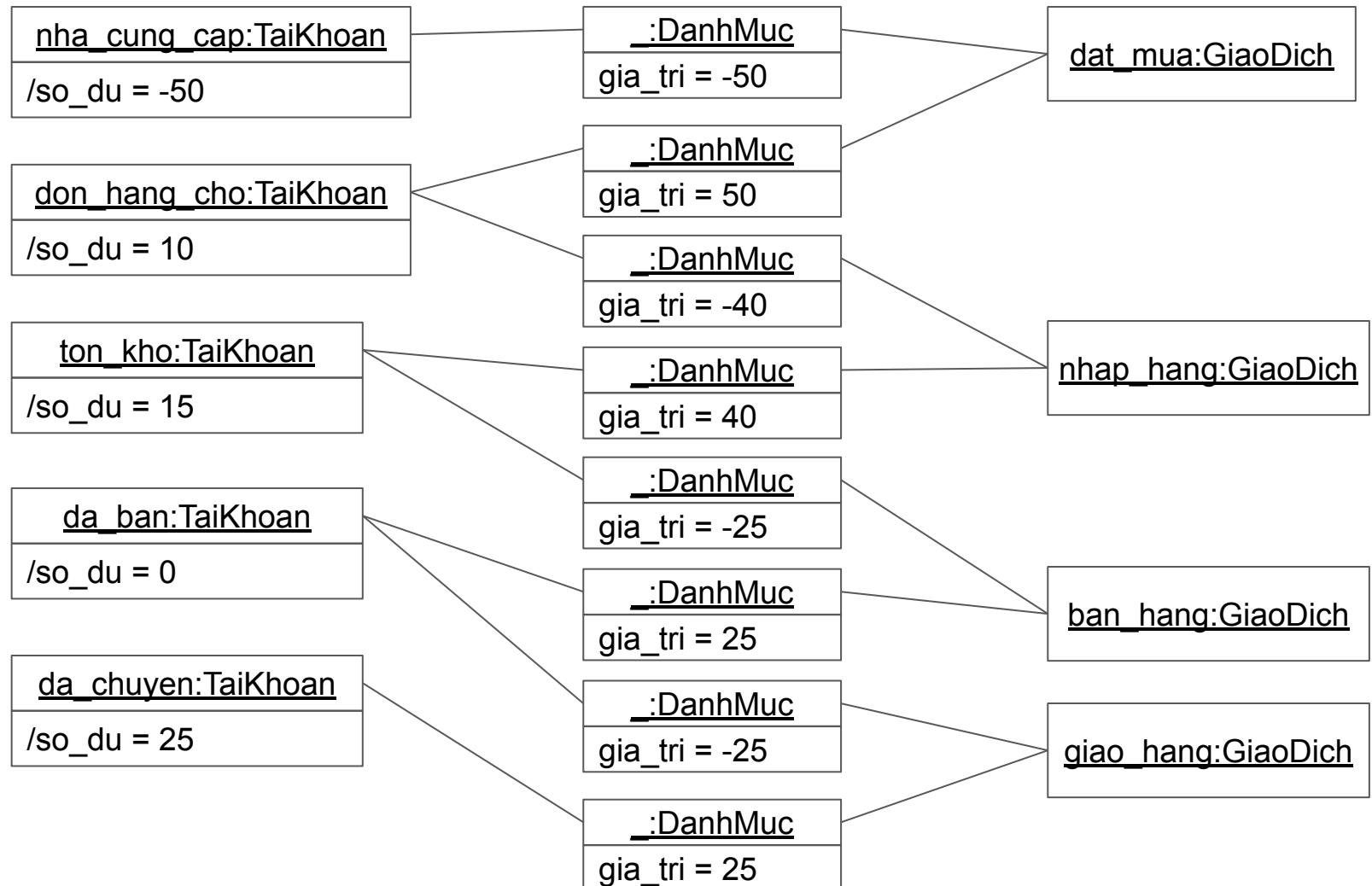
*Trạng thái sau khi nhập 1 phần hàng từ nhà cung cấp*

# Ví dụ 3.22. Các đối tượng tài khoản/giao dịch<sub>(4)</sub>



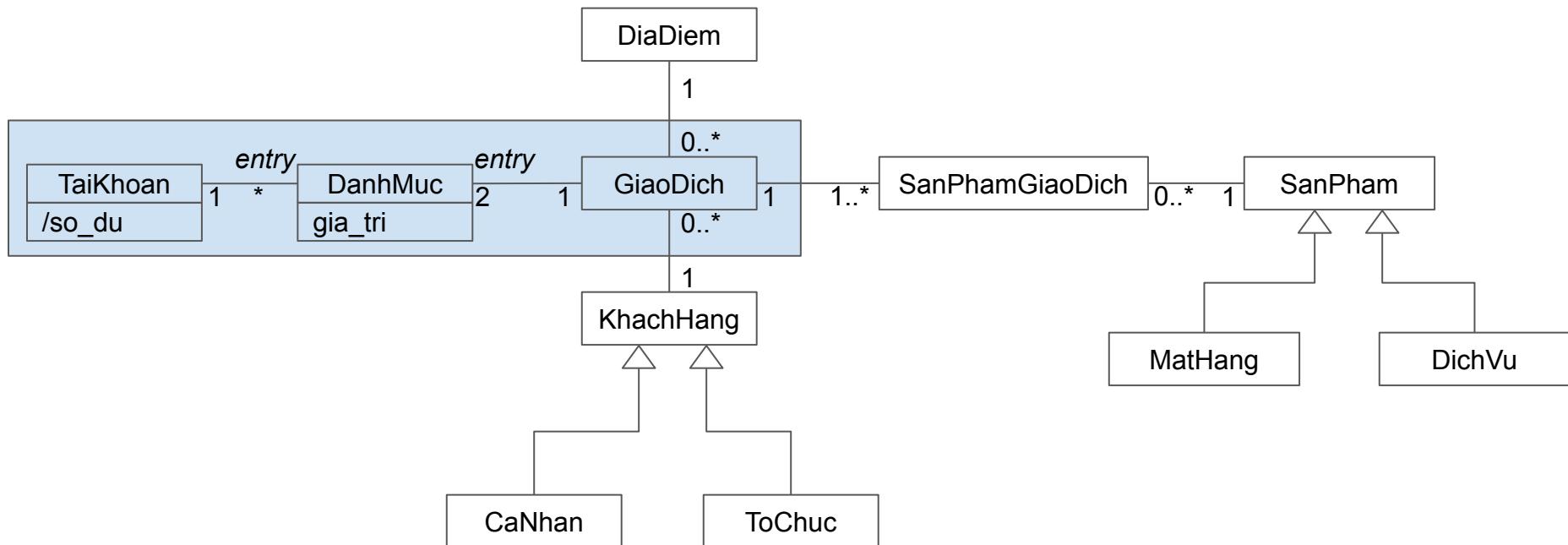
*Trạng thái sau khi bán cho khách hàng*

# Ví dụ 3.22. Các đối tượng tài khoản/giao dịch (5)

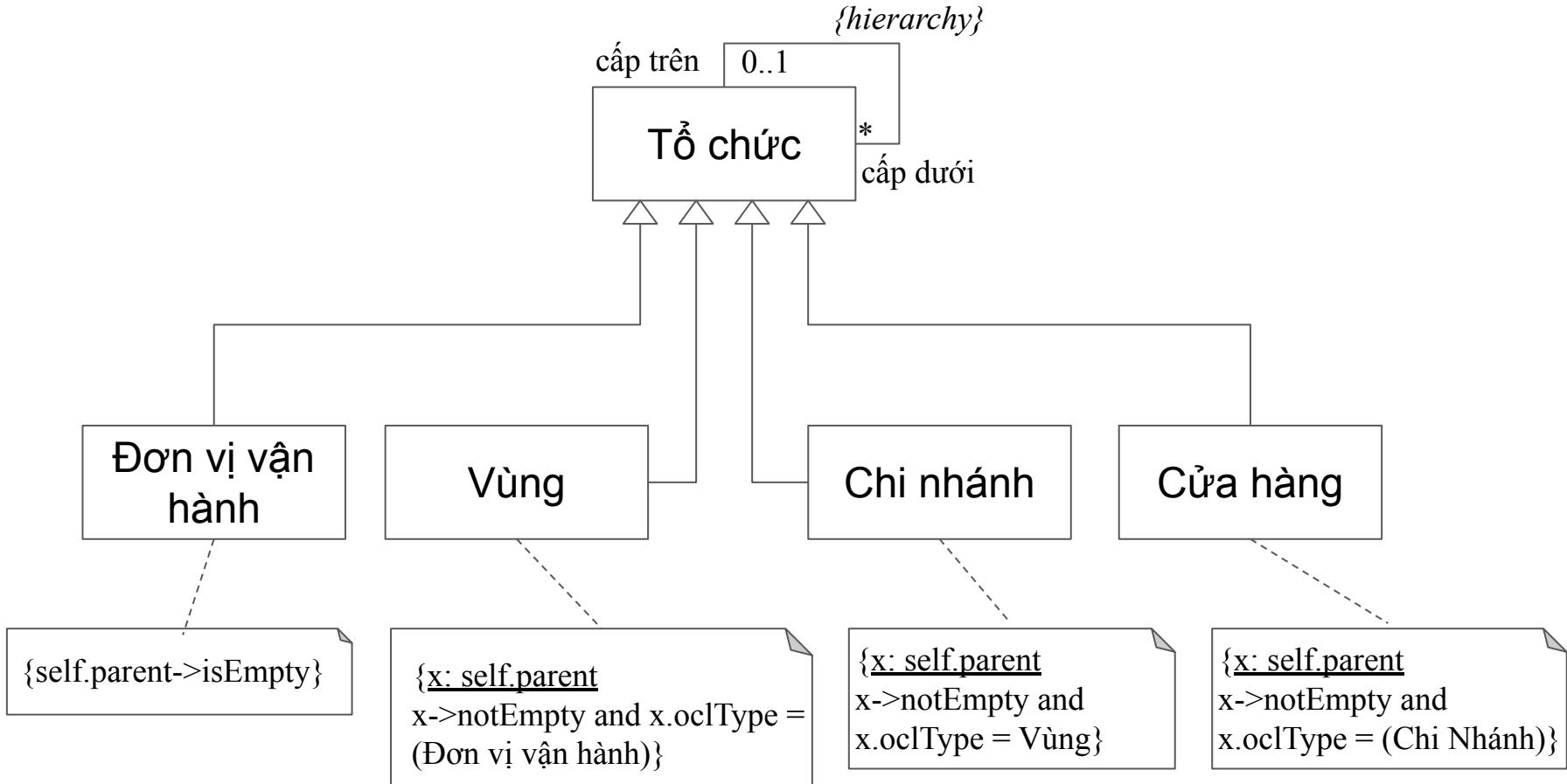


*Trạng thái sau khi giao hàng cho khách*

# Ví dụ 3.23. Sử dụng mẫu tài khoản/giao dịch



# Mẫu phân cấp tổ chức



*Có ích trong trường hợp cần thay đổi cấu trúc phân cấp, ví dụ  
xóa mức vùng*

## Ví dụ 3.24. Các đối tượng phân cấp tổ chức



## Ví dụ 3.25. Mẫu đơn hàng



# Nội dung

- Các khái niệm
- Các kỹ thuật
- Các sơ đồ
- Mô tả ràng buộc với OCL
- Đặc tả lớp với thẻ CRC
- Mẫu phân tích
- Bài tập



# Bài tập

*Xác định các thành phần thông tin, biểu diễn dữ liệu bằng đối tượng, vẽ sơ đồ lớp.*

Công ty cổ phần Thanh Hà  
Số 123, Phường Yên Hòa,

**Mẫu số 02-VT**  
*(Ban hành kèm thông tư số 133/2016/TT-BTC  
Ngày 26/08/2016 của Bộ Tài Chính)*

## PHIẾU XUẤT KHO

Ngày 19 tháng 3 năm 2001  
Số: XK0026

Ng 156  
Có 135



- Tổng số tiền (viết bằng chữ): Hai triệu ba trăm nghìn đồng chẵn
  - Số chứng từ gốc kèm theo: .....

## Người lập phiếu (Ký, họ, tên)

## Người nhận hàng (Ký, họ tên)

Thủ khoa  
(Ký họ tên)

Kế toán trưởng

Giám đốc  
(Ký, họ tên)

# Xác định đối tượng và vẽ sơ đồ lớp

*Xác định các thành phần thông tin, biểu diễn dữ liệu bằng đối tượng, vẽ sơ đồ lớp.*

Công ty cổ phần Thanh Hà  
Số 123, Phường Yên Hòa,

**Mẫu số 02-VT**  
*(Ban hành kèm thông tư số 133/2016/TT-BTC  
Ngày 26/08/2016 của Bộ Tài Chính)*

## PHIẾU XUẤT KHO

Ngày 19 tháng 3 năm 2001  
Số: XK0026

Ng 156  
Có 135



- Tổng số tiền (viết bằng chữ): Hai triệu ba trăm nghìn đồng chẵn
  - Số chứng từ gốc kèm theo: .....

Người lập phiếu  
(Ký, họ tên)

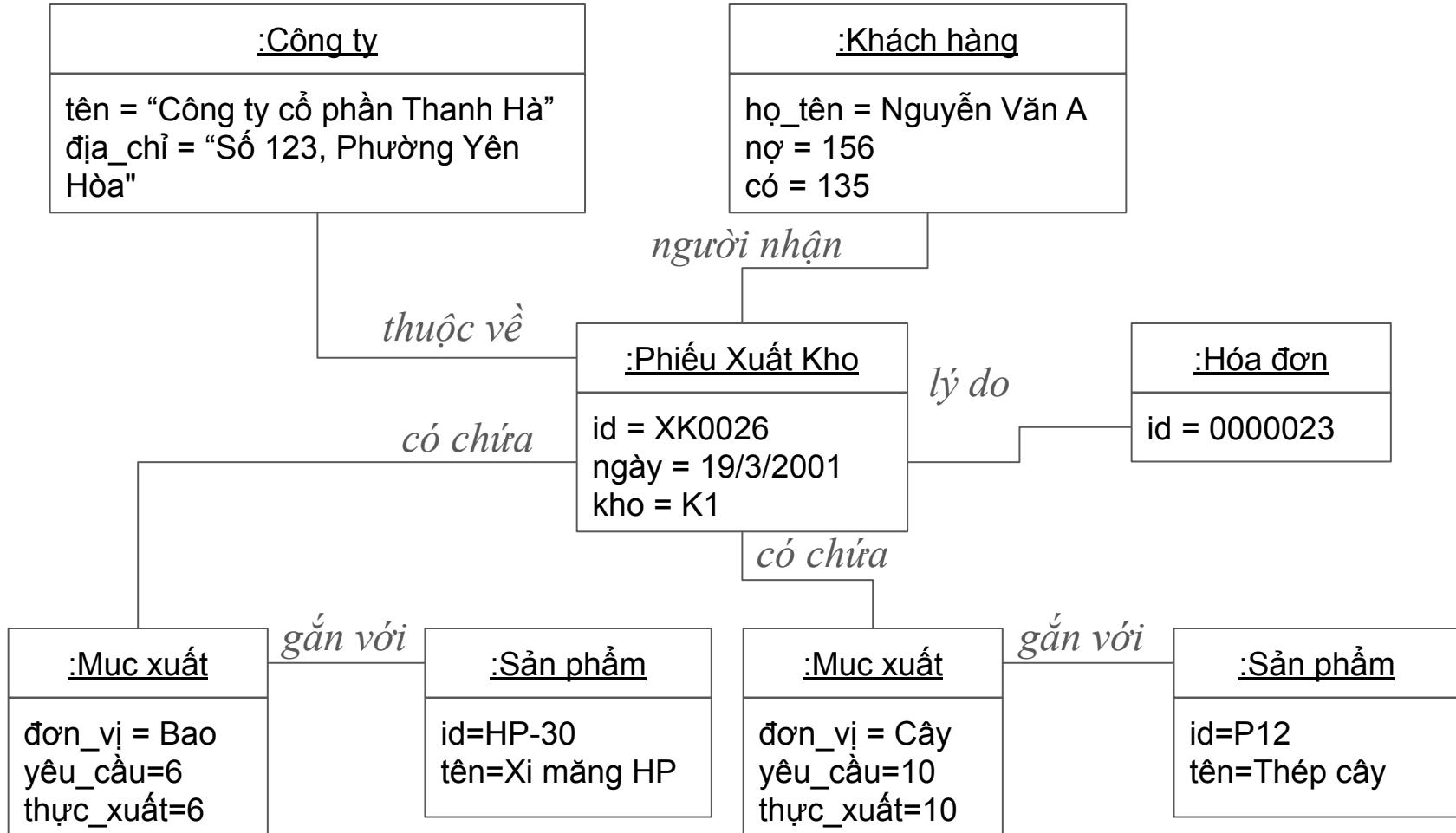
Người nhận hàng  
(Ký, họ tên)

Thủ khoa  
(Ký, ho tên)

Kế toán trưởng

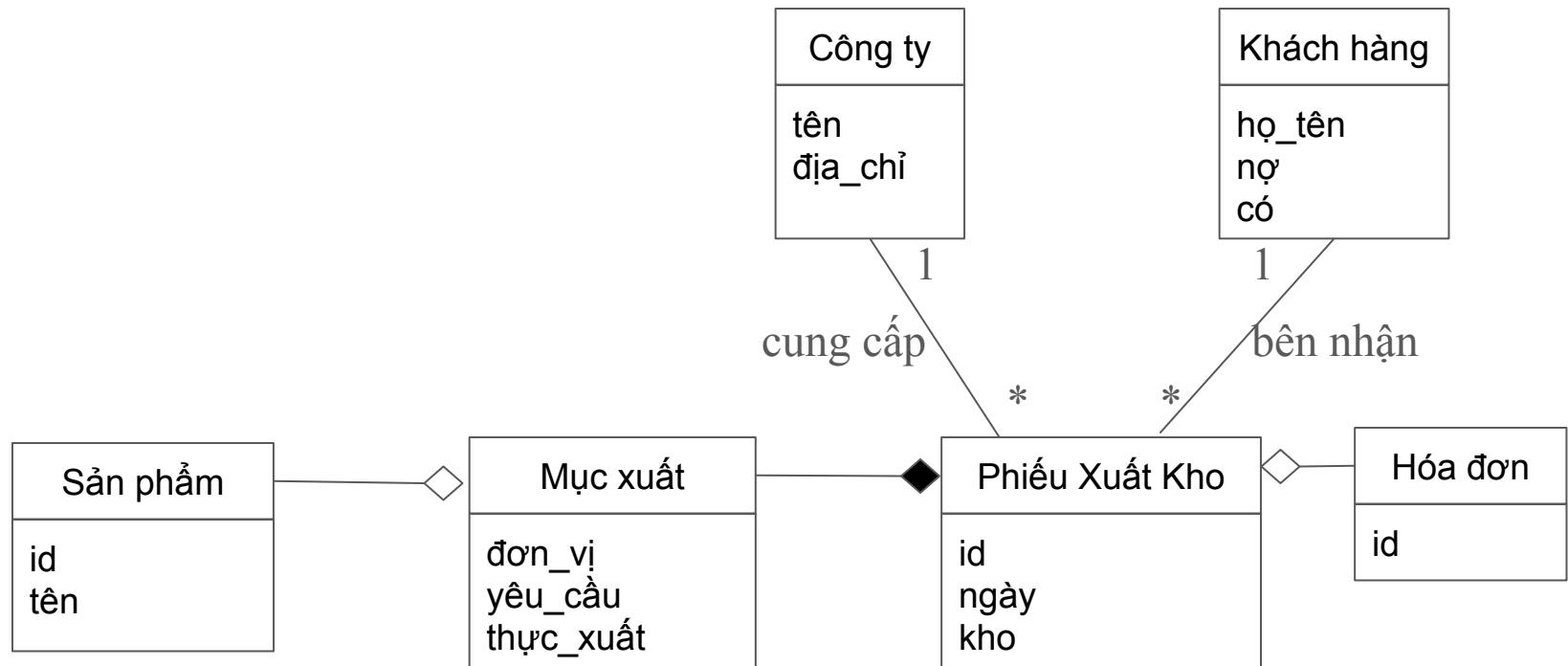
Giám đốc  
(Ký, họ tên)

# Sơ đồ đối tượng



*Lưu ý:* Lời giải để tham khảo, không phải mẫu.

# Sơ đồ lớp



*Lưu ý:* Lời giải để tham khảo, không phải mẫu.



# Phân tích và thiết kế Hệ thống

Giảng viên: Nguyễn Bá Ngọc

## Chương 4

Hà Nội-2021

# Chương 4

## Mô hình hóa hành vi

# Hành vi của hệ thống hướng đối tượng

- Đặc điểm cơ bản: *Hệ thống hướng đối tượng bao gồm nhiều đối tượng tương tác dựa trên cơ chế truyền thông điệp để đáp ứng các tương tác với tác nhân.*
- Các vấn đề mô hình hóa:
  - Mô hình hóa tương tác giữa các tác nhân và hệ thống
  - Mô hình hóa tương tác giữa các đối tượng trong hệ thống nhằm đáp ứng các hoạt động nghiệp vụ
  - Mô hình hóa sự biến đổi trạng thái của đối tượng theo tiến trình nghiệp vụ.

# Nội dung

- **Biểu diễn trạng thái của đối tượng**
  - Sơ đồ máy trạng thái
- Các mô hình tương tác
  - Sơ đồ tuần tự
  - Sơ đồ giao tiếp
- Kỹ thuật phát hiện các mối quan hệ
  - Phân tích ma trận CRUD(E)
- Tổng kết các nội dung phân tích

# Sơ đồ máy trạng thái

- Biểu diễn các trạng thái và sự thay đổi trạng thái của đối tượng
- Thường được sử dụng để biểu diễn trạng thái của các đối tượng trọng tâm, xuất hiện trong nhiều ca sử dụng.
  - Trạng thái của đối tượng được quan tâm trong các hoạt động nghiệp vụ,
  - Có liên quan đến các công việc nghiệp vụ, ...

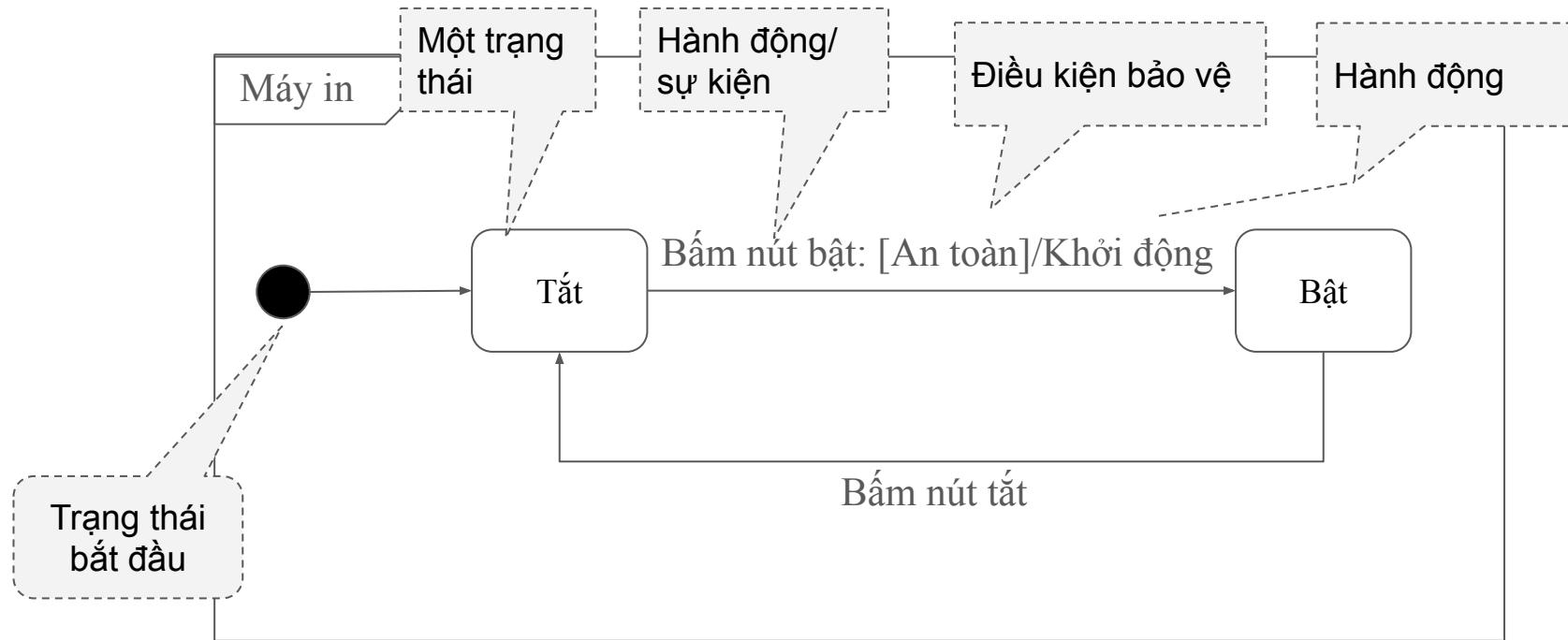
*Đạo này thế nào? Có gì mới không? ...*

# Các thành phần tiêu biểu

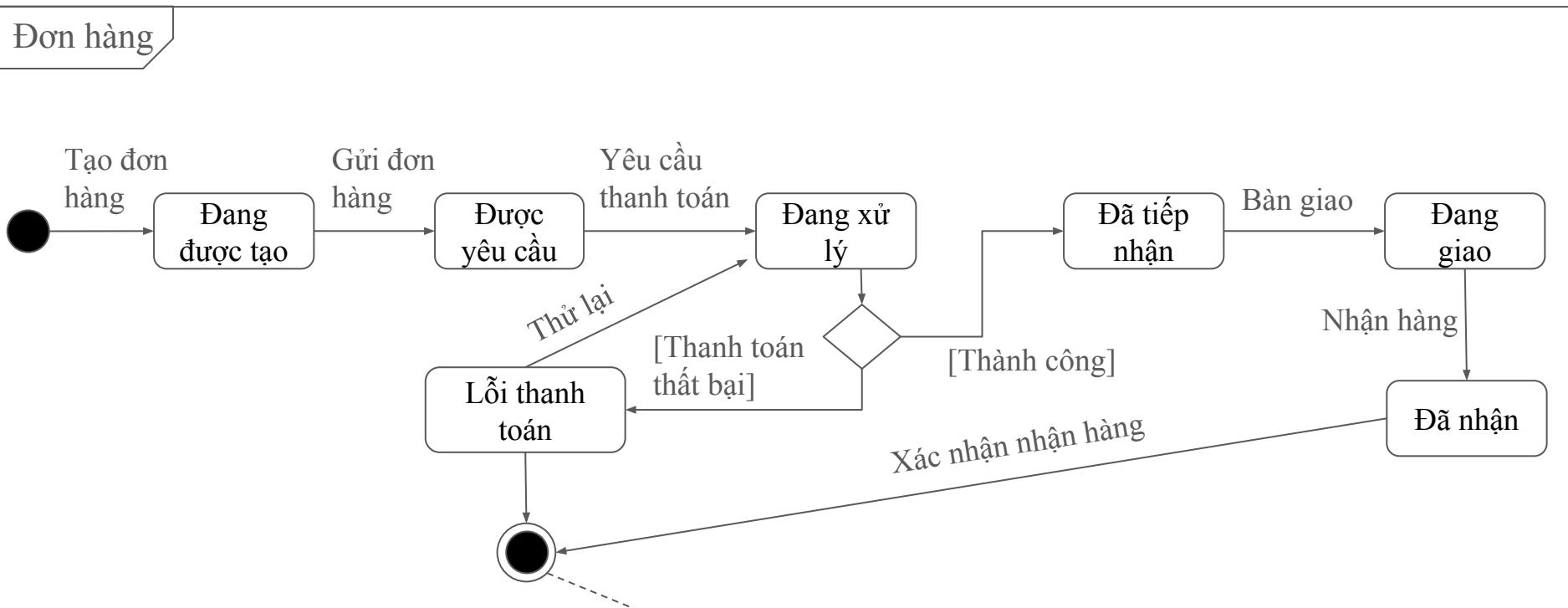
- Trạng thái: 1 giai đoạn trong 1 tiến trình, kết quả xử lý hiện tại
  - Có liên quan tới các công việc nghiệp vụ.
  - Bộ giá trị của các thuộc tính của đối tượng thỏa mãn các điều kiện nhất định.
- Sự thay đổi trạng thái của đối tượng:
  - Bước chuyển: Từ 1 trạng thái nguồn sang 1 trạng thái đích, theo chiều mũi tên.
  - Mô tả chi tiết:
    - Nguyên nhân: Sự kiện, hành động dẫn đến sự thay đổi trạng thái.
    - Điều kiện bảo vệ: Chỉ chuyển sang trạng thái đích nếu điều kiện bảo vệ được đáp ứng.
    - Biểu thức hành vi: Cái được thực hiện và hoàn thành trước khi chuyển sang trạng thái đích



# Ví dụ 4.1a. Các thành phần cơ bản



# Ví dụ 4.1b. Trạng thái đơn hàng



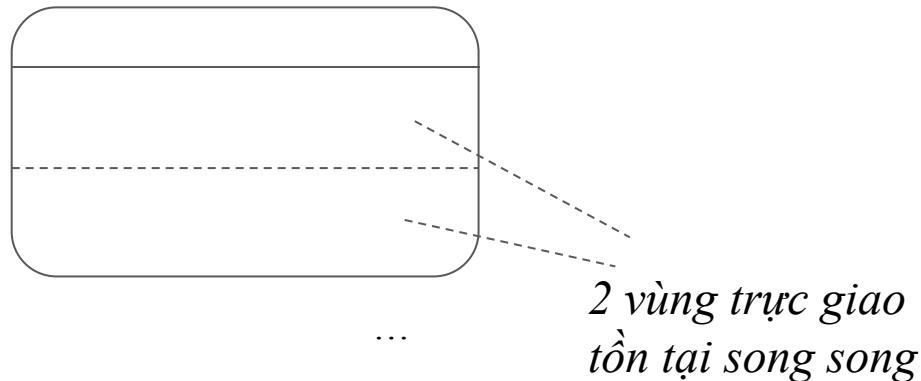
Trạng thái kết thúc

# Phân loại trạng thái

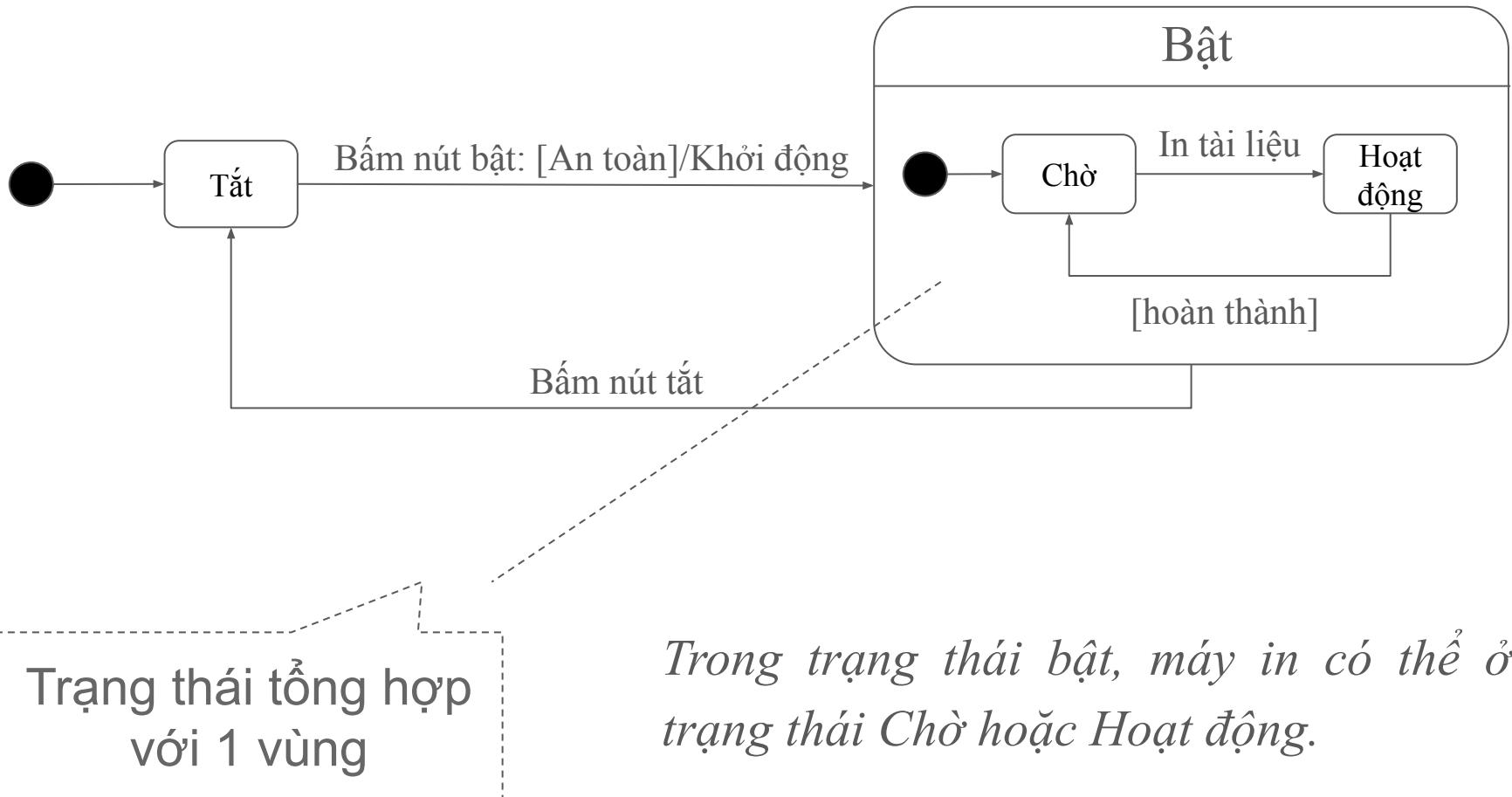
- Trạng thái đơn: Không chứa các thành phần bên trong
- Trạng thái tổng hợp: Bao gồm nhiều trạng thái con
  - Có thể bao gồm máy trạng thái hoàn chỉnh
    - Có bắt đầu và kết thúc riêng.
    - Tạo thành các máy trạng thái lồng nhau
  - Có thể bao gồm đúng 1 vùng hoặc nhiều vùng song song
- Các luồng trạng thái song song được gọi là các trạng thái tổng hợp trực giao (orthogonal)
  - Được sở hữu bởi cùng 1 trạng thái hoặc
  - cùng ở mức đỉnh của 1 máy trạng thái

# Phân Vùng

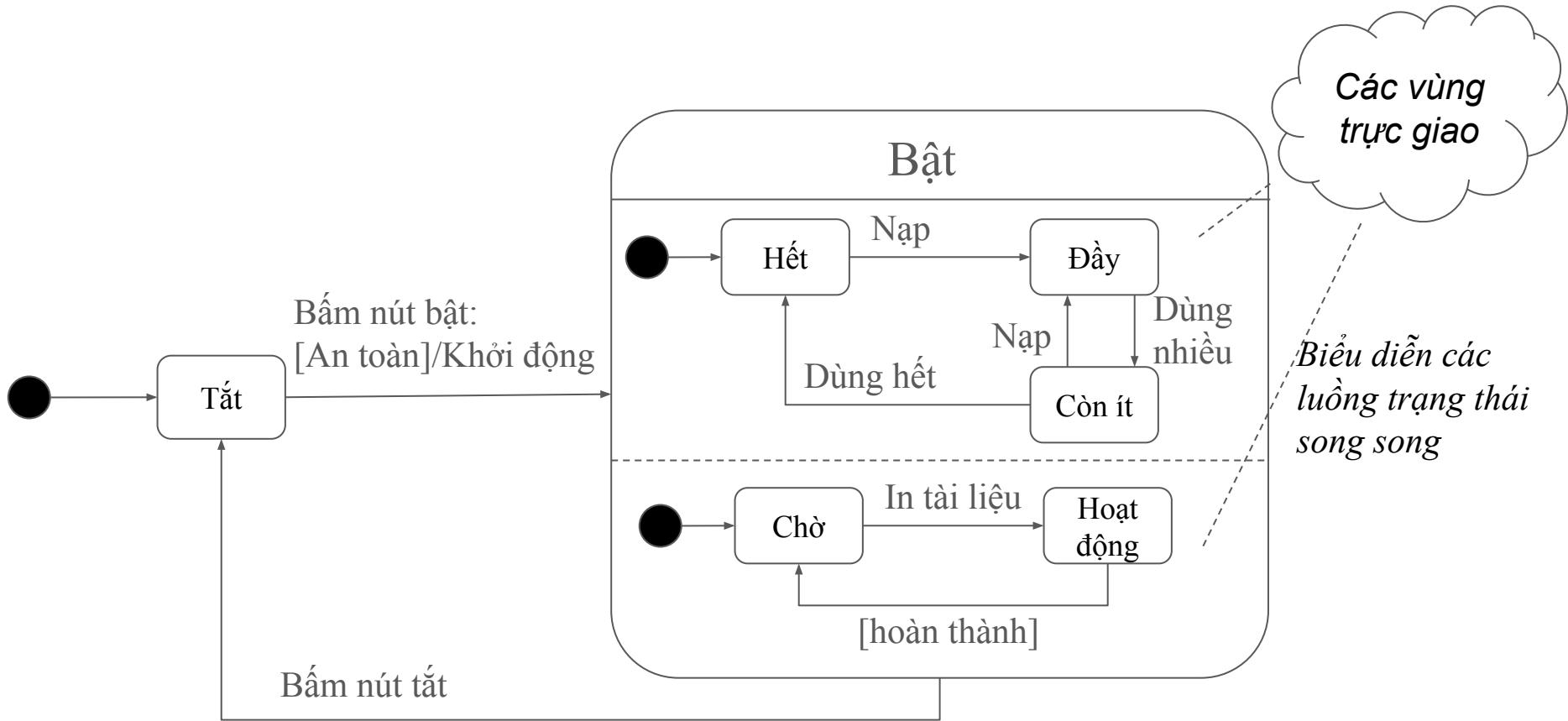
- Biểu diễn trạng thái con
  - Có thể tồn tại đồng thời trong các *vùng trực giao*
    - Được phân chia bằng các đường đứt nét
- Được kích hoạt khi đối tượng chuyển tới trạng thái chứa nó



## Ví dụ 4.2. Trạng thái tổng hợp đơn

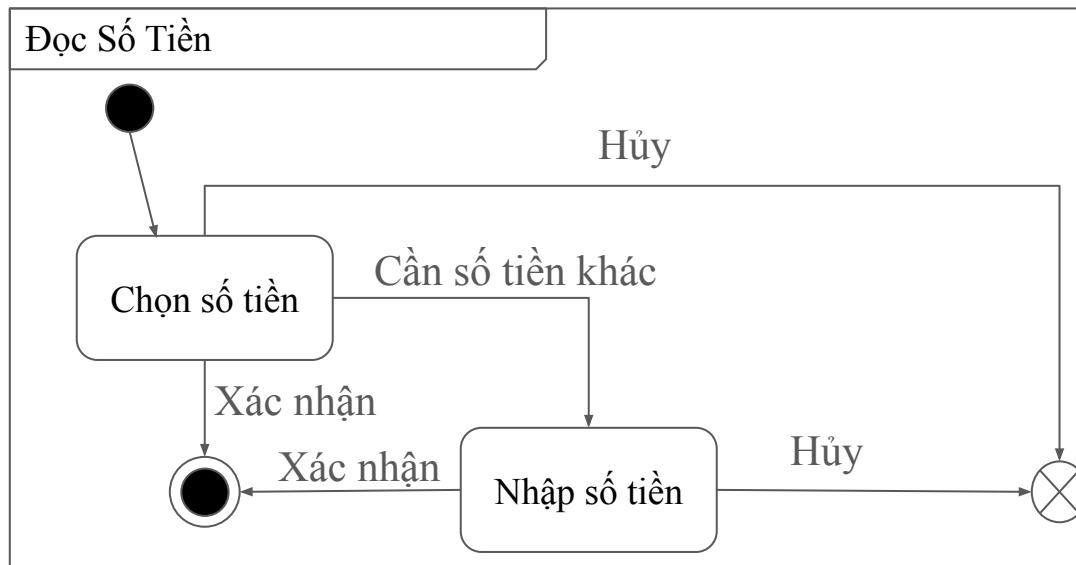
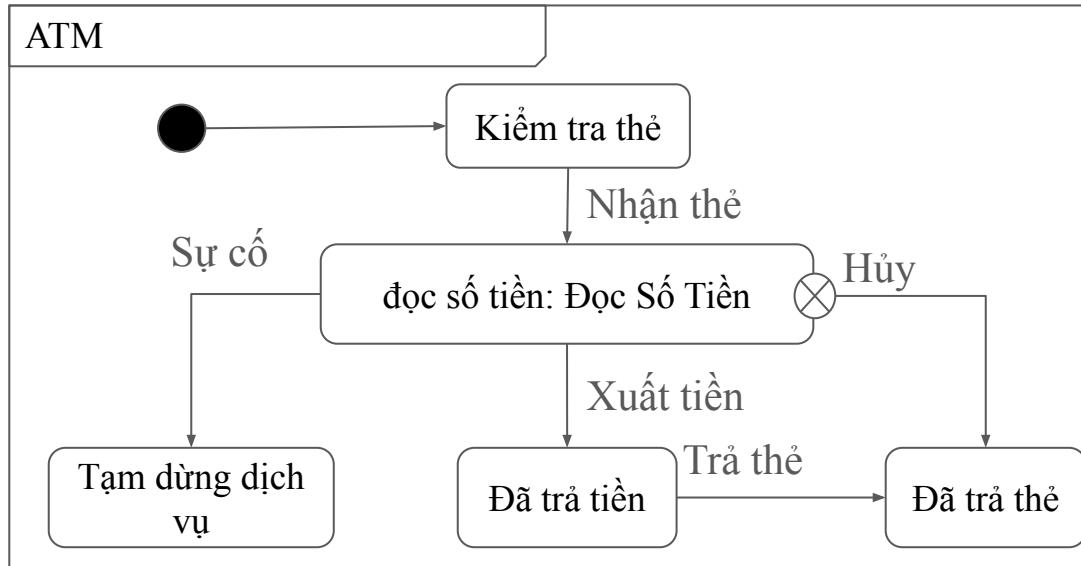


# Ví dụ 4.3. Trạng thái tổng hợp trực giao



- Mỗi luồng (diễn ra đồng thời) nằm trong 1 vùng trực giao
- Mô tả đối tượng có nhiều luồng trạng thái song song
  - Cho các thành phần (mô-đun) khác nhau của đối tượng

# Ví dụ 4.4. Máy trạng thái lồng nhau



# Các bước vẽ sơ đồ máy trạng thái

1. Nghiên cứu sơ đồ lớp và mô hình chức năng, xác định các đối tượng lĩnh vực có nhiều trạng thái, cần được biểu diễn bằng sơ đồ máy trạng thái.
  - a. Xuất hiện trong nhiều ca sử dụng.
  - b. Liên quan đến các công việc nghiệp vụ.
2. Lập danh sách các trạng thái của đối tượng.
3. Xác định các sự kiện làm thay đổi trạng thái của đối tượng.
4. Thiết lập cấu trúc trạng thái, kết hợp nhiều trạng thái nhỏ thành trạng thái tổng hợp nếu có thể.
5. Biểu diễn các bước chuyển trạng thái.
6. Bổ xung các chi tiết cho bước chuyển: Sự kiện, điều kiện bảo vệ, hành động.
7. Kiểm tra kết quả thu được.

# Nội dung

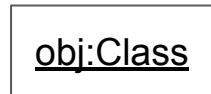
- Biểu diễn trạng thái của đối tượng
  - Sơ đồ máy trạng thái
- **Các mô hình tương tác**
  - Sơ đồ tuần tự
  - Sơ đồ giao tiếp
- Kỹ thuật phát hiện các mối quan hệ
  - Phân tích ma trận CRUD(E)
- Tổng kết các nội dung phân tích



# Sơ đồ tương tác

- Biểu diễn tương tác giữa các đối tượng theo cơ chế truyền thông điệp để hoàn thành các hoạt động của người dùng.
- Sơ đồ tuần tự: Dễ theo dõi thứ tự tuần tự của các thông điệp, là loại sơ đồ tương tác được sử dụng phổ biến nhất
- Sơ đồ giao tiếp: Tự do bố trí các đối tượng, dễ theo dõi quan hệ giữa các đối tượng.

# Các thành phần thường dùng



Đối tượng



Xử lý thông  
điệp



Các thông điệp  
song song



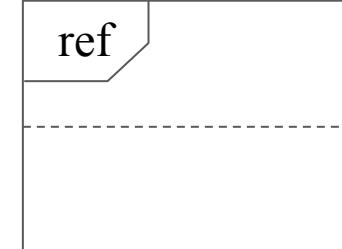
Giới hạn thời  
gian xử lý



Bất biến



Tương tác mở rộng



Các vùng trực giao



Hủy đối tượng



Thông điệp

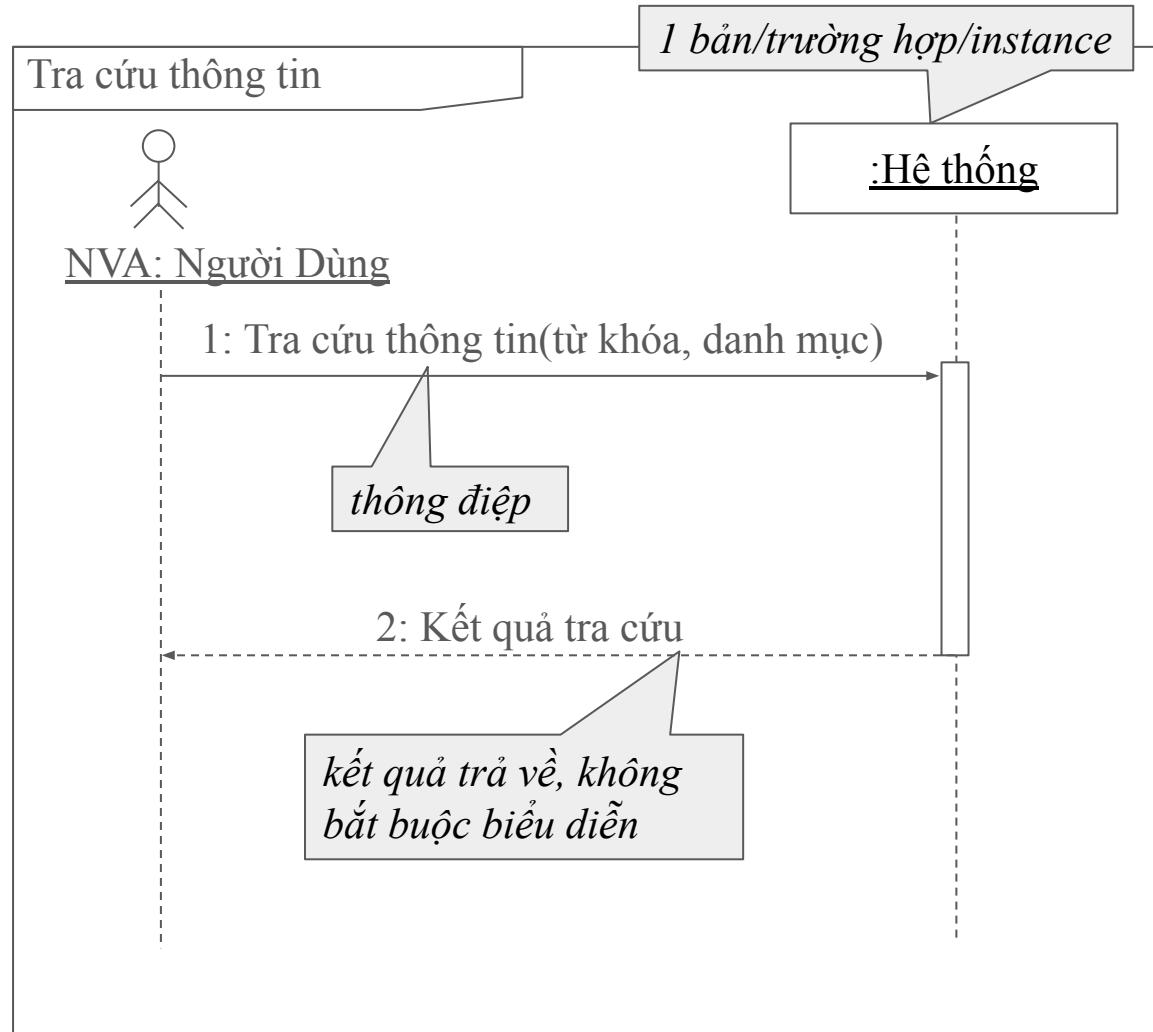


Thông điệp biến mất

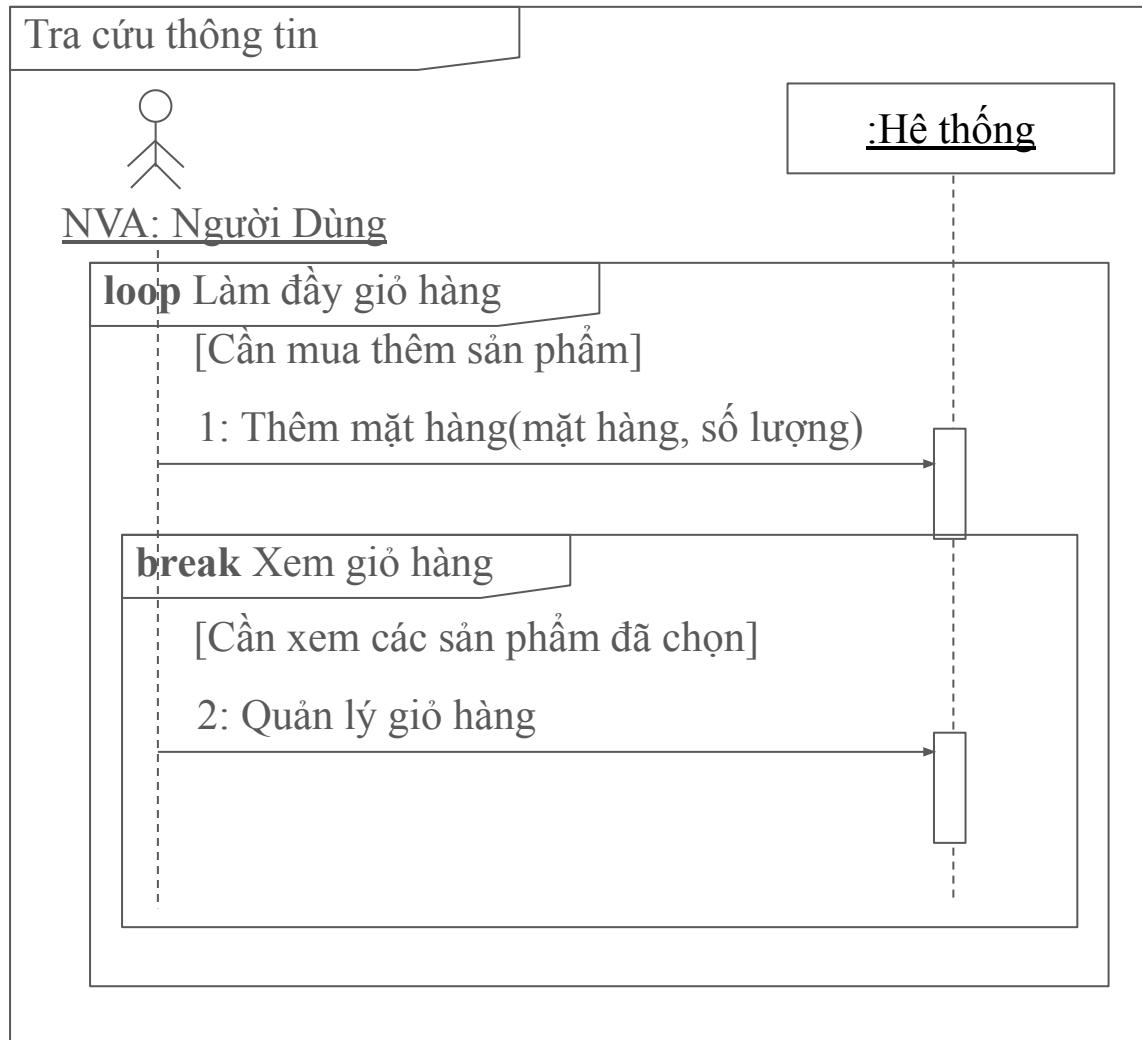


Thông điệp tìm thấy

# Ví dụ 4.5a. Sơ đồ tuần tự - Thông điệp cơ bản

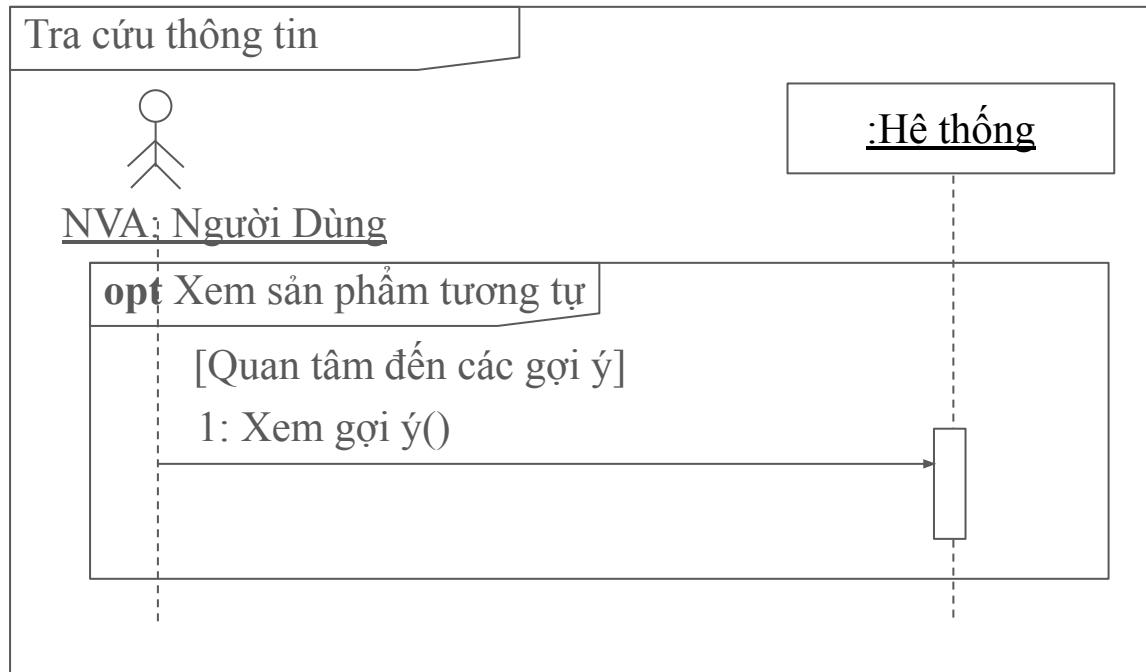


# Ví dụ 4.5b. Sơ đồ tuần tự- Vòng lặp



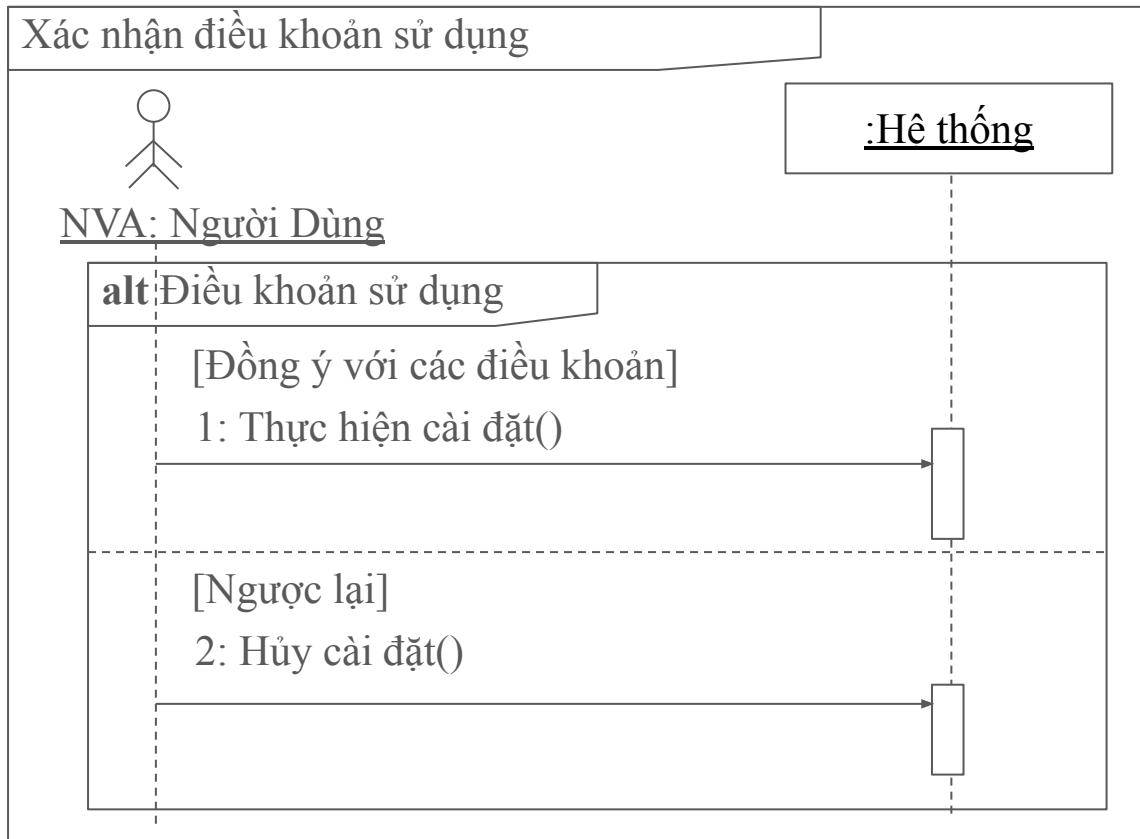
*Dừng lặp sau khi thực hiện break*

# Ví dụ 4.5c. Sơ đồ tuần tự - Vùng điều kiện



*Chỉ thực hiện nếu điều kiện bảo vệ đúng.*

# Ví dụ 4.5d. Sơ đồ tuần tự - Rẽ nhánh



*Vùng tương ứng được thực hiện nếu điều kiện đúng.*

# Mô tả thông điệp

[Biểu thức lô-gic] Tên thông điệp (Danh sách tham số)

- Điều kiện bảo vệ được mô tả trong cặp dấu []. Nếu biểu thức đúng thì thông điệp được gửi, nếu ngược lại thì thông điệp không được gửi.
- Tên thông điệp mô tả dịch vụ được yêu cầu, thường được bỏ qua trên thông điệp trả về.
- Danh sách tham số (nếu có) biểu diễn dữ liệu được gửi tới đối tượng nhận thông điệp.

# Sơ đồ tuần tự mức hệ thống

## *System Sequence Diagram (SSD)*

- Cấu trúc đơn giản
  - Chỉ gồm các tác nhân và hệ thống
  - Mô phỏng kịch bản sử dụng hệ thống để thực hiện các công việc nghiệp vụ hữu ích, trong các ca sử dụng.
- Các thông điệp tương ứng với các hoạt động nghiệp vụ ở mức khái quát cao, có thể là chức năng của hệ thống.

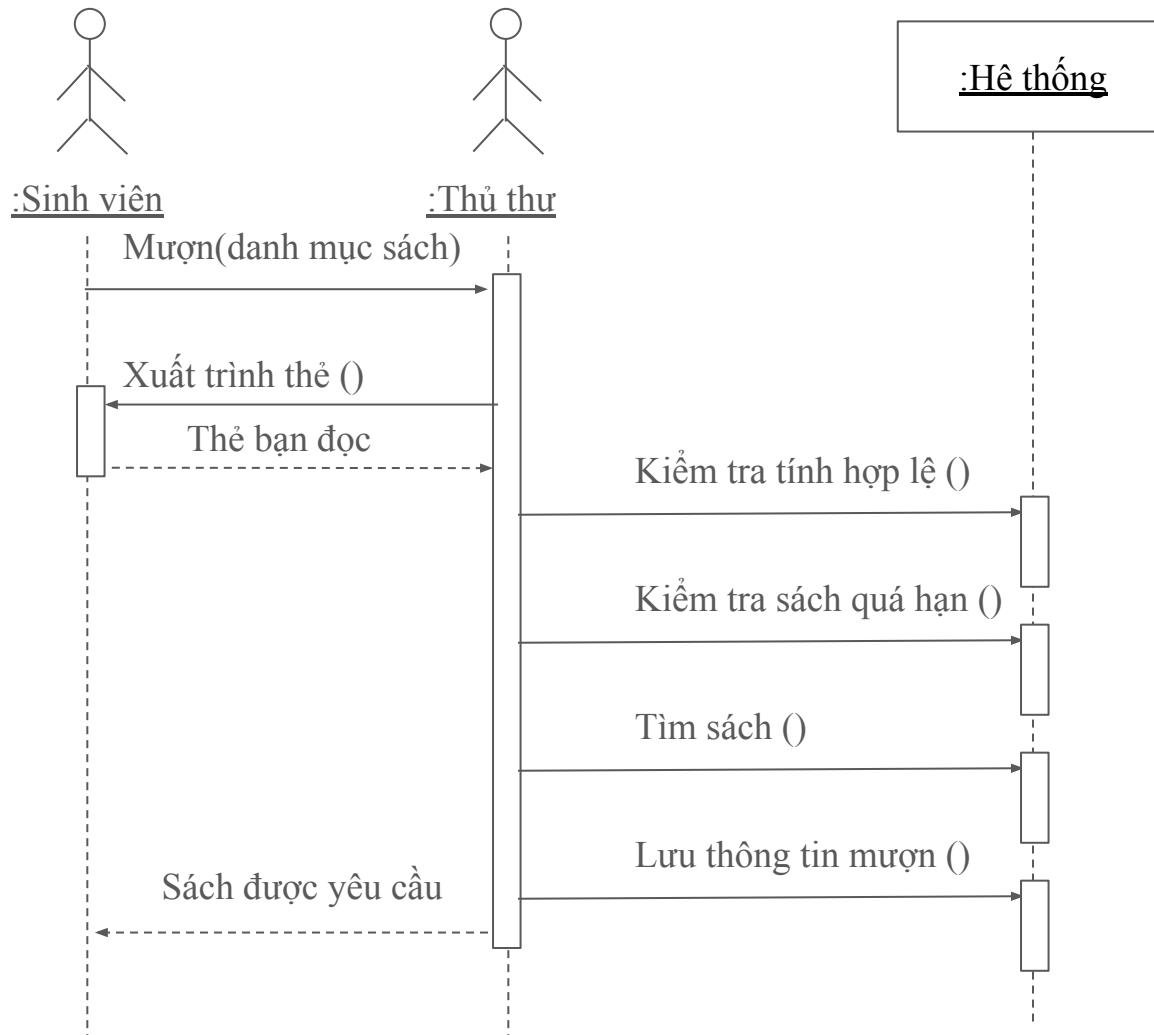
# Sơ đồ tuần tự mức nghiệp vụ

- Chi tiết hơn SSD, với mỗi thông điệp trong SSD có thể được chi tiết hóa thành nhiều thông điệp cụ thể hơn.
- Biểu diễn các thông điệp được trao đổi giữa các đối tượng trong tiến trình thực hiện kịch bản sử dụng hệ thống
  - Sử dụng các đối tượng lĩnh vực, các thứ liên quan đến các hoạt động nghiệp vụ.
- Một ca sử dụng có thể có nhiều kịch bản
  - Mỗi luồng sự kiện đều có thể có luồng tương đương
  - Một kịch bản tương ứng với một trình tự xác định trong phạm vi ca sử dụng
- Giúp hiểu tốt hơn các ca sử dụng phức tạp.

*Các chi tiết có thể tiếp tục được thêm vào theo tiến trình thiết kế.*

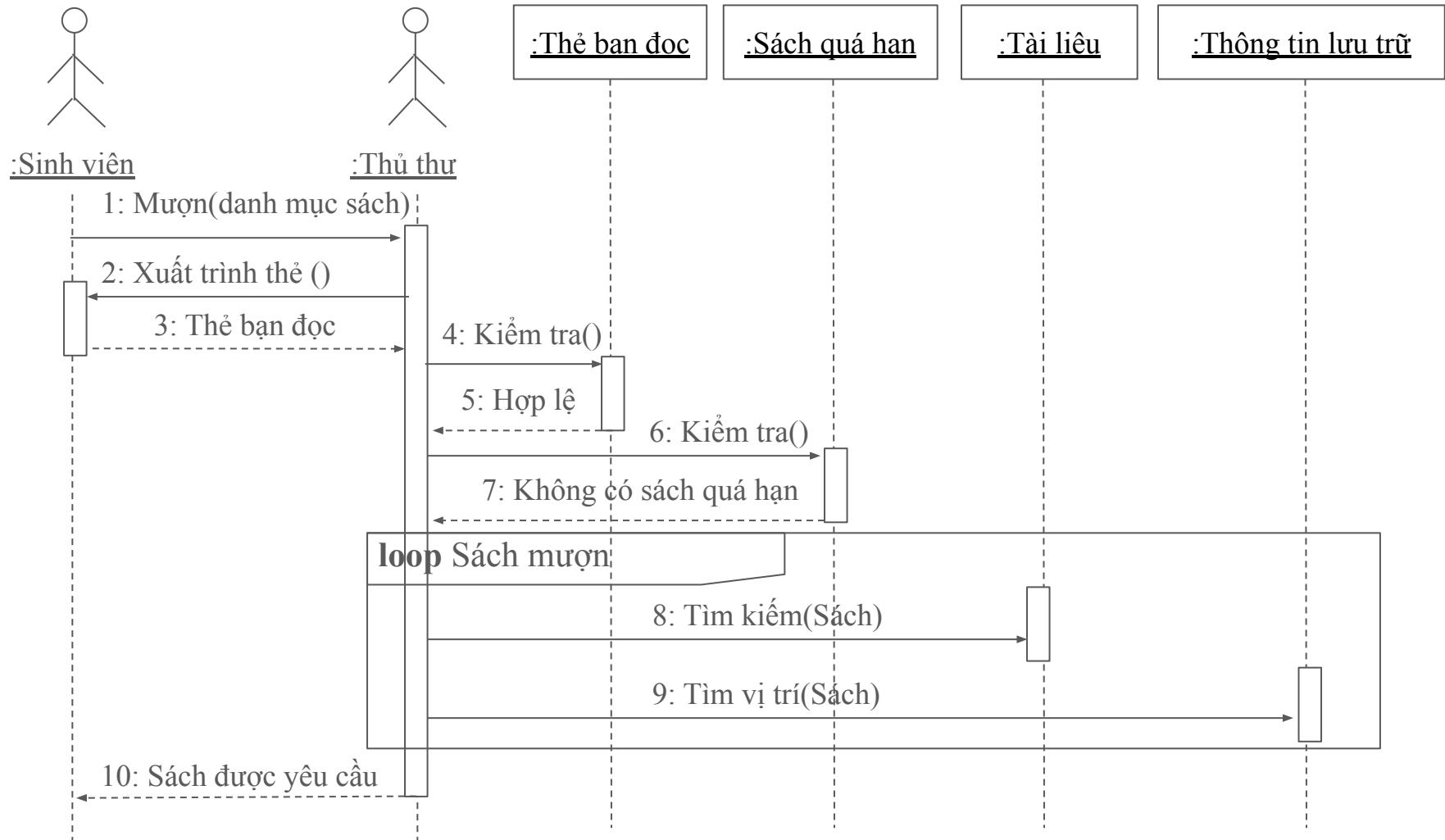
# Ví dụ 4.6. Ca sử dụng mượn sách

*Sơ đồ tuần tự mức hệ thống*



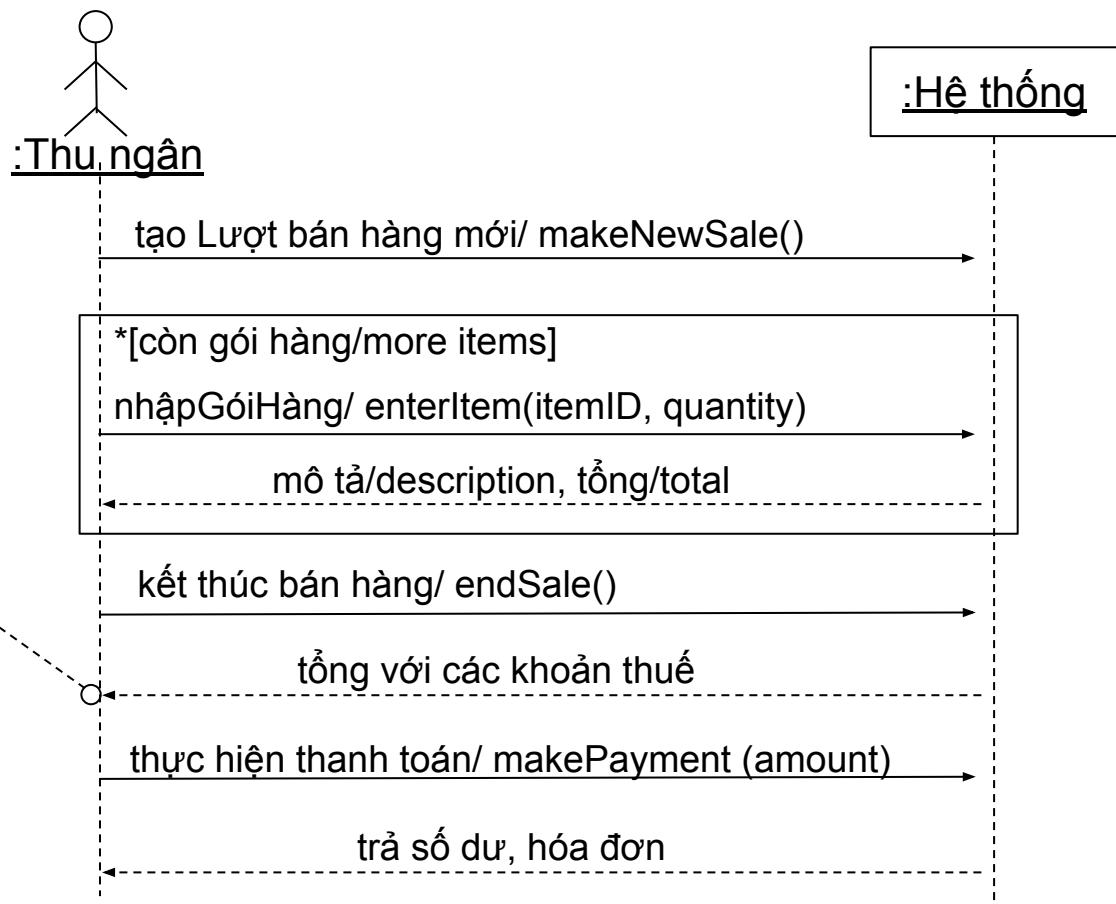
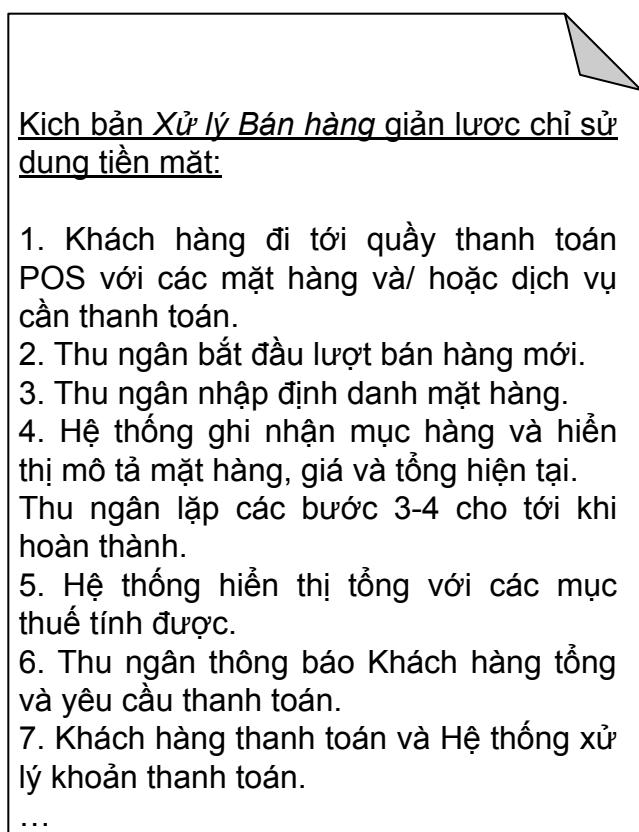
# Ví dụ 4.6. Ca sử dụng mượn sách<sub>(2)</sub>

Sơ đồ tuần tự mức nghiệp vụ



\* Các thành phần giao diện sẽ tiếp tục được thêm vào ở pha Thiết kế

# Luồng sự kiện và SSD



# Các bước vẽ sơ đồ tuần tự

1. Thiết lập ngũ cảnh
2. Xác định các tác nhân và các đối tượng
3. Thiết lập các trục thời gian
4. Biểu diễn thông điệp
  - a. Xác định đối tượng gửi và đối tượng nhận
  - b. Các tham số
  - c. Có thể bỏ qua thông điệp trả về nếu không có trao đổi dữ liệu
5. Biểu diễn xử lý thông điệp trên các trục thời gian
6. Kiểm tra mô hình

# Nội dung

- Biểu diễn trạng thái của đối tượng
  - Sơ đồ máy trạng thái

- **Các mô hình tương tác**

- Sơ đồ tuần tự
- Sơ đồ giao tiếp
- Kỹ thuật phát hiện các mối quan hệ
  - Phân tích ma trận CRUD(E)
- Tổng kết các nội dung phân tích

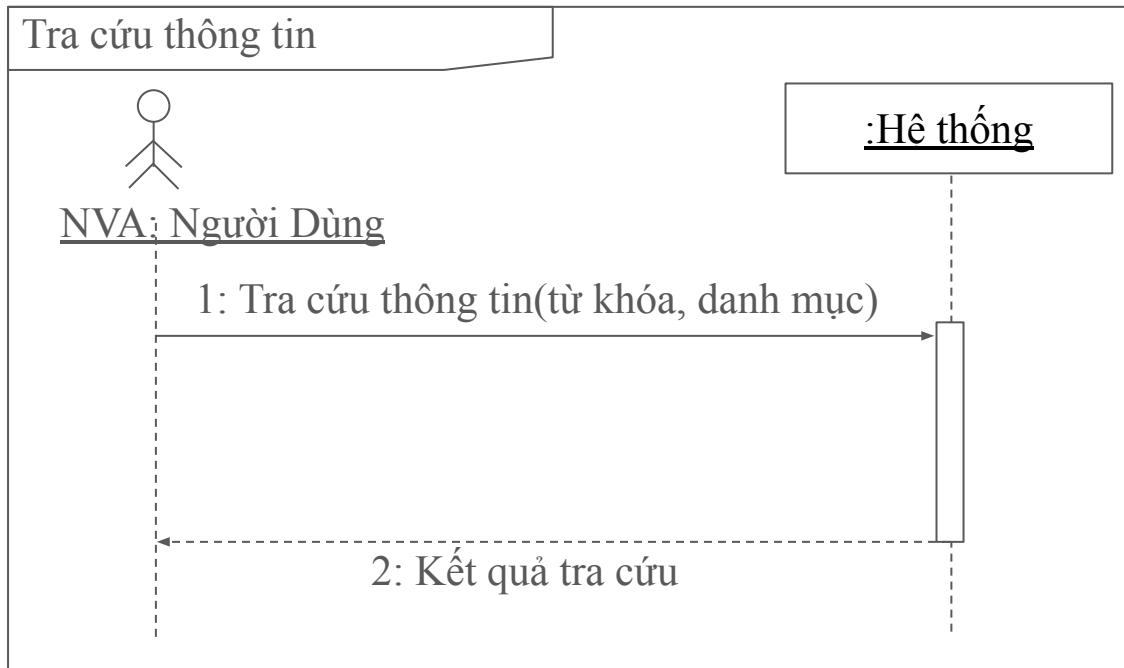
# Sơ đồ giao tiếp

- Tập trung hơn vào kiến trúc và quan hệ giữa các đối tượng
  - Cho phép tự do bố trí các đối tượng
  - Thường được sử dụng để tổng hợp kiến thức/brainstorming
- Thứ tự gửi thông điệp được xác định bằng biểu thức thứ tự/mã thông điệp
  - Thông điệp được đánh số theo 1 sơ đồ thống nhất
  - Tương tự đánh số mục lục sách.
- Có thể biểu diễn các nội dung tương đương với các sơ đồ tuần tự đơn giản, không sử dụng các thành phần cấu trúc phức tạp:
  - Tương tác mở rộng
  - Các vùng kết hợp

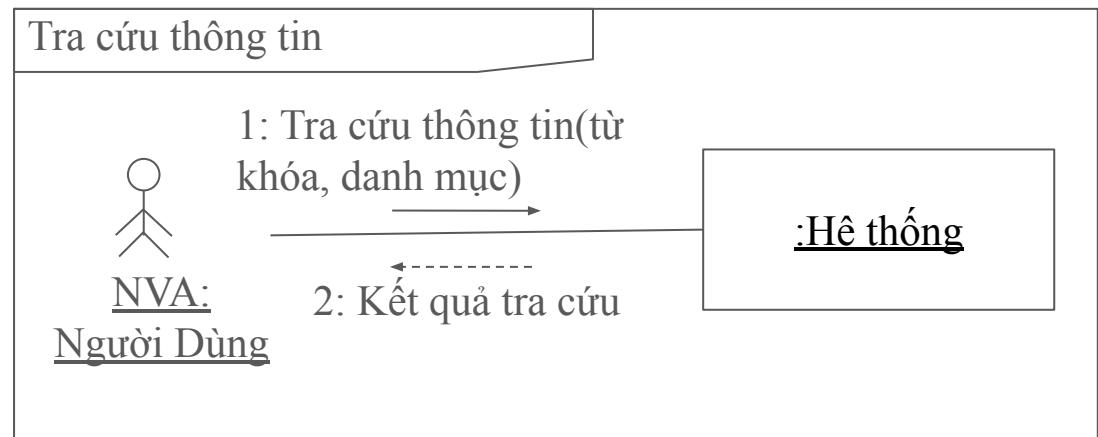
# Biểu thức thứ tự

- Danh sách các *thành phần thứ tự* được ngăn cách bởi dấu . và kết thúc bởi dấu :
  - Định dạng: Thành phần thứ tự '.' '...' '.'
- Mỗi thành phần biểu diễn 1 mức lồng nhau trong tương tác
  - Định dạng: [số nguyên|tên][điều khiển]
  - Số nguyên biểu diễn thứ tự tuần tự của thông điệp, thông điệp tiếp theo có giá trị số tăng thêm 1 đơn vị:
    - Mỗi thành phần tương ứng với 1 mức trong chuỗi thông điệp lồng nhau.
    - Các thông điệp song song có cùng số thứ tự và được phân biệt bằng tên
    - 3.2.3 là thông điệp được gửi tiếp theo sau 3.2.2 khi 3.2 được kích hoạt.
    - 3.1a và 3.1b diễn ra đồng thời khi 3 được kích hoạt
  - Mô tả điều khiển biểu diễn điều kiện hoặc vòng lặp.
    - Biểu diễn 0 hoặc nhiều thông điệp phụ thuộc vào biểu thức điều kiện.
    - '\*' [biểu thức lặp] thân vòng lặp, ví dụ: \*[i := 1..n] ...
    - [Điều kiện bảo vệ] nhánh, ví dụ: [x > y] ...

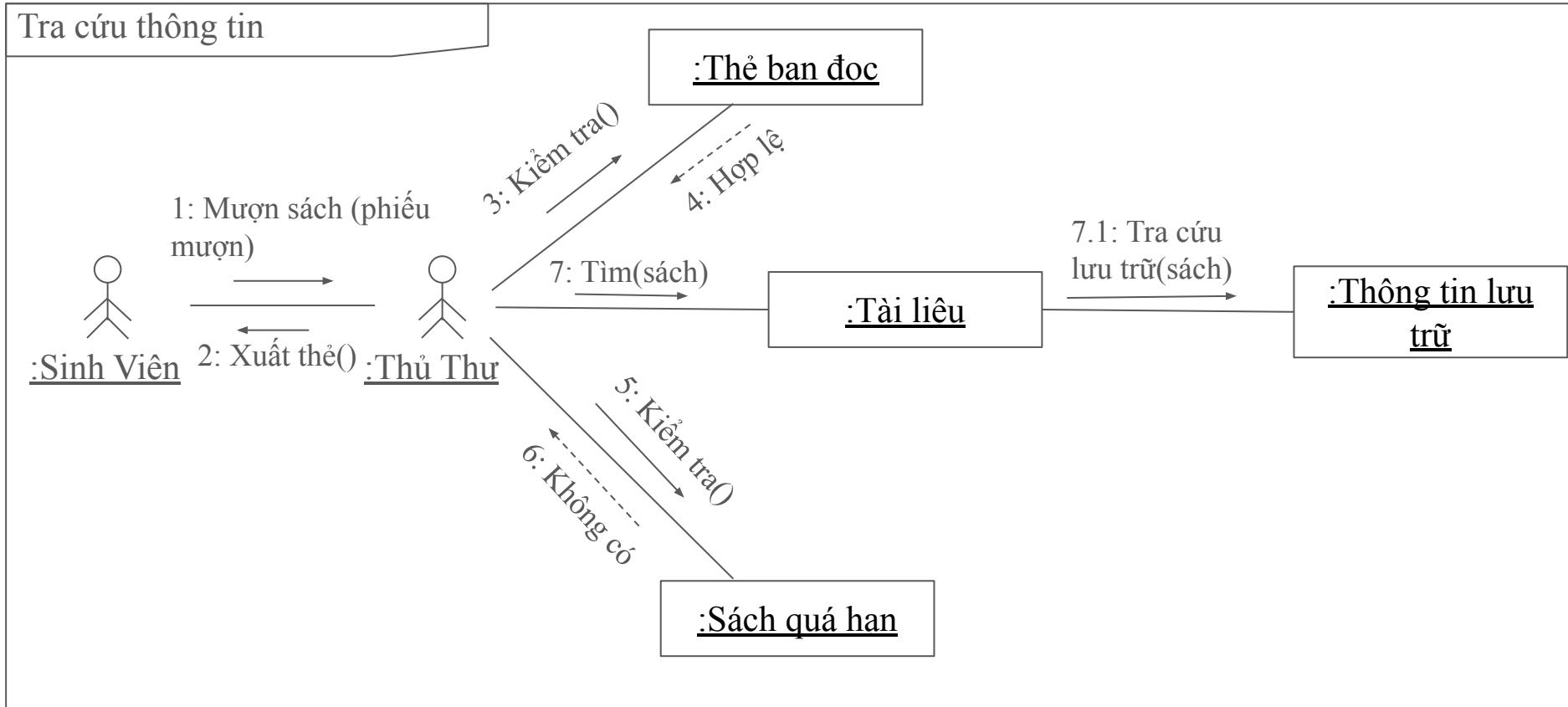
# Ví dụ 4.7. Sơ đồ giao tiếp mức hệ thống



*Biểu diễn cùng 1 nội dung  
bằng Sơ đồ tuần tự và sơ đồ  
giao tiếp*



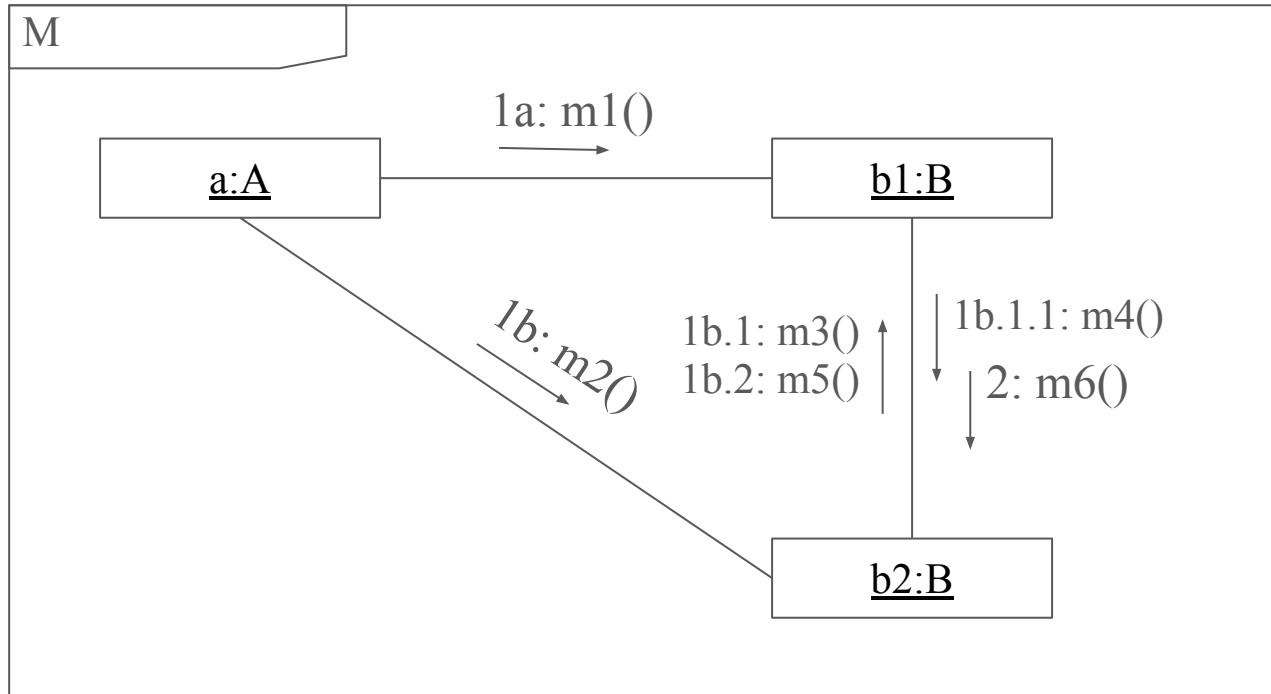
## Ví dụ 4.8. Sơ đồ giao tiếp mức nghiệp vụ



So sánh với 4.6

## Ví dụ 4.9. Biểu thức thứ tự

Tương tự đánh số đầu mục nội dung sách.



Các thông điệp  $m_1$  và  $m_2$  được gửi đồng thời từ đối tượng  $a$  tới các đối tượng  $b_1$  và  $b_2$ .

Chuỗi thông điệp lồng nhau:  $1b$ ,  $1b.1$ ,  $1b.1.1$

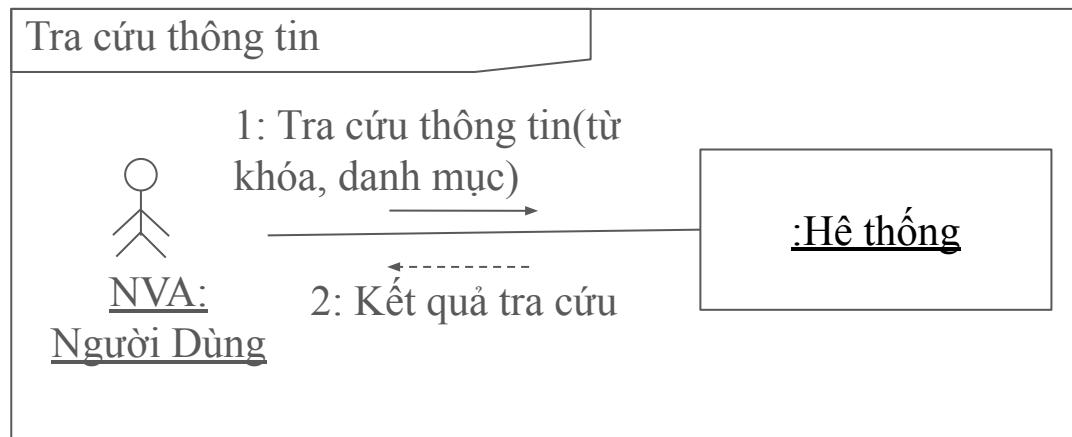
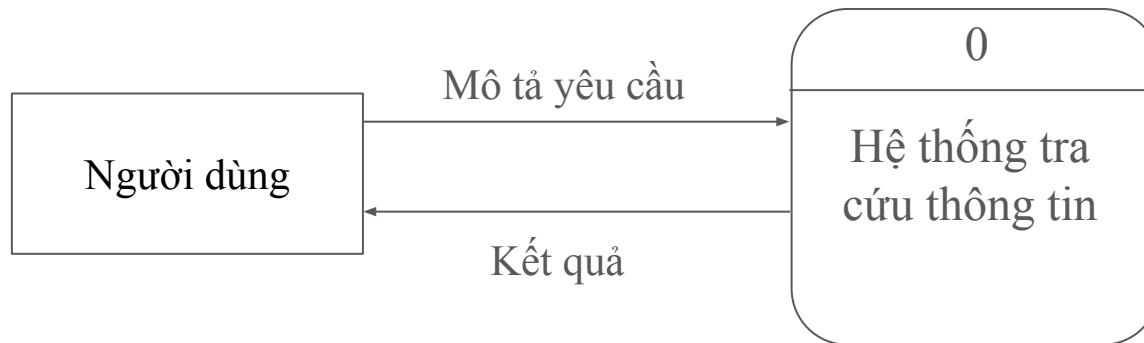
Thông điệp tuần tự:  $2:m_6$  được gửi sau khi hoàn thành  $1a:m_1$  và  $1b:m_2$ ;  $1b.2:m_5$  được gửi sau khi hoàn thành  $1b.1:m_3$

# Các bước vẽ sơ đồ giao tiếp

1. Thiết lập ngũ cảnh
2. Xác định các đối tượng, tác nhân
3. Xác định các mối quan hệ giữa các thành phần
4. Bố trí các thành phần trên sơ đồ
5. Biểu diễn các thông điệp
6. Kiểm tra mô hình

# Sơ đồ giao tiếp vs. DFD mức ngũ cảnh.

## Sơ đồ luồng dữ liệu / Data Flow Diagram - DFD



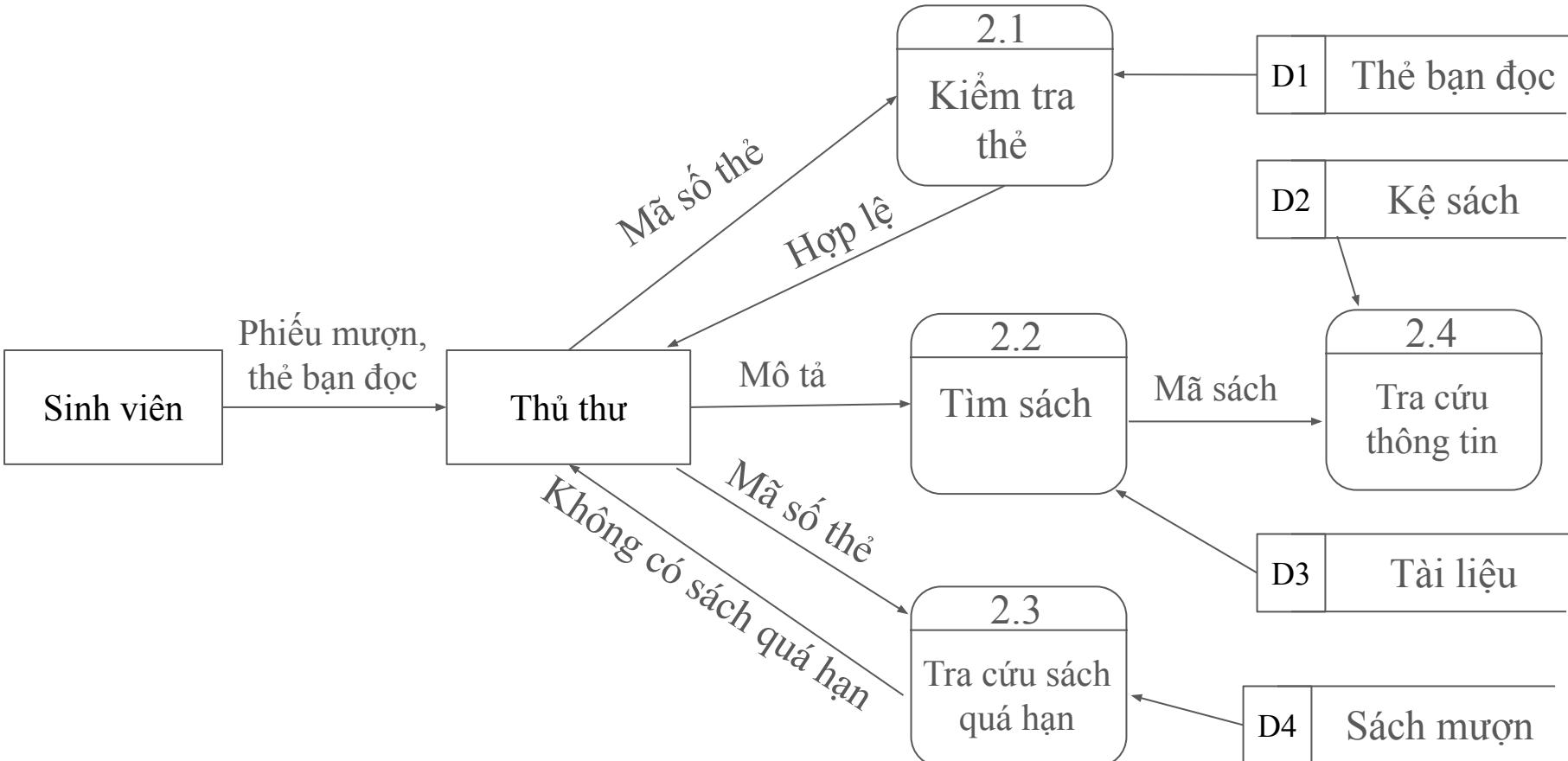
# Sơ đồ giao tiếp vs. DFD

*Các ngôn ngữ mô hình hóa khác nhau nhưng vẫn có những điểm tương đồng nhất định.*

- Tự do xếp đặt các thành phần
- Tác nhân vs. Thực thể ngoại
- Đối tượng vs. Tiến trình
- Tham số vs. Dữ liệu
- Các thông điệp trong sơ đồ giao tiếp/tương tác được chi tiết hóa dần theo tiến trình phát triển tương tự như việc chi tiết hóa dần các xử lý trong DFD qua các mức.
- Đối tượng xử lý thông điệp vs. Tiến trình + lưu trữ thực thể.

*Ca sử dụng không phải là chức năng. Kịch bản thực hiện ca sử dụng bao gồm nhiều thông điệp, xử lý thông điệp có thể tương đương với chức năng và được chi tiết hóa theo tiến trình phát triển!*

# Ví dụ 4.10. Sơ đồ luồng dữ liệu



\* So sánh với sơ đồ giao tiếp cho cùng kịch bản

# Nội dung

- Biểu diễn trạng thái của đối tượng
  - Sơ đồ máy trạng thái
- Các mô hình tương tác
  - Sơ đồ tuần tự
  - Sơ đồ giao tiếp
- **Kỹ thuật phát hiện các mối quan hệ**
  - Phân tích ma trận CRUD(E)
- Tổng kết các nội dung phân tích

# Kỹ thuật phân tích ma trận

- Hỗ trợ xác định các mối quan hệ
- Gán nhãn các trường hợp tương tác
  - **Create** - Tạo đối tượng
  - **Read** - Tra cứu thông tin được lưu trong đối tượng
  - **Update** - Cập nhật giá trị thuộc tính của đối tượng
  - **Delete** - Xóa đối tượng
  - **Execute** - Yêu cầu đối tượng thực hiện hành động
    - Quan hệ giữa các đối tượng.
- Biểu diễn dưới dạng ma trận

# Ca sử dụng và lớp lõi

Để hỗ trợ xác định các đối tượng liên quan đến các hoạt động nghiệp vụ trong phạm vi ca sử dụng:

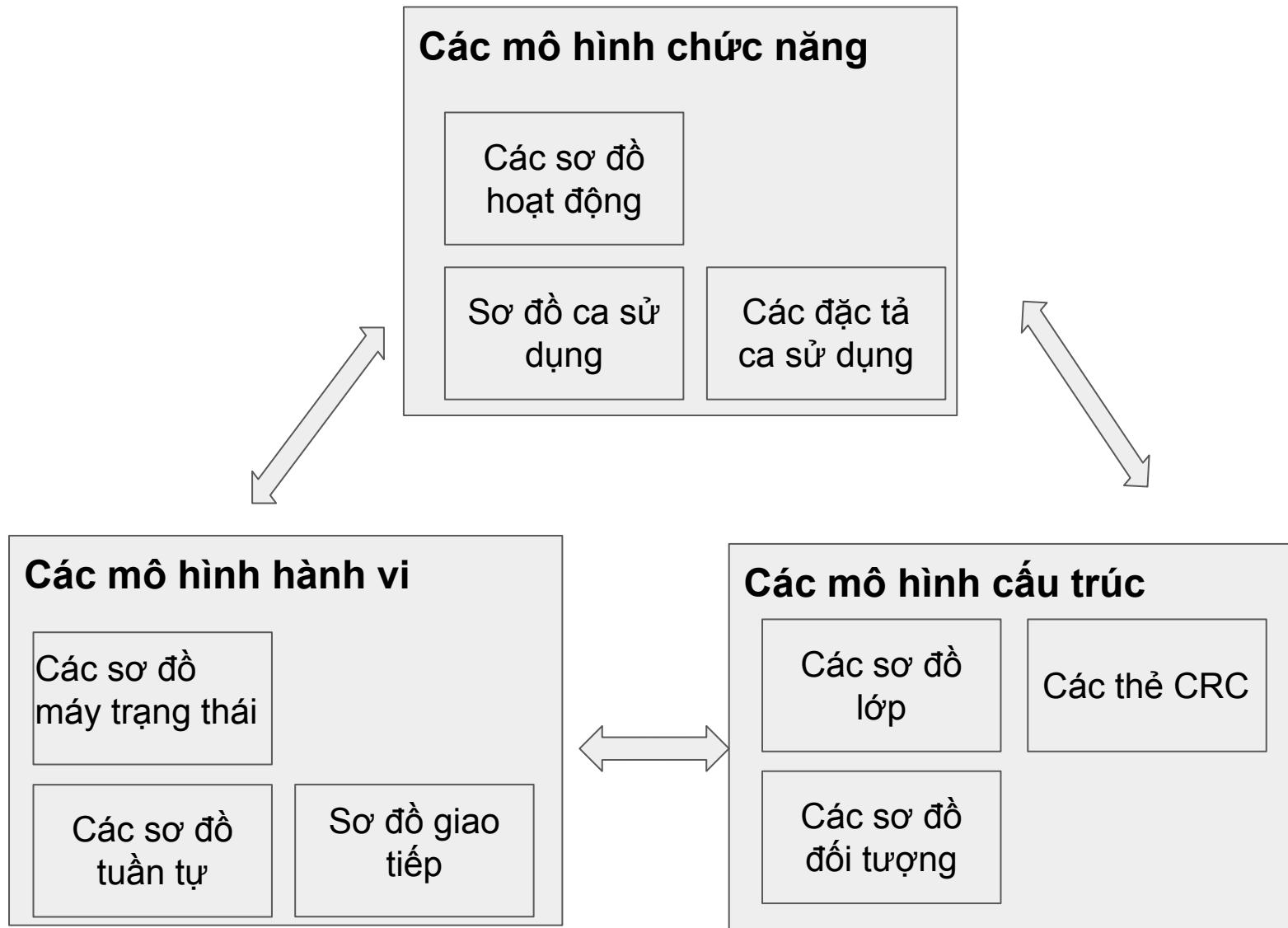
- **Phân tích CRUD:** Với mỗi lớp lõi kiểm tra xem có các ca sử dụng tạo, đọc, cập nhật và xóa đối tượng hay không?

Lớp lõi/ Ca sử dụng	Tạo tài khoản	Tìm hồ sơ	Tạo báo cáo
<b>Khách hàng</b>	C		
<b>Hóa đơn</b>	R		
<b>Đơn hàng</b>	R	R	R
...			

# Nội dung

- Các mô hình tương tác
    - Sơ đồ tuần tự
    - Sơ đồ giao tiếp
  - Biểu diễn trạng thái của đối tượng có phạm vi rộng
    - Sơ đồ máy trạng thái hành vi
  - Kỹ thuật phát hiện các mối quan hệ
    - Phân tích ma trận CRUD(E)
  - Tổng kết các nội dung phân tích
- 

# Các mô hình



# Bản đề xuất hệ thống (Chương 2)

## 1. Mục lục

2. Môi trường, nghiệp vụ & nhu cầu sử dụng hệ thống

## 3. Các mô hình chức năng

Tập đặc tả ca sử dụng, và sơ đồ ca sử dụng, các sơ đồ hoạt động.

## 4. Các mô hình cấu trúc

Tập thẻ CRC, sơ đồ lớp, và các sơ đồ đối tượng.

## 5. Các mô hình hành vi

Tập sơ đồ tuần tự, sơ đồ giao tiếp, máy trạng thái, ma trận CRUD.

## 6. Các phụ lục

Bao gồm các thông tin bổ xung có liên quan đến đề xuất, thường được sử dụng để hỗ trợ hệ thống được gợi ý. Trong đó có thể bao gồm các kết quả khảo sát, hoặc phỏng vấn, các báo cáo và thống kê, v.v.



# **Phân tích thiết kế Hệ thống**

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2021

# **Thiết kế lớp**

# Nội dung

- Vấn đề thiết kế lớp
- Các chỉ số chất lượng
- Các nguyên lý SOLID

# Nội dung

- Vấn đề thiết kế lớp
- Các chỉ số chất lượng
- Các nguyên lý SOLID

# Các hoạt động thiết kế đối tượng

- Mở rộng và tiếp tục phát triển các mô hình phân tích.
- Tạo các mô tả cho thiết kế chi tiết lớp và phương thức
  - Bổ xung các đặc tả cho mô hình hiện có
  - Xác định các tiềm năng tái sử dụng
  - Tái cấu trúc thiết kế
  - Tối ưu hóa thiết kế
  - Ánh xạ các lớp lĩnh vực ứng dụng và ngôn ngữ triển khai
- Thay đổi 1 lớp trên 1 tầng có thể khiến các lớp khác (có thể trên các tầng khác) có gắn kết với lớp đó thay đổi theo.

# Các đặc tả bổ xung

- Kiểm tra các mô hình cấu trúc và hành vi để đảm bảo các thành phần đầy đủ và cần thiết.
- Thiết lập giới hạn nhìn của các thuộc tính và phương thức của mỗi lớp (gắn với ngôn ngữ triển khai).
- Xác định nguyên mẫu của các phương thức: Tên, các tham số, kiểu trả về.
- Xác định các ràng buộc mà đối tượng phải đáp ứng
  - Tiền điều kiện;
  - Hậu điều kiện;
  - Tính chất bất biến.
  - (*Có thể được mô tả bằng OCL*).
- Các thông tin bổ xung được mô tả bằng hợp đồng, hoặc được thêm vào thẻ CRC và sơ đồ lớp.

# Các đặc tả bỗ xung<sub>(2)</sub>

- Chúng ta cũng cần xác định cách xử lý các ngoại lệ:
  - (*phát sinh nếu các ràng buộc không được đảm bảo*)
  - Hệ thống có thể xử lý đơn giản bằng cách bỏ qua?
  - Hoàn tác các thay đổi đã thực hiện dẫn đến ngoại lệ?
  - Để người dùng lựa chọn phương án xử lý ngoại lệ nếu có nhiều phương án?
- Người thiết kế xác định các lỗi mà hệ thống phải xử lý
  - Gắn kết với ngôn ngữ lập trình
  - Sử dụng mã lỗi
  - Sử dụng cơ chế exception (như trong C++ hoặc Java)

# Xác định các tiềm năng tái sử dụng

- Trong pha thiết kế chúng ta có thể sử dụng:
  - Mẫu thiết kế;
  - Nền tảng;
  - Thư viện;
  - Thành phần.
- Tiềm năng tái sử dụng có thể thay đổi theo tầng
  - Ví dụ: 1 thư viện có thể ít hữu ích trong tầng lĩnh vực ứng dụng, nhưng lại rất hữu ích trong tầng nền tảng.
- Các mẫu thiết kế
  - Rất hữu ích để gom nhóm các lớp tương tác có thể cung cấp giải pháp cho 1 vấn đề phổ biến. Có thể giải quyết “1 vấn đề thiết kế điển hình trong 1 ngũ cảnh cụ thể”.
  - (*Các chi tiết sẽ được cung cấp sau*).

# Nền tảng và thư viện

- Nền tảng cung cấp 1 tập lớp có thể được sử dụng làm khung phát triển chương trình ứng dụng
  - Có thể triển khai toàn bộ hoặc 1 phần của hệ thống.
  - CORBA, DCOM, Spring Boot, Struts, Qt, v.v..
- Thư viện bao gồm 1 tập lớp, tương tự nền tảng, được thiết kế để tái sử dụng
  - Có nhiều thư viện hỗ trợ các mảng ứng dụng
    - Ví dụ: Xử lý số học và thống kê; Phát triển giao diện đồ họa (tầng HCI); Kết nối với CSDL (tầng quản lý dữ liệu - DAM).
- Nền tảng và thư viện thường cho phép kế thừa các lớp của nó, có thể kế thừa để tái sử dụng lớp hoặc tạo đối tượng trực tiếp từ các lớp đã có.
  - (*Làm tăng tính ghép nối*).

# Thành phần

- Thành phần/component:
  - Mảng phần mềm, được đóng gói để có thể nhúng vào 1 hệ thống để cung cấp 1 tập chức năng cụ thể.
    - Ví dụ các thành phần được triển khai dựa trên các công nghệ ActiveX hoặc JavaBean.
  - Cung cấp 1 tập phương thức giao diện để tương tác với các đối tượng có trong nó.
    - Lô-gic hoạt động bên trong thành phần được ẩn sau các API.
  - Có thể được triển khai bằng các lớp thư viện và nền tảng, nhưng cũng có thể được sử dụng để triển khai nền tảng.
  - Nếu giao diện không thay đổi, thì cập nhật phiên bản mới của thành phần thường không yêu cầu thay đổi mã nguồn
    - Có thể không yêu cầu biên dịch lại.
  - Thường được sử dụng để giản lược các chi tiết triển khai.

# Tái cấu trúc

- Đóng gói phần chung / factoring
  - Lớp mới có thể được gắn kết với các lớp cũ thông qua kế thừa, tổng hợp hoặc liên kết
    - Lưu ý các vấn đề ghép nối, thống nhất, và đồng sinh.
- Triển khai các mối quan hệ trên sơ đồ lớp như những thuộc tính.
  - Các ngôn ngữ hướng đối tượng hầu như không phân biệt các quan hệ như bộ phận-tổng thể và liên quan.
    - Các mối quan hệ phải được chuyển thành các thuộc tính trong lớp.

# Đóng gói phần chung

- Tách và đóng gói phần giống nhau và khác nhau của các thứ được quan tâm
- Các lớp mới được hình thành thông qua:
  - Quan hệ khái quát hóa (thuộc loại), hoặc
    - Tạo lớp bậc cao hơn (ví dụ, tạo lớp Employee từ tập vị trí công việc)
  - Chi tiết hóa, hoặc
    - Tạo lớp cụ thể (ví dụ, tạo lớp Secretary hoặc Bookkeeper từ lớp Employee)
  - Quan hệ tổng hợp (có các thành phần)

# Tối ưu hóa thiết kế

Có thể được thực hiện để tạo thiết kế hiệu quả hơn

- Kiểm tra các đường dẫn tới đối tượng:
  - Có những trường hợp thông điệp từ 1 đối tượng tới 1 đối tượng khác phải qua nhiều bước trung gian.
  - Nếu đường dẫn dài và thông điệp được gửi thường xuyên, thì cần nhắc rút ngắn đường truyền thông điệp.
    - Có thể bổ xung thuộc tính vào đối tượng gửi thông điệp để có thể truy cập trực tiếp.

# Tối ưu hóa thiết kế: Thuộc tính

- Xác định những phương thức sử dụng thuộc tính và những đối tượng sử dụng phương thức.
- Nếu chỉ có phương thức đọc và cập nhật sử dụng thuộc tính, và chỉ có đối tượng thuộc 1 lớp gửi thông điệp đọc và cập nhật đối tượng
  - Thuộc tính đó có thể thuộc về lớp gọi thay vì lớp được gọi
  - Chuyển thuộc tính sang lớp gọi có thể giúp hệ thống hoạt động nhanh hơn.

# Tối ưu hóa thiết kế: Luồng ra

- Luồng ra bao gồm các thông điệp được gửi trực tiếp và gián tiếp trong thời gian thực hiện phương thức
  - Thông điệp trực tiếp được gửi bởi chính phương thức đó
  - Thông điệp gián tiếp được gửi bởi phương thức được gọi trong khi thực hiện phương thức đó
- Nếu kích thước luồng ra quá lớn so với các phương thức khác trong hệ thống, thì phương thức đó có thể cần được điều chỉnh.

# Tối ưu hóa thiết kế: Thú tự thực hiện

- Kiểm tra thứ tự thực hiện các lệnh trong phương thức được sử dụng thường xuyên
  - Có những trường hợp có thể sắp xếp lại các câu lệnh để đạt được hiệu quả cao hơn.
  - Ví dụ: Giả định kịch bản tìm kiếm, trong đó phạm vi tìm kiếm được thu hẹp sau khi tìm kiếm theo 1 thuộc tính.
    - Có thể tìm cách tối ưu hóa xử lý theo 1 thứ tự thuộc tính tối ưu.
    - $A \cap B \cap C$  vs.  $A \cap C \cap B$

# Tối ưu hóa thiết kế: Sử dụng bộ nhớ đệm

- Tránh tính toán lại thuộc tính suy diễn:
  - Ví dụ tạo thuộc tính tổng/total để lưu tổng giá trị đơn hàng.
  - Thiết lập cơ chế kích hoạt tiến trình tính toán.
  - Chỉ tính lại giá trị của thuộc tính suy diễn khi thay đổi các thuộc tính thành phần được sử dụng trong tính toán.

# Ánh xạ các lớp linh vực ứng dụng

- Tái cấu trúc các thành phần đa kế thừa nếu ngôn ngữ chỉ hỗ trợ đơn kế thừa.
- Tái cấu trúc các thành phần kế thừa nếu ngôn ngữ không hỗ trợ kế thừa.
- Tránh triển khai 1 thiết kế hướng đối tượng với 1 ngôn ngữ lập trình không hỗ trợ lập trình hướng đối tượng.

# Các ràng buộc và các hợp đồng

- Hợp đồng bao gồm 1 tập ràng buộc/constraints và các đảm bảo / guarantees
  - Nếu đối tượng gửi (client) đáp ứng các ràng buộc, thì đối tượng cung cấp dịch vụ (server) đảm bảo hành vi mong đợi
- Hợp đồng mô tả thông điệp được gửi giữa các đối tượng
  - Được tạo cho các phương thức công khai của lớp.
  - Cần chứa đủ thông tin để người lập trình có thể hiểu hành vi mong đợi của phương thức.
- Các loại ràng buộc:
  - Tiền điều kiện - Phải đúng trước khi phương thức được thực hiện
  - Hậu điều kiện - Phải đúng sau khi phương thức kết thúc
  - Bất biến - Phải luôn đúng đối với tất cả các đối tượng.

# Biểu diễn bất biến

Mặt trước:

Tên lớp: Order	ID: 1	Kiểu: Cụ thể, Linh vực
<b>Mô tả:</b> Biểu diễn đơn hàng		<b>Các sử dụng liên quan:</b> 1, 2, 3, ...
<b>Các trách nhiệm</b>		<b>Các đối tác</b>
...		...

Mặt sau:

Các thuộc tính:

order\_id: long

customer: Customer

cust\_id: long {cust\_id = customer.getCustID()}

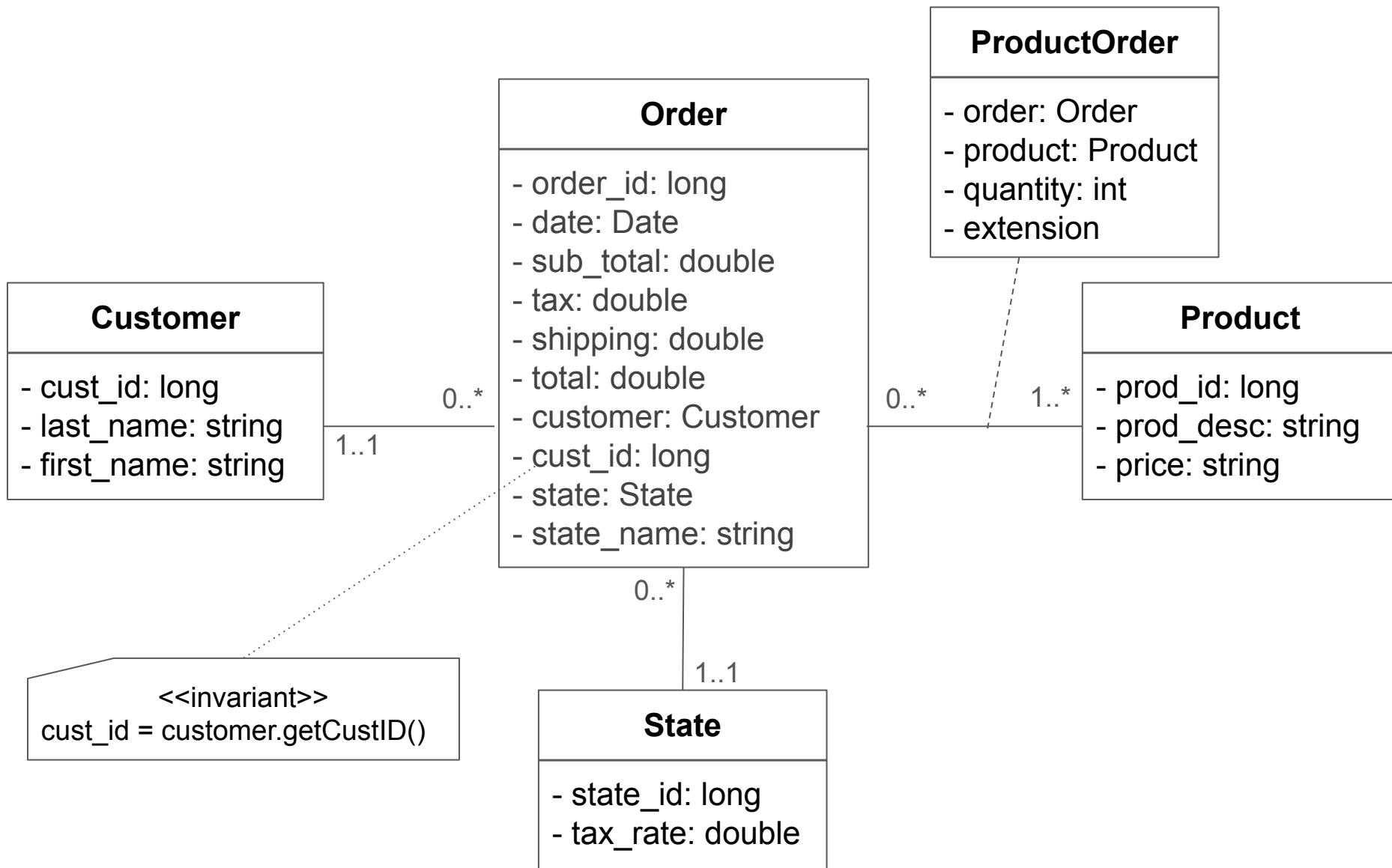
Các mối quan hệ:

**Khái quát hóa (thuộc loại):**

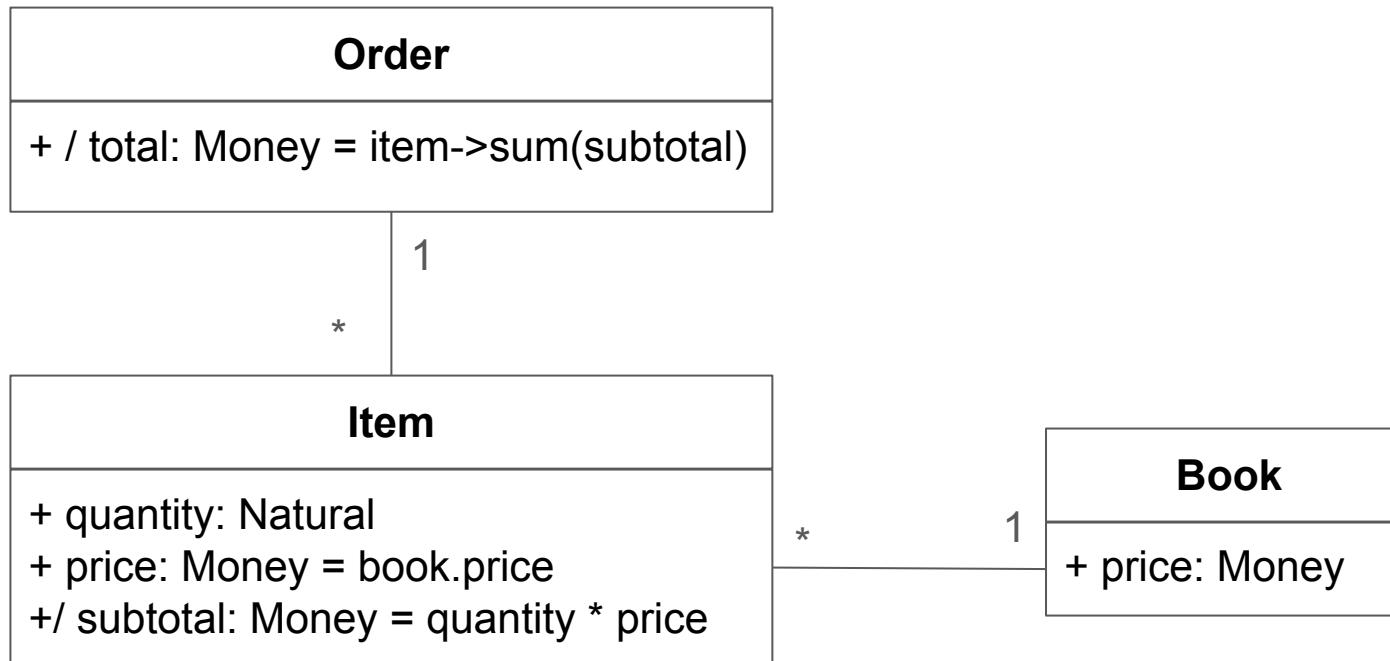
**Tổng hợp (Có các phần):**

**Các mối liên quan khác:**

# Ví dụ 5.1. Biểu diễn bất biến trên sơ đồ



# Ví dụ 5.2. Biểu diễn bất biến ngoài sơ đồ



Context Order::total:Money

derive:

`self.item->sum(subtotal)`

Context Item::subtotal:Money

derive:

`self.quantity * self.price`

Các mô tả  
OCL

# Biểu mẫu hợp đồng thông điệp

(Phi chuẩn, đặc tả thông điệp)

Tên phương thức:	Tên lớp:	Mã số:
Mã khách:		
Ca sử dụng liên quan:		
Mô tả các trách nhiệm:		
Các tham số nhận được:		
Kiểu dữ liệu trả về:		
Tiền điều kiện:		
Hậu điều kiện:		

# Đặc tả phương thức

- Mô tả các chi tiết của mỗi phương thức
  - Để người lập trình viết mã nguồn
  - (*đây là tài liệu thiết kế rất gần triển khai*)
- Không có quy chuẩn, tuy nhiên có thể bao gồm các mục:
  - Thông tin tổng quan (Tên phương thức, tên lớp, v.v...)
  - Sự kiện - Kích hoạt phương thức (ví dụ khi nhấn chuột)
  - Cấu trúc thông điệp được truyền tới bao gồm cả các tham số và giá trị trả về
  - Đặc tả giải thuật
  - Các thông tin liên quan khác

# Biểu mẫu đặc tả phương thức

Tên phương thức:	Tên lớp:	ID:
Mã thỏa thuận:	Lập trình viên:	Thời hạn:
Ngôn ngữ lập trình: <input type="checkbox"/> Visual Basic <input type="checkbox"/> Smalltalk <input type="checkbox"/> C++ <input type="checkbox"/> Java		
Kích hoạt/Sự kiện:		
Các tham số nhận được: Kiểu dữ liệu:		Ghi chú:
Thông điệp đã gửi & tham số: TênLớp.TênPhươngThức	Kiểu dữ liệu của tham số:	Ghi chú:
Tham số trả về: Kiểu dữ liệu:	Ghi chú	
Đặc tả giải thuật:		
Các ghi chú khác:		

# Nội dung

- Vấn đề thiết kế lớp
- Các chỉ số chất lượng
- Các nguyên lý SOLID



# Chỉ số chất lượng thiết kế

- Ghép nối/Coupling - Phản ánh mức độ gắn kết giữa các thành phần (lớp, phương thức, v.v..).
- Thống nhất/Cohesion - Phản ánh mức độ chuyên dụng của 1 thành phần.
- Đồng sinh/Connascence - Khái quát hóa đồng thời kết hợp chỉ số ghép nối và chỉ số thống nhất.
  - (*Xuất hiện cùng nhau/Connascence means to be born together*).

# Tính ghép nối

- Ghép nối chặt khiến thay đổi trong 1 phần của thiết kế có thể yêu cầu thay đổi trong các phần khác
- Phân loại
  - Ghép nối tương tác - hình thành từ trao đổi thông điệp
  - Ghép nối kế thừa - hình thành từ quan hệ kế thừa
  - [Coad and Edward Yourdon, *Object-Oriented Design*, 1991]
- Các giới hạn được áp dụng để giảm mức ghép nối
  - Ghép nối tương tác: Quy tắc Demeter
  - Ghép nối kế thừa: Duy trì ý nghĩa khái quát hóa/chi tiết hóa, áp dụng quy tắc khả thay/substitutability

# Quy tắc Demeter

*Thông điệp chỉ nên được gửi từ 1 đối tượng tới:*

- Chính nó;
- Đối tượng là thuộc tính của nó hoặc lớp trên;
- Đối tượng là tham số của phương thức
- Đối tượng được tạo bởi phương thức
- Đối tượng được lưu trong biến toàn cục

# Các trường hợp làm tăng mức độ ghép nối

- Phương thức gọi truyền các thuộc tính cho phương thức được gọi
- Phương thức gọi phụ thuộc vào giá trị được trả về bởi phương thức được gọi

# Các mức ghép nối tương tác

Theo thứ tự từ lỏng (được mong đợi) đến chặt (không mong muốn):

- Không có ghép nối trực tiếp / No Direct Coupling
- Dữ liệu / Data
- Dấu ấn / Stamp
- Điều khiển / Control
- Chung hoặc toàn cục / Common or Global
- Nội dung hoặc biểu diễn / Content or Pathological

# Các mức ghép nối tương tác<sub>(2)</sub>

Mức độ	Mô tả
Lỏng	Không có ghép nối trực tiếp
	Dữ liệu
	Dấu ấn
	Điều khiển
	Chung hoặc toàn cục
	Nội dung hoặc Biểu diễn
Chặt	

Nguồn: Meilir Page-Jones, *The Practical Guide to Structured Systems Design*, 2nd, 1978.

# Ghép nối kế thừa

- Có nhiều luồng quan điểm cho rằng ghép nối kế thừa là cần thiết.
- Tuy nhiên vẫn cần cân nhắc các trường hợp:
  - Phương thức được định nghĩa trong lớp dưới có nên gọi phương thức được định nghĩa trong lớp trên?
  - Phương thức được định nghĩa trong lớp dưới có nên sử dụng thuộc tính được định nghĩa trong lớp trên?
  - Phương thức được định nghĩa trong lớp trên có nên dựa vào các lớp con của nó để định nghĩa 1 phần xử lý?
    - *(Có thể triển khai dựa trên phương thức ảo và đa hình)*
- Người thiết kế cần cân đối việc phá vỡ quy tắc đóng gói và ẩn dữ liệu và tăng những ghép nối mong đợi giữa các lớp dưới và các lớp trên.

# Tính ghép nối trong thiết kế

- Hướng tới ghép nối lỏng, đưa chỉ số ghép nối về mức nhỏ nhất có thể.
- Trong thực tế có những ghép nối khó tránh:
  - Ví dụ 1 lớp HCl phụ thuộc vào 1 lớp lĩnh vực ứng dụng - lớp biểu diễn dữ liệu được hiển thị trên giao diện -
  - *Biểu mẫu thông tin sinh viên hiển thị các thông tin của 1 đối tượng thuộc lớp Student trong tầng lĩnh vực ứng dụng.*

# Tính thống nhất

- Mức độ thống nhất cao thể hiện tính chuyên dụng của mỗi thành phần: 1 lớp chỉ biểu diễn 1 loại đối tượng, 1 phương thức chỉ có 1 trách nhiệm.
- Phân loại:
  - Thống nhất phương thức
    - 1 phương thức có đảm nhận nhiều hơn 1 trách nhiệm?
    - Càng nhiều trách nhiệm thì càng phức tạp và khó triển khai.
  - Thống nhất lớp
    - Các thuộc tính và phương thức có biểu diễn 1 loại đối tượng?
    - Không nên trộn lẫn nhiều vai trò, loại đối tượng vào 1 lớp.
  - Thống nhất cây kế thừa
    - Quan hệ giữa các lớp trong 1 cây kế thừa nên biểu diễn nhất quán quan hệ "1 loại của"/"a-kind-of", không phải tổng hợp hoặc liên quan.
    - Các lớp tương thích với nguyên lý khả thay / Substitutability
- Hướng tới thống nhất ở mức độ cao nhất.

# Thống nhất phương thức

- Thể hiện tính thống nhất trong 1 phương thức
  - Các lệnh trong phương thức cùng xử lý 1 công việc.
  - Mỗi phương thức chỉ nên làm 1 việc - lãnh 1 trách nhiệm
- Các mức thống nhất phương thức theo thứ tự giảm dần từ cao (được mong đợi) đến thấp (không mong muốn)
  - Chức năng / Functional
  - Tuần tự / Sequential
  - Giao tiếp / Communicational
  - Tiến trình / Procedural
  - Tạm thời hoặc cổ điển / Temporal or Classical
  - Lô-gic / Logical
  - Ngẫu nhiên / Coincidental

# Các mức thống nhất phương thức

Mức độ	Mô tả	
Cao ↓	Chức năng	Mỗi phương thức thực hiện 1 công việc (ví dụ, tính GPA hiện tại)
	Tuần tự	Phương thức kết hợp 2 chức năng, trong đó đầu ra của chức năng đầu tiên được sử dụng như đầu vào của chức năng tiếp theo (ví dụ, chuẩn hóa và kiểm tra GPA hiện tại)
	Giao tiếp	Phương thức kết hợp 2 chức năng sử dụng chung tập thuộc tính (ví dụ, tính điểm GPA hiện tại và điểm GPA tích lũy)
	Tiến trình	Phương thức hỗ trợ nhiều chức năng liên kết yếu, ví dụ, tính GPA của sinh viên, in thông tin sinh viên, tính GPA tích lũy.
	Tạm thời hoặc cổ điển	Phương thức hỗ trợ nhiều chức năng liên quan theo thời gian (ví dụ, khởi tạo tất cả các thuộc tính)
	Lô-gic	Phương thức hỗ trợ nhiều chức năng liên quan, nhưng lựa chọn chức năng cụ thể được thực hiện dựa trên biến điều khiển được cung cấp như tham số. Ví dụ, phương thức có thể mở tài khoản tiết kiệm, tài khoản ghi nợ, hoặc tính lãi dựa theo thông điệp được gửi tới bởi phương thức gọi.
	Ngẫu nhiên	Không thể xác định mục đích của phương thức hoặc nó thực hiện nhiều chức năng không liên quan. Ví dụ, phương thức có thể cập nhật hồ sơ khách hàng, tính lãi khoản vay, và phân tích cấu trúc giá của đối thủ.

Nguồn: Page-Jones, *The Practical Guide to Structured System*, và Myers, *Composite/Structured Design*.

# Thống nhất lớp

Các yếu tố đảm bảo thống nhất lớp:

- Tất cả các thuộc tính và các phương thức của 1 lớp, cùng hỗ trợ biểu diễn 1 thứ
- Tất cả các thuộc tính và các phương thức trong 1 lớp đều cần thiết
  - Mỗi lớp chỉ nên có các thuộc tính và phương thức cần thiết để biểu diễn các đối tượng của nó.
- Ví dụ,
  - Lớp Student cần có các thuộc tính như: MSSV, Họ, Tên, Địa Chỉ
  - Nhưng không có các thuộc tính như: CPU, RAM, OS

# Các mức thống nhất lớp

*Các mức thống nhất lớp theo thứ tự giảm dần:*

- Lý tưởng / Ideal
- Hỗn hợp vai trò / Mixed-Role
- Hỗn hợp lĩnh vực / Mixed-Domain
- Hỗn hợp đối tượng / Mixed-Instance

# Các mức thống nhất lớp<sub>(2)</sub>

Mức độ	Mô tả		
Cao ↓	Lý tưởng	Lớp không có yếu tố hỗn hợp	
	Hỗn hợp vai trò	Lớp có thành phần trực tiếp kết nối đối tượng của lớp với đối tượng thuộc lớp khác trên cùng tầng (ví dụ, tầng lĩnh vực ứng dụng), nhưng các thuộc tính đó không liên quan với ý nghĩa cơ bản của lớp.	
	Hỗn hợp lĩnh vực	Lớp có thành phần trực tiếp kết nối đối tượng thuộc lớp với đối tượng thuộc lớp khác tầng, nhưng thành phần đó không liên quan đến ý nghĩa cơ bản của lớp. Trong những trường hợp này, các thành phần liên kết thuộc về lớp trên tầng kia. Ví dụ, thuộc tính cổng nằm trong 1 lớp lĩnh vực đúng ra phải nằm trong lớp thuộc tầng kiến trúc hệ thống liên quan với lớp lĩnh vực.	
	Hỗn hợp đối tượng	Lớp biểu diễn nhiều loại đối tượng. Lớp như vậy cần được phân tách thành các lớp riêng biệt. Thông thường, mỗi đối tượng chỉ sử dụng 1 phần định nghĩa của lớp.	
Nguồn: Page-Jones, <i>Fundamentals of Object-Oriented Design in UML</i> .			

# Lớp thống nhất lý tưởng

- Chứa nhiều phương thức có thể được nhìn thấy từ bên ngoài lớp,
- Mỗi phương thức công khai chỉ thực hiện 1 chức năng,
- Các phương thức chỉ sử dụng các thuộc tính và các phương thức khác được định nghĩa trong cùng lớp hoặc (các) lớp trên của nó,
- Không có ghép nối mức điều khiển giữa các phương thức công khai của nó.

[Glenford J. Myers, *Composite/Structured Design*, 1998]

# Mức thống nhất: Hỗn hợp vai trò

- Tính hỗn hợp vai trò làm giảm khả năng tái sử dụng
- Ví dụ 1: lớp Student và Room cùng thuộc tầng lĩnh vực ứng dụng, nhưng phòng học không phải thuộc tính mô tả sinh viên.
  - Thiết kế lớp Student trực tiếp gắn kết với lớp Room làm phát sinh vấn đề hỗn hợp vai trò.
- Ví dụ 2: Lớp Student và lớp Registration cũng cùng thuộc tầng lĩnh vực ứng dụng, và sinh viên là 1 phần trong đăng ký học tập
  - Thiết kế lớp Registration trực tiếp gắn kết với lớp Student không phát sinh vấn đề hỗn hợp vai trò

# Mức thống nhất: Hỗn hợp linh vực

- Phát sinh khi lớp chưa thành phần trực tiếp gắn kết với lớp trên 1 tầng khác / không thuộc tầng của nó.
- Kiểm tra tính năng có thiết yếu đối với lớp không?
  - Ví dụ 1: Lớp Order và lớp Printer thuộc các tầng khác nhau
    - Triển khai phương thức in hóa đơn trong lớp đơn hàng? Có thể triển khai lớp đơn hàng mà không có phương thức in hay không?
  - Ví dụ 2: Lớp OrderDetailView và lớp Order cũng thuộc các tầng khác nhau
    - Có thể triển khai giao diện đơn hàng mà không đọc dữ liệu đơn hàng hay không?

# Mức thống nhất: Hỗn hợp đối tượng

- Lớp biểu diễn nhiều hơn 1 nhóm đối tượng / Có thành phần không được sử dụng cho 1 số đối tượng
- Ví dụ: Giả sử chúng ta đang triển khai 1 lớp Person với các phương thức:
  - GetSalary() trả về lương nếu là nhân viên.
  - GetCashBack() trả về số dư tài khoản hoàn tiền nếu là khách hàng.
  - Với mỗi đối tượng Person là của 1 nhân viên hoặc 1 khách hàng chúng ta đều có tính năng không sử dụng đến
    - Cho thấy lớp Person đang quá rộng.
  - Giải pháp:
    - Tách các tính năng gắn với nhân viên và đưa vào 1 lớp Employee
    - Tách các tính năng gắn với khách hàng và đưa vào 1 lớp Customer
    - Các lớp Employee và Customer kế thừa lớp Person

# Đồng sinh

- Các thành phần phụ thuộc lẫn nhau khiến thay đổi trong 1 thành phần kéo theo thay đổi trong các thành phần khác.
- Kết hợp chỉ số ghép nối và chỉ số thống nhất trong các giới hạn đóng gói
- Có 3 mức đóng gói được sử dụng:
  - Đóng gói mức 0: Đóng gói trong phạm vi 1 câu lệnh
  - Đóng gói mức 1: Phương thức - tổng hợp nhiều câu lệnh
    - Kết hợp thống nhất phương thức và ghép nối tương tác.
  - Đóng gói mức 2: Lớp - Chứa cả thuộc tính và phương thức
    - Kết hợp thống nhất lớp, thống nhất cây kế thừa, và ghép nối kế thừa

# Đánh giá mức độ đồng sinh

- Mức độ đồng sinh thường được đánh giá trong phạm vi đóng gói mức 1 (mức phương thức), và đóng gói mức 2 (mức lớp).
- Mã nguồn cần hướng tới:
  - Cực tiểu hóa đồng sinh giữa các giới hạn đóng gói (ít liên kết phụ thuộc giữa các thành phần).
  - Cực đại hóa đồng sinh trong phạm vi đóng gói (nhiều quan hệ phụ thuộc trong 1 thành phần).
  - => Lớp con không nên trực tiếp truy cập thuộc tính hoặc phương thức của lớp cha

# Các trường hợp đồng sinh

- Tên / Name
- Kiểu hoặc Lớp / Type or Class
- Quy ước / Convention
- Giải thuật / Algorithm
- Vị trí / Position

# Các trường hợp đồng sinh<sub>(2)</sub>

Phân loại	Mô tả
Tên	Nếu 1 phương thức sử dụng 1 thuộc tính, thì nó được gắn với tên của thuộc tính. Nếu tên thuộc tính thay đổi, nội dung của phương thức cũng sẽ phải thay đổi.
Kiểu hoặc Lớp	Nếu 1 lớp có 1 thuộc tính thuộc kiểu A, nó được gắn với kiểu của thuộc tính. Nếu kiểu của thuộc tính thay đổi, khai báo thuộc tính cũng sẽ phải thay đổi.
Quy ước	Một lớp có 1 thuộc tính mà miền giá trị có ý nghĩa (ví dụ, số tài khoản với các giá trị trong phạm vi từ 1000 tới 1999 là các tài sản). Nếu khoảng thay đổi, thì tất cả phương thức sử dụng thuộc tính cũng cần phải thay đổi.
Giải thuật	Hai phương thức khác nhau của 1 lớp cùng phụ thuộc vào 1 giải thuật để chạy đúng (ví dụ, kiểm tra tính hợp lệ của địa chỉ email khi tạo tài khoản và khi gửi yêu cầu khôi phục mật khẩu). Nếu giải thuật cơ sở cần thay đổi, thì phương thức thêm và tìm kiếm cũng sẽ phải thay đổi.
Vị trí	Thứ tự mã nguồn trong 1 phương thức hoặc thứ tự các tham số trong 1 phương thức là đặc biệt quan trọng để phương thức chạy đúng. Nếu bất kỳ thứ tự nào sai, thì phương thức có thể hoạt động sai.

Nguồn: Meilir Page-Jones, *Comparing Techniques by Means of Encapsulation and Connascence* and Meilir Page-Jones, *Fundamentals of Object-Oriented Design in UML*.

## Ví dụ 5.3. Đồng sinh tên

```
struct student {  
    long id;  
    char fullname[256];  
};
```

Mã nguồn sử dụng cấu trúc student phụ thuộc vào các định danh của cấu trúc student.

## Ví dụ 5.4. Đồng sinh kiểu

```
char *s = NULL;
```

```
s = "Hi!"; // OK
```

```
s = 100; // NOK
```

*Mã nguồn sử dụng biến s phụ thuộc vào kiểu của biến s.*

## Ví dụ 5.5. Đóng sinh ý nghĩa

```
struct bst_node *ret = bst_put(bst, key, value);
```

```
if (ret == NULL) {  
    printf("Khóa mới.");  
} else {  
    printf("Khóa đã tồn tại.");  
}
```

*Mã nguồn sử dụng hàm bst\_put phụ thuộc vào quy ước đối với giá trị trả về.*

## Ví dụ 5.6. Đóng sinh vị trí

```
int less(int a, int b) {  
    return a < b;  
}
```

```
if (less(3, 5)) {  
    printf("OK!");  
}
```

*Hàm less kiểm tra mệnh đề  $a < b$ , mã nguồn sử dụng hàm less cần truyền tham số theo đúng thứ tự.*

## Ví dụ 5.7. Đồng sinh thuật toán

```
char *vb_encode(int *inp);
```

```
int *vb_decode(char *inp);
```

```
int arr[100]; // Sử dụng lính canh để đánh dấu điểm cuối
```

```
...
```

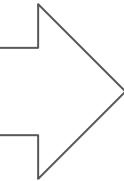
```
char *data = vb_encode(arr);
```

```
int *arr = vb_decode(data);
```

*Thuật toán giải nén phụ thuộc vào thuật toán nén.*

# Nội dung

- Vấn đề thiết kế lớp
- Các chỉ số chất lượng
- Các nguyên lý SOLID



# Nguyên lý trách nhiệm duy nhất

- Nguyên lý trách nhiệm duy nhất / Single Responsibility Principle (SRP)

*Mỗi lớp chỉ nên có 1 lý do duy nhất để thay đổi /  
A class should have only one reason to change*

[Robert .C Martin]

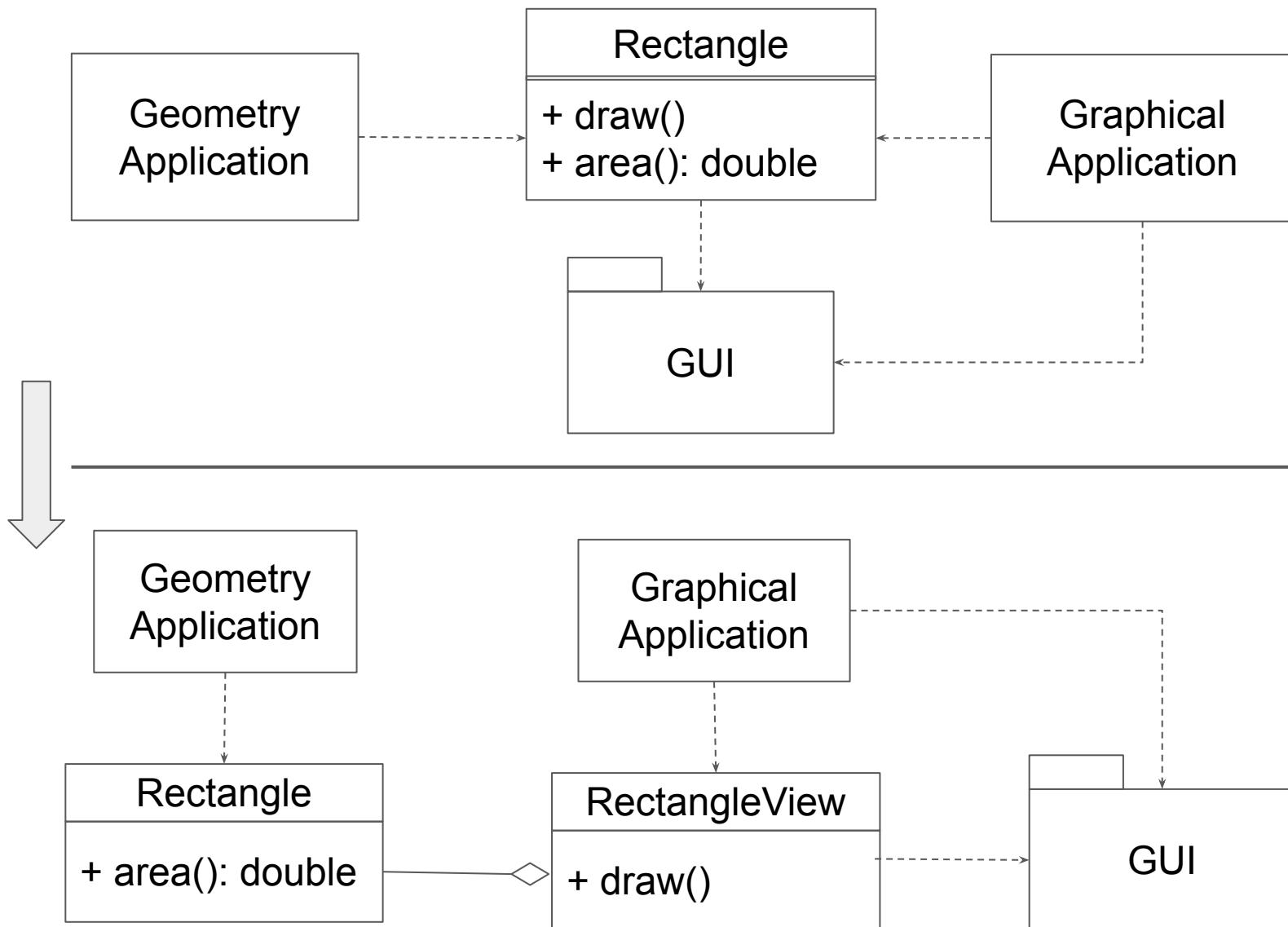
- Trách nhiệm được gán cho lớp
- Nếu phát sinh nhu cầu triển khai chức năng mới, hoặc chức năng đang có thay đổi, thì các trách nhiệm của các lớp phải thay đổi.
- Lớp có 1 trách nhiệm sẽ chỉ có 1 lý do để thay đổi.

*Mỗi thành phần chỉ nên có đảm nhận 1 trách nhiệm*

# Trách nhiệm và tính thống nhất

- Lớp có tính thống nhất ở mức cao chỉ triển khai 1 trách nhiệm (hoặc ít trách nhiệm) / Lớp chỉ triển khai 1 trách nhiệm có tính thống nhất ở mức cao.
- Lớp có tính thống nhất ở mức thấp thường vi phạm nguyên lý SRP / Lớp vi phạm SRP thường có tính thống nhất ở mức thấp.
- Ưu điểm: Các lớp 1 trách nhiệm thường nhỏ, dễ tái sử dụng, và dễ hiểu hơn, và ít thay đổi.
- Lưu ý: Phân tách trách nhiệm có thể phức tạp và có hiệu ứng phụ, áp dụng SRP khi thực sự cần thiết, thực sự có thay đổi thường xuyên.

# Ví dụ 5.8. SRP



# Nguyên lý Mở-Đóng

- Nguyên lý Mở-Đóng / Open-Close Principle (OCP)

*Các thực thể phần mềm cần có khả năng mở rộng mà không thay đổi /*

*Software entities should be open for extension, but closed for modifications.*

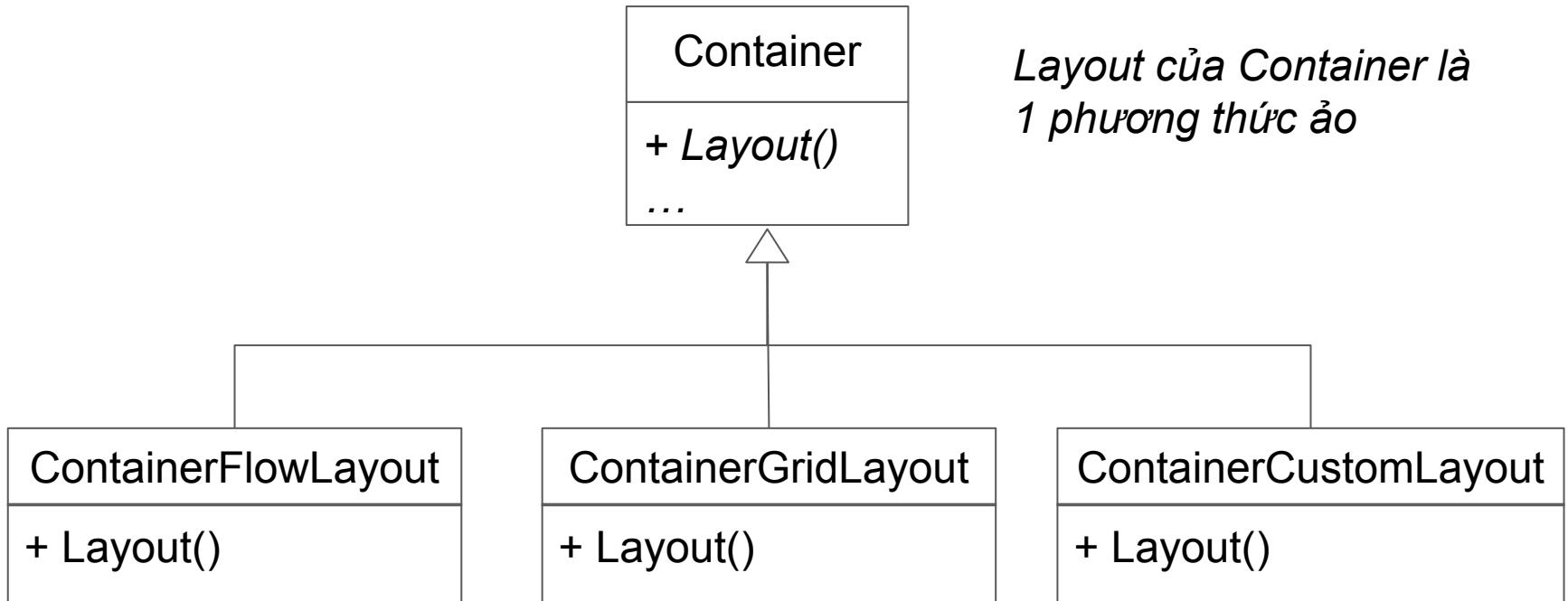
[Bertrand Meyer]

- Mở rộng hành vi của 1 thành phần:
  - Khi yêu cầu ứng dụng thay đổi, chúng ta có thể mở rộng thành phần đã có, thêm vào các hành vi mới.
- Không thay đổi mã nguồn đã có:
  - Mã nguồn đã có được giữ nguyên trong quá trình bổ xung hành vi mới.
  - Đem lại nhiều lợi ích trong quá trình phát triển.

# Trừu tượng hóa là chìa khóa

- Để mở rộng một thành phần mà không thay đổi nó, thì phần thay đổi trong triển khai của thành phần đó phải được trừu tượng hóa.
- Lập trình hướng đối tượng: Lớp ảo, giao diện, phương thức ảo, kế thừa, v.v..
- Lập trình hàm: Kiểu khái quát, hàm khái quát, v.v..
- (*Khả năng tránh thay đổi có tính tương đối: Rất khó tuyệt đối tránh thay đổi mã nguồn / Thường vẫn có những yêu cầu thay đổi dẫn đến thay đổi mã nguồn*).

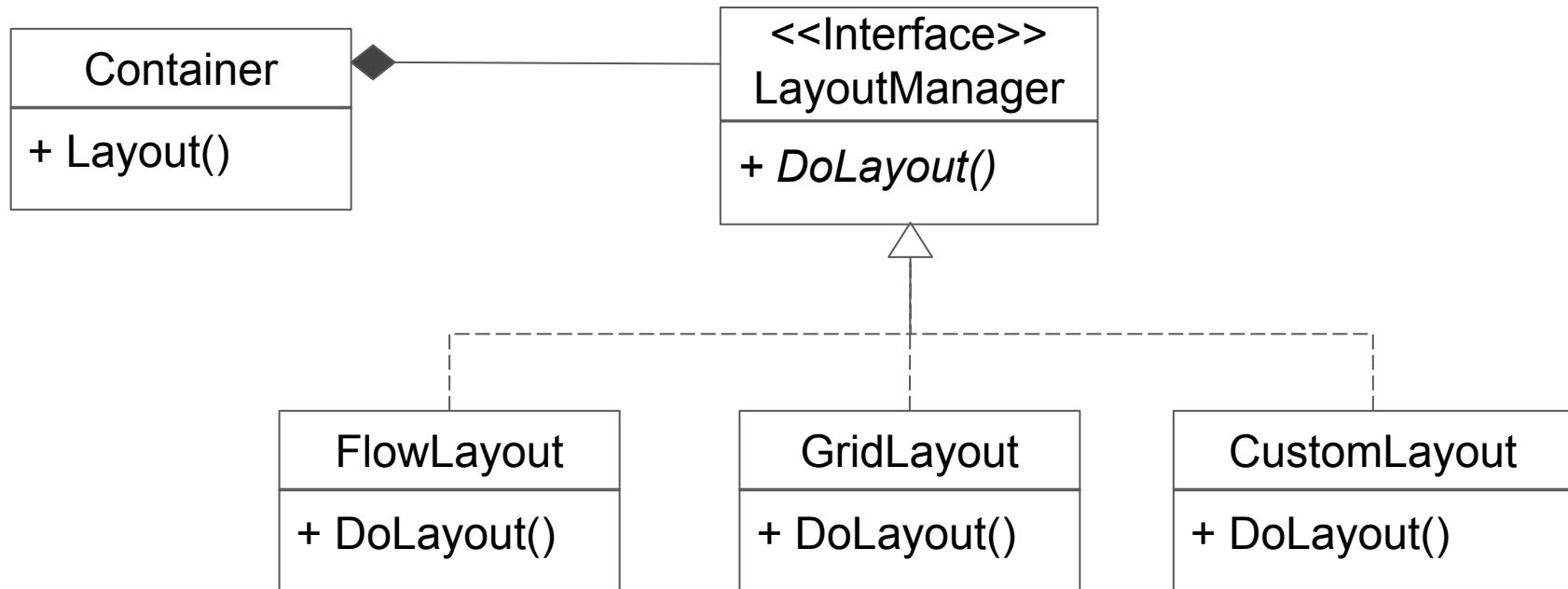
# Khái quát hóa dựa trên kế thừa



*Layout của Container là  
1 phương thức ảo*

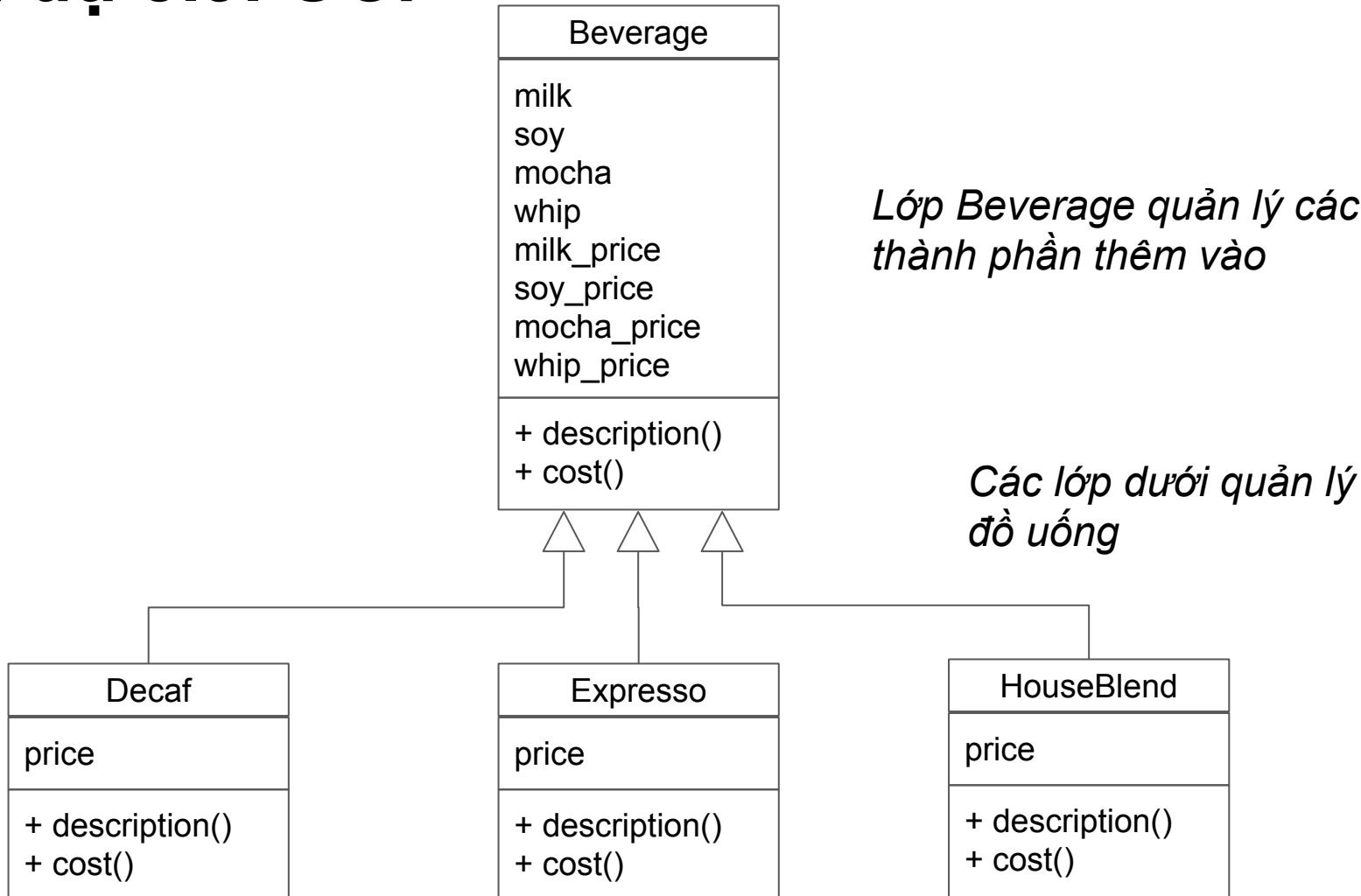
*Lớp con sẽ cung cấp triển khai cụ thể cho Layout*

# Khái quát hóa dựa trên tổng hợp



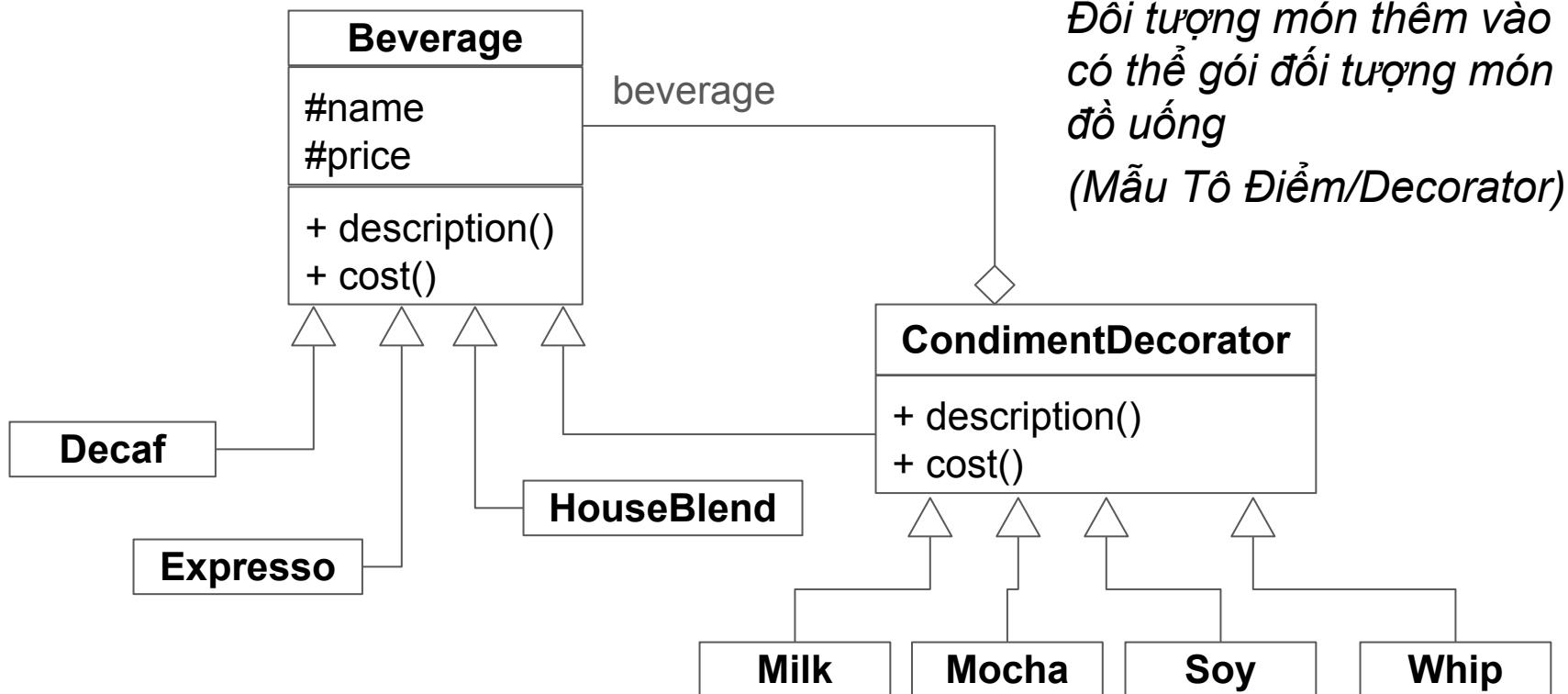
Chúng ta có thể thay đổi hành vi của *Container* bằng cách thiết lập 1 đối tượng quản lý bố cục phù hợp (kế thừa *LayoutManager*)

# Ví dụ 5.9. OCP



*Thiết kế này không tương thích OCP, để thay đổi giá của thành phần thêm vào hoặc bổ sung món mới chúng ta phải sửa lớp Beverage.*

# Ví dụ 5.9. OCP (2)



Đối tượng món thêm vào  
có thể gói đối tượng món  
đồ uống  
(Mẫu Tô Điểm/Decorator)

- Gói đối tượng Beverage trong đối tượng CondimentDecorator.
- name và price được tính theo cấu trúc đóng gói.

*Thiết kế này đáp ứng được OCP.*

# Nguyên lý khả thay Liskov

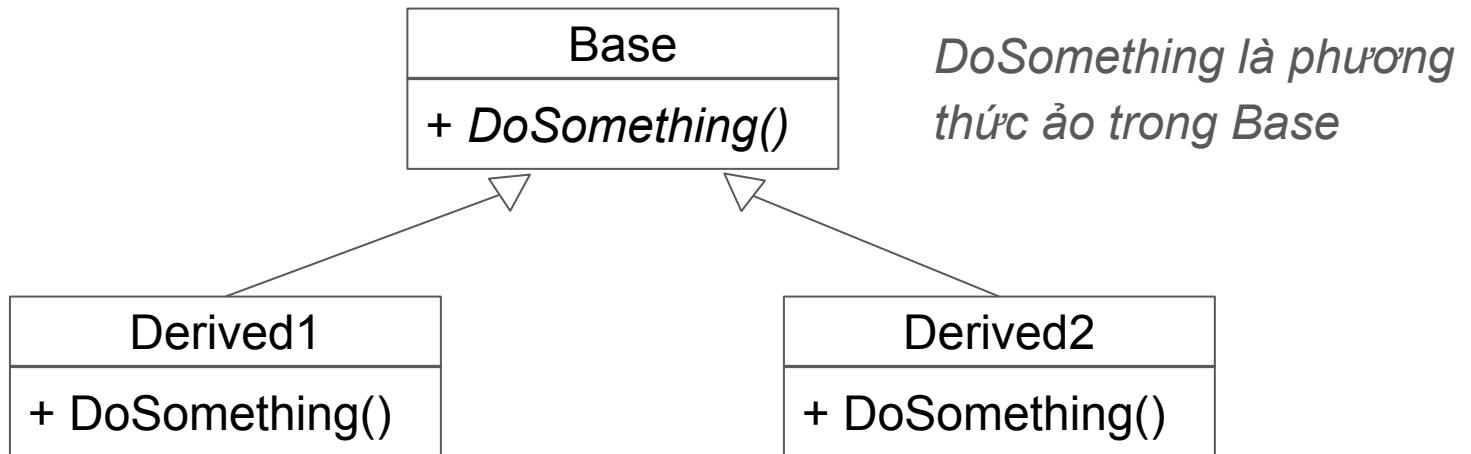
- Nguyên lý khả thay Liskov / Liskov Substitution Principle (LSP)

*Các lớp con phải có khả năng thay thế các lớp cơ sở về mặt hành vi/*

*Subtypes must be behaviorally substitutable for their base types. [Barbara Liskov, 1988]*

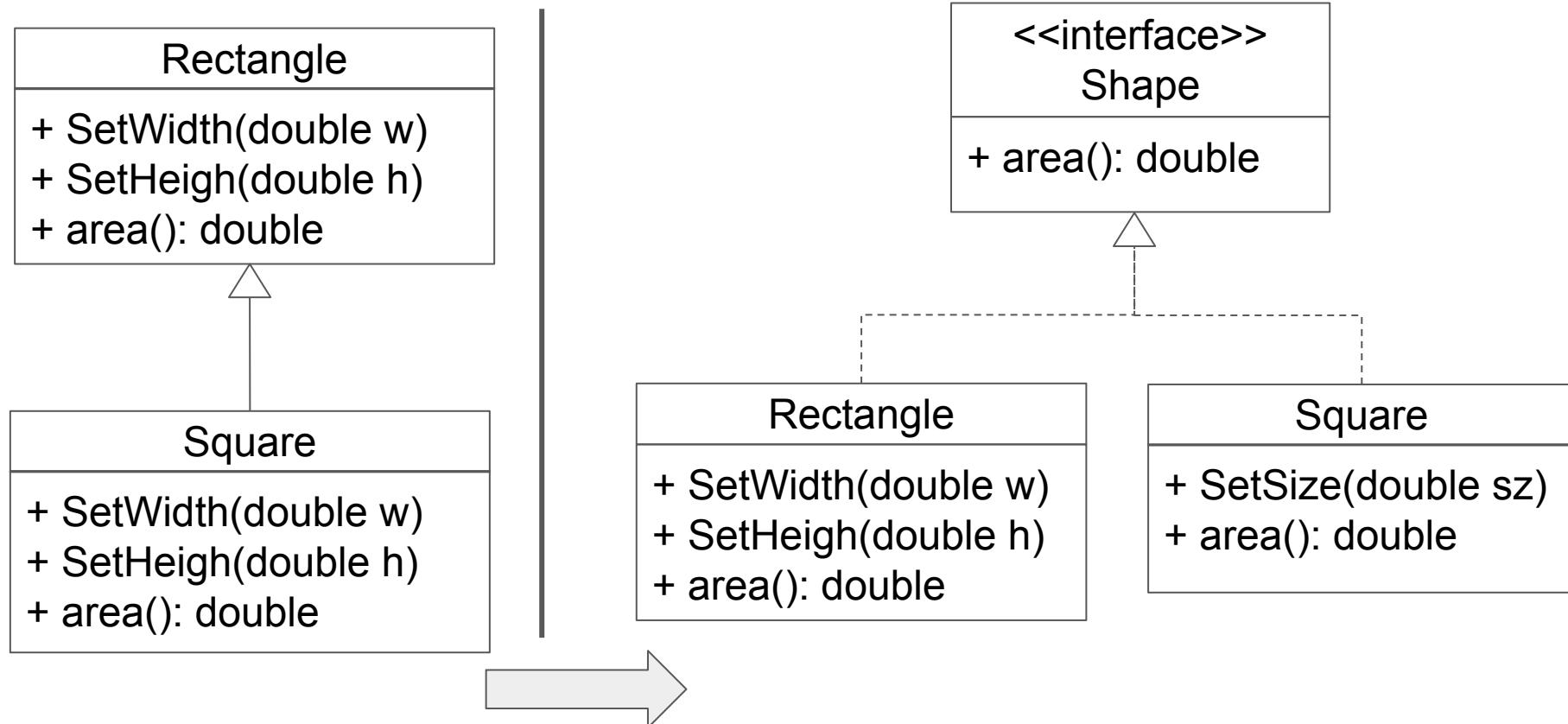
- Như dấu hiệu chất lượng thiết kế kế thừa.
- Hạn chế các vấn đề có thể phát sinh khi lập trình với các giao diện khái quát.

# Khả thay hành vi



- Ngoài định nghĩa lại phương thức DoSomething() trong Derived1 và Derived2, LSP còn ràng buộc về hành vi đối với các đối tượng thuộc các lớp Derived1 và Derived2:
  - Có thể diễn đạt một cách gần đúng là bảo toàn ý nghĩa của lớp cơ sở.

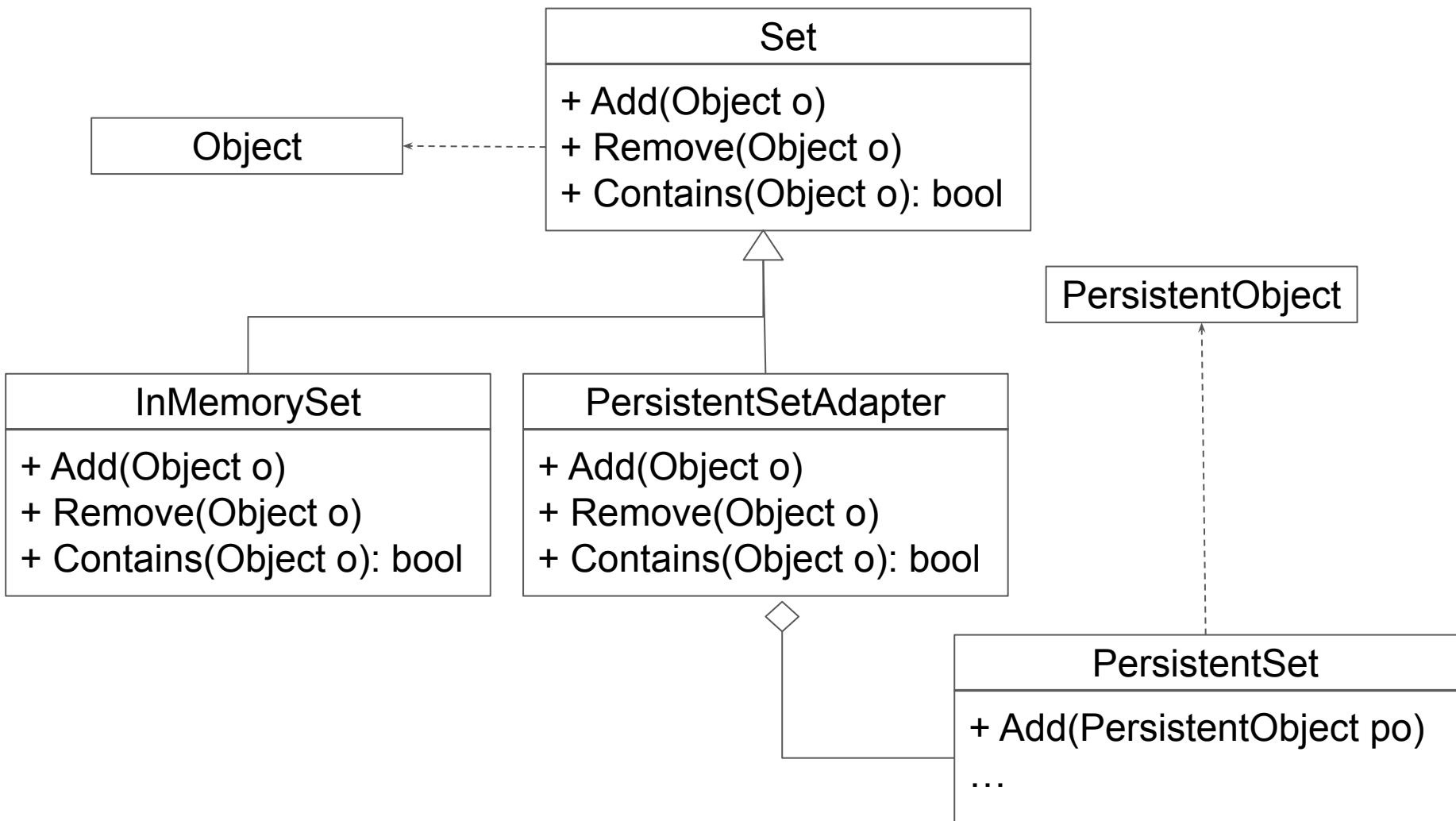
# Ví dụ 5.10. LSP



*Không tương thích LSP, SetHeigh và SetWidth có ý nghĩa khác nhau trong Rectangle và Square.*

*Đáp ứng được LSP*

# Ví dụ 5.10. LSP<sub>(2)</sub>



*Đánh giá thiết kế và đề xuất phương án nếu cần?*

# Nguyên lý tách bạch giao diện

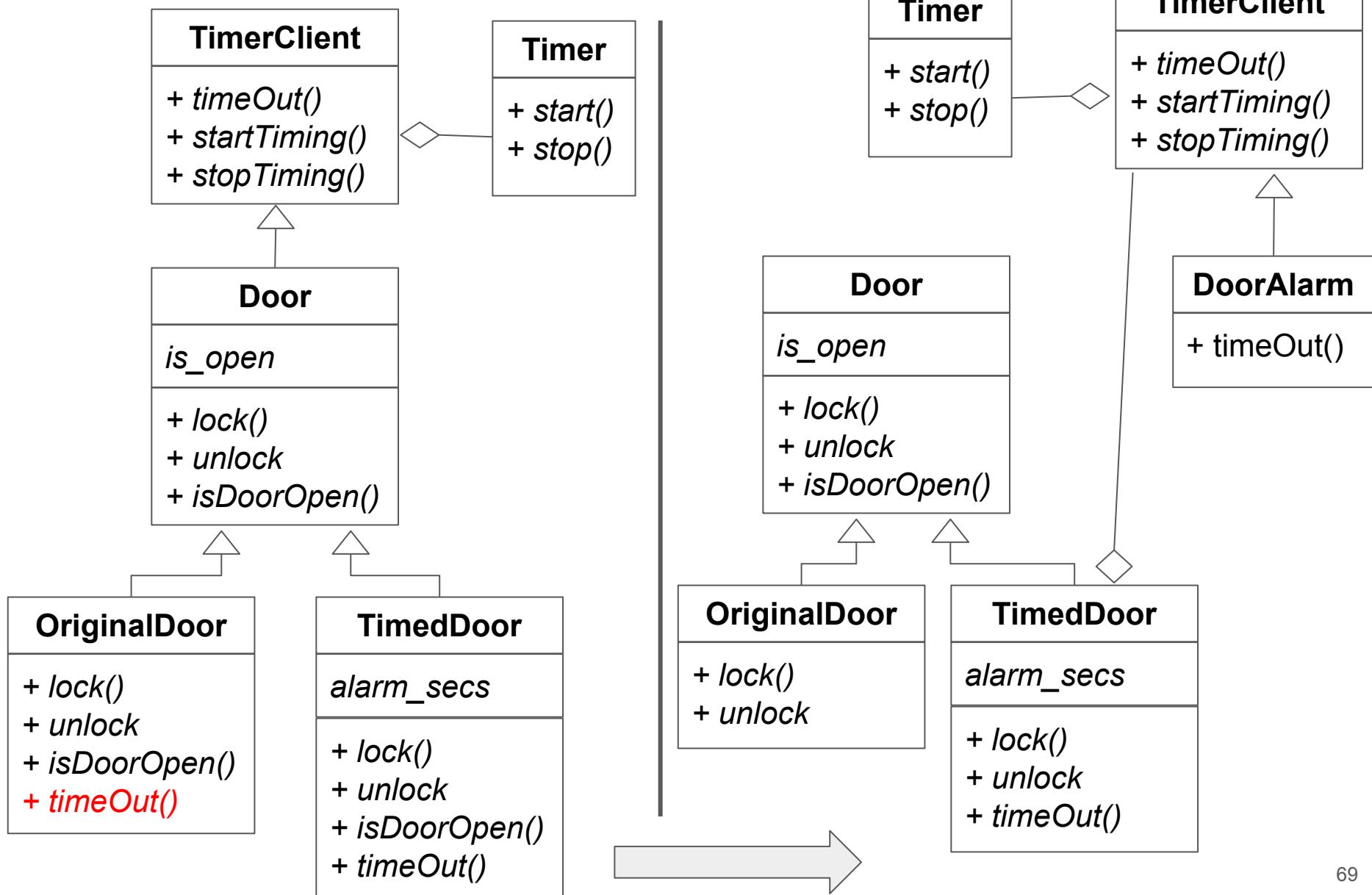
- Nguyên lý tách bạch giao diện / Interface Segregation Principle (ISP)

*Không ép phía khách phụ thuộc vào các phương thức không được sử dụng trong 1 giao diện /  
Clients should not be forced to depend on methods that they do not use*

[Robert C. Martin]

- Các giao diện chứa phương thức không liên quan ép các thành phần sử dụng giao diện đó chịu các thay đổi đúng ra không ảnh hưởng tới chúng.
- Các giao diện pha trộn cần được phân tách
- Nhưng cũng cần tránh làm nát vụn các giao diện

# Ví dụ 5.11. ISP



# Nguyên lý đảo ngược phụ thuộc

- Nguyên lý đảo ngược phụ thuộc / Dependency Inversion Principle (DIP)

Các mô-đun bậc cao không nên phụ thuộc vào các mô-đun bậc thấp. Cả 2 nên phụ thuộc vào các thành phần trừu tượng.

Các thành phần trừu tượng không nên phụ thuộc vào các thành phần cụ thể. Các thành phần cụ thể nên phụ thuộc vào các thành phần trừu tượng. /

*High-level modules should not depend on low-level modules. Both should depend on abstractions.*

*Abstractions should not depend upon details. Details should depend upon abstractions.*

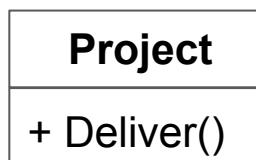
[Robert C. Martin]

# Nguyên lý đảo ngược phụ thuộc<sub>(2)</sub>

- Phần mềm với thiết kế tốt thường được chia nhỏ thành các mô-đun
  - Mô-đun bậc thấp cung cấp các dịch vụ để triển khai mô-đun bậc cao.
- Phụ thuộc vào các cơ chế trừu tượng giúp nâng cao khả năng tái sử dụng các thành phần.
- Áp dụng DIP giúp nới lỏng các phụ thuộc.

# Ví dụ 5.12. DIP

Bậc cao

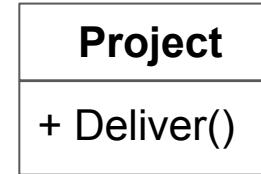


Bậc thấp



Thiết kế này không tương thích với DIP do Project phụ thuộc trực tiếp vào FrontEndDeveloper và BackEndDeveloper.

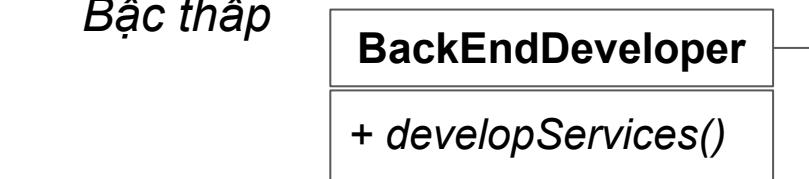
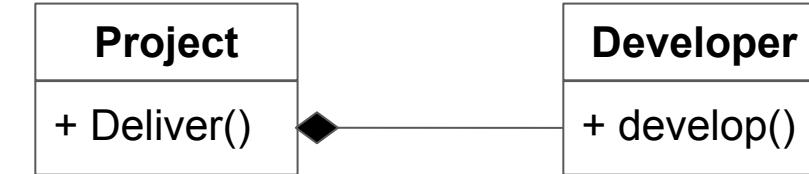
Bậc cao



Bậc thấp



Trừu tượng hóa



Triển khai cụ thể phụ thuộc vào cơ chế trừu tượng.

Đáp ứng được DIP, chiều phụ thuộc vào mô-đun bậc thấp đã được đảo ngược: Các mô-đun bậc thấp phụ thuộc vào giao diện khái quát được dùng trong mô-đun bậc cao.

# Chèn phụ thuộc

- Chèn phụ thuộc / Dependency Injection (DI)
- Phương pháp nghịch đảo phụ thuộc cụ thể, có thể được sử dụng để nghịch đảo phụ thuộc.
- Đối tượng thích hợp được tạo và đưa vào từ bên ngoài.
- Đối tượng phụ thuộc có thể được đưa vào thông qua:
  - Tham số cho hàm tạo (được sử dụng cho những phụ thuộc bắt buộc)
    - Như trong mẫu thiết kế Tô điểm / Decorator
  - Tham số cho phương thức thiết lập phụ thuộc (được sử dụng cho các phụ thuộc không bắt buộc)
    - Như trong mẫu thiết kế Trạng thái / State



# Phân tích và thiết kế Hệ thống

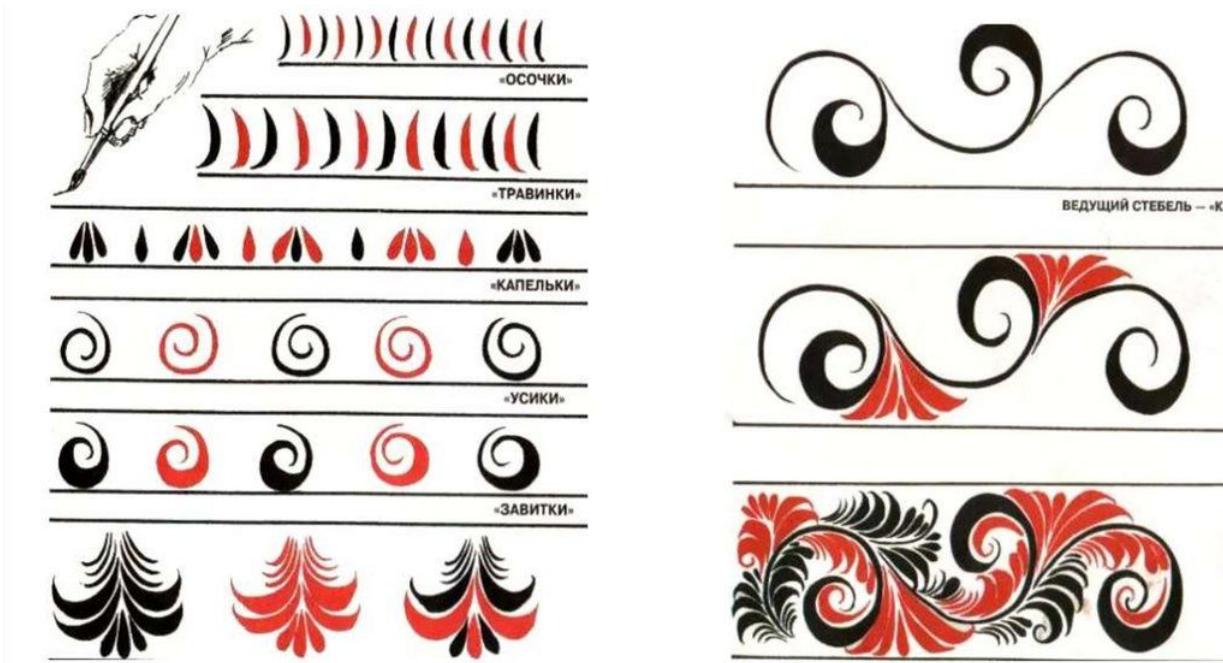
Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2025

# Mẫu thiết kế

# Chuyển ngữ

*Pattern => Mẫu hay Họa tiết? hay ...*



[ảnh sưu tầm]

Hoa văn, họa tiết là những từ trong hội họa để chỉ những thành phần được kết hợp với nhau theo quy luật để tạo nên bức tranh. *Tuy nhiên trong phân tích và thiết kế hệ thống từ mẫu được sử dụng phổ biến hơn.*

# Khái niệm

Mẫu thiết kế là giải pháp thiết kế khái quát cho vấn đề thiết kế được chấp nhận rộng rãi. Mỗi mẫu thiết kế cung cấp một giải pháp khái quát cho một vấn đề thiết kế phổ biến. Khái niệm mẫu hàm chứa nguyên lý tương tự, có thể giải quyết các vấn đề tương tự theo cách tương tự.

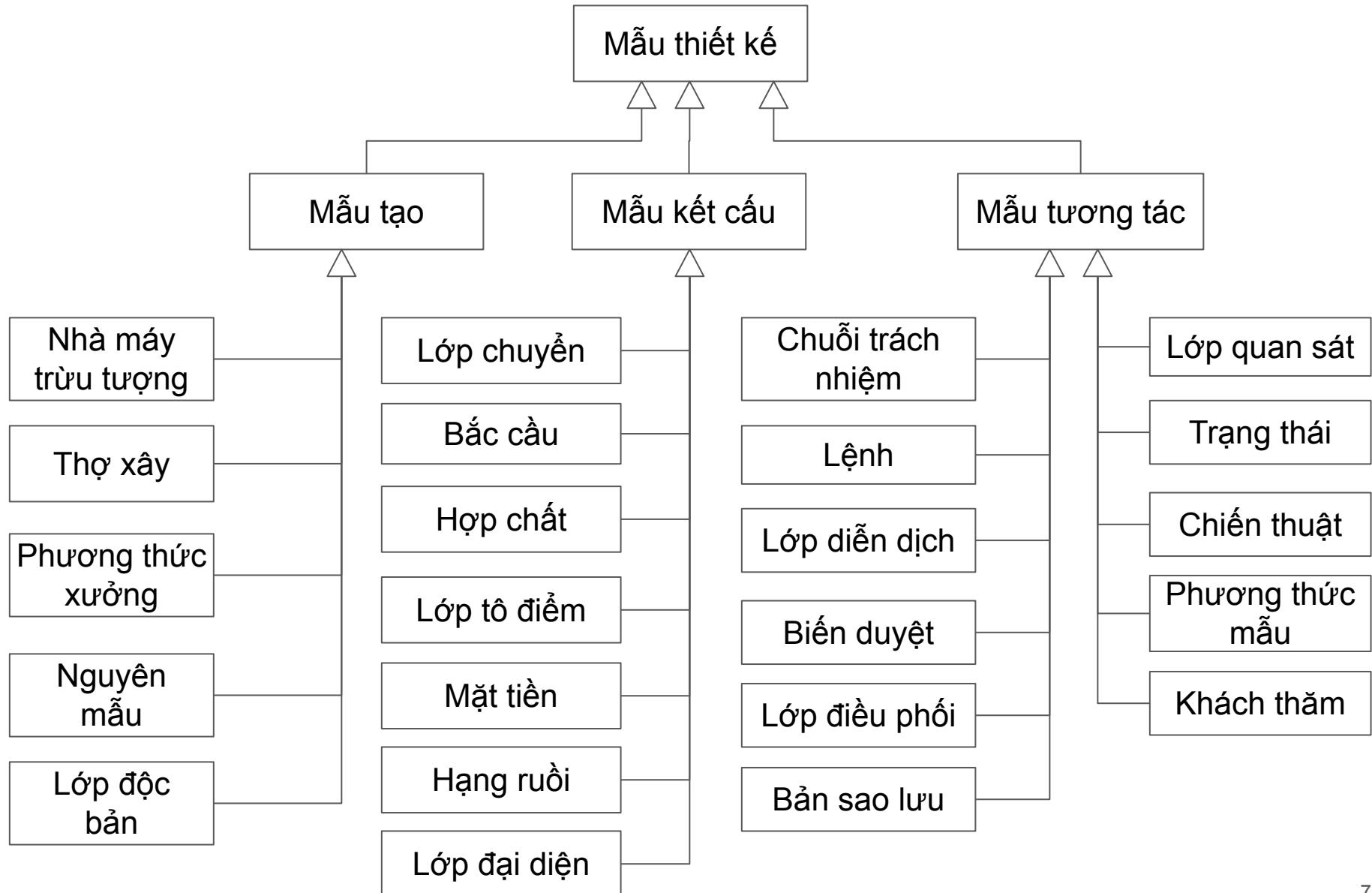
# Các thành phần của mẫu thiết kế

- **Tên** - Ngắn gọn, gợi nghĩa, làm giàu bộ từ vựng.
- **Vấn đề** - Phát biểu vấn đề thiết kế được giải quyết.
- **Giải pháp** - Các mô hình cấu trúc và hành vi.
- **Hệ quả** - Ưu nhược điểm khi áp dụng mẫu thiết kế.

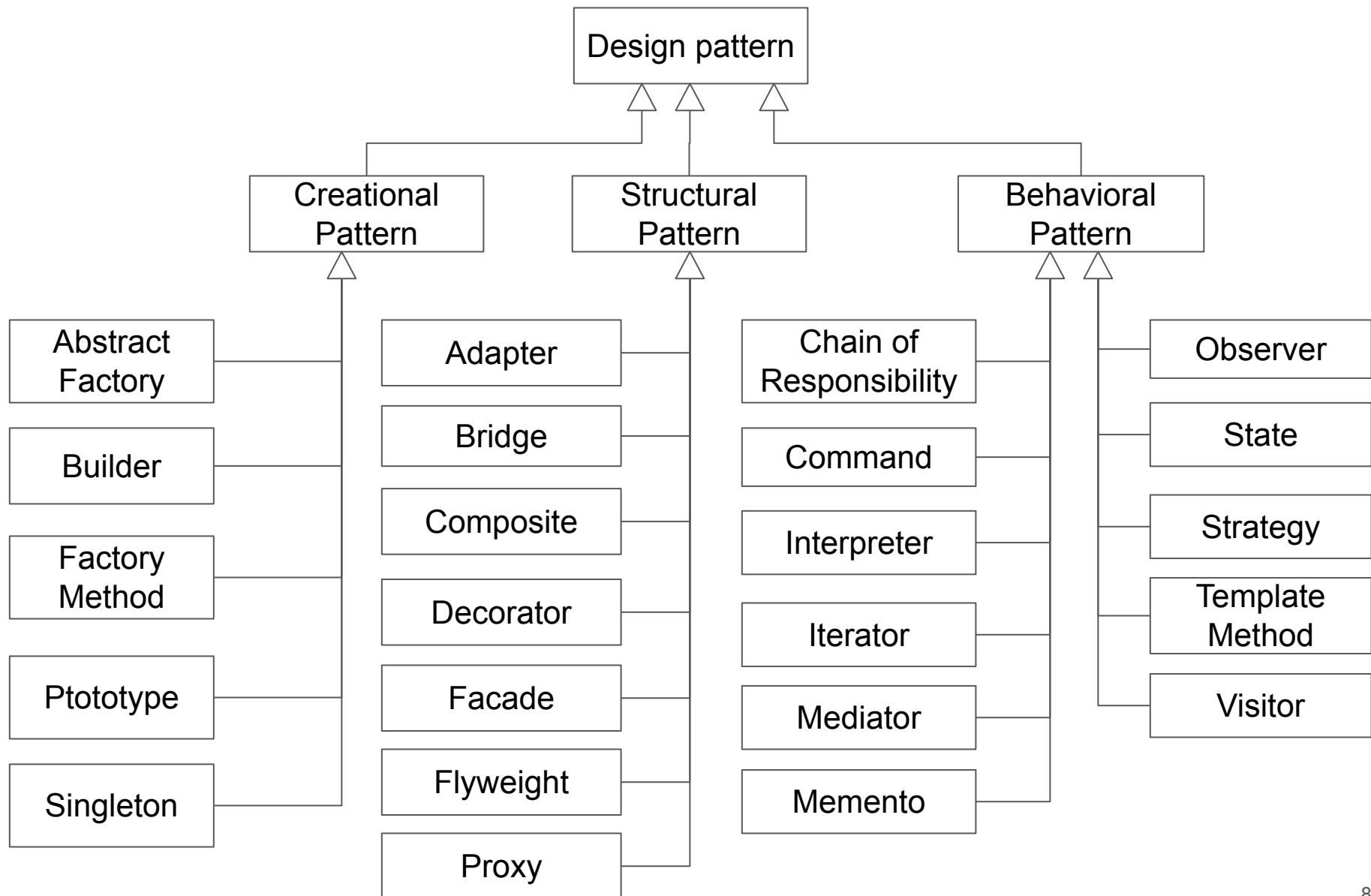
# Phân loại mẫu thiết kế

- **Mẫu tạo (Creational patterns)** – Giải quyết vấn đề tạo đối tượng.
- **Mẫu kết cấu (Structural patterns)** – Giải quyết vấn đề kết hợp đối tượng để tạo cấu trúc lớn hơn.
- **Mẫu tương tác (Behavioral patterns)** – Giải quyết vấn đề tương tác và phân chia trách nhiệm giữa các đối tượng.

## Phân loại mẫu thiết kế



# A classification of Design patterns



# Nội dung

- Mẫu tạo
- Mẫu kết cấu
- Mẫu tương tác

# Nội dung

- Mẫu tạo
- Mẫu kết cấu
- Mẫu tương tác

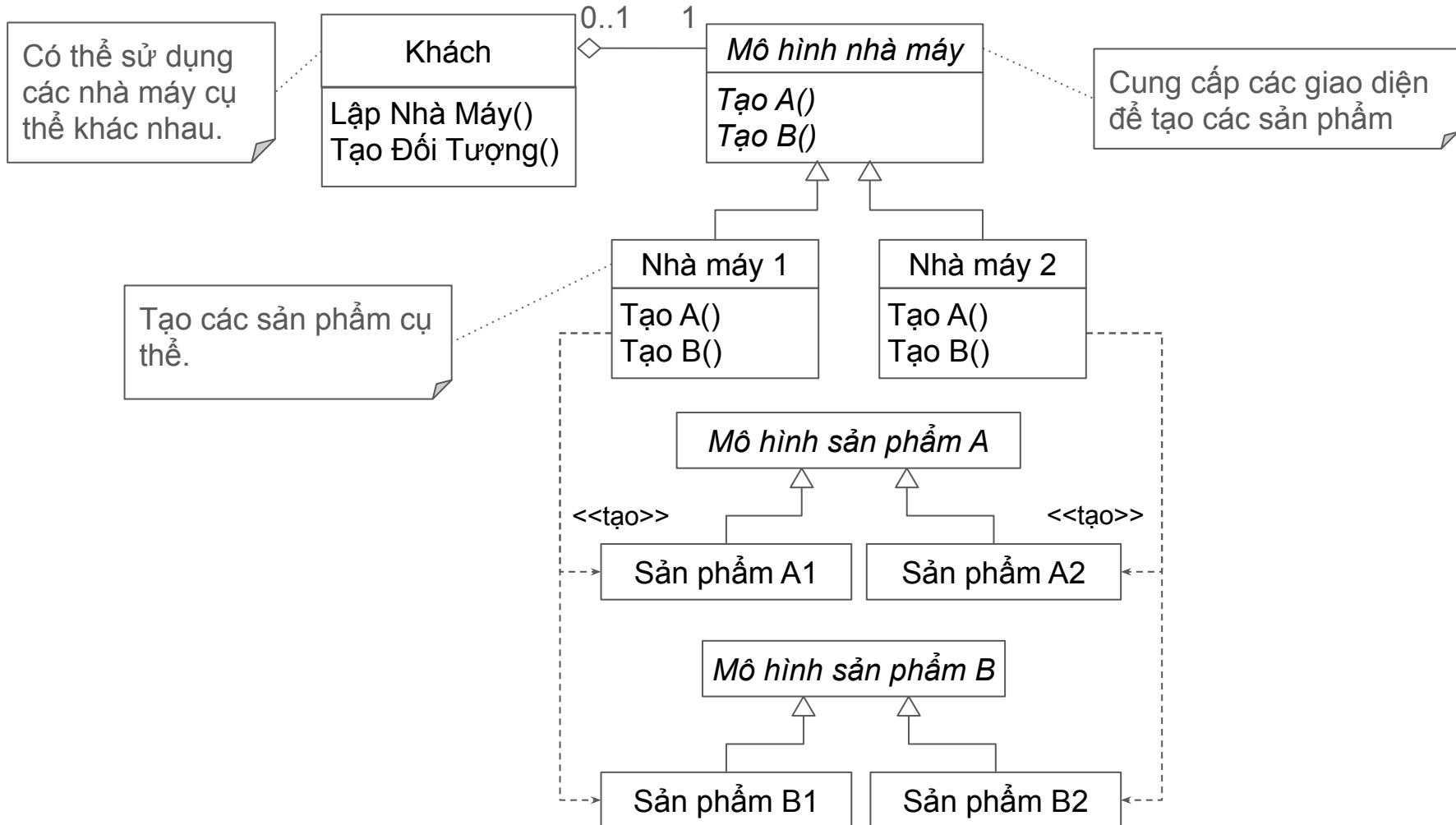


# Nhà máy trừu tượng

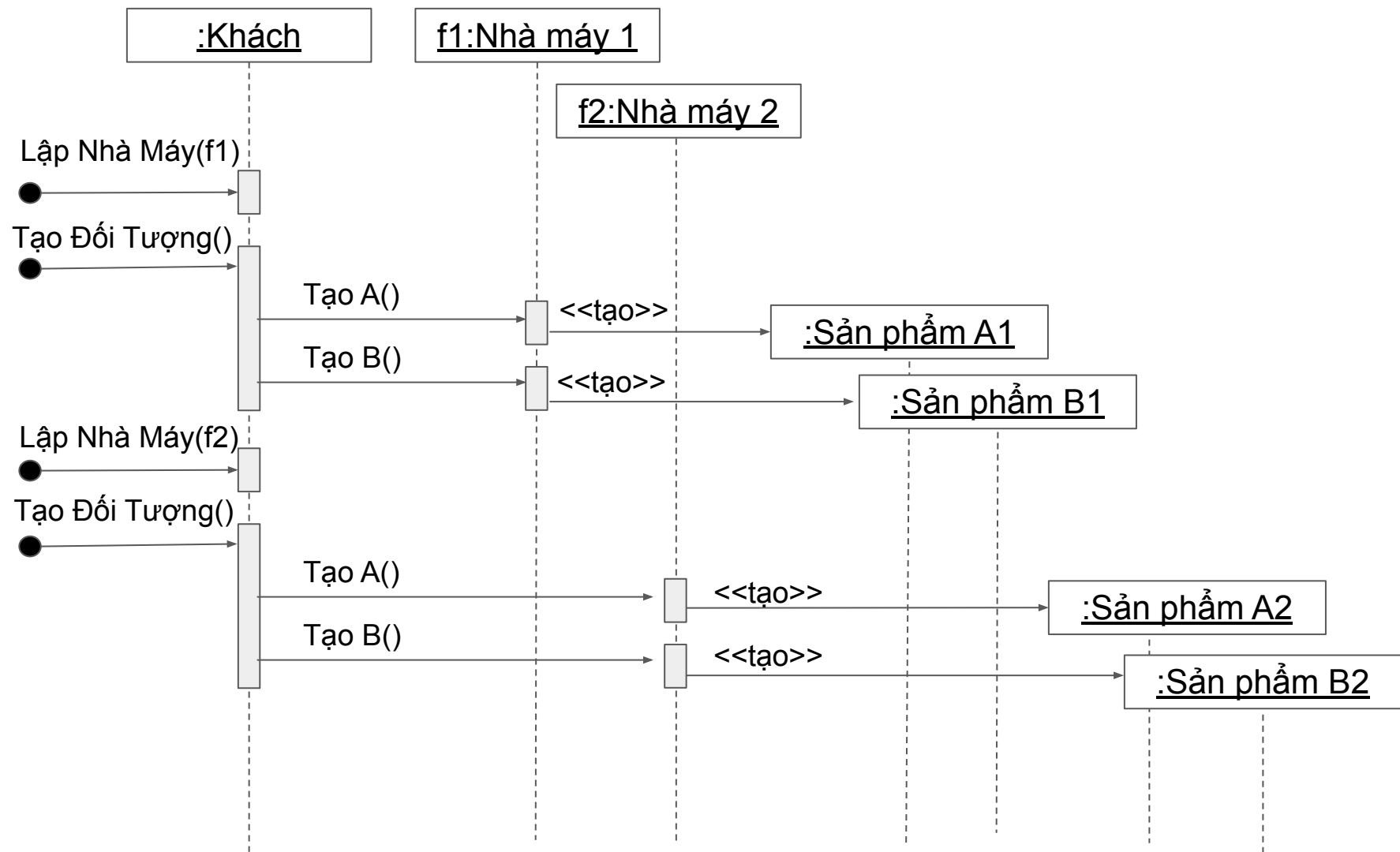
*Abstract Factory*

Cung cấp các giao diện để tạo một tập đối tượng, trong đó các đối tượng có thể thuộc các cây kế thừa khác nhau.

# Nhà máy trừu tượng: Cấu trúc



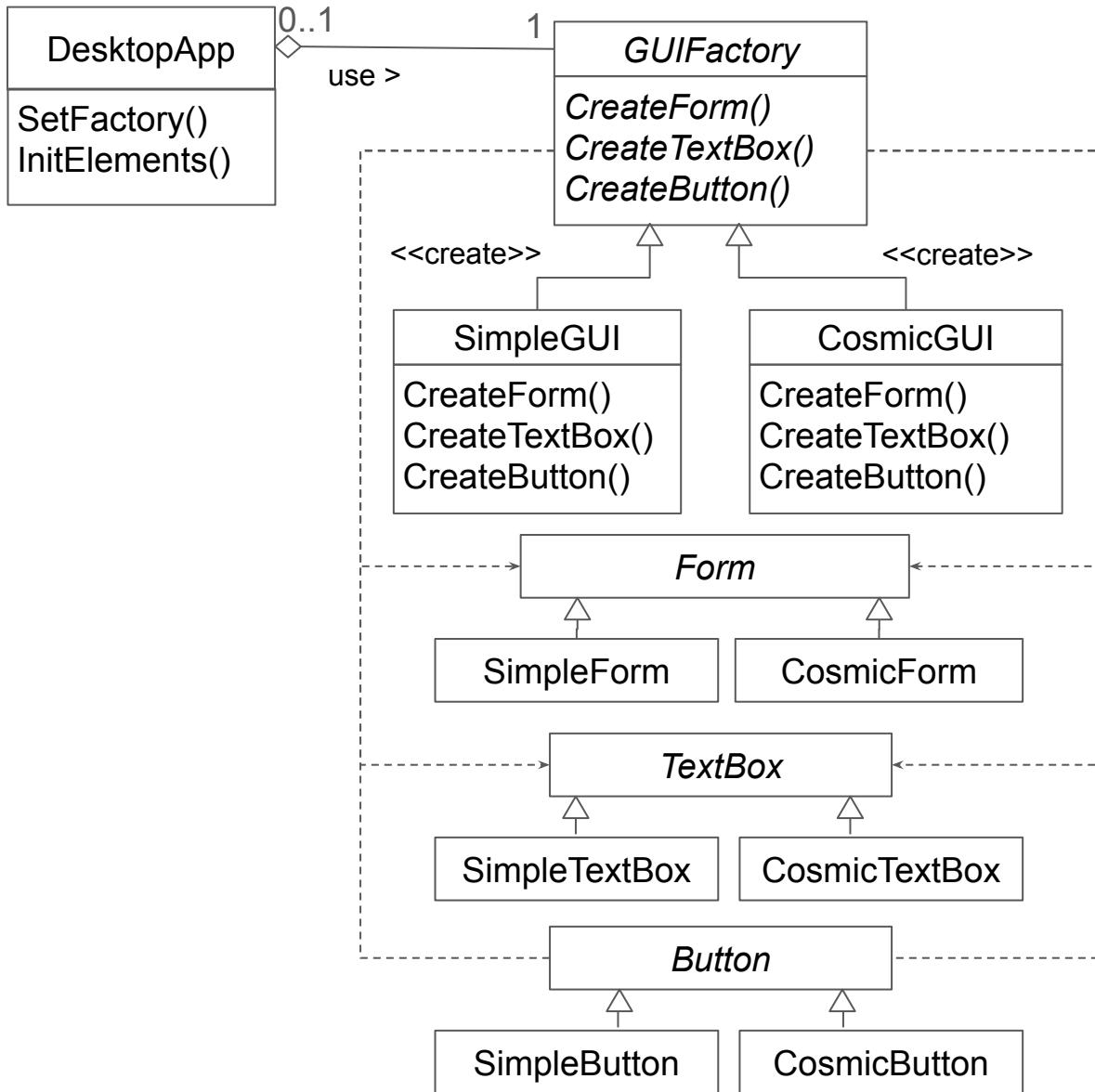
# Nhà máy trừu tượng: Hành vi



# Nhà máy trừu tượng: Hệ quả

- Cố lập các lớp cụ thể khiến chúng trong suốt đối với phía khách. Phía khách chỉ sử dụng các giao diện khái quát.
- Dễ thay thế một ê-kíp đối tượng bằng cách chuyển sang sử dụng một nhà máy cụ thể khác.
- Tăng cường tính nhất quán của ê-kíp đối tượng trong trường hợp chúng ràng buộc lẫn nhau.
- Khó bổ sung đối tượng mới vì phải thay đổi nhà máy trừu tượng và các nhà máy cụ thể đã triển khai.

# Ví dụ 1. Giao diện đồ họa

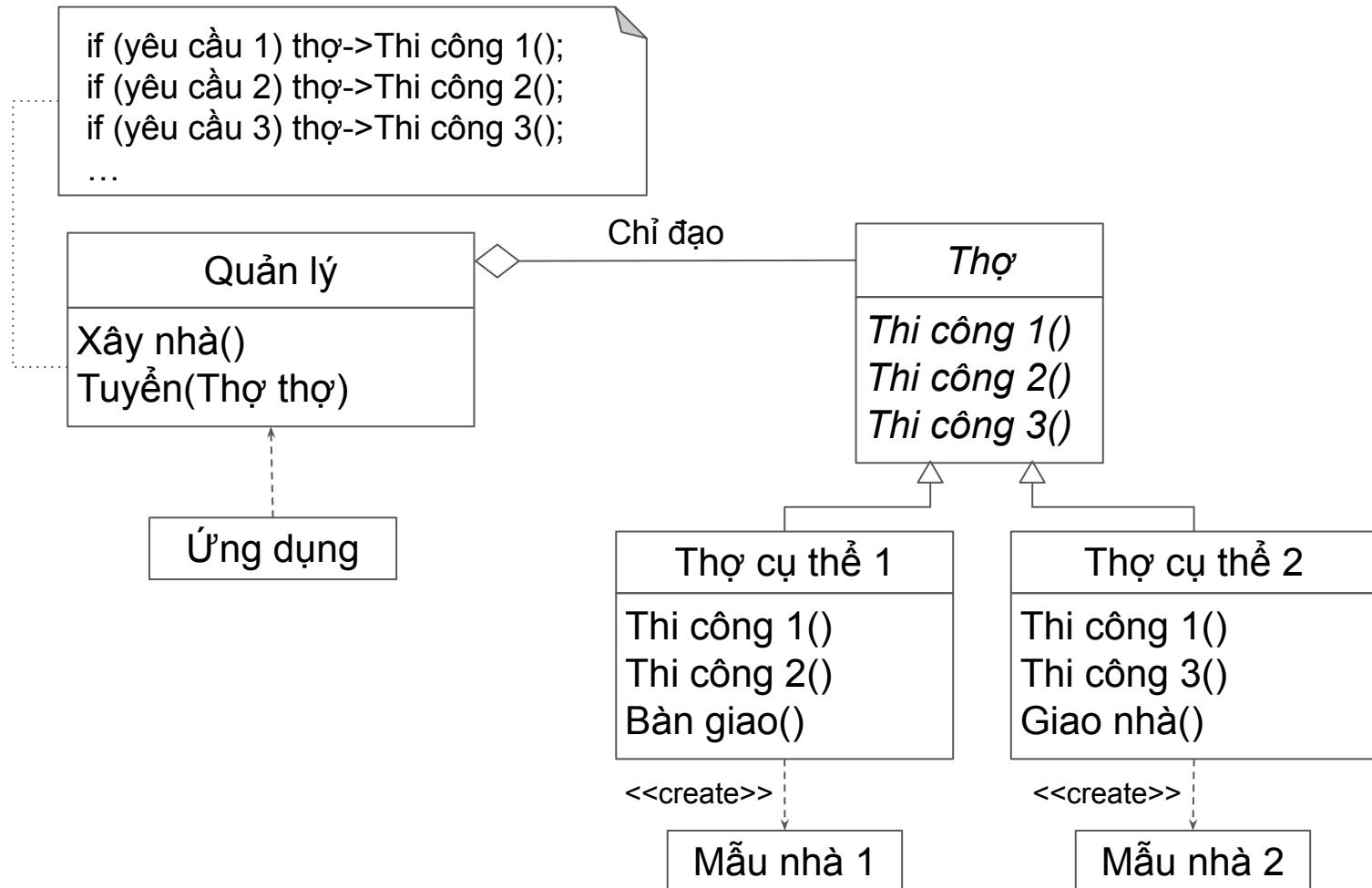


# Thợ xây

*Builder*

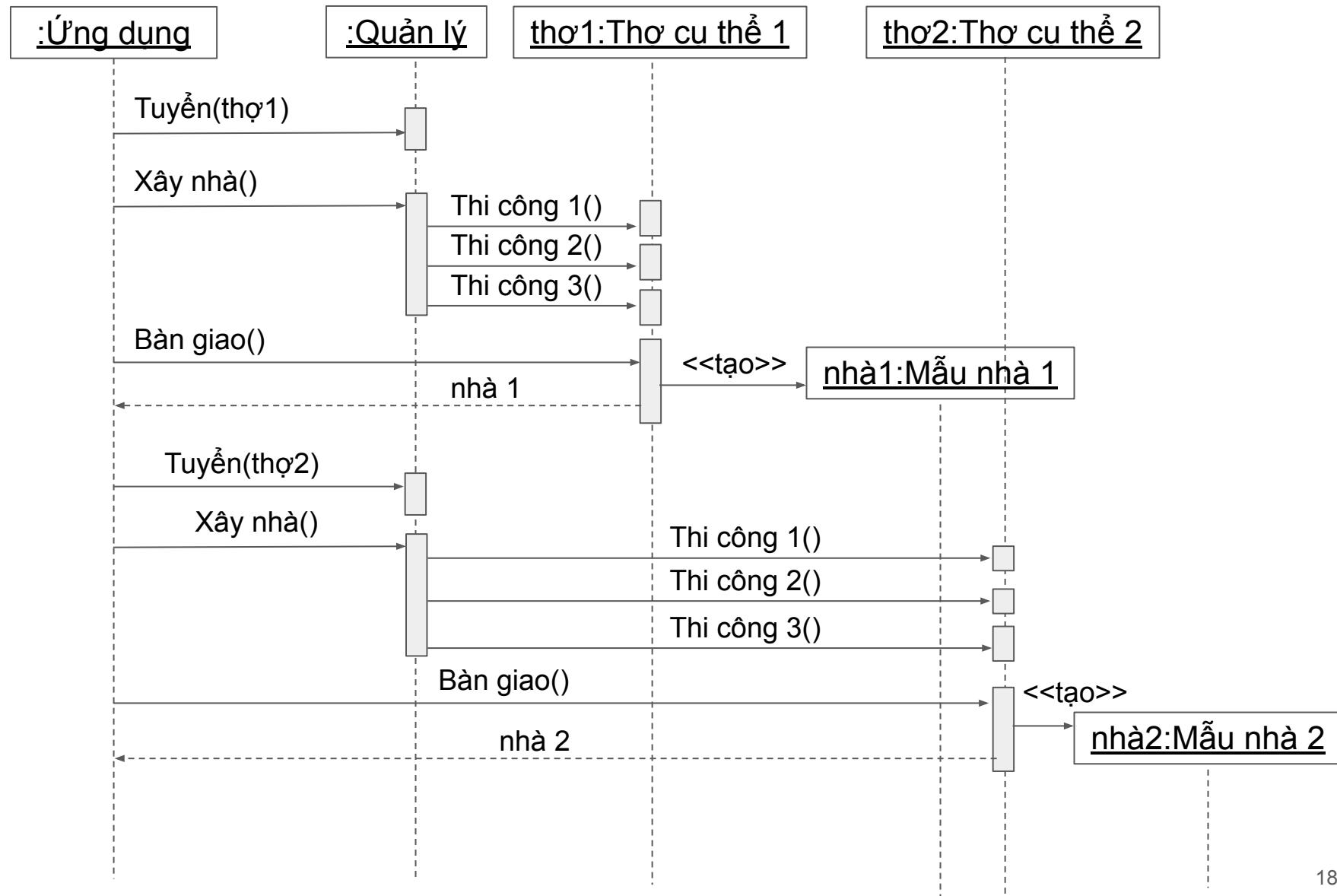
Tách riêng phần khởi tạo đối tượng phức tạp ra ngoài lớp, qua đó có thể sử dụng cùng một tiến trình để tạo nhiều biểu diễn khác nhau.

# Thợ xây: Cấu trúc



*Mẫu nhà 1 và Mẫu nhà 2 có thể thuộc các cây kế thừa khác nhau.*

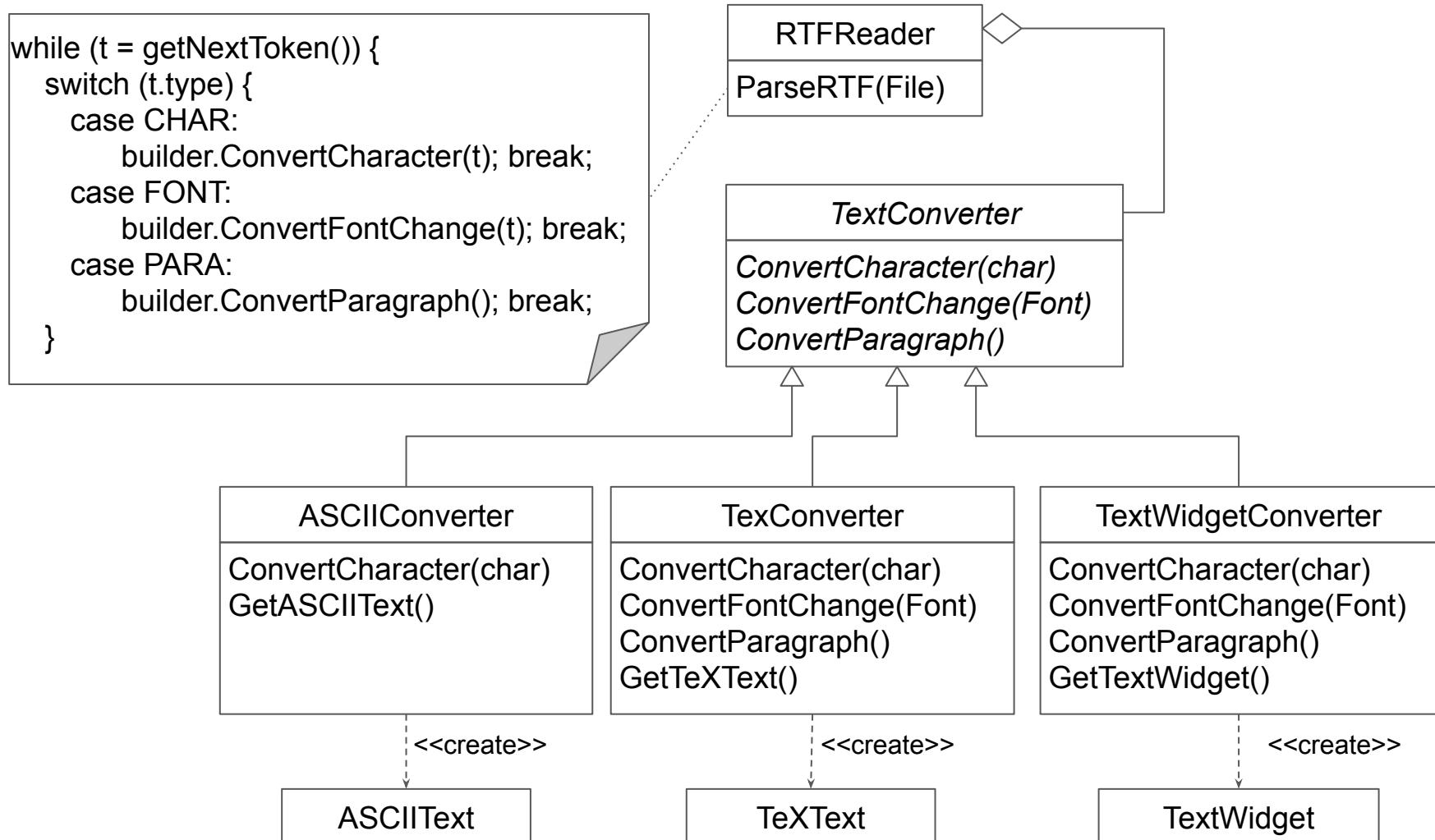
# Thợ xây: Hành vi



# Thợ xây: Hệ quả

- Dễ thay đổi biểu diễn bên trong của sản phẩm, có thể tạo lớp thợ mới nếu cần.
- Tách phần khởi tạo với biểu diễn của đối tượng làm tăng tính độc lập của các thành phần.
- Có thể kiểm soát sâu hơn tiến trình tạo đối tượng cùng với biểu diễn của nó qua các bước, so với các mẫu tạo khác.

# Ví dụ 2. Chuyển đổi định dạng văn bản

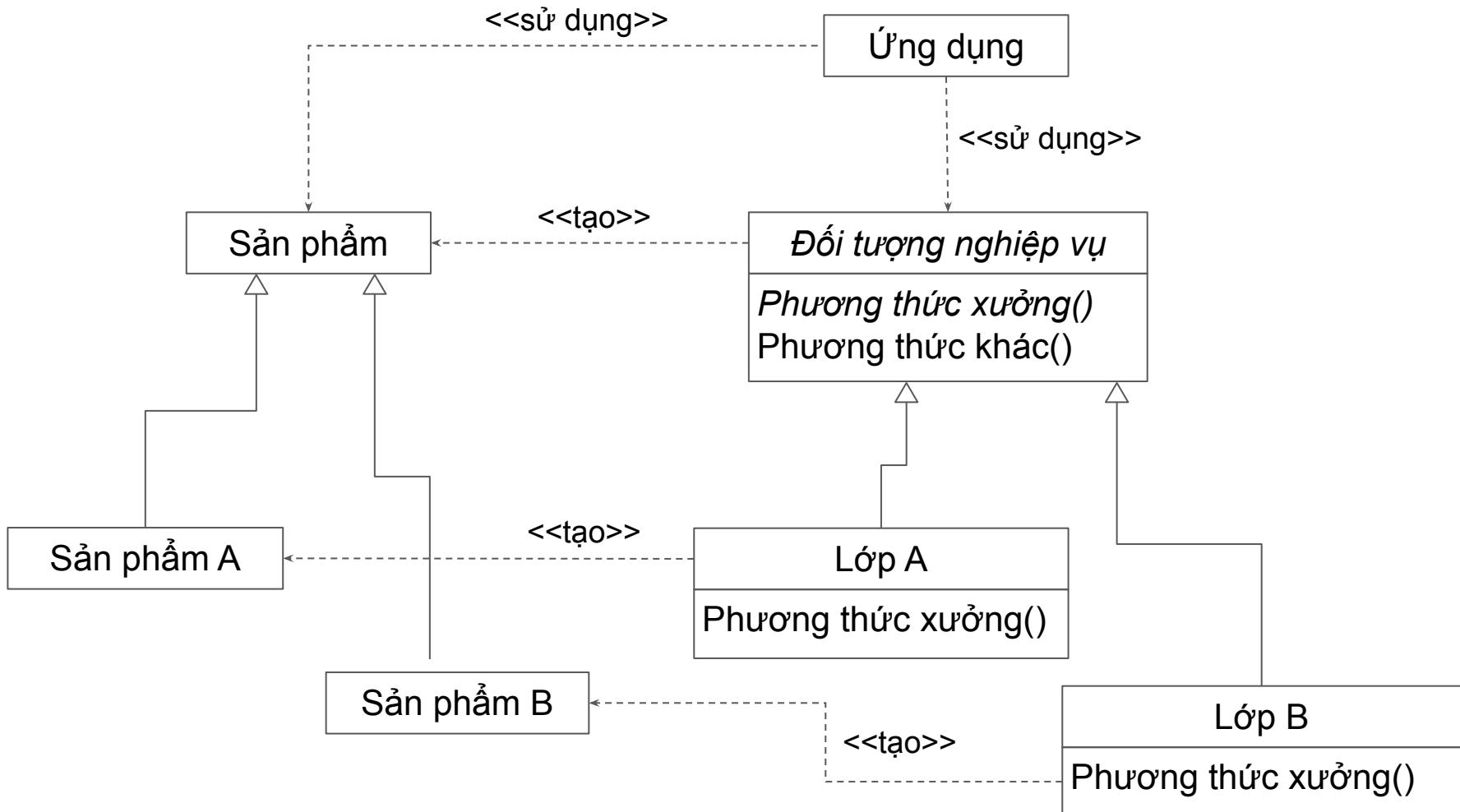


# Phương thức xưởng

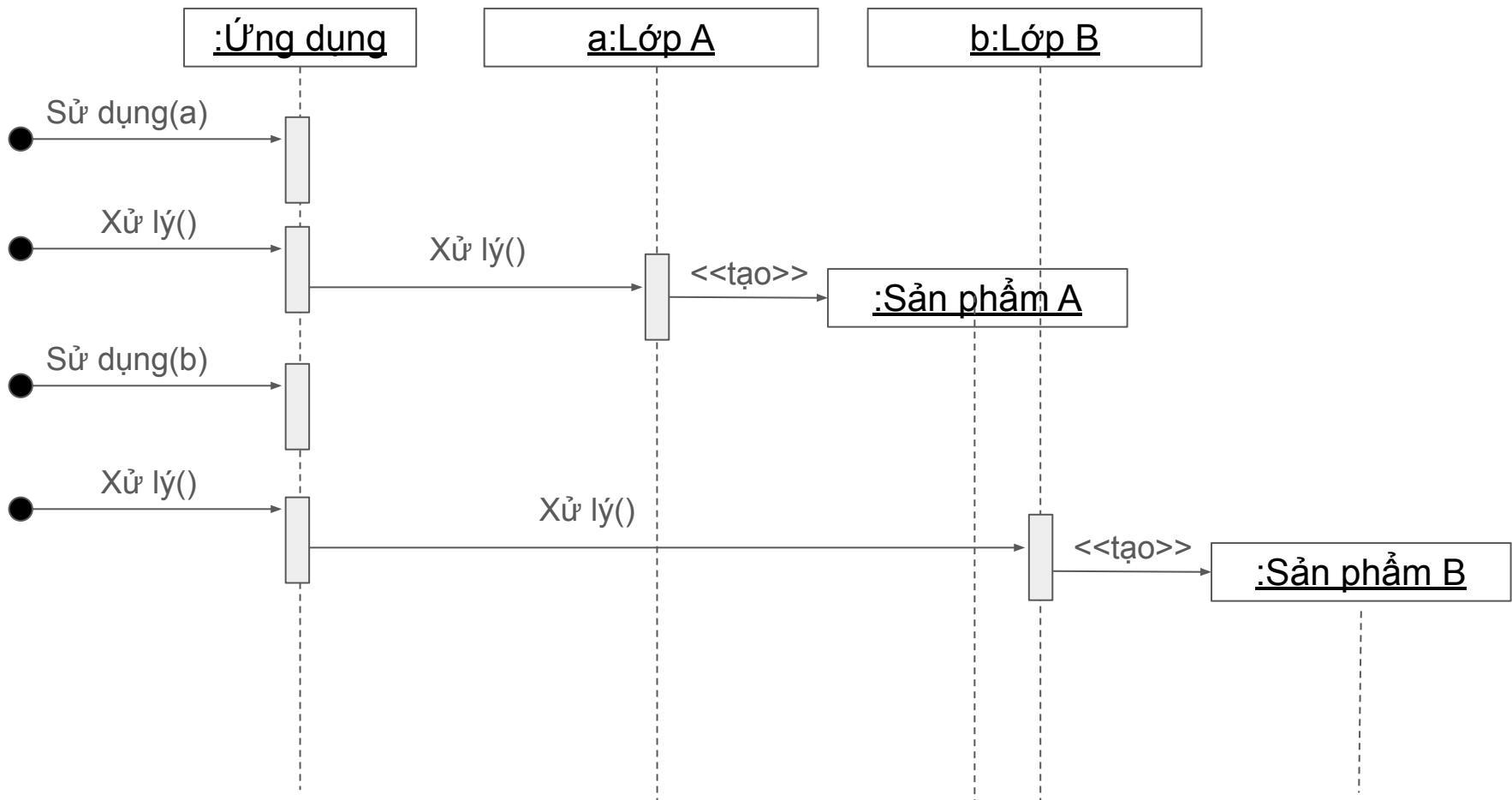
*Factory Method*

Thiết lập giao diện để tạo một đối tượng trong cây kế thừa, nhưng để các lớp dưới quyết định tạo đối tượng của lớp nào. Phương thức xưởng cho phép trì hoãn quyết định khởi tạo tới khi định nghĩa lớp dưới.

# Phương thức xưởng: Cấu trúc



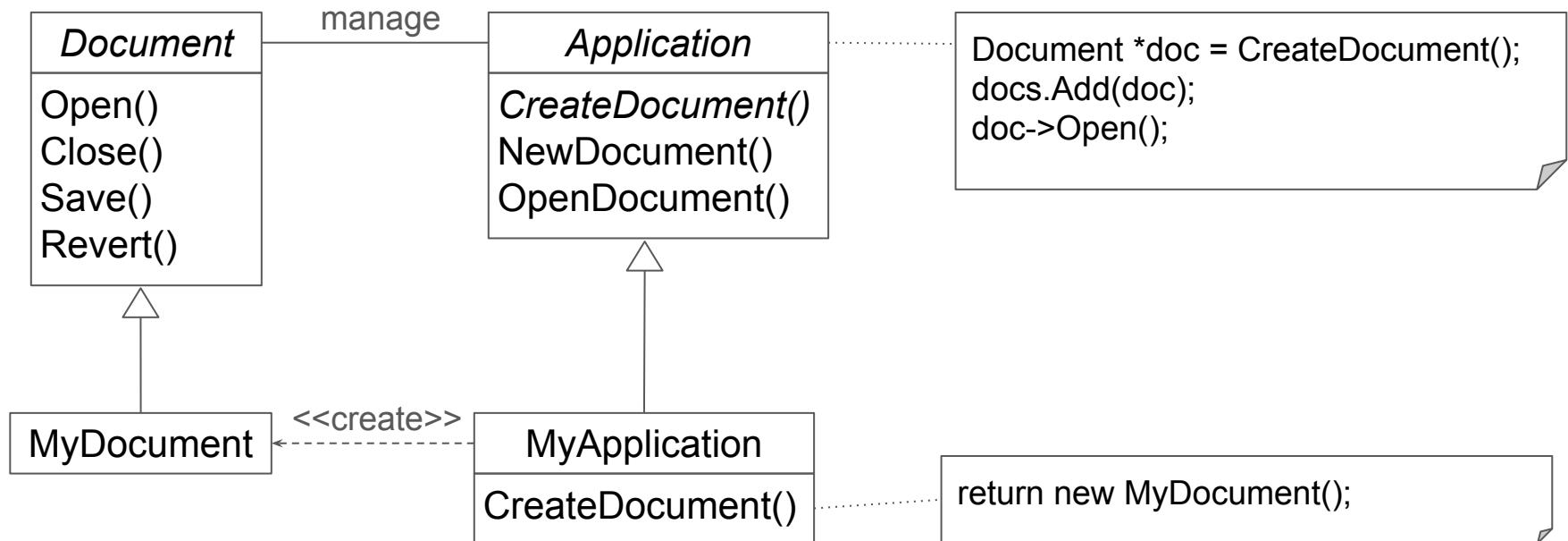
# Phương thức xưởng: Hành vi



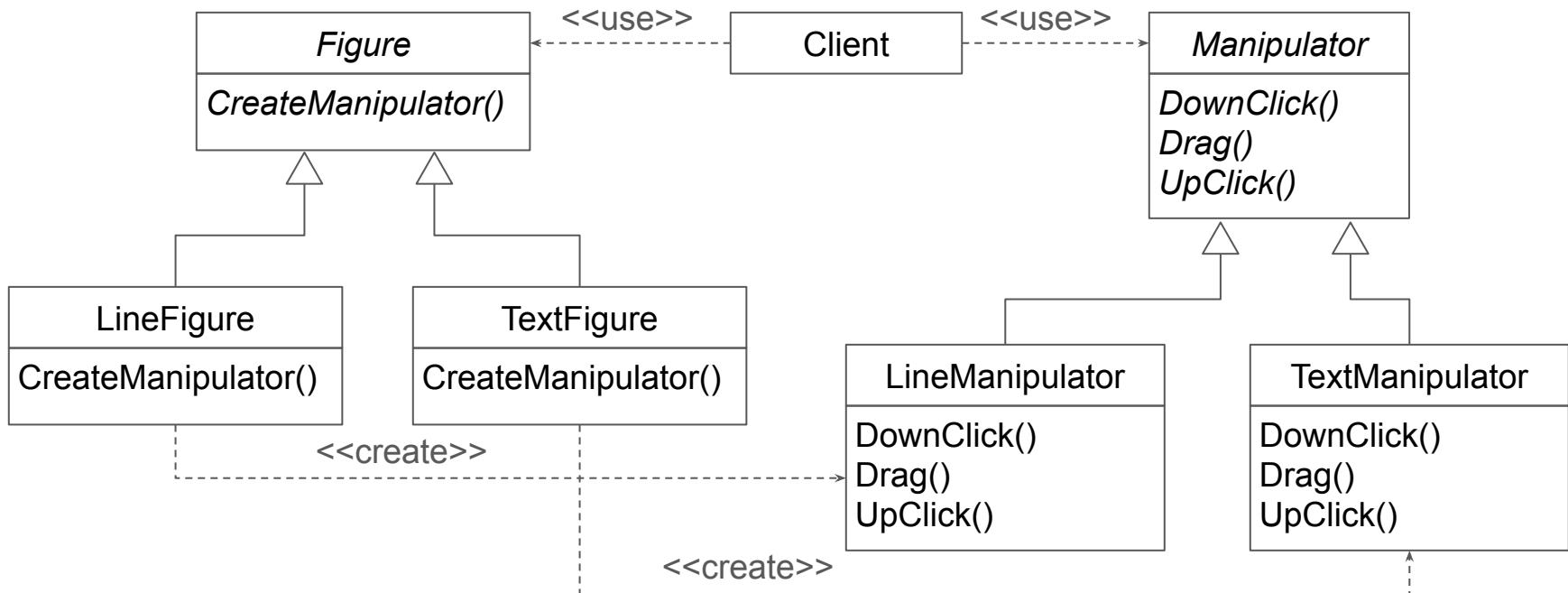
# Phương thức xuống: Hệ quả

- Cung cấp điểm mở rộng cho các lớp dưới, làm tăng tính linh động của việc tạo đối tượng.
- Có thể hình thành các cây kế thừa song hành - thường gặp khi các lớp trong cây tách 1 phần trách nhiệm của chúng thành các lớp riêng.

# Ví dụ 3. Ứng dụng xử lý tài liệu đa định dạng



# Ví dụ 4. Xử lý bản vẽ

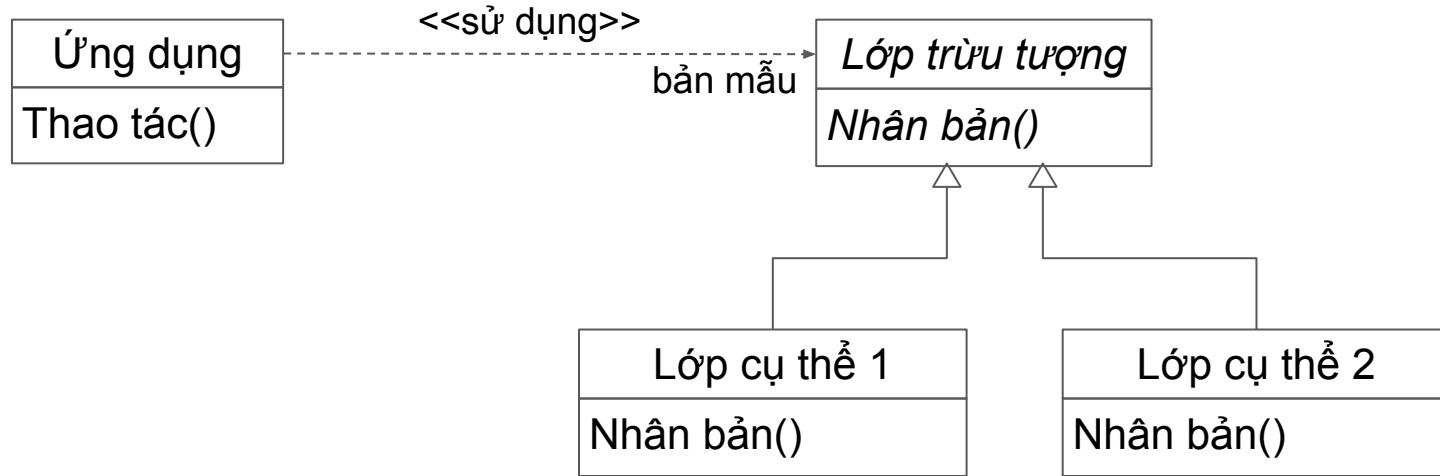


# Nguyên mẫu

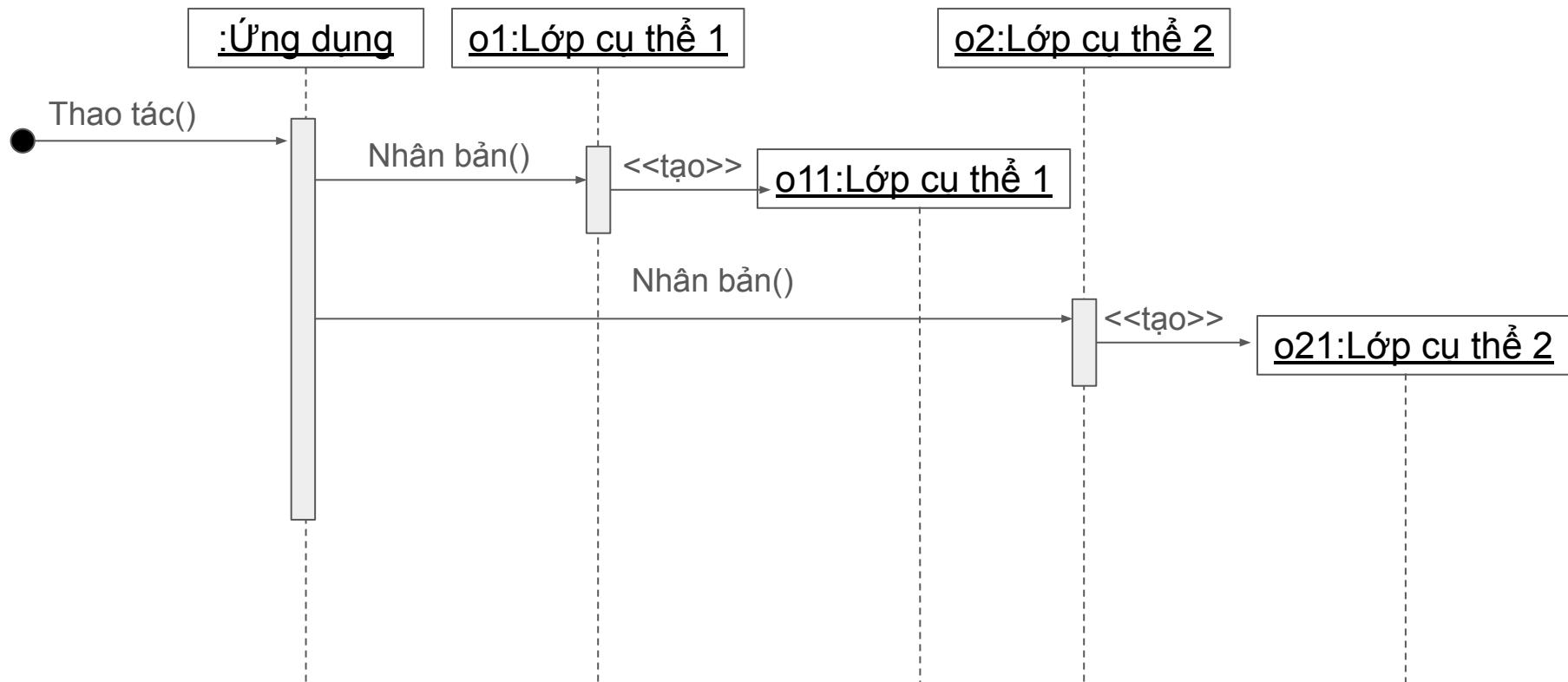
*Prototype*

Tạo một đối tượng làm bản mẫu, rồi tiếp tục tạo các đối tượng mới bằng cách sao chép nó.

# Nguyên mẫu: Cấu trúc



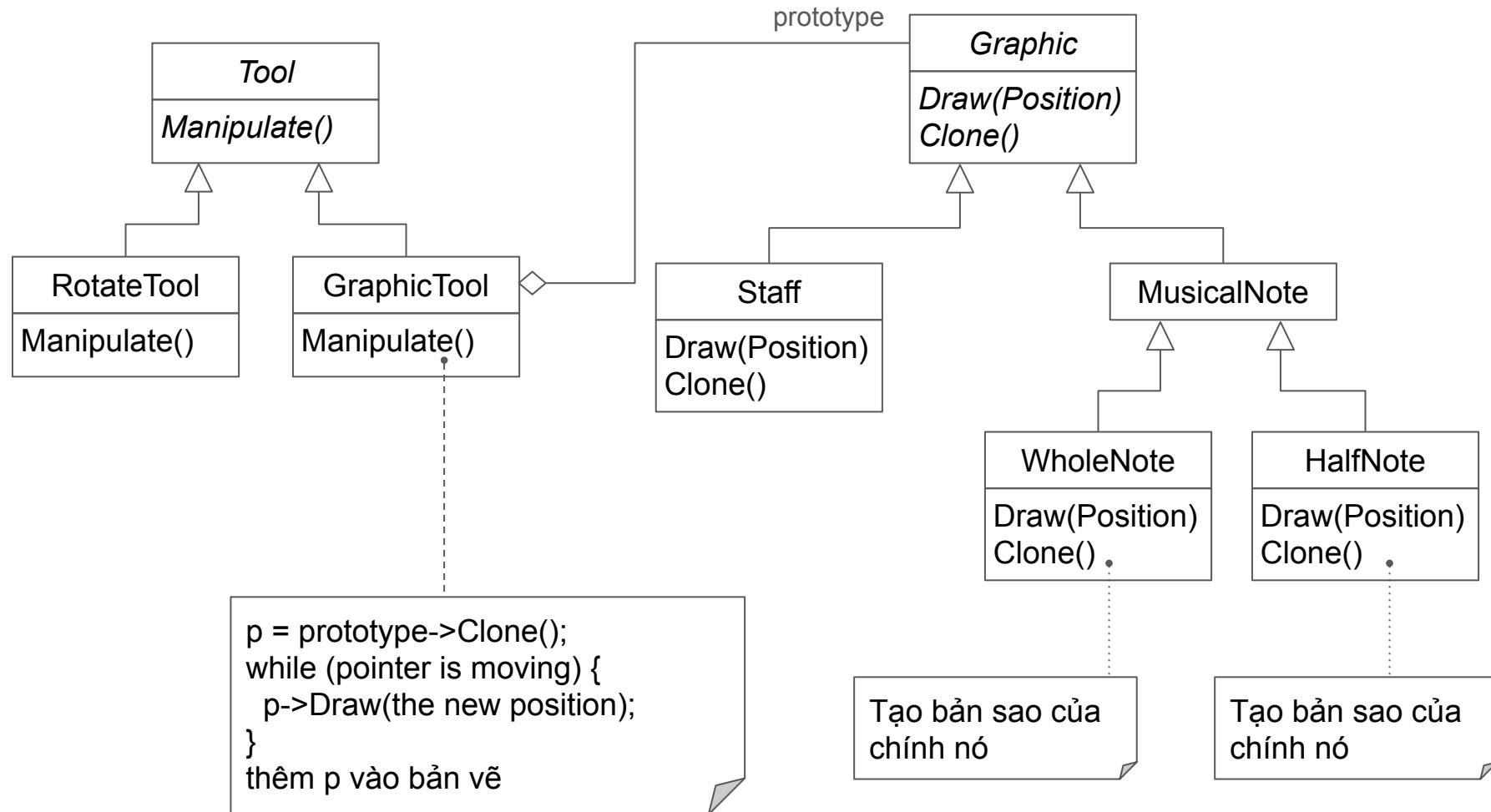
# Nguyên mẫu: Hành vi



# Nguyên mẫu: Các hệ quả

- Ứng dụng có thể đăng ký hoặc hủy bản mẫu ở thời gian thực thi.
- Hỗ trợ tạo đối tượng bằng cách thay đổi các giá trị.
- Hỗ trợ tạo đối tượng mới bằng cách thay đổi cấu trúc, kết hợp các đối tượng thành phần.
- Giảm sử dụng kế thừa, như phương thức xưởng để tạo đối tượng.
- Tùy chỉnh ứng dụng với các lớp động, hỗ trợ tạo đối tượng của các lớp được nạp ở thời gian thực thi.

# Ví dụ 5. Ứng dụng soạn nhạc



# Lớp độc bản

*Singleton*

Đảm bảo một lớp chỉ có đúng một bản thực và cung cấp khả năng truy cập toàn cục tới bản thực đó.

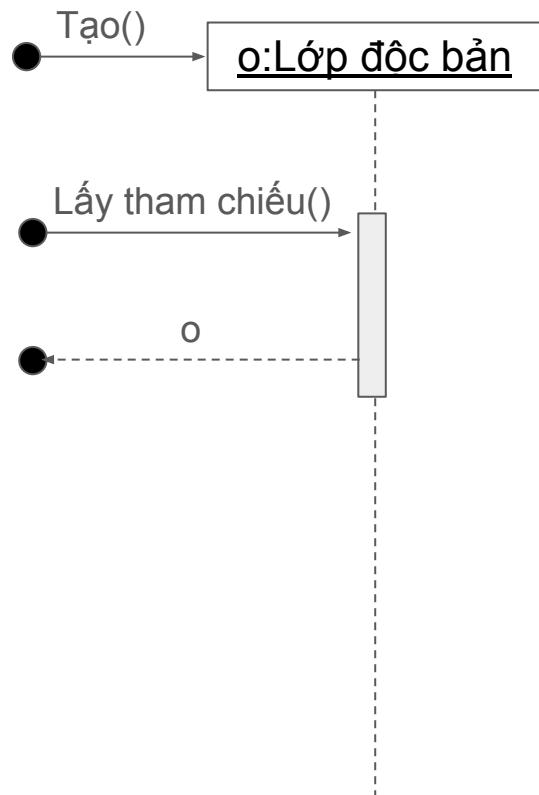
# Lớp độc bản: Cấu trúc

*Lớp độc bản*

+ Lấy tham chiếu()

Trả về tham chiếu tới 1 đối tượng duy nhất. Có thể tạo mới ở lần gọi đầu tiên.

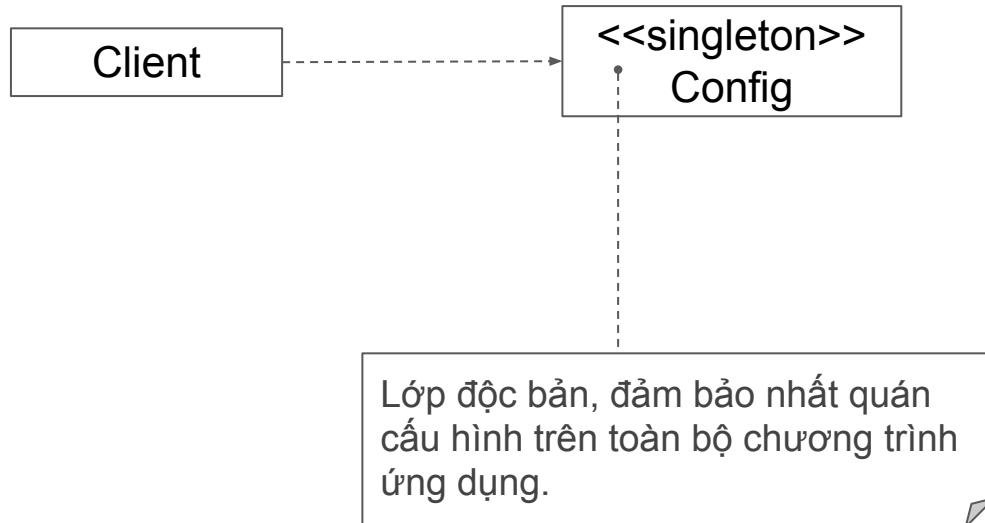
# Lớp độc bản: Hành vi



# Lớp độc bản: Các hệ quả

- Có thể thiết lập quyền truy cập tới bản thực duy nhất.
- Giản lược không gian tên, hạn chế biến toàn cục.
- Có thể kế thừa lớp độc bản và tùy chỉnh lớp mở rộng.
- Có thể thay đổi số lượng thực bản, điều chỉnh để cho phép nhiều hơn 1 thực bản.
- Có hạn chế đối với phương thức tĩnh. Ví dụ phương thức static trong C++ không thể là virtual, và không thể áp dụng đa hình.

# Ví dụ 6. Tùy chỉnh toàn cục



# Tổng kết về các mẫu tạo

- Các lớp tạo đảm nhận trách nhiệm tạo đối tượng thông qua đó có thể tham số hóa việc tạo đối tượng bằng cách thay đổi lớp tạo.
- Phương thức xưởng thay đổi đối tượng được tạo thông qua kế thừa lớp tạo, nguyên lý đơn giản nhất.
- Nhà máy trừu tượng làm việc với 1 nhóm đối tượng.
- Thợ xây cho phép quản lý sâu tiến trình tạo đối tượng phức tạp.
- Nguyên mẫu cho phép sao chép đối tượng, lớp được tạo chính là lớp tạo.
- Lớp độc bản giới hạn số lượng bản thực của lớp.

# Nội dung

- Mẫu tạo
- Mẫu kết cấu
- Mẫu tương tác



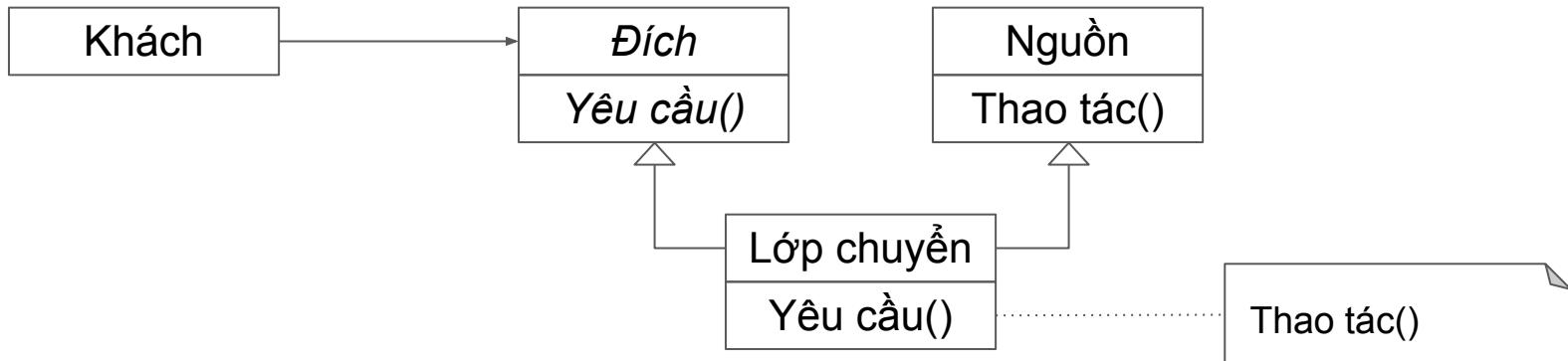
# Lớp chuyển

*Adapter*

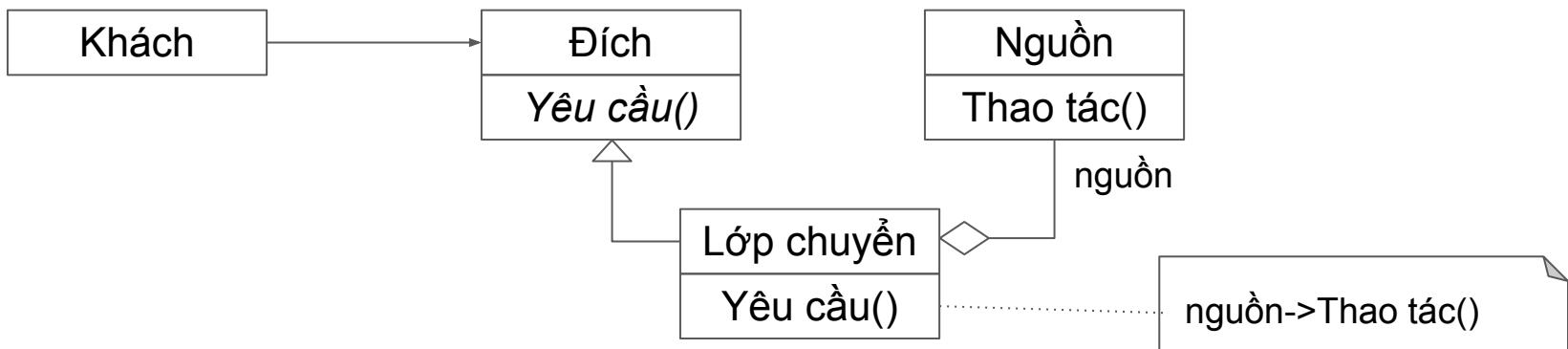
Chuyển đổi giao diện của một lớp đã có thành giao diện được yêu cầu từ phía khách, chuyển đổi một lớp với giao diện không tương thích thành một lớp với giao diện tương thích mà không thay đổi lớp đó.

# Lớp chuyển: Cấu trúc

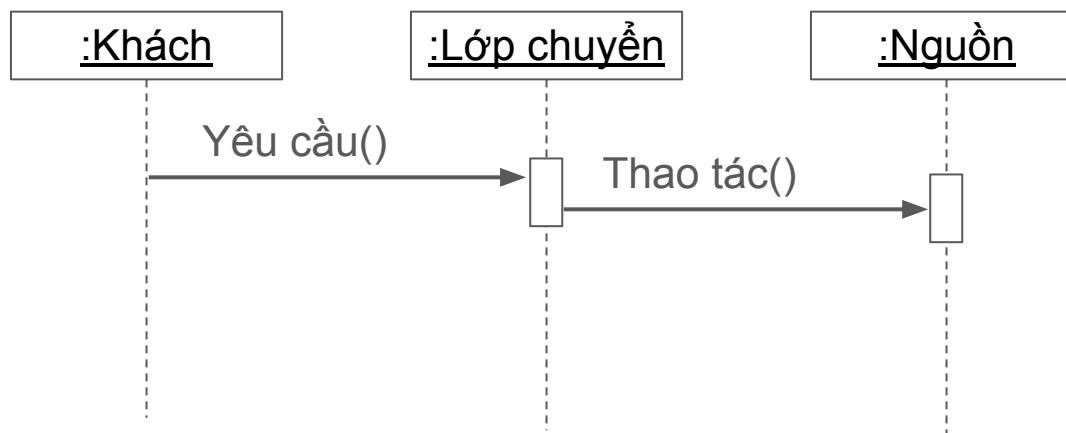
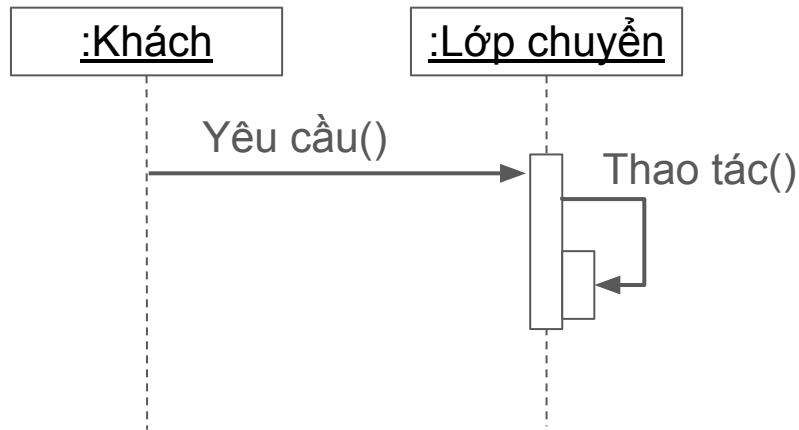
*Chuyển đổi dựa trên đa kế thừa*



*Chuyển đổi bằng cách bọc đối tượng nguồn*



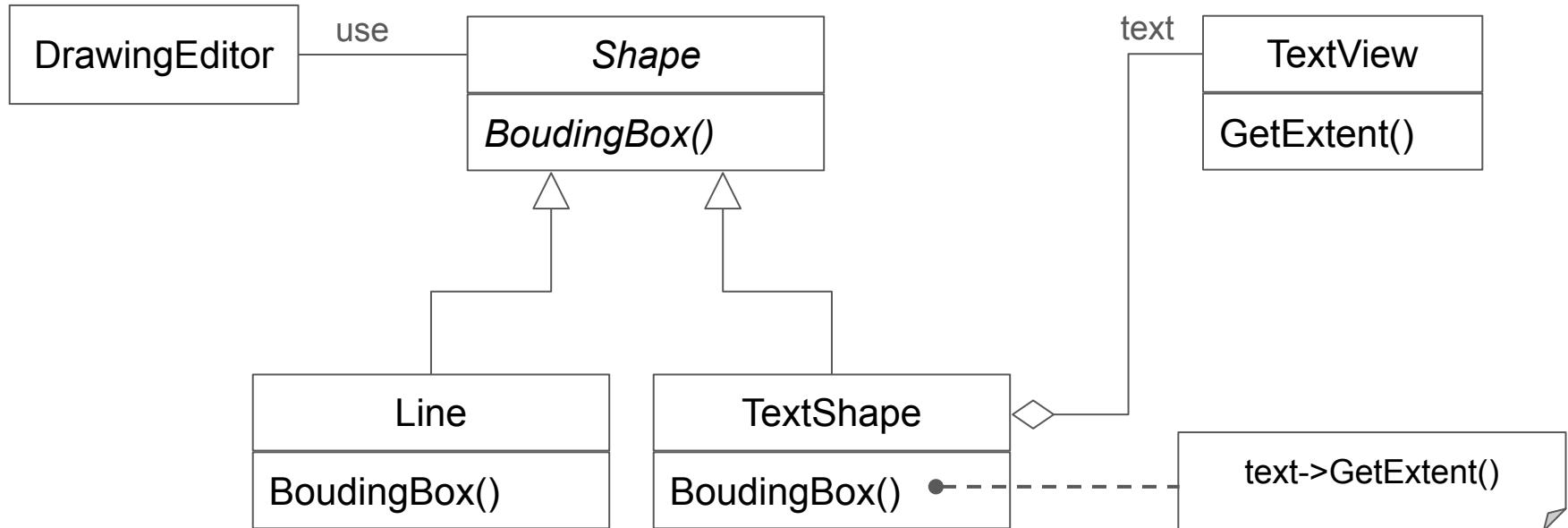
# Lớp chuyển: Hành vi



# Lớp chuyển: Các hệ quả

- Chuyển đổi dựa trên đa kế thừa:
  - Giới hạn phạm vi triển khai, trong điều kiện cho phép đa kế thừa.
  - Cho phép nạp chồng một số hành vi của lớp nguồn, bởi vì lớp chuyển kế thừa lớp được chuyển đổi.
  - Chỉ sử dụng một đối tượng, không liên kết với đối tượng được chuyển đổi.
- Chuyển đổi dựa trên đóng gói:
  - Cho phép một đối tượng chuyển hoạt động với nhiều đối tượng được chuyển đổi.
  - Không nạp chồng được hành vi của cái được chuyển đổi trong lớp chuyển.

# Ví dụ 7. Ứng dụng về

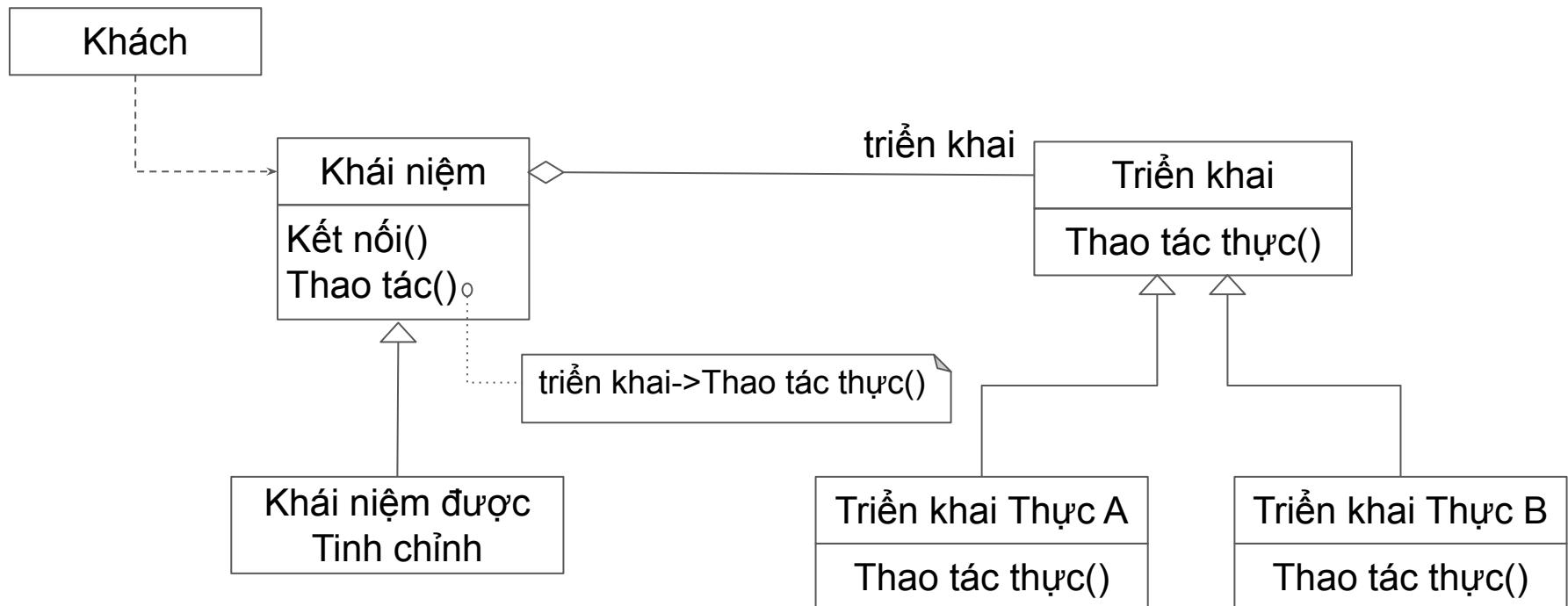


# Bắc cầu

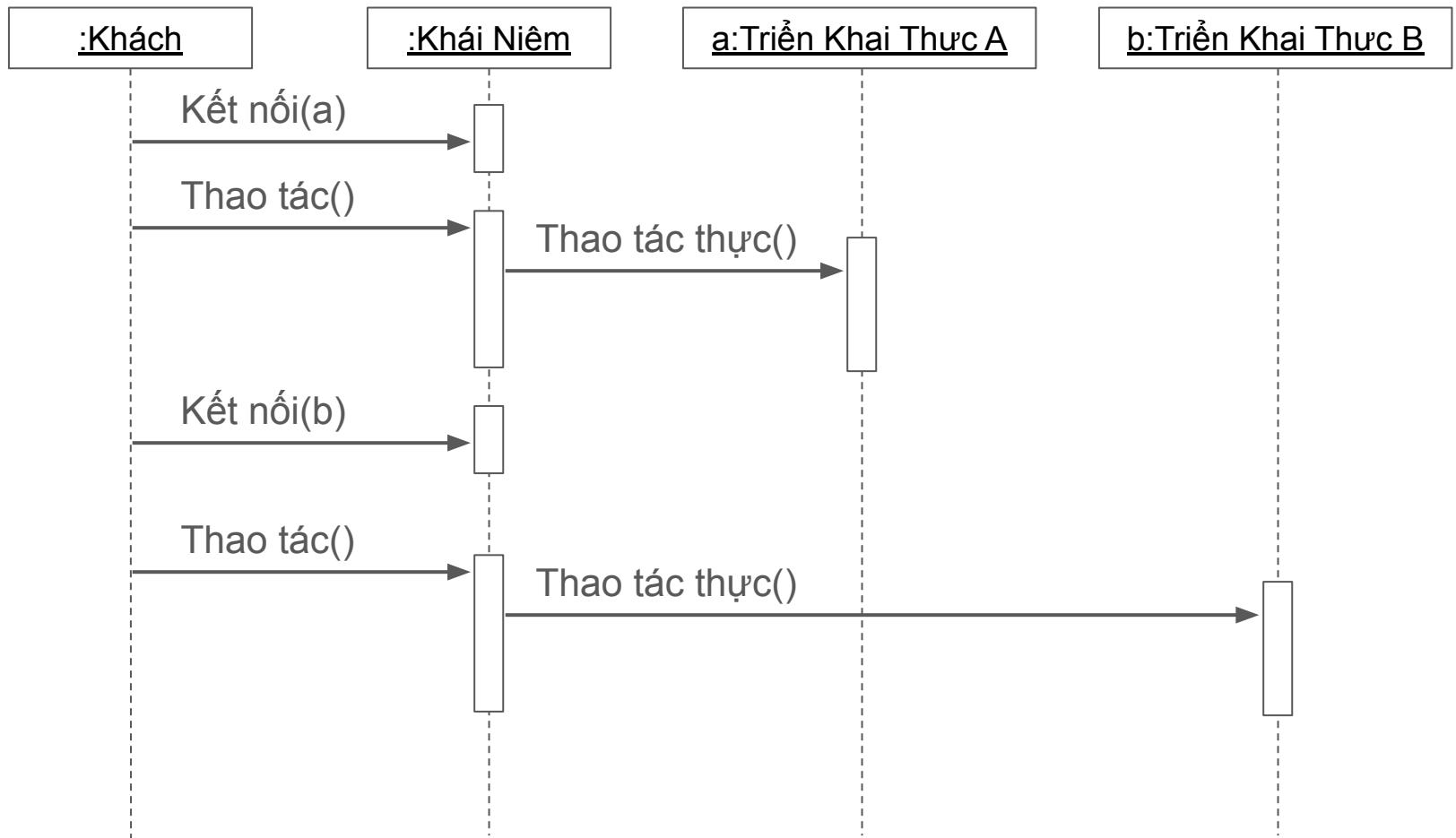
*Bridge*

Tách riêng khái niệm và triển khai để cả hai có thể thay đổi độc lập với nhau.

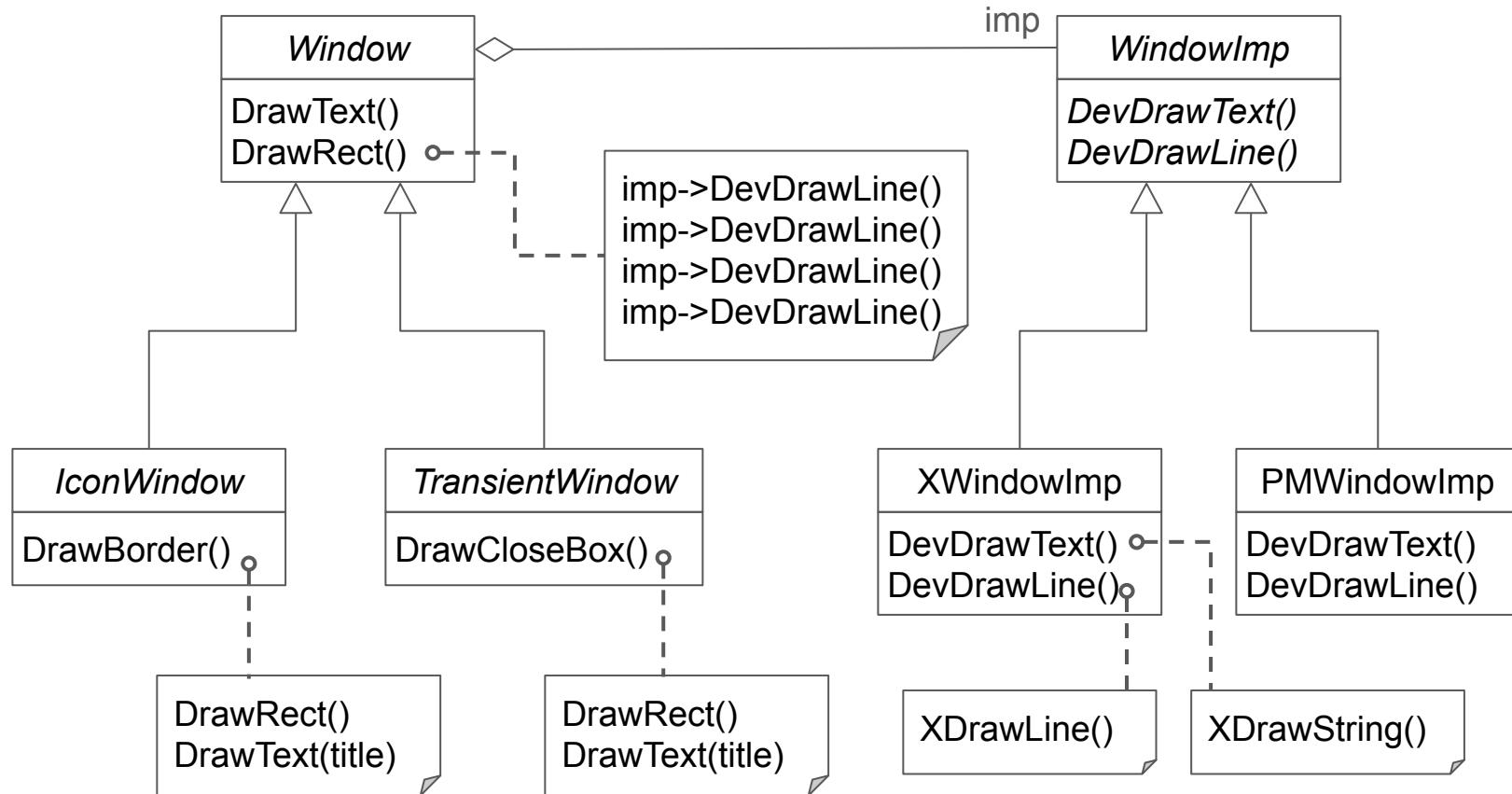
# BẮC CẦU: CẤU TRÚC



# Bắc cầu: Hành vi



# Ví dụ 8. Cửa sổ trong giao diện đồ họa



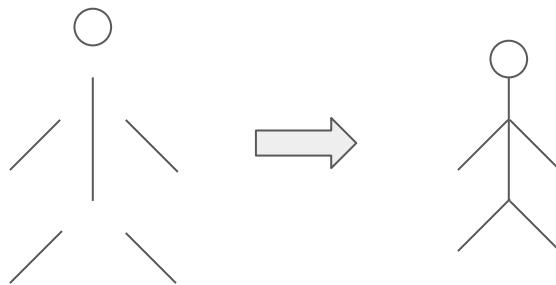
# BẮC CẦU: Các hệ quả

- Tách riêng khái niệm và triển khai. Triển khai không bị gắn chặt với khái niệm, triển khai của khái niệm có thể thay đổi ở thời gian thực thi.
- Tăng khả năng mở rộng. Có thể độc lập mở rộng cây khái niệm và cây triển khai.
- Ân các chi tiết triển khai với khách.

# Hợp chất

*Composite*

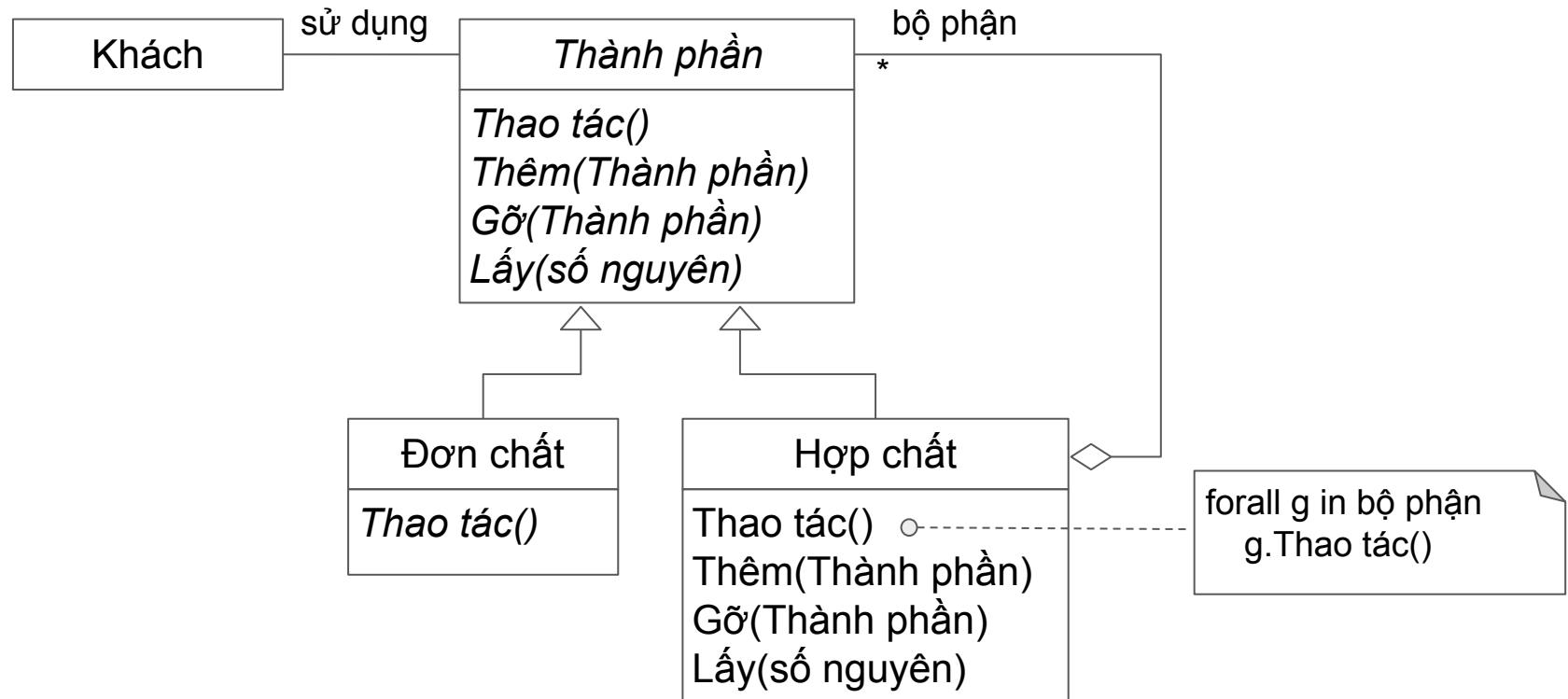
Kết hợp các đối tượng theo cấu trúc cây phân cấp với quan hệ bộ phận-tổng thể. Hợp chất cho phép khách tương tác với tổ hợp đối tượng như với một đối tượng.



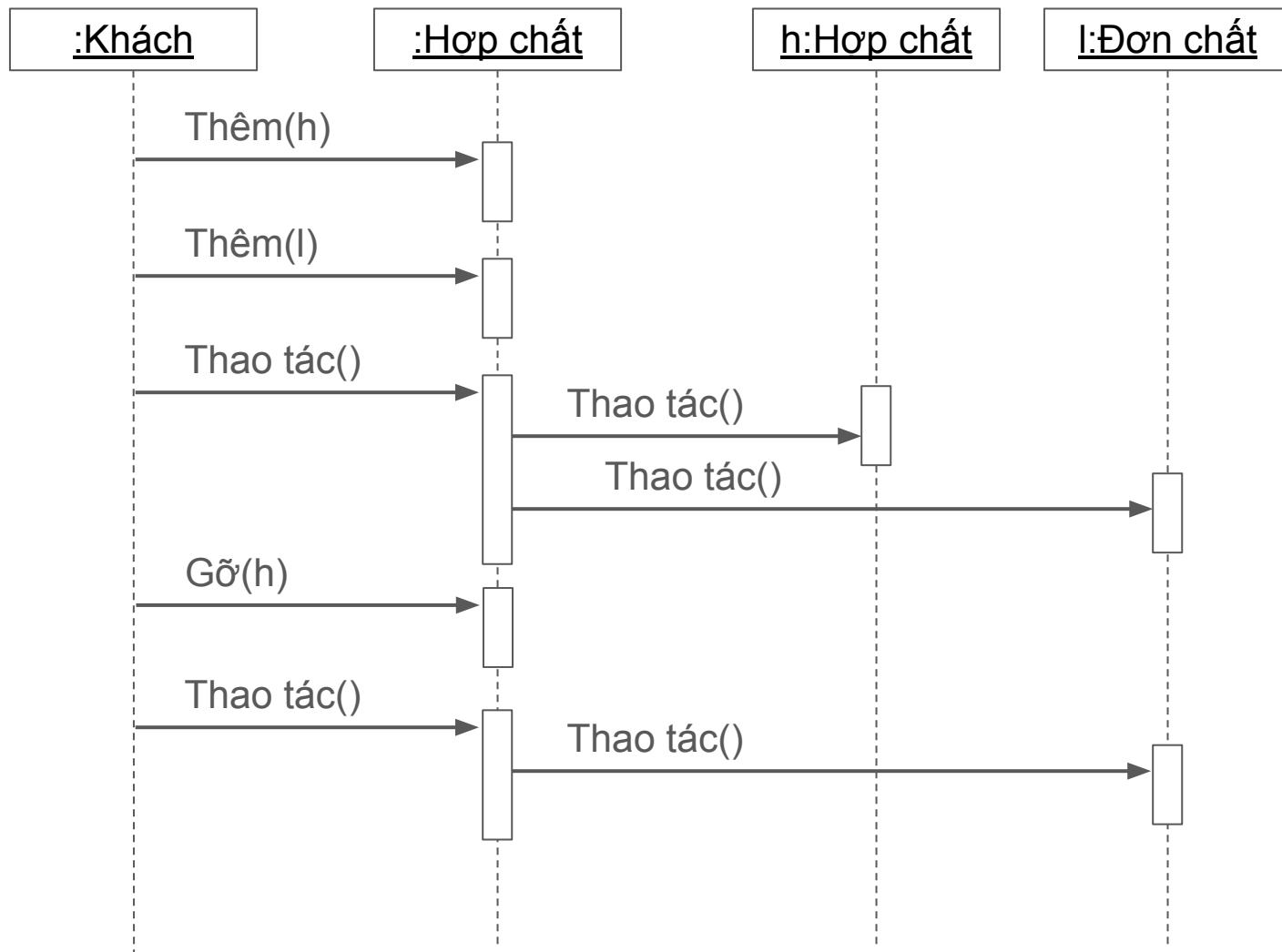
18 % Cr + 1 % Mn + 8 % Ni + 0.07 C + Fe => Inox 304

...

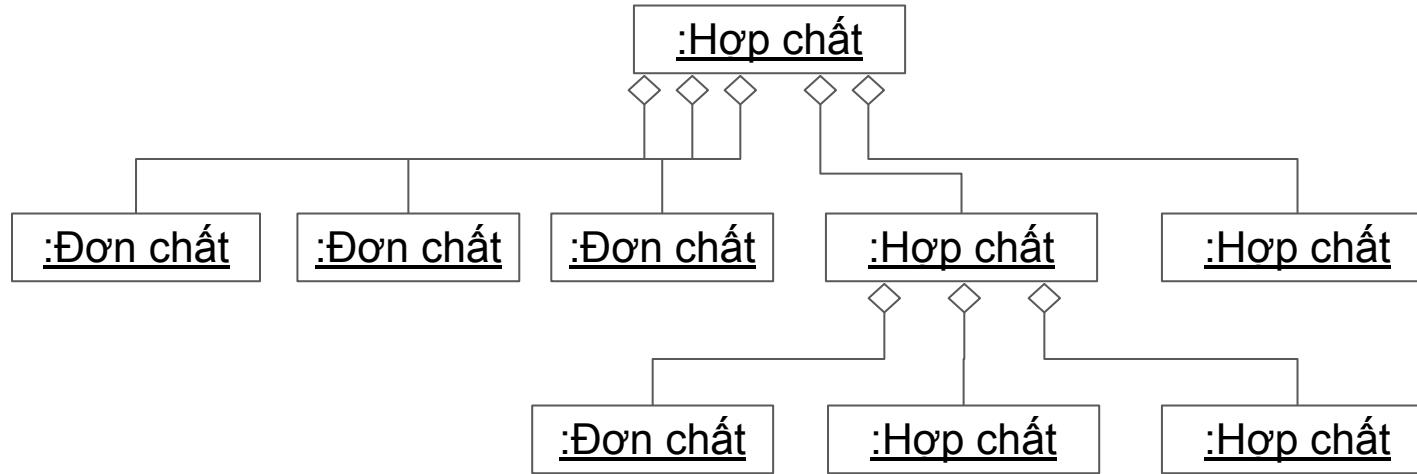
# Hợp chất: Cấu trúc



# Hợp chất: Hành vi



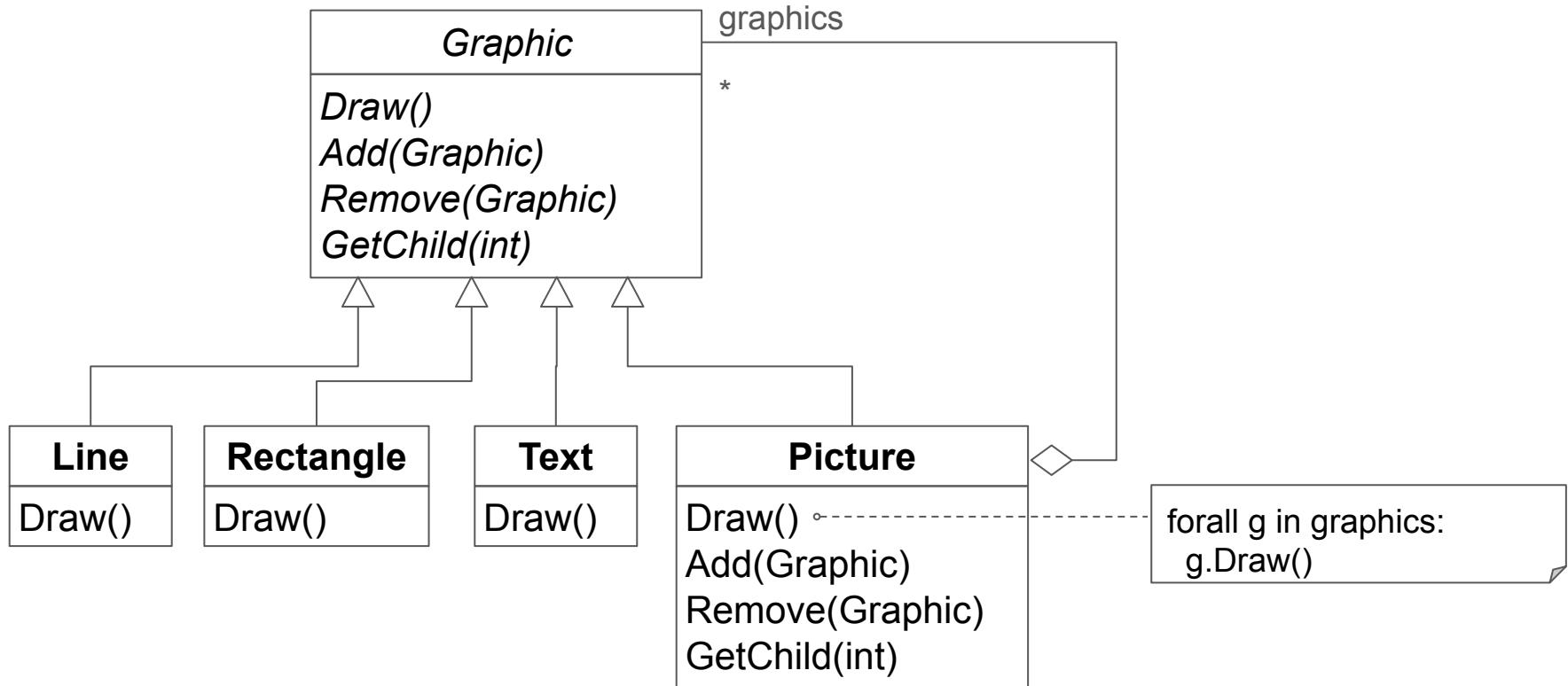
# Hợp chất: Kết hợp đối tượng



# Hợp chất: Các hệ quả

- Các đối tượng cơ bản có thể được sát nhập vào hợp chất. Các hợp chất cũng có thể tiếp tục sát nhập vào hợp chất khác tạo thành cây phân cấp.
- Khách tương tác với các đối tượng riêng và hợp chất theo cùng một cách.
- Dễ thêm các thành phần mới.
- Khó giới hạn thành phần của hợp chất.

# Ví dụ 9. Các thành phần hình vẽ



# Lớp tô điểm

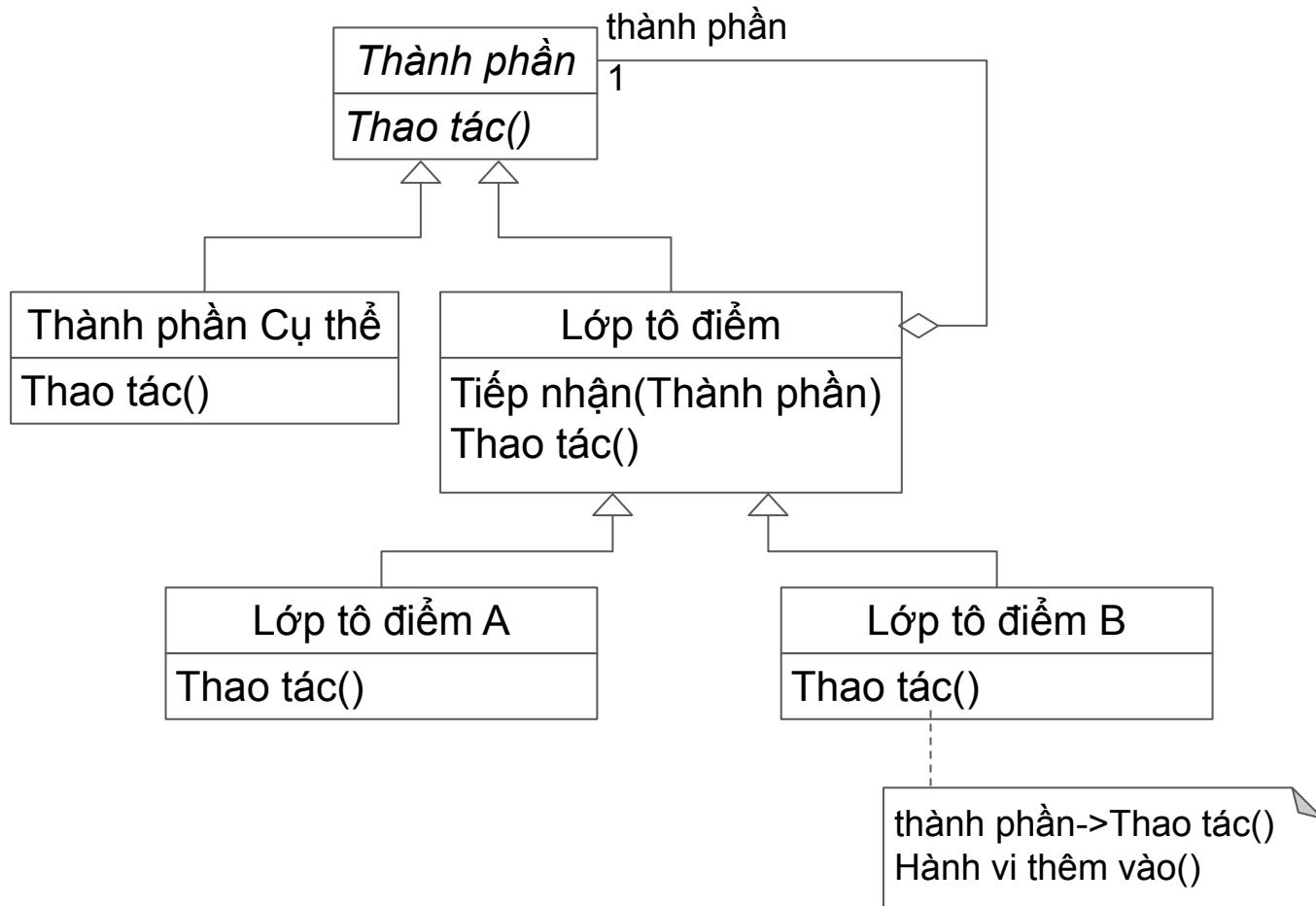
*Decorator*

Bổ xung linh động các trách nhiệm cho đối tượng trong thời gian thực thi. Lớp tô điểm là một lựa chọn khác mềm dẻo hơn so với kế thừa để mở rộng chức năng.

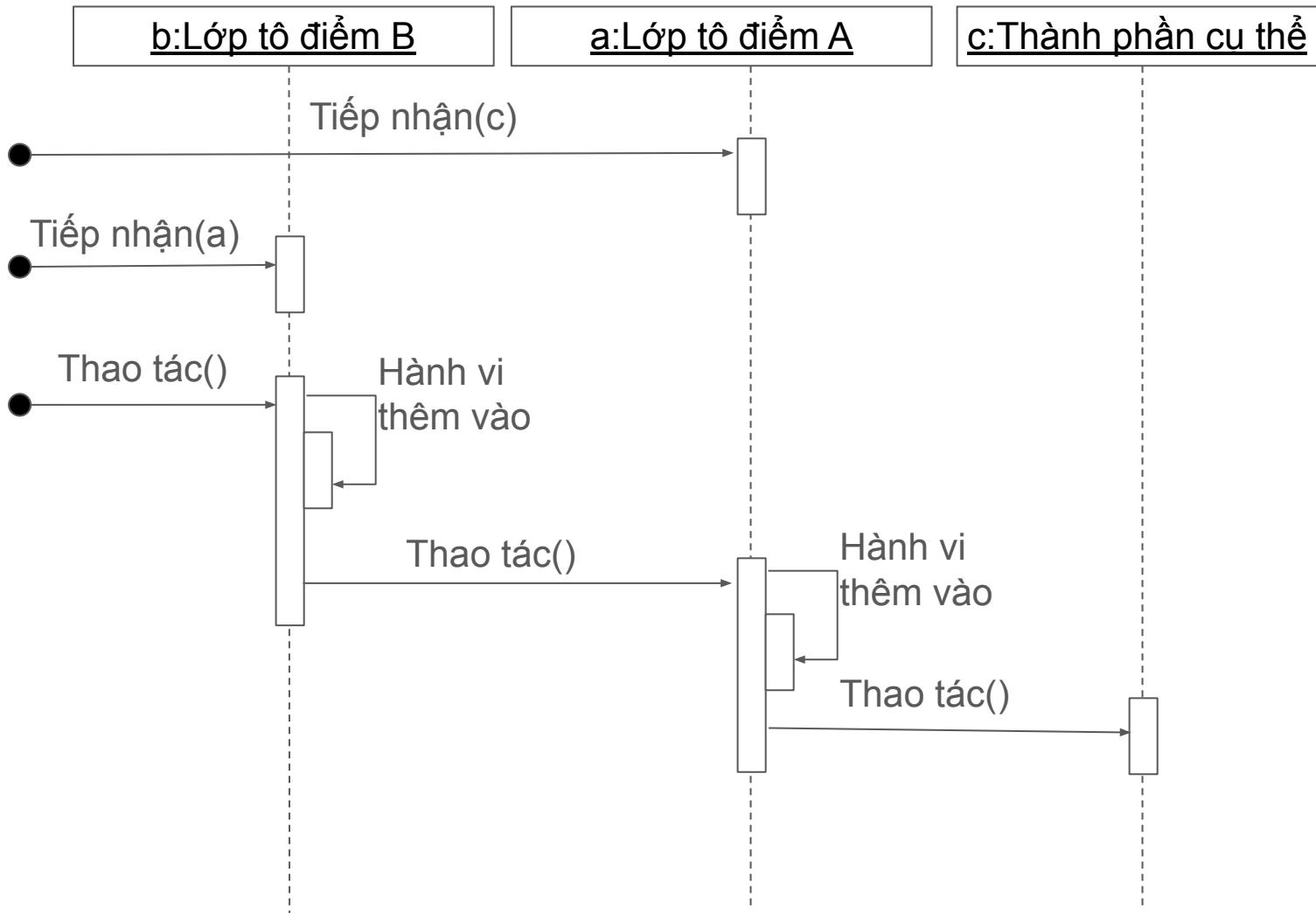
Tượng thạch cao + Màu => Tượng được tô màu

...

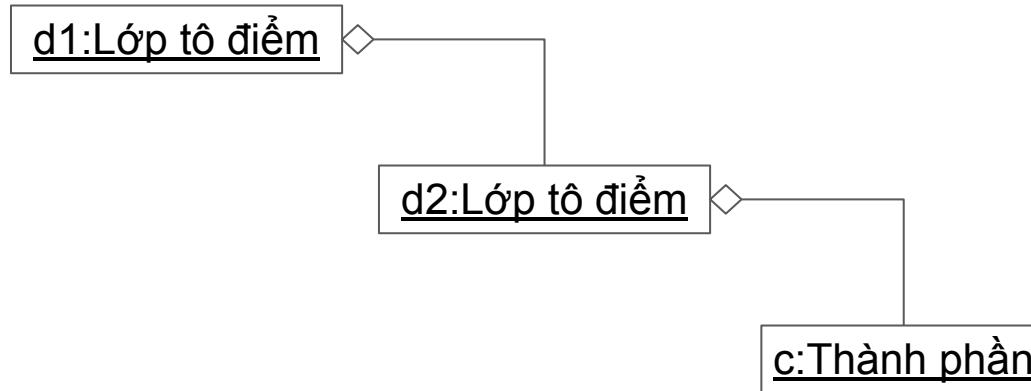
# Lớp tô điểm: Cấu trúc



# Lớp tô điểm: Hành vi



# Lớp tô điểm: Kết hợp các đối tượng

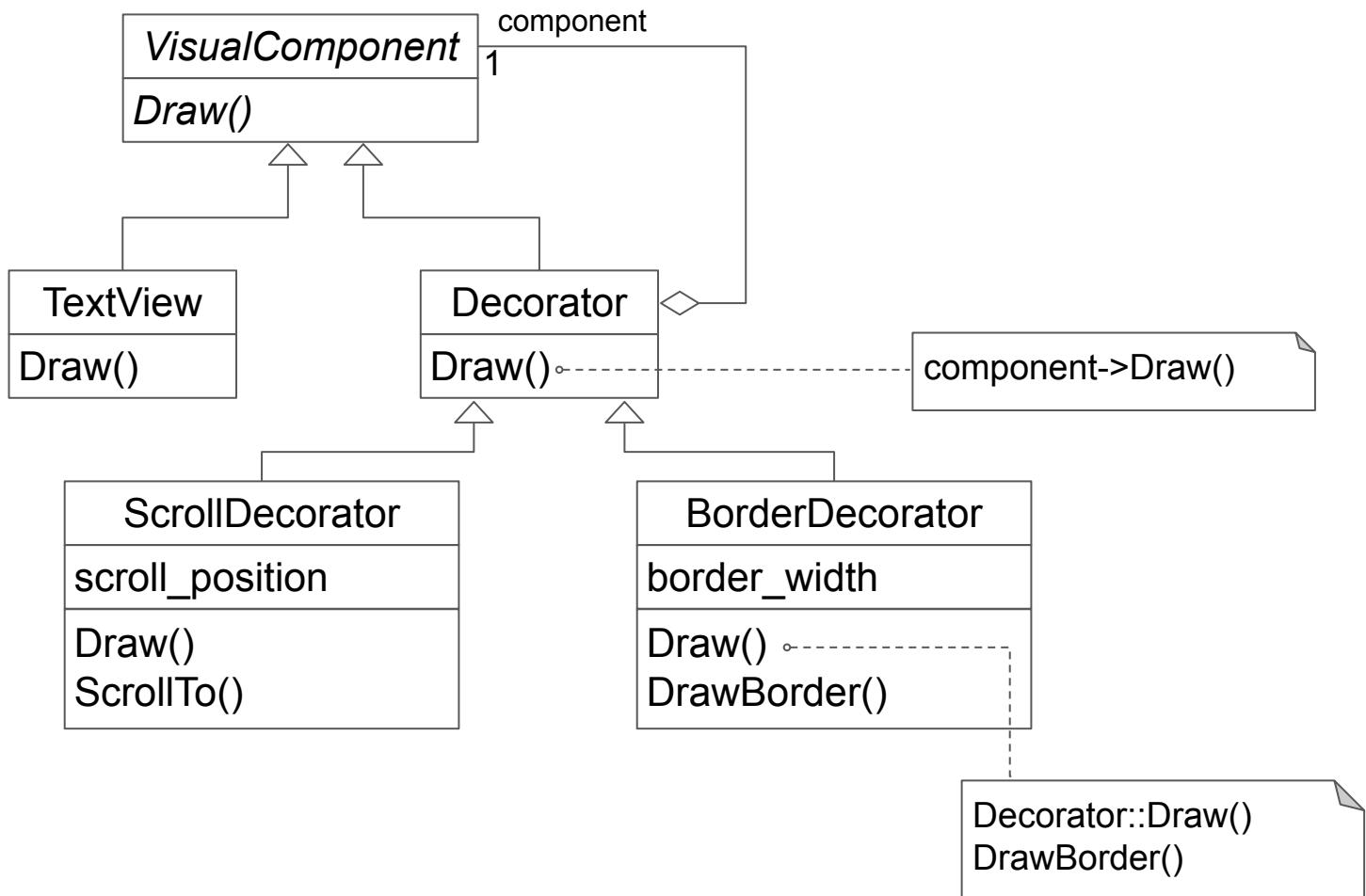


*Thành phần được gói trong đối tượng thuộc lớp tô điểm.*

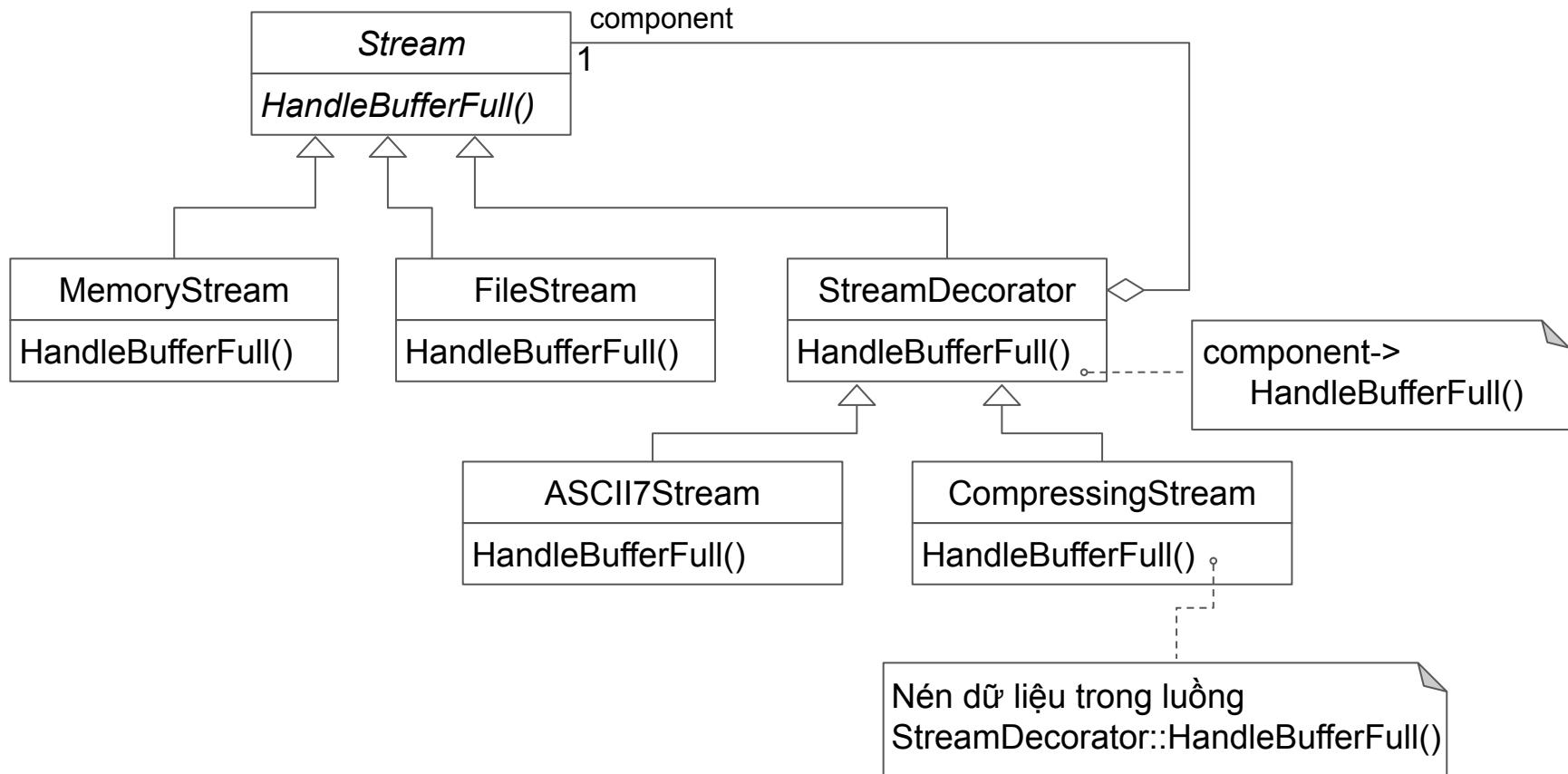
# Lớp tô điểm: Các hệ quả

- Linh động hơn kể thừa, các trách nhiệm có thể được thêm vào hoặc gỡ bỏ trong thời gian thực thi.
- Có thể tùy chỉnh, hỗ trợ hành vi phức tạp bằng cách thêm dần chức năng vào lớp đơn giản.
- Cái tô điểm và thành phần là các đối tượng khác nhau, thành phần được gói trong cái tô điểm.
- Dễ tùy chỉnh đối tượng, tuy nhiên có thể khó kết hợp chúng và gỡ rối.

# Ví dụ 10. Cửa sổ văn bản



# Ví dụ 11. Xử lý luồng dữ liệu



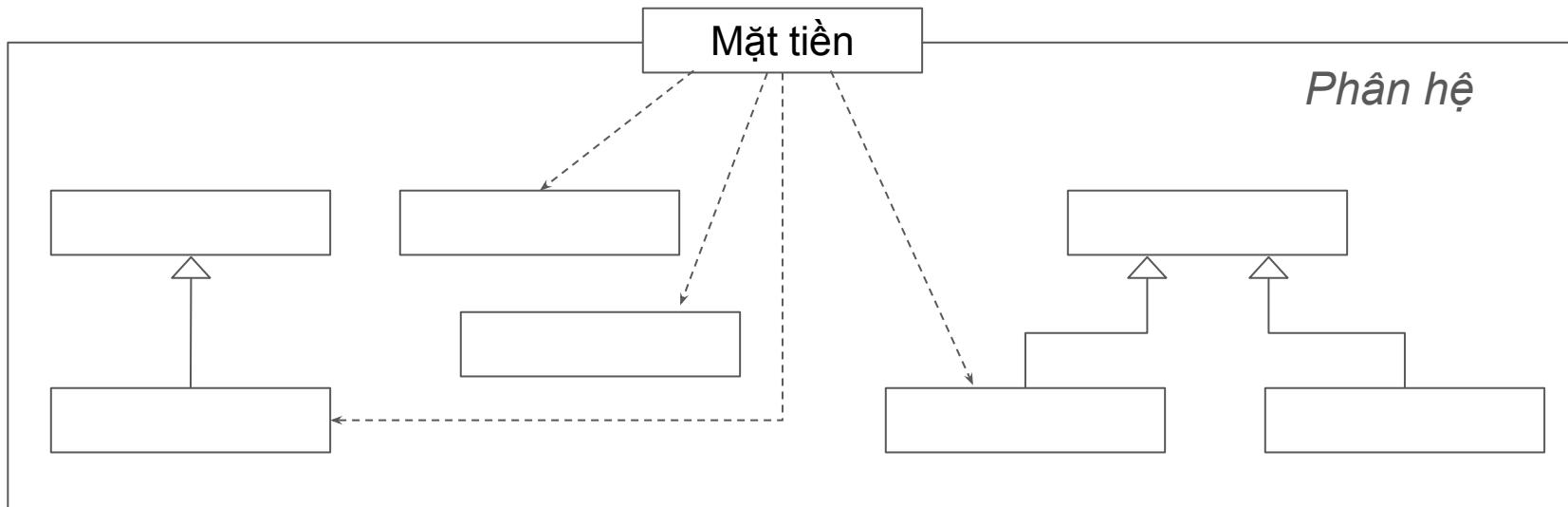
# Mặt tiền

*Facade*

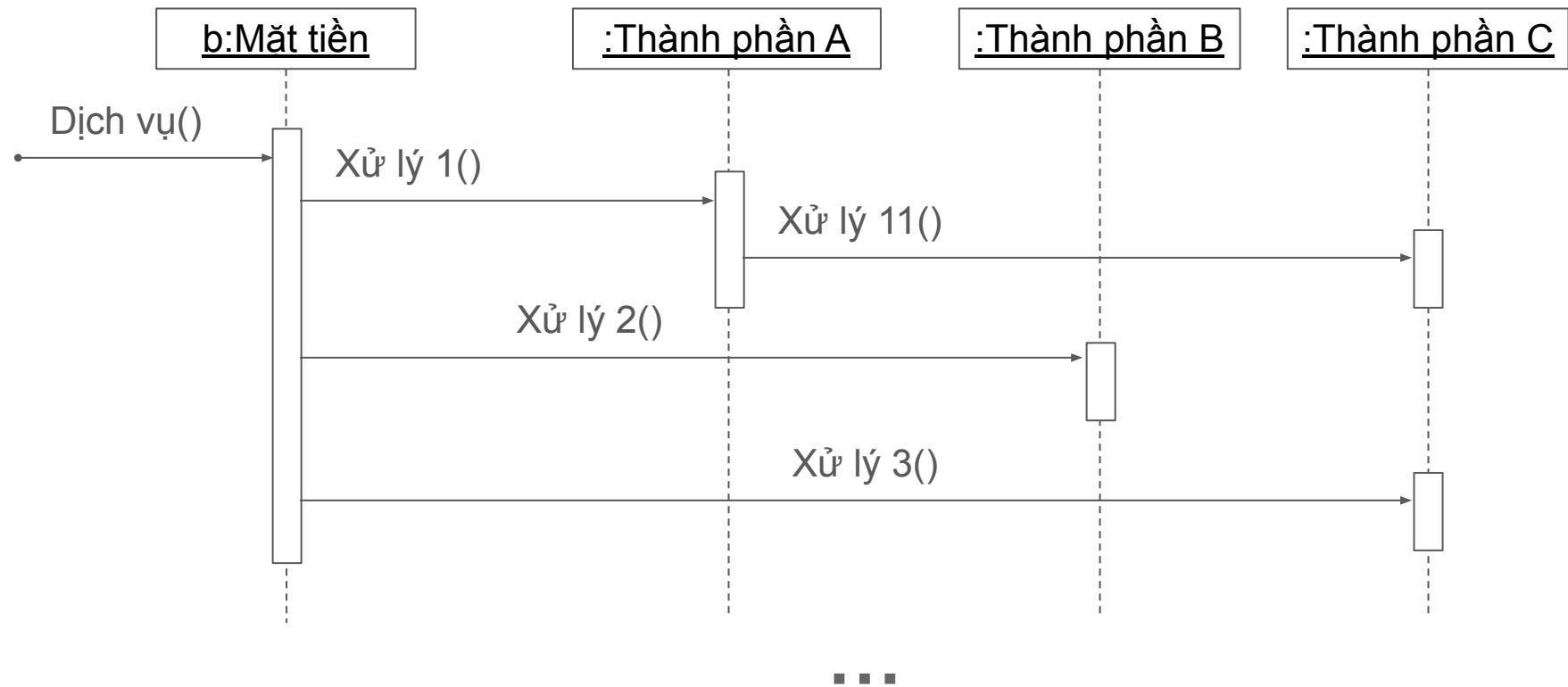
Cung cấp một giao diện hợp nhất cho một tập giao diện trong một phân hệ. Mặt tiền định nghĩa một giao diện bậc cao hơn để có thể sử dụng toàn phân hệ dễ dàng hơn.

Cung cấp các giá trị cụ thể hơn so với các tính năng nhỏ lẻ của các thành phần.

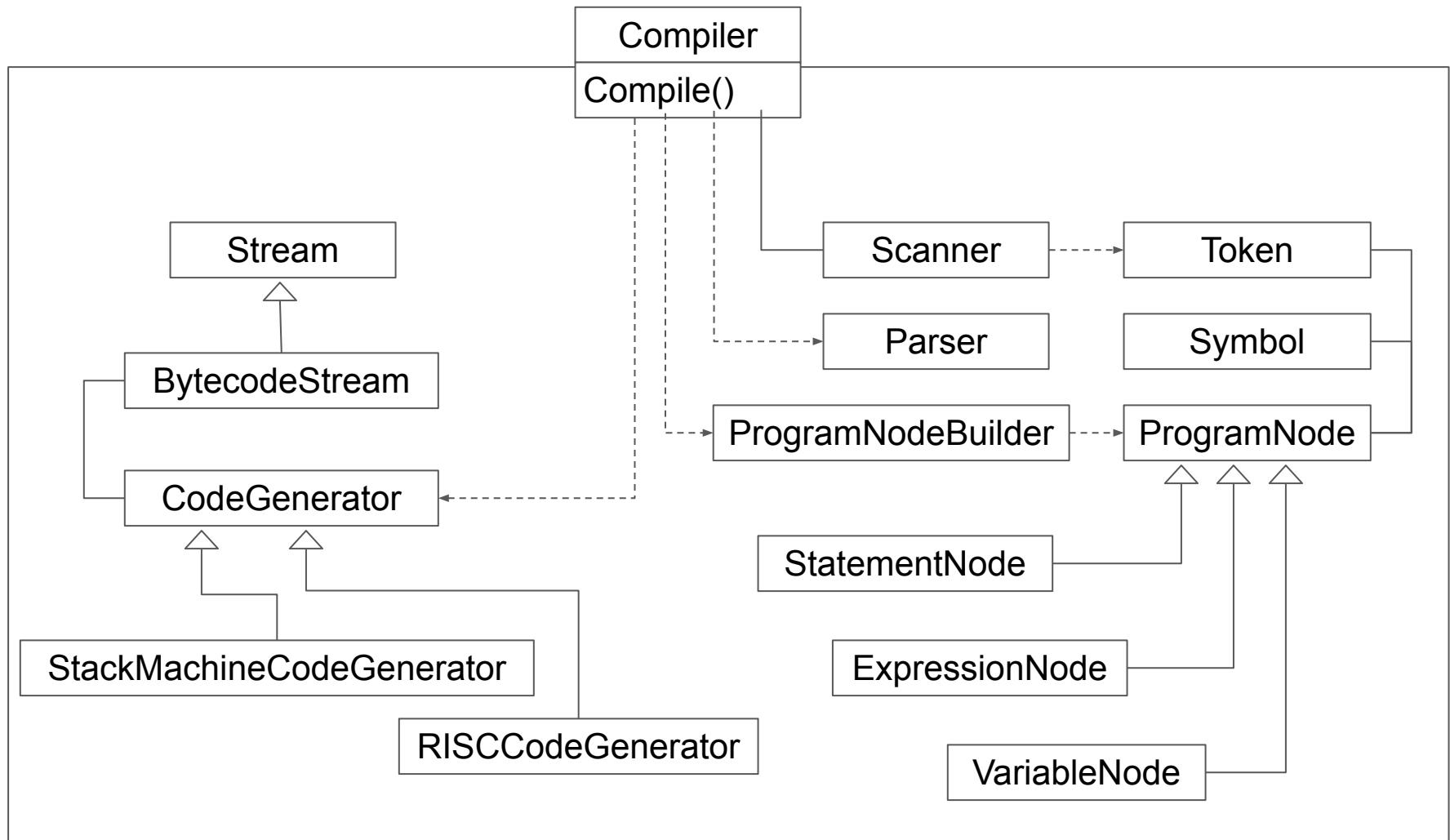
# Mặt tiền: Cấu trúc



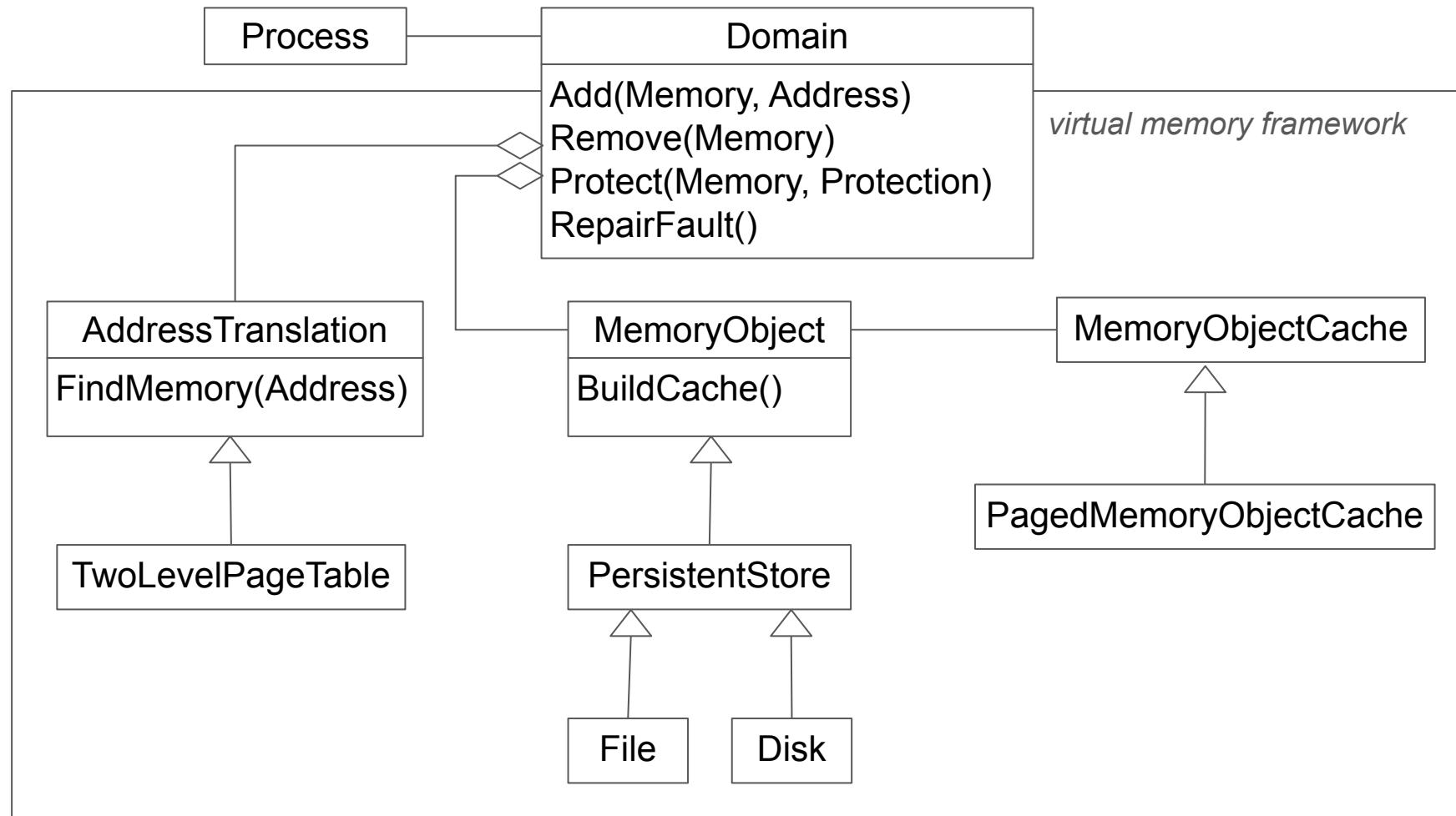
# Mặt tiền: Hành vi



# Ví dụ 12. trình biên dịch



# Ví dụ 13. Quản lý bộ nhớ



# Mặt tiền: Các hệ quả

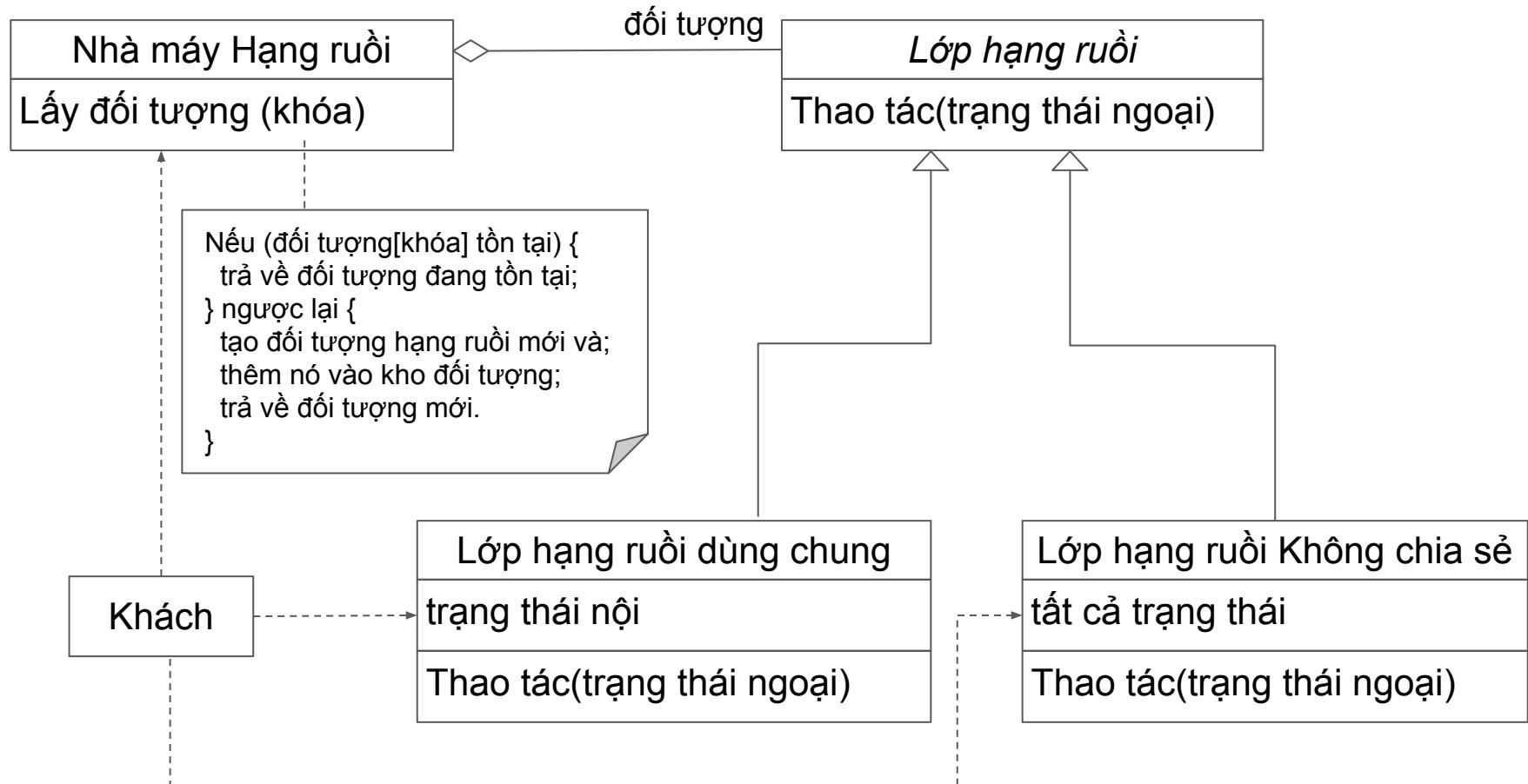
- Phân tách phạm vi sử dụng với các thành phần của phân hệ, qua đó giảm số lượng đối tượng tương tác và làm hệ thống dễ sử dụng hơn.
- Nới lỏng phụ thuộc giữa phân hệ và phía khách.
- Ứng dụng vẫn có thể sử dụng các lớp trong phân hệ nếu cần.

# Hạng ruồi

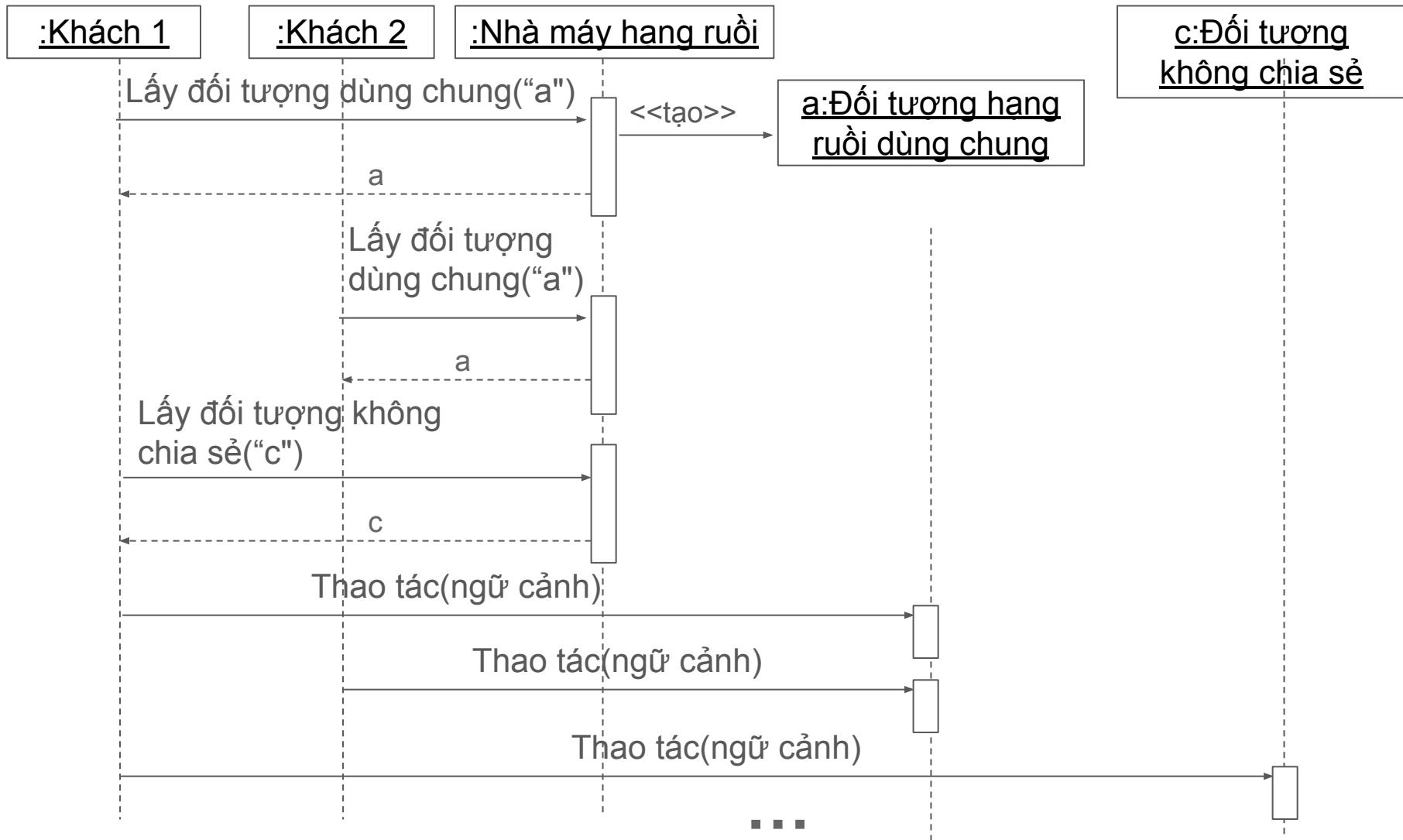
*Flyweight*

Chia sẻ để sử dụng hiệu quả các đối tượng nhỏ với số lượng lớn.

# Hạng ruồi: Cấu trúc



# Hạng ruồi: Hành vi



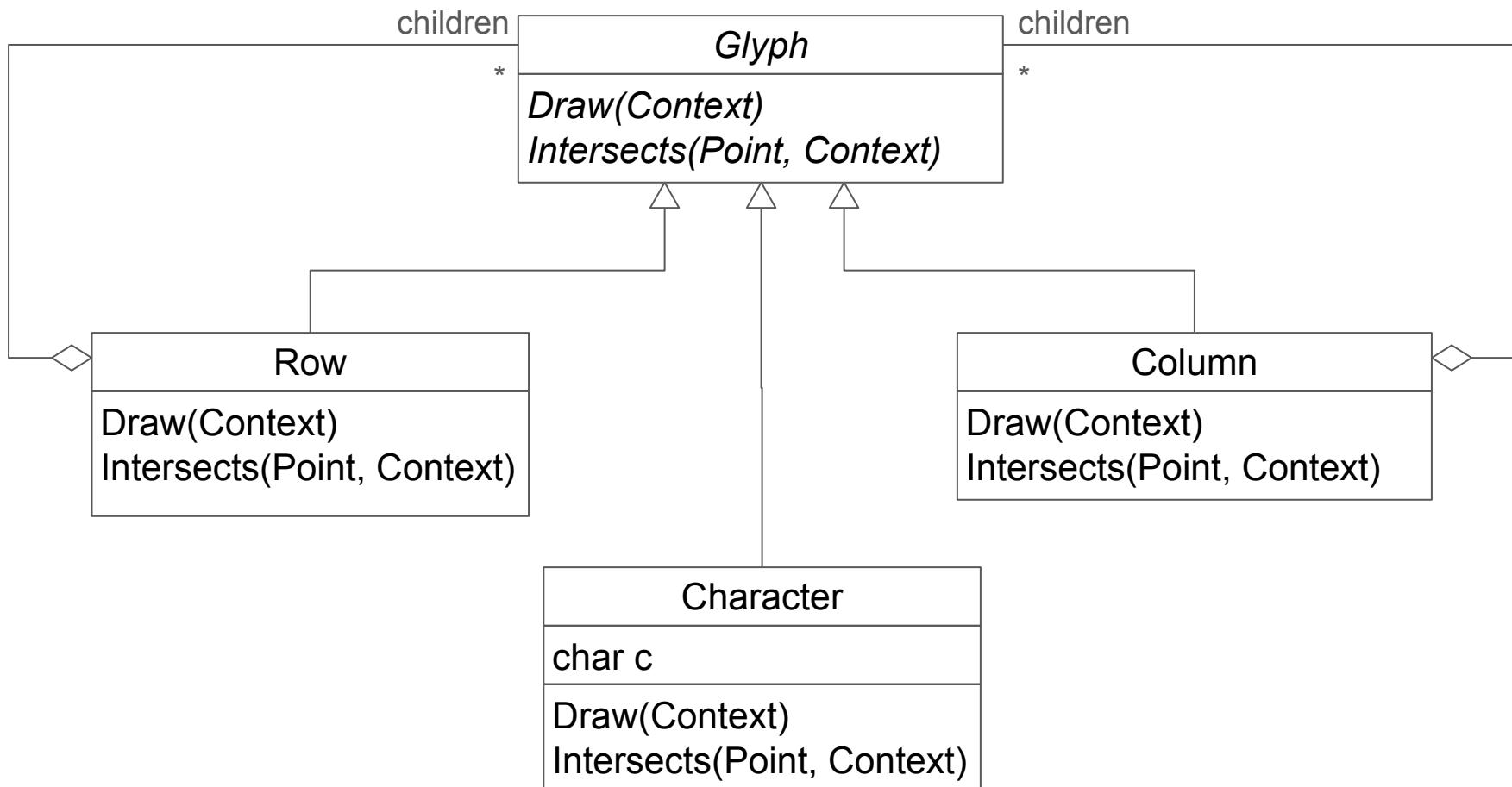
# Hạng ruồi: Kết hợp đối tượng



# Hạng ruồi: Các hệ quả

- Mức giảm dung lượng lưu trữ phụ thuộc vào tổng số lượng đối tượng được chia sẻ, dung lượng trạng thái nội của đối tượng, khả năng tính trạng thái ngoại.
- Đối tượng được chia sẻ càng nhiều thì dung lượng lưu trữ tiết kiệm được càng lớn.

# Ví dụ 14. Hiển thị văn bản

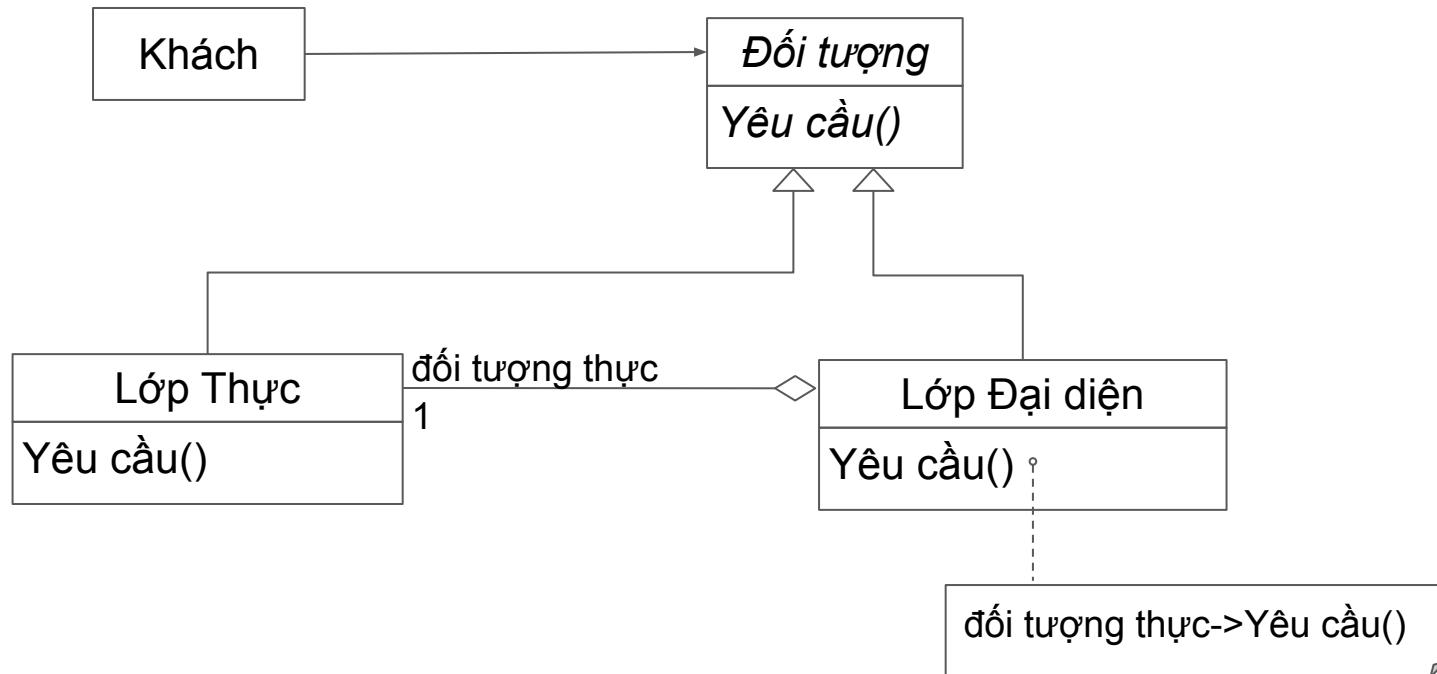


# Lớp đại diện

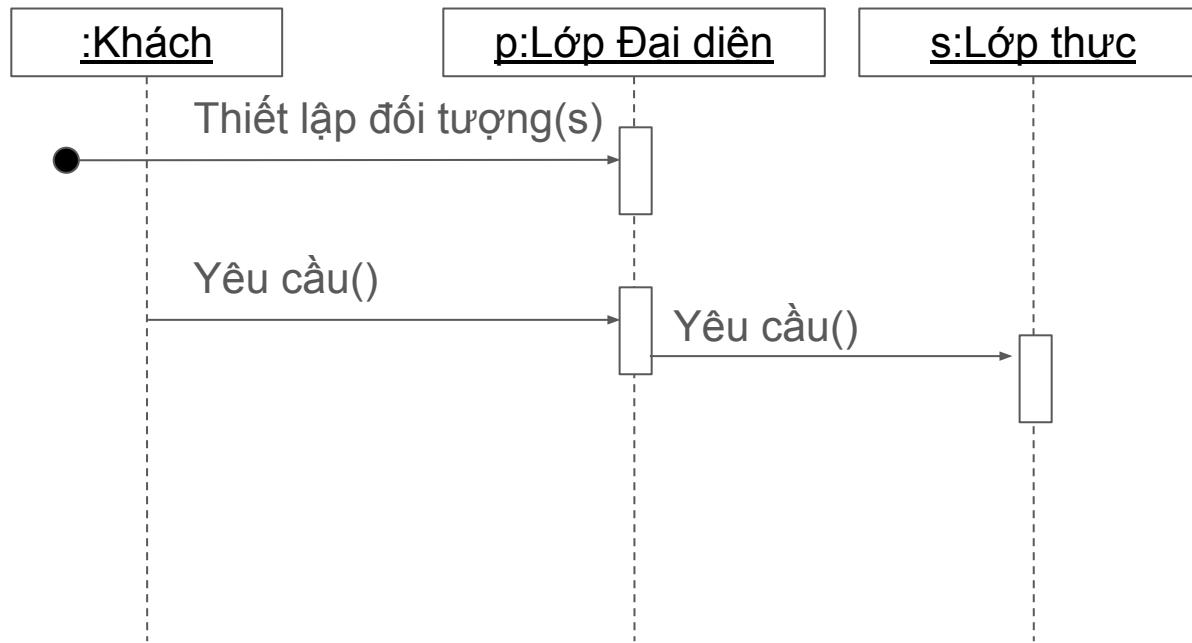
*Proxy*

Thiết lập một đối tượng đại diện cho một đối tượng khác để kiểm soát truy cập tới đối tượng đó.

# Lớp đại diện: Cấu trúc



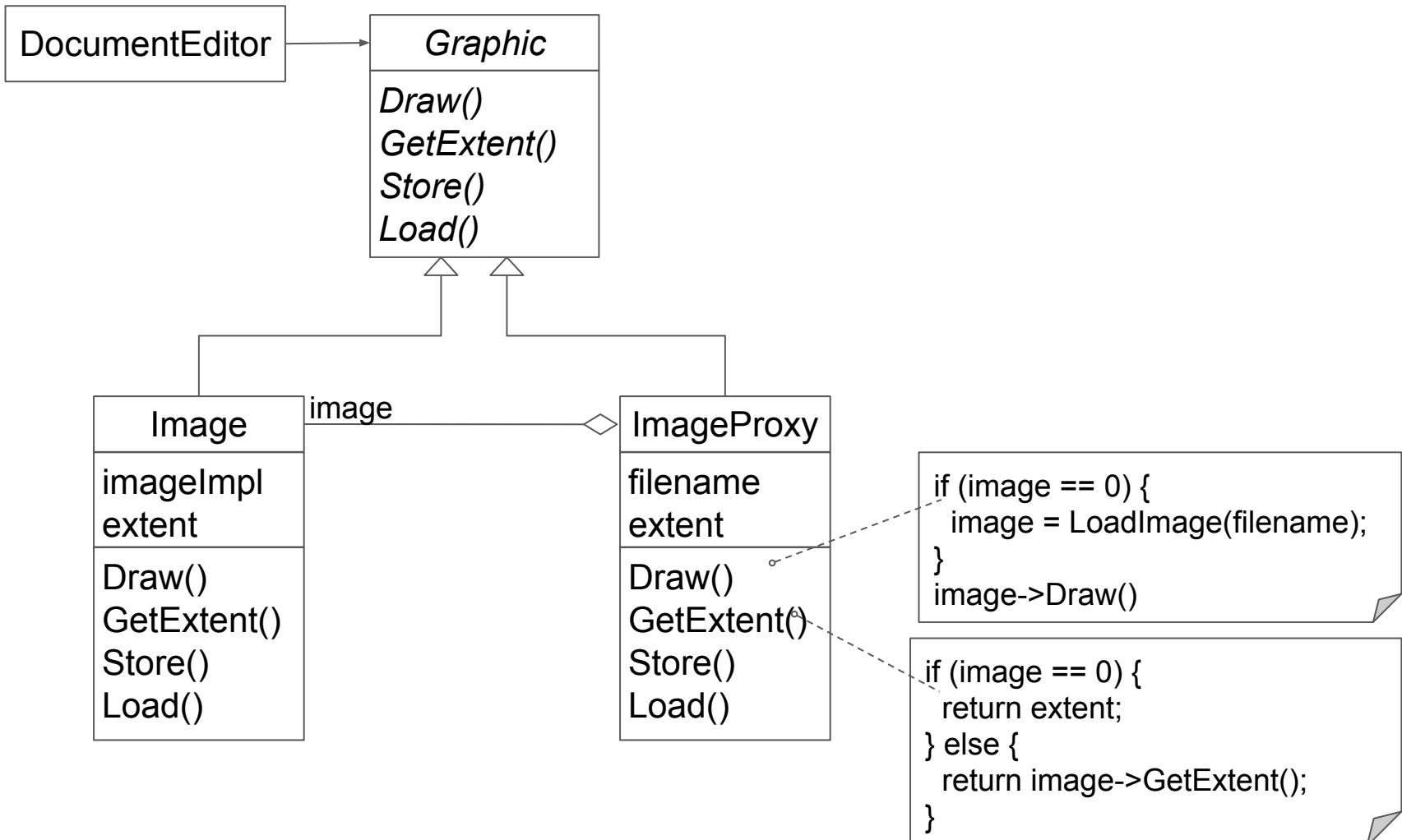
# Đại diện: Hành vi



# Đại diện: Các hệ quả

- Đối tượng đại diện truy cập từ xa có thể ẩn đi sự thật rằng nó nằm trong 1 không gian tên khác.
- Đối tượng đại diện ảo có thể thực hiện tối ưu hóa như tạo đối tượng khi cần.
- Các đối tượng đại diện cho phép thực hiện các công việc hỗ trợ khi truy cập đối tượng được đại diện.

# Ví dụ 15. Trình soạn thảo tài liệu



# Tổng kết về các mẫu kết cấu

- Các mẫu cấu trúc chia sẻ nhiều thành phần dựa trên kế thừa và kết hợp tuy nhiên mục đích sử dụng cho các trường hợp khác nhau.
- Tuy có cách kết hợp giống nhau nhưng Lớp chuyển hướng tới giải quyết vấn đề tương thích giao diện còn Bắc cầu hỗ trợ phân tách khái niệm và triển khai.
- Mặt tiền có vẻ giống lớp chuyển cho một tập đối tượng, tuy nhiên mặt tiền định nghĩa giao diện mới còn lớp chuyển sử dụng giao diện đang có.

# Nội dung

- Mẫu tạo
- Mẫu kết cấu
- Mẫu tương tác

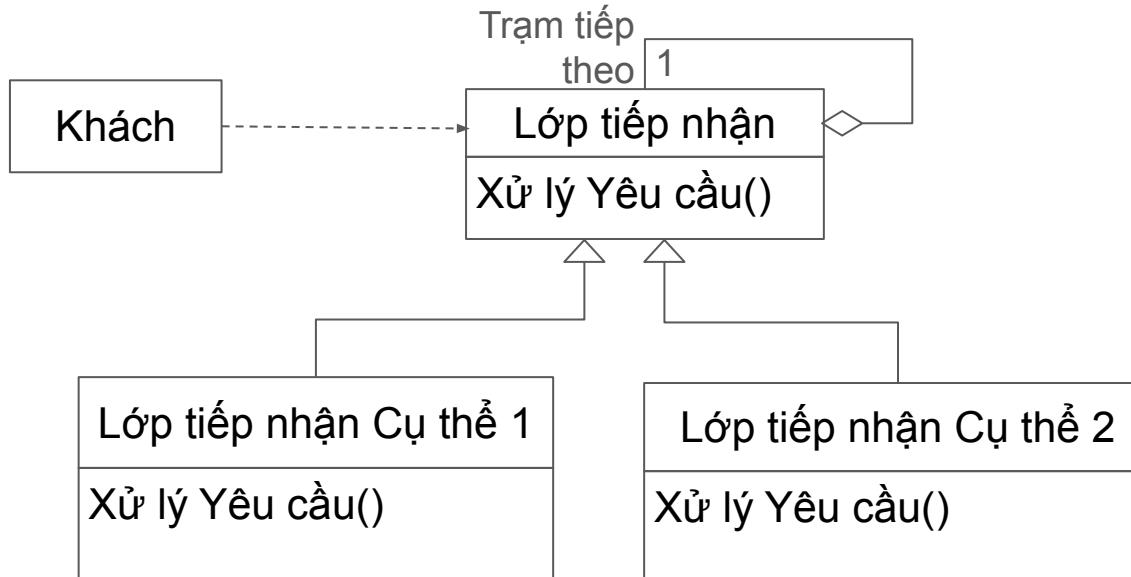


# Chuỗi trách nhiệm

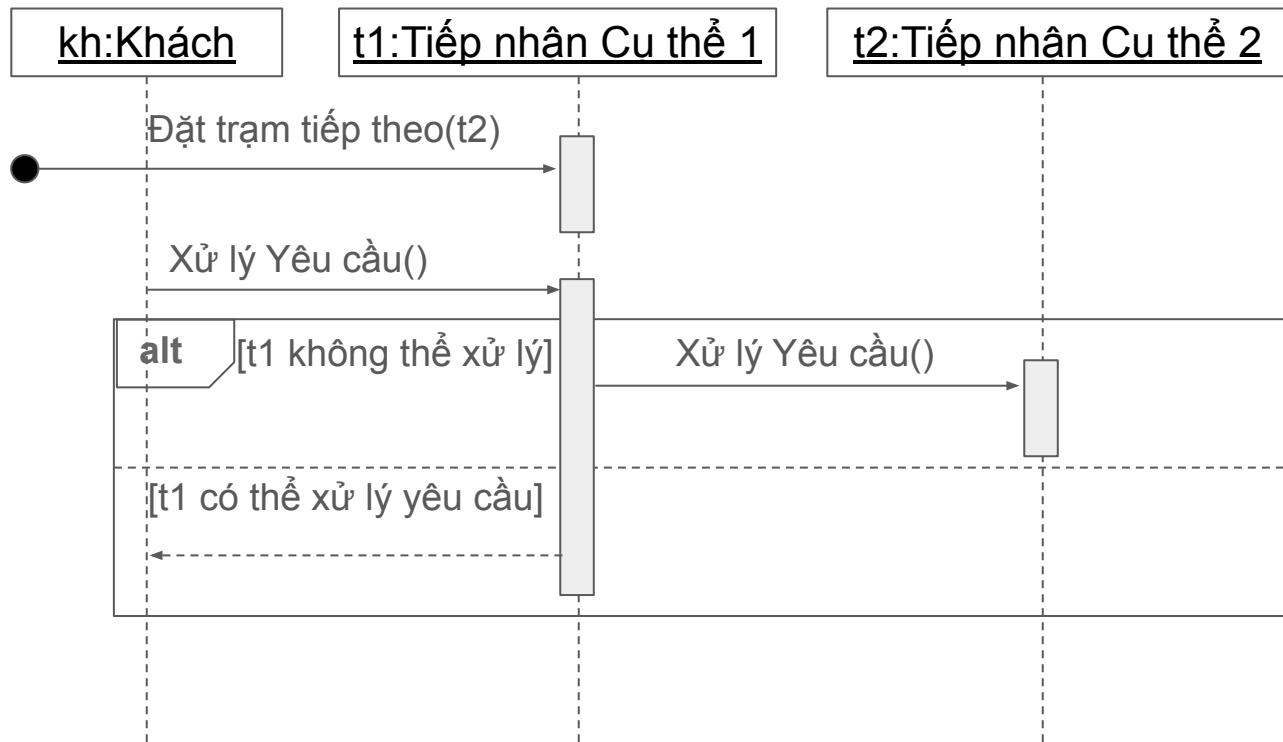
*Chain of Responsibility*

Tránh kết dính đối tượng gửi yêu cầu và đối tượng tiếp nhận nó bằng cách cho nhiều đối tượng có cơ hội xử lý yêu cầu. Xâu chuỗi các đối tượng tiếp nhận và truyền yêu cầu theo chuỗi cho tới khi có đối tượng xử lý nó.

# Chuỗi trách nhiệm: Cấu trúc



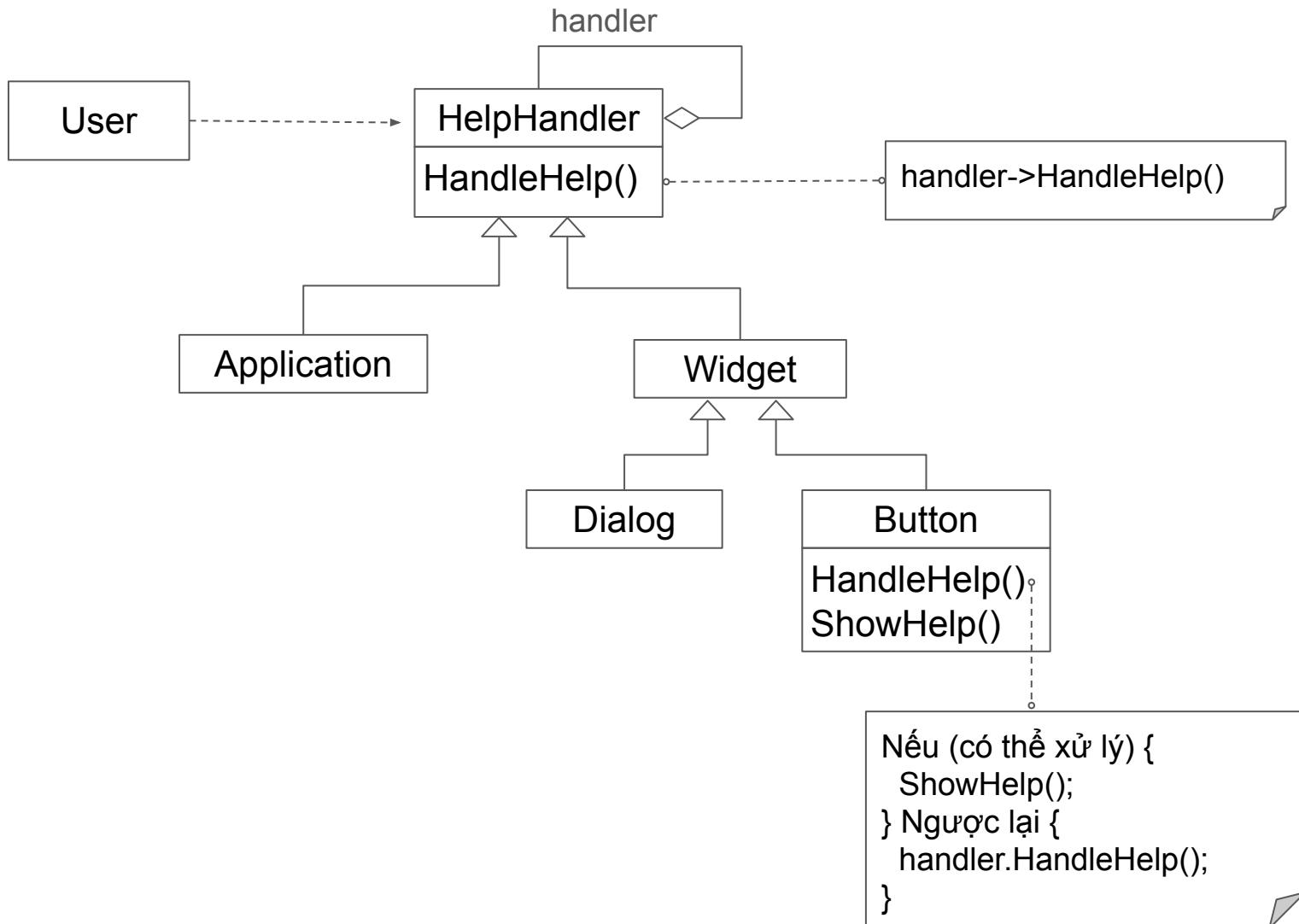
# Chuỗi trách nhiệm: Hành vi



# Chuỗi trách nhiệm: Các hệ quả

- Đối tượng gửi không biết đối tượng xử lý thông điệp và ngược lại.
- Tăng tính linh động khi gán các trách nhiệm cho các đối tượng.
- Xử lý thông điệp không được đảm bảo. Thông điệp có thể vẫn chưa được xử lý sau khi kết thúc chuỗi.

# Ví dụ 16. Xử lý yêu cầu trợ giúp

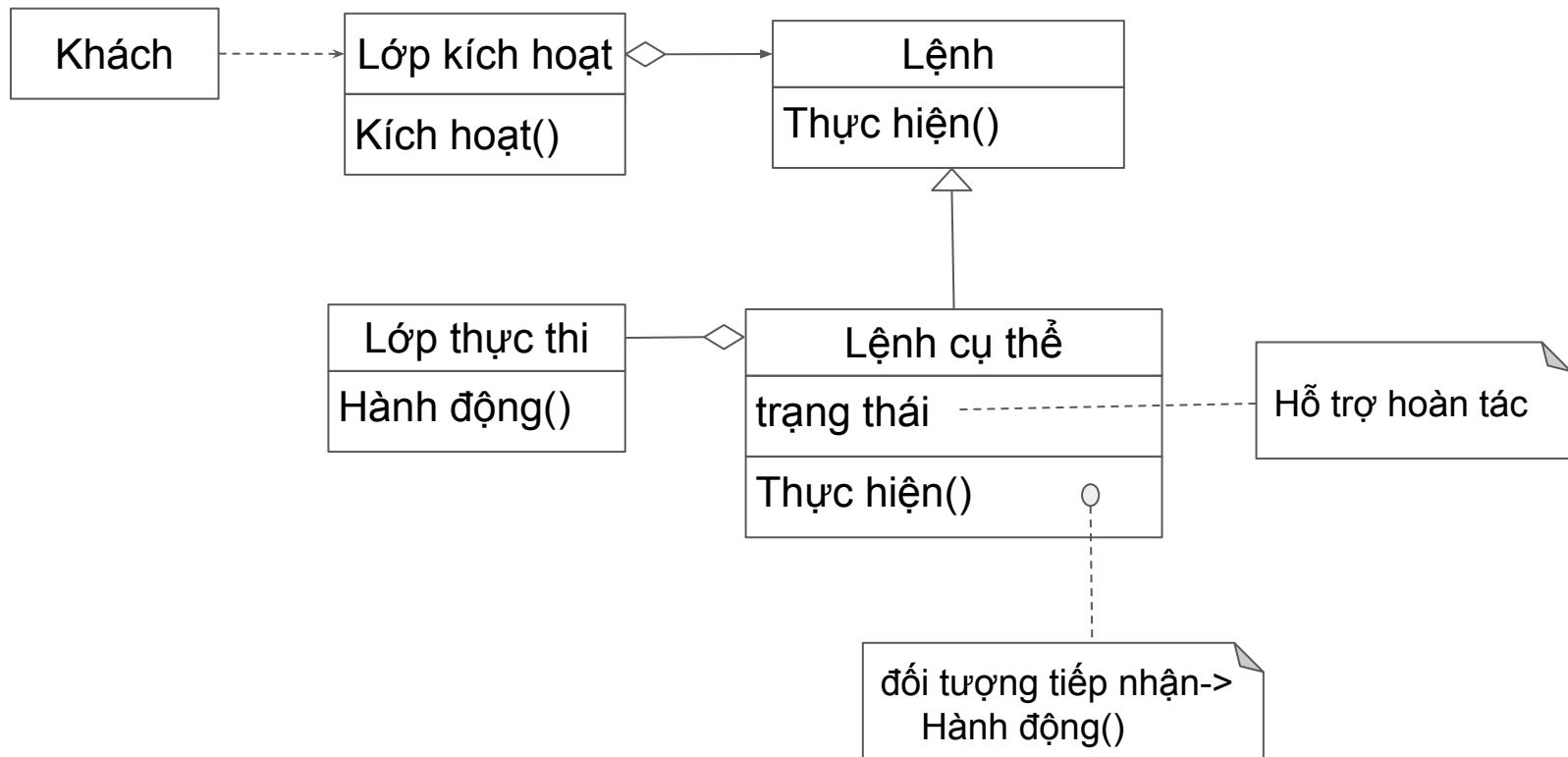


# Lệnh

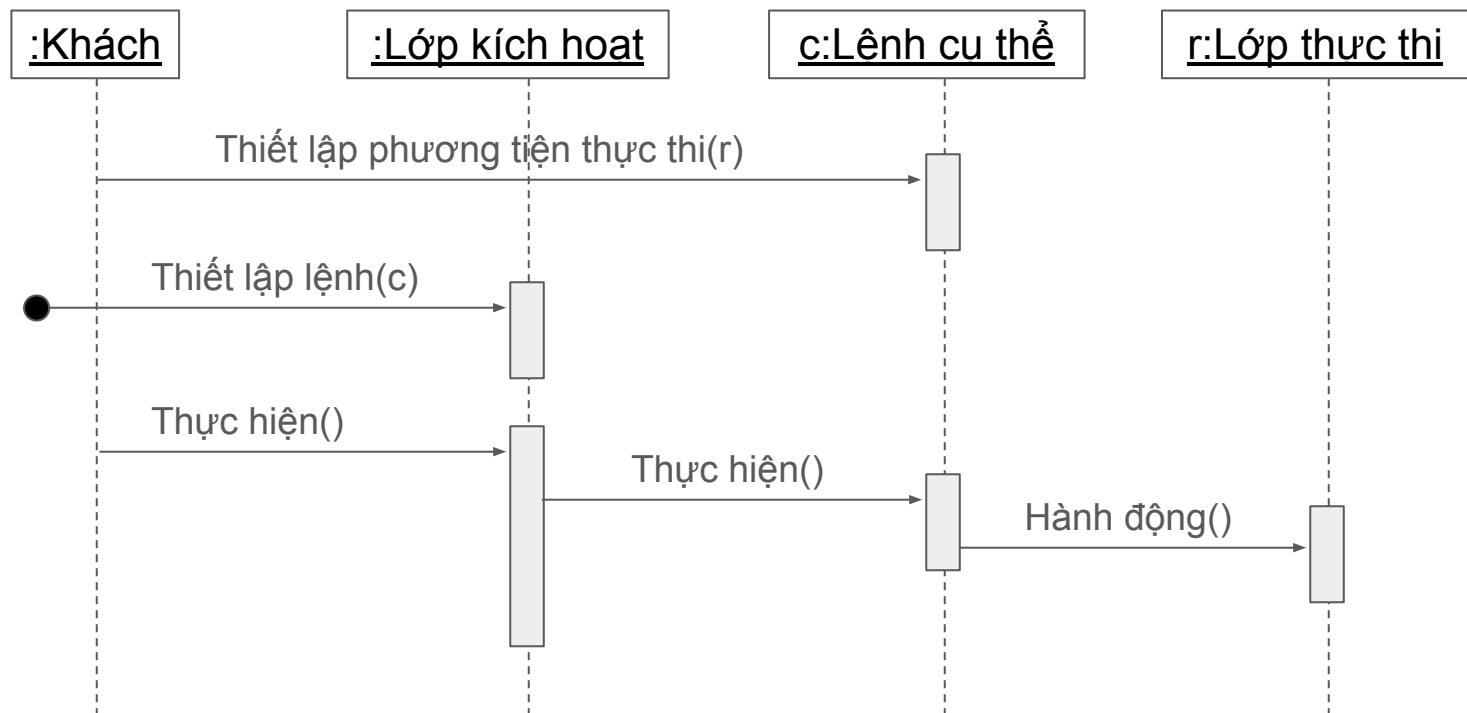
*Command*

Đóng gói yêu cầu như một đối tượng, qua đó có thể tùy chỉnh phía khách với các yêu cầu khác nhau, thiết lập hàng đợi hoặc ghi lịch sử yêu cầu, và hỗ trợ hoàn tác.

# Lệnh: Cấu trúc



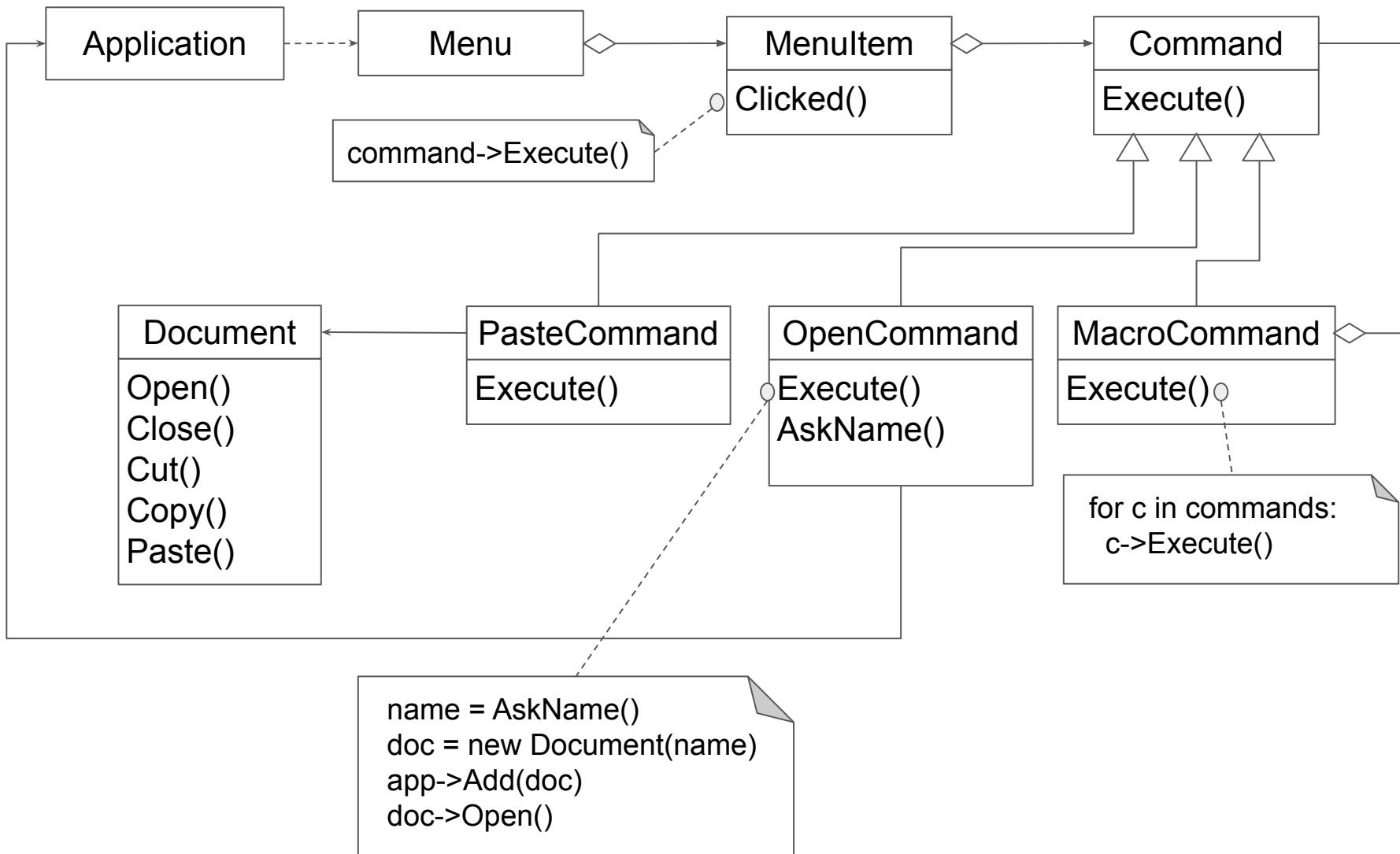
# Lệnh: Hành vi



# Lệnh: Các hệ quả

- Lệnh phân tách đối tượng kích hoạt thao tác và đối tượng biết cách thực hiện thao tác.
- Lệnh được đóng gói trong đối tượng, có thể được hiệu chỉnh và mở rộng như những đối tượng khác.
- Có thể lắp ráp các lệnh thành tổ hợp lệnh. Lệnh kết hợp là một trường hợp của mẫu hợp chất.
- Dễ bổ sung lệnh mới vì không cần thay đổi các lớp đang có.

# Ví dụ 17. Menu chương trình

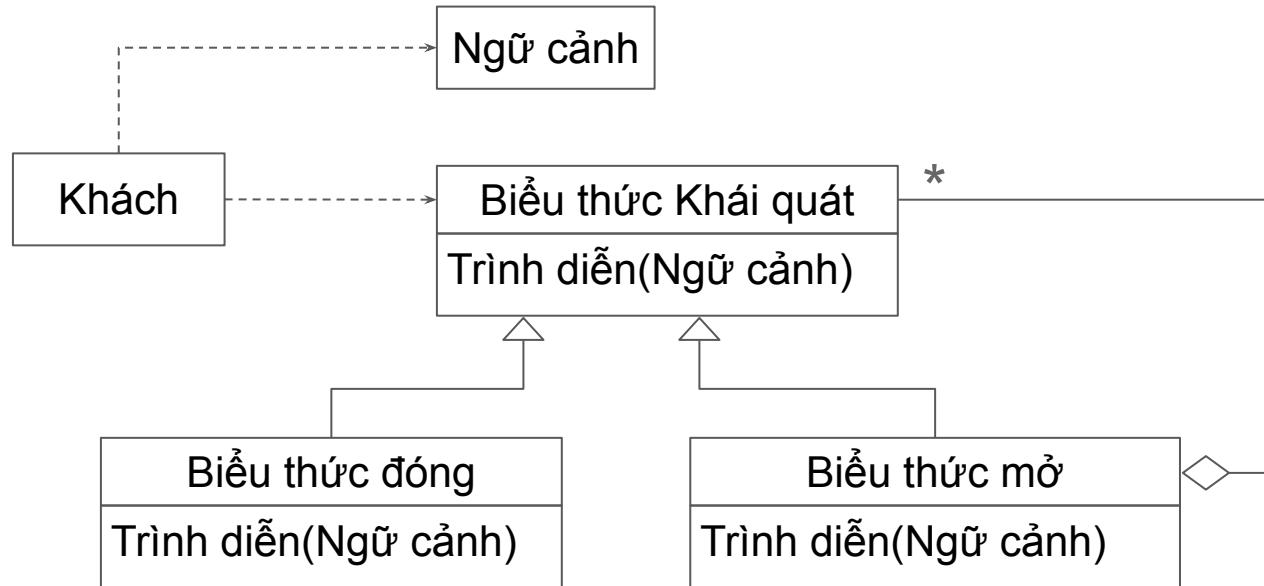


# Lớp diễn dịch

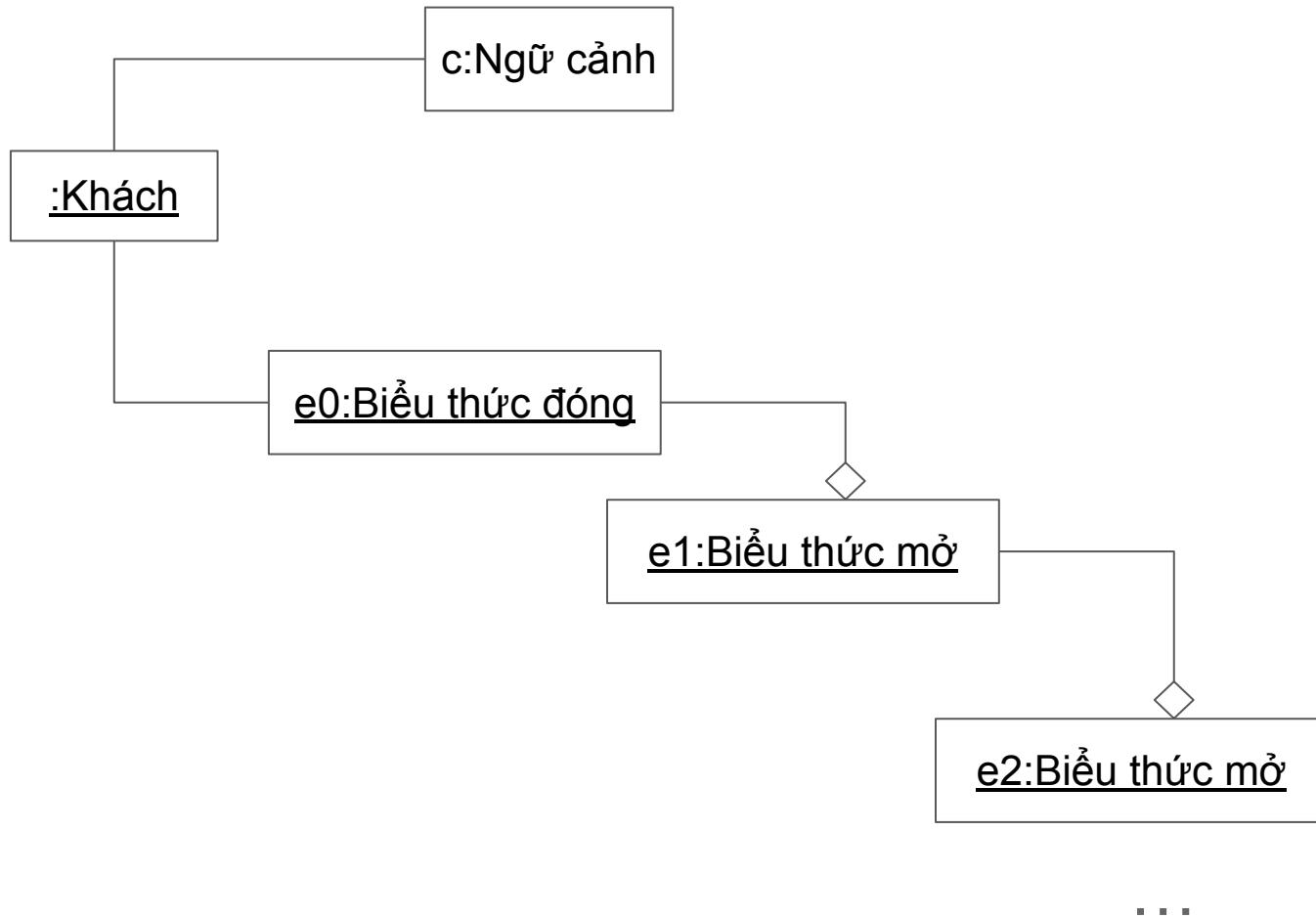
*Interpreter*

Cho một ngôn ngữ, định nghĩa một biểu diễn ngữ pháp của nó cùng với một trình diễn dịch sử dụng biểu diễn đó để trình diễn các câu trong ngôn ngữ.

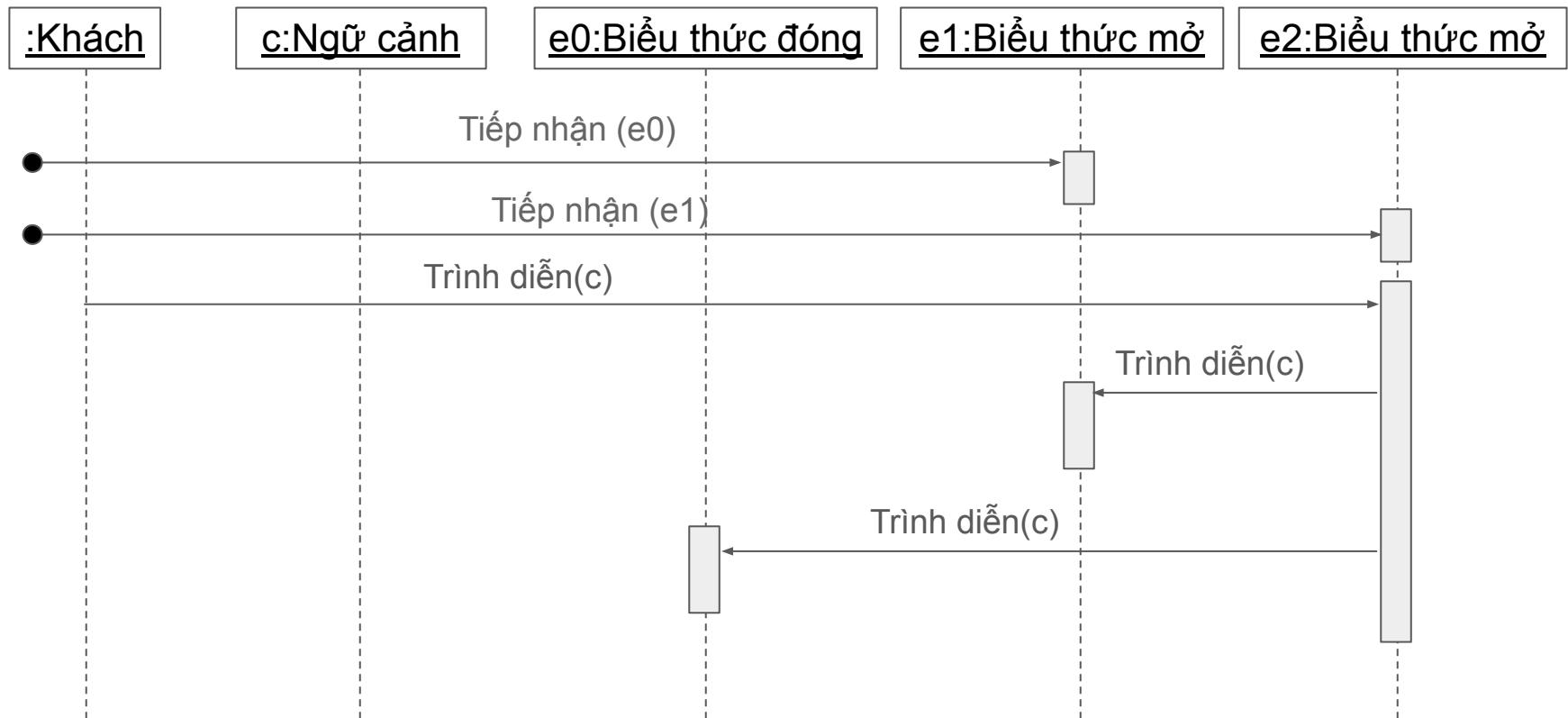
# Lớp diễn dịch: Cấu trúc



# Lớp diễn dịch: Kết hợp đối tượng



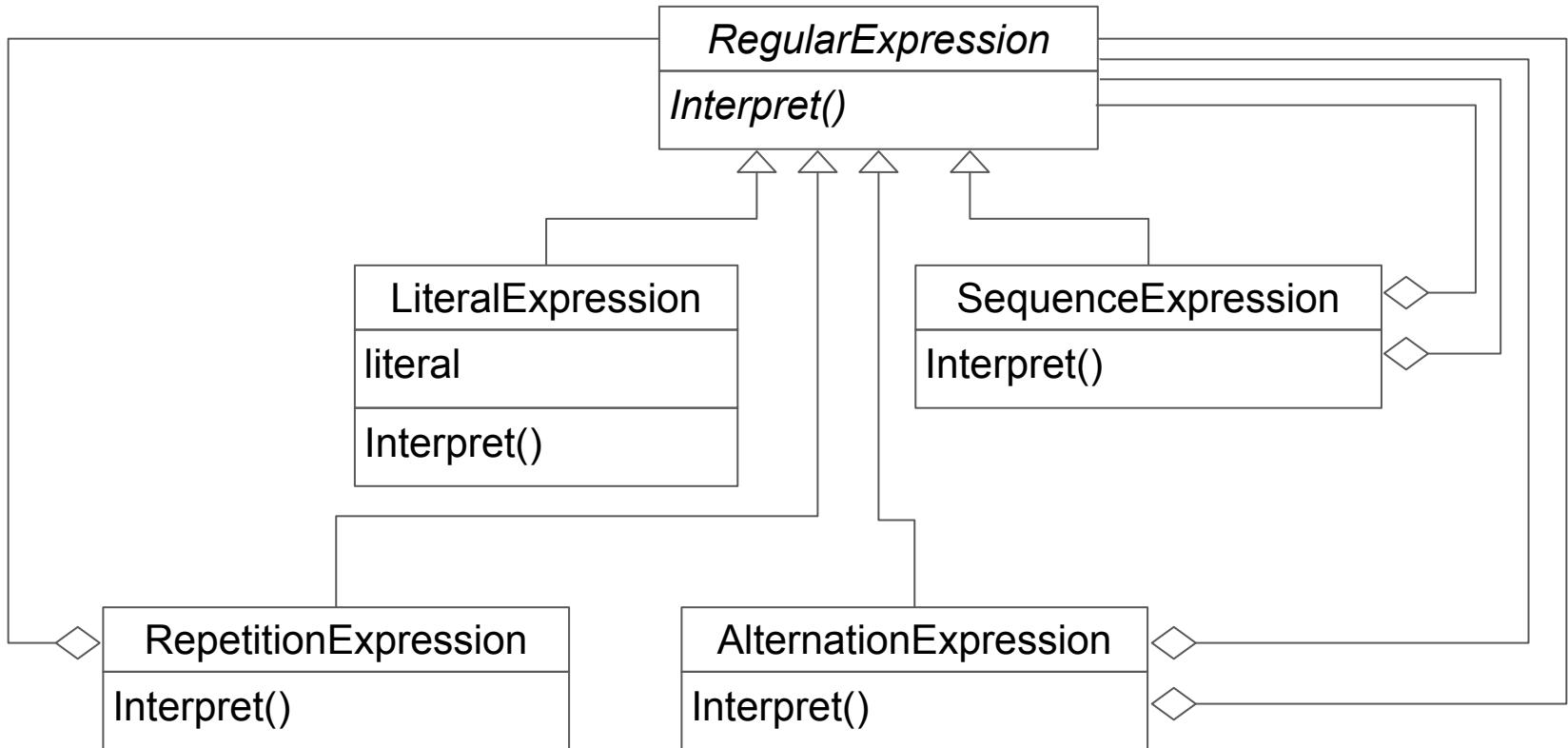
# Lớp diễn dịch: Hành vi



# Lớp diễn dịch: Các hệ quả

- Dễ thay đổi và mở rộng ngũ pháp. Các biểu thức đang tồn tại có thể được thay đổi dần, và các biểu thức mới có thể được định nghĩa bằng cách thay đổi các biểu thức đang có.
- Dễ triển khai ngũ pháp đơn giản.
- Khó duy trì ngũ pháp phức tạp.
- Cần thay đổi lớp biểu thức để thêm cách xử lý. Có thể sử dụng mẫu Khách thăm để tránh thay đổi các lớp.

# Ví dụ 18. Biểu thức chính quy

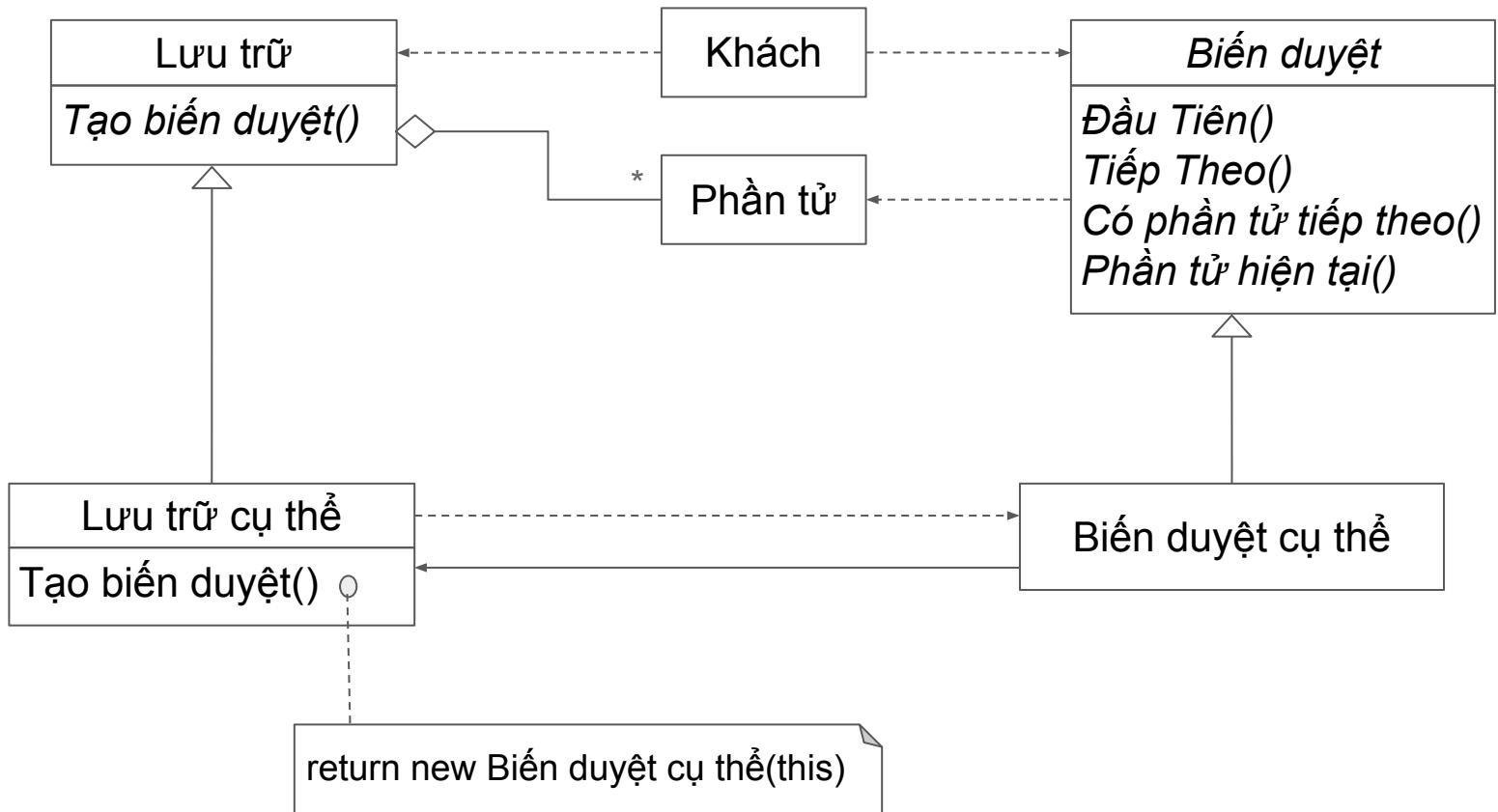


# Biến duyệt

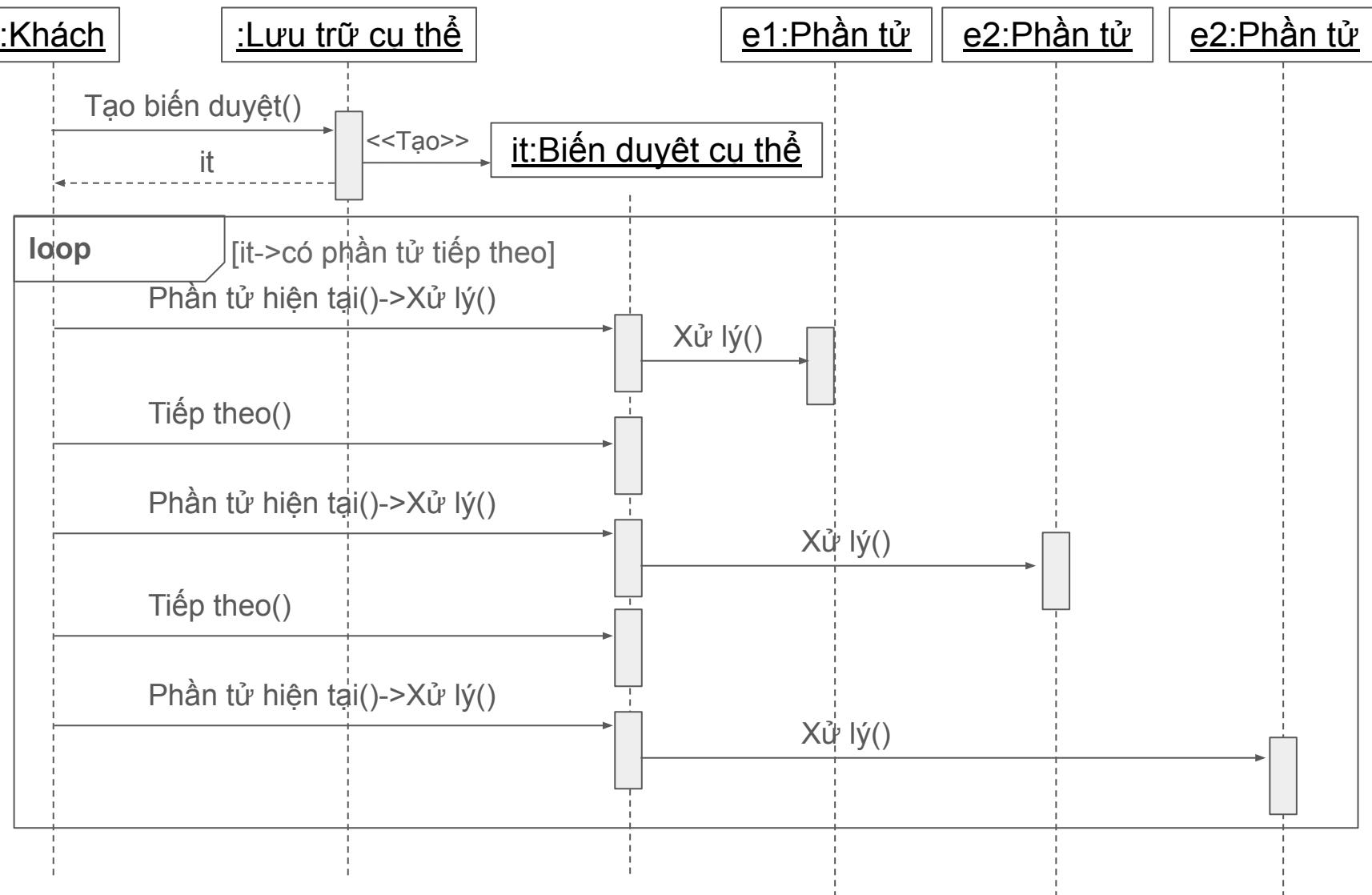
*Iterator*

Cung cấp một cách duyệt các phần tử trong lưu trữ theo thứ tự tuần tự, thống nhất, và độc lập với các biểu diễn.

# Biến duyệt: Cấu trúc



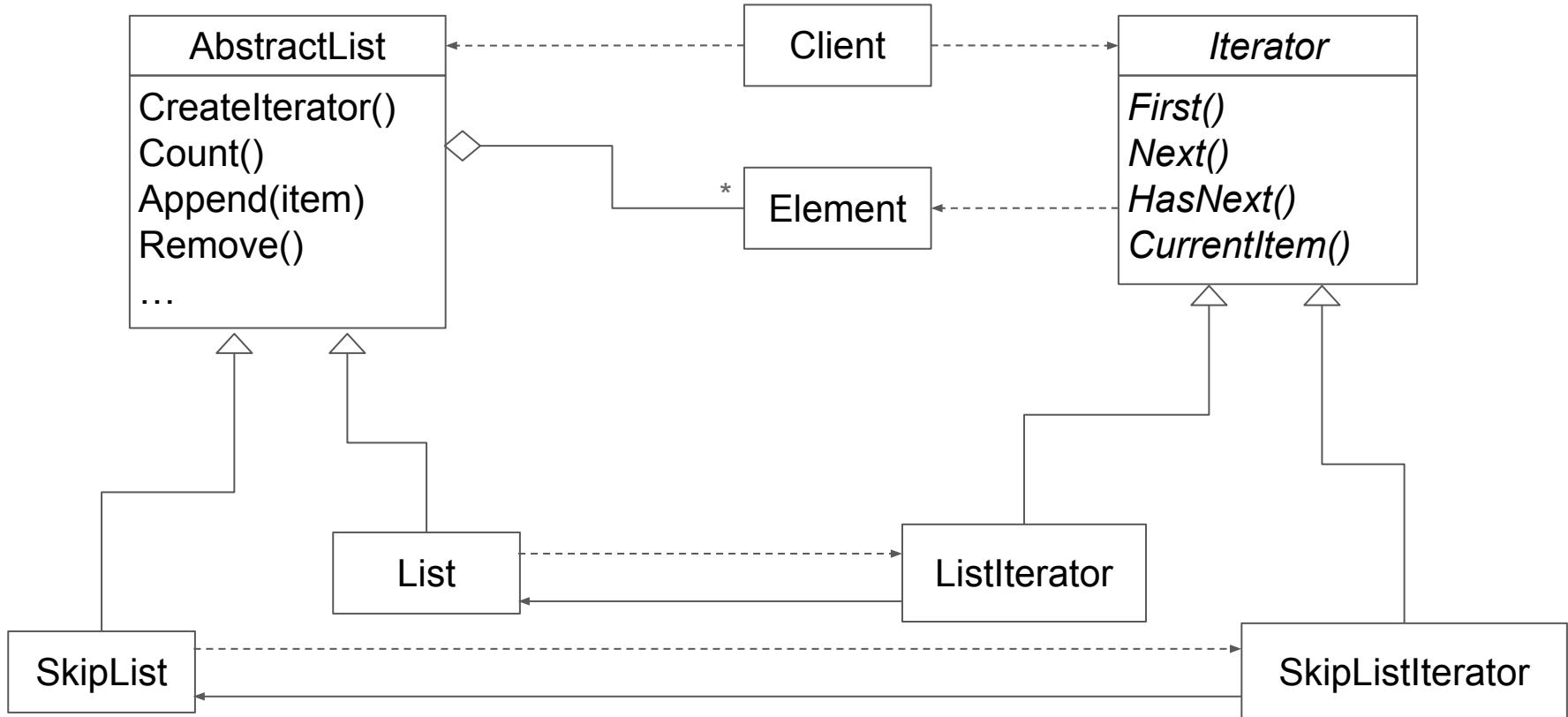
# Biến duyệt: Hành vi



# Biến duyệt: Các hệ quả

- Có thể có nhiều thứ tự duyệt đối với cấu trúc lưu trữ phức tạp, ví dụ cây nhị phân tìm kiếm. Có thể tạo nhiều triển khai biến duyệt cụ thể để duyệt theo các thứ tự khác nhau.
- Các biến duyệt làm đơn giản hóa giao diện lưu trữ.
- Các biến duyệt lưu trữ trạng thái của riêng nó, vì vậy có thể thực hiện nhiều tiến trình duyệt đồng thời với nhiều biến duyệt.

# Ví dụ 19. Cấu trúc lưu trữ

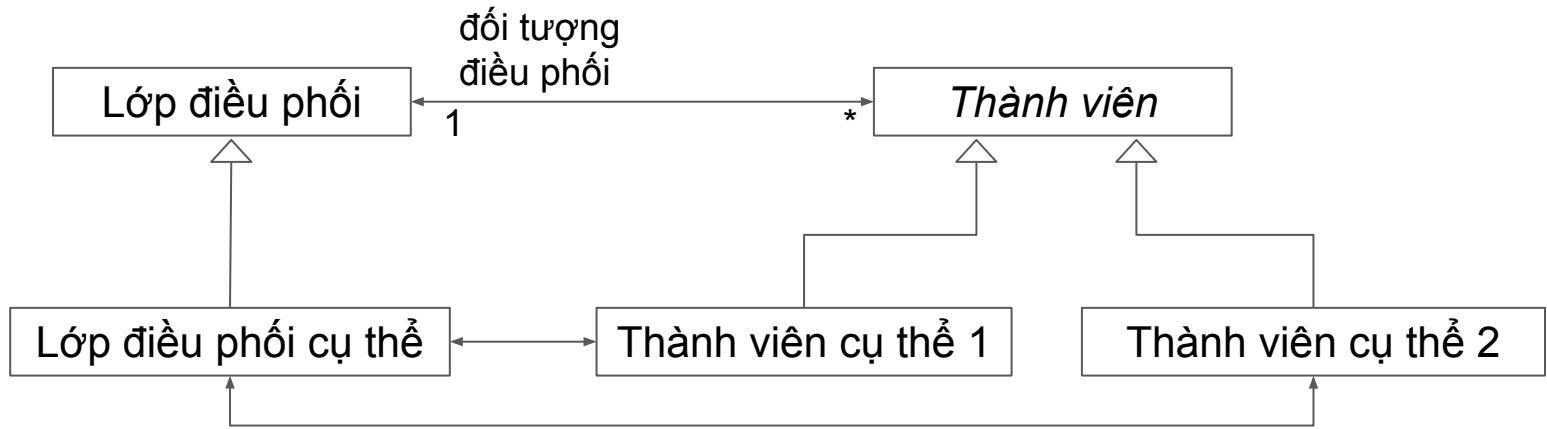


# Lớp điều phối

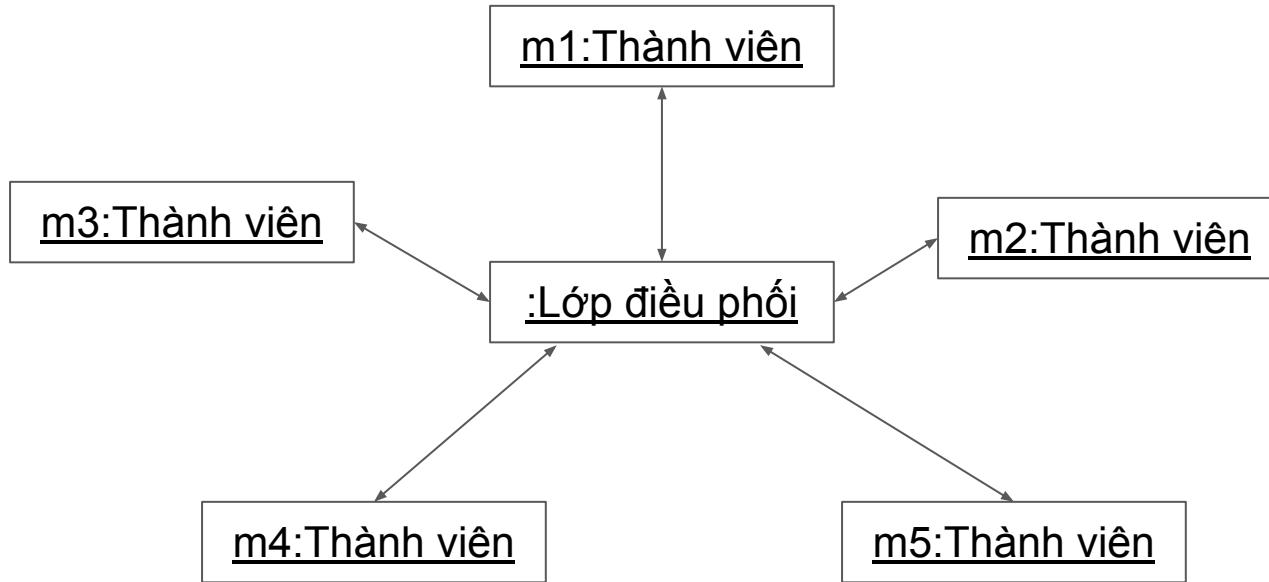
*Mediator*

Tạo một đối tượng đóng gói cách tương tác của một tập đối tượng. Mẫu lớp điều phối tạo ra liên kết lỏng bằng cách hạn chế tương tác trực tiếp giữa các đối tượng, và cho phép thay đổi tương tác của chúng độc lập.

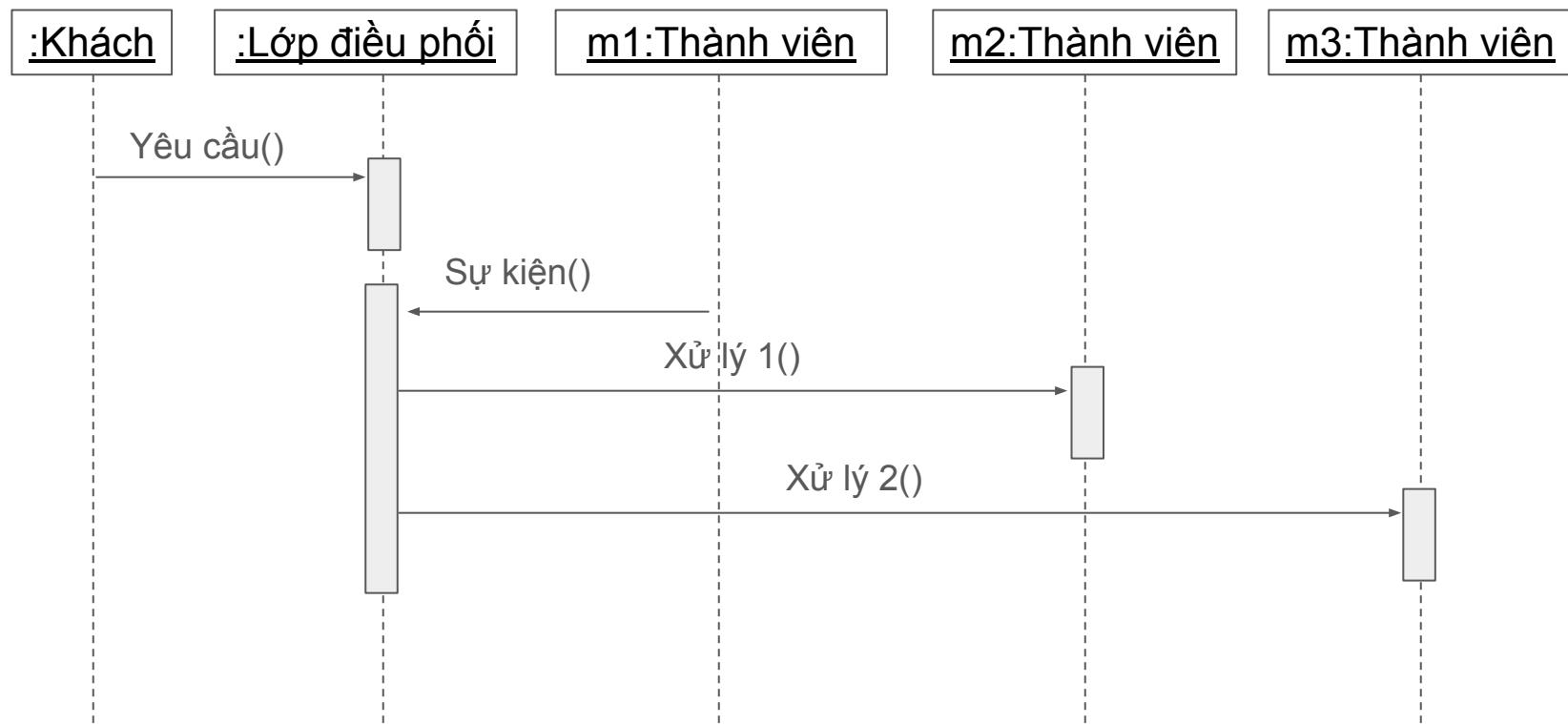
# Lớp điều phối: Cấu trúc



# Lớp điều phối: Kết hợp đối tượng



# Lớp điều phối: Hành vi

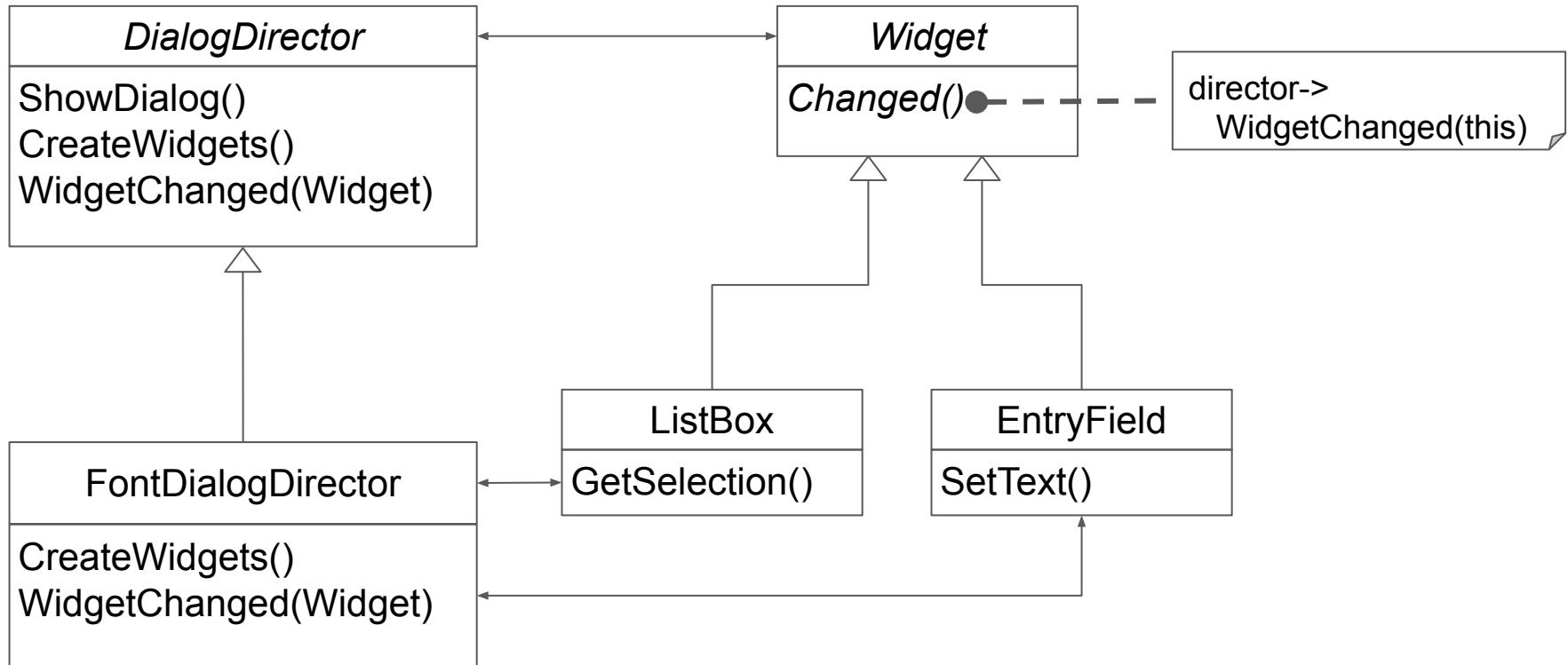


Các thành viên gửi và nhận các yêu cầu từ đối tượng điều phối. Đối tượng điều phối triển khai phối hợp bằng cách điều hướng các yêu cầu giữa các thành viên liên quan.

# Lớp điều phối: Các hệ quả

- Giảm thiểu kế thừa. Lớp điều phối tập hợp các hành vi bị phân tán giữa nhiều đối tượng. Chỉ cần kế thừa lớp điều phối để thay đổi hành vi.
- Tạo liên kết lỏng giữa các thành viên.
- Làm đơn giản hóa giao thức tương tác giữa các đối tượng. Thay thế tương tác n - n giữa các thành viên bằng tương tác 1 - n giữa đối tượng điều phối và các thành viên.
- Quản lý tập trung. Đánh đổi sự phức tạp tương tác và sự phức tạp của lớp điều phối. Lớp điều phối đóng gói các giao thức, vì vậy có thể phức tạp hơn các thành viên và khó quản lý.

# Ví dụ 20. Hộp thoại kiểu chữ

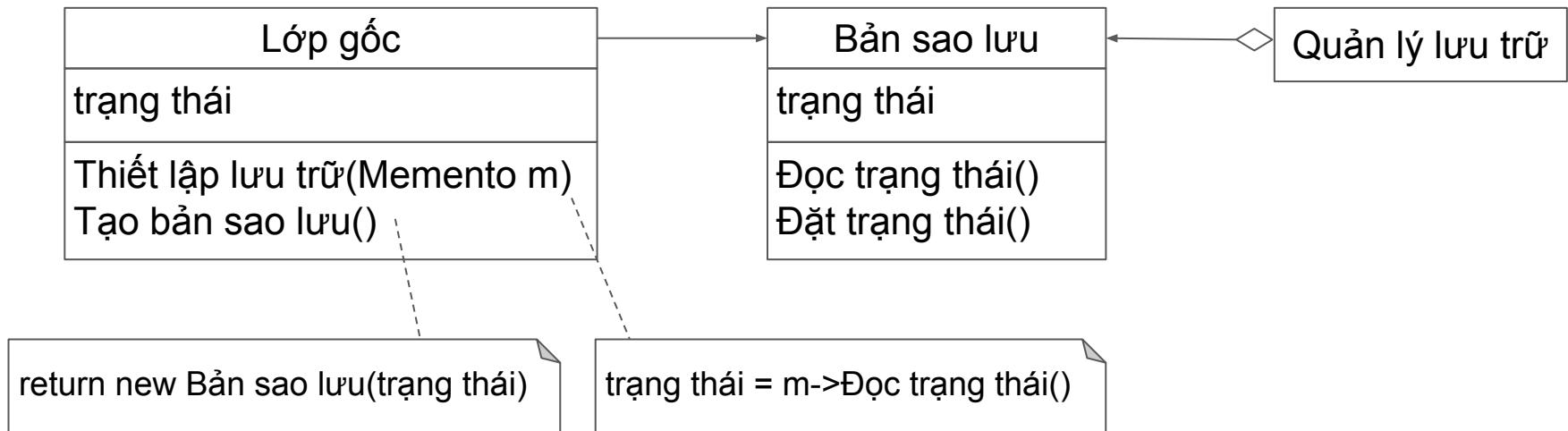


# Bản sao lưu

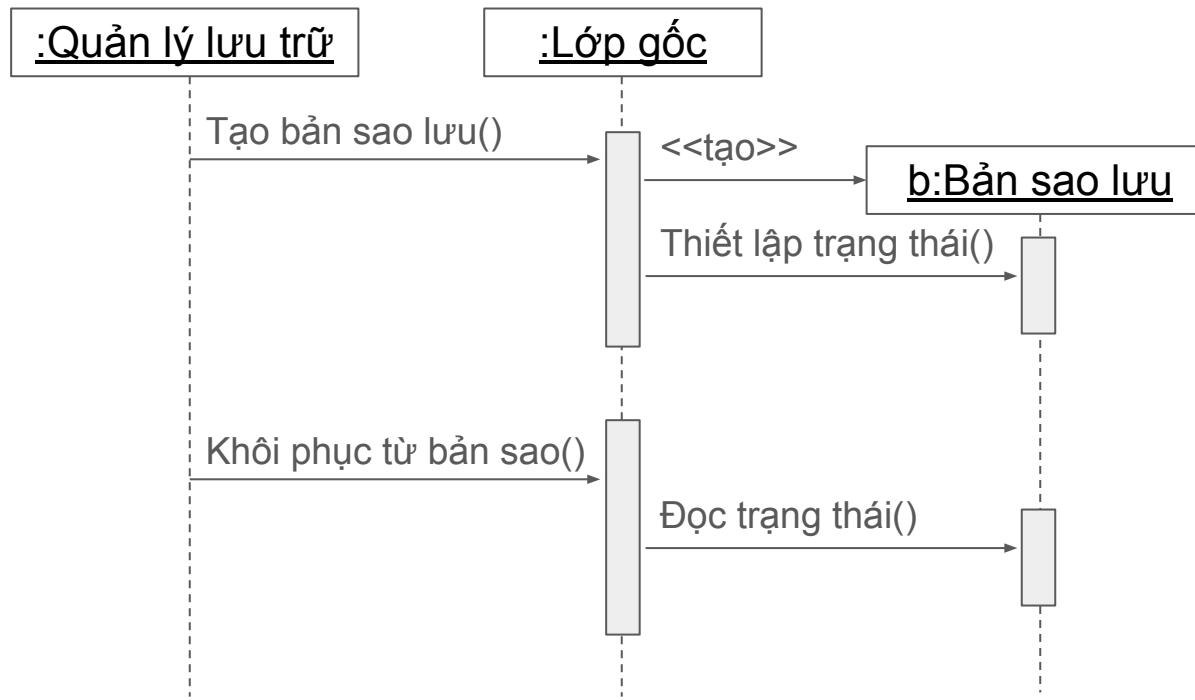
*Memento*

Thâu tóm và mở trạng thái của đối tượng cho bên ngoài để có thể khôi phục về trạng thái đó trong khi không phá vỡ quy tắc đóng gói.

# Bản sao lưu: Cấu trúc



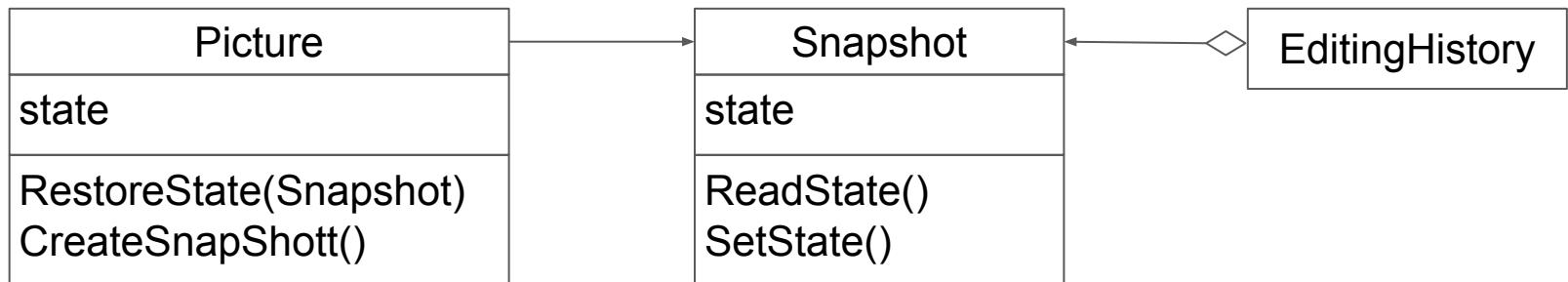
# Bản sao lưu: Hành vi



# Bản sao lưu: Các hệ quả

- Duy trì giới hạn đóng gói. Bản sao lưu tránh việc mở các thông tin được quản lý bởi lớp gốc nhưng phải được lưu ở nơi khác.
- Làm đơn giản đối tượng gốc. Để khách quản lý các trạng thái được yêu cầu của đối tượng gốc giúp đối tượng gốc đơn giản hơn.
- Sử dụng bản sao lưu có thể tốn kém nếu tốn nhiều tài nguyên để sao chép đối tượng gốc.
- Có thể khó giới hạn quyền truy cập trạng thái của bản sao lưu trong phạm vi đối tượng gốc do phụ thuộc vào ngôn ngữ triển khai.
- Có thể cần các tài nguyên bổ xung để triển khai kho lưu trữ.

# Ví dụ 21. Chỉnh sửa ảnh

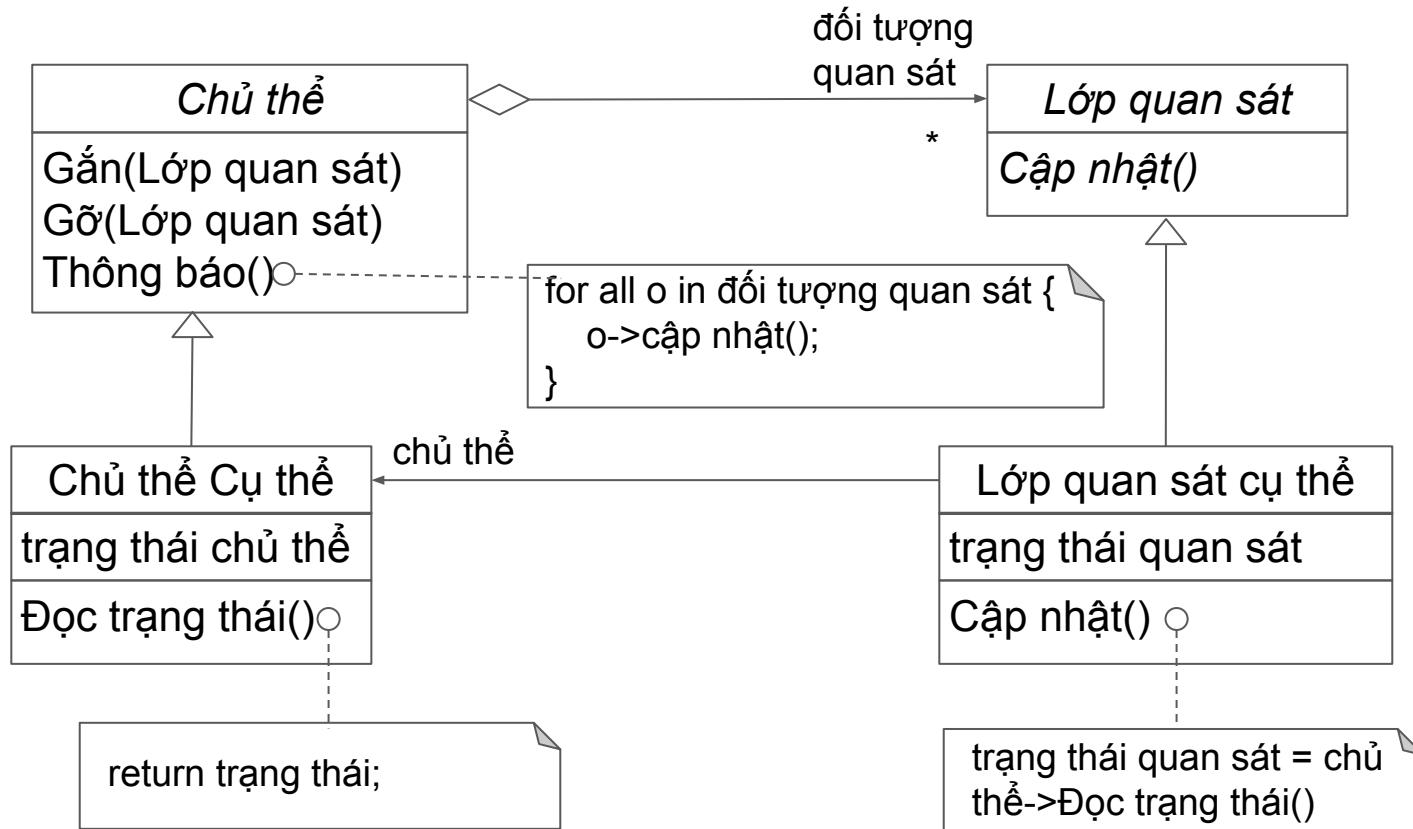


# Lớp quan sát

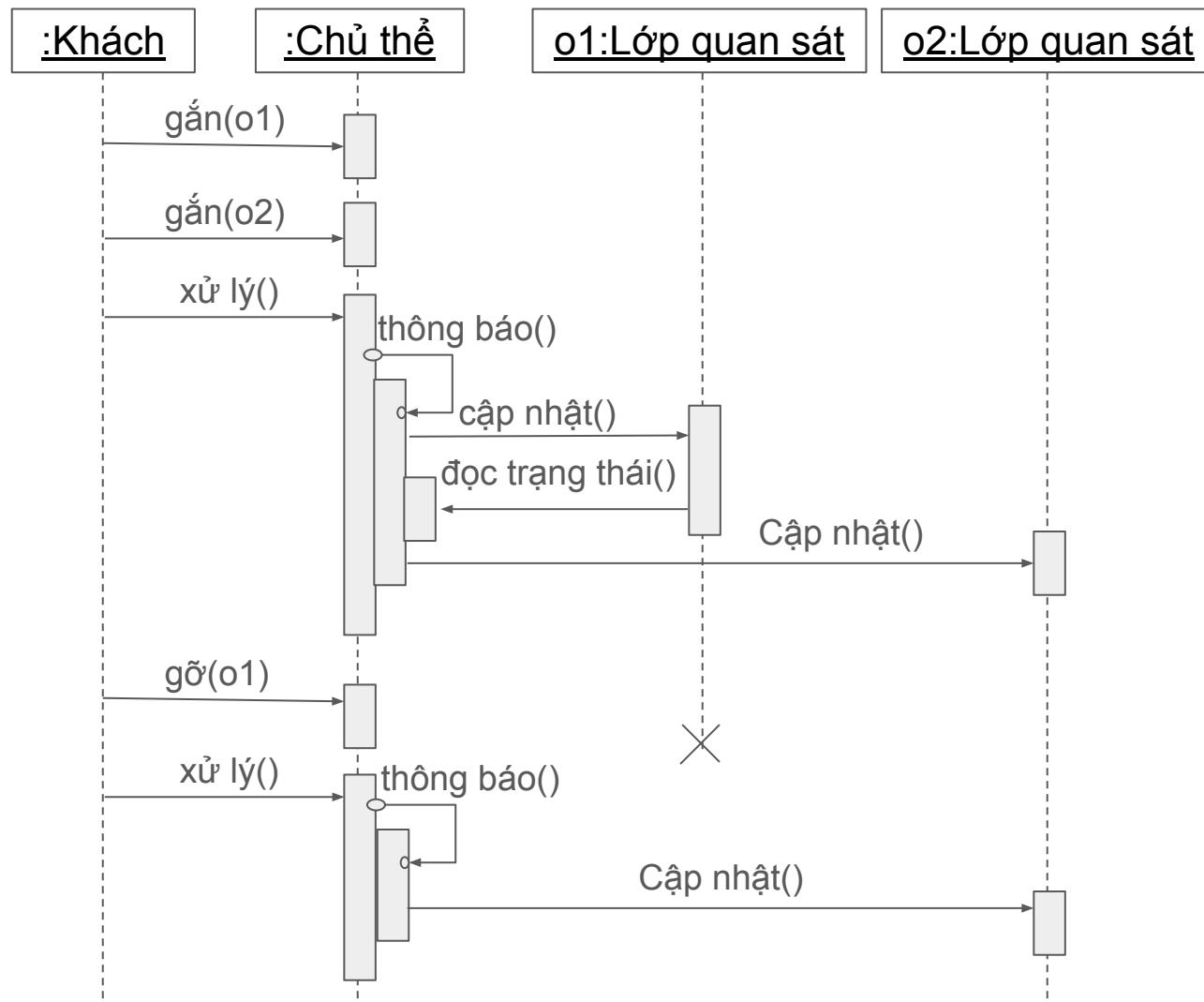
*Observer*

Thiết lập quan hệ một-nhiều giữa các đối tượng sao cho khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc vào nó được tự động thông báo và cập nhật.

# Lớp quan sát: Cấu trúc



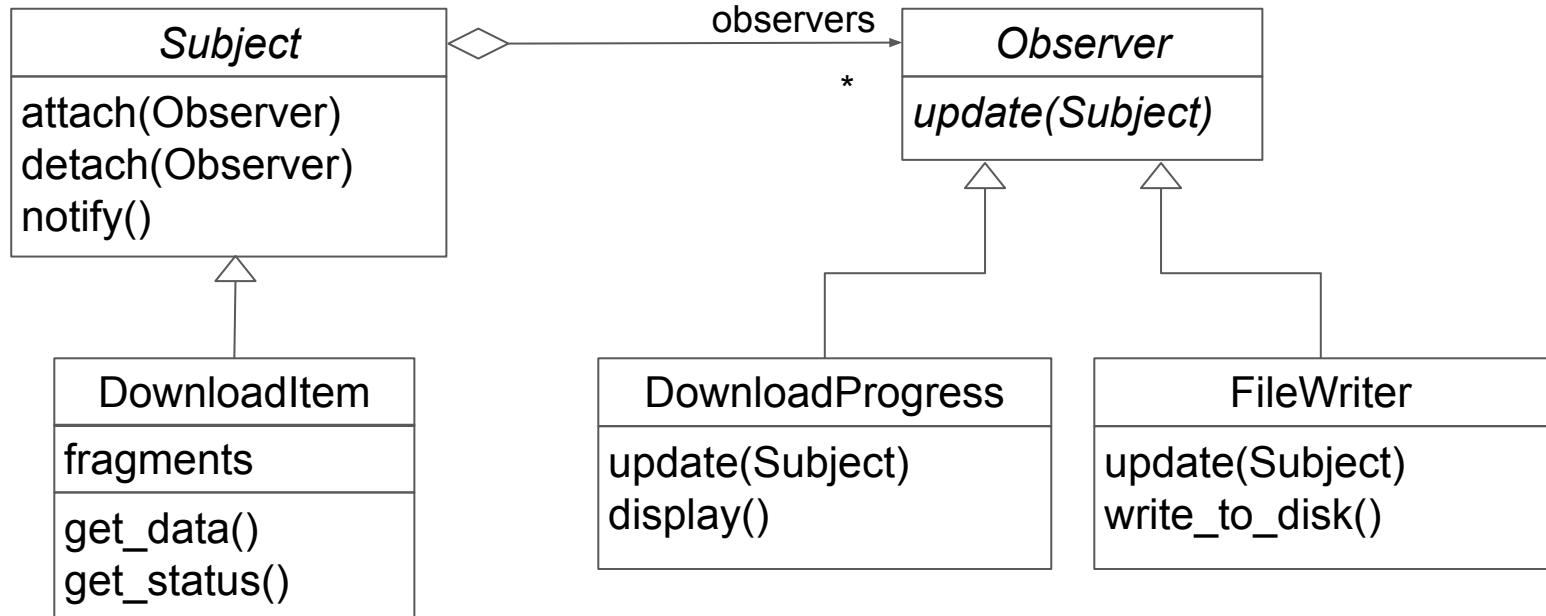
# Lớp quan sát: Hành vi



# Lớp quan sát: Các hệ quả

- Nói lỏng liên kết giữa chủ thể và đối tượng quan sát, vì vậy chủ thể và đối tượng quan sát có thể nằm trên các tầng khác nhau.
- Giao tiếp diện rộng, chủ thể chỉ có trách nhiệm thông báo đến các đối tượng quan sát đã đăng ký mà không cần quan tâm đó là những đối tượng gì. Vì vậy có thể tự do gắn và gỡ các đối tượng quan sát với chủ thể.

# Ví dụ 22. Ứng dụng tải tệp

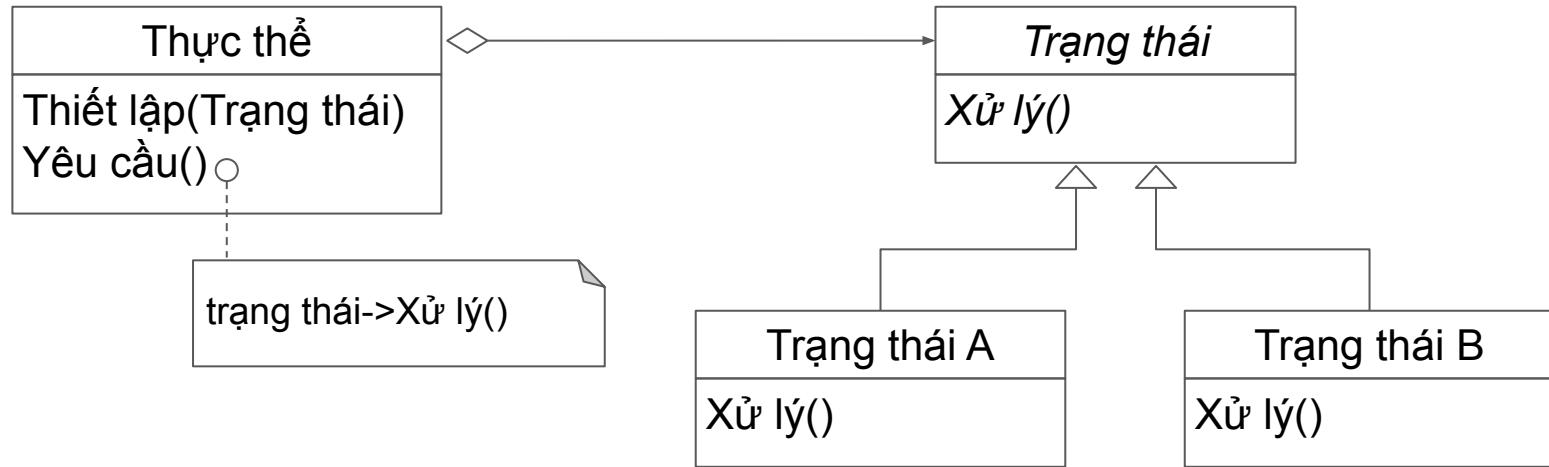


# Trạng thái

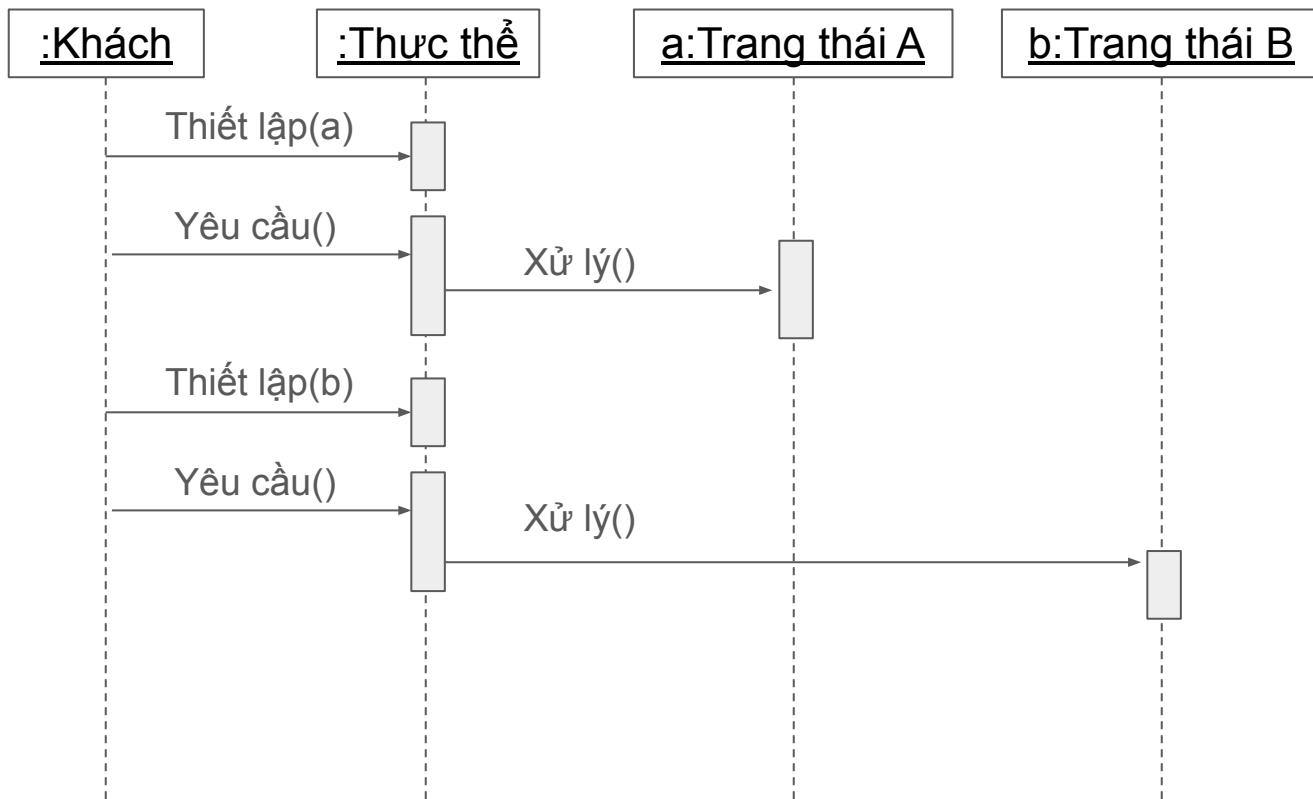
*State*

Cho phép một đối tượng thay đổi hành vi theo trạng thái của nó. Sau khi thay đổi trạng thái đối tượng có thể giống như thuộc một lớp khác.

# Trạng thái: Cấu trúc



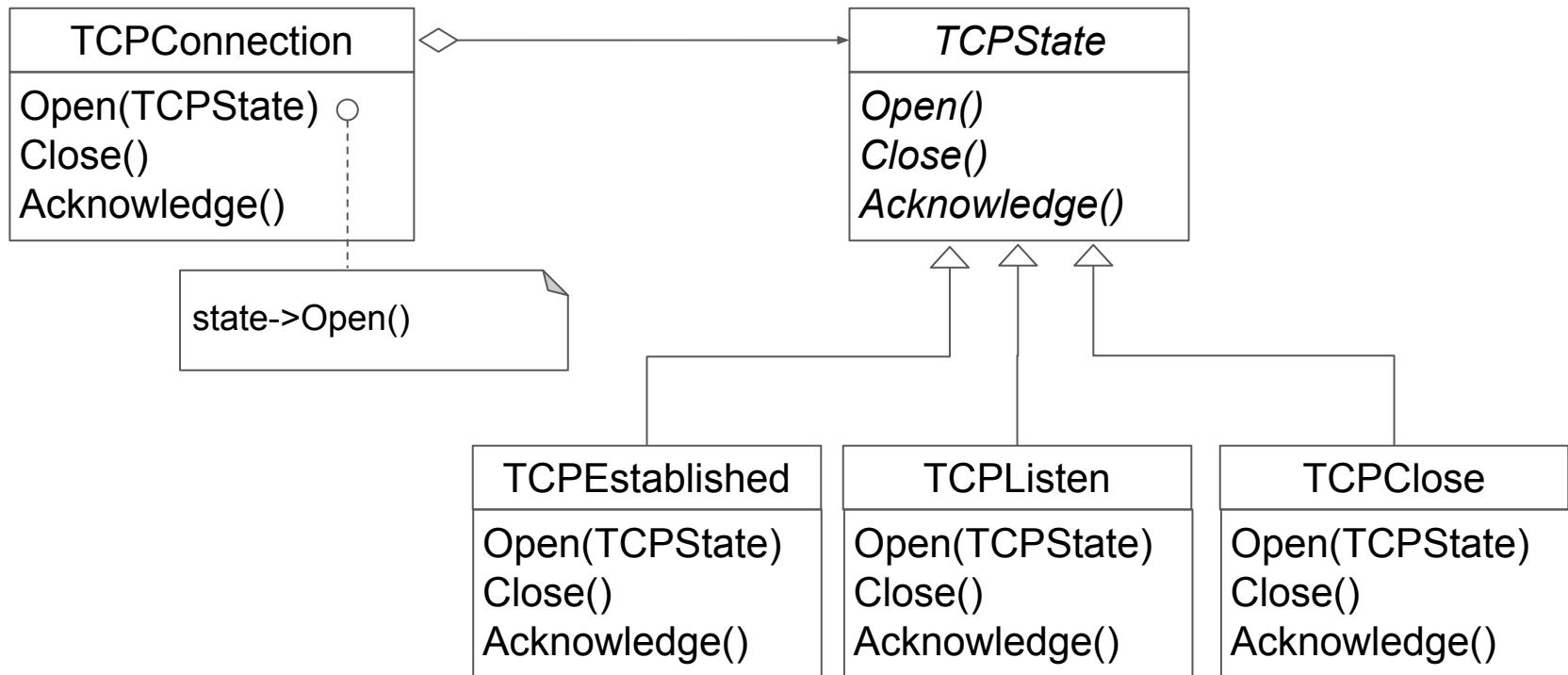
# Trạng thái: Hành vi



# Trạng thái: Các hệ quả

- Phân tách hành vi theo các trạng thái khác nhau. Các hành vi gắn với một trạng thái cụ thể được đóng gói trong một đối tượng riêng.
- Cụ thể hóa sự kiện chuyển trạng thái.
- Các thực thể có thể chia sẻ đối tượng trạng thái, tương tự mẫu thiết kế Lớp hạng ruồi.

# Ví dụ 23. Kết nối TCP-IP

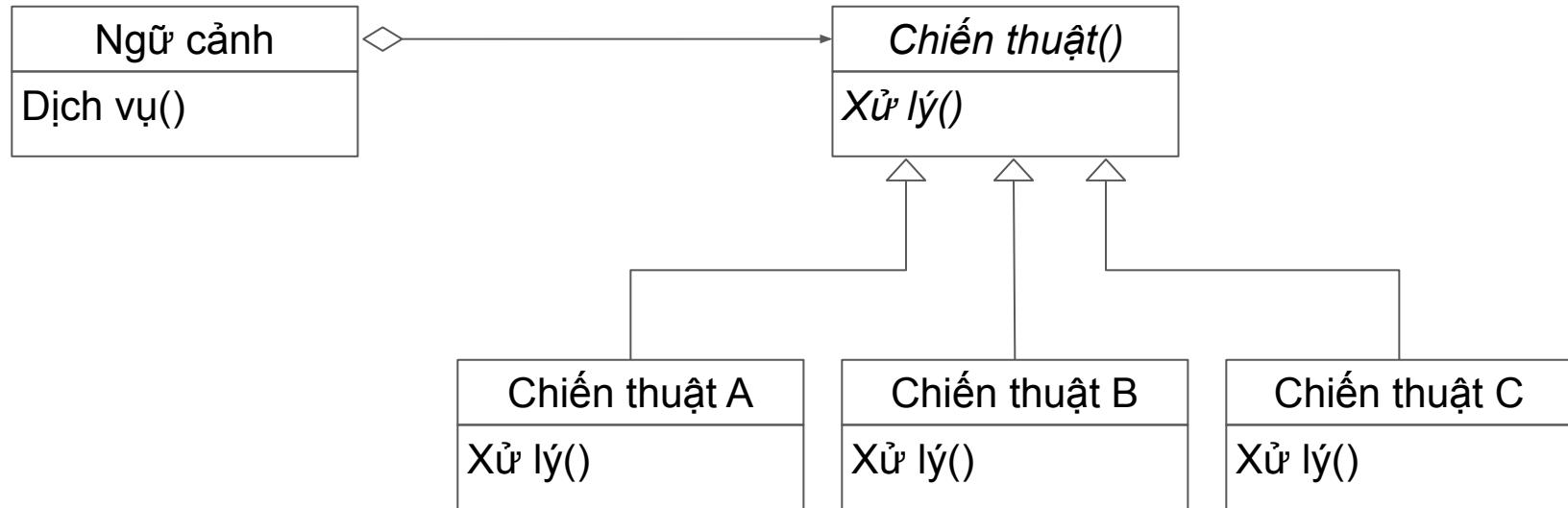


# Chiến thuật

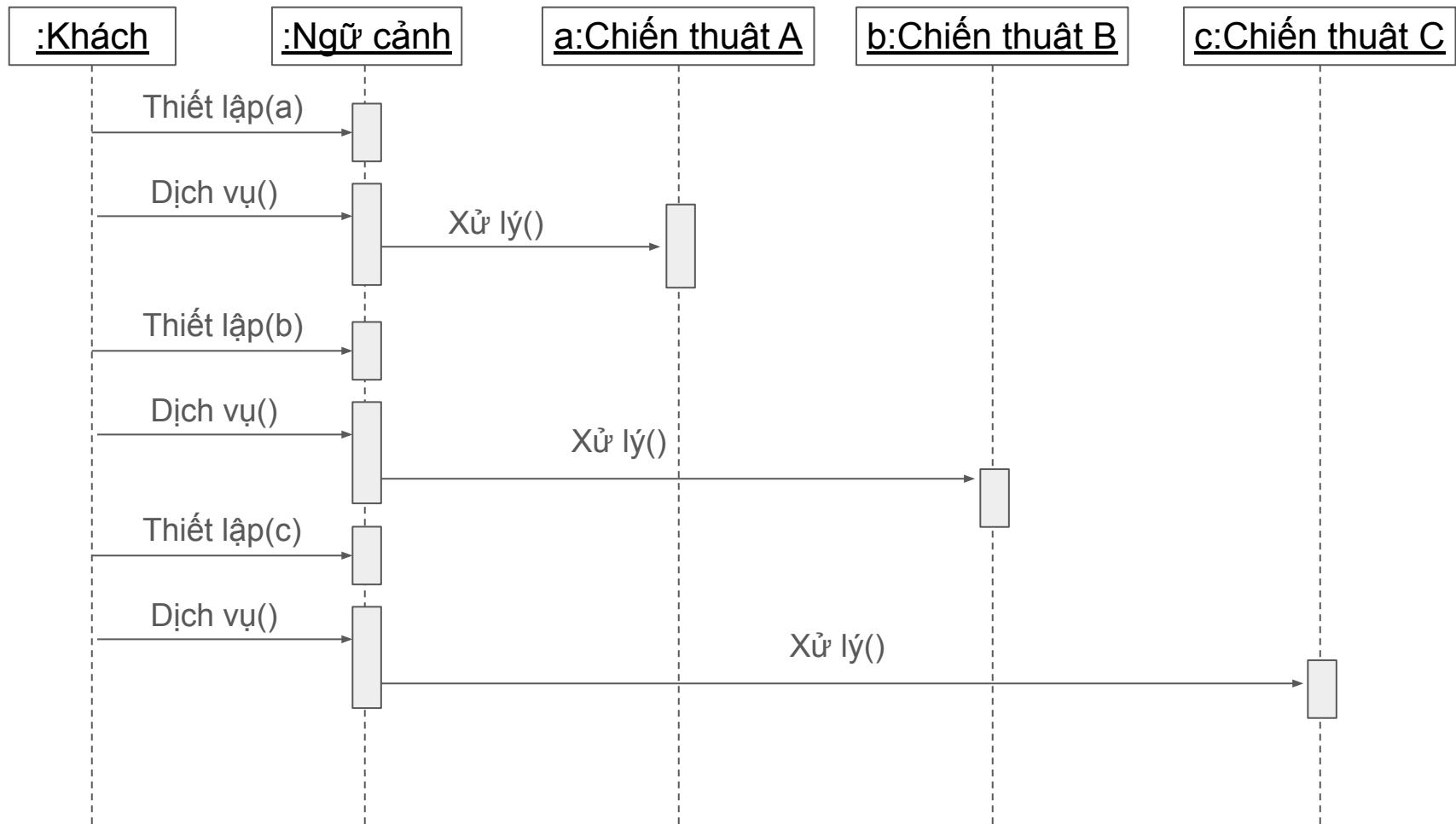
*Strategy*

Định nghĩa một cây giải thuật, đóng gói mỗi cái, và làm chúng khả thay. Chiến thuật cho phép giải thuật thay đổi độc lập so với phía sử dụng nó.

# Chiến thuật: Cấu trúc



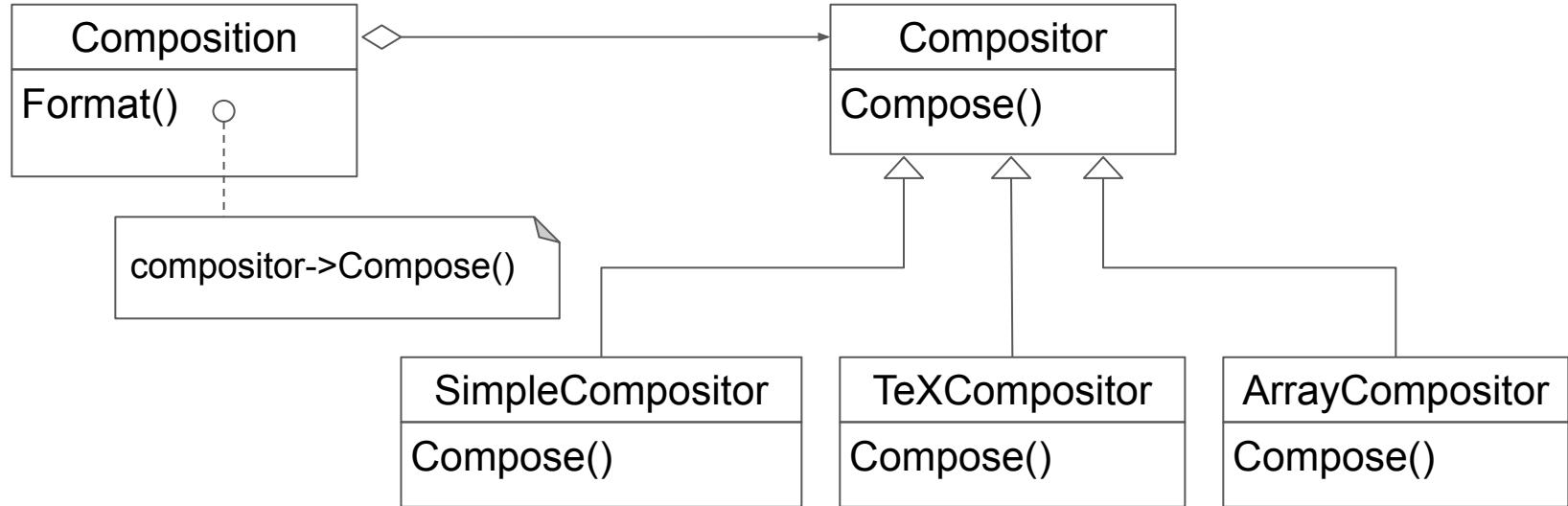
# Chiến thuật: Hành vi



# Chiến thuật: Các hệ quả

- Cây chiến thuật định nghĩa một hệ thuật toán để tái sử dụng theo ngũ cảnh.
- Đóng gói thuật toán trong lớp chiến thuật cho phép thay đổi thuật toán độc lập với ngũ cảnh của nó, giúp dễ hiểu và dễ mở rộng hơn.
- Giúp lược bớt lựa chọn rẽ nhánh. Cho phép cung cấp nhiều triển khai cho một hành vi. Phía khách cần hiểu các chiến thuật để thực hiện lựa chọn phù hợp.
- Các chiến thuật sử dụng chung một giao diện để tương tác với ngũ cảnh, vì vậy ngũ cảnh có thể phải cung cấp các tham số dư thừa cho một chiến thuật cụ thể.
- Làm tăng số lượng đối tượng trong hệ thống, tuy nhiên có thể triển khai các đối tượng không có trạng thái ngoại để chia sẻ, như mẫu thiết kế lớp hạng ruồi.

# Ví dụ 24. Tách dòng văn bản

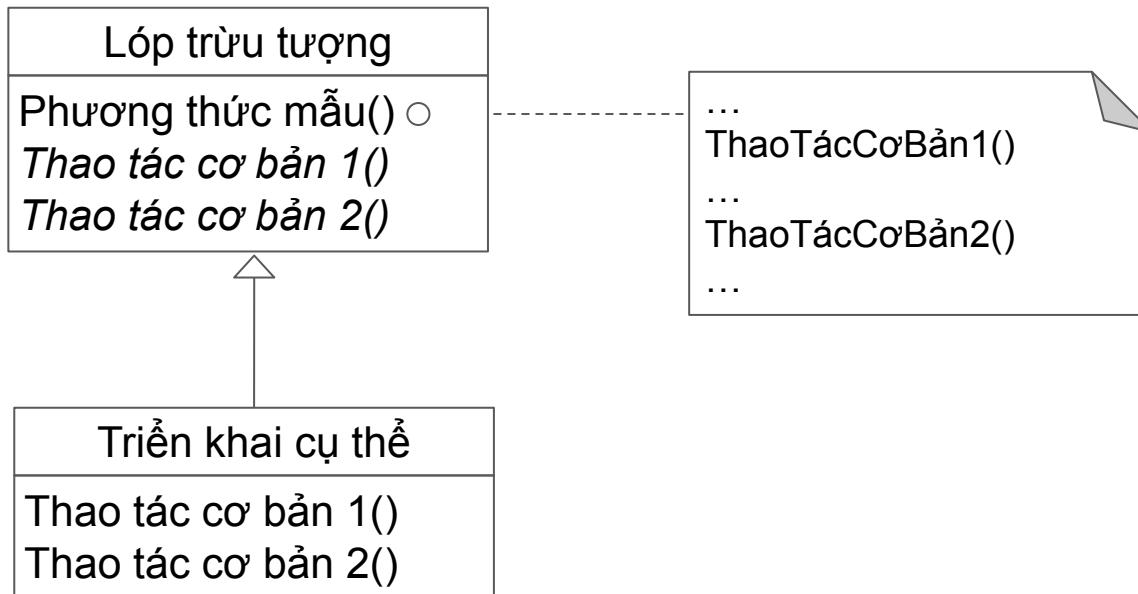


# Phương thức mẫu

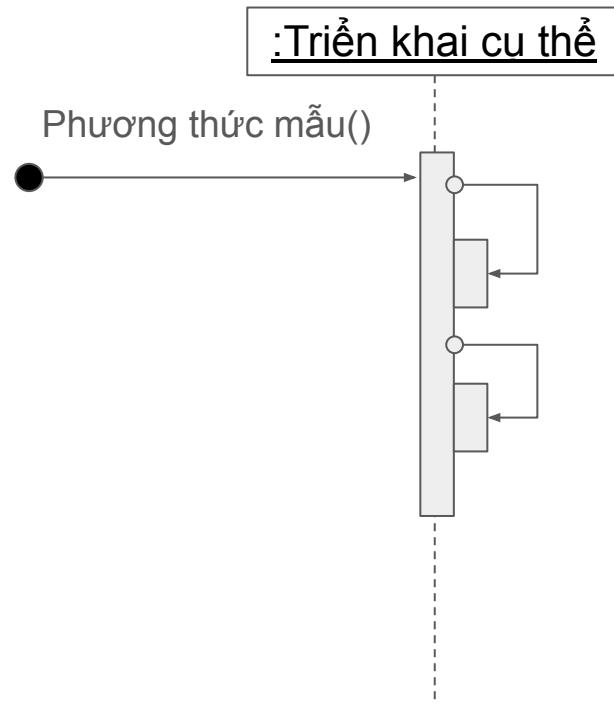
*Template Method*

Thiết lập khung giải thuật trong một phương thức, dời một số bước trong đó tới khi triển khai các lớp con. Phương thức mẫu cho phép các lớp con định nghĩa lại các bước cụ thể của một giải thuật mà không thay đổi cấu trúc giải thuật.

# Phương thức mẫu: Cấu trúc



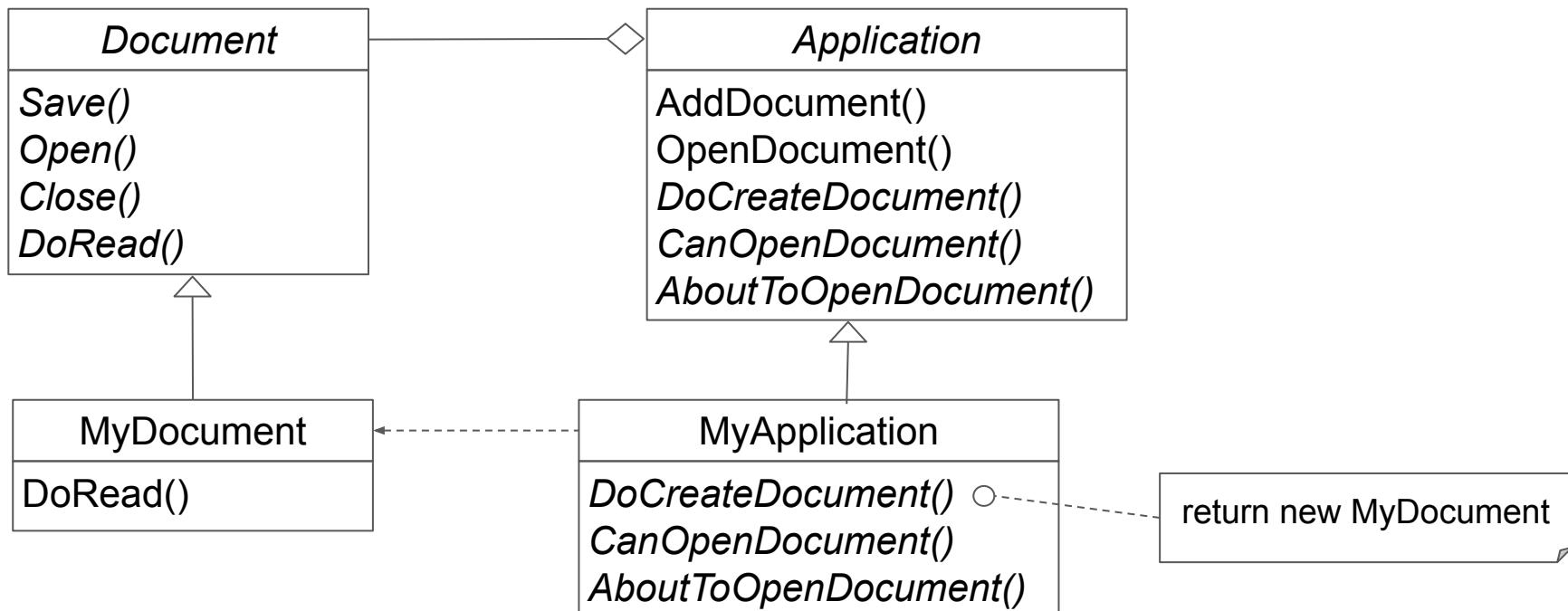
# Phương thức mẫu: Hành vi



# Phương thức mẫu: Các hệ quả

- Tách riêng hành vi chung, thường hữu ích đối với các lớp thư viện.
- Lớp dưới có thể mở rộng hành vi của lớp trên bằng cách kế thừa và định nghĩa lại thao tác.

# Ví dụ 25. Ứng dụng quản lý văn bản

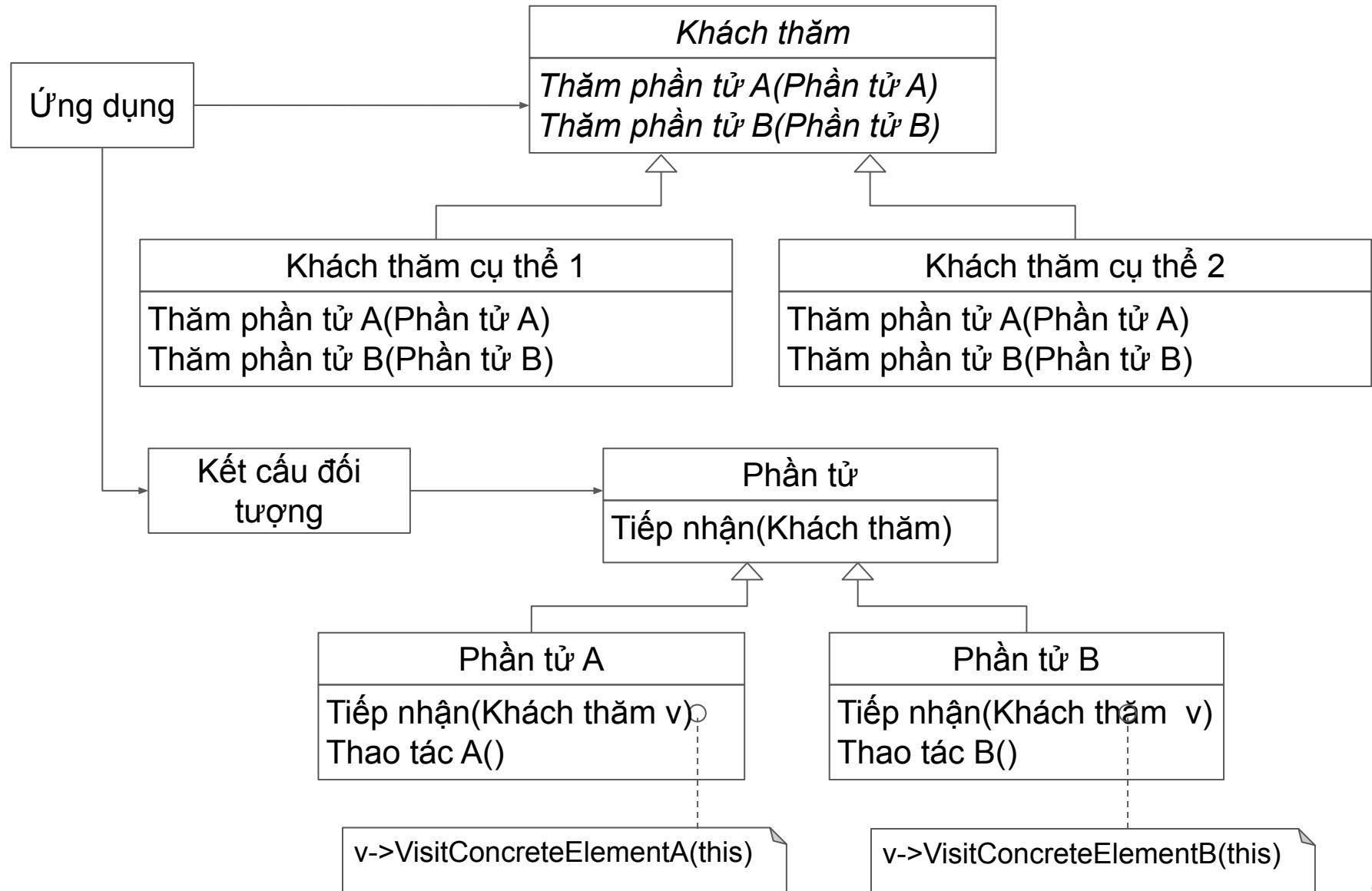


# Khách thăm

*Visitor*

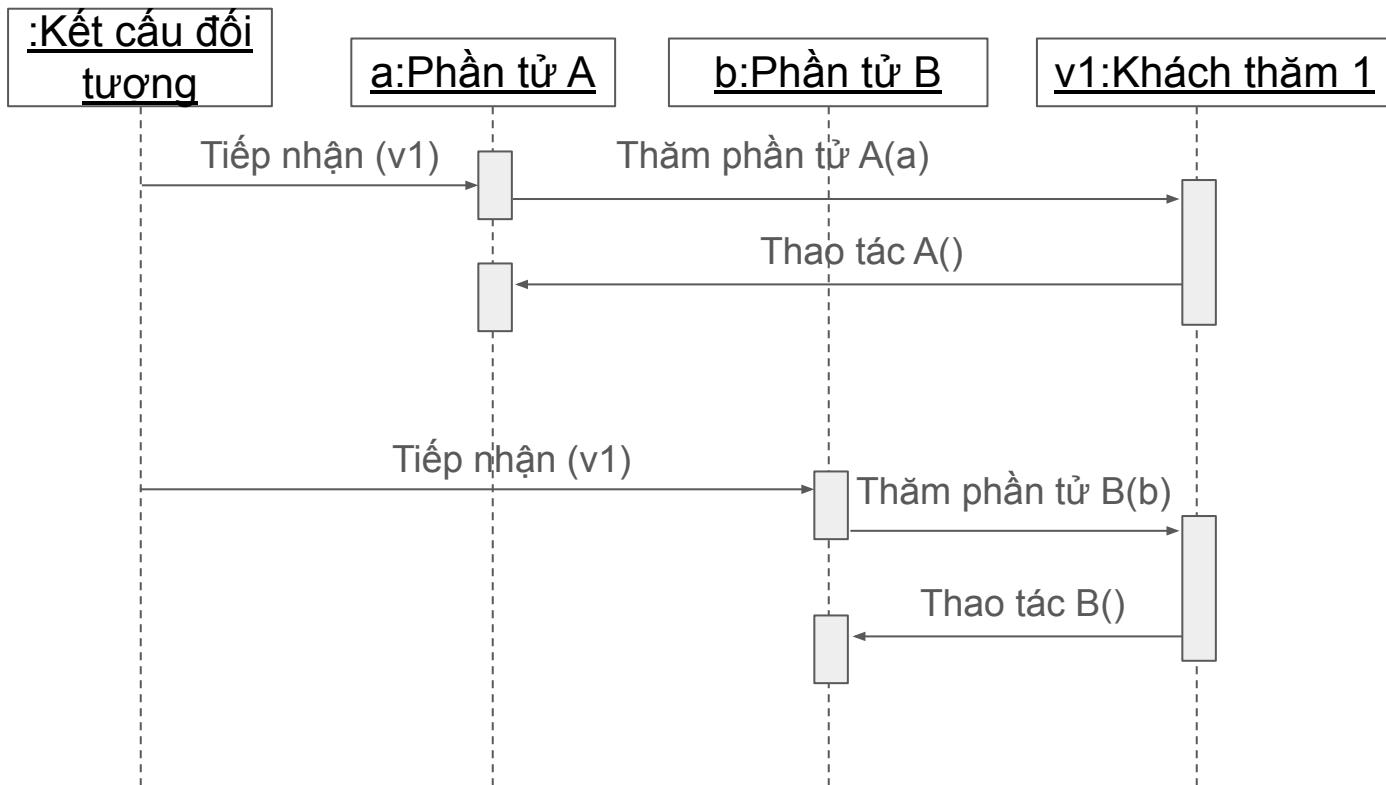
Biểu diễn một thao tác cần được thực hiện trên các đối tượng thuộc một kết cấu. Khách thăm cho phép định nghĩa một thao tác mới mà không thay đổi các lớp của các đối tượng chịu tác động.

# Khách thăm: Cấu trúc



# Khách thăm: Hành vi

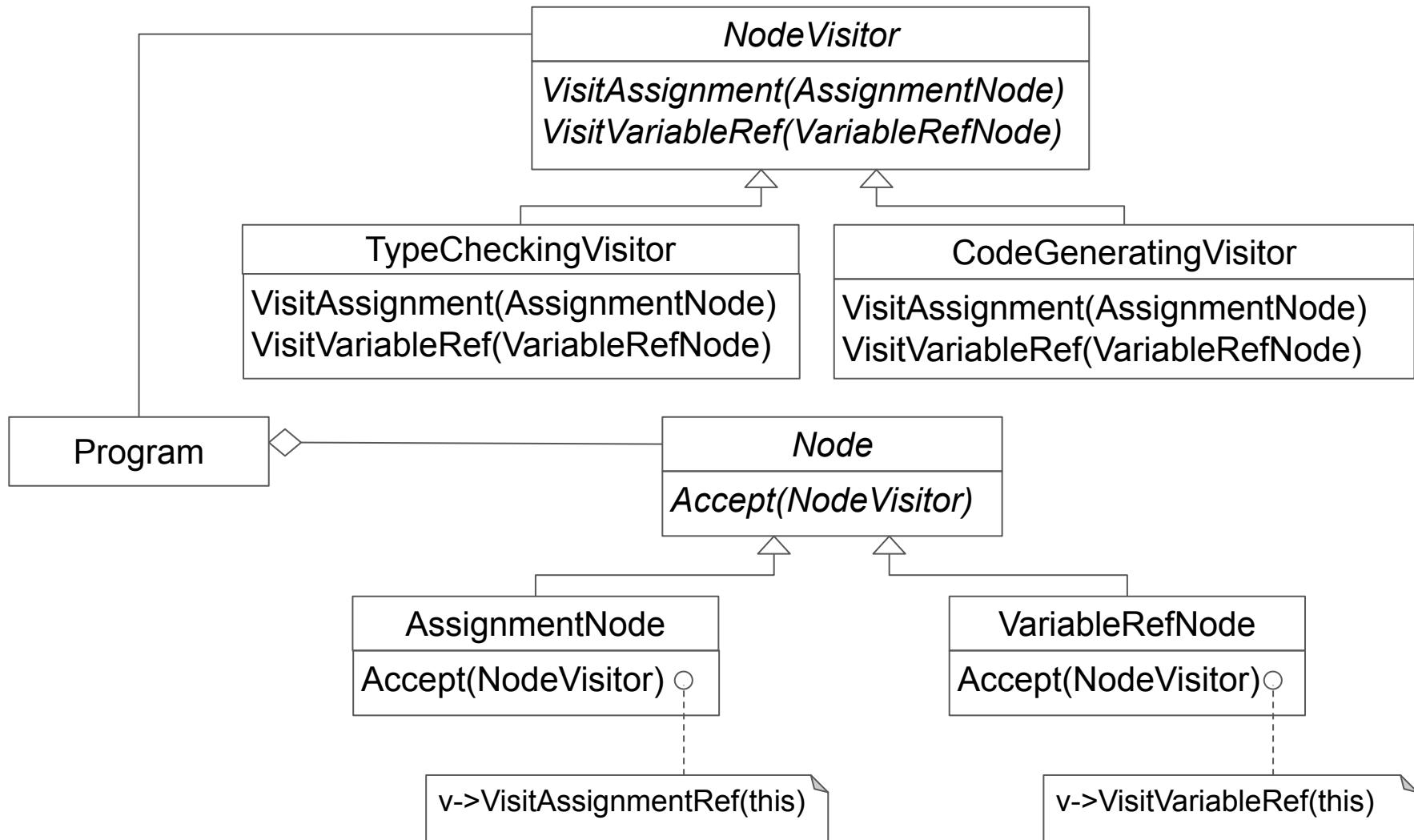
Tạo lớp v1 thay vì bỏ xung phương thức A::v1()  
và B::v1()



# Khách thăm: Các hệ quả

- Khách thăm giúp làm đơn giản hóa việc thêm các thao tác, phụ thuộc vào các thành phần của đối tượng phức tạp, chỉ cần định nghĩa lớp khách thăm mới.
- Khách thăm gom các thao tác liên quan lại và phân tách các thao tác không liên quan.
- Khó thêm các lớp thành phần cụ thể. Mỗi phần tử cụ thể mới kéo theo một thao tác khái quát trong Khách thăm và tất cả các Khách thăm cụ thể đã được triển khai.
- Các đối tượng khách thăm có thể tích lũy trạng thái khi thăm từng phần tử trong cấu trúc đối tượng.
- Mẫu khách thăm giả định giao diện của phần tử đủ mạnh để cho phép khách thăm thực hiện công việc, việc này có thể dẫn đến phá vỡ giới hạn đóng gói.

# Ví dụ 26. Cây cú pháp khái quát



# Tổng kết về mẫu tương tác

- Đóng gói các thay đổi:
  - Đối tượng chiến thuật đóng gói giải thuật;
  - Đối tượng trạng thái đóng gói hành vi theo trạng thái;
  - Đối tượng điều phối đóng gói giao thức giữa các đối tượng;
  - Đối tượng duyệt đóng gói cách truy cập và duyệt các đối tượng trong lưu trữ.

# Tổng kết về mẫu tương tác<sub>(2)</sub>

- Các mẫu tương tác hỗ trợ và củng cố lẫn nhau, ví dụ:
  - Lớp trong chuỗi trách nhiệm có thể sử dụng Phương thức mẫu với các thành phần để quyết định xử lý yêu cầu hoặc chọn đối tượng để chuyển tiếp.
  - Chuỗi trách nhiệm có thể sử dụng mẫu lệnh để biểu diễn yêu cầu như đối tượng;
  - Lớp diễn dịch có thể sử dụng mẫu Trạng thái để xác định ngũ cảnh xử lý. Đối tượng duyệt có thể duyệt qua các phần tử trong lưu trữ, và khách thăm có thể áp dụng thao tác cho các phần tử đó.

# Tổng kết về mẫu tương tác<sub>(3)</sub>

- Các mẫu tương tác cũng kết hợp tốt với các mẫu khác:
  - Trong mẫu hợp chất có thể sử dụng khách thăm để thực hiện thao tác trên các thành phần của hợp chất.
  - Hợp chất cũng có thể sử dụng chuỗi trách nhiệm để cho phép các thành phần truy cập các thuộc tính toàn cục thông qua các đối tượng trên của nó.
  - Hợp chất cũng có thể sử dụng lớp tô điểm để ghi đè các thuộc tính trên các phần của hợp chất, và có thể sử dụng lớp quan sát để gắn kết các phần khác nhau, và có thể sử dụng Trạng thái để cho phép các thành phần thay đổi hành vi theo trạng thái.
  - Hợp chất có thể được tạo bởi các lớp thợ xây và cũng có thể được coi như nguyên mẫu.

*Kết hợp các mẫu thiết kế có thể đem đến thiết kế tốt hơn cho hệ thống so với kết hợp ở mức nhỏ hơn lớp hoặc đối tượng.*



[https://docs.google.com/spreadsheets/d/1fOK6T1FdKmokeWQ\\_HCumxwtKB8E8sfWqUj0X0FI8ffA/edit?fbclid=IwY2xjawFaiCZleHRuA2FIbQIxMAABHdvzzgUiJ2KMPZNk2DFmnxn7CosmFI6TUIK4Bh4mS4CCvkApON1QGxsR8w\\_aem\\_pYUMXfMSJ9B988COAd6WyQ&gid=1793521563#gid=1793521563](https://docs.google.com/spreadsheets/d/1fOK6T1FdKmokeWQ_HCumxwtKB8E8sfWqUj0X0FI8ffA/edit?fbclid=IwY2xjawFaiCZleHRuA2FIbQIxMAABHdvzzgUiJ2KMPZNk2DFmnxn7CosmFI6TUIK4Bh4mS4CCvkApON1QGxsR8w_aem_pYUMXfMSJ9B988COAd6WyQ&gid=1793521563#gid=1793521563)

# Ví dụ đánh giá chi phí thực hiện dự án bằng Phương pháp đơn vị ca sử dụng

*Dữ liệu được sử dụng để đánh giá quy mô dự án gồm có:*

1. Tập đặc tả chi tiết ca sử dụng;
2. Biểu đồ ca sử dụng tổng quan.

*Chi tiết các bước đánh giá:*

*Danh giá trọng số tác nhân*

<b>Bảng đánh giá trọng số tác nhân chưa hiệu chỉnh</b>				
<b>Loại tác nhân</b>	<b>Mô tả</b>	<b>Điểm</b>	<b>Số lượng</b>	<b>Tổng điểm</b>
Đơn giản	Hệ thống ngoại với API được định nghĩa rõ ràng	1	0	0
Trung bình	Hệ thống ngoại sử dụng một giao diện dựa trên giao thức, ví dụ, HTTP, TCP/IP, hoặc một cơ sở dữ liệu	2	0	0
Phức tạp	Người	3	4	12
<b>Tổng trọng số tác nhân chưa hiệu chỉnh (UAW)</b>				<b>12</b>

*Danh giá trọng số ca sử dụng*

<b>Bảng đánh giá trọng số ca sử dụng chưa hiệu chỉnh</b>				
<b>Loại CSD</b>	<b>Mô tả</b>	<b>Điểm</b>	<b>Số lượng</b>	<b>Tổng điểm</b>
Đơn giản	1-3 giao dịch	5	3	15
Trung bình	4-7 giao dịch	10	4	40
Phức tạp	>7 giao dịch	15	1	15
<b>Tổng trọng số ca sử dụng chưa hiệu chỉnh (UUCW)</b>				<b>70</b>

*\*Ghi chú: Một giao dịch có thể tương đương với một hoặc một gói sự kiện trong luồng sự kiện dẫn đến thông tin được lưu vào hệ thống hoặc thông tin được lấy ra từ hệ thống. Đây là một trong số nhiều cách đánh giá độ phức tạp của ca sử dụng.*

**Số lượng đơn vị ca sử dụng chưa hiệu chỉnh: UUCP = UAW + UUCW = 12 + 70 = 82**

### *Đánh giá hệ số phức tạp kỹ thuật*

Gán cho mỗi đặc điểm kỹ thuật một giá trị trong khoảng từ 0 đến 5, với 0 nghĩa là không ảnh hưởng đến dự án, 3 là ảnh hưởng ở mức độ trung bình, và 5 là ảnh hưởng cao nhất đến dự án. Có tất cả 13 đặc điểm kỹ thuật cần đánh giá.

<b>Các hệ số phức tạp kỹ thuật</b>					
Mã số	Mô tả	Hệ số	Giá trị	Giá trị thực	Ghi chú
T1	Hệ phân tán	2	0	0	
T2	Thời gian phản hồi hoặc thông lượng	1	5	5	
T3	Sử dụng thuận tiện và hiệu quả	1	3	3	
T4	Xử lý bên trong phức tạp	1	1	1	
T5	Khả năng tái sử dụng mã nguồn	1	1	1	
T6	Dễ cài đặt	0.5	2	1	
T7	Dễ vận hành	0.5	4	2	
T8	Tính khả chuyển	2	0	0	
T9	Dễ bảo trì và cập nhật	1	2	2	
T10	Xử lý tính toán song song/dòng thời	1	0	0	
T11	Bảo mật	1	0	0	
T12	Liên kết với đối tác, sử dụng/cung cấp	1	0	0	
T13	Đào tạo đặc biệt cho người dùng	1	0	0	
<b>Tổng giá trị hệ số kỹ thuật (TFactor)</b>					<b>15</b>

**Hệ số phức tạp kỹ thuật:**  $TCF = 0.6 + (0.01 * TFactor) = 0.6 + (0.01 * 15) = 0.75$

### *Đánh giá hệ số phức tạp môi trường*

Gán cho mỗi đặc điểm môi trường một giá trị trong khoảng từ 0 đến 5. Khác với đặc điểm kỹ thuật (giá trị thể hiện ảnh hưởng của các đặc điểm lên dự án), giá trị gán với đặc điểm môi trường thể hiện chất lượng/mức độ của đặc điểm đó trong môi trường làm việc, ví dụ 0 gắn với động lực làm việc thể hiện nhóm không muốn làm việc, 3 thể hiện nhóm được khuyến khích làm việc ở mức trung bình, 5 thể hiện tinh thần làm việc của nhóm ở mức cao.

<b>Các hệ số môi trường</b>					
<b>Mã số</b>	<b>Mô tả</b>	<b>Trọng số</b>	<b>Giá trị</b>	<b>Giá trị thực</b>	<b>Ghi chú</b>
E1	Có kinh nghiệm với quy trình phát triển hệ thống	1.5	4	6	
E2	Có kinh nghiệm về ứng dụng tương tự	0.5	4	2	
E3	Kinh nghiệm về hướng đối tượng	1	4	4	
E4	Khả năng lãnh đạo nhóm	0.5	5	2.5	
E5	Động lực làm việc	1	5	5	
E6	Sự ổn định của yêu cầu	2	5	10	
E7	Nhân sự bán thời gian	-1	0	0	
E8	Sự phức tạp của ngôn ngữ lập trình	-1	4	-4.0	
<b>Tổng giá trị hệ số môi trường (EFactor)</b>				<b>25.5</b>	

$$\text{Hệ số môi trường: } EF = 1.4 + (-0.03 * EFactor) = 1.4 + (-0.03 * 25.5) = 0.635$$

**Số lượng đơn vị ca sử dụng sau hiệu chỉnh:**

$$UCP = UUCP * TCF * EF = 82 * 0.75 * 0.635 = 39.0525$$

Đặt số lượng đặc điểm môi trường không thuận lợi = (#đặc điểm trong khoảng E1...E6 được gán giá trị < 3) + (# đặc điểm trong khoảng E7...E8 được gán giá trị > 3)

Nếu số lượng đặc điểm môi trường không thuận lợi  $\leq 2$

thì PHM = 20

Ngược lại, nếu số lượng đặc điểm môi trường không thuận lợi = 3 hoặc 4

thì PHM = 28

Ngược lại

thì suy nghĩ lại về dự án; rủi ro thất bại quá cao.

$$\text{Chi phí tính bằng giờ nhân lực E} = UCP * PHM = 39.1 * 20 = 782$$

Giả sử số giờ làm việc trong một tháng là 158 (không tính các ngày cuối tuần), như vậy:

$$\text{Chi phí tính bằng tháng nhân lực E} = UCP * PHM / 158 = 4.9$$

Theo McConnel, 1996 (chỉ áp dụng với tháng nhân lực), thời gian lý tưởng đối với thực hiện dự án là:

$$T = 2.5 * \sqrt[3]{E} = 2.5 * \sqrt[3]{4.9} = 4.2$$

**Kích thước nhóm trung bình P = E/T = 4.9 / 4.2 = 1.2**

Với nhóm có số lượng thành viên > P thì thời gian hoàn thành dự án được đánh giá là cao hơn E / số lượng thành viên.

# Phân tích và thiết kế hệ thống

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2021

# Chuyển sang thiết kế

# Nội dung

- Cân bằng các mô hình phân tích
- Lựa chọn chiến lược thiết kế

# Nội dung

- Cân bằng các mô hình phân tích
- Lựa chọn chiến lược thiết kế

# Chuyển từ phân tích sang thiết kế

## Phân tích



## Thiết kế

# Các góc nhìn



# Sự gắn kết giữa các mô hình

- Chúng ta mô hình hóa hệ thống từ nhiều góc nhìn:
  - Mô hình từ mỗi góc nhìn có thể tập trung vào 1 khía cạnh khác nhau, đáp ứng nhu cầu thông tin khác nhau.
  - Một thành phần hệ thống có thể xuất hiện nhiều lần trong các mô hình khác nhau.
    - Ví dụ: Các hoạt động, luồng sự kiện và các thông điệp, v.v..
- Cần đảm bảo tính nhất quán giữa các biểu diễn và đảm bảo tính lô-gic giữa những nội dung liên quan - Tính cân bằng giữa các mô hình.

*Thử kết hợp, kiểm tra và giá các mô hình phân tích trước khi bắt đầu các hoạt động thiết kế*

# Cân bằng các mô hình phân tích

- Được thực hiện bởi 1 đội gồm người phân tích, người thiết kế và khách hàng.
- Các mục đích:
  - Kiểm tra tính đúng đắn của các mô hình
  - Tìm lỗi và điều chỉnh kịp thời

*Tiềm ẩn trách nhiệm đối với người phân tích vì những lỗi được phát hiện.*

# Cân bằng các mô hình chức năng

1. Có thể ánh xạ 2 chiều giữa các sự kiện trong các đặc tả ca sử dụng và các hoạt động trong sơ đồ hoạt động.
2. Các nút đối tượng (nếu có) trong sơ đồ hoạt động phải được sử dụng trong đặc tả ca sử dụng.
3. Thứ tự của các sự kiện trong đặc tả ca sử dụng phải nhất quán với thứ tự của các hoạt động trong sơ đồ hoạt động.
4. Đảm bảo tương ứng 1-1 giữa các ca sử dụng trong sơ đồ ca sử dụng tổng quan và các đặc tả ca sử dụng.
5. Các tác nhân có trong đặc tả ca sử dụng phải được biểu diễn trên sơ đồ ca sử dụng.
6. Các cổ đông có trong đặc tả ca sử dụng phải được biểu diễn như các tác nhân trên sơ đồ ca sử dụng.
7. Các mối quan hệ có trong đặc tả ca sử dụng phải nhất quán với các mối quan hệ có trong sơ đồ ca sử dụng.

# Cân bằng các mô hình cấu trúc

1. Tương ứng 1-1 giữa các thẻ CRC và các lớp trên sơ đồ lớp.
2. Các trách nhiệm được liệt kê trong thẻ CRC phải được thể hiện bằng các phương thức trong sơ đồ lớp.
3. Các đối tác trong thẻ CRC được thể hiện bằng mối quan hệ liên kết trong sơ đồ lớp.
4. Các thuộc tính được liệt kê trong các thẻ CRC phải được gắn kết với các thuộc tính trong lớp trên sơ đồ lớp.
5. Các thuộc tính có kiểu là lớp khác được biểu diễn bằng quan hệ tổng hợp giữa các lớp.
6. Các mối quan hệ trong các thẻ CRC phải nhất quán với các mối quan hệ được biểu diễn trong sơ đồ lớp.
7. Chỉ sử dụng các lớp liên kết cho các liên kết có các thuộc tính không có trong cả 2 lớp.
8. Các sơ đồ đối tượng phải tương thích với sơ đồ lớp: Thuộc tính, cơ số, liên kết.

# Cân bằng các mô hình hành vi

1. Cùng tác nhân & đối tượng trong mỗi cặp sơ đồ tuần tự và sơ đồ giao tiếp tương ứng.
2. Cùng thông điệp (tên, tham số, nguồn, đích, điều kiện bảo vệ) trong mỗi cặp sơ đồ tuần tự và sơ đồ giao tiếp tương ứng.
3. Cùng thứ tự thông điệp trong mỗi cặp sơ đồ tuần tự (theo trục thời gian) và sơ đồ giao tiếp (số thứ tự) tương ứng.
4. Các bước chuyển trạng thái trong sơ đồ máy trạng thái phải gắn kết với các thông điệp trong các sơ đồ tương tác (tuần tự hoặc giao tiếp).
5. Các giá trị trong ma trận CRUD(E) phải gắn kết với các thông điệp trong các sơ đồ tương tác.

# Các mô hình chức năng & cấu trúc

1. Mỗi lớp trên sơ đồ lớp phải được gắn với ít nhất 1 ca sử dụng
2. Mỗi hoạt động trong sơ đồ hoạt động và mỗi sự kiện trong đặc tả ca sử dụng phải được gắn kết với ít nhất 1 phương thức trên sơ đồ lớp.
3. Mỗi nút đối tượng trên sơ đồ hoạt động phải là 1 thuộc tính hoặc 1 đối tượng thuộc lớp có trong sơ đồ lớp
4. Mỗi thuộc tính hoặc 1 mối quan hệ trên sơ đồ lớp phải được gắn kết với ít nhất 1 chủ thể hoặc đối tượng của 1 sự kiện trong đặc tả ca sử dụng.

# Các mô hình chức năng & hành vi

1. Các sơ đồ tương tác phải được gắn với ca sử dụng
2. Các tác nhân trong các sơ đồ tương tác hoặc ma trận CRUD(E) phải được gắn với các tác nhân trong ca sử dụng
3. Các thông điệp trong các sơ đồ tương tác, các bước chuyển trong máy trạng thái và các giá trị trong ma trận CRUD(E) phải gắn kết với các hoạt động trong sơ đồ hoạt động và các sự kiện trong đặc tả ca sử dụng.
4. Tất cả các đối tượng phức tạp trong các sơ đồ hoạt động phải được biểu diễn bằng sơ đồ máy trạng thái.

# Các mô hình cấu trúc & hành vi

1. Sơ đồ máy trạng thái phải được gắn với đối tượng thuộc lớp có trong sơ đồ lớp
2. Các đối tượng trong các sơ đồ tương tác phải thuộc lớp có trong sơ đồ lớp.
3. Các thông điệp trong các sơ đồ tương tác và các bước chuyển trong sơ đồ máy trạng thái phải được gắn với phương thức trong lớp.
4. Các trạng thái của đối tượng trong sơ đồ máy trạng thái phải được biểu diễn bằng các thành phần trong sơ đồ lớp.
5. Các đối tượng trong ma trận CRUD(E) phải thuộc lớp có trong sơ đồ lớp.

# Nội dung

- Cân bằng các mô hình phân tích
- Lựa chọn chiến lược thiết kế



# Các chiến lược thiết kế

- Phát triển riêng - Xây dựng hoàn toàn trong phạm vi nội bộ
- Mua gói ứng dụng
  - Các ứng dụng văn phòng (trình soạn thảo, bảng tính, v.v..)
  - Hệ thống quản lý nhân sự, quản lý kinh doanh, v.v..
- Tích hợp hệ thống
  - Hệ thống bán hàng và hệ thống quản lý hóa đơn, v.v..
- Thuê đối tác (gia công phần mềm)
  - Hợp tác phát triển các tính năng không yêu cầu giữ bí mật

# Phát triển riêng

- Có thể đáp ứng những yêu cầu chuyên sâu.
- Có thể linh động và sáng tạo trong giải quyết vấn đề.
- Dễ dàng thay đổi các thành phần.
- Phát triển các kỹ năng cá nhân.
- Phụ thuộc vào năng lực của ban CNTT.
- Có thể tạo thêm rủi ro đáng kể.

# Mua gói phần mềm

- Mua phần mềm có trên thị trường (ví dụ phần mềm kế toán).
- Có phạm vi từ 1 thành phần, 1 công cụ đến cả 1 hệ thống thông tin hoàn thiện.
- Có thể hiệu quả nếu phù hợp với nghiệp vụ
- Có thể đã được kiểm tra và thử nghiệm kỹ.
- Nhưng có thể phải chấp nhận chức năng như được cung cấp.
  - Rủi ro phải thay đổi quy trình nghiệp vụ.
- Có thể phải tùy chỉnh đáng kể hoặc phát triển các thành phần ghép nối/thay thế.

# Tích hợp hệ thống

- Kết hợp các gói, hệ thống cũ, và phần mềm mới
  - Không hiếm trường hợp mua phần mềm có sẵn và thuê đối tác tích hợp nó với hệ thống hiện có
- Thách thức chủ yếu là tích hợp dữ liệu
  - Có thể yêu cầu chuyển đổi dữ liệu
  - Có thể phải sử dụng gói phần mềm bổ xung để viết dữ liệu theo cùng định dạng với hệ thống cũ
- Kỹ thuật đóng gói
  - Đóng gói hệ thống cũ và cung cấp các API để hệ thống mới có thể tương tác với nó.
  - Tránh can thiệp vào hệ thống hiện có.

# Gia công phần mềm

- Thuê 1 công ty - đối tác xây dựng hệ thống
- Có thể mở rộng các tài nguyên và kỹ năng hiện có.
- Yêu cầu uy tín, trao đổi thông tin 2 chiều.
- Nhưng có rủi ro mất kiểm soát trong chia sẻ thông tin quan trọng, chuyển giao công nghệ.
- Cần thận trọng lựa chọn đối tác, thận trọng chuẩn bị hợp đồng và phương thức thanh toán.

*Không thuê gia công những gì không nắm vững*

# Gia công phần mềm<sub>(2)</sub>

Các dạng hợp đồng:

- Thời gian-và-Thỏa thuận: Thanh toán theo thời gian và các khoản chi
- Giá cố định: Thanh toán theo giá như đã thỏa thuận
- Giá trị tăng cường: Thanh toán theo % lợi nhuận.

# Lựa chọn chiến lược thiết kế

	<b>Phát triển riêng</b>	<b>Sử dụng gói phần mềm</b>	<b>Gia công</b>
<b>Nhu cầu nghiệp vụ</b>	Duy nhất	Phổ biến	Không phải cốt lõi
<b>Chuyên môn hiện có</b>	Có nghiệp vụ và kỹ thuật	Không có kỹ thuật	Không có nghiệp vụ hoặc kỹ thuật
<b>Kỹ năng dự án</b>	Có mong muốn phát triển kỹ năng liên quan	Phát triển kỹ năng không nằm trong chiến lược	Quyết định gia công là 1 quyết định chiến lược
<b>Người quản lý dự án</b>	Có trình độ cao và phương pháp phát triển đã được kiểm chứng	Có thể điều phối công việc của nhà cung cấp	Có trình độ cao trong môi trường của đối tác
<b>Khung thời gian</b>	Linh động	Ngắn	Ngắn hoặc linh động

# Lựa chọn chiến lược thực hiện

- Xác định các công cụ và kỹ thuật cần để tự phát triển
- Xác định các gói phù hợp với nhu cầu người dùng
- Tìm các công ty có thể xây dựng hệ thống theo hợp đồng
- Tạo ma trận lựa chọn để cân nhắc các ưu điểm và nhược điểm của mỗi lựa chọn
  - Kết hợp với tính khả thi kỹ thuật, kinh tế, và tổ chức
  - Sử dụng cơ chế yêu cầu đề xuất (RFP) và yêu cầu thông tin (RFI) để thu thập ước lượng kinh phí & thời gian từ những nhà cung cấp tiềm năng



# Phân tích và thiết kế hệ thống

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2021

# Kiến trúc hệ thống

# Nội dung

- Mô-đun hóa hệ thống
- Các thành phần kiến trúc hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML
  - Sơ đồ gói
  - Sơ đồ thành phần
  - Sơ đồ triển khai

# Nội dung

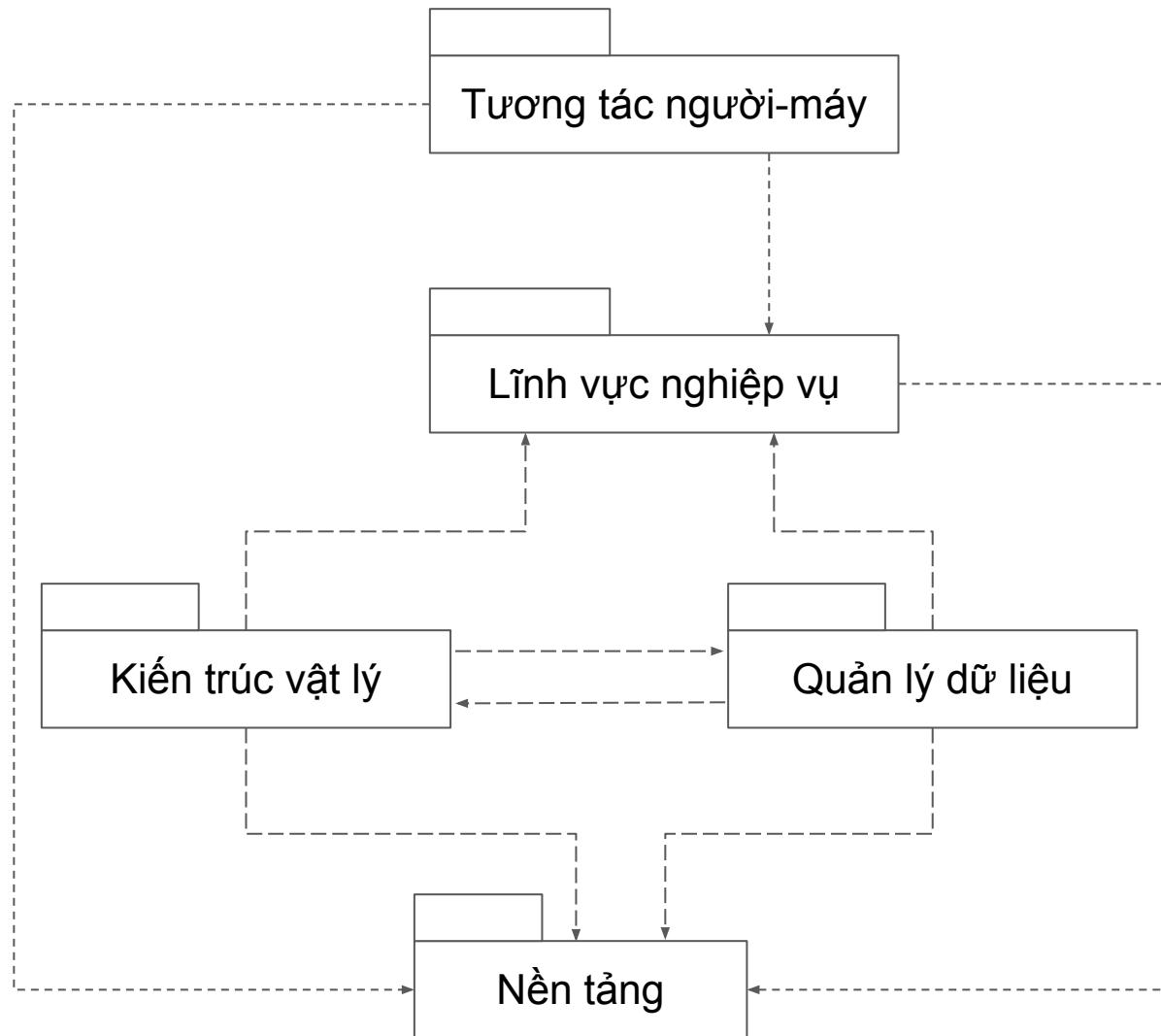
- Mô-đun hóa hệ thống
- Các thành phần kiến trúc hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML



# Mô-đun hóa hệ thống

- Mô-đun hóa: Chia nhỏ hệ thống thành nhiều mô-đun - các cấu phần hệ thống.
- Có thể vận dụng các nguyên lý thiết kế, và thử nghiệm với các kịch bản thay đổi.
- Có thể dựa trên các tương tác và các mối liên hệ:
  - Các sơ đồ lớp
  - Các sơ đồ giao tiếp
    - Ví dụ, các thông điệp và liên kết trong sơ đồ giao tiếp
    - Nếu có nhiều thông điệp được trao đổi giữa 2 lớp thì đưa cả 2 lớp vào cùng 1 mô-đun.
  - Ma trận CRUDE
  - v.v..
- Tạo mô-đun từ các phần tử có bậc liên kết cao.

# Ví dụ 1. Kiến trúc 5 tầng



# Ví dụ 1. Kiến trúc 5 tầng<sub>(2)</sub>

- **Tầng nền tảng:** Cung cấp các phần tử cơ bản để xây dựng hệ thống. Ví dụ: Array, Map, ...
- **Tầng nghiệp vụ:** Các biểu diễn và xử lý nhằm đáp ứng các hoạt động nghiệp vụ, đã được đề cập tới ở pha phân tích, và tiếp tục được phát triển ở pha thiết kế. Ví dụ lớp: Order, Customer
- **Quản lý và truy cập dữ liệu:** Các xử lý liên quan đến lưu trữ cố định. Ví dụ: CustomerDAM, FileInputStream
- **Tương tác người-máy:** Các giao diện người dùng, thường là các giao diện đồ họa. Ví dụ: Form, Button

# Nội dung

- Mô-đun hóa hệ thống
  - Các thành phần kiến trúc hệ thống
  - Mẫu kiến trúc
  - Các sơ đồ UML
- 

# Các thành phần kiến trúc

- Hạ tầng
  - Máy tính, mạng máy tính và hình trạng mạng, và phần mềm hệ thống
  - Hình thành hạ tầng để triển khai phần mềm và cung cấp dịch vụ.
- Phần mềm
  - Chương trình ứng dụng, dịch vụ Web, CSDL, v.v.., hỗ trợ tác nghiệp.
  - Được triển khai trên hạ tầng, cài đặt trên các thiết bị phần cứng và kết nối với nhau qua các giao thức và đường truyền mạng.

*Mục đích thiết kế kiến trúc tổng quan là xác định cách bố trí các thành phần phần mềm trong các thiết bị phần cứng.*

# Thiết bị tính toán

- Máy chủ - Cung cấp các tài nguyên dùng chung cho người dùng và các máy tính khác.
  - Hoạt động liên tục, độ tin cậy cao.
- Máy khách / Máy tính cá nhân
  - Máy bàn, máy tính xách tay, máy tính bảng, điện thoại thông minh, v.v..
  - Người dùng sử dụng để tương tác với hệ thống.

# Điện toán đám mây

- Mô hình cho thuê phần cứng
  - Máy chủ ở trên "Đám mây"
  - Máy khách là thiết bị của người dùng
- "Đám mây"
  - Trung tâm dữ liệu, nội bộ hoặc công khai; hoặc
  - Dịch vụ được cung cấp bởi các đối tác.
  - Tích hợp nhiều công nghệ:
    - Ảo hóa
    - Kiến trúc hướng dịch vụ
    - Mạng lưới tính toán / Grid computing
    - V.v...

# Phần mềm như dịch vụ (SaaS)

- Thường được thực hiện với hạ tầng điện toán đám mây
- Phần mềm được triển khai trên máy chủ thay vì thiết bị của người dùng.
- Các dịch vụ ứng dụng được truy cập từ xa (thường qua trình duyệt).
- Dữ liệu người dùng được cài đặt và được lưu trên máy chủ dịch vụ.

# Ứng dụng Web

- Được triển khai dựa trên các quy chuẩn Web.
- Truy cập qua URL.
- Dữ liệu được trao đổi dựa trên giao thức mạng.
- Các xử lý được thực hiện trên máy chủ.
- Dữ liệu được trả về trong trang Web.
- v.v..

# Hạ tầng mạng

- Bao gồm thiết bị mạng, phần mềm mạng.
- Hạ tầng Internet
  - Các đường truyền băng thông lớn và máy tính tốc độ cao
  - Được sở hữu bởi các chính phủ và các công ty viễn thông
- Mạng địa phương (LAN)
  - Mạng nhỏ cho 1 hệ thống thông tin nội bộ.
- Mạng toàn cầu (WWW)
  - Tất cả các tài nguyên được kết nối và truy cập qua Internet.

# Các giao thức

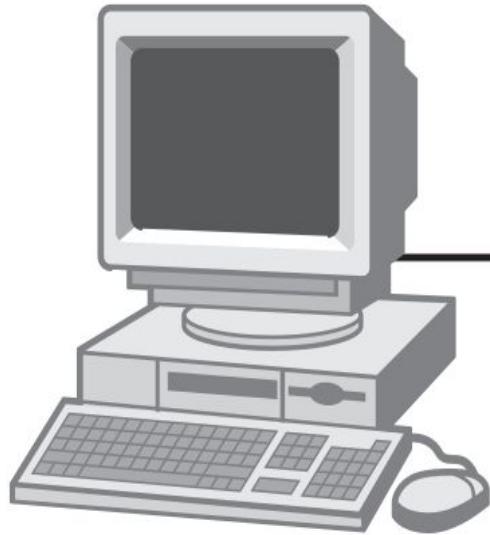
- Giao thức:
  - Một tập ngôn ngữ và quy tắc đảm bảo giao tiếp và trao đổi dữ liệu giữa phần cứng và phần mềm
- Các giao thức mạng:
  - Mạng riêng tư ảo (VPN)
    - Tạo mạng riêng tư trên Internet bằng cách sử dụng các công nghệ bảo mật và mã hóa

# Các cấu phần hệ thống tiêu biểu

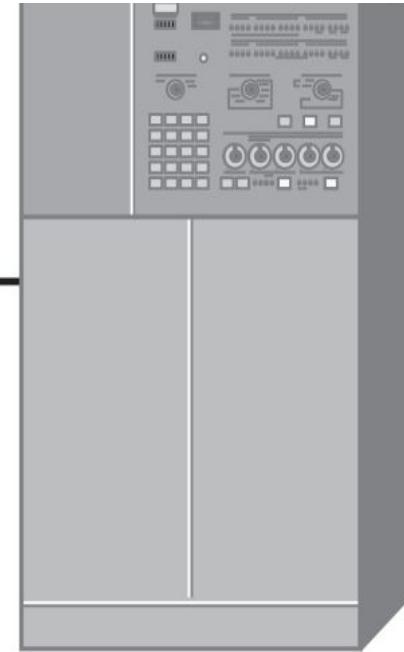
- Thiết bị phần cứng
  - Máy khách / máy tính cá nhân
  - Máy chủ
  - Hạ tầng mạng
- Thành phần phần mềm
  - Thành phần lưu trữ dữ liệu
  - Thành phần truy cập dữ liệu
  - Thành phần ứng dụng
  - Thành phần trình diễn

# Kiến trúc dựa trên máy chủ

Máy khách/Cổng  
thao tác



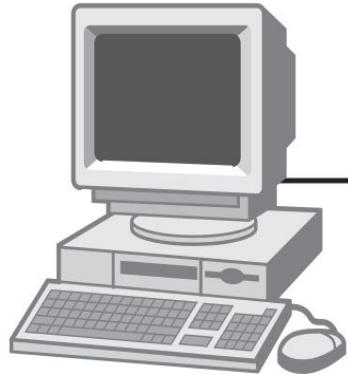
Máy chủ



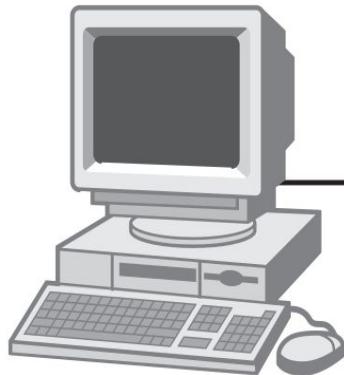
Trình diễn  
Ứng dụng  
Truy cập dữ liệu  
Lưu trữ dữ liệu

# Kiến trúc dựa trên máy khách

Các máy khách

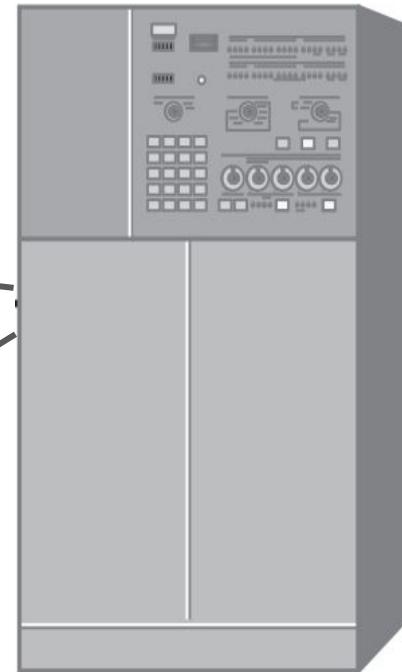


Trình diễn, ứng dụng,  
truy cập dữ liệu



Trình diễn, ứng dụng,  
truy cập dữ liệu

Máy chủ



Lưu trữ dữ liệu

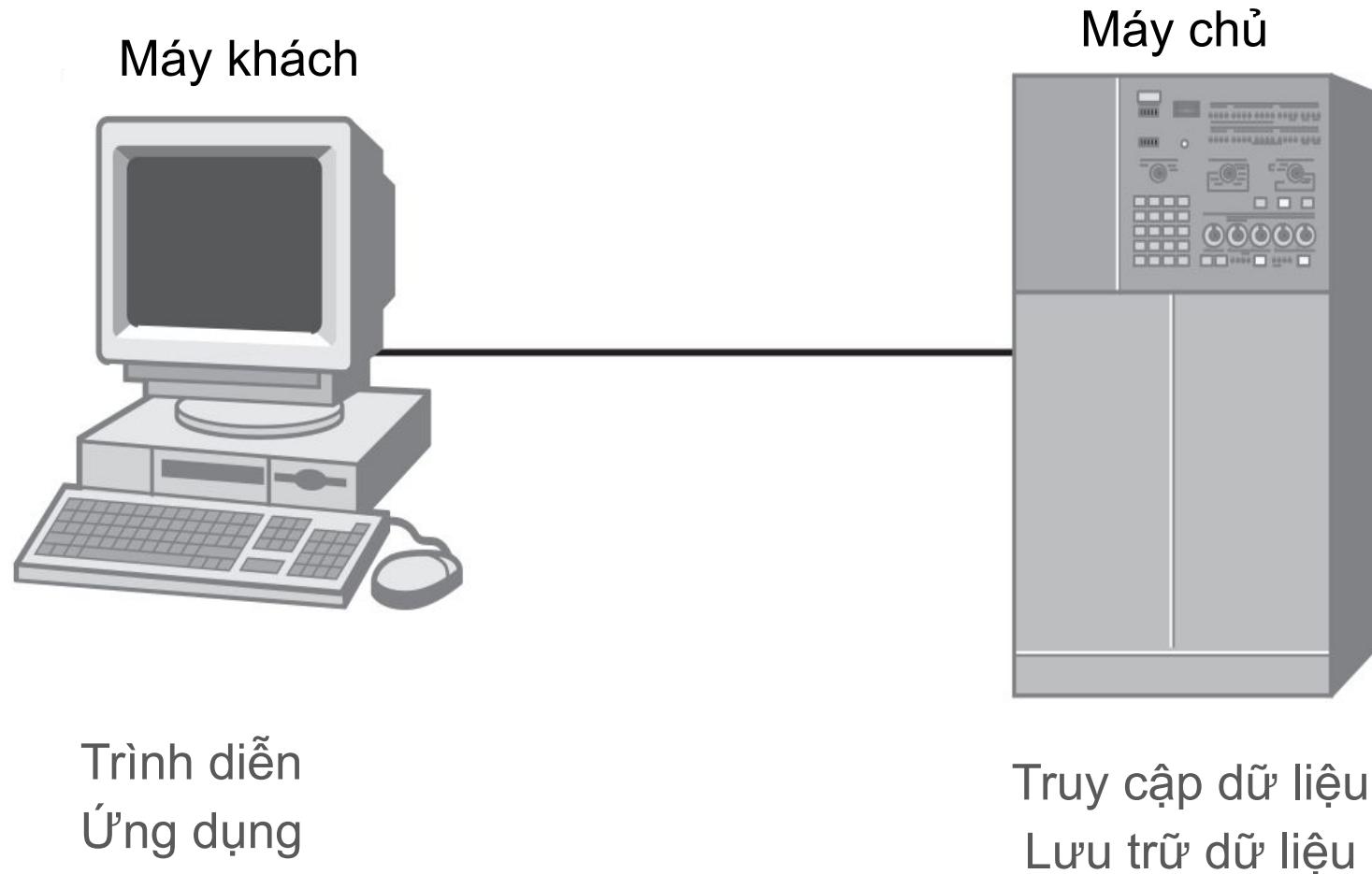
# Kiến trúc máy khách-máy chủ

- Hệ thống được thiết kế với 1 phần ở máy chủ và 1 phần ở máy khách, cố gắng phân chia hợp lý khối lượng tính toán
- Kiến trúc phổ biến nhất trong các hệ thống hiện đại
- Khối lượng tính toán của máy khách dao động
  - Máy khách mỏng chỉ thực hiện lô-gic trình diễn
  - Máy khách dày thực hiện lô-gic trình diễn và lô-gic ứng dụng
- Khả năng mở rộng cao với chi phí riêng từng phần.
- Phức tạp hơn so với phát triển ứng dụng theo kiến trúc dựa trên máy khách và kiến trúc dựa trên máy chủ, và cần hỗ trợ tương tác trong môi trường phân tán.

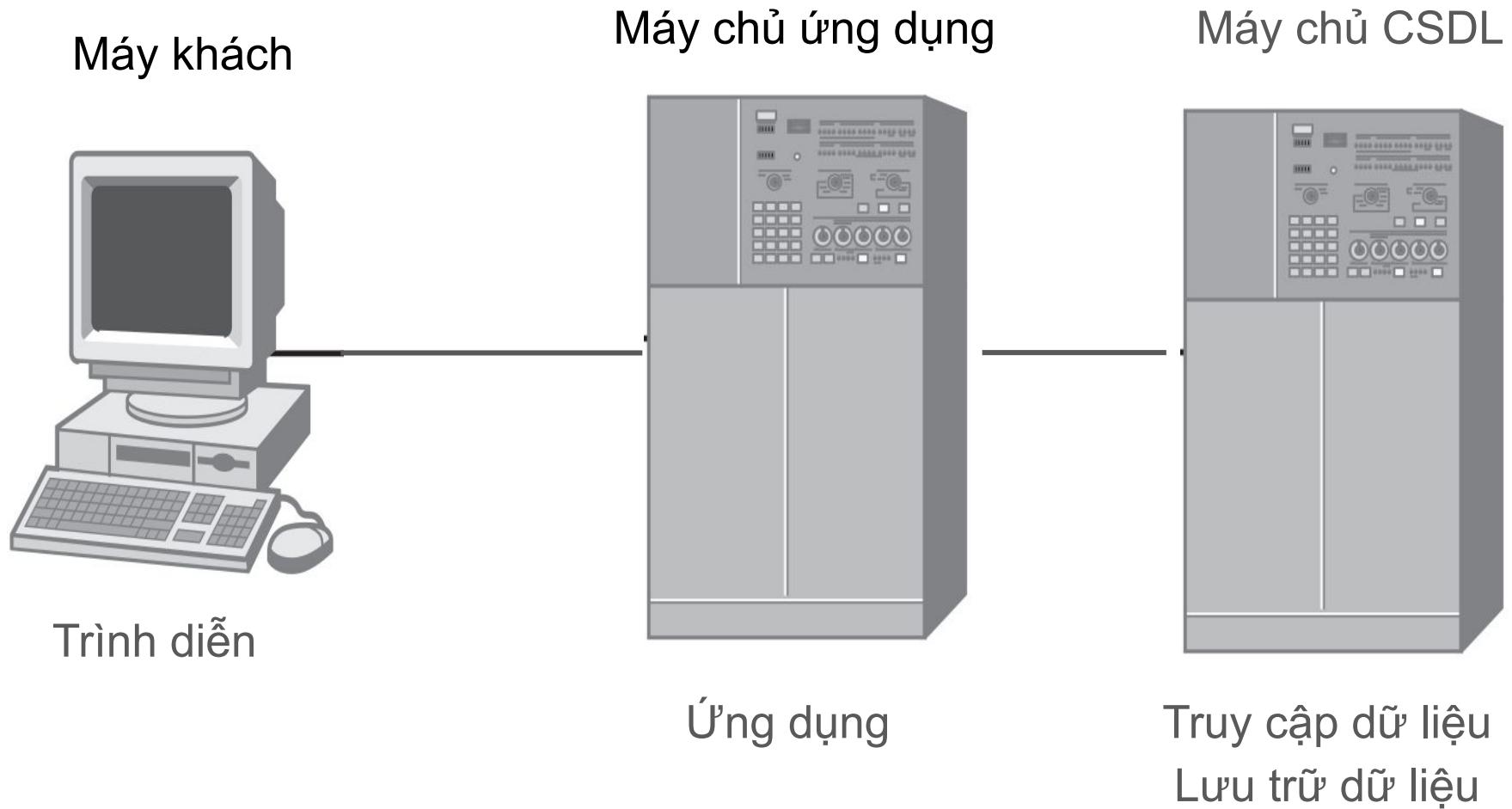
# Các mô hình kiến trúc máy khách-máy chủ

- 2-dãy: 1 máy chủ chịu trách nhiệm lưu trữ và quản lý truy cập dữ liệu; máy khách có trách nhiệm xử lý lô-gic ứng dụng và lô-gic trình diễn
- 3-dãy: Lô-gic truy cập và lưu trữ dữ liệu trên 1 máy chủ; lô-gic ứng dụng trên 1 máy chủ khác; máy khách có trách nhiệm xử lý lô-gic trình diễn.
- n-dãy: Lô-gic ứng dụng được phân chia trên nhiều máy chủ, lô-gic dữ liệu trên 1 máy khác chủ khác
  - Phổ biến trong các ứng dụng thương mại điện tử
  - Cân bằng tải tốt hơn
  - Khả năng mở rộng cao hơn hệ thống 2 hoặc 3 dãy
  - Nhu cầu sử dụng mạng cao hơn
- Các dãy máy khách-máy chủ được xác định dựa trên lô-gic chia nhỏ hệ thống.

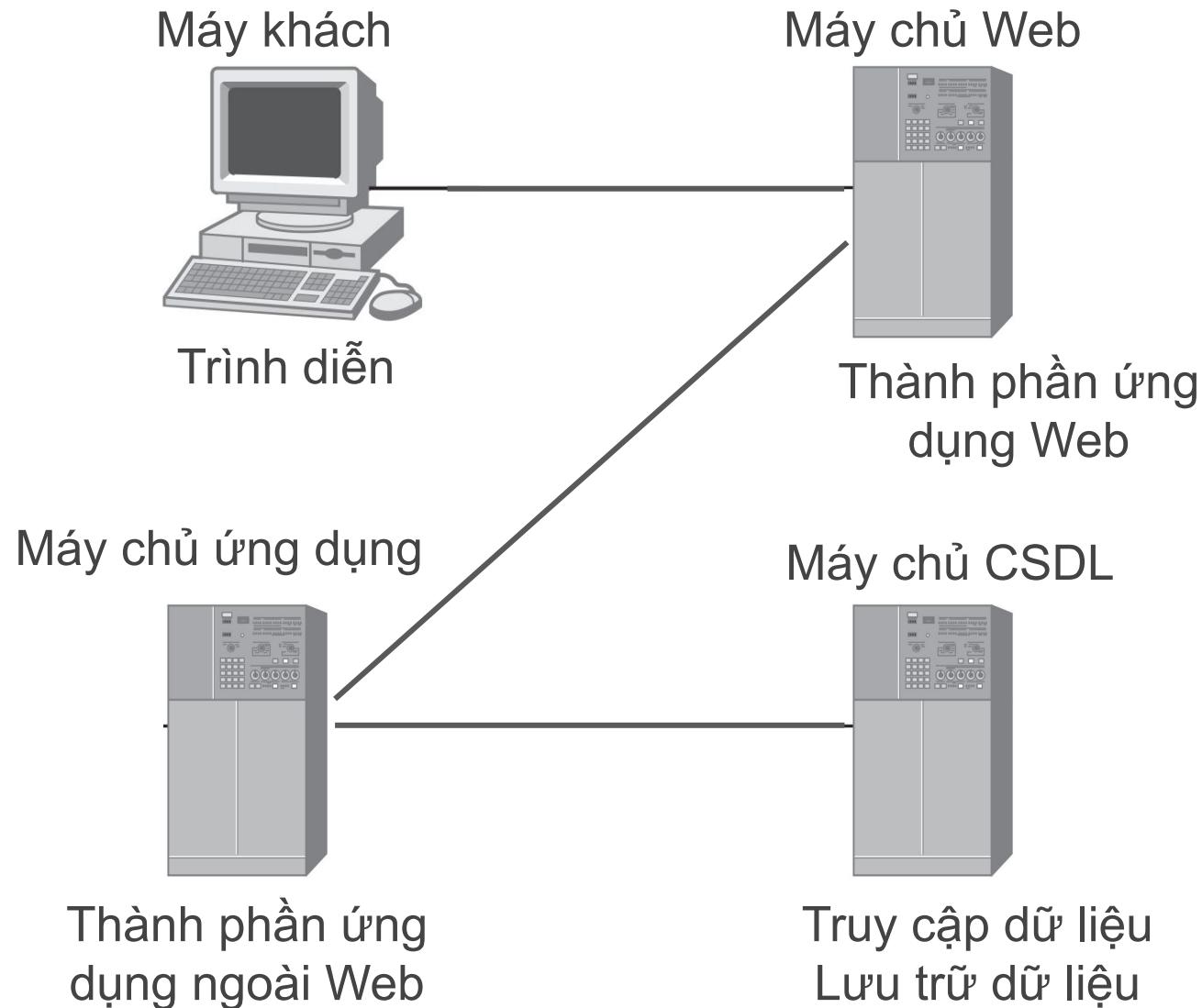
# Kiến trúc máy khách-máy chủ 2-dây



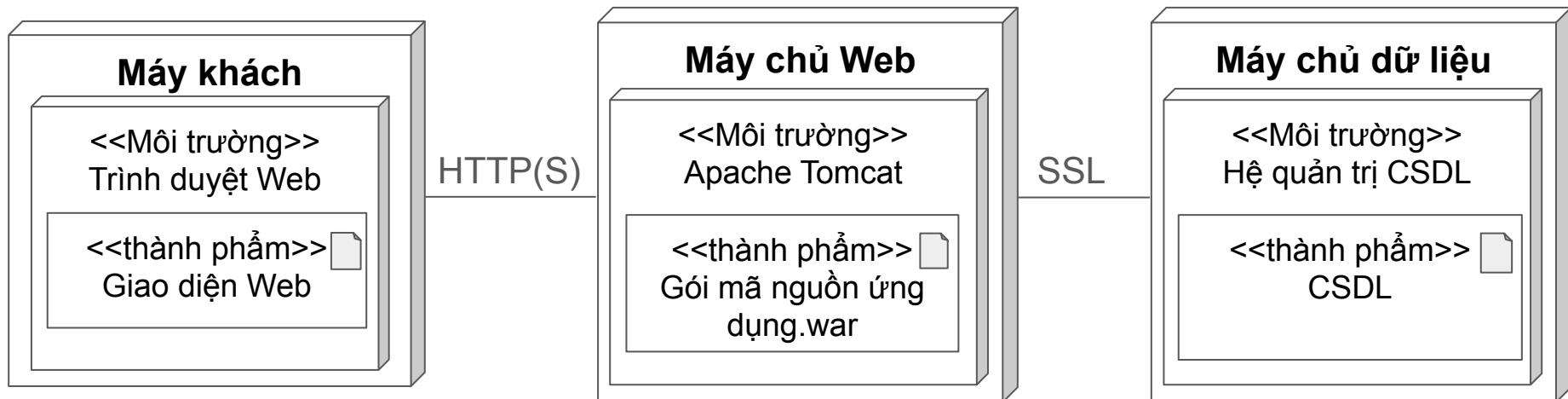
# Kiến trúc máy khách-máy chủ 3-dây



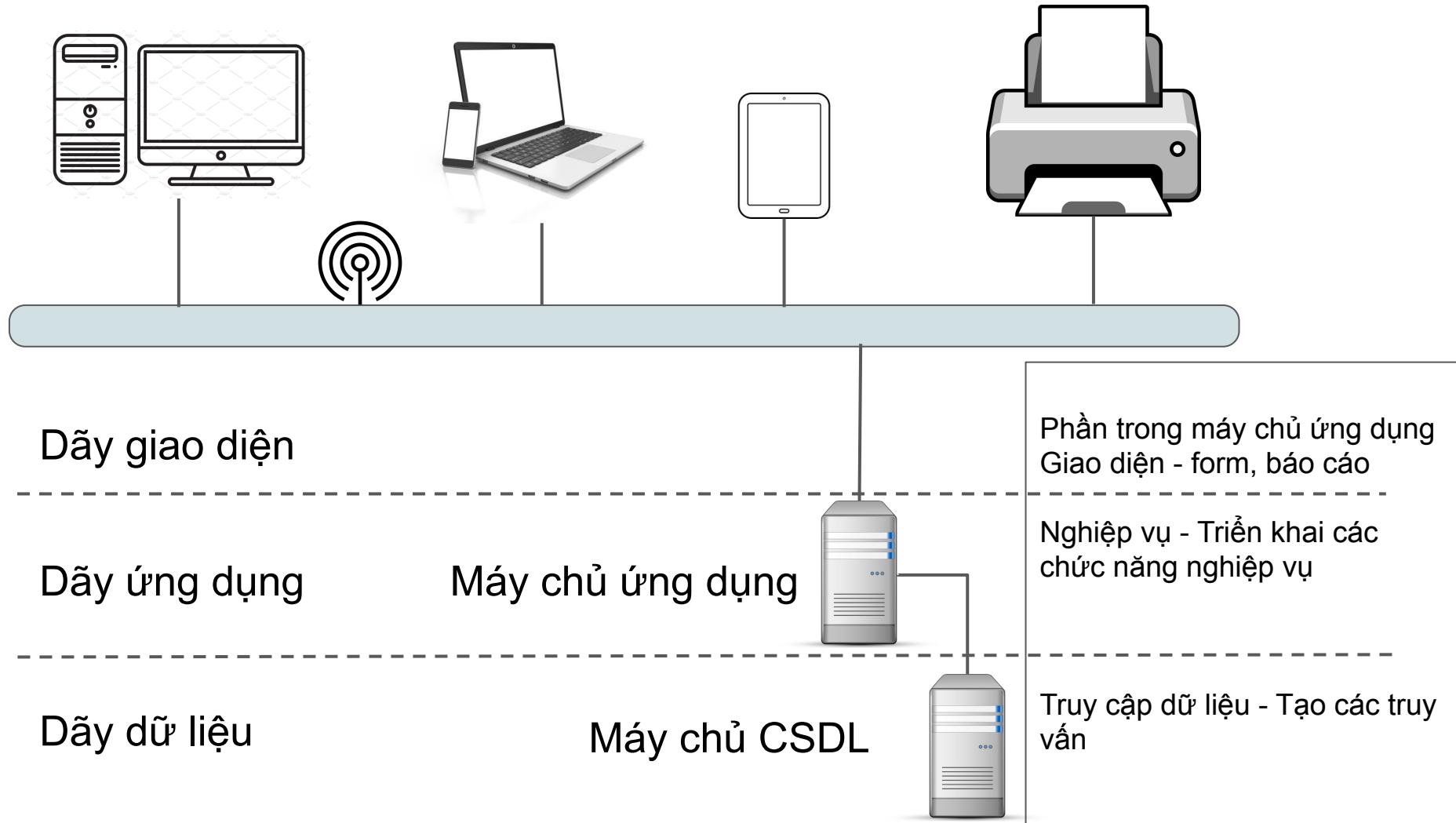
# Kiến trúc máy khách-máy chủ 4-dây



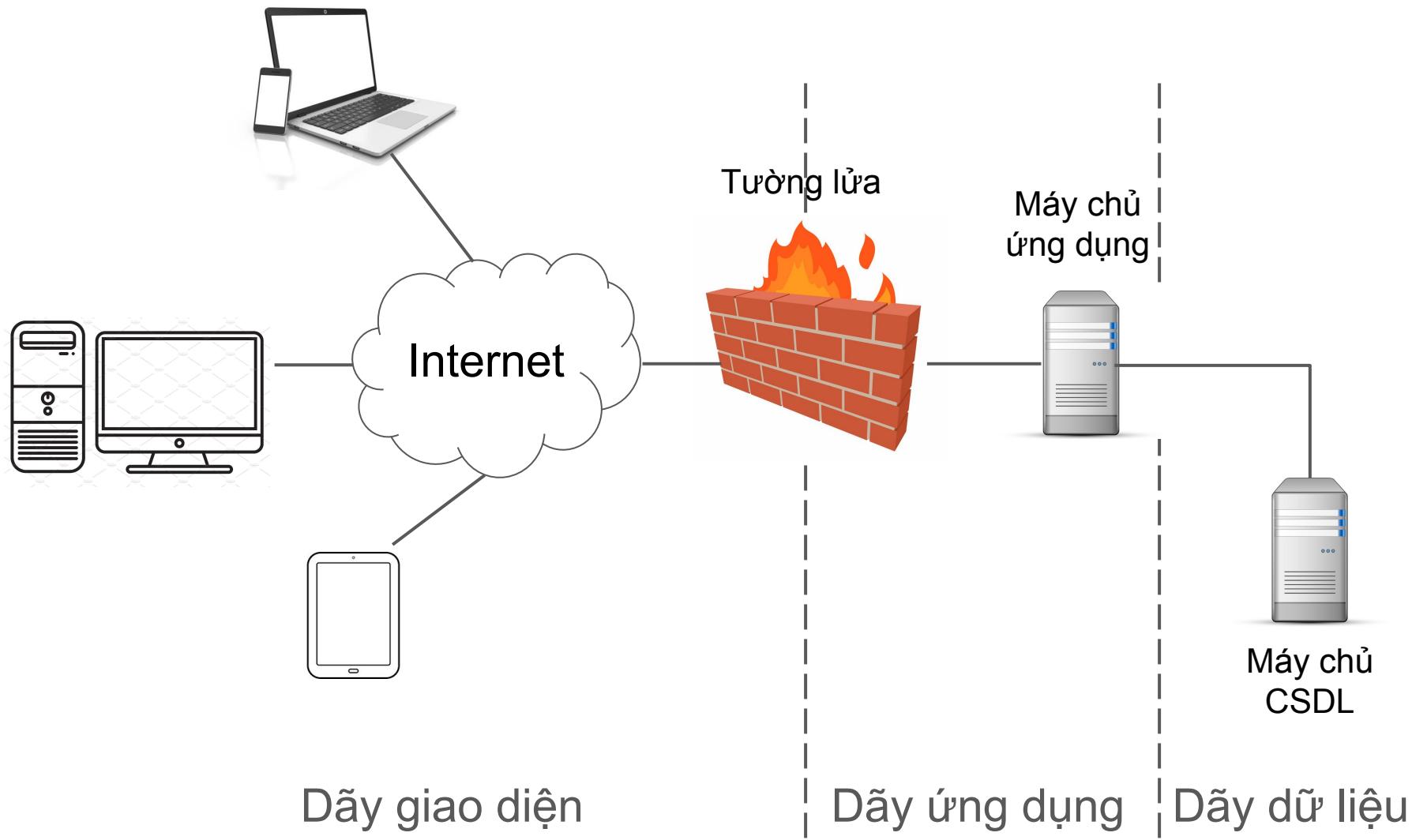
# Sơ đồ triển khai hệ thống 3-dây



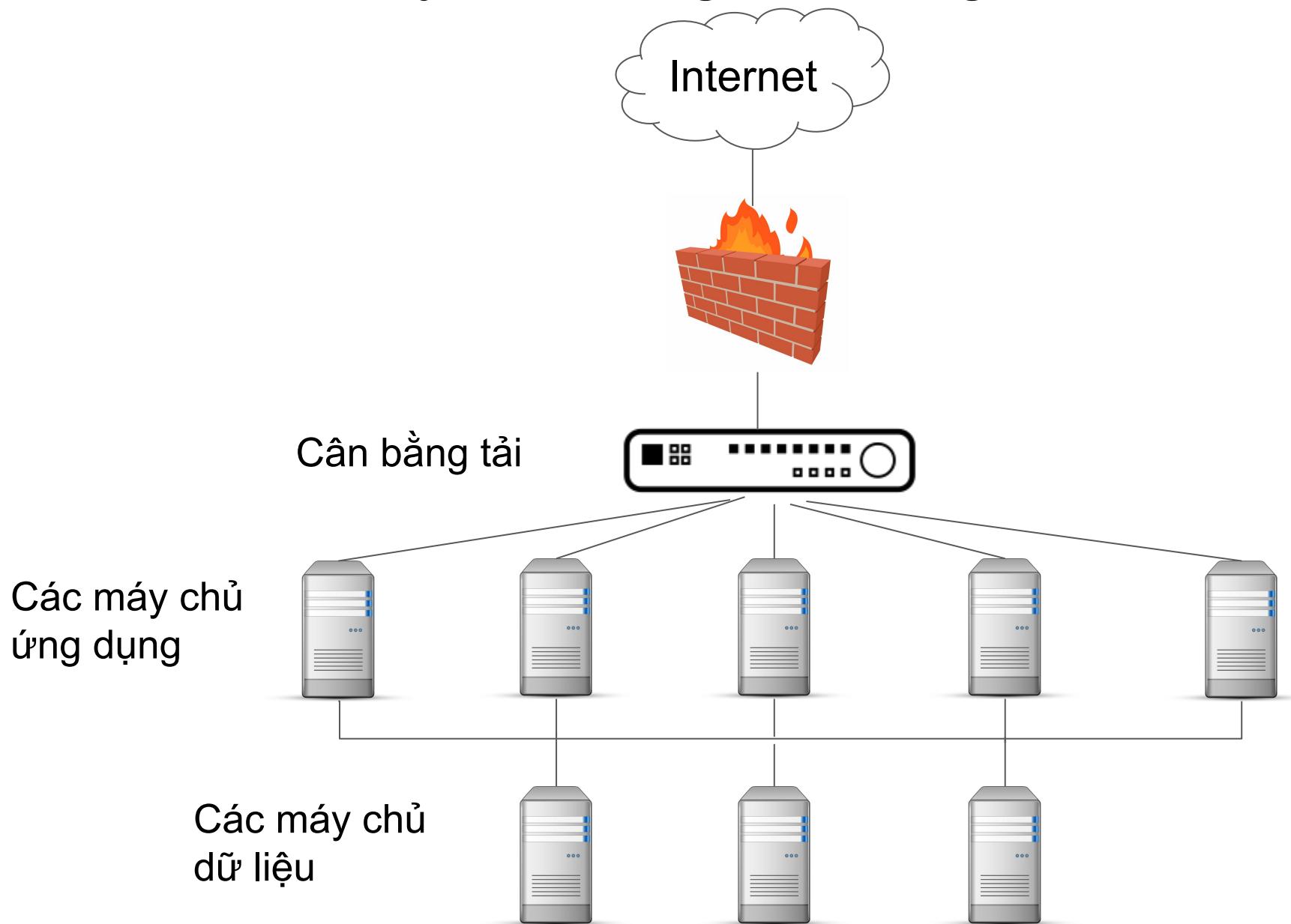
# Kiến trúc 3-dây: Triển khai nội bộ



# Kiến trúc 3-dây: Triển khai trên internet



# Kiến trúc 3-dây: Mở rộng hệ thống



# Nội dung

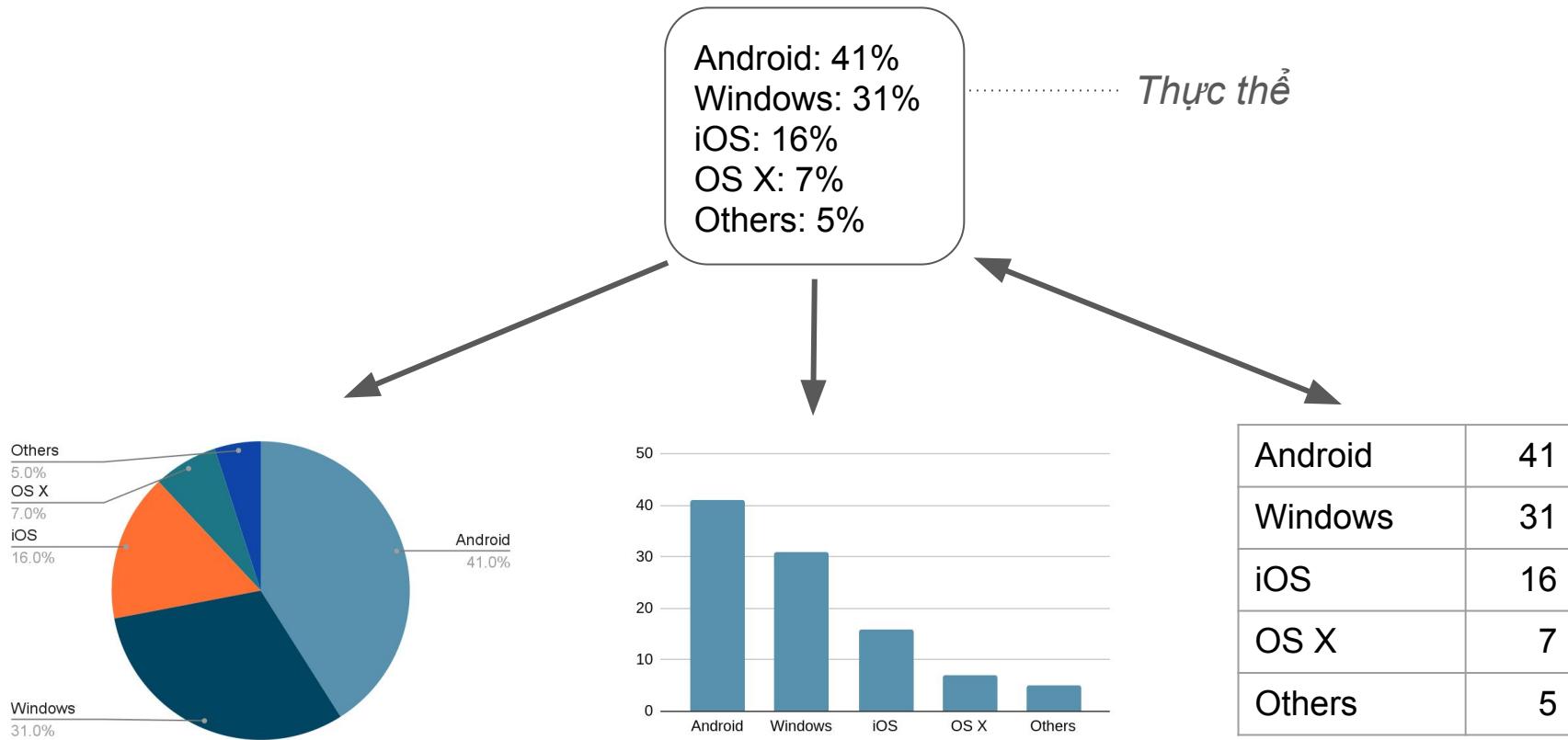
- Mô-đun hóa hệ thống
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML



# Kiến trúc MVC

- Mẫu kiến trúc Model-View-Controller (MVC) phân chia ứng dụng phần mềm thành 3 phần:
  - Thực thể/Model biểu diễn dữ liệu và xử lý nghiệp vụ.
  - Khung nhìn/View giao diện đồ họa, tương tác người dùng.
  - Điều khiển/Controller tiếp nhận và điều hướng thông điệp.
- Điều khiển có thể được tích hợp trong giao diện.
- Các thay đổi trong thực thể có thể được thông báo và cập nhật theo các cơ chế:
  - Đẩy / Push: Khi thực thể thay đổi dữ liệu được đẩy lên khung nhìn.
  - Kéo / Pull: Khi thực thể thay đổi, khung nhìn được thông báo và nó tự yêu cầu các thành phần dữ liệu.

# Ví dụ 2. Biểu diễn thị phần hệ điều hành

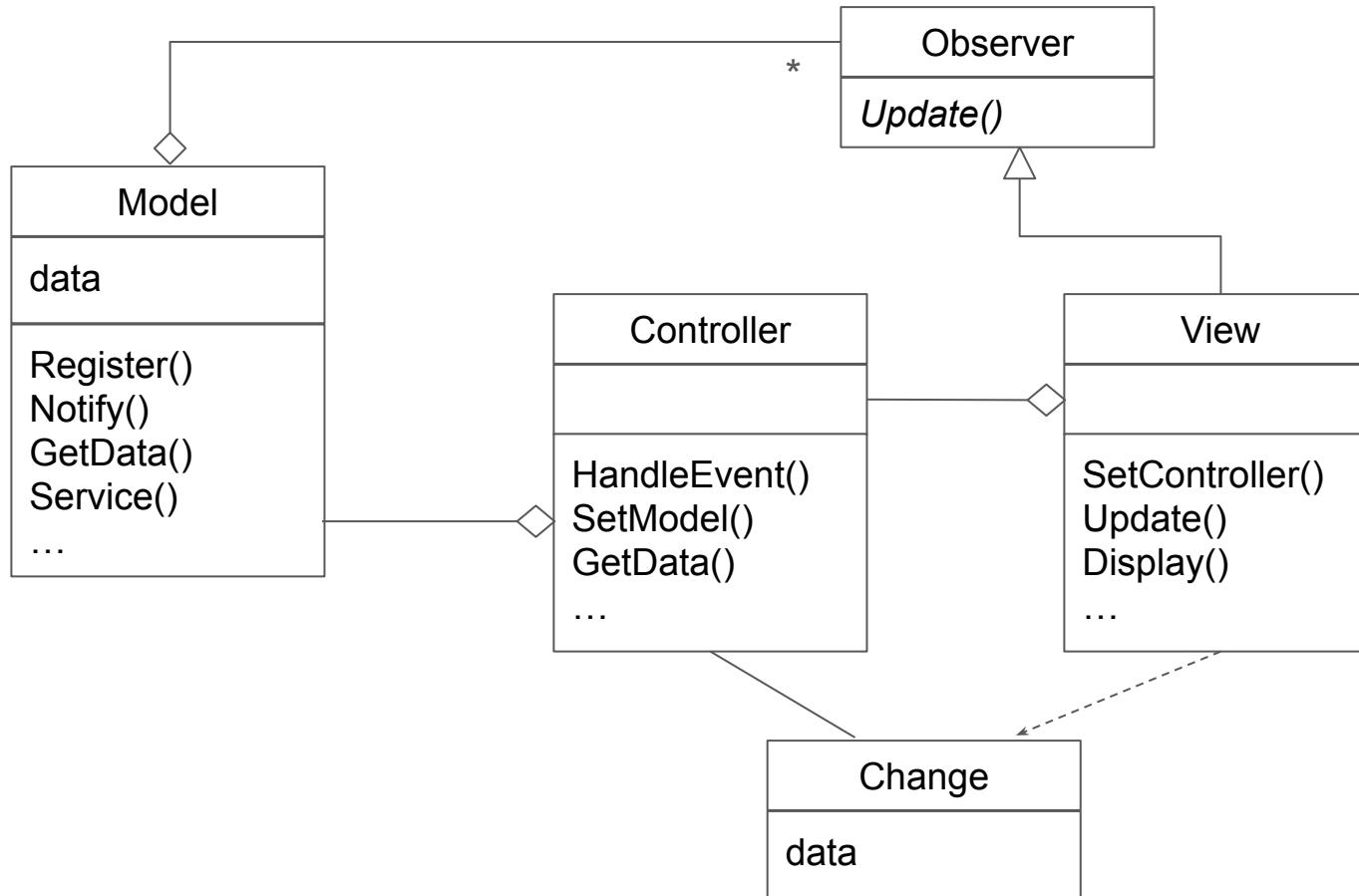


# Tùy chỉnh giao diện đồ họa

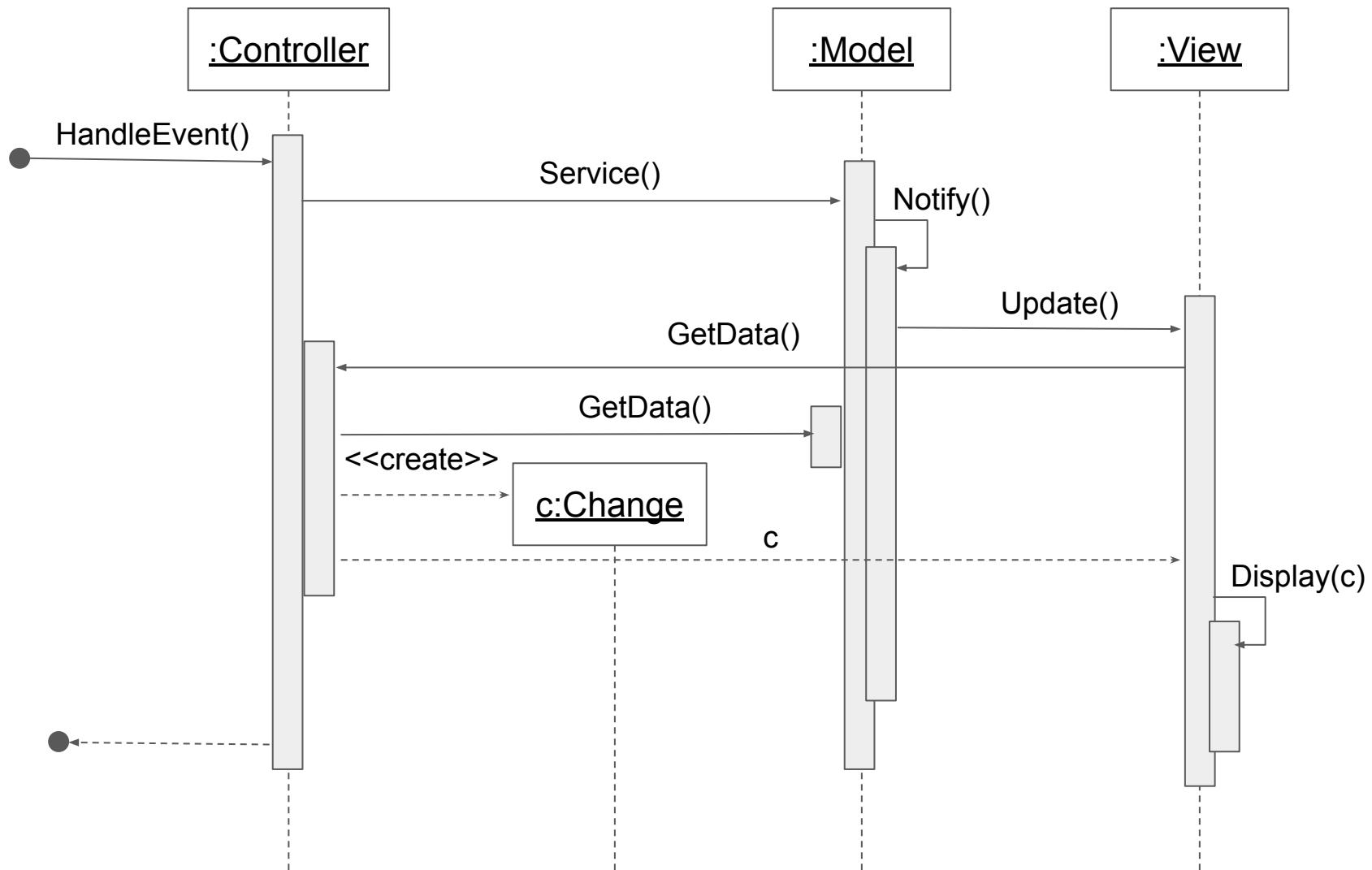
- Giao diện người dùng là phần hệ thống có nhiều nguyên nhân để thay đổi:
  - Mở rộng chức năng của hệ thống,
  - Các yêu cầu khác nhau của người dùng, v.v.
- Xây dựng phần mềm theo kiến trúc MVC cho phép giữ nguyên lô-gic ứng dụng (Thực thể) khi thay đổi giao diện, sử dụng đồng thời nhiều giao diện, và bổ xung giao diện mới.

# Ví dụ 3. MVC theo cơ chế đẩy dữ liệu

*Thực thể sẽ thông báo khung nhìn và điều khiển khi có thay đổi. Điều khiển sẽ tạo ra đối tượng dữ liệu cho khung nhìn.*



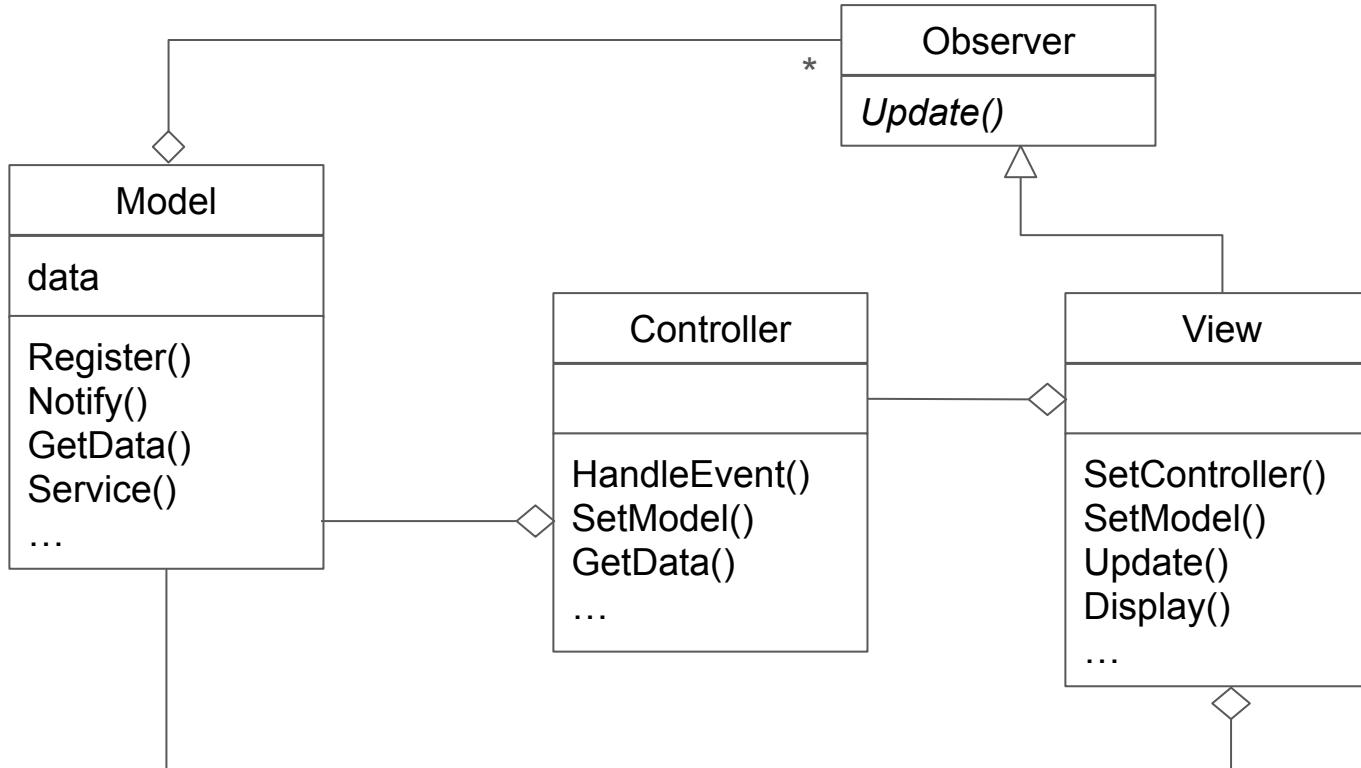
# Ví dụ 3. MVC theo cơ chế đẩy dữ liệu<sub>(2)</sub>



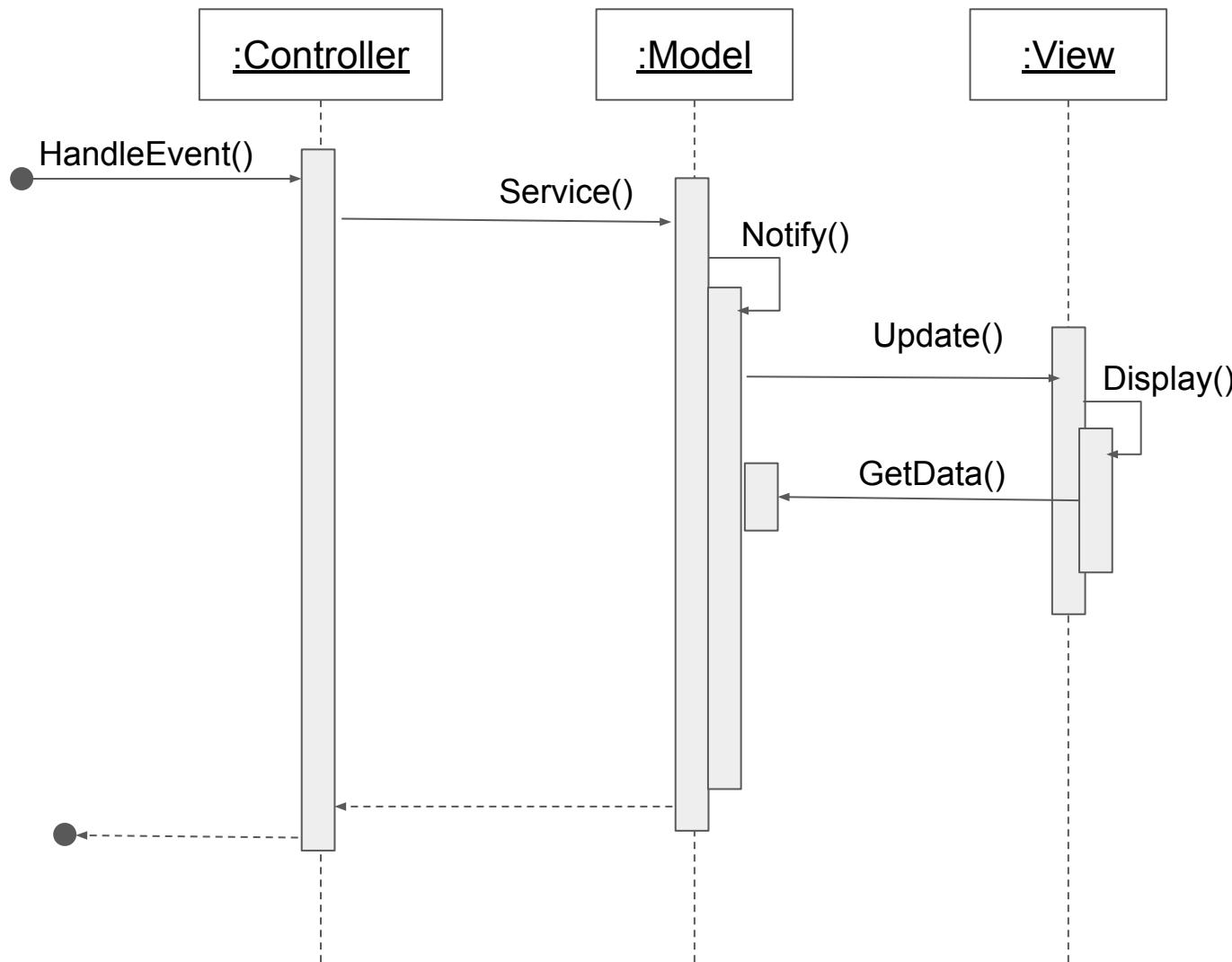
*Khung nhìn lấy dữ liệu từ đối tượng được tạo bởi điều khiển*

# Ví dụ 4. MVC theo cơ chế kéo dữ liệu

*Khung nhìn đọc dữ liệu từ thực thể khi có thay đổi*



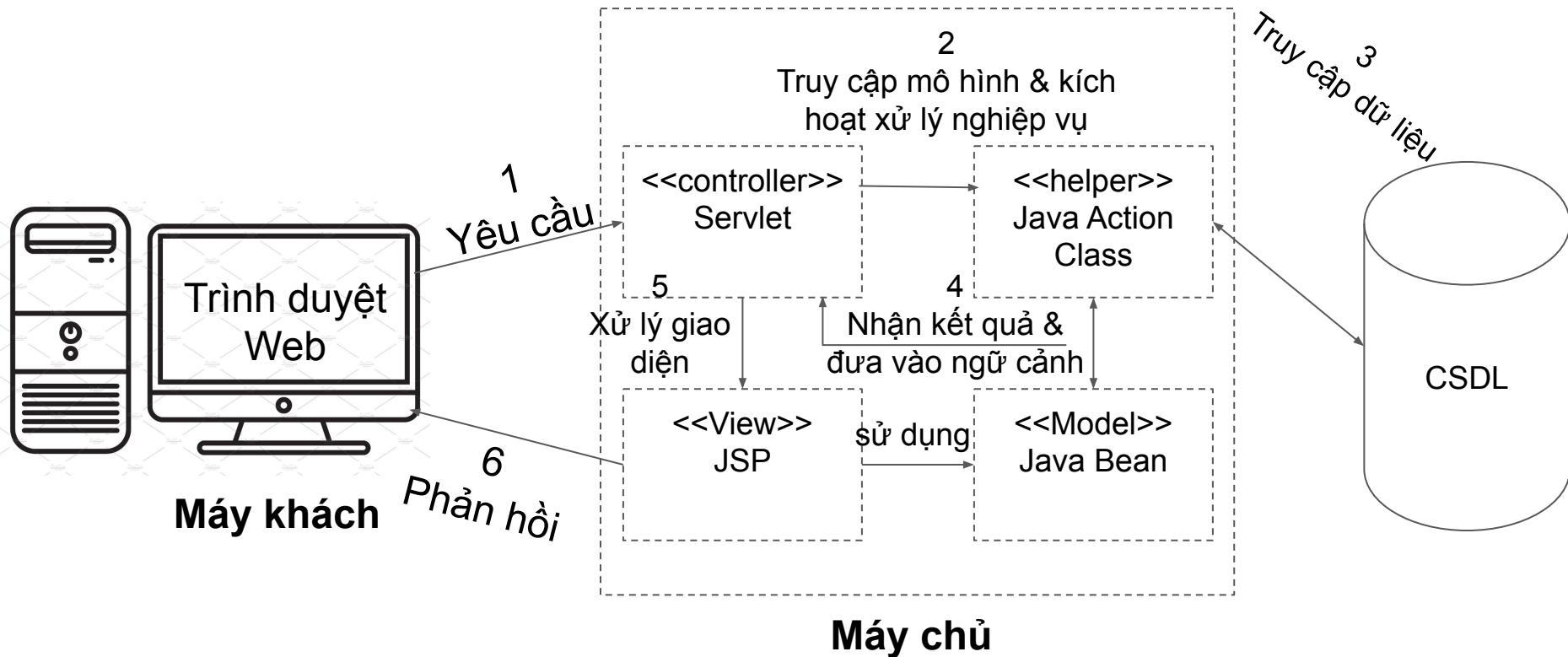
# Ví dụ 4. MVC theo cơ chế kéo dữ liệu(2)



*Khung nhìn đọc dữ liệu từ thực thể khi cần cập nhật*

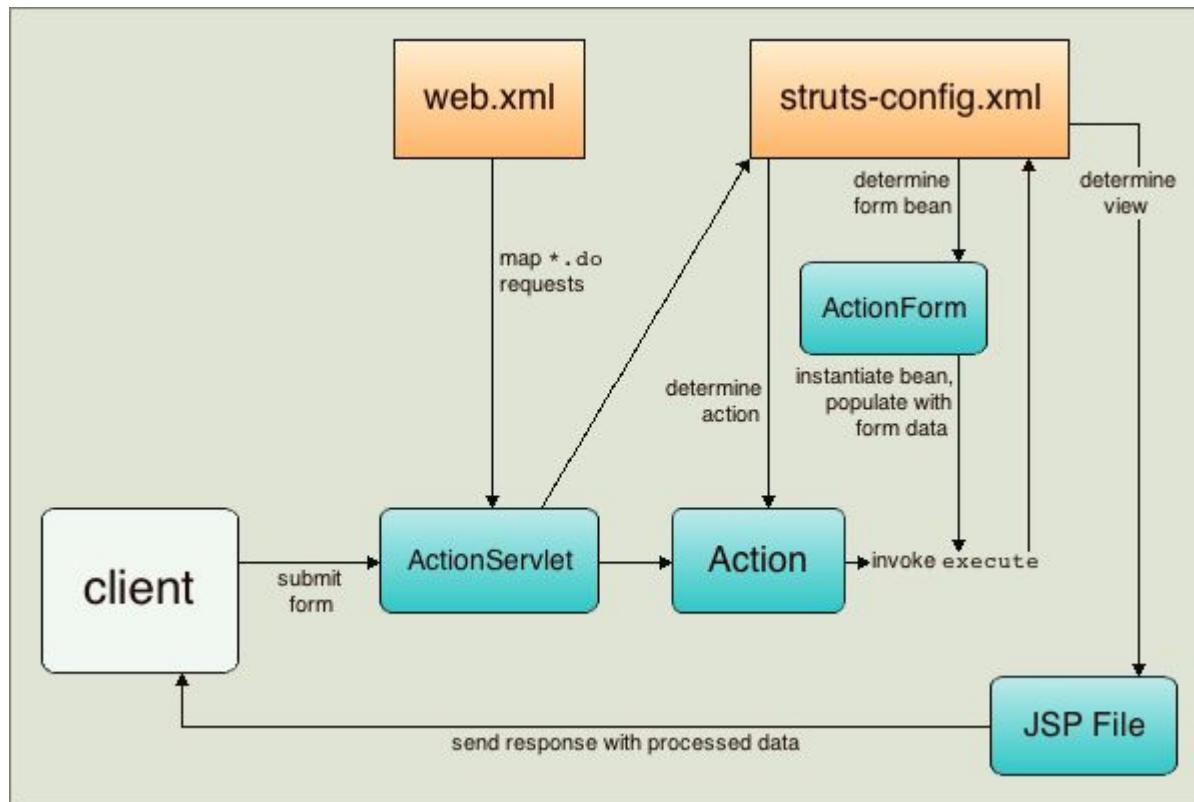
# Kiến trúc JSP Model 2

Tách biệt lô-gic trình diễn và lô-gic ứng dụng tương tự như trong MVC.



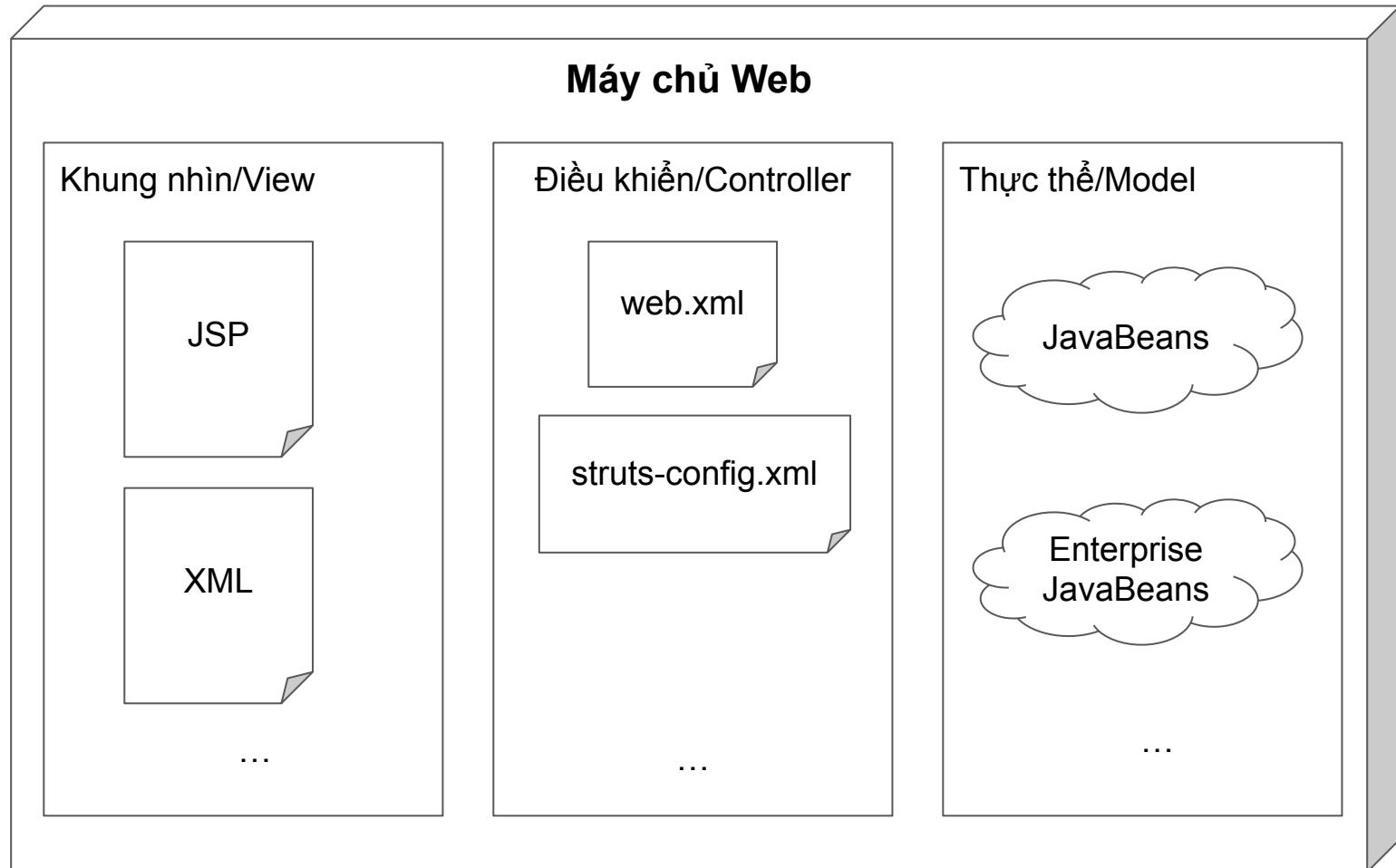
\*JSP Model 2 không định nghĩa định dạng cụ thể của Model

# Nền tảng Struts

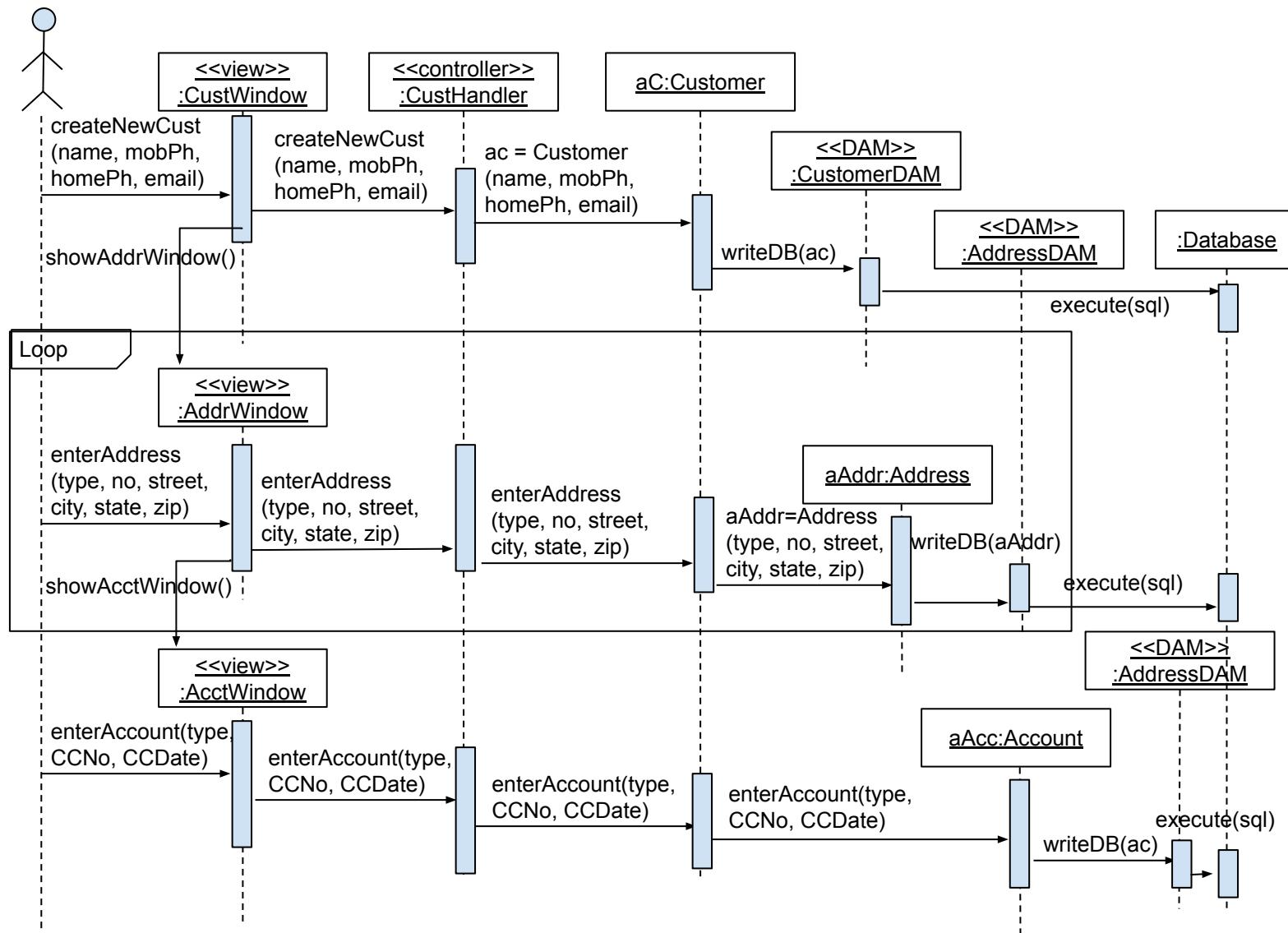


[apache.org]

# JSP Model 2 và MVC



# Sơ đồ tương tác thực tế



# Nội dung

- Mô-đun hóa hệ thống
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML

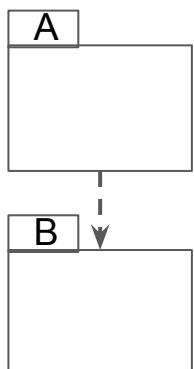
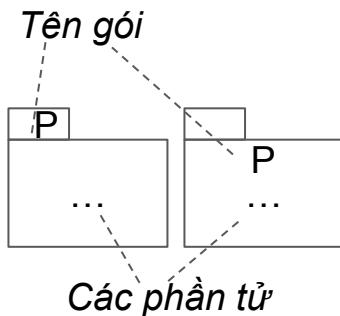


# Nội dung

- Mô-đun hóa hệ thống
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML
  - Sơ đồ gói
  - Sơ đồ thành phần
  - Sơ đồ triển khai

# Sơ đồ gói: Hệ ký hiệu

*Sơ đồ gói biểu diễn các gói cùng với các mối quan hệ*



- **Gói/Package:**
  - Nhóm lô-gic của nhiều phần tử, là không gian tên của các phần tử.
  - Có thể gom nhiều phần tử liên quan thành 1 mô-đun bậc cao, giúp giản lược mô hình.
- **Quan hệ phụ thuộc/Dependency:**
  - Nếu B thay đổi thì A cũng sẽ thay đổi theo.
  - A phụ thuộc vào B được biểu diễn bằng mũi tên nét đứt từ A tới B.

# Sơ đồ gói: Hệ ký hiệu<sub>(2)</sub>

<<merge>>

----->

- Quan hệ hợp nhất/Merge:
  - Nội dung của gói đích (theo mũi tên) được hợp nhất với nội dung của gói nguồn.
  - Các phần tử cùng tên cũng được hợp nhất.
- Quan hệ nhập/Import
  - Thêm các phần tử của gói đích vào gói nguồn.
  - Các phần tử được thêm vào có thể được nhìn thấy từ bên ngoài / nhập công khai.
- Quan hệ truy cập/access
  - Thêm các phần tử của gói đích vào gói nguồn
  - Các phần tử được thêm vào không được nhìn thấy từ bên ngoài / nhập riêng tư

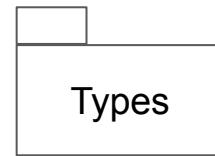
<<import>>

----->

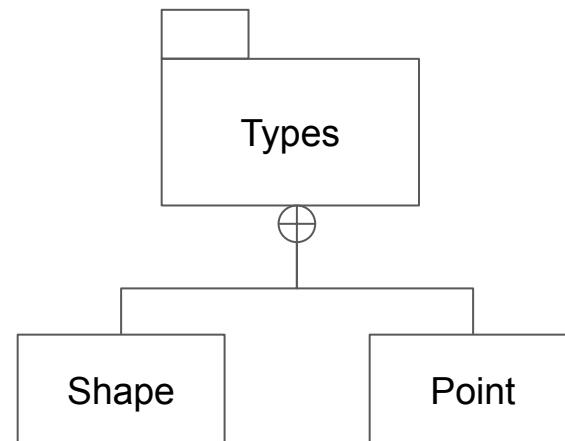
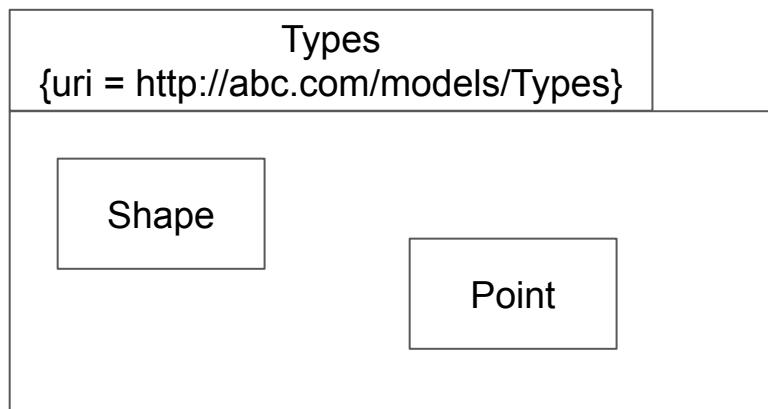
<<access>>

----->

# Biểu diễn phần tử thuộc gói

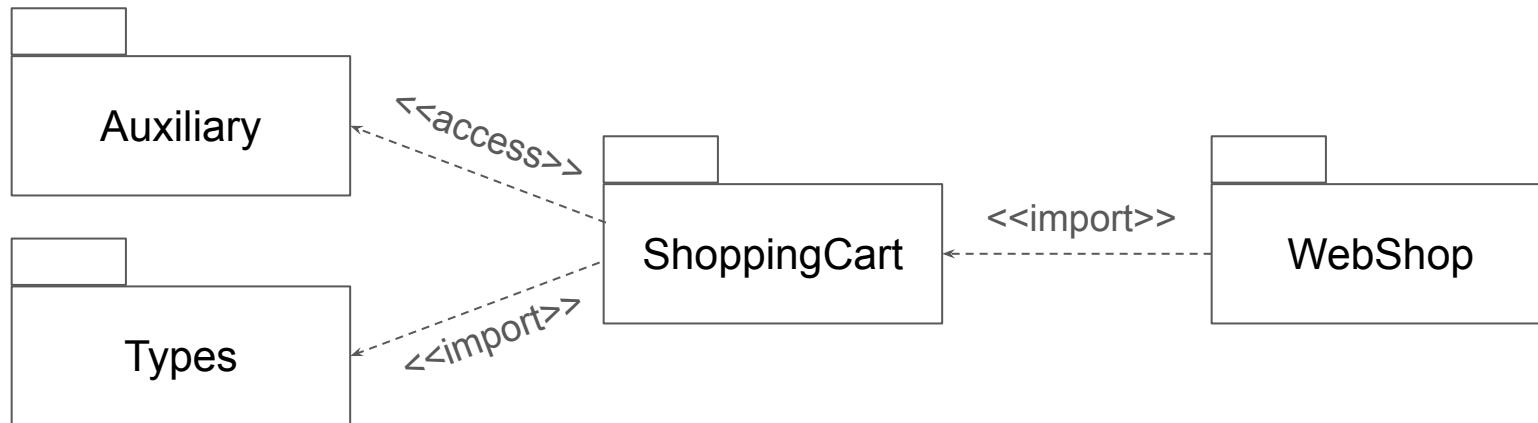


Ân nội dung của  
gói Types



Các lựa chọn biểu diễn Shape và Point thuộc gói Types

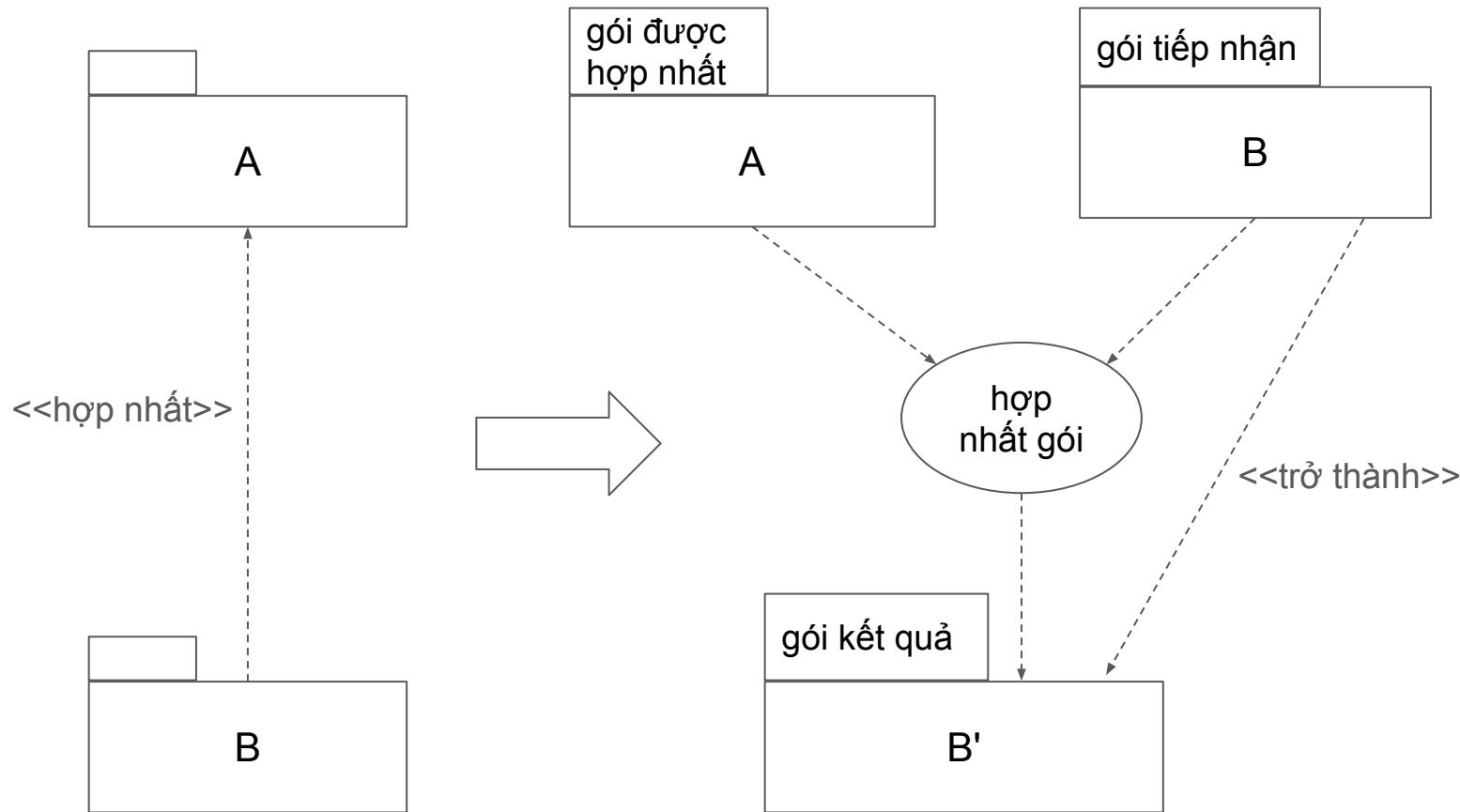
# Quan hệ nhập - <<import>>



Các phần tử của gói Types được nhập vào gói ShoppingCart và sau đó được nhập vào WebShop.

Các phần tử của Auxiliary chỉ có thể truy cập được từ ShoppingCart, nhưng không thể được từ WebShop.

# Quan hệ hợp nhất - <<merge>>



Hợp nhất các phần tử của A vào trong B.

# Nội dung

- Mô-đun hóa hệ thống
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML
  - Sơ đồ gói
  - Sơ đồ thành phần
  - Sơ đồ triển khai

# Sơ đồ thành phần

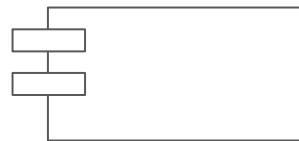
*Sơ đồ thành phần* biểu diễn các thành phần khác nhau của hệ thống cùng với các giao diện và các mối quan hệ.

- Thành phần/component biểu diễn một mô-đun hệ thống, đóng gói đầy đủ nội dung của nó.
- Hành vi của thành phần được xác định bằng các giao diện được cung cấp và được yêu cầu.
- Mỗi thành phần trong môi trường của nó có thể được thay thế bằng thành phần khác với các giao diện tương thích.
- Mỗi thành phần có thể được thể hiện như một hoặc nhiều thành phẩm.

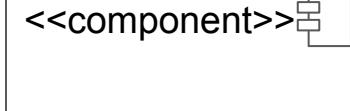
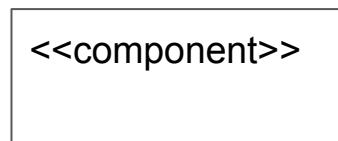
# Sơ đồ thành phần: Hệ ký hiệu

- Biểu diễn thành phần:

UML 1.X



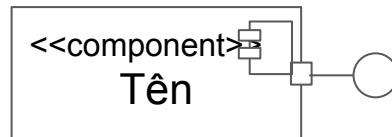
UML 2.X



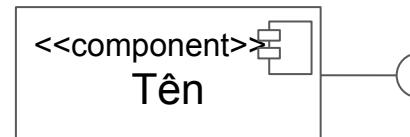
- Các giao diện được biểu diễn bằng hình tròn (cây kẹo mút - cung cấp) và nửa đường tròn (ô cắm - yêu cầu).



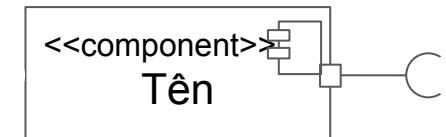
Triển khai giao diện



Cung cấp cổng

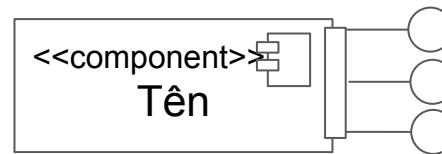


Sử dụng giao diện



Yêu cầu cổng

**Giao diện có thể được gắn với cổng**



Thành phần với cổng phức tạp

- Các quan hệ:

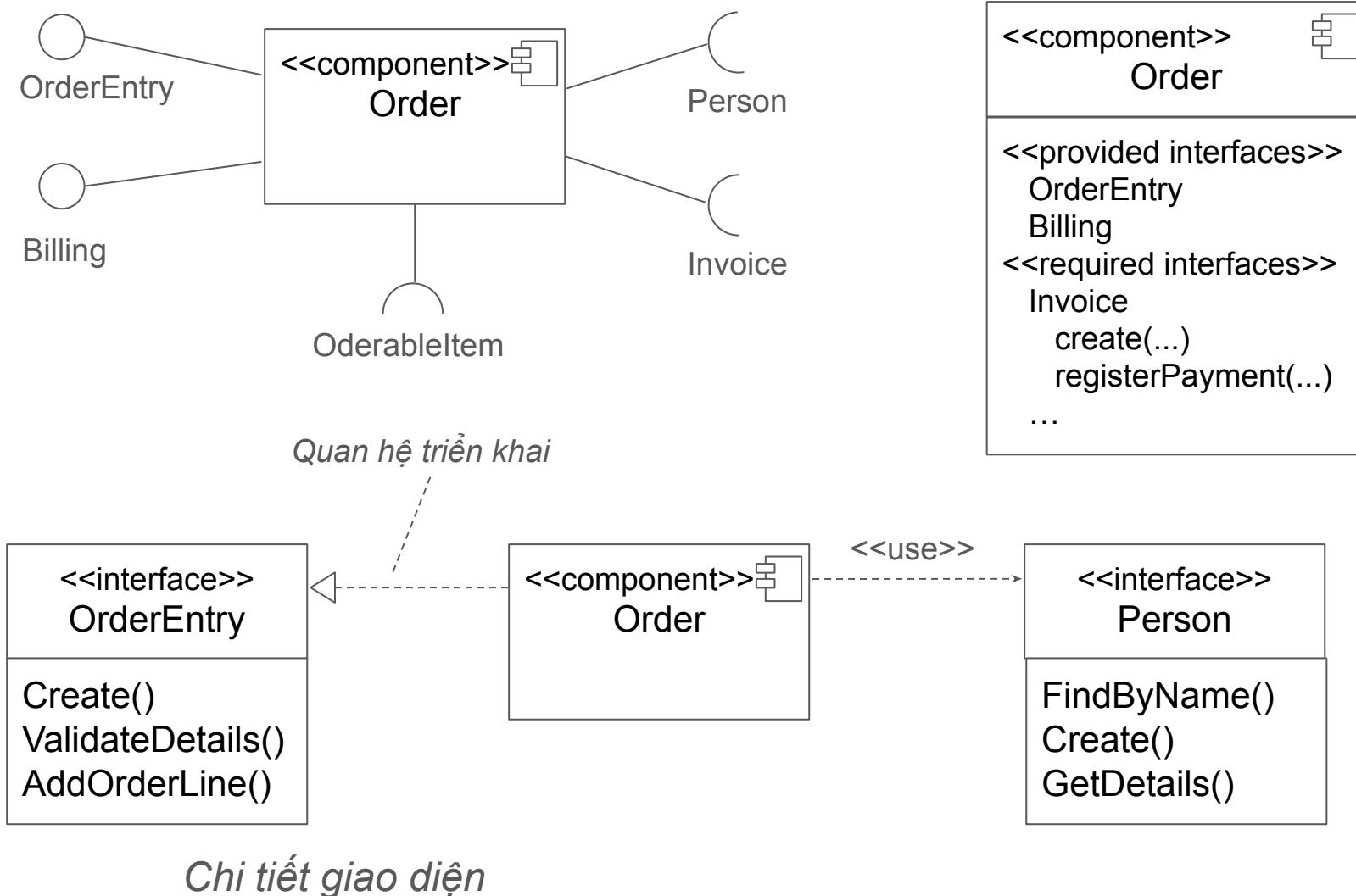
-----→ Phụ thuộc/ Sử dụng

-----→ Triển khai

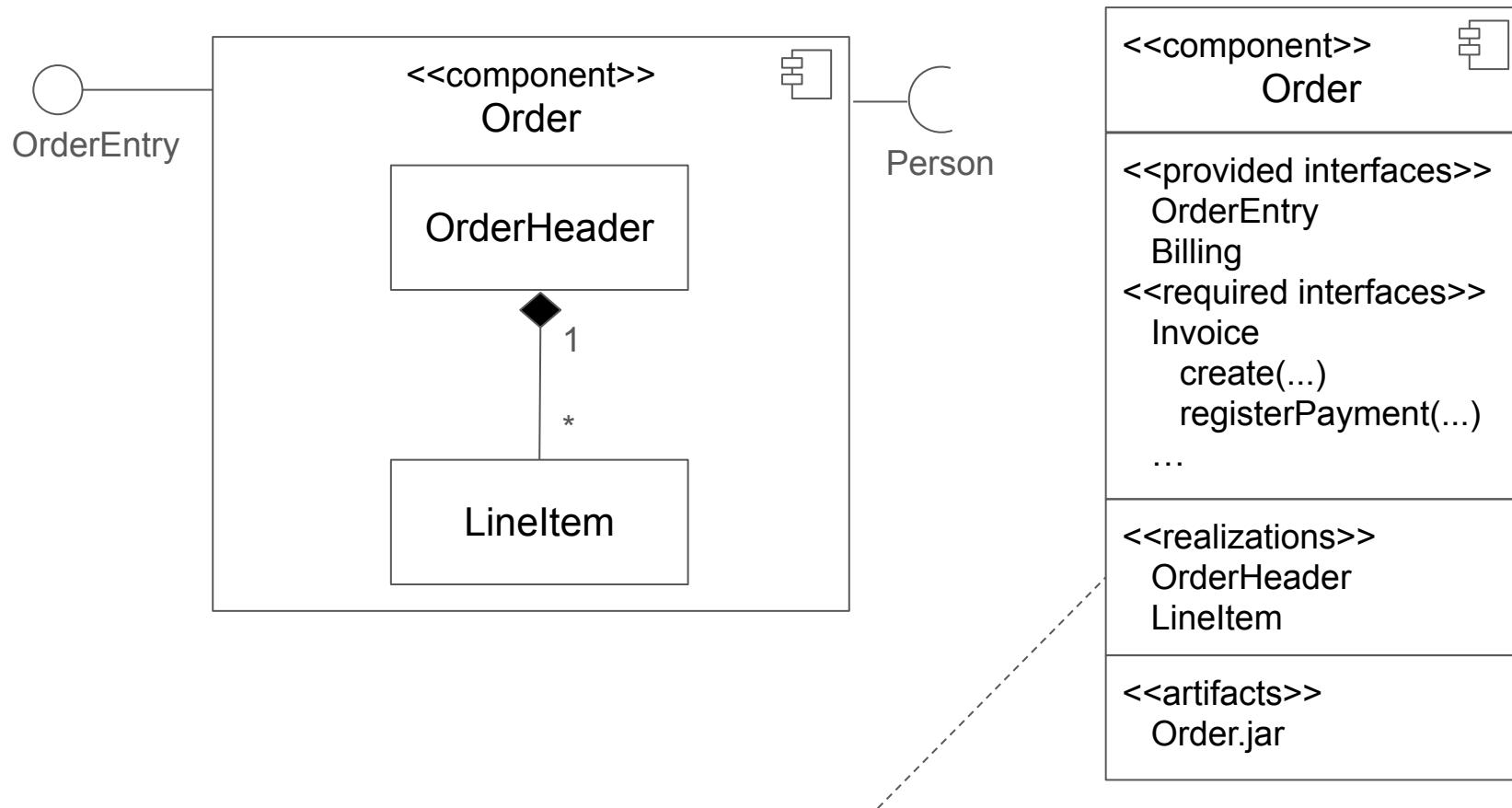
———○—— Lắp ráp

———— Chuyển tiếp

# Biểu diễn thành phần và các giao diện

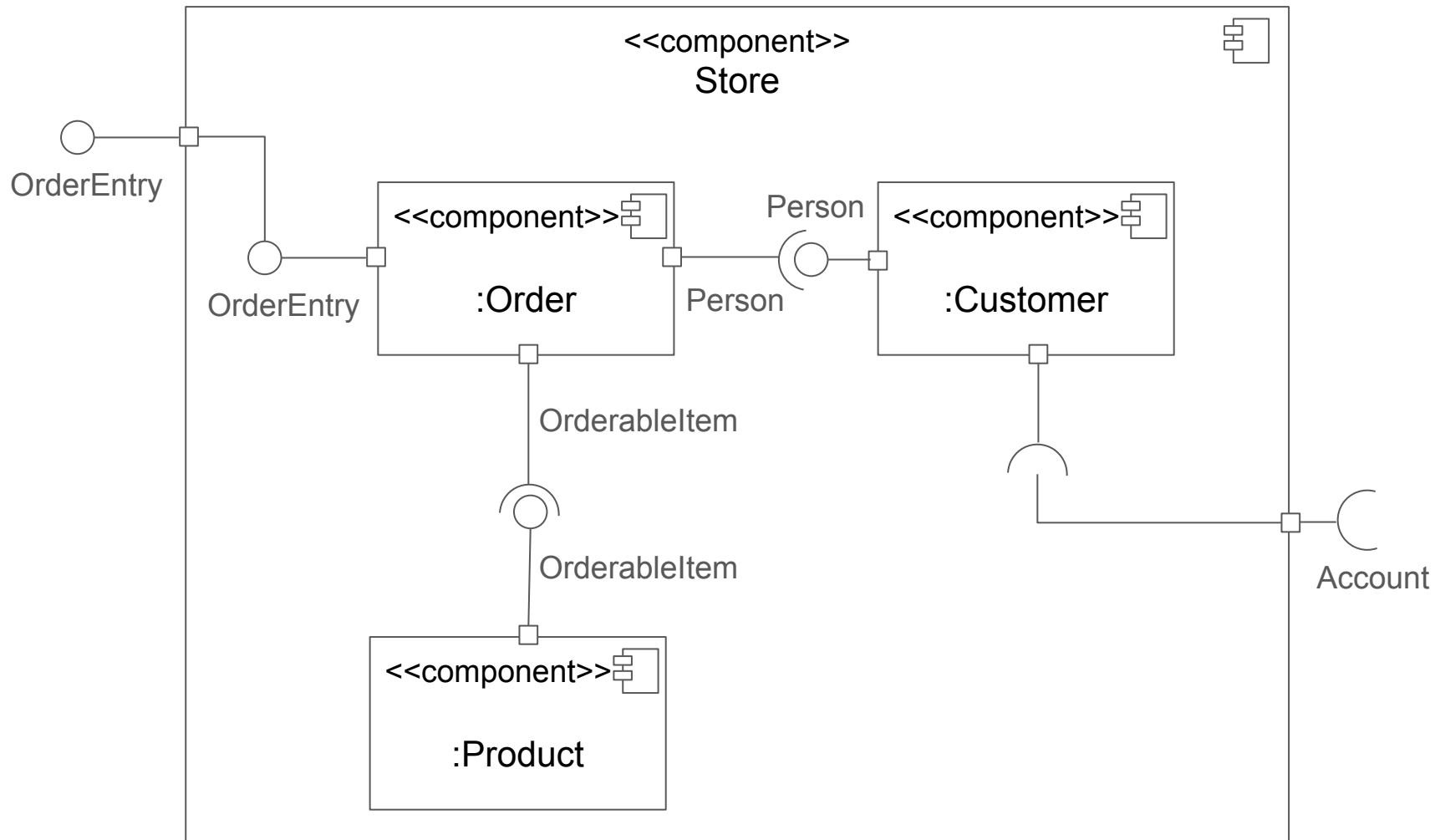


# Biểu diễn kết cấu bên trong thành phần

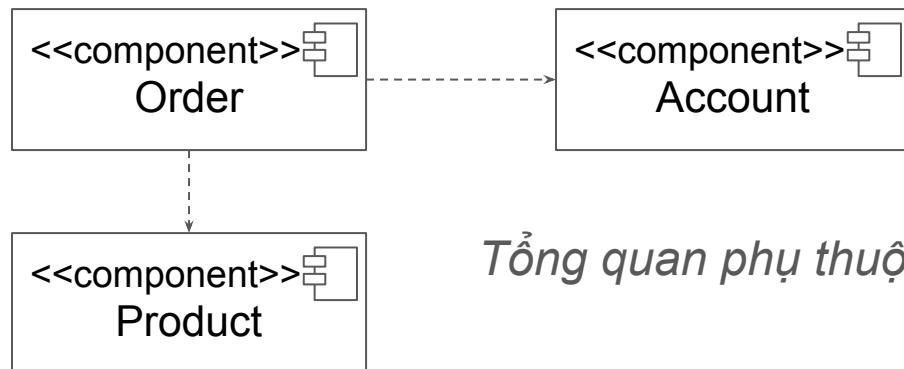


Các ngăn được bỗng xung cho các khía cạnh khác nhau

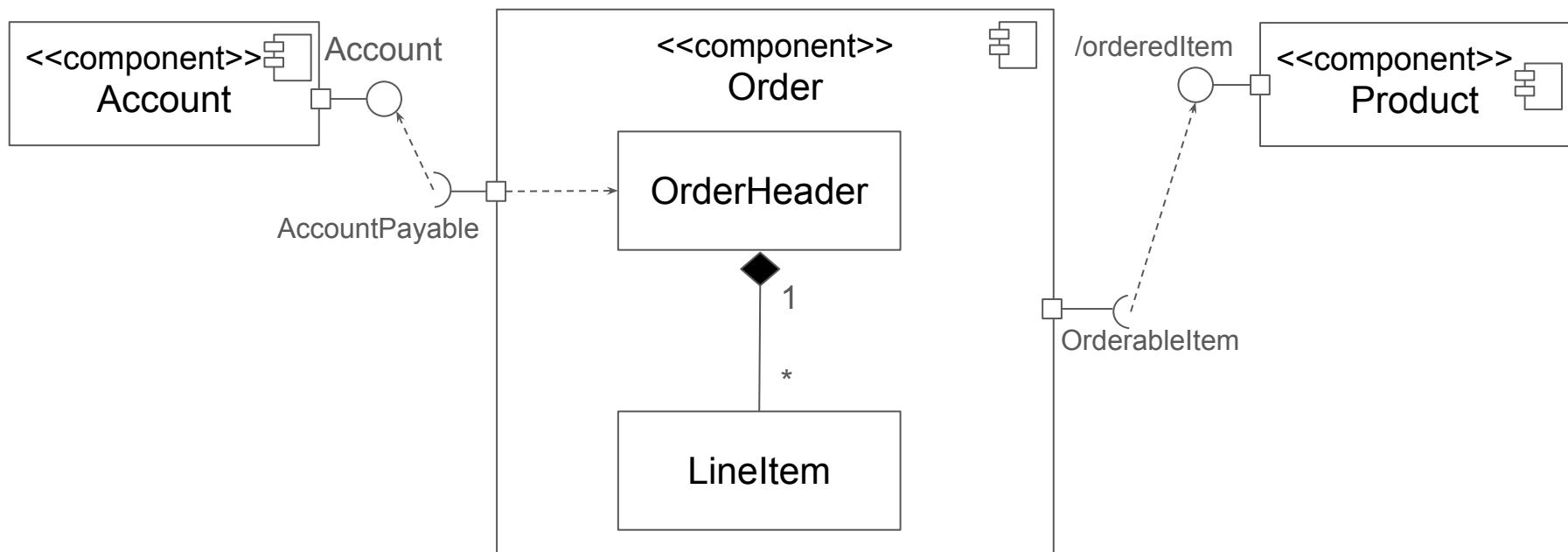
# Chuyển tiếp giao diện bên trong thành phần



# Biểu diễn phụ thuộc

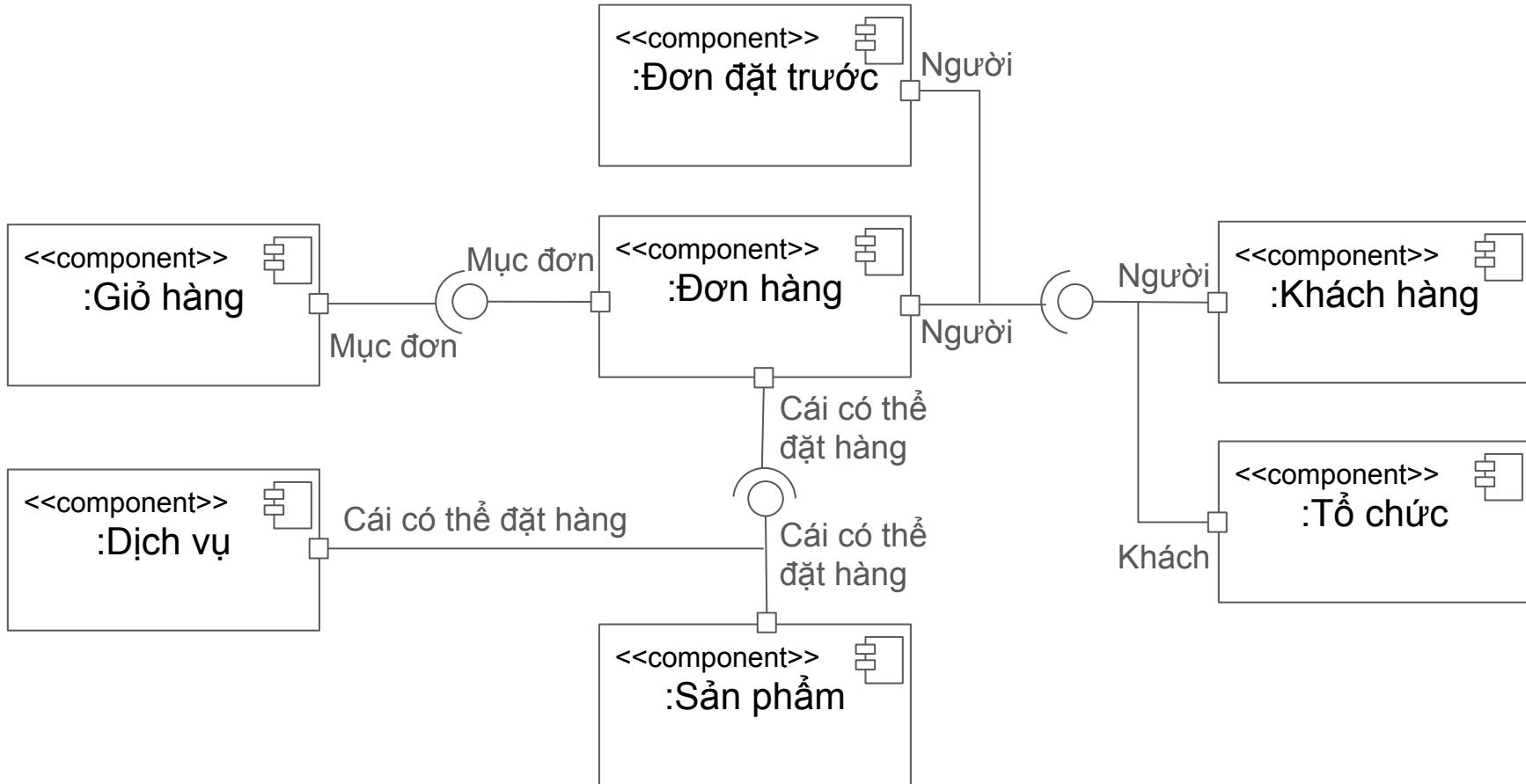


Tổng quan phụ thuộc

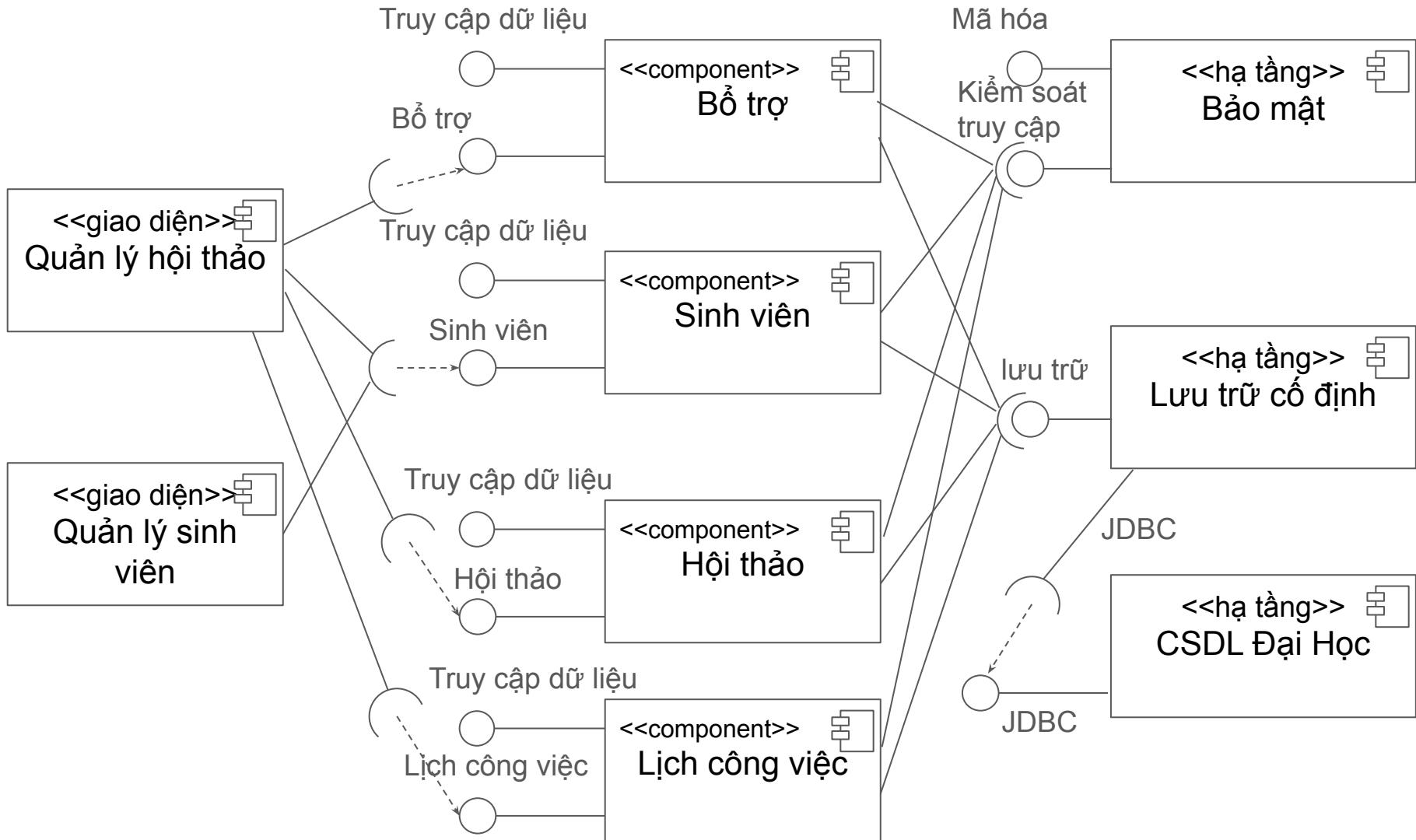


Giao diện được yêu cầu phụ thuộc vào giao diện được cung cấp

# Ví dụ 5. Kết cấu từ nhiều thành phần



# Ví dụ 6. Sơ đồ thành phần trong UML 2.x



# Nội dung

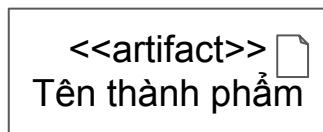
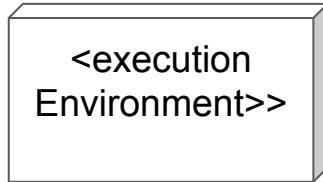
- Mô-đun hóa hệ thống
- Các thành phần kiến trúc Hệ thống
- Mẫu kiến trúc
- Các sơ đồ UML
  - Sơ đồ gói
  - Sơ đồ thành phần
  - Sơ đồ triển khai



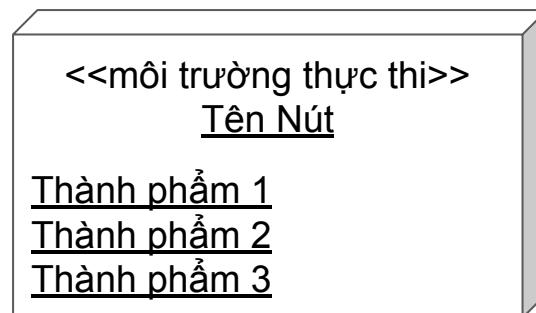
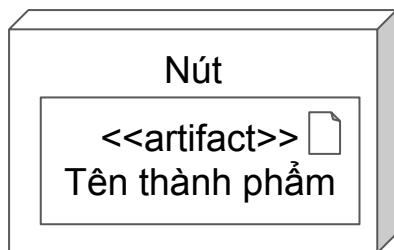
# Sơ đồ triển khai

- Biểu diễn kiến trúc vật lý, mối quan hệ giữa phần cứng và phần mềm của hệ thống được triển khai, mô tả cách hệ thống tương tác với môi trường bên ngoài.
- Các nút
  - Thường là các thiết bị tính toán: Máy chủ, Máy vi tính, điện thoại thông minh, v.v..
  - Môi trường thực thi: Linux, macOS, Windows, ...
  - Chứa các thành phần phần mềm
- Các liên kết
  - Biểu diễn kênh trao đổi dữ liệu, kết nối mạng, giao thức.

# Sơ đồ triển khai: Hệ ký hiệu



- Nút là tài nguyên tính toán:
  - Thiết bị hoặc
  - Môi trường thực thi,
  - Có thể được kết nối để biểu diễn các kênh trao đổi thông tin theo hình trạng mạng
- Thành phẩm / Artifact:
  - Cấu phần cụ thể của hệ thống
  - Các định dạng:
    - <<mã nguồn>> / <<source>>, <<tệp thực thi>> / <<executable>>, <<jar>>, v.v..
- Nút với thành phẩm được triển khai:



# Sơ đồ triển khai: Hệ ký hiệu<sub>(2)</sub>

- Đặc tả triển khai
  - Đặc tả 1 tập thuộc tính xác định các tham số thực thi của thành phẩm được triển khai trên nút.
  - Đặc tả triển khai có thể chứa thuộc tính.

<<đặc tả triển khai>>  
**Tên**

<<đặc tả triển khai>>  
**Tên**

<<đặc tả triển khai>>  
**Tên**

execution: execKind  
transaction: Boolean

execution: thread  
transaction: true

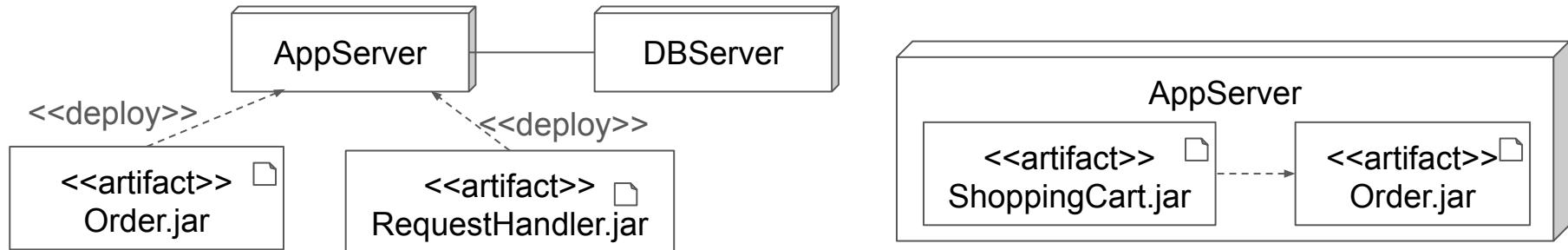
- Thành phẩm phức tạp có thể chứa thực bản của đặc tả triển khai

<<artifact>>  
Tên thành phẩm  
{execution = thread,  
transaction = true}

# Sơ đồ triển khai: Hệ ký hiệu<sub>(3)</sub>

<<deploy>>

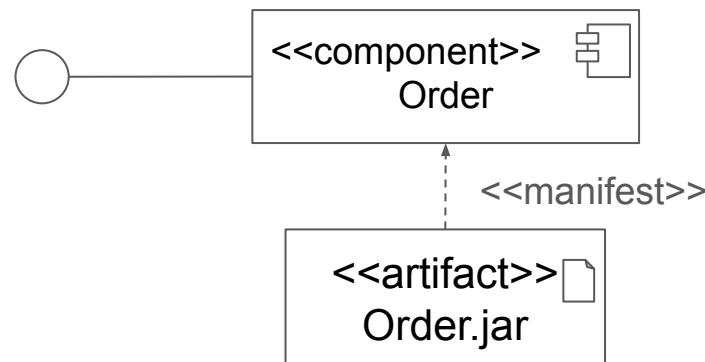
- Triển khai thành phẩm trên nút



HTTP

- Đường truyền và giao thức được biểu diễn bằng quan hệ liên kết
- Quan hệ manifest/xuất bản: Cho biết cơ sở của thành phẩm, cái hình thành nên thành phẩm.

<<manifest>>



# Sơ đồ triển khai: Thành phẩm

Thành phẩm/Artifact biểu diễn một đầu ra thực tế, được triển khai trong các nút.

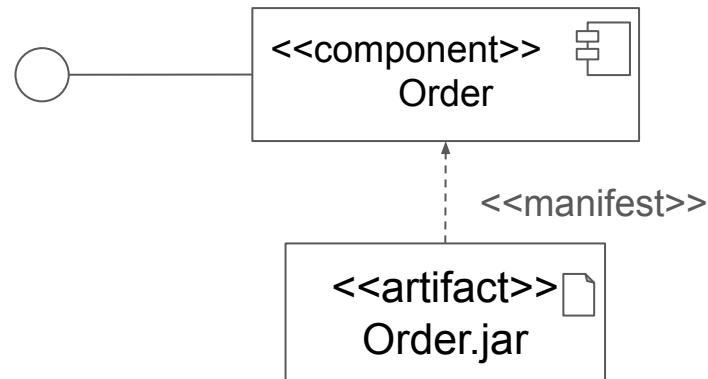


Một số định dạng thành phẩm tiêu chuẩn:

- Tệp - <<file>> một tệp cụ thể trong hệ thống tệp
- Tài liệu - <<document>> một tài liệu, có thể là mã nguồn hoặc tệp thực thi
- Thư viện - <<library>> một thư viện tĩnh hoặc động
- Tệp thực thi - <<executable>> một chương trình có thể được thực thi trên máy tính.

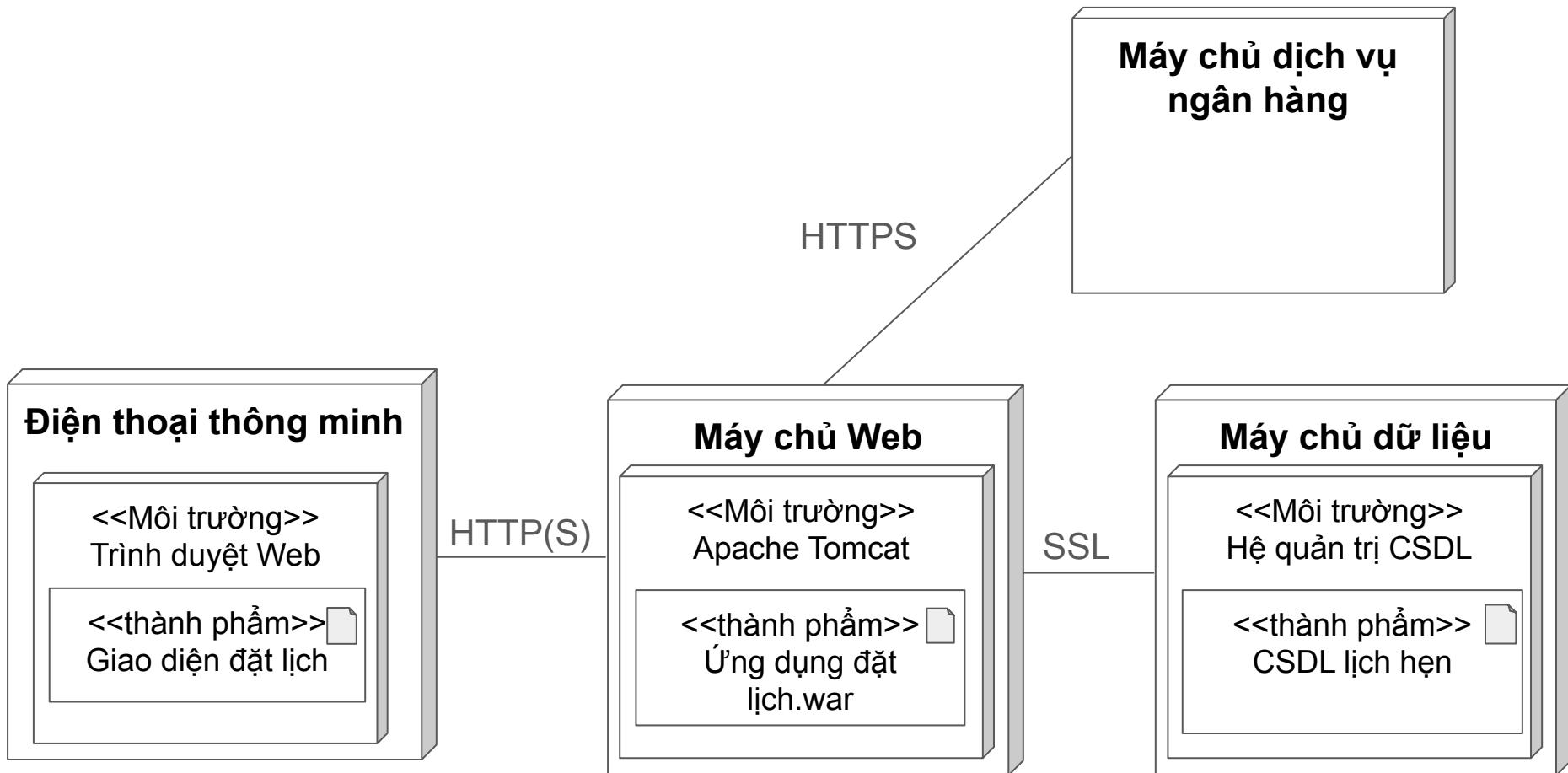
# Quan hệ manifest

- Biểu diễn triển khai vật lý của 1 hoặc nhiều cấu phần của hệ thống như các thành phẩm.
- Thường được sử dụng để biểu diễn triển khai thành phần như thành phẩm:
  - Mỗi thành phần có thể được triển khai như 1 thành phẩm
  - Mỗi thành phẩm có thể bao gồm nhiều thành phần

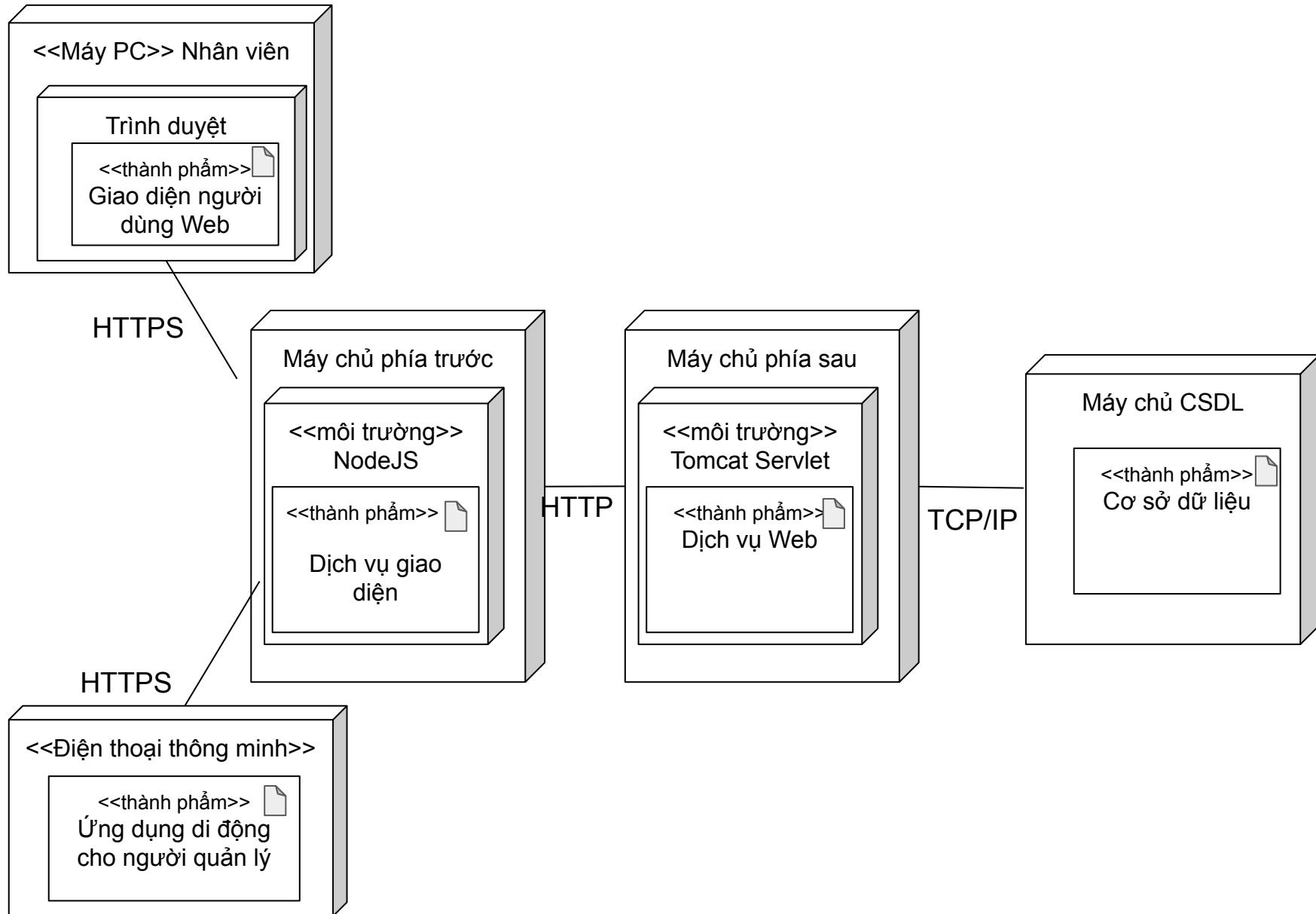


- Ngoài ra có thể sử dụng với bất kỳ phần tử nào khác nếu có thể đóng gói / PackageableElement

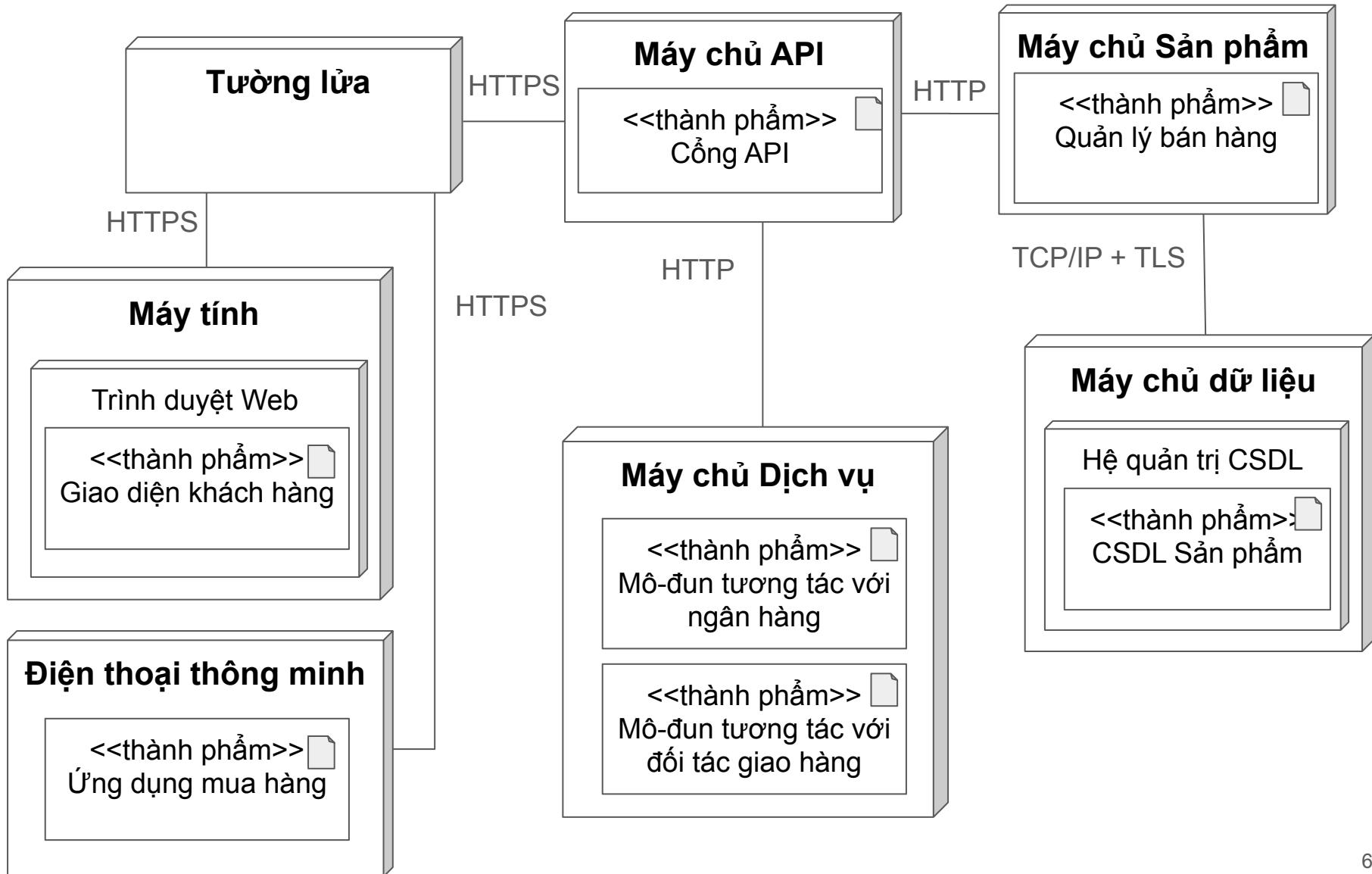
# Ví dụ 7. Sơ đồ triển khai hệ thống 3-dãy



# Ví dụ 8. Sơ đồ triển khai hệ thống 4-dây

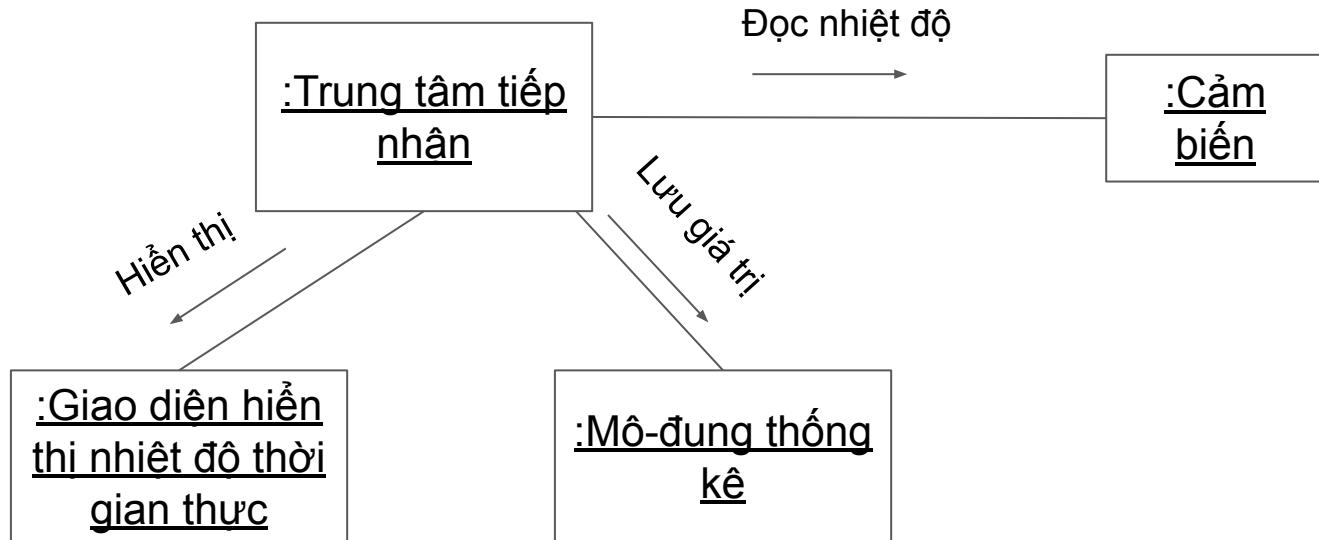


# Ví dụ 9. Sơ đồ triển khai hệ thống bán hàng



# Bài tập

Phân tích các vấn đề với kết cấu sau và phác thảo giải pháp hướng tới liên kết lỏng giữa các phần tử: Vẽ sơ đồ lớp và sơ đồ tuần tự.





# Phân tích và thiết kế hệ thống

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2022

# Nội dung

1. Tổng quan về thiết kế giao diện
2. Ngôn ngữ mô hình hóa luồng tương tác
3. Ví dụ tổng hợp

# Nội dung

- 
1. Tổng quan về thiết kế giao diện
  2. Ngôn ngữ mô hình hóa luồng tương tác
  3. Ví dụ tổng hợp

# Phân loại giao diện

*Giao diện là phương tiện để tương tác với hệ thống, theo đó i tương sử dụng các giao diện được phân loại thành:*

- Giao diện hệ thống: Hỗ trợ tương tác máy-máy, phục vụ tác nhân là hệ thống khác.
- Giao diện người dùng: Hỗ trợ tương tác người-máy, phục vụ tác nhân là người.
  - Đồng thời cũng như điện mạo hệ thống đối với người dùng - người dùng chỉ nhìn thấy các giao diện chứ không nhìn thấy các xử lý bên trong - cách tiếp cận lấy người dùng làm trọng tâm (user-centered)..

*Chúng ta sẽ tập trung vào giao diện người dùng*

# Các thành phần hỗ trợ tương tác người-máy

*Tương tác người-máy / Human-Computer Interaction (HCI)*

Thiết bị: Màn hình thường, màn hình cảm ứng, bàn phím thực, bàn phím ảo, chuột, webcam, mic, v.v..

Các giao diện người dùng: Cửa sổ, biểu mẫu, hộp thoại, hình ảnh, nút bấm, danh sách, bảng, biểu đồ, v.v..



# Các nguyên tắc thiết kế giao diện

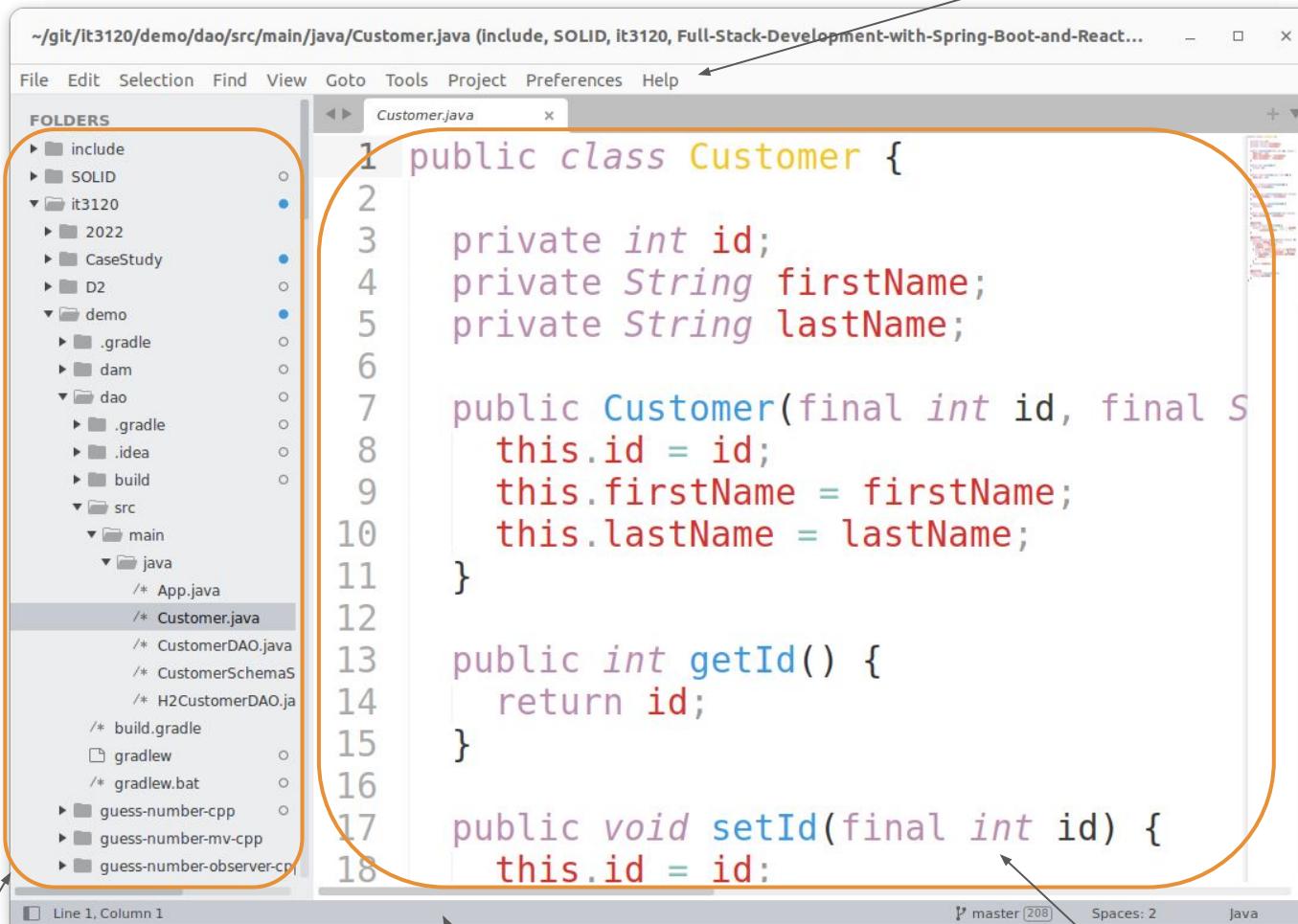
- Bố cục - Giao diện nên là 1 chuỗi các phân vùng màn hình ổn định cho các mục đích.
- Nhận biết nội dung - Người dùng nên được báo cáo về tính năng đang sử dụng và thông tin đang được hiển thị.
- Thẩm mỹ - Giao diện phải đẹp đối với người dùng, hữu ích và lôi cuốn sử dụng.
- Trải nghiệm người dùng - Giao diện phải dễ sử dụng, tự nhiên, dễ học, tạo cảm giác tích cực cho người dùng.
- Nhất quán - Các thành phần giao diện trong các ngữ cảnh khác nhau phải thống nhất - quan trọng để dễ sử dụng.
- Tiết kiệm công sức - Giao diện phải đơn giản, giúp người dùng hoàn thành công việc nhanh chóng, hạn chế lỗi.

# Bố cục

- **Bố cục/Layout:** Cách bố trí các thành phần trên màn hình
- Các thành phần được gom nhóm trong các vùng.
- Các vùng có thể tiếp tục được gom thành vùng lớn hơn
- Các vùng được bố trí theo trật tự tạo thành luồng liền mạch, tự nhiên.
  - Trật tự tự nhiên có thể thay đổi theo đối tượng người dùng, phụ thuộc vào các yếu tố văn hóa;
  - Chúng ta đọc từ trái sang phải, từ trên xuống dưới,
  - ... tuy nhiên cũng có những nơi đọc từ phải sang trái.

## Ví dụ 1. Bố cục giao diện

## Các danh mục chức năng



# Cây thư mục

## Thanh trạng thái

## Vùng soạn thảo

# Nhận biết nội dung

Giúp người dùng xác định các thông tin đang xử lý

- Sử dụng tiêu đề cửa sổ, các nhãn vùng, nhãn tên trường dữ liệu v.v..
- Trực quan tách biệt các vùng
- Sử dụng các danh mục (menu) ngũ cǎnh
- v.v...

# Ví dụ 2. Nhận biết nội dung

Các đường dẫn giúp xác định tệp hiện hành

Sơ đồ, chỉ số dòng hỗ trợ định vị nội dung

```
~/git/it3120/demo/dao/src/main/java/Customer.java (include, SOLID, it3120, Full-Stack-Development-with-Spring-Boot-and-React...)
```

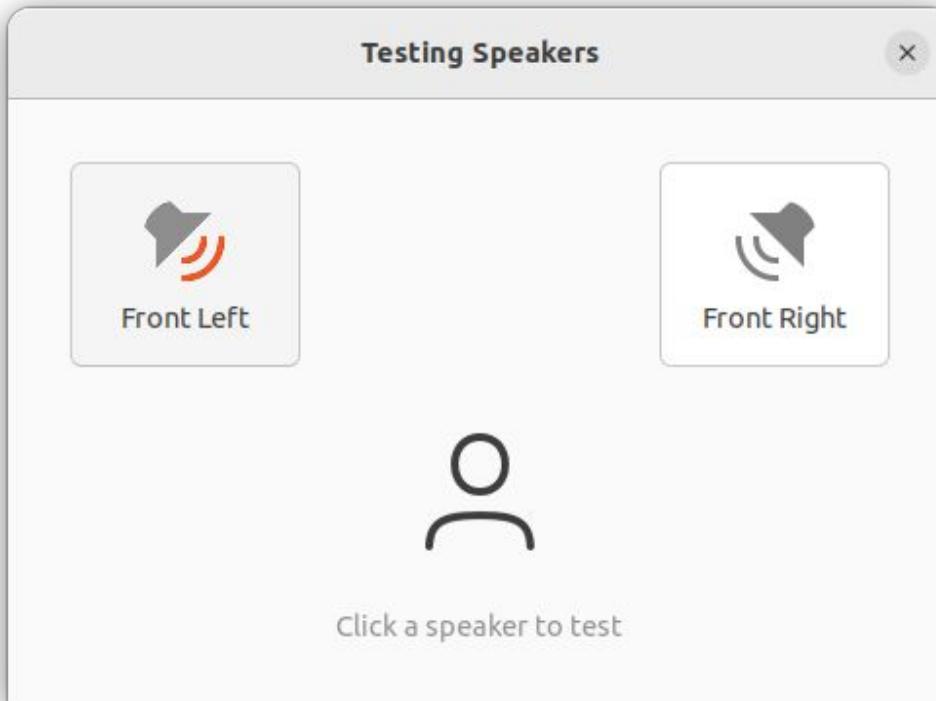
File Edit Selection Find View Goto Tools Project Preferences Help

```
1 public class Customer {  
2  
3     private int id;  
4     private String firstName;  
5     private String lastName;  
6  
7     public Customer(final int id, final S  
8         this.id = id;  
9         this.firstName = firstName;  
10        this.lastName = lastName;  
11    }  
12  
13    public int getId() {  
14        return id;  
15    }  
16  
17    public void setId(final int id) {  
18        this.id = id:  
19    }  
20}
```

Line 1, Column 1 master 208 Spaces: 2 Java

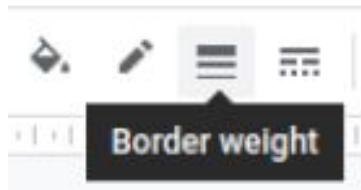
# Mô tả chức năng

- Sử dụng hình ảnh, biểu tượng, sơ đồ v.v.. giúp người dùng phán đoán chức năng và hành vi hệ thống
- Hiển thị và phản hồi trực quan thao tác người dùng
  - Cho biết đối tượng giống thứ gì và làm gì?



# Mô tả chức năng<sub>(2)</sub>

- Gợi ý chức năng
  - Cung cấp thêm thông tin về chức năng cho người dùng
  - Kích hoạt: Khi giữ chuột trên nút bấm, lần đầu mở cửa sổ mới, v.v...



# Tính thẩm mỹ

- Các giao diện phải hữu dụng, tạo động lực, đồng thời phải có tính thẩm mỹ
- Thiết kế thường càng đơn giản càng tốt
- Các khoảng trắng có vai trò quan trọng để phân tách các phần nội dung
- Mật độ thông tin phù hợp với kỹ năng người dùng
  - Mật độ thấp (< 50%) đối với người dùng mới.
  - Mật độ cao (> 50%) đối với người dùng chuyên nghiệp.
- Phong cách và khả năng đọc văn bản: Kích thước, kiểu chữ với nét hoa mĩ (serif) hoặc đơn giản (sans serif), cách sử dụng chữ hoa, v.v..
- Màu sắc và hoa văn.
- v.v..

# Độ phức tạp và hiệu quả sử dụng

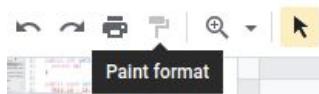
- Độ phức tạp:
  - Giao diện với ít thông tin thường đơn giản, dễ sử dụng đối với người dùng ít kinh nghiệm
    - Rất quan trọng đối với hệ thống có nhiều người dùng;
    - ... nhưng có thể không hiệu quả đối với người dùng chuyên nghiệp.
- Hiệu quả sử dụng:
  - Giao diện với nhiều thông tin có thể hiệu quả hơn đối với người dùng chuyên nghiệp (có thể yêu cầu đào tạo)
    - Rất quan trọng trong các hệ thống chuyên dụng;
    - ... nhưng có thể phức tạp đối với người dùng ít kinh nghiệm;
- Quan hệ bù trừ: Người thiết kế có thể phải lựa chọn ưu tiên người dùng mới hoặc người dùng chuyên nghiệp
  - ... hoặc hỗ trợ đồng thời các giao diện cơ bản & nâng cao.
  - Sử dụng những thiết kế tương tự làm giảm độ phức tạp.

# Tính nhất quán

- Phạm vi
  - Trong 1 chương trình
  - Trong 1 nhóm chương trình
  - Giữa các phiên bản của 1 chương trình theo thời gian
  - Giữa các phiên bản của 1 chương trình trên nhiều nền tảng
- Đặc biệt quan trọng để giảm thời gian tìm hiểu
  - Giúp người dùng phán đoán hành vi
  - Áp dụng kỹ năng học được ở 1 nơi cho những nơi khác
- Các yếu tố cơ bản
  - Những Lô-gic hoạt động chung
  - Điều khiển điều hướng
  - Thuật ngữ
  - Biểu tượng

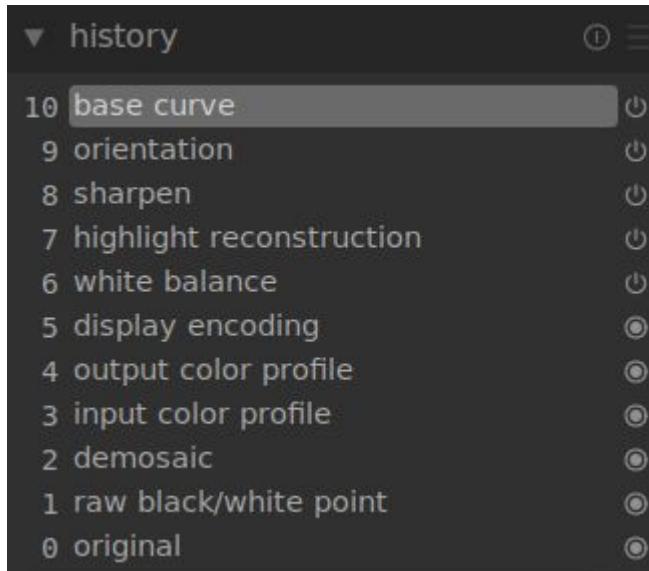
# Hạn chế lỗi thao tác

- Có thể vô hiệu các chức năng khi các điều kiện chưa được đáp ứng



*Cần đánh dấu trước khi có thể sao chép định dạng*

- Bảo vệ kết quả
  - Ghi nhớ các thay đổi
  - Hoàn tác / Khôi phục trạng thái trước thao tác lỗi



# Tối ưu thao tác người dùng

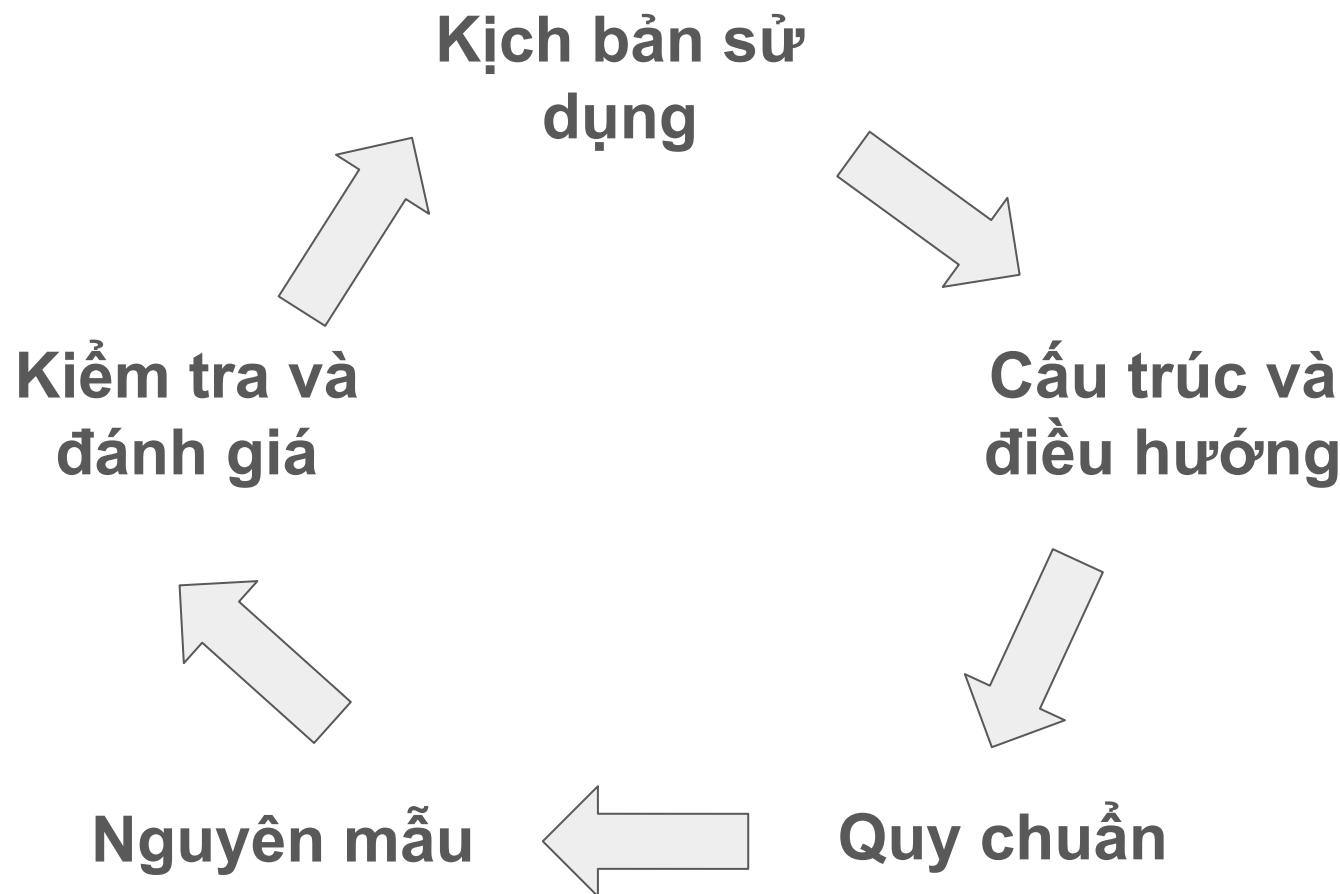
- Các giao diện cần được thiết kế để tối ưu thao tác của người dùng
  - Giúp người dùng hoàn thành công việc nhanh nhất, với số lượng thao tác tối thiểu.
  - Ở 1 góc độ nhất định có thể đánh giá độ phức tạp của thao tác người dùng thông qua số lần kích chuột
- Ví dụ quy tắc 3 lần kích chuột
  - Từ màn hình chính của hệ thống người dùng có thể tiếp cận bất kỳ mục nào với không quá 3 lần kích chuột



# Tiến trình phát triển giao diện người dùng

- Các nguyên tắc quan trọng: *Dẫn dắt bởi ca sử dụng, lặp và tăng dần, tập trung vào người dùng sớm và xuyên suốt dự án, kiểm thử thiết kế để đảm bảo nhu cầu sử dụng.*
- Cơ sở: Ca sử dụng, sơ đồ tuần tự và sơ đồ giao tiếp.
  - *Tập trung vào người dùng và công việc của họ, cái họ cần cho công việc.*
- Các bước thực hiện:
  - Biên soạn các kịch bản sử dụng với các bước tương tác chi tiết, có thể sử dụng các thành phần giao diện thực tế.
    - *(Có thể phát hiện thêm các yêu cầu mới).*
  - Thiết kế cấu trúc và điều hướng;
  - Biên soạn quy chuẩn;
  - Tạo nguyên mẫu.
  - Kiểm tra, đánh giá.

# Thiết kế GD theo nguyên tắc lặp và tăng dần



# Biên soạn kịch bản sử dụng

- Kịch bản sử dụng mô tả các bước được thực hiện bởi người dùng để hoàn thành 1 nhiệm vụ / 1 phần công việc của họ
- Mỗi kịch bản sử dụng tương ứng với 1 luồng sự kiện trong 1 ca sử dụng thiết yếu.
  - Tuy nhiên không phải là bản sao 1-1
  - Được mô tả với các chi tiết về các công cụ tác nghiệp.
- Cần lưu ý các kịch bản được sử dụng nhiều nhất / phổ biến nhất để ưu tiên tạo những giao diện phù hợp, dễ dùng cho những tình huống đó.

# Các ca sử dụng thực tế

- Phụ thuộc vào triển khai: Mô tả chi tiết cách sử dụng hệ thống sau khi được triển khai
- Các ca sử dụng thiết yếu được phát triển thành thực tế bằng cách mô tả chi tiết giao diện người dùng thực tế
- Đối với ứng dụng đa nền tảng, ví dụ, máy bàn, máy tính bảng, và điện thoại thông minh, ca sử dụng thực tế cần được phát triển cho từng nền tảng mà ca sử dụng được triển khai.

# Thiết kế cấu trúc và điều hướng

- Xác định các thành phần giao diện và lô-gic điều hướng
  - Các thành phần điều khiển điều hướng giúp người dùng di chuyển giữa các giao diện trong hệ thống
  - Các tương tác để đáp ứng nhu cầu sử dụng
- Đồng thời thông báo về kết quả thực hiện các hành động: Thành công, thất bại, đang được thực hiện, v.v..
- Có thể biểu diễn bằng IFML và các ca sử dụng thực tế.
  - IFML - Interaction Flow Modeling Language (Ngôn ngữ mô hình hóa luồng tương tác - thêm 1 quy chuẩn từ OMG).
- Nguyên lý cơ bản:
  - Sử dụng 1 trật tự ổn định cho các thành phần (ví dụ, Tệp > Tạo mới; Mở vs. Tệp > Mở; Tạo mới)

# Các hình thức điều khiển điều hướng

- Các nút bấm
- Các liên kết
- Danh mục / Menus
  - Người dùng được cung cấp 1 danh sách lựa chọn
  - Có nhiều hình thức (thanh menu, menu ngữ cảnh, v.v..)
- Vuốt và chạm (đối với màn hình cảm ứng)
- Kéo và thả
- Ngôn ngữ
  - Hình thức - Người dùng sử dụng hệ lệnh của hệ thống
  - Tự nhiên - Hệ thống trình diễn ngôn ngữ tự nhiên
- Điều khiển bằng giọng nói
- v.v..

# Các thông báo

- Thông báo lỗi - Cung cấp thông tin về lỗi phát sinh
  - Ví dụ, Có trường dữ liệu bắt buộc bị để trống
- Thông báo xác nhận - Hạn chế lỗi / Yêu cầu xác nhận đối với các thao tác quan trọng (và không thể hoàn tác)
  - Ví dụ, Bạn có chắc chắn muốn xóa hay không?
- Thông báo kết quả thành công
  - Ví dụ, đơn hàng đã được tạo
- Thông báo trạng thái
  - Ví dụ, đã hoàn thành 90% công việc
- Các thông điệp trợ giúp - Cung cấp thông tin bổ xung để hỗ trợ người dùng thực hiện công việc.
- v.v...

# Biên soạn quy chuẩn giao diện

- Quy chuẩn giao diện bao gồm các thành phần giao diện cơ bản được sử dụng thống nhất trong toàn hệ thống
- Các thành phần tiêu biểu thường được chuẩn hóa:
  - Khuôn mẫu giao diện: Bố cục tổng quan, Gam màu, lô-gic sắp xếp các cửa sổ, v.v..
  - Biểu diễn đối tượng:
    - Tên của các thứ trong giao diện
    - Thuật ngữ cho các hành động
    - Các biểu tượng giao diện, ví dụ biểu tượng trên các nút bấm, v.v.
- Các hình tượng tương tác có thể gợi ý hành vi
  - Cho biết đối tượng giống với thứ gì trong thế giới thực từ đó có thể phán đoán chức năng của đối tượng.
  - Ví dụ, giỏ hàng trong hệ thống bán hàng (giống giỏ hàng trong siêu thị) được sử dụng để lưu sản phẩm được chọn.

# Các hình tượng trong tương tác người-máy

- Hình tượng thao tác trực tiếp
  - Hình tượng trong đó các đối tượng được hiển thị (hình ảnh / biểu tượng) giống đối tượng vật lý.
  - Sử dụng giống như sử dụng các đối tượng vật lý.
- Hình tượng bàn làm việc
  - Màn hình được bố cục thành các vùng, với 1 không gian làm việc rộng ở chính giữa và tập hợp các công cụ được bố trí trên biên.
- Hình tượng tài liệu
  - Hình tượng trong đó dữ liệu được biểu diễn trực quan như các trang giấy hoặc biểu mẫu
- Hình tượng hội thoại
  - Hình tượng trong đó người dùng và máy tính tương tác giống như đang thảo luận thông qua các giao diện.

# Ví dụ 3. Hình tượng thao tác trực tiếp

- Giỏ hàng lưu các sản phẩm được chọn



- Kéo tệp vào (biểu tượng) thùng tái chế để xóa

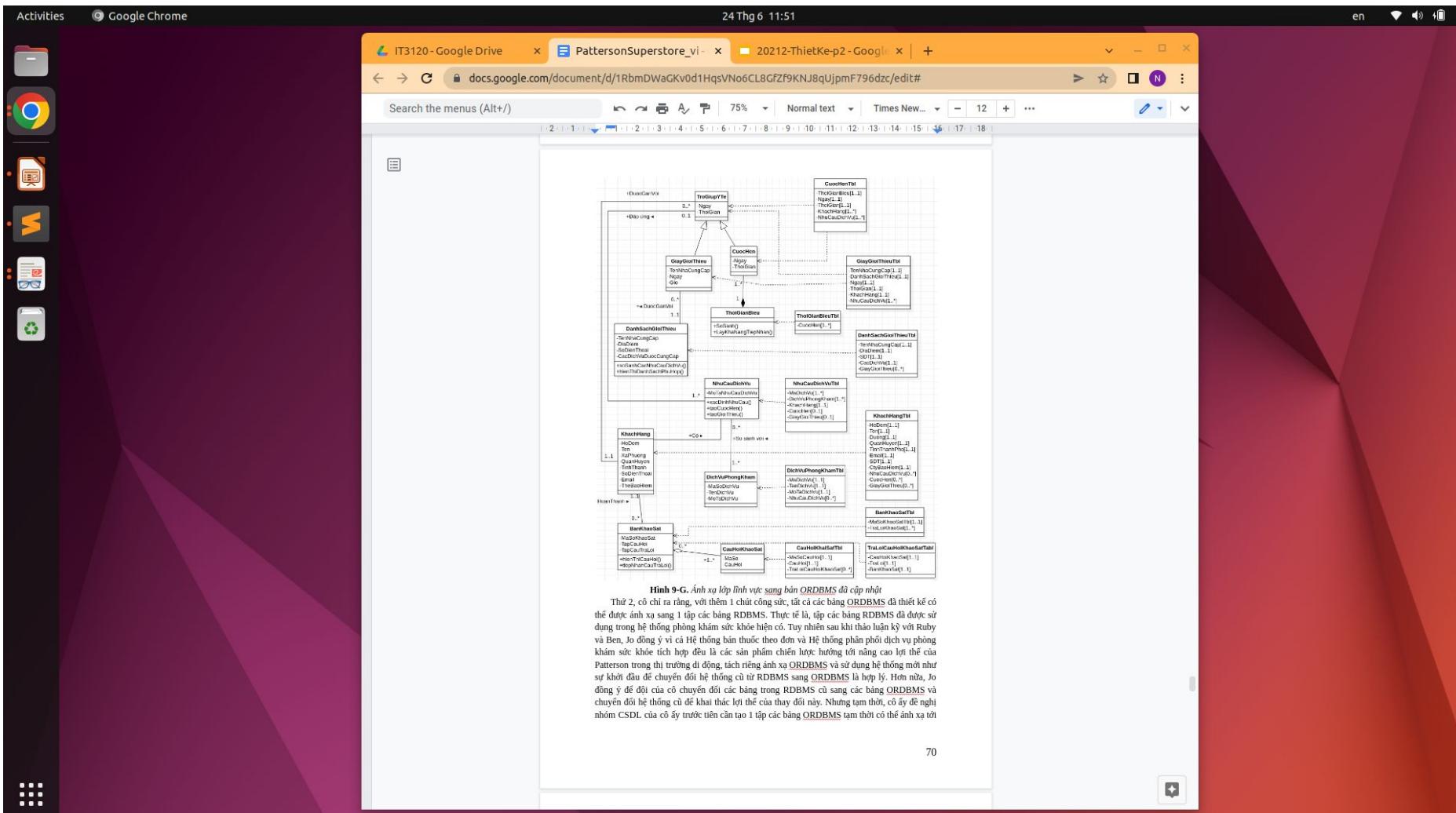


- Các công cụ vẽ



- v.v...

# Ví dụ 4. Hình tượng bàn làm việc & tài liệu

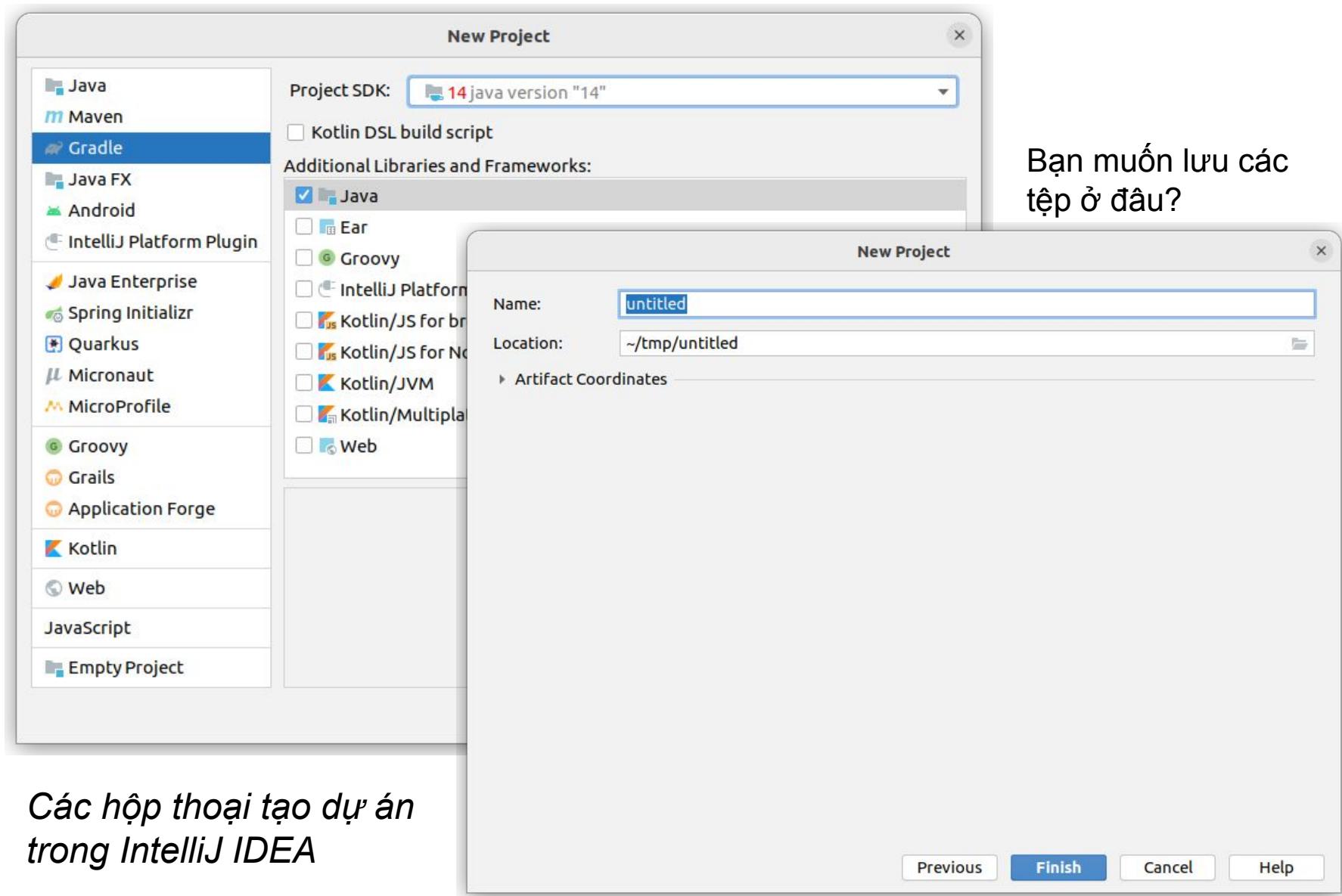


Hình 9-G. Ánh xạ lớp linh vực sang bản ORDBMS đã cập nhật

Thứ 2, cô chỉ ra rằng, với thêm 1 chút công sức, tất cả các bảng ORDBMS đã thiết kế có thể được ánh xạ sang 1 tập các bảng RDBMS. Thực tế là, tập các bảng RDBMS đã được sử dụng trong hệ thống phòng khám sức khỏe hiện có. Tuy nhiên sau khi thử luận kỹ với Ruby và Ben, Jo đồng ý vì cả Hệ thống bán thuốc theo đơn và Hệ thống phân phối dịch vụ phòng khám sức khỏe tích hợp đều là các sản phẩm chiến lược hướng tới nâng cao lợi thế của Patterson trong thị trường di động, tách riêng ánh xạ ORDBMS và sử dụng hệ thống mới như sự khởi đầu để chuyển đổi hệ thống cũ từ RDBMS sang ORDBMS là hợp lý. Hơn nữa, Jo đồng ý để đội của cô chuyển đổi các bảng trong RDBMS cũ sang các bảng ORDBMS và chuyển đổi hệ thống cũ để khai thác lợi thế của thay đổi này. Nhưng tạm thời, cô ấy để nghe nhóm CSDL của cô ấy trước tiên cần tạo 1 tập các bảng ORDBMS tạm thời có thể ánh xạ tới

# Ví dụ 5. Hội thoại

Bạn muốn tạo loại dự án nào?



Các hộp thoại tạo dự án  
trong IntelliJ IDEA

# Tạo nguyên mẫu giao diện

- Các giao diện có thể được:
  - Vẽ / phác thảo bằng các công cụ vẽ thông thường.
  - Tạo bằng các phần mềm chuyên dụng cho phép giả lập các thao tác chuyển giao diện.
  - Tạo bằng ngôn ngữ lập trình như trong pha thực thi:
    - Tuy nhiên chỉ giả lập các xử lý thực tế:
    - Ví dụ tạo nguyên mẫu trang Web với HTML, CSS, Javascript, và dữ liệu mô phỏng
    - => Gần nhất với giao diện khi triển khai thực tế
- Tương thích và nhất quán với các mô hình cấu trúc và điều hướng.
- Các giao diện có thể được tập hợp và bố trí theo cấu trúc điều hướng tạo thành các bảng phân cảnh / Storyboard
  - Hữu ích - Giúp người dùng hình dung về hoạt động của hệ thống thực tế (trong tương lai).

# Thiết kế nhập

- Các biểu mẫu được sử dụng để nhập dữ liệu
- Dữ liệu có thể:
  - Có cấu trúc: Ngày, sản phẩm, v.v...
  - Phi cấu trúc: Bình luận, mô tả.
- Các nguyên lý cơ bản
  - Nhận dữ liệu từ nguồn (ví dụ Mã vạch vs. RFID);
  - Xử lý trực tuyến vs. xử lý theo gói;
  - Tối ưu số thao tác nhập (sử dụng giá trị mặc định cho những giá trị được dùng thường xuyên).

# Các loại đầu vào

- Các hộp nhập dữ liệu:
  - Hộp nhập văn bản
  - Hộp nhập số, có thể hỗ trợ dạng tự động
    - Ví dụ: Nhập số điện thoại
  - Hộp mật khẩu ẩn các ký tự với các dấu \* và không cho cắt dán hoặc sao chép
  - v.v..
- Các hộp lựa chọn
  - Hộp đánh dấu / Check box, có thể lựa chọn nhiều mục
  - Hộp chọn loại trừ / Radio box - chọn 1 mục trong nhóm.
  - Hộp danh sách - Biểu diễn 1 danh sách lựa chọn
- Thanh kéo / Slider - Có thể di chuyển con trỏ trên 1 thang tương ứng với 1 khoảng giá trị.
- v.v..

# Kiểm tra đầu vào

- Dữ liệu cần được kiểm tra trước khi được đưa vào hệ thống để đảm bảo tính đúng đắn.
- Chỉ tiếp nhận dữ liệu hợp lệ.
- Các kiểm tra tiêu biểu:
  - Định dạng
  - Khoảng giá trị
  - Chữ số kiểm tra - Giúp giảm lỗi nhập số
    - Ví dụ chữ số cuối trong mã vạch
  - Tính nhất quán với các trường dữ liệu liên quan
  - Các ràng buộc trong CSDL

# Thiết kế xuất

- Các báo cáo được tạo từ dữ liệu được xuất bởi hệ thống
- Các nguyên lý cơ bản:
  - Mục đích và tần suất sử dụng ảnh hưởng đến bố cục của báo cáo
  - Quản lý lượng thông tin trong 1 báo cáo - Chỉ cung cấp những gì cần thiết và đặt những thông tin quan trọng nhất ở vị trí đầu tiên.
  - Giảm thiểu độ lệch, đặc biệt trong thiết kế đồ họa (biểu đồ).

# Các loại đầu ra

- Báo cáo chi tiết - Người dùng cần thông tin đầy đủ
- Báo cáo ngắn gọn - Các chi tiết được tổng hợp (ví dụ, tổng, trung bình)
- Báo cáo ngoại lệ
- Tài liệu xoay vòng - Đầu ra quay lại thành đầu vào
- Biểu đồ - Đề thuận tiện so sánh trực quan
- Có thể xuất báo cáo trên thiết bị điện tử (xem trên màn hình) hoặc bản cứng (in trên giấy)

# Kiểm tra giao diện

- Lý tưởng, đánh giá các giao diện khi hệ thống đang được thiết kế - Trước khi bắt đầu xây dựng nó
  - Giúp phát hiện và giải quyết sớm các vấn đề
  - Người dùng có thể muốn thay đổi giao diện sau khi nhìn thấy nó.
- Huy động tối đa các thành viên tham gia

# Các phương pháp đánh giá giao diện

- Theo kinh nghiệm - Đổi chiều thiết kế với các nguyên lý và quy tắc đã biết
- Thuyết trình & đánh giá - Nhóm thiết kế trình diễn nguyên mẫu với người dùng và giải thích lô-gic hoạt động
- Tương tác - Người dùng thử sử dụng nguyên mẫu và thảo luận trực tiếp với các thành viên đội dự án.
- Kiểm thử kín - Người dùng thử sử dụng nguyên mẫu ngôn ngữ trong phạm vi phòng thí nghiệm.

# Các yêu cầu phi chức năng

*Thiết kế giao diện có thể bị ảnh hưởng bởi các yêu cầu phi chức năng (được xác định từ giai đoạn đầu tiên), ví dụ:*

- Các yêu cầu vận hành
  - Triển khai hệ thống theo hình thức ứng dụng Web
- Độ tin cậy
  - Có thể khôi phục hoạt động của hệ thống trong tình huống sự cố trong giới hạn 5 p
- Hiệu năng
  - Cập nhật trạng thái theo thời gian thực với độ trễ không quá 1s
- Các yêu cầu bảo mật
  - Sử dụng kết nối HTTPS

# Nội dung

1. Tổng quan về thiết kế giao diện
2. Ngôn ngữ mô hình hóa luồng tương tác
3. Ví dụ tổng hợp



# Các thành phần mô hình

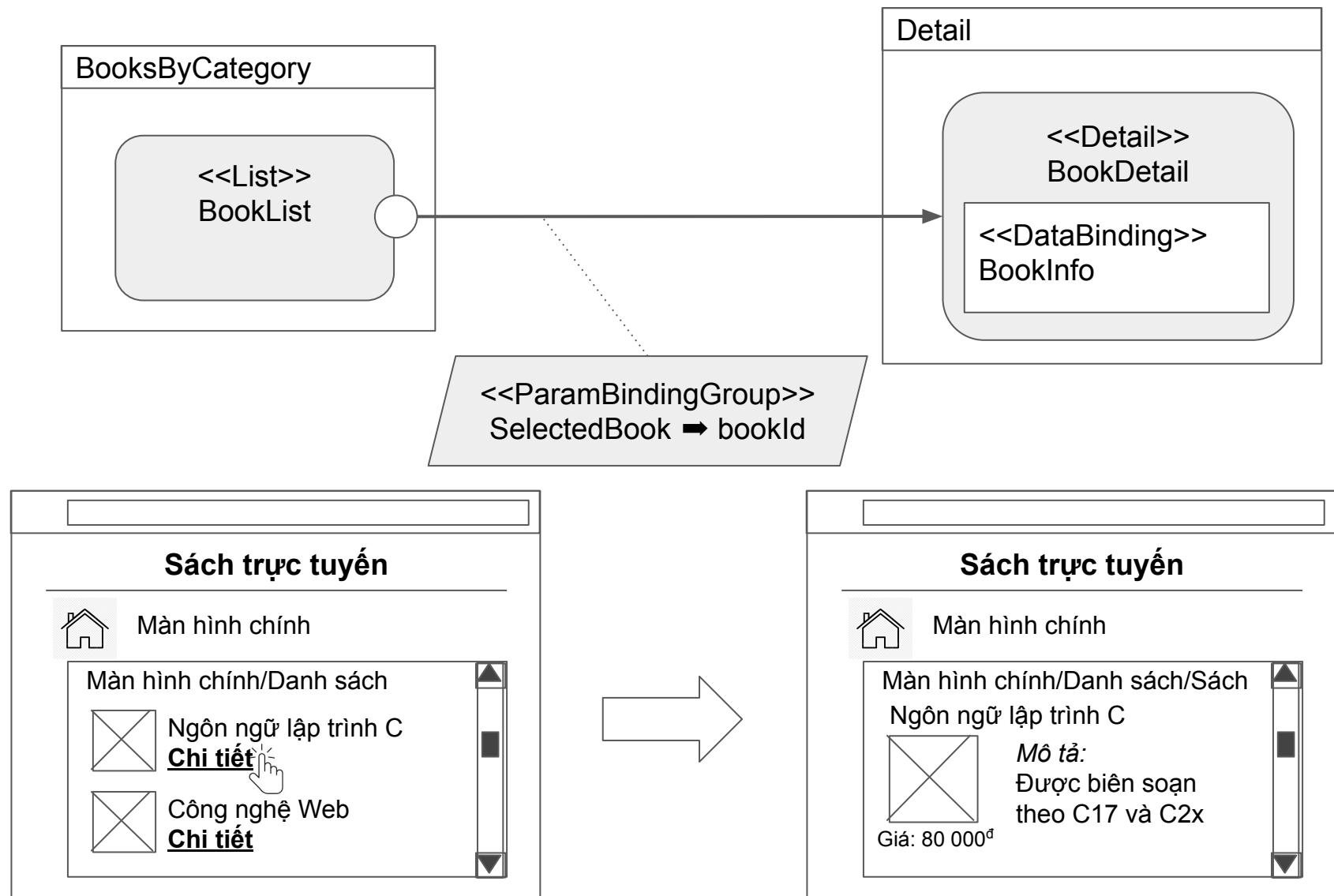
*Sơ đồ luồng tương tác (trong Ngôn ngữ mô hình hóa luồng tương tác - **I**nteraction **F**low **M**odeling **L**anguage - IFML) biểu diễn các khía cạnh cơ bản của giao diện người dùng*

- **Bố cục:** Chia màn hình giao diện thành các phân vùng, các phân vùng có thể lồng nhau, có thể được thiết lập trạng thái hiển thị, trạng thái khả dụng.
  - Trong IFML phân vùng được biểu diễn bằng thành phần khung/chứa đựng - ViewContainer.
- **Nội dung:** Các thành phần nội dung trong phân vùng
  - Thành phần nội dung được biểu diễn bằng thành phần hiển thị - ViewComponent.

## Các thành phần mô hình<sub>(2)</sub>

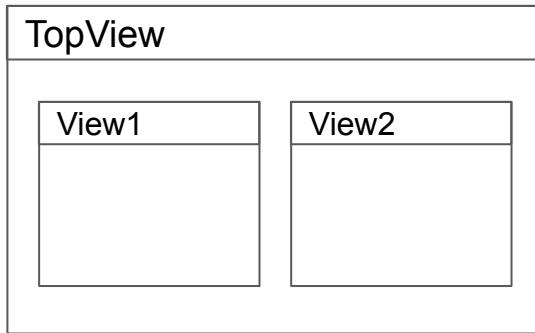
- **Sự kiện:** Các diễn biến ảnh hưởng tới trạng thái giao diện người dùng - Events - Có thể được kích hoạt bởi tương tác người dùng, tự động theo trạng thái hệ thống hoặc bởi hệ thống ngoại.
- **Điều hướng:** Biểu diễn hệ quả của 1 sự kiện trên giao diện người dùng như cập nhật và hiển thị nội dung, kích hoạt hành động, hoặc sự kết hợp của nhiều hiệu ứng.
- **Truyền tham số:** Xác định mối quan hệ đầu ra-đầu vào giữa các thành phần giao diện, phân vùng giao diện, và các hành động.

# Ví dụ 6. Sơ đồ luồng tương tác

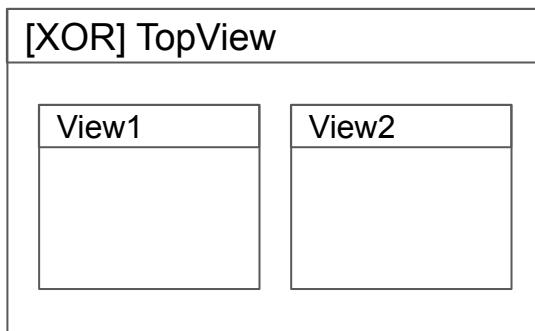
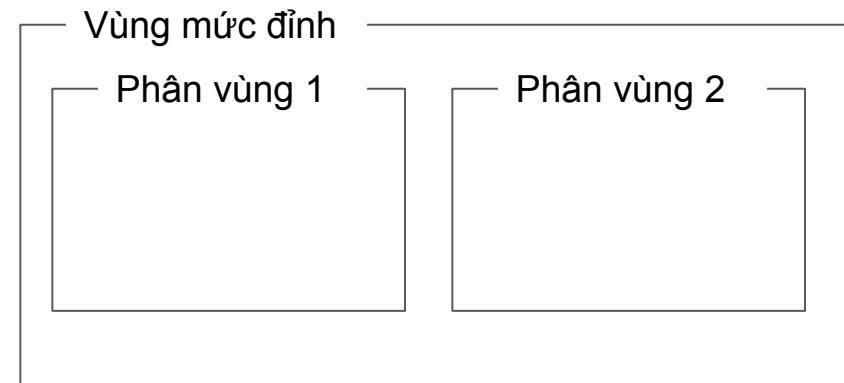


# Các phân vùng lồng nhau

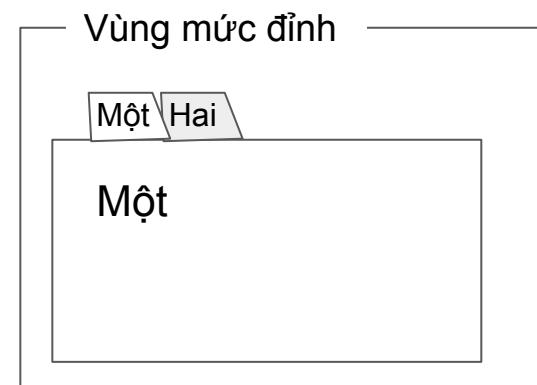
Các phân vùng con cùng nằm trong 1 phân vùng có thể được hiển thị đồng thời hoặc loại trừ lẫn nhau.



Đồng thời

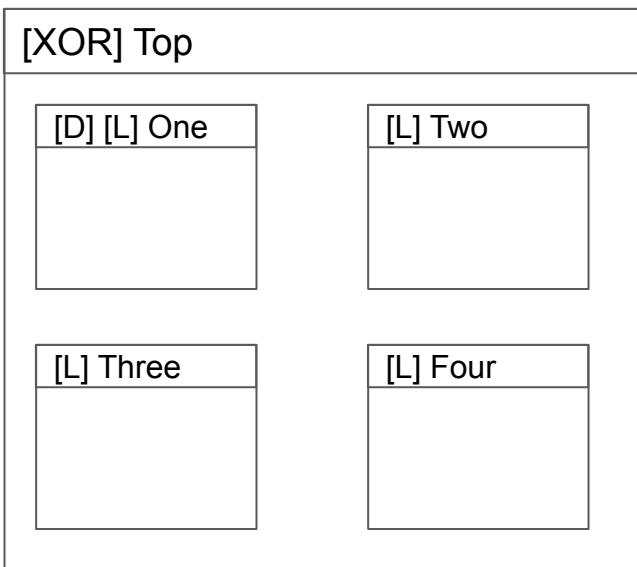


Loại trừ

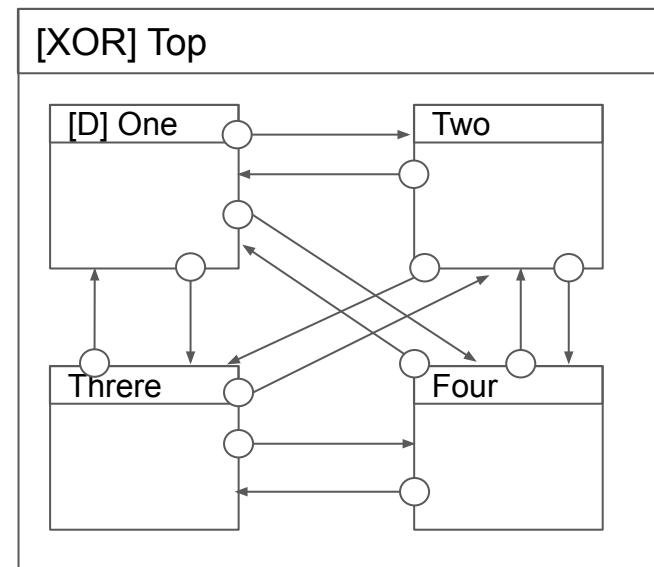


# Các thuộc tính của phân vùng

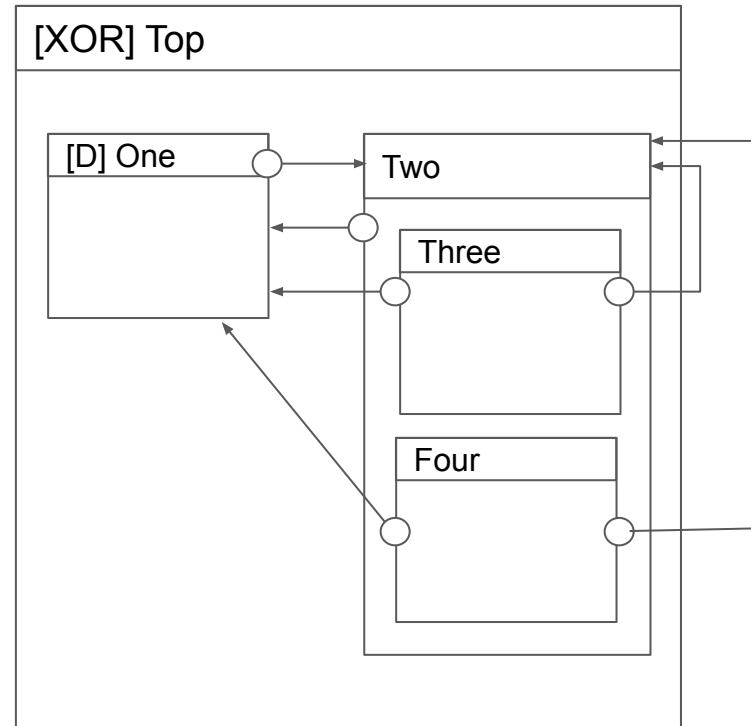
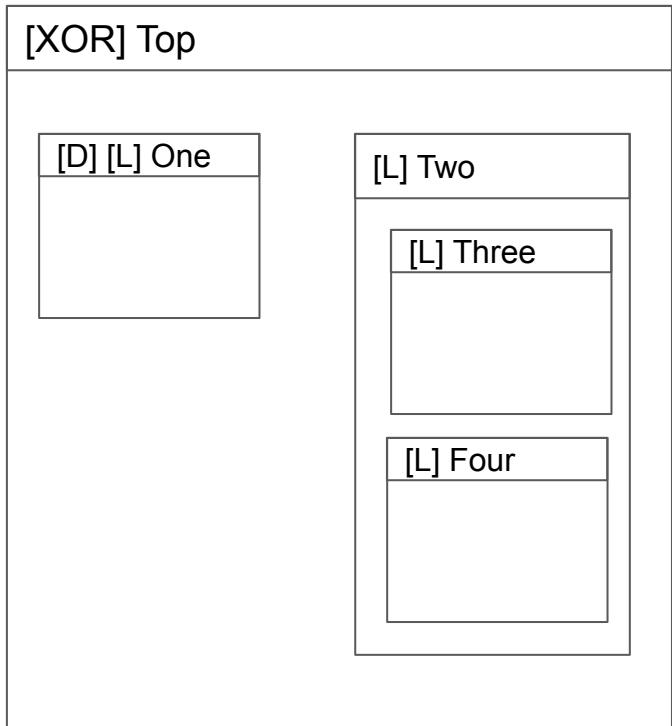
- Phân vùng mặc định - Được chọn để hiển thị nếu không có quy ước cụ thể khác khi phân vùng chưa nó được kích hoạt. Được ký hiệu bằng ký tự D.
- Phân vùng đánh dấu - Có thể hướng tới từ bất kỳ phân vùng nào nằm trong phân vùng chứa nó. Được ký hiệu bằng ký tự L.



< tương đương >



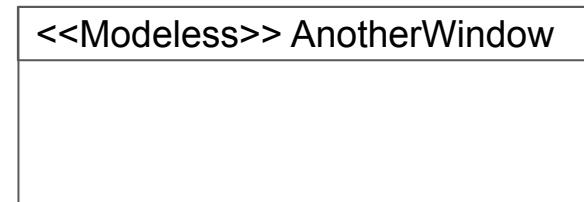
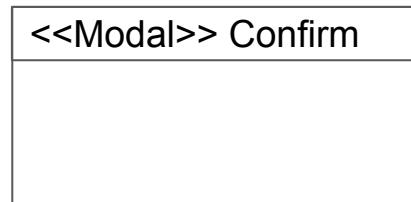
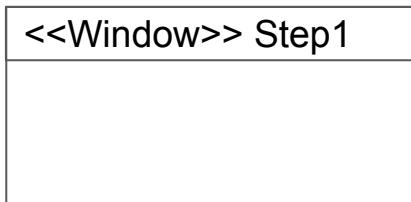
# Các thuộc tính của phân vùng<sub>(2)</sub>



# Cửa sổ

Cửa sổ là 1 loại thành phần khung đặc biệt, có thể được thiết lập chế độ khóa (Modal) hoặc nới lỏng (Modeless).

- Một cửa sổ khóa được mở sẽ vô hiệu khả năng tương tác với các cửa sổ ở chế độ nền cho tới khi được đóng lại.
- Một cửa sổ nới lỏng được mở vẫn cho phép tương tác với các thành phần giao diện khác.



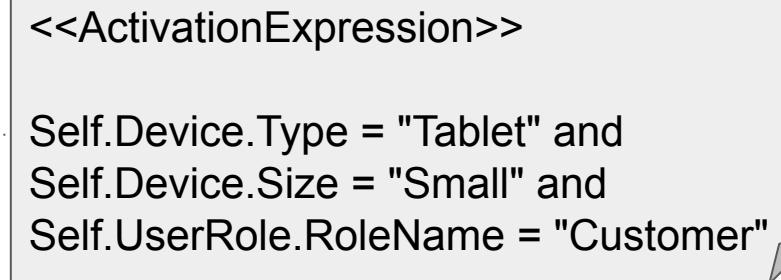
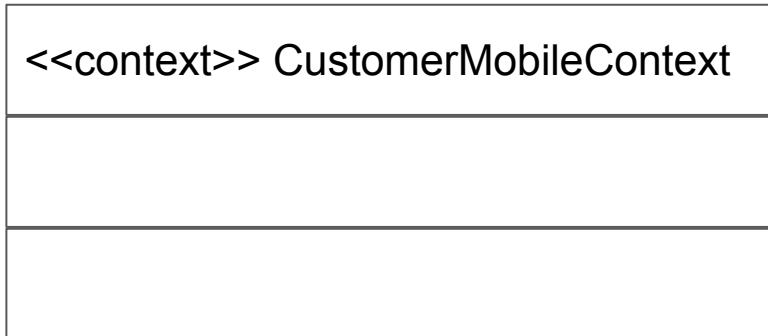
# Ngữ cảnh và chiều ngữ cảnh

- Ngữ cảnh (Context) mô tả các đặc điểm trạng thái hệ thống ở thời gian thực thi và xác định cách giao diện người dùng thích nghi với các đặc điểm đó. Chiều ngữ cảnh (ContextDimension) là 1 thành phần của ngữ cảnh.
- Các mở rộng của chiều ngữ cảnh trong IFML
  - Vai trò người dùng (UserRole) - Bao gồm các thuộc tính mà hồ sơ người dùng phải đáp ứng để kích hoạt ngữ cảnh.
  - Thiết bị (Device) - Biểu diễn các đặc điểm của thiết bị.
  - Vị trí (Position) - Biểu diễn thông tin vị trí và định hướng của thiết bị được sử dụng bởi chương trình ứng dụng.

# Biểu thức kích hoạt

Biểu thức kích hoạt (ActivationExpression) xác định điều kiện Boolean để kích hoạt ngữ cảnh (hoặc thành phần IFML khác)

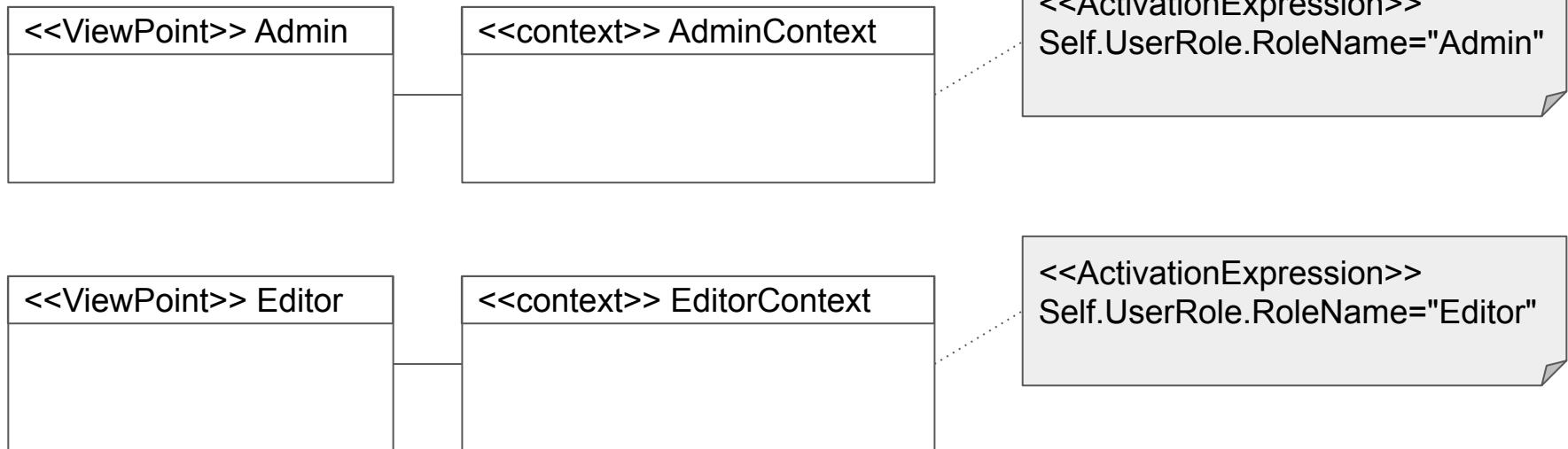
- Kích hoạt nếu đúng và vô hiệu nếu sai.



*Biểu thức kích hoạt xác định các điều kiện để kích hoạt ngữ cảnh Người dùng thiết bị di động*

# Điểm nhìn

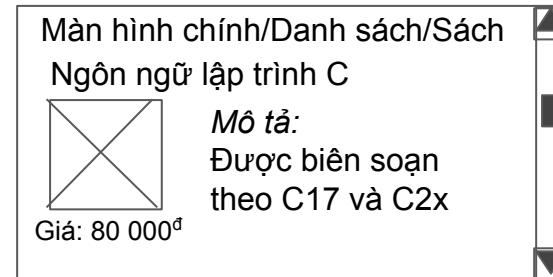
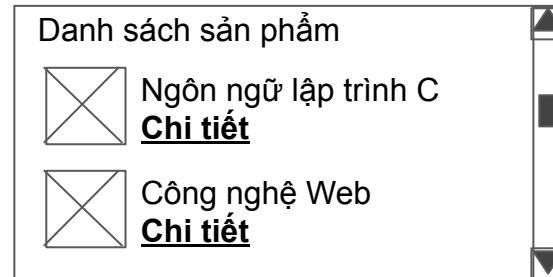
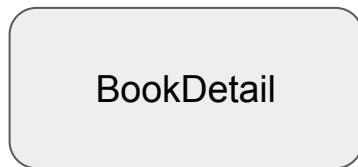
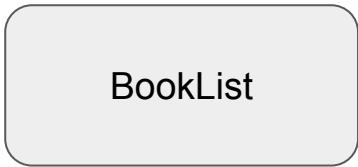
Mỗi điểm nhìn (ViewPoint) là 1 đặc tả toàn bộ mô hình giao diện người dùng có hiệu lực khi 1 ngũ cảng cụ thể được bật.



# Thành phần hiển thị

- Thành phần hiển thị (ViewComponent) - Thành phần có thể được nhìn thấy trong giao diện người dùng được sử dụng để xuất dữ liệu hoặc nhận dữ liệu từ người dùng.
- Mỗi thành phần hiển thị có thể được hợp thành từ nhiều bộ phận được gọi là phần tử hiển thị ViewComponentPart
  - Phần tử hiển thị (ViewComponentPart) có thể chỉ tồn tại trong thành phần hiển thị/không tồn tại bên ngoài phạm vi thành phần hiển thị.
- Sự kiện và điều hướng: Các thành phần hiển thị có thể tiếp nhận tương tác bằng cách gắn kết với các sự kiện.

# Ví dụ 7. Các thành phần hiển thị



# Nạp nội dung và dữ liệu

- Thành phần hiển thị xuất nội dung được gắn với nó lên màn hình giao diện.
- Nạp nội dung/ContentBinding: Xác định nguồn nội dung của thành phần hiển thị bằng URI.
- Nạp dữ liệu/DataBinding: Dữ liệu bên trong thành phần hiển thị được xác định bởi kiểu dữ liệu, điều kiện lọc và các thuộc tính được sử dụng để hiển thị.
- Hành vi động/DynamicBehavior: Nhận dữ liệu từ cổng dịch vụ

WebViewer

```
<<ContentBinding>>  
http://mysite.com/index.html
```

MessageList

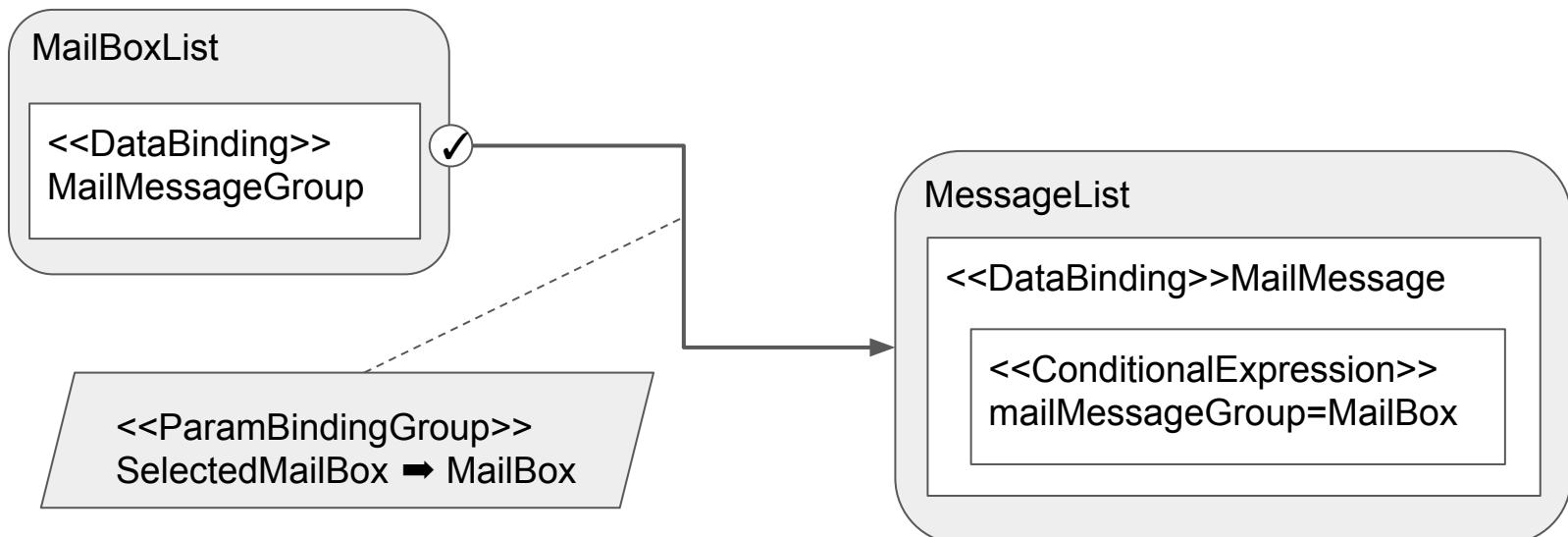
```
<<DataBinding>>  
MailMessage
```

TweetList

```
<<DynamicBehavior>>  
TwitterAPI.Search(query, resNum)
```

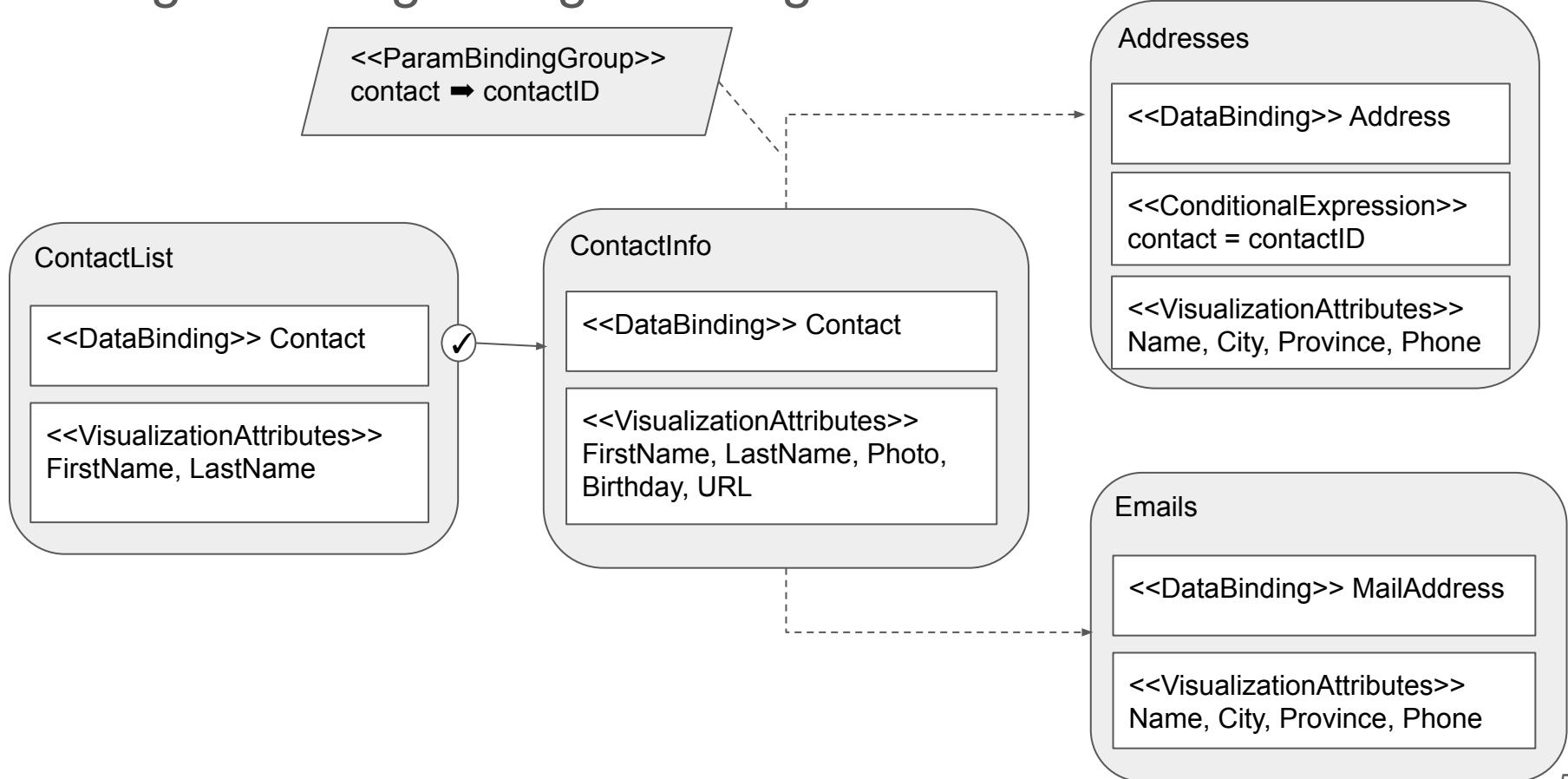
# Truyền tham số

- Kết nối tham số đầu ra của 1 thành phần hiển thị với tham số đầu vào của 1 thành phần hiển thị khác.
- Trong trường hợp truyền đồng thời nhiều tham số các tham số được gom thành nhóm - Truyền nhóm tham số.



# Luồng dữ liệu

Luồng dữ liệu là luồng tương tác cho biết các tham số được truyền từ 1 thành phần nguồn tới 1 thành phần đích mà không có tương tác người dùng.



# Mở rộng IFLM

- Các thành phần mở rộng được ký hiệu bởi nhãn loài và có thể chứa các phần tử bỗ xung.
- IFLM cho phép mở rộng thành phần hiển thị cơ bản (ViewComponent) với các đặc tả của người dùng.
- Ví dụ các mở rộng của thành phần hiển thị:

<<List>> ListName

<<Details>> DetailsName

<<Form>> FormName

<<SimpleField>> Field1: type1

<<SelectionField>> Sel1

# Các thành phần mở rộng để xuất dữ liệu

- Danh sách/List: Mở rộng của ViewComponent để hiển thị 1 danh sách các đối tượng. Nếu danh sách được gắn với 1 sự kiện thì bất kỳ đối tượng nào trong nó cũng có thể được sử dụng để kích hoạt sự kiện và truyền như đối số.
- Danh sách đa lựa chọn/MultiChoiceList: Hỗ trợ nhiều sự kiện (checking, unchecking, submit) và cho phép chọn nhiều phần tử.
- Chi tiết/Detail: Được sử dụng để hiển thị các thuộc tính của đối tượng. Đối tượng trong nó có thể được sử dụng để kích hoạt sự kiện và truyền đi như đối số.
- Sự kiện chọn>SelectEvent: Hỗ trợ lựa chọn 1 hoặc nhiều phần tử từ 1 tập hợp. Khi được kích hoạt nó khiến các giá trị được lựa chọn được truyền đi như đối số.

# Biểu mẫu

- Biểu mẫu/Form là 1 thành phần hiển thị có thể bao gồm nhiều trường dữ liệu.
- Trường/Field là bộ phận của biểu mẫu được xác định bởi kiểu dữ liệu của giá trị mà nó tiếp nhận hoặc hiển thị cho người dùng.
- Trường đơn giản/SimpleField Là 1 loại trường tiếp nhận giá trị có định kiểu. Giá trị của nó là tham số đầu ra và có thể được truyền tới thành phần khác hoặc hành động.
- Trường lựa chọn/SelectionField là 1 loại trường cho phép chọn 1 hoặc nhiều giá trị từ 1 tập đã được định nghĩa.
- Sự kiện gửi/SubmitEvent là 1 loại sự kiện biểu diễn việc gửi 1 hoặc nhiều giá trị. Nó kích hoạt truyền tham số từ thành phần chứa sự kiện tới đích của luồng điều hướng.

# Ví dụ 8. Biểu mẫu

<<Form>>  
BookEntry

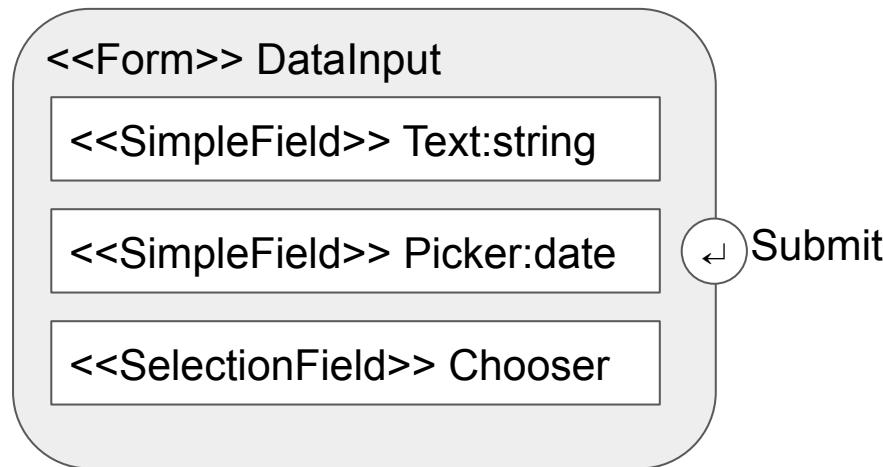
Thông tin về sách

Tên:

Giá:  Đơn vị:  ▾

Mô tả:

## Ví dụ 8. Biểu mẫu<sub>(2)</sub>



Nhập dữ liệu

Văn bản:

Chọn ngày:

Lựa chọn:

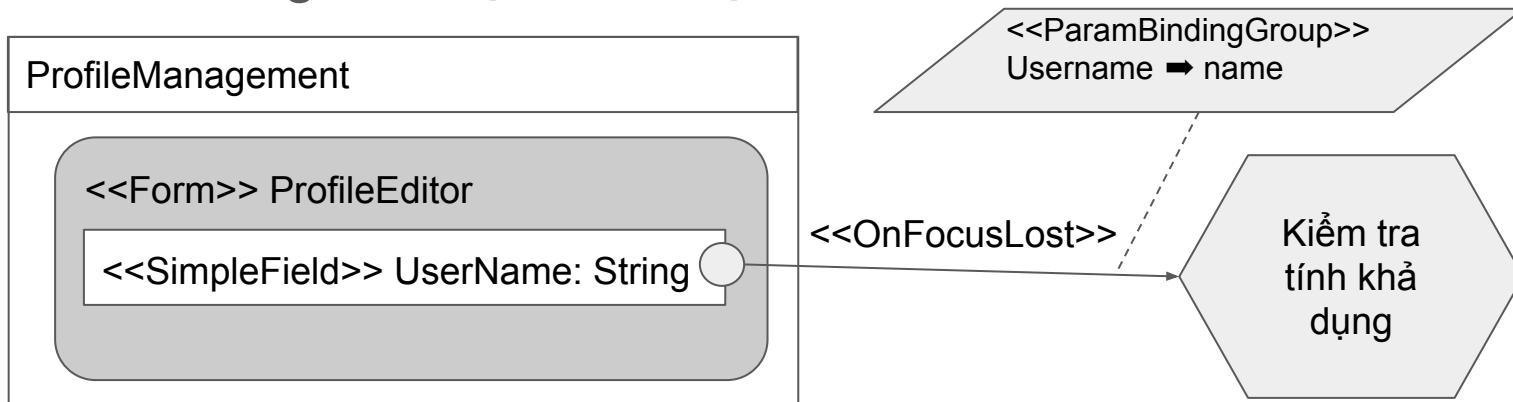
Gửi

The screenshot shows a user interface for data entry. It features a title 'Nhập dữ liệu' and three input fields. The first field is labeled 'Văn bản:' and contains an empty text input. The second field is labeled 'Chọn ngày:' and includes a date input showing '28/04/2023' and a button with three dots. The third field is labeled 'Lựa chọn:' and has a dropdown menu open, displaying 'Mục 1'. A large 'Gửi' button is located at the bottom right.

# Các mở rộng cho giao diện cửa sổ

- Mở rộng sự kiện:

- Khi mất tiêu điểm/OnFocusLost: Sự kiện được kích hoạt khi người dùng rời trường dữ liệu. Có thể được gắn với trường đơn giản hoặc toàn bộ biểu mẫu.

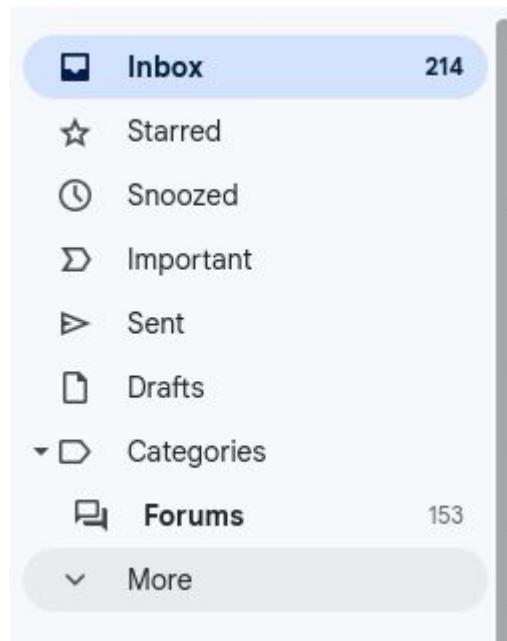


- Bắt đầu kéo/OnDragStart - Bắt đầu kéo. Sự kiện đích được xác định bởi OnDropEvent.
- Thả/OnDrop - Được kích hoạt khi kết thúc tương tác thả.

# Các mở rộng cho giao diện cửa sổ<sub>(2)</sub>

- Các thành phần hiển thị:

- Cây/Tree: Được sử dụng để hiển thị dữ liệu phân cấp, ví dụ cây danh mục email.
- Bảng/Table: Hiển thị dữ liệu theo dạng bảng và hỗ trợ người dùng soạn thảo.



A screenshot of a product catalog table titled 'Danh mục sản phẩm'. The table has a header row with columns for Mã (Code), Tên (Name), Đơn giá (Unit Price), and Đơn vị (Unit). There are five data rows below the header. Each row contains a code, name, price, unit, and a blue 'Xóa' (Delete) link. A dashed line points from the 'Table' bullet point in the list above to the table in the screenshot.

Mã	Tên	Đơn giá	Đơn vị	
T110	Tua vít	10000	cái	<a href="#">Xóa</a>
K221	Kìm mỏ nhọn	15000	cái	<a href="#">Xóa</a>
D335	Dây thép	30000	kg	<a href="#">Xóa</a>
O668	Óc vít	50000	kg	<a href="#">Xóa</a>

# Các mở rộng cho giao diện Web

- Phân vùng: Các mở rộng của ViewContainer
  - Trang - page - 1 mở rộng của thành phần khung biểu diễn 1 trang Web có địa chỉ URL riêng.
  - Khu vực - Area - 1 mở rộng của thành phần khung loại trừ biểu diễn 1 nhóm trang hoặc các khu vực khác, được gom nhóm theo mục đích ứng dụng cụ thể.
  - Miền - SiteView - 1 mở rộng của thành phần khung loại trừ biểu diễn các khu vực của ứng dụng Web và các trang được gom lại theo mục đích ứng dụng cụ thể, thường để đáp ứng nhu cầu của 1 tác nhân.

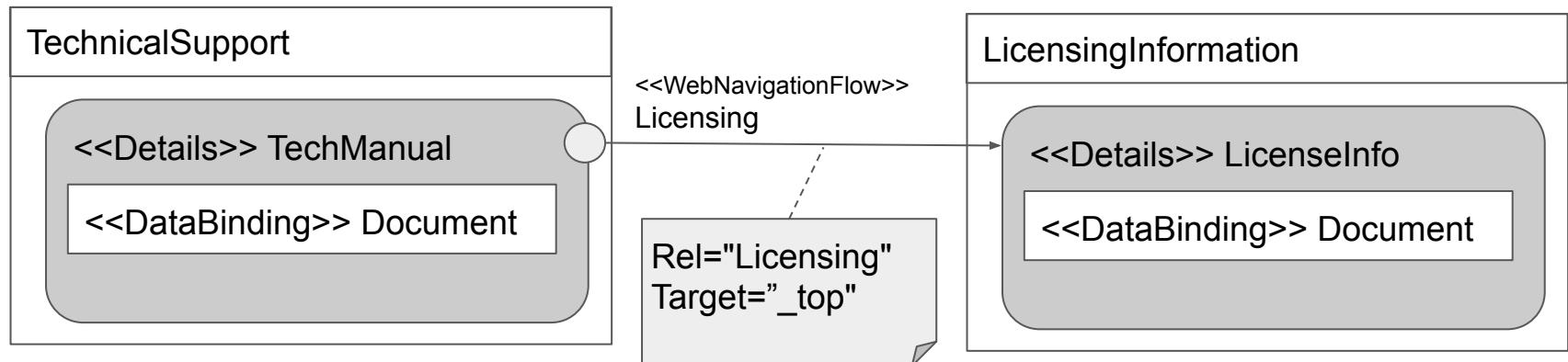
# Các mở rộng cho giao diện Web<sub>(2)</sub>

Miền/Khu vực/Trang được bổ xung thêm các thuộc tính mới cần thiết cho môi trường Web:

- Nhãn URL/URL label: Chuỗi cho biết địa chỉ cố định của miền/trang/khu vực. Đối với các trang động, URL thường bao gồm các tham số để tính nội dung của nó.
- Bảo mật/Security: Nếu thuộc tính có giá trị là "được bảo mật" thì tất cả các trang của khu vực được truy cập bằng giao thức HTTPS.
- Bảo vệ/Protection: Nếu thuộc tính có giá trị là "được bảo vệ" thì tất cả trang trong khu vực hoặc miền, hoặc 1 trang riêng lẻ được giới hạn truy cập.

# Các mở rộng cho giao diện Web<sub>(3)</sub>

- Liên kết/Link: Luồng điều hướng Web (WebNavigationFlow) là 1 mở rộng của luồng điều hướng (NavigationFlow) với các thuộc tính bổ xung:
  - Quan hệ/Rel: Xác định mối quan hệ giữa tài liệu hiện thời và tài liệu được liên kết; giá trị của nó được mã hóa bằng HTML.
  - Đích/Target: Xác định đích mở tài liệu, thường là 1 cửa sổ trình duyệt; có thể là cùng cửa sổ trình duyệt với tài liệu hiện tại hoặc 1 cửa sổ mới.



# Các mở rộng cho giao diện Web<sub>(4)</sub>

- Thành phần hiển thị:
  - Danh sách sắp xếp động/DynamicSortedList: Là mở rộng của thành phần danh sách (List) cho phép người dùng sắp xếp dữ liệu bằng các thuộc tính trực quan.
  - Danh sách cuộn/ScrollableList: Là mở rộng của danh sách cho phép gom các phần tử thành khối (phân trang) và cho phép di chuyển đến các khối.
  - Danh sách lồng nhau/NestedList: Là mở rộng của danh sách với thành phần có thể là danh sách.

# Mở rộng cho giao diện di động

- Ngữ cảnh:
  - Thiết bị/Device có các chiều sau:
    - Kích thước đường chéo/DiagonalSize: Kích thước vật lý theo đường chéo của màn hình.
    - Phân khúc kích thước/SizeCategory: Các kích thước có thể được gom thành các lớp như (Nhỏ, Bình thường, Lớn, rất lớn).
    - Phân khúc mật độ/DensityCategory: Các giá trị mật độ được gom thành các lớp (Thấp, Trung bình, Cao, Rất cao).
    - Kích thước theo điểm ảnh/PixelSize: Kích thước thực tế theo chiều ngang và chiều dọc của màn hình được đo bằng điểm.
    - Mật độ: Số lượng điểm ảnh trên đơn vị diện tích được đo bằng dpi (dots per inch).
    - Kết nối mạng/Network connectivity: Có thể được sử dụng để tùy chỉnh chất lượng của nội dung trên giao diện dựa trên tốc độ kết nối mạng.
    - V.v..

# Mở rộng cho giao diện di động<sub>(2)</sub>

- Phân vùng:
  - Nhãn dòng <<system>>: Thành phần khung được gán nhãn <<system>> biểu diễn 1 vùng cố định của giao diện, được quản lý bởi hệ điều hành hoặc nền tảng giao diện khác.
- Thành phần hiển thị và sự kiện/Component and event:
  - Nhãn dòng <<system>>: Thành phần hiển thị được gán nhãn <<system>> nhấn mạnh giao diện sử dụng các thành phần của hệ thống.
  - <<nhấn>> và <<chạm>>: <<press>> and <<touch>>.

# Nội dung

1. Tổng quan về thiết kế giao diện
2. Ngôn ngữ mô hình hóa luồng tương tác
3. Ví dụ tổng hợp

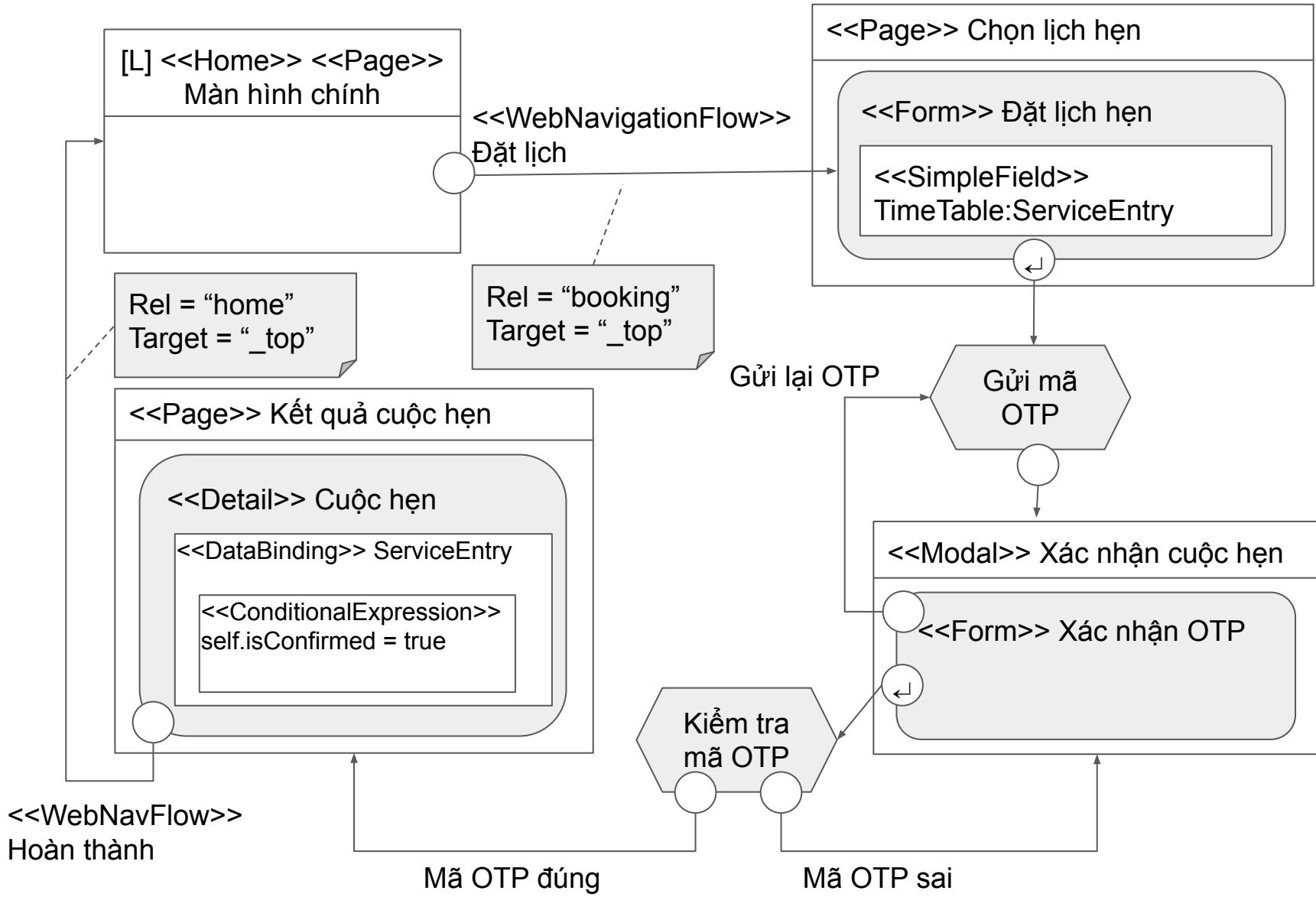
Ví dụ thiết kế giao diện Web  
cho kịch bản Đặt lịch hẹn dịch vụ Spa

# Kịch bản sử dụng

## **Kịch bản Khách hàng đặt lịch hẹn dịch vụ Spa**

1. Khách hàng yêu cầu Chọn lịch hẹn
2. Hệ thống hiển thị lịch làm việc
3. Khách hàng lựa chọn thời gian phù hợp
4. Hệ thống gửi mã xác nhận tới khách hàng
5. Khách hàng nhập mã xác nhận
6. Hoàn thành đặt lịch nếu mã xác nhận hợp lệ.
7. Nếu mã xác nhận không hợp lệ
  - Khách hàng có thể nhập lại
  - hoặc yêu cầu gửi lại mã xác nhận  
*(rồi quay lại bước 5).*
  - hoặc kết thúc không đặt lịch hẹn.

# Mô hình hóa luồng tương tác



# Quy chuẩn giao diện

- Các nút bấm
  - Gửi yêu cầu - Đặt lịch hẹn theo thông tin đã chọn
  - Xác nhận - Chứng minh người dùng thực đang gửi yêu cầu
  - Hoàn thành - Đóng cửa sổ đặt lịch hẹn và quay về màn hình chính
- Các danh mục mức đǐnh
  - Đặt lịch hẹn - Bắt đầu thực hiện kịch bản Đặt Lịch hẹn
- Các liên kết
  - Gửi lại mã - Gửi lại mã OTP để định danh người dùng
- Hộp nhập mã 

123567

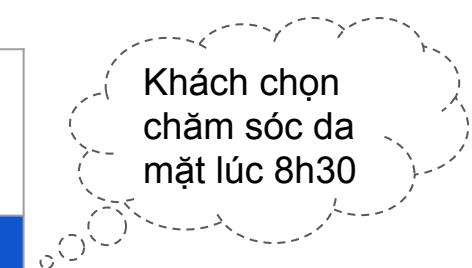
  - 1 dòng, độ dài tối đa 6 ký tự
  - Hiện ký tự
  - Cỡ chữ 18pt

# Quy chuẩn giao diện<sub>(2)</sub>

- Thời gian biểu:
  - Hiển thị theo ngày, khoảng cách 30 phút
  - Dòng là thời gian, cột là dịch vụ
  - Để trống ô có thể chọn, ô đã có lịch hẹn được tô màu đỏ, ô được chọn được tô màu xanh.

Ngày 10/04/2023

	Gội đầu		Chăm sóc da mặt
	Ghế 1	Ghế 2	
8h30			
9h00			
11h00			
11h30			
13h30			
14h00			



# Quy chuẩn giao diện<sub>(3)</sub>

- Chi tiết cuộc hẹn:
  - Hiển thị thông tin cuộc hẹn đã được đăng ký thành công  
**Đặt lịch hẹn thành công!**

**Tên dịch vụ:** Chăm sóc da mặt

**Thời gian:** 9h00

**Ngày:** 10/04/2023

# Nguyên mẫu giao diện: Trang chính

The wireframe illustrates a main page interface. At the top, there is a header section with a large rectangular input field. Below the header, on the right side, is a circular icon labeled "Người dùng". On the left side, there is a button labeled "Đặt lịch hẹn". The central area features a large diamond-shaped placeholder with the text "<>". At the bottom, there is another diamond-shaped placeholder with the text "<< Thông tin liên hệ>>".

Người dùng

Đặt lịch hẹn

<<Bài viết về thẩm mỹ viện>>

<< Thông tin liên hệ>>

# Nguyên mẫu giao diện: Đặt lịch hẹn

Người dùng

Màn hình chính/Đặt lịch hẹn

Ngày 10/04/2023

	Gội đầu		Chăm sóc da mặt
	Ghế 1	Ghế 2	
8h30			
9h00			
11h00			
11h30			
13h30			
14h00			

Gửi yêu cầu

76

# Nguyên mẫu giao diện: Xác nhận lịch hẹn

Màn hình chính/Đặt lịch hẹn/Xác nhận

Người dùng

Mã OTP đã nhận:  (Chưa nhận được mã)  
[Gửi lại](#)

Xác nhận

<< Thông tin liên hệ>>

# Nguyên mẫu giao diện: Trạng thái đặt lịch



Người dùng

Màn hình chính/Đặt lịch hẹn/Kết quả

**Đặt lịch hẹn thành công!**

**Tên dịch vụ:** Chăm sóc da mặt

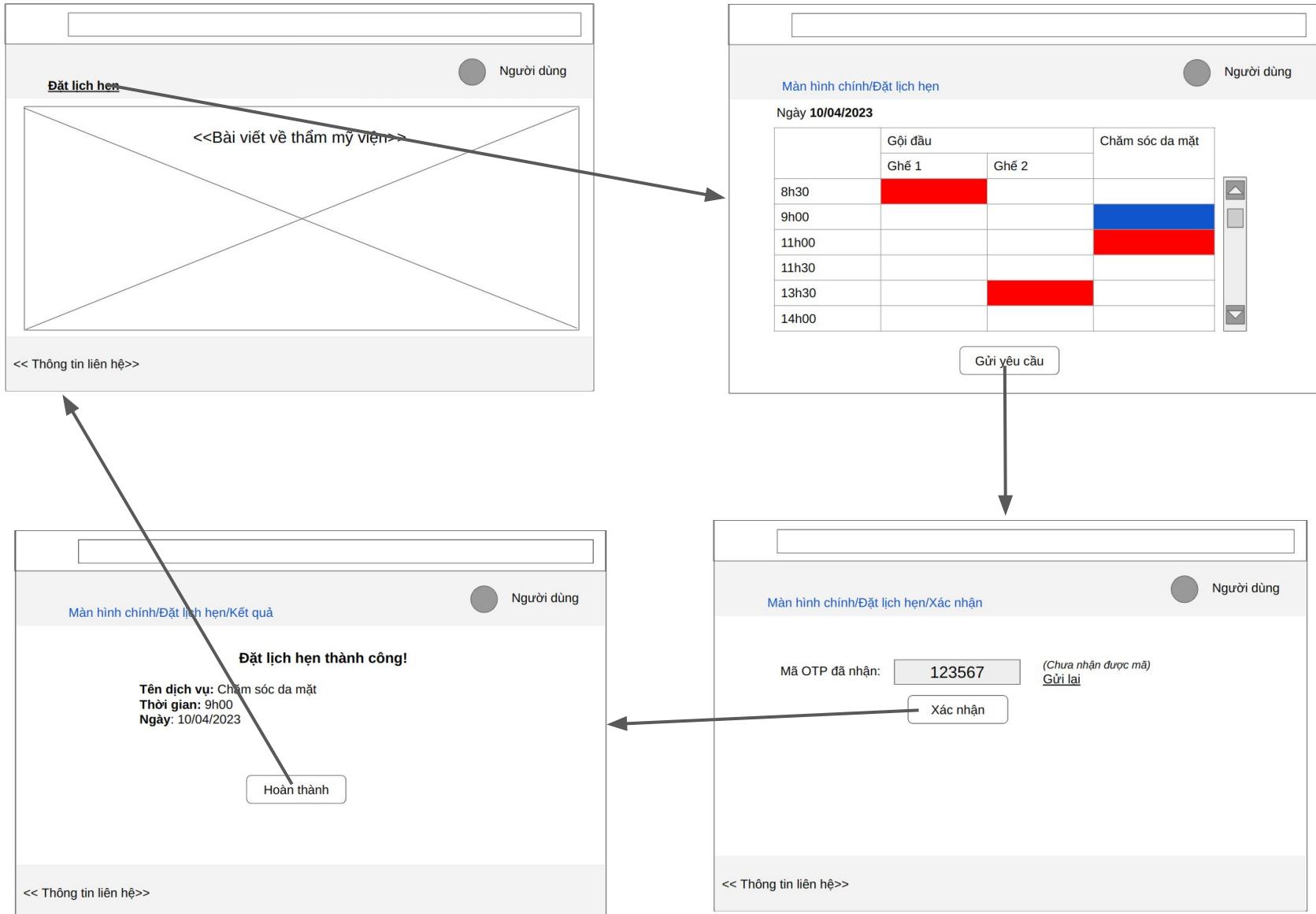
**Thời gian:** 9h00

**Ngày:** 10/04/2023

Hoàn thành

<< Thông tin liên hệ>>

# Bảng phân cảnh Đặt lịch hẹn





# Phân tích và thiết kế hệ thống

Giảng viên: Nguyễn Bá Ngọc

Hà Nội-2022

# Thiết kế lưu trữ cố định

# Nội dung

- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán

# Nội dung

- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán



# Tổng quan về thiết kế lưu trữ cố định

- CSDL (DB) và Hệ quản trị CSDL (DBMS) là thành phần cơ bản thiết yếu của hệ thống thông tin.
- Trong quá trình thiết kế lưu trữ cố định mô hình lĩnh vực được biến đổi thành mô hình CSDL cho hệ thống.
- Hệ quản trị CSDL được sử dụng để triển khai, quản lý và truy cập CSDL.

# Các lựa chọn lưu trữ cố định

- Phát triển riêng, dựa trên tính năng đọc/ghi tệp cơ bản.
- CSDL quan hệ
- CSDL đối tượng-quan hệ
- CSDL hướng đối tượng
- Các mô hình dữ liệu khác:
  - Lưu trữ khóa-giá trị / Key-Value data stores
  - Lưu trữ tài liệu / Document data stores
  - CSDL hướng cột / Columnar database
  - CSDL đồ thị / Graph database
  - V.v..

# CSDL quan hệ

- Lựa chọn lưu trữ dữ liệu phổ biến nhất
- CSDL bao gồm 1 tập bảng
  - Trong mỗi bảng khóa chính xác định 1 dòng duy nhất
  - Mỗi quan hệ giữa các bảng được thiết lập bằng khóa ngoại
    - Cơ chế đảm bảo toàn vẹn dữ liệu giữa các bảng, ví dụ: Từ chối tạo đơn hàng cho 1 khách hàng không tồn tại.
- Ngôn ngữ truy vấn có cấu trúc / Structured Query Language (SQL) được sử dụng để truy cập dữ liệu
  - Quy chuẩn được hỗ trợ bởi nhiều hệ quản trị CSDL
  - Cho phép thao tác với từng bảng và kết hợp nhiều bảng để lấy dữ liệu cần thiết

# CSDL quan hệ-đối tượng

*Khả năng biểu diễn mạnh hơn CSDL quan hệ*

- CSDL quan hệ với khả năng lưu trữ đối tượng.
- Đối tượng được lưu trữ bằng các kiểu dữ liệu do người dùng tự định nghĩa
  - SQL được mở rộng để xử lý các kiểu dữ liệu phức tạp.
  - Hỗ trợ kế thừa ở mức hạn chế.
- Hỗ trợ trường dữ liệu đa trị.

# CSDL hướng đối tượng

*Khả năng biểu diễn mạnh hơn CSDL đối tượng-quan hệ*

- Triển khai các khái niệm hướng đối tượng
- Có thể đặc tả cấu trúc và hành vi của đối tượng
  - Tương thích tốt hơn với các ngôn ngữ hướng đối tượng.
- Sử dụng ngôn ngữ tương tự / được phát triển từ SQL
- Hỗ trợ kế thừa ở mức độ nhất định

*Chiếm thị phần nhỏ trong ứng dụng thực tế*

# Các yêu cầu phi chức năng

- Các yêu cầu phi chức năng có ảnh hưởng đến thiết kế lưu trữ cố định
- Yêu cầu vận hành: Ảnh hưởng tới lựa chọn phần cứng và hệ điều hành.
- Yêu cầu hiệu năng: Các vấn đề tốc độ và dung lượng.
- Yêu cầu bảo mật: Quản lý truy cập, mã hóa.
- v.v..

# Nội dung

- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán



# Ánh xạ đối tượng lĩnh vực ứng dụng

- Với mô hình dữ liệu Hướng đối tượng
  - Mỗi lớp có thể tương ứng với 1 lớp trong lưu trữ.
  - Có thể biểu diễn quan hệ kế thừa.
- Với mô hình dữ liệu Quan hệ-đối tượng
  - Mỗi lớp có thể tương ứng với 1 bảng trong lưu trữ.
  - Có thể tạo đối tượng để biểu diễn thuộc tính đa trị.
- Với mô hình dữ liệu quan hệ
  - Mỗi lớp chỉ chứa các thuộc tính cơ bản có thể tương ứng với 1 bảng trong CSDL quan hệ.
  - Cần tái cấu trúc quan hệ kế thừa và các thuộc tính đa trị.

# Ánh xạ CSDL Hướng Đối tượng

- Trường hợp thuận lợi nhất, tương thích khái niệm.
- Mỗi lớp cụ thể tương ứng với 1 lớp trong lưu trữ đối tượng.
- Có thể cần tái cấu trúc kết cấu đa kế thừa nếu không được hỗ trợ bởi CSDL:
  - R1a - Mô phỏng kế thừa. Thêm thuộc tính vào lớp con trong CSDL để chứa ID đối tượng của đối tượng lớp trên duy trong đa kế thừa.
  - R1b - Lớp con đầy đủ. Sao chép các thuộc tính của lớp trên xuống lớp con để giản lược quan hệ kế thừa.
  - (*Tương tự ánh xạ quan hệ kế thừa sang các mô hình đơn giản hơn.*)

# Ánh xạ CSDL Quan hệ-Đối tượng

*Mỗi đối tượng trong ORDBMS có 1 ID duy nhất*

**R1.** Ánh xạ tất cả lớp lĩnh vực cụ thể và lớp lĩnh vực ảo có nhiều lớp con trực tiếp tới bảng ORDBMS.

**R2.** Ánh xạ thuộc tính đơn trị tới cột của bảng.

**R3.** Ánh xạ phương thức và thuộc tính suy diễn tới thủ tục lưu trữ (stored procedure) hoặc mô-đun chương trình.

**R4.** Ánh xạ quan hệ tổng hợp và quan hệ liên kết đơn trị (0..1 hoặc 1..1) tới cột có thể lưu ID đối tượng. Áp dụng cho cả 2 đầu quan hệ.

**R5.** Ánh xạ các thuộc tính đa trị tới cột có thể lưu 1 tập ID của đối tượng.

**R6.** Ánh xạ nhóm thuộc tính lặp sang 1 bảng mới và tạo liên kết 1-nhiều từ bảng gốc sang bảng mới / thêm khóa ngoại vào bảng mới.

**R7.** Ánh xạ các quan hệ tổng hợp và liên kết đa trị (0..\* hoặc 1..\*) tới cột có thể lưu 1 tập ID đối tượng. Thực hiện cho cả 2 đầu quan hệ.

# Ánh xạ CSDL quan hệ-đối tượng<sub>(2)</sub>

**R8.** Đối với các quan hệ tổng hợp và liên kết hỗn hợp (một-nhiều), ở phía đơn trị (1..1) hoặc (0..1) của mỗi quan hệ, thêm 1 cột để lưu 1 tập ID của các đối tượng ở phía đa trị. Ở phía đa trị (1..\* hoặc 0..\*), thêm 1 cột để lưu ID của đối tượng thuộc đầu đơn trị.

**R9a** - Mô phỏng kế thừa. Thêm cột vào bảng tương ứng với lớp con để chứa ID đối tượng được lưu trong bảng tương ứng với lớp cha. Cơ số của liên kết mới từ lớp con tới lớp cha phải là 1..1 / 1 đối tượng lớp con xác định đúng 1 đối tượng lớp cha. Thêm cột vào bảng biểu diễn lớp cha để lưu ID đối tượng của lớp con. Nếu lớp cha là cụ thể, thì cơ số từ lớp cha tới lớp con là 0..1: 1 đối tượng lớp cha có thể xác định 1 đối tượng lớp con, nếu ngược lại thì cơ số là 1..1: xác định đúng 1 đối tượng lớp con. Ràng buộc XOR (OR loại trừ) cần được bổ xung cho các liên kết này. Thực hiện với tất cả các lớp cha. Hoặc

**R9b** - Lớp con đầy đủ. Mở rộng lớp con bằng cách sao chép các thuộc tính của lớp cha xuống các lớp con và loại lớp cha trừu tượng khỏi thiết kế. Hoặc

**R9c** - Làm phẳng cây kế thừa. Sao chép tất cả các thuộc tính trong cây vào lớp gốc, tạo 1 bảng cho cây kế thừa.

# Ánh xạ tới CSDL quan hệ

- R1.** Ánh xạ lớp lĩnh vực cụ thể và lớp lĩnh vực trừu tượng có nhiều lớp con trực tiếp tới bảng RDBMS.
- R2.** Ánh xạ thuộc tính đơn trị tới cột của bảng.
- R3.** Ánh xạ phương thức và thuộc tính suy diễn tới thủ tục lưu trữ hoặc mô-đun phần mềm.
- R4.** Ánh xạ các quan hệ tổng hợp và liên kết đơn trị tới 1 cột có thể lưu khóa của bảng liên quan / thêm 1 khóa ngoại vào bảng. Thực hiện cho cả 2 đầu liên kết.
- R5.** Ánh xạ các thuộc tính đa trị và các nhóm lặp tới các bảng mới và tạo quan hệ 1-nhiều từ bảng gốc tới bảng mới / thêm khóa ngoại vào bảng mới.
- R6.** Ánh xạ các quan hệ tổng hợp và liên kết đa trị tới 1 bảng liên kết mới để kết nối 2 bảng ban đầu. Sao chép khóa chính từ các bảng gốc vào bảng liên kết/thêm các khóa ngoại vào bảng liên kết.

# Ánh xạ tới CSDL quan hệ<sub>(2)</sub>

**R7.** Đối với các quan hệ tổng hợp và liên kết hỗn hợp, sao chép khóa chính từ phía đơn trị (1..1 hoặc 0..1) của mỗi quan hệ vào 1 cột mới trong bảng của phía đa trị (1..\* hoặc 0..\*) để lưu khóa của bảng liên quan / bổ xung 1 khóa ngoại vào bảng ở phía đa trị của mỗi quan hệ.

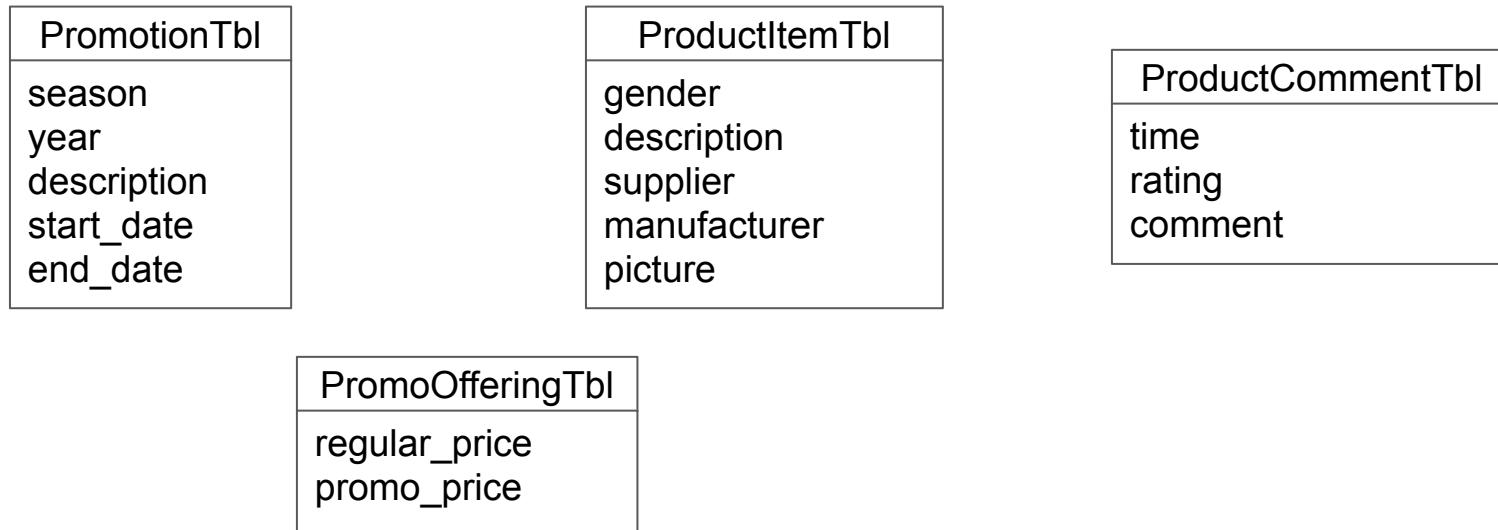
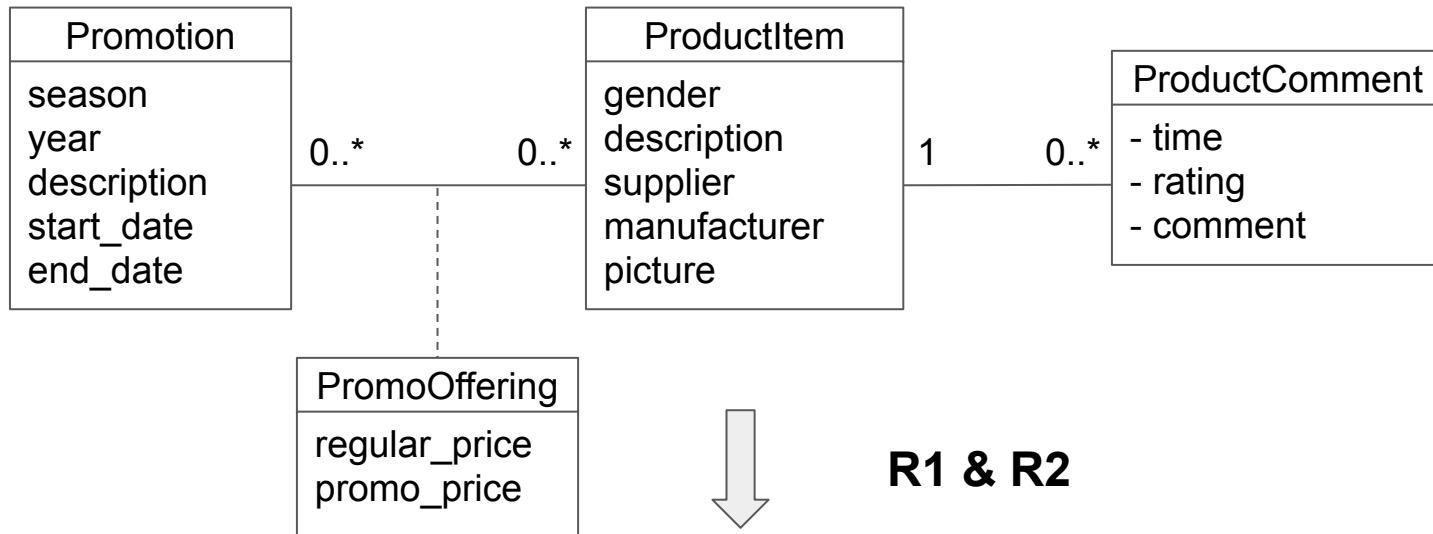
Đối với quan hệ Khái quát hóa/Kế thừa:

**R8a** - Mô phỏng kế thừa. Đảm bảo khóa chính của đối tượng thuộc lớp con giống với khóa chính của đối tượng thuộc lớp cha. Cơ số của liên kết từ lớp con tới lớp cha phải là 1..1. Nếu lớp cha là lớp cụ thể thì cơ số từ lớp cha tới lớp con là 0..1, nếu ngược lại thì cơ số là 1..1. Bên cạnh đó ràng buộc XOR (OR loại trừ) phải được thêm vào giữa các liên kết. Áp dụng quy tắc này cho tất cả các lớp cha. Hoặc

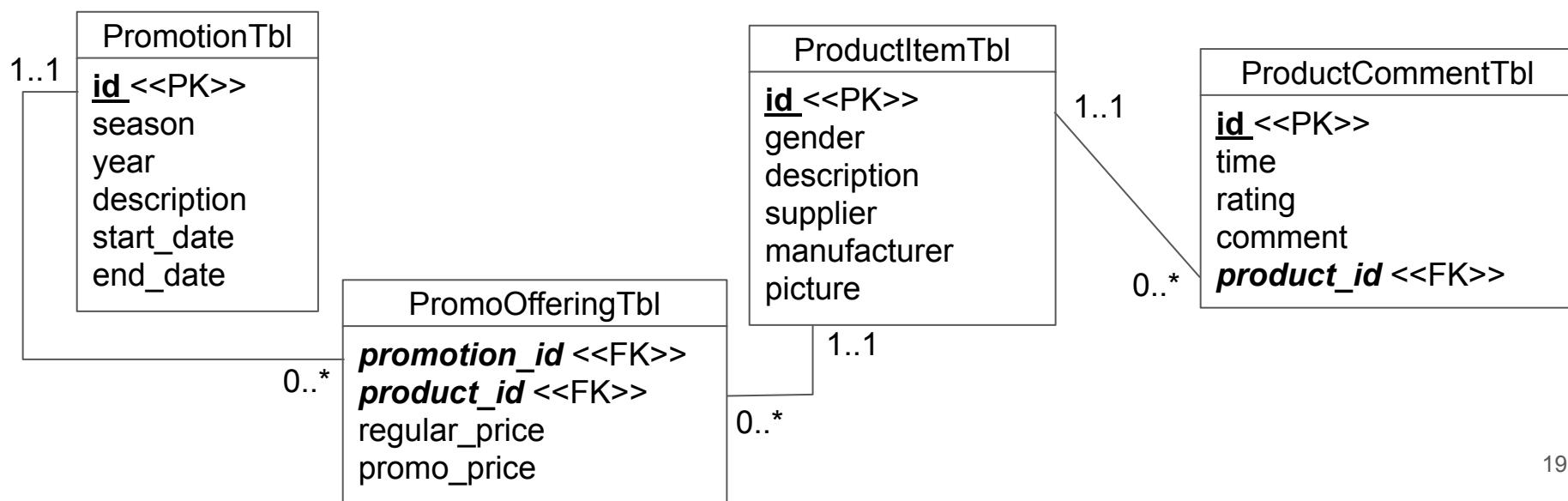
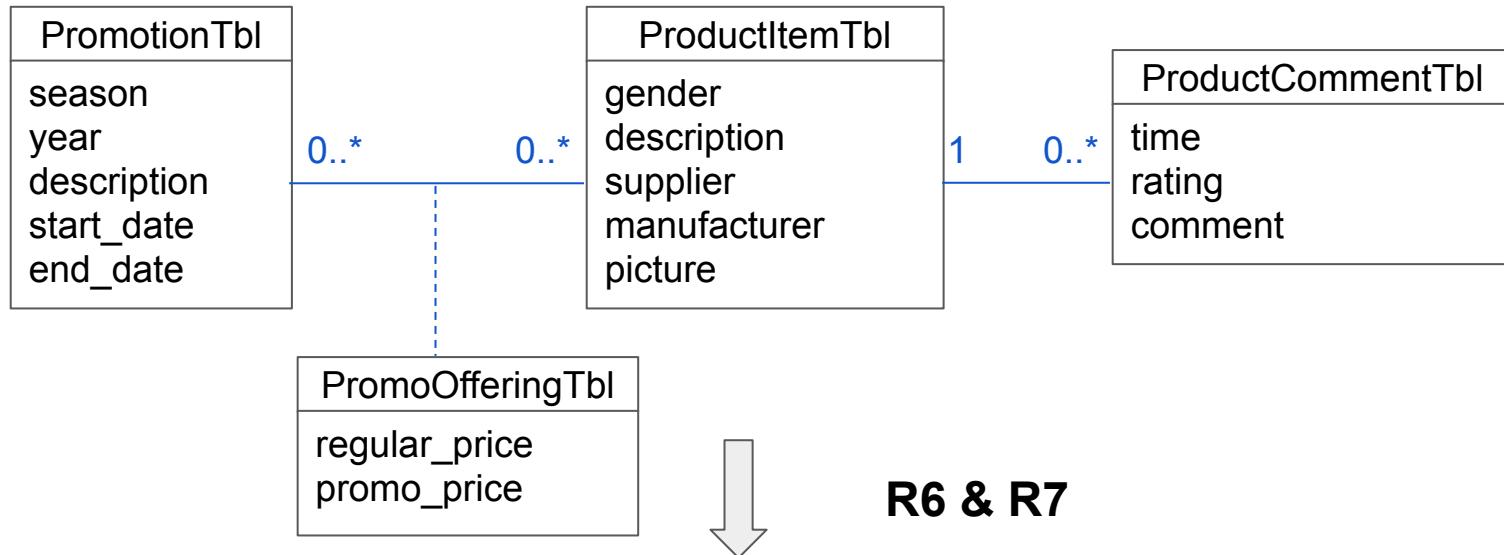
**R8b** - Lớp con đầy đủ. Mở rộng lớp con bằng cách sao chép thuộc tính của lớp cha vào tất cả các lớp con là xóa lớp cha khái quát khỏi thiết kế. Hoặc

**R8c** - Làm phẳng cây kế thừa. Sao chép tất cả các thuộc tính trong cây vào lớp gốc, tạo 1 bảng cho cây kế thừa.

# Ví dụ 1. Ánh xạ sang CSDL quan hệ



# Ví dụ 1. Ánh xạ sang CSDL quan hệ<sub>(2)</sub>



# Đặc tả bảng

<b>Bảng ProductCommentTbl</b>				
<b>STT</b>	<b>Tên cột</b>	<b>Kiểu dữ liệu</b>	<b>Ràng buộc</b>	<b>Ghi chú</b>
1	id	INT	AUTO_INCREMENT	Mã bình luận
2	time	TIMESTAMP	NOT NULL	Thời điểm bình luận
3	rating	INT	NOT NULL	Điểm bình luận
4	comment	TEXT	NOT NULL	Nội dung bình luận
5	product_id	INT	NOT NULL	Khóa của sản phẩm
Khóa chính		id		
Khóa ngoại		product_id = ProductTbl.id		

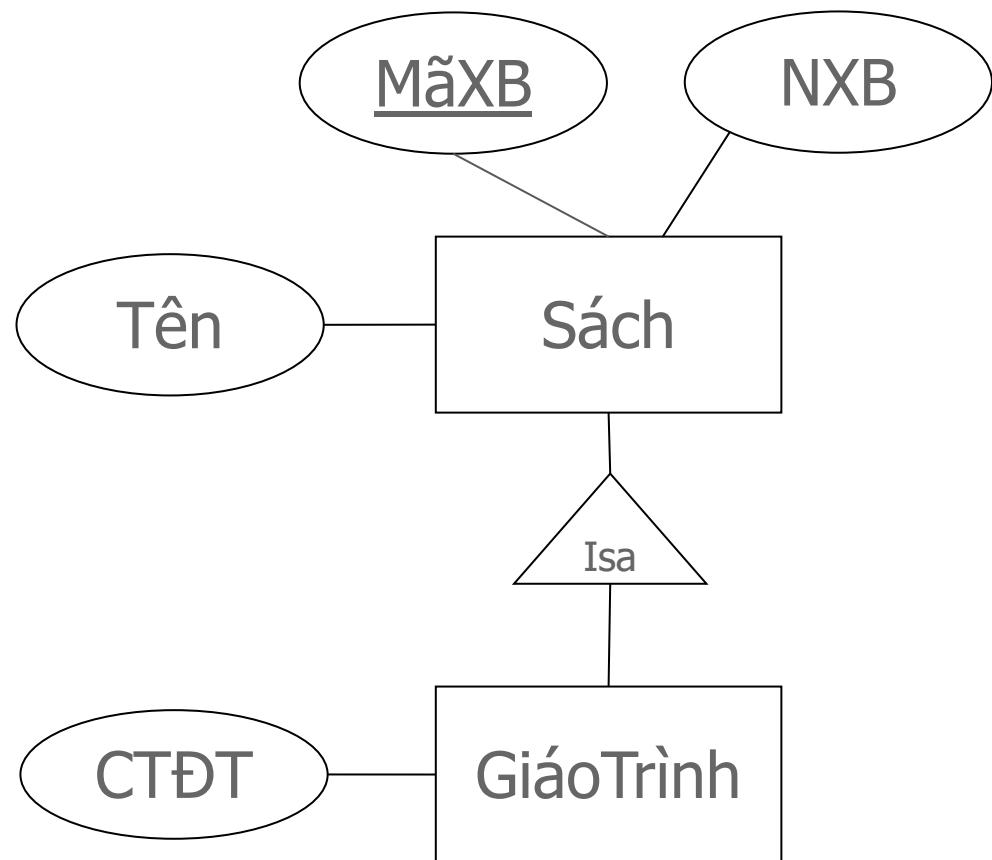
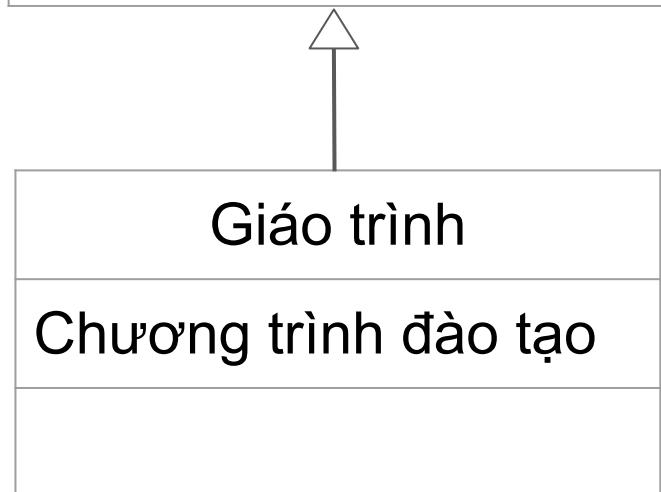
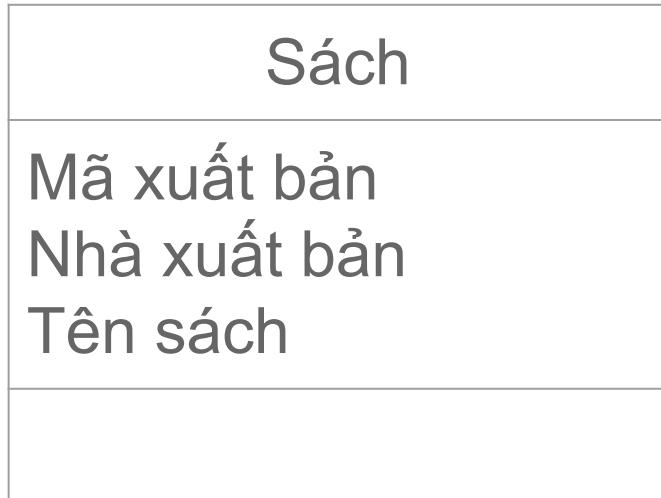
...

# Lưu trữ lớp con với RDBMS

Có nhiều cách lưu khác nhau:

1. *Mô phỏng kế thừa*: Tạo cho mỗi lớp con 1 bảng lưu các thuộc tính riêng, đồng thời kết nối với bảng tương ứng với lớp cha.
2. *Lớp con đầy đủ*: Tạo cho mỗi lớp con 1 bảng lưu đầy đủ các thuộc tính, bao gồm cả thuộc tính riêng và thuộc tính kế thừa từ các lớp cha.
3. *Làm phẳng cây kế thừa*: Tạo 1 bảng cho 1 cây kế thừa với tất cả các thuộc tính trong cây kế thừa.
  - Các đối tượng nhận giá trị NULL cho các thuộc tính không thuộc về chúng.
  - Dữ liệu có thể ở dạng phi chuẩn.

## Ví dụ 2. Quan hệ kế thừa



# Ví dụ 3. Mô phỏng kế thừa

Sách

Tên	NXB	<u>Mã XB</u>	
LT HĐT	KHKT	K123	
Tin ĐC	BKB	B111	

Giáo trình

CTĐT	<u>Mã XB</u>
KHMT	B111

*Lớp con có thể phân tán trên nhiều bảng*

*Thuận tiện xử lý các truy vấn kiểu Tìm tất cả sách (bao gồm cả các giáo trình) được xuất bản bởi 1 nhà xuất bản cụ thể. Tìm kiếm theo thuộc tính của nút gốc trong phạm vi tất cả đối tượng thuộc cây kế thừa.*

# Ví dụ 4. Lớp con đầy đủ

Sách

Tên	NXB	<b><u>Mã XB</u></b>
LT HĐT	KHKT	K123

Giáo trình

*Không có liên kết giữa bảng của lớp cha và bảng của lớp con*

Tên	NXB	<b><u>Mã XB</u></b>	CTĐT
Tin ĐC	BKB	B111	KHMT

*Thuận tiện để xử lý các truy vấn như “tìm CTĐT của các giáo trình được xuất bản bởi BKB.” (sử dụng cả thuộc tính riêng của lớp con và thuộc tính kế thừa từ lớp cha)*

# Ví dụ 5. Làm phẳng cây kế thừa

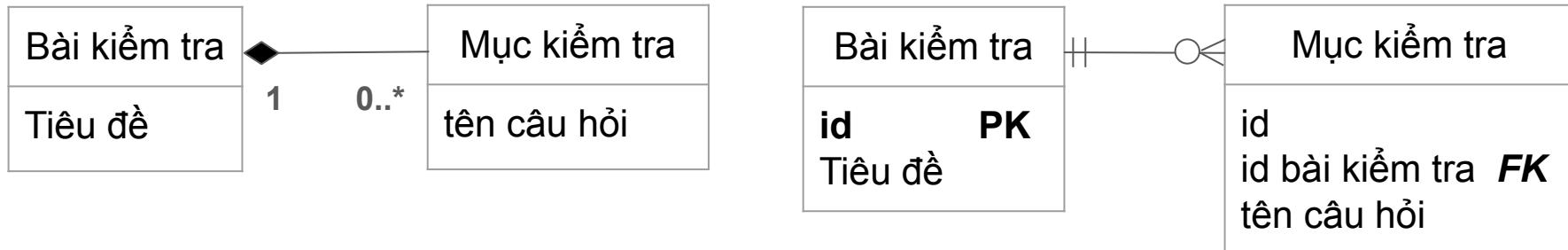
<u>Mã số</u>	Phân loại sách
1	Sách
2	Sách Giáo Trình

Tên	NXB	<u>Mã XB</u>	CTĐT	Kiểu
Tin ĐC	BKB	B111	KHMT	1
LT HĐT	KHKT	KT123	NULL	2

Có thể tiết kiệm dung lượng nếu có ít thuộc tính thường xuyên có giá trị NULL

!Lưu ý: Dữ liệu ở dạng phi chuẩn.

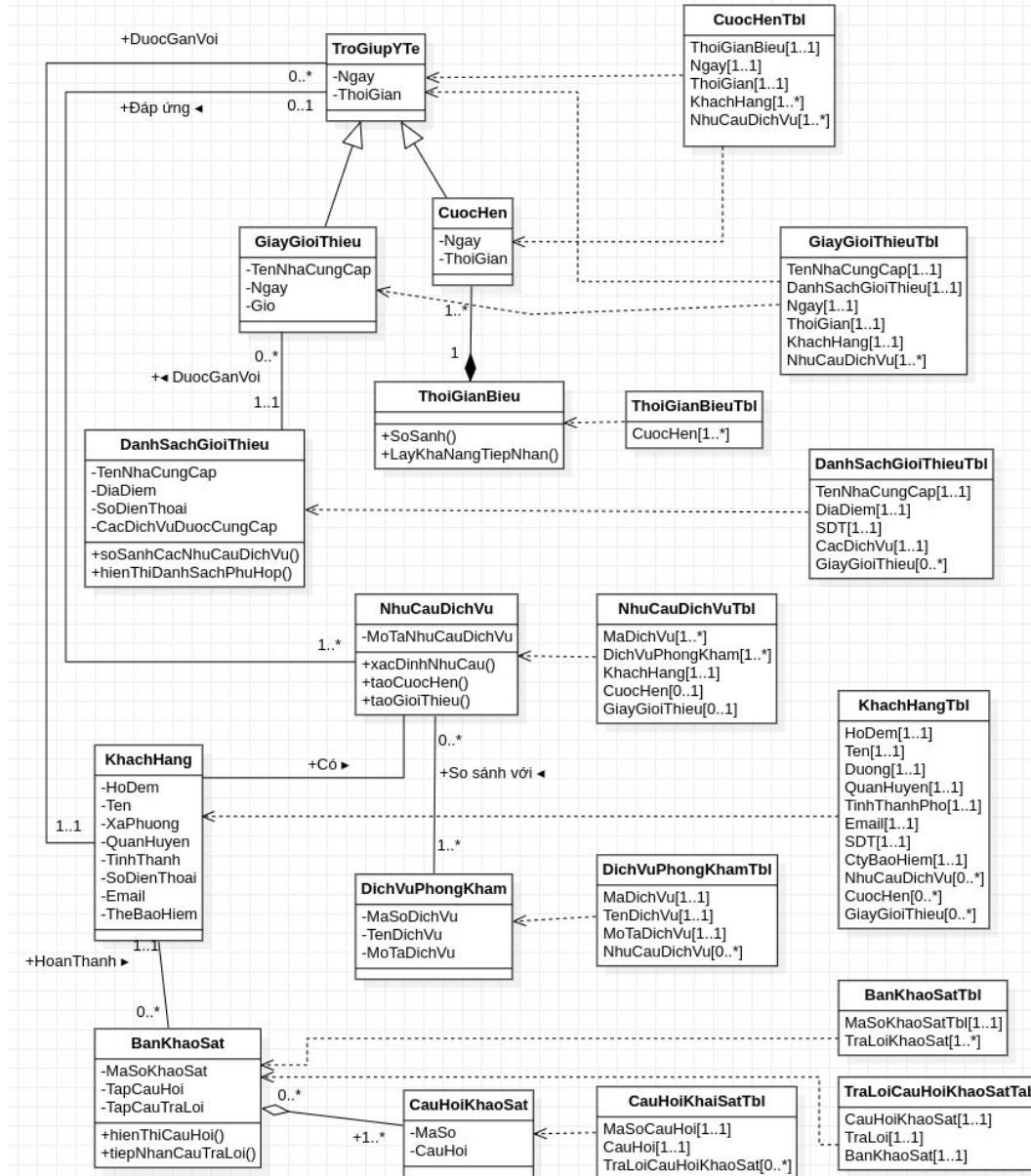
# Ví dụ 6. Lưu trữ đối tượng: Quan hệ 1-nhiều



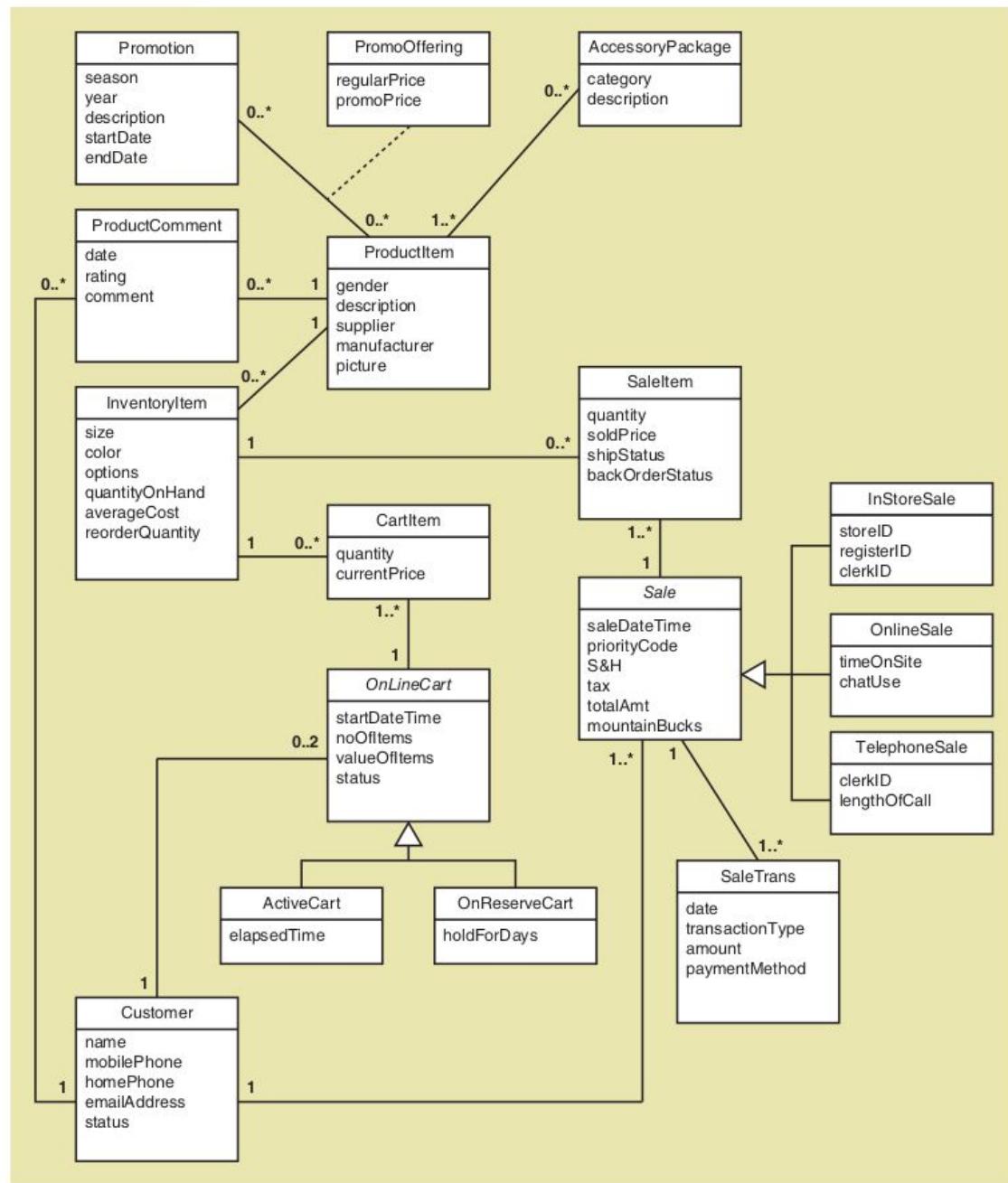
<b>id</b>	<b>Tiêu đề</b>
1	Mô hình hóa chức năng
3	Kiến trúc hệ thống
...	...

<b>id</b>	<b>id bài KT</b>	<b>Tên câu hỏi</b>
1	1	Khái niệm tác nhân
2	1	Khái niệm ca sử dụng
3	1	Đặc tả chi tiết ca sử dụng
...	...	...

# Ví dụ 7. Ánh xạ lớp linh vực tới ORDBMS



# Ví dụ 8. Ánh xạ RDBMS



# Ví dụ 8. Ánh xạ RDBMS: Tạo bảng

Áp dụng R1, R2, & R8c: Lớp => Bảng, Thuộc tính => Cột, Làm phẳng cây kế thừa

Bảng	Các cột
AccessoryPackage	Category, Description
CartItem	Quantity, CurrentPrice
Customer	Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	StartTime, NumberofItems, ValueofItems, Status, ElapsedTime, HoldForDays
ProductComment	Date, Rating, Comment
ProductItem	Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	RegularPrice, PromoPrice
Promotion	Season, Year, Description, StartDate, EndDate
Sale	SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	Date, TransactionType, Amount, PaymentMethod

# Ví dụ 8. Ánh xạ RDBMS: Bổ xung khóa chính

Bảng	Các cột
AccessoryPackage	<b>AccessoryPackageID</b> , Category, Description
CartItem	<b>CartItemID</b> , Quantity, CurrentPrice
Customer	<b>AccountNumber</b> , Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	<b>InventoryItemID</b> , Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	<b>OnlineCartID</b> , StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	<b>ProductCommentID</b> , Date, Rating, Comment
ProductItem	<b>ProductItemID</b> , Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	<b>PromoOfferingID</b> , RegularPrice, PromoPrice
Promotion	<b>PromotionID</b> , Season, Year, Description, StartDate, EndDate
Sale	<b>SaleID</b> , SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	<b>SaleItemID</b> , Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	<b>SaleTransactionID</b> , Date, TransactionType, Amount, PaymentMethod

# Ví dụ 8. Ánh xạ RDBMS: Các liên kết

Áp dụng R4, R6, & R7: Bổ xung các khóa ngoại

Bảng	Các cột
Accessory Package	<b>AccessoryPackageID</b> , Category, Description
AccessoryPackageContents	<b>AccessoryPackageID</b> , <b>ProductItemID</b>
Cartitem	<b>CartItemID</b> , <i>InventoryItemID</i> , <i>OnlineCartID</i> , Quantity, CurrentPrice
Customer	<b>AccountNumber</b> , Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	<b>InventoryItemID</b> , <b>ProductItemID</b> , Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	<b>OnlineCartID</b> , <i>CustomerAccountNumber</i> , StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ProductComment	<b>ProductCommentID</b> , <b>ProductItemID</b> , <i>CustomerAccountNumber</i> , Date, Rating, Comment
ProductItem	<b>ProductItemID</b> , Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	<b>PromoOfferingID</b> , <b>PromotionID</b> , <b>ProductItemID</b> , RegularPrice, PromoPrice
Promotion	<b>PromotionID</b> , Season, Year, Description, StartDate, EndDate
Sale	<b>SaleID</b> , <i>CustomerAccountNumber</i> , SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks, StoreID, RegisterID, ClerkID, TimeOnSite, ChatUse, LengthOfCall
SaleItem	<b>SaleItemID</b> , <i>InventoryItemID</i> , <b>SaleID</b> , Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	<b>SaleTransactionID</b> , <b>SaleID</b> , Date, TransactionType, Amount, PaymentMethod

# Ví dụ 8. Ánh xạ RDBMS: Mô phỏng kế thừa

R8a: Tạo cho mỗi lớp con 1 bảng & thiết lập liên kết với bảng của lớp cha

Bảng	Các cột
Accessory Package	<b>AccessoryPackageID</b> , Category, Description
AccessoryPackageContents	<b>AccessoryPackageID</b> , <b>ProductItemID</b>
Cartitem	<b>CartItemID</b> , <i>InventoryItemID</i> , <i>OnlineCartID</i> , Quantity, CurrentPrice
Customer	<b>AccountNumber</b> , Name, MobilePhone, HomePhone, EmailAddress, Status
InventoryItem	<b>InventoryItemID</b> , <b>ProductItemID</b> , Size, Color, Options, QuantityOnHand, AverageCost, ReorderQuantity
OnlineCart	<b>OnlineCartID</b> , <i>CustomerAccountID</i> , StartDateTime, NumberOfItems, ValueOfItems, Status, ElapsedTime, HoldForDays
ActiveCart	<b>OnlineCartID</b> , ElapsedTime
OnReserveCart	<b>OnlineCartID</b> , HoldForDays
ProductComment	<b>ProductCommentID</b> , <b>ProductItemID</b> , <i>CustomerAccountNumber</i> , Date, Rating, Comment
ProductItem	<b>ProductItemID</b> , Gender, Description, Supplier, Manufacturer, Picture
PromoOffering	<b>PromoOfferingID</b> , <b>PromotionID</b> , <b>ProductItemID</b> , RegularPrice, PromoPrice
Promotion	<b>PromotionID</b> , Season, Year, Description, StartDate, EndDate

# Ví dụ 8. Ánh xạ RDBMS: Mô phỏng kế thừa<sub>(2)</sub>

R8a: Tạo cho mỗi lớp con 1 bảng & thiết lập liên kết với bảng của lớp cha

Bảng	Các cột
Sale	<b>SaleID</b> , CustomerAccountNumber, SaleDateTime, PriorityCode, ShippingAndHandling, Tax, TotalAmount, MountainBucks
InStoreSale	<b>SaleID</b> , StoreID, RegisterID, ClerkID
OnlineSale	<b>SaleID</b> , TimeOnSite, ChatUse
TelephoneSale	<b>SaleID</b> , ClerkID, LengthOfCall
SaleItem	<b>SaleItemID</b> , InventoryItemID, <b>SaleID</b> , Quantity, SoldPrice, ShipStatus, BackOrderStatus
SaleTransaction	<b>SaleTransactionID</b> , <b>SaleID</b> , Date, TransactionType, Amount, PaymentMethod

# Nội dung

- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán



# Tối ưu hóa CSDL quan hệ

- Tối ưu hóa lưu trữ
  - Chuẩn hóa các bảng
  - Giảm dư thừa dữ liệu và loại bỏ các giá trị NULL
- Tối ưu hóa tốc độ truy cập
  - Giải chuẩn 1 số bảng / giảm nhu cầu kết nối các bảng để truy xuất dữ liệu để giảm thời gian xử lý
  - v.v..

# Tối ưu hóa CSDL quan hệ

- Dữ liệu phi chuẩn: Các luật chuẩn hóa không được đáp ứng

3 mức chuẩn hóa đầu tiên gồm có:

- Dạng chuẩn 1: Không có trường đa giá trị/Thuộc tính chỉ có giá trị nguyên tố
- Dạng chuẩn 2: Thuộc chuẩn 1 và tất cả các thuộc tính không nằm trong khóa đều phụ thuộc vào toàn bộ khóa chính
- Dạng chuẩn 3: Thuộc chuẩn 2 và mọi thuộc tính không khóa chỉ phụ thuộc vào khóa / Không có thuộc tính không nằm trong khóa phụ thuộc vào thuộc tính khác không nằm trong khóa

# Ví dụ 9. Chuẩn hóa: Dữ liệu phi chuẩn

SSN	Name	Department	Salary	Dependents
111-22-3333	Mary Smith	Accounting	40,000	John, Alice, Dave
222-33-4444	Jose Pena	Marketing	50,000	---
333-44-5555	Frank Collins	Production	35,000	Jan, Julia

SSN	Name	Department	Salary	Dependent	Dependent	Dependent
111-22-3333	Mary Smith	Accounting	40,000	John	Alice	Dave
222-33-4444	Jose Pena	Marketing	50,000			
333-44-5555	Frank Collins	Production	35,000	Jan	Julia	

# Ví dụ 9. Chuẩn hóa: Dữ liệu ở dạng chuẩn

SSN	Name	Department	Salary
111-22-3333	Mary Smith	Accounting	40,000
222-33-4444	Jose Pena	Marketing	50,000
333-44-5555	Frank Collins	Production	35,000

RecordID	SSN	Dependent
1	111-22-3333	John
2	111-22-3333	Alice
3	111-22-3333	Dave
4	333-44-5555	Jan
5	333-44-5555	Julia

# Ví dụ 10. Chuẩn hóa

Dữ liệu phi chuẩn

Sample Records:

Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate	Prod. 1 Number	Prod. 1 Desc.	Prod. 1 Price	Prod. 1 Qty.	Prod. 2 Number	Prod. 2 Desc.	Prod. 2 Price	Prod. 2 Qty.
239	11/23/00	1035	Black	John	MD	0.05	555	Cheese Tray	\$45.00	2				
260	11/24/00	1035	Black	John	MD	0.05	444	Wine Gift Pack	\$60.00	1				
273	11/27/00	1035	Black	John	MD	0.05	222	Bottle Opener	\$12.00	1				
241	11/23/00	1123	Williams	Mary	CA	0.08	444	Wine Gift Pack	\$60.00	2				
262	11/24/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	\$12.00	2				
287	11/27/00	1123	Williams	Mary	CA	0.08	222	Bottle Opener	\$12.00	2				
290	11/30/00	1123	Williams	Mary	CA	0.08	555	Cheese Tray	\$45.00	3				
234	11/23/00	2242	DeBerry	Ann	DC	0.065	555	Cheese Tray	\$45.00	2				
237	11/23/00	2242	DeBerry	Ann	DC	0.065	111	Wine Guide	\$15.00	1	444	Wine Gift Pack	\$60.00	1
238	11/23/00	2242	DeBerry	Ann	DC	0.065	444	Wine Gift Pack	\$60.00	1				
245	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	\$12.00	1				
250	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	\$12.00	1				
252	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	\$12.00	1	444	Wine Gift Pack	\$60.00	2
253	11/24/00	2242	DeBerry	Ann	DC	0.065	222	Bottle Opener	\$12.00	1	444	Wine Gift Pack	\$60.00	1



Null Cells

--	--	--	--	--

# Ví dụ 10. Chuẩn hóa: Dạng chuẩn 1

Sample Records:

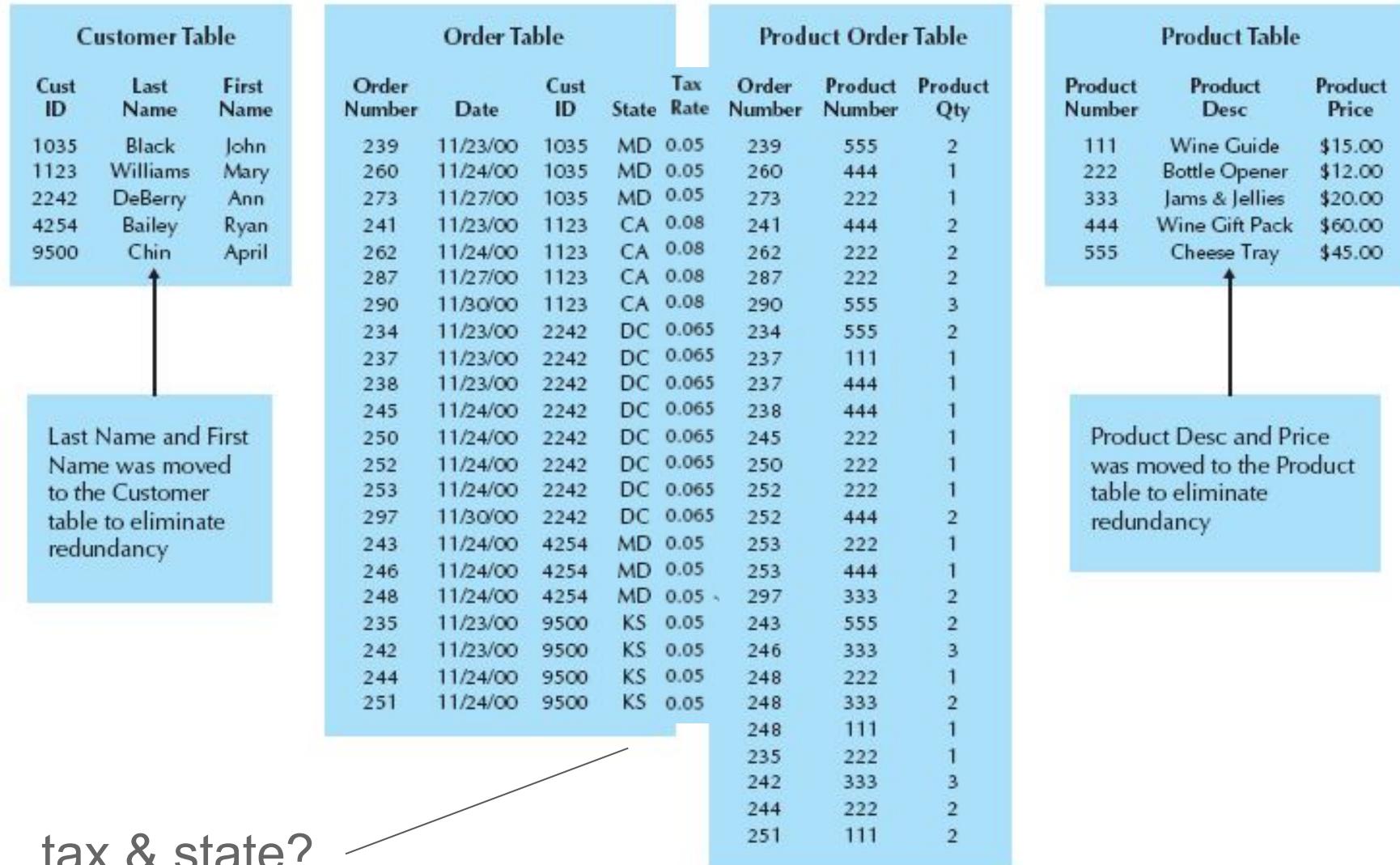
Order Table						
Order Number	Date	Cust ID	Last Name	First Name	State	Tax Rate
239	11/23/00	1035	Black	John	MD	0.05
260	11/24/00	1035	Black	John	MD	0.05
273	11/27/00	1035	Black	John	MD	0.05
241	11/23/00	1123	Williams	Mary	CA	0.08
262	11/24/00	1123	Williams	Mary	CA	0.08
287	11/27/00	1123	Williams	Mary	CA	0.08
290	11/30/00	1123	Williams	Mary	CA	0.08
234	11/23/00	2242	DeBerry	Ann	DC	0.065
237	11/23/00	2242	DeBerry	Ann	DC	0.065
238	11/23/00	2242	DeBerry	Ann	DC	0.065
245	11/24/00	2242	DeBerry	Ann	DC	0.065
250	11/24/00	2242	DeBerry	Ann	DC	0.065
252	11/24/00	2242	DeBerry	Ann	DC	0.065
253	11/24/00	2242	DeBerry	Ann	DC	0.065
297	11/30/00	2242	DeBerry	Ann	DC	0.065
243	11/24/00	4254	Bailey	Ryan	MD	0.05
246	11/24/00	4254	Bailey	Ryan	MD	0.05
248	11/24/00	4254	Bailey	Ryan	MD	0.05
235	11/23/00	9500	Chin	April	KS	0.05
242	11/23/00	9500	Chin	April	KS	0.05
244	11/24/00	9500	Chin	April	KS	0.05
251	11/24/00	9500	Chin	April	KS	0.05

Order 237 has 2 products

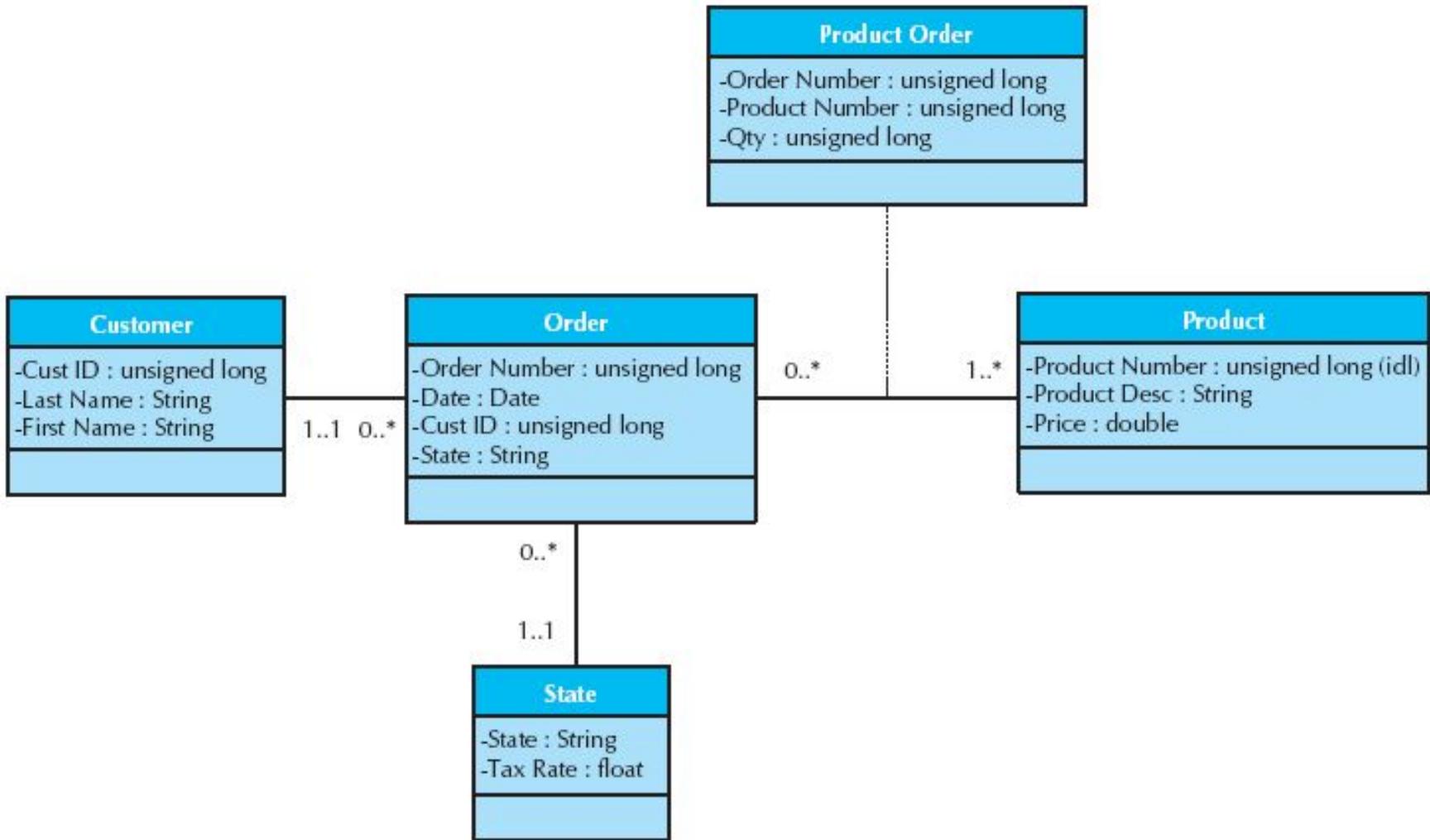
Product Order Table				
Order Number	Product Number	Product Desc	Product Price	Product Qty
239	555	Cheese Tray	\$45.00	2
260	444	Wine Gift Pack	\$60.00	1
273	222	Bottle Opener	\$12.00	1
241	444	Wine Gift Pack	\$60.00	2
262	222	Bottle Opener	\$12.00	2
287	222	Bottle Opener	\$12.00	2
290	555	Cheese Tray	\$45.00	3
234	555	Cheese Tray	\$45.00	2
237	111	Wine Guide	\$15.00	1
237	444	Wine Gift Pack	\$60.00	1
238	444	Wine Gift Pack	\$60.00	1
245	222	Bottle Opener	\$12.00	1
250	222	Bottle Opener	\$12.00	1
252	222	Bottle Opener	\$12.00	1
253	222	Bottle Opener	\$12.00	1
252	444	Wine Gift Pack	\$60.00	2
253	222	Bottle Opener	\$12.00	1
253	444	Wine Gift Pack	\$60.00	1
297	333	Jams & Jellies	\$20.00	2
243	555	Cheese Tray	\$45.00	2
246	333	Jams & Jellies	\$20.00	3
248	222	Bottle Opener	\$12.00	1
248	333	Jams & Jellies	\$20.00	2
248	111	Wine Guide	\$15.00	1
235	222	Bottle Opener	\$12.00	1
242	333	Jams & Jellies	\$20.00	3
244	222	Bottle Opener	\$12.00	2
251	111	Wine Guide	\$15.00	2

Order 248 has 3 products

# Ví dụ 10. Chuẩn hóa: Dạng chuẩn 2



# Ví dụ 10. Chuẩn hóa: 3NF



# Nội dung

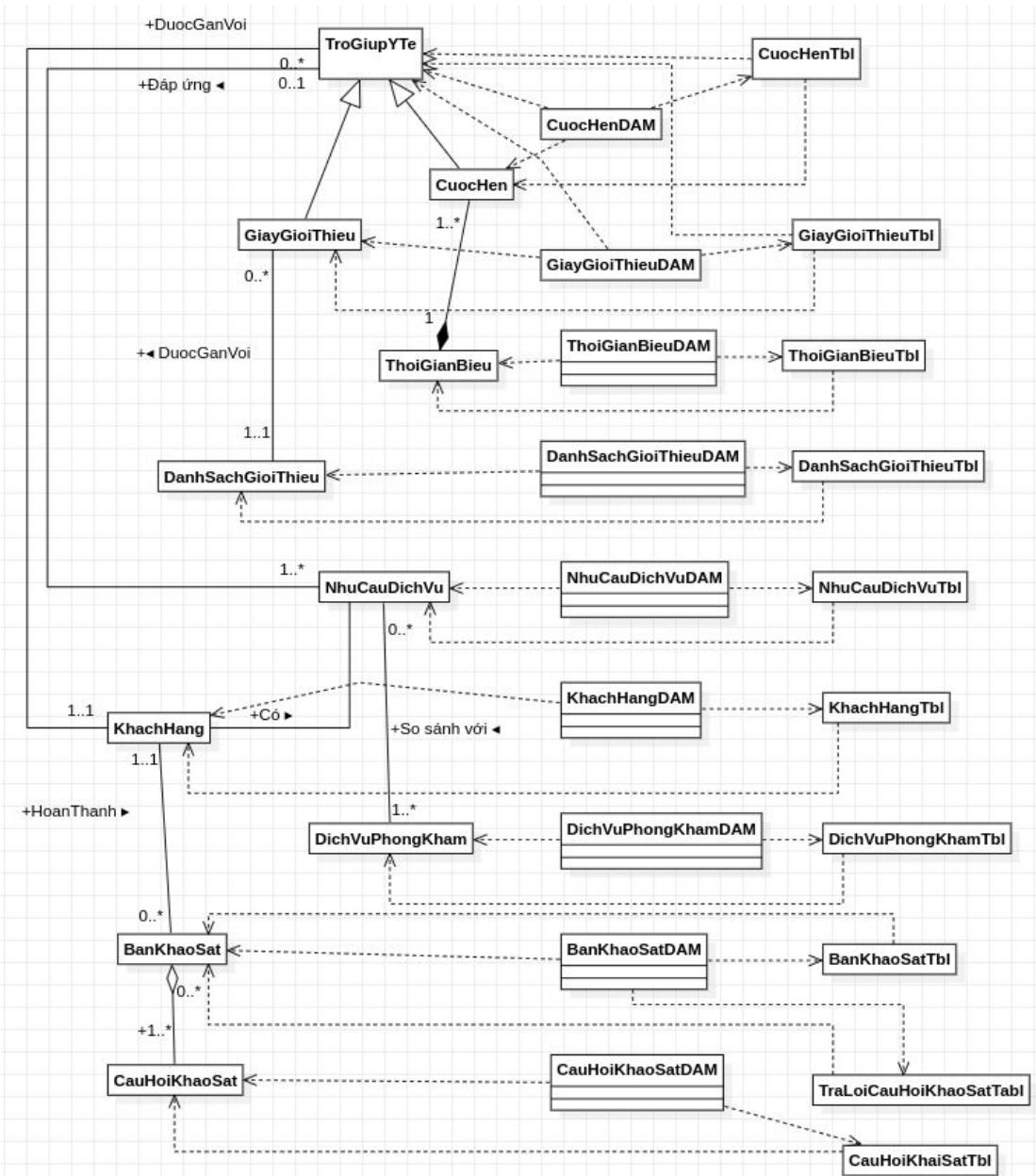
- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán



# Các lớp truy cập & quản lý dữ liệu

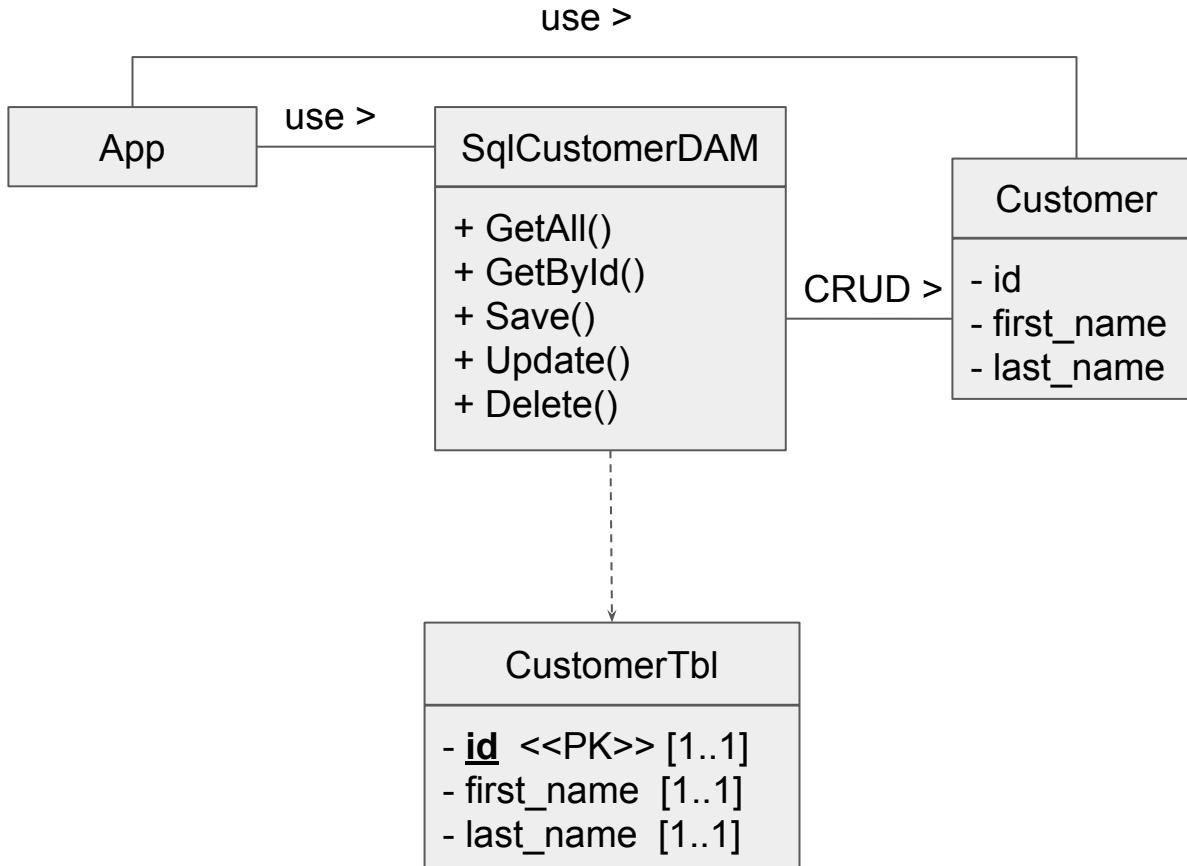
- Các lớp truy cập và quản lý dữ liệu / Data Access and Management (DAM) Classes
- Giữ vai trò trung gian kết nối các lớp lĩnh vực ứng dụng và cơ sở dữ liệu
- Tạo 1 hoặc nhiều lớp DAM cho mỗi lớp lĩnh vực ứng dụng
  - Do dữ liệu phân tán trên nhiều bảng
  - Phân tách giao diện (ISP)
  - v.v..

# Ví dụ 11. Các lớp DAM



# Mẫu thiết kế DAO

# Sử dụng lớp DAM và RDBMS

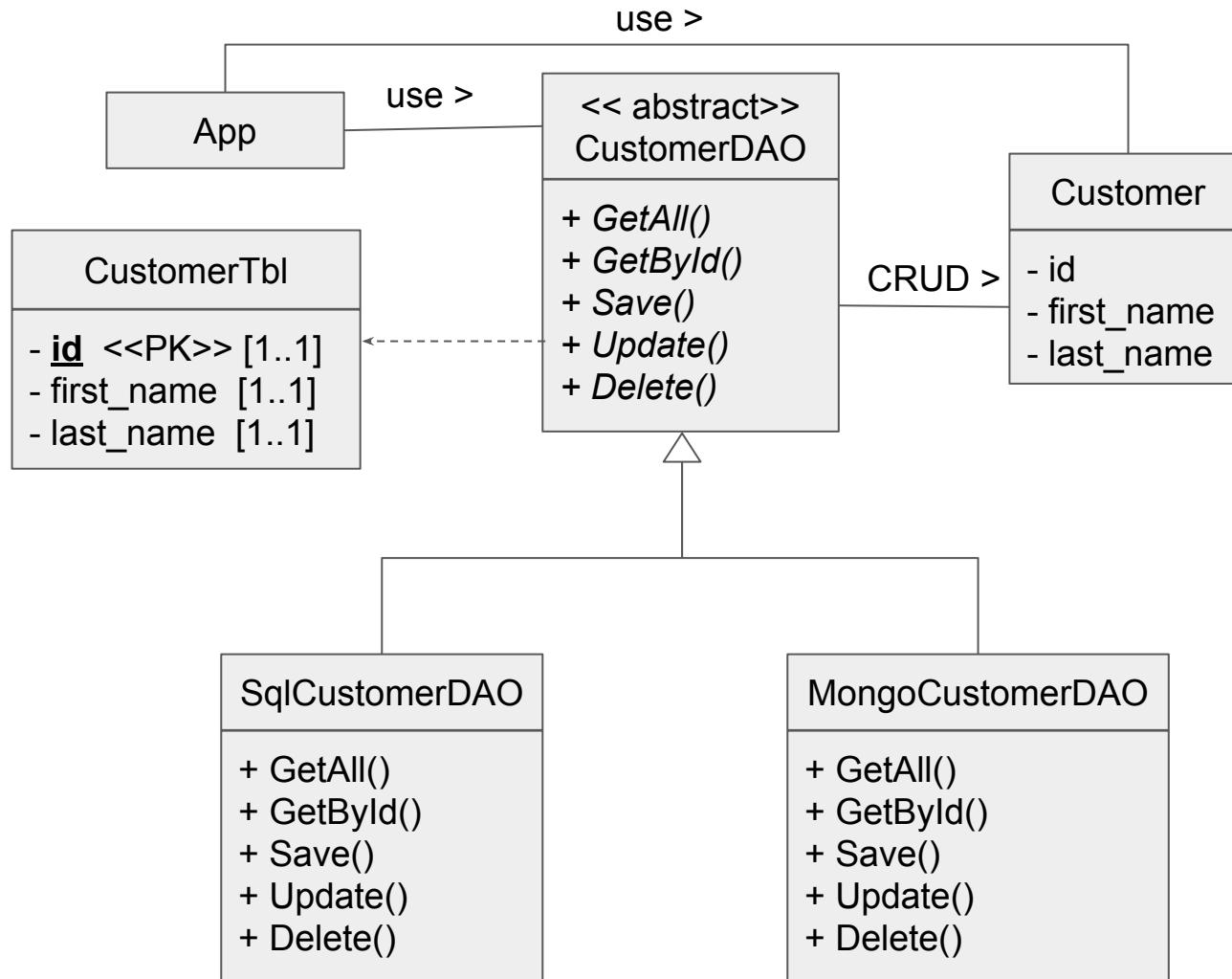


# Đặc tả bảng

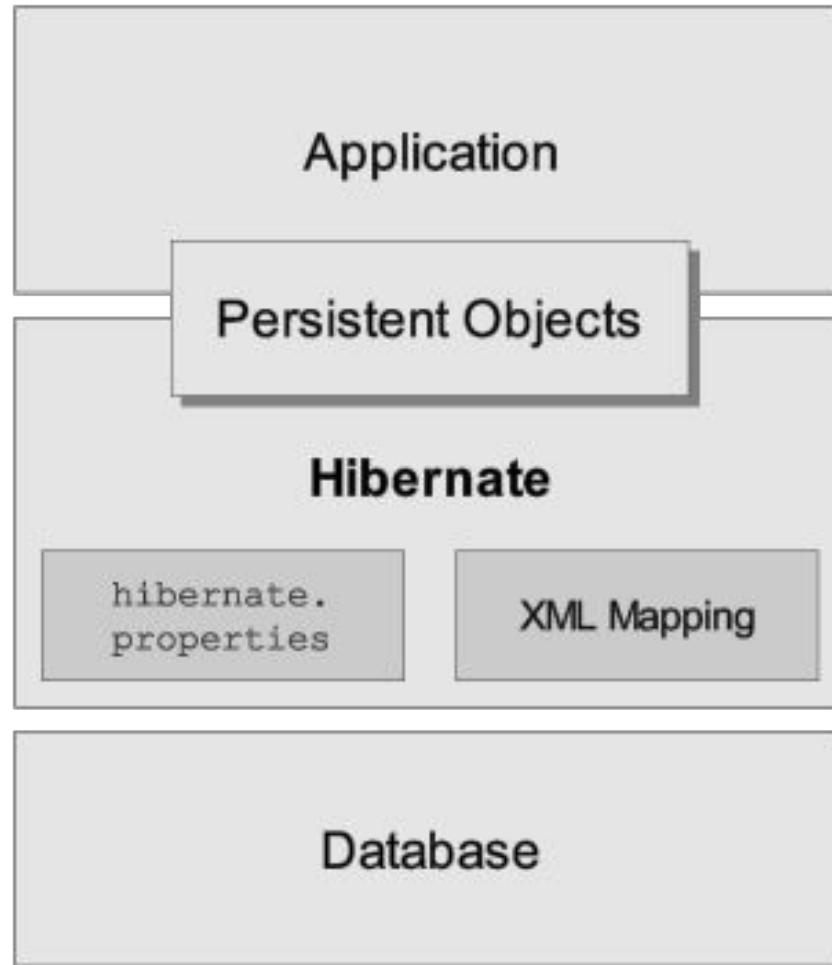
CustomerTbl
- <b>id</b> <>PK>> [1..1]
- first_name [1..1]
- last_name [1..1]

Bảng CustomerTbl				
STT	Tên cột	Kiểu dữ liệu	Ràng buộc	Ghi chú
1	id	INT	AUTO_INCREMENT	Mã khách hàng
2	first_name	VARCHAR (255)	NOT NULL	Tên
3	last_name	VARCHAR(255)	NOT NULL	Họ và Đệm
Khóa chính		id		
Khóa ngoại		Không có		

# Mẫu thiết kế DAO



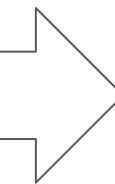
# Nền tảng Hibernate



[<https://docs.redhat.com/>]

# Nội dung

- Tổng quan
- Các quy tắc ánh xạ
- Tối ưu hóa CSDL quan hệ
- Tầng truy cập và quản lý dữ liệu
- CSDL phân tán

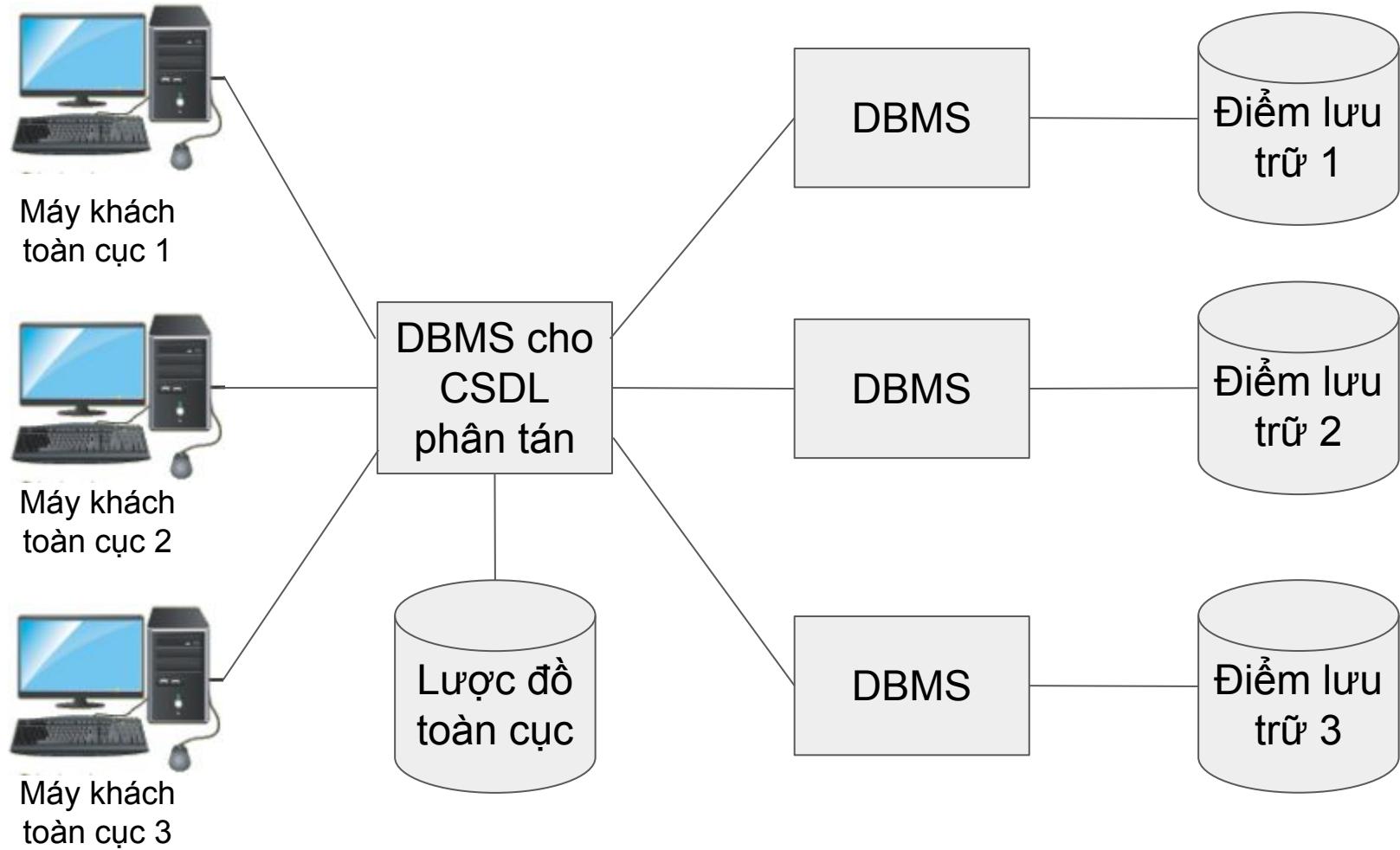


# Kiến trúc CSDL phân tán

- CSDL phân tán đồng nhất / Homogeneous distributed database được lưu ở nhiều nơi với cùng DBMS và 1 lược đồ toàn cục.
- CSDL phân tán không đồng nhất / Heterogeneous distributed database được lưu ở nhiều với các DBMS khác nhau và có thể có lược đồ cục bộ.

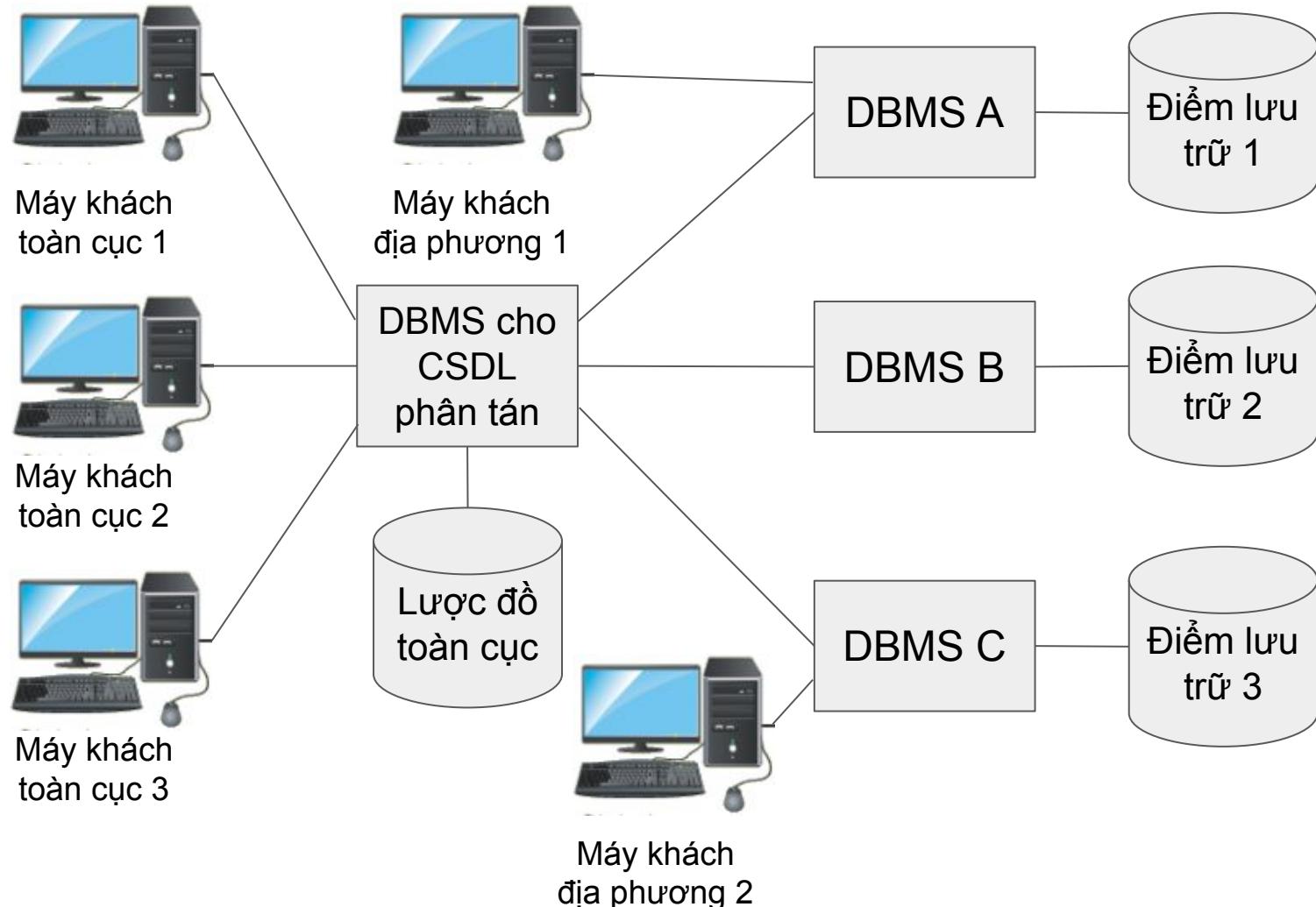
# CSDL phân tán đồng nhất

Truy cập thông qua 1 Hệ quản trị CSDL và lược đồ chung



# CSDL phân tán không đồng nhất

Truy cập thông qua các DBMS riêng. Có thể có DBMS và lược đồ chung



# Các cách triển khai

- Lắp dữ liệu - Mỗi điểm lưu trữ có 1 bản sao
  - Đồng bộ: Cập nhật tất cả các bản ghi khi thay đổi 1 bản ghi bất kỳ.
- Phân mảng theo chiều ngang - Mỗi điểm chứa 1 nhóm dòng

AcctNumb	LastName	FirstName	SSN	TypeOfAcct	Balance	DateLastActivity
01-85562-1	Jones	Bill	878-77-9890	Checking	\$ 7,908.39	5/9/2014
01-85444-2	Johnson	Harold	676-44-3433	Checking	\$25,698.33	5/2/2013
02-45443-2	Williams	Jonathon	343-44-2322	Checking	\$ 3,938.77	4/4/2012
01-34999-1	Redd	Mary	898-79-3487	Savings	\$12,898.71	12/2/2013
01-23989-2	Chun	Tun	233-59-6765	Savings	\$ 8,932.67	1/8/2014
01-87889-4	Gang	Bao	322-48-3545	Checking	\$ 568.33	3/4/2014
01-32339-2	Jiang	Rui	550-43-5454	Savings	\$35,788.23	7/8/2014
02-39988-1	Ma	Shuo	343-98-2345	Checking	\$ 1,893.55	8/23/2014

U.S.  
accounts

Hong Kong  
accounts

# Các cách triển khai<sub>(2)</sub>

- Phân mảng theo chiều dọc - Mỗi điểm lưu 1 nhóm cột

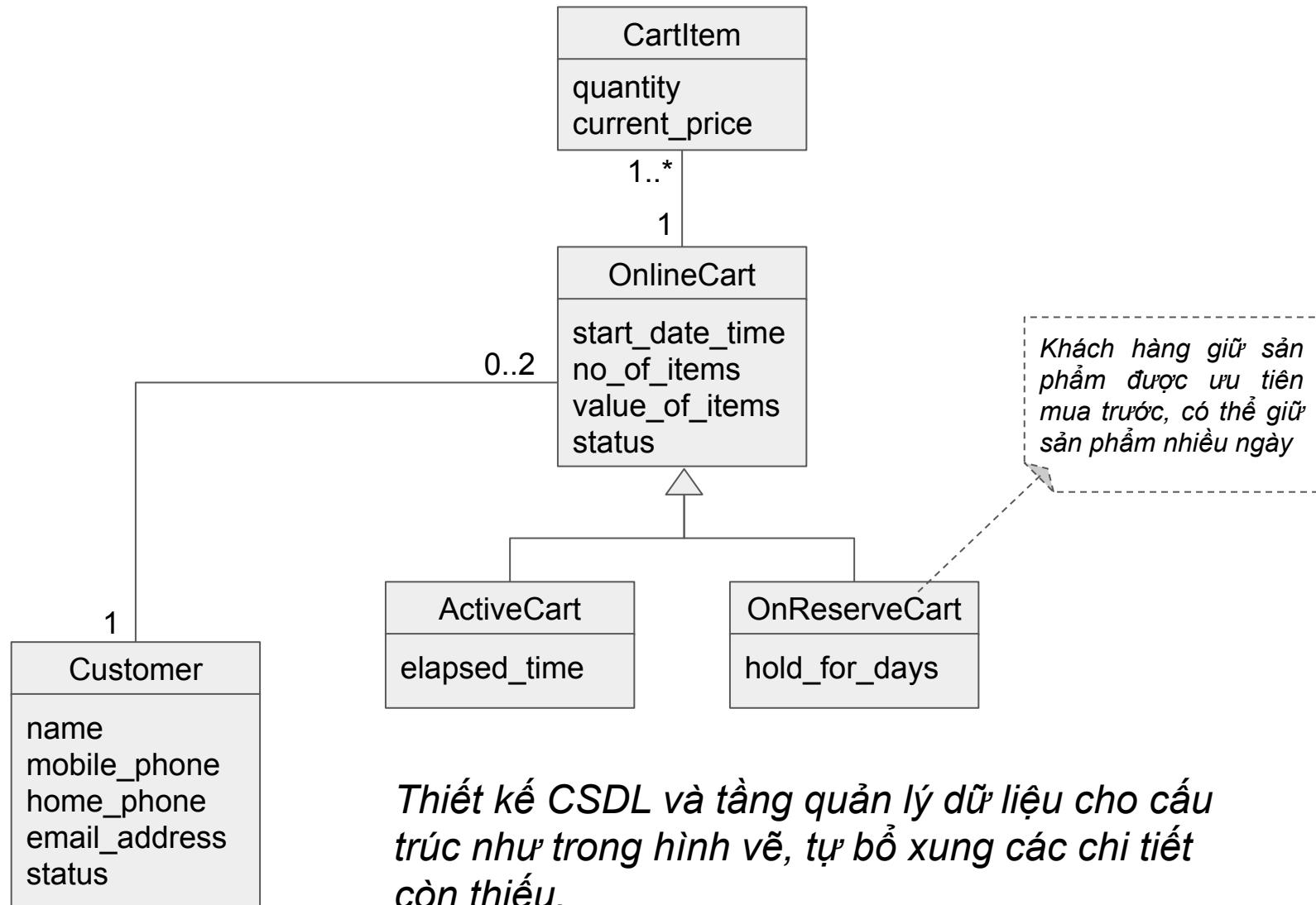
PartNumber	Description	Manufacturer	QtyOnHand	SchematicNo	InspectionNo	QtyOnHand2
4568-AC9	Screw assembly	Westco Inc	348	42-596	56	346
7618-IF44	Handle assembly	Japan Tools	276	16-443	43	434
7678-AD22	Door1 assembly	Tokyo Hardware	58	76-454	65	765
4890-XX88	Door2 assembly	Tokyo Hardware	97	78-443	34	446
9890-CD87	Interior module	Open Electronics	454	23-794	67	454
6766-DY65	Interior seal assembly	Sealants Inc	611	56-545	23	2132
8769-DD77	Connection assembly	Open Electronics	546	90-787	22	722
2311-AB28	Crank assembly	Westco Inc	768	33-571	12	121
3432-RB88	Double pulley assembly	Westco Inc	564	90-443	43	342

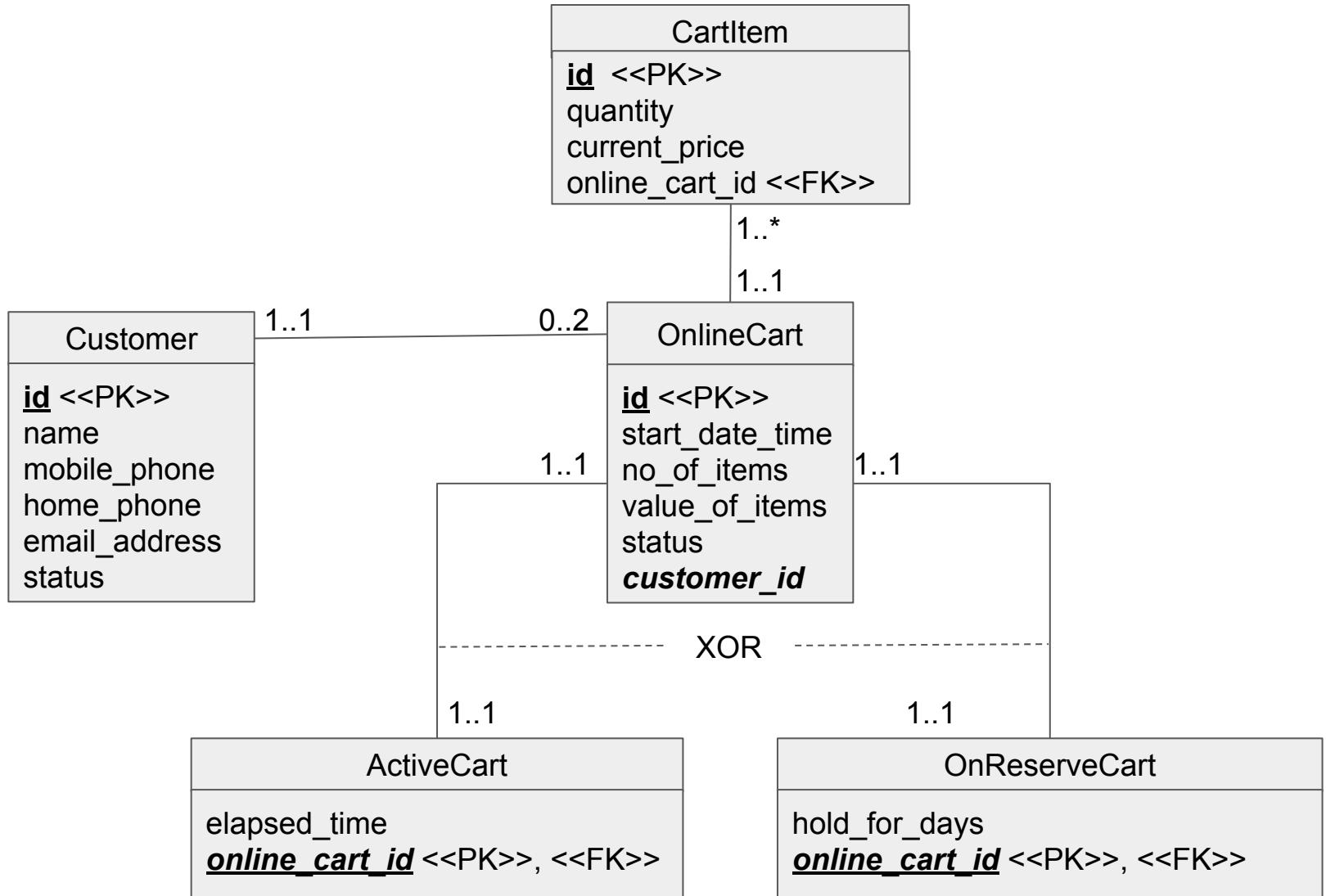
- Kết hợp lắp, phân mảng theo chiều ngang và phân mảng theo chiều dọc

# Bảo vệ CSDL

- Lịch sử giao dịch: Kỹ thuật lưu lại tất cả các cập nhật
  - Giúp ngăn chặn gian lận
  - Khôi phục sau sự cố
- Kiểm soát xung đột và cập nhật
  - Giao dịch - 1 phần công việc bao gồm nhiều bước, trong đó hoặc tất cả các bước cùng hoàn thành hoặc không chấp nhận kết quả của bước nào.
  - Khóa CSDL - kỹ thuật kiểm soát truy cập (1 phần) CSDL.
  - Khóa dùng chung / khóa đọc - Cho phép nhiều người dùng cùng đọc dữ liệu.
  - Khóa loại trừ / khóa ghi - Chỉ 1 người dùng có thể truy cập phần CSDL bị khóa.

# Bài tập thiết kế tầng quản lý dữ liệu





- ...
- Đặc tả các bảng
- Tạo các lớp DAM hoặc DAO, ...

