

Phân tích và thiết kế Hệ thống

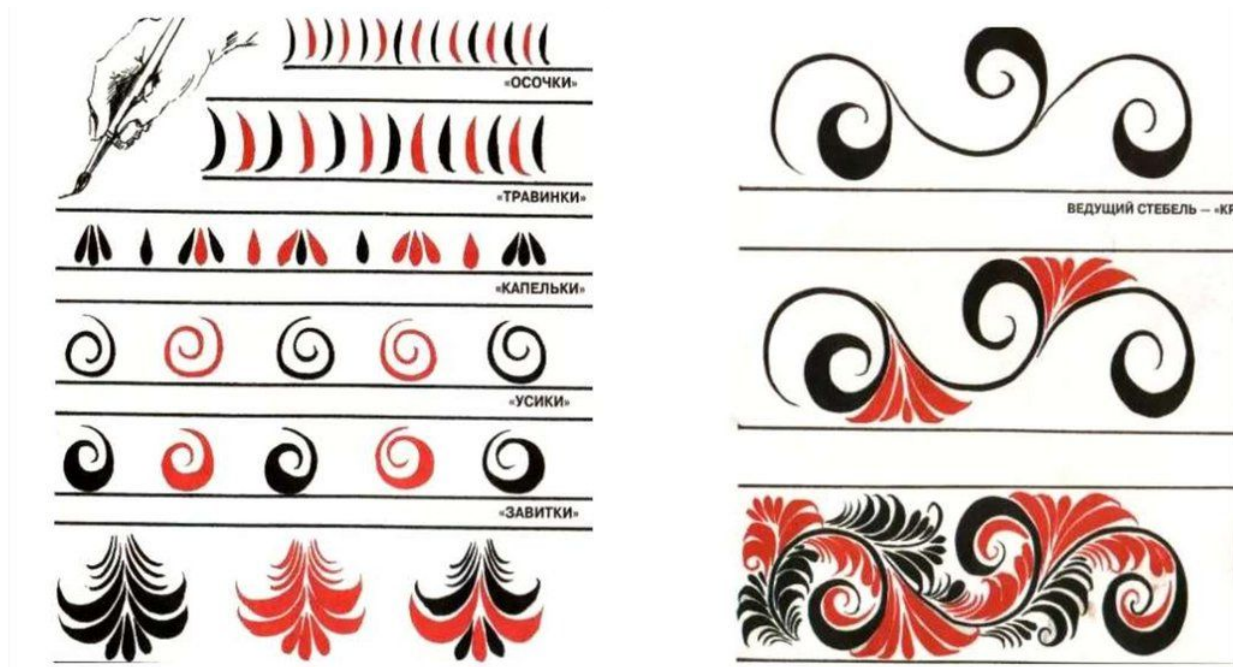
Giảng viên: Nguyễn Bá Ngọc

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.

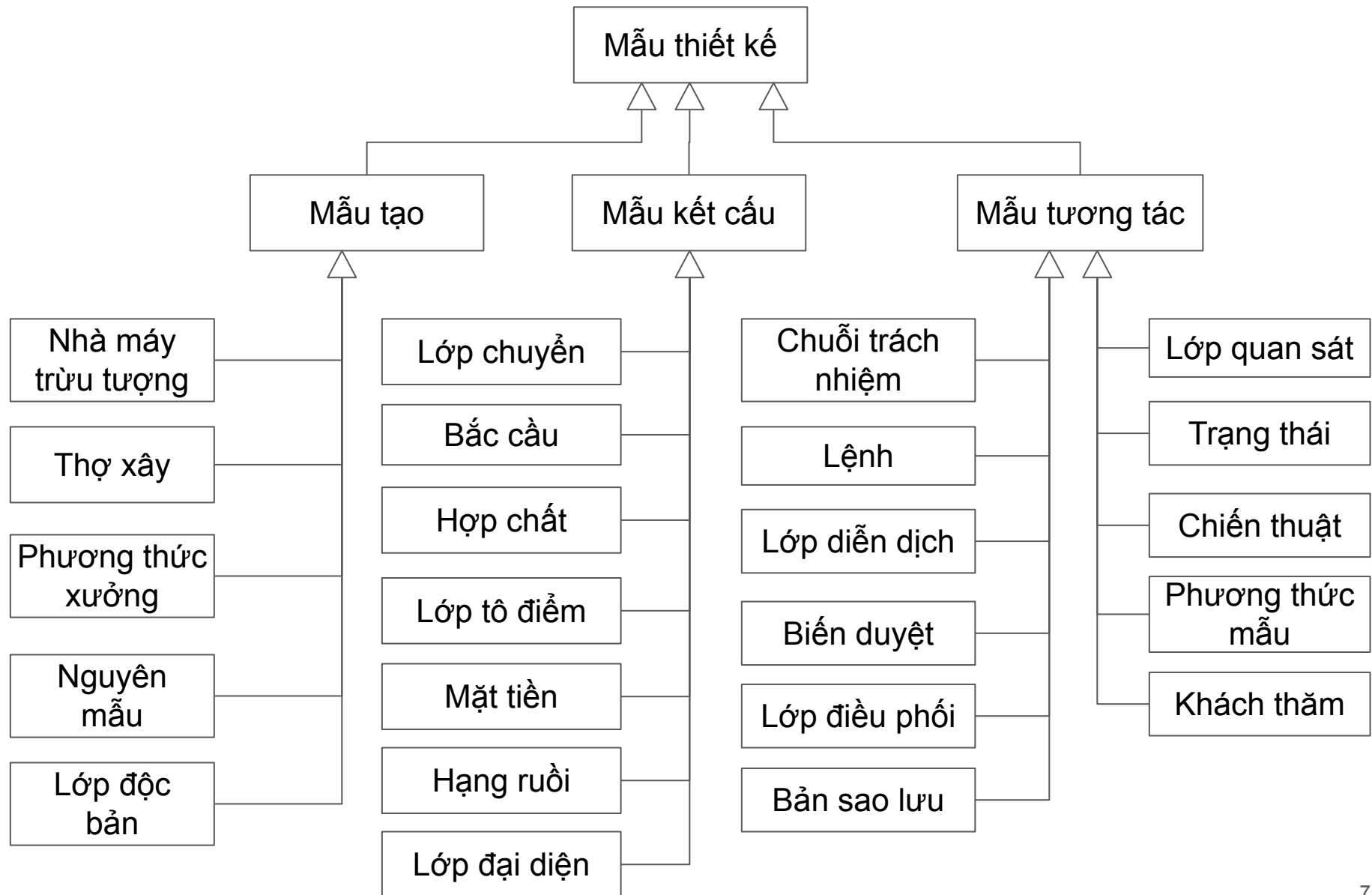
Các thành phần của mẫu thiết kế

- **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.

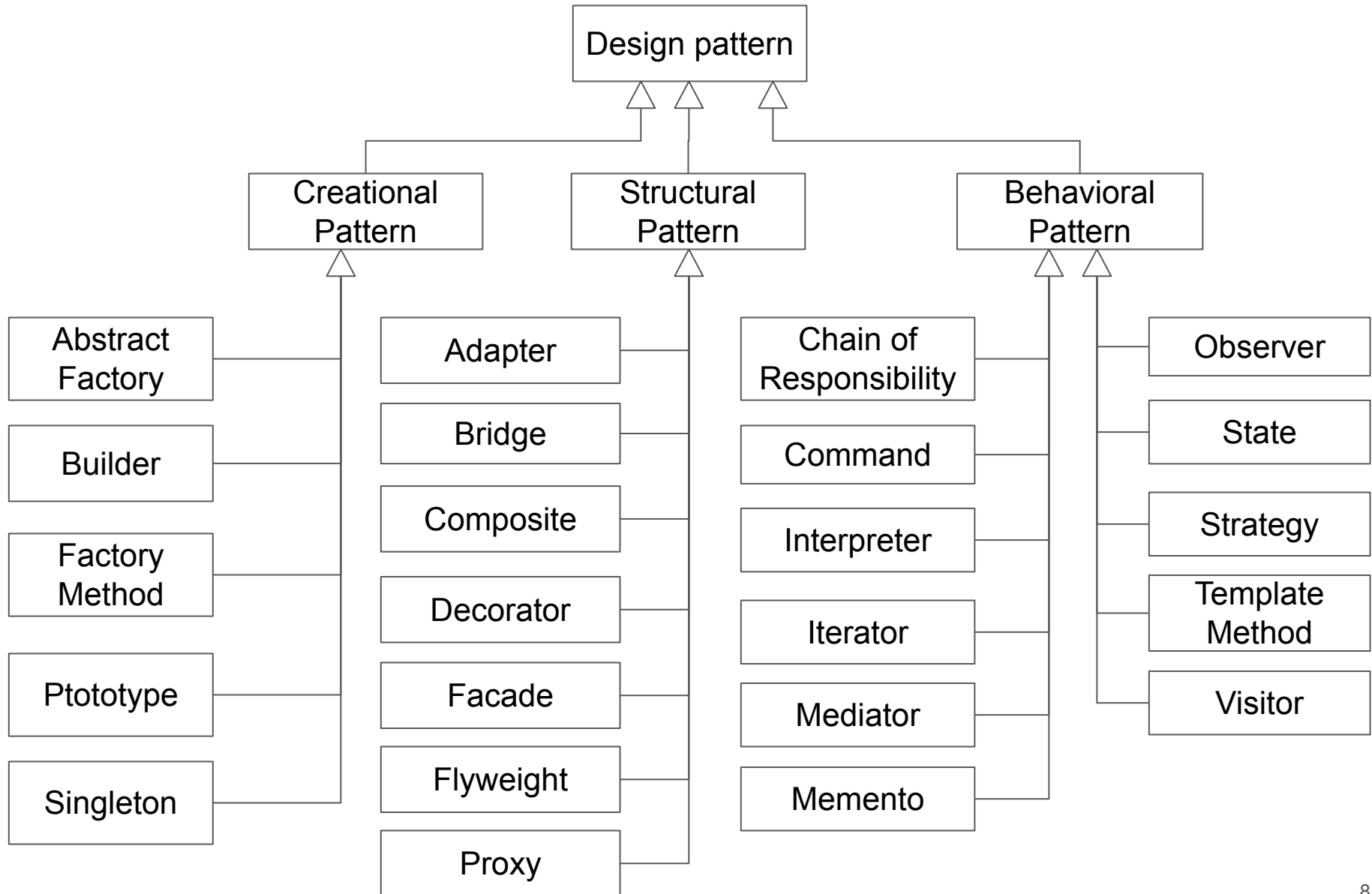
Phân loại mẫu thiết kế

- **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

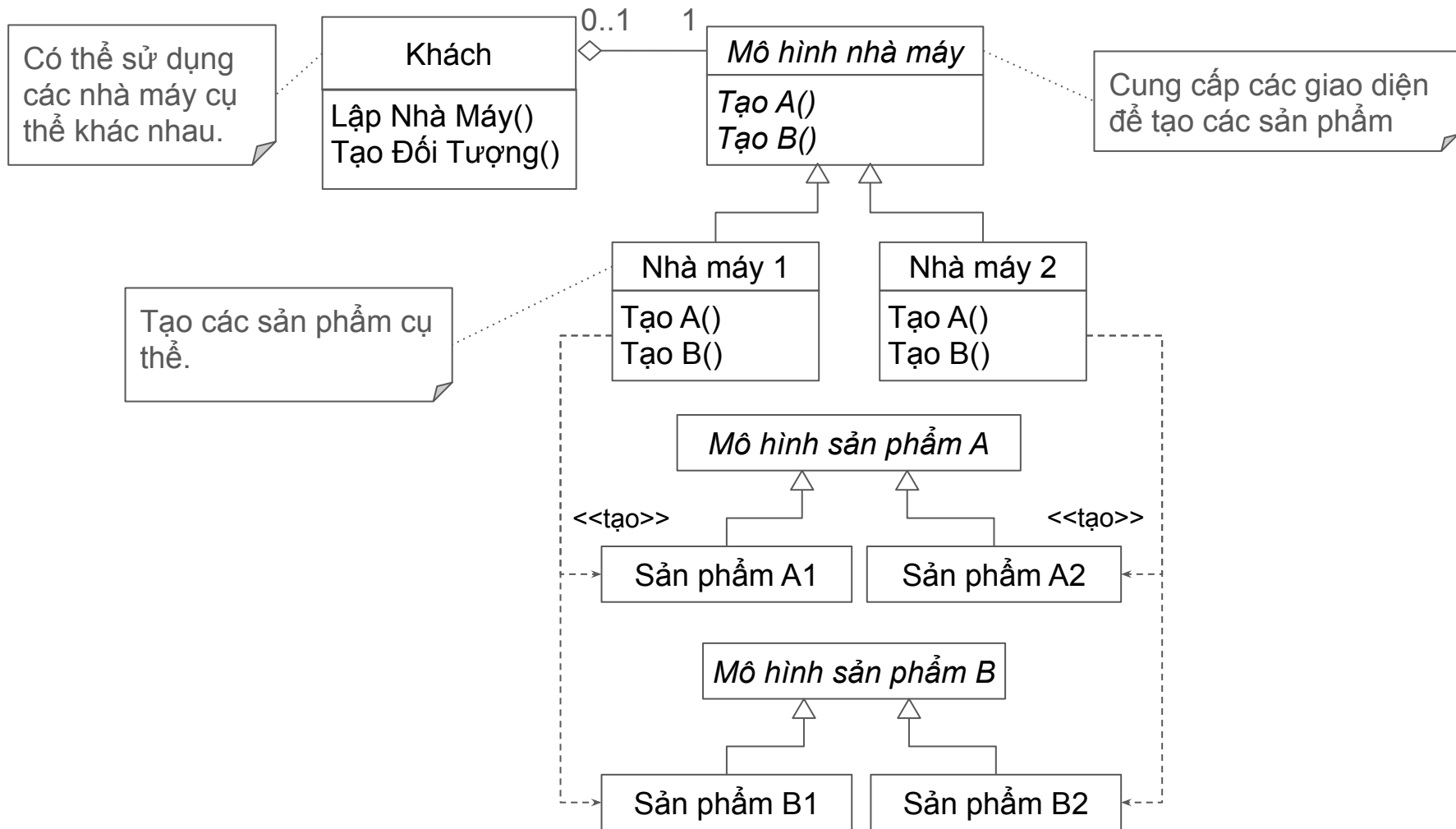
- Mẫu tạo
- Mẫu kết cấu
- Mẫu tương tác

Nhà máy trừu tượng

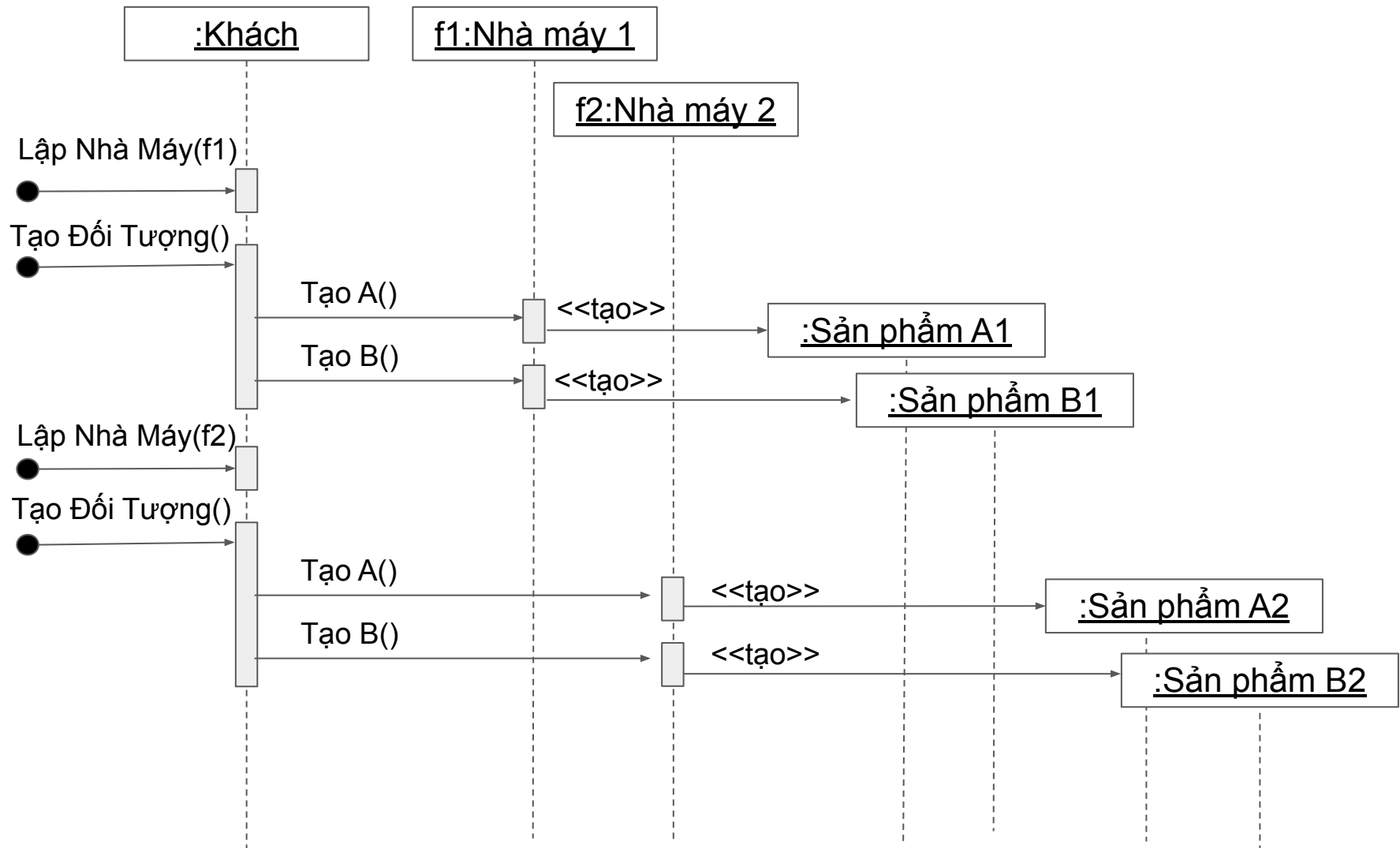
Abstract Factory

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



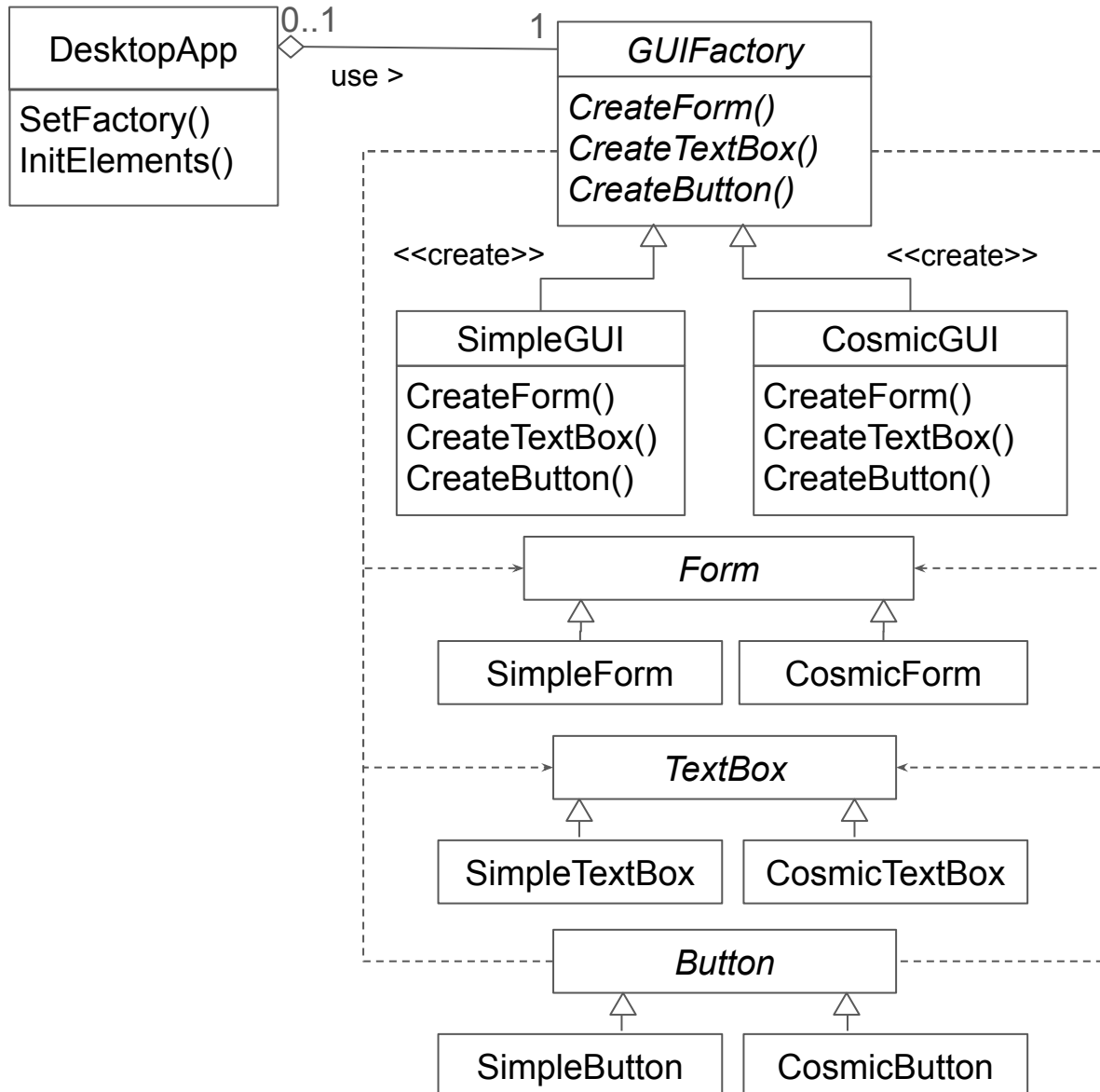
Nhà máy trừu tượng: Hành vi



Nhà máy trừu tượng: Hệ quả

- Cô 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

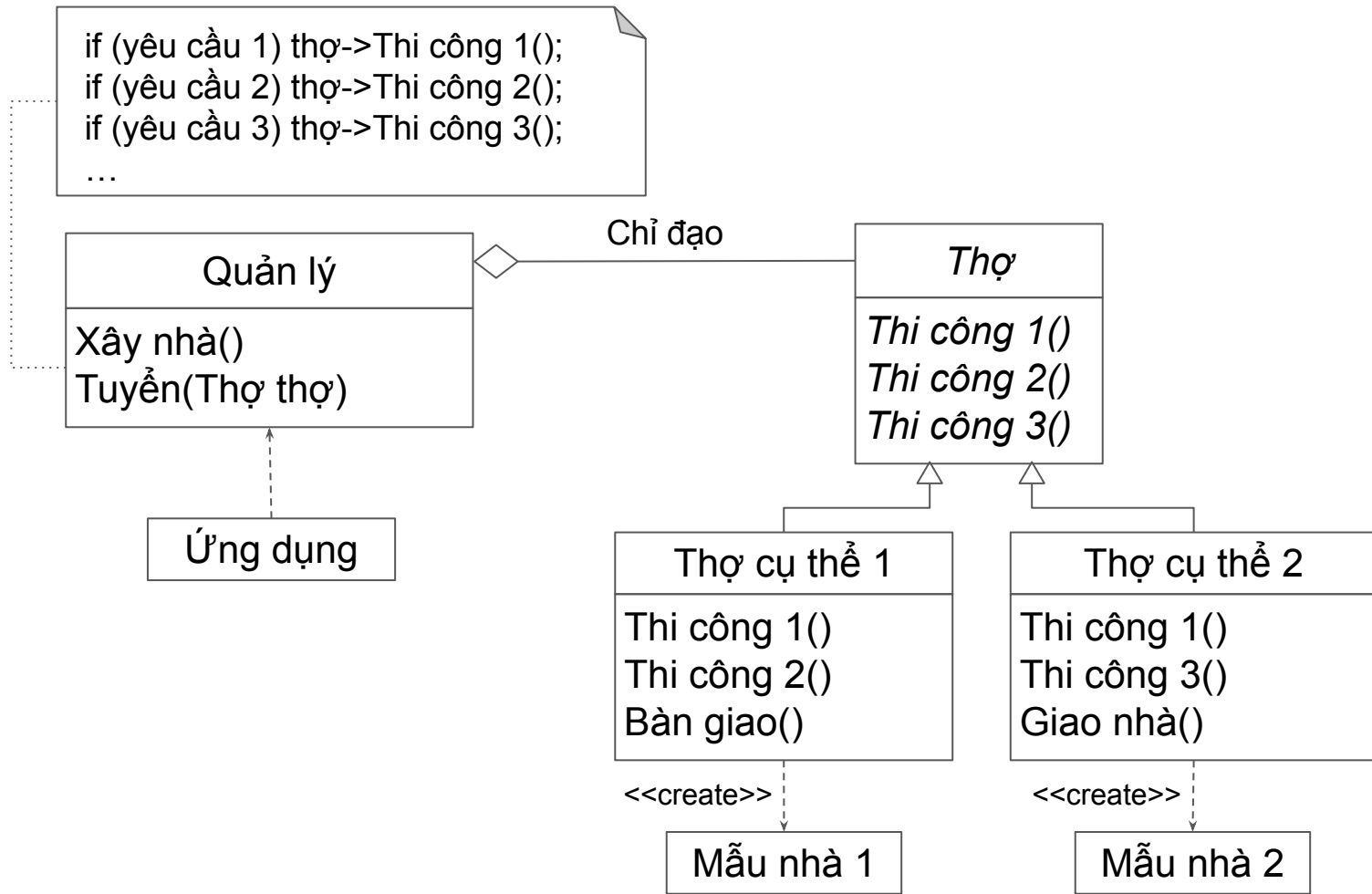


Thợ xây

Builder

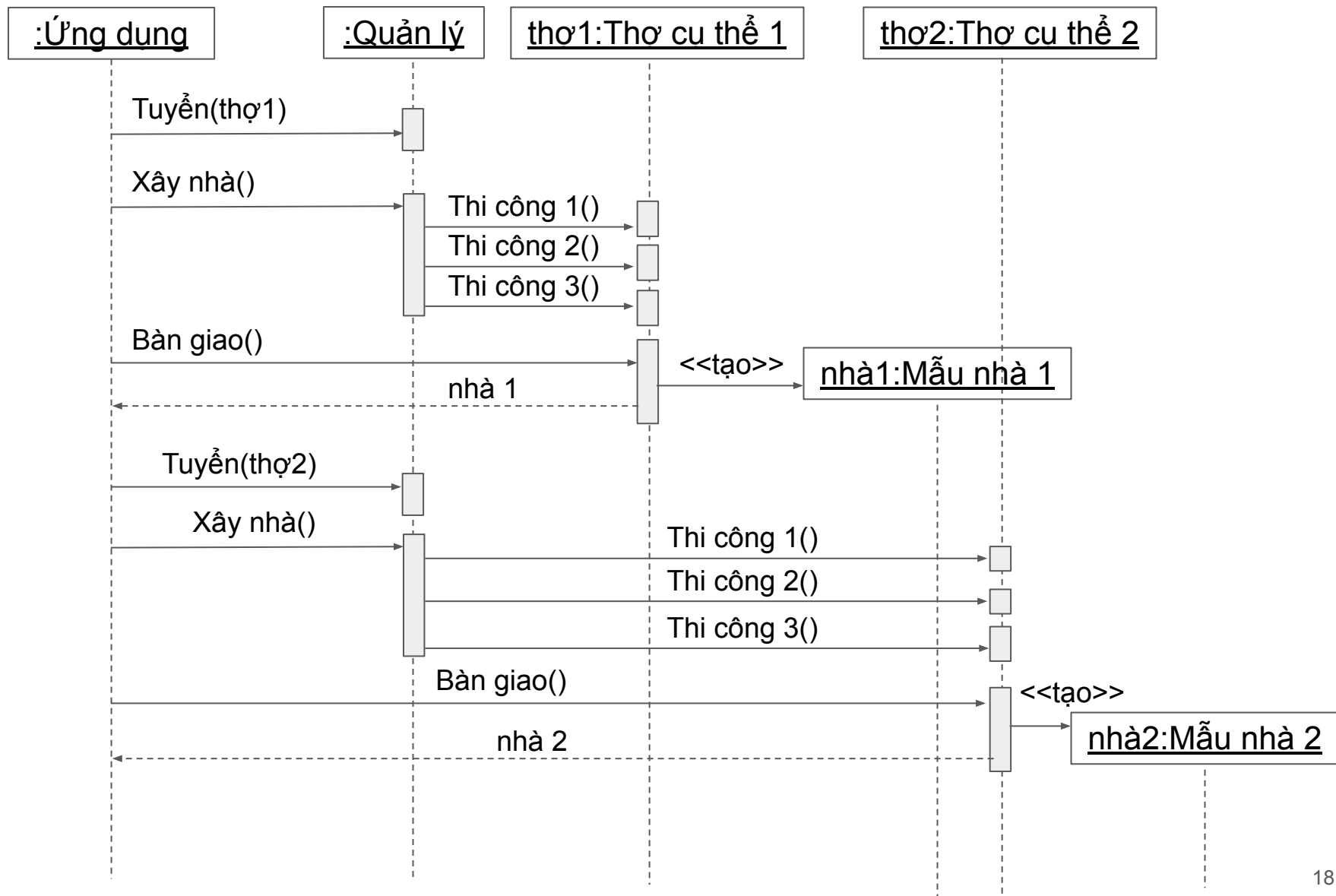
Tách riêng phần khởi tạo đối tượng phức tạp ra ngoài lớp, qua đó có thể sử dụng cùng một tiến trình để tạo nhiều biểu diễn khác nhau.

Thợ xây: Cấu trúc



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

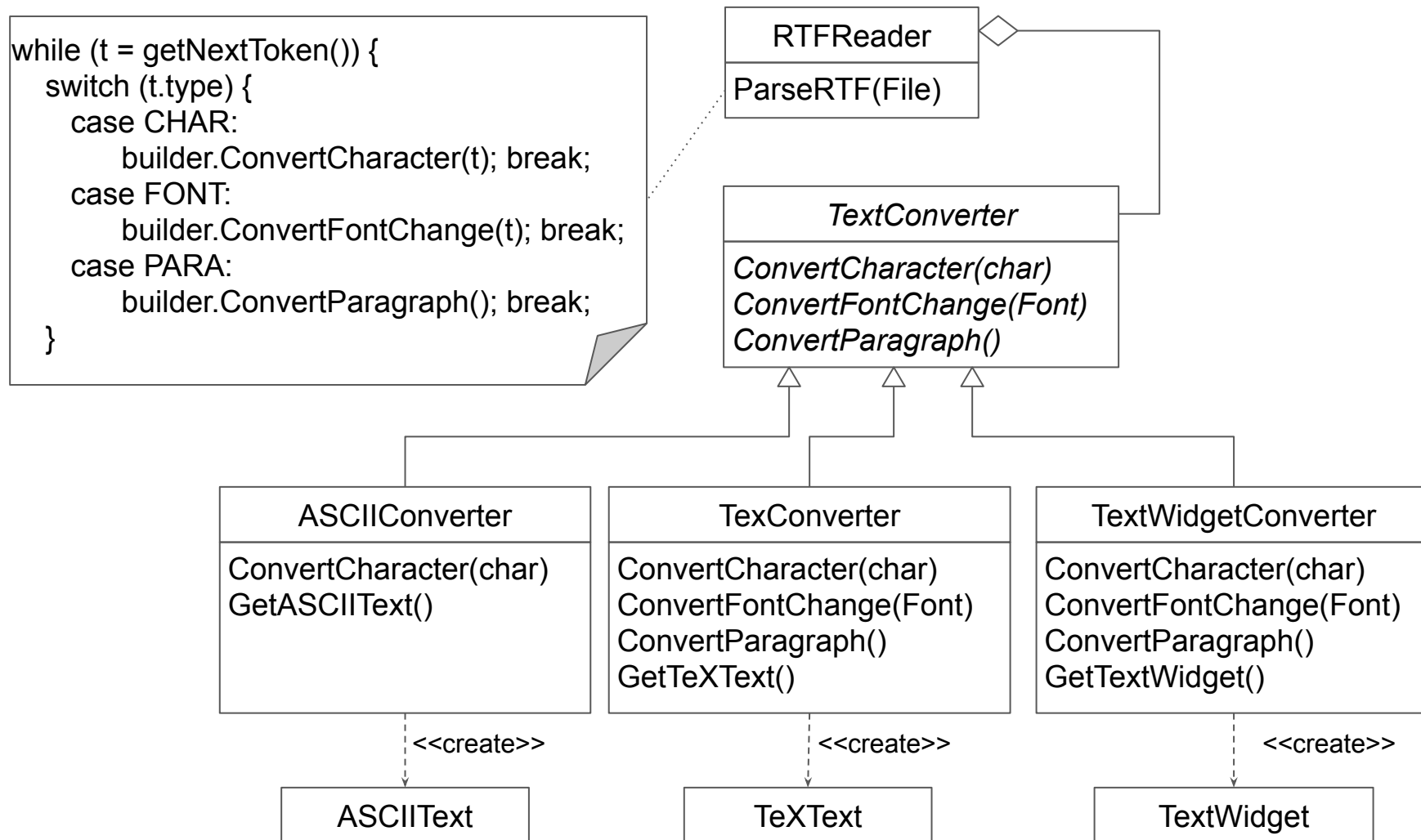
Thợ xây: Hành vi



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

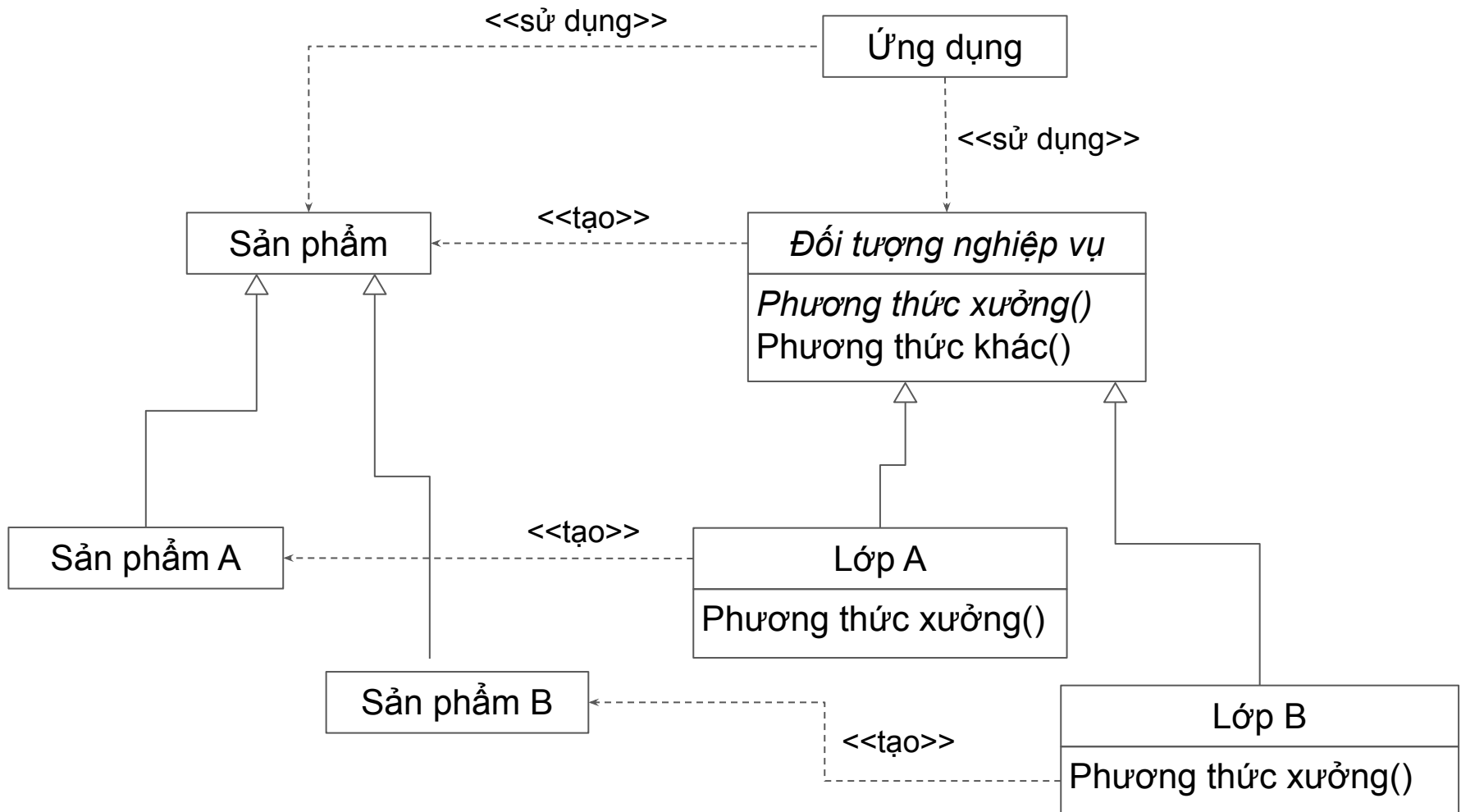


Phương thức xưởng

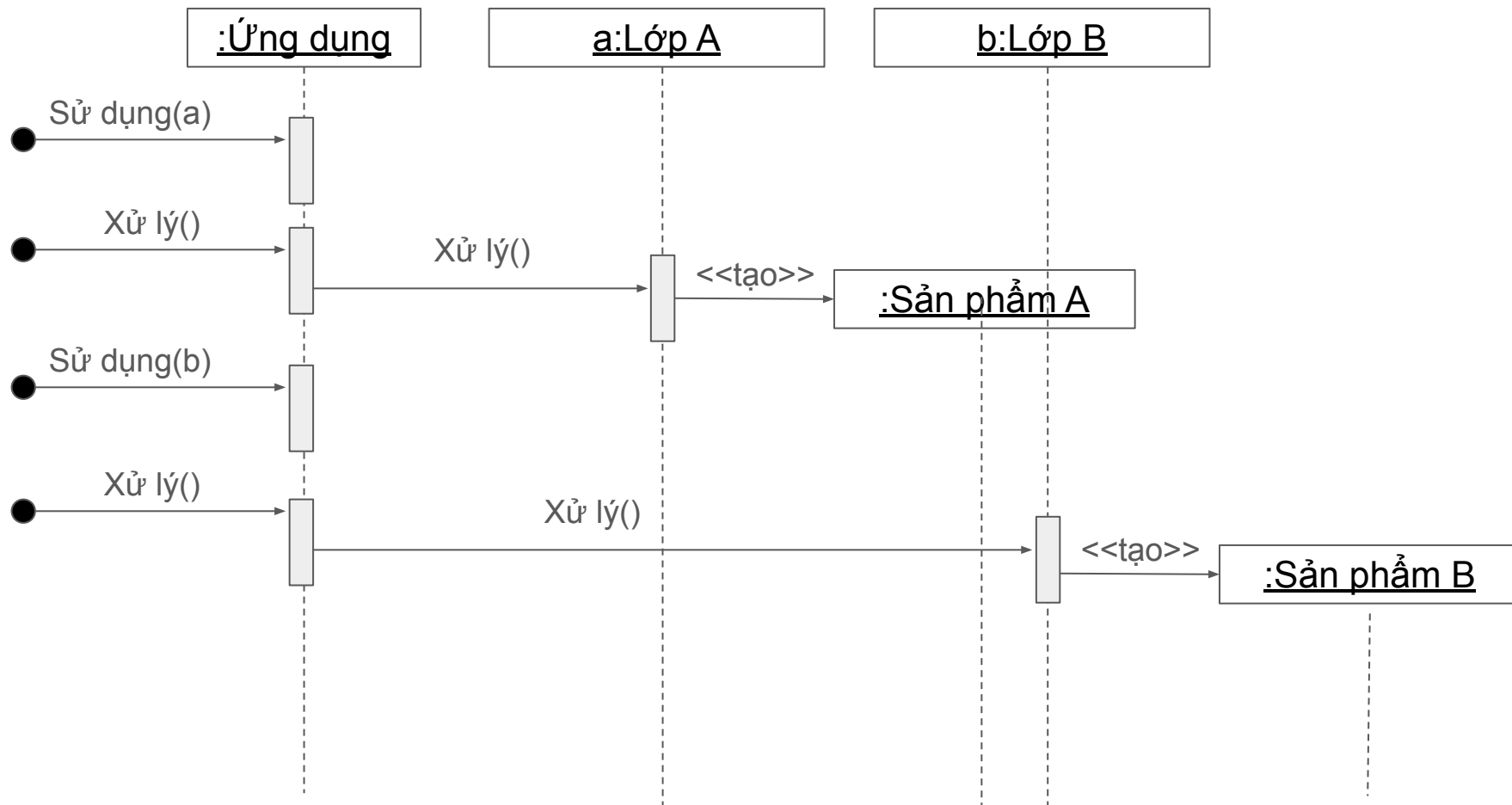
Factory Method

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 xướng: Cấu trúc



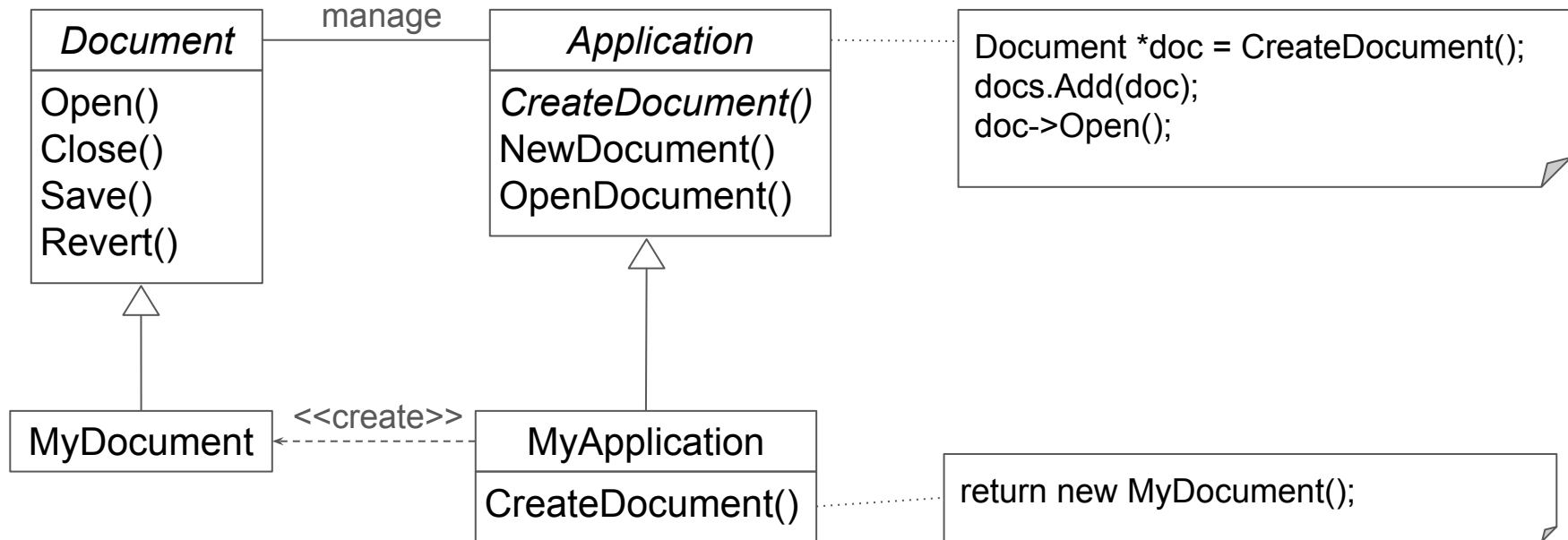
Phương thức xướng: Hành vi



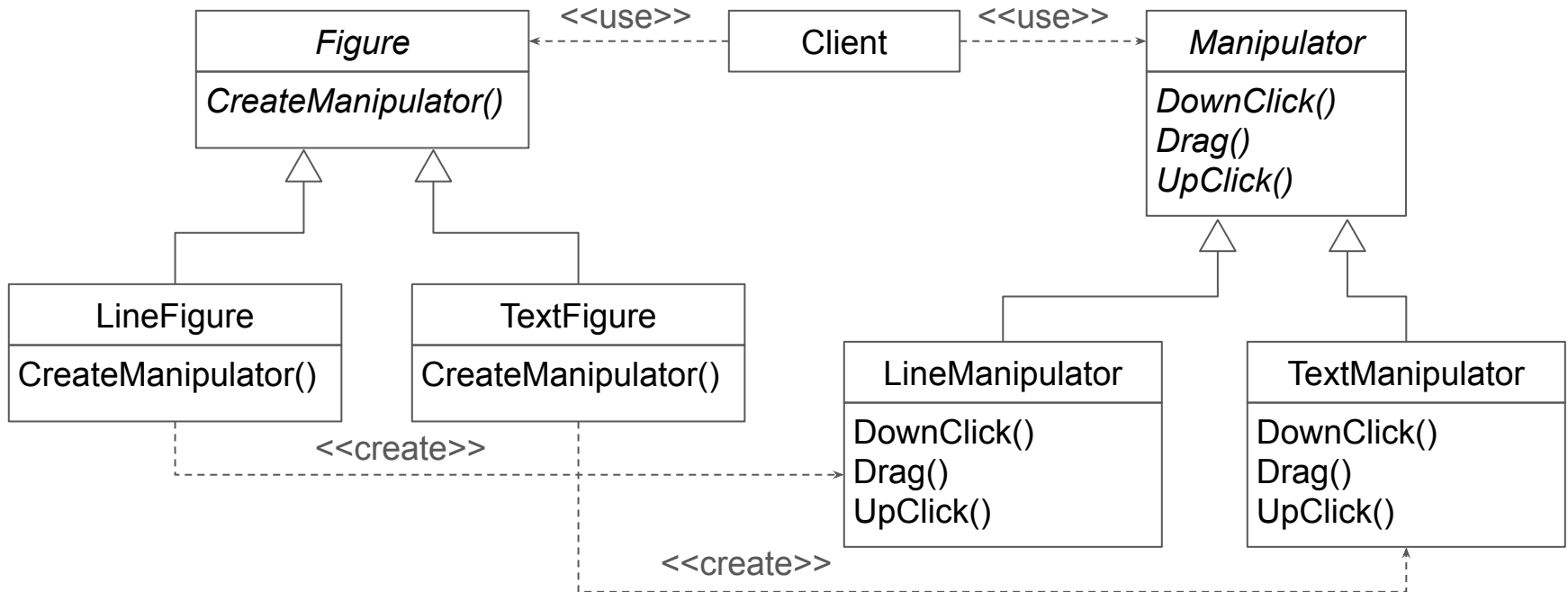
Phương thức xử lý: Hệ quả

- 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

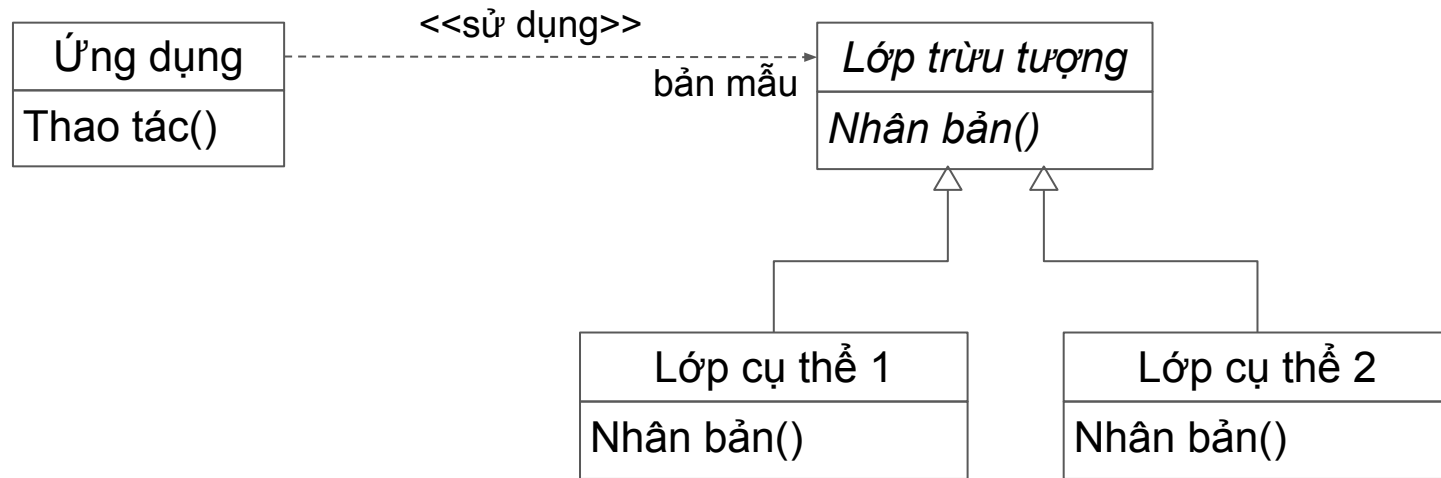


Nguyên mẫu

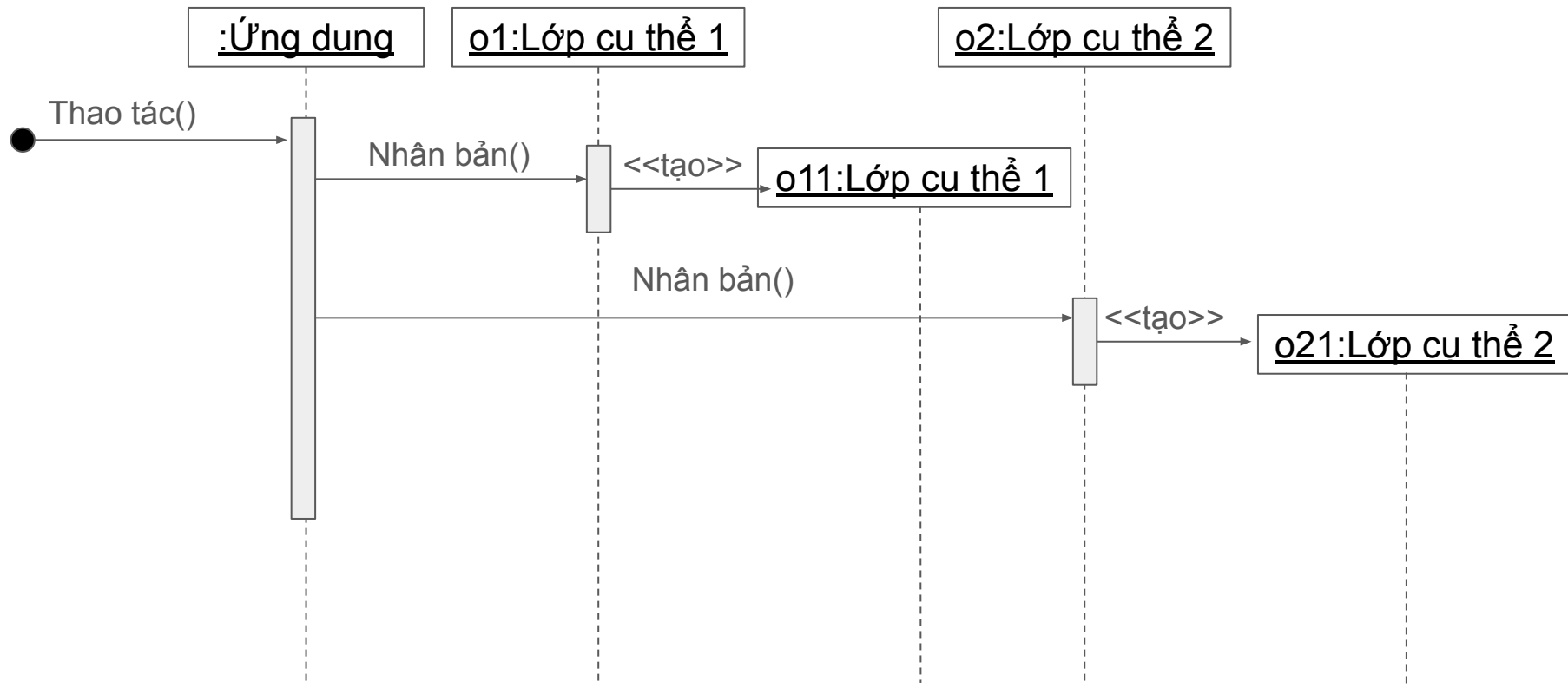
Prototype

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

Nguyên mẫu: Cấu trúc



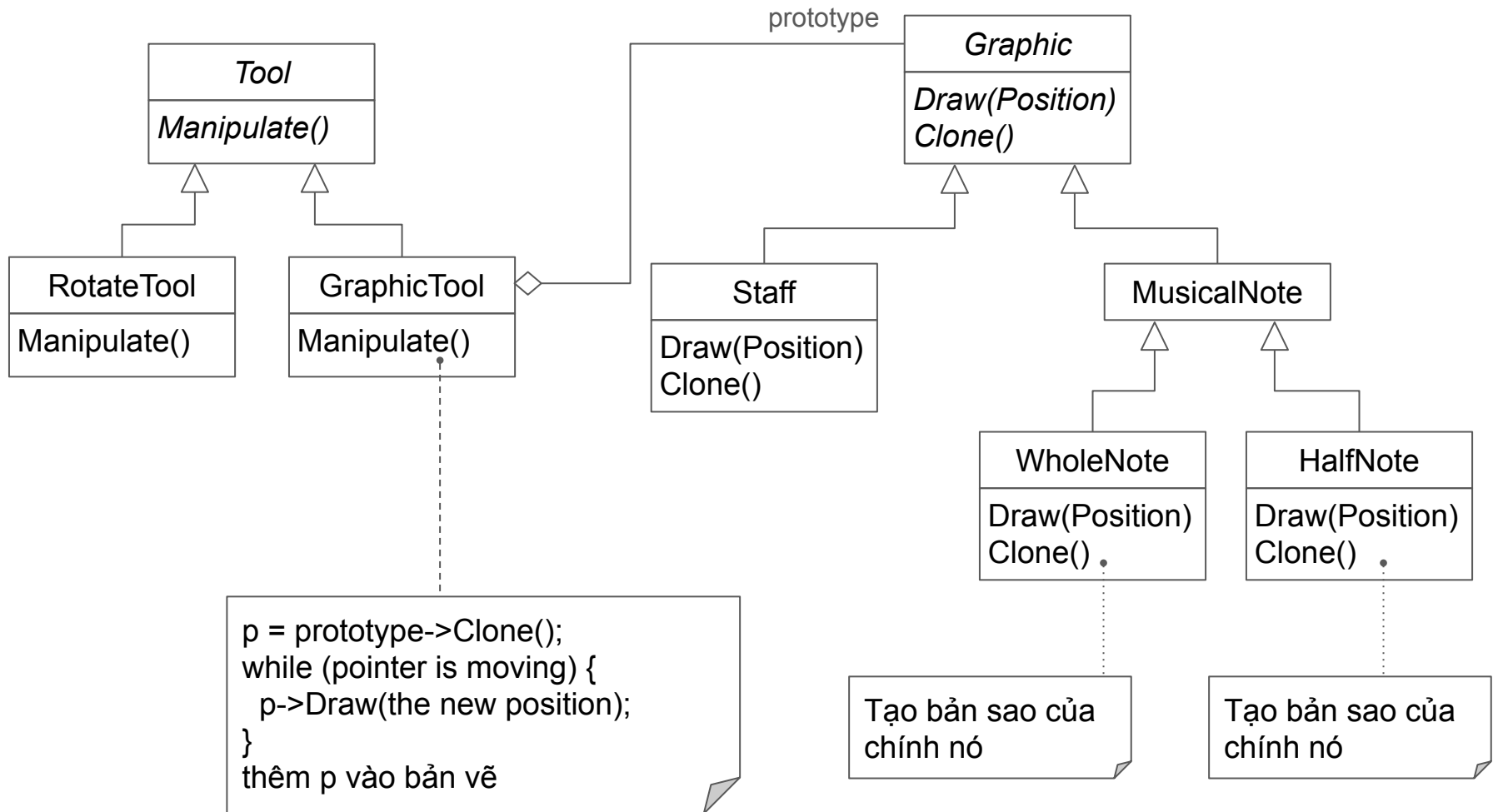
Nguyên mẫu: Hành vi



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 xướng 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

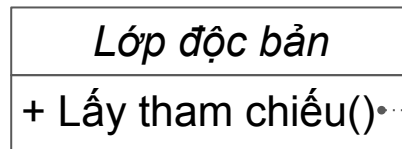


Lớp độc bản

Singleton

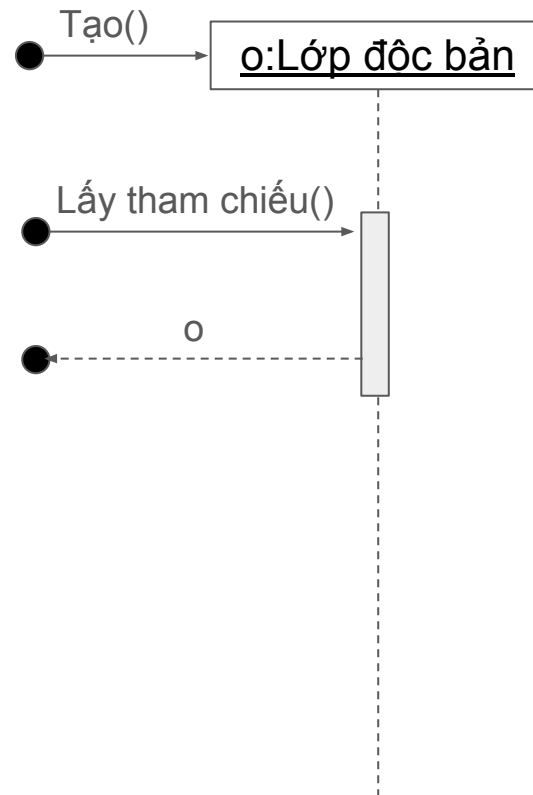
Đảm bảo một lớp chỉ có đúng một bản thực và cung cấp khả năng truy cập toàn cục tới bản thực đó.

Lớp độc bản: Cấu trúc



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

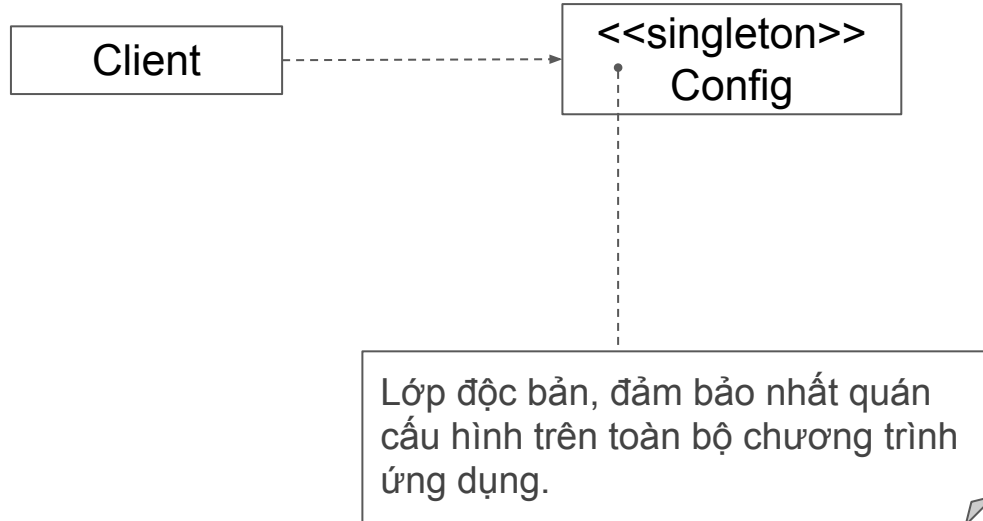
Lớp độc bản: Hành vi



Lớp độc bản: Các hệ quả

- Có thể thiết lập quyền truy cập tới bản thực duy nhất.
- Giảm 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.

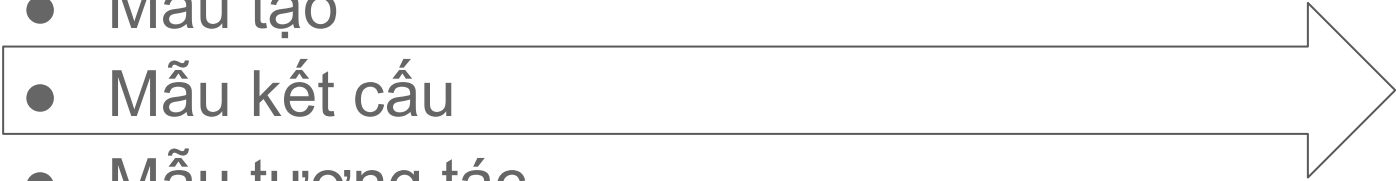
Ví dụ 6. Tùy chỉnh toàn cục



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 xướng thay đổi đối tượng được tạo thông qua kế thừa lớp tạo, nguyên lý đơn giản nhất.
- Nhà máy trùu 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
- 

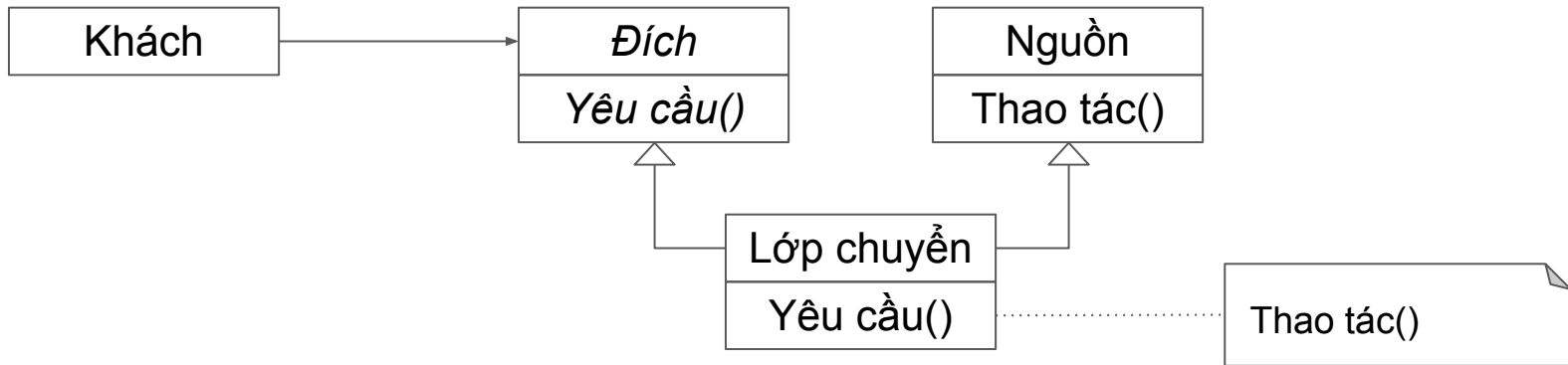
Lớp chuyển

Adapter

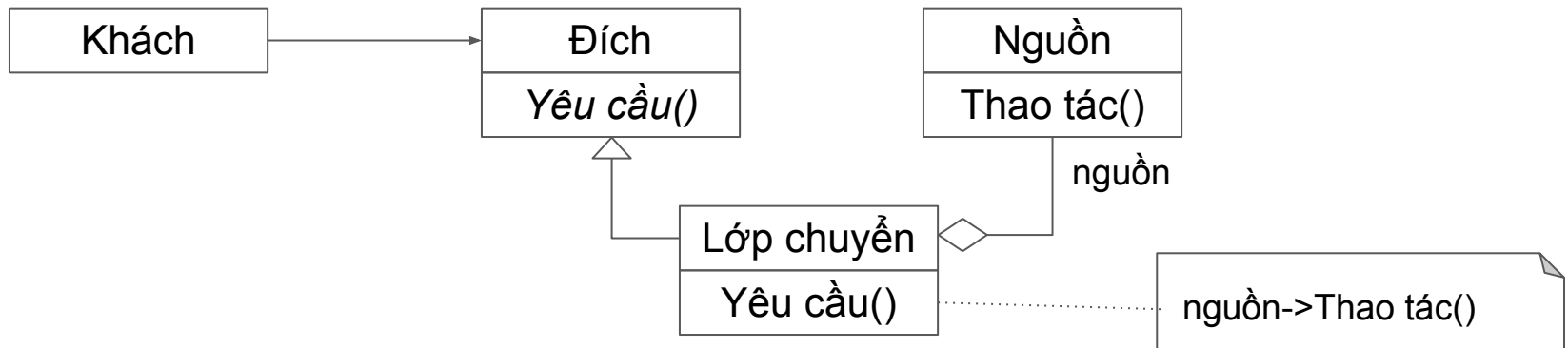
Chuyển đổi giao diện của một lớp đã có thành giao diện được yêu cầu từ phía khách, chuyển đổi một lớp với giao diện không tương thích thành một lớp với giao diện tương thích mà không thay đổi lớp đó.

Lớp chuyển: Cấu trúc

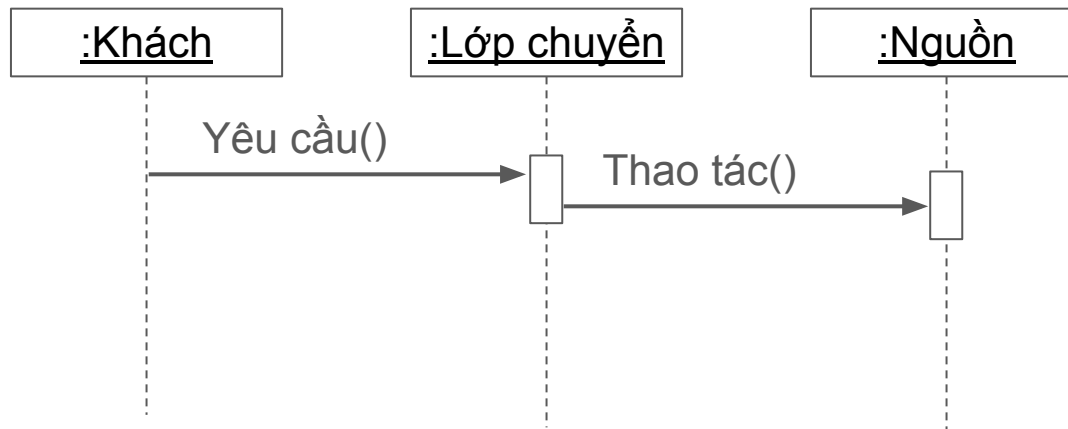
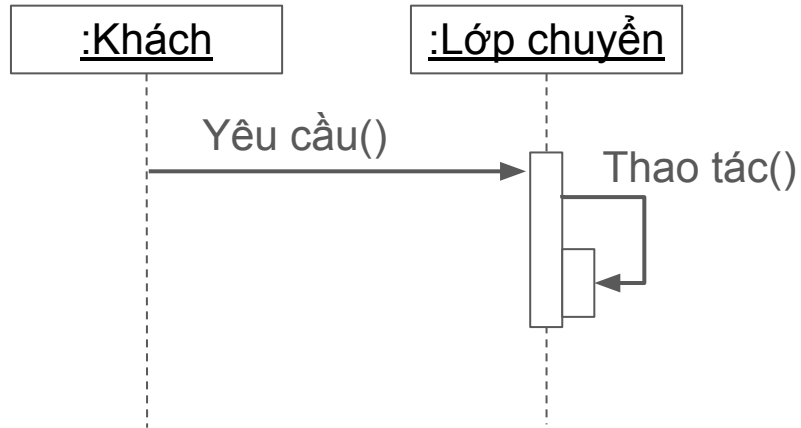
Chuyển đổi dựa trên đa kế thừa



Chuyển đổi bằng cách bọc đối tượng nguồn



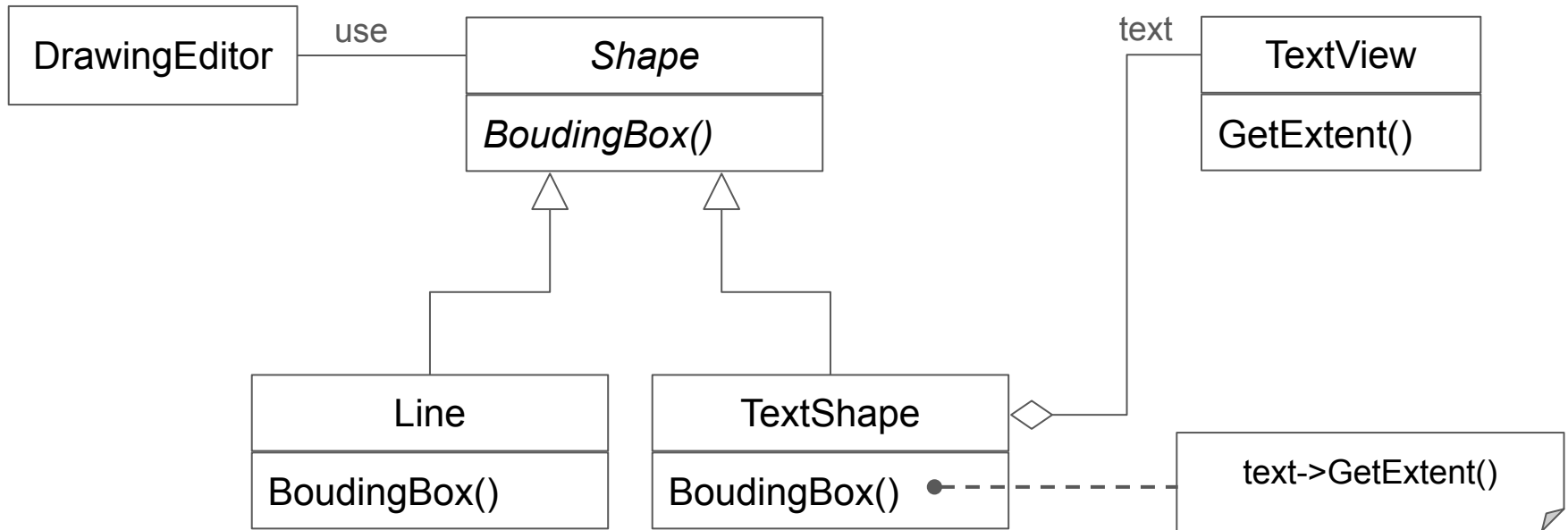
Lớp chuyển: Hành vi



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 vẽ

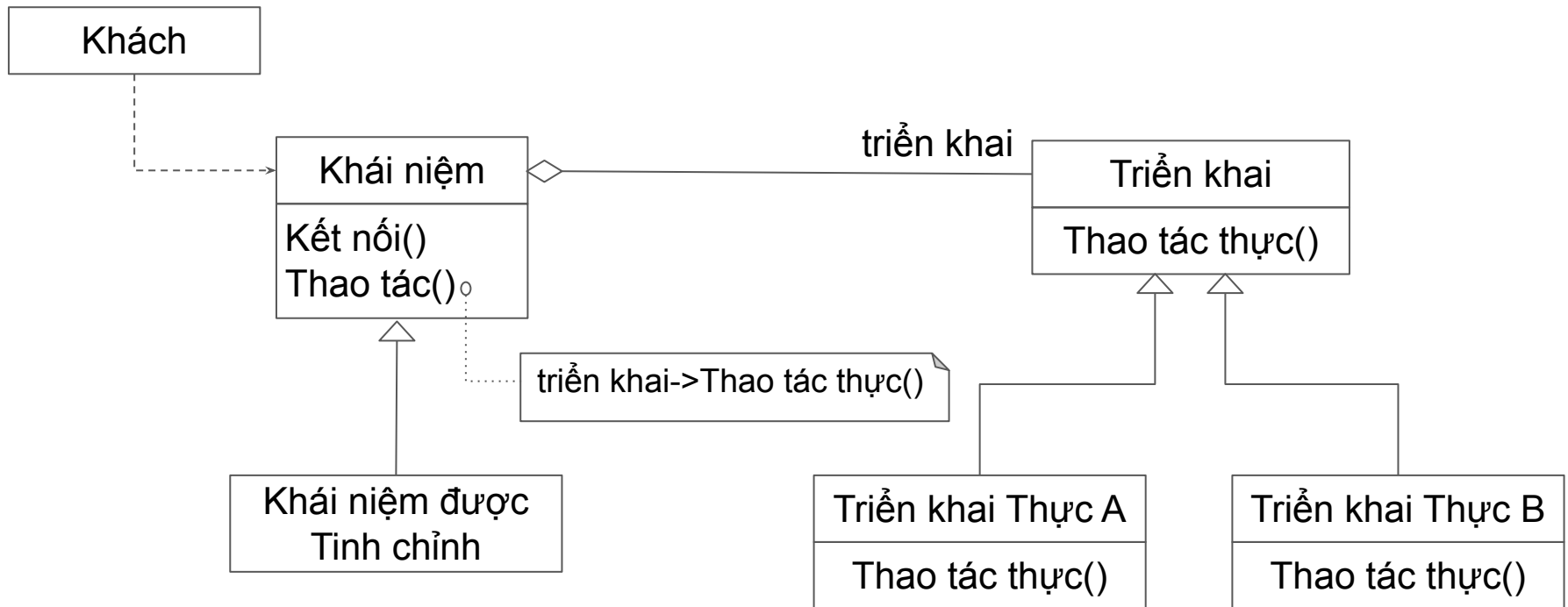


Bắc cầu

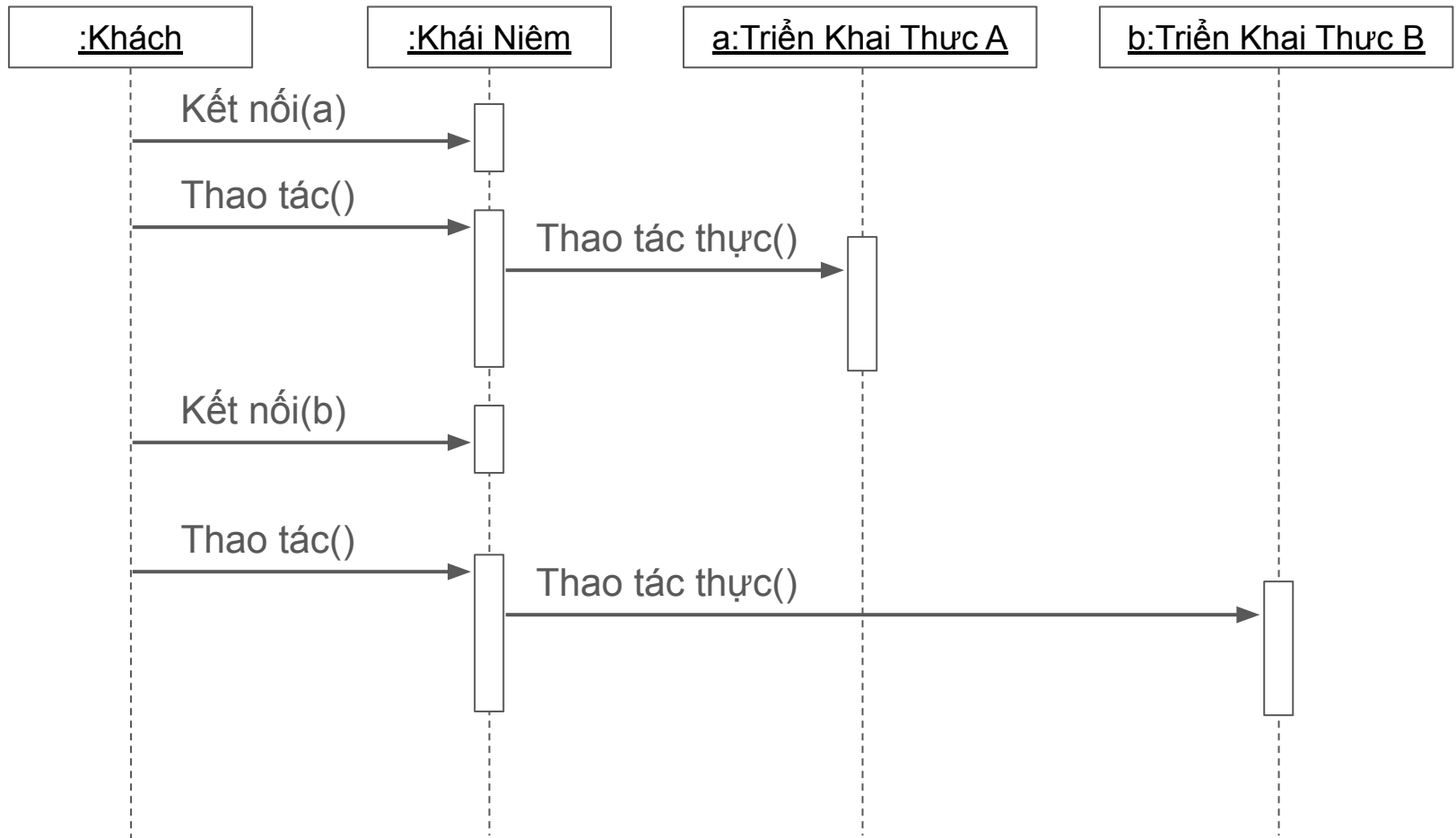
Bridge

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

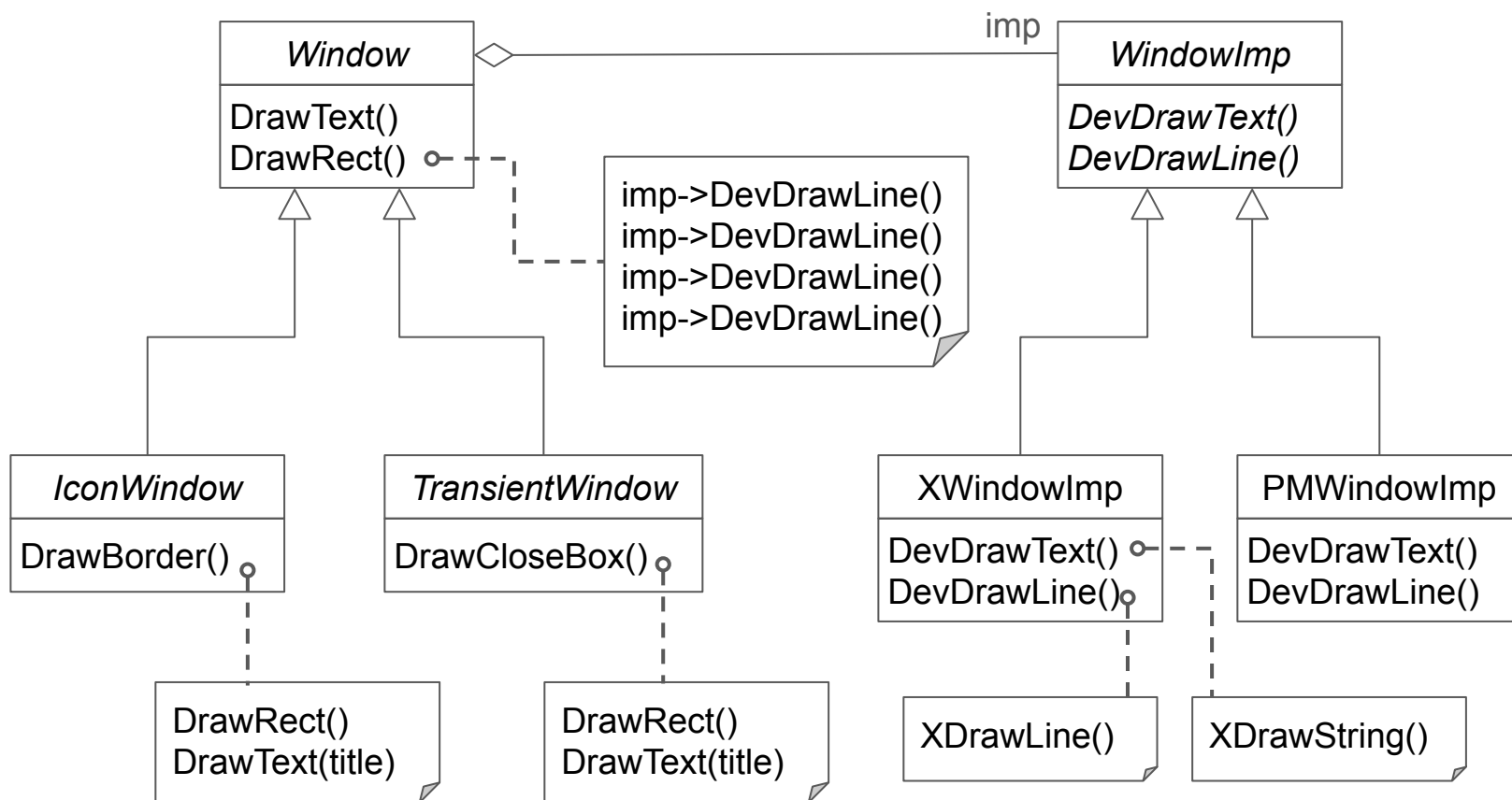
Bắc cầu: Cấu trúc



Bắc cầu: Hành vi



Ví dụ 8. Cửa sổ trong giao diện đồ họa



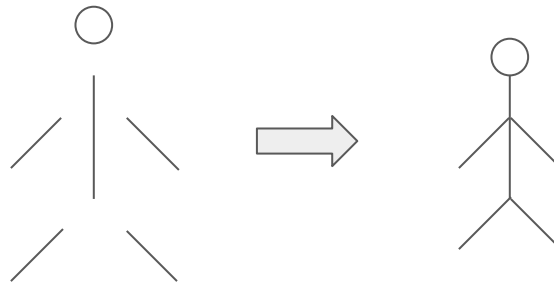
Bắc cầu: Các hệ quả

- 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.
- Ẩn các chi tiết triển khai với khách.

Hợp chất

Composite

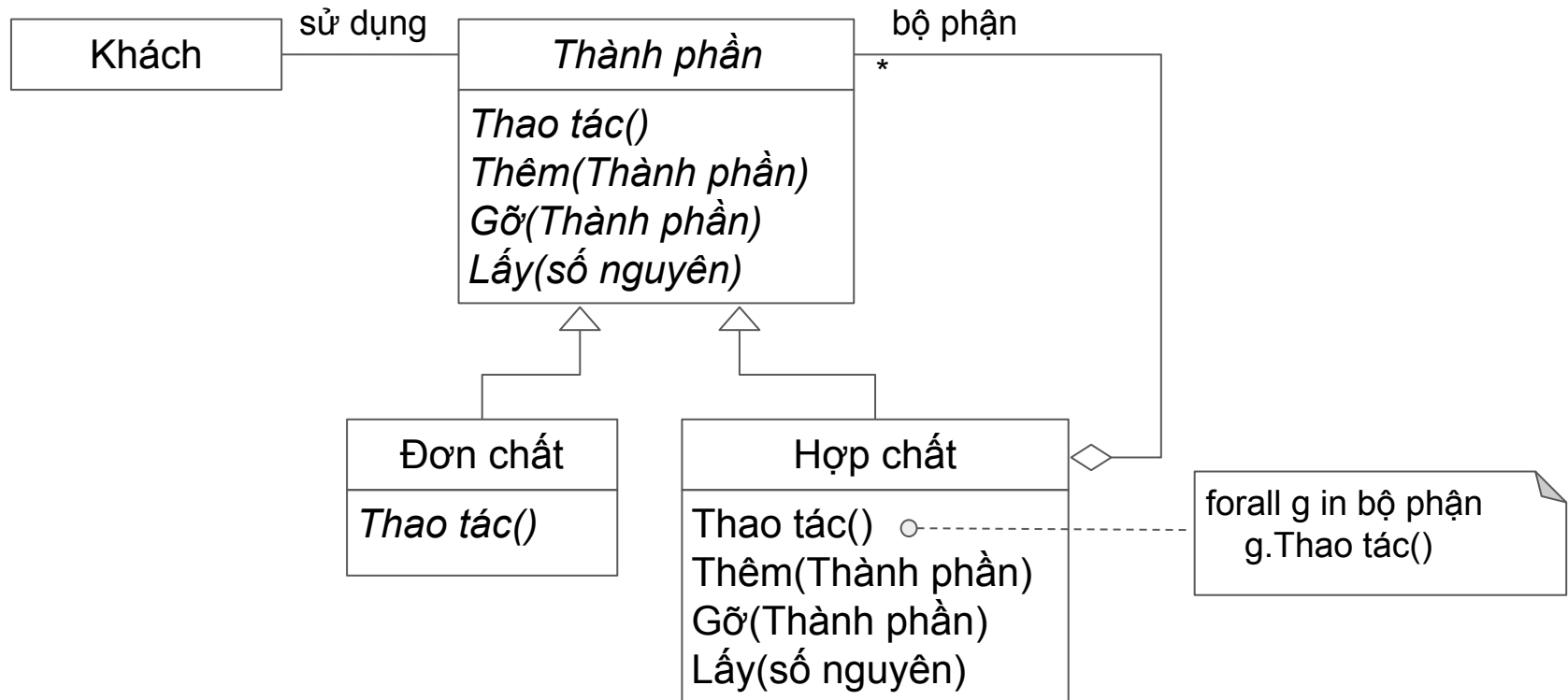
Kết hợp các đối tượng theo cấu trúc cây phân cấp với quan hệ bộ phận-tổng thể. Hợp chất cho phép khách tương tác với tổ hợp đối tượng như với một đối tượng.



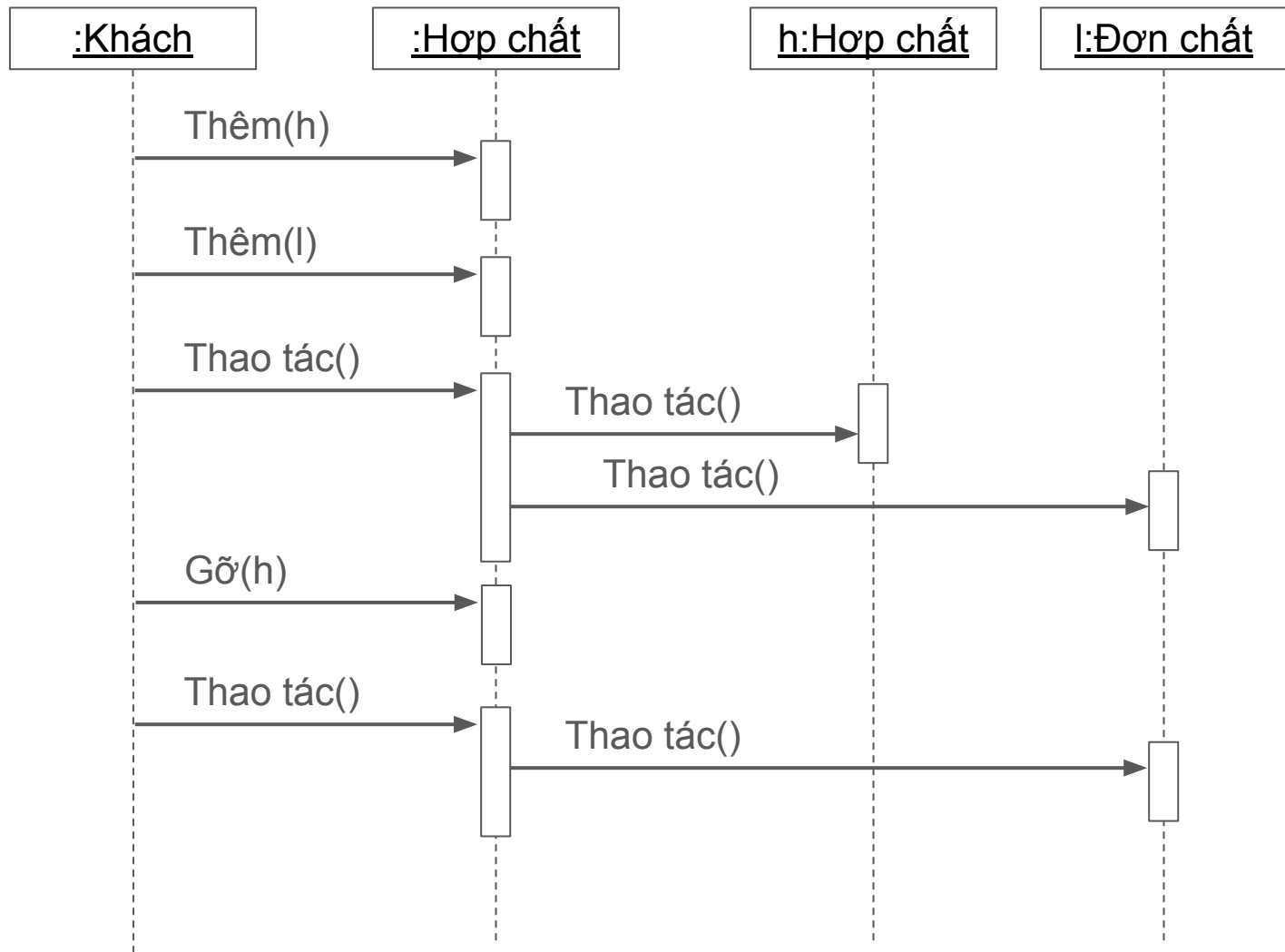
18 % Cr + 1 % Mn + 8 % Ni + 0.07 C + Fe => Inox 304

...

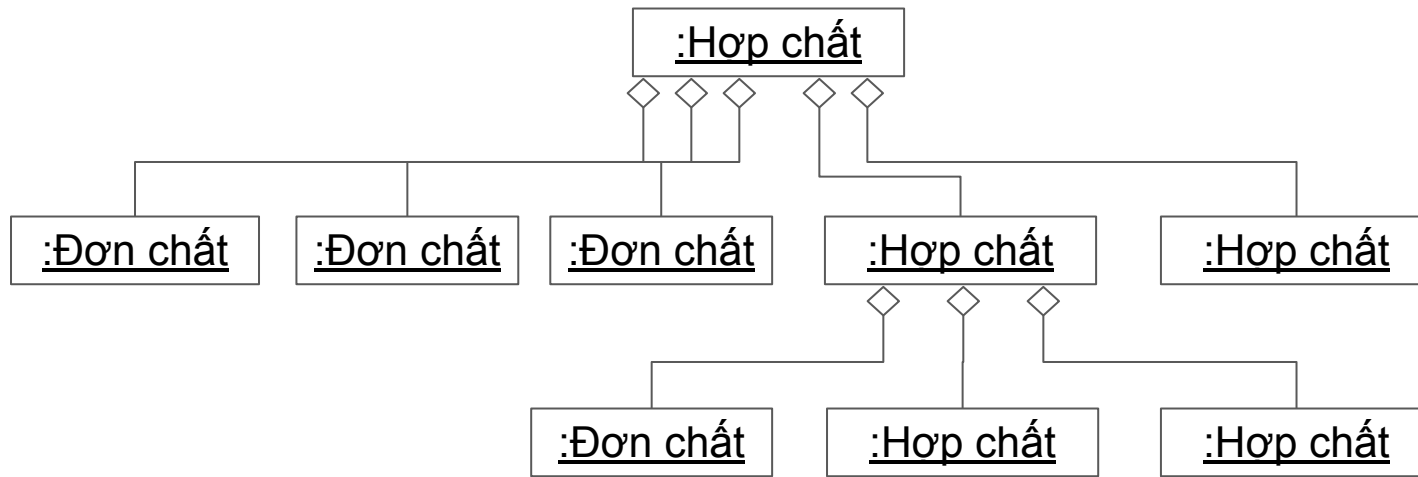
Hợp chất: Cấu trúc



Hợp chất: Hành vi



Hợp chất: Kết hợp đối tượng

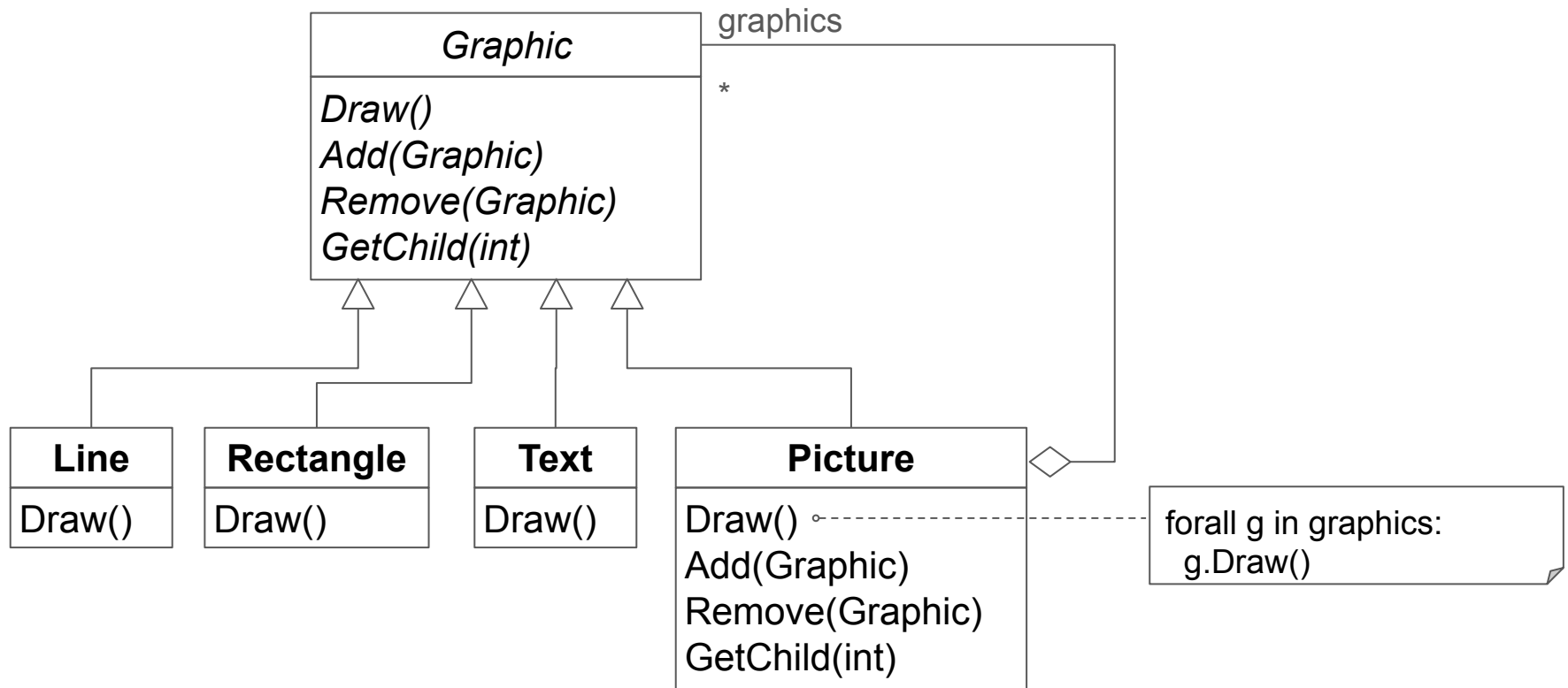


...

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

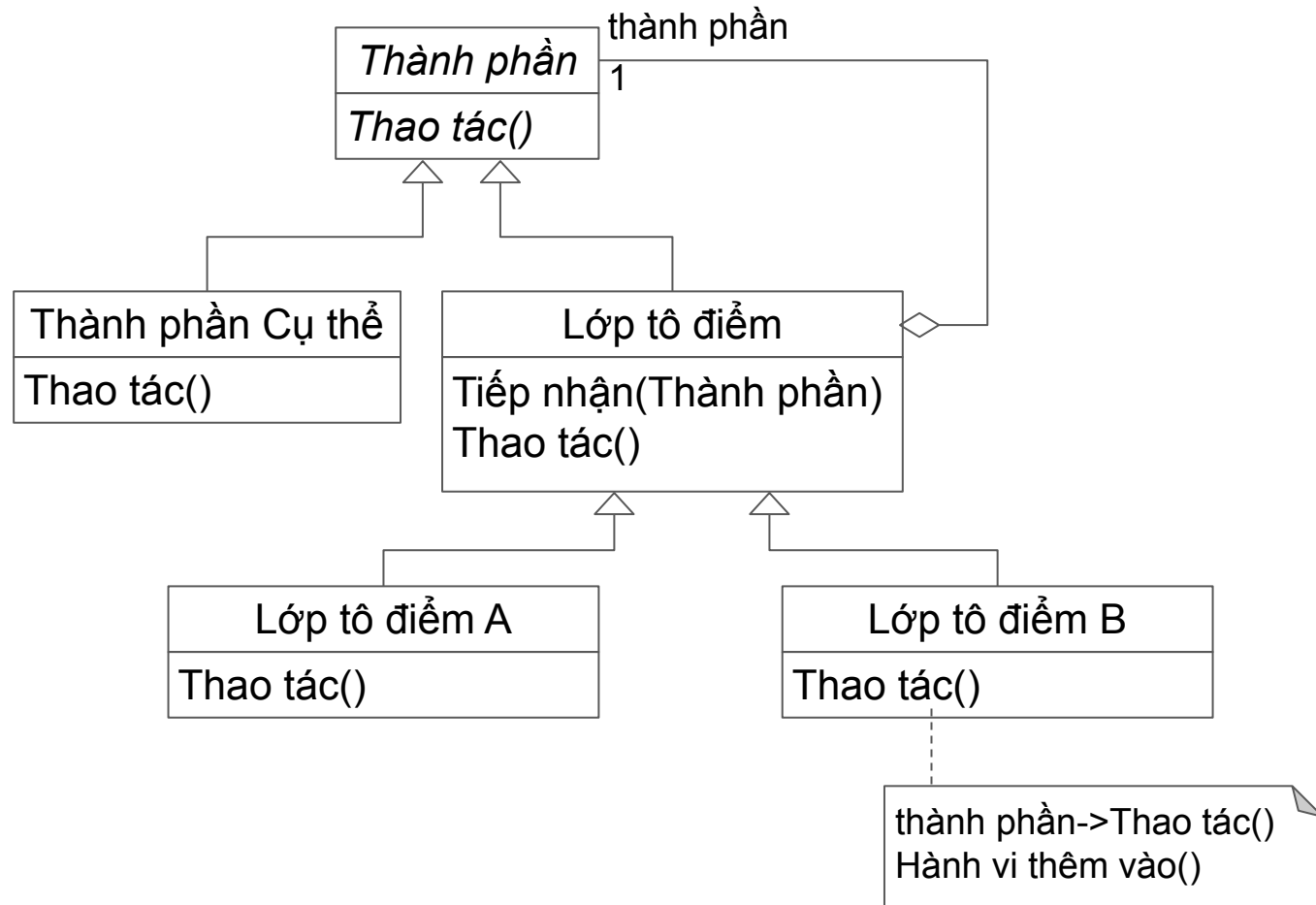
Decorator

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.

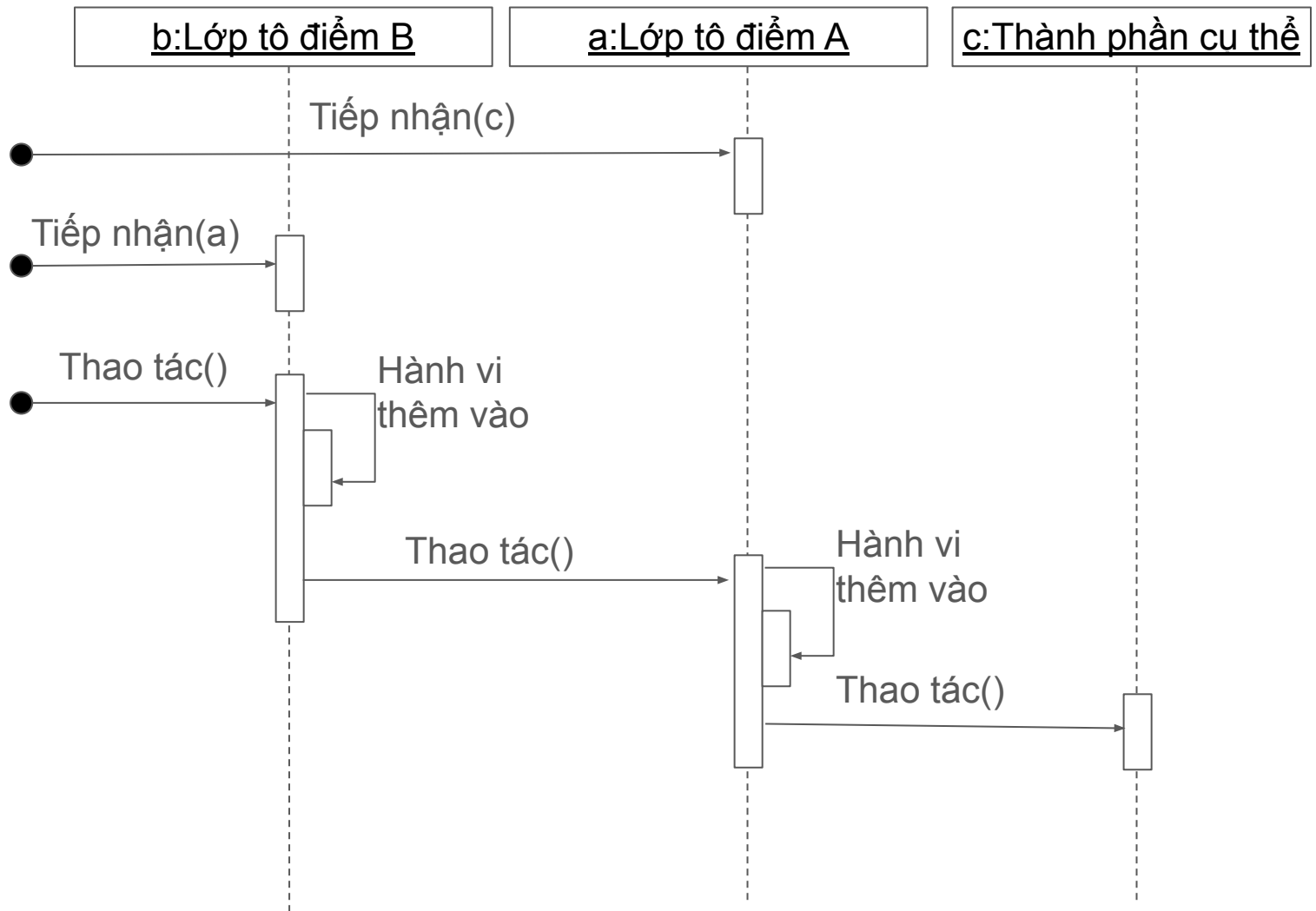
Tượng thạch cao + Màu => Tượng được tô màu

...

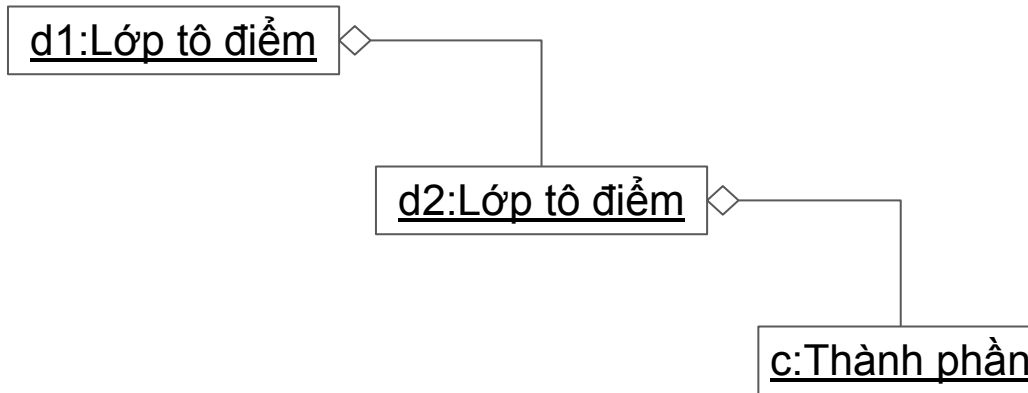
Lớp tô điểm: Cấu trúc



Lớp tô điểm: Hành vi



Lớp tô điểm: Kết hợp các đối tượng

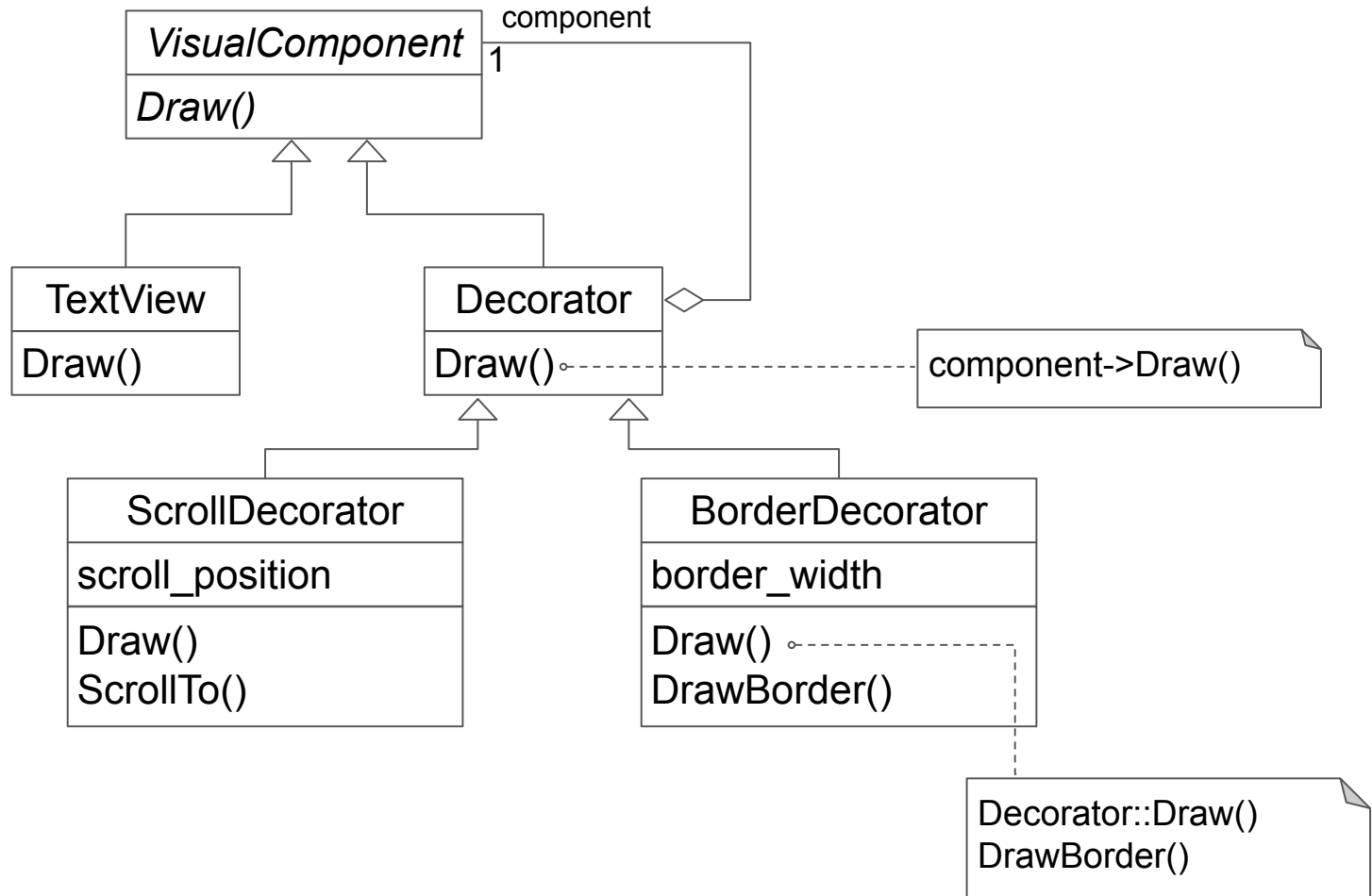


Thành phần được gói trong đối tượng thuộc lớp tô điểm.

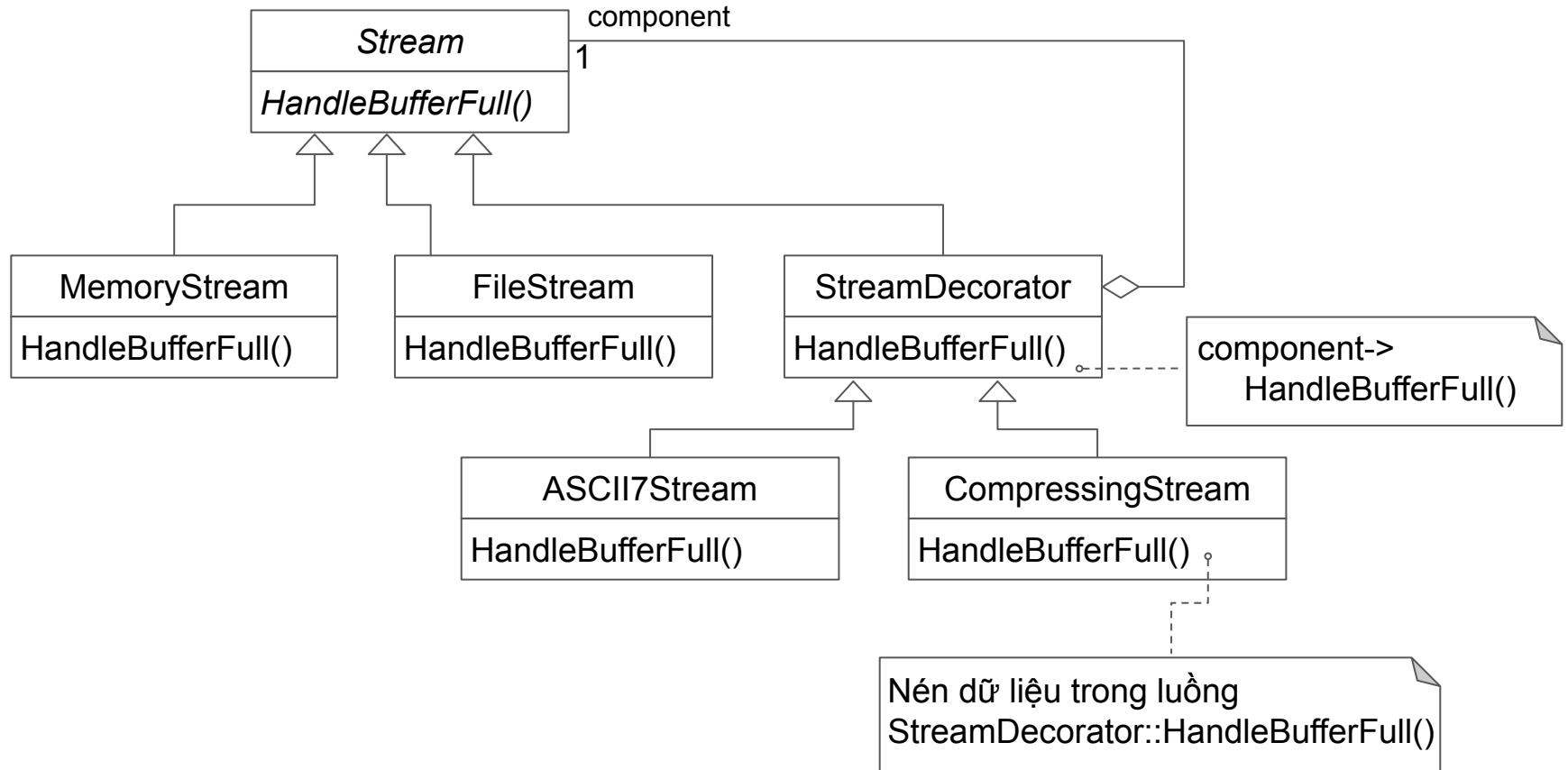
Lớp tô điểm: Các hệ quả

- Linh động hơn kế 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 các 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



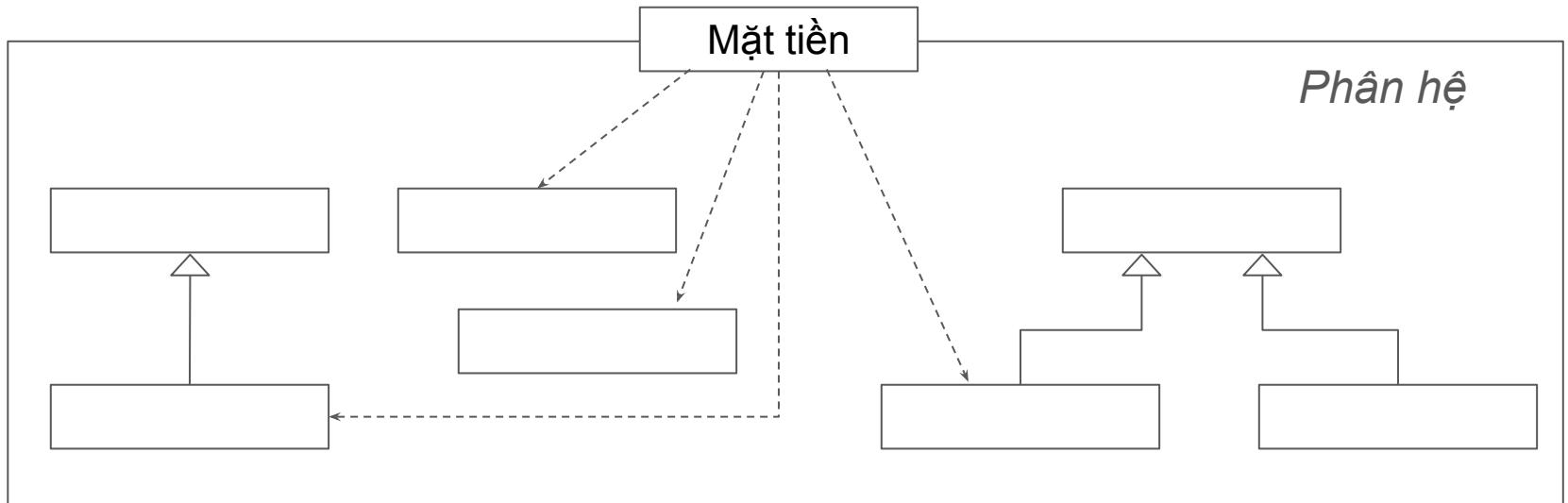
Mặt tiền

Facade

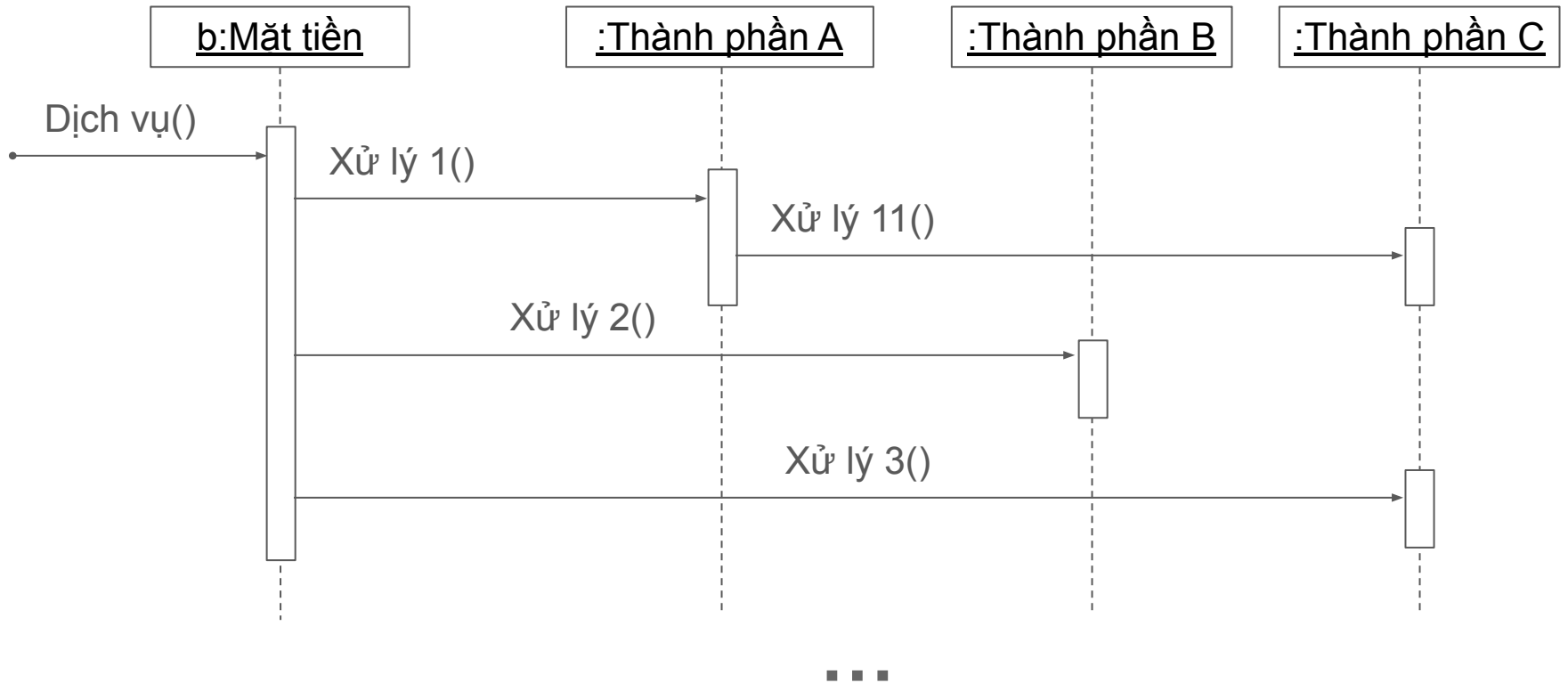
Cung cấp một giao diện hợp nhất cho một tập giao diện trong một phân hệ. Mặt tiền định nghĩa một giao diện bậc cao hơn để có thể sử dụng toàn phân hệ dễ dàng hơn.

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

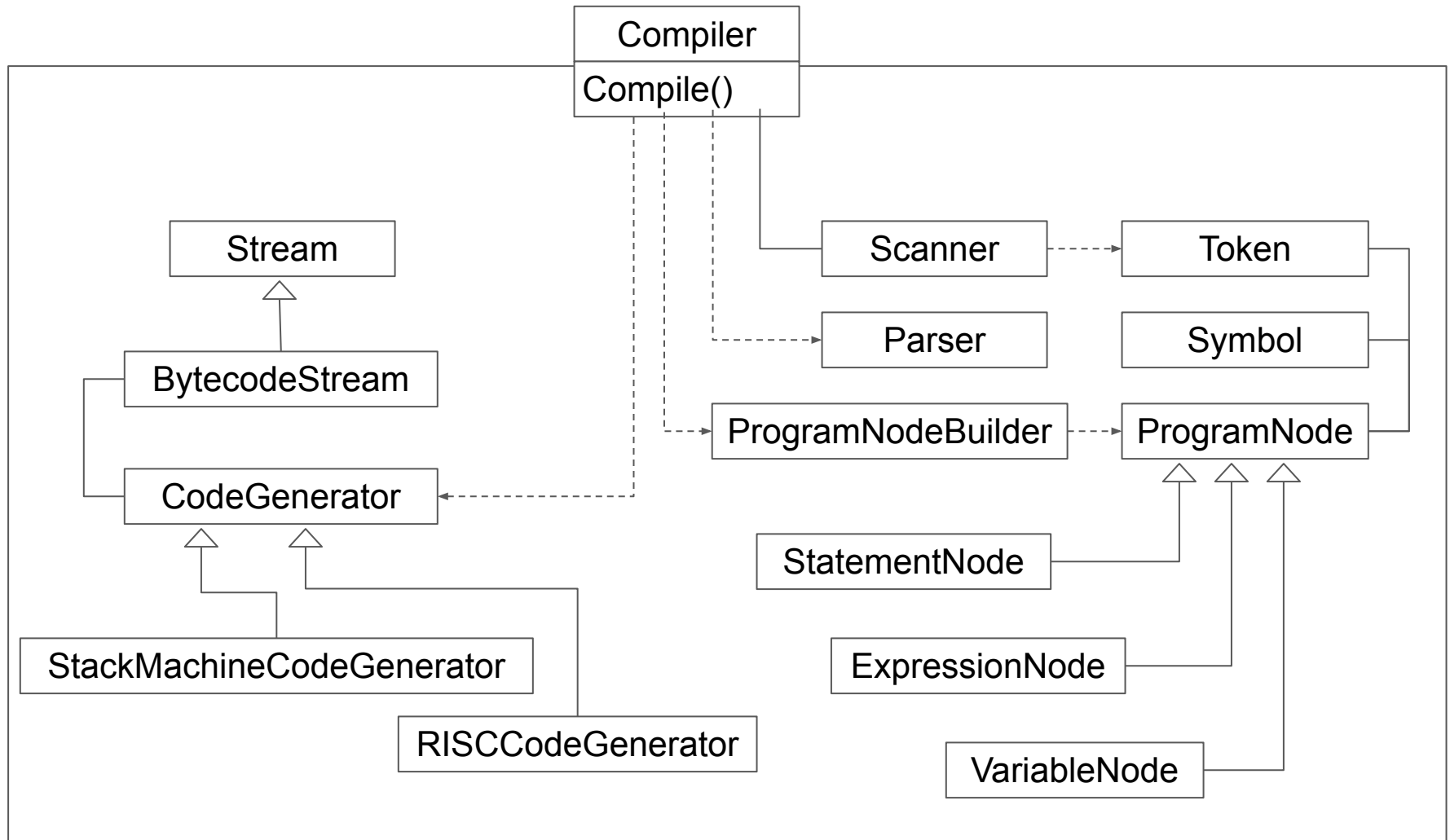
Mặt tiền: Cấu trúc



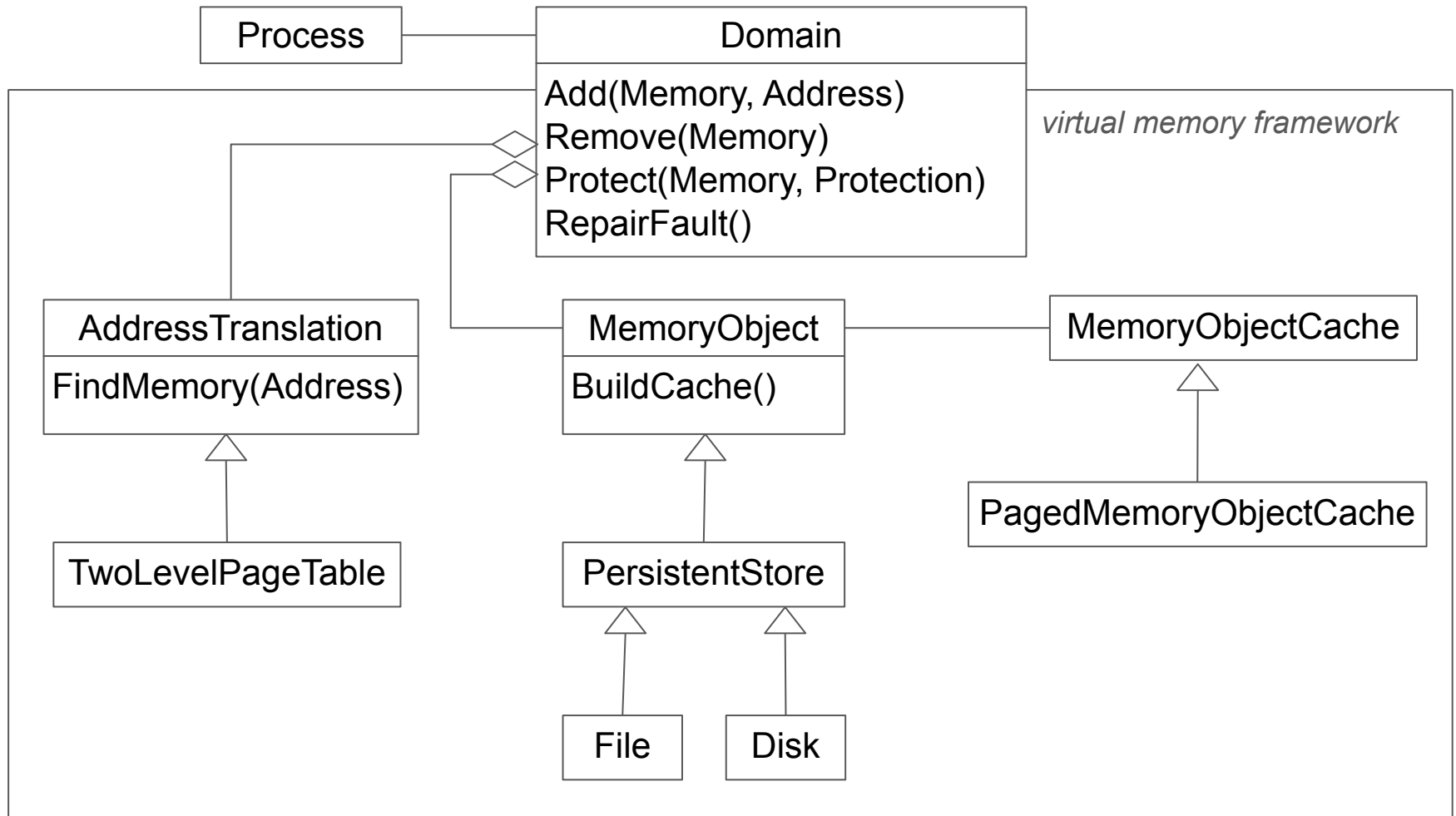
Mặt tiền: Hành vi



Ví dụ 12. trình biên dịch



Ví dụ 13. Quản lý bộ nhớ



Mặt tiền: Các hệ quả

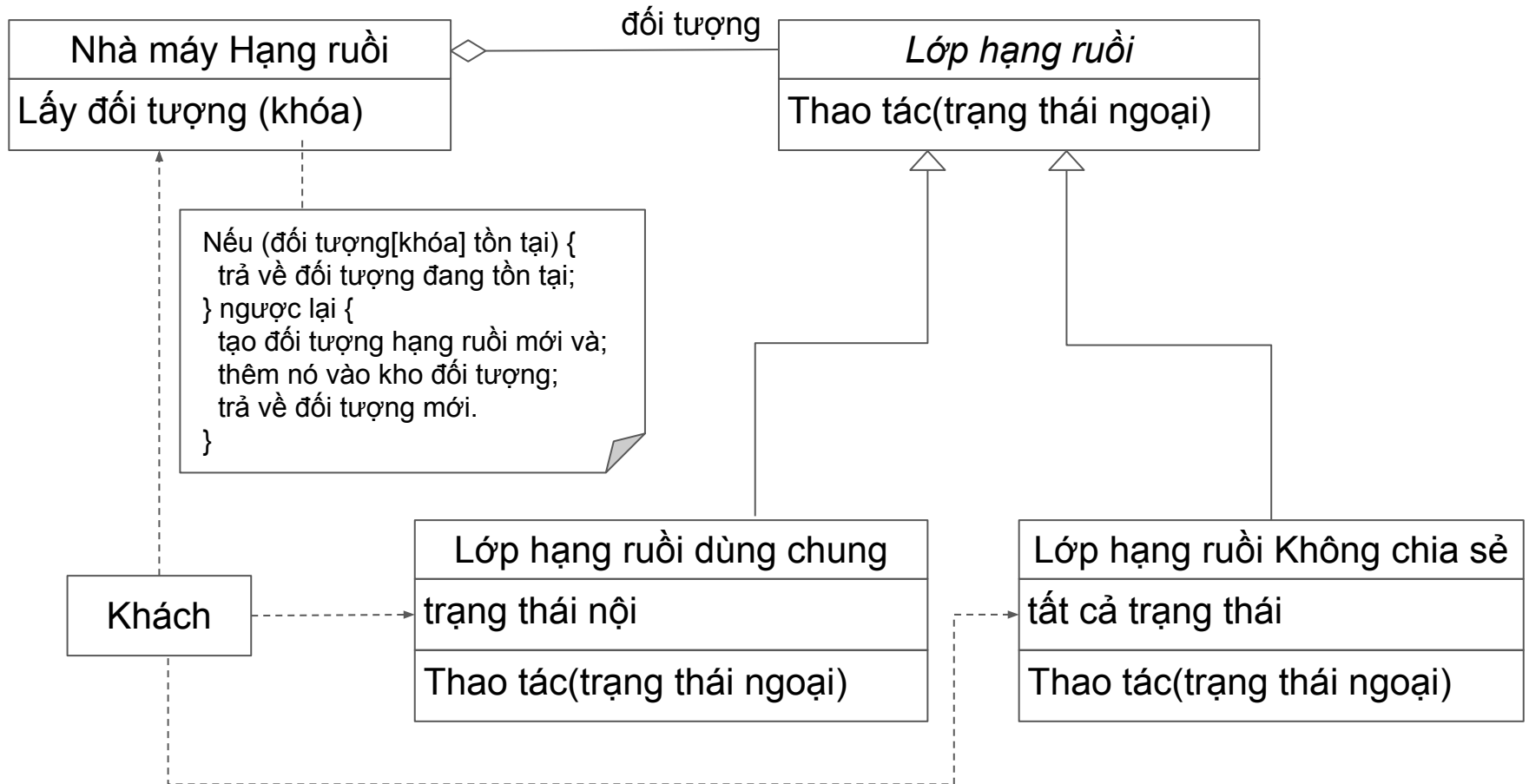
- Phân tách phạm vi sử dụng với các thành phần của phân hệ, qua đó giảm số lượng đối tượng tương tác và làm hệ thống dễ sử dụng hơn.
- Nói lảng phụ thuộc giữa phân hệ và phía khách.
- Ứng dụng vẫn có thể sử dụng các lớp trong phân hệ nếu cần.

Hạng ruồi

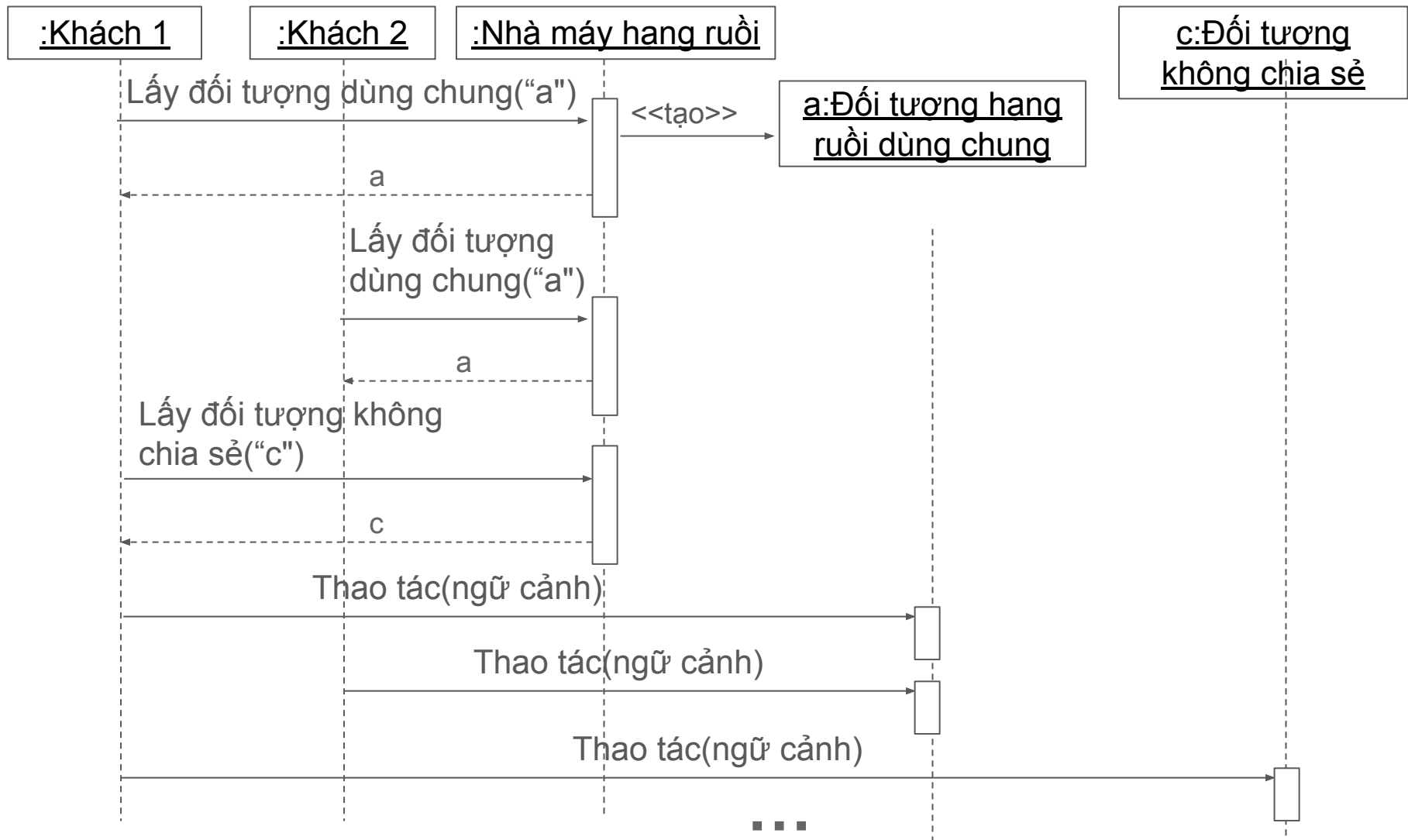
Flyweight

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

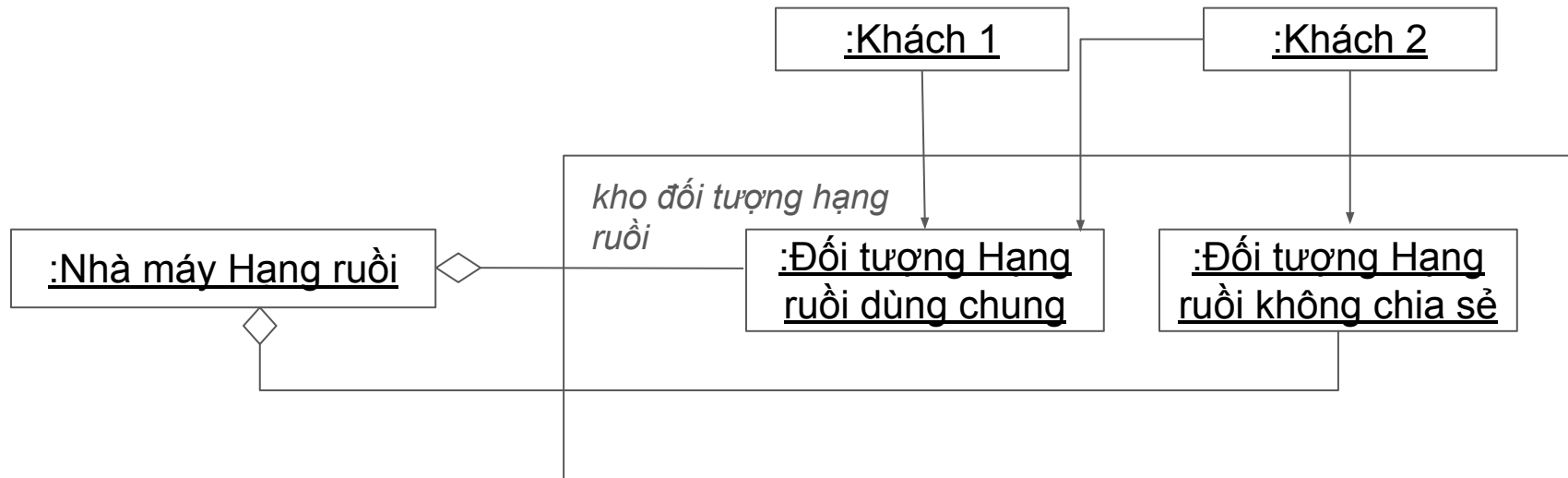
Hạng ruồi: Cấu trúc



Hạng ruồi: Hành vi



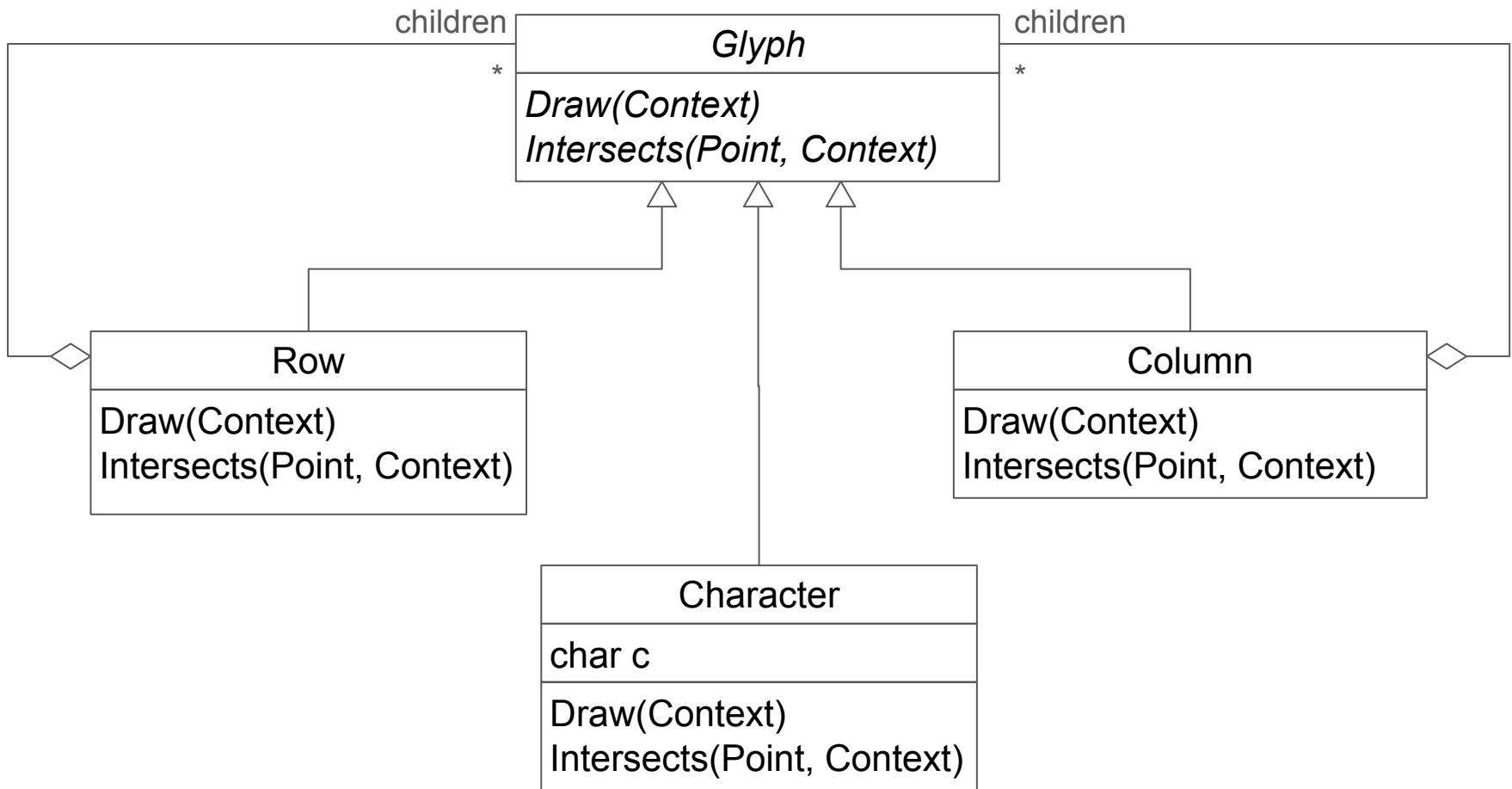
Hạng ruồi: Kết hợp đối tượng



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. Hiện thị văn bản

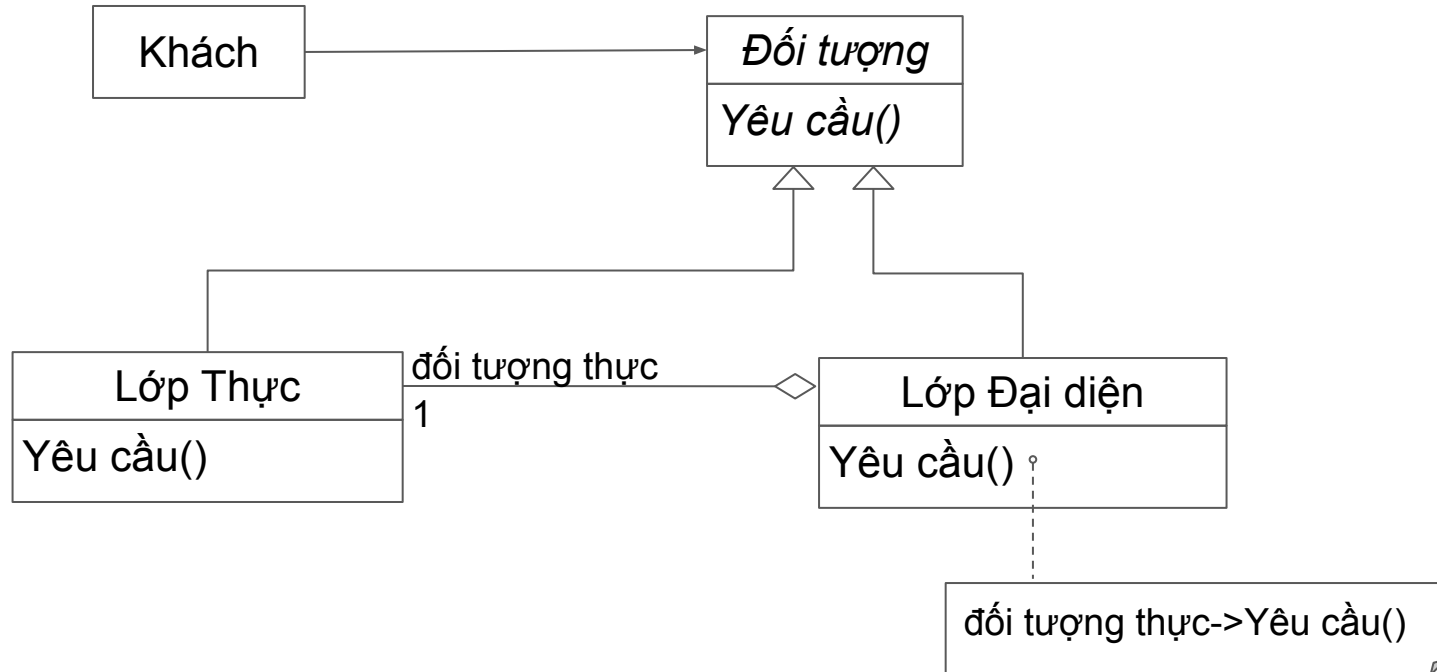


Lớp đại diện

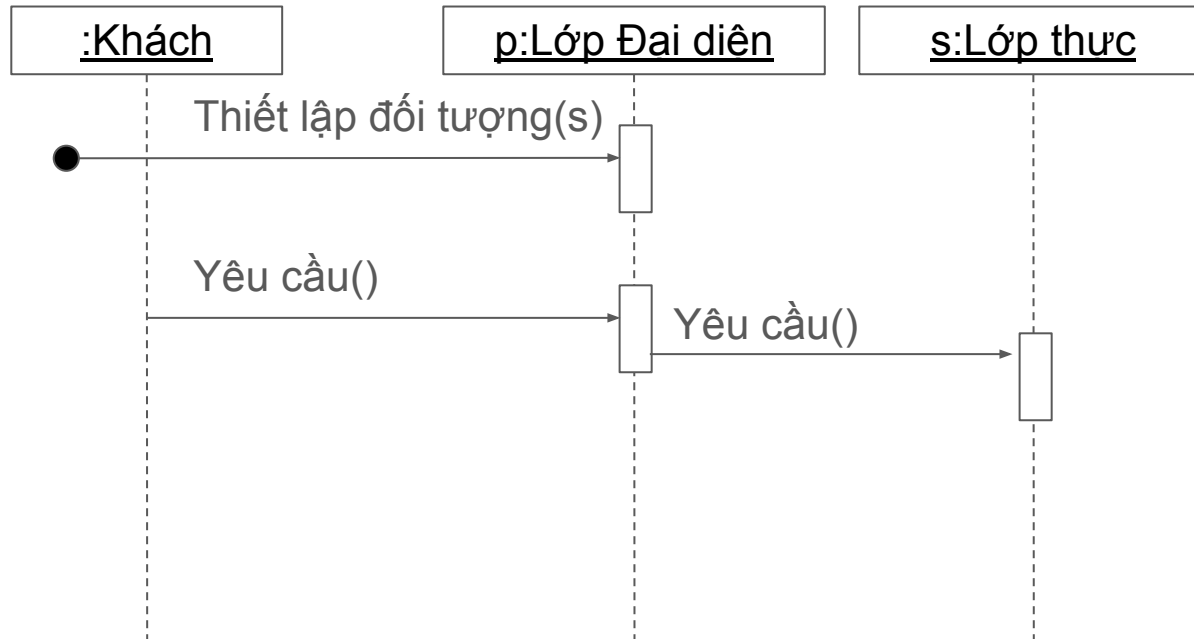
Proxy

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



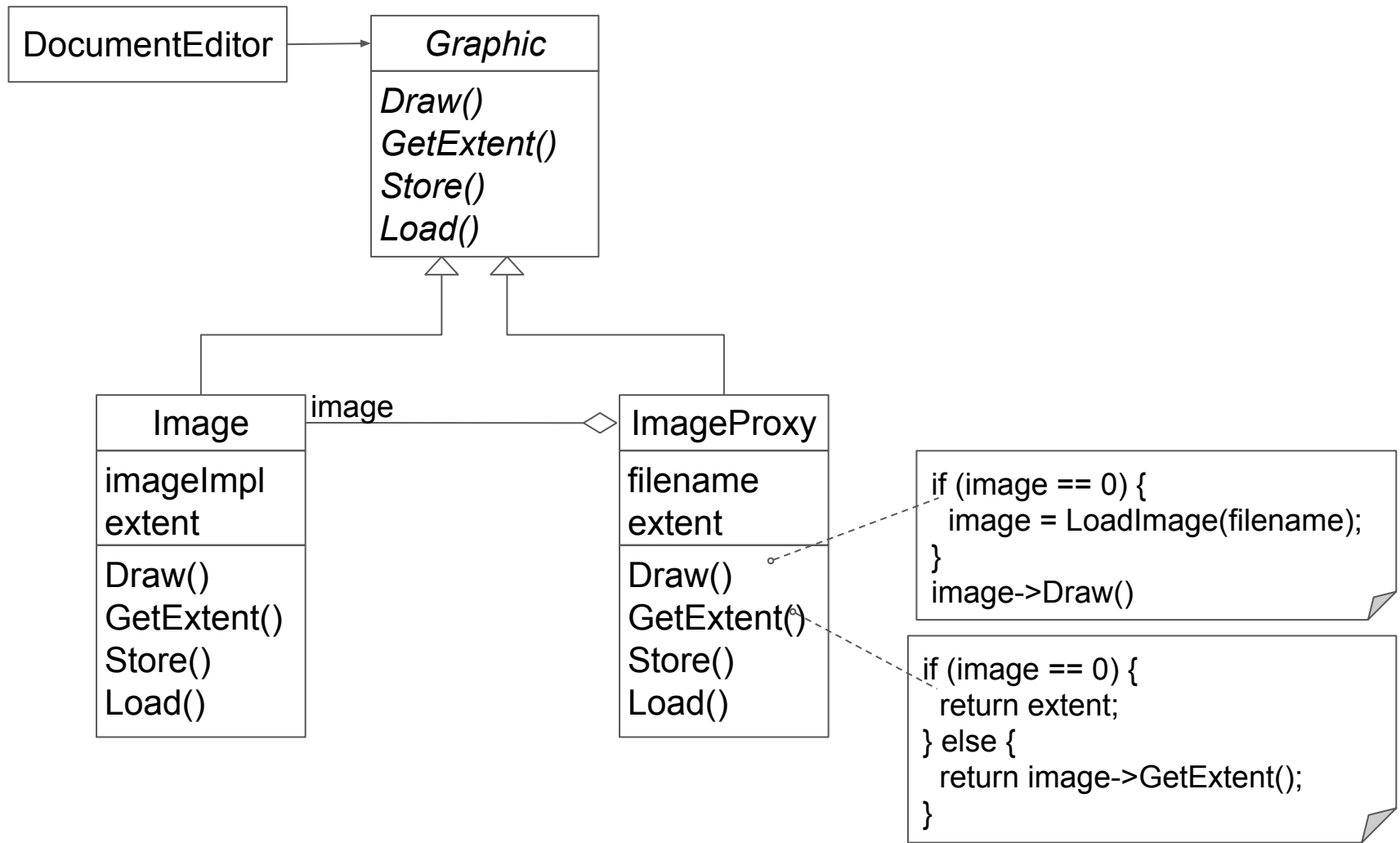
Đại diện: Hành vi



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 bổ 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

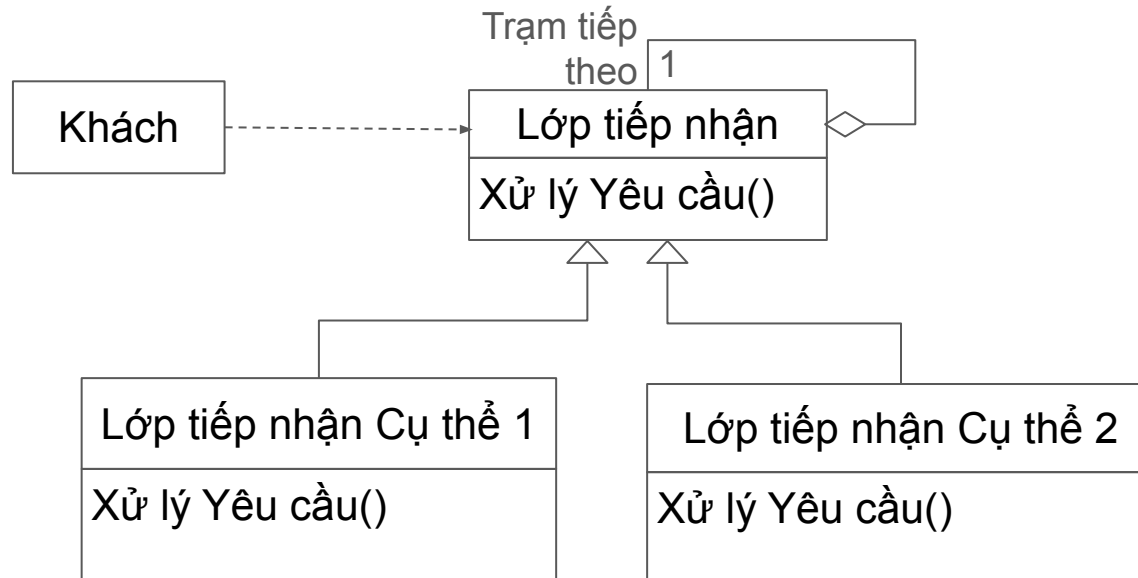


Chuỗi trách nhiệm

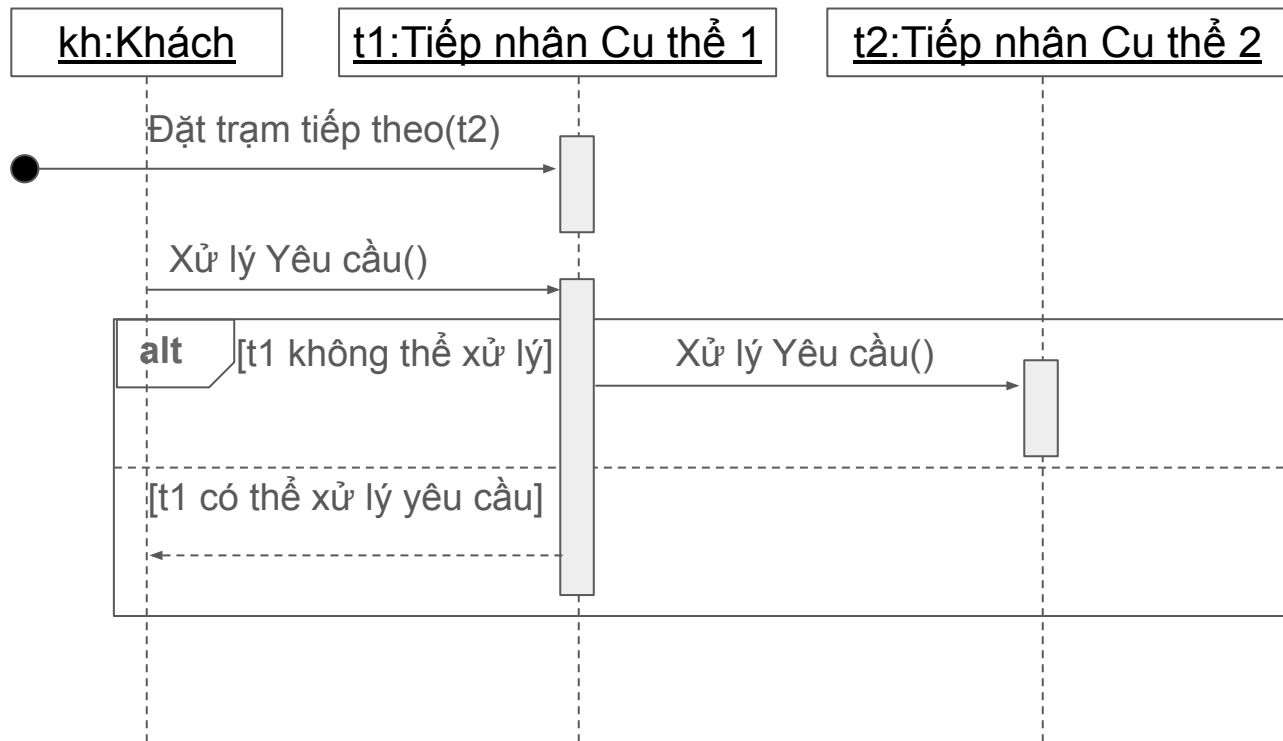
Chain of Responsibility

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



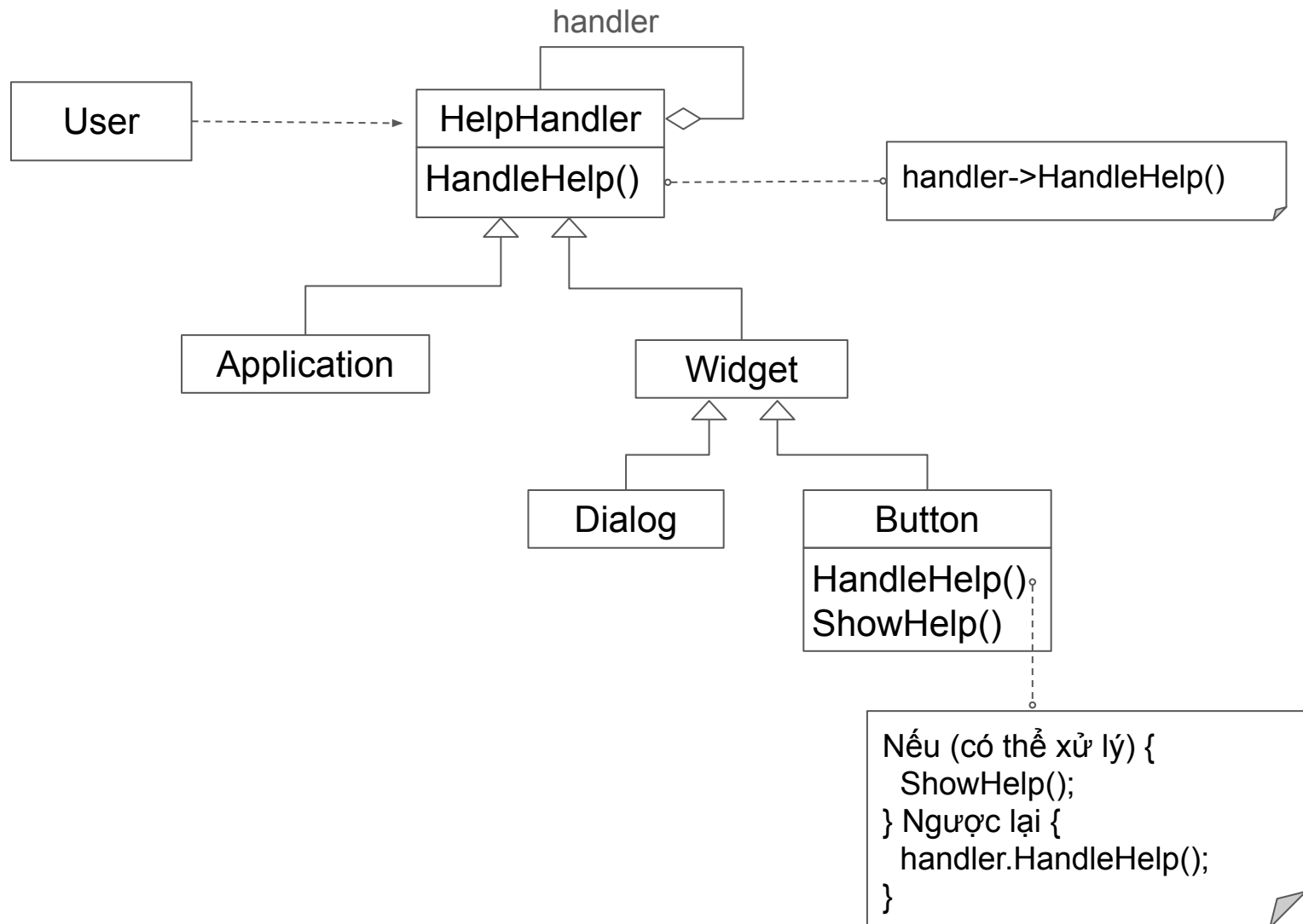
Chuỗi trách nhiệm: Hành vi



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

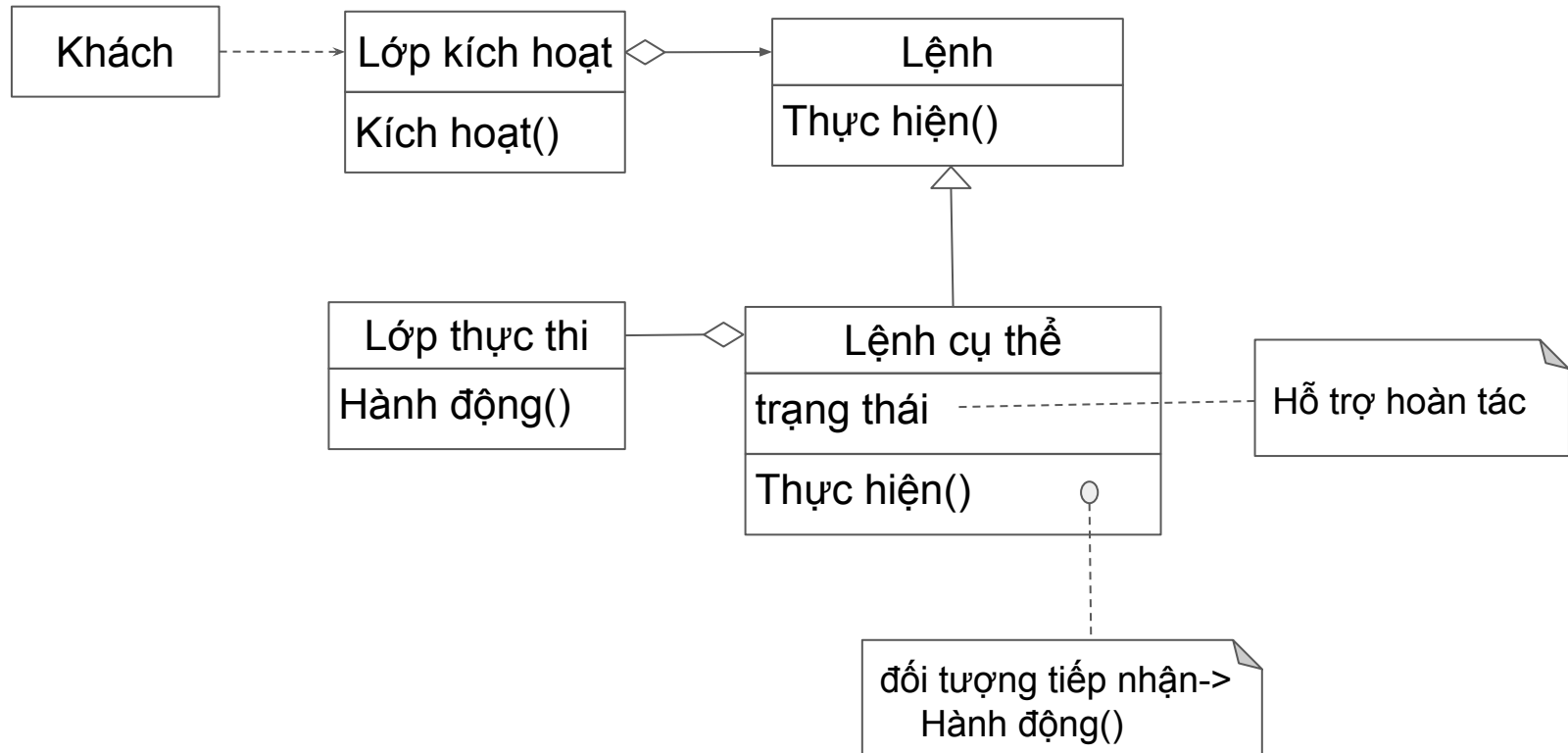


Lệnh

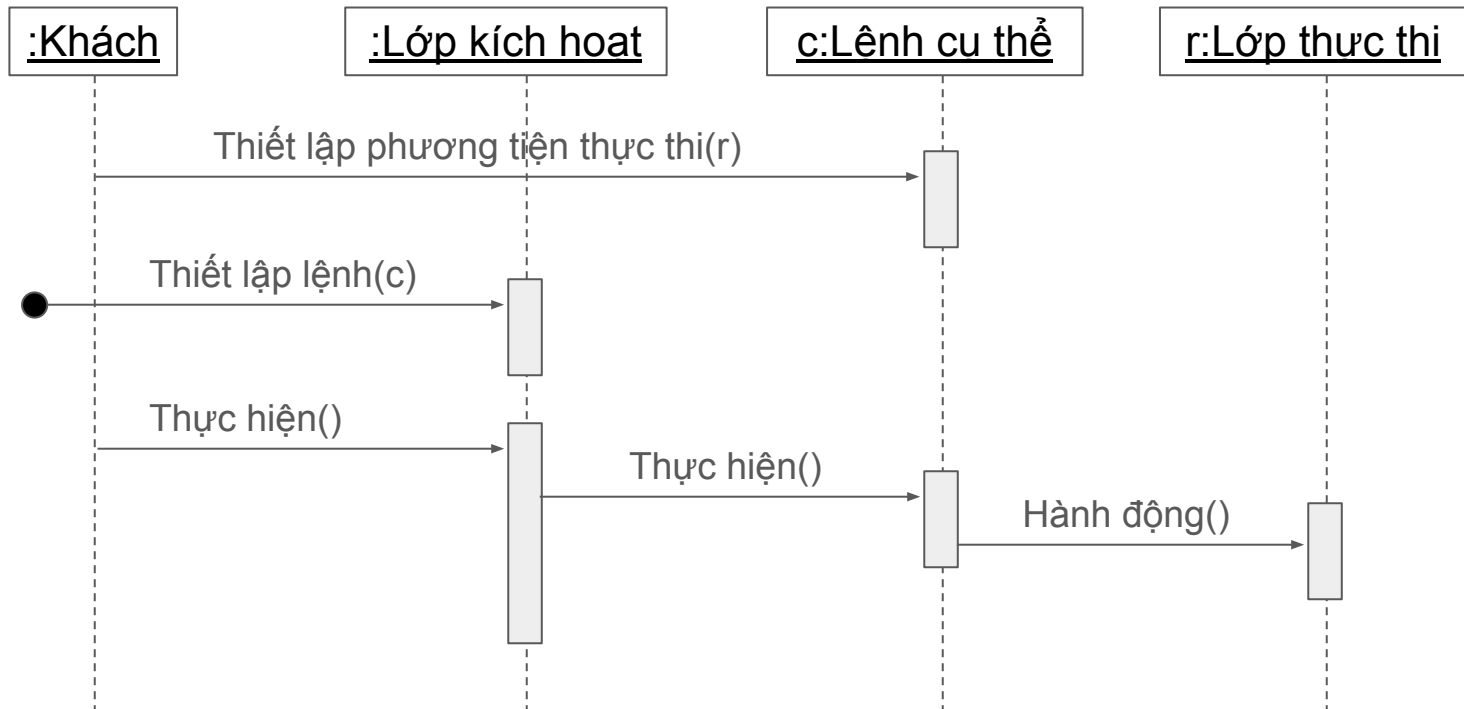
Command

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



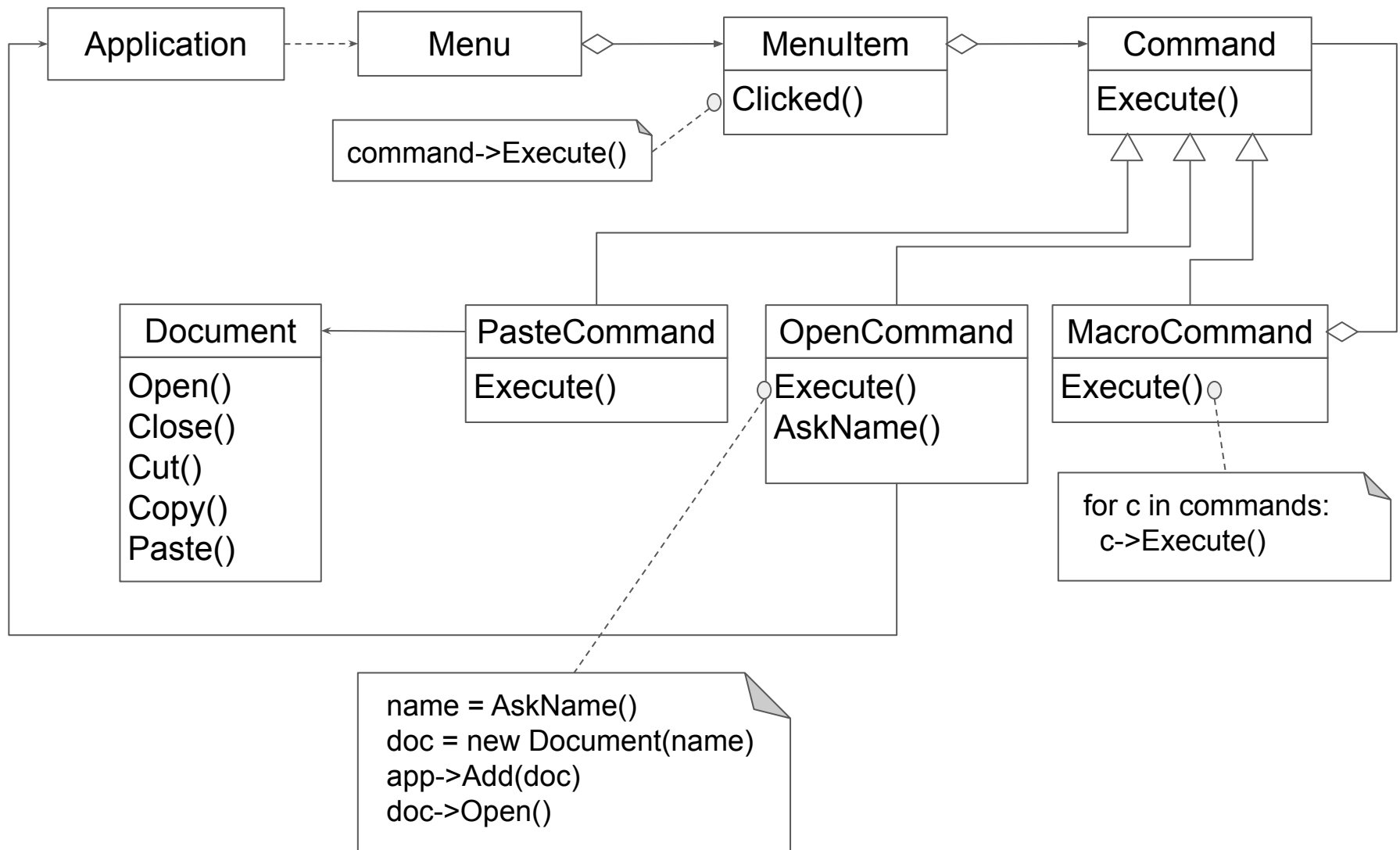
Lệnh: Hành vi



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 xung lệnh mới vì không cần thay đổi các lớp đang có.

Ví dụ 17. Menu chương trình

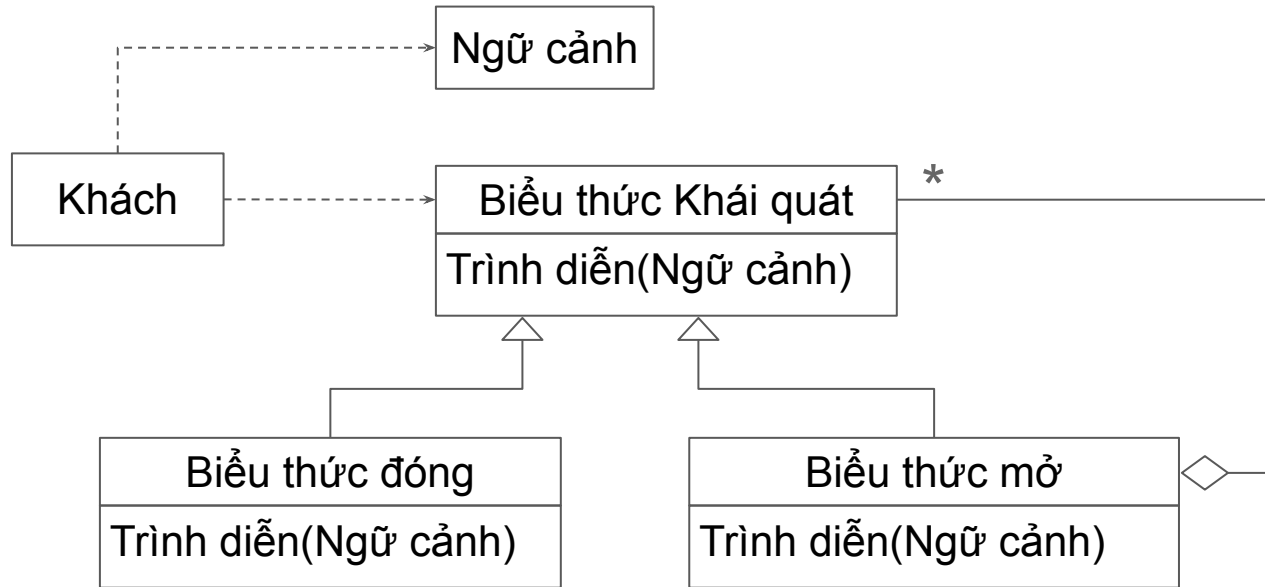


Lớp diễn dịch

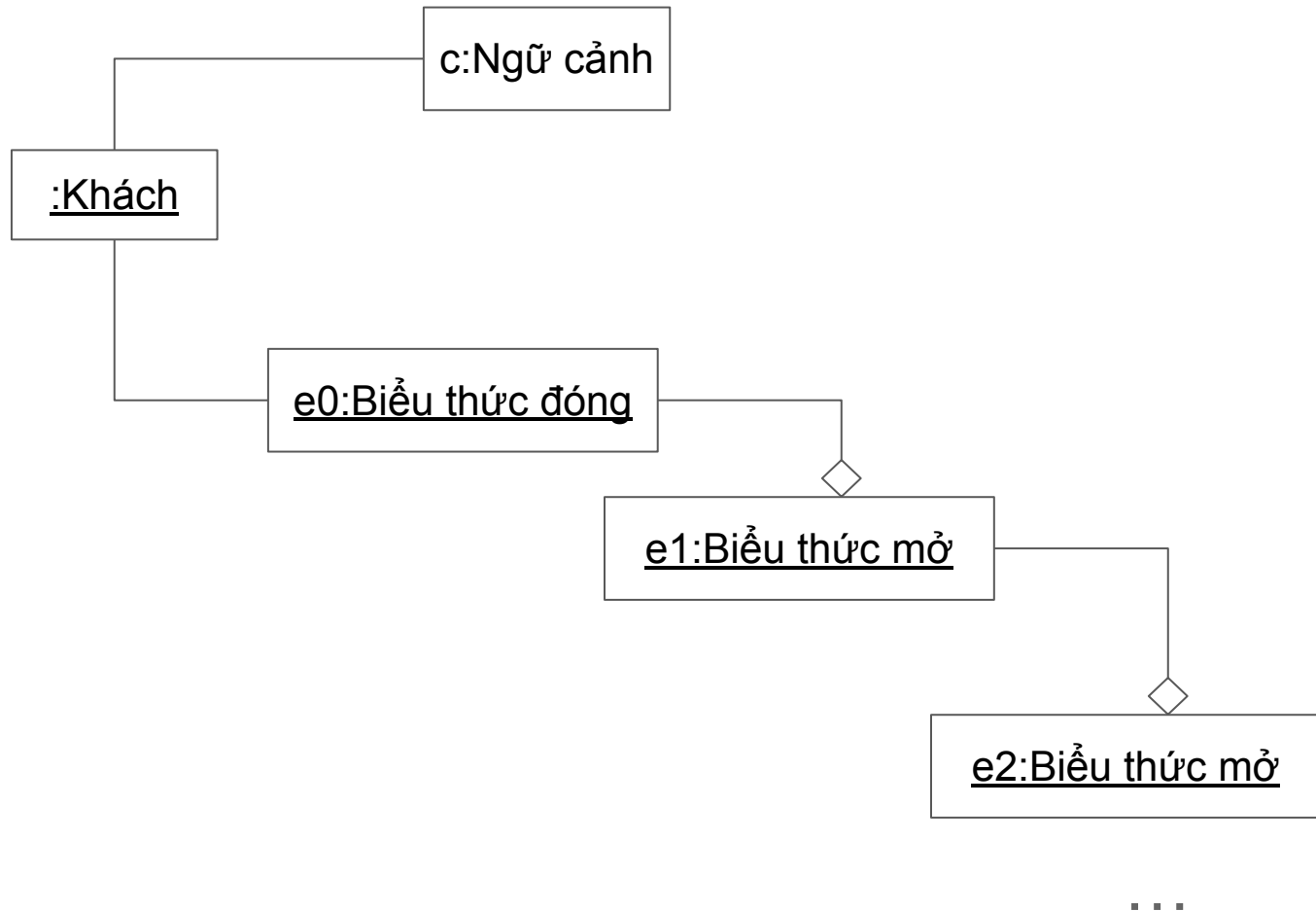
Interpreter

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.

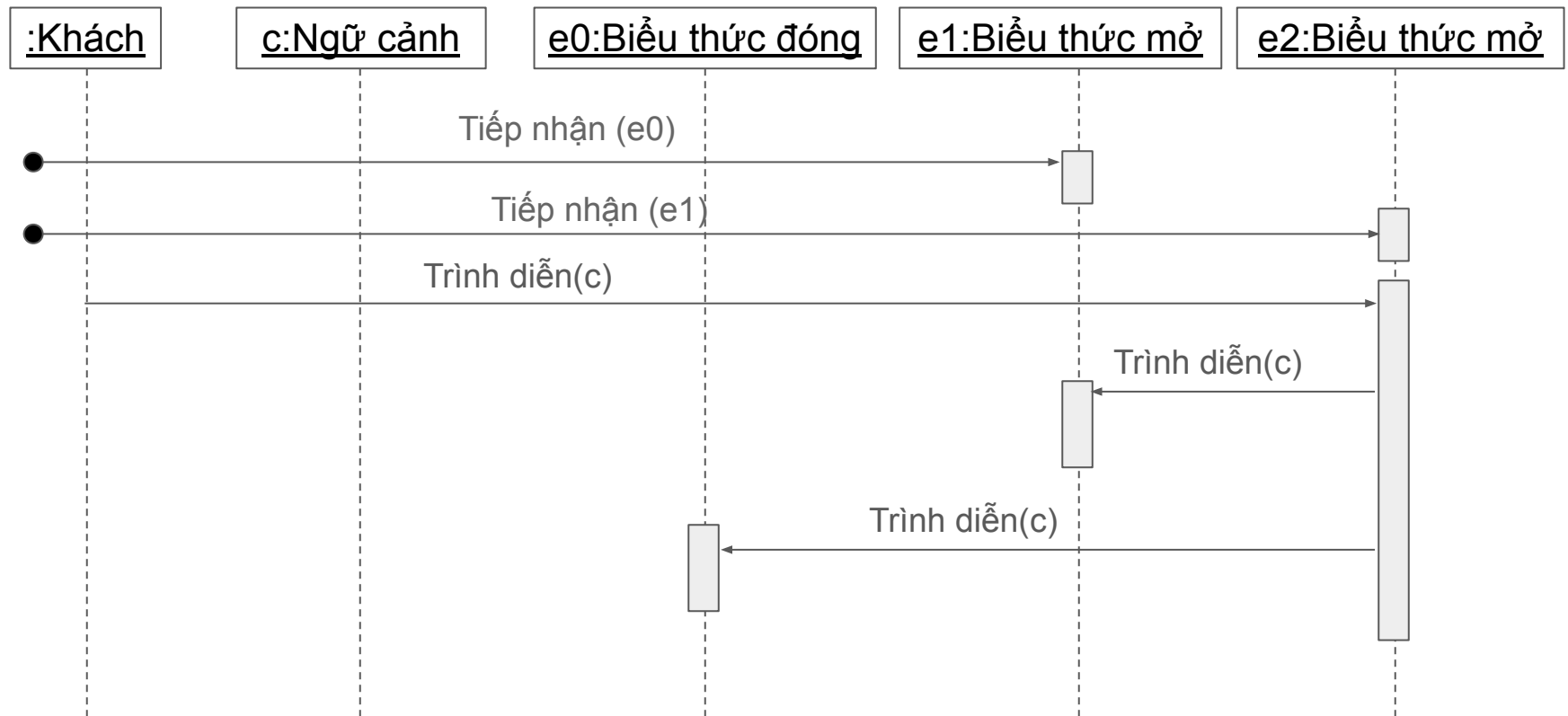
Lớp diễn dịch: Cấu trúc



Lớp diễn dịch: Kết hợp đối tượng



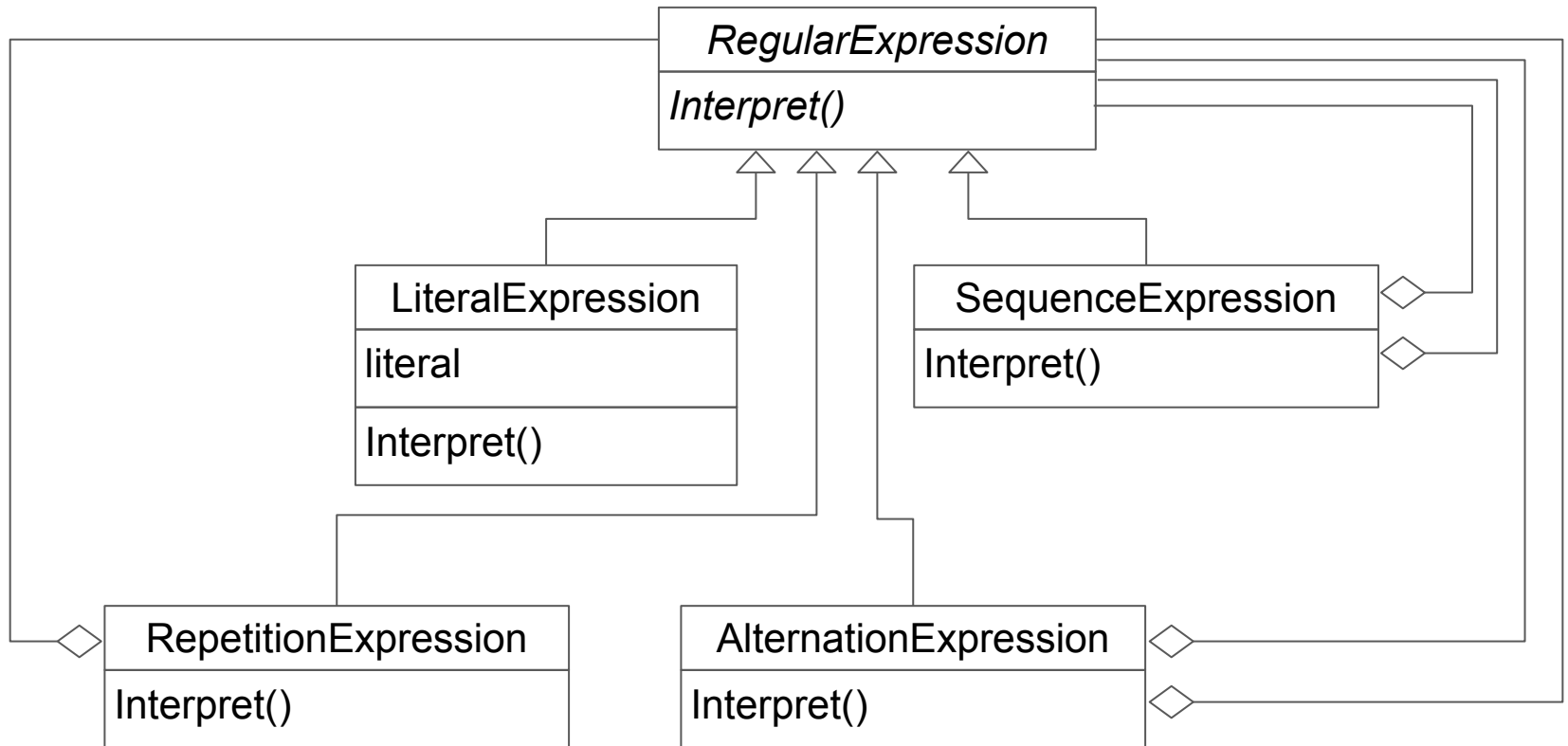
Lớp diễn dịch: Hành vi



Lớp diễn dịch: Các hệ quả

- Dễ thay đổi và mở rộng ngữ 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 ngữ pháp đơn giản.
- Khó duy trì ngữ 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

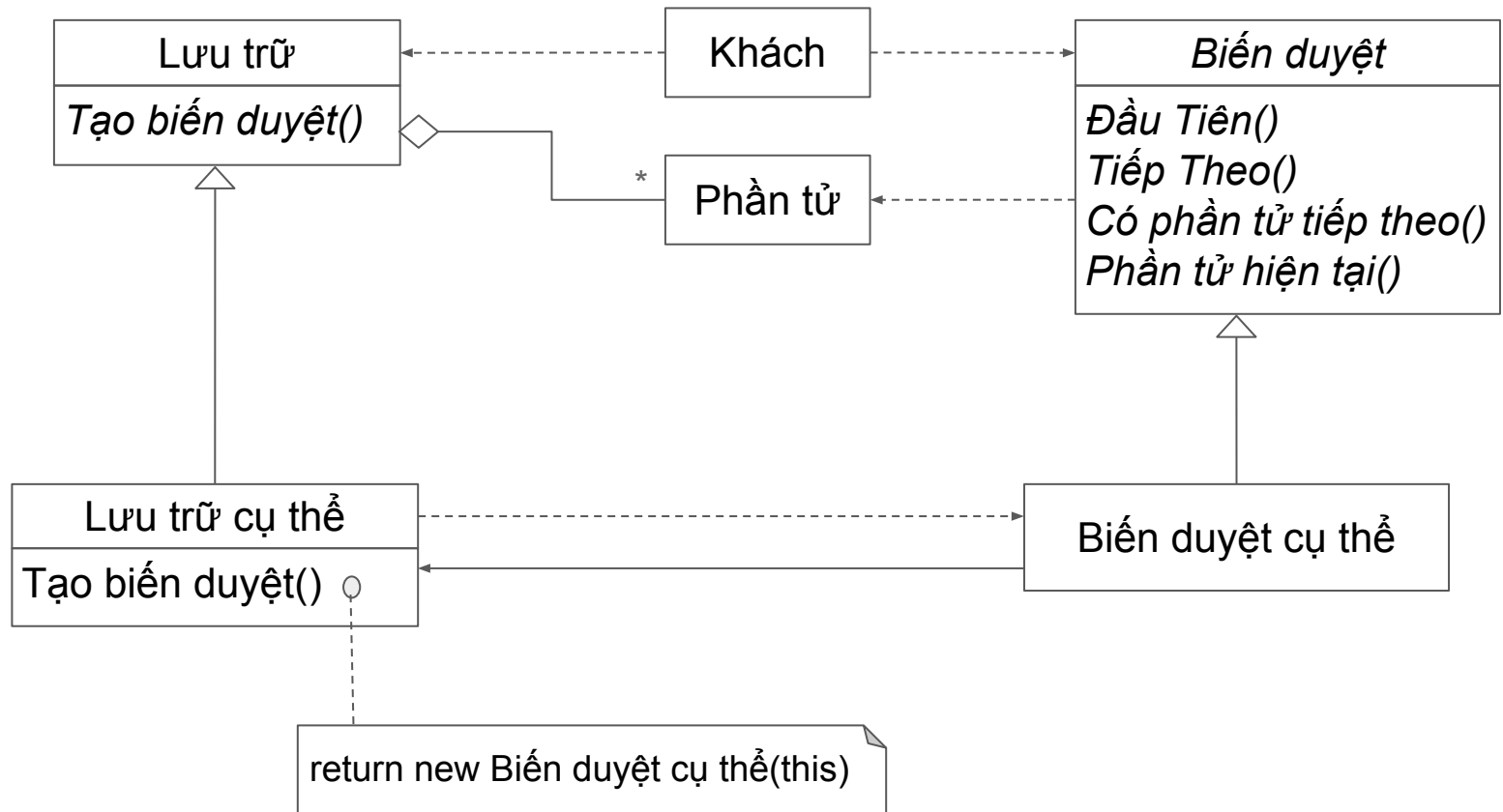


Biến duyệt

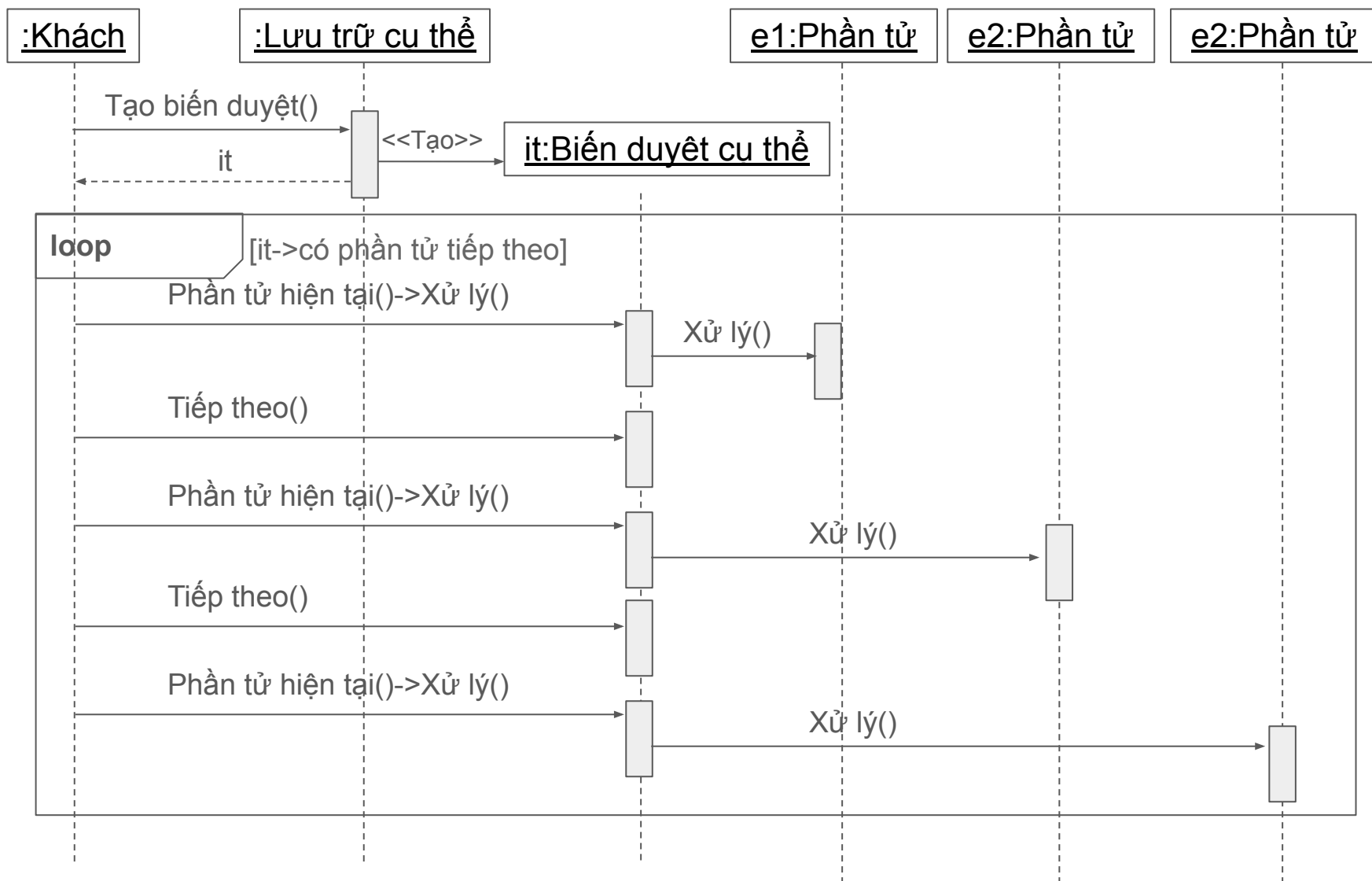
Iterator

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



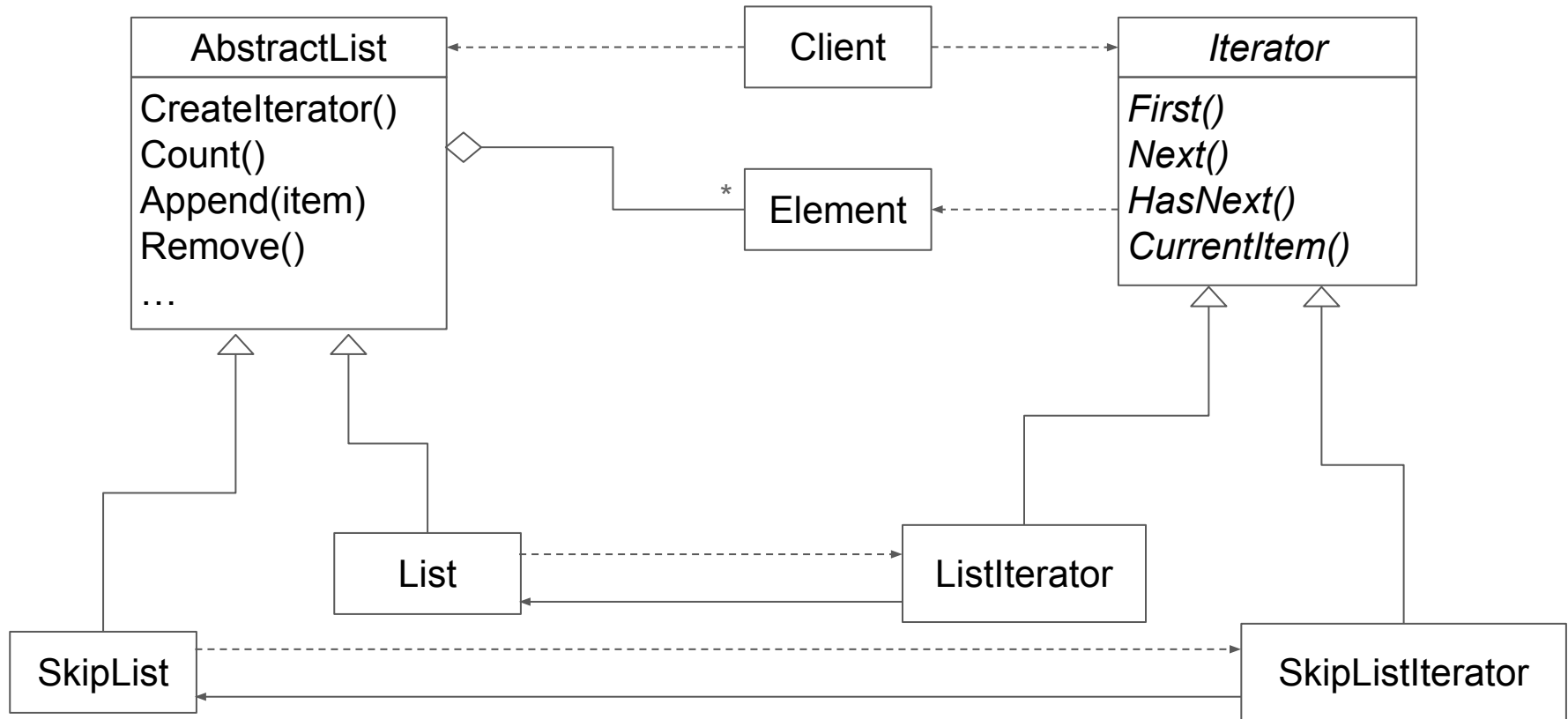
Biến duyệt: Hành vi



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 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

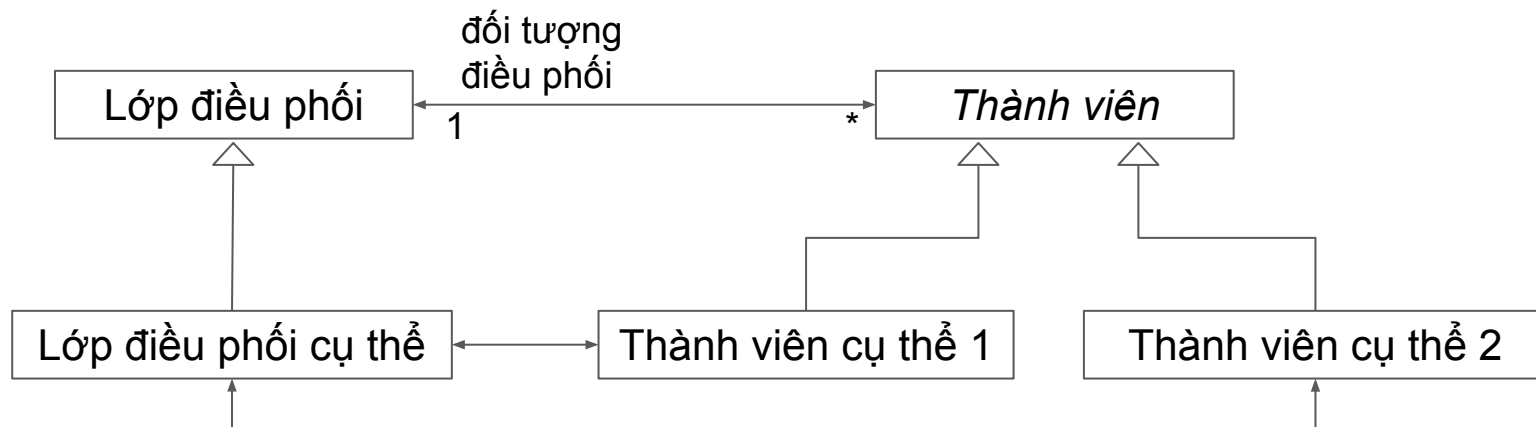


Lớp điều phối

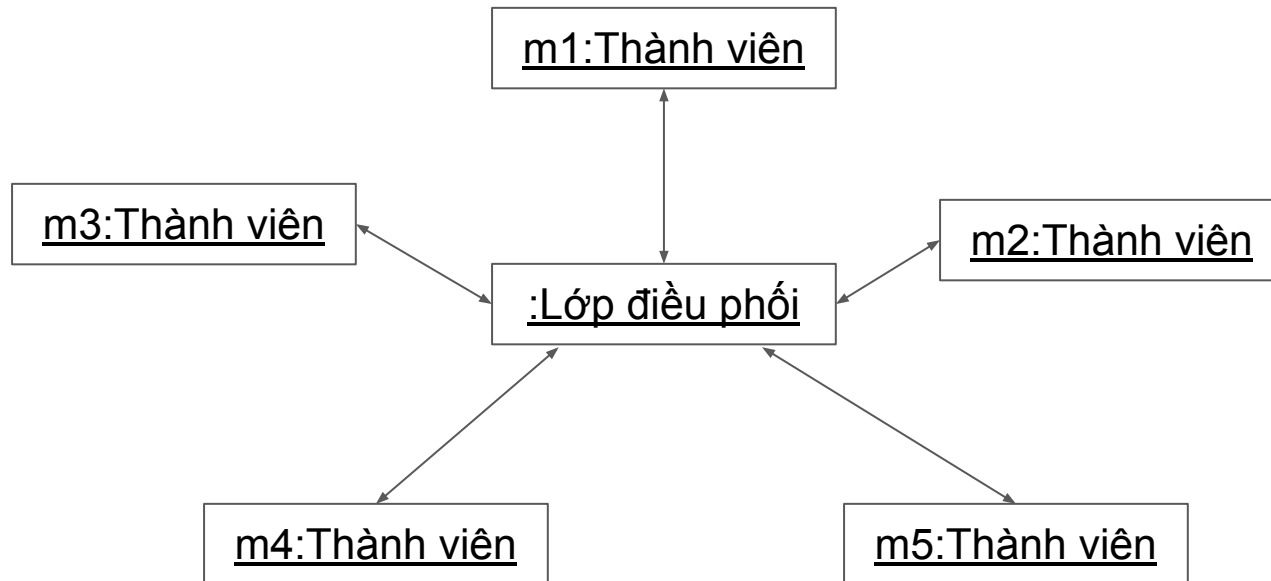
Mediator

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.

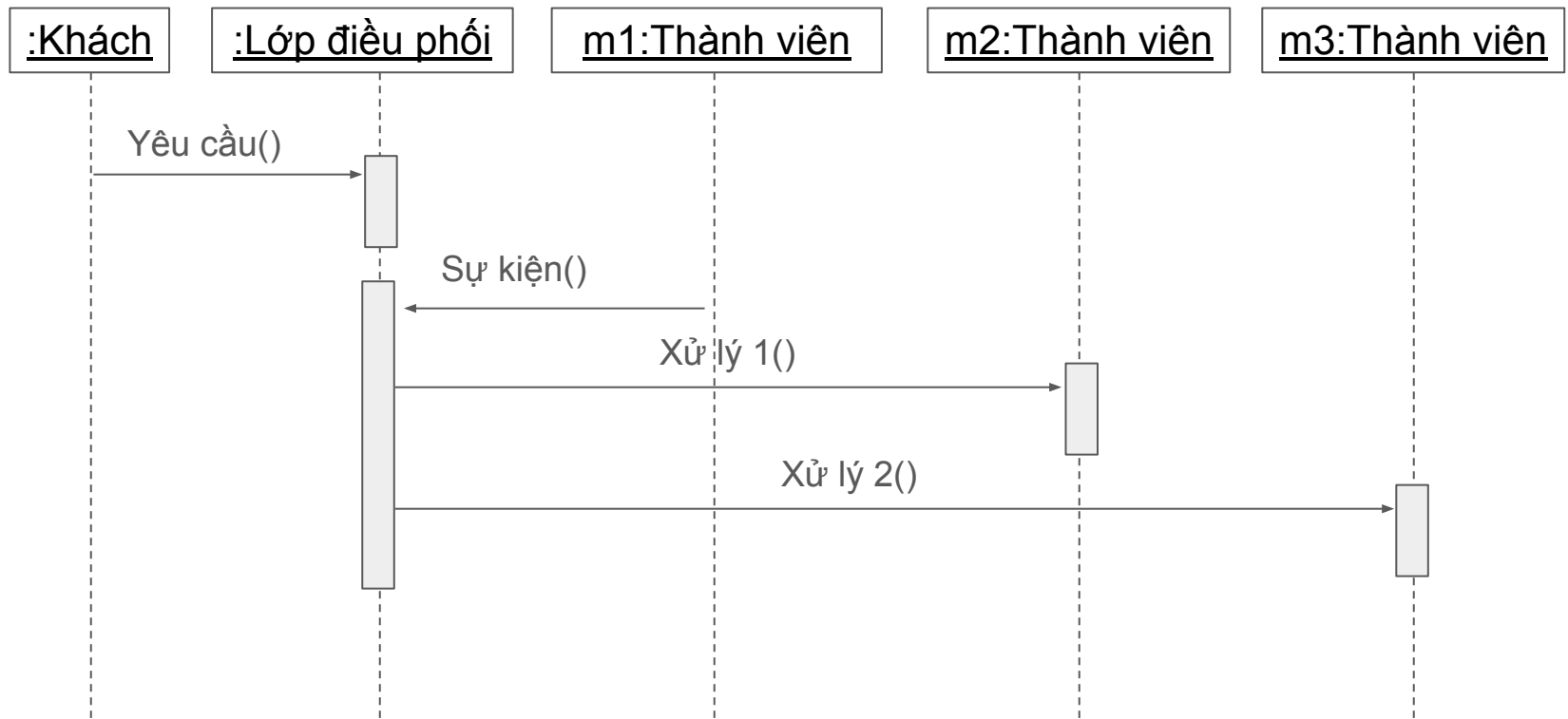
Lớp điều phối: Cấu trúc



Lớp điều phối: Kết hợp đối tượng



Lớp điều phối: Hành vi

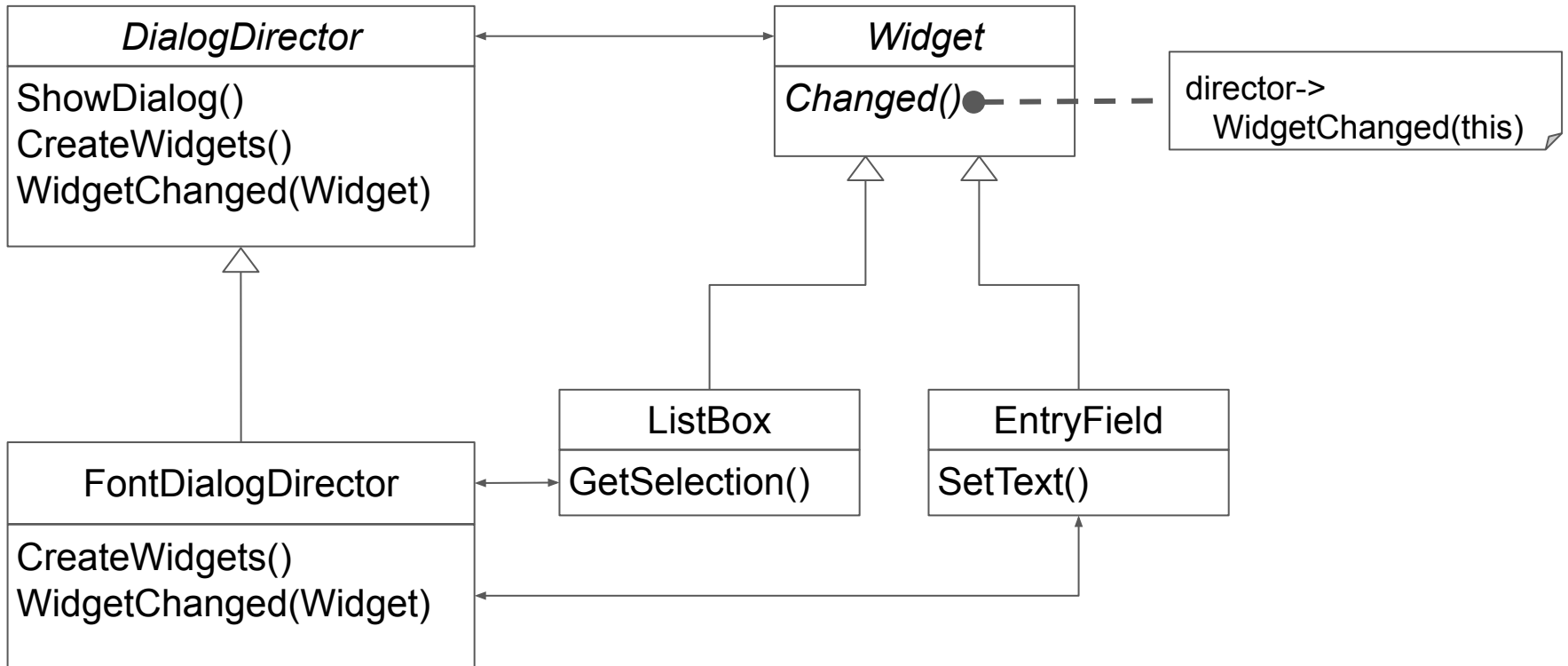


Các thành viên gửi và nhận các yêu cầu từ đối tượng điều phối. Đối tượng điều phối triển khai phối hợp bằng cách điều hướng các yêu cầu giữa các thành viên liên quan.

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

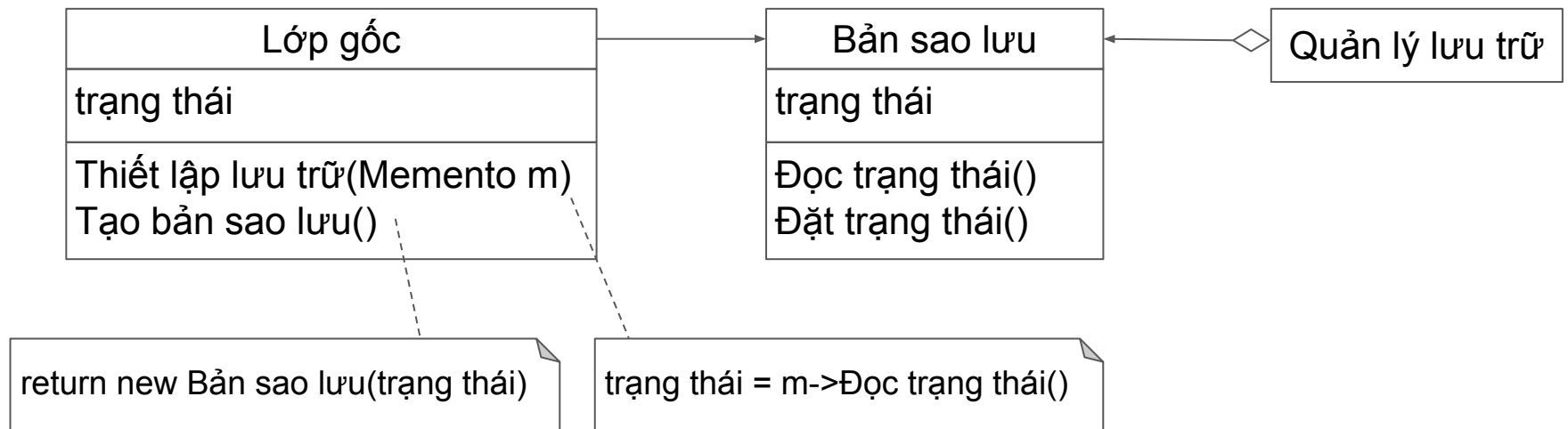


Bản sao lưu

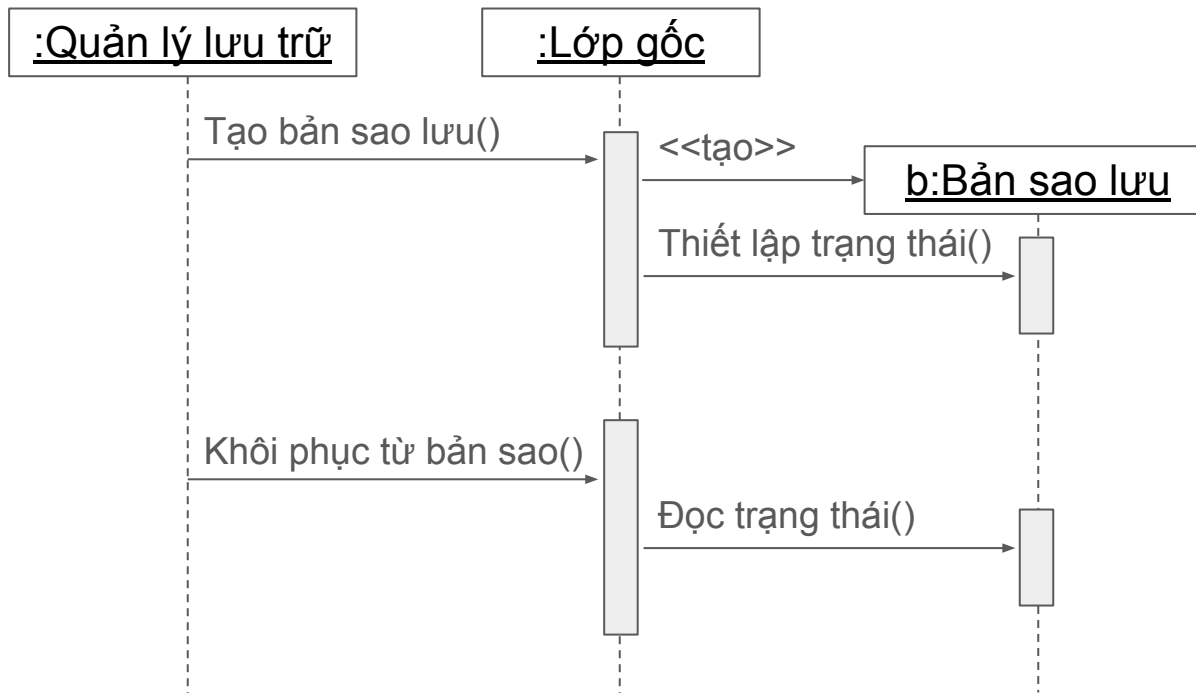
Memento

Thu thập 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



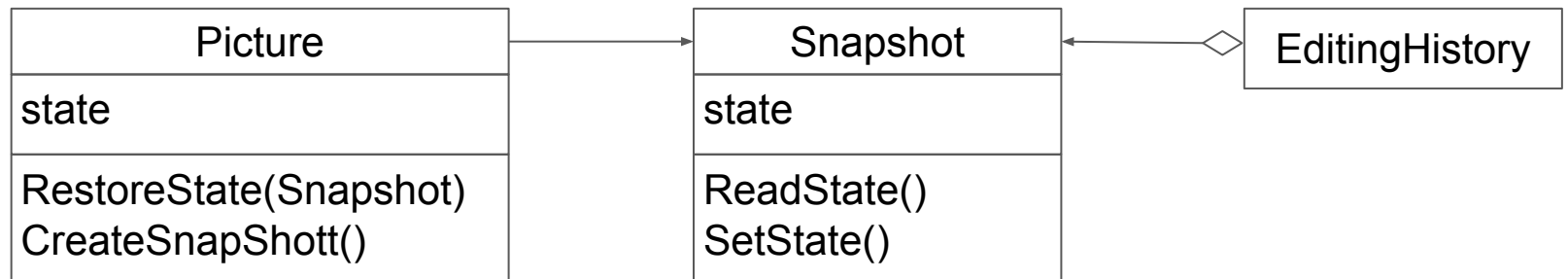
Bản sao lưu: Hành vi



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

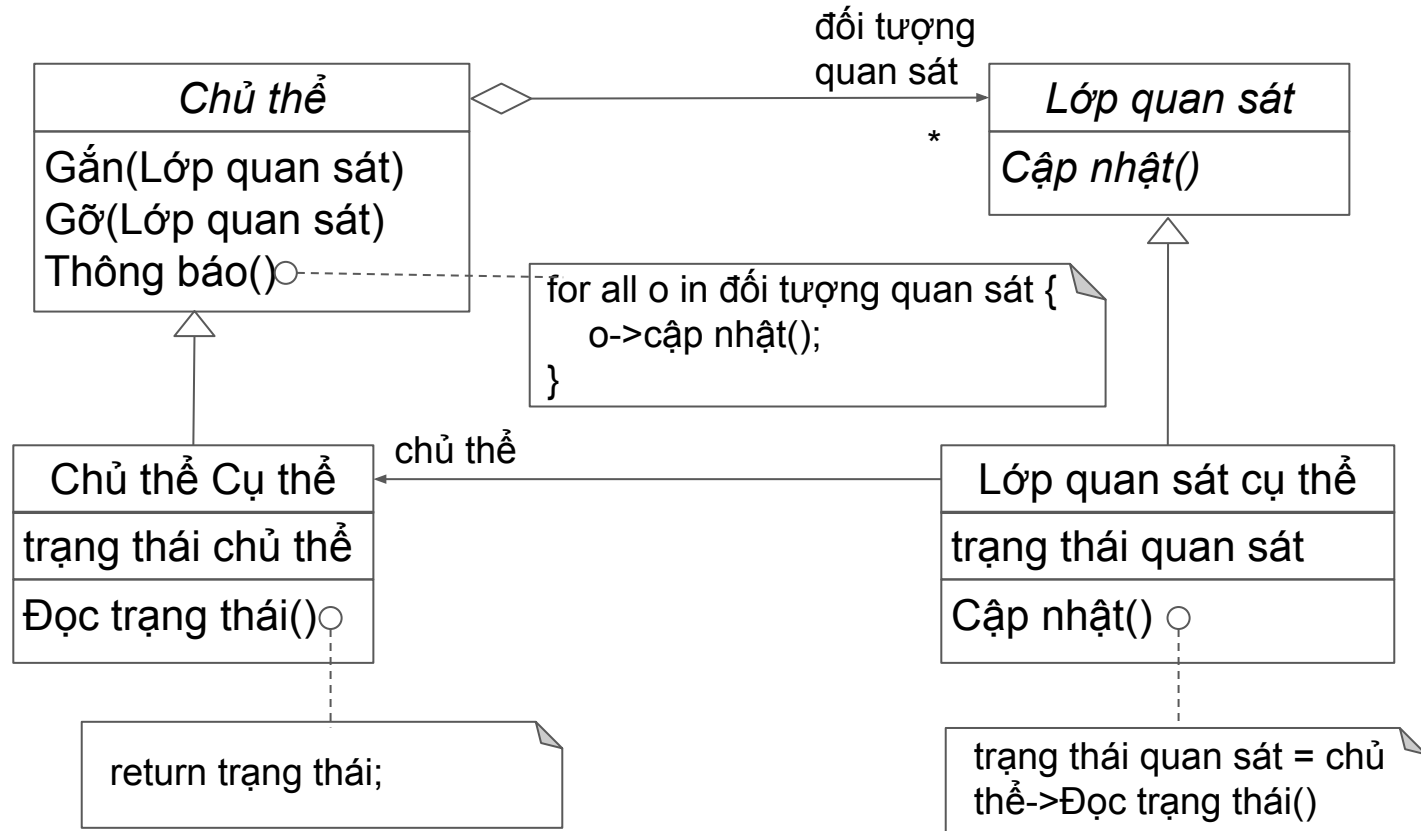


Lớp quan sát

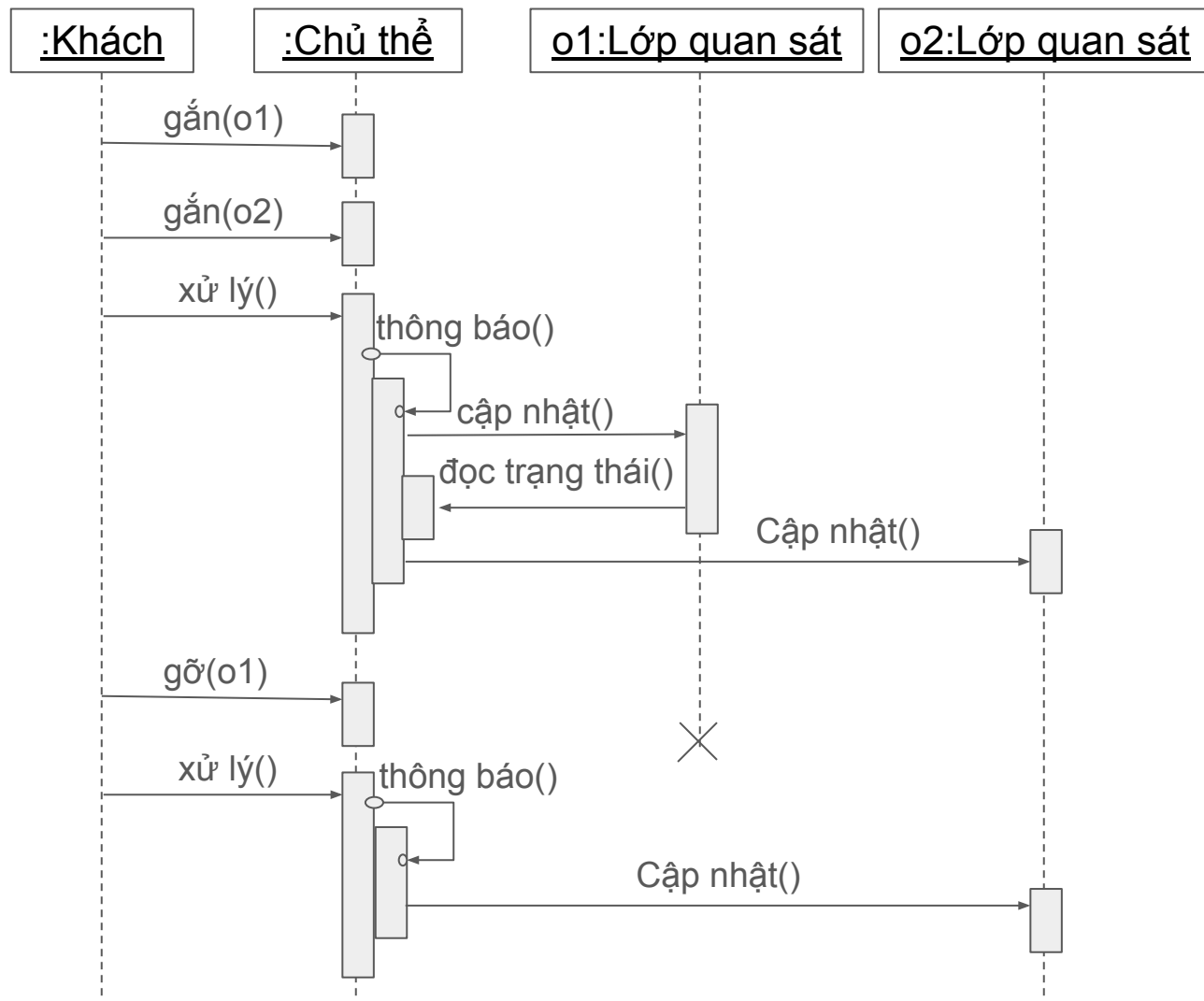
Observer

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



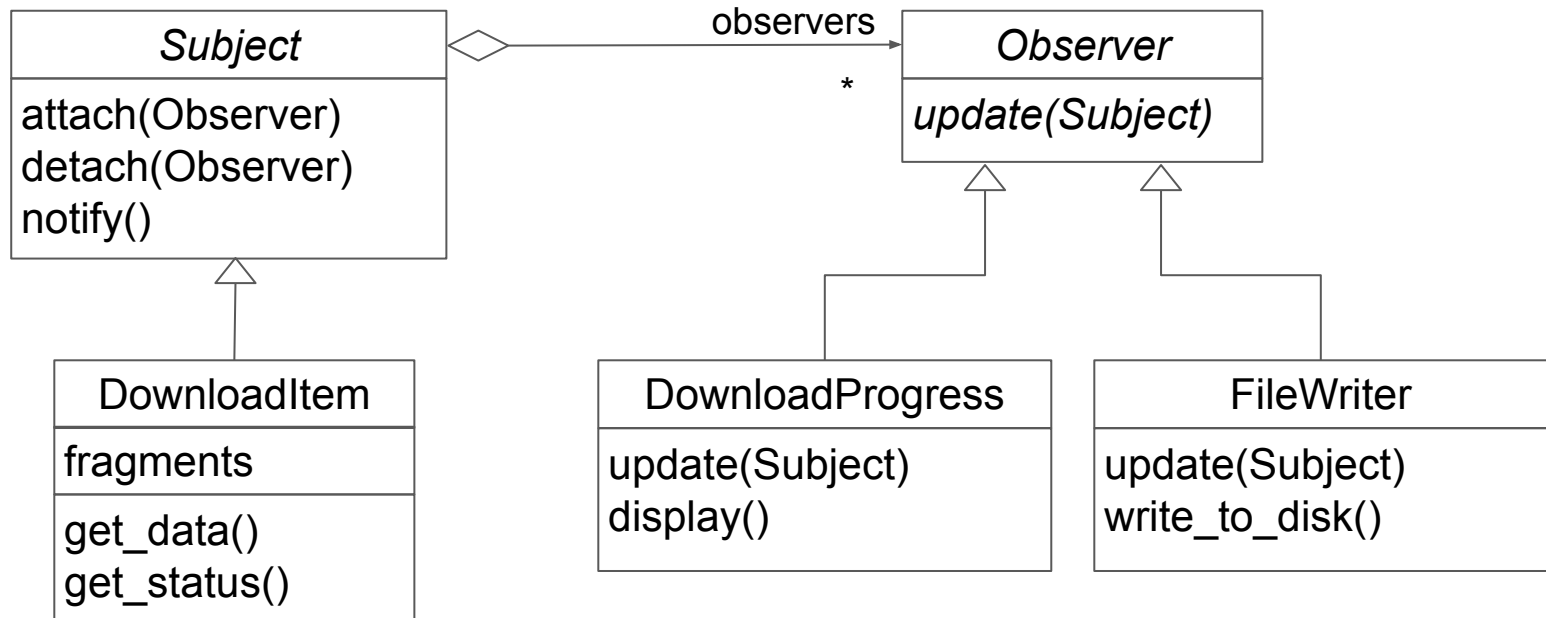
Lớp quan sát: Hành vi



Lớp quan sát: Các hệ quả

- Nói lòng 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

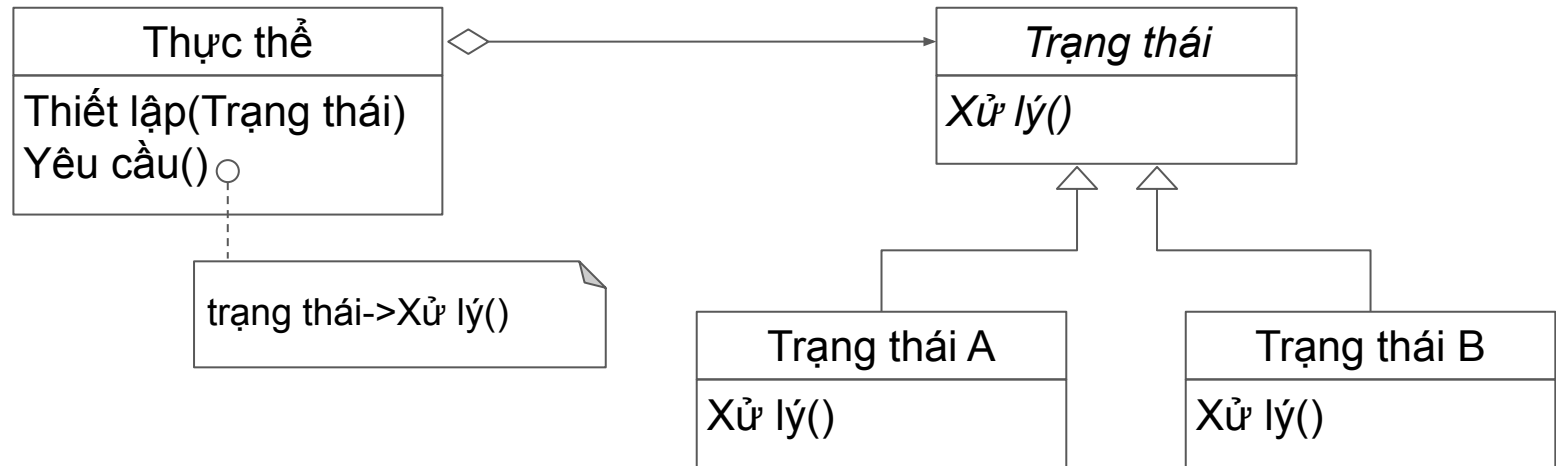


Trạng thái

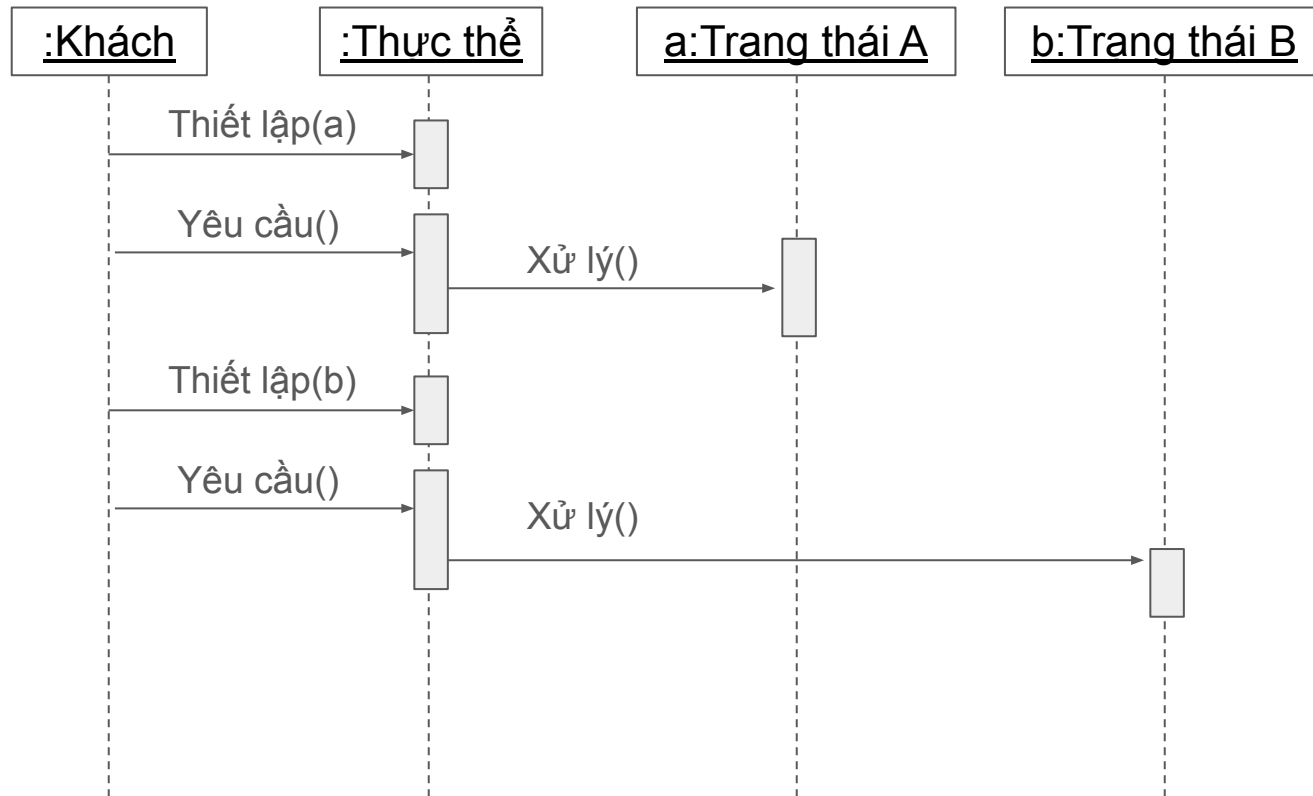
State

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



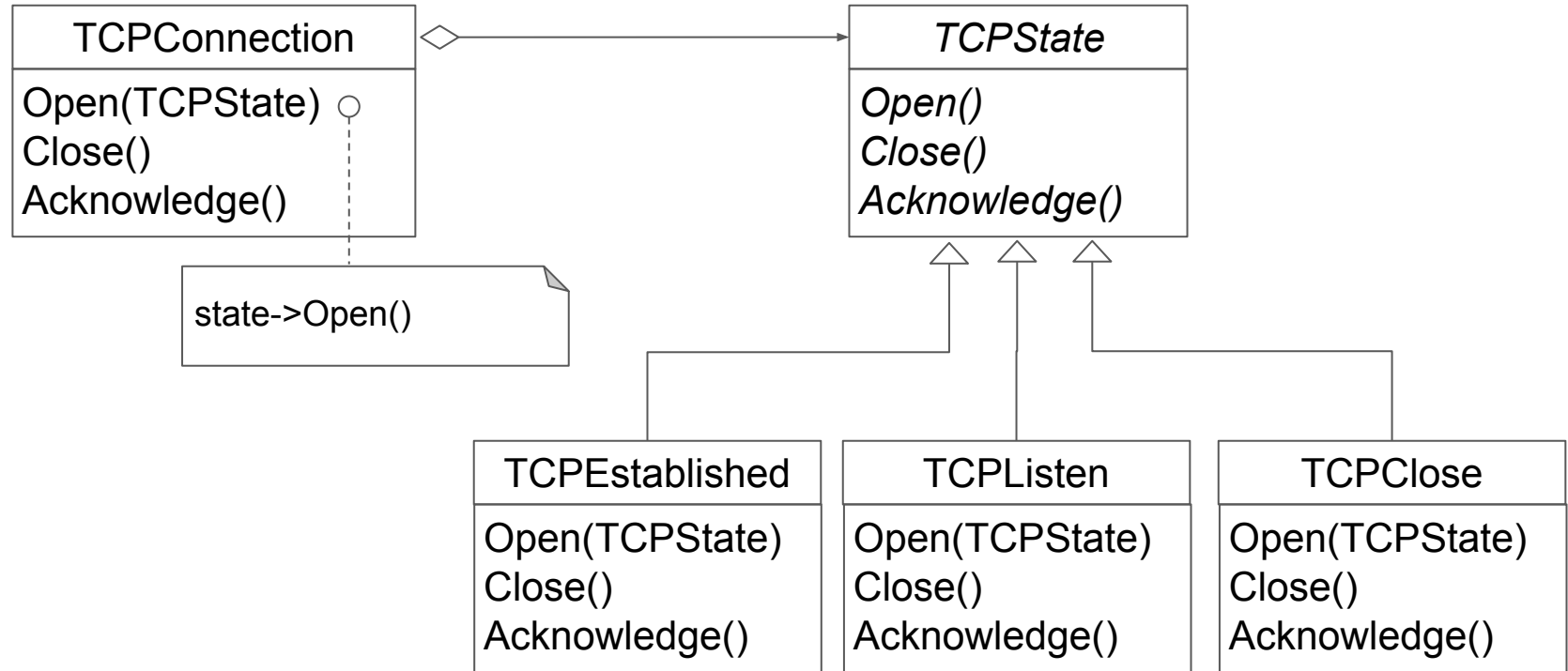
Trạng thái: Hành vi



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

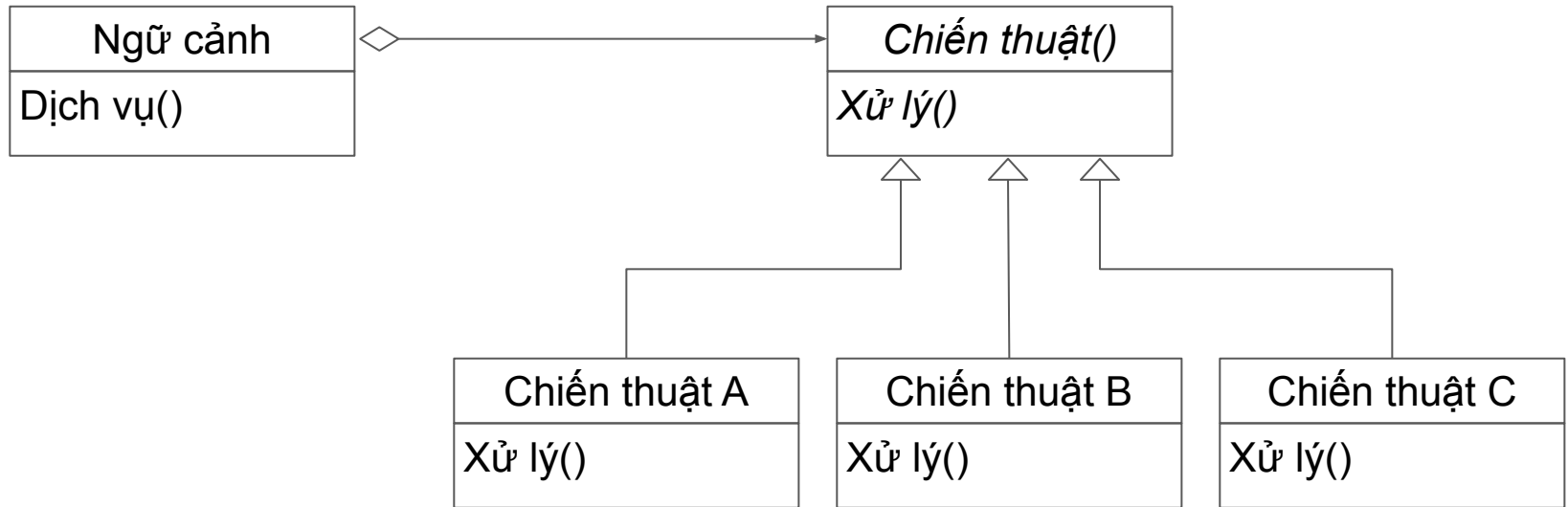


Chiến thuật

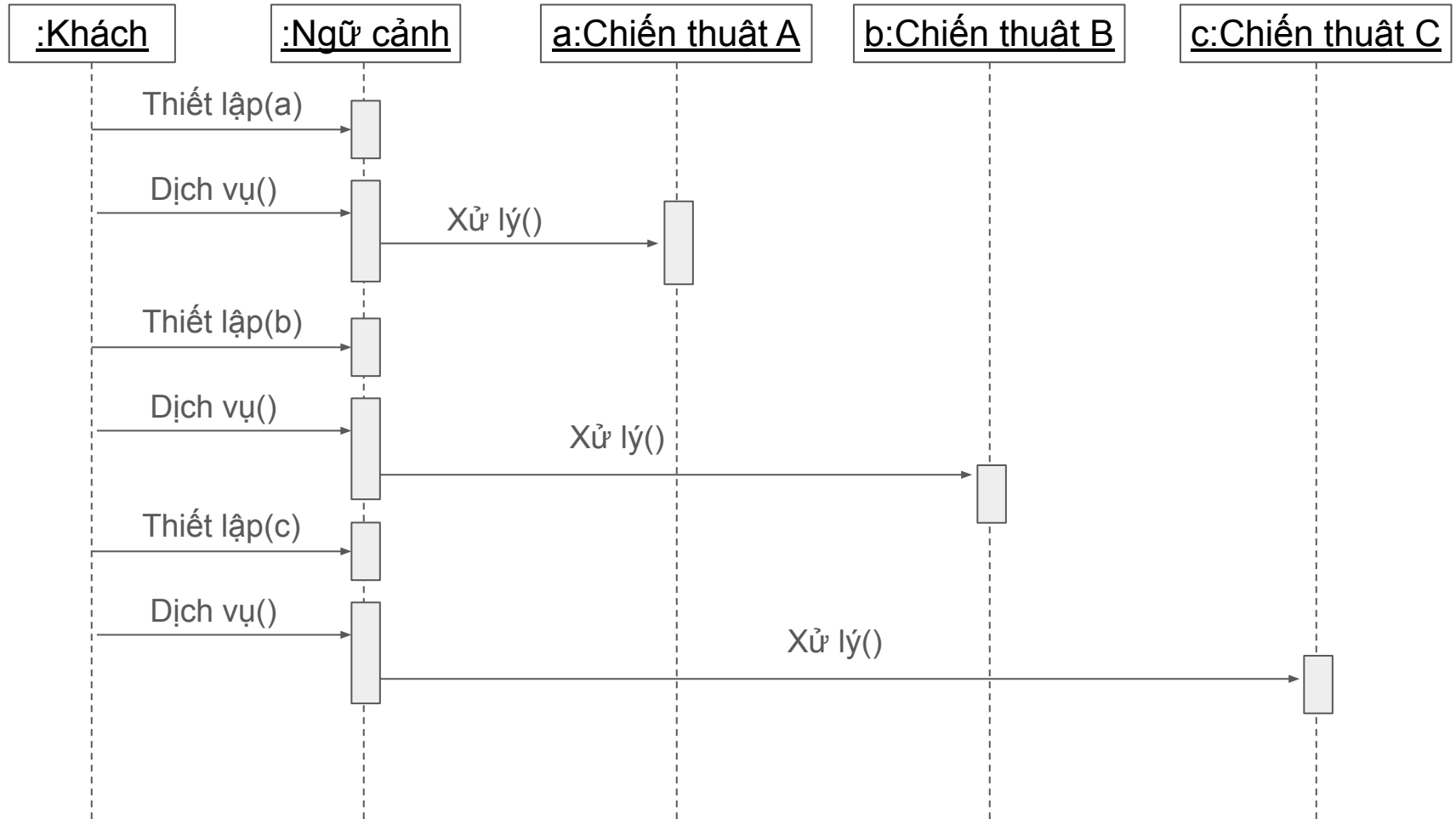
Strategy

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



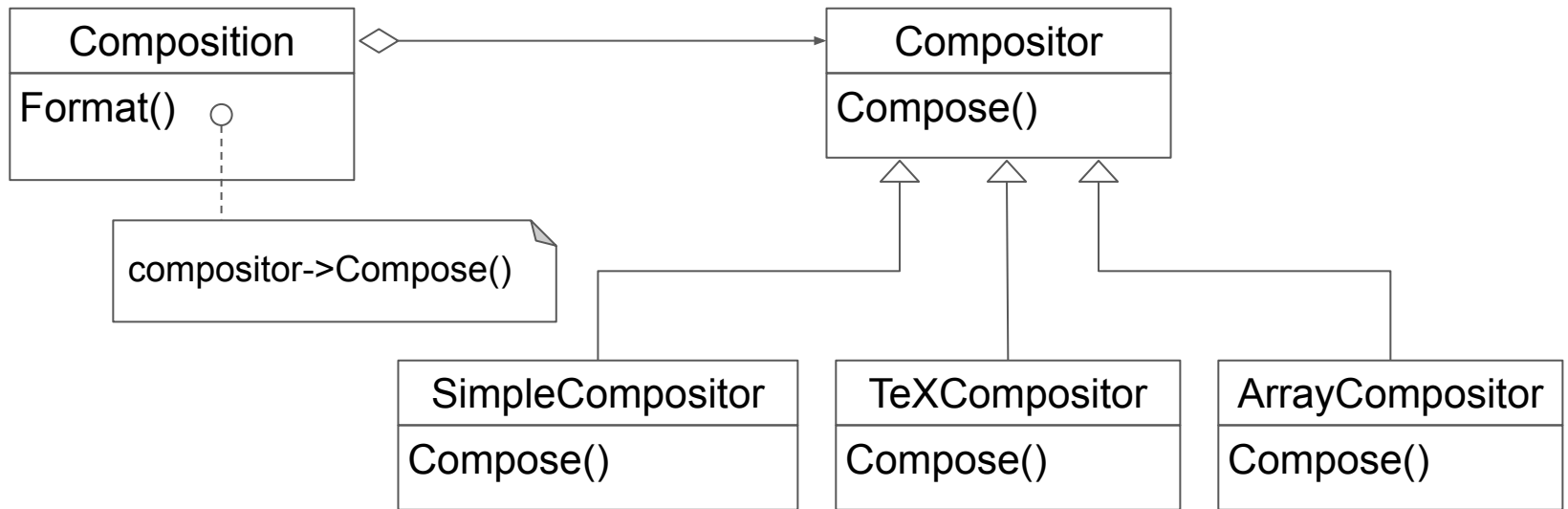
Chiến thuật: Hành vi



Chiến thuật: Các hệ quả

- Cây chiến thuật định nghĩa một hệ thuật toán để tái sử dụng theo ngữ cảnh.
- Đóng gói thuật toán trong lớp chiến thuật cho phép thay đổi thuật toán độc lập với ngữ 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 ngữ cảnh, vì vậy ngữ 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

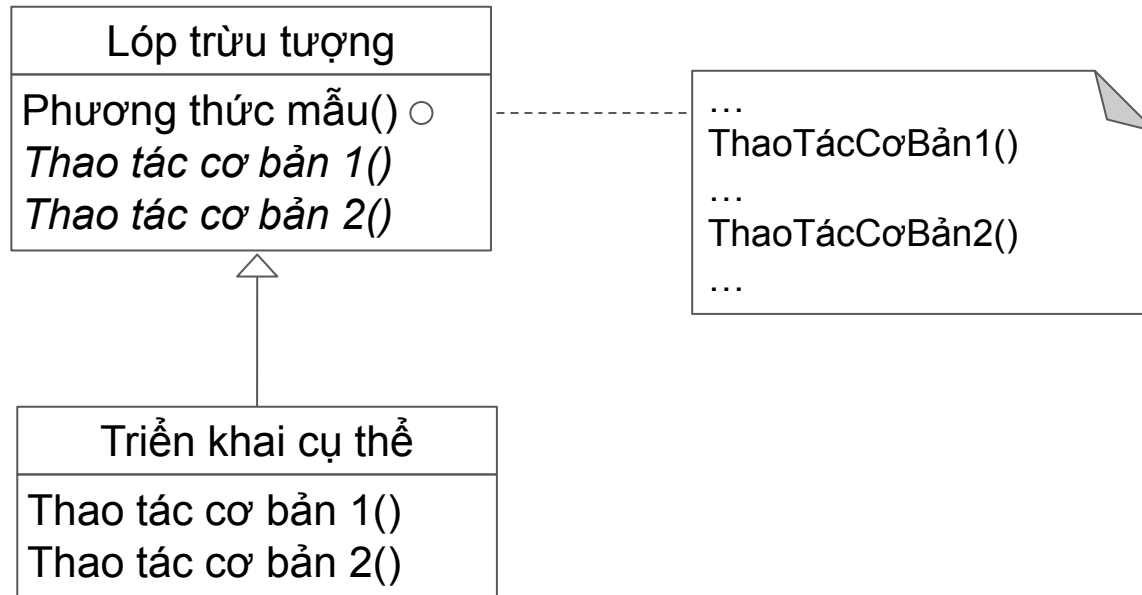


Phương thức mẫu

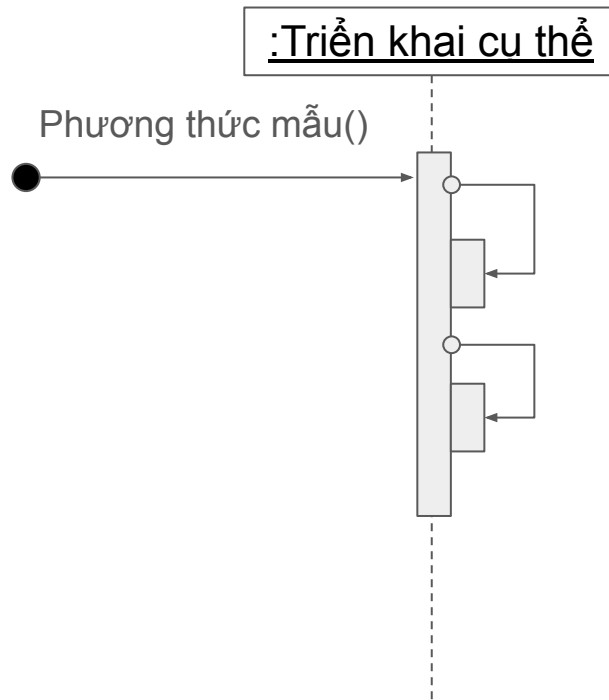
Template Method

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



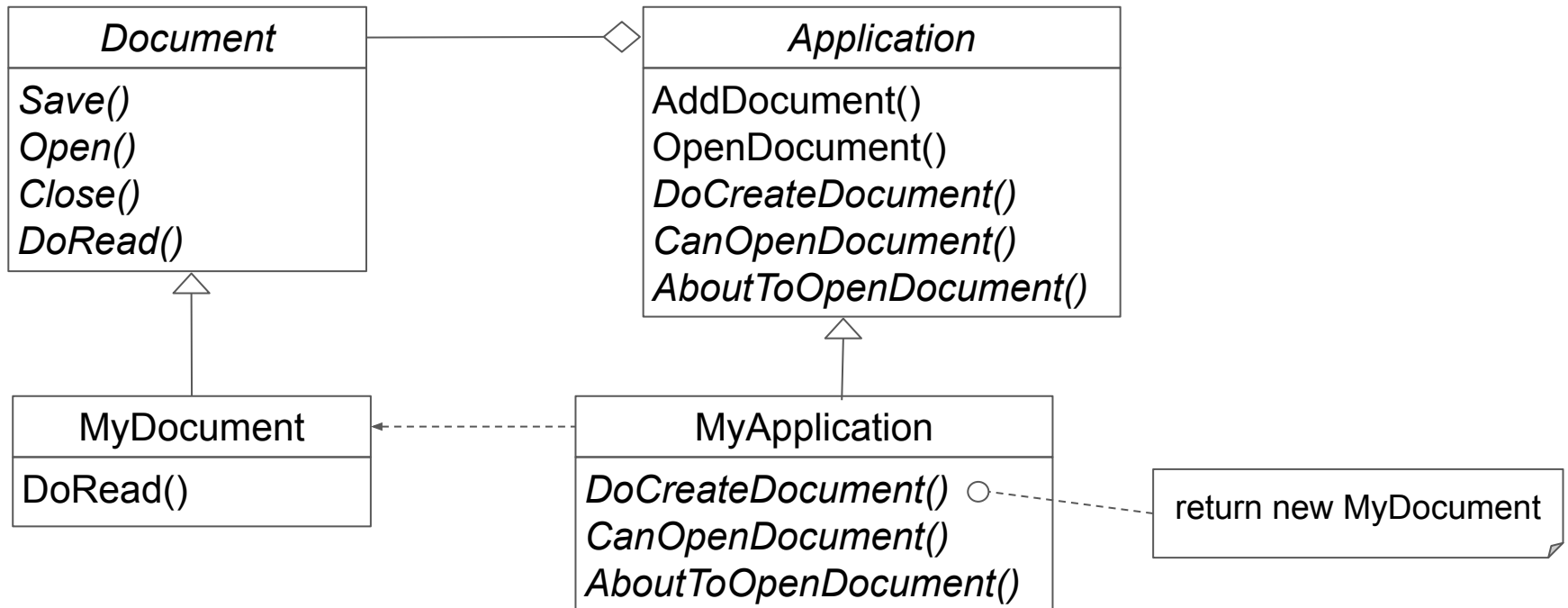
Phương thức mẫu: Hành vi



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

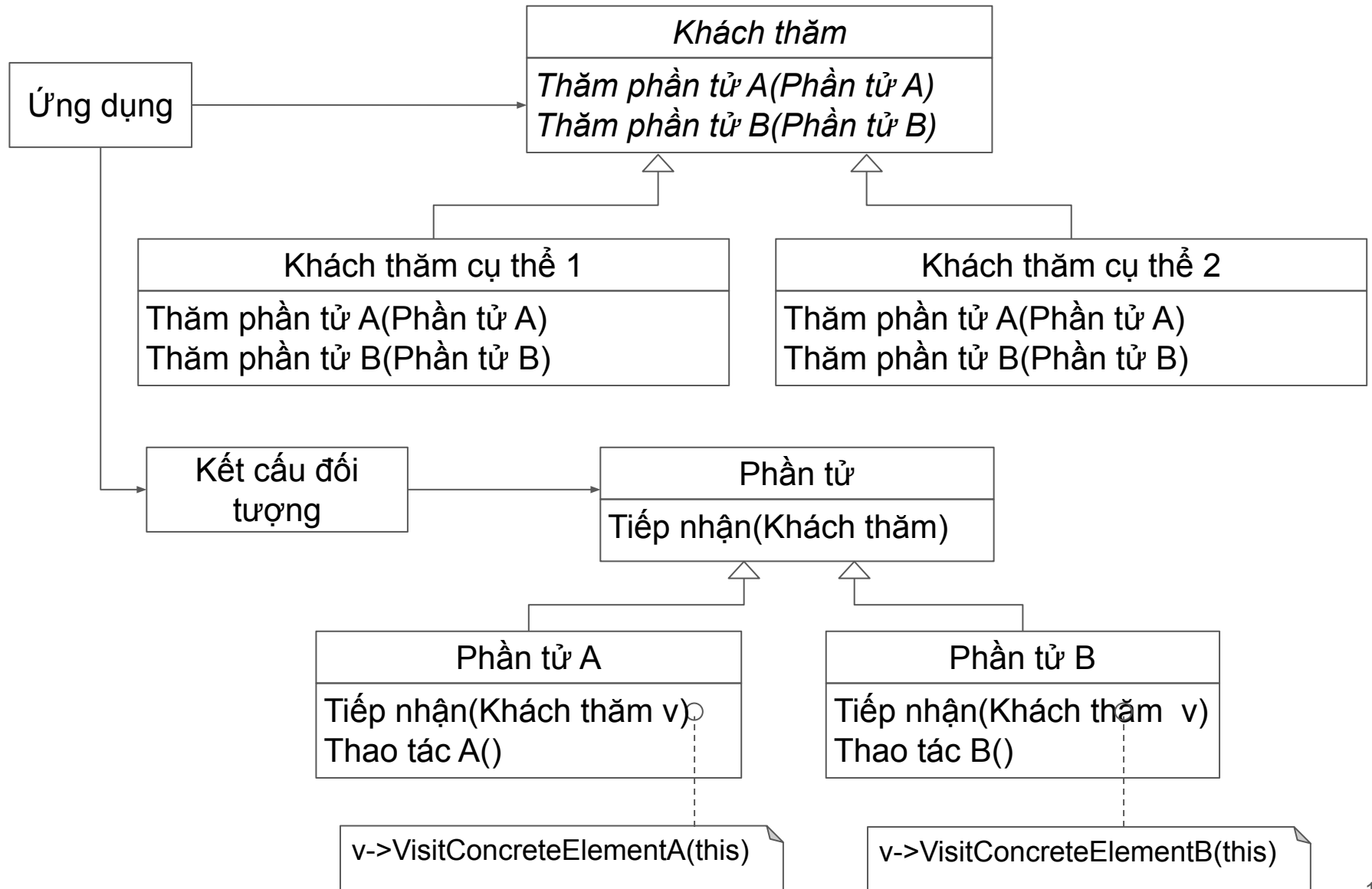


Khách thăm

Visitor

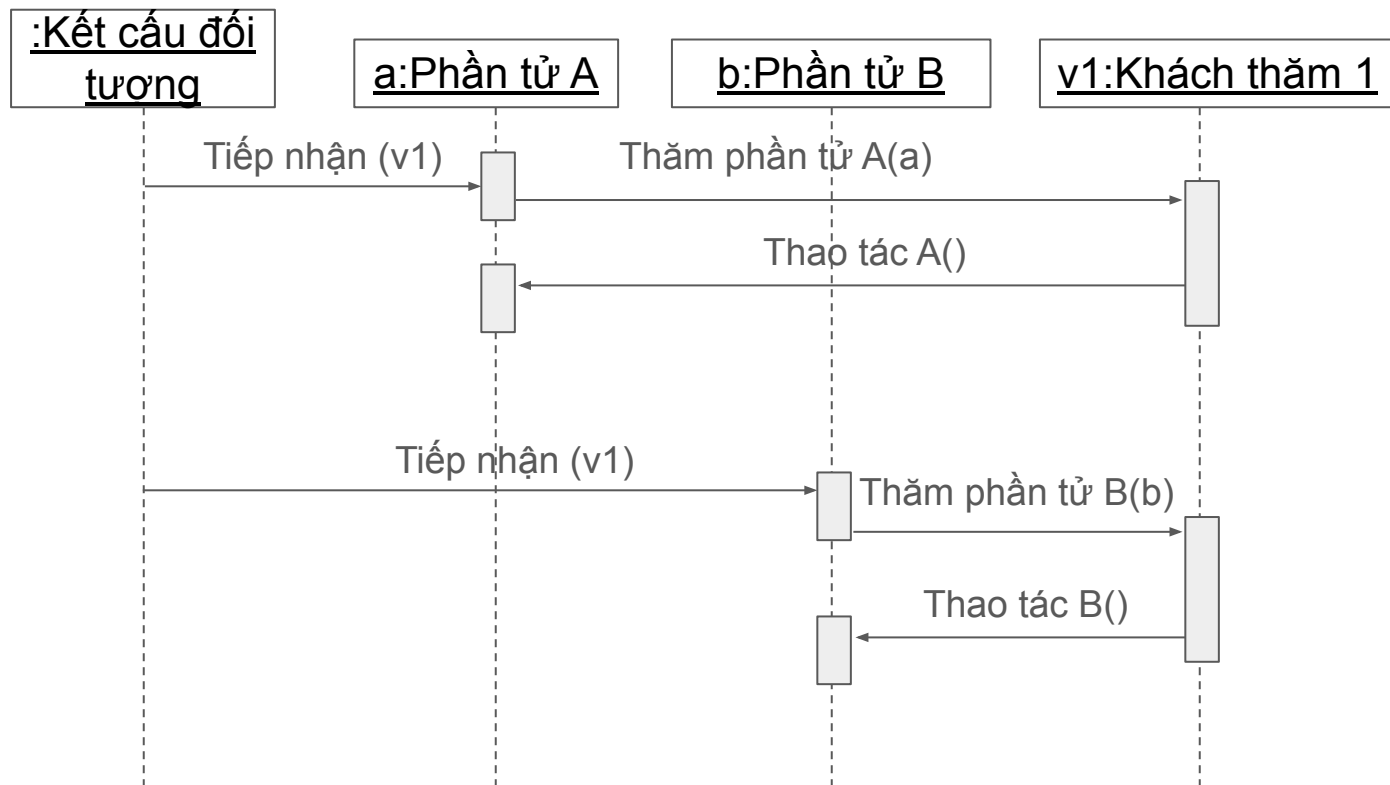
Biểu diễn một thao tác cần được thực hiện trên các đối tượng thuộc một kết cấu. Khách thăm cho phép định nghĩa một thao tác mới mà không thay đổi các lớp của các đối tượng chịu tác động.

Khách thăm: Cấu trúc



Khách thăm: Hành vi

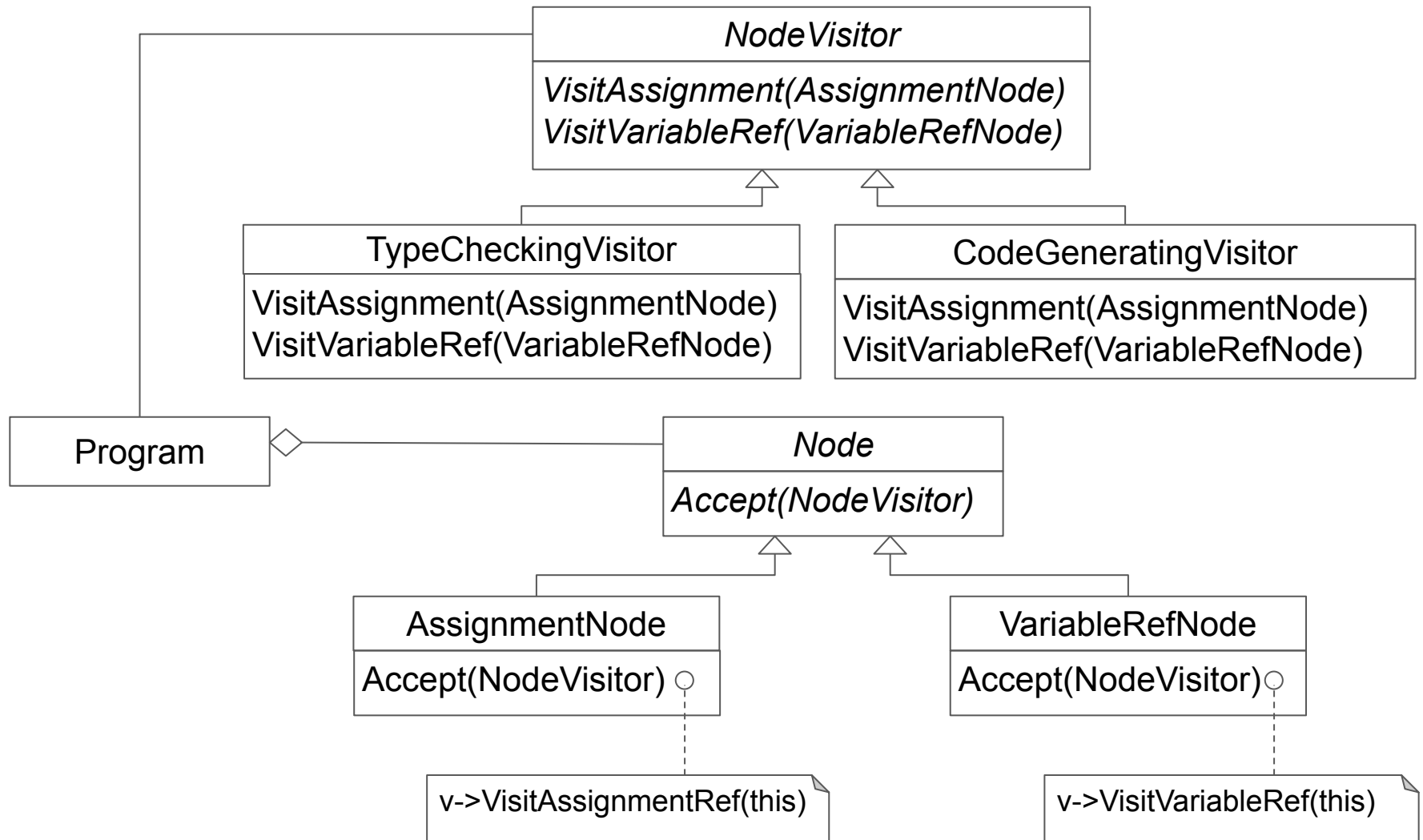
Tạo lớp v1 thay vì bổ xung phương thức A::v1() và B::v1()



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₍₂₎

- 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 ngữ 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₍₃₎

- 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