Structural pattern

Giảng viên: Huỳnh Tuấn Anh
Khoa CNTT - Đại học Nha Trang

Structural pattern

- Liên quan tới cách tổ chức các đối tượng để hình thành các cấu trúc lớn hơn
- Mô tả cách để kết hợp các đối tượng để thực hiện một chức năng mới

- Adapter pattern
- Bridge pattern
- Composite pattern
- Decorator pattern
- Façade pattern
- Proxy pattern

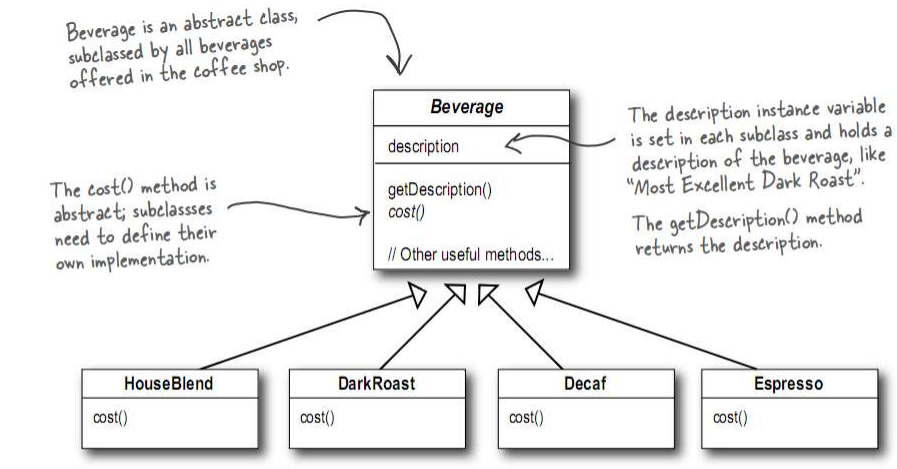
3

Decorator pattern

Example

5

Example: Cost of Beverage

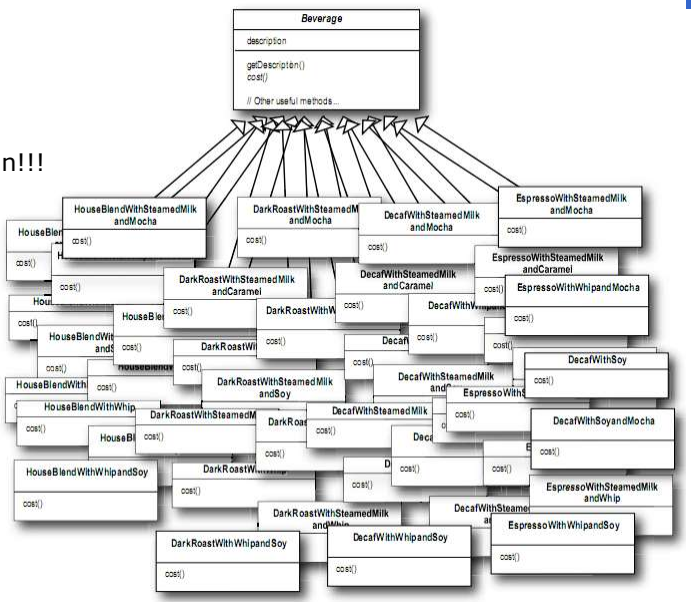


Thêm một số thành phần phụ vào 4 loại nước đã có ?

6

- 1st idea

Class explosion!!!



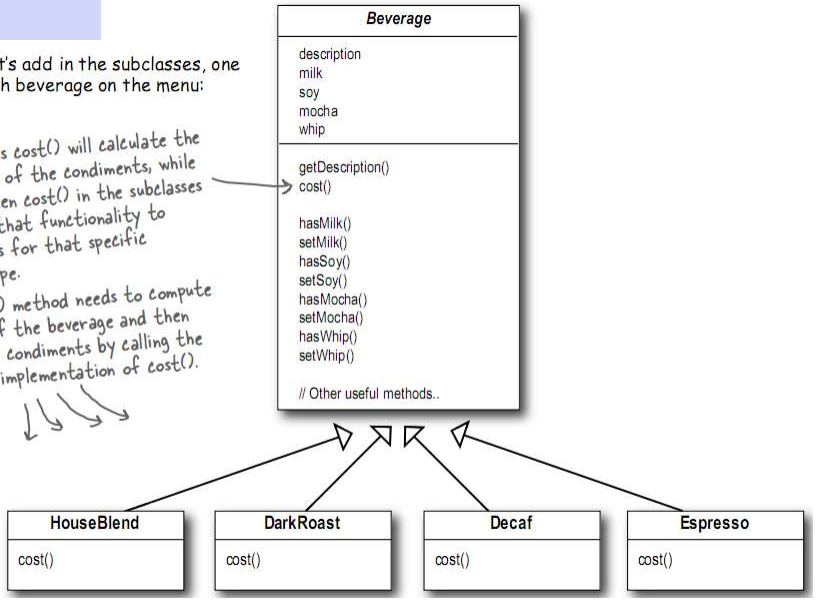
7

2nd idea

Now let's add in the subclasses, one for each beverage on the menu:

The superclass cost() will calculate the costs for all of the condiments, while the overridden cost() in the subclasses will extend that functionality to include costs for that specific beverage type.

Each cost() method needs to compute the cost of the beverage and then add in the condiments by calling the superclass implementation of cost().



8

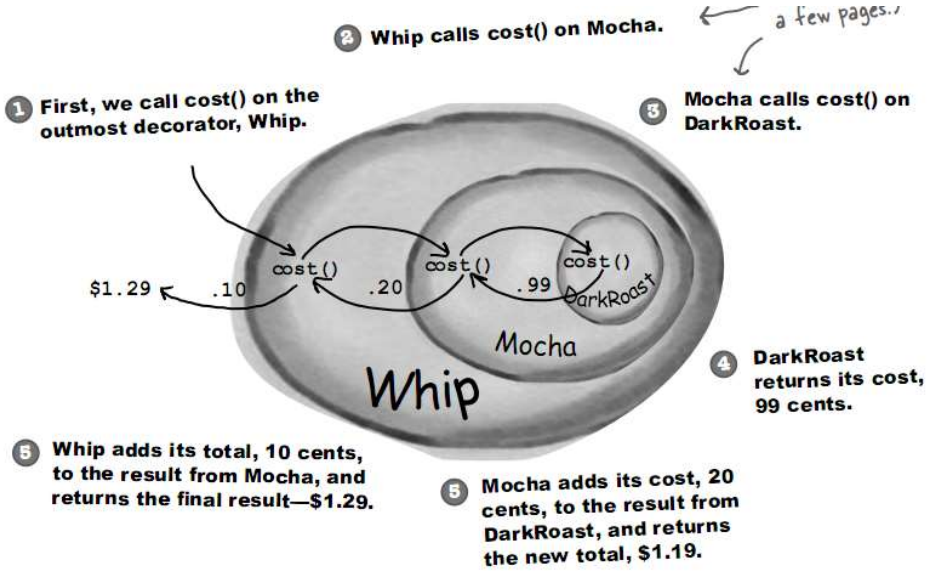
- The Open-Closed Principle:

Design Principle

Classes should be open for extension, but closed for modification.

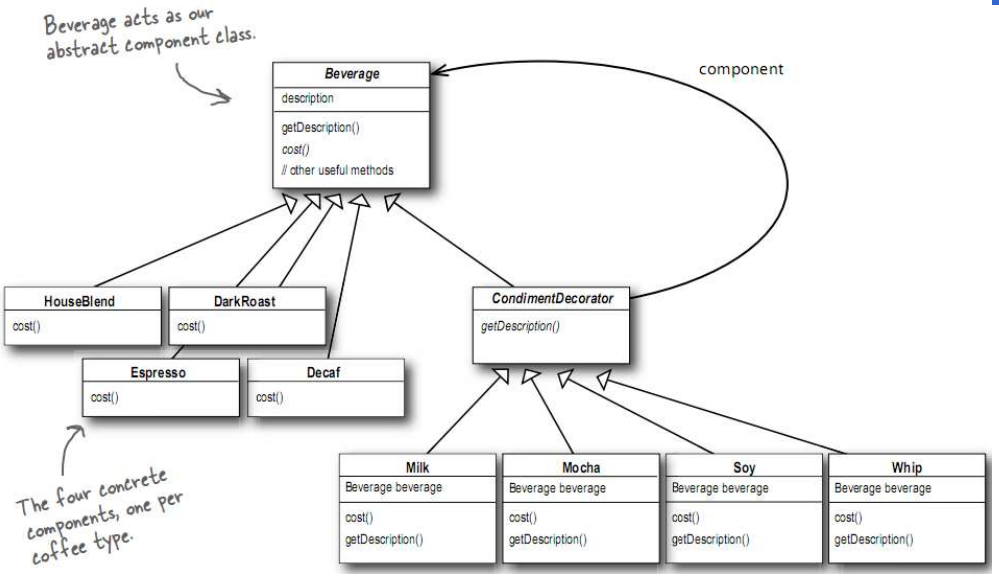
9

- 3rd idea



10

- 3rd idea...



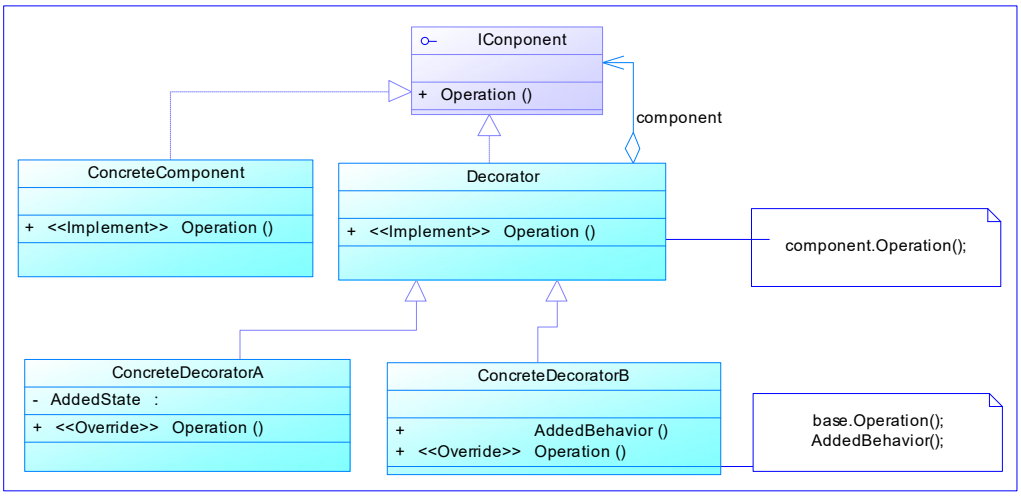
11

Decorator pattern

- Mục đích:
  - Cho phép thêm mới các trạng thái và hành vi vào một đối tượng lúc run-time bằng cách dùng kỹ thuật subclassing để mở rộng các chức năng của lớp.

12

- Cấu trúc



13

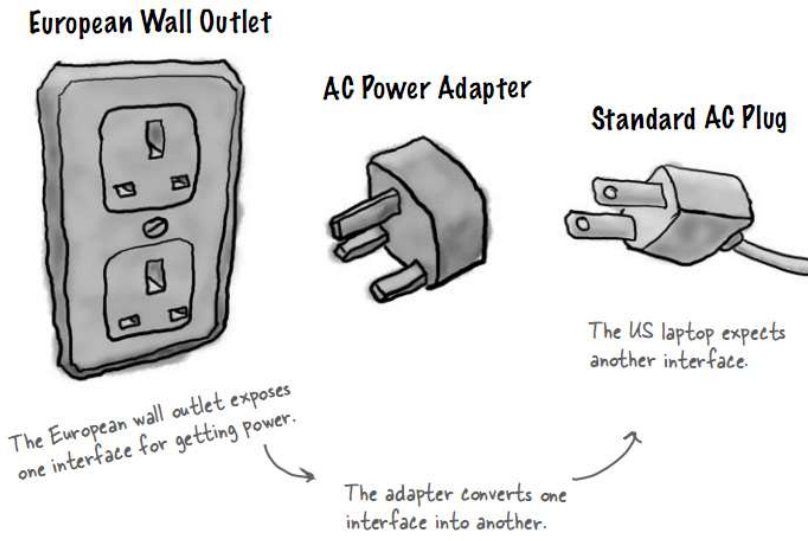
Questions

- Vai trò của lớp Decorator, có thể không cần dùng lớp Decorator được không?
- Nêu mối liên hệ giữa ConcreteComponent và ConcreteDecorator
- Trường hợp sử dụng của mẫu Decorator
  - Client sử dụng một component theo một interface không thay đổi nhưng muốn sử dụng các phiên bản mở rộng của component, nhưng:
    - Không thể mở rộng thành phần component bằng cách thừa kế, hoặc:
    - Việc mở rộng thành phần component có thể dẫn đến việc bùng nổ lớp

14

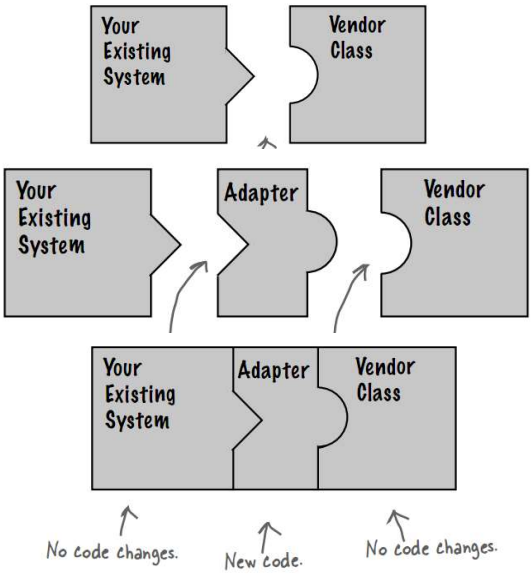
Adapter pattern

Adapter pattern: Example



16

Adapter pattern: Example



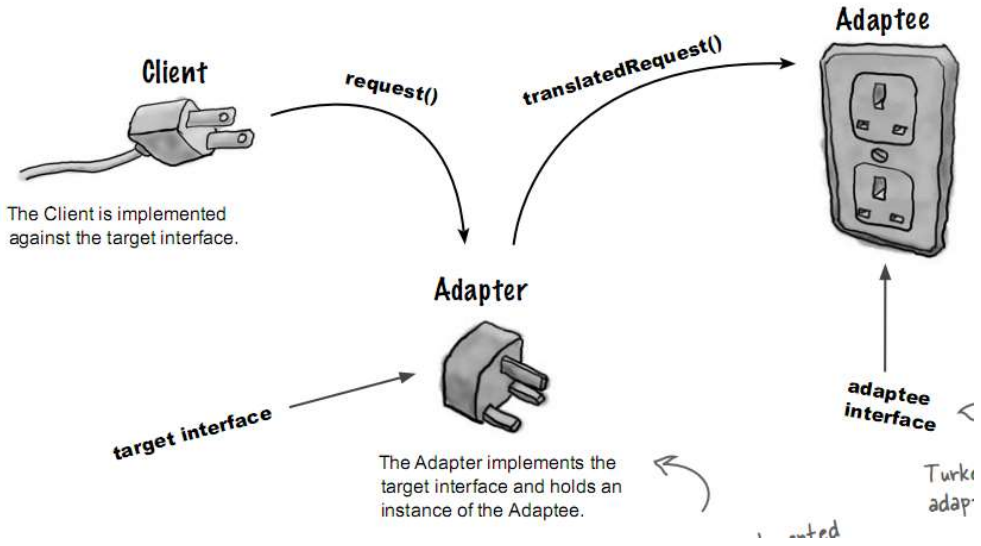
17

Adapter pattern

- Mục đích:
  - Chuyển giao diện của một lớp thành một giao diện khác mà client sử dụng
  - Cho phép các lớp có giao diện không tương thích cùng làm việc với nhau.

18

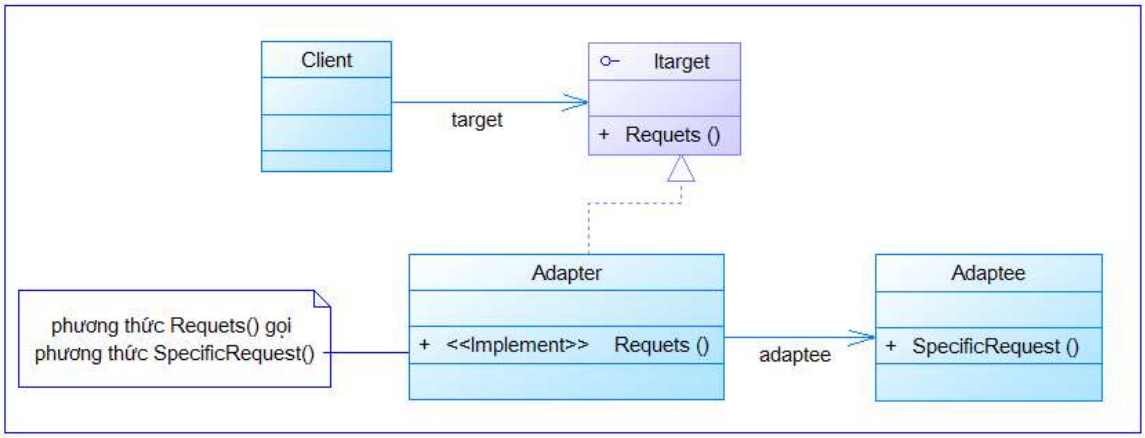
Adapter pattern: Idea



19

Adpater pattern

- Cấu trúc:



20

Questions

- Vai trò của lớp Adapter ?
- Adapter chỉ được sử dụng cho một lớp Adaptee duy nhất?
- Trường hợp sử dụng của mẫu Adapter:
  - Giao diện của Client và Library không tương thích nhau và không thể cập nhật cả Client lẫn Library.

21

Façade pattern

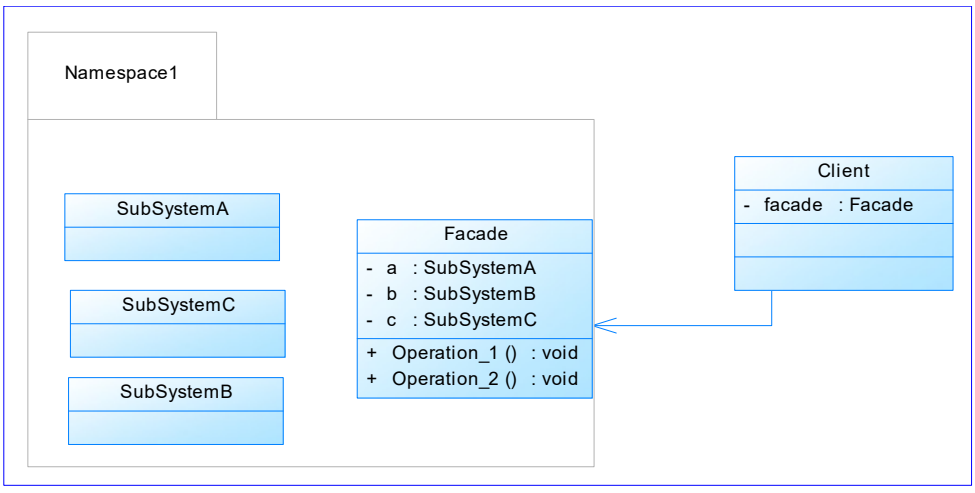
Façade pattern

- Mục đích
  - Cung cấp một interface hợp nhất cho một tập các interface trong một subsystem
  - Định nghĩa một interface ở mức cao làm cho việc sử dụng subsystem trở nên dễ dàng hơn

23

Façade pattern

- Cấu trúc:



24

Trường hợp sử dụng của mẫu Façade

- Hệ thống có nhiều hệ thống con, mỗi hệ thống con được truy cập theo những giao diện khác nhau gây sự khó hiểu/phức tạp khi truy cập các hệ thống này.
- Cần phải đưa ra một giao diện đơn giản để truy cập một hệ thống bao gồm nhiều hệ thống con phức tạp.
- Cần thiết lập các "cổng truy cập riêng" cho các client khác nhau.

25

- Each unit should only talk to its friends; don't talk to strangers.
- Reduce the interactions between objects to just a few close "friend

Design Principle

Principle of Least Knowledge - talk only to your immediate friends.

26

Proxy pattern

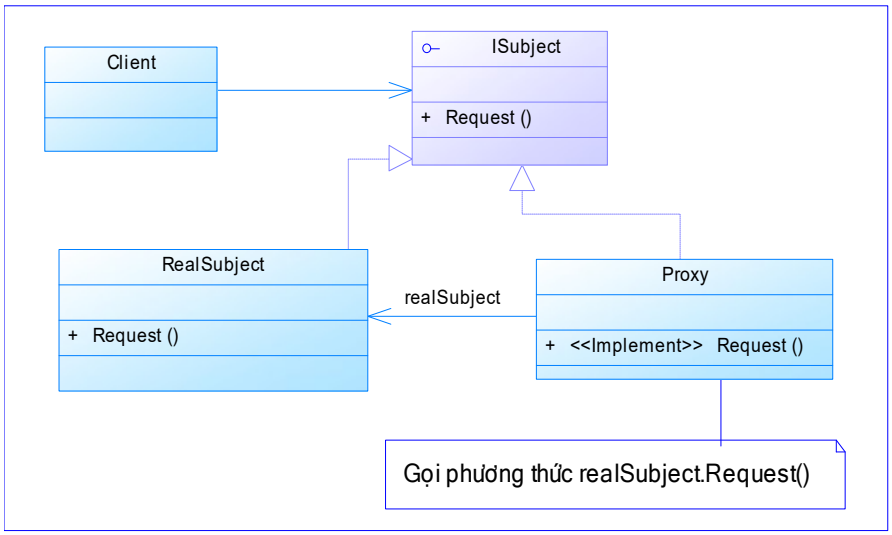
Proxy pattern

- Mục đích:
  - Cung cấp một đối tượng thay thế hay một trình giữ chỗ cho một đối tượng khác để kiểm soát việc truy cập tới đối tượng đó.
  - Sử dụng Proxy pattern để tạo một đối tượng đại diện để kiểm soát truy cập tới một đối tượng khác:
    - Ở xa
    - Expensive to create
    - Cần được bảo vệ

28

Proxy pattern

- Cấu trúc

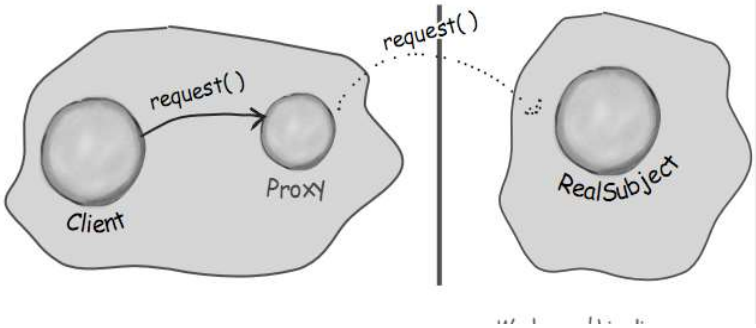


29

Remote proxy

Remote Proxy

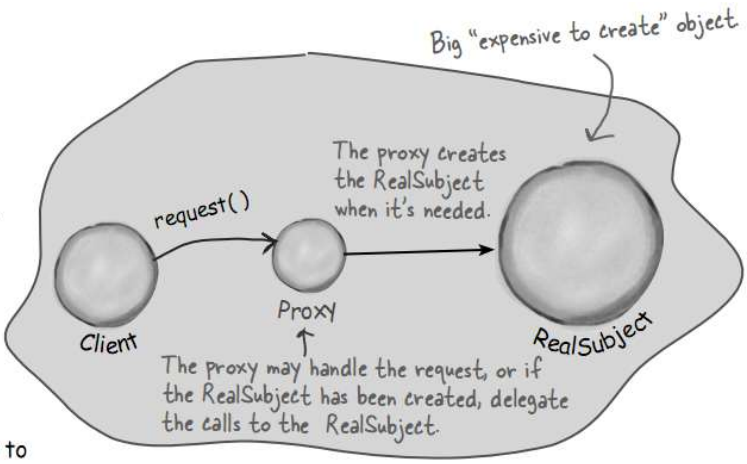
With Remote Proxy, the proxy acts as a local representative for an object that lives in a different JVM. A method call on the proxy results in the call being transferred over the wire, invoked remotely, and the result being returned back to the proxy and then to the Client.



30

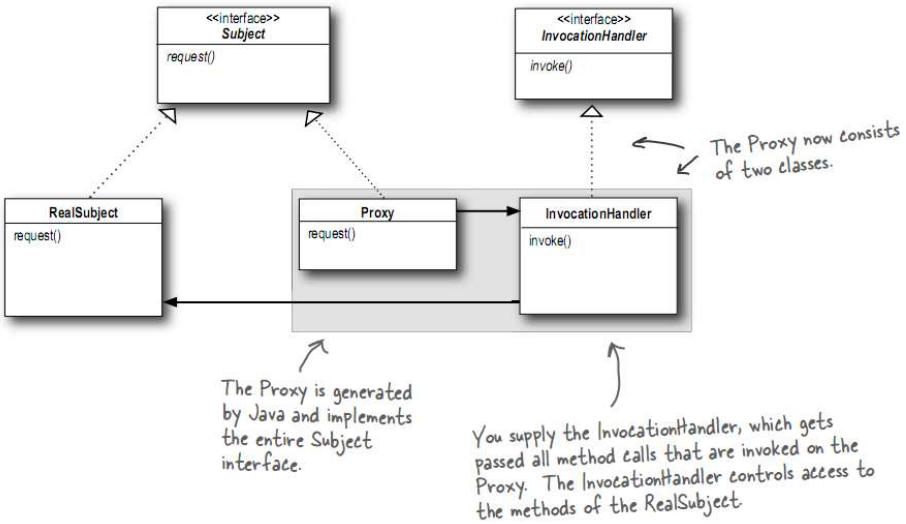
Virtual Proxy

Virtual Proxy acts as a representative for an object that may be expensive to create. The Virtual Proxy often defers the creation of the object until it is needed; the Virtual Proxy also acts as a surrogate for the object before and while it is being created. After that, the proxy delegates requests directly to the RealSubject.



31

Protection proxy



32

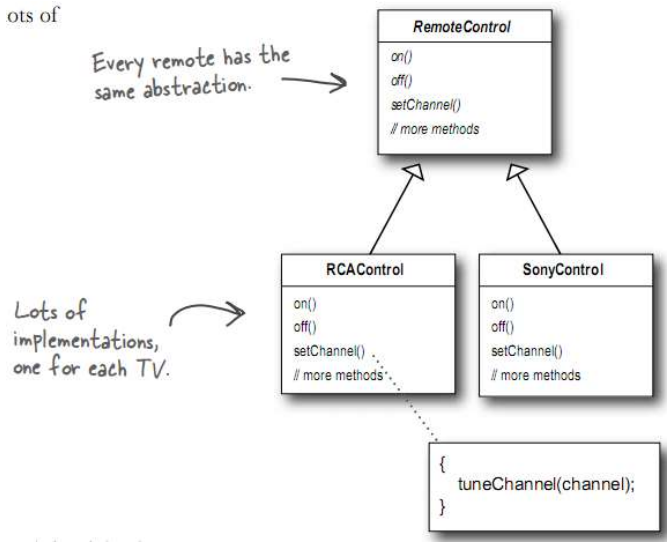
Questions

33

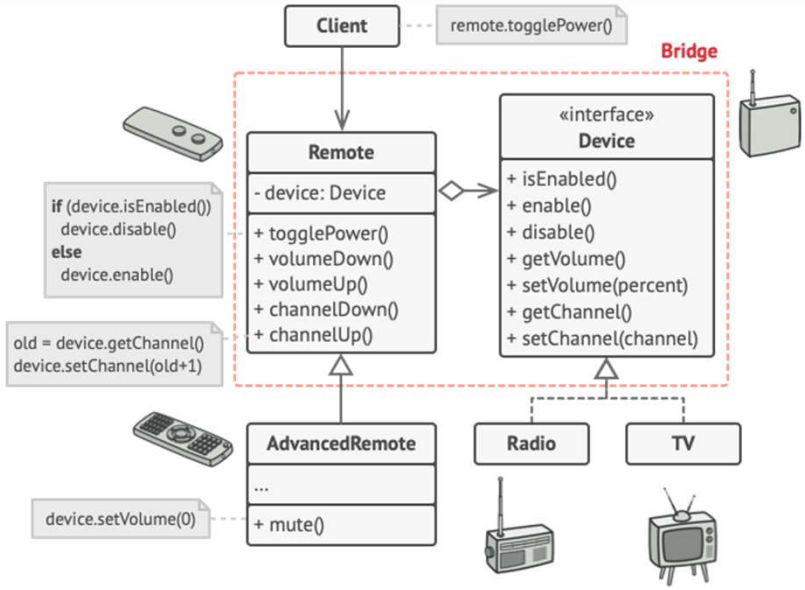
Bridge pattern

Example: Remote Control

- Các loại remote TV có cùng chung một giao diện trừu tượng và nhiều cách thực thi khác nhau cho nhiều loại TV
- Vấn đề:
  - Remote lần đầu tiên thiết kế không chuẩn và cần phải cải tiến
    - Cả TV và Remote đều thay đổi



35



36

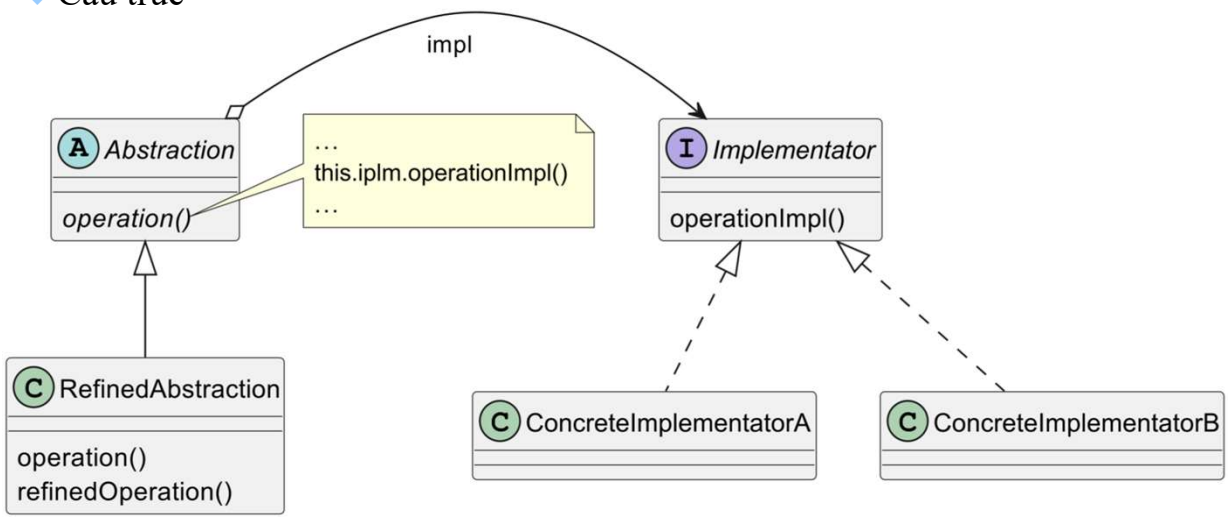
Bridge pattern

- Mục đích:
  - Tách rời phần trừu tượng ra khỏi sự thực thi của nó sao cho cả hai có thể biến đổi không phụ thuộc nhau

37

Bridge pattern

- Cấu trúc



38

Bridge pattern

- Ưu điểm:
  - Phần thực thi và phần sử dụng không trực tiếp kết nối với nhau mà được kết nối với nhau bằng các thành phần trừu tượng (Abstract class, interface)
  - Phần Abstraction và phần Implementation có được thể mở rộng không phụ thuộc nhau
  - Thay đổi các lớp concrete abstraction không ảnh hưởng đến client
- Hạn chế:
  - Gia tăng sự phức tạp

39

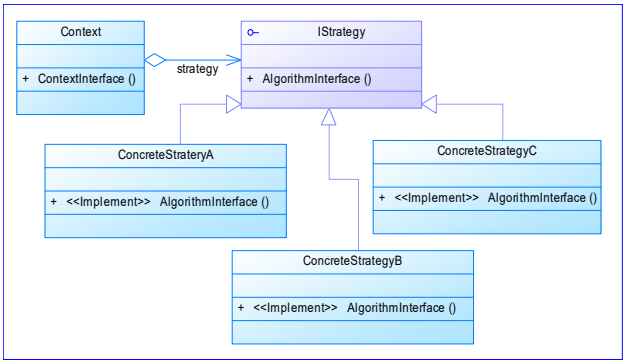
Trường hợp sử dụng

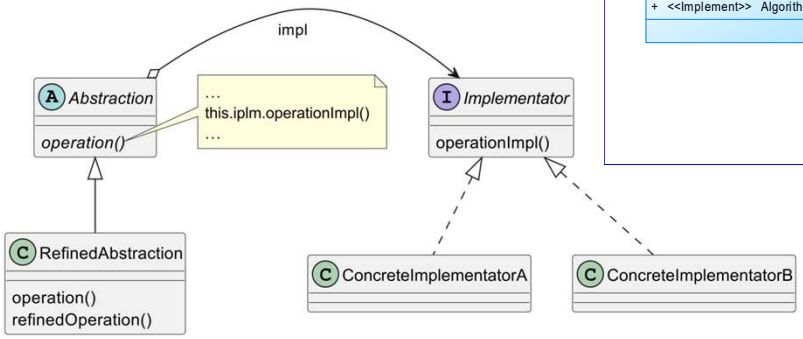
- Tách rời phần trừu tượng và phần thực thi để có thể mở rộng một cách độc lập
- Các phần trừu tượng và thực thi không thể được quyết định lúc compile time.
- Các sự thay đổi ở phần trừu tượng không ảnh hưởng đến ứng dụng
- Tách rời phần thực thi chi tiết ra khỏi client

40

So sánh Bridge và Stratery

- Sơ đồ gần giống nhau nhưng mục đích sử dụng khác nhau





41

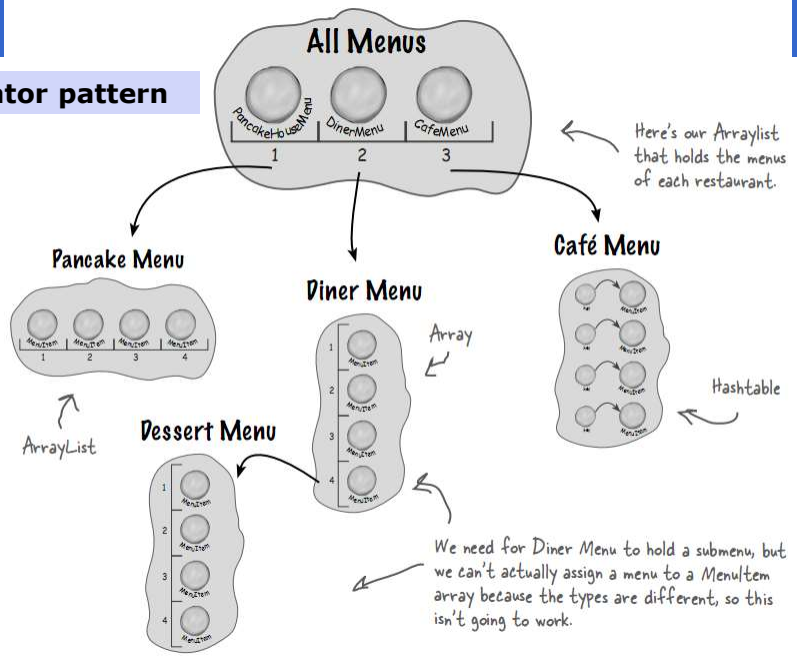
So sánh Bridge và Stratery

- Mục đích khác nhau
  - Stratery: Cho phép chọn lựa một thuật toán cụ thể trong một họ các thuật toán
  - Bridge: Cho phép tách rời phần trừu tượng và việc thực thi, cho phép cả hai đều có thể mở rộng một cách độc lập.
- Sử dụng ở client:
  - Bridge: Sử dụng thông qua lớp được mở rộng
  - Stratery: Sử dụng thông qua thành phần Context

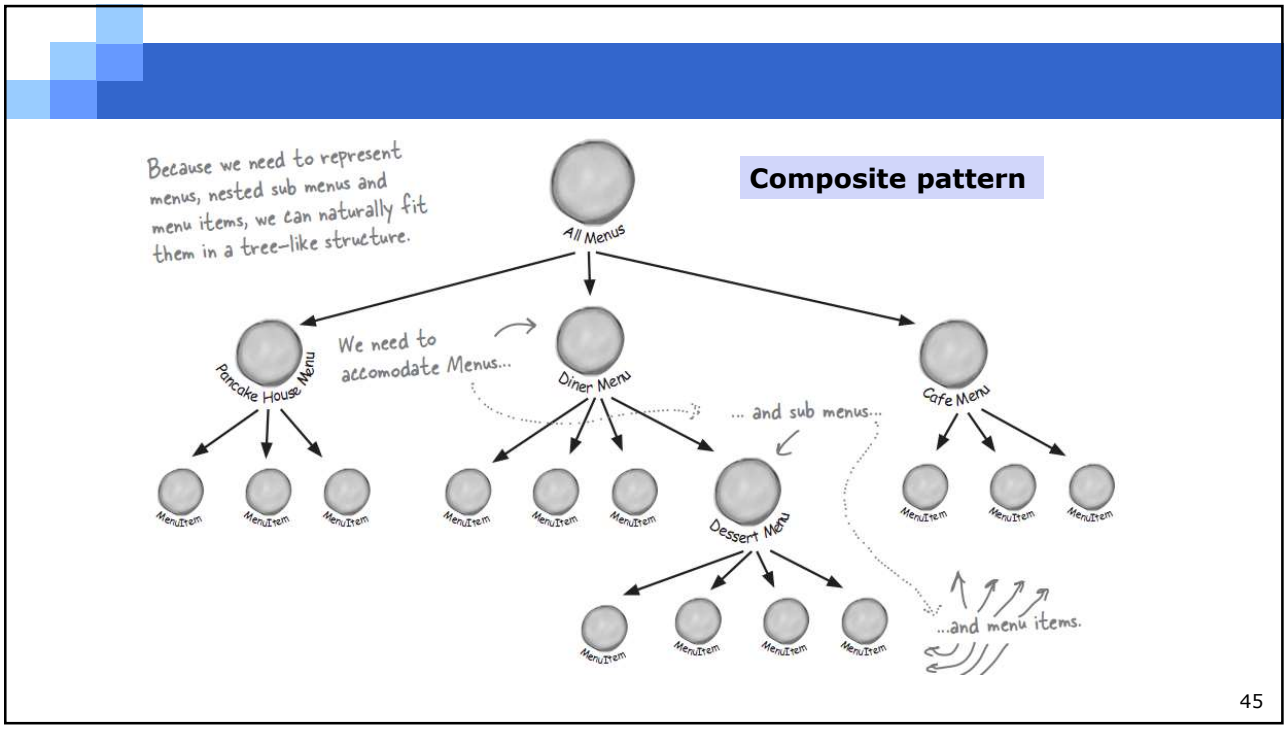
42

Composite pattern

Iterator pattern



44



45

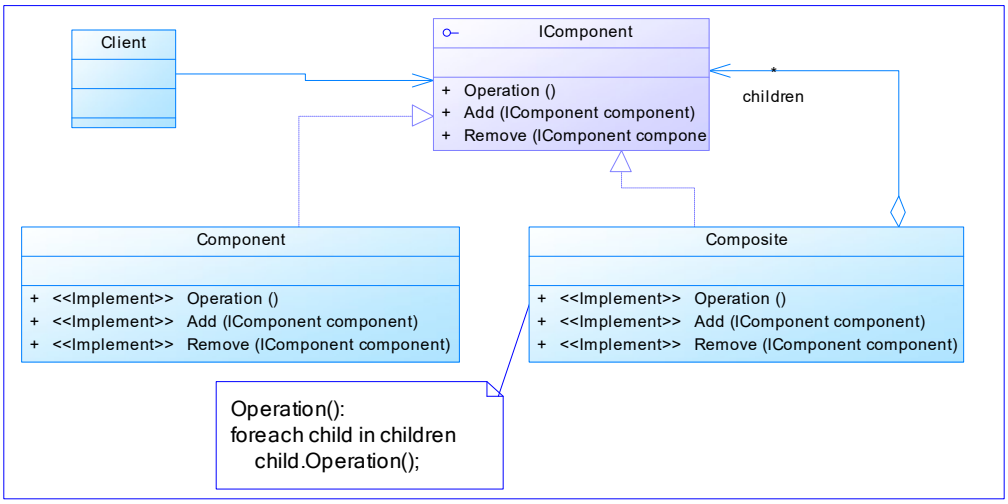
Composite pattern

- Mục đích:
  - Sắp xếp các đối tượng vào các cấu trúc cây để biểu diễn các phân cấp part-whole giữa các đối tượng
  - Đối xử với các đối tượng riêng lẻ và nhóm các đối tượng theo các cách giống nhau

46

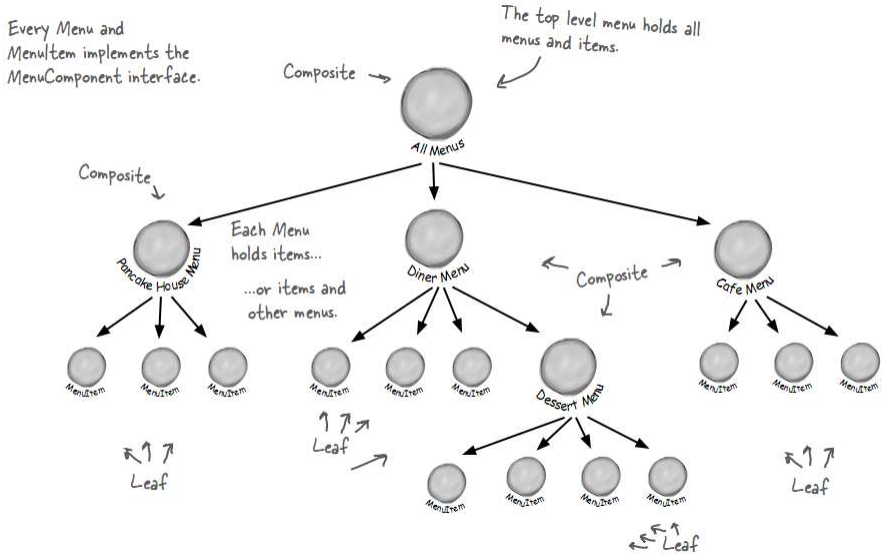
Composite pattern

- Cấu trúc:



47

Composite pattern: Example



48

Composite pattern

- Composite Pattern cho phép xây dựng các cấu trúc của các đối tượng dưới dạng cây với các node là:
  - composition of objects
  - individual objects
- Sử dụng cấu trúc Composite ta có thể áp dụng cùng một Operation cho cả hai loại đối tượng Composite và Individual → Làm mất đi sự khác biệt đối với hai loại đối tượng

49

Questions

- So sánh giữa Iterator Pattern và Composite Pattern. Có thể thay thế Composite Pattern bằng Iterator Pattern được không.
- Trường hợp sử dụng:
  - Tổ chức các đối tượng riêng lẻ theo cấu trúc phân cấp, và:
  - Client truy cập truy cập các thành phần riêng lẻ và phức hợp theo cách giống nhau

50

Tài liệu tham khảo

- Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates. Head First Design pattern. O'Reilly 2006.
- **Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides**. Design Patterns Elements of Reusable Object-Oriented Software. Addison-Wesley 1995
- http://www.dofactory.com/Patterns/Patterns.aspx

51