



Behavioral patterns

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

04/09/2024



Behavioral patterns

- ❖ Liên quan tới các giải thuật và các quan hệ giữa các đối tượng, cách các đối tượng giao tiếp với nhau

04/09/2024

Behavioral patterns

- ❖ Chain of Responsibility.
- ❖ Command pattern
- ❖ Interpreter pattern
- ❖ Iterator pattern
- ❖ Mediator pattern
- ❖ Memento pattern
- ❖ Observer pattern
- ❖ State pattern
- ❖ Strategy pattern
- ❖ Template Method
- ❖ Visitor pattern

3



Observer pattern

04/09/2024

observer pattern

- ❖ Mục đích: Định nghĩa một phụ thuộc one-to-many giữa các đối tượng sao cho khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc nó được thông báo và được cập nhật một cách tự động.

5

04/09/2024

Các ví dụ về observer pattern

- ❖ Chương trình bảng tính Excel
- ❖ Data binding
- ❖ Đặt mua báo dài hạn
 - Độc giả đăng ký mua báo dài hạn với tòa soạn
 - Khi có một tờ báo mới xuất bản thì nó được đại lý phân phối đến độc giả.
 - **Publishers + Subscribers = Observer Pattern**



6

Five minute drama: a subject for observation

In today's skit, two post-bubble software developers encounter a real live head hunter...

04/09/2024

1 Software Developer #1

This is Ron, I'm looking for a Java development position, I've got five years experience and...

2 Headhunter/Subject

Uh, yeah, you and everybody else, baby. I'm putting you on my list of Java developers, don't call me, I'll call you!

3 Software Developer #2

Hi, I'm Jill, I've written a lot of EJB systems, I'm interested in any job you've got with Java development.

4 Subject

I'll add you to the list, you'll know along with everyone else.

7

5 Meanwhile for Ron and Jill life goes on: if a Java job comes along, they'll get notified, after all, they are observers.

04/09/2024

6 Subject

Hey observers, there's a Java opening down at JavaBeans-R-Us, jump on it! Don't blow it!

Swahaha, money in the bank, baby!

7 Observer

Thanks, I'll send my resume right over.

Observer

This guy is a real jerk, who needs him. I'm looking for my own job.

8 Observer

Jill lands her own job!

You can take me off your call list, I found my own job!

9 Subject

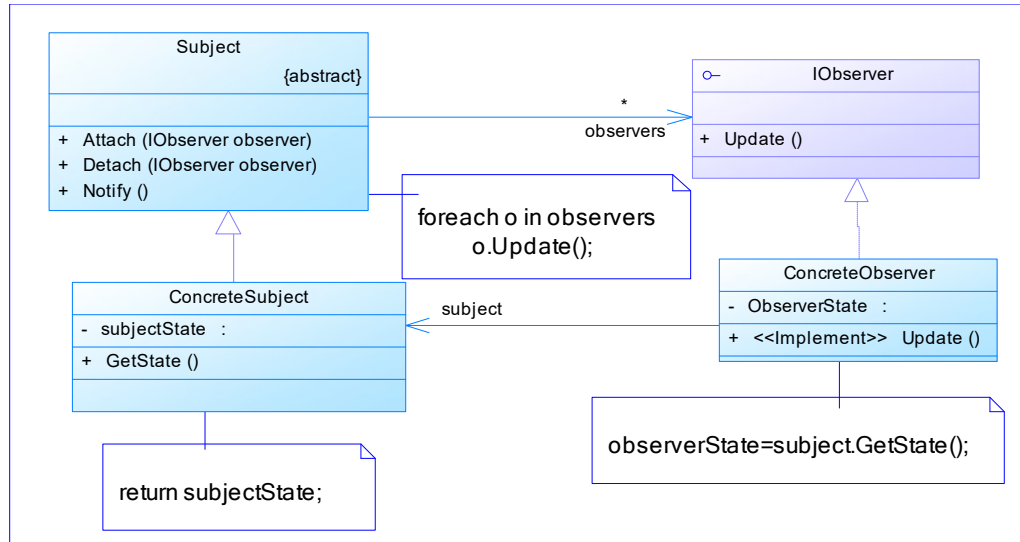
Argghh!!! Mark my words Jill, you'll never work in this town again if I have anything to do with it. You're off my call list!!!

8

04/09/2024

Observer pattern

❖ Cấu trúc



9

04/09/2024

Questions

- ❖ Giải thích vai trò của hàm `Update()` từ đó cho biết vai trò của `IObserver`
- ❖ Có thể thay thế Observer pattern bằng mô hình các đối tượng dùng chung dữ liệu được không?
- ❖ Loosely coupled design: Hãy phân tích mối quan hệ giữa 2 `ConcreteSubject` và `ConcreteObserver`
- ❖ Viết mã lệnh cho cấu trúc của Observer pattern

10

04/09/2024

Trường hợp sử dụng của Observer Pattern

- ❖ Sự thay đổi dữ liệu của một đối tượng có thể kéo theo sự cập nhật trạng thái của các đối tượng khác (Ví dụ: thay đổi dữ liệu kéo theo việc cập nhật giao diện tự động để hiển thị dữ liệu).
- ❖ Sự thay đổi một đối tượng kéo theo một hiệu ứng phụ nào đó tùy thuộc vào thành phần client sử dụng.
- ❖ Viết/thiết kế các callback trong các library/framework.

11

04/09/2024

Bài tập

- ❖ Tìm hiểu các mô hình lập trình, framework, library có sử dụng Observer Pattern

12

04/09/2024

***Design Principle***

*Strive for loosely coupled designs
between objects that interact.*

13

A decorative graphic on the left side of the slide consists of a staircase of light blue squares of varying sizes, arranged in a diagonal pattern from the bottom-left towards the top-right. To the right of this staircase is a large, solid blue rectangular area that extends towards the right edge of the slide. Below this blue area is a horizontal bar of dark blue color.

Command pattern

04/09/2024

These top secret drop boxes have revolutionized the spy industry. I just drop in my request and people disappear, governments change overnight and my dry cleaning gets done. I don't have to worry about when, where, or how; it just happens!



15

04/09/2024

Command pattern

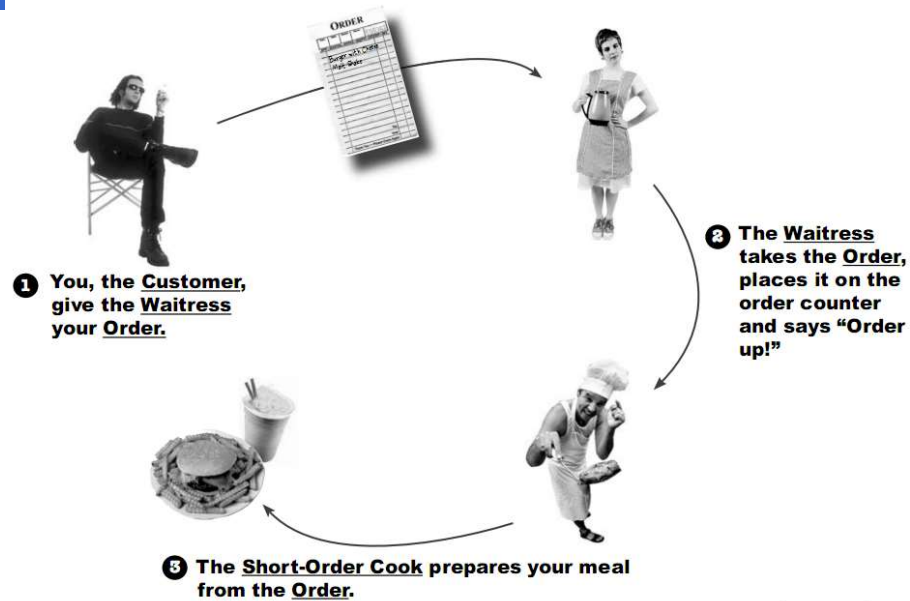
❖ Mục đích:

- Đóng gói requests thành một đối tượng.
- Cho phép tham số hóa các client với các requests khác nhau
- Tách rời request một hành động ra khỏi đối tượng thực hiện hành động đó

16

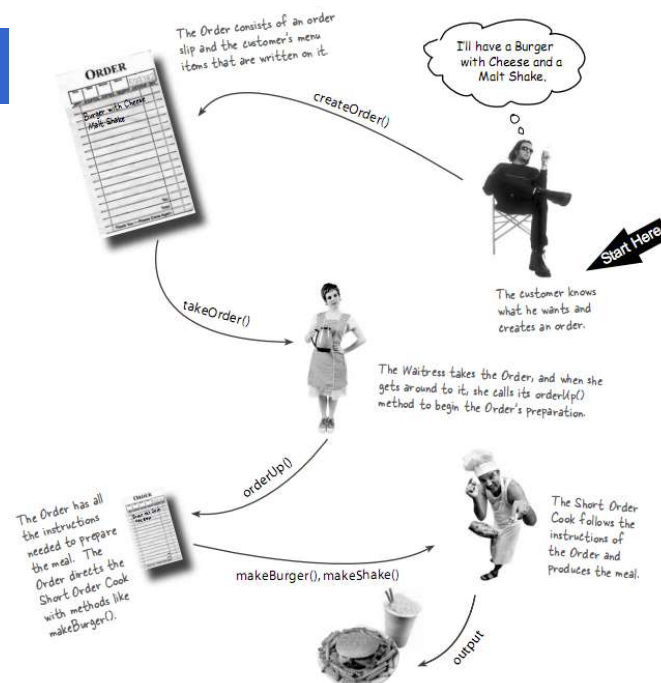
04/09/2024

Example: Diner operates



17

04/09/2024

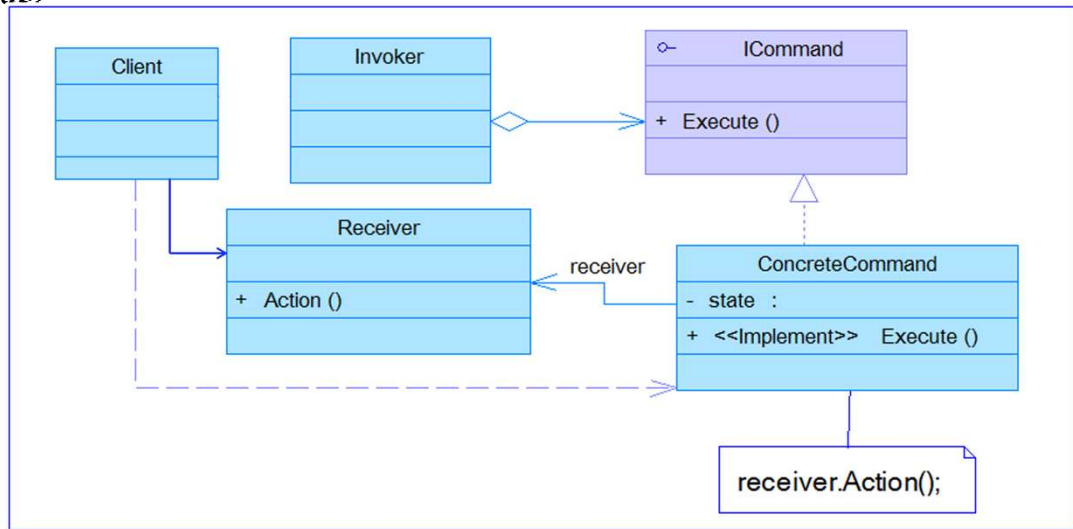


18

04/09/2024

Command pattern

❖ Cấu trúc:

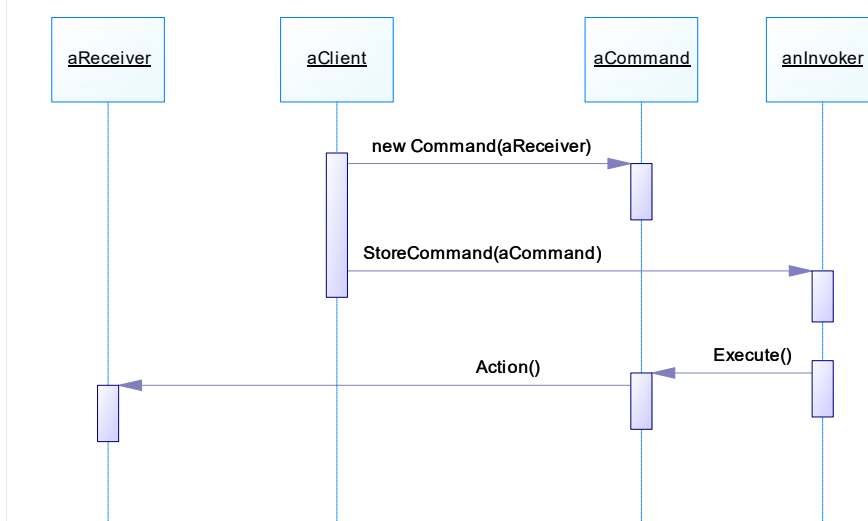


19

04/09/2024

Collaboration

Command pattern sequence diagram

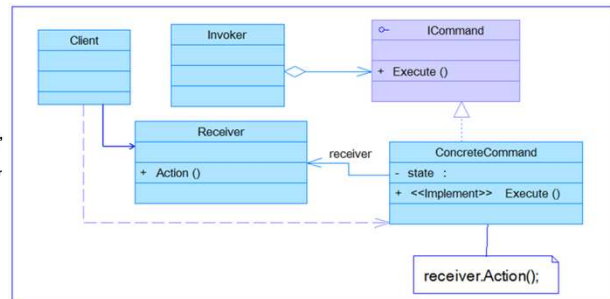
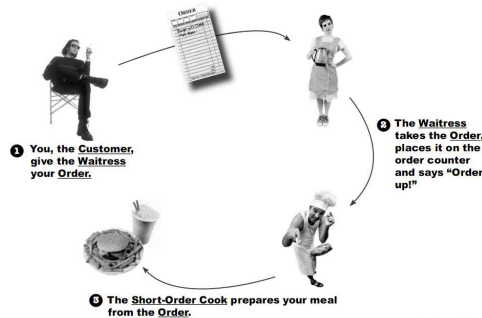


20

04/09/2024

Matching between Command & Diner Example

Diner Example	Command pattern
Customer	Client
Order	Command
Waitress	Invoker
TakeOrder()	setCommand()
Short Order Cook	Receiver
orderUp()	Execute()

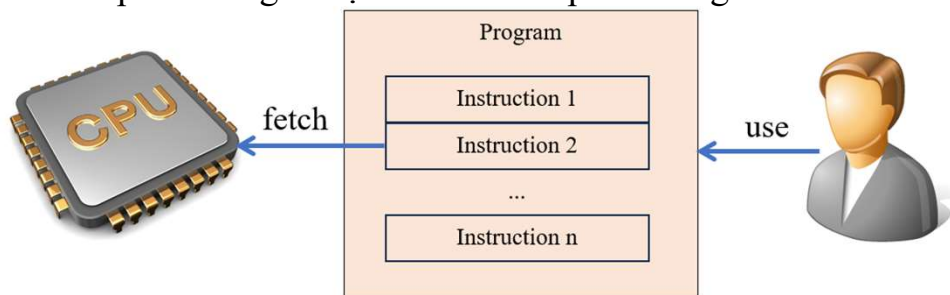


21

04/09/2024

Ví dụ

- ❖ Liên kết các thành phần trong ví dụ và các thành phần trong sơ đồ của mẫu Command

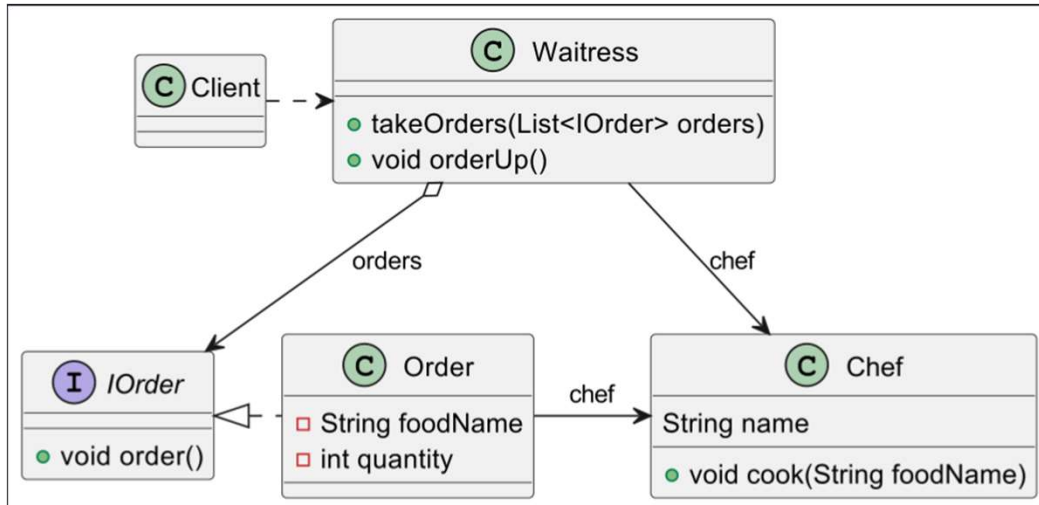


Command Pattern	Các thành phần trong sơ đồ
Invoker	?
Command	?
Receiver	?
User	?

22

04/09/2024

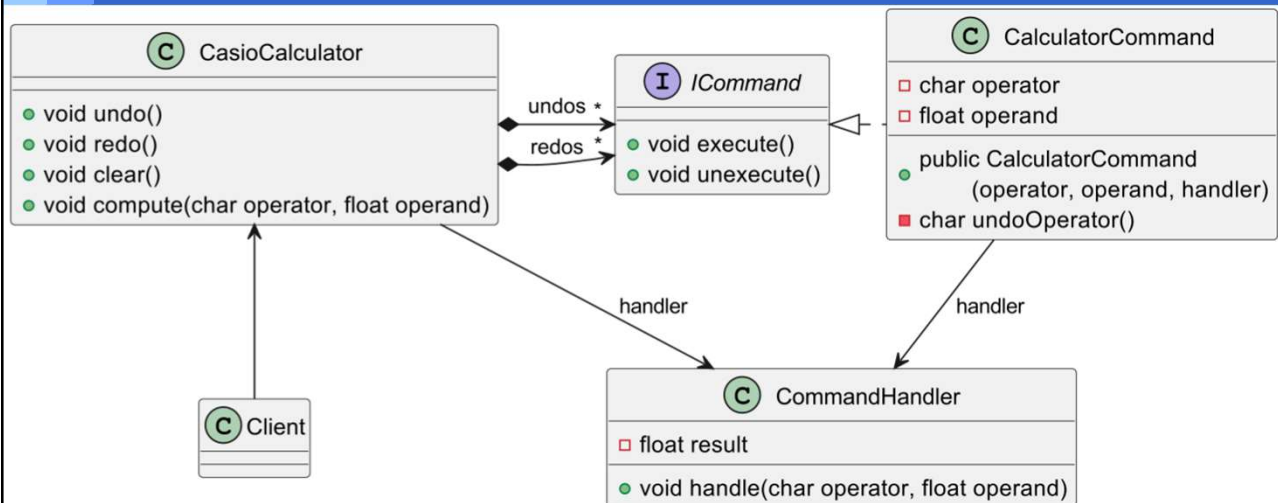
Ví dụ



23

04/09/2024

Ví dụ: Calculator với chức năng undo, redo

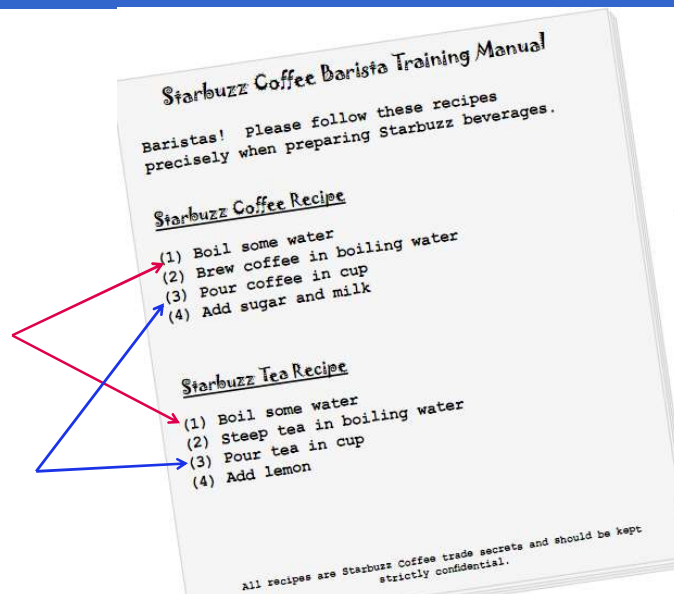


24

Template method

Tea and Coffee examples

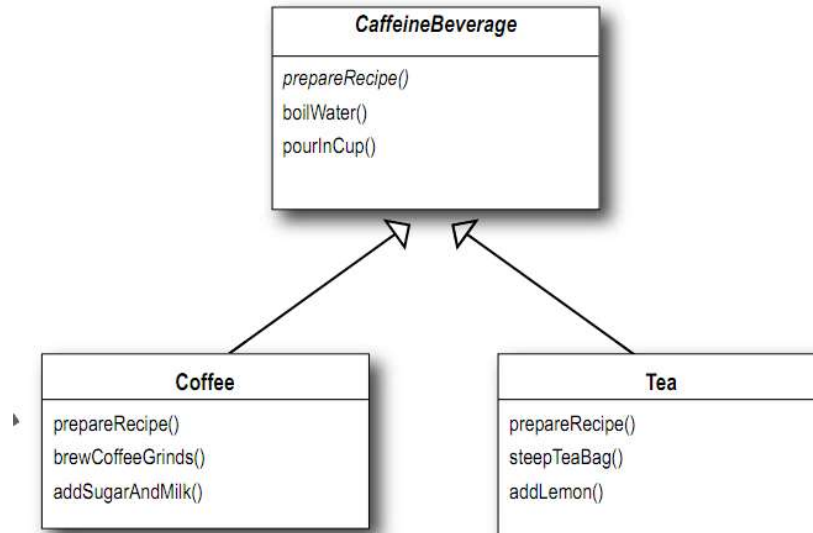
04/09/2024



26

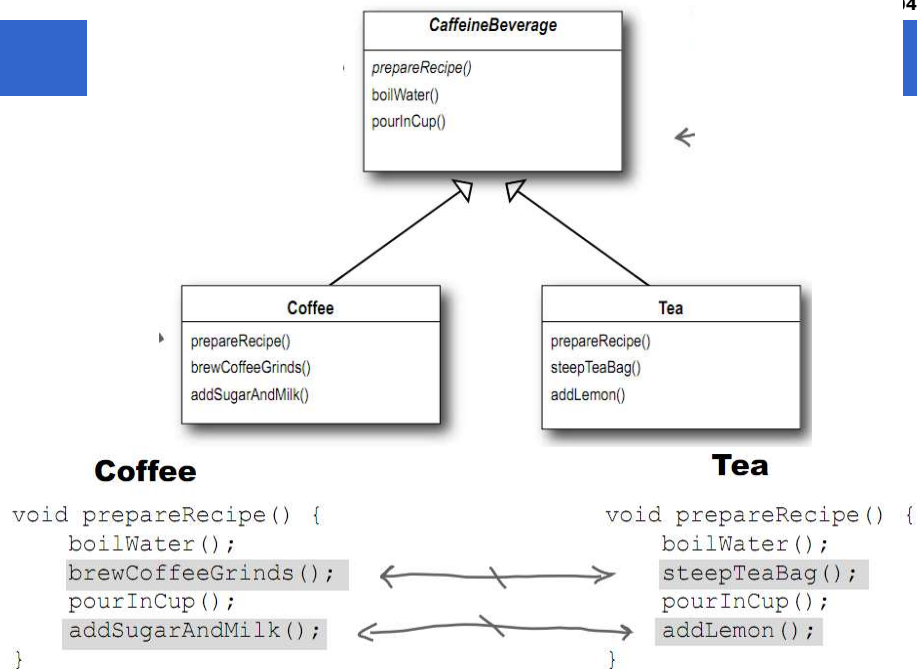
04/09/2024

Tea and Coffee example



27

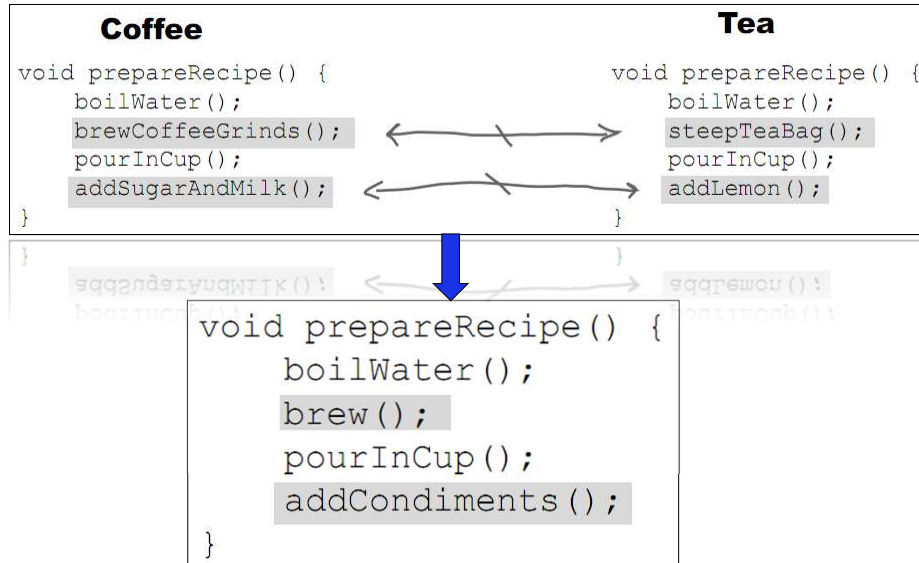
04/09/2024



28

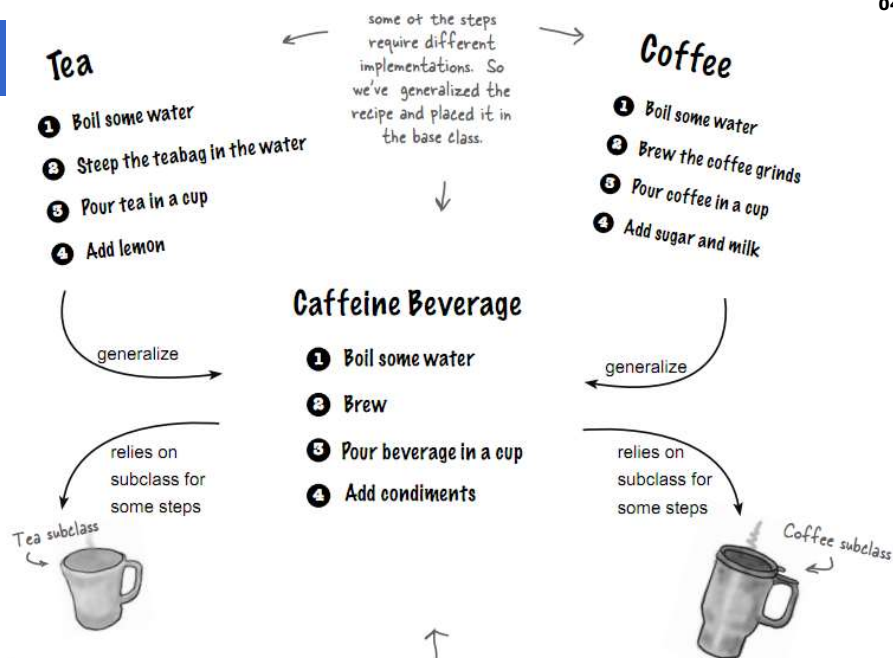
04/09/2024

Tea and Coffee example...



29

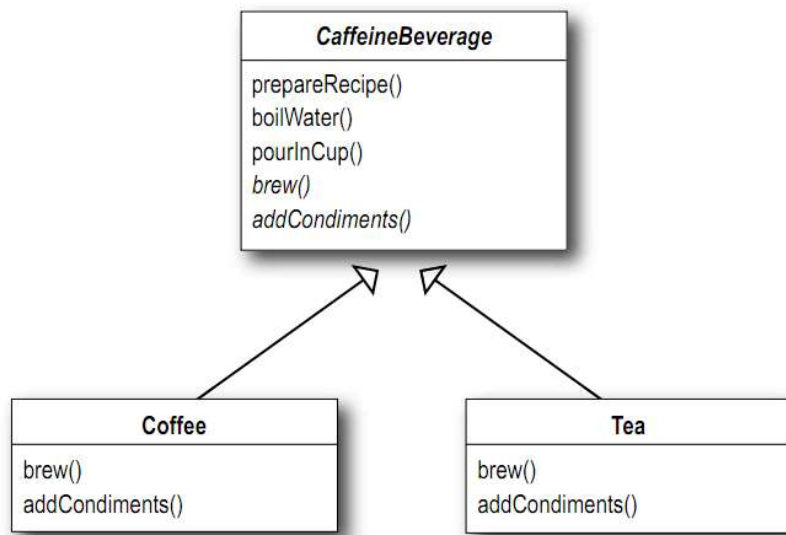
04/09/2024



30

04/09/2024

Tea and Coffee example...



31

04/09/2024

Template method

❖ Mục đích:

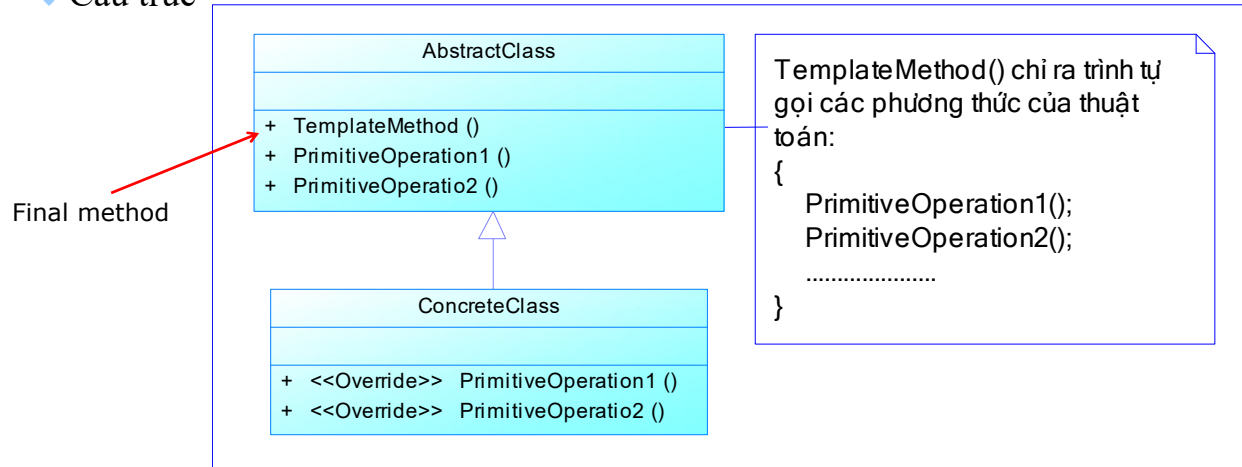
- Định nghĩa khung sườn của một thuật toán bao gồm nhiều bước trong một phương thức.
 - Một số bước được khai báo abstract ở lớp cơ sở và ủy quyền cho lớp con cài đặt
 - Việc cài đặt các bước ở lớp con không làm ảnh hưởng đến cấu trúc của thuật toán.

32

04/09/2024

Template method

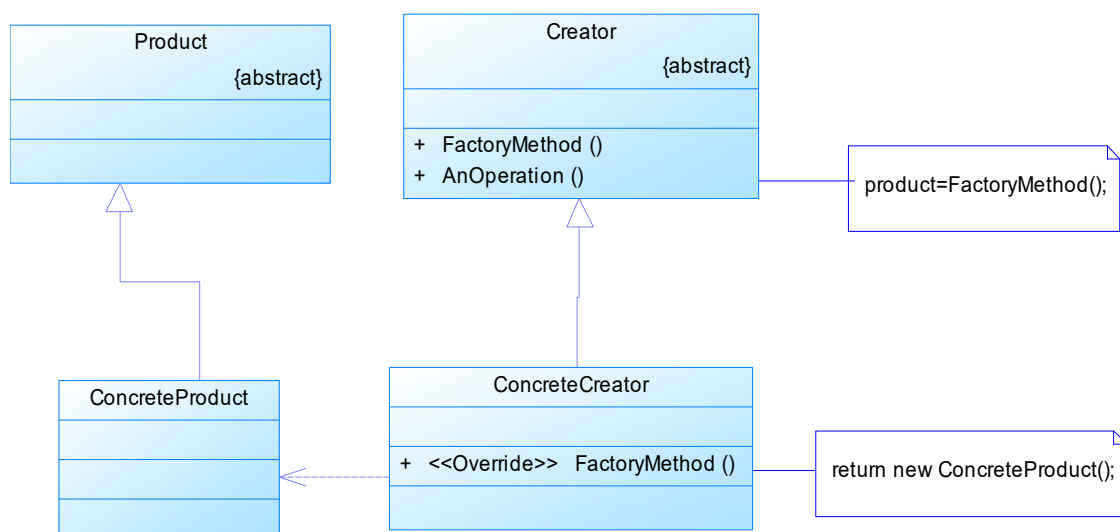
❖ Cấu trúc



33

04/09/2024

Liên hệ giữa factory method và template method



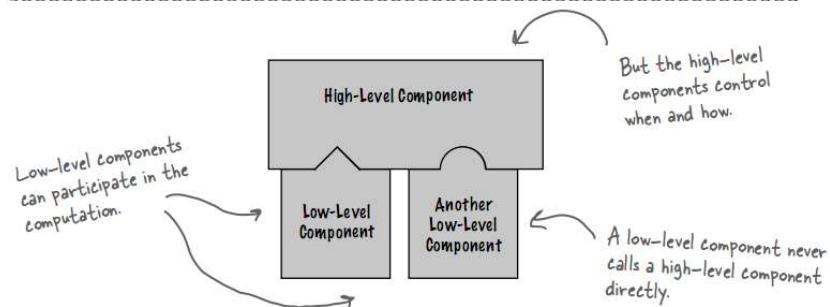
34

04/09/2024



The Hollywood Principle

Don't call us, we'll call you.



35

04/09/2024

Questions

- ❖ Có thể dùng interface để thay thế AbstractClass không? Nếu được hãy vẽ lại sơ đồ cấu trúc của template method?
- ❖ Các ConcreteClass có cần phải thực thi toàn bộ các phương thức của lớp AbstractClass?
- ❖ So sánh Template method với Strategy pattern, factory method

36

04/09/2024

Trường hợp sử dụng Template Method

- ❖ Thiết kế một lớp gồm nhiều phương thức, trong đó việc cài đặt của một số phương thức phụ thuộc (có lời gọi đến) các phương thức khác của chính lớp này nhưng:
 - Các phương thức được gọi này có nhiều tùy biến, nhiều cách cài đặt khác nhau.
 - Các phương thức phụ thuộc (template method) bất biến và có thể cài đặt được.
 - Các phương thức được gọi bởi các phương thức template có thể được khai báo abstract, việc cài đặt các phương thức này sẽ được ủy quyền cho các phương thức con

37



State pattern

04/09/2024

State pattern

❖ Mục đích:

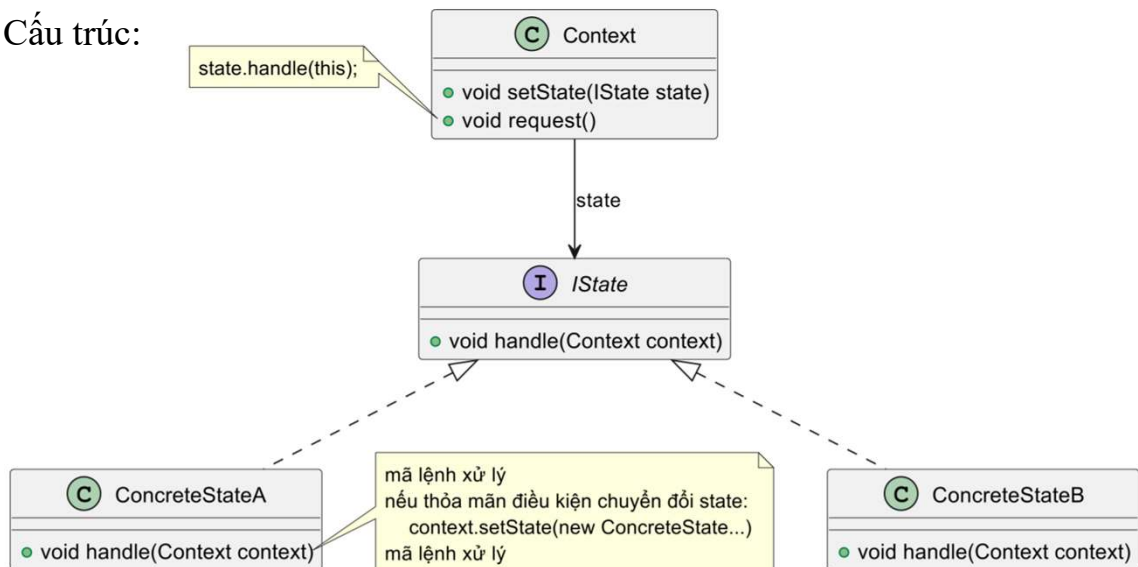
- Cho phép một đối tượng thay đổi hành vi của nó (kết quả của một phương thức) khi trạng thái bên trong của nó thay đổi.
- Mỗi lớp cụ thể sẽ có trạng thái (state) được cài đặt trong các lớp riêng. Các lớp cài đặt state sẽ chịu trách nhiệm chuyển đổi state cho đối tượng khi thỏa một điều kiện nào đó.

39

04/09/2024

State pattern

❖ Cấu trúc:



40

04/09/2024

State pattern: Sử dụng

❖ Một đối tượng có nhiều trạng thái và:

- Hành vi của đối tượng phụ thuộc vào các trạng thái của đối tượng và trạng thái này có thể được thay đổi lúc run-time
- Có nhiều operation phụ thuộc vào trạng thái của đối tượng. Thông thường chỉ vài operation cùng phụ thuộc vào trạng thái của đối tượng.

❖ Sử dụng State pattern:

- Đối tượng: thể hiện của lớp Context
- Mỗi trạng thái: thể hiện của lớp ConcreteState
- Chuyển trạng thái (state) cụ thể cho đối tượng được thực hiện trong phương thức handle khi điều kiện để trạng thái đó xuất hiện được đáp ứng
 - context.setState(...)

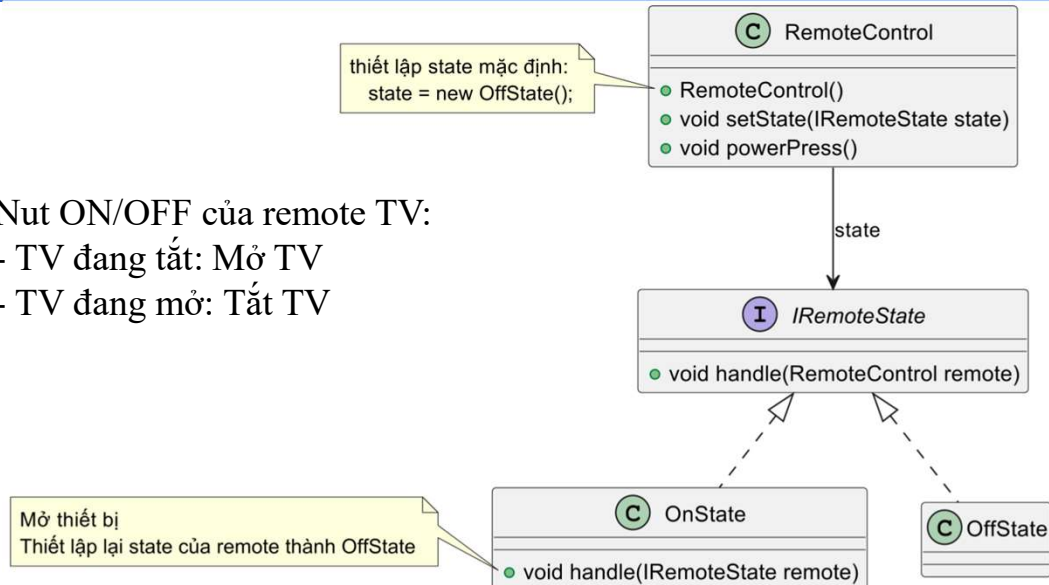
41

04/09/2024

Ví dụ

Nút ON/OFF của remote TV:

- TV đang tắt: Mở TV
- TV đang mở: Tắt TV



42



Chain of Responsibility

04/09/2024



Chain of Responsibility

❖ Mục đích:

- Tách rời thành phần gửi và đối tượng thực hiện request

❖ Ý tưởng:

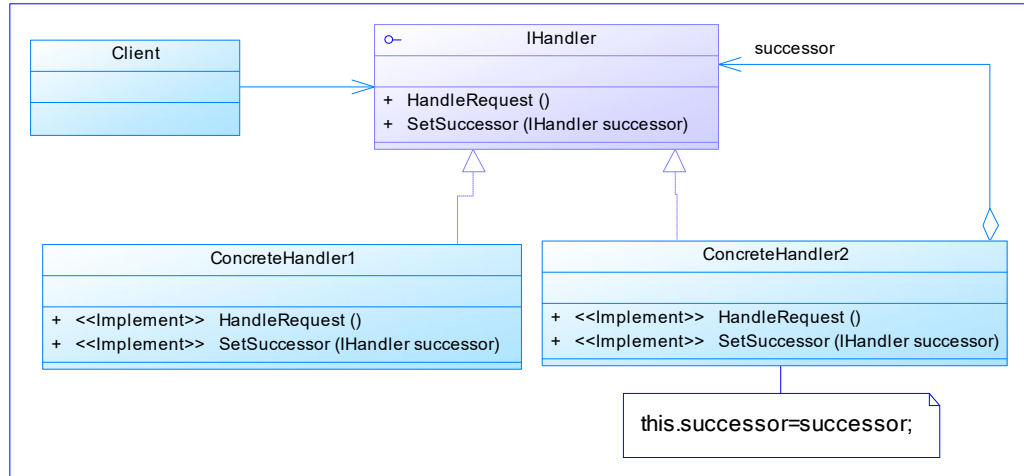
- Kết nối các đối tượng thực hiện request thành một chuỗi
- Chuyển request dọc theo chuỗi cho đến khi gặp được đối tượng có khả năng xử lý được nó

44

04/09/2024

Chain of Responsibility

❖ Cấu trúc:



45

04/09/2024

Question

❖ Có thể thay đổi Chain of Responsibility bằng cách dùng cấu trúc lệnh `if...else` của các ngôn ngữ lập trình được không?

46

04/09/2024

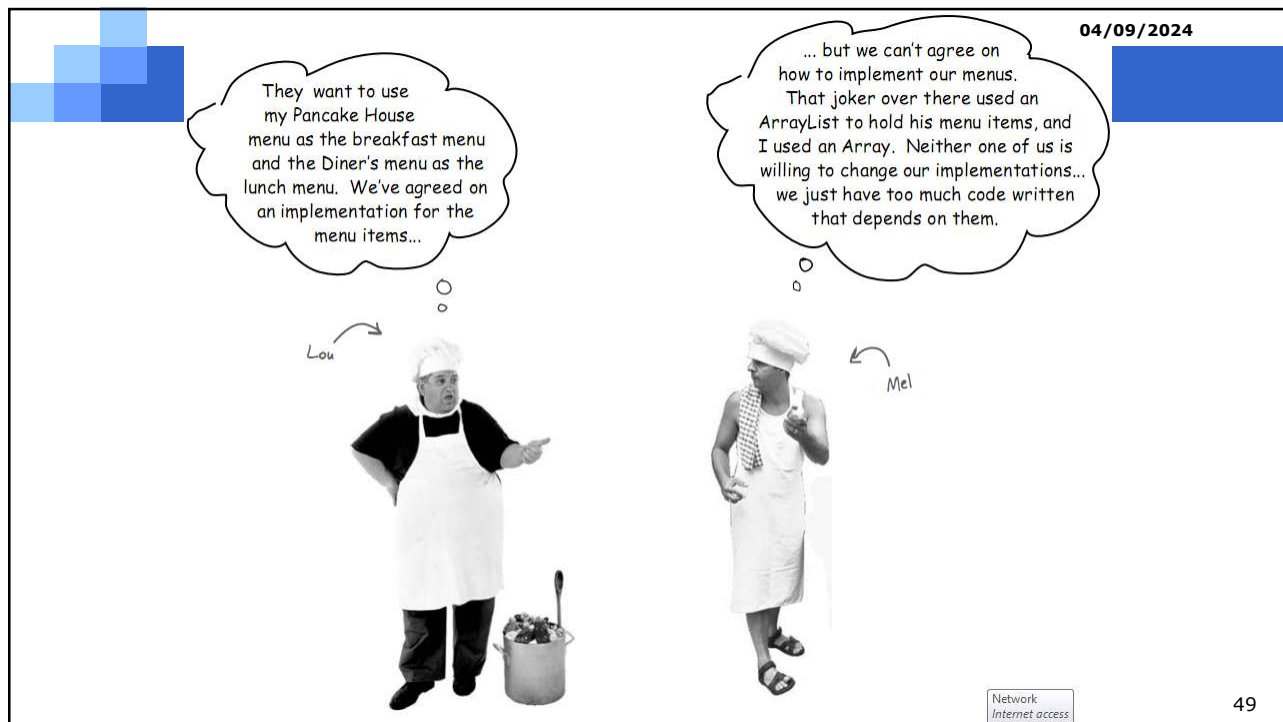
Sử dụng

- ❖ “Người gọi” không biết phải gọi request cho “người nhận” nào trong tập các “người nhận”
- ❖ Chuỗi các “người nhận” có thể thay đổi lúc run-time

47



Iterator pattern



Iterator

04/09/2024

❖ Vấn đề:

- Các tập hợp khác nhau được biểu diễn theo các cách khác nhau
- Client truy cập tới các phần tử của tập hợp theo một cách duy nhất
- Client không cần biết cấu trúc của từng tập hợp cụ thể

50

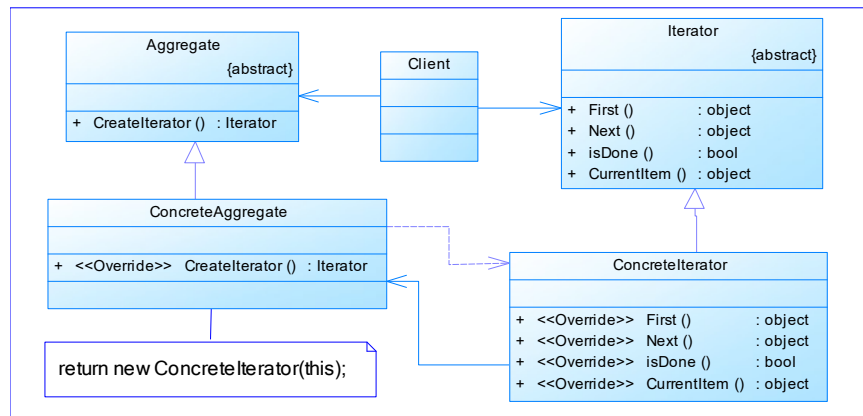
04/09/2024

Iterator

❖ Mục đích:

- Cung cấp một cách truy cập các phần tử của một tập hợp một cách tuần tự mà không cần biết cấu trúc của tập hợp đó.

❖ Cấu trúc



51

04/09/2024

Questions

- ❖ Có thể gộp chung hai lớp **Iterator** và **Aggregate** được không? Nêu ưu và khuyết điểm của cách làm này.
- ❖ Vận dụng mẫu **Iterator**: Mẫu **Iterator** đã được cài đặt cho hầu hết các tập hợp, việc sử dụng mẫu **Iterator** chính là sử dụng các cài đặt sẵn có của iterator trên các tập hợp này. Sau đây là một số khuyến nghị:
 - **Iterator** là giao diện chung cho tất cả các tập hợp cài đặt một phương pháp thứ hai để duyệt tập hợp. Do đó, để tránh trường hợp client phải phụ thuộc vào một tập hợp cụ thể, nên sử dụng tập hợp qua giao diện **Iterator** nếu có thể.

52

04/09/2024

***Design Principle***

A class should have only one reason to change.

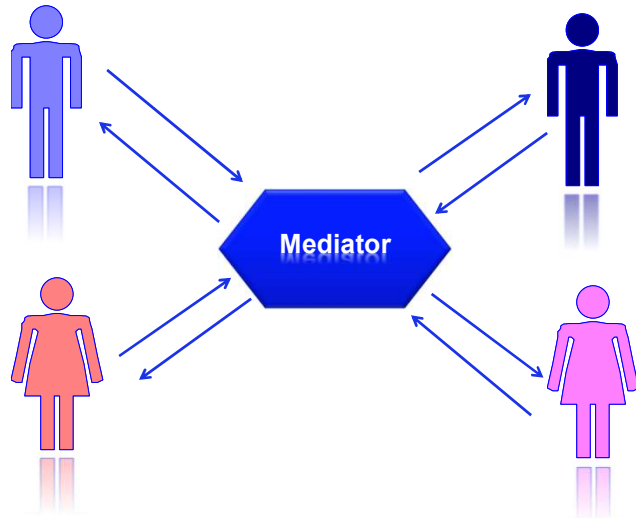
Mỗi class chỉ nên thiết kế với một single responsibility

53

Mediator pattern

04/09/2024

Mediator: Example



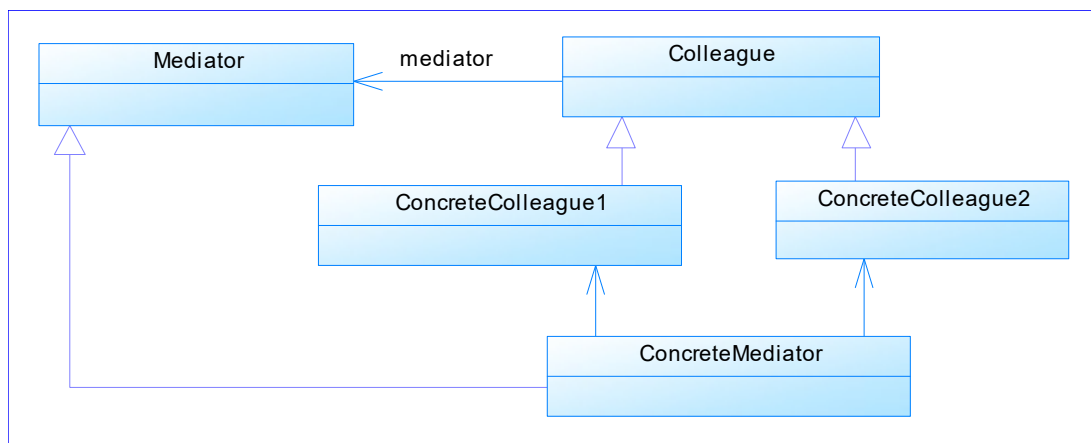
55

04/09/2024

Mediator pattern

❖ Mục đích:

- Đóng gói các cách tương tác giữa một tập các đối tượng



56

04/09/2024

Mediator

❖ Ưu điểm

- Gia tăng việc tái sử dụng các đối tượng được hỗ trợ bởi Mediator bằng cách tách rời chúng ra khỏi hệ thống
- Đơn giản hóa việc duy trì hệ thống bằng cách tập trung logic điều khiển
- Đơn giản hóa và giảm sự thay đổi các message được gửi giữa các đối tượng trong hệ thống

❖ Hạn chế

- Mediator có thể rất phức tạp nếu không được thiết kế một cách phù hợp.

❖ Sử dụng

- Mediator thường được sử dụng để phối hợp các thành phần GUI với nhau

57

04/09/2024

Trường hợp sử dụng

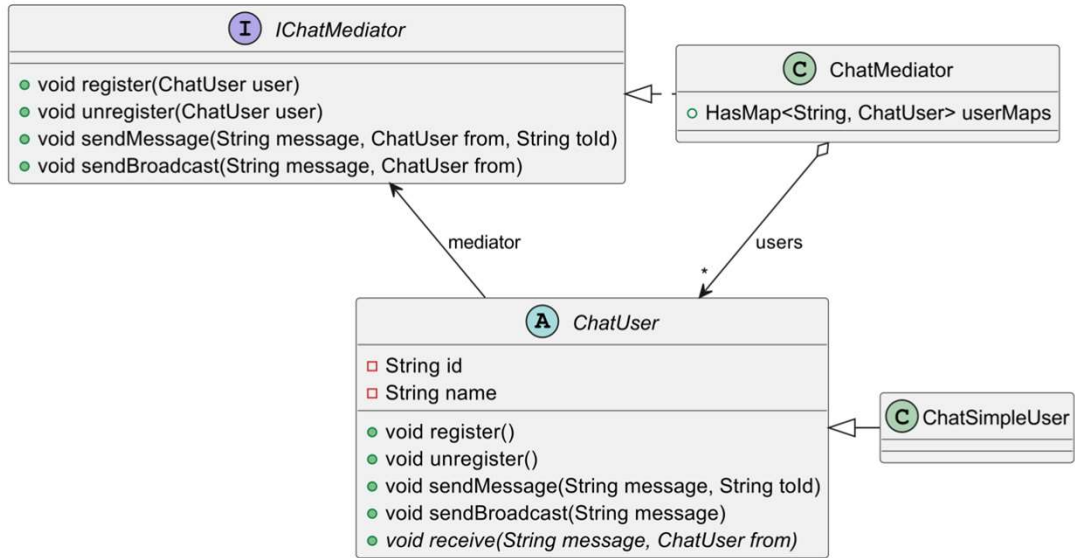
❖ Mediator được sử dụng trong trường hợp

- Một nhóm các object giao tiếp/tương tác với nhau, việc tự quản lý các mối liên kết giữa các object gặp khó khăn.
- Cần tập trung quản lý các liên kết giữa các object, đây cũng chính là nguyên lý: Mỗi lớp chỉ nên thực hiện một nhiệm vụ đơn

58

04/09/2024

Ví dụ: Chat



59

Memento pattern

04/09/2024

Memento pattern

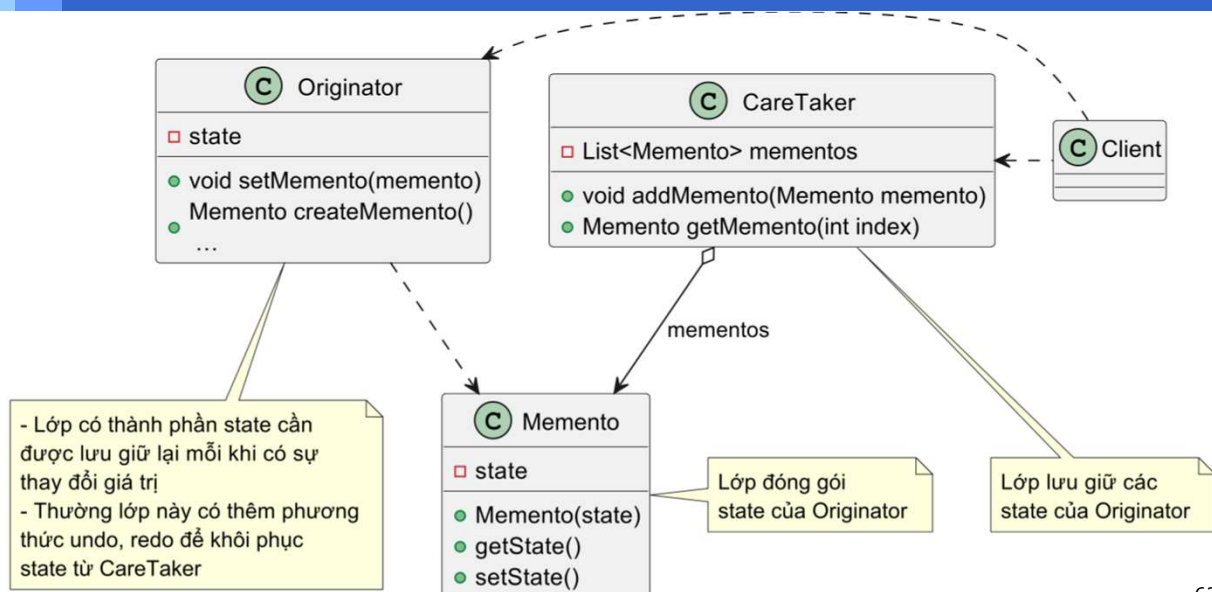
❖ Mục đích:

- Lưu lại các trạng thái của một đối tượng để khôi phục lại sau này mà không vi phạm nguyên tắc đóng gói.
 - Các trạng thái của đối tượng được lưu giữ trong một lớp riêng
 - Việc khôi phục lại trạng thái của một đối tượng do chính đối tượng đó đảm nhận

61

04/09/2024

Cấu trúc



62

04/09/2024

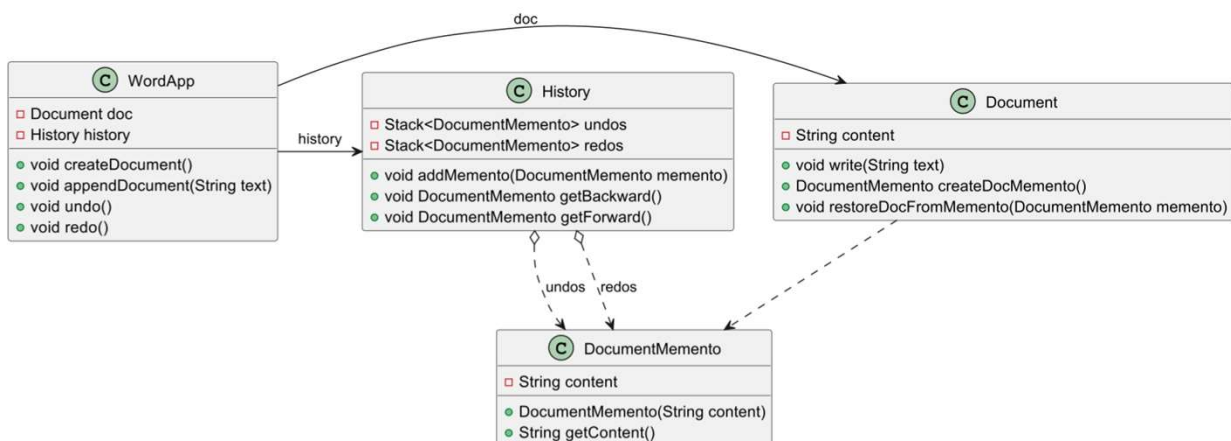
Trường hợp sử dụng

- ❖ Sử dụng memento pattern trong trường hợp muốn lưu giữ các bản sao về trạng thái của một đối tượng để có thể khôi phục trạng thái của một đối tượng
 - Ví dụ: Truy cập dữ liệu theo Transaction, nếu bị lỗi có thể khôi phục dữ liệu về trạng thái trước đó
- ❖ Sử dụng memento pattern trong trường hợp không muốn cho truy cập các trạng thái của một đối tượng thông qua các getter, setter. Tuy nhiên ứng dụng vẫn có các thành phần muốn truy cập các trạng thái này

63

04/09/2024

Ví dụ: WordApp



64



Visitor pattern



Vấn đề

04/09/2024

- ❖ Xử lý thông tin của một phần tử trong một cấu trúc đối tượng cho trước
- ❖ Việc xử lý thông tin như thế nào chưa thể xác định lúc compile-time

04/09/2024

Visitor pattern

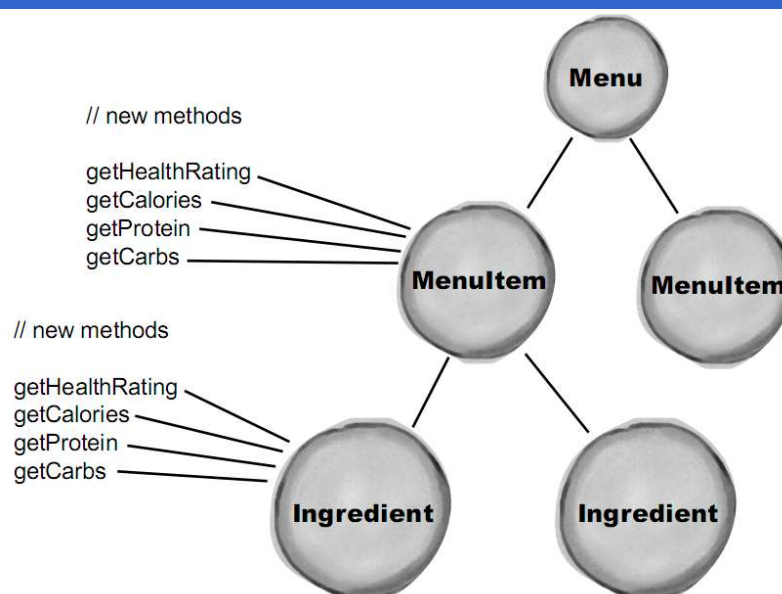
❖ Mục đích:

- Thực hiện một thao tác trên một phần tử của một cấu trúc phức hợp (composite structure)
- Định nghĩa phương thức mới thao tác trên một phần tử của cấu trúc mà không cần thay đổi các lớp đã được định nghĩa trên cấu trúc đó

67

04/09/2024

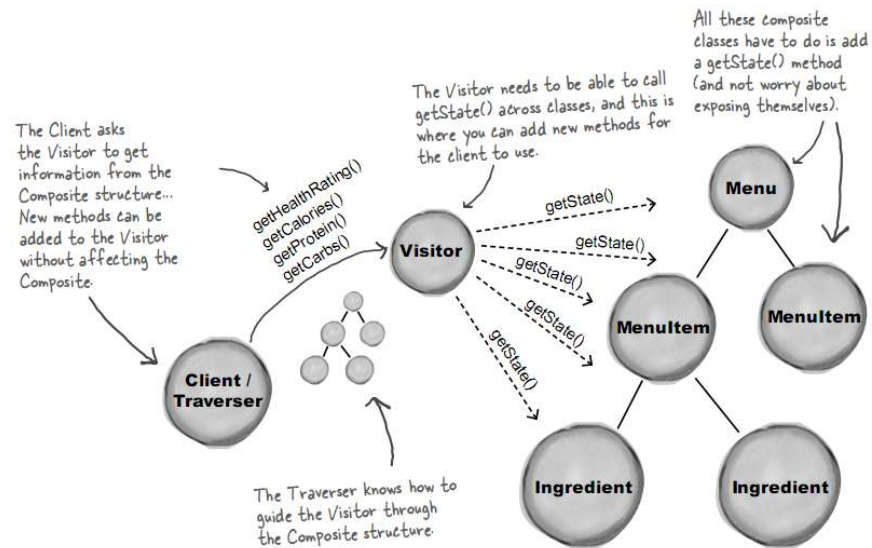
Visitor pattern: Example



68

04/09/2024

Visitor pattern: Example

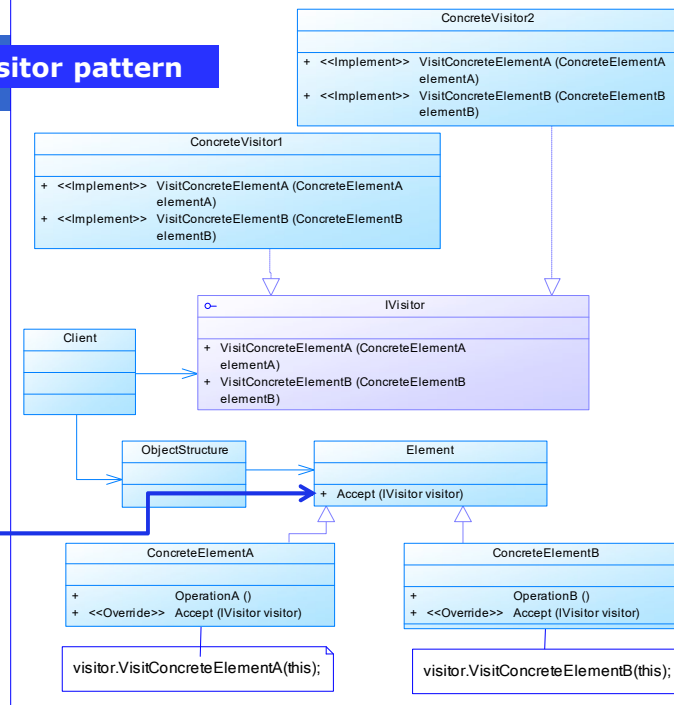


69

Giải pháp: Visitor pattern

04/09/2024

Định nghĩa trước một khuôn dạng để cho phép các đối tượng dạng IVisitor truy cập tới các field bên trong



70

04/09/2024

Visitor pattern

❖ Ưu điểm

- Cho phép thêm các thao tác xử lý tới một Composite structure mà không cần thay đổi cấu trúc đó
- Việc thêm mới một operation khá dễ dàng
- Mã lệnh của các operation được thực hiện bởi Visitor được tập trung

❖ Hạn chế

- Tính chất đóng gói (encapsulation) của các lớp bị phá vỡ khi sử dụng Visitor
- Việc thay đổi nó để phù hợp với cấu trúc Composite gặp nhiều khó khăn do phải dùng hàm duyệt cấu trúc của Composite.

71

04/09/2024

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>

72