



THIẾT KẾ PHẦN MỀM

ThS. Dương Hữu Thành,
Khoa CNTT, Đại học Mở TP.HCM,
thanh.dh@ou.edu.vn.

1

The background of this slide is identical to the one above, featuring the same blue-toned graphic of hands interacting with a digital interface containing gears and a network of connections.

Nội dung chính

- 1** Thiết kế phần mềm
- 2** Thiết kế giao diện
- 3** Thiết kế xử lý
- 4** Thiết kế dữ liệu
- 5** Thiết kế hướng đối tượng
- 6** Mẫu thiết kế
- 7** Nguyên lý thiết kế phần mềm

2

2



Thiết kế phần mềm

- Thiết kế phần mềm nhằm **mô tả chi tiết** tổ chức, hoạt động các đơn vị xử lý phần mềm dựa trên kết quả yêu cầu phần mềm.
- Thiết kế là cơ sở cho việc hiện thực phần mềm, xác định giải pháp, cách thức giải quyết vấn đề (**HOW**).

Dương Hữu Thành

3



Thiết kế giao diện

- Thiết kế giao diện nhằm mô tả chi tiết **cách thức giao tiếp** giữa người sử dụng và phần mềm.
- Kết quả của thiết kế giao diện
 - Danh sách các màn hình
 - Thông tin chi tiết từng màn hình
 - Màn hình chính
 - Sơ đồ các màn hình

Dương Hữu Thành

4



Thiết kế giao diện

- **Tính tiện dụng:** dễ sử dụng, trực quan, dễ huấn luyện người dùng, .v.v.
- **Tính hiệu quả:** sử dụng đơn giản, tự nhiên, tránh những thao tác dư thừa.
- **Tính tiến hóa:** tuỳ chọn nội dung, hình thức trình bày, sự kiện xử lý.
- Khi thiết kế nên **theo dõi thói quen** người sử dụng và cần trao đổi thông nhất khi có những thói quen không hợp lý với người dùng.

Dương Hữu Thành

5



Thiết kế giao diện

- Các thành phần trên màn hình giao diện thường có:
 - Các thành phần nhập liệu.
 - Các thành phần hiển thị thông tin kết quả.
 - Các nút xử lý.
- Chú ý:
 - Các thành phần **nhập liệu** quyết định **tính đúng đắn** của màn hình giao diện.
 - Các thành phần **thông tin kết quả** quyết định **tính hiệu quả và tiện dụng** của giao diện.

Dương Hữu Thành

6



Thiết kế giao diện

- Các thành phần nhập liệu
 - Ô nhập liệu
 - Lựa chọn
 - Radio Button (chỉ được chọn một)



Tiêu đề 1



Tiêu đề 2



Tiêu đề 3

- Checkbox (được chọn nhiều)



Tiêu đề 1



Tiêu đề 2



Tiêu đề 3



Thiết kế giao diện

- Thành phần thông tin kết quả cho phép người dùng **xem thông tin kết quả** hoặc thông tin tương ứng với dữ liệu nhập trên màn hình giao diện.



- Các nút xử lý cho phép người dùng chọn các **tác vụ cần thực hiện** trên màn hình giao diện.



Quy tắc thiết kế giao diện

- Trong quyển sách thiết kế giao diện, tiến sĩ **Theo Mandel [Man97]** đưa ra 3 quy tắc vàng khi thiết kế giao diện.
 - Người dùng được phép linh hoạt tương tác giao diện (**Place the user in control**).
 - Hạn chế bắt người dùng nhớ quá nhiều thông tin khi tương tác giao diện (**Reduce the user's memory load**).
 - Thiết kế giao diện nhất quán (**Make the interface consistent**).

Dương Hữu Thành

9

9



Place the user in control

- **Không bắt buộc** người dùng thực hiện các thao tác không cần thiết.
- Hỗ trợ nhiều **tương tác linh hoạt** do người dùng khác nhau sẽ có những sở thích khác nhau.
- Cho phép người dùng **tạm ngắt** (interrupt) công việc đang làm và có khả năng **phục hồi** (undo) khi quay lại làm tiếp.

Dương Hữu Thành

10

10



Place the user in control

- Cho phép **tuỳ chỉnh** một số thao tác quen thuộc lặp lại để tương tác nhánh chóng, hiệu quả hơn.
- **Ẩn chi tiết xử lý** kỹ thuật bên dưới hệ thống với người dùng, chẳng hạn không hiển thị thông tin lỗi liên quan bên trong hệ thống trên màn hình người dùng.
- Thiết kế các **thao tác trực tiếp** (direct interaction) với các đối tượng xuất hiện trên màn hình, chẳng hạn kéo một object vào thùng rác để xoá.

Dương Hữu Thành

11

11



Reduce the User's Memory Load

- Giảm một số yêu cầu ghi nhớ của người dùng khi thực thi tác vụ phức tạp.
- Thiết lập các **giá trị mặc định** có ý nghĩa.
- Định nghĩa các **shortcut, hot-keys** trực quan, dễ nhớ, chẳng hạn Ctrl+P là thực hiện in, trong đó ký tự P là ký tự đầu của chức năng cần làm.

Dương Hữu Thành

12

12



Reduce the User's Memory Load

- Việc bố cục giao diện trực quan **dựa trên những tác vụ trong thế giới thực.**
- Hiển thị thông tin với người dùng theo quy trình, từng bước, hạn chế hiển thị cùng lúc quá nhiều thông tin không cần thiết.
 - Những thông tin chung, quan trọng hiển thị đầu tiên.
 - Chi tiết từng thông tin hiển thị các bước tiếp theo.

Dương Hữu Thành

13

13



Make the interface consistent

- **Thông tin hiển thị** theo quy tắc thiết kế nhất quán giữa các màn hình.
 - Màu sắc nhất quán, không dùng quá nhiều màu, đặc biệt các màu sắc sỡ, thiết kế phù hợp giữa các màu sắc tương phản.
 - Các thông điệp (message) trên giao diện phải nhất quán, súc tích, lịch sự, nên kèm theo lời giải thích, gợi ý ý nghĩa của thông điệp.

Dương Hữu Thành

14

14



Make the interface consistent

- **Cơ chế nhập liệu** (input) nhất quán trong ứng dụng.
- **Cơ chế điều hướng** từ tác vụ này sang tác vụ khác cũng phải được định nghĩa nhất quán.
- Cho phép người dùng thấy được tác vụ đang thực hiện nằm trong ngữ cảnh có nghĩa nào của hệ thống.

Dương Hữu Thành

15

15



Make the interface consistent

- Một số mô hình tương tác đã trở thành **thói quen** của người dùng thì **không nên thay đổi** nếu không có lý do thuyết phục, chẳng hạn Ctrl+S thường dùng để lưu sự thay đổi thì không nên thay đổi nếu không cần thiết.
- Kiểm tra các dữ liệu nhập trên giao diện cả về **ràng buộc bản chất** và **ràng buộc nghiệp vụ** của nó.

Dương Hữu Thành

16

16



Thiết kế xử lý

- Mục tiêu của thiết kế xử lý là nhằm mô tả tất cả các **hàm xử lý** của phần mềm tương ứng với các yêu cầu phần mềm.
- Kết quả của giai đoạn này là danh sách các xử lý và chi tiết các xử lý quan trọng.

STT	Tên xử lý	Điều kiện gọi thực hiện	Ý nghĩa	Ghi chú

Dương Hữu Thành

17



Thiết kế xử lý

- Ví dụ giao diện đặt vé xe khách trực tuyến.

Họ tên	<input type="text"/>
Email	<input type="text"/>
Điện thoại	<input type="text"/>
Ngày sinh	<input type="text"/>
Nơi đi	<input type="text"/>
Nơi đến	<input type="text"/>
Ngày đi	<input type="text"/> 
Chuyến đi	<input type="text"/>

A1	B1
A2	B2
A3	B3
A4	B4
A5	B5
A6	B6
...	...

Dương Hữu Thành

18



Thiết kế xử lý

- Các xử lý trên giao diện

STT	Tên xử lý	Điều kiện gọi thực hiện	Ý nghĩa
1	NoiDi_Change	Chọn nơi đi và có thông tin nơi đến và ngày đi.	Nạp danh sách các chuyến đi vào select box “Chuyến đi”
2	NoiDen_Change	Chọn nơi đến và có thông tin nơi đi, và ngày đi.	Nạp danh sách các chuyến đi vào select box “Chuyến đi”
3	NgayDi_Change	Chọn ngày đi và có đủ thông tin nơi đi, nơi đến.	Nạp danh sách các chuyến đi vào select box “Chuyến đi”
4	ChuyenDi_Change	Chọn một chuyến đi mới	Nạp danh sách ghế và trạng thái ở vùng bên phải.
5	DatVe_Click	Click vào nút Đặt vé	Tiến hành ghi nhận thông tin đặt vé và xử lý thanh toán.
6	HuyGiaoDich_Click	Click vào nút Huỷ giao dịch	Tiến hành kết thúc giao dịch hiện tại.

Dương Hữu Thành

19



Thiết kế dữ liệu

- Thiết kế dữ liệu nhằm mô tả cách thức **tổ chức lưu trữ dữ liệu** của phần mềm trong máy tính.
- Kết quả của thiết kế dữ liệu
 - Danh sách các bảng dữ liệu.
 - Mối quan hệ giữa các bảng dữ liệu.
 - Thông tin chi tiết từng bảng như danh sách thuộc tính, khoá chính.

Dương Hữu Thành

20



Thiết kế dữ liệu

- **Tính đúng đắn**: lưu đầy đủ và đúng ngữ nghĩa các thông tin có trong nghiệp vụ liên quan.
- **Tính tiến hóa**: lưu thông tin về tổ chức và quy định có trong nghiệp vụ liên quan.
- **Tính hiệu quả**: lưu tiết kiệm không gian bộ nhớ, truy xuất nhanh thông tin cần thiết.
- **Tính bảo mật**: lưu thông tin người sử dụng phần mềm cùng với quyền tương ứng.

Dương Hữu Thành

21

21



Thuộc tính

- Thuộc tính **khoa** (primary key)
- Thuộc tính có giá trị **rời rạc**
 - Số điện thoại: di động, nhà, cơ quan.
 - Điểm kiểm tra học sinh THPT: điểm 15 phút, điểm 1 tiết, điểm giữa kỳ, điểm cuối kỳ.
 - Giá vé tour du lịch: giá người lớn và trẻ em.
- Thuộc tính **đa trị** (multivalued attributes)
 - Địa chỉ: số nhà, tên đường, quận/huyện.

Dương Hữu Thành

22

22



Thuộc tính

- Thuộc tính **suy diễn**

Receipt		ReceiptDetail	
<u>id</u>	<u>INT</u>		
created_date	DATE		
creator_id	INT		
<u>total_amount</u>	<u>DECIMAL</u>		

thuộc tính suy diễn, nó vẫn có thể suy ra từ công thức sau nếu không lưu trữ

$$\text{total_amount} = \sum_{\text{ReceiptDetails}} \text{quantity} * \text{unit_price}$$

Dương Hữu Thành

23



Thuộc tính

- Thuộc tính suy diễn nếu được truy cập thường xuyên bởi nhiều người dùng thì có thể lưu trữ để tăng hiệu năng của hệ thống.
- Thuộc tính suy diễn được truy cập ít và số lượng người truy cập không nhiều thì không cần lưu trữ để tiết kiệm bộ nhớ.

Dương Hữu Thành

24

Quan hệ các thực thể

- Quan hệ một-một (1-1).

Account	
<u>user_id</u>	INT
username	VARCHAR(50)
password	VARCHAR(100)
active	BIT
joined_date	DATE

Thuộc

UserInfo	
<u>user_id</u>	INT
first_name	VARCHAR(50)
last_name	VARCHAR(50)
phone	VARCHAR(50)
email	VARCHAR(20)

Dương Hữu Thành

25

25

Quan hệ các thực thể

- Quan hệ một-nhiều (1-n).

Receipt	
<u>id</u>	INT
...	...
creator_id	INT

Được tạo

Account	
<u>user_id</u>	INT
...	...
user_role	INT

Có

UserRole	
<u>role_id</u>	INT
name	VARCHAR(50)
note	VARCHAR(255)

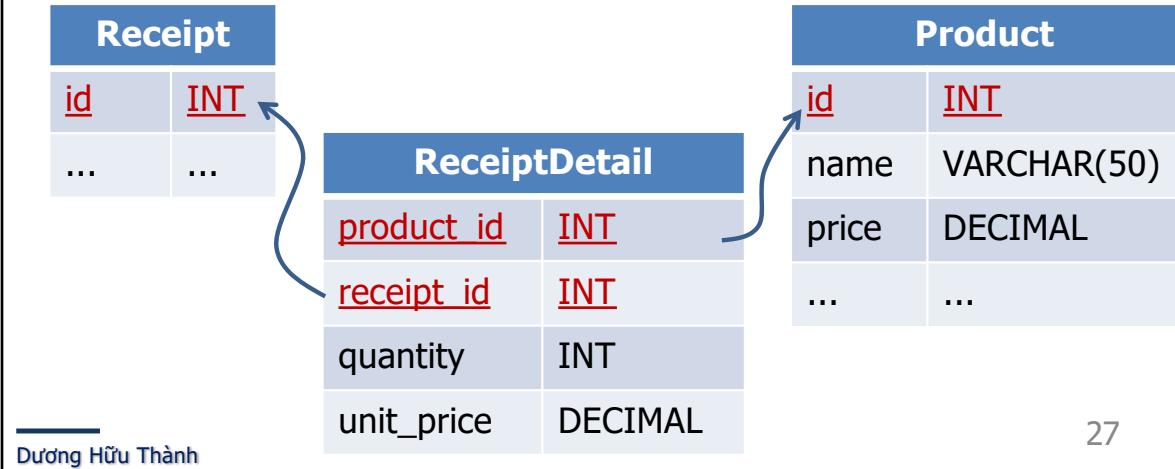
Dương Hữu Thành

26

26

Quan hệ các thực thể

- Quan hệ nhiều-nhiều (n-n).
 - Sản phẩm thuộc nhiều hóa đơn.
 - Hóa đơn có nhiều sản phẩm được mua.



27

Thiết kế hướng đối tượng

- Thiết kế phần mềm hướng đối tượng bao gồm việc **phát triển nhiều mô hình hệ thống** khác nhau.
- Phương pháp này **phù hợp với các dự án lớn**, nhiều nhóm làm việc.
- Do đòi hỏi nhiều công sức (effort) cho việc phát triển và bảo trì các mô hình, nên nó sẽ không hiệu quả về mặt chi phí cho các hệ thống nhỏ.

Thiết kế dữ liệu từ sơ đồ lớp

- Mỗi lớp đối tượng được ánh xạ thành một bảng dữ liệu.
- Nếu lớp đối tượng **có cấu trúc kiểu mảng** hoặc **có cấu trúc phức tạp** thì tách thành bảng riêng.
- Nếu lớp có thuộc tính **kiểu enum** thì tách thành một bảng và có khoá ngoại tham chiếu đến.
- Các thuộc tính có **giá trị rời rạc** thì tách thành bảng danh mục.

Dương Hữu Thành

29

- Mục tiêu quan trọng của thiết kế dữ liệu là **hạn chế sự trùng lặp thông tin** trong dữ liệu.
- Xét lớp sau

User
- id : int
- firstName : String
- lastName : String
- city : String
- country : String

- Chuyển thành bảng dữ liệu

<u><u>id</u></u>	first_name	last_name	city	country
------------------	------------	-----------	------	---------

Dương Hữu Thành

30

Thiết kế dữ liệu từ sơ đồ lớp

- Một thể hiện dữ liệu minh họa cho lớp trên

id	first_name	last_name	city	country
1	Thanh	Duong	TPHCM	Viet Nam
2	Vy	Nguyen	TPHCM	Viet Nam
3	Chau	Nguyen	Can Tho	Viet Nam
4	Phuong	Vo	Can Tho	Viet Nam
5	Ngoc	Le	TPHCM	Viet Nam
6	Tran	Le	Can Tho	Viet Nam

Trùng lắp thông tin

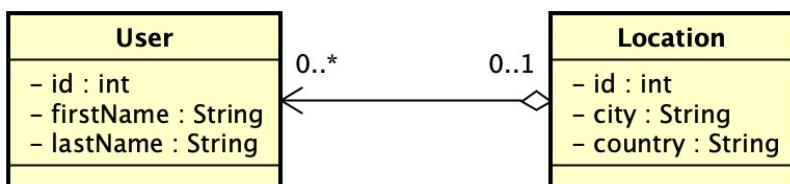
Dương Hữu Thành

31

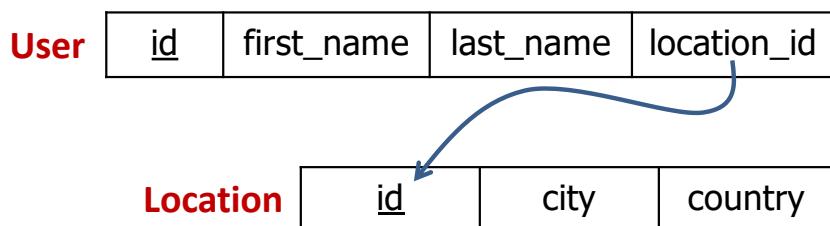
31

Thiết kế dữ liệu từ sơ đồ lớp

- Ta thấy có sự trùng lắp thông tin của trường city và country. Do đó ta sửa sơ đồ lớp như sau:



- Các bảng dữ liệu tương ứng



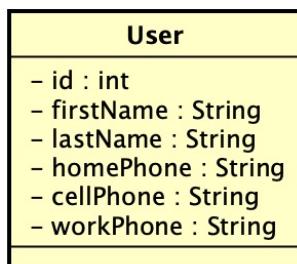
Dương Hữu Thành

32

32

Thiết kế dữ liệu từ sơ đồ lớp

- Tương tự giả sử User cần có thông tin điện thoại của người dùng bao gồm home phone, mobile phone, work phone, v.v. với lớp như sau:



Dương Hữu Thành

33

33

Thiết kế dữ liệu từ sơ đồ lớp

- Bảng dữ liệu tương ứng lớp này có thể sẽ xuất hiện nhiều giá trị null của các số điện thoại trong bảng dữ liệu, điều này lãng phí bộ nhớ.

id	firstName	lastName	homePhone	cellPhone	workPhone
1	Thanh	Duong		09844614891	(028) 38386603
2	Huynh	Le	(028) 62812345		
3	Khanh	Bui			(028) 38386603
4	Tran	Nguyen		09645618891	
5	Phap	Nguyen	028) 62812345		

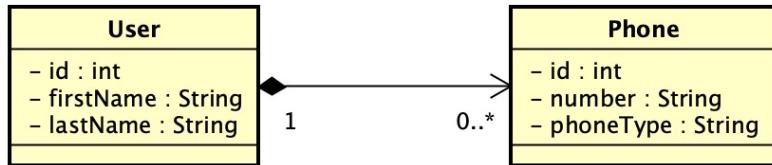
Dương Hữu Thành

34

34

Thiết kế dữ liệu từ sơ đồ lớp

- Thiết kế lại sơ đồ lớp



- Các bảng dữ liệu tương ứng

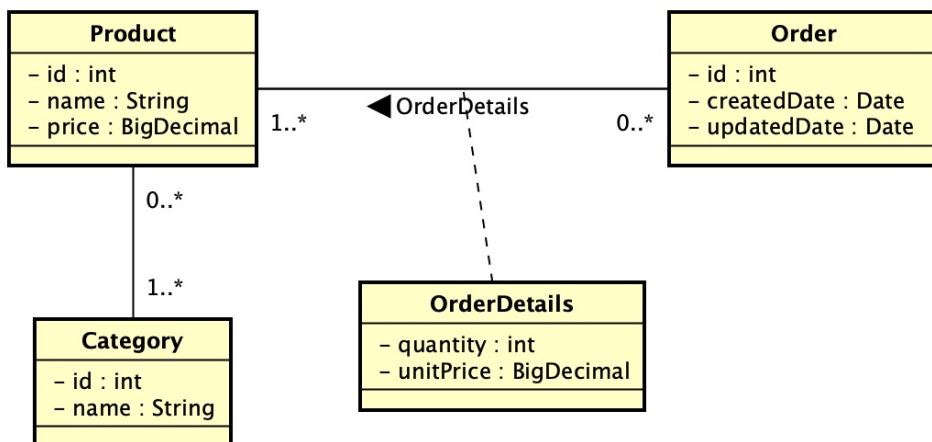
User	<u>id</u>	first_name	last_name	
Phone	<u>id</u>	number	phone_type	user_id

- Quan hệ kết hợp nhiều-nhiều

- Tạo một **bảng trung gian** với **2 khoá ngoại** lần lượt tham chiếu đến hai bảng trong mỗi quan hệ, **giá trị tổ hợp** của **2 khoá ngoại** này phải là duy nhất.
 - Trong bảng này chứa các thuộc tính quan hệ giữa hai lớp.

Thiết kế dữ liệu từ sơ đồ lớp

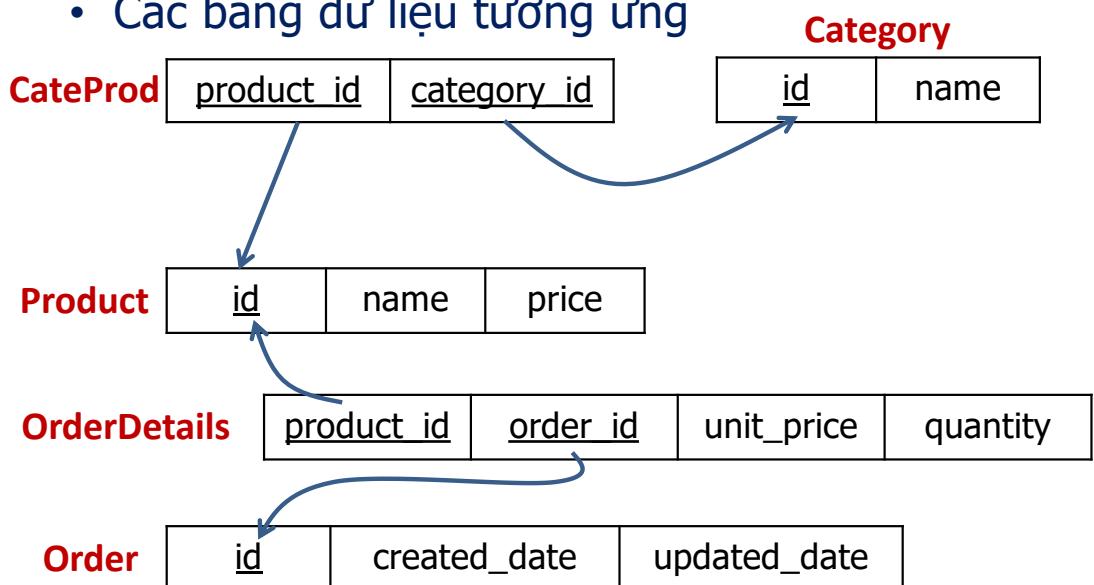
- Ví dụ sơ đồ lớp có quan hệ nhiều-nhiều giữa Product và Order, Product và Category.



Dương Hữu Thành

37

- Các bảng dữ liệu tương ứng



Dương Hữu Thành

38

Thiết kế dữ liệu từ sơ đồ lớp

- Quan hệ kế thừa
 - Lớp con thêm thuộc tính **định danh giống** thuộc **tính định danh** của lớp cha.
 - Thiết lập **quan hệ khoá ngoại** của thuộc tính này trong bảng đại diện lớp con tới bảng đại diện lớp cha.

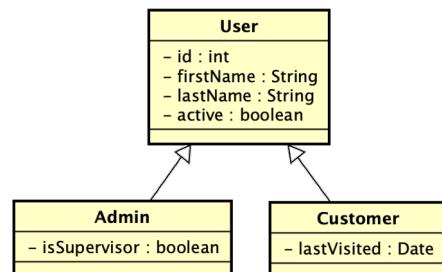
Dương Hữu Thành

39

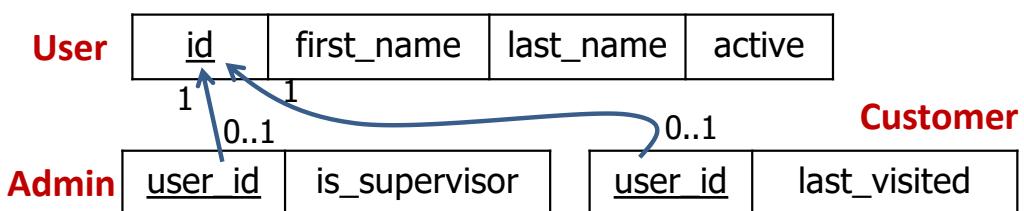
39

Thiết kế dữ liệu từ sơ đồ lớp

- Ví dụ ta có 2 loại người dùng trong hệ thống



- Các bảng dữ liệu tương ứng

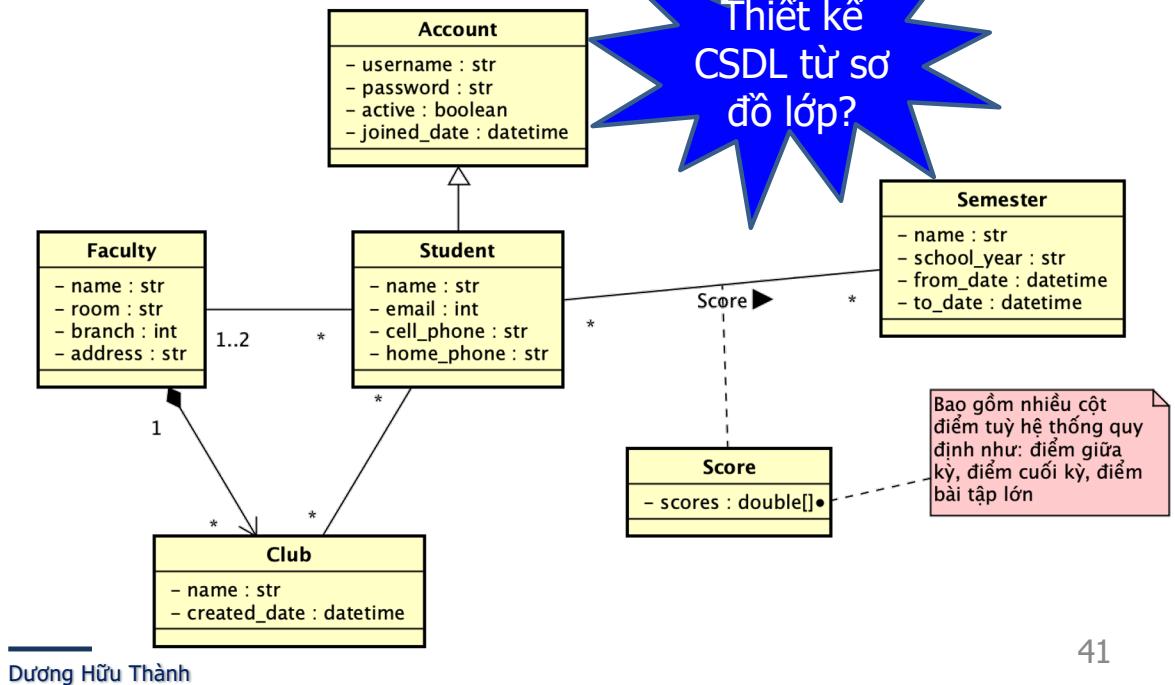


Dương Hữu Thành

40

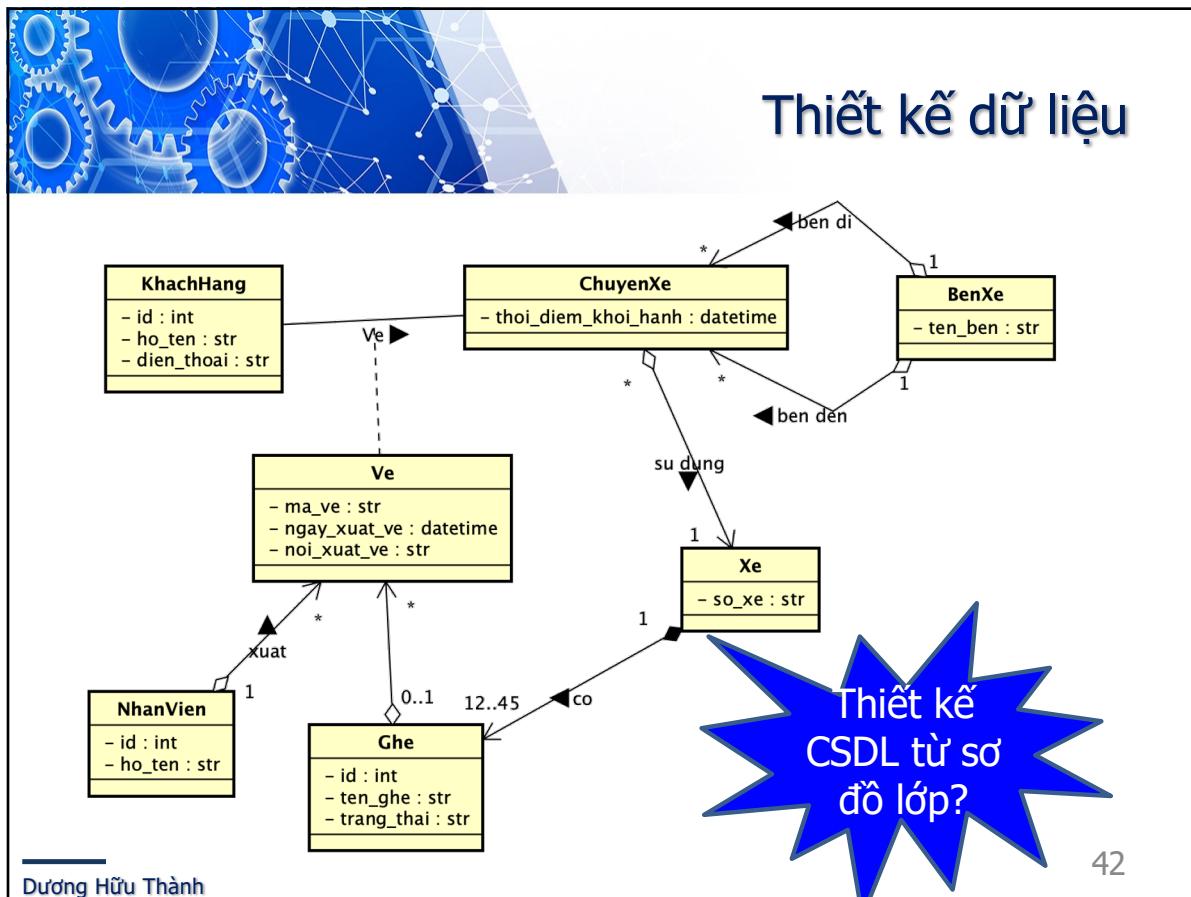
40

Thiết kế dữ liệu từ sơ đồ lớp



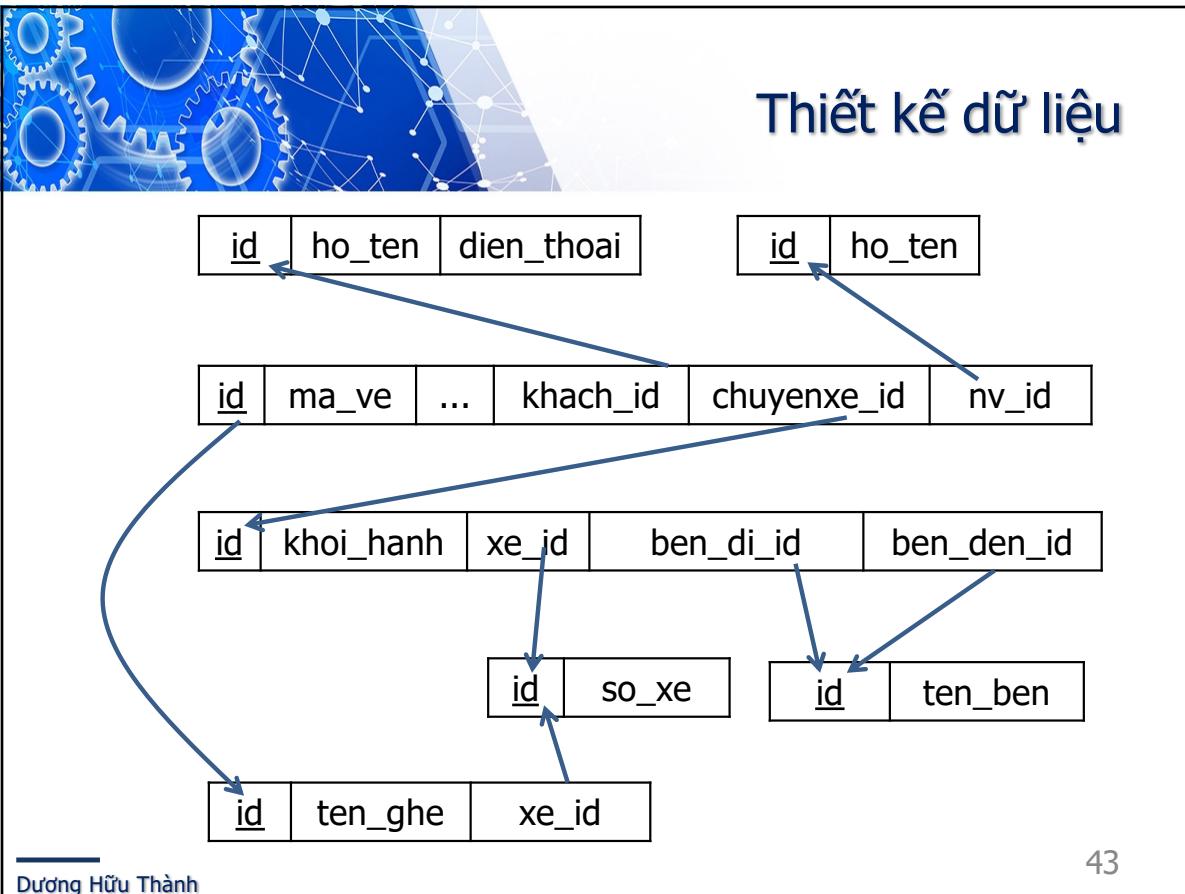
41

Thiết kế dữ liệu



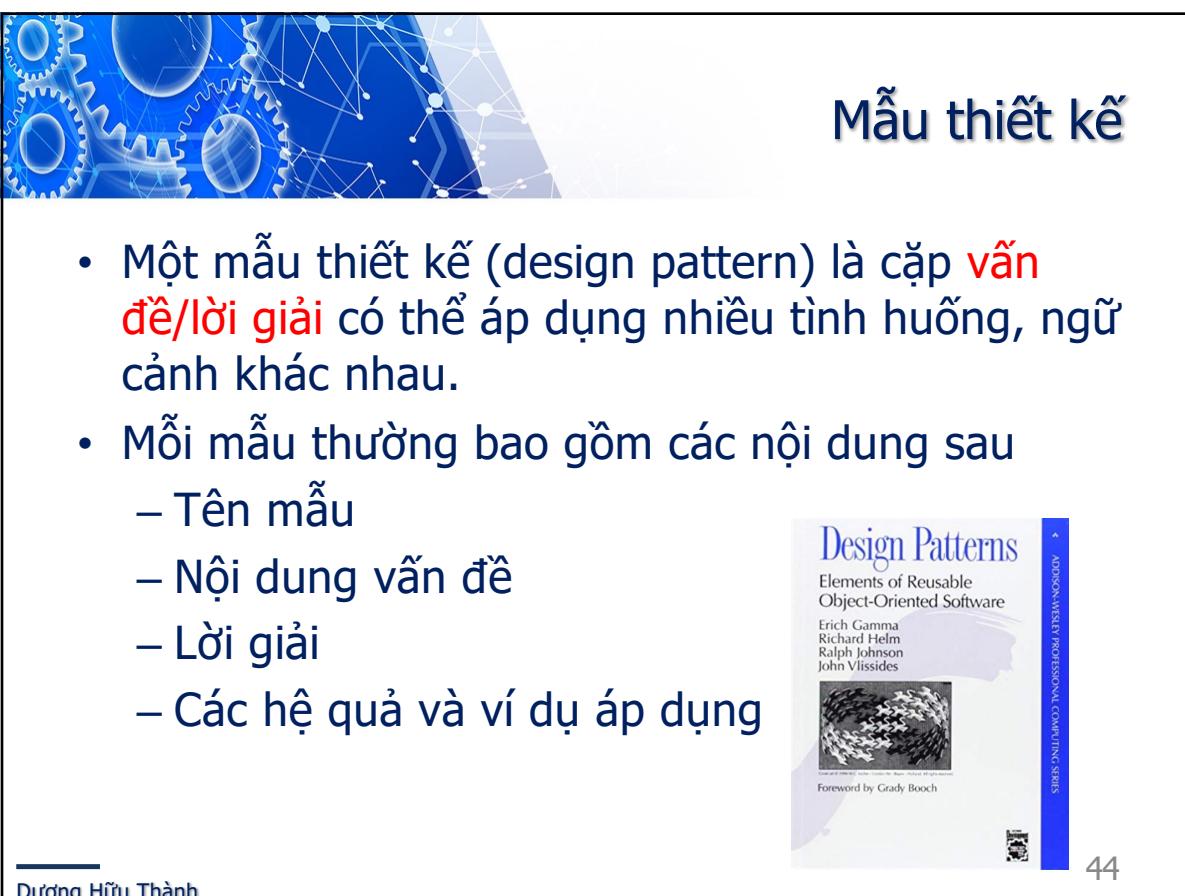
42

Thiết kế dữ liệu



Dương Hữu Thành

43



Dương Hữu Thành

44



Mẫu thiết kế

- Phổ biến nhất là **23 mẫu thiết kế** của 4 tác giả: Erich Gamma, Richard Helm, Ralph Johnson và John Vlissides, còn được gọi là mẫu GoF (Gang of Four) (Gamma et al., 1995).
- Các mẫu quan trọng khác được giới thiệu trong hàng loạt sách của các tác giả từ công ty công nghệ Siemens (Buschmann et al., 1996; Buschmann et al., 2007a; Buschmann et al., 2007b; Kircher and Jain, 2004; Schmidt et al., 2000)

Dương Hữu Thành

45

45



Mẫu thiết kế

- Các mẫu có ảnh hưởng rất lớn đến thiết kế hướng đối tượng, chúng là những **giải pháp đã được kiểm chứng** và được xem như các "**tử vong**" khi nói về thiết kế - có thể giải thích về thiết kế bằng cách mô tả mẫu được sử dụng.
- Cơ sở của các mẫu thiết kế là **kế thừa** (inheritance) và **đa hình** (polymorphism).

Dương Hữu Thành

46

46



Mẫu thiết kế GOF

- Gamma và các cộng sự chia phần “Nội dung vấn đề” thành 2 phần
 - **Motivation**: tính huống cụ thể thiết kế phần mềm sử dụng mẫu đang đề cập.
 - **Applicability**: các tình huống trong thiết kế có thể sử dụng mẫu đang đề cập.

Dương Hữu Thành

47

47



Mẫu thiết kế GOF

- Phần “Lời giải” chia thành
 - Cấu trúc (structure): mô tả mẫu bằng các ký hiệu đồ họa.
 - Các thành viên (participants): trình bày ý nghĩa và vai trò các lớp và đối tượng tham gia vào mẫu thiết kế.
 - Các cộng tác (collaborations)
 - Thực thi (implementation): chú ý cài đặt, mã nguồn minh họa.

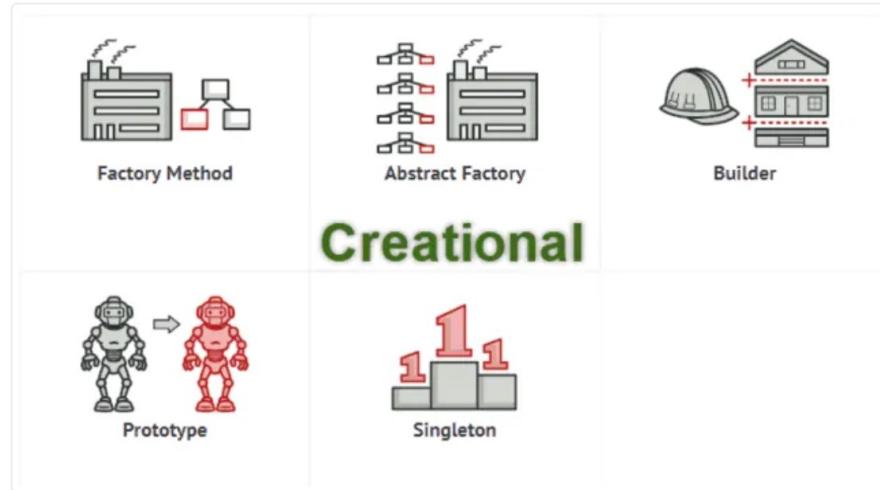
Dương Hữu Thành

48

48

Mẫu thiết kế GOF

- Creational Pattern: cách tạo thể hiện đối tượng.



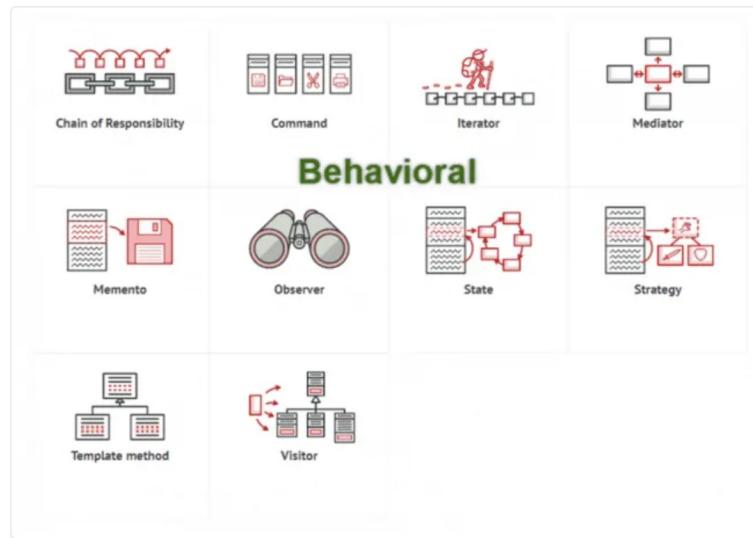
Dương Hữu Thành

49

49

Mẫu thiết kế GOF

- Behavioral Pattern: cách thức tương tác giữa các đối tượng.



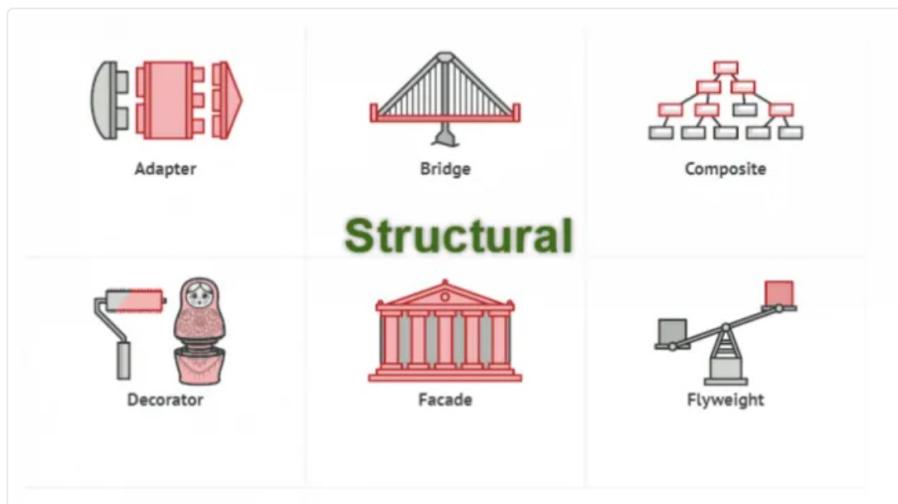
Dương Hữu Thành

50

50

Mẫu thiết kế GOF

- Structural Pattern: cách kết hợp, liên hệ các lớp đối tượng.



Dương Hữu Thành

51



Mẫu thiết kế GOF

- Áp dụng các mẫu thiết kế
 - Ta cần xác định **vấn đề gấp phải** khi thiết kế để tìm mẫu thích hợp.
 - Việc áp dụng mẫu thiết kế chỉ có ý nghĩa sử dụng ý tưởng các mẫu, có thể điều chỉnh thích hợp với hệ thống phát triển.
 - Việc sử dụng mẫu hiệu quả thường yêu cầu có kinh nghiệm trong thiết kế phần mềm để nhận ra những tình huống sử dụng mẫu.

Dương Hữu Thành

52

52

Mẫu Observer

- Tên mẫu: Observer
- Mô tả: định nghĩa mối quan hệ phụ thuộc 1 – n giữa các đối tượng, khi trạng thái một đối tượng thay đổi thì tất cả các đối tượng phụ thuộc nó sẽ được thông báo và cập nhật tự động.

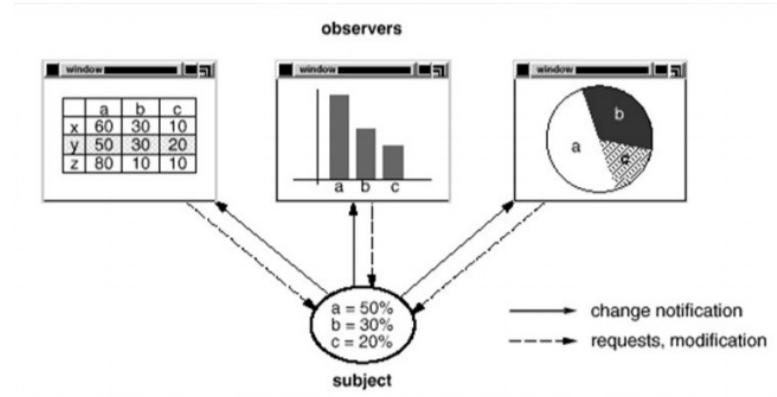
Dương Hữu Thành

53

53

Mẫu Observer

- Tình huống cụ thể sử dụng mẫu Observer



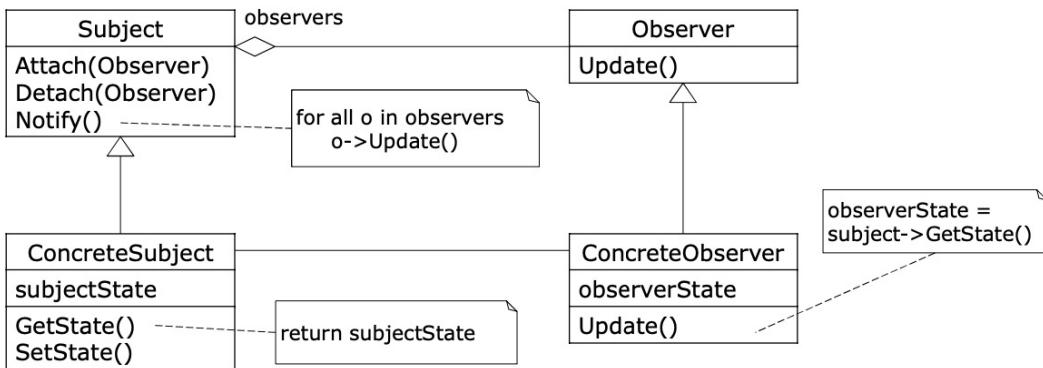
Dương Hữu Thành

54

54

Mẫu Observer

- Cấu trúc mẫu Observer



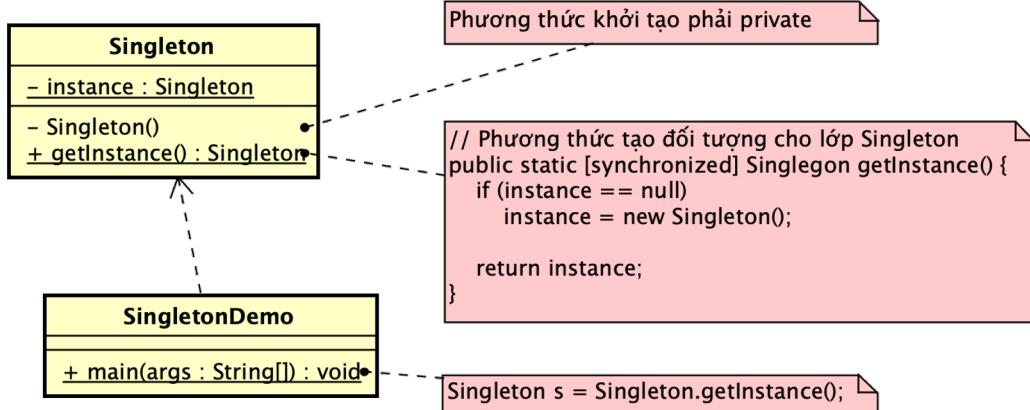
Dương Hữu Thành

55

55

Mẫu Singleton

- Mẫu này cho phép chỉ tạo duy nhất một thể hiện của lớp.



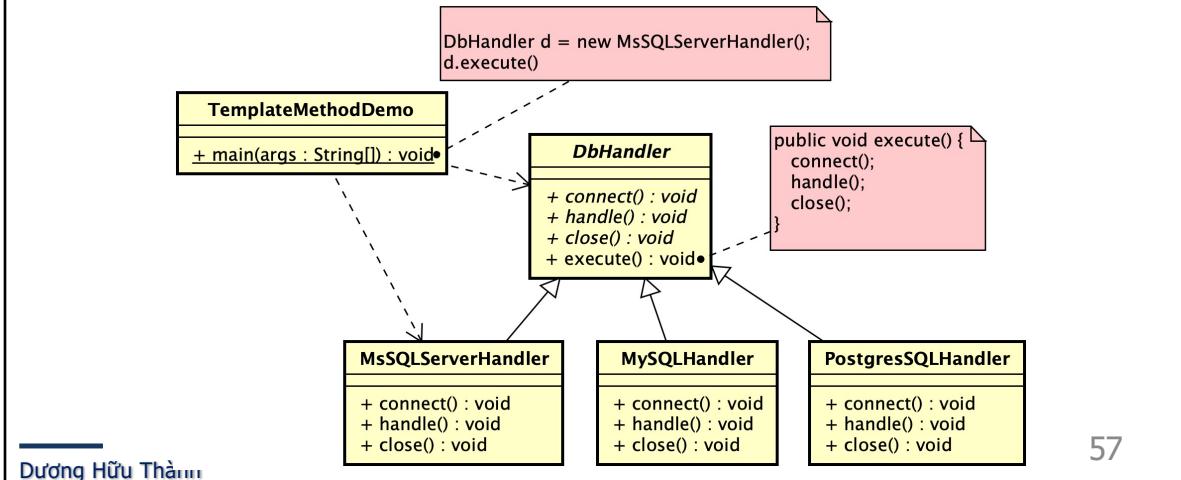
Dương Hữu Thành

56

56

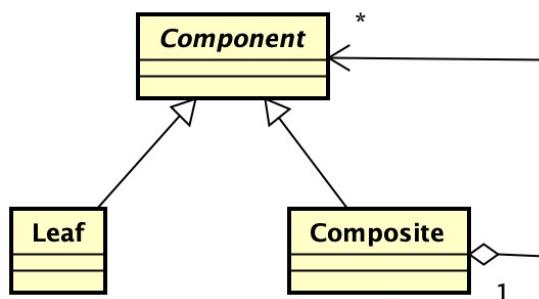
Mẫu Template Method

- Mẫu này định nghĩa khuôn mẫu thực thi chung, từng bước cụ thể trong khuôn mẫu sẽ do lớp con định nghĩa.



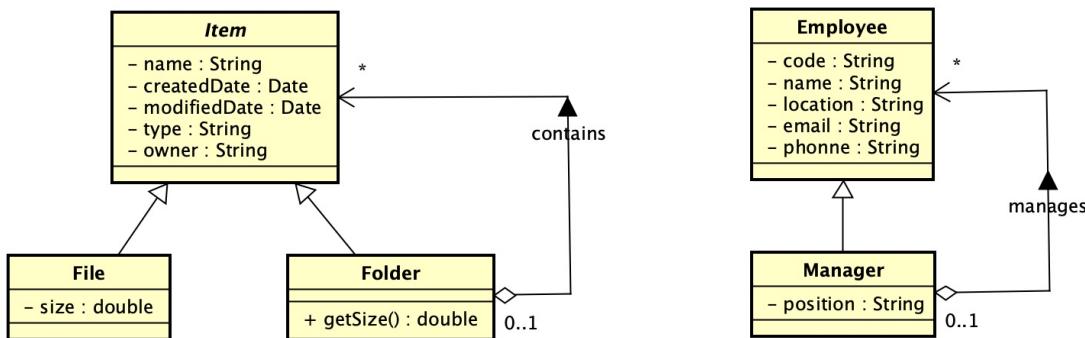
Mẫu Composite

- Mẫu này cho phép tạo ra cấu trúc cây phân tầng dạng đệ quy.



Mẫu Composite

- Ví dụ



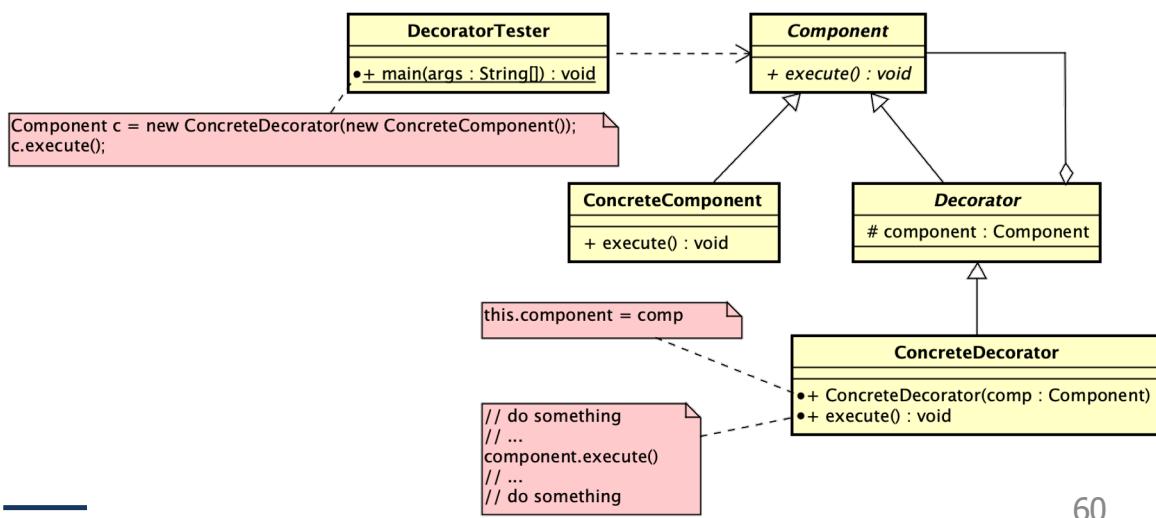
Dương Hữu Thành

59



Mẫu Decorator

- Mẫu này cho phép hành vi của một đối tượng có thể mở rộng linh hoạt lúc thực thi.

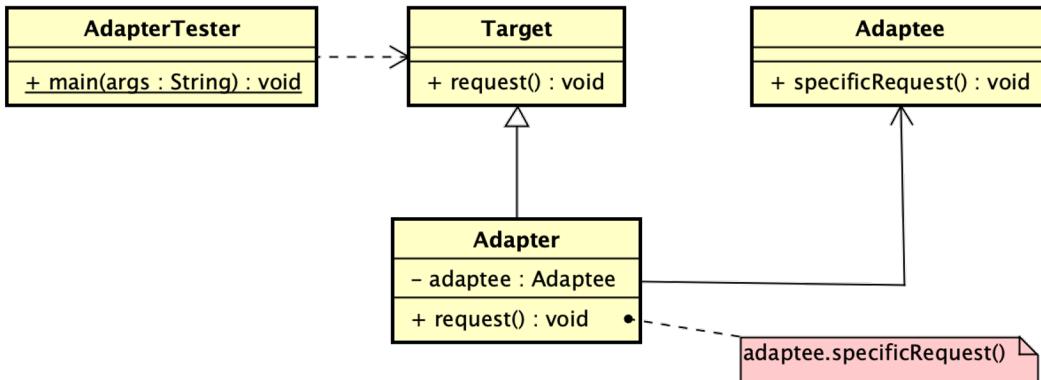


Dương Hữu Thành

60

Mẫu Adapter

- Mẫu này cho phép hai hay nhiều thành phần không tương thích có thể làm việc với nhau.



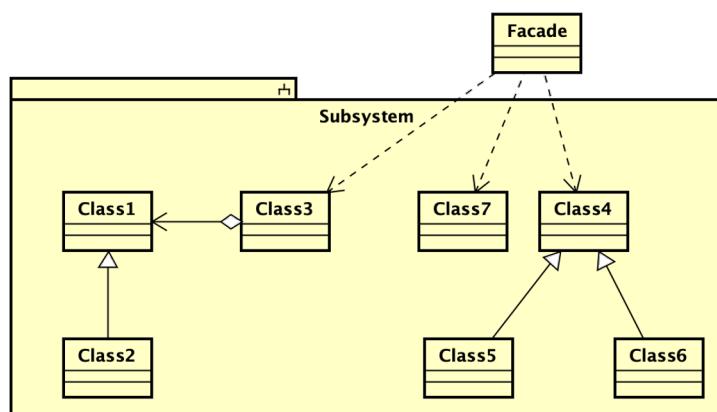
Dương Hữu Thành

61



Mẫu Facade

- Mẫu này cung cấp một interface thống nhất cho các interface trong hệ thống con giúp cho hệ thống con được sử dụng dễ dàng hơn.



Dương Hữu Thành

62



Nguyên lý thiết kế phần mềm

- DRY (Don't Repeat Yourself)
 - Mã nguồn lặp dẫn đến dễ phát sinh lỗi, khó khăn trong việc bảo trì phần mềm, thay đổi chương trình.
- KISS (Keep It Simple Stupid!)
 - Sự phức tạp không cần thiết nên bỏ qua.
- YAGNI (You Aren't Gonna Need It)
 - Chỉ thực hiện các chức năng thật sự cần thiết.

Dương Hữu Thành

63

63



Nguyên lý thiết kế phần mềm

- SoC (Separation of Concerns)
 - Tách hệ thống thành các component càng ít phụ thuộc nhau càng tốt.
 - Khi các thành phần được tích hợp lại sẽ tương tác với nhau thông qua interface mà không cần quan tâm chi tiết bên trong của component được hiện thực như thế nào.

Dương Hữu Thành

64

64



Nguyên lý thiết kế phần mềm

- **Loose Coupling:** giữa các component càng ít phụ thuộc nhau càng tốt.
 - Ngược lại gọi là tight coupling, điều này sẽ dẫn đến khó khăn trong việc tái sử dụng component, cũng như chỉnh sửa hay bảo trì hệ thống.
- **High Cohesion:** các thành phần trong một component càng phụ thuộc nhau càng tốt.

Dương Hữu Thành

65

65



Nguyên lý SOLID

- SOLID là từ viết tắt của 5 nguyên tắc được đề xuất bởi Robert C Martin năm 2000, tuân thủ 5 nguyên tắc này giúp chương trình dễ đọc, dễ bảo trì, dễ dàng mở rộng và nâng cấp.
- Các nguyên lý của SOLID
 - Single Responsibility Principle
 - Open/Closed Principle
 - Liskov Substitution Principle
 - Interface Segregation Principle
 - Dependency Inversion Principle

Dương Hữu Thành

66

66



Nguyên lý SOLID

- **Single Responsibility Principle:** một module hoặc class chỉ nên đảm nhận **một nhiệm vụ duy nhất**.
 - Một module hay lớp đảm nhận nhiều nhiệm vụ dễ sinh ra lỗi, việc thay đổi mã nguồn khó khăn, khó bảo trì, khó khoanh vùng lỗi khi phát hiện vấn đề.

Dương Hữu Thành

67

67



Nguyên lý SOLID

- **Open/Close Principle:** dễ dàng mở rộng một lớp (**open for extension**), nhưng không thay đổi nội dung bên trong nó (**close for modification**).
 - Nguyên lý này ưu tiên áp dụng cho các thành phần thường xuyên mở rộng, nâng cấp.
 - Nó giúp hạn chế thay đổi chương trình, không ảnh hưởng chương trình đang hoạt động khi mở rộng, nâng cấp chương trình.

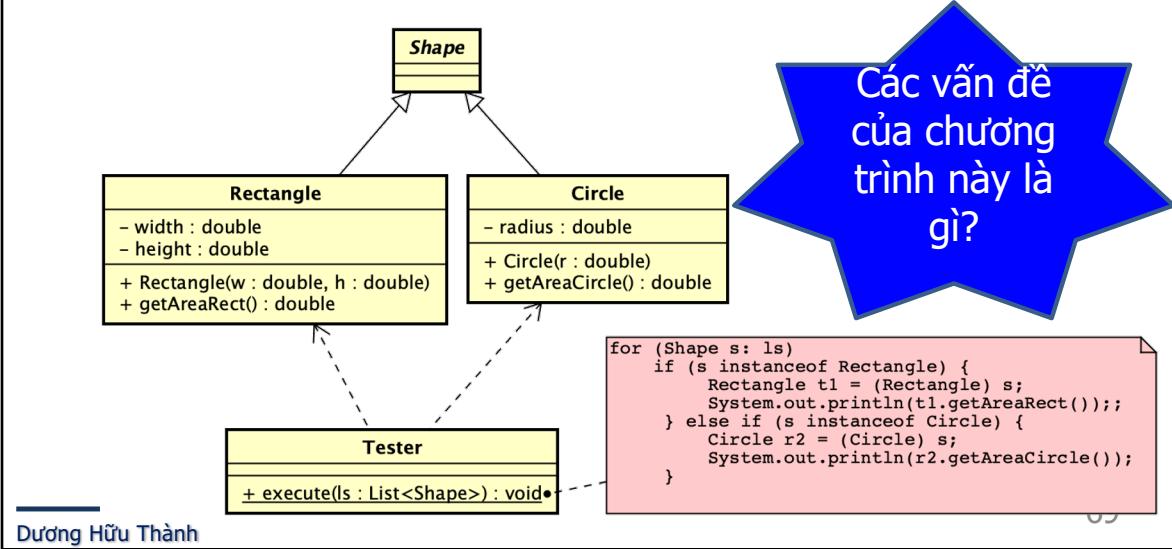
Dương Hữu Thành

68

68

Nguyên lý SOLID

- Ví dụ bài toán tính diện tích một hình nào đó.
- Chương trình được thiết kế ban đầu như sau:



Dương Hữu Thành

69

Nguyên lý SOLID

- Chương trình cần phải chỉnh sửa phương thức `execute()` của lớp `Tester()` mỗi khi thêm một hình mới cần tính diện tích vào chương trình, điều này dẫn đến nhiều rủi ro cho chương trình đang hoạt động ổn định khi mở rộng, nâng cấp.
→ Vi phạm nguyên lý Open/Close.

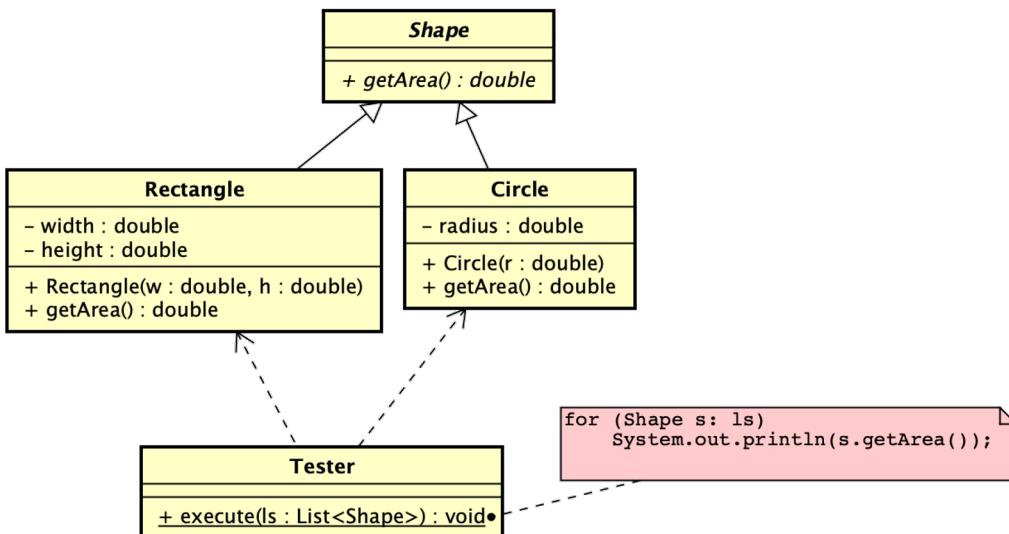
Dương Hữu Thành

70

70

Nguyên lý SOLID

- Chương trình thiết kế lại như sau:



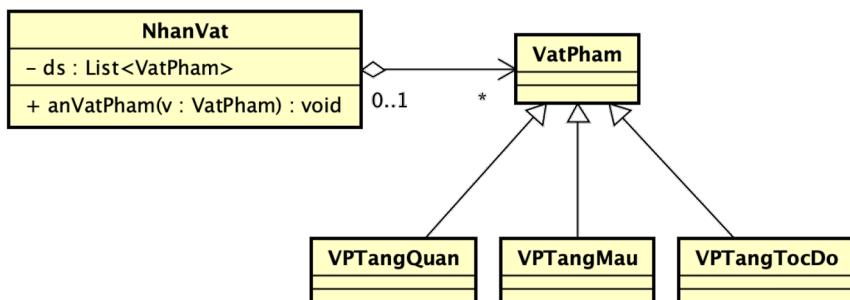
Dương Hữu Thành

71

71

Nguyên lý SOLID

- Ví dụ một trò chơi cho phép người chơi ăn các vật phẩm khác nhau. Để đảm bảo quy tắc này, ta thiết kế sơ đồ lớp như sau:



- Khi có vật phẩm mới, ta chỉ việc tạo lớp kế thừa lại **VatPham** và hiện thực cái riêng của nó, không ảnh hưởng những thứ đang hoạt động.

Dương Hữu Thành

72

72

Nguyên lý SOLID

- **Liskov Substitution Principle:** các đối tượng lớp con có thể thay đổi các đối tượng lớp cha nhưng không gây ra lỗi.
- Ví dụ:
 - Một công ty có 2 loại nhân viên A, B có cách thức tính lương khác nhau, trong đó các nhân viên phải điểm danh mỗi ngày làm, chức năng yêu cầu có mã nhân viên để lưu trữ.
 - Sau đó bổ sung cộng tác viên tính lương chỉ theo số ngày làm, không phải nhân viên chính thức nên không có mã số để điểm danh.

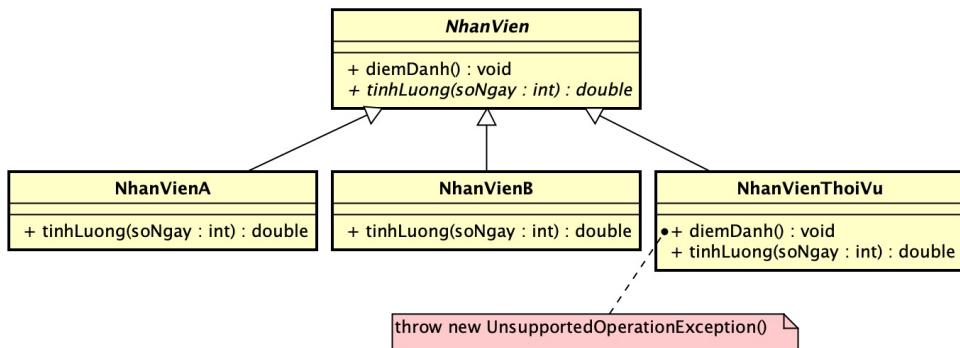
Dương Hữu Thành

73



Nguyên lý SOLID

- Sơ đồ lớp thiết kế như sau



- Khi đó đoạn chương trình sau có vấn đề:

```
NhanVien n = new NhanVienThoiVu();
n.diemDanh();
```

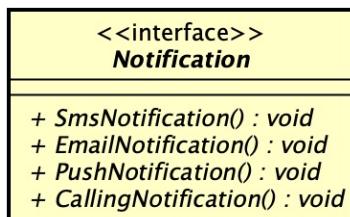
Dương Hữu Thành

74



Nguyên lý SOLID

- **Interface Segregation Principle:** nên tách một interface lớn thành các interface con với các mục đích khác nhau.
- Ví dụ: giao diện Notification đảm nhiệm nhiều cách thức gửi thông báo, lớp con phải hiện thực tất cả các cách thức này mặc dù có thể chỉ cần sử dụng một trong các cách thức đó.



Dương Hữu Thành

75



Nguyên lý SOLID

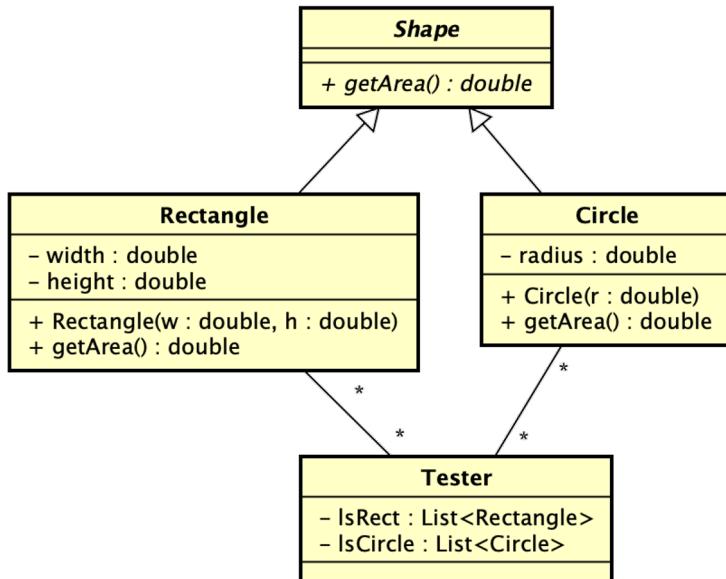
- **Dependency Inversion Principle:**
 - Các thành phần phần mềm **không** nêu phụ thuộc cái cụ thể, cái riêng mà nêu phụ thuộc cái tổng quát, cái chung của nó.
 - Nói cách khác, các thành phần cấp cao (high-level) **không** nêu phụ thuộc các thành phần cấp thấp (low-level), tất cả chỉ nêu phụ thuộc lớp trừu tượng hoặc giao diện.

Dương Hữu Thành

76

Nguyên lý SOLID

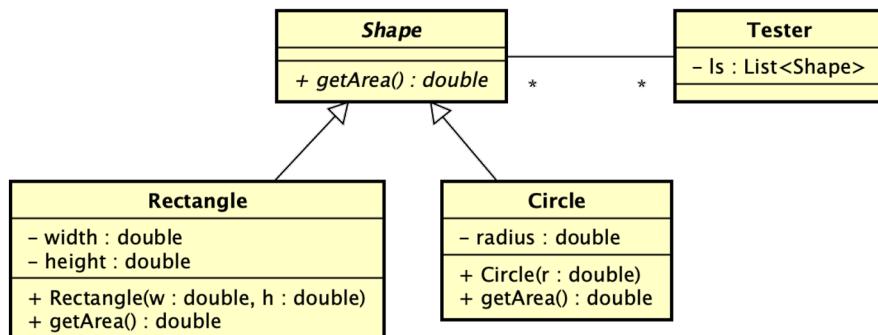
- Ví dụ xét sơ đồ:



Dương Hữu Thành

77

- Lớp QuanLyHinh phụ thuộc hai lớp con khiến chương trình thiếu linh hoạt, khó mở rộng khi có thêm các hình mới.
- Thiết kế lại chỉ phụ thuộc lớp trừu tượng.



Dương Hữu Thành

78



Q&A