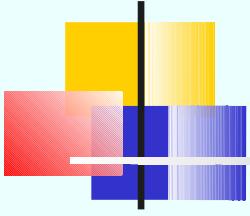


# Giới thiệu môn học Công nghệ phần mềm

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

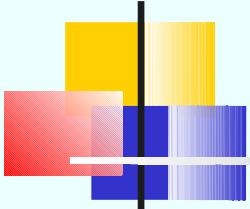


# Công cụ hỗ trợ

---

Visual Paradigm (VP) for UML: download bản free tại:

<http://www.visual-paradigm.com/product/vpuml/>



# Các khái niệm liên quan (1)

---

Software : phần mềm

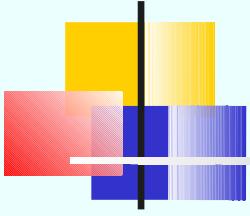
Software engineering: công nghệ / kỹ nghệ  
phần mềm

Software process: tiến trình phần mềm

Software development: phát triển phần mềm

Software life-cycle models: mô hình vòng đời  
phần mềm

Phase: một pha, một bước, một giai đoạn  
phát triển phần mềm



# Các khái niệm liên quan (2)

---

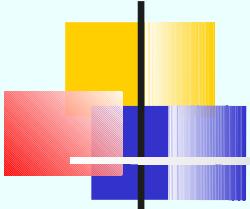
Developer: người phát triển phần mềm

Development team: đội phát triển phần mềm

Quality Assurance (QA): đội đảm bảo chất lượng phần mềm

User: người sử dụng phần mềm

Client: người đặt hàng phần mềm



# Các khái niệm liên quan (3)

---

Methodology, paradigm: phương pháp luận, mô hình lặp lại các bước để phát triển phần mềm

Cost: chi phí phát triển phần mềm

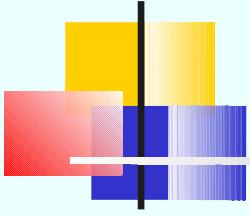
Price: giá bán của phần mềm

Technique: kỹ thuật

Mistake, fault, failure, error: lỗi

Defect: các thiếu sót

Bug: lỗi trong code



# Các khái niệm liên quan (4)

---

Requirements: yêu cầu, lấy yêu cầu

Description: đặc tả yêu cầu

Analysis: phân tích yêu cầu / phần mềm

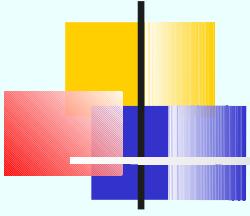
Design: thiết kế

Implementation: cài đặt

Delivery: triển khai

Maintenance: bảo trì

Testing: kiểm thử

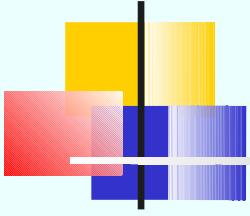


# Các khái niệm liên quan (5)

---

Object-oriented software: phần mềm hướng đối tượng

Object-oriented software engineering: công nghệ phần mềm hướng đối tượng



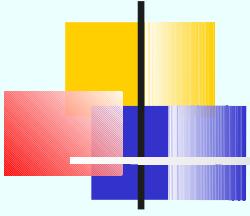
# Một số câu hỏi (1)

---

Phân biệt client và user?

Trả lời:

...



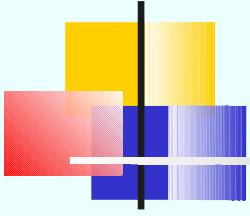
# Một số câu hỏi (2)

---

Phân biệt cost và price?

Trả lời:

...



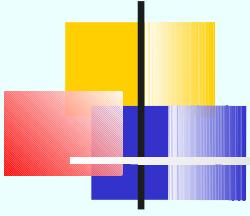
# Một số câu hỏi (3)

---

Phân biệt fault, failure và bug?

Trả lời:

...



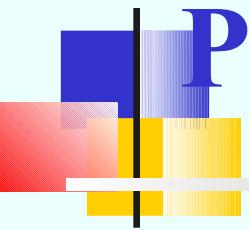
# Một số câu hỏi (4)

---

Phân biệt việc phát triển phần mềm và sản xuất phần mềm?

Trả lời:

...

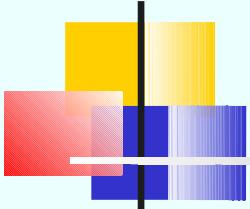


## Công nghệ phần mềm

# Phạm vi của công nghệ phần mềm

*Giảng viên: TS. Nguyễn Mạnh Hùng*

*Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

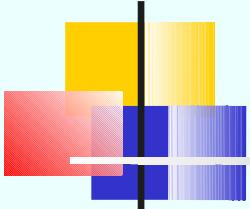


# Khía cạnh lịch sử (1)

---

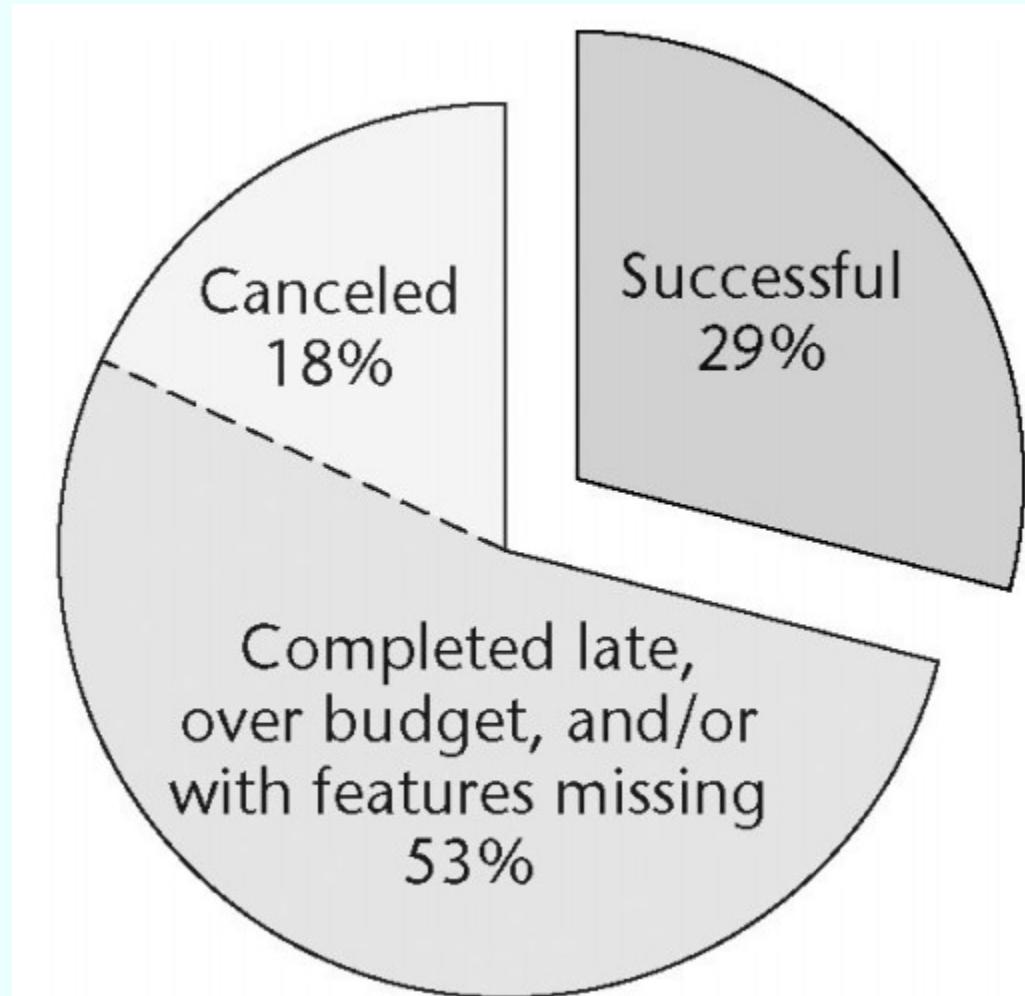
Năm 1968, NATO đã nhóm họp ở Đức để tìm giải pháp thoát khỏi khủng hoảng phần mềm:

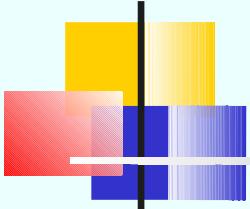
- Phần mềm hoàn thành và chuyển giao trễ thời hạn
- Vượt chi phí dự đoán
- Vẫn còn tiềm tàng lỗi



# Khía cạnh lịch sử (2)

Dữ liệu thống kê từ  
9236 dự án phần  
mềm năm 2004:





# Khía cạnh lịch sử (3)

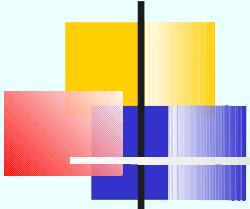
---

Khảo sát năm 2000 trên các cty phần mềm:

- 78% dự án có tranh chấp đều kết thúc bằng kiện tụng

Với các dự án bị kiện:

- 67% dự án bị kiện do chức năng không đúng yêu cầu khách hàng
- 56% dự án bị kiện vì dời hẹn giao sản phẩm quá nhiều lần
- 45% dự án bị kiện là do còn lỗi nghiêm trọng đến mức sản phẩm không sử dụng được

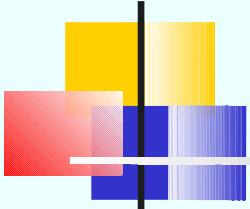


# Khía cạnh lịch sử (4)

---

Sự khủng hoảng phần mềm không thể giải quyết dứt điểm:

- Nó còn có thể tồn tại lâu dài
- Hiện nay vẫn chưa có dự đoán chính xác thời điểm kết thúc

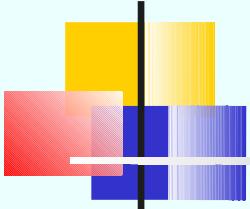


# Khía cạnh kinh tế

---

Xem xét khía cạnh kinh tế của các tình huống:

- Có ngôn ngữ lập trình mới, có nên dùng ngôn ngữ mới này cho dự án?
- Có công cụ phân tích thiết kế mới, dễ dùng hơn, có nên sử dụng cho dự án?
- Có công nghệ code/test mới, có nên ứng dụng vào dự án?



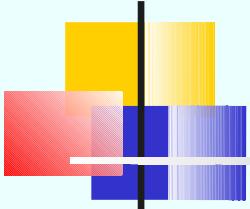
# Khía cạnh bảo trì (1)

---

Mô hình vòng đời phát triển phần mềm:

- Ví dụ: mô hình thác nước (waterfall)

1. Requirements phase
2. Analysis (specification) phase
3. Design phase
4. Implementation phase
5. Postdelivery maintenance
6. Retirement

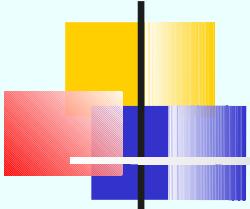


# Khía cạnh bảo trì (2)

---

Các dạng bảo trì:

- Bảo trì sửa chữa (corrective)
- Bảo trì phát triển (perfective)
- Bảo trì tương thích (adaptive)



# Khía cạnh bảo trì (3)

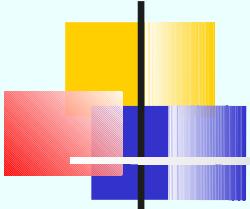
---

Khi nào thì coi một hành động là của pha bảo trì?

- **Định nghĩa cổ điển** (dựa trên thời gian): một hành động là của pha bảo trì khi nó thực hiện sau khi bàn giao và cài đặt sản phẩm

Ví dụ:

- Nếu một lỗi được phát hiện sau khi bàn giao phần mềm thì việc sửa lỗi là của pha bảo trì
- Nếu cùng lỗi đó nhưng được phát hiện trước khi bàn giao phần mềm thì việc sửa lỗi thuộc pha cài đặt



# Khía cạnh bảo trì (4)

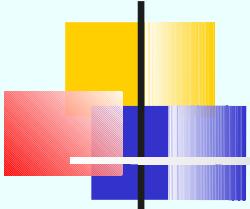
---

Khi nào thì coi một hành động là của pha bảo trì?

- Định nghĩa hiện đại: một hành động là của pha bảo trì khi nó làm thay đổi phần mềm vì lí do hoàn thiện hay tương thích

Ví dụ:

- Nếu khách hàng bổ sung thêm yêu cầu từ pha phân tích, thiết kế hay cài đặt thì việc thay đổi đó sẽ được coi là bảo trì sản phẩm

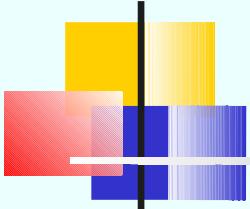


# Khía cạnh bảo trì (5)

---

Tầm quan trọng của pha bảo trì:

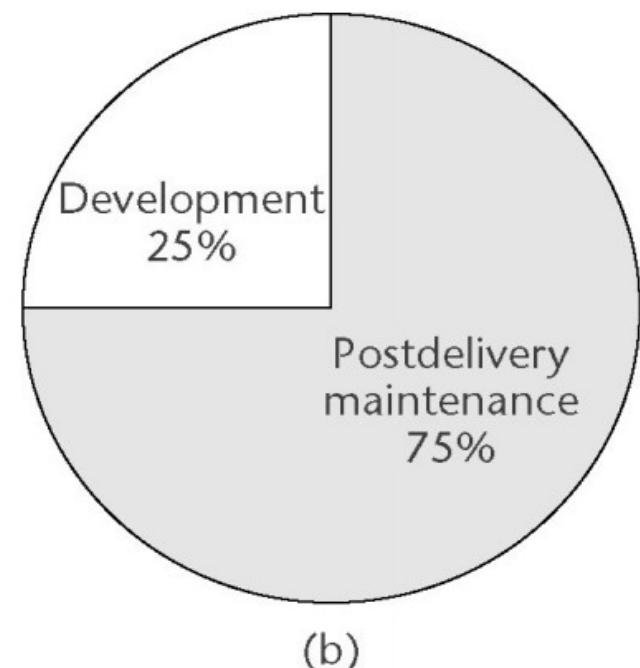
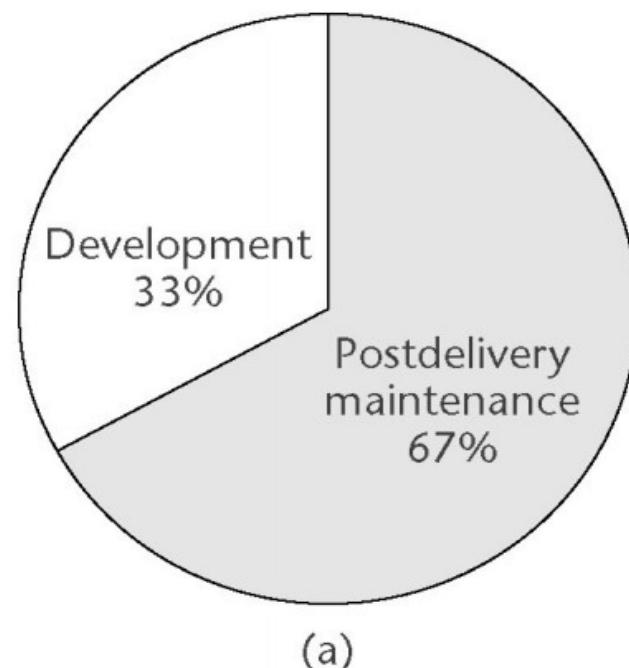
- Phần mềm không tốt thì sẽ bị vứt bỏ, chứ không được bảo trì
- Chỉ những phần mềm tốt mới được bảo trì, thời gian bảo trì có thể 10- 20 năm, có thể cả đời
- Bản thân phần mềm là một công cụ hỗ trợ, hoặc phương tiện làm việc, do đó nó sẽ thay đổi thường xuyên theo yêu cầu công việc

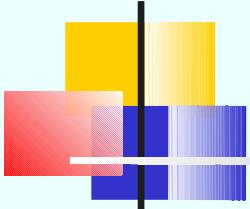


# Khía cạnh bảo trì (6)

Thời gian (chi phí) cho bảo trì luôn chiếm tỉ trọng lớn nhất:

- (a): 1976 – 1981
- (b): 1992 - 1998

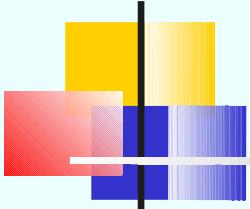




# Khía cạnh bảo trì (7)

Thời gian (chi phí) cho các pha gần như không thay đổi nhiều:

	Various Projects between 1976 and 1981	132 More Recent Hewlett-Packard Projects
Requirements and analysis (specification) phases	21%	18%
Design phase	18	19
Implementation phase		
Coding (including unit testing)	36	34
Integration	24	29

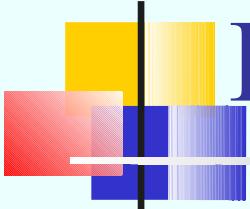


# Khía cạnh bảo trì (8)

---

Chi phí tương quan giữa các pha:

- Nếu giảm 10% chi phí cho pha cài đặt → sẽ giảm được khoảng 0.85% chi phí cho dự án
- Nếu giảm 10% chi phí cho bảo trì → sẽ giảm được 7.5% chi phí toàn bộ dự án!



# Khía cạnh phân tích- thiết kế (1)

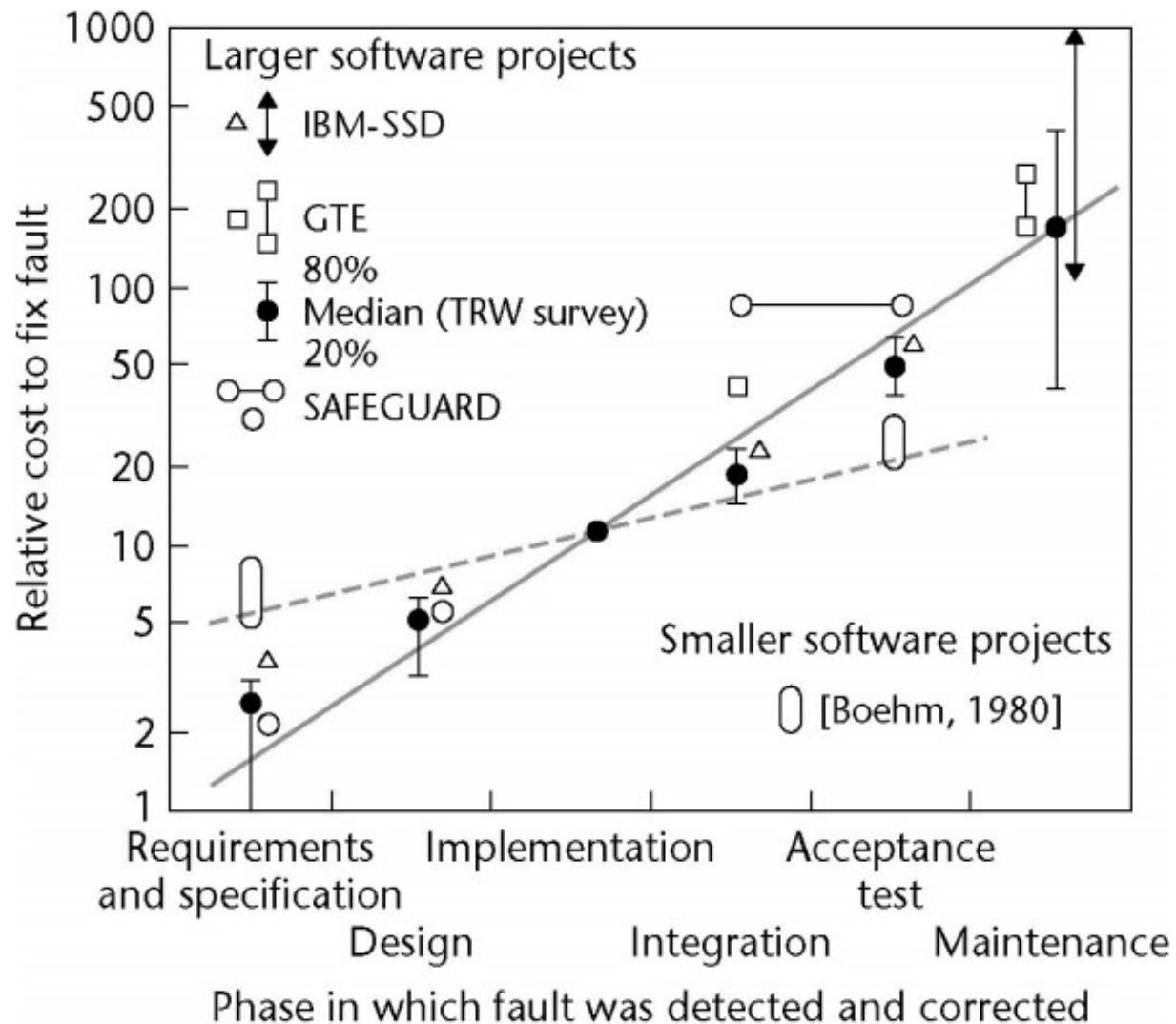
---

Nguyên tắc cơ bản:

- Lỗi được phát hiện càng sớm thì chi phí để sửa lỗi càng thấp!
- Và ngược lại!

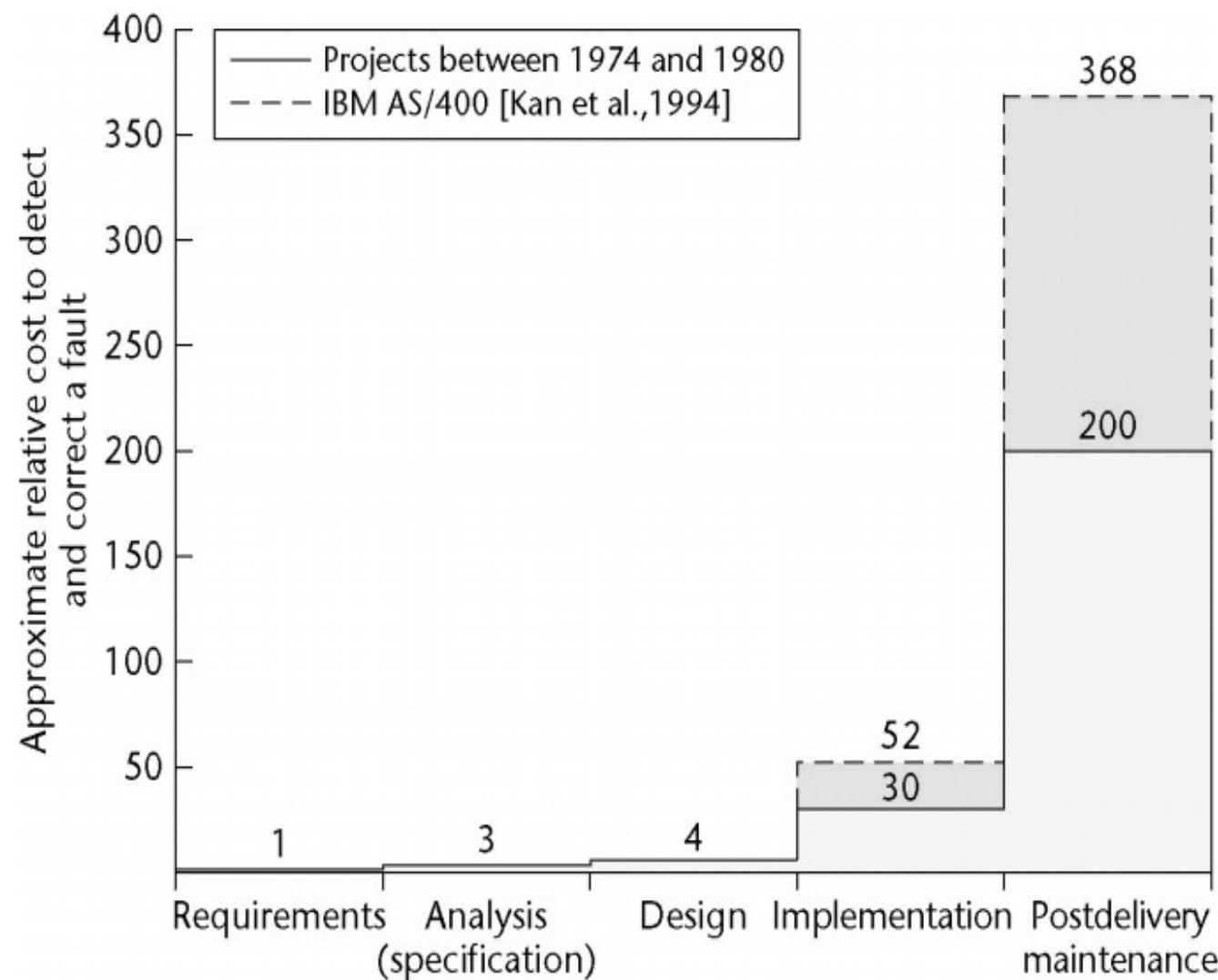
# Khía cạnh phân tích- thiết kế (2)

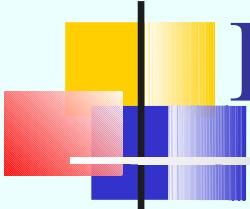
Ví dụ:



# Khía cạnh phân tích- thiết kế (3)

Ví dụ (tt):





# Khía cạnh phân tích- thiết kế (4)

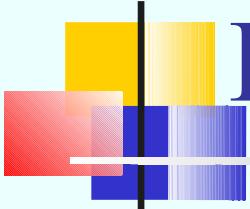
---

Để sửa một lỗi phát hiện sớm trong các pha yêu cầu, phân tích, và thiết kế:

- Chỉ cần thay đổi tài liệu các pha tương ứng

Để sửa một lỗi phát hiện muộn trong cài đặt hoặc bảo trì:

- Lần ngược lại các pha trước để sửa lại tài liệu
- Sửa lại code vì phân tích thiết kế đã bị sửa
- Test lại phần sửa/ test phần tương thích với phần còn lại
- Cài đặt lại hệ thống cho khách hàng



# Khía cạnh phân tích- thiết kế (5)

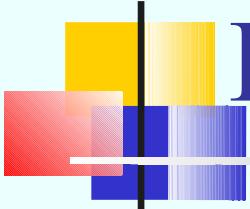
---

Thông kê cho thấy:

- 60-70% lỗi phát hiện ra là nằm trong các pha yêu cầu, phân tích, và thiết kế
- Nhưng thời điểm phát hiện ra các lỗi đấy là trong pha cài đặt và bảo trì

Ví dụ của công ty Jet Propulsion Laboratory:

- 1.9 lỗi/ trang đặc tả (specification)
- 0.9 lỗi/ trang thiết kế
- 0.3 lỗi/ trang code

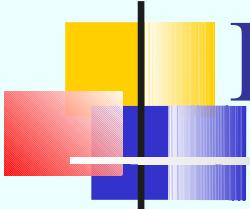


# Khía cạnh phân tích- thiết kế (6)

---

## Kết luận:

- Phải cải thiện chất lượng của pha lấy yêu cầu, phân tích và thiết kế
- Phát hiện lỗi càng sớm càng tốt
- Giảm thiểu tổng số lỗi phát hiện của toàn dự án



# Khía cạnh nhóm phát triển

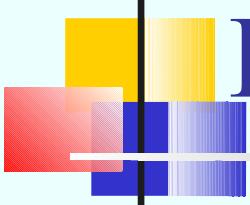
---

Có thể:

- Phát triển một phần mềm lớn, với chỉ một người làm tất cả các khâu!

Nhưng, phần mềm thường phát triển bởi một nhóm phát triển:

- Vấn đề tương tác, tích hợp giữa các modul
- Vấn đề giao tiếp và cộng tác giữa các thành viên của nhóm



# Khía cạnh lập kế hoạch

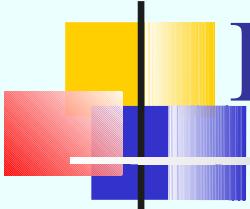
---

Câu hỏi:

- Tại sao không có pha lập kế hoạch?

Trả lời:

- ...



# Khía cạnh kiểm thử

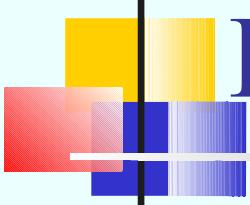
---

Câu hỏi:

- Tại sao không có pha kiểm thử (test)?

Trả lời:

- ...



# Khía cạnh làm tài liệu

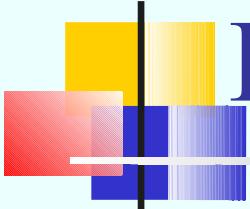
---

Câu hỏi:

- Tại sao không có pha làm tài liệu?

Trả lời:

- ...



# Kết luận

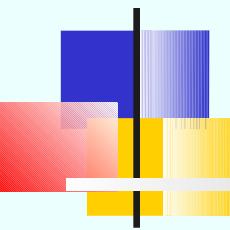
---

Để phát triển được một phần mềm không dễ dàng:

- Các vấn đề về kinh tế
- Các vấn đề về kỹ thuật
- Các vấn đề cho mỗi giai đoạn phát triển
- Các vấn đề về con người

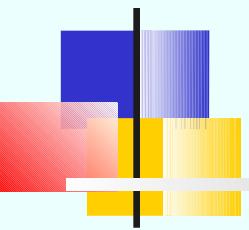
Với mục đích:

- Phát hiện lỗi càng sớm càng tốt
- Giảm thiểu được số lượng lỗi
- Giảm thiểu thời gian (và chi phí) phát triển



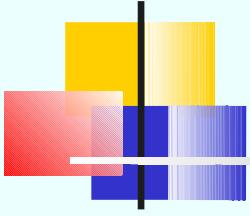
# Questions?

---



# Công nghệ phần mềm Tiến trình phần mềm

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

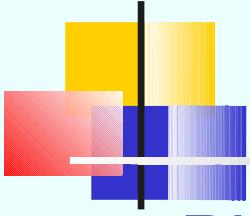


# Requirement workflow (1)

---

Mục đích:

- Xác định rõ cái mà khách hàng cần
- Không phải cái khách hàng muốn

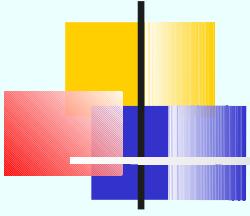


# Requirement workflow (2)

---

Phương pháp:

- Xác định rõ hiểu lĩnh vực ứng dụng của phần mềm:
  - Làm rõ các khái niệm chuyên ngành trong lĩnh vực tương ứng
- Xây dựng mô hình nghiệp vụ của khách hàng:
  - Làm việc với chuyên gia nghiệp vụ
  - Sử dụng công cụ UML
  - Đánh giá tính khả thi: kĩ thuật, chi phí...

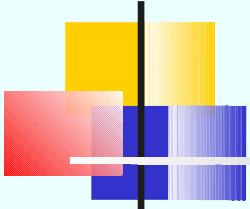


# Requirement workflow (3)

---

Kết quả cần đạt được:

- Thời hạn giao sản phẩm (deadline)
- Độ tin cậy (reliability)
- Chi phí (cost)
- Ngoài ra còn phải thống nhất một số yêu cầu khác: portability, respond time, parallel running

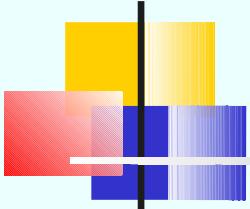


# Analysis workflow (1)

---

Mục tiêu:

- Phân tích, phân rã và mịn hóa yêu cầu của khách hàng
- Tại sao không làm việc này ngay trong pha requirement?



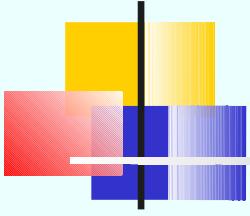
# Analysis workflow (2)

---

Vấn đề ngôn ngữ:

- Tài liệu phải thống nhất được cả hai bên khách hàng và đội phát triển
- Khách hàng chỉ hiểu ngôn ngữ tự nhiên: ngôn ngữ không chính xác
- Đội phát triển chỉ làm việc được trên ngôn ngữ kĩ thuật: chính xác và khoa học

→ Tạo ra hai loại tài liệu đặc tả: bằng ngôn ngữ tự nhiên và bằng ngôn ngữ kĩ thuật

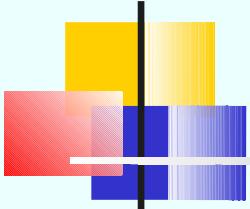


# Analysis workflow (3)

---

→ Tại sao việc làm tài liệu đặc tả lại quan trọng  
đến vậy?

- Trả lời: ...



# Analysis workflow (4)

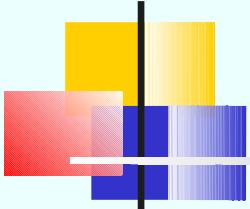
---

Kết quả cần đạt được:

- Tài liệu đặc tả đúng yêu cầu của khách hàng

Yêu cầu về tài liệu không được:

- Mâu thuẫn (contradictions)
- Có khái niệm và định lượng mờ (omissions)
- Không đầy đủ (incompleteness)



# Analysis workflow (5)

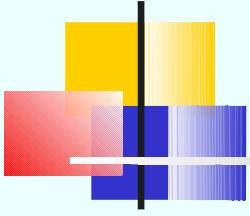
---

Kết quả cần đạt được:

- Bản kế hoạch (tạm thời) về quản lý dự án phần mềm

Yêu cầu về bản kế hoạch:

- Ước lượng chi phí
- Ước lượng thời gian
- Các điểm mốc quan trọng (milestone)
- Các sản phẩm phải có sau mỗi điểm mốc

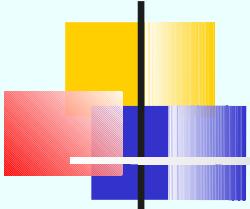


# Design workflow (1)

---

## Mục đích:

- Mịn hóa và mô hình hóa kết quả pha phân tích cho đến khi có thể code được từng modul trên một ngôn ngữ lập trình tương ứng

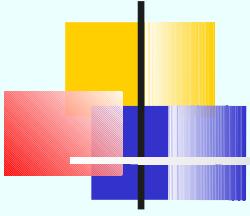


# Design workflow (2)

---

Các vấn đề xem xét:

- Chọn ngôn ngữ lập trình
- Tính sử dụng lại (reusability)
- Tính thiết kế mở (open-design)
- Tính khả chuyền (portability)

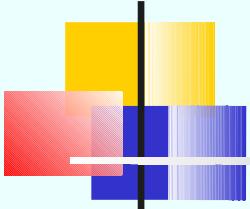


# Design workflow (3)

---

Phương pháp:

- Trích các lớp
- Xác định quan hệ giữa các lớp (thiết kế kiến trúc)
- Thiết kế các thuộc tính và phương thức (method) cho mỗi lớp (thiết kế chi tiết)

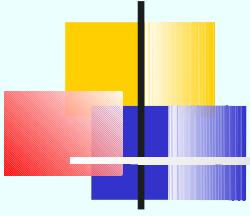


# Design workflow (4)

---

Kết quả cần đạt được:

- Bản mẫu các lớp, thuộc tính và phương thức + thuật toán xử lý trong các phương thức để có thể cài đặt được ngay

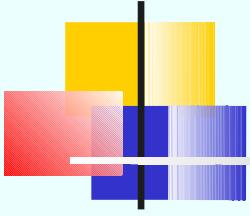


# Implementation workflow (1)

---

Mục tiêu:

- Cài đặt hệ thống theo kết quả pha thiết kế

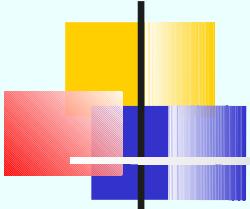


# Implementation workflow (2)

---

Phương pháp:

- Cài đặt theo class, modul
- Tích hợp các class, modul

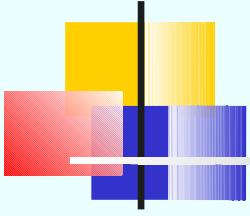


# Test workflow (1)

---

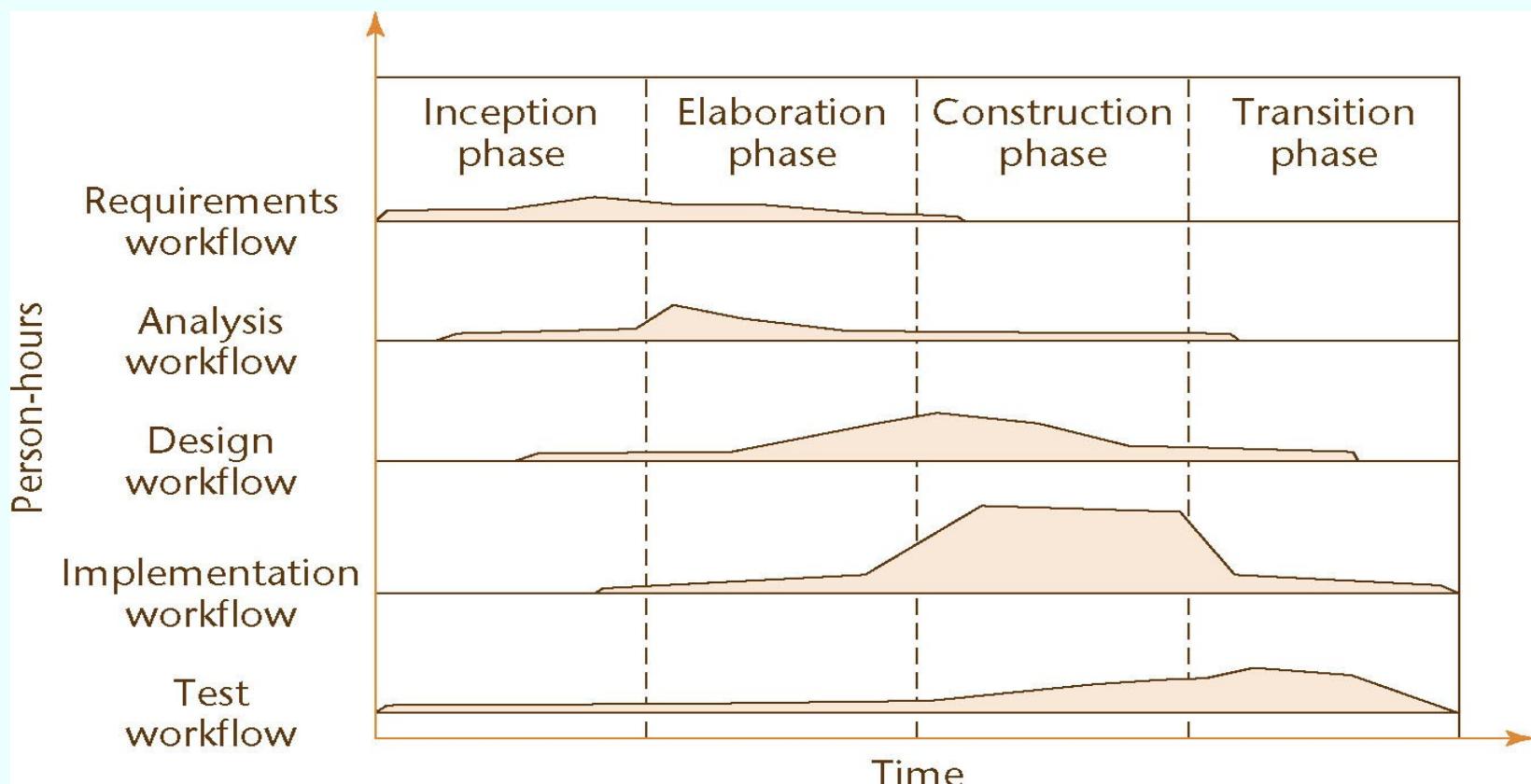
Nội dung test các sản phẩm đầu ra của từng pha:

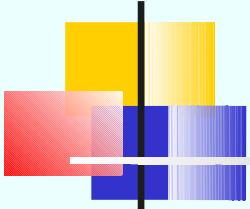
- Yêu cầu: test tài liệu
- Phân tích: test tài liệu, kế hoạch và ước lượng
- Thiết kế: test tài liệu
- Cài đặt: unit test, integrated test, product test, acceptance test. Đối với phần mềm COTS thì test phản alpha và beta.



# Unified Process (1)

Mỗi pha tương ứng một bước trong chu kì tăng trưởng (increasement):



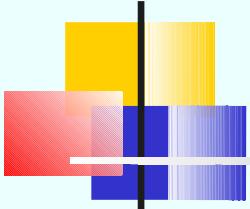


# Unified Process (2)

---

Các pha phát triển:

- Inception: Đánh giá
- Elaboration: Thiết lập
- Construction: Xây dựng
- Transition: Chuyển tiếp



# Unified Process (3)

---

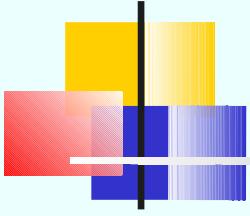
Mỗi bước thực hiện tương ứng với:

- 1 trong 5 workflows
- 1 trong 4 pha

Quan hệ:

- Workflow tương ứng với cách nhìn kĩ thuật
- Pha tương ứng cách nhìn nghiệp vụ

→ Tại sao mỗi bước phải có hai cách nhìn khác nhau?

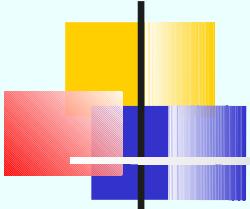


# Inception phase (1)

---

Mục tiêu:

- Xác định xem phần mềm làm ra có kinh tế và khả thi hay không

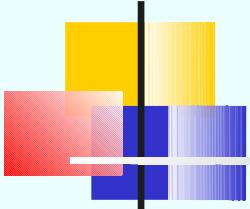


# Inception phase (2)

---

Thực hiện:

- Tìm hiểu lĩnh vực chuyên môn
- Xây dựng mô hình nghiệp vụ
- Nêu rõ giới hạn của sản phẩm
- Bắt đầu xây dựng phân tích kinh doanh

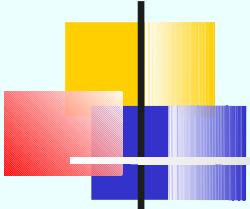


# Inception phase (3)

---

Phân tích kinh doanh:

- Giá phát triển có mang tính kinh tế?
- Bao lâu sẽ quay vòng vốn?
- Nếu từ bỏ dự án thì chi phí hết bao nhiêu?
- Nếu sản phẩm dạng COTS, có cần có chiến dịch tiếp thị sản phẩm?
- Sản phẩm có thể giao đúng hẹn không?
- Thiệt hại gì nếu giao sản phẩm cho khách hàng trễ hẹn?

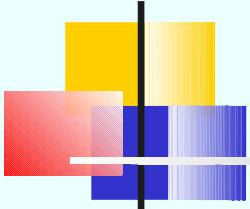


# Inception phase (4)

---

Phân tích rủi ro khi phát triển phần mềm:

- Liệu team có đủ kinh nghiệm cần thiết?
- Có cần phần cứng mới cho sản phẩm?
- Nếu có, thì thiệt hại gì nếu người ta giao phần cứng trễ hẹn?
- Trong trường hợp đó, có nên đặt hàng một nhà cung cấp phần cứng khác để dự phòng không?
- Có cần công cụ hỗ trợ nào không?
- Nếu có, liệu chúng có sẵn hay không, hay có cần toàn bộ chức năng của nó hay không?

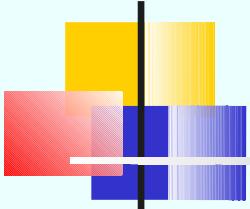


# Elaboration phase (1)

---

Mục tiêu:

- Mịn hóa các kết quả sau pha inception và requirement
- Phân tích rủi ro theo mức độ nghiêm trọng
- Mịn hóa bản phân tích kinh doanh có trong pha inception
- Xem xét lại SPMP

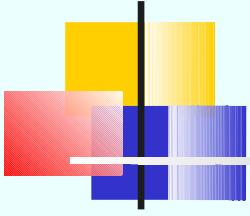


# Elaboration phase (2)

---

Phương pháp:

- Sử dụng các kĩ thuật và phương pháp trong pha inception và requirement

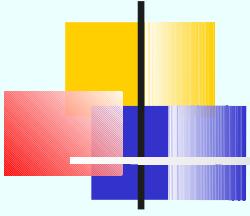


# Construction phase (1)

---

Mục tiêu:

- Xây dựng phiên bản đầu tiên hoạt động được của sản phẩm

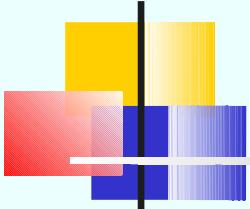


# Construction phase (2)

---

Phương pháp:

- Cài đặt
- Kiểm thử
- Làm tài liệu

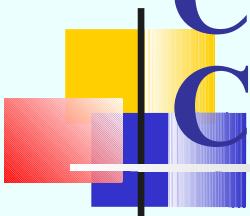


# Transition phase (1)

---

Mục tiêu:

- Đảm bảo tất cả các yêu cầu của khách hàng đã được thực hiện một cách đúng đắn
- Các lỗi đã được sửa
- Các tài liệu hướng dẫn sử dụng đã hoàn chỉnh



# Capability maturity model - CMM

---

- Không phải một mô hình vòng đời phát triển phần mềm
- Một bộ các tiêu chuẩn đánh giá chiến lược hoàn thiện tiến trình phần mềm: SW-CMM
- Các tài liệu hướng dẫn sử dụng đã hoàn chỉnh



- Ra đời năm 1986 bởi SEI
- Hoàn thiện tiến trình phần mềm
- Hoàn thiện quản lý tiến trình, hoàn thiện kỹ thuật
- Có 5 levels



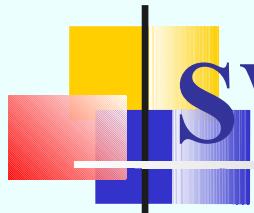
# SW – CMM: level 1

---

Mức khởi đầu (initial):

- Các tiến trình phần mềm là không dự đoán được
- Việc quản lí chỉ bao gồm việc xử lí các rủi ro gấp phả

→ Tất cả các cty đều đạt chuẩn level 1

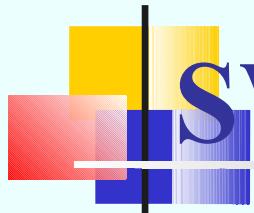


## SW – CMM: level 2

---

Mức có khả năng lặp lại (repeatable):

- Các quyết định quản lý dựa vào các dự án tương tự trước đó
- Có phương pháp đo các tiêu chí
- Kết quả dự án này có thể được dùng để ước lượng chi phí và thời gian cho các dự án tiếp theo
- Khi có lỗi xảy ra, việc khắc phục lỗi được thực hiện ngay

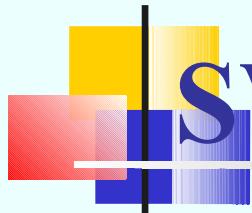


## SW – CMM: level 3

---

Mức có được định nghĩa (defined):

- Có tài liệu kĩ thuật và quản lí
- Liên tục có cố gắng để nâng cao chất lượng sản phẩm
- Việc xem xét lại luôn được thực hiện để đảm bảo chất lượng sản phẩm

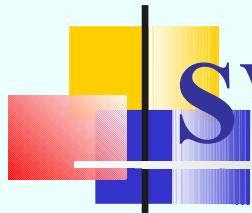


# SW – CMM: level 4

---

Mức có được quản lí (managed):

- Chất lượng và quy trình sản xuất luôn được giám sát
- Việc điều chỉnh chất lượng sản phẩm theo kết quả thống kê cũng được thực hiện

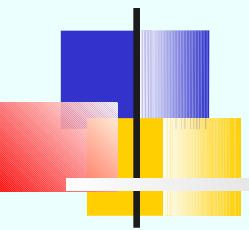


# SW – CMM: level 5

---

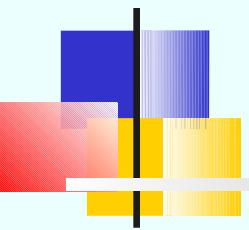
Mức có tối ưu hóa (optimize):

- Không ngừng cải thiện: chất lượng sản phẩm theo thống kê và điều khiển quy trình phát triển
- Ghi nhận phản hồi và kinh nghiệm có được sau mỗi sản phẩm để cải tiến các sản phẩm tiếp theo



# Questions?

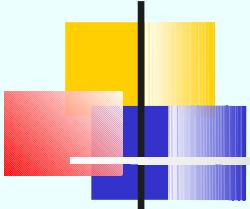
---



# Công nghệ phần mềm

## Một số mô hình vòng đời phát triển phần mềm

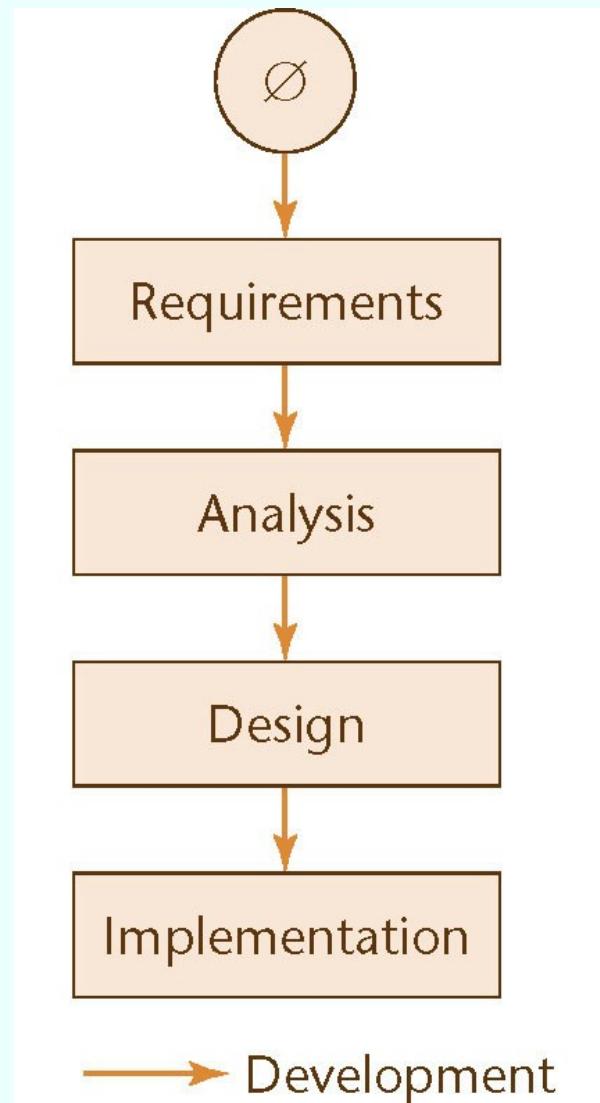
*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

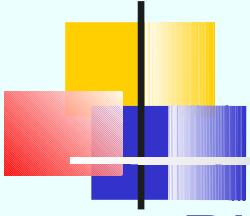


# Mô hình trên lí thuyết

Trên lí thuyết thì:

- Các pha được tiến hành tuần tự
- Bắt đầu phát triển hoàn toàn từ không có gì



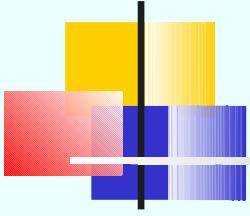


# Thực tế...

---

Phát triển phần mềm hoàn toàn khác:

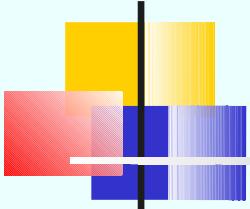
- Lỗi có thể xảy ra mọi lúc mọi nơi trong tiến trình phát triển
- Khách hàng thay đổi hoặc không nắm rõ yêu cầu



# Vấn đề thay đổi yêu cầu (1)

---

- Khách hàng có thể thay đổi yêu cầu ngay khi phần mềm đang được phát triển
- Ngay cả khi thay đổi có lí do hợp lí, thì mọi thay đổi đều ảnh hưởng đến phần mềm
- Các thay đổi có thể dẫn đến lỗi hồi quy (regression fault)
- Nếu thay đổi quá nhiều → phải thiết kế và cài đặt lại phần mềm

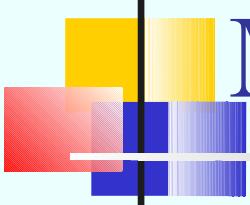


# Vấn đề thay đổi yêu cầu (2)

---

Yêu cầu thay đổi là việc không tránh khỏi:

- Khách hàng là công ty đang phát triển thì yêu cầu thay đổi thường xuyên
- Mỗi cá nhân/khách hàng đều có quyền thay đổi yêu cầu của mình
- → hiện chưa có giải pháp triệt để để giải quyết vấn đề này!

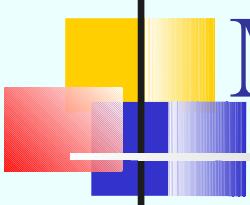


# Mô hình lặp và tăng trưởng (1)

---

Thực tế:

- Các pha phát triển không kết thúc khi chuyển sang pha khác, nó kéo dài liên tục trong suốt vòng đời phát triển → gọi là các workflow
- Bản chất của tiến trình phát triển phần mềm là lặp: lặp lại các bước nhiều lần, kết quả lần sau sẽ tốt hơn lần trước



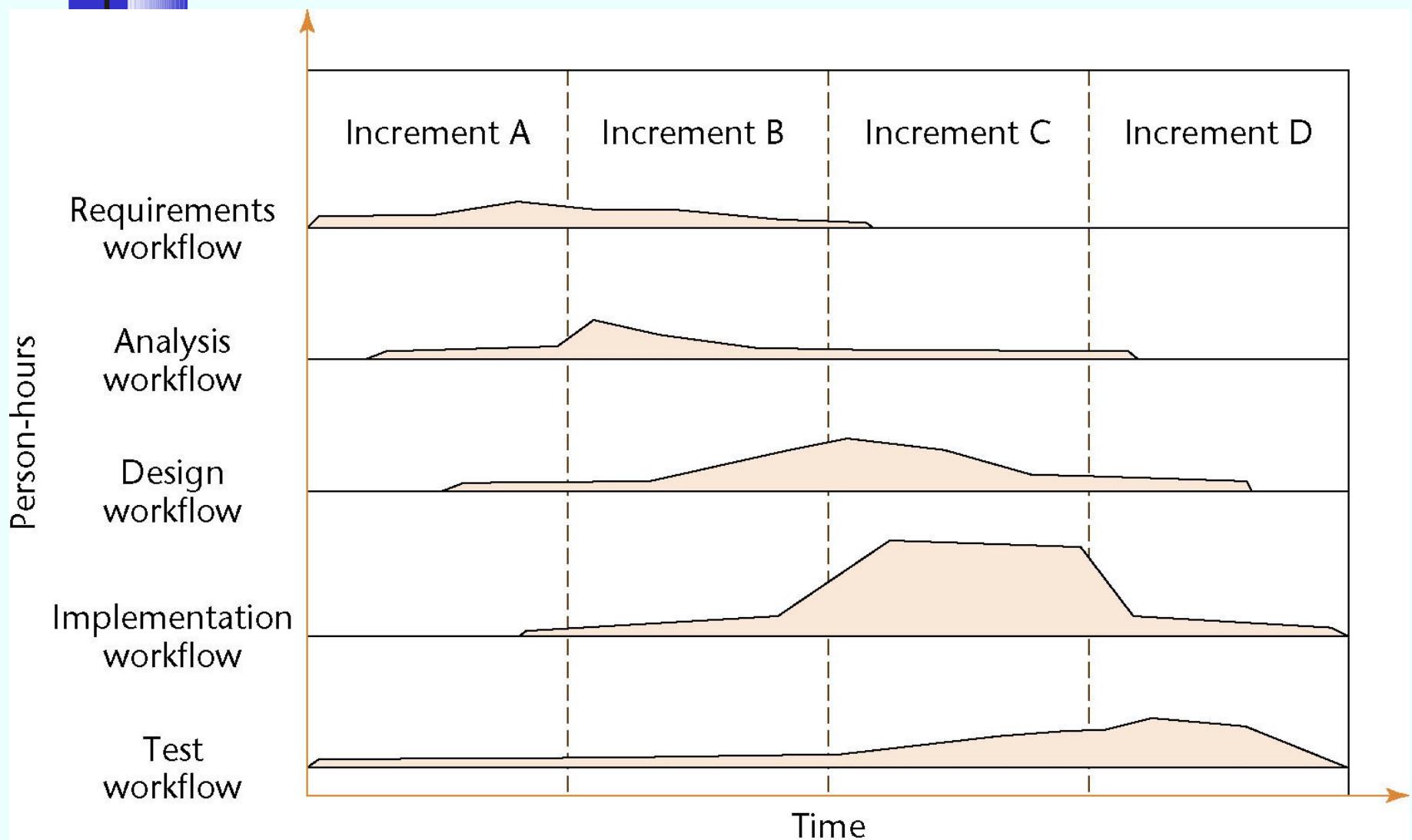
# Mô hình lặp và tăng trưởng (2)

---

## Luật Miller:

- Tại mỗi thời điểm, người ta chỉ có thể tập trung vào tối đa khoảng 7 vấn đề
- để xử lý các vấn đề lớn, sử dụng phương pháp làm mịn từng bước:
- Tập trung xử lý các việc quan trọng trước
  - Các việc ít quan trọng hơn xử lý sau
- gọi là tiến trình tăng trưởng

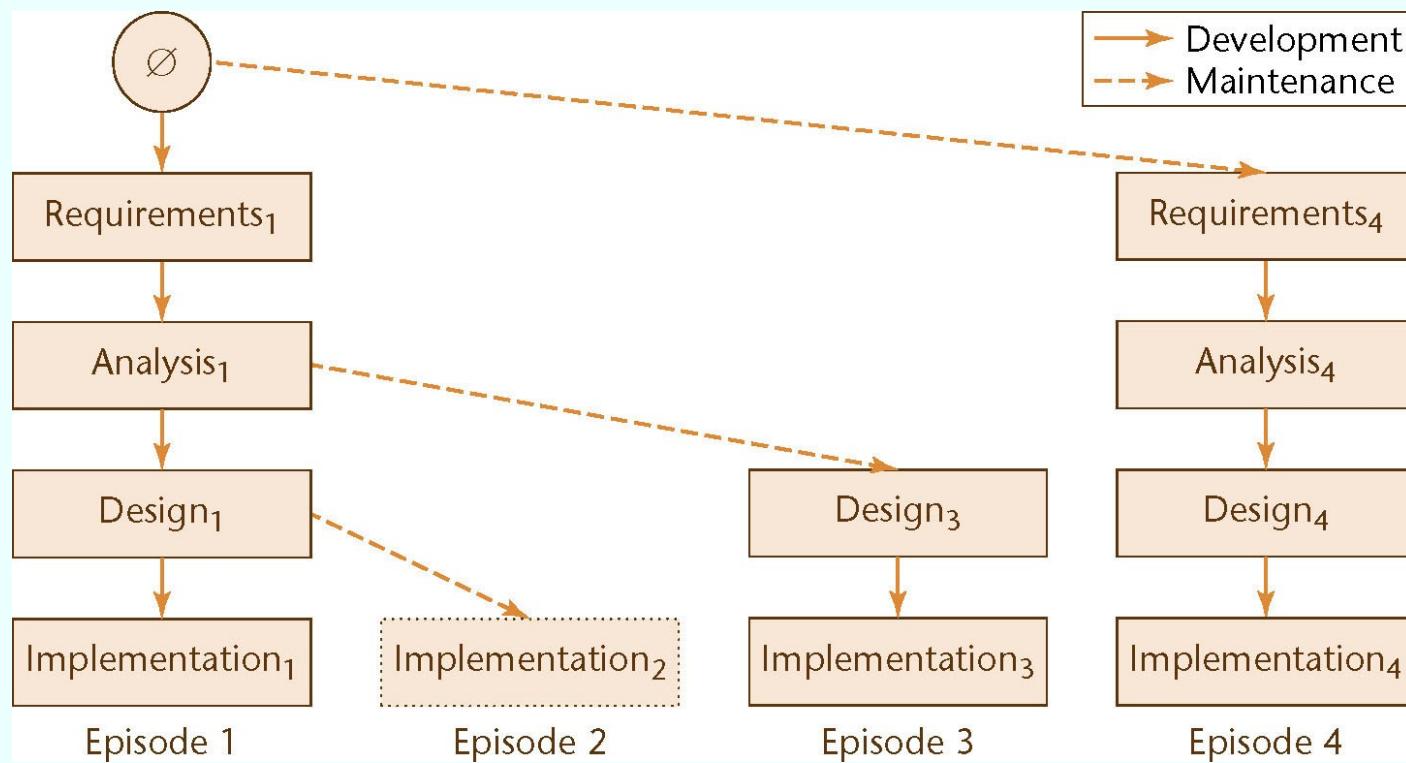
# Mô hình lắp và tăng trưởng (3)

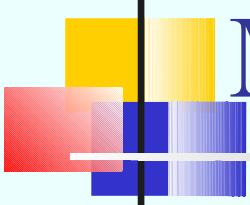


# Mô hình lắp và tăng trưởng (4)

Lắp và tăng trưởng kết hợp nhau:

- Không có các pha đơn lẻ, mà mỗi pha được lắp lại nhiều lần



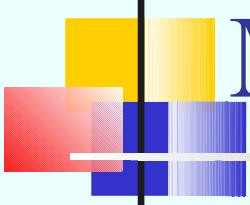


# Mô hình lắp và tăng trưởng (5)

---

Dùng khái niệm workflow thay vì phase:

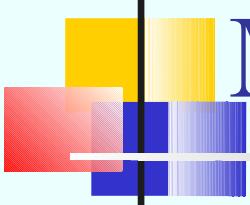
- Thực tế không tồn tại tuần tự các pha
- Tất cả 5 workflow đều hoạt động trong suốt vòng đời PTPM
- Tại mỗi giai đoạn, có một workflow chiếm vị trí trọng tâm



# Mô hình lắp và tăng trưởng (6)

---

- Có thể coi dự án là một tập các dự án nhỏ, mỗi dự án nhỏ tương ứng với một lần tăng trưởng
- Mỗi dự án nhỏ đều có artifact cho mỗi workflow:
  - Mở rộng các artifacts (tăng trưởng)
  - Kiểm thử các artifacts (test)
  - Thay đổi artifacts (lắp)
- Mỗi dự án nhỏ thực hiện một phần của dự án ban đầu

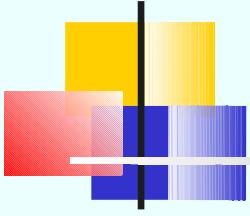


# Mô hình lắp và tăng trưởng (7)

---

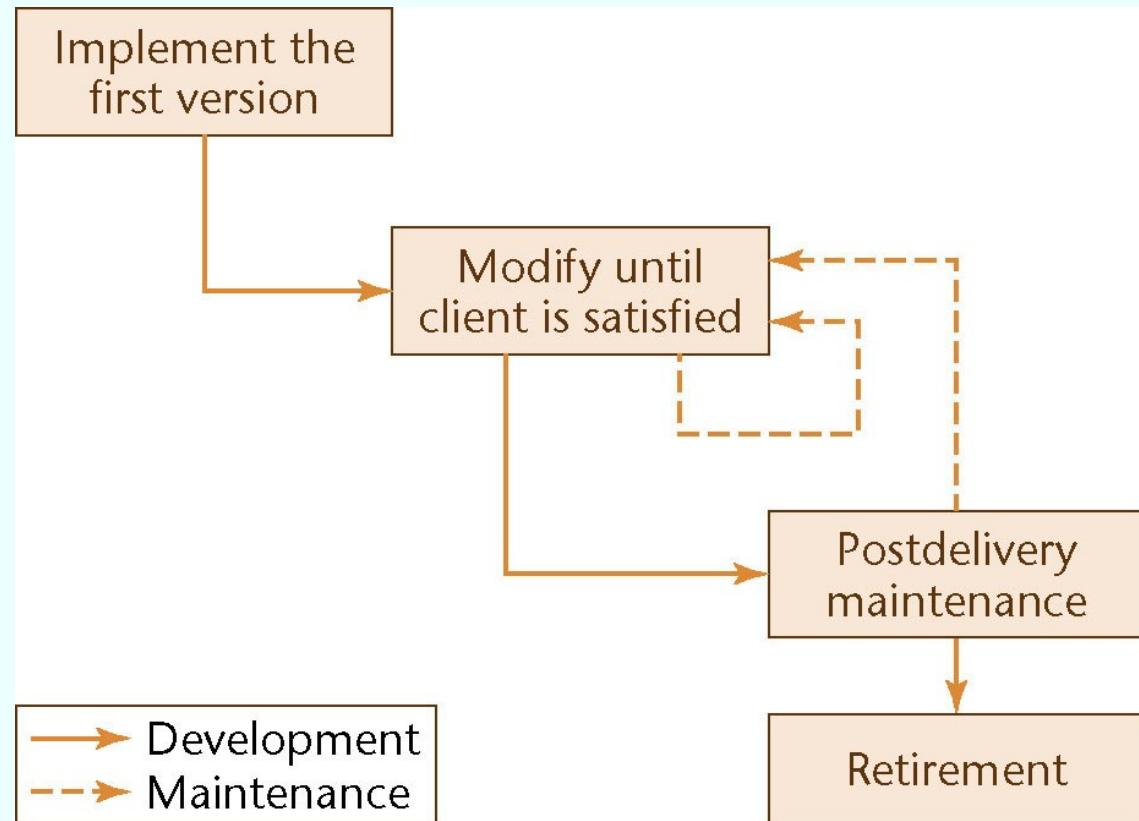
Ưu điểm:

- Mỗi bước lắp đều có test workflow → có thể phát hiện và sửa lỗi sớm
- Thiết kế kiến trúc ngay từ đầu: mô hình theo modul ngay từ đầu, và việc tính toán tránh lỗi khi kết hợp được tính toán trước
- Có thể có phiên bản dùng được của sản phẩm ngay từ giai đoạn đầu
- Khách hàng có thể dựa vào phần đã hoàn thành để nêu chính xác yêu cầu trong những modul sau

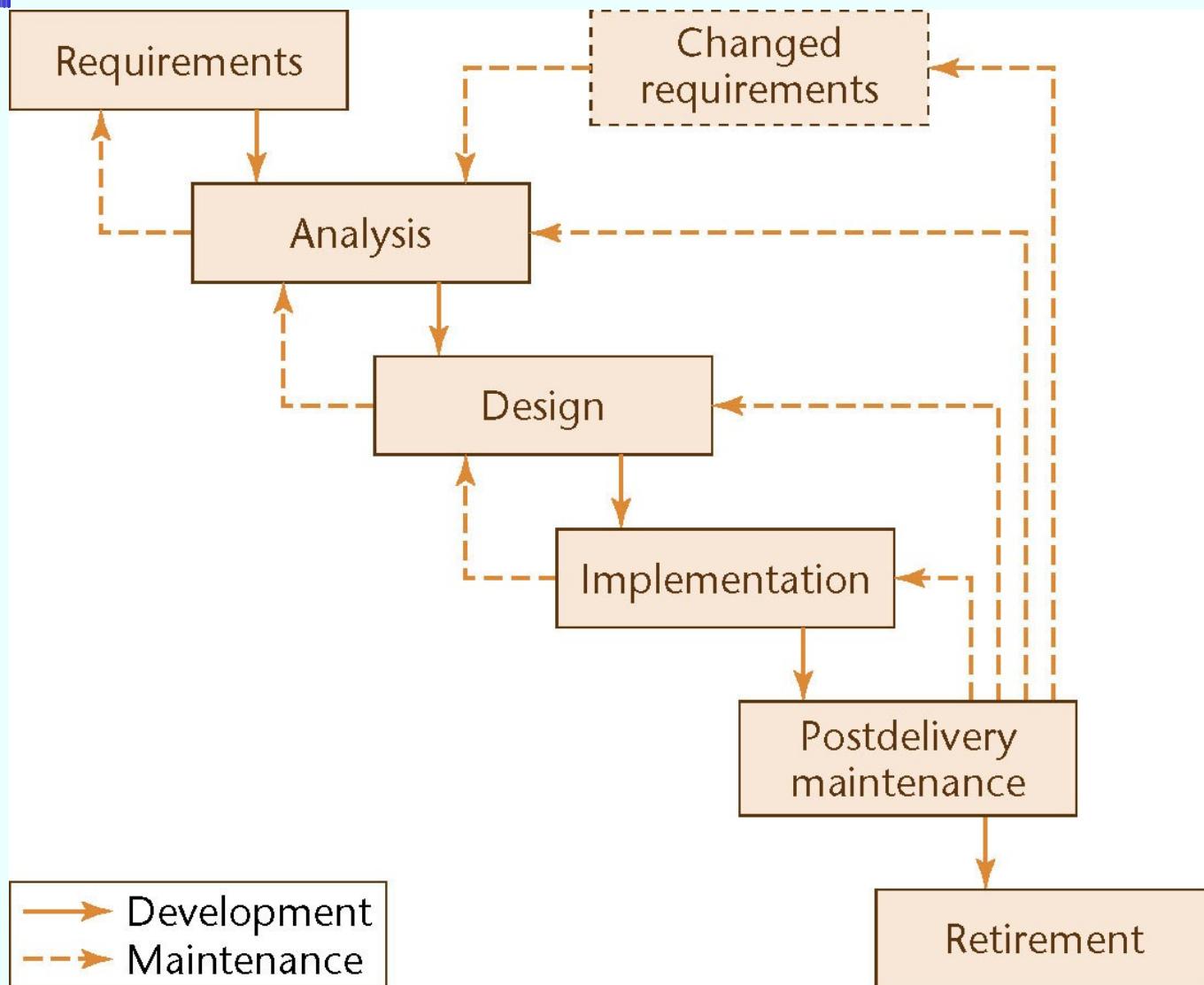


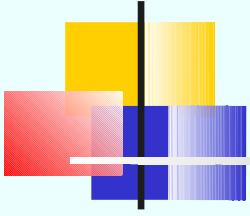
# Mô hình xây và sửa

- Không có đặc tả, không có thiết kế
- Chỉ cài đặt
- Bảo trì mò mẫm
- Ưu điểm?
- Nhược điểm?



# Mô hình thác nước (1)





# Mô hình thác nước (2)

---

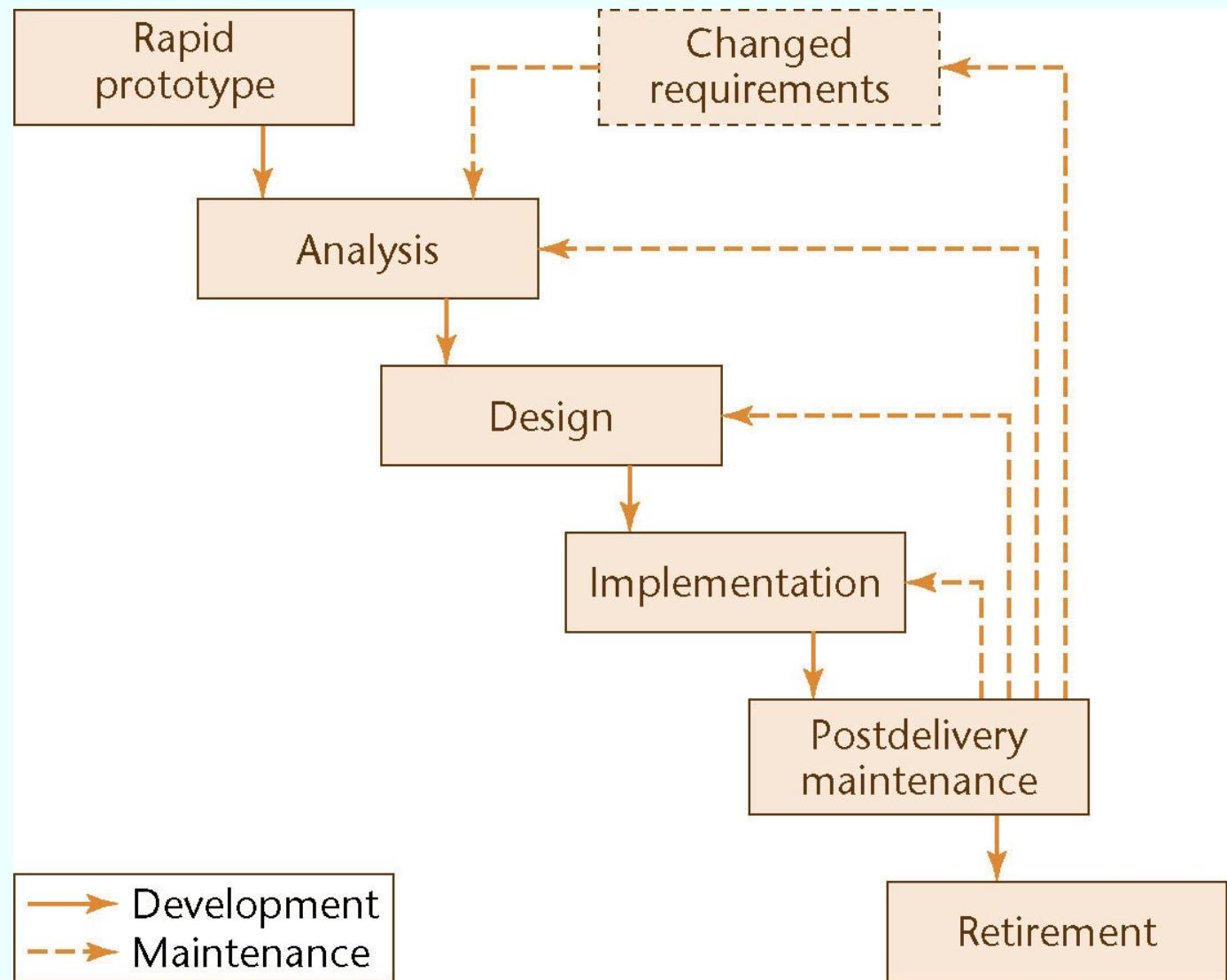
Đặc trưng:

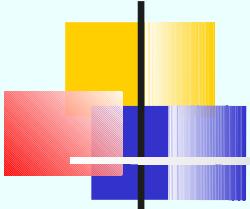
- Các vòng lặp phản hồi sau mỗi pha
- Làm tài liệu cuối mỗi pha

Ưu và nhược điểm?

- ...

# Mô hình bản mẫu nhanh (1)





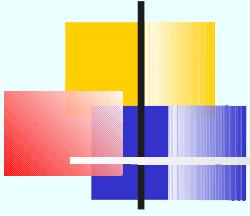
# Mô hình bản mẫu nhanh (2)

---

Đặc trưng:

- Tiến hành làm bản mẫu nhanh (rapid prototype) trong pha lấy yêu cầu
- Các pha còn lại làm theo thứ tự tuyến tính

Ưu và nhược điểm?

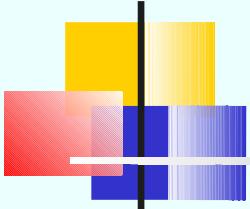


# Tiến trình linh hoạt (1)

---

Trích chọn các story của sản phẩm:

- Ước lượng thời gian và chi phí
- Chọn story tiếp theo để phát triển
- Mỗi story được chia nhỏ thành các task
- Viết các test case cho các task trước khi cài đặt

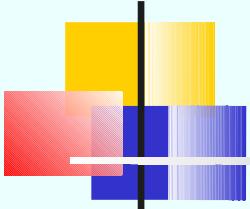


# Tiến trình linh hoạt (2)

---

Phát triển mỗi story:

- Không có pha đặc tả
- Thiết kế linh hoạt và có thể thay đổi theo yêu cầu của khách hàng
- Luôn có đại diện của khách hàng trong team
- Lập trình theo cặp
- Liên tục tích hợp các task

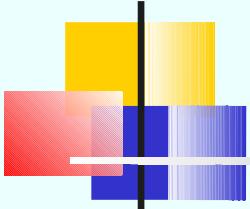


# Tiến trình linh hoạt (3)

---

Sử dụng phương pháp họp đứng (stand-up meeting):

- Toàn bộ team đứng vòng tròn và nhìn thấy được nhau
- Thời gian họp cố định hàng ngày, nhưng không quá 15p
- Họp để thấy vấn đề, nếu có, chư không giải quyết vấn đề
- Lần lượt mỗi người trả lời các câu hỏi giống nhau

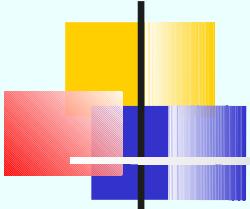


# Tiến trình linh hoạt (4)

---

Các câu hỏi khi họp đúng:

- Tôi đã làm được gì từ buổi họp hôm qua?
- Hôm nay tôi đang làm cái gì?
- Có vấn đề gì với việc đang làm hôm nay?
- Chúng ta có quên làm phần nào không?
- Tôi đã học thêm được gì khi làm việc với team?

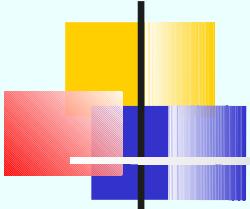


# Tiến trình linh hoạt (5)

---

## Chiến lược:

- Mỗi story chỉ phát triển liên tục và phải hoàn thiện sau 2-3 tuần
  - Cứ sau 3 tuần hoàn thành một bước lặp và bàn giao tính năng mới cho khách hàng
  - Sử dụng kĩ thuật timeboxing để quản lí thời gian
- mô hình này yêu cầu cố định thời gian, không cố định tính năng của sản phẩm



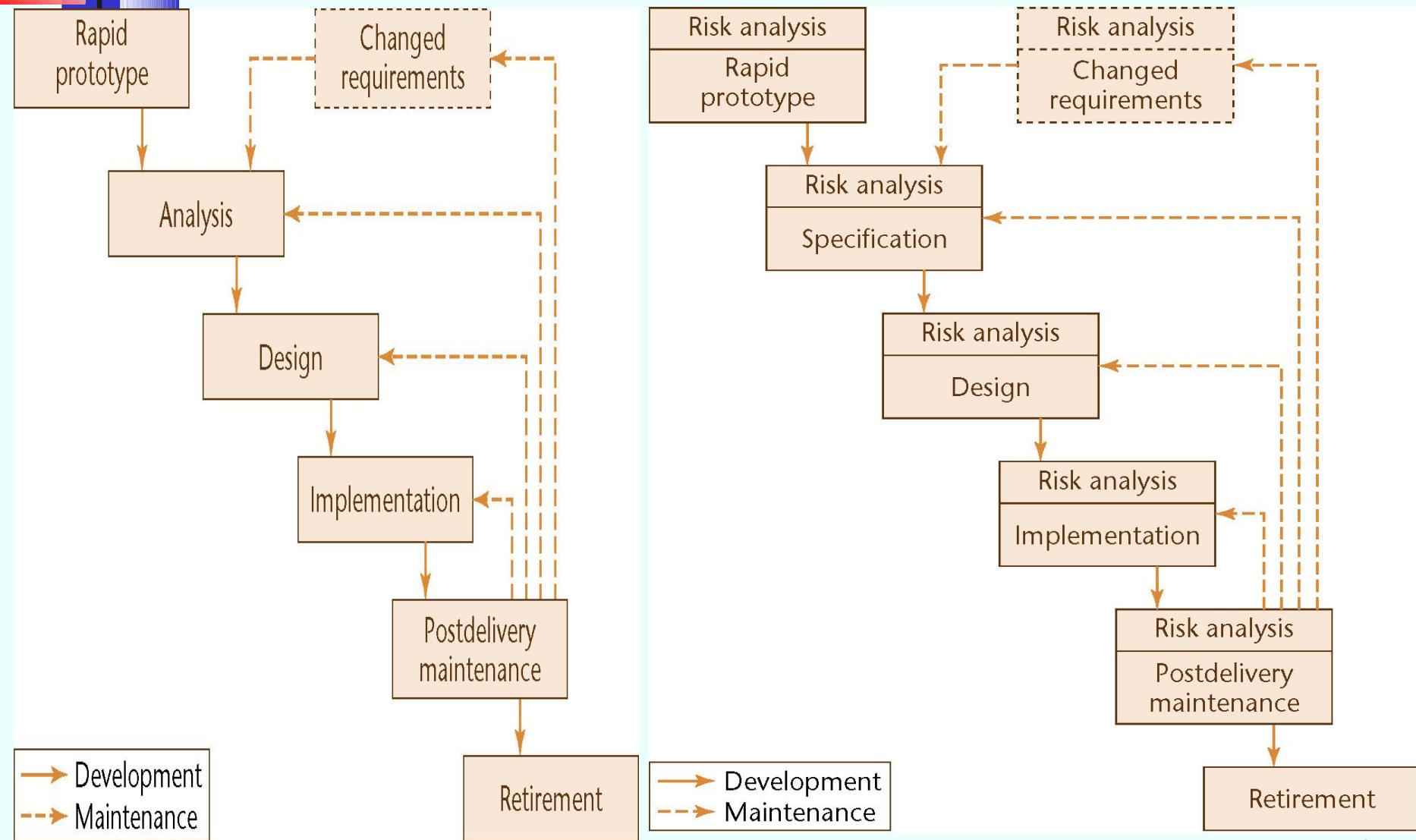
# Tiến trình linh hoạt (6)

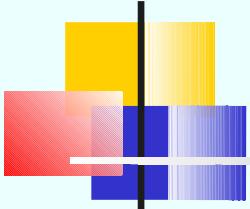
---

Ưu điểm và nhược điểm:

- Phương pháp họp đứng (stand-up meeting)?
- Kỹ thuật timeboxing?

# Mô hình xoắn ốc (1)





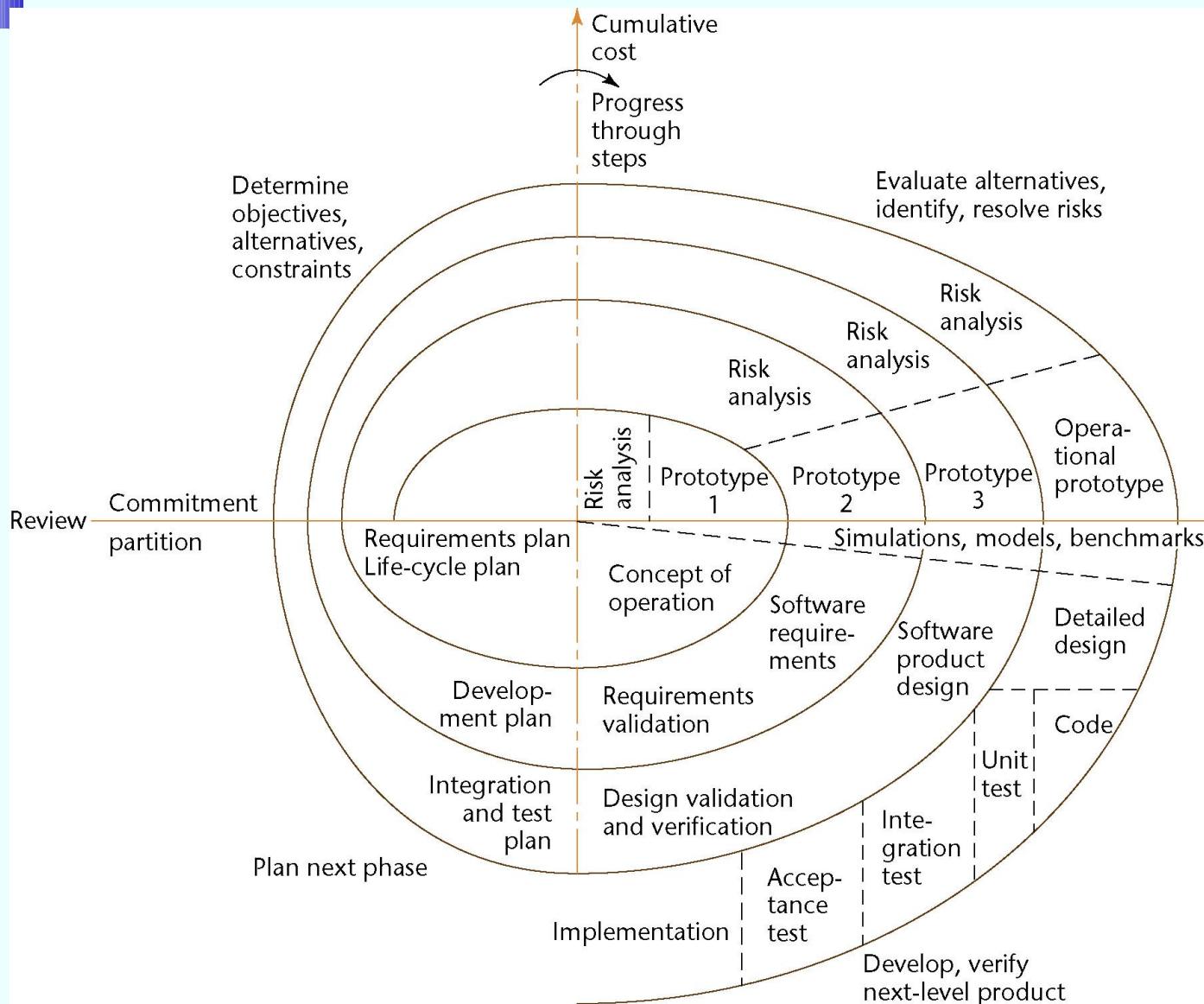
# Mô hình xoắn ốc (2)

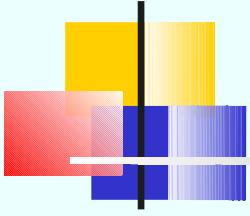
---

Đặc trưng:

- Có nhiều vòng lặp nhau, nhưng vòng lặp sau phát triển rộng hơn vòng trước
- Mỗi pha của mỗi lần lặp:
  - Bắt đầu bằng việc quyết định và phân tích rủi ro
  - Kết thúc bằng việc đánh giá lỗi và lập kế hoạch cho pha tiếp theo
  - Nếu các rủi ro đều không xử lý được thì dừng lại ngay lập tức

# Mô hình xoắn ốc (3)



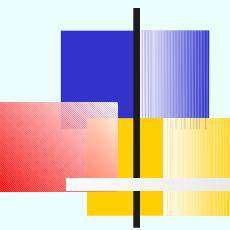


# Mô hình xoắn ốc (4)

---

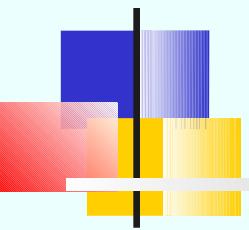
## Ưu điểm và nhược điểm?

- ...



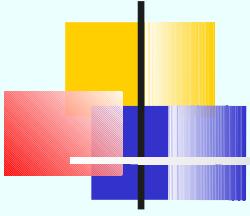
# Questions?

---



# Công nghệ phần mềm Nhóm (team) phát triển phần mềm

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

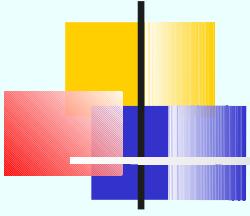


# Tổ chức nhóm PTPM

---

Trên lí thuyết thì:

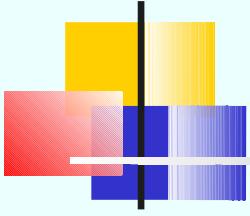
- Nếu một sản phẩm phần mềm phải giao trong 3 tháng, nhưng đòi hỏi khối lượng công việc là 12 tháng/người
- → Dùng 4 người phát triển phần mềm đó thì có đúng hạn và chất lượng không?



# Chia sẻ công việc (1)

---

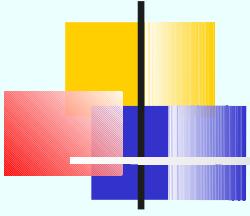
- Một nông dân cày đám ruộng hết 10 ngày  
→ nếu có 10 nông dân cày đồng thời thì  
chỉ hết có 1 ngày
- Một phụ nữ trong 9 tháng thì sinh được 1 baby  
→ 9 phụ nữ có thể sinh 1 baby trong 1 tháng?



# Chia sẻ công việc (2)

---

- Không giống việc sinh baby, phát triển phần mềm là một dạng công việc có thể chia sẻ được
- Cũng không giống cày ruộng, PTPM cần đến các kỹ năng hợp tác trong nhóm hiệu quả



# Tổ chức nhóm code (1)

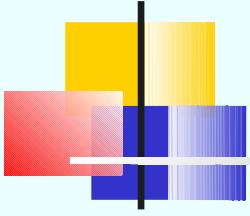
---

Xét ví dụ:

- A và B phải code hai modul M1 và M2

Các lỗi sau có thể xảy ra:

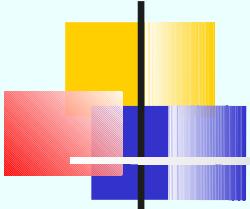
- A và B cùng code M1, nghĩ rằng người còn lại code M2
- A code M1, B code M2. M1 gọi M2 truyền 4 tham số, nhưng M2 nhận 5 tham số
- Hai bên đều có 4 tham số, nhưng thứ tự và kiểu tham số bên gọi khác bên định nghĩa



# Tổ chức nhóm code (2)

---

- Không phải vấn đề về năng lực kỹ thuật  
→ mà là vấn đề quản lý con người và công việc!

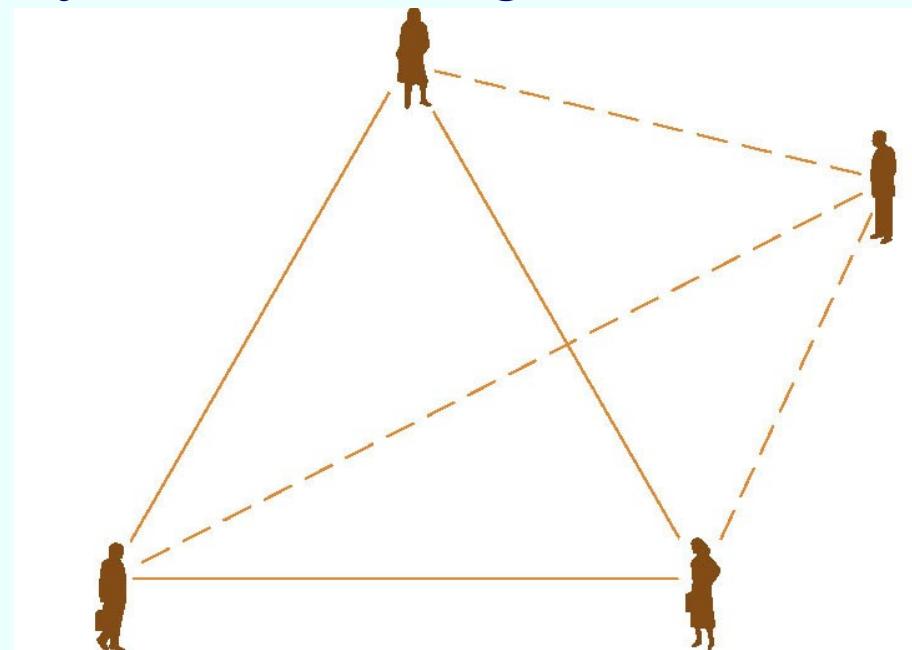


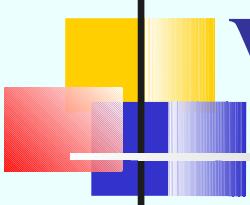
# Vấn đề giao tiếp (1)

---

Xét ví dụ:

- Đôi phát triển có 3 người → có 3 kênh giao tiếp.  
Nhưng dự án sắp đến hạn mà còn quá nhiều việc
- Giải pháp trực quan: tuyển thêm 1 người  
→ cần 6 kênh giao tiếp!





# Vấn đề giao tiếp (2)

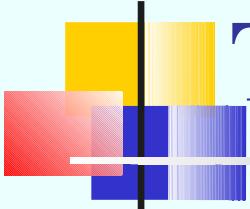
---

3 người cũ sẽ phải diễn giải cho người mới:

- Các việc đã hoàn thành
- Các việc chưa hoàn thành
- Cách hoàn thiện các việc còn dang dở

Luật Brooks:

- Khi đưa thêm người mới vào dự án đang nguy cơ bị trễ, thì không giải quyết được vấn đề trễ, thậm chí còn làm dự án bị trễ thêm!



# Tổ chức nhóm PTPM

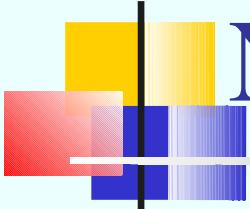
---

Thông thường:

- Nhóm PTPM làm việc với nhau trong suốt tiến trình PTPM, nhưng quan trọng nhất là giai đoạn code
  - người ta quan tâm đến việc tổ chức nhóm code

Có hai loại nhóm code:

- Nhóm code bình đẳng (dân chủ)
- Nhóm code có sếp

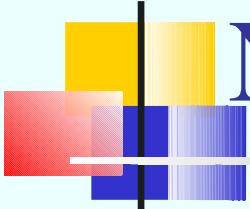


# Nhóm code bình đẳng (1)

---

Dựa trên các nguyên tắc:

- Các thành viên bình đẳng về chức vụ
- Mỗi người tự do thiết kế, code và test modul của mình
- Việc có lỗi được coi là việc bình thường
- Cả đội sẽ xây dựng một tính năng hay cả sản phẩm, và sản phẩm là của cả đội



# Nhóm code bình đẳng (2)

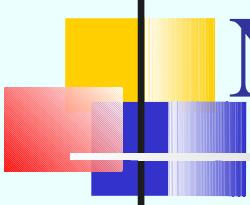
---

Thuận lợi:

- Các thành viên nắm chắc phần code của mình
- Khả năng code mạnh, nhất là giải quyết các dự án khó

Khó khăn:

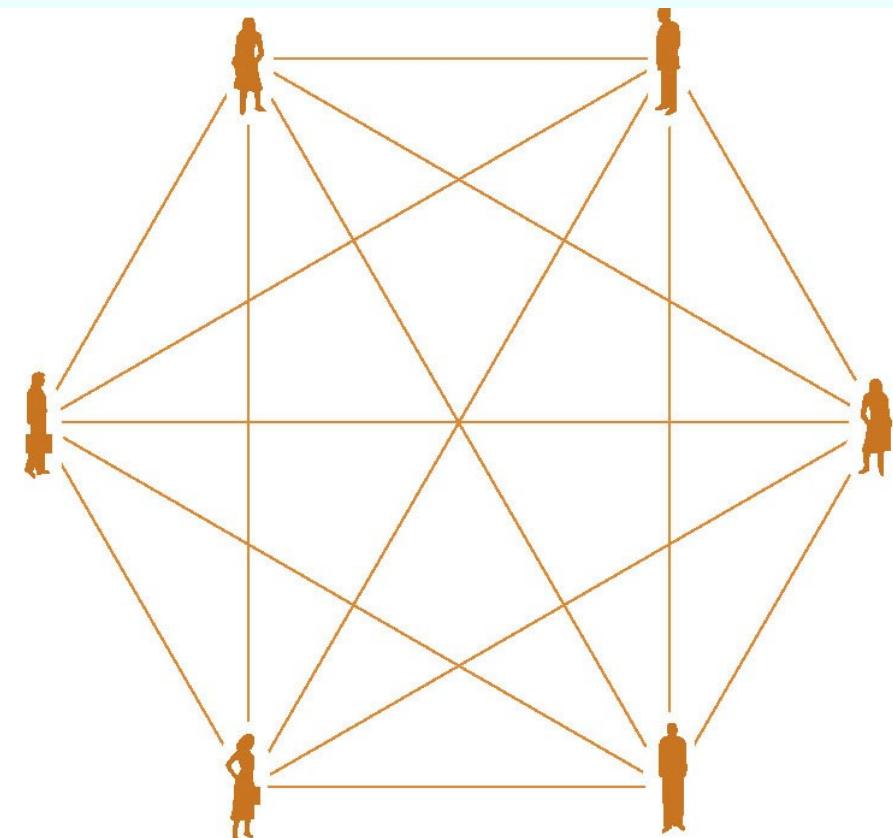
- Việc tự test code của mình thường không hiệu quả
- Khó khăn về mặt quản lý  
→ Đội phải được phát triển một cách tự nhiên

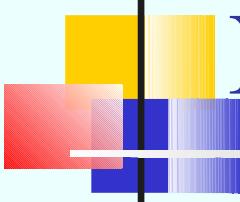


# Nhóm code có sếp – kiểu cũ (1)

Xem xét nhóm có 6 người:

- Có 15 cặp giao tiếp
- Tổng các nhóm con có 2, 3, 4, 5, 6 người là 57
- Nhóm này không thể giải quyết 1 việc cần 6 tháng/ người trong 1 tháng!

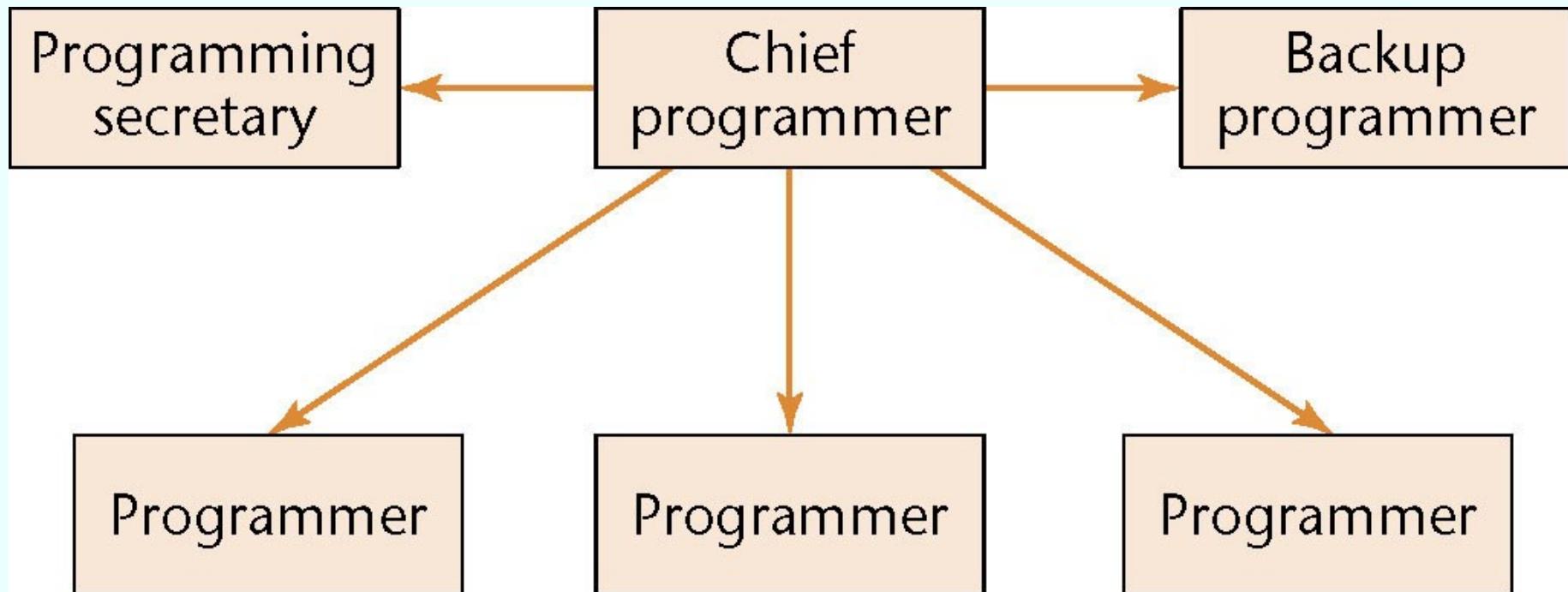


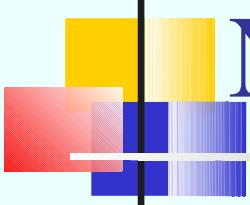


# Nhóm code có sếp – kiểu cũ (2)

Xem xét giải pháp:

- Có 6 thành viên, nhưng chỉ còn 5 cặp giao tiếp!
- Dựa trên ý tưởng nhóm bác sĩ của 1 ca phẫu thuật



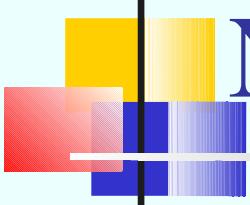


# Nhóm code có sếp – kiểu cũ (3)

---

Sếp của nhóm code:

- Có kĩ năng cao trong quản lí và code
- Thực hiện phần thiết kế kiến trúc
- Phân công công việc code cho các thành viên
- Code các phần chính và khó nhất
- Tạo các giao diện để tích hợp các modul
- Xem lại code của tất cả các thành viên
- Chịu trách nhiệm về từng dòng code của nhóm

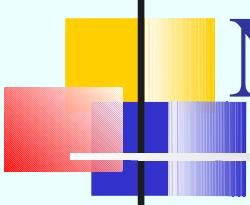


# Nhóm code có sếp – kiểu cũ (4)

---

Sếp dự bị của nhóm code:

- Dự bị cho sếp của nhóm
- Có kĩ năng tương đương sếp trong quản lí và code
- Nắm rõ dự án tương đương sếp
- Lập kế hoạch test hộp đen (black-box) và các công việc độc lập với tiến trình thiết kế

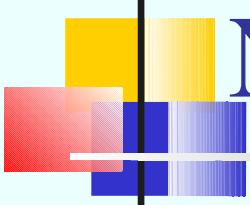


# Nhóm code có sếp – kiểu cũ (5)

---

Thư ký lập trình của nhóm code:

- Có kỹ năng cao, trả lương cao, và là thành viên chủ chốt của nhóm
- Chịu trách nhiệm về tài liệu cho toàn bộ dự án:
  - Liệt kê danh sách mã nguồn
  - Ngôn ngữ điều khiển công việc (JCL)
  - Dữ liệu test
  - Biên dịch code, kiểm tra code convention
  - Chạy các test case

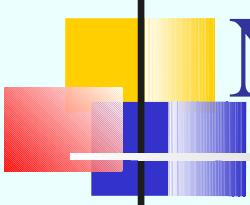


# Nhóm code có sếp – kiểu cũ (6)

---

Lập trình viên:

- Chỉ làm việc duy nhất là lập trình
- Các việc khác liên quan đã có thư ký lập trình lo!

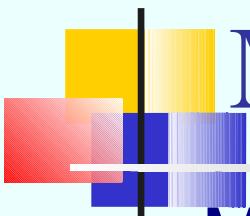


# Nhóm code có sếp – kiểu cũ (7)

---

Khó khăn:

- Sếp, dù bị đều phải có đồng thời kĩ năng cao trong cả quản lí và code. Nhưng thường người quản lí giỏi thì code kém và ngược lại
- Sếp dù bị phải có kĩ năng tương đương sếp, nhưng phải làm dù bị cho sếp và trả lương thấp hơn → khó ai chấp nhận!
- Thư ký không làm gì ngoài việc làm tài liệu cả ngày → lập trình viên thường gét việc làm tài liệu!



# Mô hình nhóm kết hợp (1)

---

Mục đích:

- Kết hợp ưu điểm của cả hai mô hình:
  - Nhóm bình đẳng: tinh thần phát hiện và sửa lỗi cao
  - Nhóm có sếp: quản lí và giao tiếp tốt

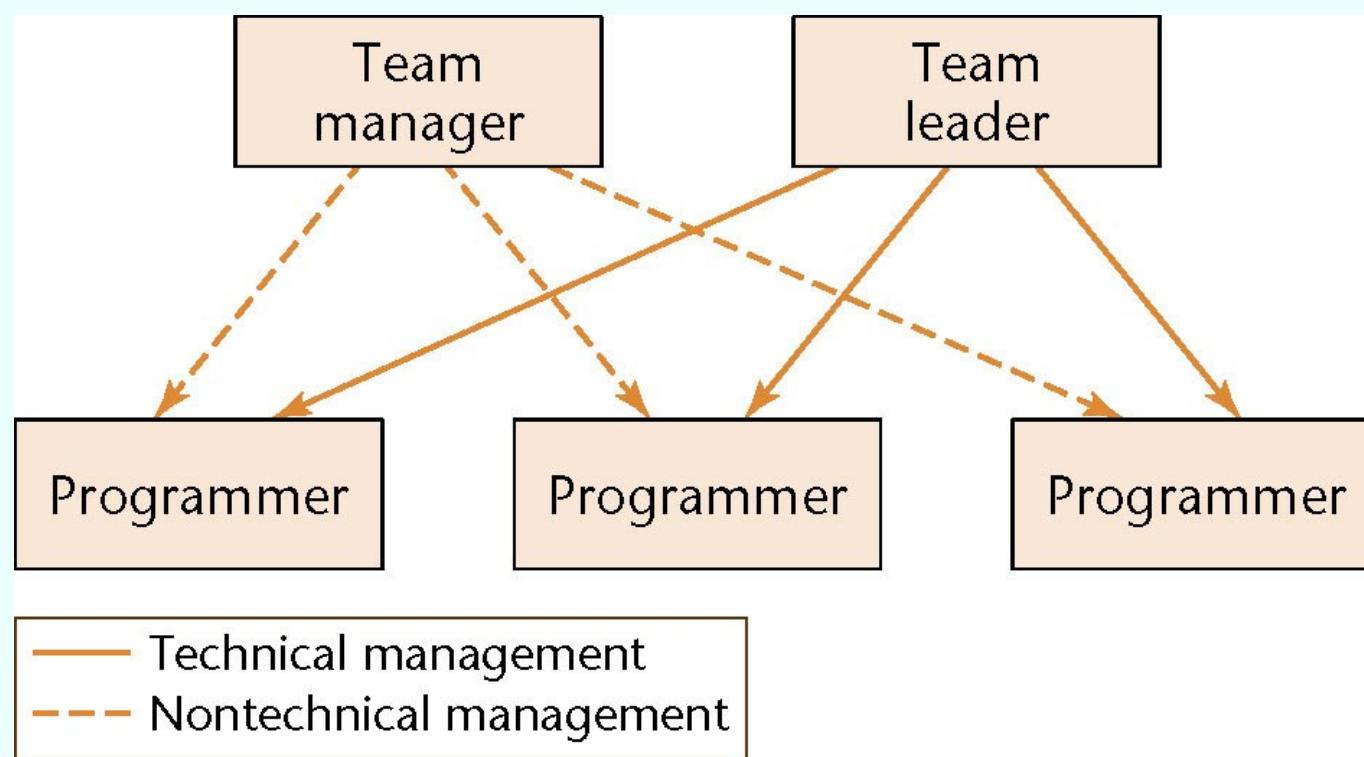
Thực tế, trong mô hình CPT:

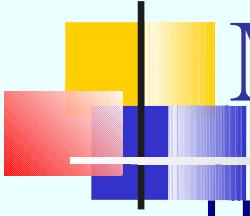
- Sếp chịu trách nhiệm về từng dòng code nên phải review toàn bộ code
  - Sếp cũng chịu trách nhiệm quản lí nên có thể không cần review code
- Giảm bớt trách nhiệm của sếp!

# Mô hình nhóm kết hợp (2)

Giải pháp:

- Sắp chỉ quản lí các vấn đề phi kĩ thuật
- Tạo ra team leader để quản lí kĩ thuật



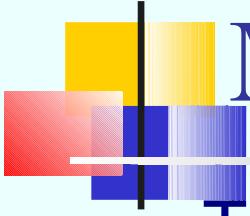


# Mô hình nhóm kết hợp (3)

---

## Hoạt động:

- Sếp chỉ quản lí các vấn đề phi kĩ thuật : thu nhập, bình đẳng, năng lực của các thành viên
- Team leader chỉ quản lí kĩ thuật: review toàn bộ code và hỗ trợ kĩ thuật cho các thành viên
- Sếp không review code nhưng khi họp có thể tham gia để hỗ trợ kĩ thuật cho các thành viên



# Mô hình nhóm kết hợp (4)

---

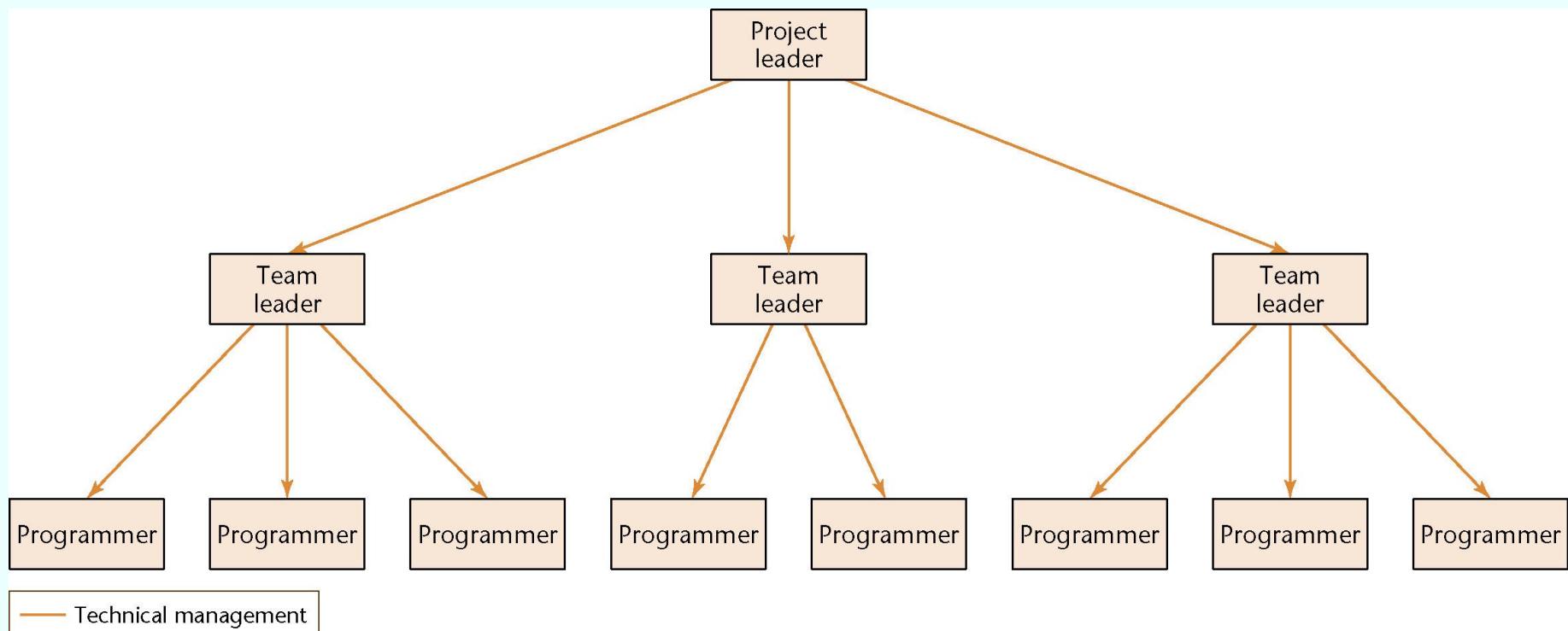
Tính khả thi:

- Tìm một team leader dễ dàng hơn nhiều một sếp
- Các thành viên chỉ bị quản lí bởi 1 sếp duy nhất
- Sếp chỉ cần kỹ năng cao về quản lí, team leader chỉ cần kỹ năng cao về code → dễ tìm hơn

# Mô hình nhóm kết hợp (5)

Với dự án lớn:

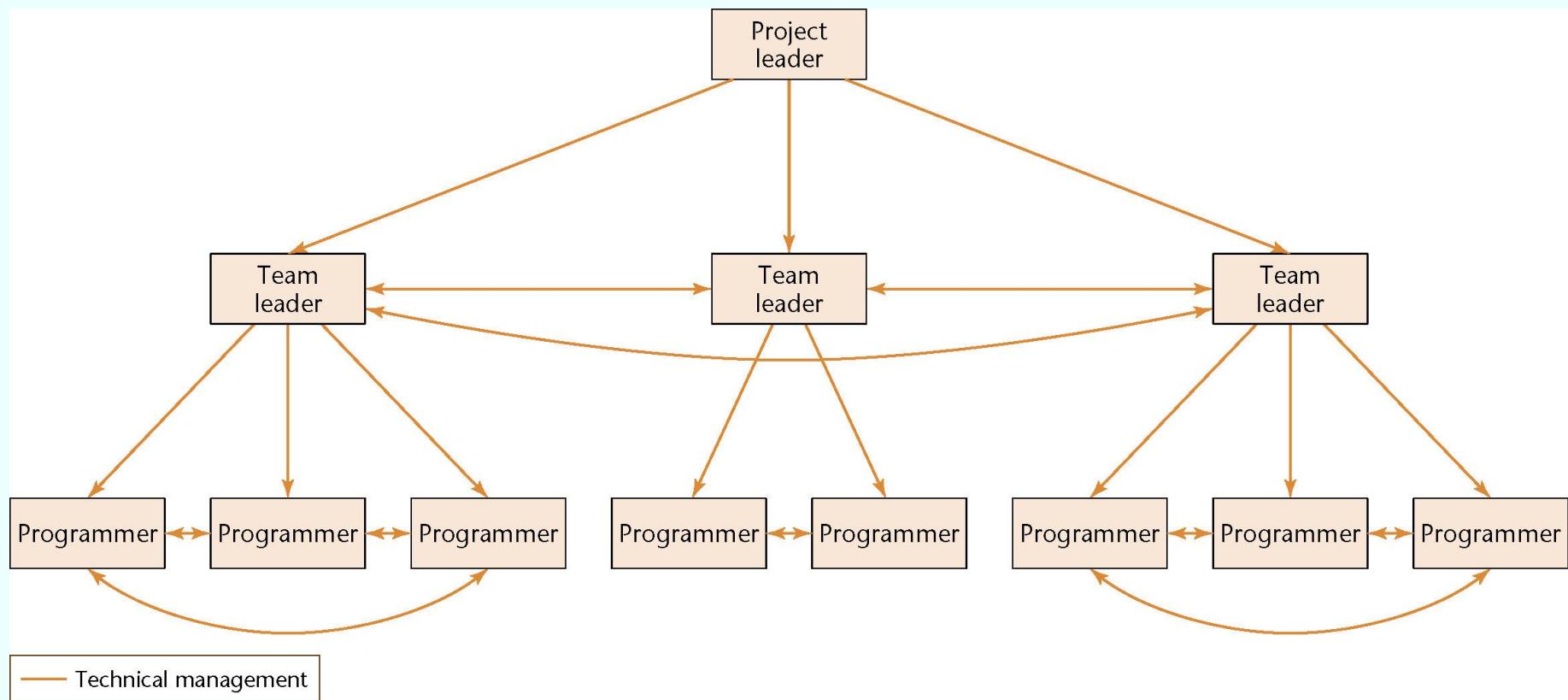
- Thêm một tầng quản lý kỹ thuật

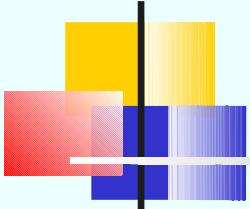


# Mô hình nhóm kết hợp (6)

Vấn đề ra quyết định:

- Dùng phương pháp nhóm bình đẳng





# Nhóm code cho mô hình XP

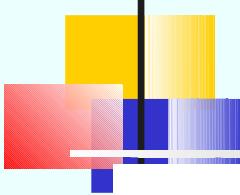
---

Mô hình:

- Lập trình theo cặp, mỗi cặp chung máy
- Test chéo: người này test code của người kia

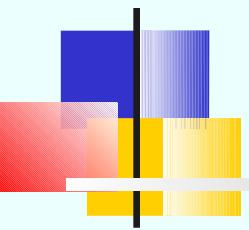
Ưu điểm:

- Khi một thành viên rời nhóm, thì thành viên mới dễ dàng gia nhập vì có thể học với người cùng cặp
- Phát huy được ưu điểm của lập trình bình đẳng



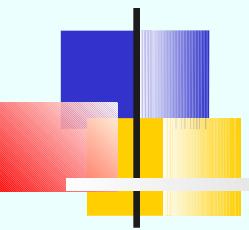
# People - CMM





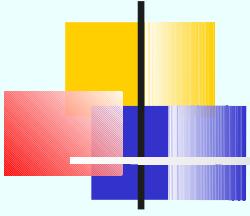
# Questions?

---



# Công nghệ phần mềm Pha lấy yêu cầu

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

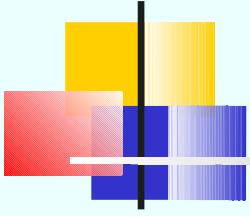


# Pha lấy yêu cầu (1)

---

Mục đích:

- Xác định rõ cái khách hàng cần
- Không phải xác định cái khách hàng muốn

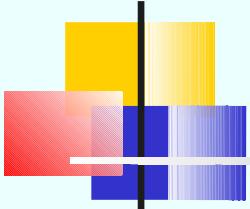


# Pha lấy yêu cầu (2)

---

## Thực hiện:

- Tìm hiểu và nắm rõ lĩnh vực của phần mềm
- Xây dựng mô hình nghiệp vụ của khách hàng
- Xác định rõ yêu cầu của khách hàng dựa trên mô hình nghiệp vụ
- Lặp lại các bước trên cho đến khi khách hàng đồng ý

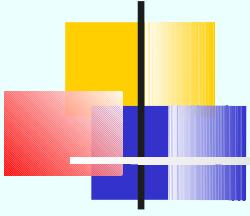


# Pha lấy yêu cầu (3)

---

Tìm hiểu domain của ứng dụng:

- Xây dựng danh sách từ chuyên môn (glossary)
- Mỗi từ / khái niệm / cụm từ được giải thích nghĩa rõ ràng theo đúng chuyên ngành hẹp của ứng dụng

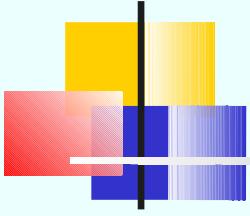


# Pha lấy yêu cầu (4)

---

Xây dựng mô hình nghiệp vụ:

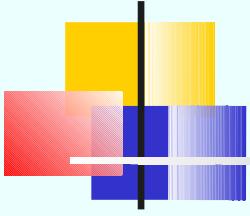
- B1: Phỏng vấn với đại diện khách hàng để có bản mô tả nghiệp vụ toàn bộ các hoạt động của khách hàng
- B2: Sử dụng UML để biểu diễn yêu cầu của khách hàng: Use case
- Chỉ các yêu cầu chức năng mới được mô hình hóa trong UML, các yêu cầu phi chức năng sẽ được áp dụng từ bước thiết kế



# Use case (1)

---

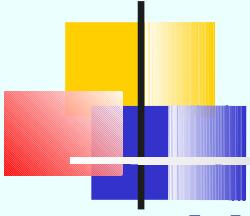
- Use case mô hình một tương tác giữa người dùng với hệ thống phần mềm
- Ví dụ: trong UML



# Use case (2)

---

- Trong VP, chọn use case diagram

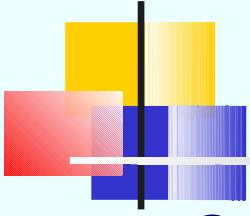


# Actor (1)

---

Một use case thường có:

- Actor: tác nhân – người dùng tương ứng với use case đó
- Actor thường là người khởi tạo use case hoặc là tác nhân chính để use case hoạt động
- Một người dùng có thể làm nhiều actor khác nhau
- Một actor có thể tham gia vào nhiều use case khác nhau
- Actor có thể là một tổ chức khác hoặc một thiết bị đầu cuối như máy in, điện thoại, tổng đài thông tin

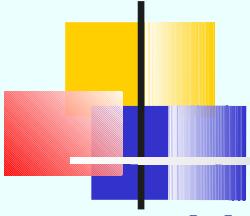


# Actor (2)

---

Các actor có thể có quan hệ kế thừa:

- Nhân viên quản trị mạng (Admin), nhân viên bán hàng (Seller), nhân viên lễ tân (Receptionist) đều có thể coi là nhân viên của khách sạn (Hotel employee)

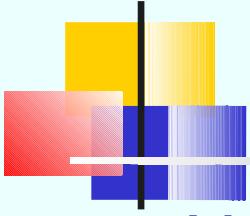


# Actor (3)

---

Use case có 2 actor:

- Chỉ có nhân viên lễ tân (Receptionist) là thao tác với phần mềm, nhưng phải có khách hàng có mặt tại quầy thì việc checkin mới diễn ra. Do đó, UC này cần có 2 actor.



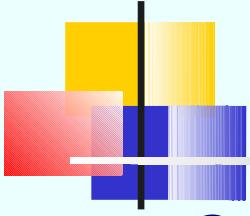
# Actor (4)

---

Nếu nhiều actor có cùng hoạt động liên quan đến cùng 1 use case thì sao?

- Nhân viên lễ tân (Receptionist) phải login, khi login chỉ cần một mình nhân viên lễ tân là login vào được.
- Nhân viên bán hàng (Seller) cũng phải login, khi login cũng chỉ cần một mình nhân viên bán hàng là login vào được.

→ Vậy phải biểu diễn trong UML như thế nào?

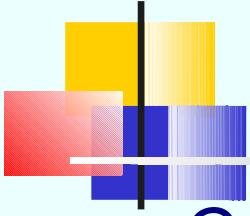


# Actor (5)

---

Cách biểu diễn **sai**:

- Cách biểu diễn này **sai** vì nó nói lên rằng việc login chỉ diễn ra được khi có **đồng thời cả Seller và Receptionist**. Thực tế không phải vậy, chỉ cần một trong 2 người login cũng được.

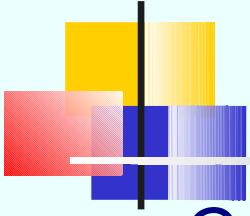


# Actor (6)

---

Cách biểu diễn chấp nhận được (1):

- Tạo một actor trừu tượng và cho nó là actor duy nhất của UC Login

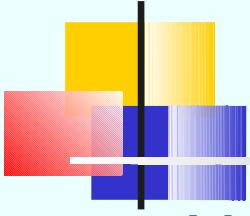


# Actor (7)

---

Cách biểu diễn chấp nhận được (2):

- Tạo một UC login trừu tượng rồi cho 2 dạng login khác nhau tương ứng với mỗi actor



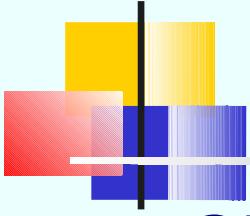
# Actor (8)

---

Nếu nhiều actor có cùng hoạt động liên quan đến cùng 1 use case thì sao?

- Cách 1: dùng actor trừu tượng
- Cách 2: dùng use case trừu tượng

→ Vậy cách nào tốt hơn?

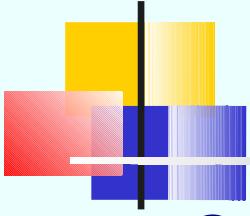


# Quan hệ giữa các use case

---

Giữa các use case có thể có các quan hệ:

- Quan hệ include (bao gồm)
- Quan hệ extend (mở rộng)
- Quan hệ generalize (kế thừa)

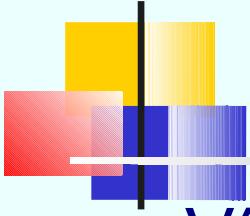


# Quan hệ include (1)

---

Quan hệ “include”:

- Úc A có quan hệ include với úc B nếu việc hoàn thành B là một phần công việc để hoàn thành A
- Nếu không hoàn thành B thì A không thể hoàn thành
- Việc hoàn thành B có thể lặp lại nhiều lần, thì người ta tạo ra úc riêng để tránh trùng lặp
- Quan hệ này được biểu diễn bằng một mũi tên nét đứt đi từ A đến B. Mũi tên có nhãn « include »

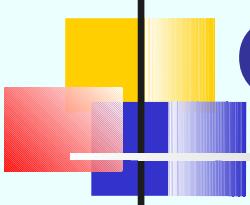


# Quan hệ include (2)

---

Ví dụ phần mềm quản lý đặt chỗ cho khách sạn:

- Khi khách hàng gọi điện đến cho nhân viên bán hàng yêu cầu **đặt phòng**, nhân viên phải **tìm kiếm phòng trống**, khi đó, ứng **Đặt phòng** sẽ include ứng **Tìm phòng trống**

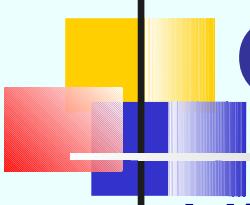


# Quan hệ extend (1)

---

Quan hệ “extend”:

- Uc A có quan hệ extend với uc B nếu việc hoàn thành A là một tùy chọn công việc để hoàn thành B
- Trong một số trường hợp, làm B bao gồm làm A
- Nhưng trong một số trường hợp khác, làm B không cần làm A
- Quan hệ này được biểu diễn bằng một mũi tên nét đứt đi từ A đến B. Mũi tên có nhãn « extend »

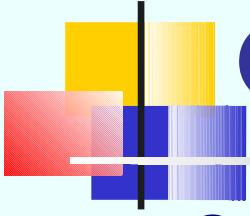


# Quan hệ extend (2)

---

Ví dụ phần mềm quản lý đặt chỗ cho khách sạn:

- Admin **login** vào thì có thể chọn chức năng **quản lý phòng**, hoặc chức năng **tạo báo cáo**, hoặc không cần thực hiện thêm chức năng nào cũng được.

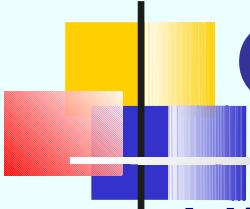


# Quan hệ generalize (1)

---

## Quan hệ kế thừa:

- Úc A có quan hệ kế thừa với úc B nếu B là một phần dạng tổng quát của A, hay A là một thể hiện chi tiết của B
- Quan hệ này được biểu diễn bằng một mũi tên nét liền (đầu hình tam giác rỗng) đi từ A đến B.



# Quan hệ generalize (2)

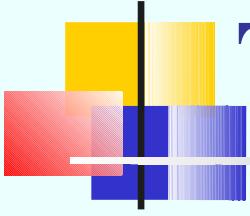
---

Ví dụ phần mềm quản lý đặt chỗ cho khách sạn:

- Nhân viên bán hàng có thể login theo role của mình và nhận đặt phòng qua điện thoại
- Nhân viên lễ tân cũng có thể login theo role của mình và nhận đặt phòng tại chỗ cho khách

## Ví dụ 1

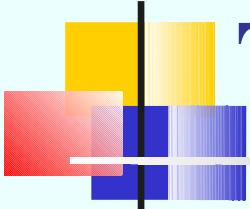
Trang web Hotels.com



# Trang chủ (1)

---

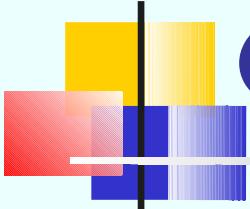
Phần trên:



# Trang chủ (2)

---

Phần dưới:

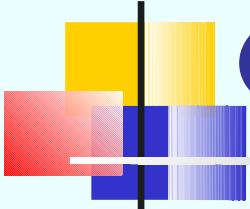


# Các use case ban đầu

---

Như vậy vào hệ thống, có thể thực hiện 2 việc:

- Tìm kiếm khách sạn
- Xem danh sách khách sạn phổ biến

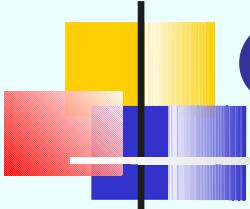


# Chi tiết các use case (1)

---

Click vào nút search khách sạn, hiện ra kết quả:

- Phía trên là menu cho phép xem kết quả sắp xếp theo: most popular, star, rating, distance, price
- Phía dưới là danh sách chi tiết các khách sạn còn phòng

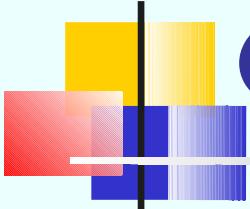


# Chi tiết các use case (2)

---

Như vậy các use case liên quan đến **tìm kiếm**:

- **Xem danh sách chi tiết** các khách sạn là kết quả tất yếu của thao tác tìm kiếm
- Có thể **tùy chọn xem danh sách** theo: most popular, star, rating, distance, price

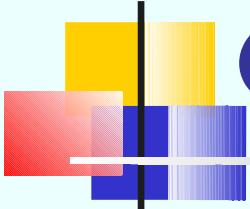


# Chi tiết các use case (3)

---

Click vào tên một khách sạn (tùy chọn từ trang xem danh sách chi tiết):

- Hiện ra thông tin chi tiết của khách sạn
- Bên phải là nút đặt chỗ

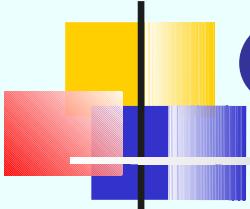


# Chi tiết các use case (4)

---

Như vậy:

- Xem chi tiết khách sạn là một tùy chọn từ hành động xem danh sách khách sạn
- Đặt chỗ cũng là một tùy chọn từ hành động xem chi tiết khách sạn

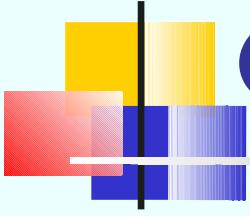


# Chi tiết các use case (5)

---

Click vào nút book của một hạng phòng:

- Trang thanh toán hiện ra yêu cầu nhập thông tin thanh toán và xác nhận thanh toán

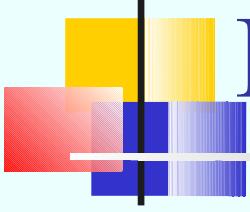


# Chi tiết các use case (6)

---

Như vậy:

- Muốn đặt phòng thì phải thanh toán thì mới hoàn thành được mục đích đặt phòng

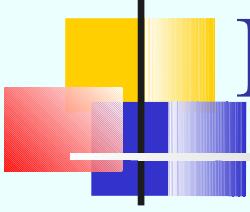


# Bài tập 1

---

Chi tiết use case còn lại:

- Từ trang chủ, click vào xem chi tiết 1 hotel thì hiện lên trang chi tiết hotel + cho phép tìm kiếm phòng trống cho riêng hotel đấy

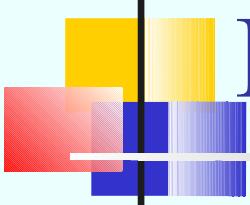


# Bài tập 1 (tt)

---

Chi tiết use case còn lại:

- Từ trang chủ, click vào xem chi tiết 1 hotel thì hiện lên trang chi tiết hotel + cho phép tìm kiếm phòng trống cho riêng hotel đấy

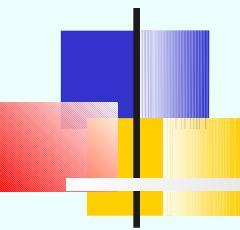


# Bài tập 1 (tt)

---

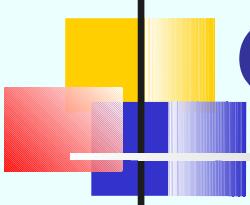
Chi tiết use case còn lại:

- Chọn ngày và click vào check available thì hiện ra chi tiết thông tin khách sạn + các phòng để đặt chỗ (có nút đặt chỗ để chuyển sang trang đặt chỗ)



## Ví dụ 2

### Game: Line 98

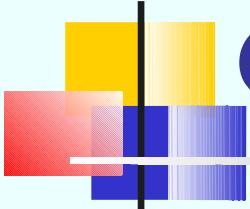


# Giao diện chính (1)

---

Có thể:

- Chơi game
- Trong quá trình chơi có thể chọn các chức năng trên menu: save, load, config (sound, skin, type of game),...

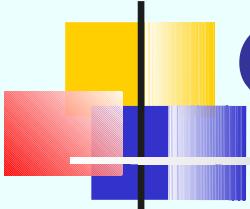


# Giao diện chính (2)

---

Có thể:

- Trong quá trình chơi có thể chọn các chức năng trên menu: save, load, config (sound, skin, type of game),...

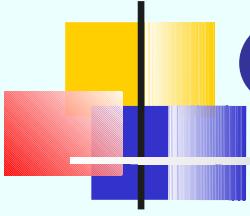


# Giao diện chính (3)

---

Có thể:

- Khi kết thúc game thì bảng xếp hạng hiện ra
- Nếu click vào menu high score thì bảng xếp hạng cũng hiện ra

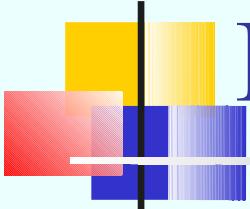


# Các use case

---

Như vậy:

- Chỉ có 1 hành động chính là chơi game
- Các hành động còn lại là tùy chọn trong khi chơi game
- Riêng xem bảng xếp hạng là hành động vừa tùy chọn (click vào menu) vừa bắt buộc (khi hết game)

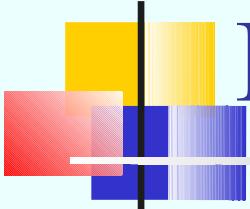


# Bài tập 2

---

Quan sát, phân tích và vẽ sơ đồ use case cho:

- Game solitaire của Window
- Game trồng vườn của Facebook

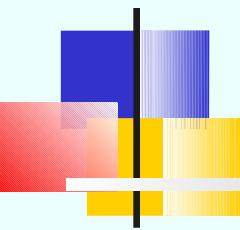


# Bài tập nhóm phải nộp (r1)

---

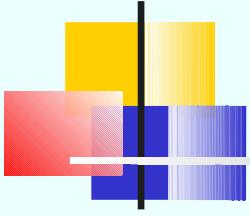
## Mỗi thành viên trong nhóm:

- Tìm hiểu một hệ thống trong thực tế của người ta đã có mà tương tự như đề tài của nhóm đã đăng kí, hệ thống của mỗi người tìm hiểu phải khác nhau
- Mô tả các hoạt động nghiệp vụ của từng chức năng như trong bài
- Sau đó vẽ sơ đồ use case tương ứng cho từng chức năng của hệ thống đó
- Mỗi người nộp báo cáo riêng, ghi đầy đủ họ tên, lớp, nhóm ngay đầu báo cáo
- Điểm đánh giá cá nhân



# Case study

# Phần mềm quản lí đặt phòng cho khách sạn

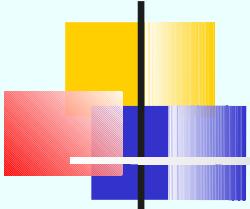


# Mục đích

---

Phần mềm:

- Hỗ trợ quản lý việc đặt phòng, nhận phòng, trả phòng và thanh toán cho một khách sạn

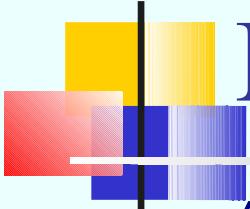


# Mô tả (1)

---

Phạm vi phần mềm:

- Hỗ trợ quản lý cho 1 khách sạn
- Chỉ có nhân viên khách sạn có thẩm quyền mới được thao tác, sử dụng phần mềm: người quản lý khách sạn, nhân viên quản trị hệ thống, nhân viên bán hàng, nhân viên lễ tân

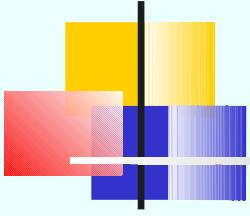


## Mô tả (2)

---

Đối với tất cả các nhân viên:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login có thể thay đổi mật khẩu cá nhân
- Khi xong công việc, hoặc hết ca làm việc phải logout khỏi hệ thống

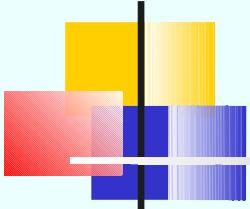


## Mô tả (3)

---

Người quản lý khách sạn được phép:

- Xem các báo cáo, bao gồm báo cáo doanh thu theo thời gian, báo cáo doanh thu theo phòng, báo cáo tỉ lệ phòng trống theo thời gian

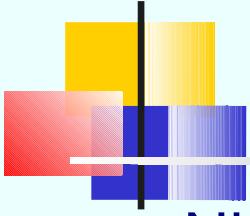


# Mô tả (4)

---

Nhân viên quản trị hệ thống được phép:

- Quản lý các tài khoản của người sử dụng hệ thống (thêm, sửa, xóa tài khoản)

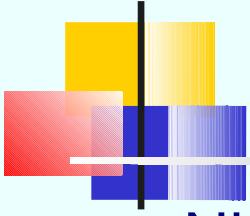


# Mô tả (5)

---

Nhân viên bán hàng được phép:

- Nhận đặt phòng cho khách hàng qua điện thoại
- Nhận hủy thông tin đặt phòng qua điện thoại

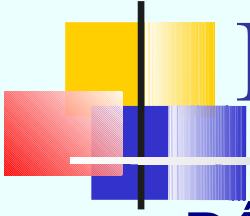


# Mô tả (6)

---

Nhân viên lễ tân được phép:

- Nhận đặt chỗ trực tiếp từ khách hàng
- Nhận hủy đặt chỗ trực tiếp từ khách hàng
- Nhận checkin khách hàng
- Nhận checkout và thanh toán cho khách hàng

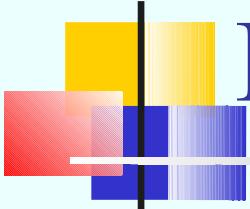


# BM: nhân viên nói chung (1)

---

Đối với tất cả các nhân viên:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login, trên menu trang chủ tương ứng với từng nhân viên đều có menu để chọn chức năng thay đổi mật khẩu, và chức năng logout

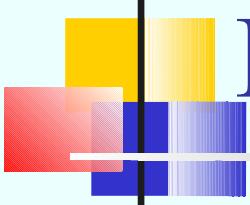


# BM: nhân viên nói chung (2)

---

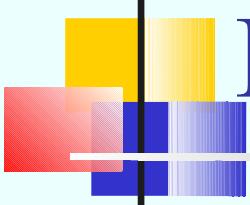
Vậy có thể có các use case:

- Login
- Change password (Thay đổi mật khẩu)
- Logout
- Uc Change password và uc Logout là mở rộng từ uc Login



# BM: nhân viên nói chung (3)

---

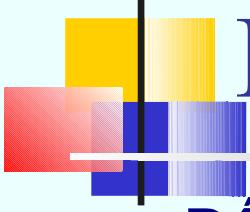


# BM: Nhân viên nói chung (4)

---

Mô tả các use case:

- Login: Use case này cho phép nhân viên đăng nhập theo tài khoản của mình
- Change password: use case này cho phép nhân viên thay đổi mật khẩu đăng nhập của mình sau khi đăng nhập
- Logout: use case này cho phép nhân viên đăng xuất sau khi hoàn thành nhiệm vụ hoặc hết phiên làm việc của mình

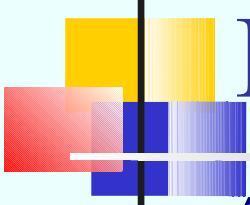


# BM: người quản lí (1)

---

Đối với người quản lí:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login, trong menu chính có chọn xem báo cáo, và chọn quản lí thông tin các phòng
- Khi chọn xem báo cáo thì có thể tùy chọn các loại báo cáo khác nhau: báo cáo doanh thu theo thời gian (nhập thời gian), báo cáo doanh thu theo phòng (nhập mã phòng), báo cáo tỉ lệ phòng trống theo thời gian (nhập thời gian)
- Nếu không nhập dữ liệu (thời gian, mã phòng) thì thống kê tất cả

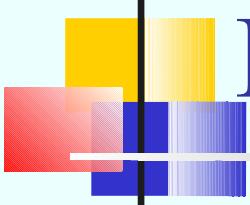


# BM: người quản lí (2)

---

Đối với người quản lí (tt):

- Trong menu chính quản lí phòng có chọn: thêm phòng, sửa phòng, xóa phòng
- Khi chọn thêm thì form thêm phòng hiện ra để nhập thông tin phòng: name, type, displayPrice, description, và nút thêm
- Khi chọn sửa hoặc xóa thì hiện lên form yêu cầu nhập tên phòng cần xóa để tìm ra một danh sách phòng có tên đã nhập. Khi chọn tên tương ứng để xóa thì hiện form tương tự khi thêm, với các ô có sẵn thông tin để sửa. Khi chọn tên tương ứng để xóa thì hiện lên ô xác nhận và xóa.



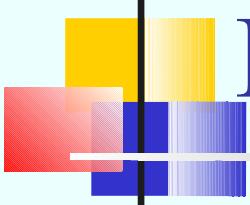
# BM: người quản lí (3)

---

Vậy phải có các use case:

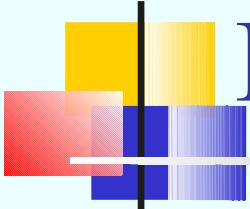
- Login: Nhưng để xuất hiện menu của người quản lí ngay sau khi login thì ta gọi là uc Manager login
- View a report: xem báo cáo
- Manage room: quản lí thông tin phòng

Các hoạt động này là tùy chọn sau khi login, và có thể thực hiện nhiều lần không cần login lại, nên nó là extend từ uc Manager login



# BM: người quản lí (4)

---

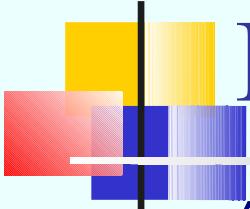


# BM: người quản lí (4)

---

Mô tả các use case:

- Manager login: Use case này cho phép người quản lí đăng nhập theo tài khoản của mình
- View a report: use case này cho phép người quản lí xem một báo cáo về doanh thu hoặc tỉ lệ phòng trống
- Manage room: use case này cho phép người quản lí thêm, hoặc sửa, hoặc xóa thông tin về phòng của khách sạn

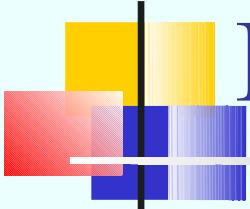


# BM: quản trị hệ thống (1)

---

Đối với nhân viên quản lí hệ thống:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login, trong menu chính quản lí người dùng có chọn: thêm user, sửa user, xóa user
- Khi chọn thêm thì form thêm user hiện ra để nhập thông tin user: username, password, fullname, birthday, address, mail, role, description, và nút thêm
- Khi chọn sửa hoặc xóa thì hiện lên form yêu cầu nhập tên người cần xóa để tìm ra một danh sách người dùng có tên đã nhập. Khi chọn tên tương ứng để xóa thì hiện form tương tự khi thêm, với các ô có sẵn thông tin để sửa. Khi chọn tên tương ứng để xóa thì hiện lên ô xác nhận và xóa.

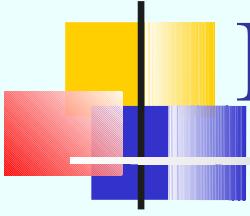


# BM: quản trị hệ thống (2)

---

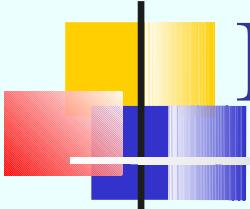
Vậy phải có các use case:

- Login: Nhưng để xuất hiện menu của người quản trị hệ thống ngay sau khi login thì ta gọi là uc Admin login
- Manage account: hoạt động này là tùy chọn sau khi login, và có thể thực hiện nhiều lần không cần login lại, nên nó là extend từ uc Admin login



# BM: quản trị hệ thống (3)

---

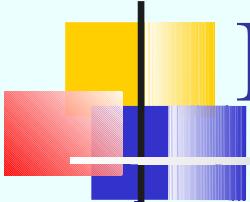


# BM: quản trị hệ thống (4)

---

Mô tả các use case:

- Admin login: Use case này cho phép người quản trị hệ thống đăng nhập theo tài khoản của mình
- Manage an account: use case này cho phép người quản trị hệ thống thêm, sửa, xóa một tài khoản của người dùng hệ thống khi có yêu cầu từ chính người dùng đó.

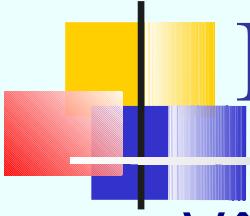


# BM: nhân viên bán hàng (1)

---

Đối với nhân viên bán hàng:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login, trong menu chính có chọn đặt chỗ và hủy đặt chỗ
- Khi khách hàng gọi đến yêu cầu đặt chỗ, nhân viên phải tìm phòng trống theo thời gian khách đưa ra, hệ thống hiện danh sách phòng trống theo yêu cầu, nhân viên yêu cầu khách hàng chọn phòng và lưu thông tin đặt phòng, bao gồm cả thông tin của khách hàng
- Khi khách hàng gọi điện đến yêu cầu hủy đặt chỗ, nhân viên tìm thông tin đặt chỗ theo tên khách hàng, hệ thống hiện lên danh sách đặt phòng, nhân viên xác nhận thông tin với khách hàng và xóa thông tin đặt phòng

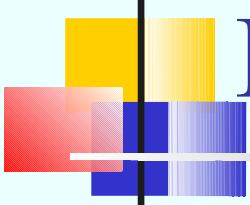


# BM: nhân viên bán hàng (2)

---

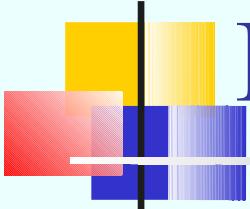
Vậy phải có các use case:

- Login: Nhưng để xuất hiện menu của nhân viên bán hàng ngay sau khi login thì ta gọi là uc Seller login
- Booking by phone: hoạt động này là tùy chọn sau khi login, và có thể thực hiện nhiều lần không cần login lại, nên nó là extend từ uc Seller login
- Cancel by phone: hoạt động này là tùy chọn sau khi login, và có thể thực hiện nhiều lần không cần login lại, nên nó là extend từ uc Seller login



# BM: nhân viên bán hàng (3)

---

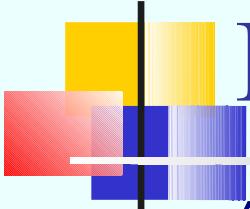


# BM: nhân viên bán hàng (4)

---

Mô tả các use case:

- Seller login: Use case này cho phép nhân viên bán hàng đăng nhập theo tài khoản của mình
- Booking by phone: use case này cho phép nhân viên bán hàng đặt phòng khi có yêu cầu từ khách hàng qua điện thoại.
- Cancel by phone: use case này cho phép nhân viên bán hàng hủy đặt phòng khi có yêu cầu từ khách hàng qua điện thoại.

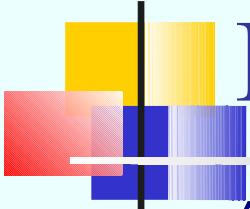


# BM: nhân viên lễ tân (1)

---

Đối với nhân viên lễ tân:

- Phải login để thực hiện các hoạt động nghiệp vụ của mình
- Sau khi login, trong menu chính có chọn: đặt phòng, hủy đặt phòng, checkin, checkout cho khách yêu cầu tại chỗ
- Khi chọn đặt phòng hay hủy đặt phòng thì phần mềm hoạt động tương tự các chức năng với nhân viên bán hàng

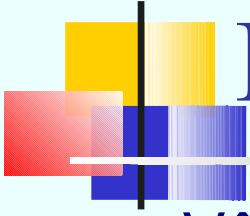


# BM: nhân viên lễ tân (2)

---

Đối với nhân viên lễ tân (tt):

- Khi chọn checkin thì hệ thống cho phép chọn tìm kiếm đặt phòng theo tên khách hàng, hệ thống hiện danh sách đặt phòng, nhân viên chọn cập nhật phòng tương ứng với khách hàng
- Khi chọn checkout thì hệ thống cho phép tìm phòng theo mã, hiện thông tin chi tiết hóa đơn và in ra cho khách hàng thanh toán, sau đó cập nhật lại trạng thái phòng



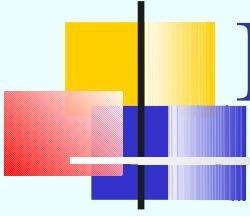
# BM: nhân viên lễ tân (3)

---

Vậy phải có các use case:

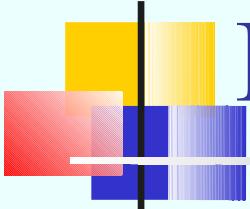
- Login: Nhưng để xuất hiện menu của nhân viên lễ tân ngay sau khi login thì ta gọi là uc Receptionist login
- Booking on site: đặt phòng trực tiếp
- Cancel on site: hủy đặt phòng trực tiếp
- Checkin: nhận phòng đã đặt
- Checkout: trả phòng và thanh toán

Các hoạt động này là tùy chọn sau khi login, và có thể thực hiện nhiều lần không cần login lại, nên nó là extend từ uc Receptionist login



# BM: nhân viên lẽ tân (4)

---

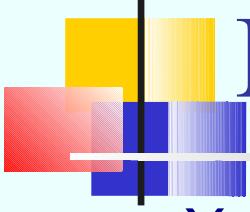


# BM: nhân viên lễ tân (5)

---

Mô tả các use case:

- Receptionist login: Use case này cho phép nhân viên lễ tân đăng nhập theo tài khoản của mình
- Booking on site: use case này cho phép nhân viên lễ tân đặt phòng khi có yêu cầu từ khách hàng tại quầy.
- Cancel on site: use case này cho phép nhân viên lễ tân hủy đặt phòng khi có yêu cầu từ khách hàng tại quầy.
- Checkin: use case này cho phép nhân viên lễ tân cập nhật thông tin khách đã nhận phòng
- Checkout: use case này cho phép nhân viên lễ tân cập nhật thông tin khách trả phòng và thanh toán cho khách hàng

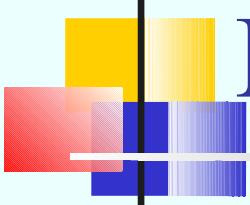


# Mịn hóa BM (1)

---

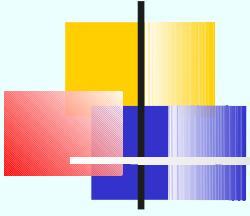
Xem xét lại các use case:

- Nhân viên bán hàng và nhân viên lễ tân cùng có uc đặt chỗ và hủy đặt chỗ
  - Có thể gộp chung lại thành uc đặt chỗ và hủy đặt chỗ chung về phía khách hàng, và tách ra đối với mỗi kiểu nhân viên



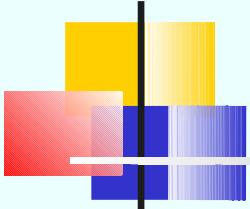
# Mịn hóa BM (2)

---



# Kết quả sơ đồ UC

---

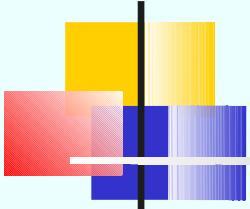


# Chi tiết BM (1)

---

Chi tiết thêm một số use case:

- Trong uc Booking phải tìm kiếm phòng trống
- Trong uc Cancel phải tìm kiếm thông tin đặt phòng theo tên khách hàng (hoặc theo ngày đặt)
- Trong uc Checkin cũng phải tìm kiếm thông tin đặt phòng theo tên khách hàng
- Trong uc Checkout cũng phải tìm kiếm thông tin đặt phòng theo phòng (hoặc theo khách hàng)
- Trong uc Manage account, khi sửa và xóa account cũng phải tìm kiếm thông tin account theo tên user
- Trong uc Manage room, khi sửa và xóa room cũng phải tìm kiếm thông tin về phòng
- Trong uc Checkout phải tạo hóa đơn thanh toán

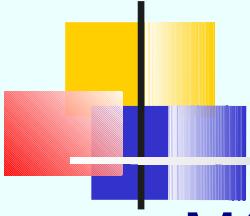


# Chi tiết BM (2)

---

Như vậy cần bổ sung thêm một số use case:

- Search free room: tìm kiếm phòng trống
- Search booking: tìm kiếm thông tin đặt phòng theo tên khách hàng (hoặc theo ngày đặt)
- Search account: tìm kiếm thông tin account theo tên user
- Search room: tìm kiếm phòng theo tên
- Payment: thanh toán cho khách hàng khi checkout

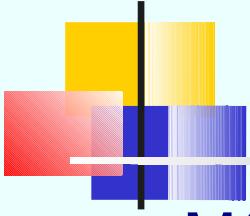


# Chi tiết BM (3)

---

Mô tả use case:

- Search free room: Use case này cho phép uc Booking tìm kiếm phòng trống trong một khoảng thời gian theo yêu cầu của khách hàng

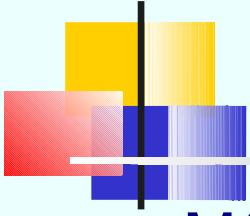


# Chi tiết BM (4)

---

Mô tả use case:

- Search booking: use case này cho phép tìm kiếm thông tin đặt phòng theo tên khách hàng (hoặc theo ngày đặt)

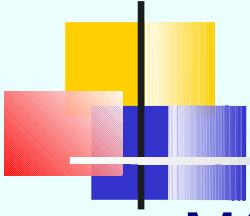


# Chi tiết BM (5)

---

Mô tả use case:

- Search account: cho phép uc Update account và uc Delete account tìm kiếm thông tin account theo tên user

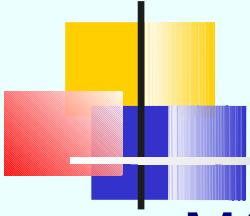


# Chi tiết BM (6)

---

Mô tả use case:

- Search room: cho phép uc Update room và uc Delete room tìm kiếm thông tin phòng theo tên

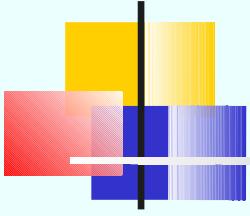


# Chi tiết BM (7)

---

Mô tả use case:

- Payment: cho phép cập nhật thông tin thanh toán cho khách hàng khi checkout

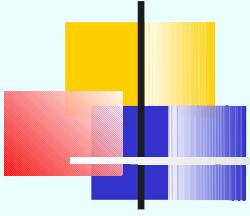


# Bài tập (1)

---

Các nhóm làm bài tập với đề bài tập lớn của mình:

- Lập danh sách các từ chuyên môn của ứng dụng
- Mô tả các hoạt động nghiệp vụ của dự án
- Xác định các UC ban đầu
- Chi tiết và làm mịn các UC ban đầu này

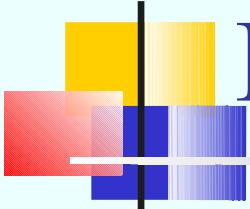


# Bài tập (2)

---

Các nhóm làm bài tập lớn:

- Tinh chỉnh các UC đã khởi tạo từ bài tập số 1 để được sơ đồ UC và mô tả các UC phiên bản cuối cùng của dự án

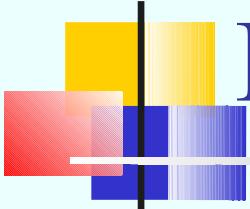


# Bài tập nhóm phải nộp (r2)

---

Tất cả các thành viên trong nhóm làm chung:

- Mô tả yêu cầu chi tiết đối với hệ thống do nhóm sẽ phát triển
- Mô tả các hoạt động nghiệp vụ của từng chức năng
- Sau đó vẽ sơ đồ uc cho từng chức năng
- Minh họa mô hình nghiệp vụ (BM) như trong bài

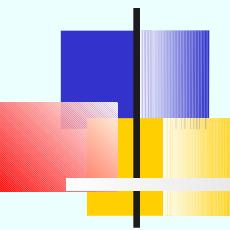


# Bài tập nhóm phải nộp (r2)

---

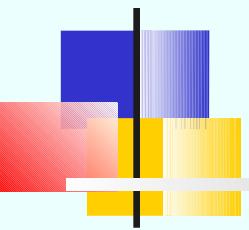
Báo cáo cần có:

- Danh sách các từ chuyên môn trong lĩnh vực của ứng dụng (glossary)
- Mô tả hệ thống chi tiết bằng ngôn ngữ tự nhiên
- Sơ đồ tổng quan các use case của toàn hệ thống (chỉ cần sơ đồ)
- Với mỗi sơ đồ use case, vẽ sơ đồ chi tiết và mô tả các use case như trong bài
- Nộp báo cáo chung, ghi đầy đủ họ tên, lớp, nhóm ngay đầu báo cáo
- Điểm đánh giá chung cả nhóm



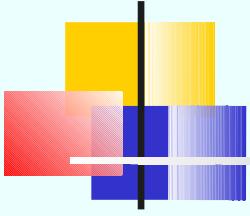
# Questions?

---



# Công nghệ phần mềm Pha phân tích

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

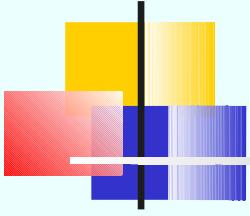


# Pha phân tích (1)

---

Mục đích:

- Giúp đội phát triển hiểu sâu hơn yêu cầu của khách hàng
- Đặc tả yêu cầu của khách hàng dưới dạng có thể làm đầu vào cho thiết kế và cài đặt được

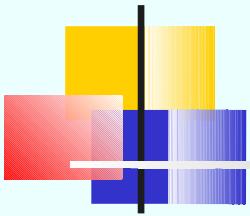


# Pha phân tích (2)

---

Thực hiện:

- Trích các lớp: lớp thực thể, lớp biên, lớp điều khiển
- Xác định quan hệ (ban đầu giữa các lớp)

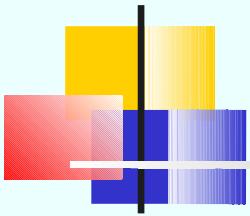


# Pha phân tích (3)

---

Lớp thực thể (còn gọi là lớp model):

- Dùng để biểu diễn dữ liệu để xử lý, trao đổi giữa các đối tượng trong hệ thống
- Thường chỉ có các thuộc tính và các phương thức truy nhập *get/set*

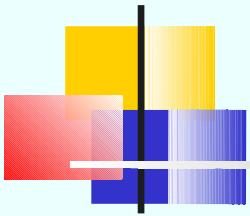


# Pha phân tích (4)

---

Lớp biên (còn gọi là lớp view):

- Dùng để biểu diễn các dạng giao diện, giao tiếp giữa người dùng và hệ thống
- Mỗi lớp biên thường liên quan đến một thiết bị đầu vào, hoặc đầu ra của hệ thống

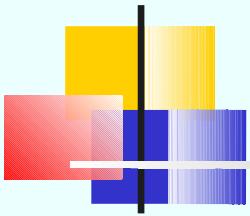


# Pha phân tích (5)

---

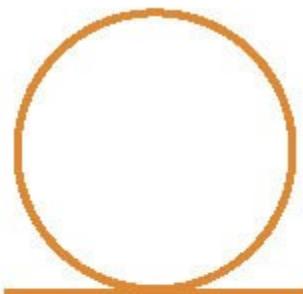
Lớp điều khiển (còn gọi là lớp control):

- Dùng để mô hình các tính toán và thuật toán phức tạp trong hệ thống
- Có thể chỉ cần dùng một lớp điều khiển cho các hệ thống đơn giản, mỗi phương thức là một hàm xử lí, tính toán độc lập

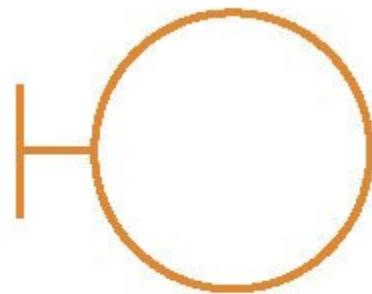


# Pha phân tích (6)

Biểu diễn các dạng lớp trong UML:



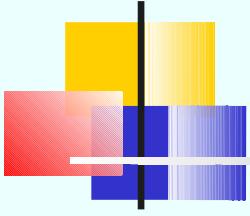
**Entity Class**



**Boundary Class**



**Control Class**

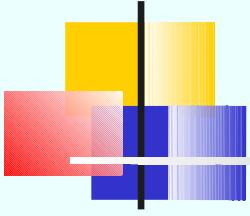


# Trích các lớp

---

Lặp lại 3 bước chính sau:

- B1: Mô hình hóa chức năng
- B2: Mô hình hóa các lớp
- B3: Mô hình hóa hoạt động



# Mô hình hóa chức năng

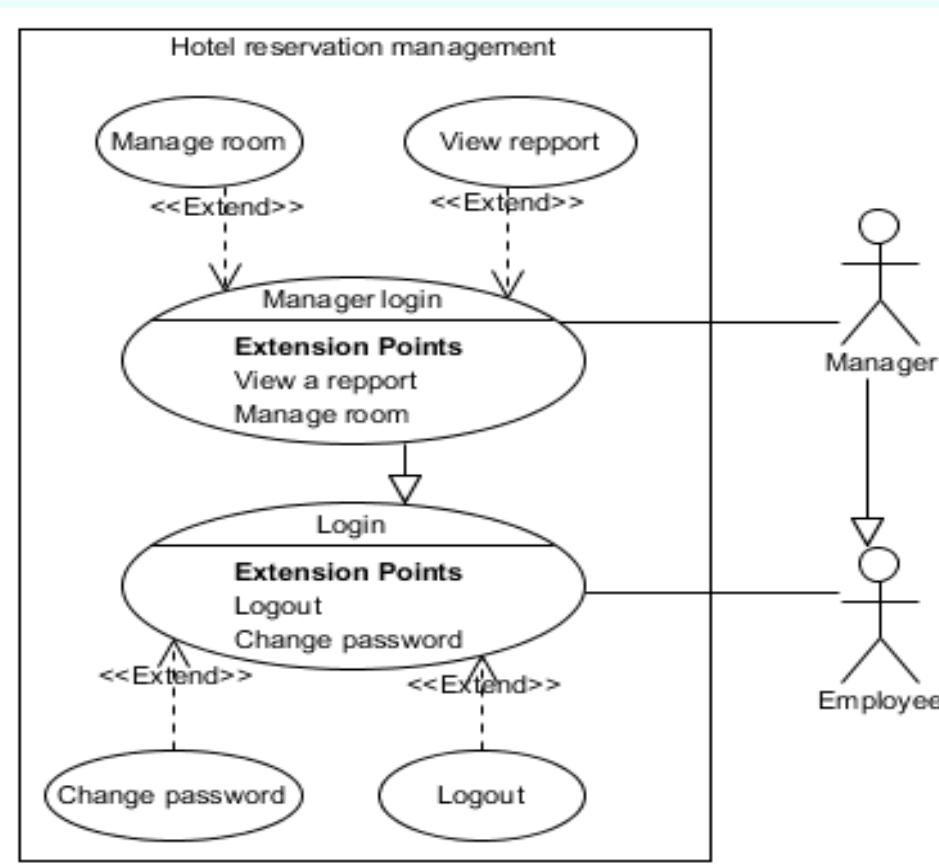
---

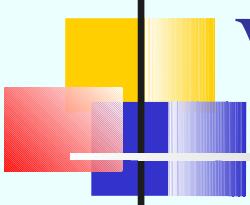
## Thực hiện:

- Với mỗi use case (kết quả từ pha yêu cầu), viết ít nhất một scenario cho use case đấy
- Một scenario là một kịch bản cụ thể khi người sử dụng tương tác với hệ thống

# Viết scenario (1)

Ví dụ với bài toán đặt phòng khách sạn, các use case của người quản lý:



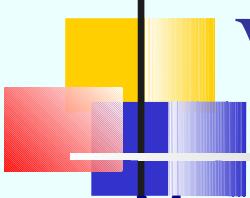


# Viết scenario (2)

---

Mô tả các use case:

- Manager login: Use case này cho phép người quản lý đăng nhập theo tài khoản của mình
- View a report: use case này cho phép người quản lý xem một báo cáo về doanh thu hoặc tỉ lệ phòng trống
- Manage room: use case này cho phép người quản lý thêm, hoặc sửa, hoặc xóa thông tin về phòng của khách sạn

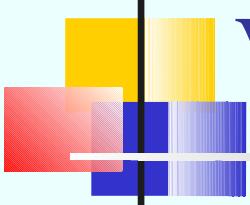


# Viết scenario (3)

---

Manage room: scenario chuẩn cho thêm phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn thêm thông tin một phòng mới.
2. Giao diện quản lí phòng hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút thêm phòng.
4. Giao diện thêm phòng hiện ra với các ô nhập: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 2 nút: nút thêm phòng, và nút hủy bỏ.
5. Nhân viên A nhập các thông tin phòng mới vào các ô và click nút thêm phòng
6. Thông báo thêm phòng thành công hiện ra.
7. A click vào nút ok
8. Hệ thống quay về trang chủ người quản lí

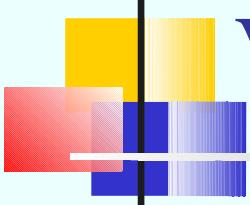


# Viết scenario (4)

---

Manage room: scenario ngoại lệ cho thêm phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn thêm thông tin một phòng mới.
2. Giao diện quản lí phòng hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút thêm phòng.
4. Giao diện thêm phòng hiện ra với các ô nhập: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 2 nút: nút thêm phòng, và nút hủy bỏ.
5. Nhân viên A nhập các thông tin phòng mới vào các ô và click nút thêm phòng
6. Thông báo phòng với id vừa nhập đã tồn tại hiện ra.
7. A click vào nút ok

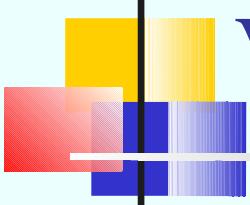


# Viết scenario (5)

---

Manage room: scenario ngoại lệ cho thêm phòng (tt)

8. Hệ thống quay lại giao diện nhập phòng với các thông tin đã nhập lần trước
9. Nhân viên A nhập lại id mới và click nút thêm phòng
10. Thông báo thêm phòng thành công hiện ra.
11. A click vào nút ok
12. Hệ thống quay về trang chủ người quản lí

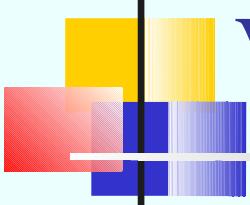


# Viết scenario (6)

---

Manage room: scenario chuẩn cho sửa phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn sửa thông tin phòng 305.
2. Giao diện quản lí phòng hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút sửa phòng.
4. Giao diện tìm kiếm phòng hiện ra với một ô nhập tên phòng và một nút tìm kiếm
5. A nhập 305 vào ô tên phòng và click vào nút tìm kiếm
6. Giao diện kết quả tìm kiếm hiện ra gồm một bảng các phòng có tên 305, mỗi dòng có đầy đủ thông tin một phòng với các cột: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 1 nút chọn sửa.

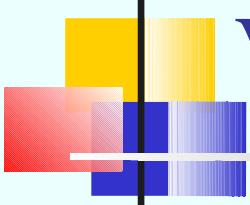


# Viết scenario (7)

---

Manage room: scenario chuẩn cho sửa phòng (tt)

7. A chọn click vào nút chọn sửa của dòng thứ nhất.
8. Giao diện sửa phòng hiện ra với các ô chứa sẵn thông tin phòng đã chọn gồm có: id phòng(không sửa được), tên phòng, kiểu phòng, giá hiển thị, mô tả, 1 nút hủy bỏ và 1 nút sửa.
9. A sửa thông tin loại phòng và mô tả phòng, và click vào nút sửa.
10. Thông báo sửa phòng thành công hiện ra.
11. A click vào nút OK
12. Hệ thống quay lại trang chủ của người quản lí.

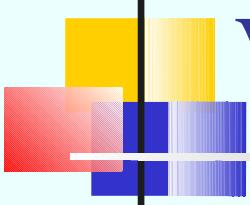


# Viết scenario (8)

---

Manage room: scenario chuẩn cho xóa phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn xóa thông tin phòng 503.
2. Giao diện quản lí phòng hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút sửa phòng.
4. Giao diện tìm kiếm phòng hiện ra với một ô nhập tên phòng và một nút tìm kiếm
5. A nhập 503 vào ô tên phòng và click vào nút tìm kiếm
6. Giao diện kết quả tìm kiếm hiện ra gồm một bảng các phòng có tên 503, mỗi dòng có đầy đủ thông tin một phòng với các cột: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 1 nút chọn xóa.

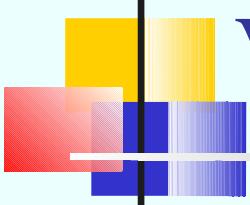


# Viết scenario (9)

---

Manage room: scenario chuẩn cho xóa phòng (tt)

7. A chọn click vào nút chọn xóa của dòng thứ nhất.
8. Một dialog hiện ra yêu cầu nhân viên quản lý xác nhận có muốn xóa thông tin phòng 503 hay không
9. A click vào nút xác nhận có muốn xóa.
10. Thông báo xóa phòng thành công hiện ra.
11. A click vào nút OK
12. Hệ thống quay lại trang chủ của người quản lý.

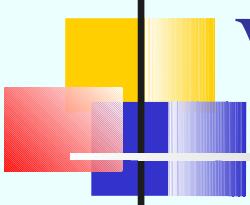


# Viết scenario (10)

---

Yêu cầu của một scenario:

- Luôn phải nêu rõ ý định, mục đích của actor trước khi bắt đầu scenario hoặc ngay trong bước 1
- Các bước luôn đánh số thứ tự từ 1
- Mỗi một bước chỉ có một hành động đơn. Nếu có hai hành động trở lên thì các hành động đó phải cùng một chủ thể
- Hai bước liên tiếp mà có cùng một chủ thể thì nên gộp lại thành một bước
- Scenario kết thúc khi chủ thể ban đầu đạt được mục đích ban đầu
- Một scenario có thể có nhiều ngoại lệ, các ngoại lệ cần được mô tả hết



# Viết scenario (11)

---

Yêu cầu của một scenario:

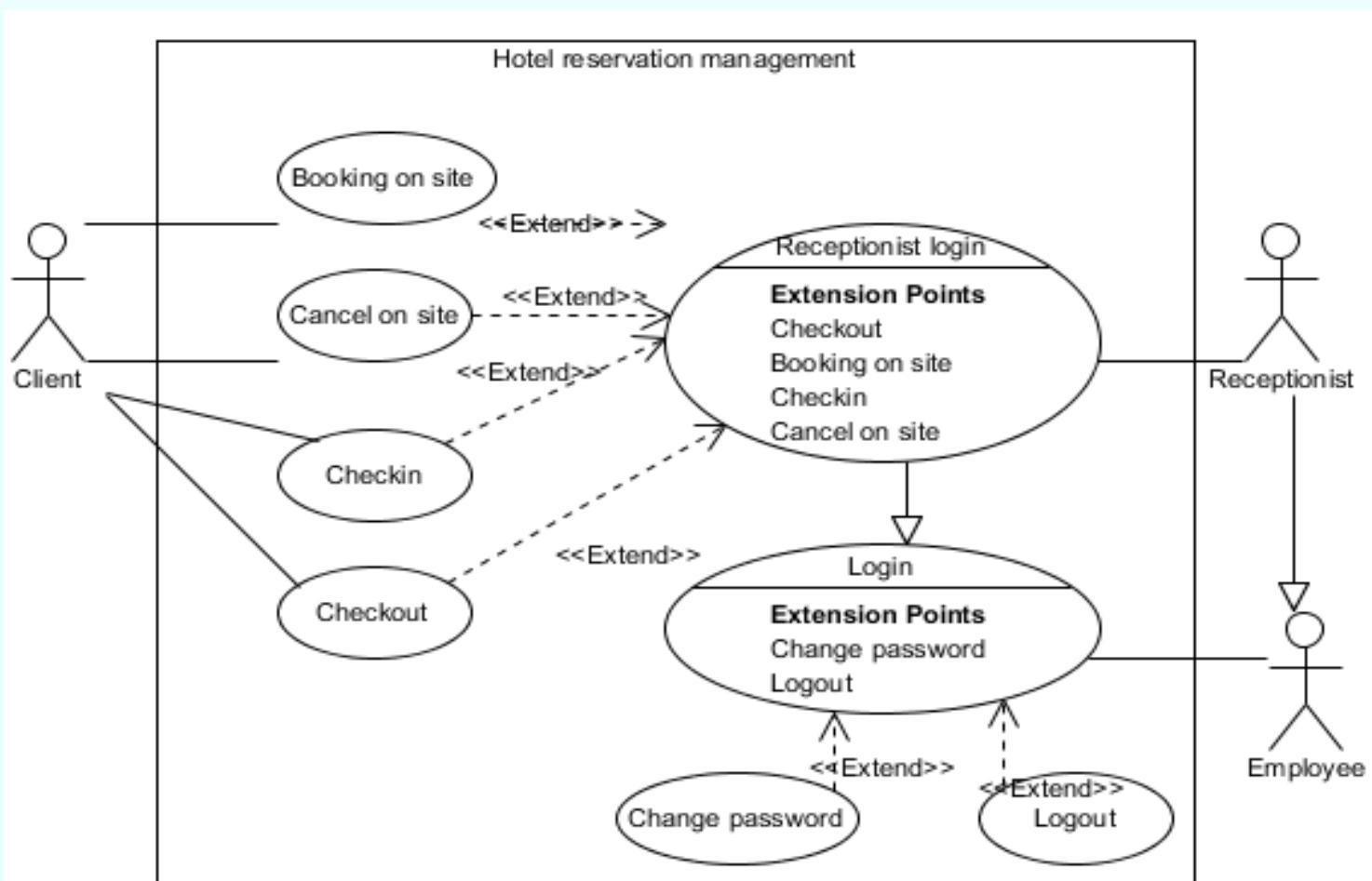
- Thông tin trong mỗi bước là phải cụ thể, không được viết chung chung

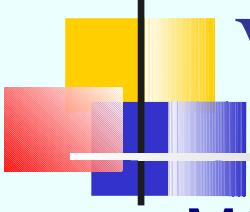
Ví dụ viết thế này là **không được**:

1. Nhân viên click vào nút xóa phòng
2. Giao diện tìm kiếm phòng hiện ra
3. Nhân viên nhập tên phòng vào
4. Giao diện kết quả hiện ra
5. Nhân viên chọn xóa 1 phòng
6. Hệ thống thông báo xóa thành công

# Viết scenario (12)

Ví dụ với bài toán đặt phòng khách sạn, các use case của nhân viên lễ tân:



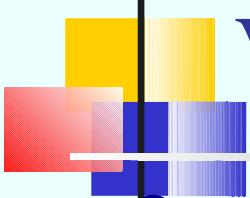


# Viết scenario (13)

---

Mô tả các use case:

- Receptionist login: Use case này cho phép nhân viên lễ tân đăng nhập theo tài khoản của mình
- Booking on site: use case này cho phép nhân viên lễ tân đặt phòng khi có yêu cầu từ khách hàng tại quầy.
- Cancel on site: use case này cho phép nhân viên lễ tân hủy đặt phòng khi có yêu cầu từ khách hàng tại quầy.
- Checkin: use case này cho phép nhân viên lễ tân cập nhật thông tin khách đã nhận phòng
- Checkout: use case này cho phép nhân viên lễ tân cập nhật thông tin khách trả phòng và thanh toán cho khách hàng

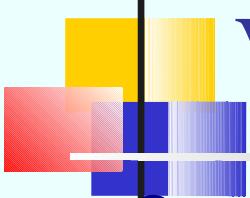


# Viết scenario (14)

---

## Scenario chuẩn cho đặt phòng tại chỗ

1. Nhân viên tiếp tân A chọn chức năng quản lý đặt phòng sau khi login. A muốn thêm thông tin đặt phòng do khách hàng B yêu cầu.
2. Giao diện quản lý đặt phòng hiện ra với 3 nút: thêm, sửa, hủy đặt phòng
3. Nhân viên A click vào nút thêm đặt phòng.
4. Giao diện tìm phòng trống hiện ra với các ô nhập: ngày bắt đầu, ngày kết thúc và nút tìm kiếm.
5. Nhân viên A hỏi khách hàng B ngày bắt đầu, kết thúc mong muốn.
6. Khách hàng B nói với nhân viên A ngày bắt đầu, kết thúc.
7. Nhân viên A nhập ngày bắt đầu, kết thúc và click nút tìm kiếm.

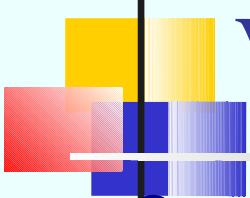


# Viết scenario (15)

---

Scenario chuẩn cho đặt phòng tại chỗ (tt)

8. Giao diện kết quả tìm kiếm phòng trống hiện ra với bảng danh sách các phòng trong khoảng ngày đã chọn. Mỗi phòng tương ứng với một dòng với các thông tin: id phòng, tên phòng, kiểu phòng, giá, mô tả, và nút chọn đặt.
9. Nhân viên A thông báo danh sách các phòng trống cho khách hàng B chọn.
10. Khách hàng B chọn phòng thứ 2 trong danh sách.
11. A click vào nút chọn đặt của dòng thứ hai.
12. Giao diện nhập thông tin khách hàng hiện ra, bao gồm các ô nhập thông tin khách hàng: họ tên, số CMND/passport, kiểu giấy id, địa chỉ, mô tả, ghi chú. Một nút thêm, một nút tìm kiếm.

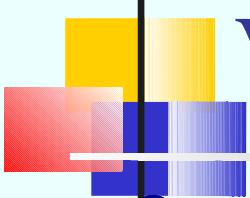


# Viết scenario (16)

---

Scenario chuẩn cho đặt phòng tại chỗ (tt)

13. Nhân viên A hỏi khách hàng B tên
14. Khách hàng B nói tên mình là B cho nhân viên A.
15. Nhân viên A nhập tên B vào ô họ tên và click nút tìm kiếm.
16. Giao diện kết quả tìm kiếm thông tin khách hàng hiện ra gồm một bảng danh sách các khách hàng có tên B, mỗi khách hàng chứa các thuộc tính tương ứng với các cột: họ tên, số CMND/passport, kiểu giấy id, địa chỉ, mô tả, ghi chú, một nút chọn. Dưới cùng là nút thêm khách hàng mới.
17. Nhân viên A xác nhận các thông tin với khách hàng B
18. Khách hàng B xác nhận thông tin của mình chưa có trong danh sách đã tìm thấy.

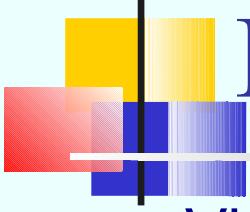


# Viết scenario (17)

---

Scenario chuẩn cho đặt phòng tại chỗ (tt)

19. Nhân viên A click vào nút thêm mới khách hàng ở phía dưới.
20. Giao diện nhập thông tin khách hàng mới hiện ra với các ô nhập: họ tên, số CMND/passport, kiểu giấy id, địa chỉ, mô tả, ghi chú, một nút chọn. Dưới cùng là nút thêm khách hàng mới.
21. Nhân viên A nhập các thông tin khách hàng B và click vào nút thêm mới
22. Giao diện xác nhận đặt phòng hiện lên với đầy đủ các thông tin: phòng đặt, ngày checkin, ngày checkout, giá đặt, khách hàng đặt. Dưới cùng là nút xác nhận và nút hủy bỏ.
23. Nhân viên A click vào nút xác nhận
24. Hệ thống báo đặt chỗ thành công và quay về trang chủ của nhân viên lễ tân



# Bài tập trên lớp

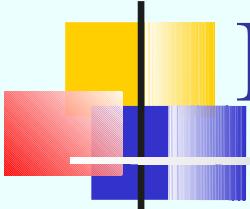
---

Viết tất cả các scenario ngoại lệ cho:

- Sửa thông tin phòng
- Xóa thông tin phòng
- Đặt chỗ

Viết scenario chuẩn và ngoại lệ cho chức năng:

- Checkin
- Sửa đặt phòng
- Trả phòng + thanh toán tại chỗ

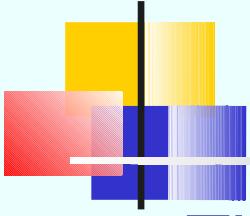


# Bài tập về nhà

---

Với chủ đề bài tập lớn của nhóm:

- Chọn một use case trong modul của mình (mà bản thân cho là khó nhất và hay nhất)
- Viết lại phần mô tả modul đó trong pha yêu cầu
- Vẽ lại sơ đồ use case
- Viết 1 scenario chuẩn và tất cả các scenario ngoại lệ cho use case đã chọn

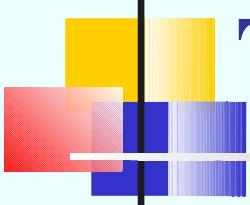


# Mô hình hóa các lớp (1)

---

Thực hiện:

- Trích các lớp thực thể và các thuộc tính của chúng
- Xác định quan hệ và tương tác giữa các lớp này
- Biểu diễn các thông tin này trên sơ đồ lớp (khởi tạo)



# Trích lớp thực thể (1)

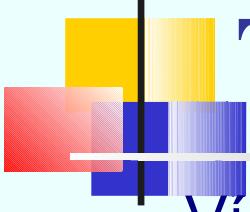
---

Kĩ thuật trích danh từ để trích các lớp:

- Mô tả hoạt động của ứng dụng trong một đoạn văn
- Trích các danh từ xuất hiện trong đoạn văn đó, coi như là các ứng cử viên của lớp thực thể
- Xét duyệt từng danh từ và đề xuất nó là **lớp thực thể** hay là **thuộc tính của lớp thực thể**

Lưu ý:

- Có thể thay đoạn văn trong bước 1 bằng cách tập hợp các scenario đã viết trong bước trước

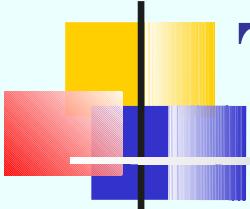


# Trích lớp thực thể (2)

---

Ví dụ mô tả bài toán đặt phòng khách sạn:

- Hệ thống phục vụ hoạt động quản lý đặt phòng của một khách sạn. Trong đó, nhân viên quản lý có thể quản lý thông tin phòng và xem các báo cáo. Nhân viên quản trị có thể quản lý các tài khoản người dùng hệ thống. Nhân viên bán hàng có thể đặt phòng, thay đổi và hủy đặt phòng cho khách hàng thông qua điện thoại. Nhân viên tiếp tân có thể đặt phòng, thay đổi đặt phòng, hủy đặt phòng, làm thủ tục checkin, checkout và thanh toán trực tiếp tại chỗ cho khách hàng. Khi thanh toán có thể xuất hóa đơn theo yêu cầu của khách hàng, bao gồm tiền phòng và chi phí các dịch vụ gia tăng của khách sạn mà khách hàng đã dùng.

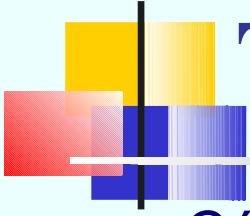


# Trích lớp thực thể (3)

---

Ví dụ mô tả bài toán đặt phòng khách sạn:

- **Hệ thống** phục vụ hoạt động quản lý đặt phòng của một khách sạn. Trong đó, **nhân viên quản lý** có thể quản lý thông tin **phòng** và xem các báo cáo. **Nhân viên quản trị** có thể quản lý các **tài khoản người dùng** hệ thống. **Nhân viên bán hàng** có thể đặt phòng, thay đổi và hủy đặt phòng cho khách hàng thông qua **điện thoại**. **Nhân viên tiếp tân** có thể đặt phòng, thay đổi đặt phòng, hủy đặt phòng, làm thủ tục checkin, checkout và thanh toán trực tiếp tại chỗ cho **khách hàng**. Khi thanh toán có thể xuất hóa đơn theo yêu cầu của **khách hàng**, bao gồm tiền phòng và **chi phí** các dịch vụ gia tăng của khách sạn mà **khách hàng** đã dùng.



# Trích lớp thực thể (4)

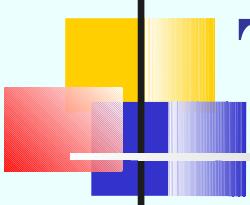
---

Các danh từ:

- Hệ thống, phòng, khách sạn, nhân viên quản lý, báo cáo, nhân viên quản trị, tài khoản người dùng, nhân viên bán hàng, khách hàng, điện thoại, nhân viên tiếp tân, hóa đơn, yêu cầu, tiền phòng, chi phí, dịch vụ gia tăng.

Đánh giá:

- **Điện thoại** nằm ngoài phạm vi của phần mềm → loại
- **Hệ thống, yêu cầu, tiền phòng, chi phí** là các danh từ trừu tượng → loại
- **Báo cáo** nên là một lớp biên hơn là lớp thực thể
- **Nhân viên quản lý, nhân viên quản trị, nhân viên bán hàng, nhân viên tiếp tân** đều có thể là các danh từ cụ thể của tài khoản người dùng

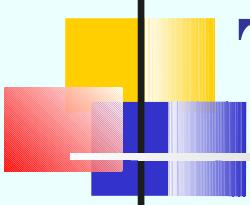


# Trích lớp thực thể (5)

---

Như vậy chỉ còn các lớp thực thể:

- Phòng: Room
- Khách sạn: Hotel
- Tài khoản người dùng: User
- Hóa đơn: Bill
- Khách hàng: Client
- Dịch vụ gia tăng: Service



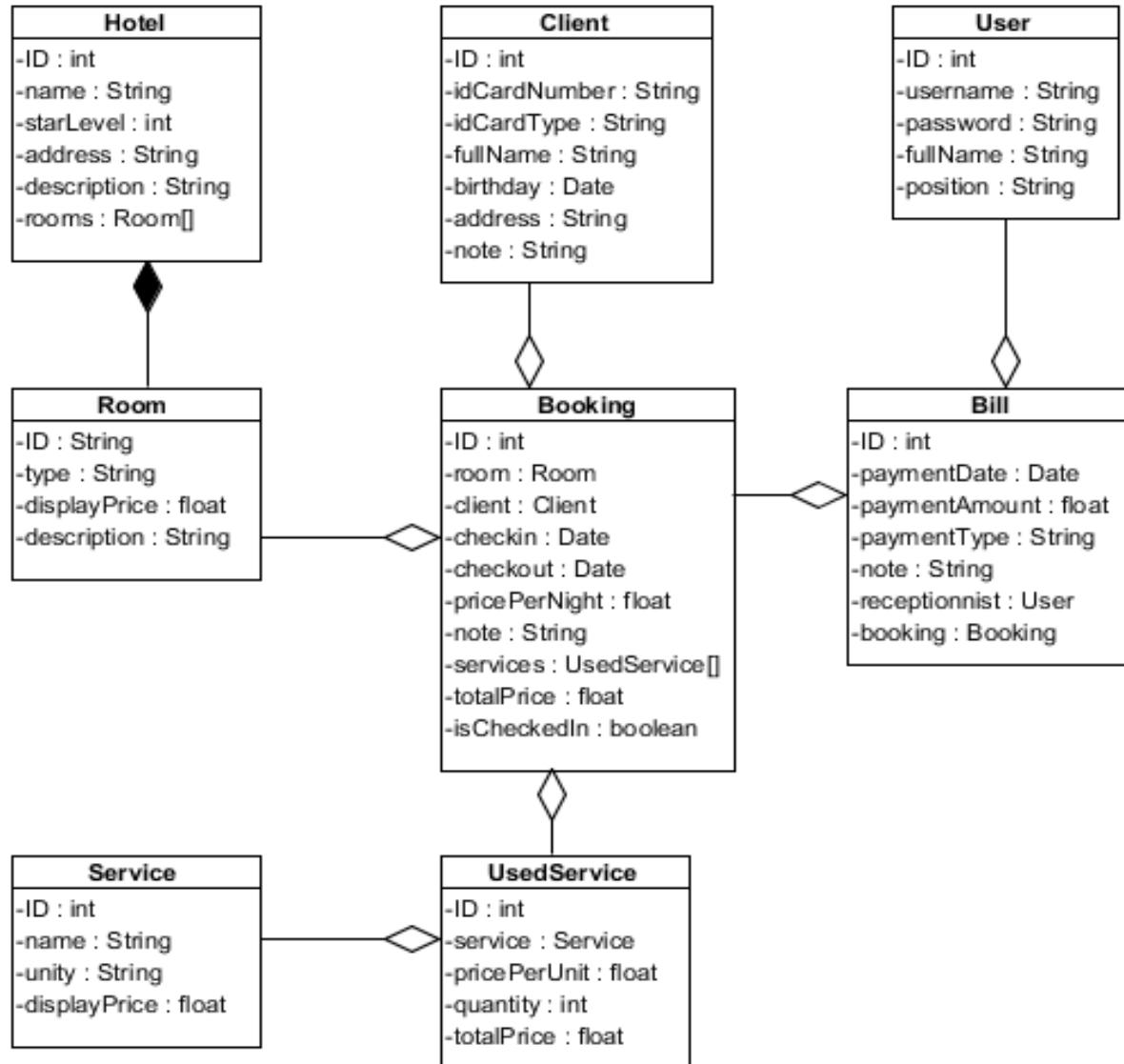
# Trích lớp thực thể (6)

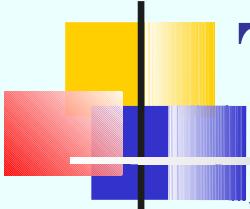
---

Quan hệ giữa các lớp thực thể:

- Một Hotel có nhiều Room, một Room phải thuộc vào một Hotel nhất định
- Một Room có thể đặt bởi nhiều Client, một Client lại có thể đặt nhiều Room tại nhiều thời điểm khác nhau  
→ Đề xuất thêm một lớp **Booking**
- Một Booking có thể dùng nhiều Service khác nhau, một Service lại có thể được sử dụng bởi nhiều Booking khác nhau → Đề xuất thêm lớp **UsedService**
- Một Booking có thể được thanh toán nhiều lần khác nhau nên có thể có nhiều Bill
- Mỗi Bill có tối đa một User lập và nhận thanh toán.

# Trích lớp thực thể (7)



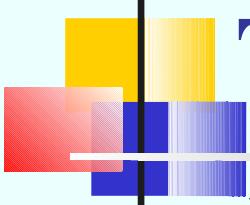


# Trích lớp điều khiển (1)

---

Đề xuất các lớp điều khiển:

- Toàn bộ hệ thống dùng chung một lớp điều khiển
- Mỗi modul dùng riêng một lớp điều khiển

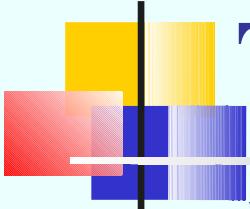


# Trích lớp điều khiển (2)

---

Đề xuất mỗi modul dùng riêng lớp điều khiển:

- Lớp điều khiển cho modul Manager: ManagerCtr
- Lớp điều khiển cho modul Admin: AdminCtr
- Lớp điều khiển cho modul Seller: SellerCtr
- Lớp điều khiển cho modul Receptionist: ReceptCtr

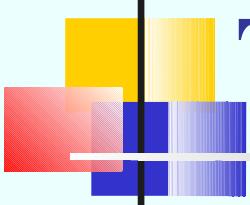


# Trích lớp biên (1)

---

Đề xuất các lớp biên:

- Mỗi giao diện (trang web, form) nên để là một lớp biên
- Mỗi báo cáo, biểu mẫu nên để là một lớp biên
- Các thông báo, các thông điệp xác nhận có thể xem xét tạo thành một lớp biên hoặc là thành phần của một lớp biên

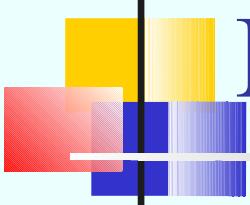


# Trích lớp biên (2)

---

Đề xuất các lớp biên cho modul quản lý phòng của Manager:

- Giao diện chính: RoomManagerFrm
- Chức năng thêm: form thêm (AddRoomFrm)
- Chức năng sửa: form tìm kiếm (SearchEditRoomFrm), form kết quả (chung với SearchEditRoomFrm), form sửa (EditRoomFrm)
- Chức năng xóa: form tìm kiếm (SearchDeleteRoomFrm), form kết quả dùng chung với SearchDeleteRoomFrm.
- Các dialog và cửa sổ con đều là thành phần của các form chính



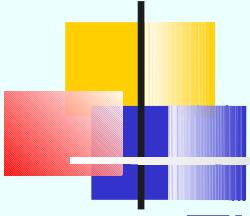
# Mô hình hóa các lớp (2)

---

Dùng thẻ CRC để mô hình hóa quan hệ giữa các lớp:

- C: class. Biểu diễn tên lớp
- R: responsibility. Trách nhiệm của lớp
- C: collaboration. Quan hệ của lớp

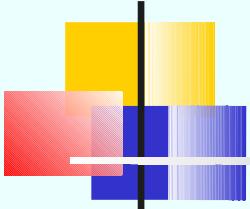
Trong VP: chọn new CRC Card diagram



# Mô hình hóa các lớp (3)

## Thẻ CRC cho lớp điều khiển modul Manager:

ManagerCtr	
Responsibilities:	
Name	Collaborator
Hiện giao diện quản lý phòng	
Hiện giao diện thêm phòng	
Hiện giao diện tìm kiếm phòng để sửa	
Hiện giao diện kết quả tìm kiếm phòng để sửa	
Hiện giao diện sửa phòng	
Hiện giao diện tìm kiếm phòng để xóa	
Hiện giao diện kết quả tìm kiếm phòng để xóa	
Lưu thông tin phòng mới vào CSDL	
Tìm kiếm phòng trong CSDL theo tên	
Cập nhật thông tin một phòng vừa sửa vào CSDL	
Xóa thông tin một phòng khỏi CSDL	
Đóng gói dữ liệu vào đối tượng Room	



# Mô hình hóa các lớp (4)

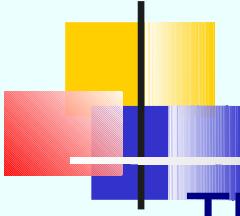
---

Thẻ CRC cho lớp điều khiển (tt):

- Vì hướng đối tượng che giấu dữ liệu của các lớp nên không thể viết:
  - Lớp điều khiển hiện giao diện quản lý phòng

Mà phải viết theo dạng tương tác giữa các lớp:

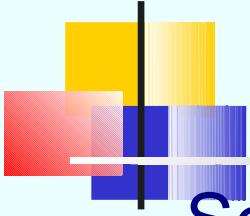
- Lớp **ManagerCtr** gửi thông điệp yêu cầu lớp **RoomManagerFrm** hiển thị giao diện chính



# Mô hình hóa các lớp (5)

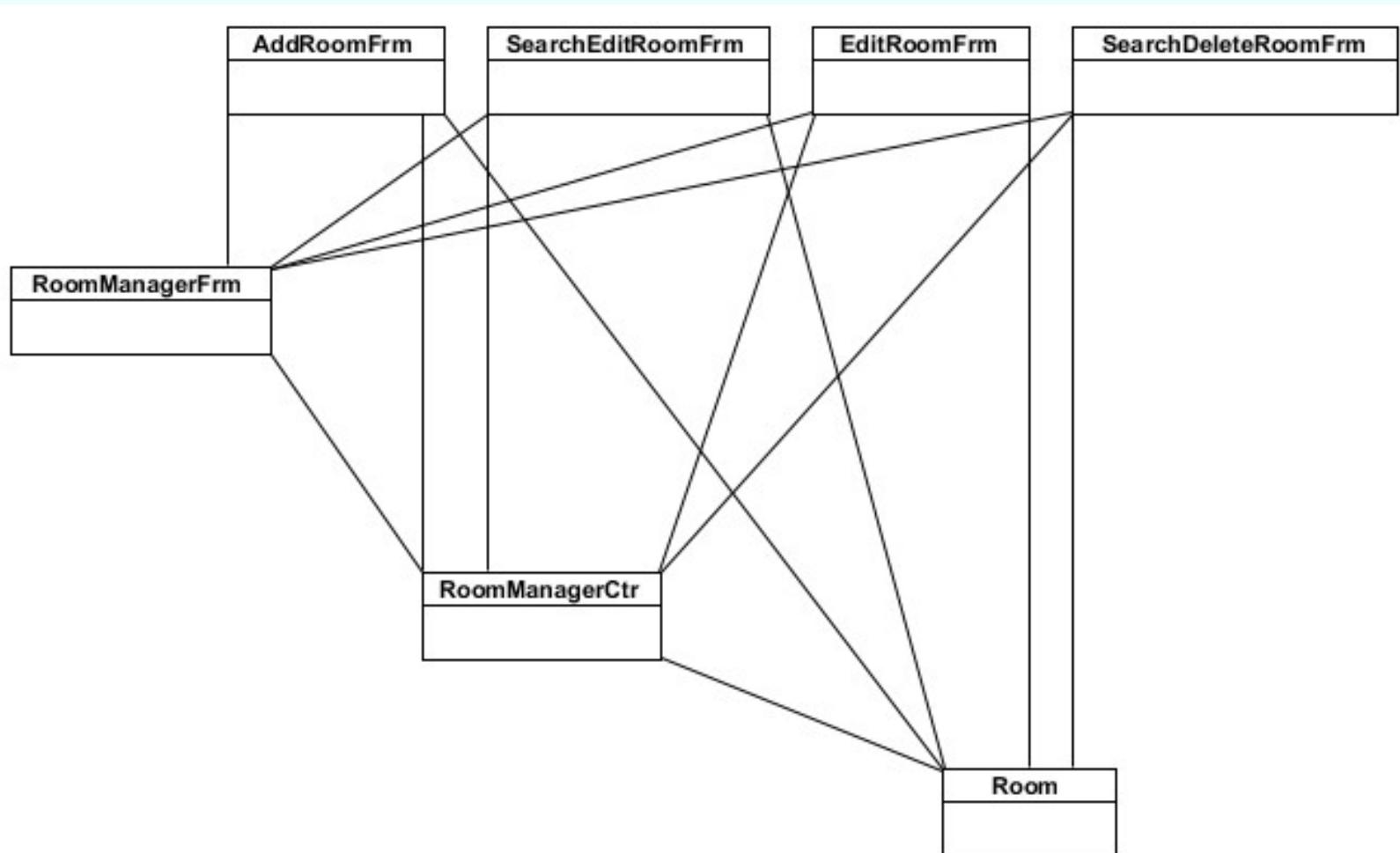
## Thẻ CRC cho lớp điều khiển (tt):

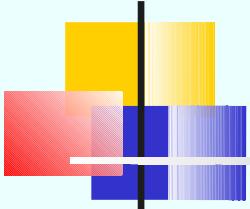
ManagerCtr	
Responsibilities:	
Name	Collaborator
Yêu cầu lớp RoomManagerFrm hiện giao diện quản lý phòng	RoomManagerFrm
Yêu cầu lớp AddRoomFrm hiện giao diện thêm phòng	AddRoomFrm
Yêu cầu lớp SearchEditRoomFrm hiện giao diện tìm kiếm phòng để sửa	SearchEditRoomFrm
Yêu cầu lớp SearchEditRoomFrm hiện giao diện kết quả tìm kiếm phòng để sửa	SearchEditRoomFrm
Yêu cầu lớp EditRoomFrm hiện giao diện sửa phòng	EditRoomFrm
Yêu cầu lớp SearchDeleteRoomFrm hiện giao diện tìm kiếm phòng để xóa	SearchDeleteRoomFrm
Yêu cầu lớp SearchDeleteRoomFrm hiện giao diện kết quả tìm kiếm phòng để xóa	SearchDeleteRoomFrm
Lưu thông tin phòng mới vào CSDL	
Tìm kiếm phòng trong CSDL theo tên	
Cập nhật thông tin một phòng vừa sửa vào CSDL	
Xóa thông tin một phòng khỏi CSDL	
yêu cầu lớp Room đóng gói dữ liệu vào đối tượng Room	Room



# Mô hình hóa các lớp (6)

Sơ đồ lớp cho modul quản lý phòng:





# Mô hình hoạt động (1)

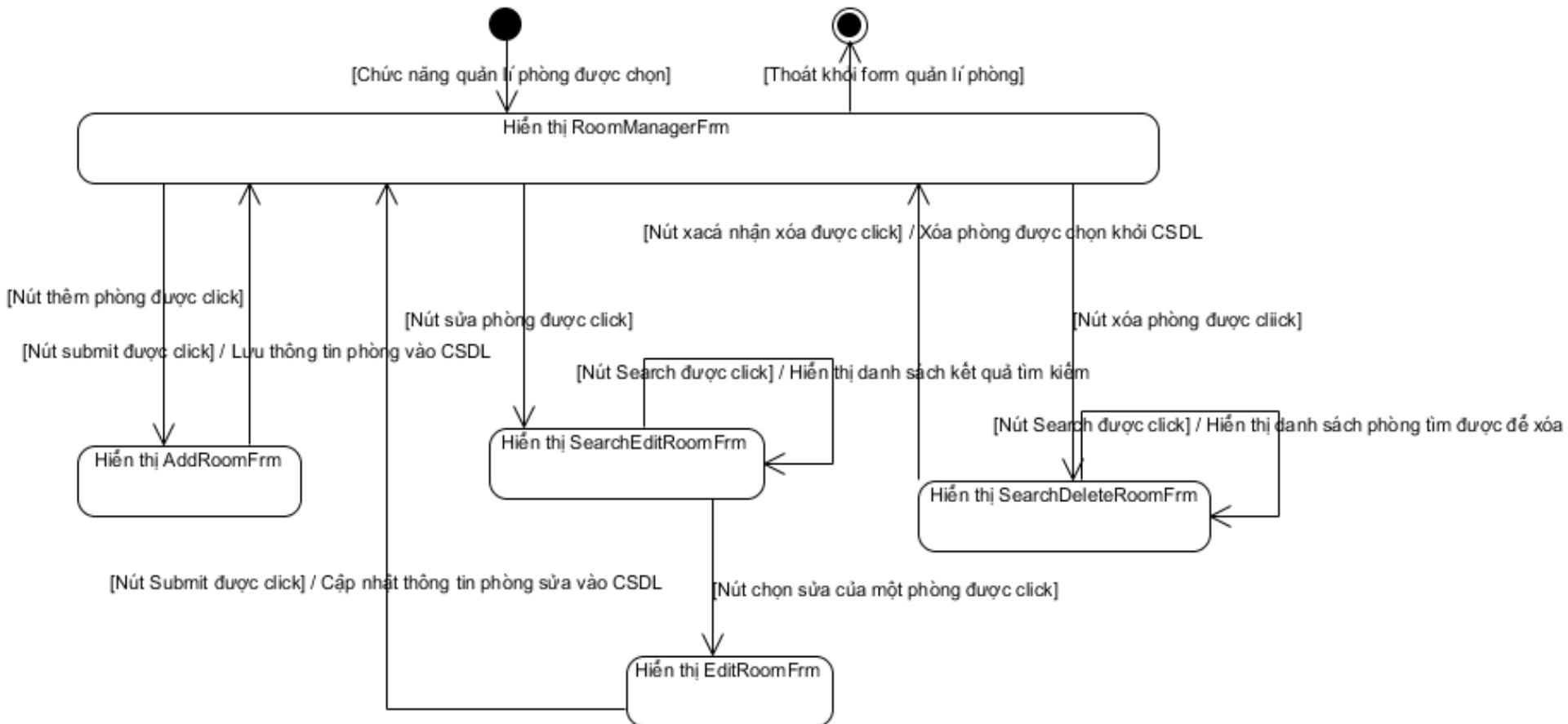
---

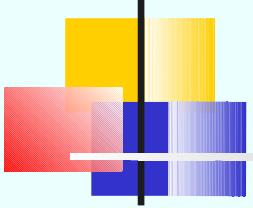
Sử dụng sơ đồ statechart:

- Mỗi trạng thái hệ thống được mô tả bằng một hình chữ nhật
- Khi có một sự kiện (event) xảy ra, thì trạng thái này sẽ chuyển sang trạng thái kia. Chuyển trạng thái biểu diễn bằng một mũi tên, nhãn là tên của sự kiện
- Các sự kiện và hoạt động tương tác được trích ra từ các scenario
- Trong VP: chọn new state machine diagram

# Mô hình hoạt động (2)

## Modul quản lý phòng:



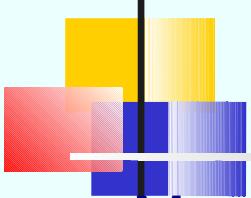


# Scenario cuối pha phân tích (1)

---

Trong scenario này:

- Các chủ thể hành động nằm trong phần mềm được thay thế bằng tên các lớp đã trích được
- Tương tác giữa các chủ thể chuyển thành hành động gửi thông điệp yêu cầu thực hiện hành động nào đó

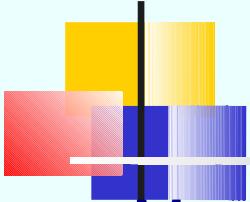


# Scenario cuối pha phân tích (2)

---

Manage room: scenario chuẩn cho thêm phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn thêm thông tin một phòng mới.
2. Lớp RoomManagerFrm hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút thêm phòng.
4. Lớp RoomManagerFrm gọi lớp AddRoomFrm yêu cầu hiển thị
5. Lớp AddRoomFrm hiện ra với các ô nhập: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 2 nút: nút thêm phòng, và nút hủy bỏ.
6. Nhân viên A nhập các thông tin phòng mới vào các ô và click nút thêm phòng
7. Lớp AddRoomFrm gọi lớp Room để đóng gói thông tin trên form thành một đối tượng kiểu Room



# Scenario cuối pha phân tích (3)

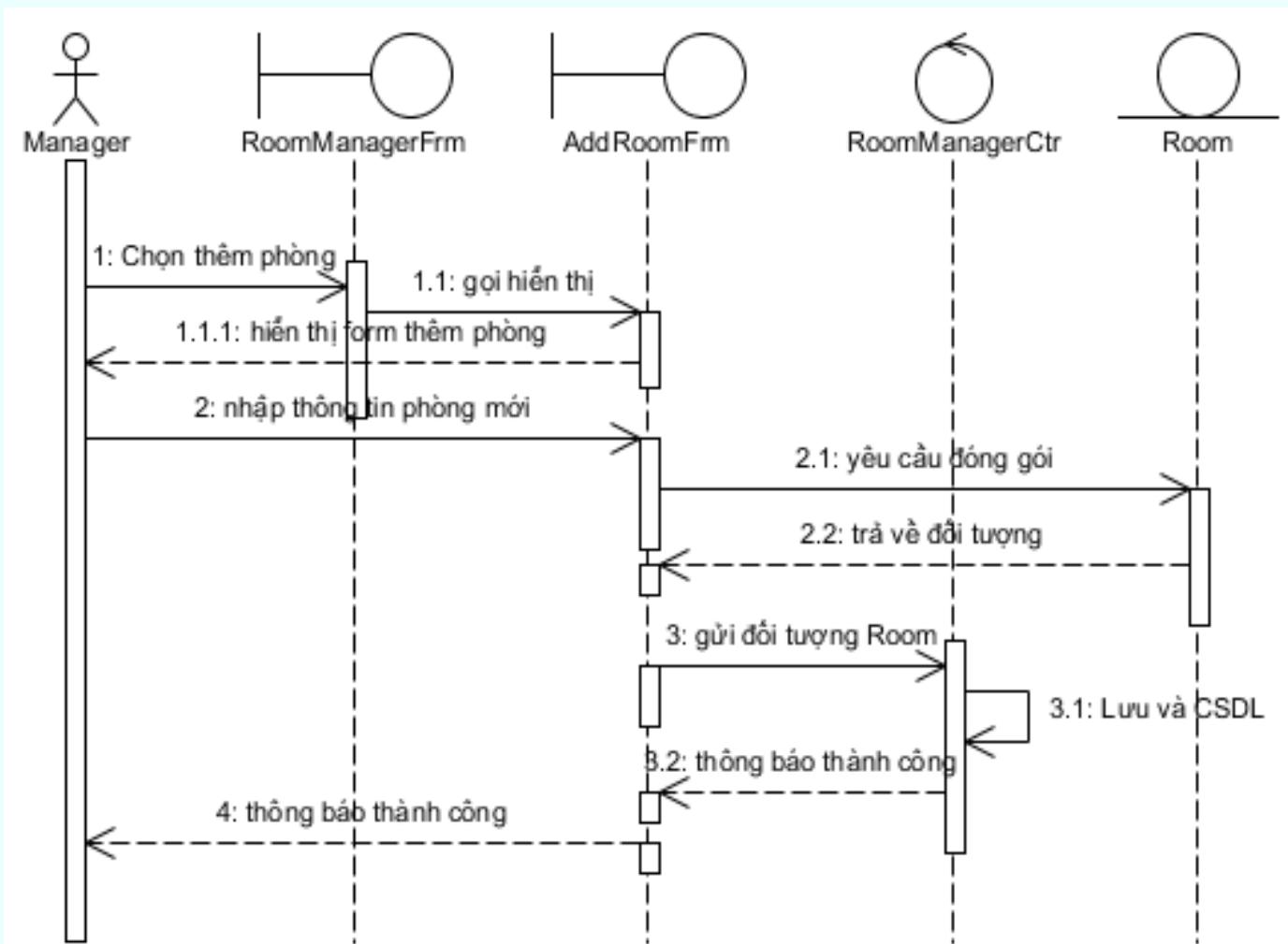
---

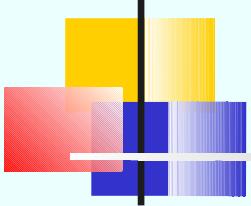
Manage room: scenario chuẩn cho thêm phòng (tt)

8. Lớp Room đóng gói thông tin và trả lại cho lớp AddRoomFrm một đối tượng kiểu Room
9. Lớp AddRoomFrm chuyển đổi từ đối tượng Room cho lớp RoomManagerCtr
10. Lớp RoomManagerCtr lưu thông tin phòng vào CSDL
11. Lớp RoomManagerCtr thông báo cho lớp AddRoomFrm đã thêm thành công
12. Lớp AddRoomFrm thông báo thêm phòng thành công.

# Scenario cuối pha phân tích (4)

Sơ đồ tuần tự cho scenario chuẩn cho thêm phòng



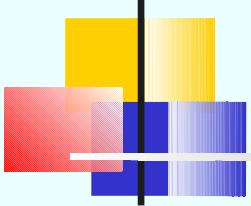


# Scenario cuối pha phân tích (5)

---

Manage room: scenario chuẩn cho sửa phòng

1. Nhân viên quản lí A chọn chức năng quản lí phòng sau khi login. A muốn sửa thông tin phòng 305.
2. Lớp RoomManagerFrm hiện ra với 3 nút: thêm, sửa, xóa phòng
3. Nhân viên A click vào nút sửa phòng.
4. Lớp RoomManagerFrm gọi lớp SearchEditRoomFrm hiển thị
5. Lớp SearchEditRoomFrm hiện ra với một ô nhập tên phòng và một nút tìm kiếm
6. A nhập 305 vào ô tên phòng và click vào nút tìm kiếm
7. Lớp SearchEditRoomFrm gửi thông tin tên phòng 305 cho lớp RoomMaanagerCtr

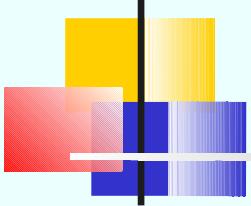


# Scenario cuối pha phân tích (6)

---

Manage room: scenario chuẩn cho sửa phòng (tt)

8. Lớp RoomManagerCtr tìm kiếm các phòng có tên 305 trong CSDL
9. Lớp RoomManagerCtr gửi kết quả đến lớp Room để đóng gói thành danh sách các đối tượng Room
10. Lớp Room gửi trả cho lớp RoomManagerCtr danh sách các đối tượng Room
11. Lớp RoomManagerCtr gửi danh sách các đối tượng Room cho lớp SearchEditRoomFrm để hiển thị
12. Lớp SearchEditRoomFrm hiện kết quả tìm kiếm gồm một bảng các phòng có tên 305, mỗi dòng có đầy đủ thông tin một phòng với các cột: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, và 1 nút chọn sửa.



# Scenario cuối pha phân tích (7)

---

Manage room: scenario chuẩn cho sửa phòng (tt)

13. Nhân viên A click chọn sửa vào phòng thứ 2 trong danh sách

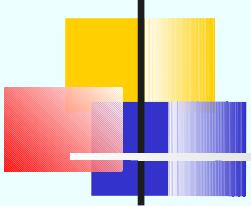
14. Lớp SearchEditRoomFrm gọi và truyền đối tượng vừa được chọn cho lớp EditRoomFrm

15. Lớp EditRoomFrm hiện ra với đầy đủ thông tin của đối tượng được chọn sửa: id phòng, tên phòng, kiểu phòng, giá hiển thị, mô tả, 1 nút hủy bỏ và một nút sửa.

16. Nhân viên A sửa một số thông tin về loại phòng, mô tả và giá của phòng và click vào nút sửa.

17. Lớp EditRoomFrm gửi thông tin trên form đến lớp Room để đóng gói đối tượng Room

18. Lớp Room đóng gói thông tin thành một đối tượng Room



# Scenario cuối pha phân tích (8)

---

Manage room: scenario chuẩn cho sửa phòng (tt)

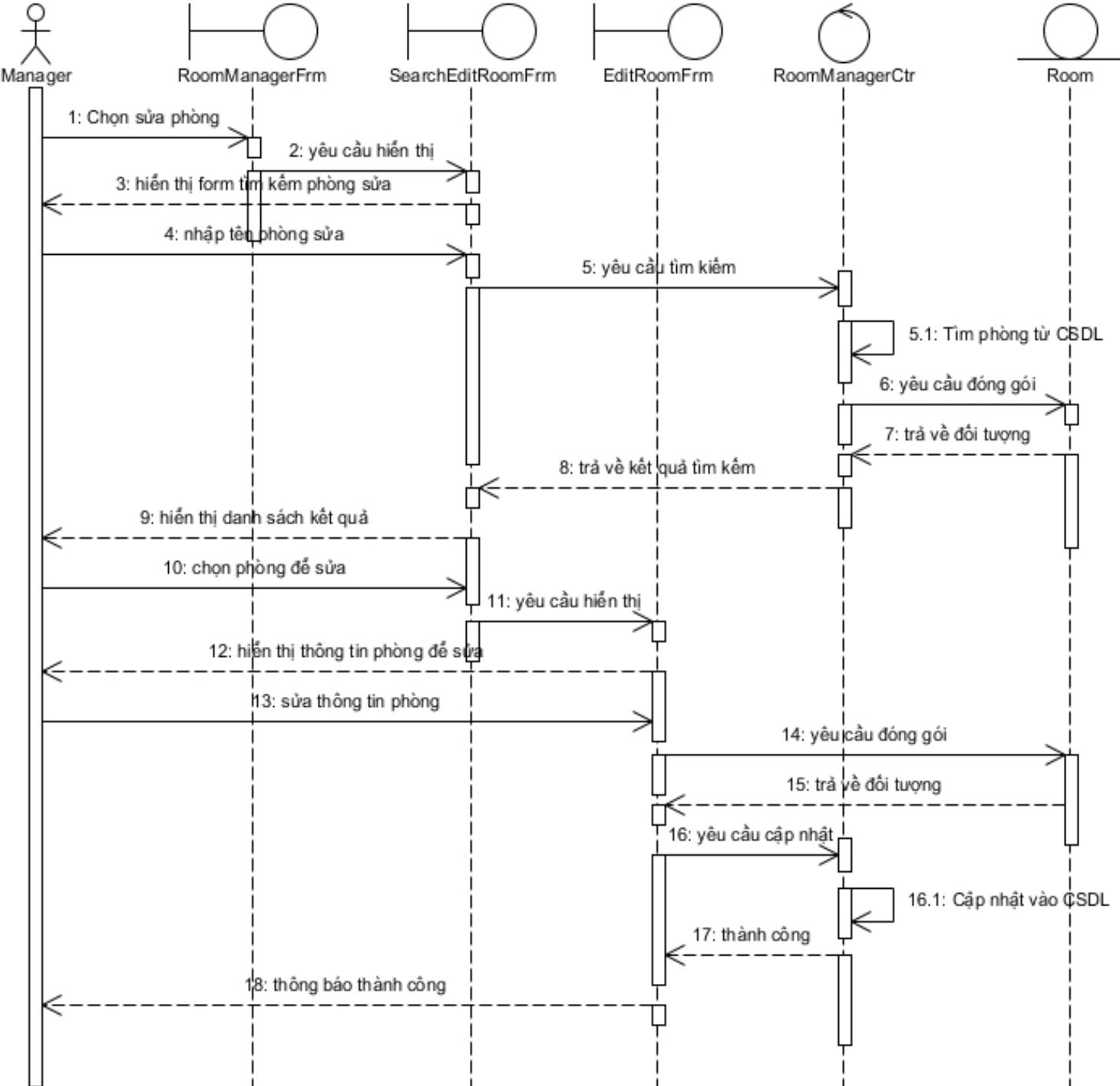
19. Lớp EditRoomFrm truyền đối tượng Room cho lớp RoomManagerCtr

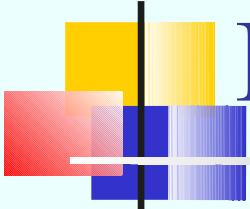
20. Lớp RoomManagerCtr cập nhật thông tin phòng vào CSDL.

21. Lớp RoomManagerCtr thông báo cho lớp EditRoomFrm đã cập nhật thành công

22. Lớp EditRoomFrm thông báo cập nhật thành công cho nhân viên A

# Sơ đồ tuần tự cho scenario chuẩn cho sửa phòng



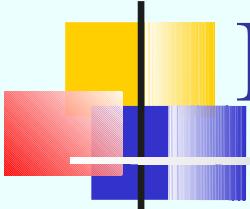


# Bài tập trên lớp

---

Thực hiện các bước sau lần lượt với các modul của Admin, của Seller, và của Receptionist:

- Trích các lớp biên, các lớp điều khiển
- Viết thẻ CRC cho lớp điều khiển
- Vẽ sơ đồ trạng thái cho modul
- Vẽ lại sơ đồ lớp cho modul
- Viết mỗi UC với một scenario phiên bản 2
- Thực tế hóa mỗi scenario này của mỗi UC thành sơ đồ tuần tự

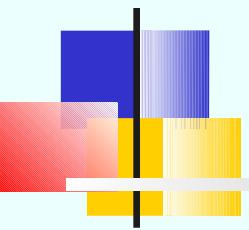


# Bài tập về nhà

---

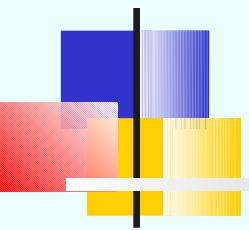
Với mỗi modul của cá nhân:

- Vẽ lại sơ đồ chi tiết các UC của modul cá nhân
- Với mỗi UC, trích các scenario chuẩn và các ngoại lệ tương ứng ( không cần xử lí các ngoại lệ sai kiểu dữ liệu đầu vào)
- Trích các lớp thực thể, trích các lớp biên, các lớp điều khiển. Vẽ sơ đồ lớp từ các lớp đã trích được.
- Xây dựng thẻ CRC cho các lớp điều khiển
- Xây dựng sơ đồ hoạt động (statechart) cho modul
- Viết lại các scenario với các lớp đã trích được
- Thực tế hóa mỗi scenario của mỗi UC thành sơ đồ tuần tự (hoặc cộng tác)



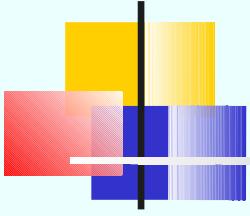
# Questions?

---



# Công nghệ phần mềm Pha thiết kế

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

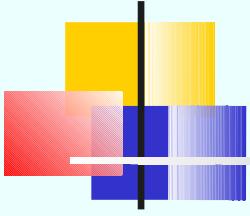


# Pha thiết kế (1)

---

## Mục đích:

- Chuyển tài liệu phân tích dù dạng đặc tả nghiệp vụ hệ thống, sang dạng có thể cài đặt và kiểm thử được

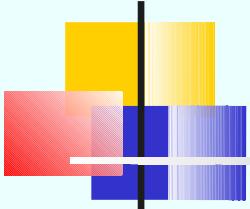


# Pha thiết kế (2)

---

Thực hiện:

- B1: Thiết kế CSDL (nếu có)
- B2: Hoàn thiện sơ đồ lớp có được trong pha phân tích → sơ đồ lớp chi tiết
- B3: Thiết kế chi tiết hoạt động bên trong của các lớp, các phương thức của lớp

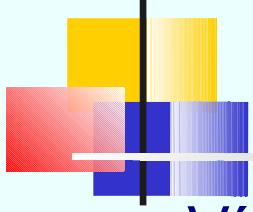


# Thiết kế CSDL (1)

---

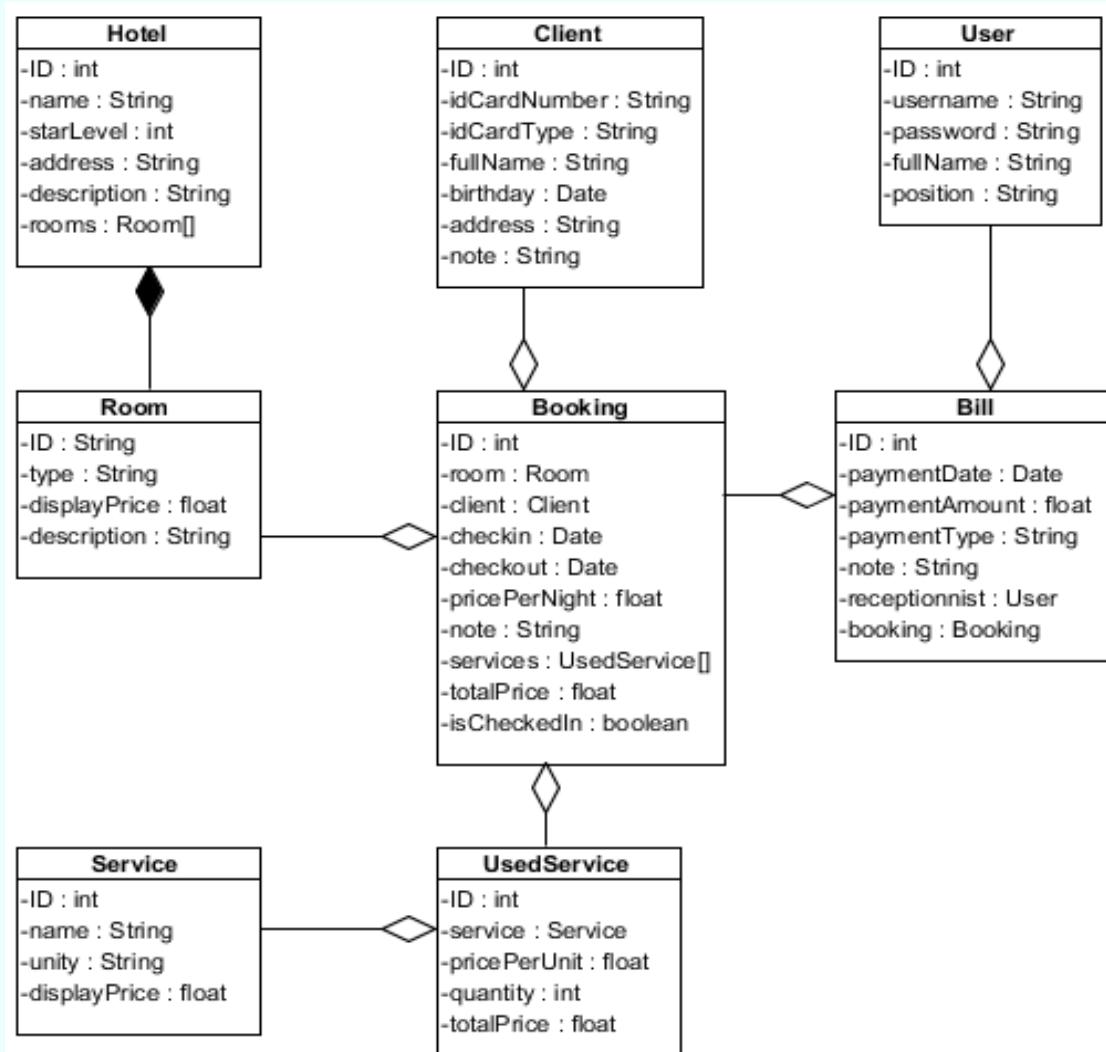
Xây dựng CSDL từ sơ đồ lớp thực thể của hệ thống:

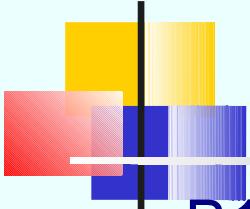
- B1: Mỗi lớp thực thể đề xuất thành một bảng tương ứng trong CSDL
- B2: Với mỗi lớp thực thể, lấy các thuộc tính kiểu cơ bản (không phải kiểu lớp thực thể khác) làm thuộc tính cho bảng tương ứng với lớp thực thể đó.
- B3: Giữa hai lớp có quan hệ thành phần, liên kết, hợp thì giữa hai bảng tương ứng phải có quan hệ n-n, 1-n hoặc 1-1, tùy từng trường hợp.
- B4: Định nghĩa khóa chính và khóa ngoài tương ứng với các quan hệ giữa các bảng
- B5: Gộp bảng nếu có quan hệ 1-1, tách bảng nếu có quan hệ n-n.



# Thiết kế CSDL (2)

Ví dụ với phần mềm quản lý đặt phòng khách sạn:



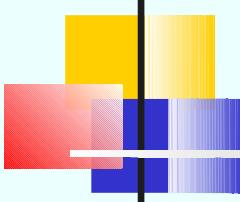


# Thiết kế CSDL (3)

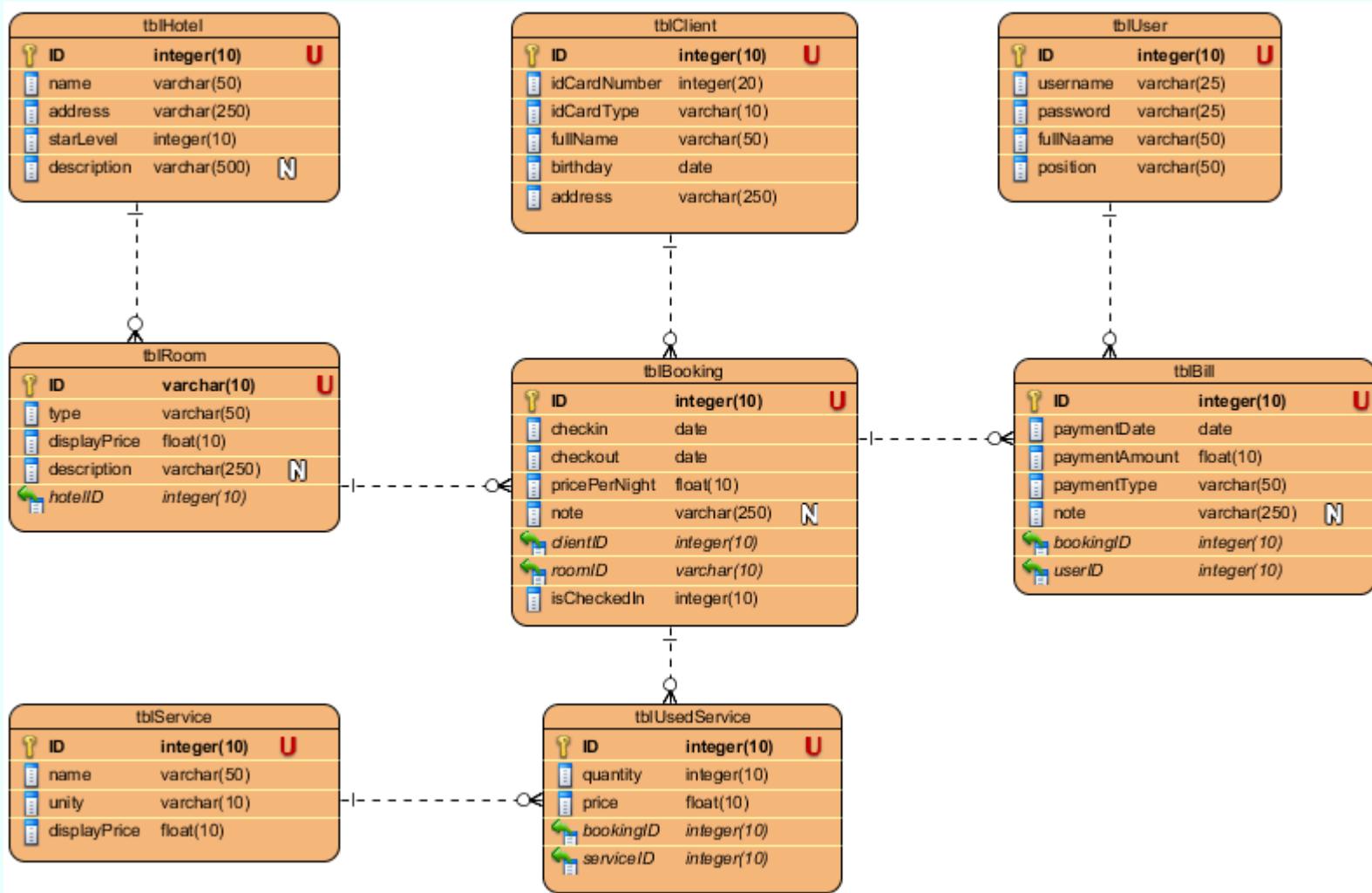
---

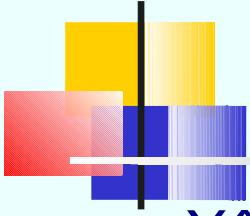
B1: Mỗi lớp thực thể đề xuất thành một bảng tương ứng trong CSDL:

- Lớp Hotel → bảng tblHotel
- Lớp Room → bảng tblRoom
- Lớp Client → bảng tblClient
- Lớp User → bảng tblUser
- Lớp Service → bảng tblService
- Lớp UsedService → bảng tblUsedService
- Lớp Booking → bảng tblBooking
- Lớp Bill → bảng tblBill



# Thiết kế CSDL (4)



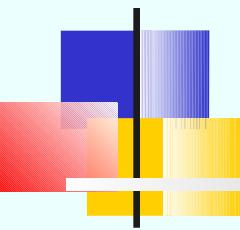


# Bài tập (1)

---

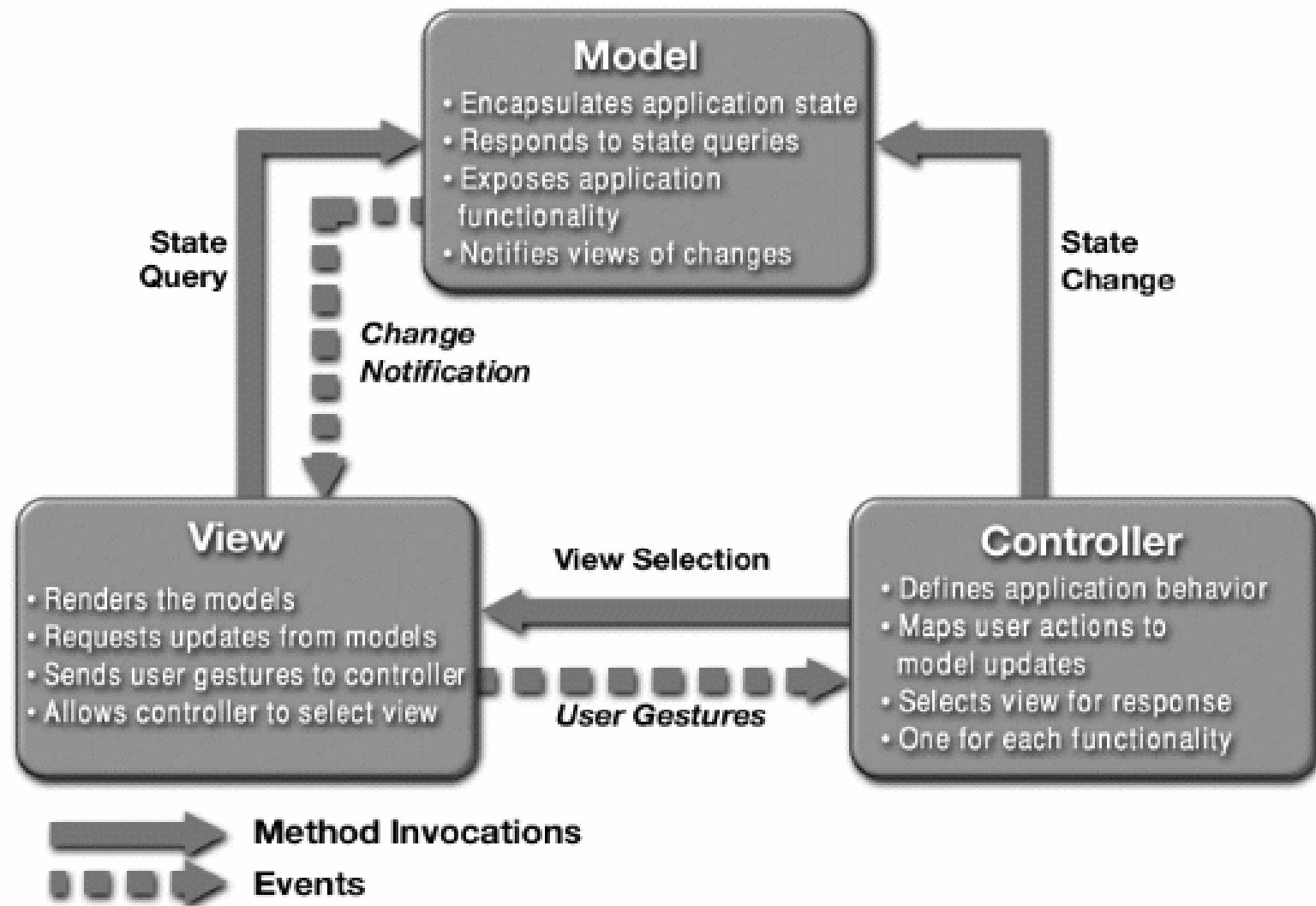
Xây dựng CSDL cho bài tập nhóm:

- Trình bày lại (đã chỉnh sửa theo kết quả phâ phân tích) sơ đồ lớp thực thể của toàn hệ thống
- Trình bày sơ đồ quan hệ giữa các bảng trong CSDL sau khi áp dụng các bước chuyển đổi trong bài
- Cả nhóm nộp chung

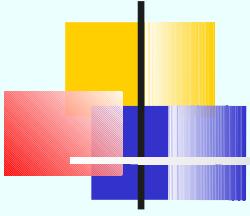


# **Thiết kế hệ thống theo mô hình MVC**

# Mô hình MVC (1)



[image source: <http://www.oracle.com/technetwork/>]

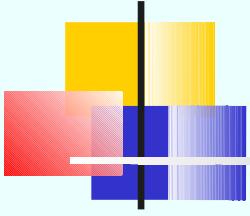


# Mô hình MVC (2)

---

M - model:

- Đóng gói dữ liệu, thông tin
- Chức năng biểu diễn, vận chuyển thông tin để trình diễn (view) và xử lí (control)

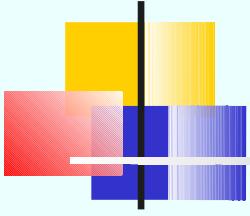


# Mô hình MVC (3)

---

C - control:

- Định nghĩa các hành vi, hoạt động, xử lí của hệ thống
- Đối chiếu hành động của user (nhận từ view), vào tập chức năng để xử lí, đồng thời chọn hành động đưa view ra để show

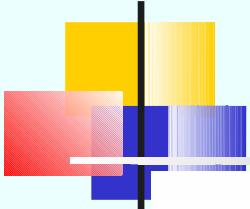


# Mô hình MVC (4)

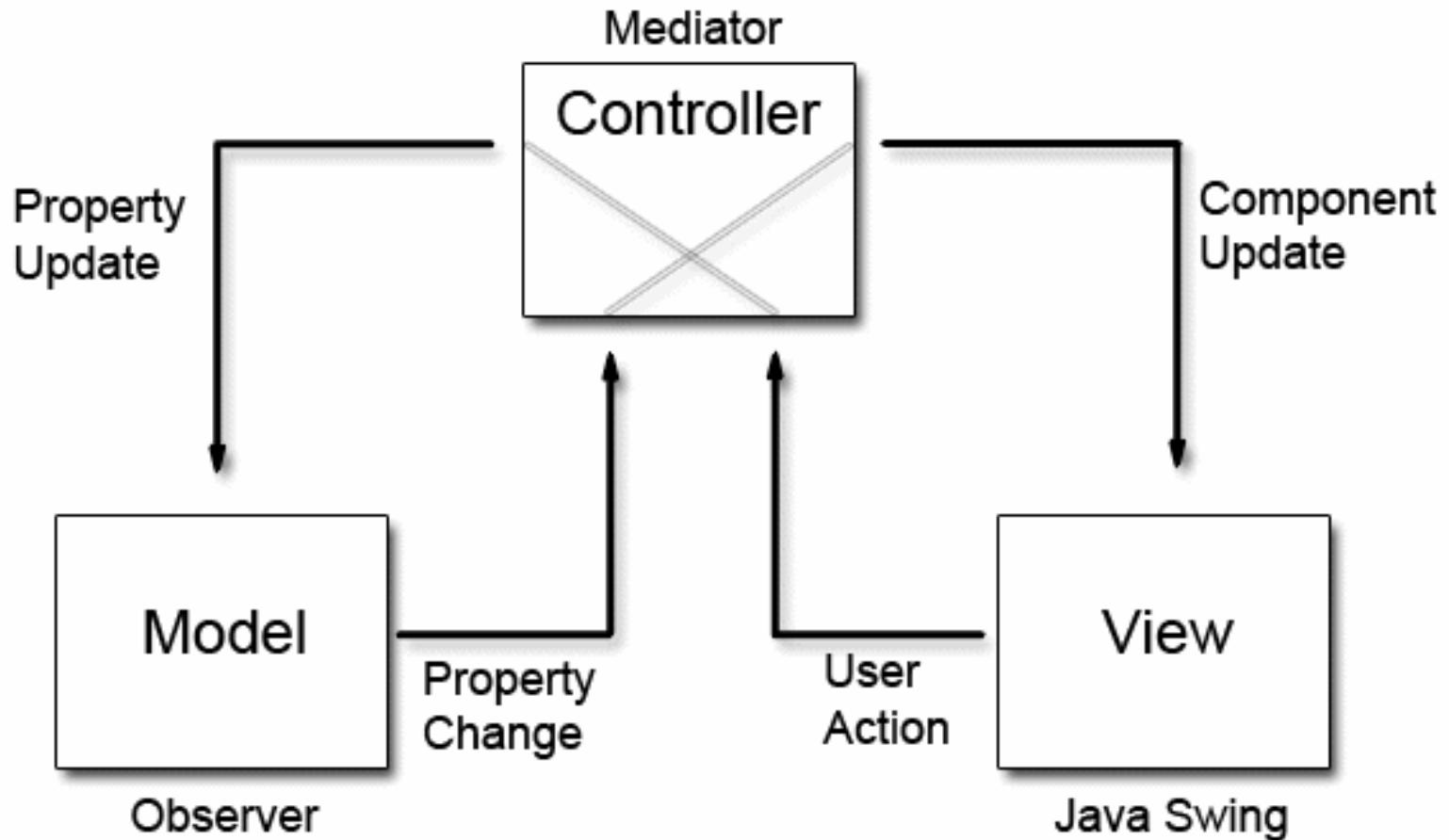
---

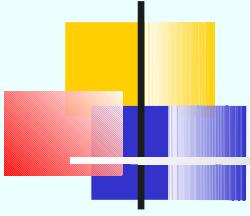
V - view:

- Giao diện với người dùng
- Show các kết quả xử lí của tầng control
- Thu nhận các hoạt động, yêu cầu của người sử dụng và chuyển cho tầng control xử lí

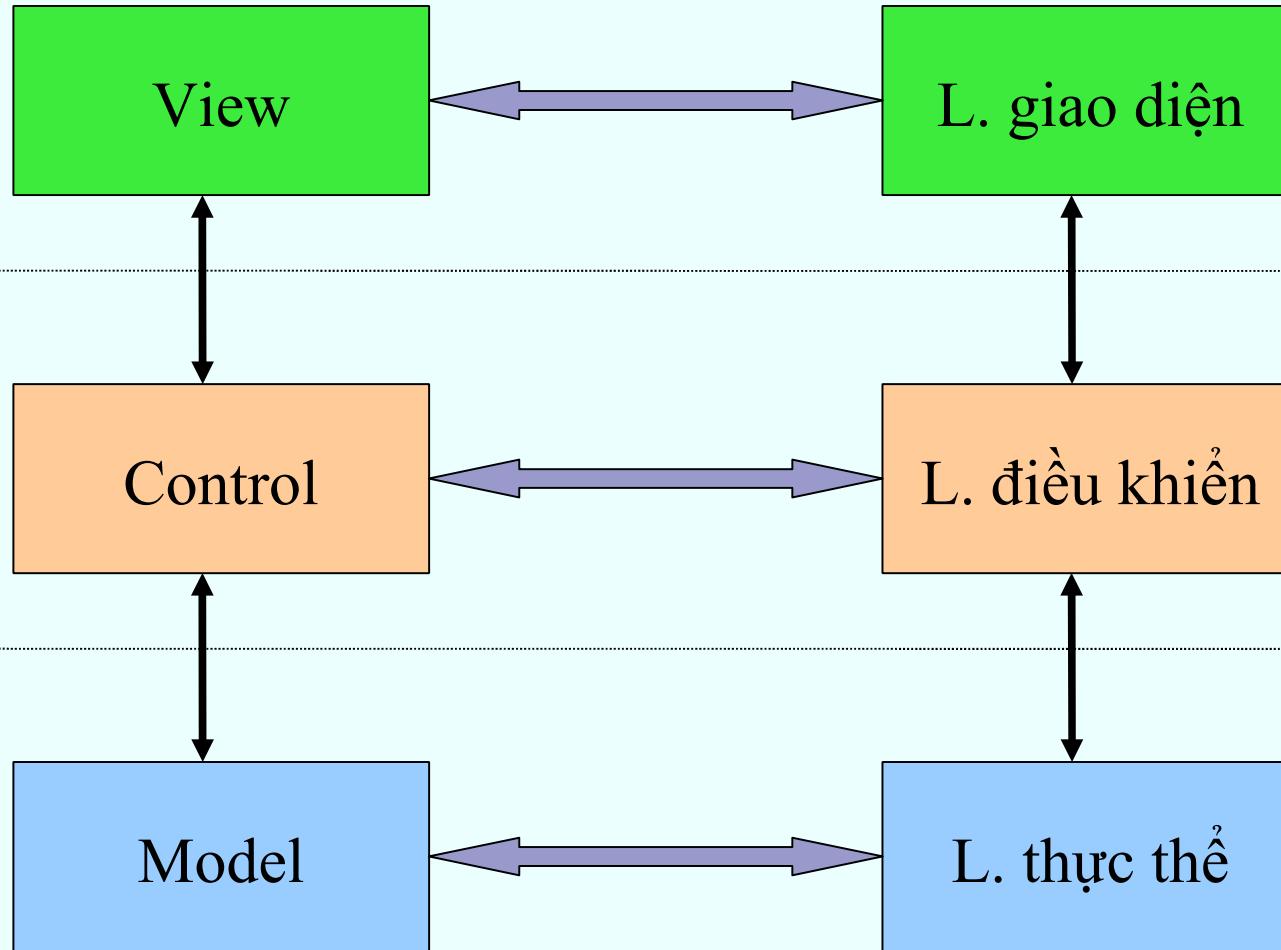


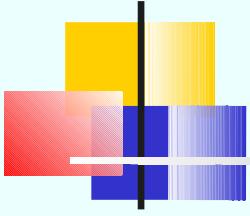
# MVC cải tiến (1)





# MVC cải tiến (2)

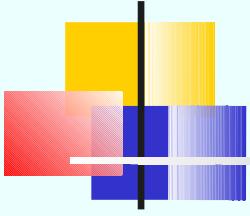




# Các lớp thực thể

---

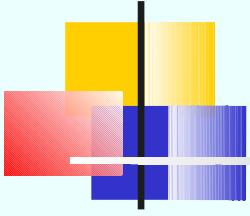
- Đóng gói dữ liệu, thông tin
- Chỉ chứa các thuộc tính và các phương thức truy cập các thuộc tính (javaBean)
- Chức năng biểu diễn, vận chuyển thông tin để trình diễn (view) và xử lí (control)



# Các lớp điều khiển

---

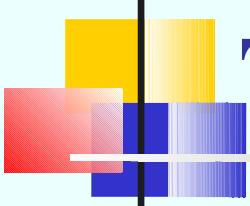
- Cập nhật thông tin vào DB (thông tin chưa trong các thực thể)
- Thực hiện các tính toán, xử lý trung gian
- Đổi chiều hành động của user (nhận từ view), vào tập chức năng để xử lý, đồng thời chọn hành động đưa view ra để show



# Các lớp giao diện

---

- Các frame, cửa sổ của ứng dụng (javaSwing)
- Các trang giao diện web: html, jsp
- Các bảng, mẫu biểu, báo cáo in ra



# Thiết kế hệ thống theo MVC

---

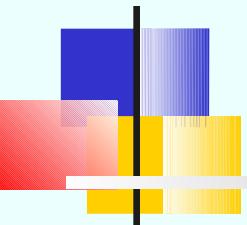
Có thể áp dụng một số dạng mô hình MVC phổ biến:

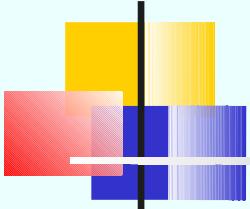
- Mô hình MVC dùng thực thể thuần (cỗ điển)
- Mô hình MVC dùng bean
- Mô hình MVC cải tiến (hiện đại)

Lưu ý:

Với bài tập lớn, các nhóm nên chọn thiết kế theo 1 trong 3 dạng kiến trúc trên.

# Thiết kế theo mô hình MVC với thực thể thuần



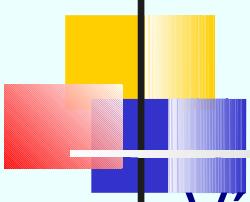


# MVC với thực thể thuần (1)

---

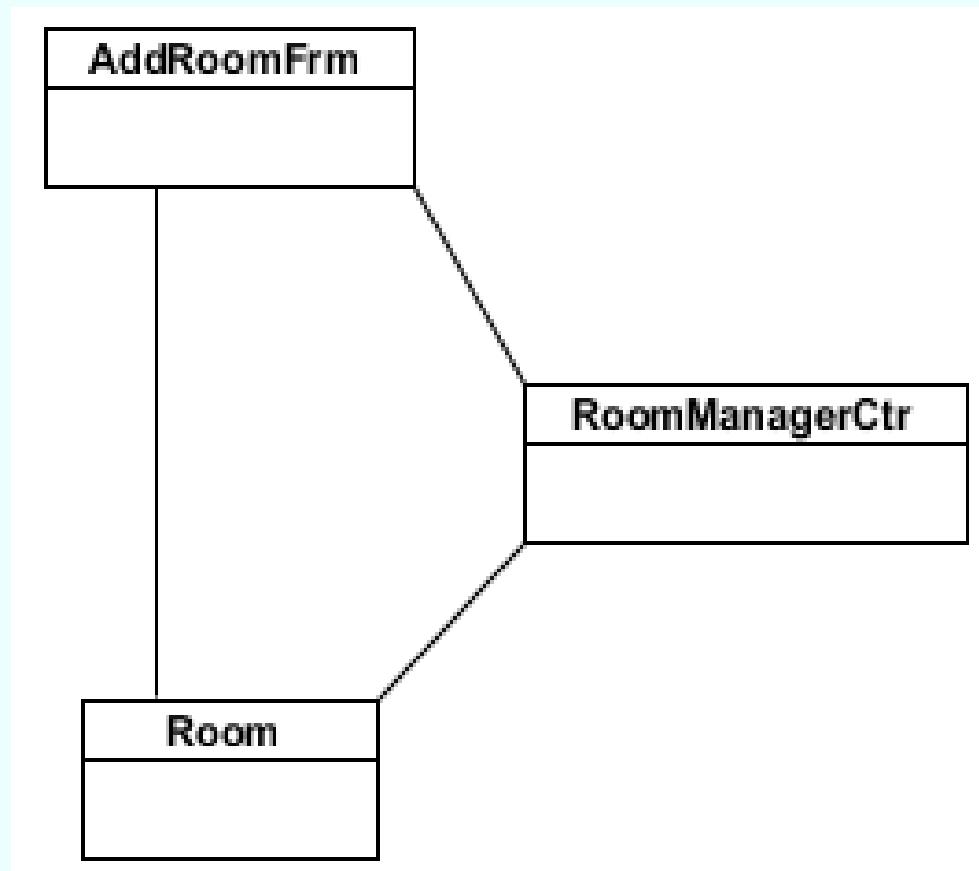
## Đặc trưng:

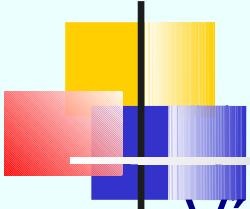
- Lớp thực thể chỉ chứa các thuộc tính và các phương thức get/set cho mỗi thuộc tính (còn gọi là các lớp thực thể thuần)
- Các thao tác liên quan đến CSDL đều đặt trong lớp điều khiển (dạng lớp DAO – Data Access Object)



# MVC với thực thể thuần (2)

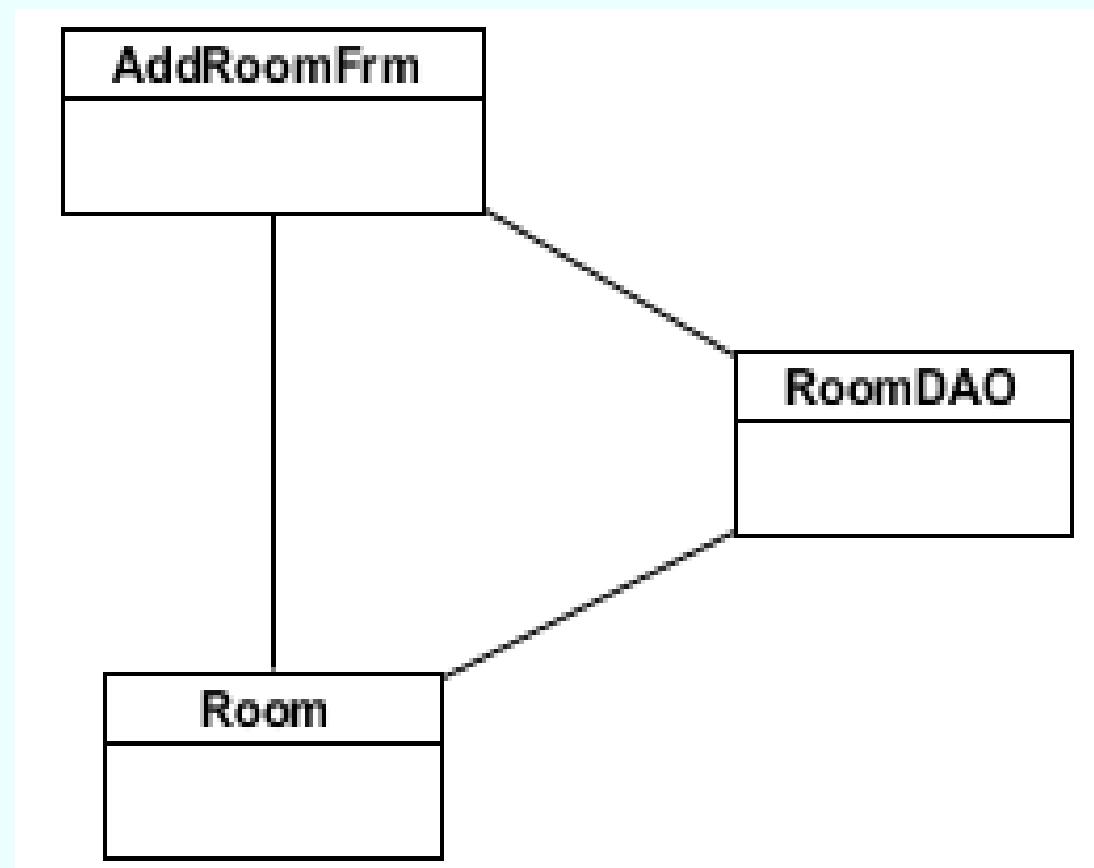
Ví dụ modul quản lí phòng của Manager, sơ đồ lớp cuối  
pha phân tích của chức năng thêm phòng:

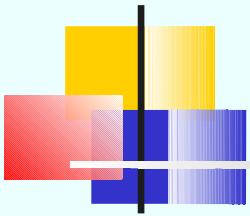




# MVC với thực thể thuần (3)

Ví dụ chức năng thêm phòng của Manager, sơ đồ lớp  
theo MVC dùng thực thể thuần:



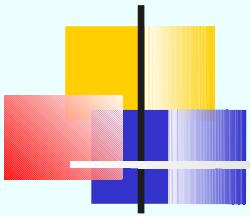


# Hoàn thiện sơ đồ lớp (1)

---

Thực hiện:

- Định nghĩa kiểu thuộc tính cho lớp
- Định nghĩa khuôn mẫu các phương thức cho lớp

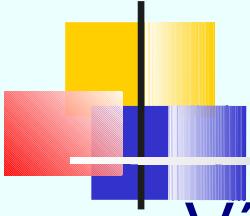


# Hoàn thiện sơ đồ lớp (2)

---

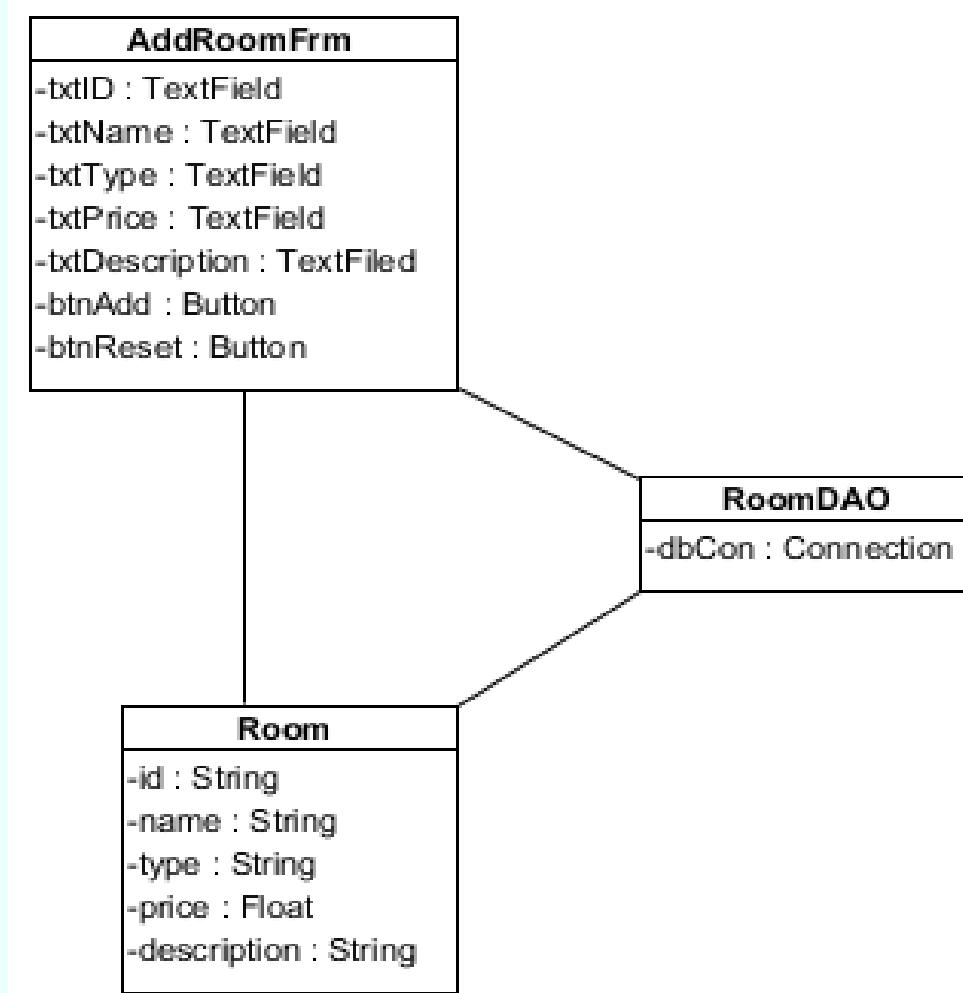
Định nghĩa kiểu thuộc tính cho lớp:

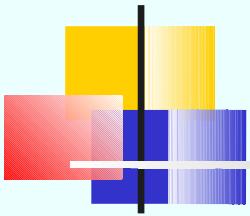
- Tên thuộc tính đã xác định trong pha phân tích
- Chọn kiểu dữ liệu cụ thể cho từng thuộc tính dựa vào giới hạn lưu trữ của thuộc tính
- Điện thuộc tính (tên:kiểu) vào sơ đồ lớp



# Hoàn thiện sơ đồ lớp (3)

Ví dụ với chức năng thêm phòng:



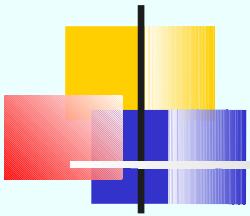


# Hoàn thiện sơ đồ lớp (4)

---

Định nghĩa khuôn mẫu phương thức cho lớp:

- Dùng thẻ CRC để xác định phương thức nào nên gán cho lớp nào
- Định nghĩa khuôn mẫu cho từng phương thức

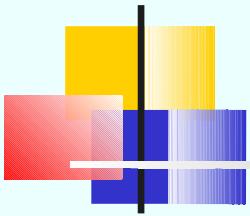


# Hoàn thiện sơ đồ lớp (5)

---

Gán phương thức cho lớp:

- **Nguyên lý A: Che giấu thông tin.** Các thuộc tính của lớp phải để dạng private → cần các phương thức **get/set** tương ứng cho phép các đối tượng khác truy nhập vào các thuộc tính này
- Áp dụng cho các lớp thực thể: các thuộc tính để private, mỗi thuộc tính có một cặp phương thức get/set tương ứng



# Hoàn thiện sơ đồ lớp (6)

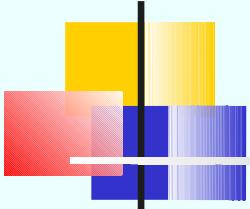
---

Ví dụ với lớp thực thể Room:

- Các thuộc tính để chế độ private (có dấu “-” đầu tên thuộc tính)
- Mỗi thuộc tính có một cặp phương thức get/set tương ứng: ví dụ thuộc tính name có các phương thức:

public void setName(String name)

public String getName()

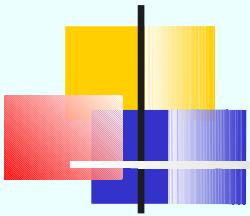


# Hoàn thiện sơ đồ lớp (7)

---

Gán phương thức cho lớp (tt):

- **Nguyên lí B:** Nếu có nhiều đối tượng **X** gọi đến một hành động **k** của đối tượng **Y**, thì phương thức để thực hiện hành động **k** nên gán cho lớp của đối tượng **Y**, mà không nên gán cho lớp của đối tượng **X**

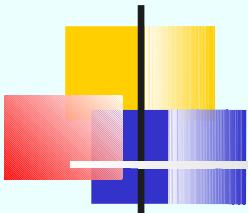


# Hoàn thiện sơ đồ lớp (8)

---

Gán phương thức cho lớp (tt):

- **Nguyên lí C:** Thiết kế hướng trách nhiệm. Nếu một hành động mà không thể gán thành phương thức cho lớp khác, thì lớp của đối tượng cần thực hiện hành động đó phải chứa phương thức tương ứng hành động đó



# Hoàn thiện sơ đồ lớp (9)

Định nghĩa khuôn mẫu các phương thức:

- Kiểu dữ liệu trả về
- Số lượng, thứ tự và kiểu dữ liệu truyền vào

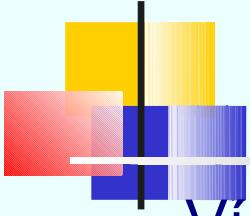
Ví dụ:

Không nên định nghĩa phương thức lưu thông tin một **Room** với các tham số là các thuộc tính:

```
public void saveRoom(int id, string name...)
```

Mà nên truyền vào là một kiểu đối tượng cửa lớp **Room** đã đóng gói:

```
public void saveRoom(Room room)
```

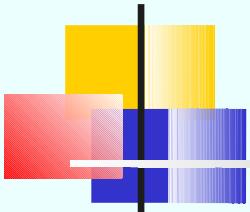


# Hoàn thiện sơ đồ lớp (10)

Ví dụ thẻ CRC của lớp RoomDAO và AddRoomFrm cho chức năng thêm phòng:

RoomDAO	
Responsibilities:	
Name	Collaborator
Yêu cầu lớp AddRoomFrm hiển thị giao diện nhập thông tin phòng	AddRoomFrm
Lưu thông tin phòng vào CSDL	
Yêu cầu lớp AddRoomFrm hiển thị thông báo thêm thành công	AddRoomFrm

AddRoomFrm	
Responsibilities:	
Name	Collaborator
Hiển thị form nhập thông tin phòng	
Yêu cầu lớp Room đóng gói thông tin nộp trên form thành đối tượng	Room
Yêu cầu lớp RoomDAO lưu thông tin phòng vào CSDL	RoomDAO
Xử lý sự kiện khi nút Add bị click	
Xử lý sự kiện khi nút Reset bị click	

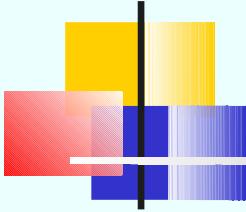


# Hoàn thiện sơ đồ lớp (11)

---

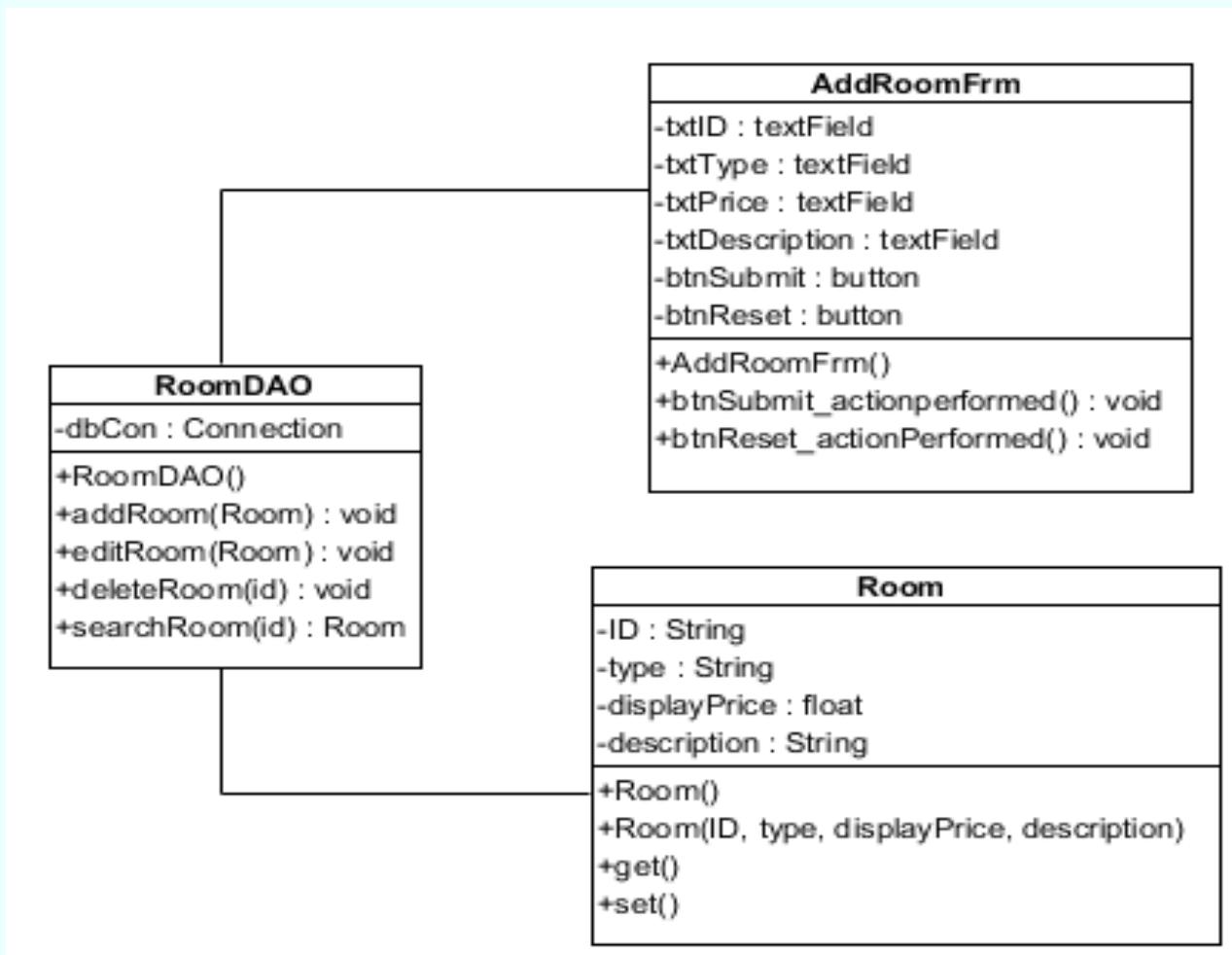
Áp dụng nguyên lí B và C:

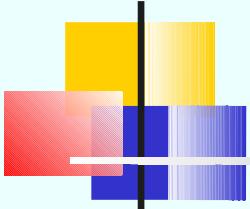
- Lớp AddRoomFrm phải có các phương thức: hiển thị form (hàm khởi tạo), xử lý sự kiện nút Add và nút Reset bị click, thông báo thành công
- Lớp RoomDAO phải có phương thức lưu thông tin phòng vào CSDL
- Lớp Room phải có các phương thức đóng gói thông tin đối tượng (hàm khởi tạo hoặc các phương thức set)



# Hoàn thiện sơ đồ lớp (12)

Kết quả thu được sơ đồ lớp như sau:



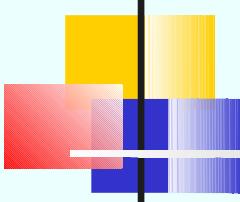


# Sơ đồ tuần tự pha thiết kế (1)

---

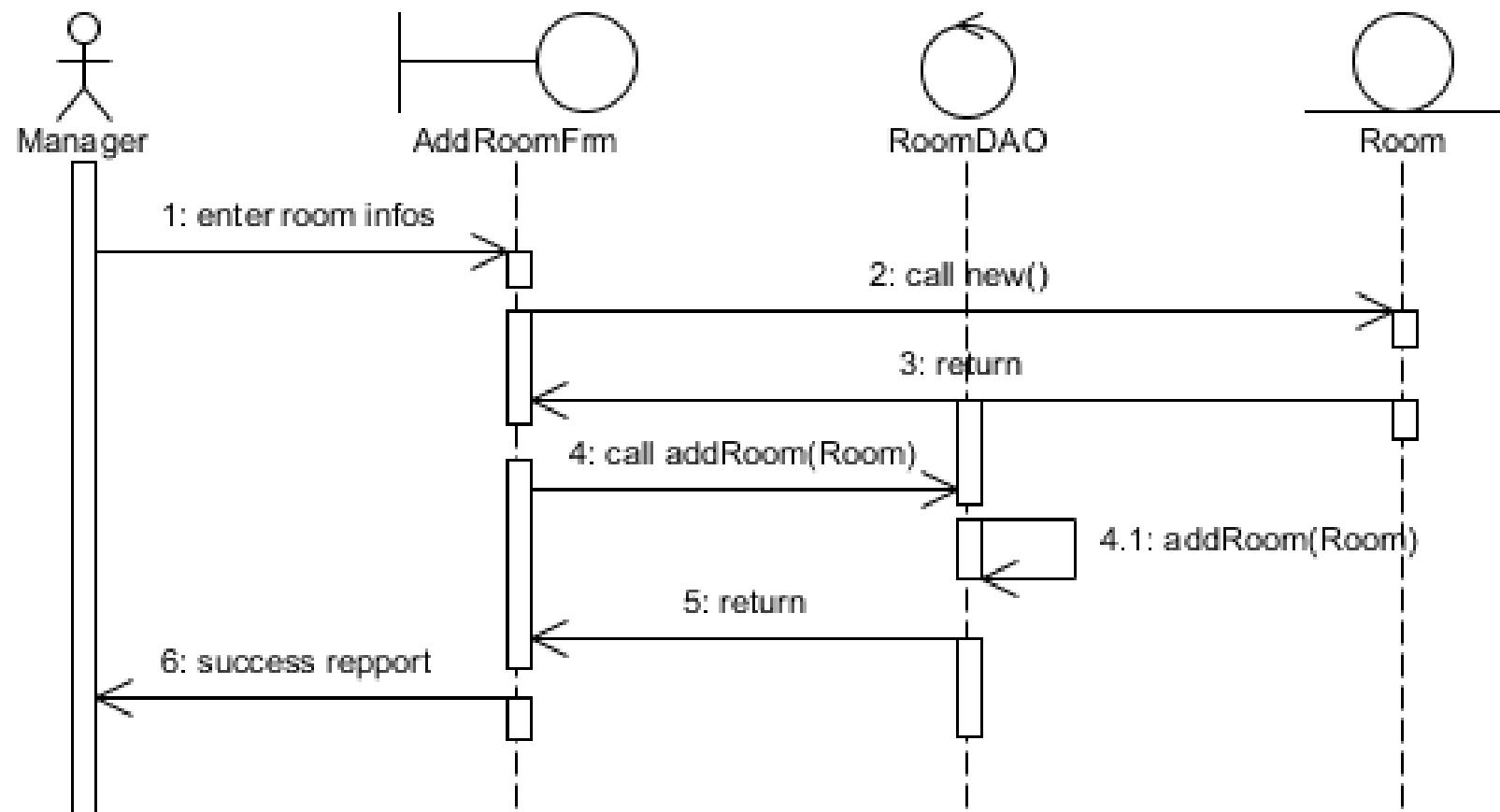
Đặc trưng:

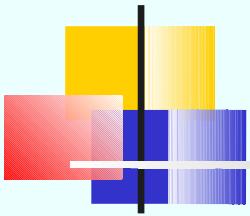
- Tên các lớp theo đúng sơ đồ lớp đã thiết kế
- Nhãn các mũi tên phải là tên các phương thức của các lớp trong sơ đồ lớp đã thiết kế



# Sơ đồ tuần tự pha thiết kế (2)

Ví dụ chức năng thêm phòng của Manager:



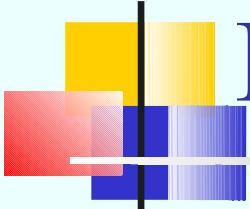


# Bài tập

---

Thiết kế tương tự cho các chức năng và modul:

- Sửa thông tin phòng
- Xóa thông tin phòng
- Khách hàng đặt chỗ tại quầy với nhân viên tiếp tân
- Khách hàng checkin tại quầy với nhân viên tiếp tân
- Khách hàng trả phòng và thanh toán tại quầy với nhân viên tiếp tân



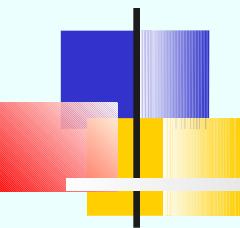
# Bài tập về nhà

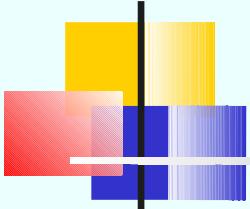
---

Với mỗi modul cá nhân:

- Vẽ lại sơ đồ UC chi tiết của hệ thống và của modul
- Vẽ lại sơ đồ các lớp sau pha phân tích
- Định nghĩa các thuộc tính và kiểu thuộc tính của mỗi lớp
- Dùng kĩ thuật thẻ CRC và 3 nguyên lí thiết kế phương thức để gán các phương thức cho các lớp
- Định nghĩa khuôn mẫu cho từng phương thức
- Điền tất cả vào sơ đồ lớp để thu được sơ đồ lớp chi tiết (theo mô hình MVC dùng thực thể thuần)
- Vẽ lại sơ đồ tuần tự sau pha thiết kế

# Thiết kế theo mô hình MVC với thực thể bean



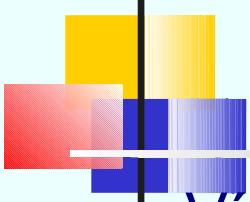


# MVC với thực thể bean (1)

---

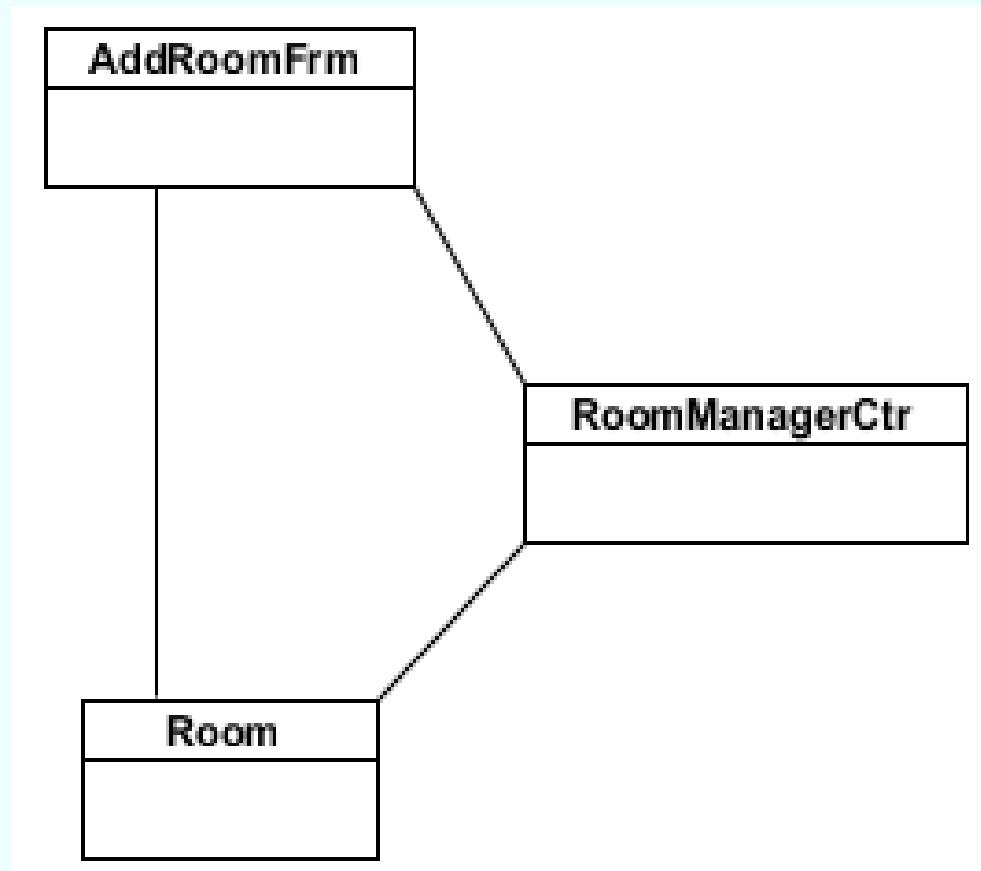
## Đặc trưng:

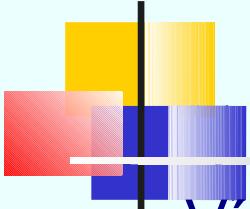
- Lớp thực thể chứa các thuộc tính và các phương thức get/set cho mỗi thuộc tính, và
- Các thao tác liên quan đến CSDL mà liên quan đến lớp thực thể nào thì đều đặt trong lớp điều khiển đó
- Các lớp thực thể kiểu này được gọi là lớp bean
- Trong nhiều trường hợp, không còn cần đến lớp điều khiển nữa vì lớp bean đã kiêm luôn vai trò của lớp điều khiển



# MVC với thực thể bean (2)

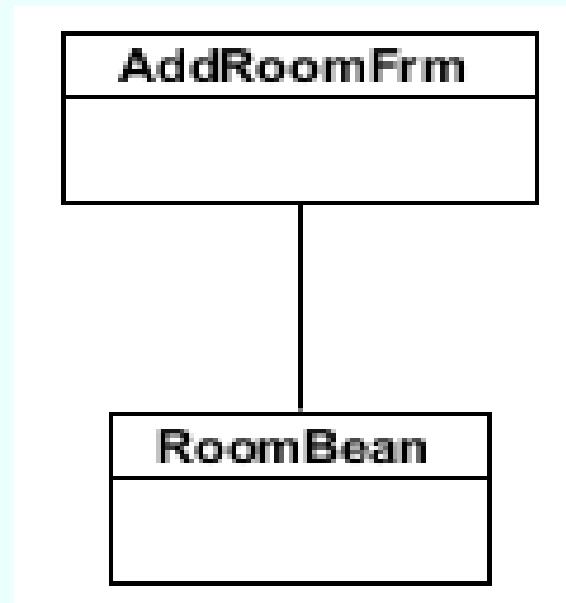
Ví dụ modul quản lí phòng của Manager, sơ đồ lớp cuối  
pha phân tích của chức năng thêm phòng:

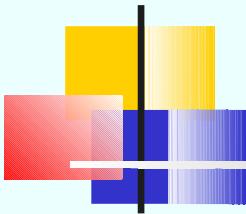




# MVC với thực thể bean (3)

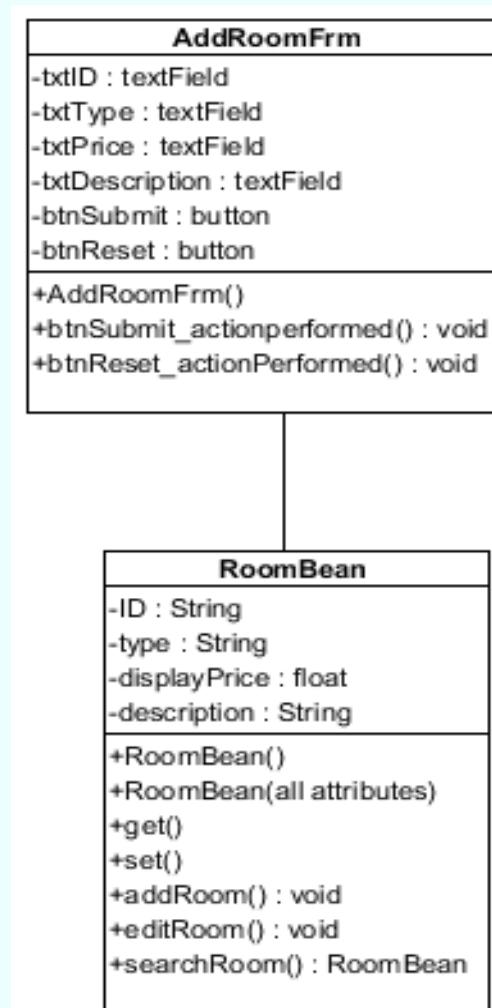
Ví dụ chức năng thêm phòng của Manager, sơ đồ lớp  
theo MVC dùng thực thể bean:

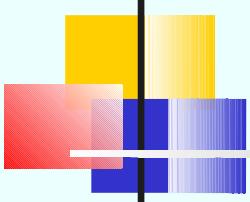




# Hoàn thiện sơ đồ lớp

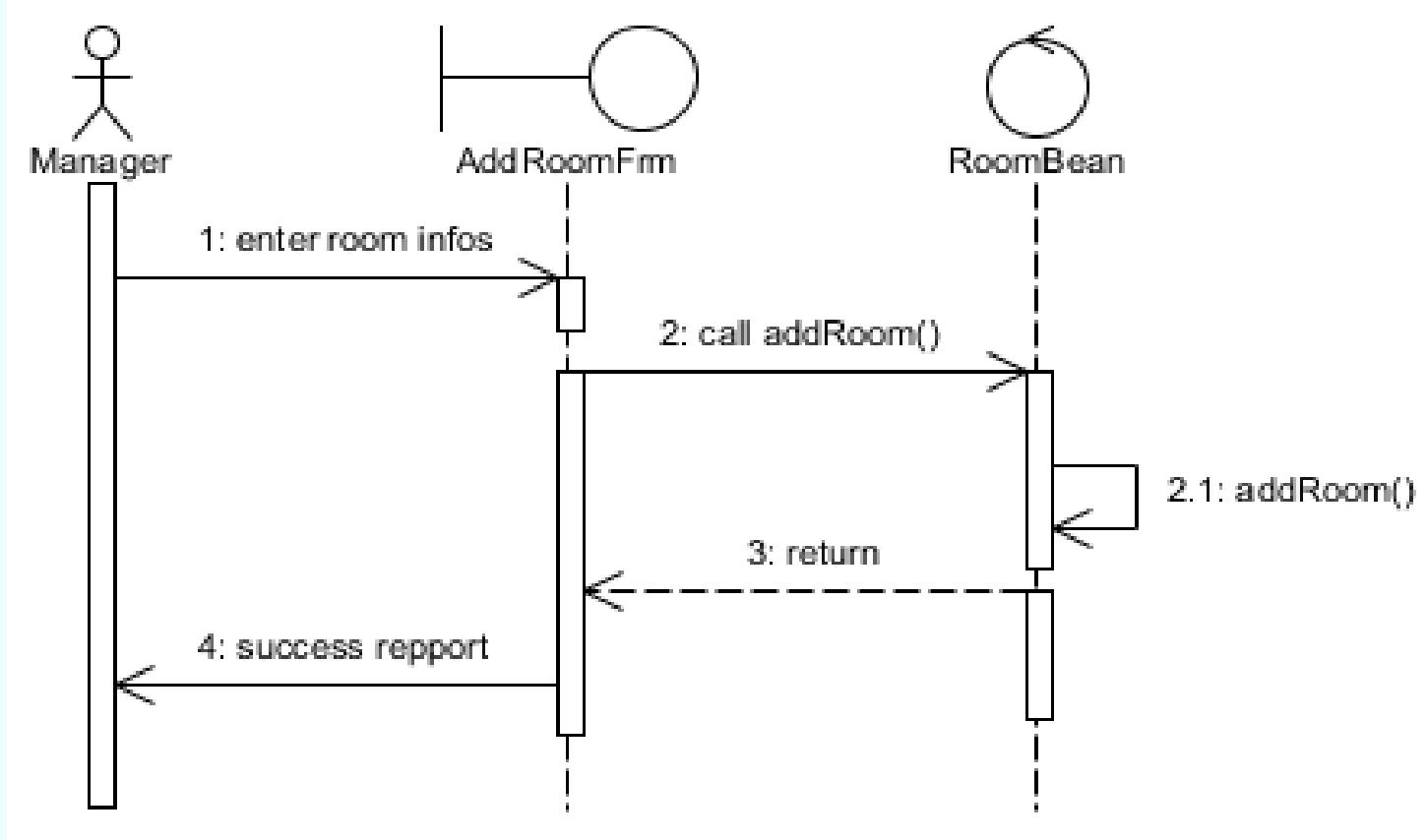
Kết quả thu được sơ đồ lớp sau khi áp dụng các nguyên lí A, B, và C như sau:

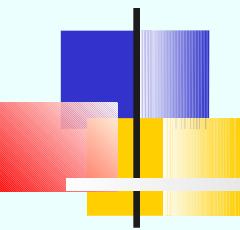




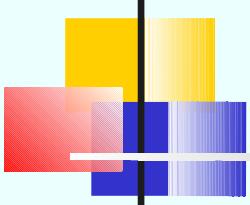
# Sơ đồ tuần tự pha thiết kế

Sơ đồ tuần tự cho cách thiết kế dùng bean:





# Thiết kế theo mô hình MVC cải tiến (hiện đại)

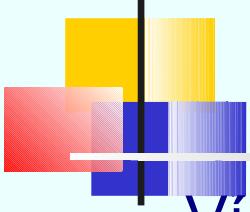


# MVC cải tiến (1)

---

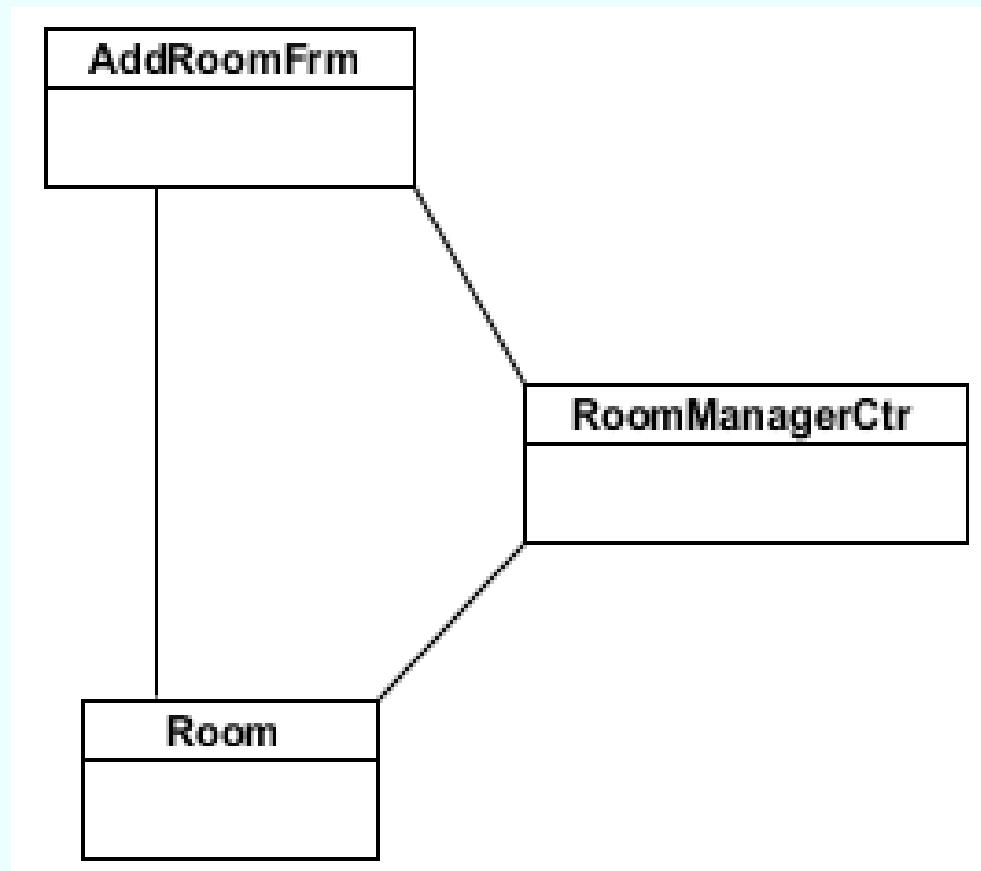
## Đặc trưng:

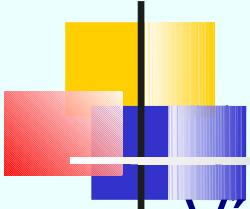
- Lớp thực thể thuần
- Các thao tác liên quan đến CSDL đều đặt trong lớp điều khiển
- Các lớp điều khiển giành quyền điều khiển toàn bộ các lớp view và control



# MVC cải tiến (2)

Ví dụ modul quản lí phòng của Manager, sơ đồ lớp cuối  
pha phân tích của chức năng thêm phòng:



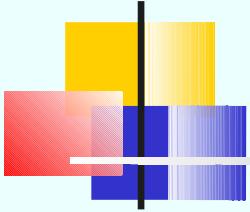


# MVC cải tiến (3)

---

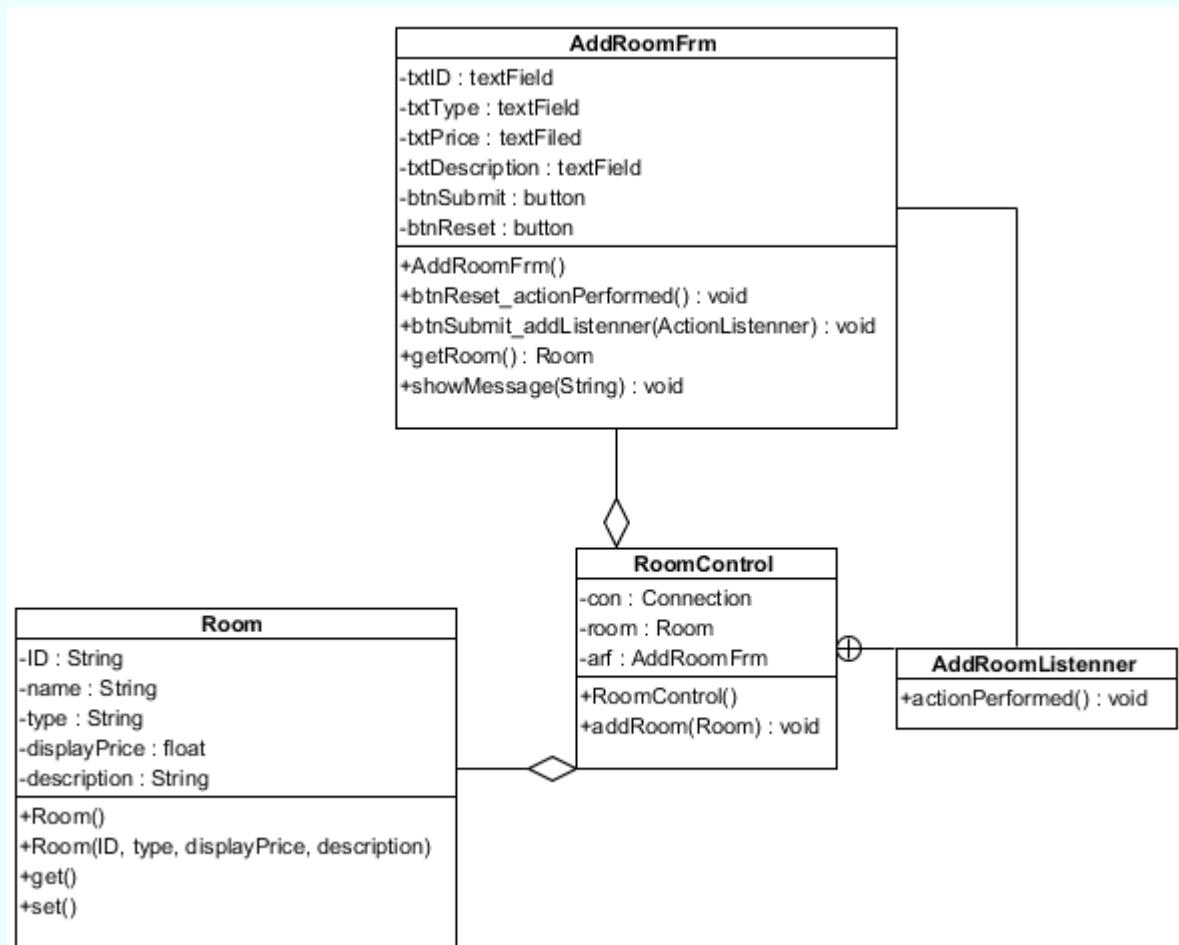
Ví dụ chức năng thêm phòng của Manager, sơ đồ lớp theo MVC cải tiến:

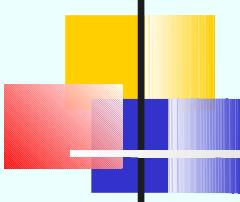
- Khi nút Add trên form của lớp biên AddRoomFrm bị click thì lớp biên này không được xử lí gì mà phải truyền sự kiện này xuống cho lớp điều khiển
- Để làm được việc này, trên lớp biên có phương thức truyền quyền điều khiển cho lớp điều khiển, và
- Trong lớp điều khiển có một lớp nội tại (inner class) dùng để nhận sự kiện do lớp biên truyền xuống để xử lí



# Hoàn thiện sơ đồ lớp

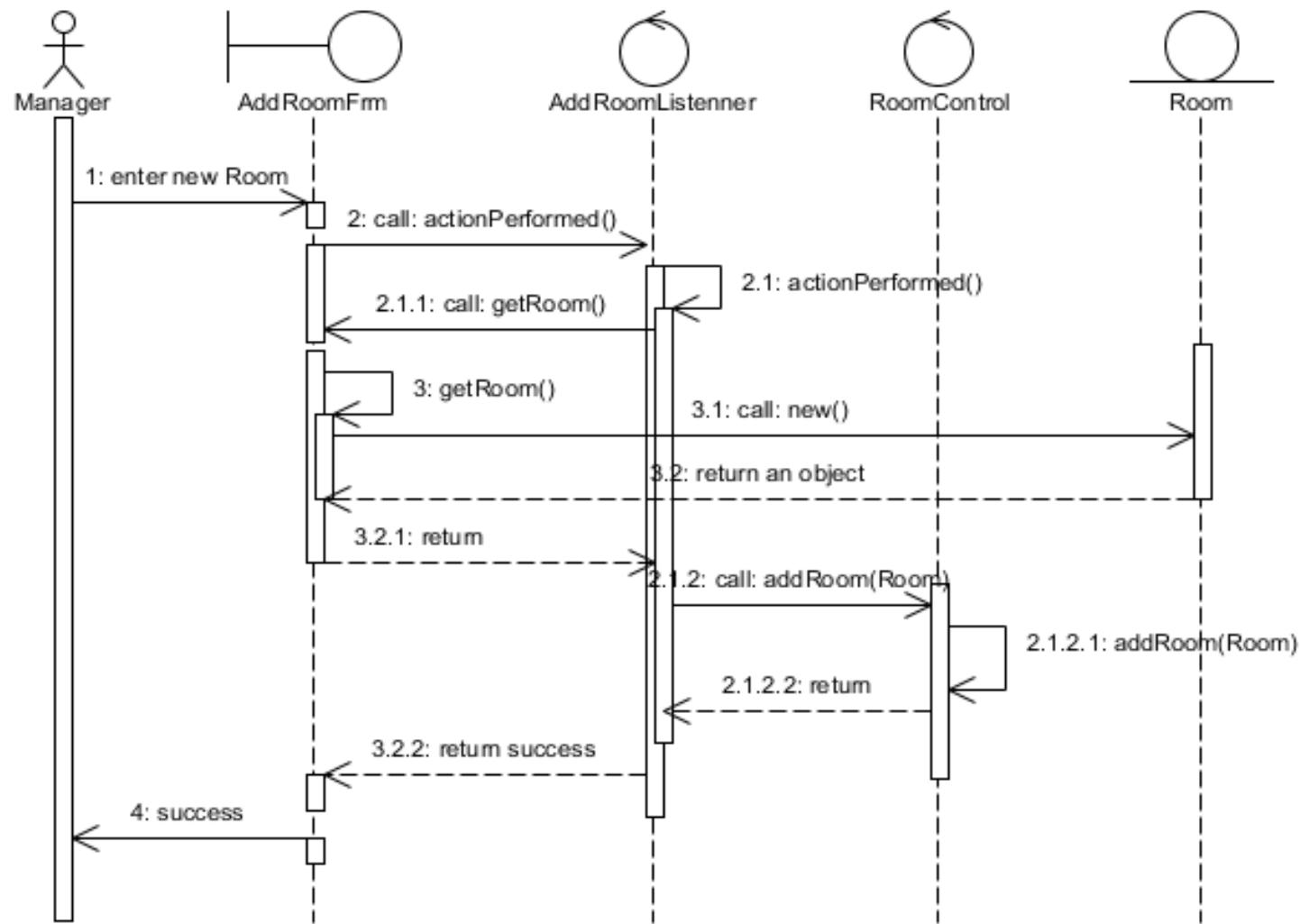
Kết quả thu được sơ đồ lớp sau khi áp dụng các nguyên lí A, B, và C như sau:

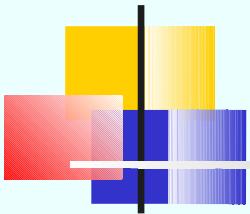




# Sơ đồ tuần tự pha thiết kế

Sơ đồ tuần tự cho cách thiết kế dùng MVC cải tiến:



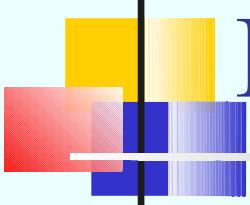


# Bài tập

---

Thiết kế tương tự (dung MVC với thực thể bean và dùng MVC cải tiến) cho các chức năng và modul:

- Sửa thông tin phòng
- Xóa thông tin phòng
- Khách hàng đặt chỗ tại quầy với nhân viên tiếp tân
- Khách hàng checkin tại quầy với nhân viên tiếp tân
- Khách hàng trả phòng và thanh toán tại quầy với nhân viên tiếp tân

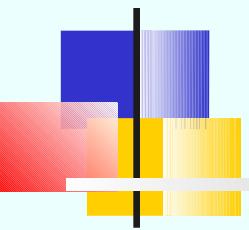


# Bài tập về nhà

---

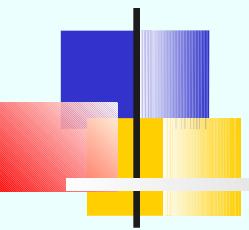
Với mỗi modul cá nhân:

- Vẽ lại sơ đồ UC chi tiết của hệ thống và của modul
- Vẽ lại sơ đồ các lớp sau pha phân tích
- Thiết kế hệ thống theo mô hình MVC dùng thực thể bean và theo mô hình MVC cải tiến
- Vẽ lại sơ đồ tuần tự sau pha thiết kế (cho mỗi loại)



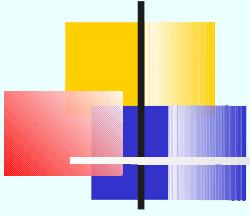
# Questions?

---



# Công nghệ phần mềm Pha cài đặt

*Giảng viên: TS. Nguyễn Mạnh Hùng  
Học viện Công nghệ Bưu chính Viễn thông (PTIT)*

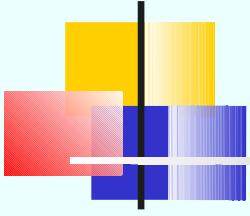


# Pha cài đặt (1)

---

Mục đích:

- Cài đặt thành chương trình
- Kiểm thử chương trình

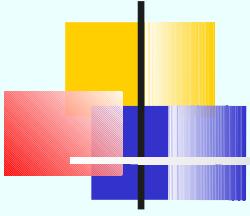


# Chuẩn bị kiểm thử (1)

---

## Thực hiện:

- Viết test case cho mỗi phương thức và mỗi lớp trước khi cài đặt chúng
- Test case dưới dạng hộp đen (black-box test):
  - Chỉ rõ đầu vào
  - Đầu ra mong muốn

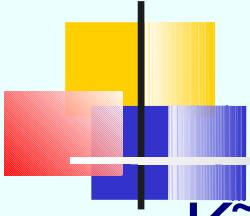


# Chuẩn bị kiểm thử (2)

---

Về nguyên tắc:

- Phải test hết tất cả các trường hợp có thể có của các kiểu dữ liệu
- Tuy nhiên nếu làm vậy số trường hợp phải test là quá lớn
- → dùng kỹ thuật test biên



# Chuẩn bị kiểm thử (3)

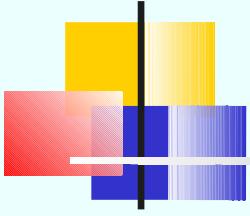
---

## Kỹ thuật test biên:

- Nếu một tham số đầu vào có một giới hạn biên  $x$ , thì phải test ít nhất 4 trường hợp:
  - 1: giá trị đầu vào bất kì **cách xa**  $x$
  - 2: giá trị đầu vào **ngay trên**  $x$
  - 3: giá trị đầu vào **ngay dưới**  $x$
  - 4: giá trị đầu vào **đúng bằng**  $x$

## Ví dụ:

- Nếu phép chia cho số nguyên có điều kiện số bị chia # 0 thì phải test khi số bị chia: -1, 0, 1, và một số  $>100$

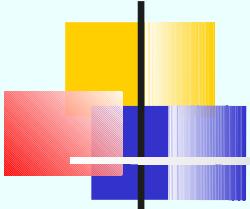


# Chuẩn bị kiểm thử (4)

---

Kĩ thuật test biên (tt):

- Nếu một tham số đầu vào có 2 giới hạn biên  $x_1$  và  $x_2$  thì phải test ít nhất 7 trường hợp:
  - 1,2: giá trị đầu vào **đúng bằng**  $x_1$ , **ngay trên**  $x_1$
  - 3,4: giá trị đầu vào **ngay dưới**  $x_2$ , **đúng bằng**  $x_2$
  - 5: giá trị đầu vào **đúng bằng**  $(x_1+x_2)/2$
  - 6: giá trị đầu vào **nhỏ hơn**  $x_1$
  - 7: giá trị đầu vào **lớn hơn**  $x_2$

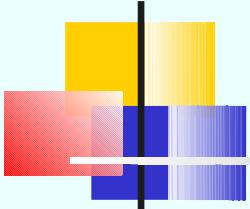


# Chuẩn bị kiểm thử (5)

---

Kĩ thuật test chức năng thao tác CSDL:

- Nếu chức năng thêm một đối tượng vào CSDL thì phải test ít nhất 3 trường hợp:
  - 1: thêm một đối tượng **chưa có** trong CSDL
  - 2: thêm một đối tượng **đã có** trong CSDL
  - 3: thêm **liên tục 2 lần** một đối tượng **chưa có** trong CSDL

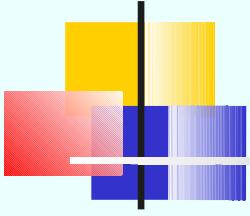


# Chuẩn bị kiểm thử (6)

---

Kĩ thuật test chức năng thao tác CSDL (tt):

- Nếu chức năng sửa một đối tượng trong CSDL thì phải test ít nhất 3 trường hợp:
  - 1: sửa một đối tượng **chưa có** trong CSDL
  - 2: sửa một đối tượng **đã có** trong CSDL
  - 3: sửa **liên tục 2 lần** một thuộc tính của đối tượng **đã có** trong CSDL

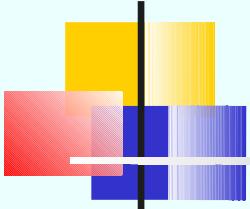


# Chuẩn bị kiểm thử (7)

---

Kĩ thuật test chức năng thao tác CSDL (tt):

- Nếu chức năng xóa một đối tượng trong CSDL thì phải test ít nhất 3 trường hợp:
  - 1: xóa một đối tượng **chưa có** trong CSDL
  - 2: xóa một đối tượng **đã có** trong CSDL
  - 3: xóa **liên tục 2 lần** một đối tượng **đã có** trong CSDL

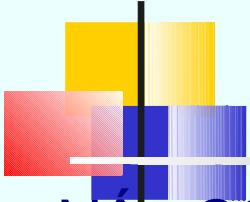


# Chuẩn bị kiểm thử (8)

---

Kĩ thuật test chức năng thao tác CSDL (tt):

- Nếu chức năng tìm kiếm một số đối tượng trong CSDL thì phải test ít nhất 2 trường hợp:
  - 1: tìm kiếm một đối tượng **chưa có** trong CSDL
  - 2: tìm kiếm một đối tượng **đã có** trong CSDL

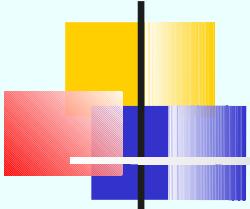


# Chuẩn bị kiểm thử (9)

---

Nếu CSDL có 3 bảng: **KhachHang** lưu thông tin khách hàng, bảng **SanPham** lưu thông tin sản phẩm, bảng **HoaDon** lưu thông tin một khách hàng mỗi lần mua một số sản phẩm. Khi đó, chức năng thêm một hóa đơn vào trong CSDL thì phải test ít nhất 2 trường hợp:

- 1: thêm một hóa đơn **chưa có** trong CSDL
- 2: thêm một hóa đơn **đã có** trong CSDL
- 3: thêm một hóa đơn mà **khách hàng chưa có** trong CSDL
- 4: thêm một hóa đơn mà **sản phẩm chưa có** trong CSDL
- 5: thêm một hóa đơn mà **cả khách hàng và sản phẩm chưa có** trong CSDL



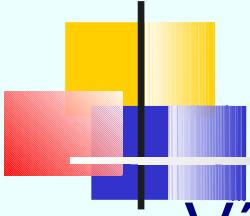
# Chuẩn bị kiểm thử (10)

---

Ví dụ test case cho chức năng thêm một quyển sách vào CSDL:

- B1: Dữ liệu hiện thời:

<code>id</code>	<code>name</code>	<code>author</code>	<code>release year</code>
0	Java core	Steve Jobs	2011
1	Software Engineering	Scatches	2003



# Chuẩn bị kiểm thử (11)

---

Ví dụ test case cho chức năng thêm một quyển sách vào CSDL (tt):

- B2: Các thao tác, và kết quả mong đợi (có thể dùng requirement diagram/test case trong VP):

Các bước thao tác

1. NV chọn chức năng thêm sách

2. Nhân viên nhập:

Id=2,

Tên = Data mining

Tác giả = Jennings

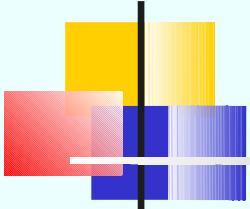
Năm xb = 2012

Và click nút submit 1 lần

Kết quả mong đợi

Giao diện thêm sách hiện ra, gồm các ô nhập: id, tên, tác giả, năm xb và nút submit

Thông báo thêm thành công và CSDL sẽ có như sau:



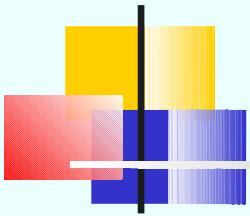
# Chuẩn bị kiểm thử (12)

---

Ví dụ test case cho chức năng thêm một quyển sách vào CSDL (tt):

- B3: kết quả mong đợi trong CSDL:

<b>id</b>	<b>name</b>	<b>author</b>	<b>release year</b>
0	Java core	Steve Jobs	2011
1	Software Engineering	Scatches	2003
2	Data mining	Jennings	2012

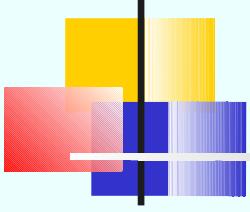


# Bài tập

---

Viết test case cho các chức năng sau:

- Sửa thông tin phòng
- Xóa thông tin phòng
- Đặt phòng tại quầy
- Checkin
- Trả phòng và thanh toán

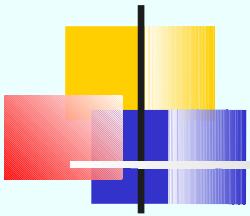


# Bài tập về nhà

---

Với mỗi modul cá nhân:

- Viết các test case cho từng chức năng của modul

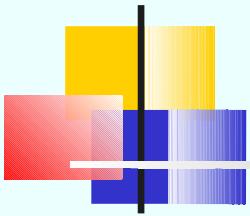


# Cài đặt (1)

---

Thực hiện theo thứ tự:

- Cài đặt các lớp thực thể
- Cài đặt các lớp giao diện
- Cài đặt các lớp điều khiển

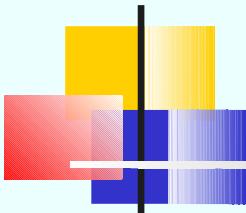


# Cài đặt (2)

---

## Chú thích code:

- Chú thích code là cần thiết và quan trọng cho pha bảo trì
- Nên chú thích code đầu mỗi lớp và đầu mỗi phương thức

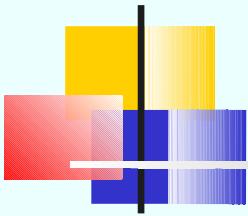


# Cài đặt (3)

---

Chú thích code cần có các thông tin tối thiểu:

- Tên của phần code, mô tả ngắn gọn chức năng và hoạt động của đoạn code
- Tên người code và ngày code
- Ngày được duyệt và tên người duyệt
- Tên các tham số, theo abc, giải thích ngắn gọn ý nghĩa từng tham số
- Tên các file truy cập/thay đổi bởi đoạn code
- Đầu vào – đầu ra
- Khả năng xử lí ngoại lệ
- Danh sách các thay đổi, cập nhật từ lần code đầu



# Cài đặt (4)

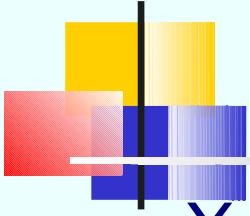
---

## Đặt tên biến:

- Tên biến nên đặt theo cách gợi nhớ, trừ các biến chạy
- Bắt đầu bằng chữ thường
- Thống nhất với nhau

## Ví dụ:

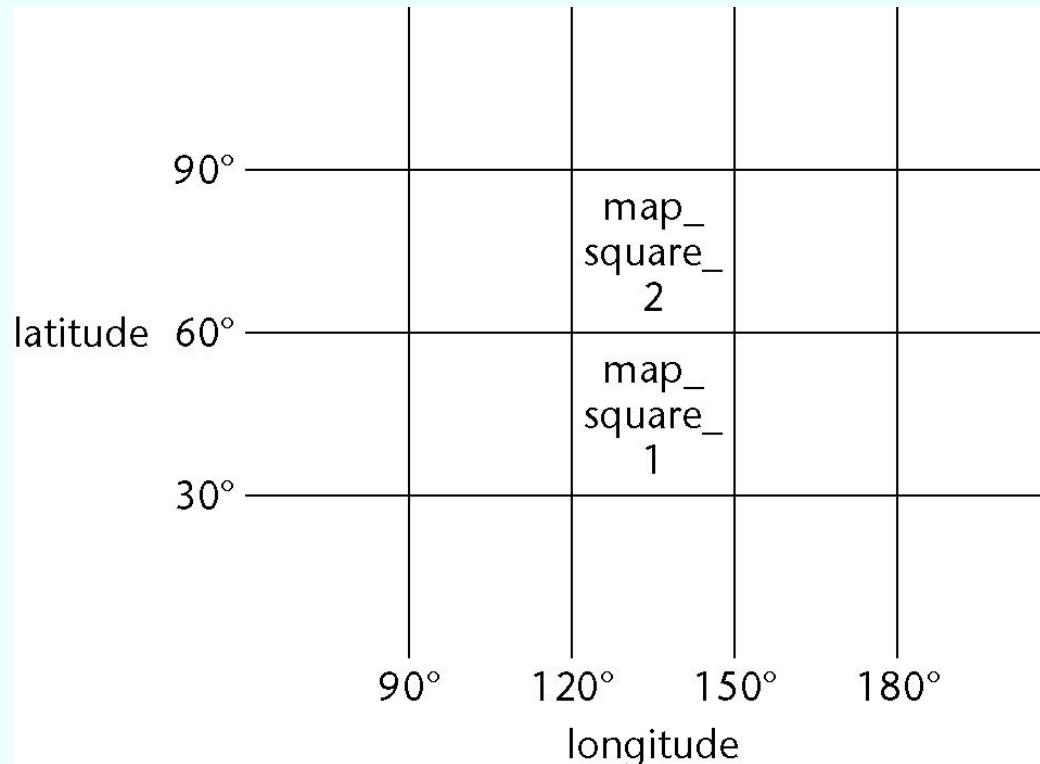
- Có thể sử dụng: frequencyAverage, frequencyTotal
- Có thể sử dụng: averageFrequency, totalFrequency
- Nhưng **không thể** đặt tên cùng nhau:  
FrequencyAverage, totalFrequency

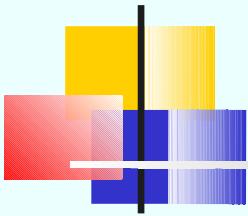


# Cài đặt (5)

Xét ví dụ với câu lệnh if:

- Xác định xem một điểm có tọa độ nằm trong vùng map\_square1 hoặc map\_square2 hoặc không



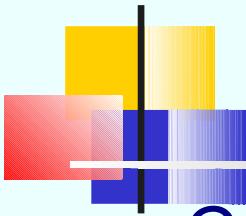


# Cài đặt (6)

## Cách 1:

- Trình bày không chấp nhận được

```
if (latitude > 30 && longitude > 120) {if (latitude <= 60 && longitude <= 150)
mapSquareNo = 1; else if (latitude <= 90 && longitude <= 150) mapSquareNo = 2
else print "Not on the map";} else print "Not on the map";
```

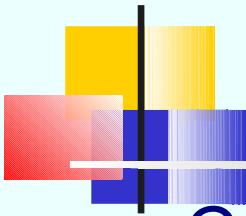


# Cài đặt (7)

## Cách 2:

- Khuôn dạng được, nhưng cấu trúc không được

```
if (latitude > 30 && longitude > 120)
{
    if (latitude <= 60 && longitude <= 150)
        mapSquareNo = 1;
    else
        if (latitude <= 90 && longitude <= 150)
            mapSquareNo = 2;
        else
            print "Not on the map";
}
else
    print "Not on the map";
```

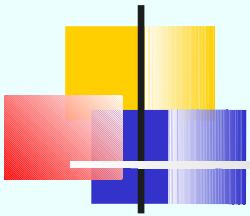


# Cài đặt (8)

## Cách 3:

- Chấp nhận được

```
if (longitude > 120 && longitude <= 150 && latitude > 30 && latitude <= 60)
    mapSquareNo = 1;
else
    if (longitude > 120 && longitude <= 150 && latitude > 60 && latitude <= 90)
        mapSquareNo = 2;
    else
        print "Not on the map";
```



# Cài đặt (9)

Nguyên tắc dùng lệnh if:

- Nếu có lện if-if liền nhau dạng

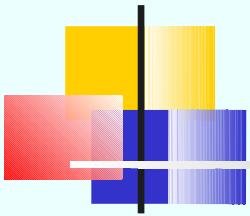
If (điều kiện 1)

If (điều kiện 2) làm x

→ thì nên chuyển thành:

If (điều kiện 1) && (điều kiện 2) làm x

- Nếu câu lệnh if lồng nhau có độ sâu 3 tầng trở lên thì nên xem xét lại dùng cách khác để điều khiển

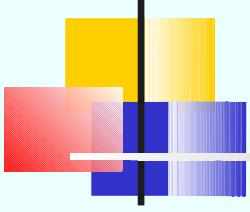


# Cài đặt (10)

---

Nguyên tắc phân chia và code modul/ method:

- Một method (modul) chỉ nên có tối đa 30-50 câu lệnh
- Nếu có nhiều hơn thì nên tách thành các modul con

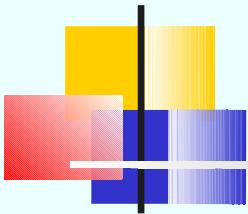


# Tích hợp (1)

---

Các kỹ thuật tích hợp:

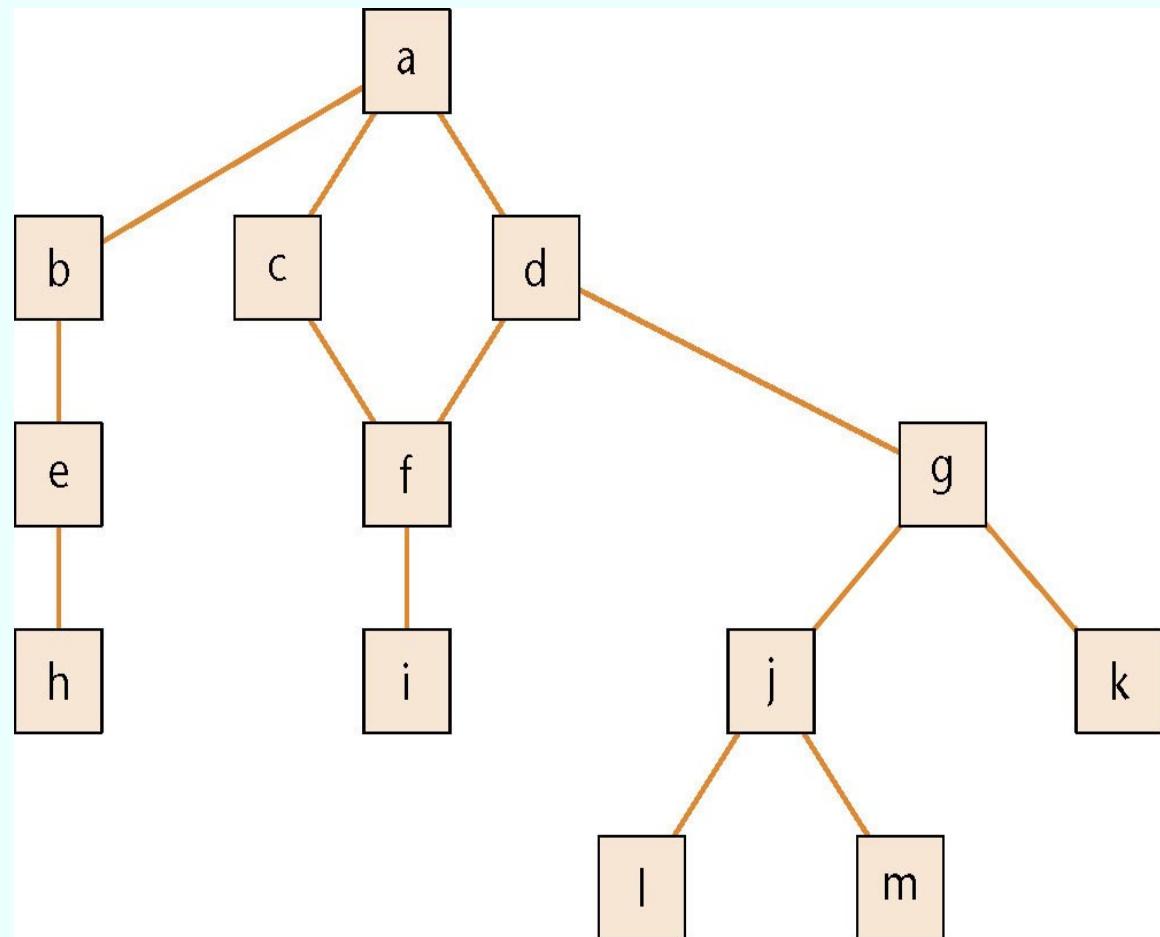
- Top-down: Tích hợp từ trên xuống
- Bottom-up: Tích hợp từ dưới lên
- Sandwich: Tích hợp theo cả hai chiều trên xuống và dưới lên

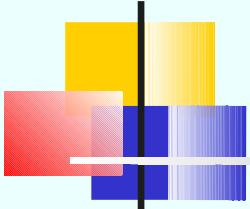


# Tích hợp (2)

Top-down:

- Thứ tự tích hợp là:  
a,b,c,d,e,f,g,h,i,j  
,k,l,m

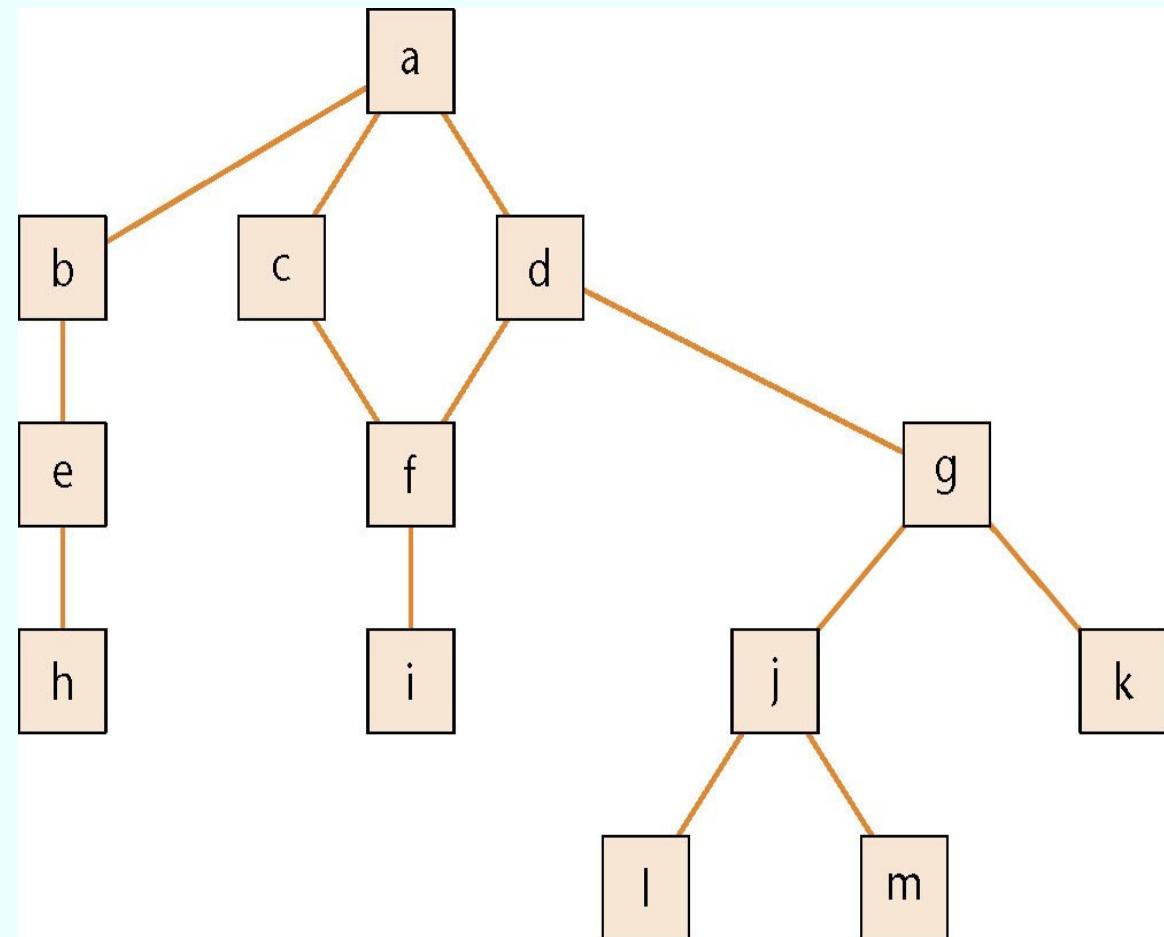


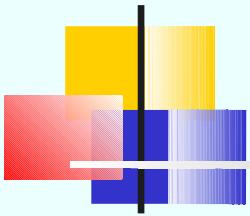


# Tích hợp (3)

Top-down (tt):

- Để test modul a, phải coi các modul b,c,d là các hằng số (stubs)

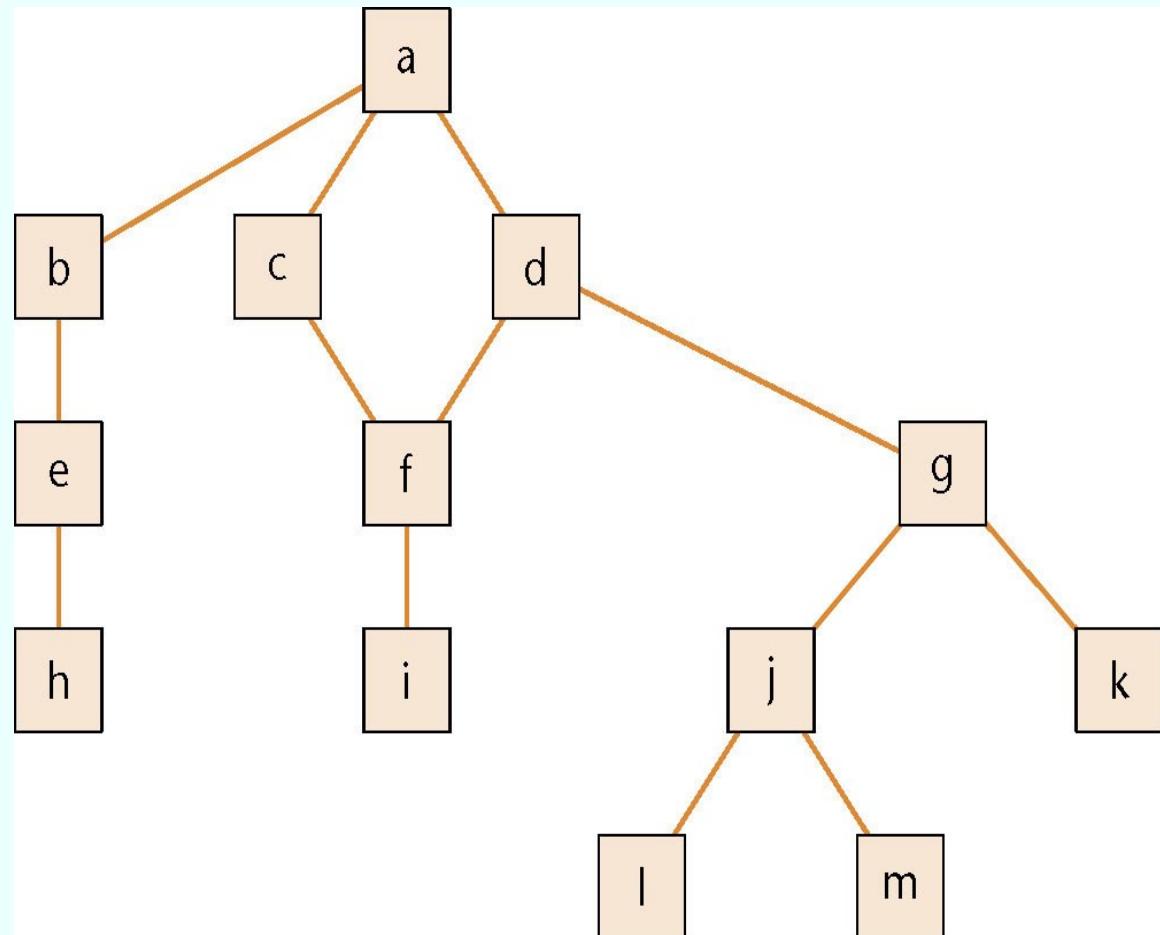


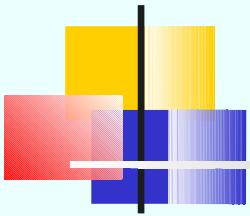


# Tích hợp (4)

Bottom-up:

- Thứ tự tích hợp là:  
l,m,h,i,j,k,e,f,g,b  
,c,d,a

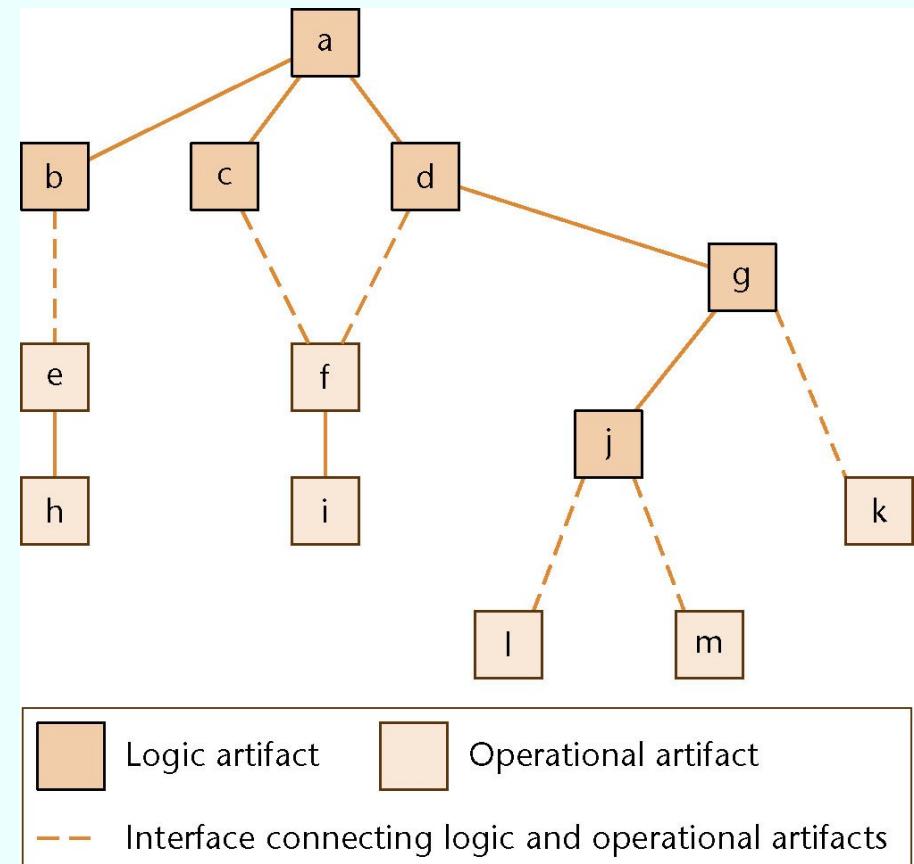


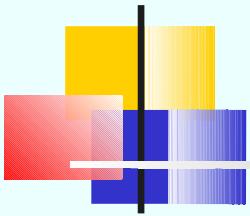


# Tích hợp (5)

## Sandwich:

- Các modul logic thì tích hợp top-down
- Các modul thực hiện trực tiếp thì tích hợp bottom-up



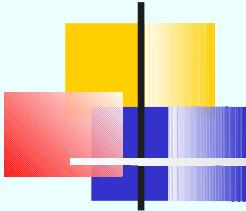


# Kiểm thử (1)

---

Với mỗi modul/ method:

- Chạy các test case đã viết trong phần đầu, lưu kết quả chạy thành nhật kí chạy test case



# Kiểm thử (2)

Ví dụ với modul thêm một sách:

- Chạy các test case đã viết trong phần đầu, lưu kết quả chạy thành nhật ký chạy test case

Các test case

Kết quả

1. thêm một sách chưa có id  
trong CSDL                              passed

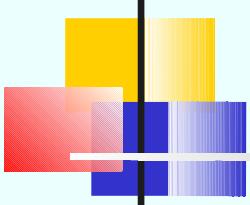
2. Thêm một sách đã có id trong  
CSDL                                      error

3. Thêm liên tục 2 lần một sách  
chưa có id trong csdl                error



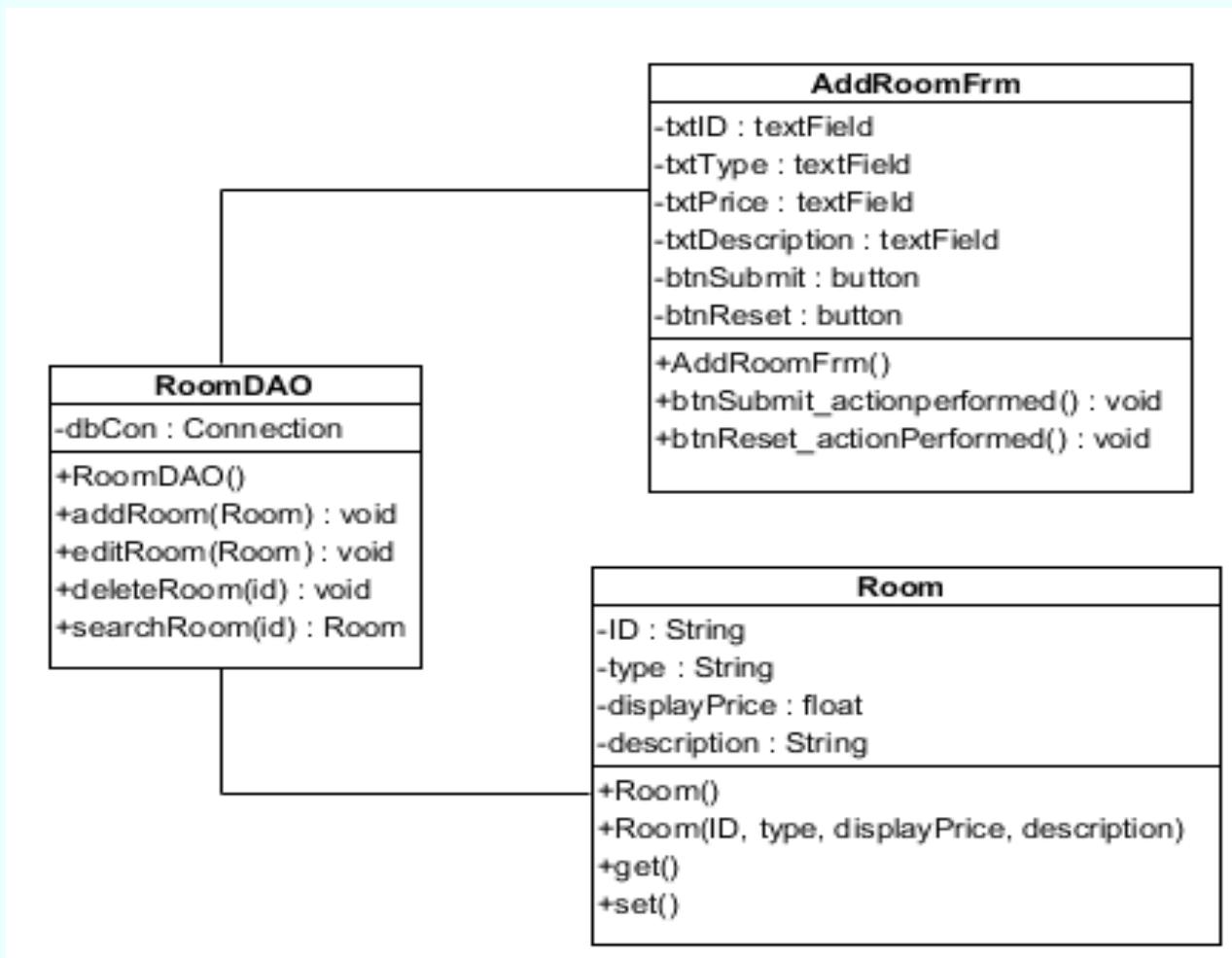
# Ví dụ Cài đặt theo thiết kế MVC thuận

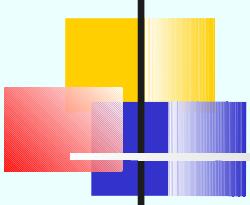
*Tham khảo chi tiết tại:*  
<http://coderandcode.blogspot.com/2014/02/variations-of-mvc-model.html>



# Sơ đồ lớp pha thiết kế

Với chức năng thêm phòng:



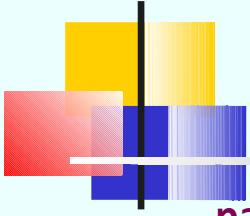


# Các bảng CSDL liên quan

---

Bảng chứa thông tin phòng:

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI
🔑 id	VARCHAR(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>				
❖ name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
❖ type	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
❖ displayPrice	FLOAT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
❖ description	VARCHAR(250)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



# Lớp Room (1)

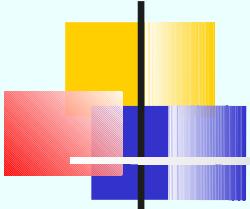
---

```
package mvcPure;

public class Room {
    private String id;
    private String name;
    private String type;
    private float displayPrice;
    private String description;

    public Room() {
        super();
    }

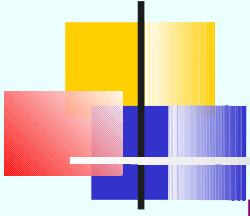
    public Room(String id, String name, String type,
               float displayPrice, String description) {
        super();
        this.id = id;
        this.name = name;
        this.type = type;
        this.displayPrice = displayPrice;
        this.description = description;
    }
}
```



# Lớp Room (2)

---

```
public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getType() {
    return type;
}
public void setType(String type) {
    this.type = type;
}
public float getDisplayPrice() {
    return displayPrice;
}
public void setDisplayPrice(float displayPrice) {
    this.displayPrice = displayPrice;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}
```



# Lớp RoomDAO (1)

---

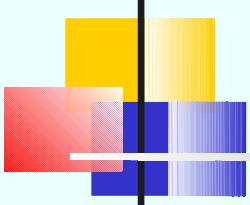
```
package mvcPure;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class RoomDAO {
    private Connection con;

    public RoomDAO(){
        String dbUrl = "jdbc:mysql://localhost:3306/hotel";
        String dbClass = "com.mysql.jdbc.Driver";

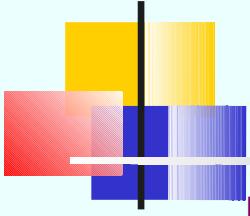
        try {
            Class.forName(dbClass);
            con = DriverManager.getConnection (dbUrl,
                "root", "12345678");
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```



# Lớp RoomDAO (2)

---

```
public void addRoom(Room room){  
    String sql = "INSERT INTO tblRoom(id, name, type, displayPrice,  
        description) VALUES(?, ?, ?, ?, ?)";  
    try{  
        PreparedStatement ps = con.prepareStatement(sql);  
        ps.setString(1, room.getId());  
        ps.setString(2, room.getName());  
        ps.setString(3, room.getType());  
        ps.setFloat(4, room.getDisplayPrice());  
        ps.setString(5, room.getDescription());  
  
        ps.executeUpdate();  
    }catch(Exception e){  
        e.printStackTrace();  
    }  
}
```



# Lớp AddRoomFrm (1)

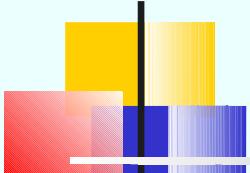
---

```
package mvcPure;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AddRoomFrm extends JFrame implements ActionListener{
    private JTextField txtID;
    private JTextField txtName;
    private JTextField txtType;
    private JTextField txtDisplayPrice;
    private JTextField txtDescription;
    private JButton btnSubmit;
    private JButton btnReset;
```



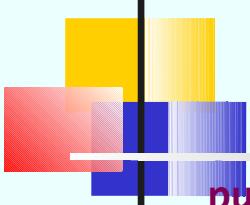
# Lớp AddRoomFrm (2)

---

```
public AddRoomFrm(){
    super("Room management pure-MVC");
    txtID = new JTextField(15);
    txtName = new JTextField(15);
    txtType = new JTextField(15);
    txtDisplayPrice = new JTextField(15);
    txtDescription = new JTextField(15);
    btnSubmit = new JButton("Submit");
    btnReset = new JButton("Reset");

    JPanel content = new JPanel();
    content.setLayout(new GridLayout(6,2));
    content.add(new JLabel("ID:")); content.add(txtID);
    content.add(new JLabel("Name:")); content.add(txtName);
    content.add(new JLabel("Type:")); content.add(txtType);
    content.add(new JLabel("Display price:")); content.add(txtDisplayPrice);
    content.add(new JLabel("Description:")); content.add(txtDescription);
    content.add(btnReset); content.add(btnSubmit);
    btnSubmit.addActionListener(this);
    btnReset.addActionListener(this);
    this.setContentPane(content);
    this.pack();

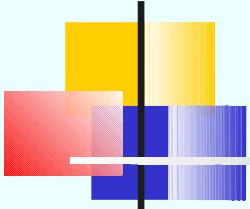
    this.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){
            System.exit(0);
        }
    });
}
```



# Lớp AddRoomFrm (3)

---

```
public void actionPerformed(ActionEvent e) {  
    JButton btn = (JButton) e.getSource();  
    if(btn.equals(btnSubmit)){  
        btnSubmit_actionperformed();  
    }else if(btn.equals(btnReset)){  
        btnReset_actionperformed();  
    }  
}  
  
public void btnSubmit_actionperformed() {  
    Room room = new Room();  
    room.setId(txtID.getText());  
    room.setName(txtName.getText());  
    room.setType(txtType.getText());  
    room.setDisplayPrice(Float.parseFloat(txtDisplayPrice.getText()));  
    room.setDescription(txtDescription.getText());  
  
    RoomDAO rd = new RoomDAO();  
    rd.addRoom(room);  
    JOptionPane.showMessageDialog(this, "Add room successfully!");  
}  
  
public void btnReset_actionperformed() {  
    txtID.setText(""); txtName.setText("");  
    txtType.setText(""); txtDisplayPrice.setText("");  
    txtDescription.setText("");  
}
```



# Lớp Test

---

```
package mvcPure;

public class Test {

    public static void main(String[] args) {
        AddRoomFrm arf = new AddRoomFrm();
        arf.setVisible(true);
    }
}
```

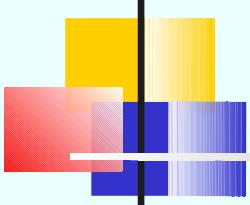
Lưu ý trước khi chạy, phải:

- Cài đặt CSDL và bật server MySQL
- Add driver của Jdbc mysql vào library của project



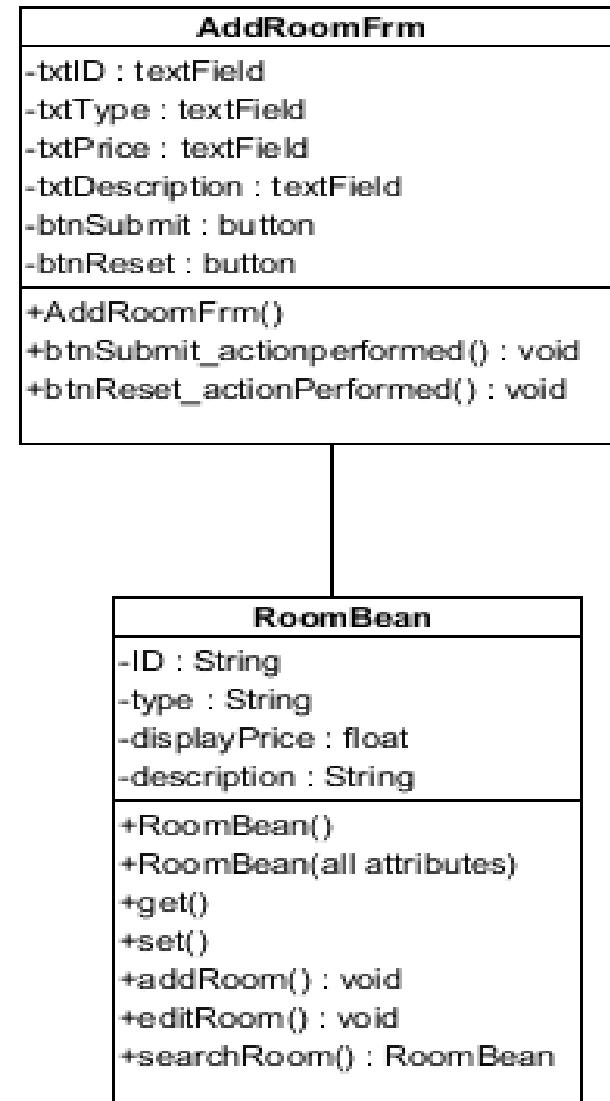
# Ví dụ Cài đặt theo thiết kế MVC bean

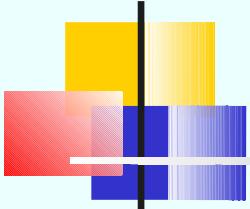
*Tham khảo chi tiết tại:*  
<http://coderandcode.blogspot.com/2014/02/variations-of-mvc-model.html>



# Sơ đồ lớp pha thiết kế

Với chức năng thêm phòng:





# Lớp RoomBean (1)

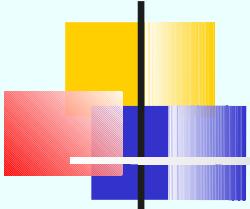
---

```
package mvcBean;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class RoomBean {
    private String id;
    private String name;
    private String type;
    private float displayPrice;
    private String description;

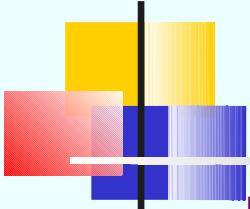
    public RoomBean() {
        super();
    }
    public RoomBean(String id, String name, String type, float
DisplayPrice, String description) {
        super();
        this.id = id;
        this.name = name;
        this.type = type;
        this.displayPrice = displayPrice;
        this.description = description;
    }
}
```



# Lớp RoomBean (2)

---

```
public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getType() {
    return type;
}
public void setType(String type) {
    this.type = type;
}
public float getDisplayPrice() {
    return displayPrice;
}
public void setDisplayPrice(float displayPrice) {
    this.displayPrice = displayPrice;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}
```

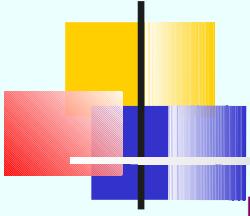


# Lớp RoomBean (3)

---

```
public void addRoom(){
    String dbUrl = "jdbc:mysql://localhost:3306/hotel";
    String dbClass = "com.mysql.jdbc.Driver";
    String sql = "INSERT INTO tblRoom(id, name, type, displayPrice,
                  description) VALUES(?, ?, ?, ?, ?)";
    try{
        Class.forName(dbClass);
        Connection con = DriverManager.getConnection (
            dbUrl, "root", "12345678");
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, getId());
        ps.setString(2, getName());
        ps.setString(3, getType());
        ps.setFloat(4, getDisplayPrice());
        ps.setString(5, getDescription());

        ps.executeUpdate();
        con.close();
    }catch(Exception e){
        e.printStackTrace();
    }
}
```



# Lớp AddRoomFrm (1)

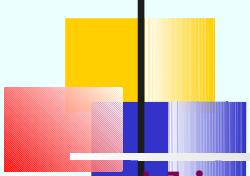
---

```
package mvcBean;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AddRoomFrm extends JFrame implements ActionListener{
    private JTextField txtID;
    private JTextField txtName;
    private JTextField txtType;
    private JTextField txtDisplayPrice;
    private JTextField txtDescription;
    private JButton btnSubmit;
    private JButton btnReset;
```



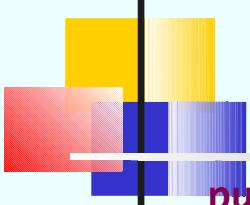
# Lớp AddRoomFrm (2)

---

```
public AddRoomFrm(){
    super("Room management pure-MVC");
    txtID = new JTextField(15);
    txtName = new JTextField(15);
    txtType = new JTextField(15);
    txtDisplayPrice = new JTextField(15);
    txtDescription = new JTextField(15);
    btnSubmit = new JButton("Submit");
    btnReset = new JButton("Reset");

    JPanel content = new JPanel();
    content.setLayout(new GridLayout(6,2));
    content.add(new JLabel("ID:")); content.add(txtID);
    content.add(new JLabel("Name:")); content.add(txtName);
    content.add(new JLabel("Type:")); content.add(txtType);
    content.add(new JLabel("Display price:")); content.add(txtDisplayPrice);
    content.add(new JLabel("Description:")); content.add(txtDescription);
    content.add(btnReset); content.add(btnSubmit);
    btnSubmit.addActionListener(this);
    btnReset.addActionListener(this);
    this.setContentPane(content);
    this.pack();

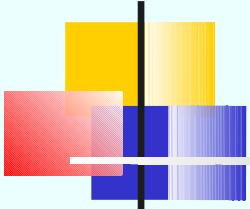
    this.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){
            System.exit(0);
        }
    });
}
```



# Lớp AddRoomFrm (3)

---

```
public void actionPerformed(ActionEvent e) {  
    JButton btn = (JButton) e.getSource();  
    if(btn.equals(btnSubmit)){  
        btnSubmit_actionperformed();  
    }else if(btn.equals(btnReset)){  
        btnReset_actionperformed();  
    }  
}  
  
public void btnSubmit_actionperformed() {  
    RoomBean room = new RoomBean();  
    room.setId(txtID.getText());  
    room.setName(txtName.getText());  
    room.setType(txtType.getText());  
    room.setDisplayPrice(Float.parseFloat(txtDisplayPrice.getText()));  
    room.setDescription(txtDescription.getText());  
  
    room.addRoom();  
    JOptionPane.showMessageDialog(this, "Add room successfully!");  
}  
  
public void btnReset_actionperformed() {  
    txtID.setText(""); txtName.setText("");  
    txtType.setText(""); txtDisplayPrice.setText("");  
    txtDescription.setText("");  
}
```



# Lớp Test

---

```
package mvcBean;

public class Test {

    public static void main(String[] args) {
        AddRoomFrm arf = new AddRoomFrm();
        arf.setVisible(true);
    }
}
```

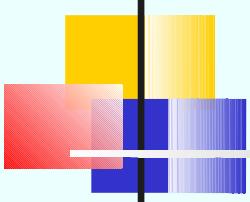
Lưu ý trước khi chạy, phải:

- Cài đặt CSDL và bật server MySQL
- Add driver của Jdbc mysql vào library của project



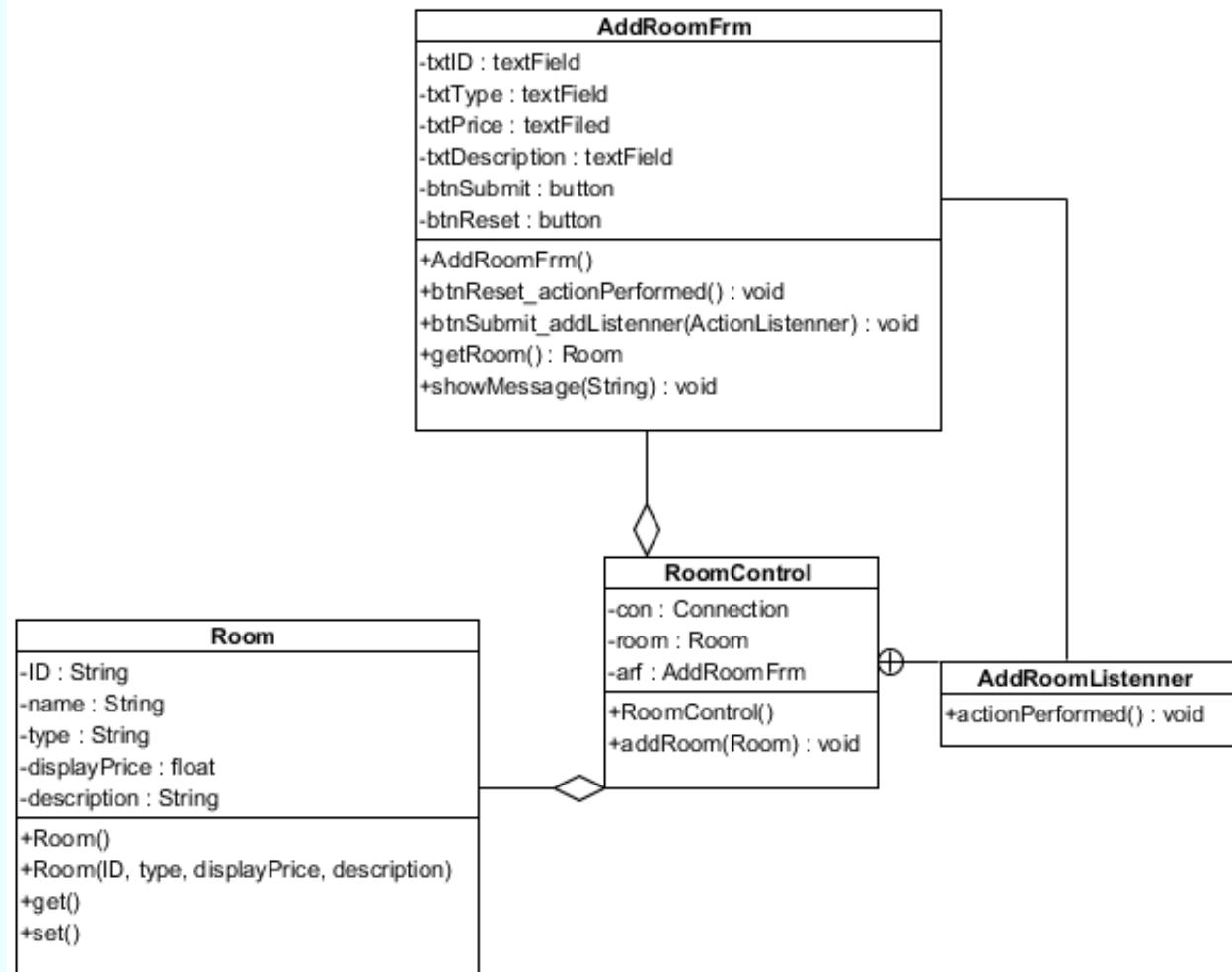
# Ví dụ Cài đặt theo thiết kế MVC cải tiến

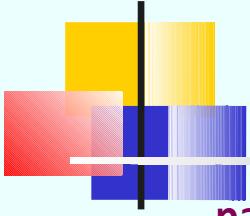
*Tham khảo chi tiết tại:*  
<http://coderandcode.blogspot.com/2014/02/variations-of-mvc-model.html>



# Sơ đồ lớp pha thiết kế

Với chức năng thêm phòng:





# Lớp Room (1)

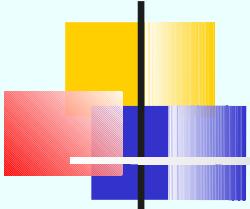
---

```
package mvcNew;

public class Room {
    private String id;
    private String name;
    private String type;
    private float displayPrice;
    private String description;

    public Room() {
        super();
    }

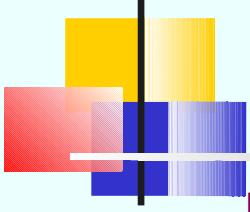
    public Room(String id, String name, String type,
               float displayPrice, String description) {
        super();
        this.id = id;
        this.name = name;
        this.type = type;
        this.displayPrice = displayPrice;
        this.description = description;
    }
}
```



# Lớp Room (2)

---

```
public String getId() {
    return id;
}
public void setId(String id) {
    this.id = id;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
public String getType() {
    return type;
}
public void setType(String type) {
    this.type = type;
}
public float getDisplayPrice() {
    return displayPrice;
}
public void setDisplayPrice(float displayPrice) {
    this.displayPrice = displayPrice;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}
```



# Lớp AddRoomFrm (1)

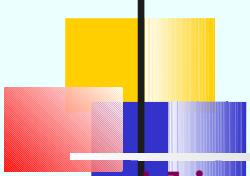
---

```
package mvcNew;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class AddRoomFrm extends JFrame implements ActionListener{
    private JTextField txtID;
    private JTextField txtName;
    private JTextField txtType;
    private JTextField txtDisplayPrice;
    private JTextField txtDescription;
    private JButton btnSubmit;
    private JButton btnReset;
```



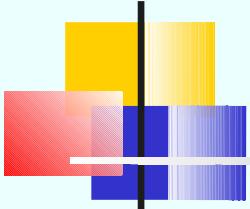
# Lớp AddRoomFrm (2)

---

```
public AddRoomFrm(){
    super("Room management pure-MVC");
    txtID = new JTextField(15);
    txtName = new JTextField(15);
    txtType = new JTextField(15);
    txtDisplayPrice = new JTextField(15);
    txtDescription = new JTextField(15);
    btnSubmit = new JButton("Submit");
    btnReset = new JButton("Reset");

    JPanel content = new JPanel();
    content.setLayout(new GridLayout(6,2));
    content.add(new JLabel("ID:")); content.add(txtID);
    content.add(new JLabel("Name:")); content.add(txtName);
    content.add(new JLabel("Type:")); content.add(txtType);
    content.add(new JLabel("Display price:")); content.add(txtDisplayPrice);
    content.add(new JLabel("Description:")); content.add(txtDescription);
    content.add(btnReset); content.add(btnSubmit);
    btnSubmit.addActionListener(this);
    btnReset.addActionListener(this);
    this.setContentPane(content);
    this.pack();

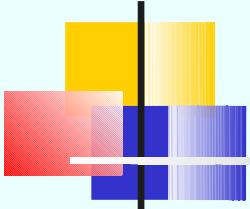
    this.addWindowListener(new WindowAdapter(){
        public void windowClosing(WindowEvent e){
            System.exit(0);
        }
    });
}
```



# Lớp AddRoomFrm (3)

---

```
public void actionPerformed(ActionEvent e) {  
    JButton btn = (JButton) e.getSource();  
    if(btn.equals(btnReset)){  
        btnReset_actionperformed();  
    }  
}  
  
public void btnReset_actionperformed() {  
    txtID.setText("");  
    txtName.setText("");  
    txtType.setText("");  
    txtDisplayPrice.setText("");  
    txtDescription.setText("");  
}
```



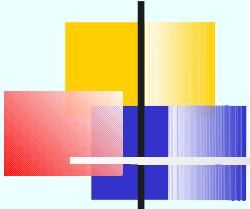
# Lớp AddRoomFrm (4)

---

```
public Room getRoom(){
    Room room = new Room();
    room.setId(txtID.getText());
    room.setName(txtName.getText());
    room.setType(txtType.getText());
    room.setDisplayPrice(
        Float.parseFloat(txtDisplayPrice.getText()));
    room.setDescription(txtDescription.getText());
    return room;
}

public void showMessage(String msg){
    JOptionPane.showMessageDialog(this, msg);
}

public void addSubmitListener(ActionListener log) {
    btnSubmit.addActionListener(log);
}
```



# Lớp RoomControl (1)

---

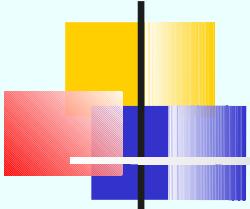
```
package mvcNew;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class RoomControl {
    private Connection con;
    private Room room;
    private AddRoomFrm arf;

    public RoomControl(){
        String dbUrl = "jdbc:mysql://localhost:3306/hotel";
        String dbClass = "com.mysql.jdbc.Driver";
        try {
            Class.forName(dbClass);
            con = DriverManager.getConnection (dbUrl, "root", "12345678");
        }catch(Exception e) {
            e.printStackTrace();
        }

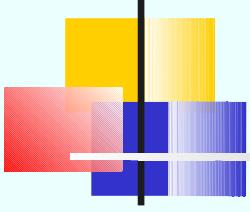
        arf = new AddRoomFrm();
        arf.addSubmitListener(new AddRoomListener());
        arf.setVisible(true);
    }
}
```



# Lớp RoomControl (2)

---

```
public void addRoom(Room room){  
    String sql = "INSERT INTO tblRoom(id, name, type, displayPrice,  
        description) VALUES(?, ?, ?, ?, ?)";  
    try{  
        PreparedStatement ps = con.prepareStatement(sql);  
        ps.setString(1, room.getId());  
        ps.setString(2, room.getName());  
        ps.setString(3, room.getType());  
        ps.setFloat(4, room.getDisplayPrice());  
        ps.setString(5, room.getDescription());  
  
        ps.executeUpdate();  
    }catch(Exception e){  
        e.printStackTrace();  
    }  
}  
  
class AddRoomListener implements ActionListener {  
    public void actionPerformed(ActionEvent e) {  
        try {  
            room = arf.getRoom();  
            addRoom(room);  
            arf.showMessageDialog("Add room successfully!");  
        } catch (Exception ex) {  
            ex.printStackTrace();  
        }  
    }  
}
```



# Lớp Test

---

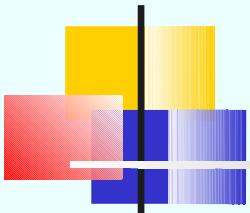
```
package mvcNew;

public class Test {

    public static void main(String[] args) {
        RoomControl rc = new RoomControl();
    }
}
```

Lưu ý trước khi chạy, phải:

- Cài đặt CSDL và bật server MySQL
- Add driver của Jdbc mysql vào library của project

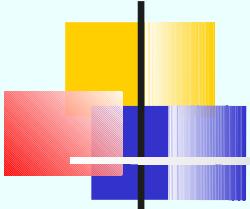


# Bài tập

---

Cài đặt theo kiến trúc đã thiết kế các modul sau:

- Chức năng sửa thông tin phòng
- Chức năng xóa thông tin phòng
- Chức năng đặt phòng
- Chức năng checkin
- Chức năng trả phòng và thanh toán

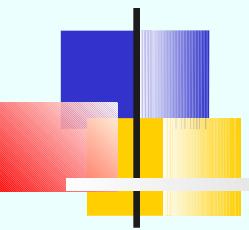


# Bài tập về nhà

---

Cài đặt modul cá nhân theo kiến trúc đã thiết kế:

- Trình bày sơ đồ lớp đã thiết kế
- Trình bày sơ đồ cơ sở dữ liệu đã thiết kế
- Cài đặt các lớp theo đúng thiết kế
- Demo chương trình



# Questions?

---