Phân tích thiết kế hướng đối tượng

Bài 6: Phân tích đối tượng

TS. Nguyễn Hiếu Cường
Bộ môn CNPM, Khoa CNTT, Trường ĐH GTVT
Email: cuonggt@gmail.com

Hà Nội, 2025

Phân tích hệ thống

- Mô hình ca sử dụng
  - Hệ thống cần làm gì (dưới góc nhìn của người dùng)
  - Các chức năng được yêu cầu của hệ thống
- Nhưng, phần mềm bao gồm các đối tượng

- Các đối tượng được xác định từ đâu?

Vai trò của phân tích đối tượng

- Là cầu nối giữa phân tích yêu cầu và thiết kế
- Giúp hiểu rõ:
  - Những thực thể nghiệp vụ nào cần quản lý
  - Chúng có quan hệ với nhau ra sao
- Tạo nền cho xây dựng biểu đồ lớp phân tích
- Mô hình phân tích: tập trung vào khái niệm nghiệp vụ

Đầu vào/đầu ra của phân tích đối tượng

- Đầu vào
  - Mô tả use case, các kịch bản (basic/alternative flows)
  - Thuật ngữ nghiệp vụ từ các bên (stakeholders)
  - Tài liệu hiện có (biểu mẫu, báo cáo, quy trình hiện hành)
- Đầu ra
  - Danh sách ban đầu các đối tượng tiềm năng (candidate objects)
  - Đặc tả sơ bộ về các đối tượng đó (tên, ý nghĩa)
  - Biểu đồ lớp phân tích

Ví dụ

- Suzy có:
  - Một chuồng để nuôi các con vật làm cảnh
  - Chuồng gồm nhiều ngăn, mỗi ngăn chỉ nuôi 1 con vật (chó, mèo)
- Suzy cần một chương trình để quản lý các con vật trên
- Chương trình có các nhiệm vụ? (use cases)
  - *Thêm/bớt* một con vật vào các ngăn trong chuồng
  - *Xác định* con vật trong mỗi ngăn: tên gì, loại gì, "kêu" thế nào?
  - *Mở rộng*, sửa chữa một cách dễ dàng

Phân tích

- Có những đối tượng gì trong bài toán?
  - Các con vật → Animal, Cat, Dog, ...
  - Cái chuồng → Kennel
- Các lớp cần có (giả thiết Suzi chỉ nuôi mèo và chó)?
  - Animal Mô tả các đặc tính chung của các loại con vật
  - Cat Mô tả loại đối tượng con mèo, kế thừa lớp Animal
  - Dog Mô tả loại đối tượng con chó, kế thừa lớp Animal
  - Kennel Chuồng nuôi các con vật, gồm nhiều ngăn

Thiết kế

- Lớp Animal

  → Các đặc tính chung của một con vật, là cơ sở cho Cat và Dog

- Lớp Cat

  → Các đặc tính riêng của con mèo, ví dụ: tiếng "kêu" (kiểu của mèo)

- Lớp Dog

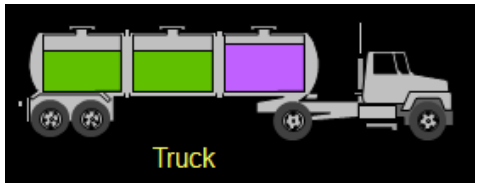
  → Các đặc tính riêng của con chó, ví dụ: tiếng "kêu" (kiểu của chó)

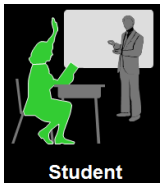
- Lớp Kennel

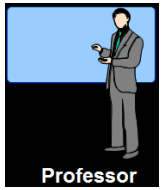
  → Các thao tác "thêm", "bớt" một con vật; "Liệt kê" danh sách

Đối tượng

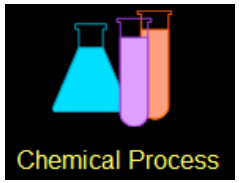
- Thế giới thực bao gồm các *đối tượng* (object)!
  - Đối tượng vật lý



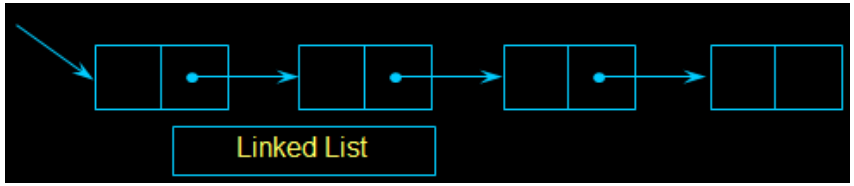




  - Đối tượng khái niệm



  - Đối tượng phần mềm



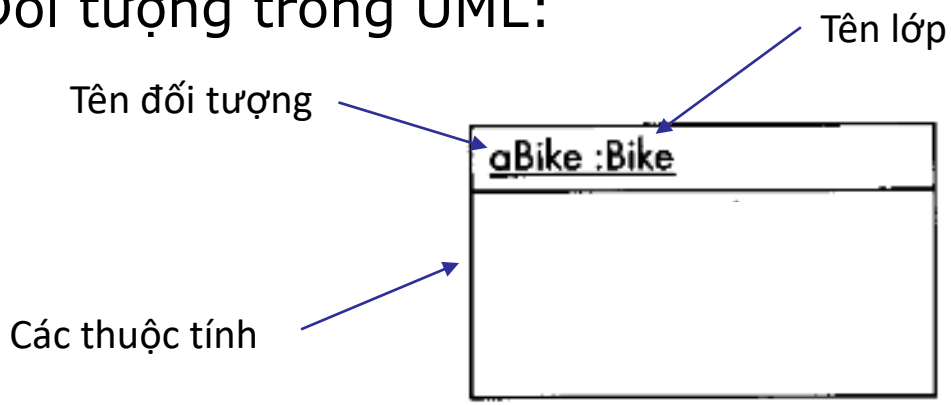
- Mỗi đối tượng gồm các *thuộc tính* và các *thao tác*

Lớp

- Lớp (class) là định nghĩa trừu tượng (abstract definition) của các đối tượng có cùng những đặc tính chung
  - Đối tượng là thể hiện cụ thể (instance) của một lớp
- Lớp có tác dụng
  - Trừu tượng hoá dữ liệu (data abstraction)
  - Bao gói thông tin (encapsulation)
  - Che giấu thông tin (information hiding)

Đối tượng

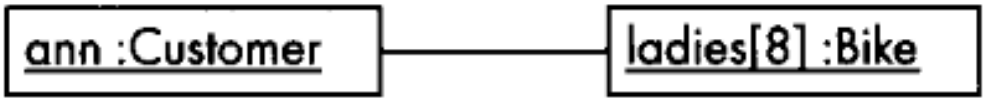
- Một đối tượng gồm những gì?
  - Trạng thái (state): định hình bởi giá trị các thuộc tính của đối tượng
  - Ứng xử (behaviour): thể hiện bởi các hành động có thể của đối tượng
  - Định danh (identity): mỗi đối tượng là duy nhất trong bộ nhớ
- Đối tượng trong UML:



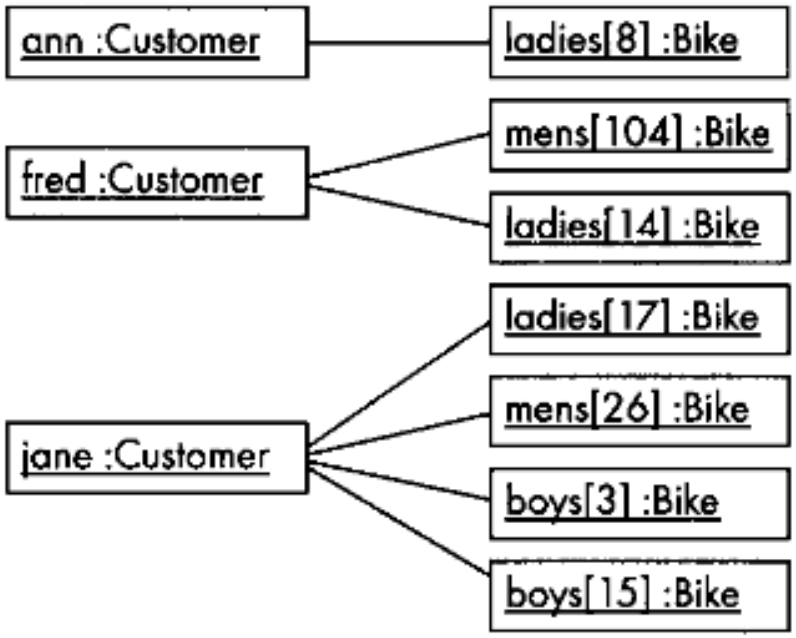
Sự phụ thuộc giữa các đối tượng

- Đối tượng A phụ thuộc vào đối tượng B

- Ví dụ: Khách hàng tên **ann** thuê xe đạp **ladies[8]**

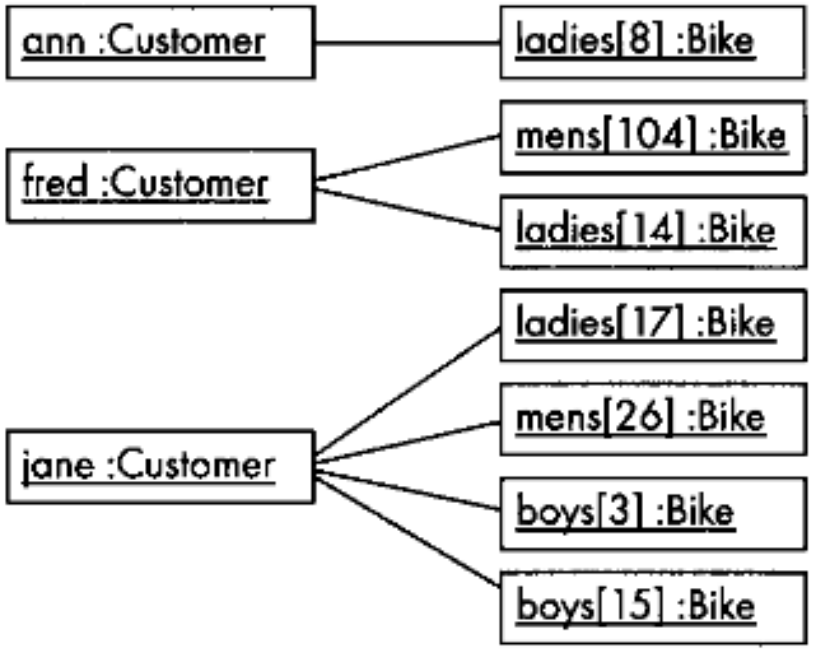


Sơ đồ đối tượng (Object diagram)



Ưu nhược điểm

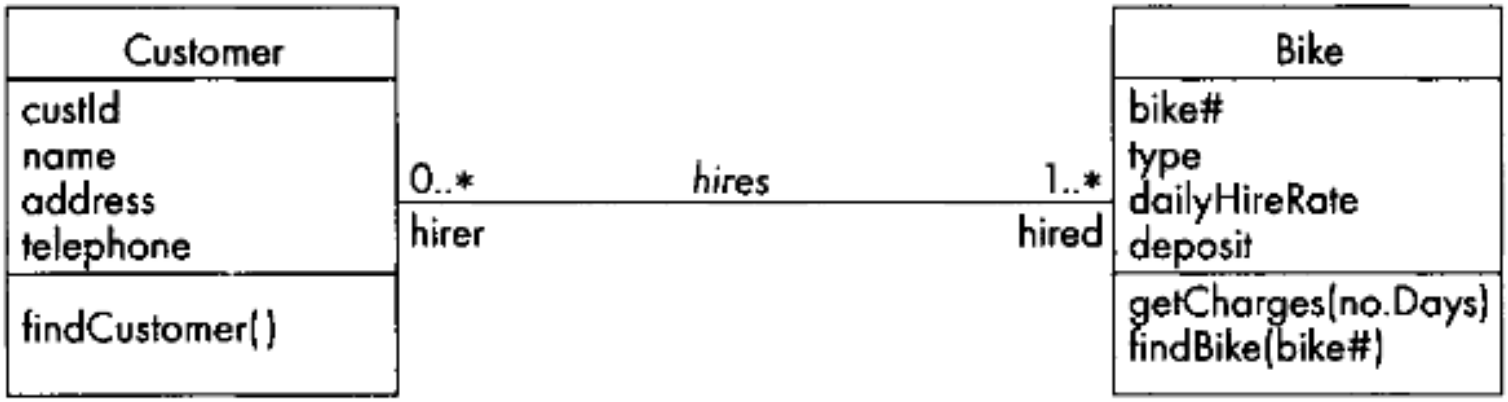
- Ưu điểm
  - Mô tả quan hệ giữa các đối tượng
  - Làm rõ hơn các hệ thống phức tạp **tại một thời điểm**
- Nhược điểm?



Sơ đồ lớp (Class diagram)

Liên kết giữa các lớp

- Lớp A liên kết với lớp B = các đối tượng của A <u>có thể tương tác</u> được với các đối tượng của B



0..*	multiplicity
hires	association name
hirer, hired	role name

Các kiểu liên kết

- Có nhiều kiểu liên kết giữa các lớp:

1. **Kết hợp** (association)
   - Là liên kết căn bản và phổ biến nhất
2. **Kết tập** (aggregation)
   - Liên kết "tổng thể - thành phần"
3. **Gộp** (composition)
   - Liên kết "tổng thể - thành phần", nhưng "chặt" hơn
4. **Kế thừa** (inheritance)
   - Liên kết "là một" hoặc "là một loại"

Kết hợp (Association)

- Các đối tượng của hai lớp có thể tương tác với nhau
- Tên liên kết có thể được ghi rõ (nếu cần)

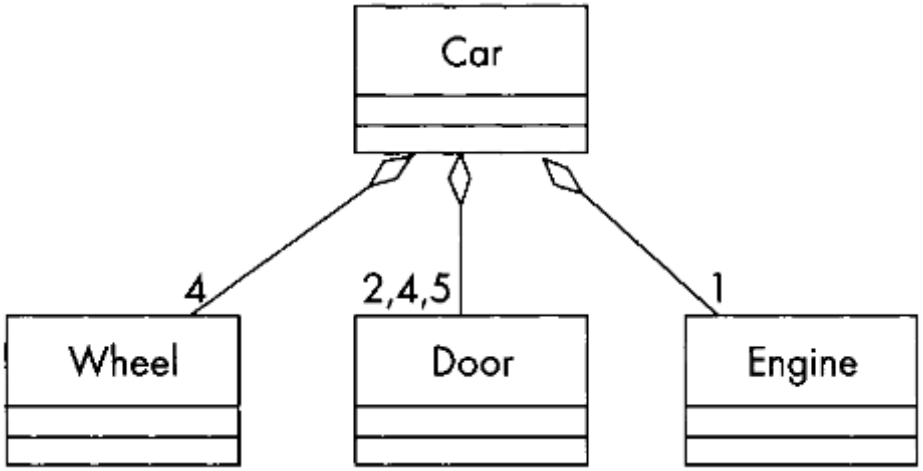
Liên kết bội (Multiplicity)

- Xác định giới hạn số đối tượng có thể tham gia vào liên kết

Meaning	*Example*	*Notation*
an exact number	exactly one	1 (or may be omitted)
	exactly six	6
many	zero or more	0..*
	one or more, lots of	1..*, *
a specific range	one to four, zero to six	1..4, 0..6,
a choice	two or four or five	2, 4, 5

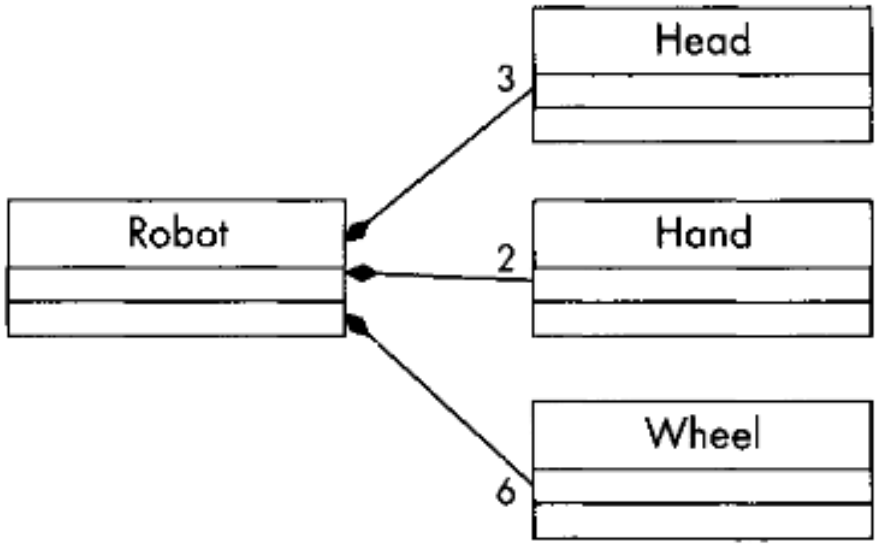
Kết tập (Aggregation)

- Thể hiện quan hệ "tổng thể - thành phần" (whole-part)
  - Trong mô tả thường có các cụm từ: "là một phần của" (is a part of) hoặc "gồm có" (consist of)

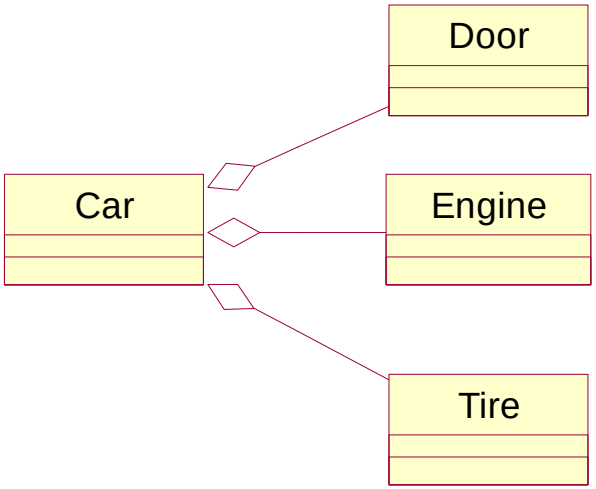


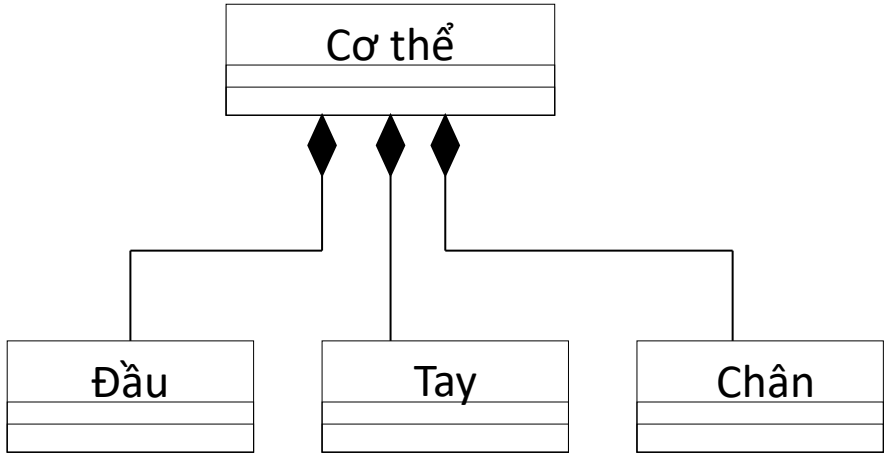
Gộp (Composition)

- Là một dạng quan hệ "tổng thể - thành phần"
  - "Chặt" hơn quan hệ kết tập: đối tượng thành phần không thể tồn tại độc lập với đối tượng tổng thể



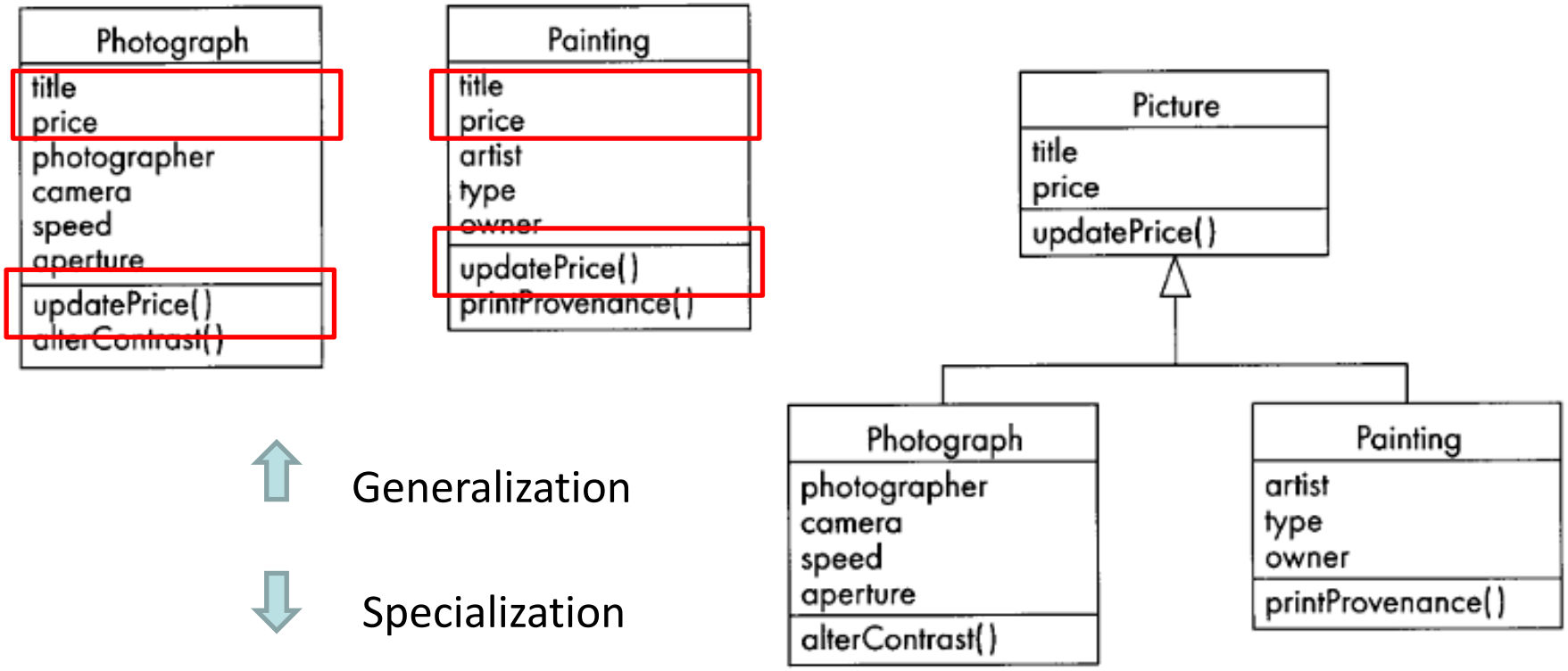
Cài đặt các quan hệ kết tập và gộp





Kế thừa

- Là quan hệ "là một" (is a) hoặc "là một loại" (is a kind of)

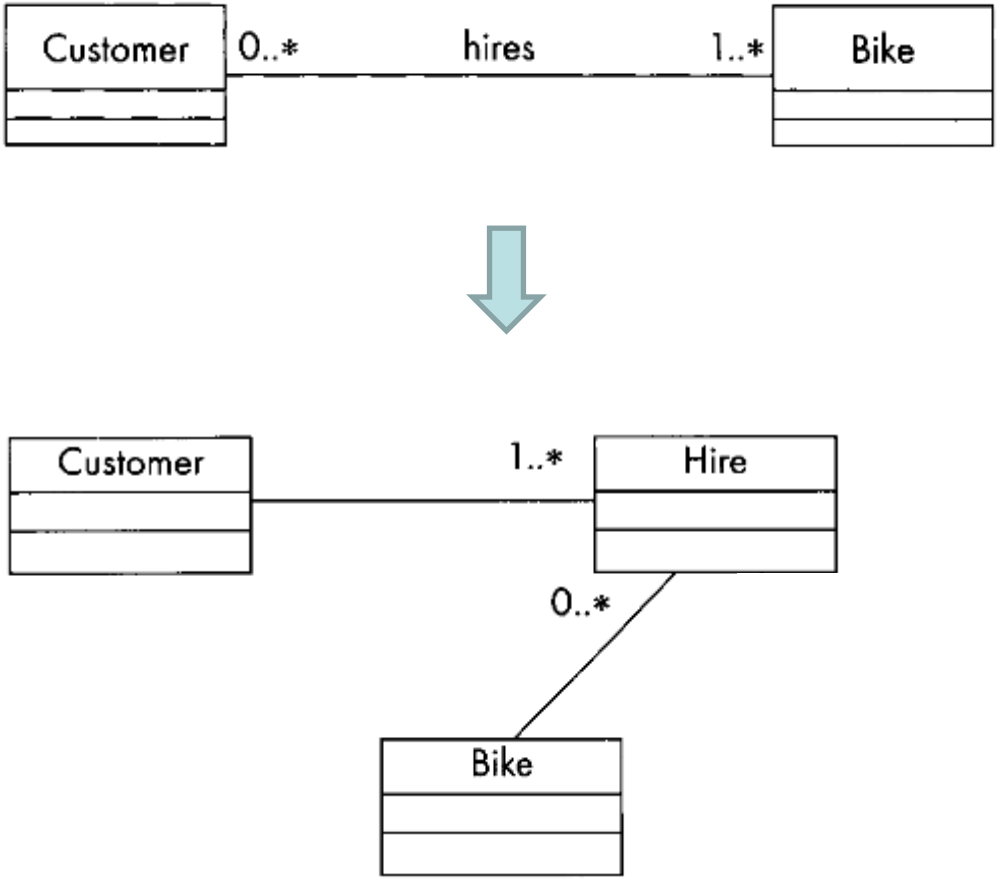


Lớp kết hợp

- Ví dụ:

- Có nhận xét gì về liên kết "hires"?
  - Liên kết "hires" bao gồm nhiều thuộc tính (startDate, endDate)
  - Đặt các thuộc tính này ở Customer hay Bike đều có sự bất tiện
  - Trong trường hợp này nên chuyển nó thành một *lớp kết hợp*
- Lớp kết hợp (association class)

Lớp kết hợp



Mục tiêu của phân tích đối tượng

- Xác định các lớp khái niệm trong miền bài toán (Concepts)
- Ghi nhận các thuộc tính (attributes) quan trọng
- Xác định quan hệ giữa các khái niệm
- Chưa quan tâm tới UI, database, hay coding chi tiết
  - Tránh "Thiết kế sớm" (Premature Design)

Thế nào là một lớp tốt?

1. **Problem domain**: Các lớp nên phản ánh đúng đối tượng trong phạm vi bài toán

   Ví dụ: Nếu đang phân tích hệ thống thư viện, lớp Book hay Member là tốt, nhưng lớp System hay LibraryManagement lại quá chung chung

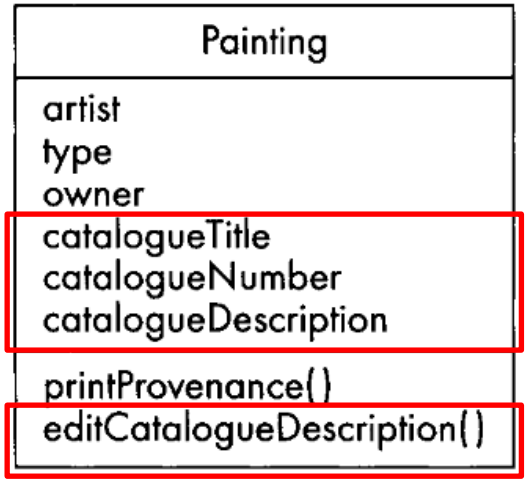
2. **Functionality**: Một lớp cần có cả dữ liệu và hành vi, tránh lớp chỉ toàn dữ liệu hoặc chỉ toàn hành vi

   Ví dụ: Phải có dữ liệu gắn với hành vi xử lý dữ liệu đó → tránh biến lớp thành "struct dữ liệu" hoặc "tập hợp các hàm" rời rạc

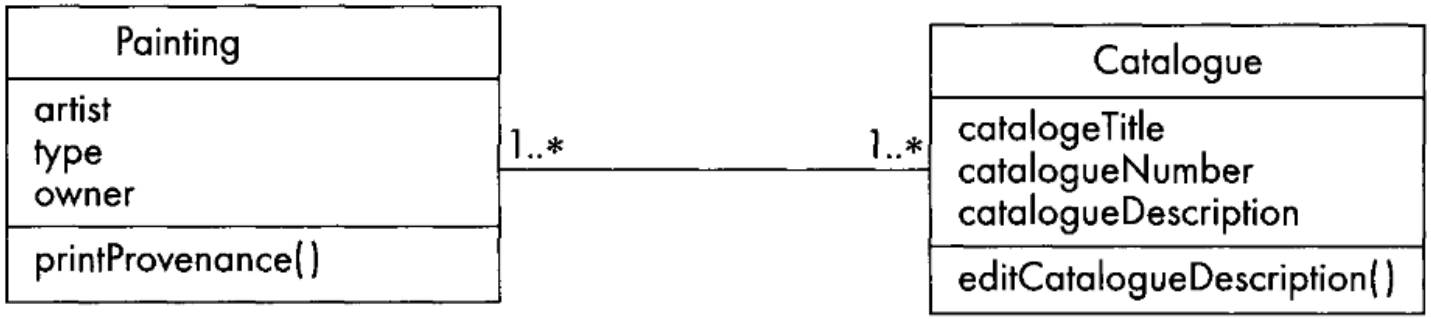
3. **Cohesion**: Lớp phải có tính cố kết cao (high cohesion), nên tập trung vào chỉ một trách nhiệm chính

Ví dụ

- Lớp sau có tốt không?



- Lớp trên không có tính cố kết cao! Cần chỉnh lại?



Câu hỏi

1. Sự khác nhau giữa đối tượng và lớp? Ký hiệu UML của chúng?
2. Các đặc trưng của một đối tượng là gì?
3. Các đối tượng giao tiếp với nhau như thế nào?
4. Hãy trình bày về 4 loại liên kết giữa các lớp?
5. Trình bày về một số tiêu chuẩn để coi một lớp là tốt?
6. Cho use case *Borrow Book*, hãy vẽ conceptual class diagram. Có thể thêm `BorrowForm` hoặc `LoanController` vào diagram không?

   `BorrowForm` và `LoanController` không phải là lớp domain → Premature design

Bài tập

1. Hãy tìm các định nghĩa phù hợp cho mỗi khái niệm *(Concept)*

Concept	*Definition*
aggregation	1 a relationship between two classes where one is a specialization of another
association	2 the ability of one operation to be implemented by different methods
attribute	3 abstracting common features into a superclass
class	4 code implementing an operation
data hiding	5 concealing internal details of an object
encapsulation	6 creation of an object
generalization	7 data item defined as part of a class or object
inheritance	8 instance of a class
instatiation	9 interface of a method
message	10 packaging together data and operations
method	11 relationship between classes
object	12 request for a service to be executed
operation	13 template for objects
polymorphism	14 whole-part relationship

Bài tập

2. Căn cứ vào tính chất của các khái niệm trong thực tế, hãy xác định quan hệ giữa các lớp trong từng trường hợp sau:

a hotel room, booking, guest

b club member, adult member, junior member

c exam paper, instruction, question, solution

d animal, mammal, bird, reptile, dog, horse, parrot

e sentence, word, letter, punctuation

f academic staff, lecturer, professor, student.