Phân tích thiết kế hướng đối tượng

Bài 12: Thiết kế tổng thể và chi tiết

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

Hà Nội, 2018

Phân tích và Thiết kế

- Phân tích
  - Những yêu cầu phải thực hiện – What?
  - Chủ yếu quan tâm đến các yêu cầu chức năng
  - Độc lập với cài đặt (implementation independent)
- Thiết kế
  - Làm sao thực hiện được các yêu cầu chỉ ra ở bước phân tích – How?
  - Quan tâm cả yêu cầu chức năng và phi chức năng
  - Phụ thuộc cài đặt: architecture, user interface, database...

Thiết kế

- Lựa chọn chiến lược thiết kế
  - Tự phát triển hoàn toàn (Customdevelopment)
  - Mua một phần (Packaged software)
  - Thuê ngoài (Outsourcing)
- Thiết kế tổng thể
  - Kiến trúc hệ thống: phân tầng, phân gói...
- Thiết kế chi tiết
  - Thiết kế chi tiết các lớp
  - Thiết kế giao diện, thiết kế cơ sở dữ liệu...

Tự phát triển

- Ưu điểm
  - Đáp ứng những nhu cầu chuyên biệt
  - Linh hoạt trong giải quyết vấn đề, dễ dàng thay đổi
  - Bảo mật về công nghệ và chiến lược kinh doanh
  - Nâng cao kỹ năng cho nhóm phát triển
- Nhược điểm
  - Có thể vượt quá nguồn lực cho phép
  - Có thể không hiệu quả: “re-invent the wheel”...

Mua một phần

- Ưu điểm
  - Có thể hoàn thiện nhanh hơn, được test tốt hơn...
  - Sử dụng các thành phần đã có sẵn
- Nhược điểm
  - Phải chấp nhận những gì đã được cung cấp
  - Không linh hoạt khi cần thay đổi
  - Vấn đề tích hợp hệ thống

Thuê ngoài

- Tại sao thuê ngoài?
  - Chúng ta không có đủ kỹ thuật, kinh nghiệm...
  - Đa dạng hóa sản phẩm
  - Có thể đảm bảo thời gian, chất lượng hơn
- Những vấn đề cần lưu ý?
  - Hiểu rõ vấn đề mình cần thuê ngoài
  - Lựa chọn nhà cung cấp
  - Khả năng tích hợp hệ thống
  - Bảo mật thông tin: công nghệ, chiến lược kinh doanh…

Lựa chọn chiến lược thiết kế

	Use Custom Development When…	Use a Packaged System When…	Use Outsourcing When…
Business Need	The business need is unique.	The business need is common.	The business need is not core to the business.
In-house Experience	In-house functional and technical experience exists.	In-house functional experience exists.	In-house functional or technical experience does not exist.
Project Skills	There is a desire to build in-house skills.	The skills are not strategic.	The decision to outsource is a strategic decision.
Project Management	The project has a highly skilled project manager and a proven methodology.	The project has a project manager who can coordinate the vendor's efforts.	The project has a highly skilled project manager at the level of the organization that matches the scope of the outsourcing deal.
Time frame	The time frame is flexible.	The time frame is short.	The time frame is short or flexible.

Kiến trúc hệ thống

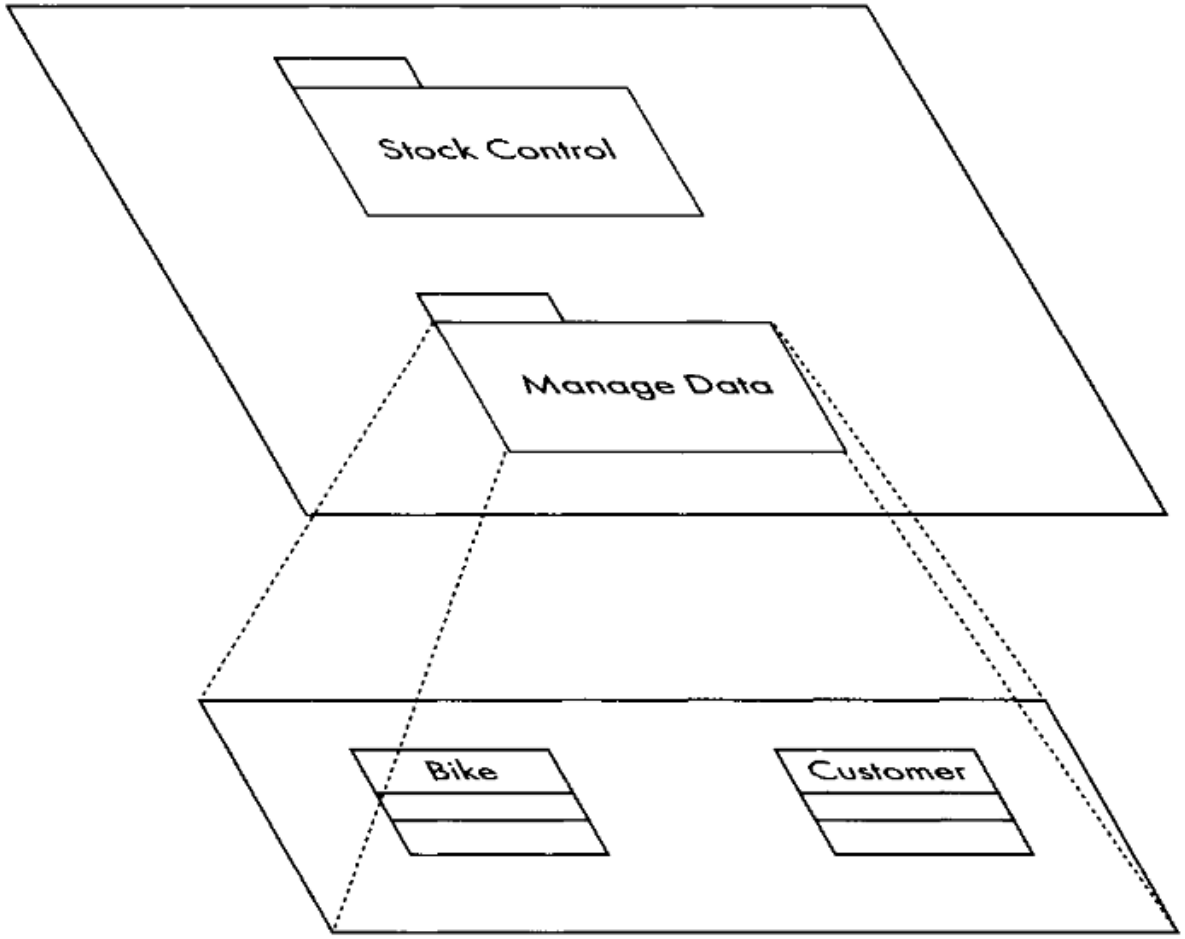
- Kiến trúc logic của phần mềm (Logical software architecture)
  - Class diagram
  - Package diagram
- Kiến trúc vật lý của phần mềm (Physical software architecture)
  - Component diagram
- Kiến trúc phần cứng (Hardware architecture)
  - Deployment diagram

Phân hoạch (partition)

- Hệ thống có thể phân hoạch thành các gói (package)
- Gói dùng để nhóm các phần tử mô hình:
  - Các ca sử dụng (trong biểu đồ ca sử dụng)
  - Các lớp (trong biểu đồ lớp)
  - Các đối tượng (tương tác với nhau trong biểu đồ cộng tác)
- Ví dụ
  - Phân tách hệ thống Wheels thành 2 gói, mỗi gói có các lớp liên quan:
    - Gói Stock Control: Hire, Payment...
    - Gói Manage Data: Bike, Customer...

Ví dụ (phân hoạch thành các gói)

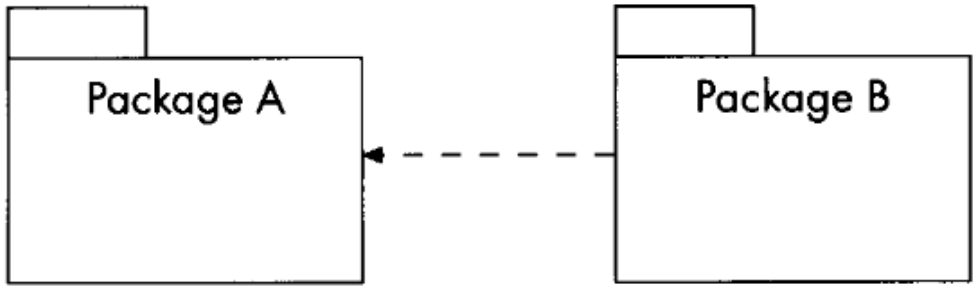
Level 1 Package diagram of subsystem



Level 2 class diagram for Manage Data subsystem

Phụ thuộc giữa các gói

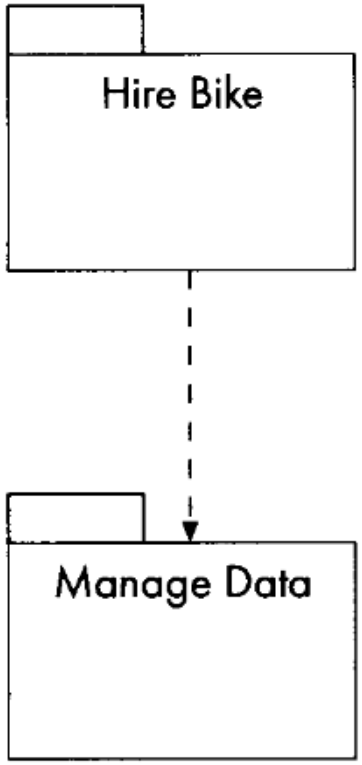
- Phụ thuộc giữa hai gói (dependency) nếu sự thay đổi ở gói này có ảnh hưởng đến gói kia
- Gói B phụ thuộc vào gói A
  - Nếu sự thay đổi ở gói A ảnh hưởng đến gói B
  - Ví dụ: Nếu có một lớp của gói B phụ thuộc vào một lớp của gói A thì gói B phụ thuộc gói A



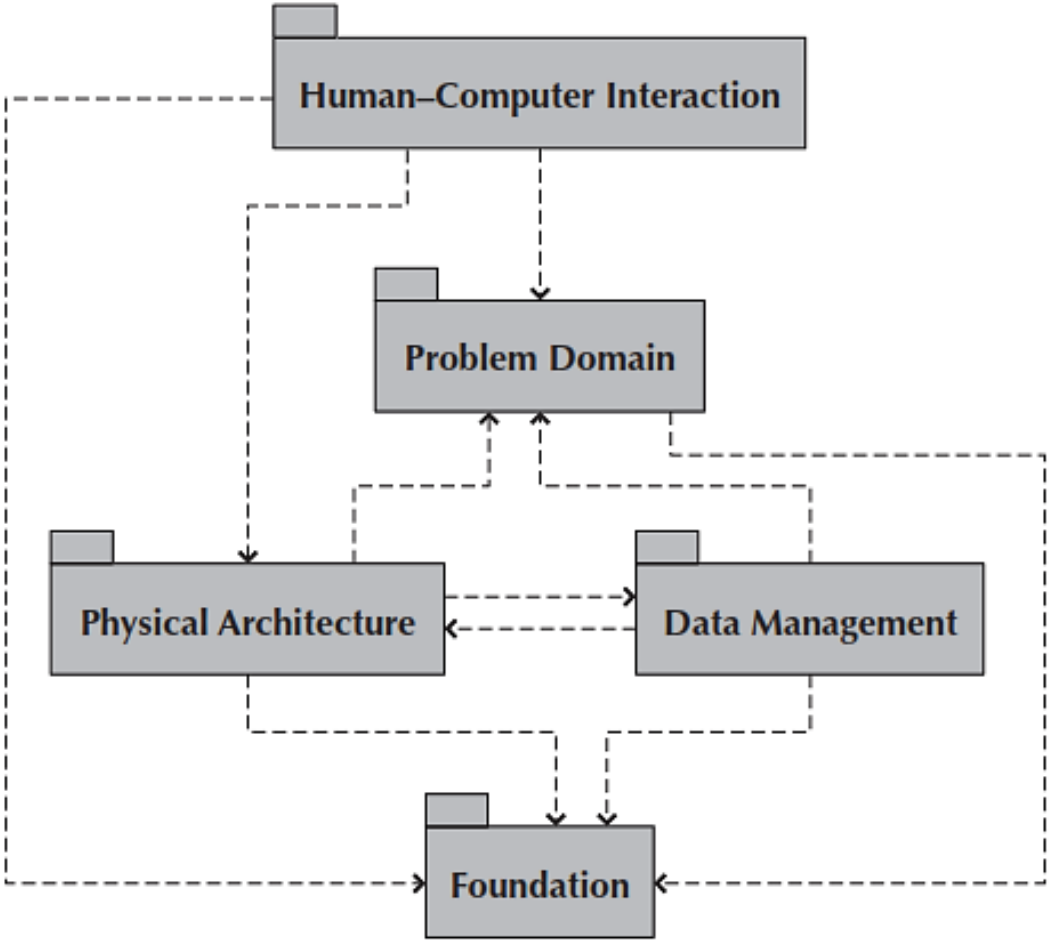
Ví dụ (phụ thuộc)

- Gói Hire Bike có các lớp: Hire và Payment
- Một số lớp trong gói Hire Bike cần sử dụng các dịch vụ của lớp Bike
- Bike là một lớp nằm trong gói Manage Data

➔ Gói Hike Bike phụ thuộc gói Manage Data



Ví dụ (biểu đồ gói)

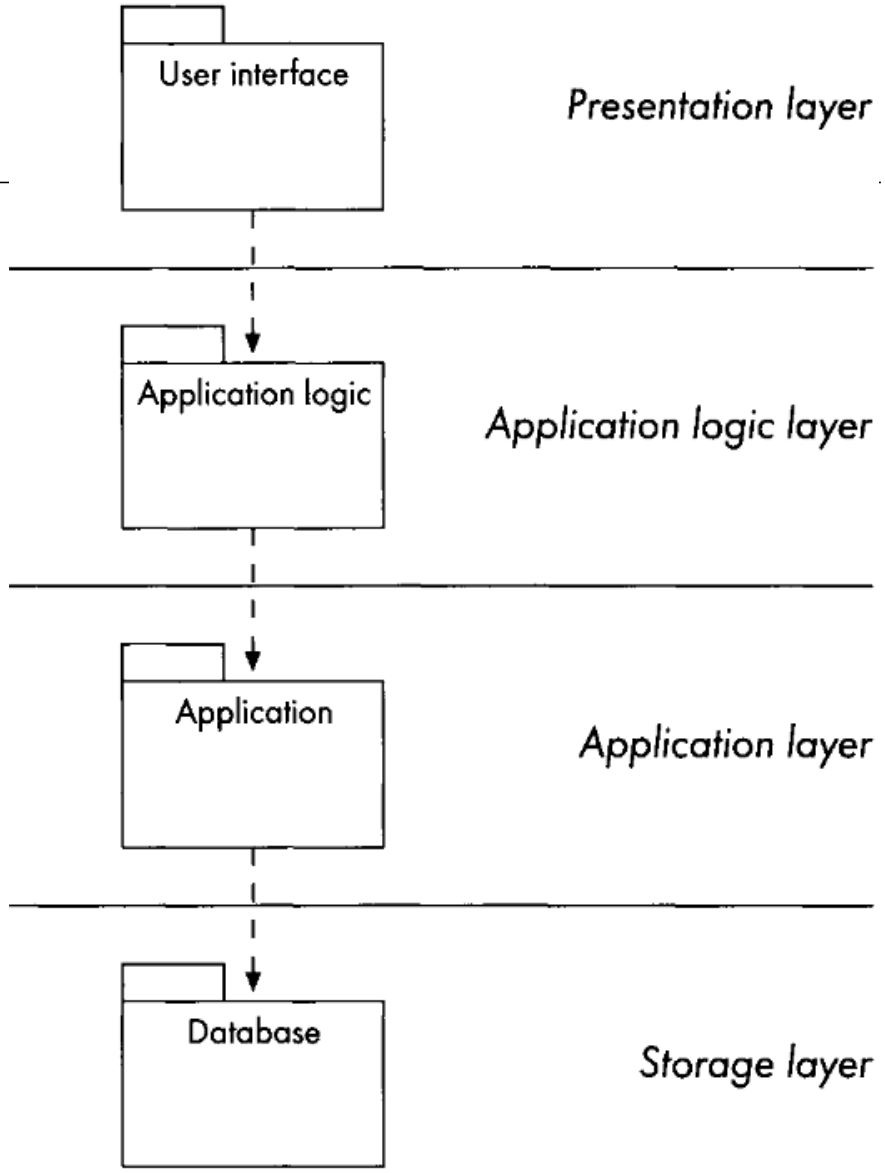


Phân tầng (layering)

- Trong bước phân tích chủ yếu tập trung vào nội dung bài toán (problem domain)
- Chuyển đổi sang mô hình thiết kế cần bổ sung thêm thông tin môi trường hệ thống (data management, user interface…)
- Có thể phân chia các thành phần kiến trúc trên thành các tầng (layers)

Ví dụ

Một kiến trúc 4-tầng



Các loại lớp thiết kế

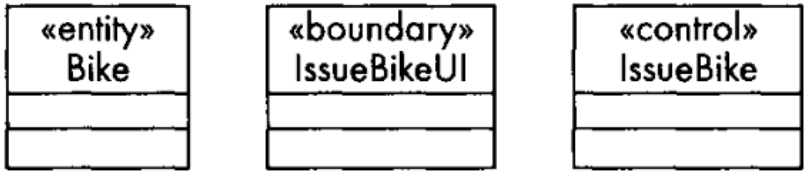
- Phân tích: làm rõ hệ thống cần những gì (what?)
  → Lớp được phát hiện là các lớp để mô tả vấn đề - các lớp thực thể

- Thiết kế: làm sao để hoàn thành những yêu cầu trên (how?)
  → Có thể bổ sung thêm các lớp biên và các lớp điều khiển

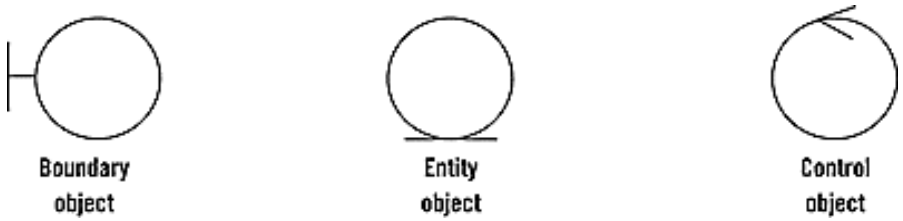
- Cần xác định các lớp biên và lớp điều khiển cho mỗi ca sử dụng
  - Lớp thực thể (entity class): thực thể của hệ thống (Customer, Bike…)
  - Lớp biên (boundary class): giao diện của hệ thống (màn hình, form…)
  - Lớp điều khiển (control class): thực hiện các thao tác kết nối giữa các lớp biên và các lớp thực thể.

Biểu diễn các loại lớp

- Dùng khuôn mẫu (stereotype)



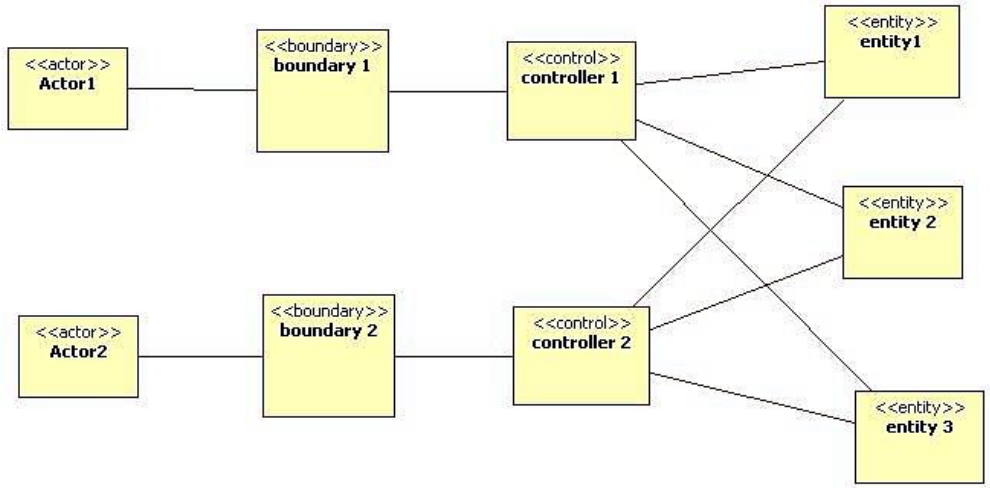
- Dùng ký hiệu

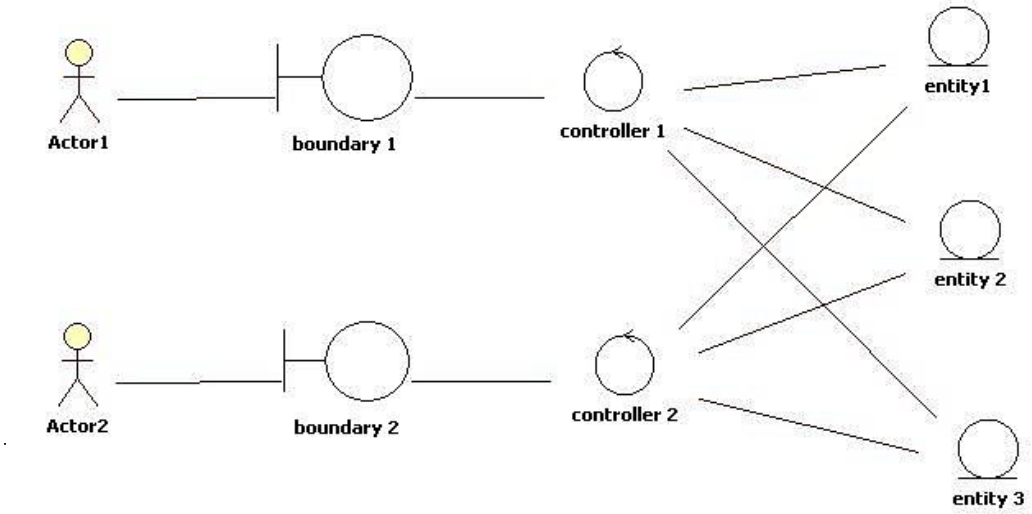


Quy tắc giao tiếp giữa các đối tượng

- Tác nhân chỉ có thể giao tiếp với các đối tượng lớp biên
- Đối tượng lớp biên chỉ có thể giao tiếp với các đối tượng lớp điều khiển và các tác nhân
- Đối tượng lớp điều khiển có thể giao tiếp với các đối tượng lớp biên, lớp thực thể và các đối tượng của điều khiển khác, nhưng không được giao tiếp với các tác nhân
- Đối tượng lớp thực thể chỉ có thể giao tiếp với đối tượng lớp điều khiển

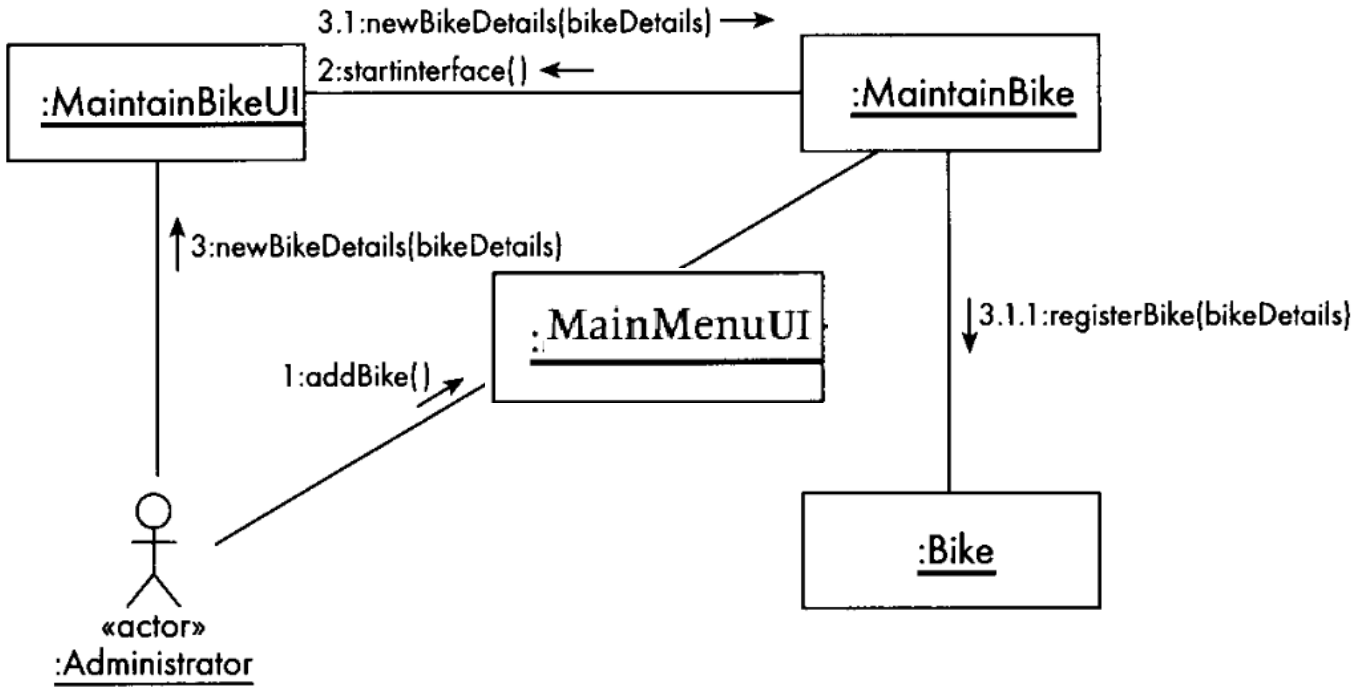
Ví dụ

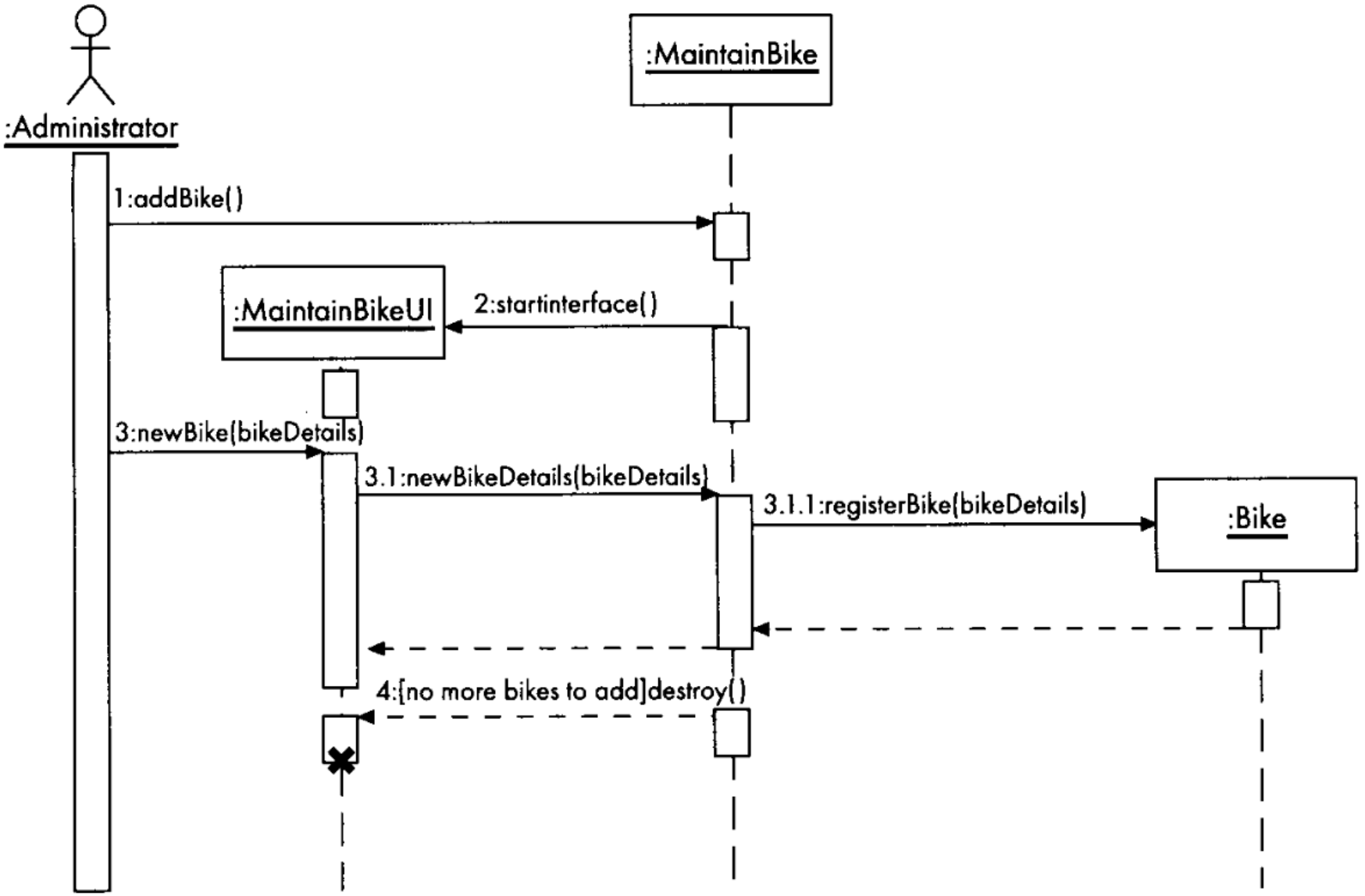

<<actor>>
Actor1
<<boundary>>
boundary 1
<<control>>
controller 1
<<entity>>
entity1
<<entity>>
entity 2
<<actor>>
Actor2
<<boundary>>
boundary 2
<<control>>
controller 2
<<entity>>
entity 3

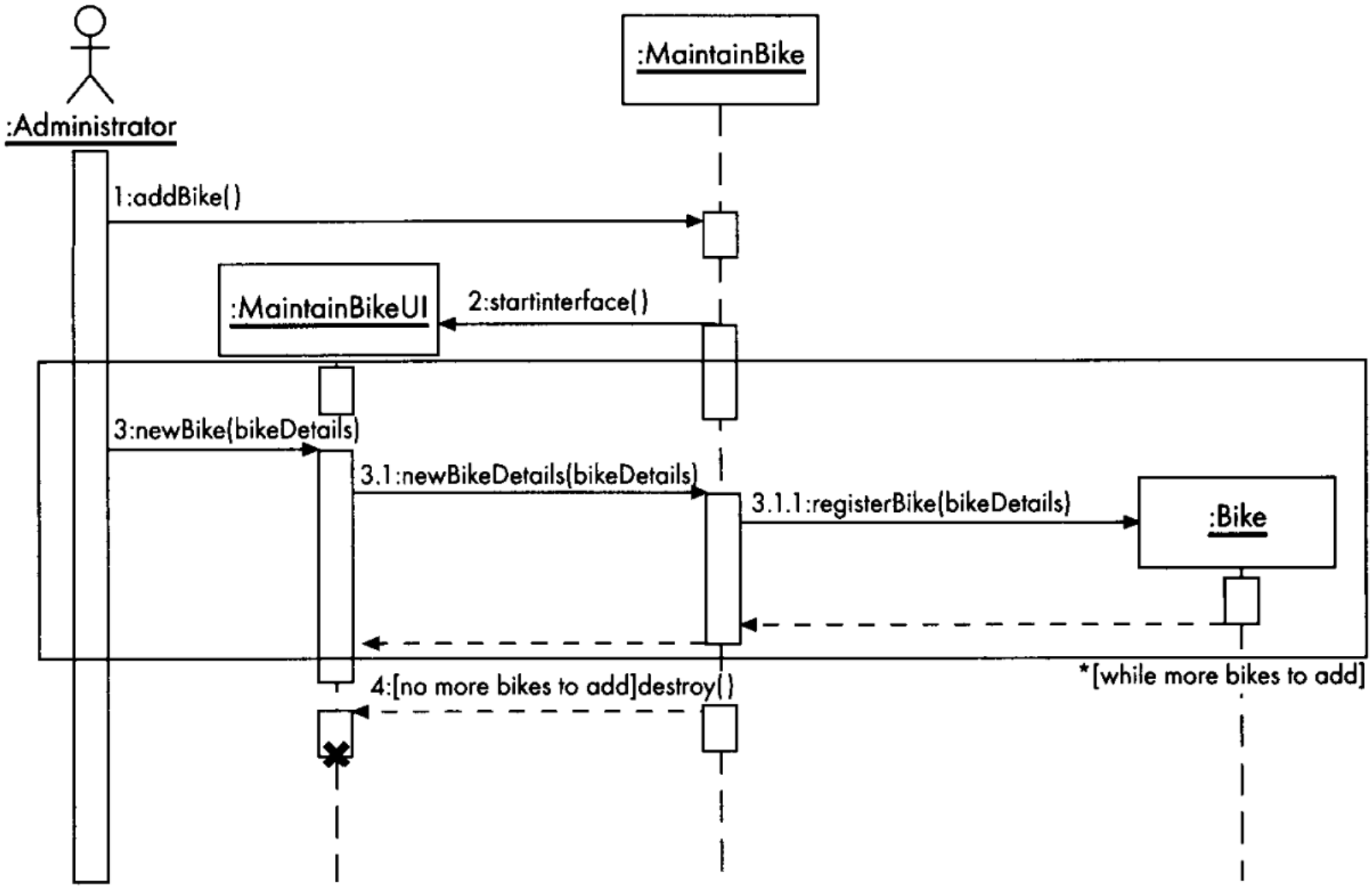

Actor1
boundary 1
controller 1
entity1
entity 2
Actor2
boundary 2
controller 2
entity 3

Ví dụ

- Biểu đồ cộng tác cho ca sử dụng "Add bike"



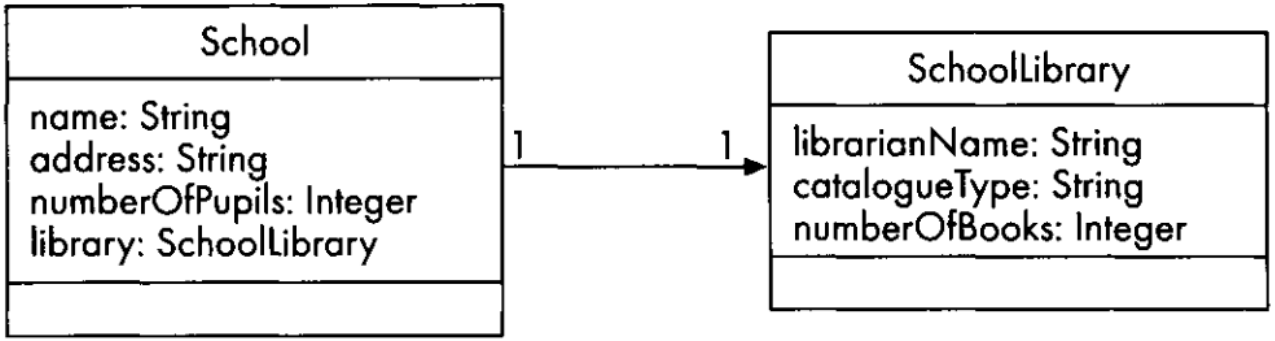

:Administrator
:MaintainBike
1:addBike()
:MaintainBikeUI
2:startinterface()
3:newBike(bikeDetails)
3.1:newBikeDetails(bikeDetails)
3.1.1:registerBike(bikeDetails)
:Bike
4:[no more bikes to add]destroy()


:Administrator
:MaintainBike
1:addBike()
:MaintainBikeUI
2:startinterface()
3:newBike(bikeDetails)
3.1:newBikeDetails(bikeDetails)
3.1.1:registerBike(bikeDetails)
:Bike
*[while more bikes to add]
4:[no more bikes to add]destroy()

Thiết kế các liên kết

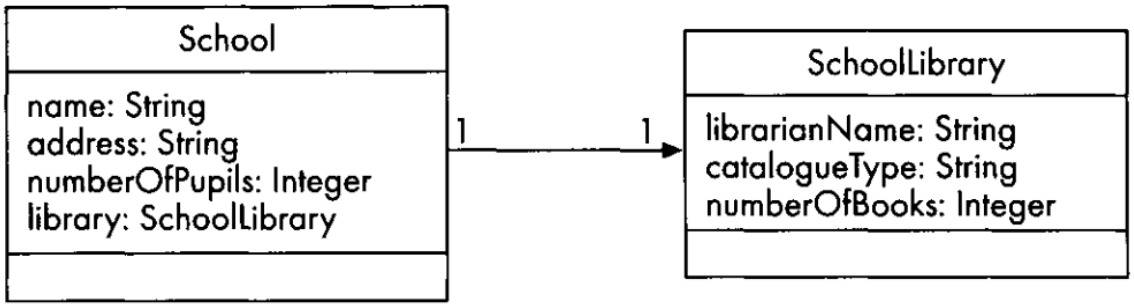
- Những đối tượng nào cần giao tiếp với nhau và hướng thực hiện của các thông điệp
  - Đối tượng School có thể gửi thông điệp tới đối tượng SchoolLibrary

    (Trong lớp School có một thuộc tính kiểu SchoolLibrary)

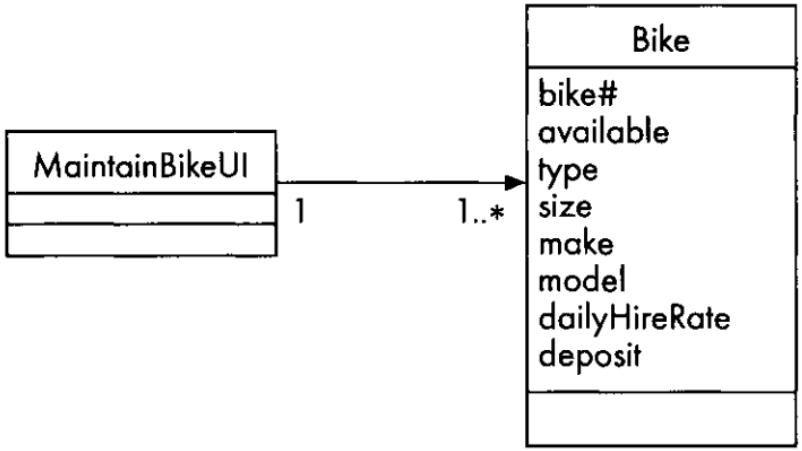


Thiết kế các liên kết

- Liên kết 1 – 1

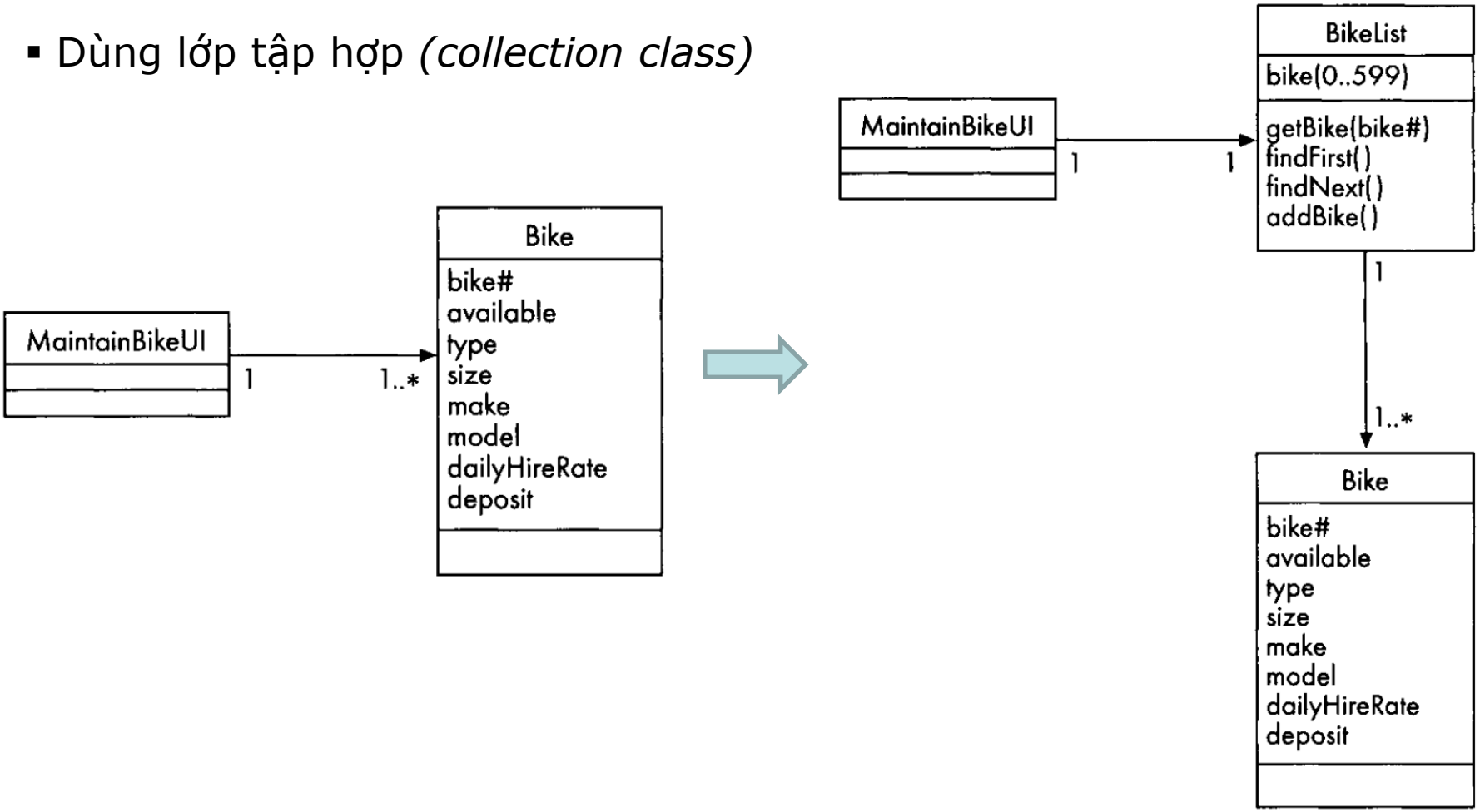


- Liên kết 1 – n?



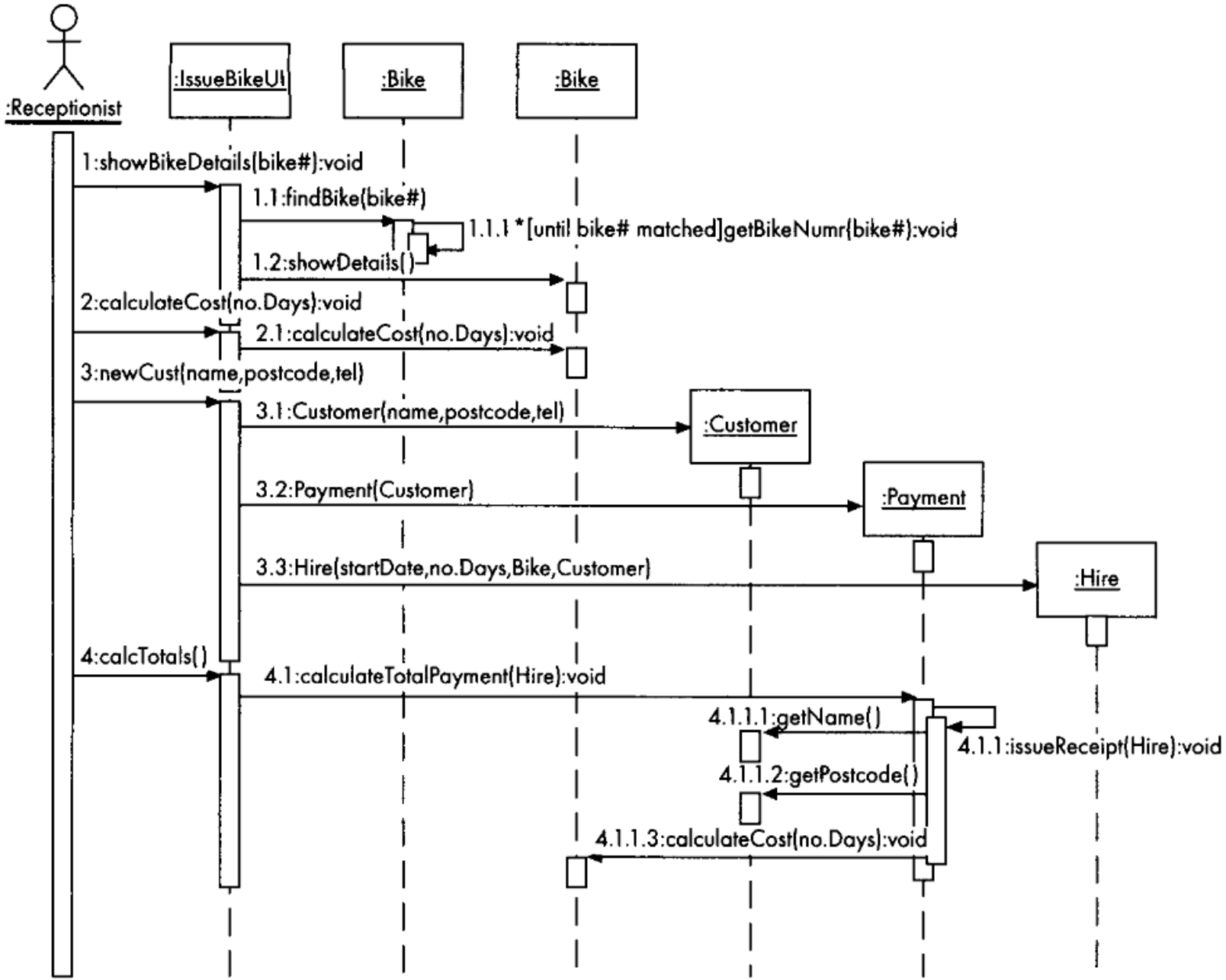
Liên kết 1 – n

- Dùng lớp tập hợp *(collection class)*



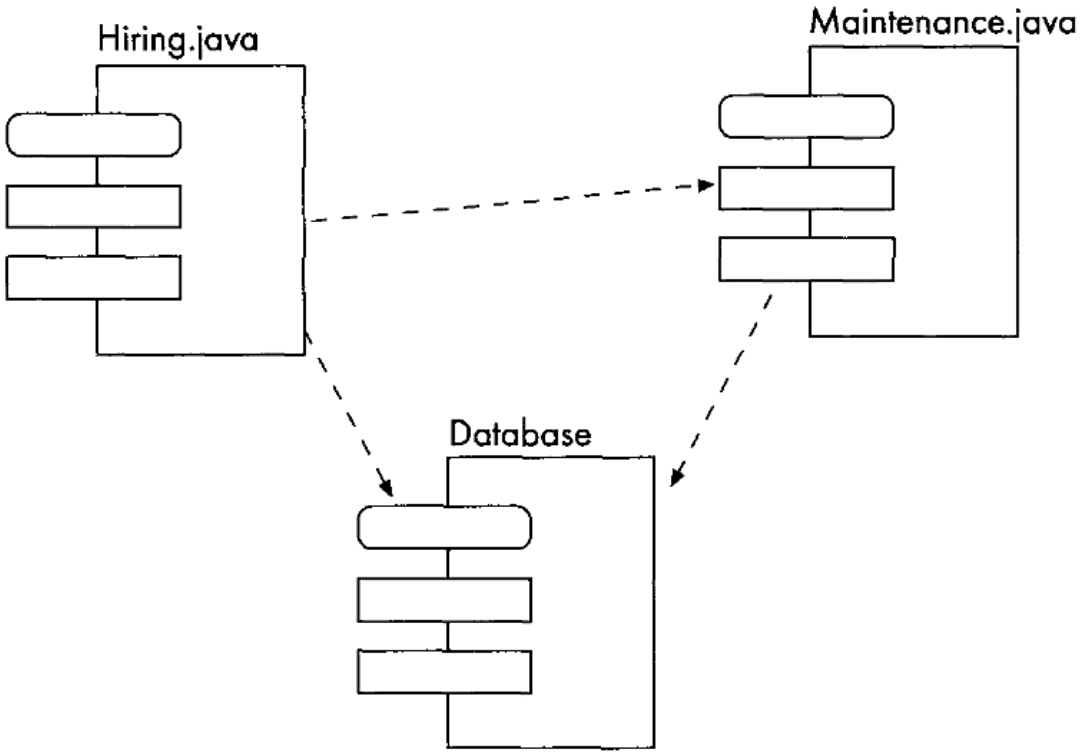
Ví dụ

- Ca sử dụng “Issue bike” (Cho thuê xe) với kịch bản thuê xe thành công…


:Receptionist
:IssueBikeUI
:Bike
:Bike
1:showBikeDetails(bike#):void
1.1:findBike(bike#)
1.1.1 *[until bike# matched]getBikeNumr(bike#):void
1.2:showDetails()
2:calculateCost(no.Days):void
2.1:calculateCost(no.Days):void
3:newCust(name,postcode,tel)
3.1:Customer(name,postcode,tel)
:Customer
3.2:Payment(Customer)
:Payment
3.3:Hire(startDate,no.Days,Bike,Customer)
:Hire
4:calcTotals()
4.1:calculateTotalPayment(Hire):void
4.1.1.1:getName()
4.1.1:issueReceipt(Hire):void
4.1.1.2:getPostcode()
4.1.1.3:calculateCost(no.Days):void

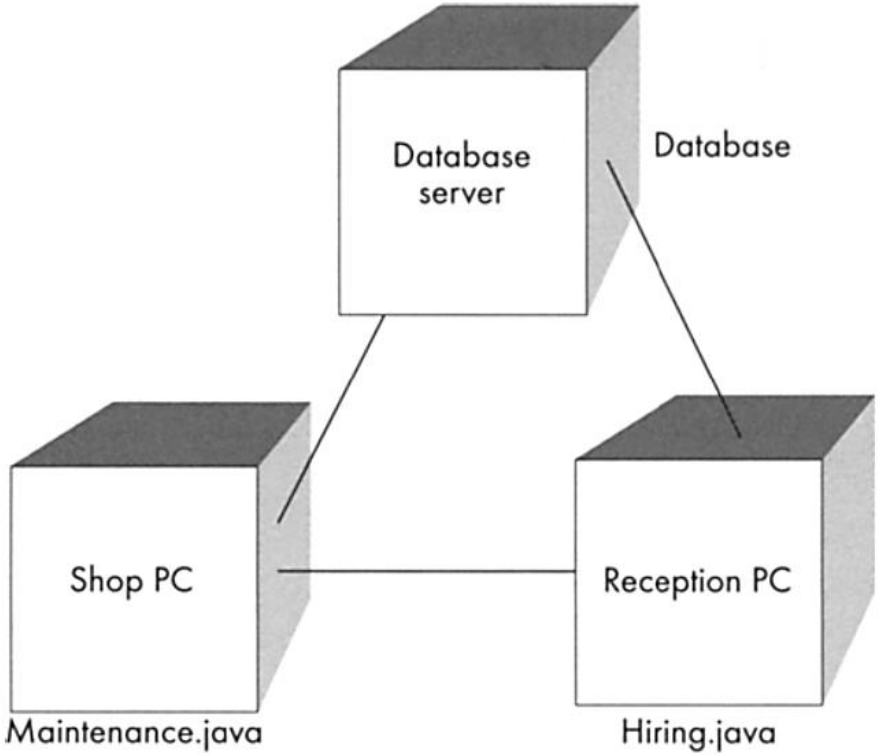
Biểu đồ thành phần (Component diagram)

- Mô tả các thành phần phần mềm và sự phụ thuộc giữa chúng
- Thành phần có thể là các gói hoặc các file (source, binary, data)



Biểu đồ triển khai (Deployment diagram)

- Mô tả sự bố trí các thành phần phần cứng của hệ thống
- Thành phần: Máy chủ, máy trạm, máy in...



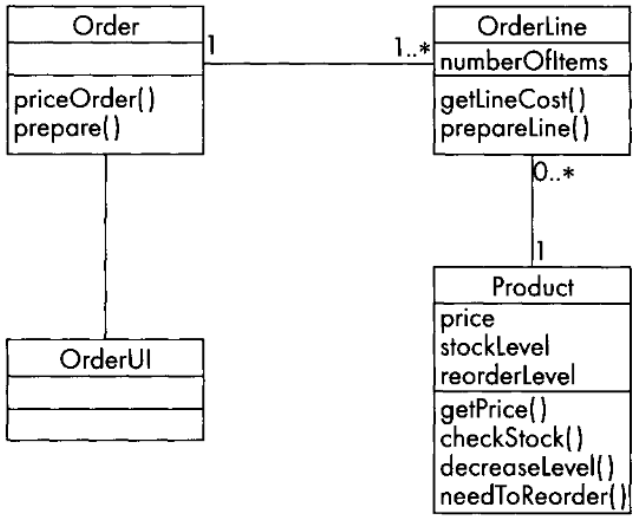
Câu hỏi

1. Đường điều hướng (navigable path) là gì?
2. Ý nghĩa của điều hướng một chiều?
3. Nêu khác biệt giữa mô hình phân tích và mô hình thiết kế?
4. Kiến trúc phân tầng (layered architecture) là gì?
5. Biểu đồ gói (package diagram) dùng làm gì?
6. Sự phụ thuộc (dependency) giữa các gói là gì?
7. Biểu đồ thành phần (component diagram) dùng làm gì?
8. Biểu đồ triển khai (deployment diagram) dùng làm gì?

Bài tập

1. Hãy vẽ biểu đồ trình tự thể hiện:

:OrderUI sends a priceOrder() message to :Order; :Order sends a getLineCost() message to each :OrderLine; each :OrderLine sends a getPrice() message to the appropriate :Product, it then returns the lineCost which is the price*numberOfItems. Your answer should use an iteration clause.



Bài tập

2. Hãy vẽ biểu đồ trình tự thể hiện:

An :OrderUI sends a prepare() message to an :Order; the :Order sends a prepareLine() message to each :OrderLine; the :OrderLine must check if there is sufficient stock of its :Product and adjust the stock level as appropriate.

Bài tập

3. Hãy vẽ biểu đồ trình tự thể hiện:

Each time the stock level is decreased, the :Product checks whether it needs to reorder. It does this by comparing the stockLevel with reorderLevel. If it needs to reorder, the :Product sends a message to the :Supplier.

Bài tập

4. We have grouped classes into a Stock package and an Ordering package as follows:

Stock package	*Ordering package*
Product	Customer
Supplier	Order
	OrderLine

Each :Product knows its price. Each :OrderLine has a numberOfItems attribute and is linked to a :Product. Each :OrderLine must work out its line cost by asking for the price from the appropriate :Product and multiplying this by the numberOfItems. What is the dependency between the two packages?