
Phân Tích & Thiết Kế Hướng Đối Tượng Sử Dụng UML

Giới Thiệu Về Hướng Đối Tượng

Mục tiêu: Giới thiệu về Hướng Đối Tượng

- ✍ Tìm hiểu các nguyên tắc cơ bản của hướng đối tượng (object orientation – OO)
- ✍ Tìm hiểu các khái niệm cơ bản và các thuật ngữ của hướng đối tượng kết hợp với hệ thống ký hiệu của UML
- ✍ Đánh giá chính xác sức mạnh của OO
- ✍ Tìm hiểu một số cơ chế mô hình hoá cơ bản của UML

Giới thiệu về Hướng Đối Tượng: Các chủ đề

- ★ ✎ Các nguyên tắc cơ bản của OO
- ✎ Các khái niệm cơ bản của OO
- ✎ Sức mạnh của OO
- ✎ Các cơ chế mô hình hoá cơ bản của UML

Các nguyên tắc cơ bản của OO

Hướng Đối Tượng

**Trừu tượng hoá
Abstraction**

**Tính đóng gói
Encapsulation**

**Tính đơn thể[?]
Modularity**

**Tính phân cấp
Hierarchy**

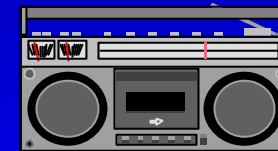
Thế nào là trừu tượng hoá ?



Người bán hàng



Khách hàng

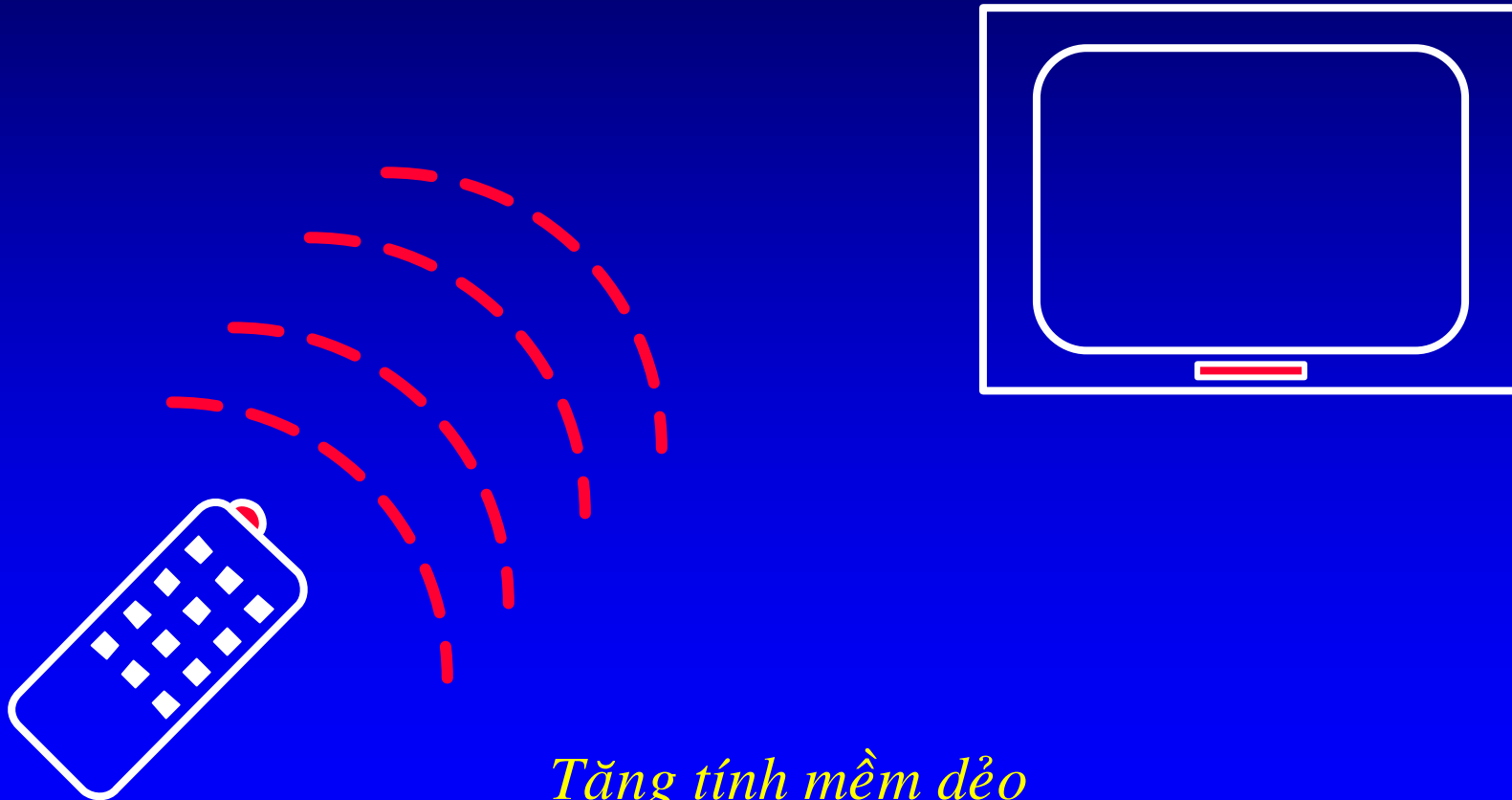


Sản phẩm

Quản lý được độ phức tạp

Encapsulation là gì?

- ✍ Che dấu cài đặt bên trong với clients
- ✍ Clients phụ thuộc vào interface

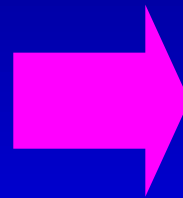


Tính đơn thể là gì ?

✍ Phân chia nhỏ một vấn đề phức tạp thành nhiều phần nhỏ, đơn giản hơn quản lý được

Nhận
Đơn đặt hàng

Hệ thống xử lý
đơn đặt hàng

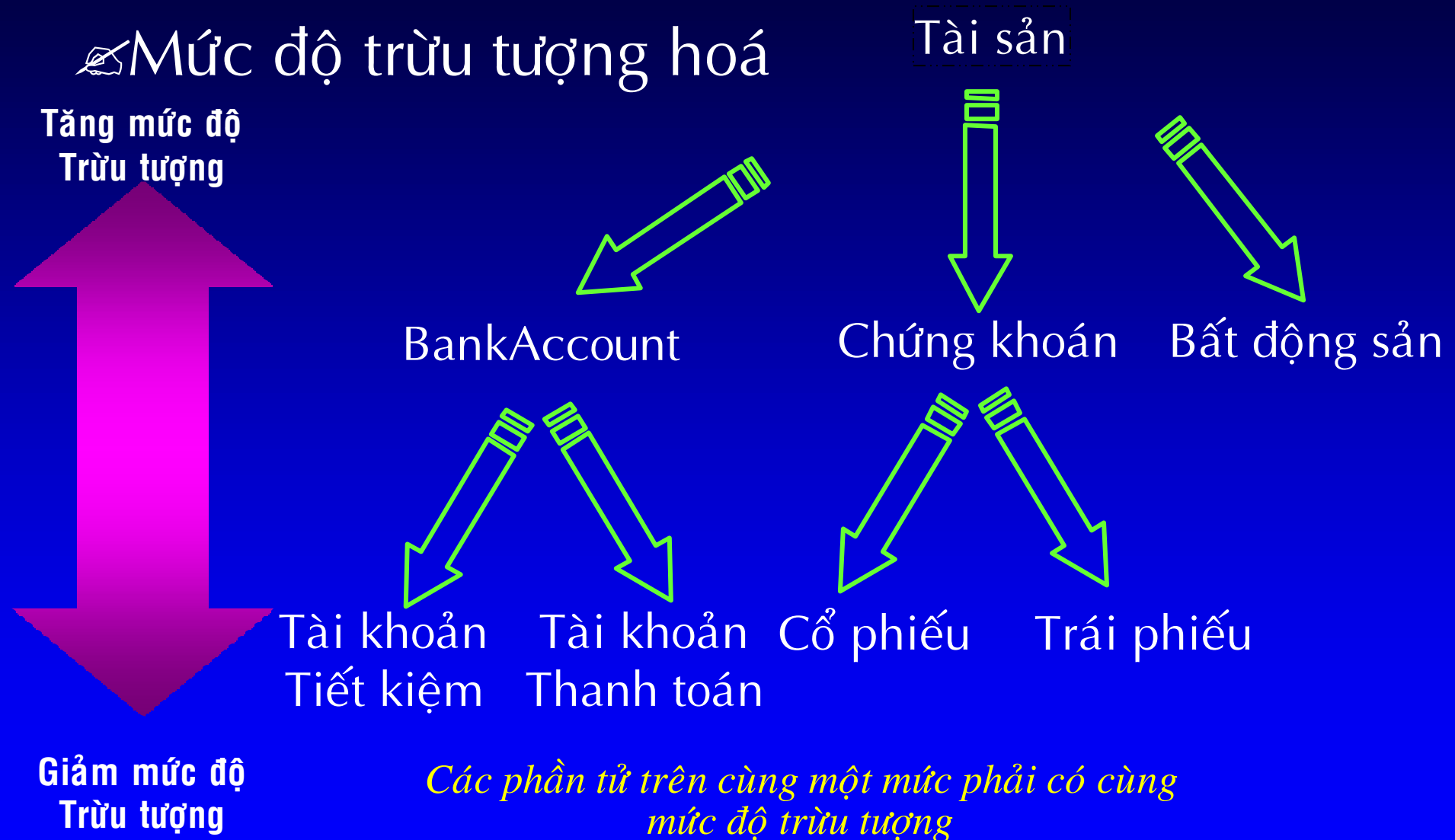


Thực hiện
đơn đặt hàng





Tính tiền

Quản lý được độ phức tạp

Sự phân cấp (Hierarchy) là gì ?



Giới thiệu về Hướng Đối Tượng: Các chủ đề

-  Các nguyên tắc cơ bản của OO
- ★  Các khái niệm cơ bản của OO
-  Sức mạnh của OO
-  Các cơ chế mô hình hoá cơ bản của UML

Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

 Interface (Polymorphism)

 Component

 Package

 Subsystem

 Relationships

Các khái niệm cơ bản của Hướng đối tượng

★ ✎ Object

✎ Class

✎ Attribute

✎ Operation

✎ Interface (Polymorphism)

✎ Component

✎ Package

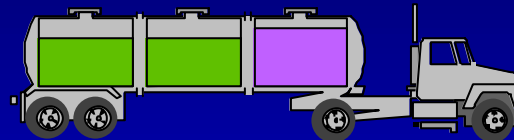
✎ Subsystem

✎ Relationships

Object là gì ?

✍ Một cách không hình thức, một đối tượng biểu diễn một thực thể, dạng vật lý, khái niệm, hoặc phần mềm

✍ Thực thể vật lý



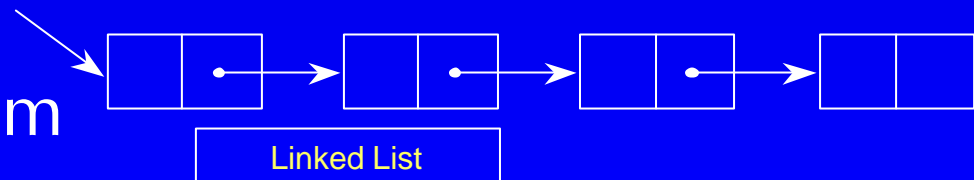
Truck

✍ Thực thể khái niệm



Chemical Process

✍ Thực thể phần mềm



Một định nghĩa hiệu quả hơn

- ✍ Một đối tượng là một khái niệm, sự trừu tượng, hoặc một vật với giới hạn rõ ràng và có ý nghĩa với một ứng dụng cụ thể
- ✍ Một đối tượng có:
 - ✍ Trạng thái
 - ✍ Hành vi
 - ✍ Định danh (Identity)

Biểu diễn đối tượng

✍ Một đối tượng được biểu diễn bởi một hình chữ nhật với tên được gạch dưới

: Professor

Chỉ có tên Class

ProfessorClark

Chỉ có tên đối tượng

ProfessorClark :
Professor

Tên class và tên đối tượng

$$a + b = 10$$

Professor Clark



Các khái niệm cơ bản của Hướng đối tượng

 Object

★  Class

 Attribute

 Operation

 Interface (Polymorphism)

 Component

 Package

 Subsystem

 Relationships

Class là gì?

- ✍ Class là mô tả của một nhóm đối tượng có chung các thuộc tính (attributes), hành vi (operations), các mối quan hệ và ngữ nghĩa
 - ✍ Một đối tượng là một thể hiện của class
- ✍ Một class là sự trừu tượng mà trong đó:
 - ✍ Nhấn mạnh các tính chất quan trọng
 - ✍ Bỏ qua các tính chất khác

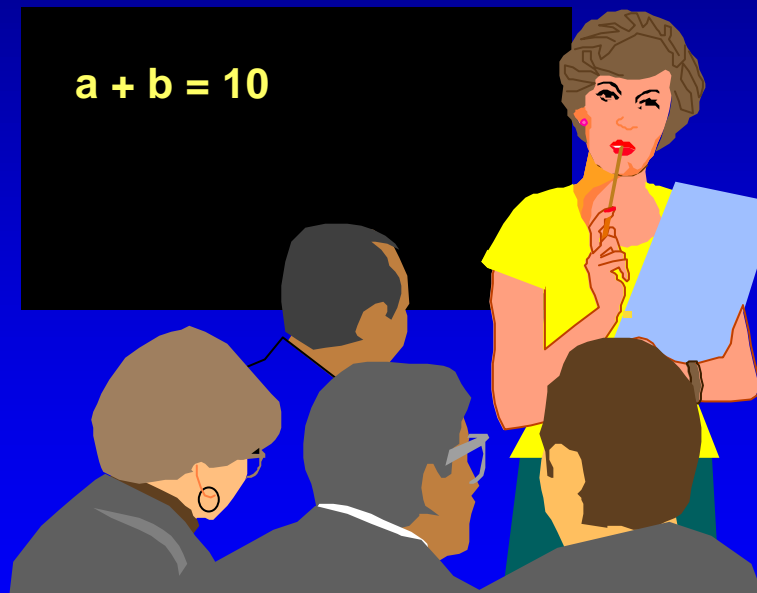
Nguyên tắc OO : Trừu tượng hoá

Ví dụ về Class

Class Course

Properties

Tên
Địa điểm
Thời gian
Số tín chỉ
Giờ bắt đầu
Giờ kết thúc

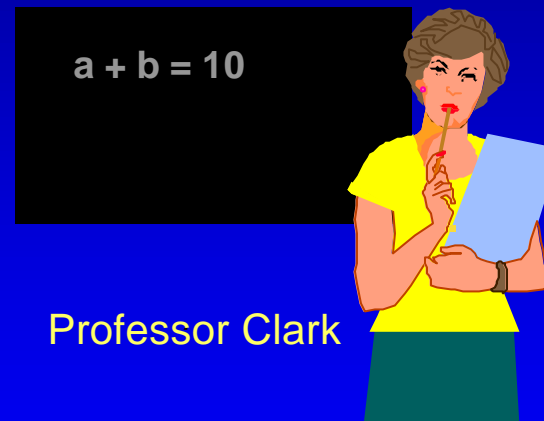


Behavior

Thêm một sinh viên
Huỷ một sinh viên
Lấy danh sách giáo sư
Xác định hết chỗ chưa

Biểu diễn Class

✍ Một class biểu diễn bằng một hình chữ nhật gồm ba phần



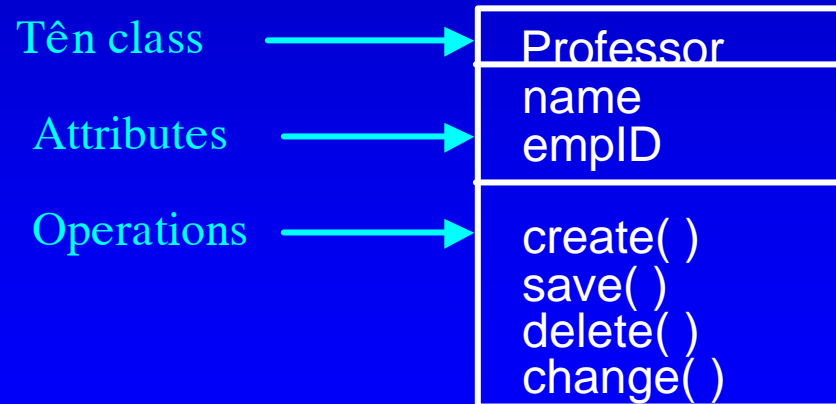
Các phần trong một Class

✍ Một class bao gồm ba phần

✍ Phần đầu chứa tên class

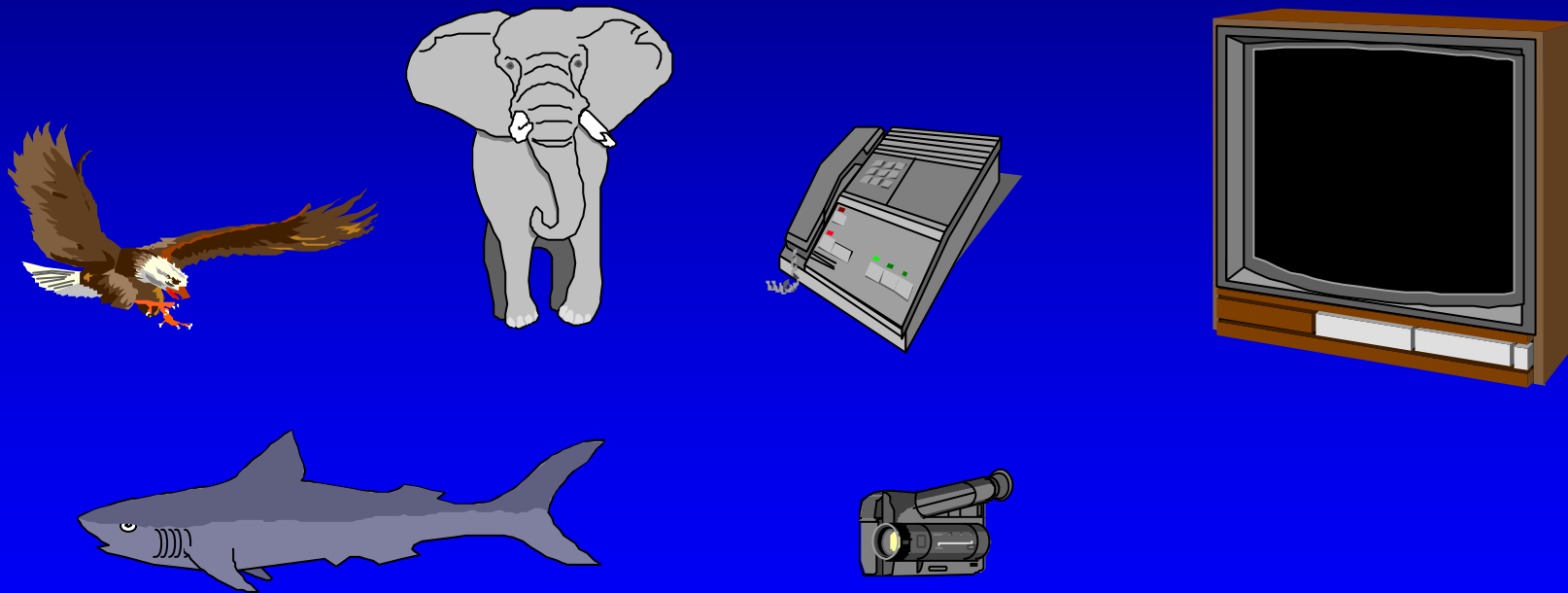
✍ Phần thứ hai cho thấy cấu trúc của lớp (attributes)

✍ Phần thứ ba cho thấy các hành vi của lớp (operations)



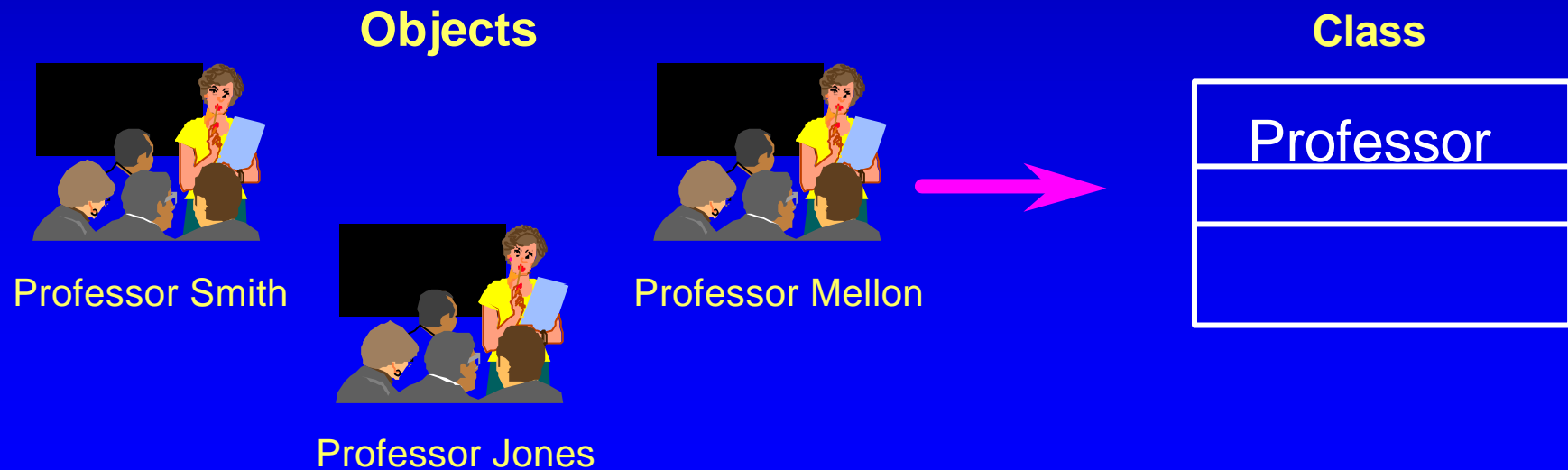
Các lớp đối tượng

 Bạn nhìn thấy bao nhiêu class?



Quan hệ giữa class và đối tượng

- ✍ Một class là một định nghĩa trừu tượng của một đối tượng
- ✍ Nó định nghĩa cấu trúc và hành vi của mỗi đối tượng trong lớp
- ✍ Nó được dùng như khuôn mẫu để tạo đối tượng
- ✍ Các đối tượng được nhóm thành các class



Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

★  Attribute

 Operation

 Interface (Polymorphism)

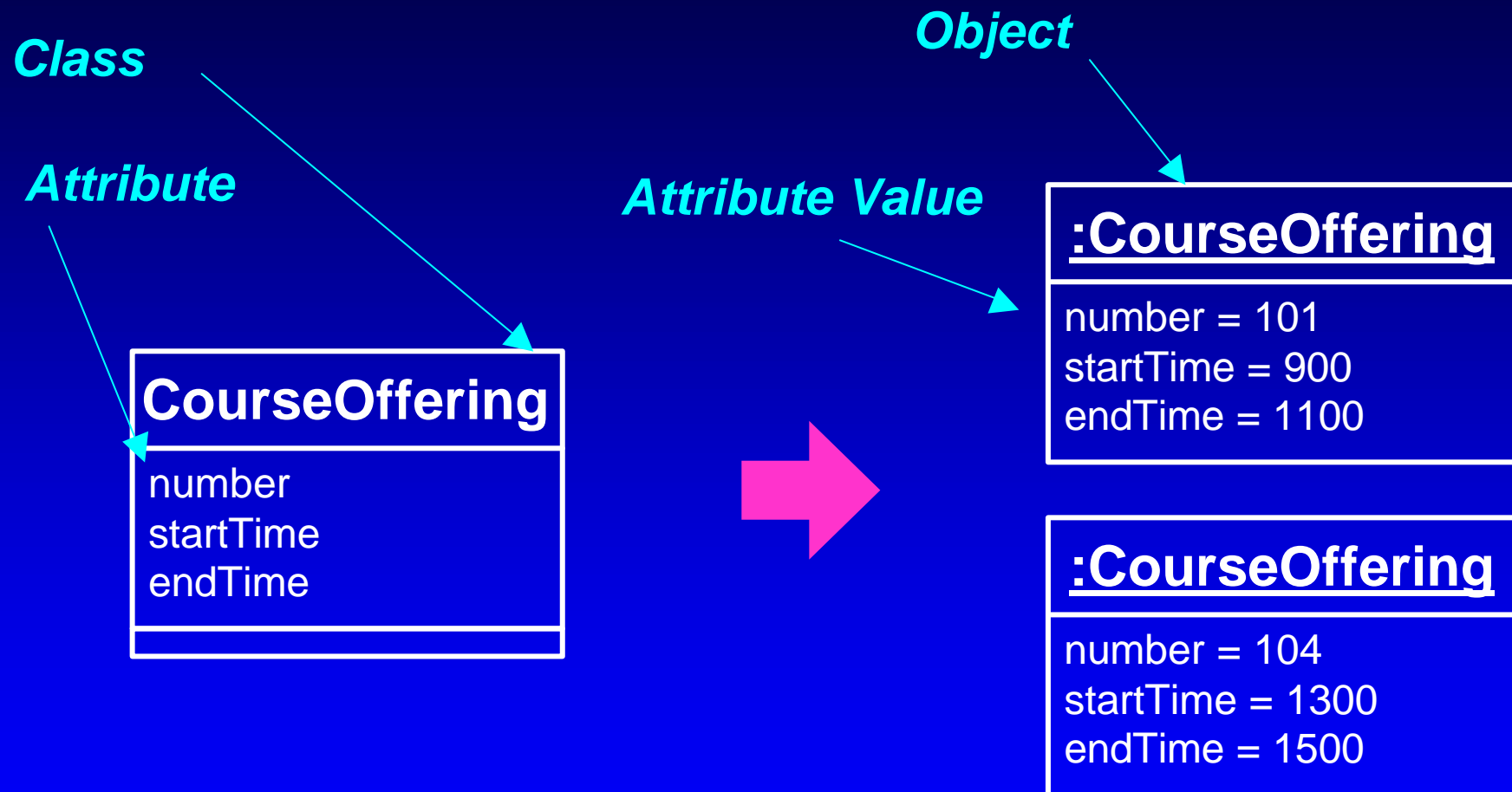
 Component

 Package

 Subsystem

 Relationships

Thuộc tính (Attribute) là gì?



Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

★  Operation

 Interface (Polymorphism)

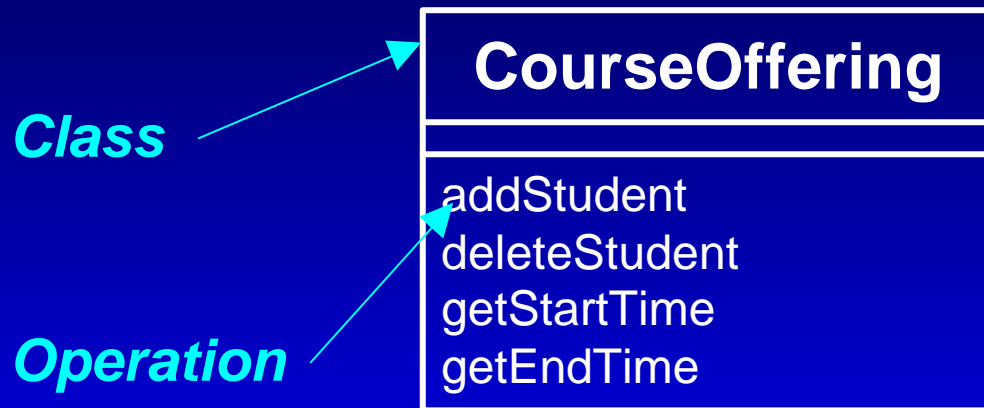
 Component

 Package

 Subsystem

 Relationships

Hành vi (Operation) là gì?



Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

★  Interface (Polymorphism)

 Component

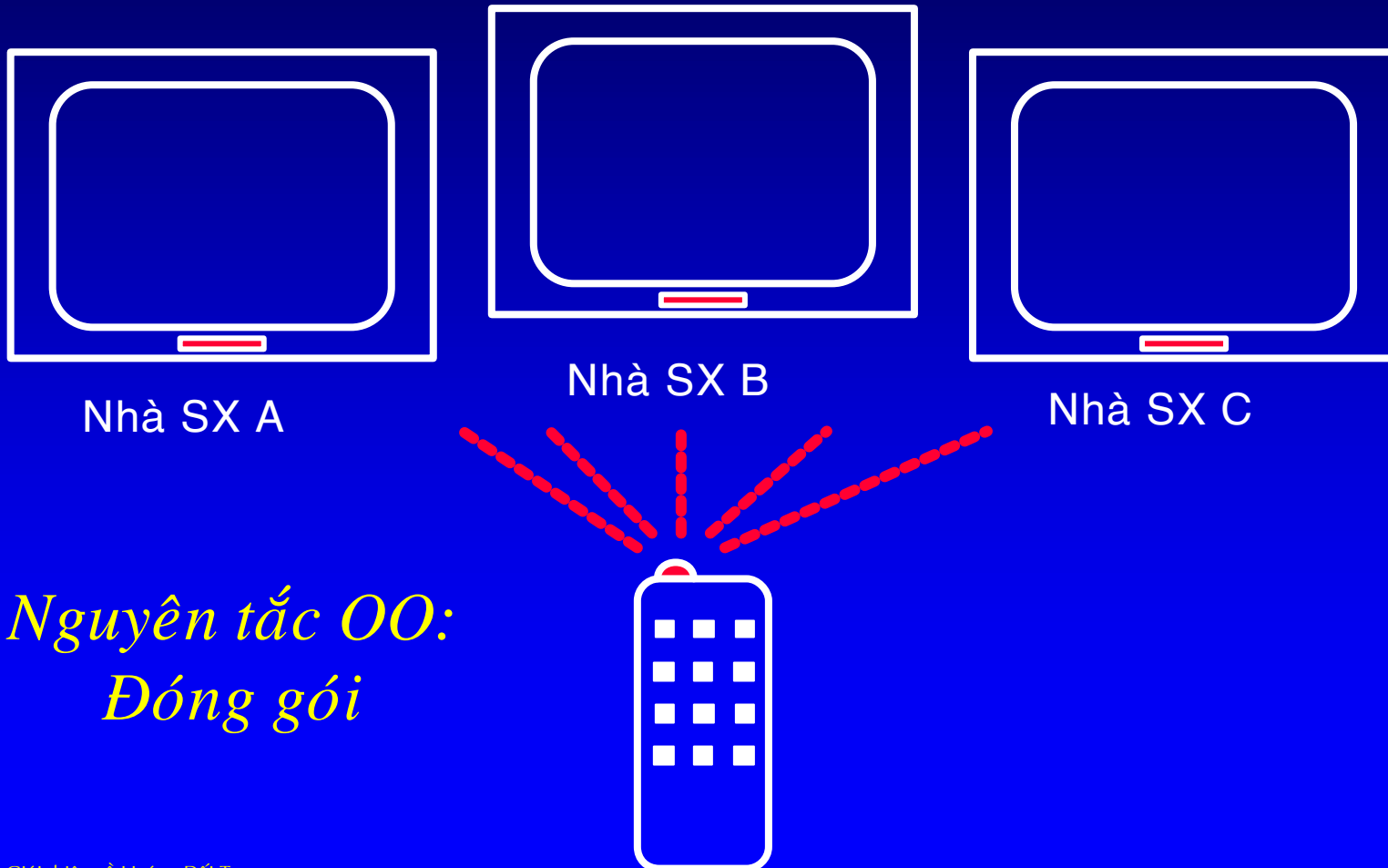
 Package

 Subsystem

 Relationships

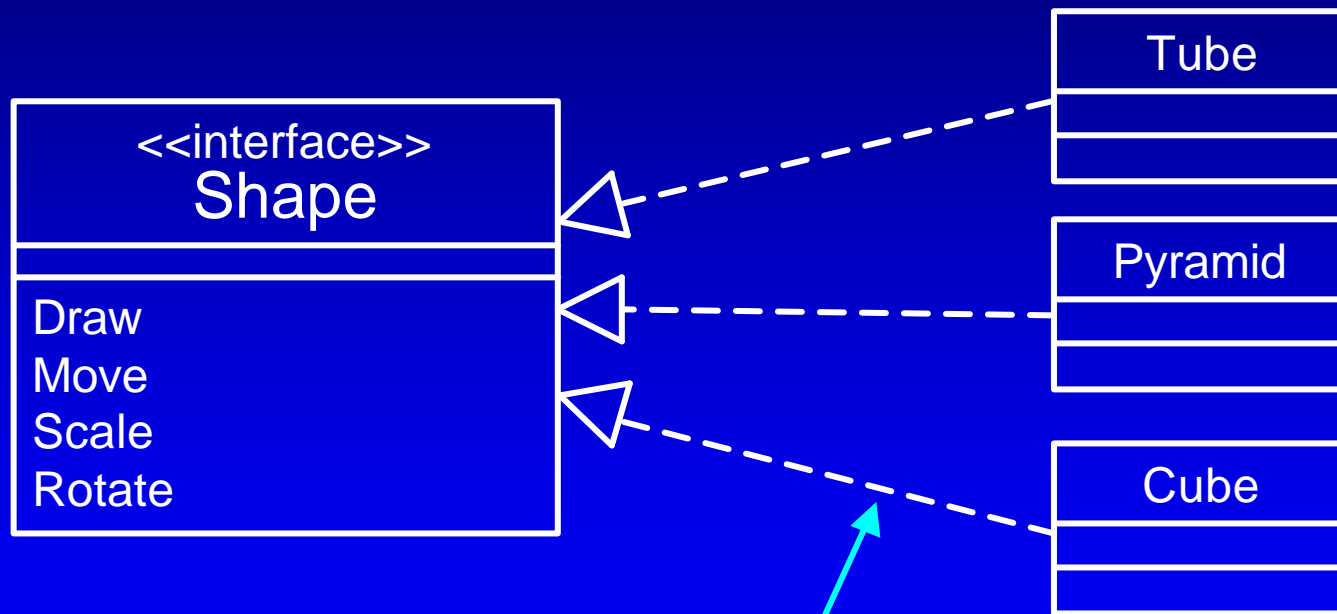
Polymorphism là gì?

✍ Khả năng che dấu nhiều cài đặt khác nhau bên dưới một giao diện (interface) duy nhất



Interface là gì?

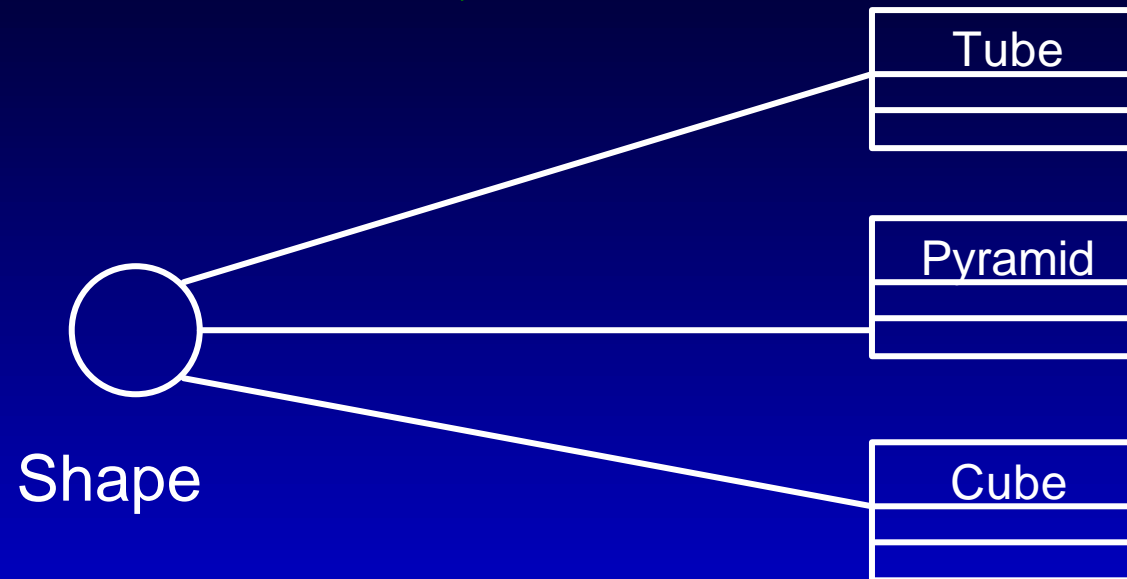
- ✂ Interface hình thức hoá polymorphism
- ✂ Interface hỗ trợ kiến trúc “plug-and-play”



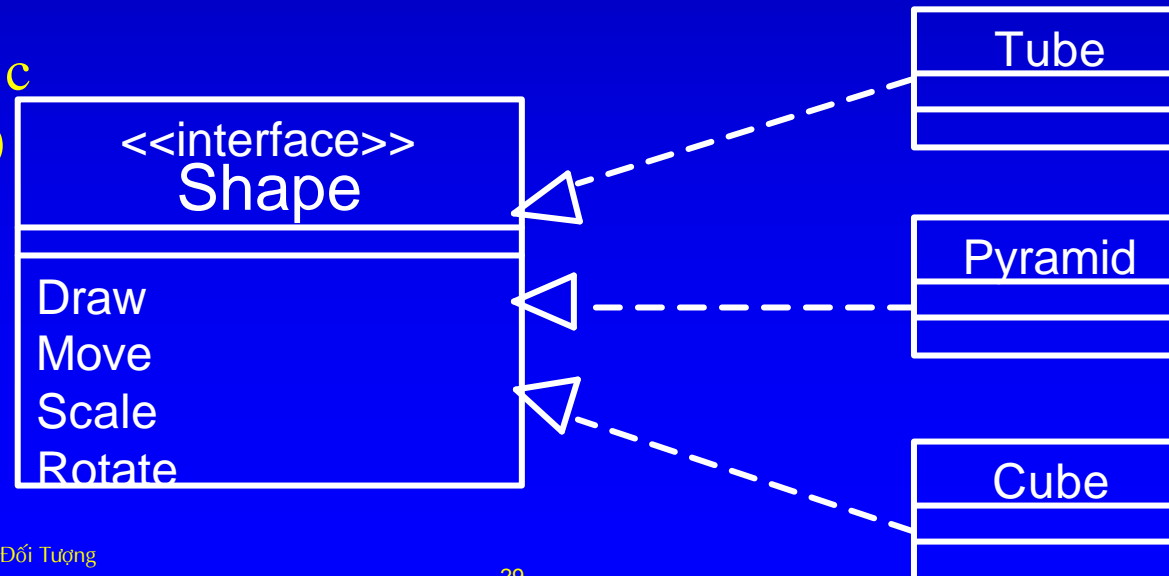
Quan hệ Realization

Biểu diễn Interface

Biểu diễn rút gọn



Biểu diễn chính tắc (Class/Stereotype)



Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

 Interface (Polymorphism)

★  Component

 Package

 Subsystem

 Relationships

Component là gì?

✍ Một phần không tầm thường của hệ thống, gần như độc lập và có thể thay thế được, giữ một chức năng rõ ràng trong hệ thống

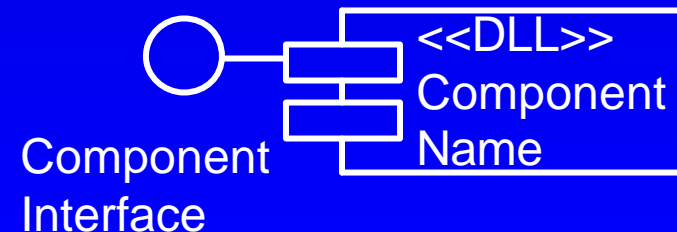
✍ Một component có thể là

✍ Một source code component

✍ Một run time components hoặc

✍ Một executable component

*Nguyên tắc OO:
Đóng gói*



Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

 Interface (Polymorphism)

 Component

★  Package

 Subsystem

 Relationships

Package là gì?

- ✍ Một package là một cơ chế để tổ chức các phần tử vào thành các nhóm
- ✍ Một phần tử trong mô hình có thể chứa các phần tử khác



*Nguyên tắc OO:
Tính đơn thể*

- ✍ Dùng để
 - ✍ Tổ chức mô hình đang phát triển
 - ✍ Một đơn vị trong quản trị cấu hình

Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

 Interface (Polymorphism)

 Component

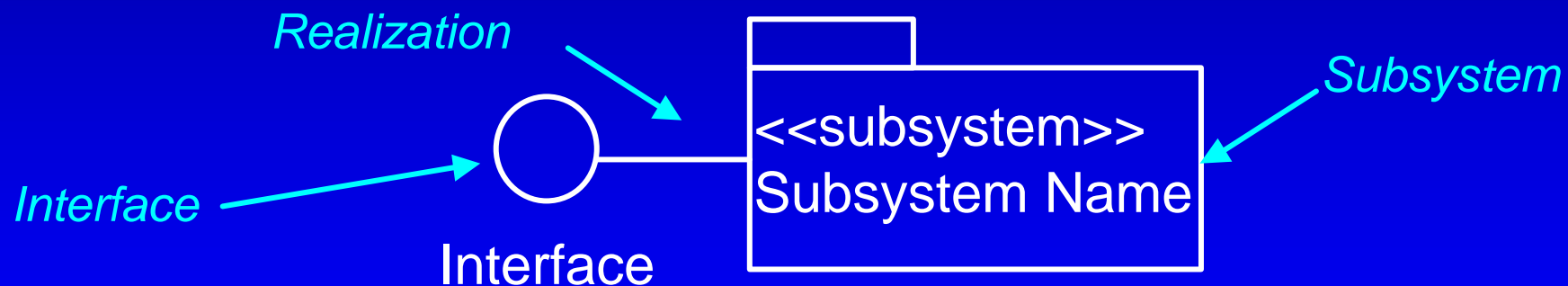
 Package

★  Subsystem

 Relationships

Subsystem là gì?

- ✍ Tổ hợp của một package (có thể chứa các phần tử khác trong mô hình) và một class (có hành vi)
- ✍ Hiện thực hoá một hoặc nhiều interface định nghĩa cho hành vi của nó

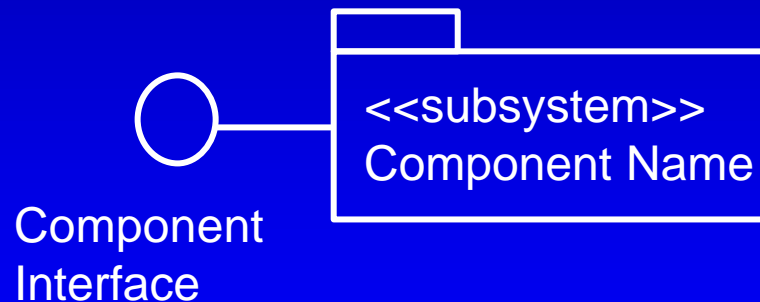


Nguyên tắc OO: Đóng gói và Tính đơn thể

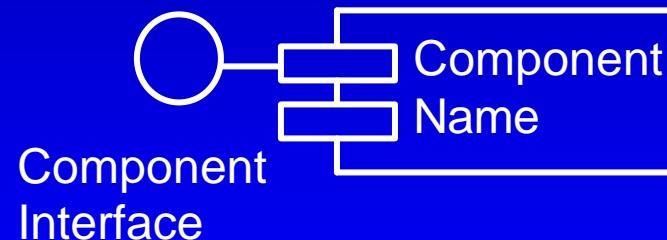
Subsystem và Component

- ✍ Component là thể hiện ở mức vật lý của một khái niệm trừu tượng trong thiết kế
- ✍ Subsystem có thể dùng để biểu diễn các component trong thiết kế

Design Model



Implementation Model



Nguyên tắc OO: Đóng gói và Tính đơn thể

Các khái niệm cơ bản của Hướng đối tượng

 Object

 Class

 Attribute

 Operation

 Interface (Polymorphism)

 Component

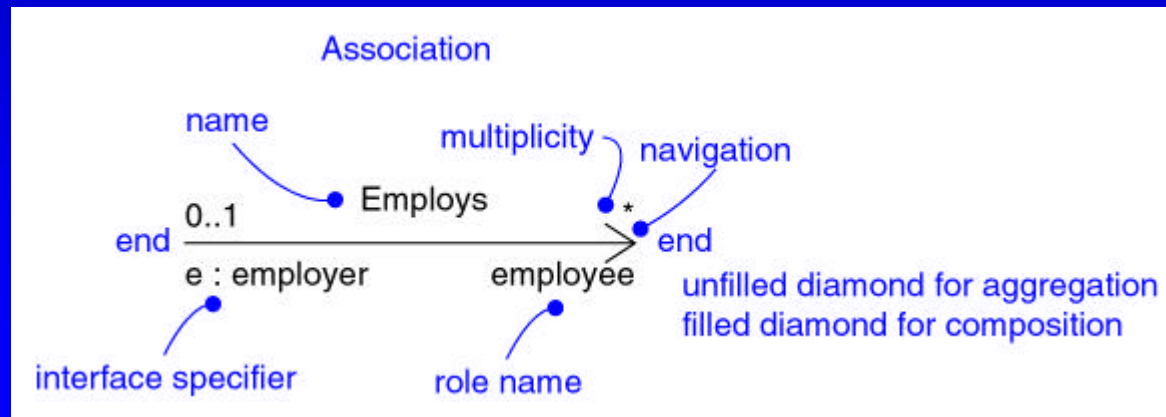
 Package

 Subsystem

★  Relationships

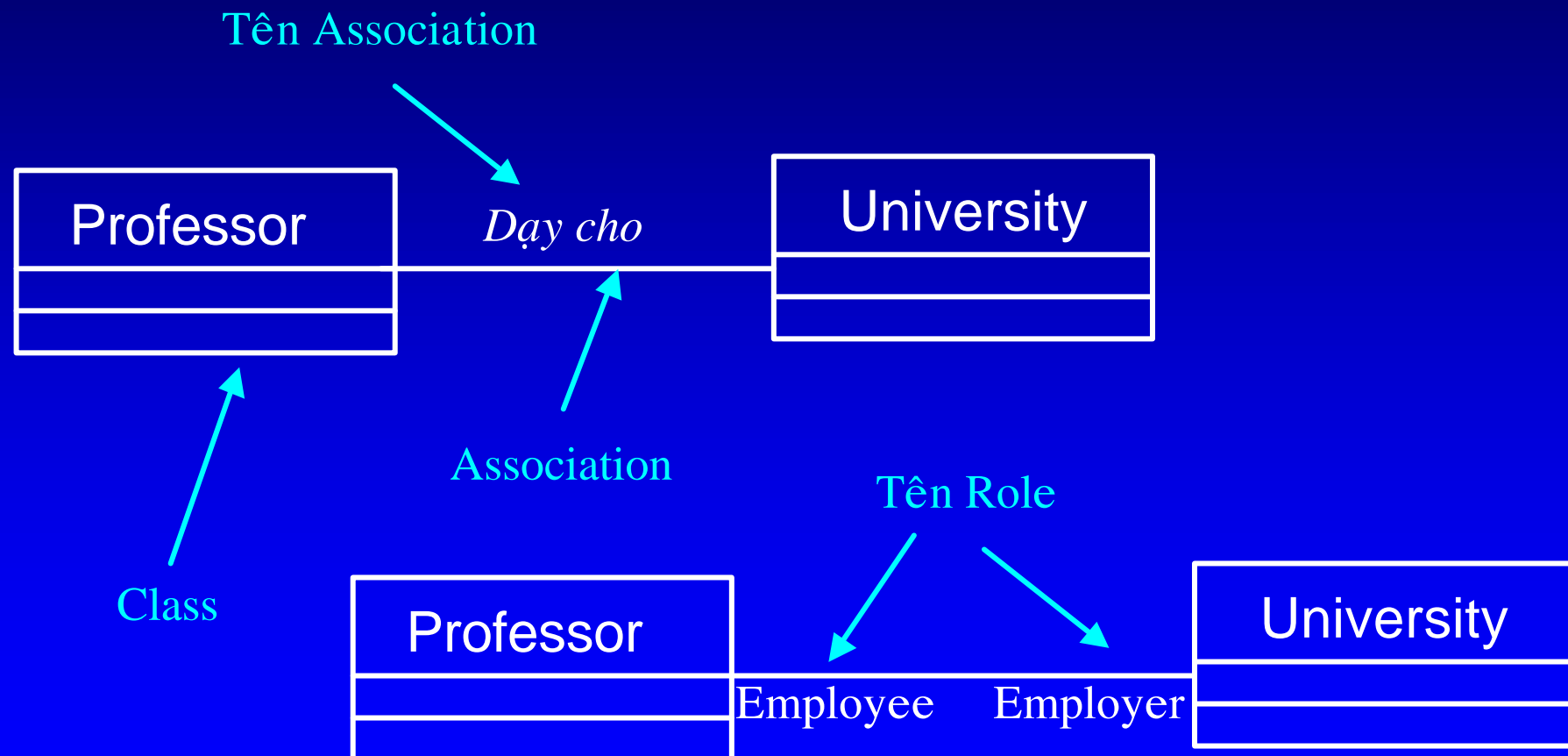
Các mối quan hệ

- ✎ Association (Kết hợp)
- ✎ Aggregation (Thu nạp)
- ✎ Composition (Cấu thành)
- ✎ Dependency (Phụ thuộc)
- ✎ Generalization (Tổng quát hóa)
- ✎ Realization (Hiện thực hoá)



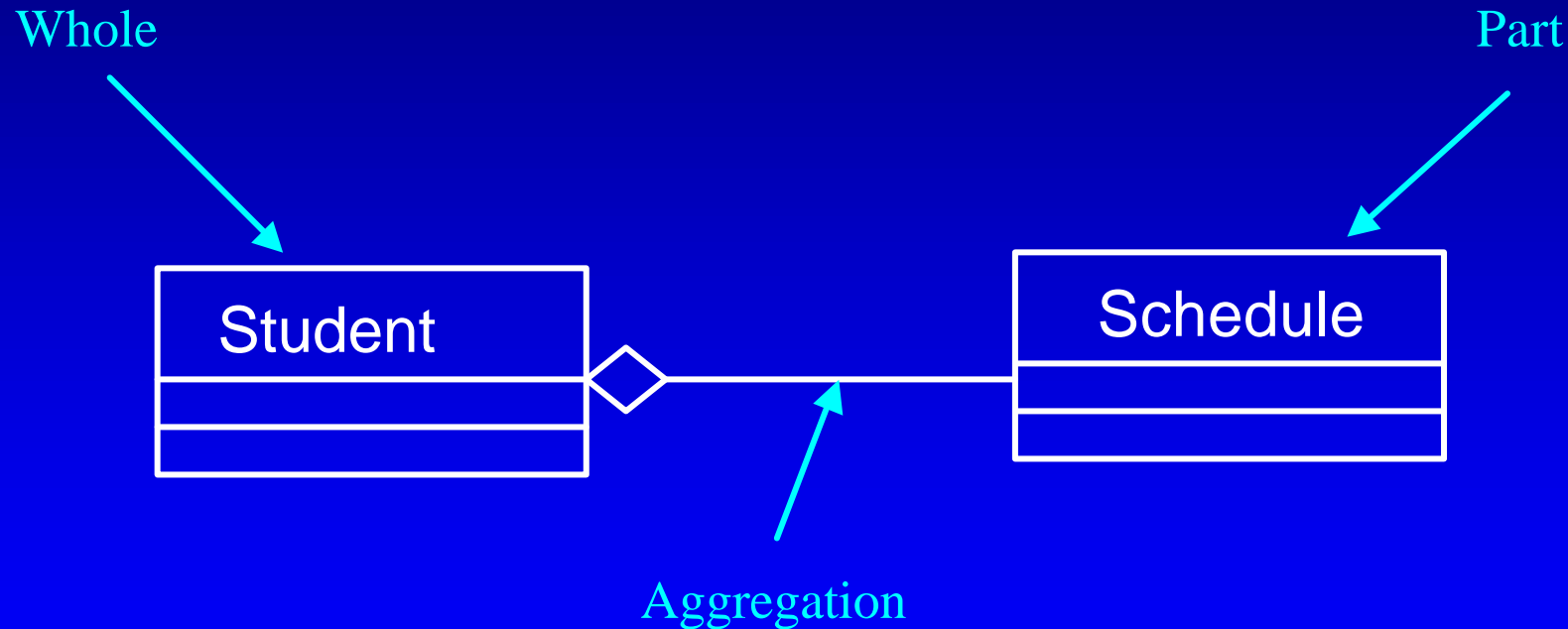
Mối quan hệ: Association

✍ Mô hình hoá một liên kết ngữ nghĩa giữa các class



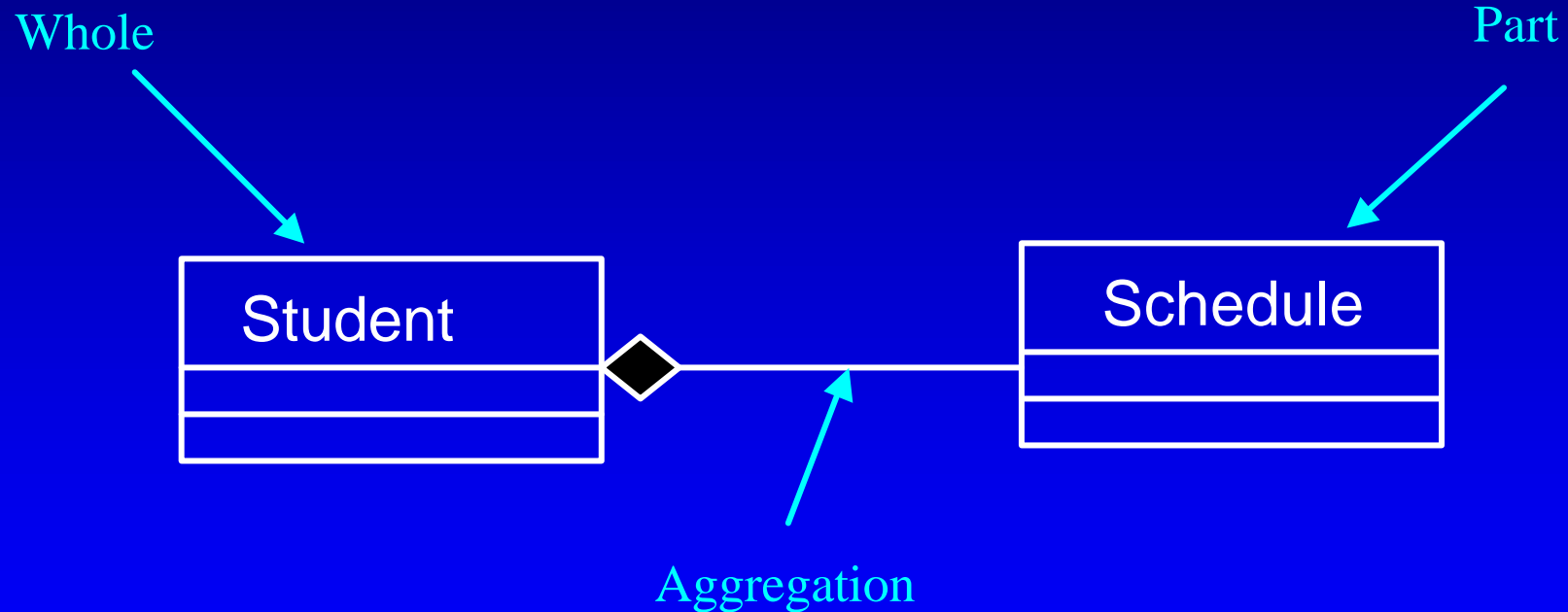
Mối quan hệ: Aggregation

✍ Một dạng đặc biệt của association mô hình hoá mối quan hệ toàn thể-bộ phận giữa một thực thể và các bộ phận của nó



Mối quan hệ: Composition

- ✍ Một dạng aggregation có tính sở hữu cao và cùng chu kỳ sống
- ✍ Các bộ phận không thể sống lâu hơn thực thể



Association: Bản số và Chiều

- ✍ Bản số xác định số đối tượng tham gia vào một mối quan hệ
 - ✍ Số các thể hiện của một class quan hệ với MỘT thể hiện của một class khác
 - ✍ Được chỉ ra ở mỗi đầu của quan hệ association
- ✍ Association và aggregation mặc định là hai chiều, nhưng người ta thường giới hạn theo một chiều
 - ✍ Mũi tên được thêm vào để chỉ chiều của mối quan hệ

Association: Bản số

 Không xác định

 Chỉ một

 Không hoặc nhiều

 Một hoặc nhiều

 Không hoặc một

 Khoảng được chỉ định

 Các khoảng không liên tục

1

0..*

*

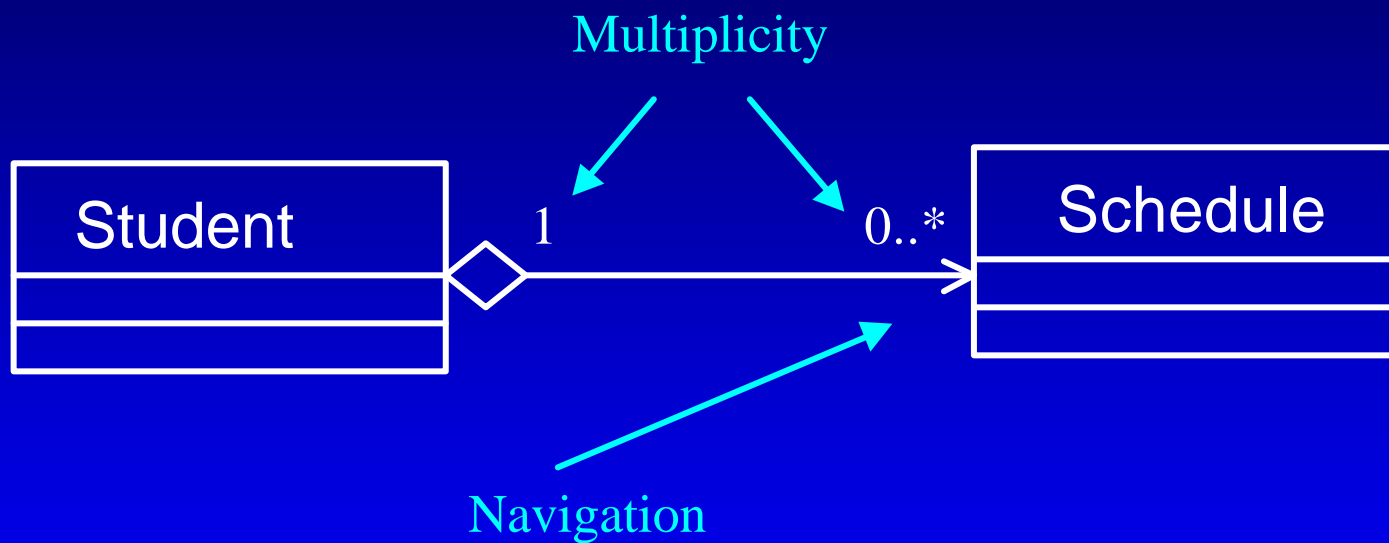
1..*

0..1

2..4

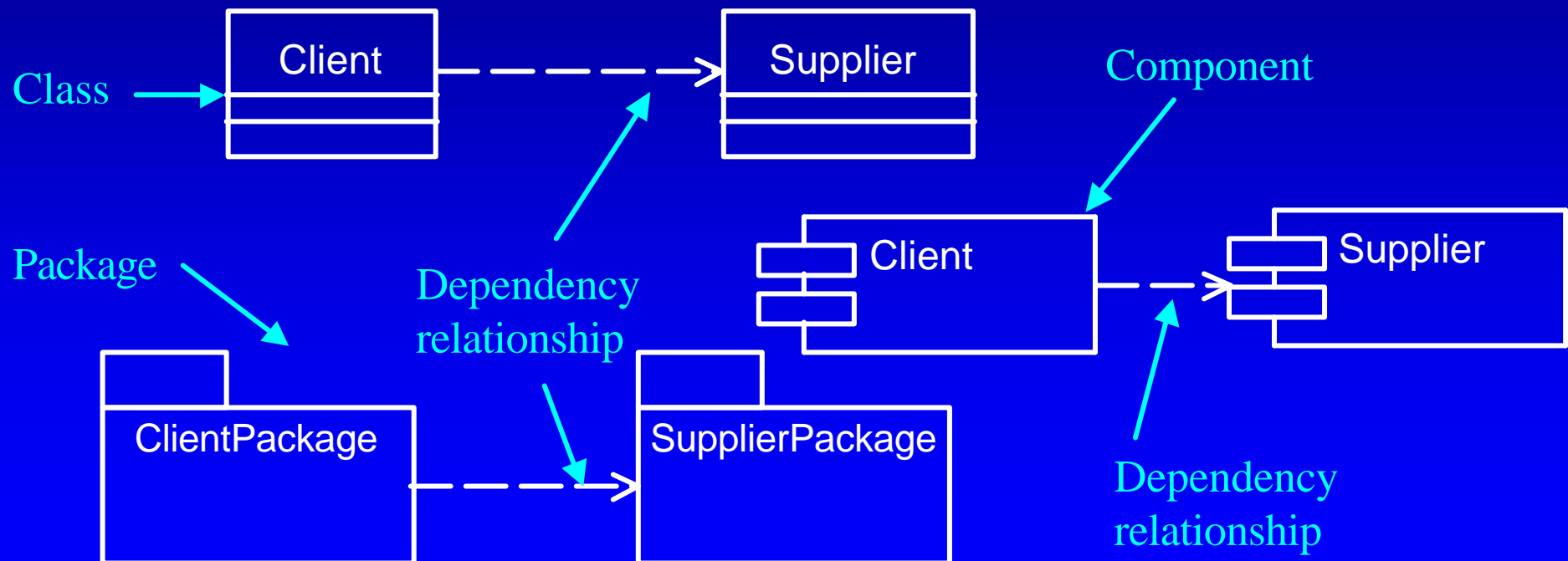
2, 4..6

Ví dụ: Bản số và Chiều



Mối quan hệ: Dependency

- ✍ Quan hệ giữa hai phần tử trong mô hình mà thay đổi ở phần tử này có thể gây ra thay đổi ở phần tử kia
- ✍ Quan hệ “sử dụng”, không cấu trúc



Mối quan hệ: Generalization

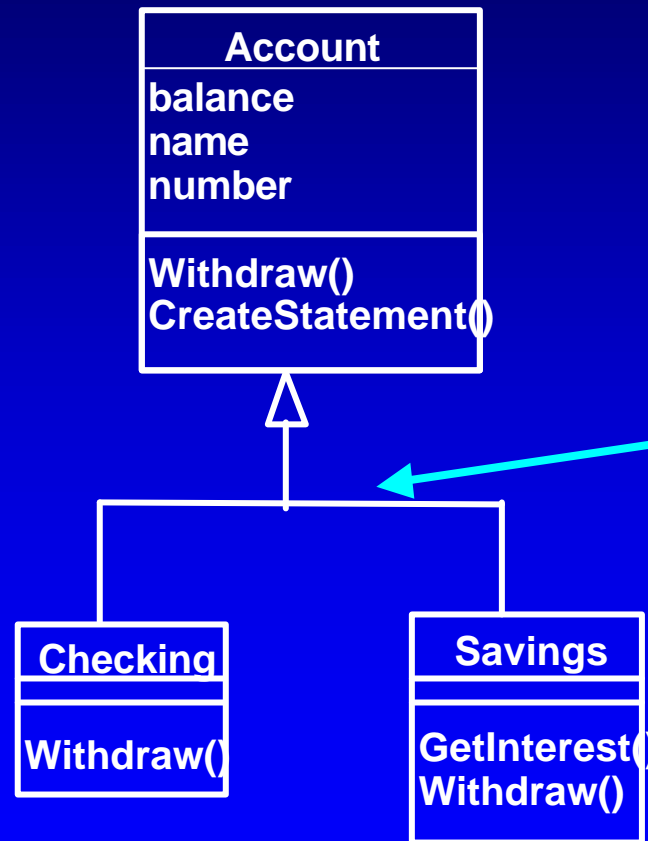
- ✍ Quan hệ giữa các class trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi của một hoặc nhiều class khác
- ✍ Xác định một sự phân cấp các mức độ trừu tượng trong đó một subclass kế thừa từ một hoặc nhiều superclass
 - ✍ Đơn kế thừa
 - ✍ Đa kế thừa
- ✍ Generalization là quan hệ “là một dạng của”

Ví dụ: Đơn kế thừa

✍ Một class kế thừa từ một class khác

Tổ tiên

Superclass
(cha)



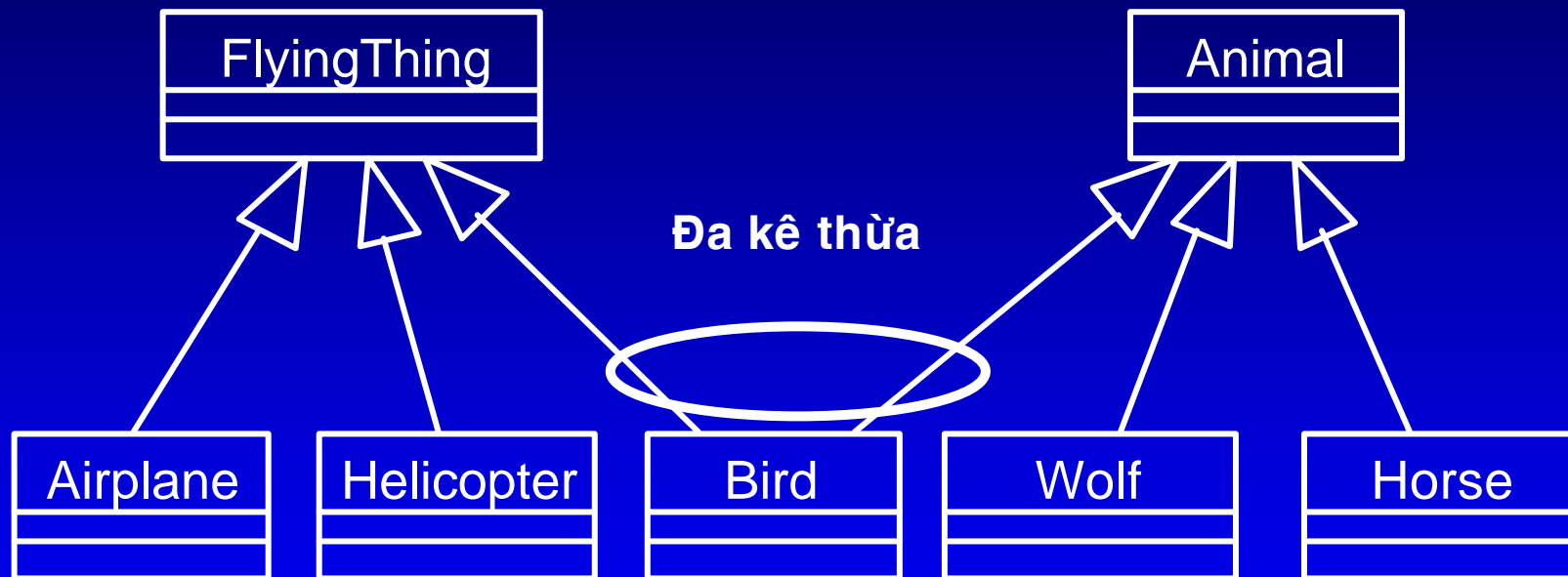
Generalization
Relationship

Subclasses

Hậu duệ

Ví dụ: Đa kế thừa

✍ Một class kế thừa từ nhiều class khác



Chỉ sử dụng đa kế thừa khi thật cần, và luôn phải cẩn thận !

Cái gì được kế thừa?

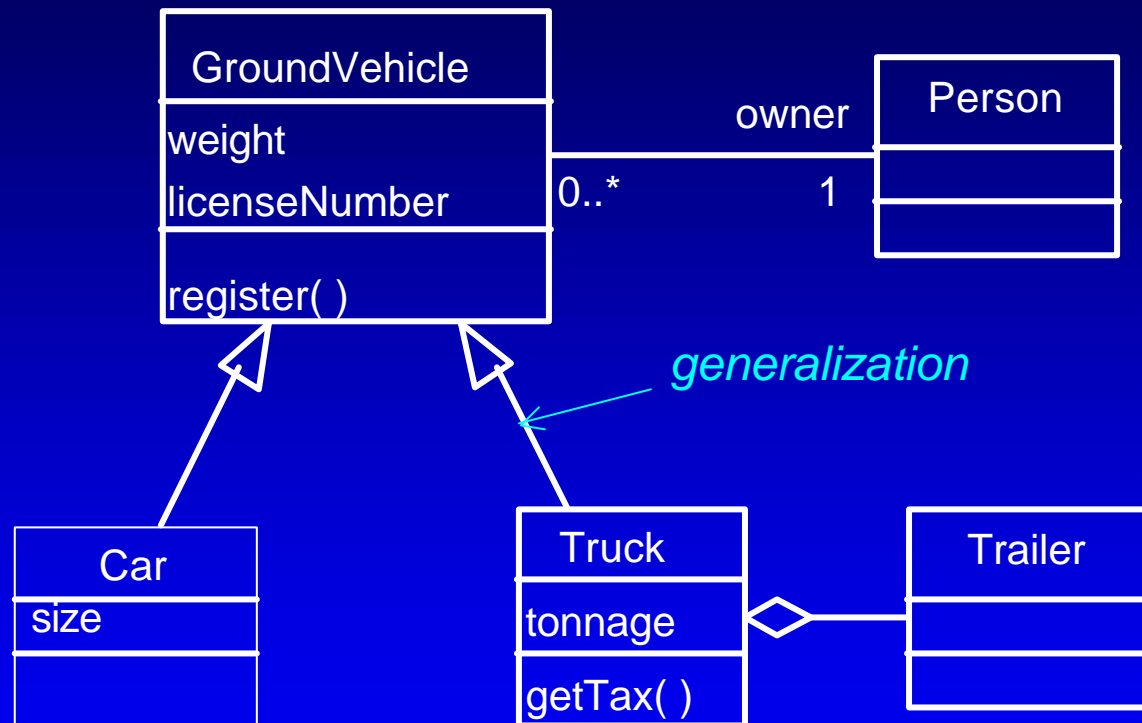
- ✍ Một subclass kế thừa các thuộc tính, hành vi và các mối quan hệ từ cha nó
- ✍ Một subclass có thể:
 - ✍ Bổ sung thuộc tính, hành vi và các mối quan hệ
 - ✍ Định nghĩa lại các hành vi (**nên cẩn thận!**)
- ✍ Các thuộc tính, hành vi và các mối quan hệ chung được đặt ở mức cao nhất có thể trong cấu trúc phân cấp

Sự kế thừa làm nổi bật các điểm tương đồng giữa các class

Ví dụ: Cái gì được kế thừa

*Superclass
(cha)*

Subclass

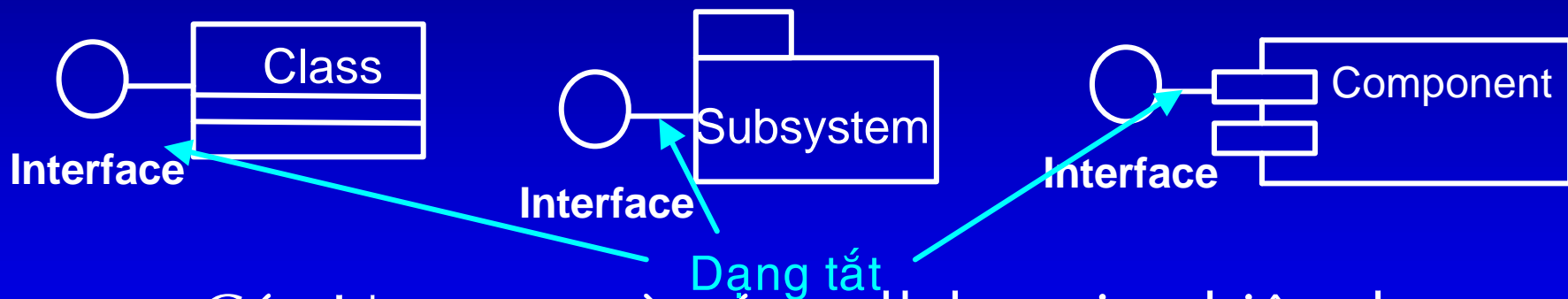


Mối quan hệ: Realization

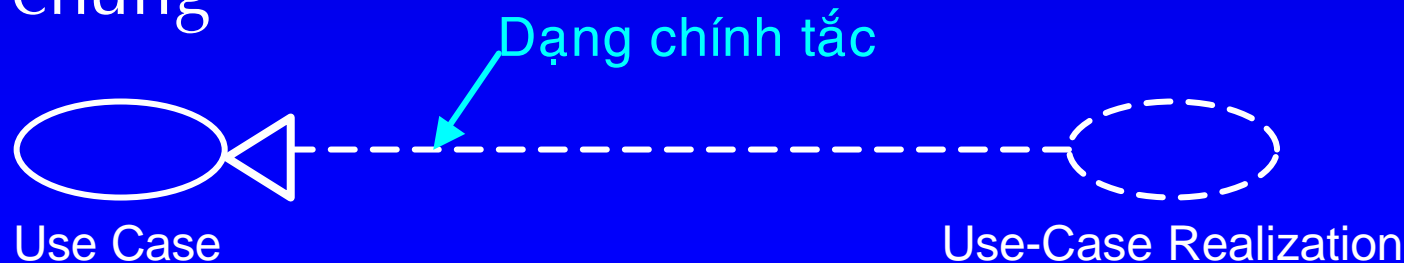
✍ Một classifier đóng vai trò một hợp đồng mà một classifier khác đồng ý thực hiện

✍ Xuất hiện giữa:

✍ Các Interface và các classifier hiện thực chúng



✍ Các Use case và các collaboration hiện thực chúng



Giới thiệu về Hướng Đối Tượng: Các chủ đề

 Các nguyên tắc cơ bản của OO

 Các khái niệm cơ bản của OO

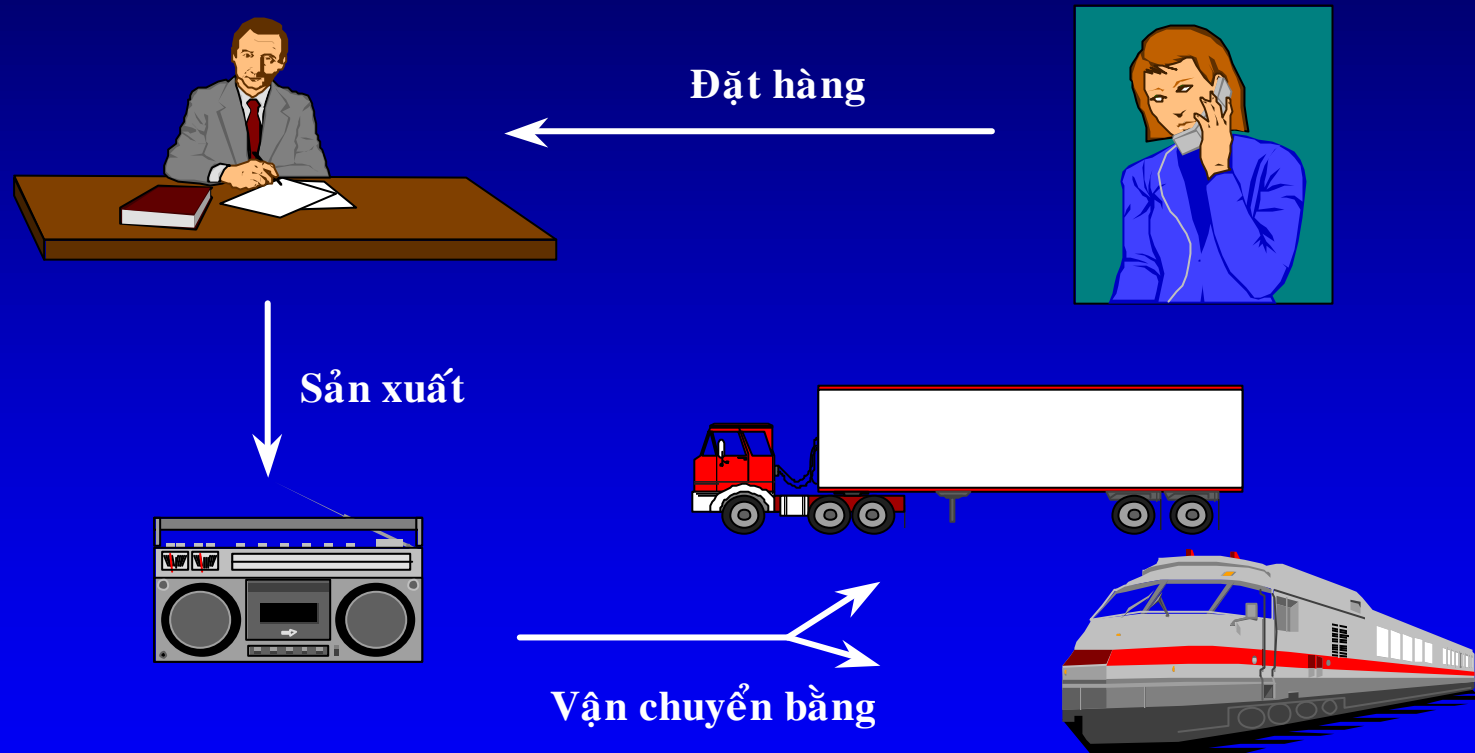
★  Sức mạnh của OO

 Các cơ chế mô hình hoá cơ bản của UML

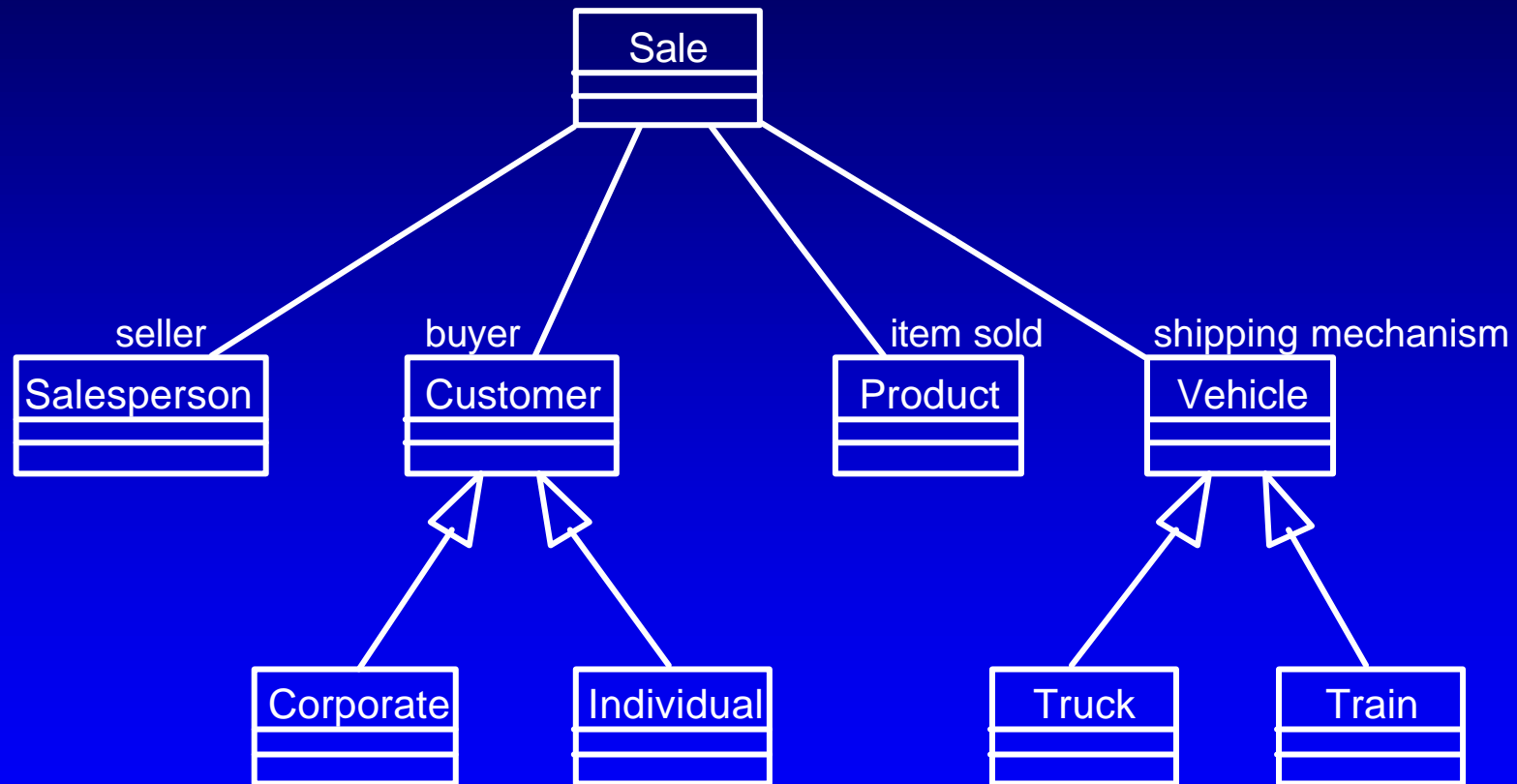
Sức mạnh của Hướng đối tượng

- ✍ Một mô hình chung
- ✍ Có tính dễ dùng lại
- ✍ Mô hình phản ánh chính xác thế giới thực
 - ✍ Mô tả chính xác hơn các tập dữ liệu và các xử lý
 - ✍ Được phân rã dựa trên các phân chia tự nhiên
 - ✍ Dễ hiểu và dễ bảo trì
- ✍ Tính ổn định
 - ✍ Một thay đổi nhỏ trong yêu cầu không gây ra sự thay đổi lớn trong hệ thống đang phát triển

Một ví dụ đơn giản: Sales Order System

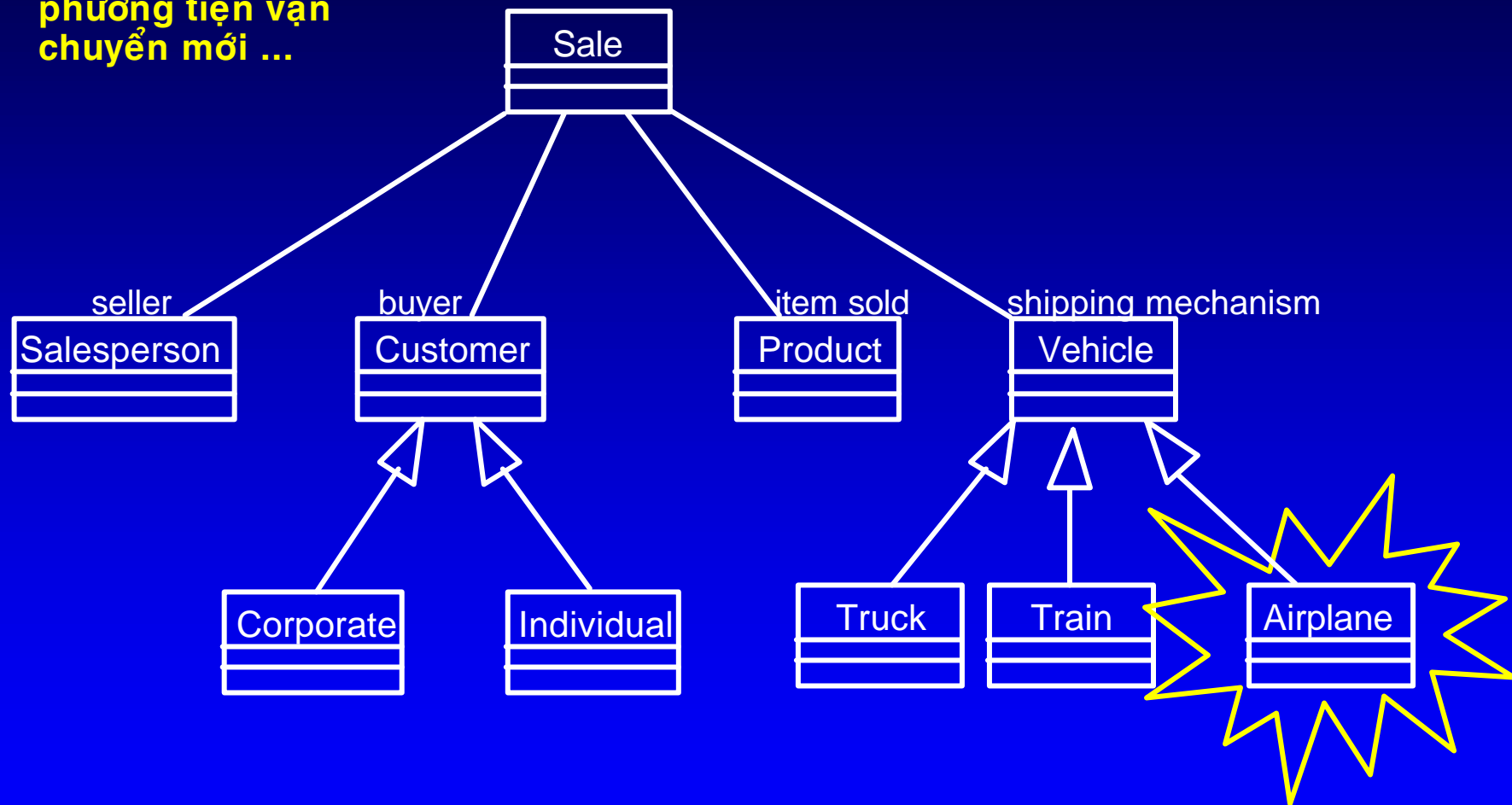


Class Diagram của ví dụ “bán hàng”



Hiệu ứng của sự thay đổi yêu cầu

Giả sử bạn cần
phương tiện vận
chuyển mới ...



Việc thay đổi liên quan đến việc thêm 1 subclass mới

Giới thiệu về Hướng Đối Tượng: Các chủ đề

 Các nguyên tắc cơ bản của OO

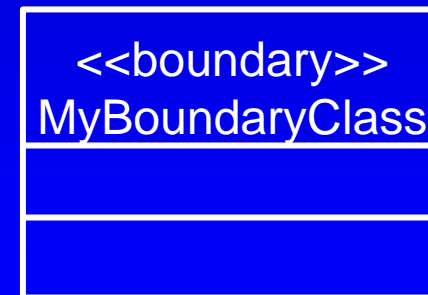
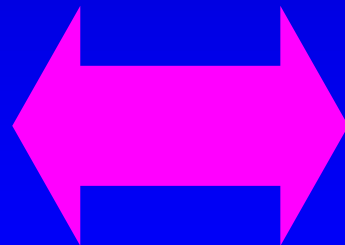
 Các khái niệm cơ bản của OO

 Sức mạnh của OO

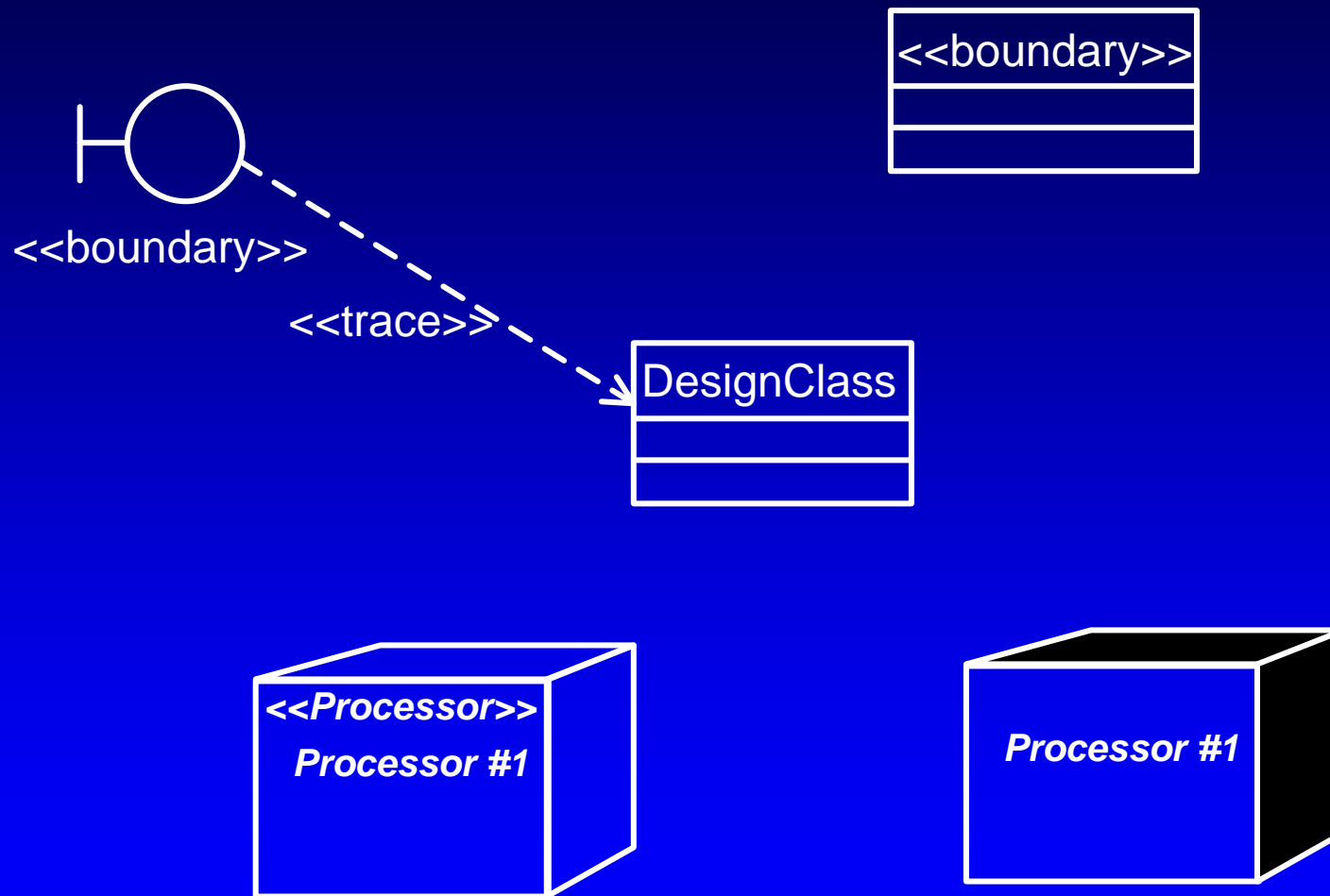
★  Các cơ chế mô hình hoá cơ bản của UML

Các khuôn mẫu (Stereotype)

- ✍ Phân lớp và mở rộng các phần tử trong hệ thống ký hiệu UML
- ✍ Định nghĩa một phần tử của mô hình mới dựa trên một phần tử khác
- ✍ Có thể áp dụng cho mọi phần tử mô hình
- ✍ Được biểu diễn với tên đặt trong dấu << >> hoặc bằng các icon khác

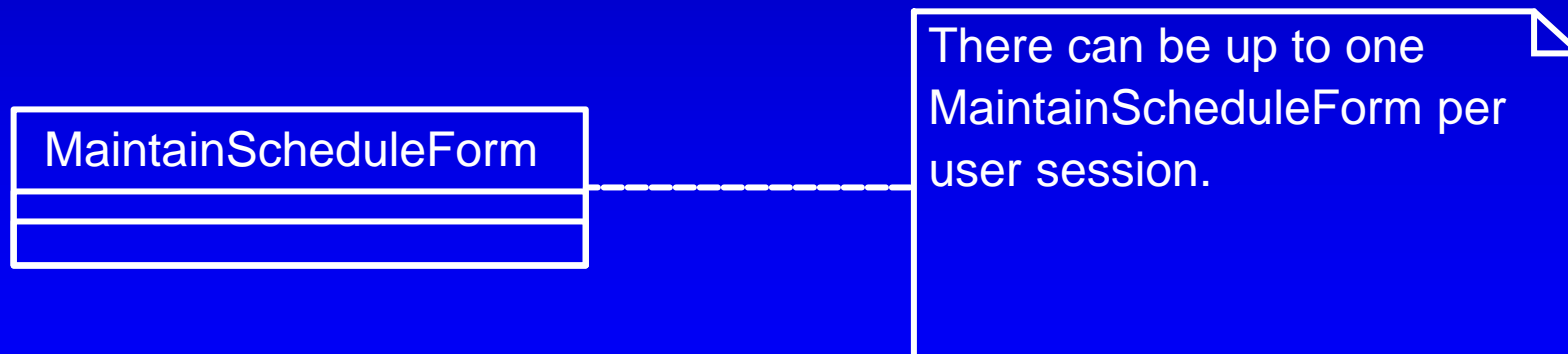


Ví dụ: Stereotype



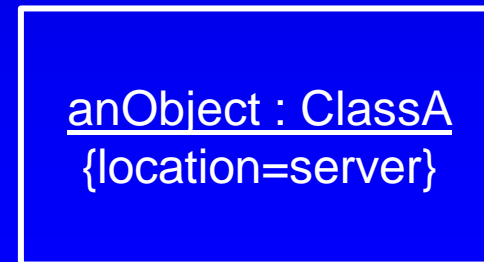
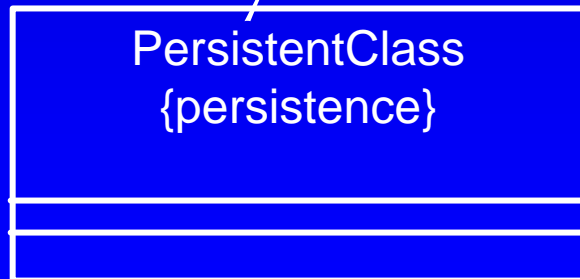
Các ghi chú (note)

- ✍ Có thể đặt ghi chú cho mọi phần tử UML
- ✍ Ghi chú dùng để thêm thông tin cho các lược đồ
- ✍ Nó là hình chữ nhật bị bẻ góc
- ✍ Ghi chú có thể móc nối với một phần tử bằng một đường đứt nét



Các giá trị đính (Tagged Values)

- ✍ Là sự mở rộng của các thuộc tính hoặc của các phần tử UML
- ✍ Là một số thuộc tính được định nghĩa sẵn bởi UML
 - ✍ Persistence
 - ✍ Location (chẳng hạn client, server)
- ✍ Là các thuộc tính có thể được tạo bởi các nhà mô hình hoá UML phục vụ cho mục đích bất kỳ



Các ràng buộc (Constraints)

✍️ Hỗ trợ việc thêm các luật mới hoặc hiệu chỉnh các luật đang tồn tại



Câu hỏi ôn tập

- ✍ Bốn nguyên tắc cơ bản của OO là gì ? Mô tả ngắn gọn về mỗi nguyên tắc.
- ✍ Đối tượng là gì ? Class là gì ? Những điểm khác nhau giữa chúng ?
- ✍ Thuộc tính (Attribute) là gì ?
- ✍ Hành vi (Operation) là gì ?
- ✍ Interface là gì ? Polymorphism là gì ?
- ✍ Component là gì ?

(còn tiếp)

Câu hỏi ôn tập (tt)

- ✍ Package là gì?
- ✍ Subsystem là gì ? Nó có quan hệ như thế nào với Component? Nó có quan hệ như thế nào với package? Nó có quan hệ như thế nào với class?
- ✍ Tên của 4 quan hệ UML cơ bản ? Mô tả từng quan hệ.
- ✍ Mô tả sức mạnh của OO.
- ✍ Cho biết tên và mô tác một số cơ chế tổng quát trong UML.
- ✍ Stereotype là gì? Cho biết tên của một số stereotype dùng chung.