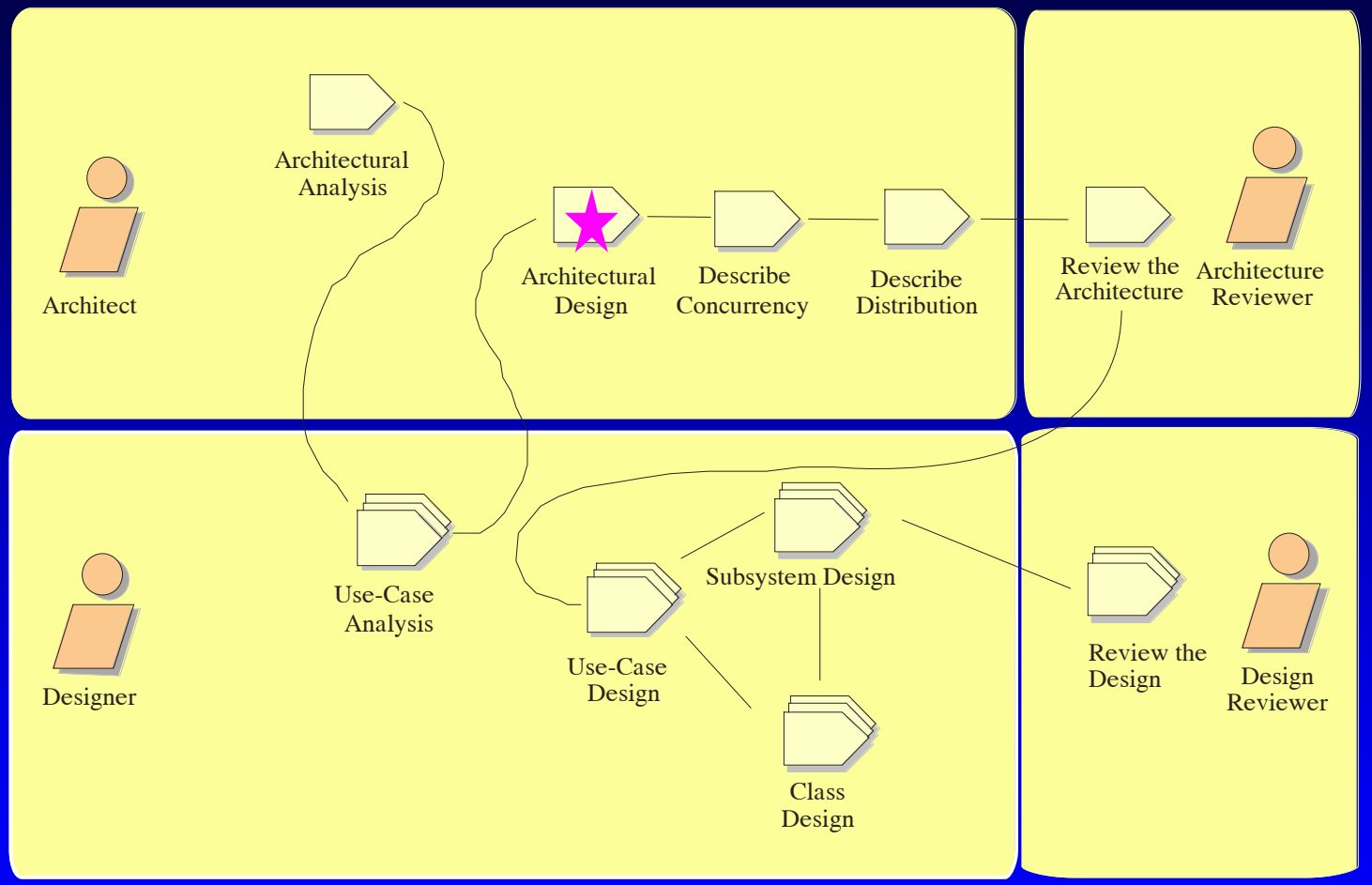
Phân tích và Thiết kế Hướng đối tượng dùng UML

Module 8: Thiết kế kiến trúc

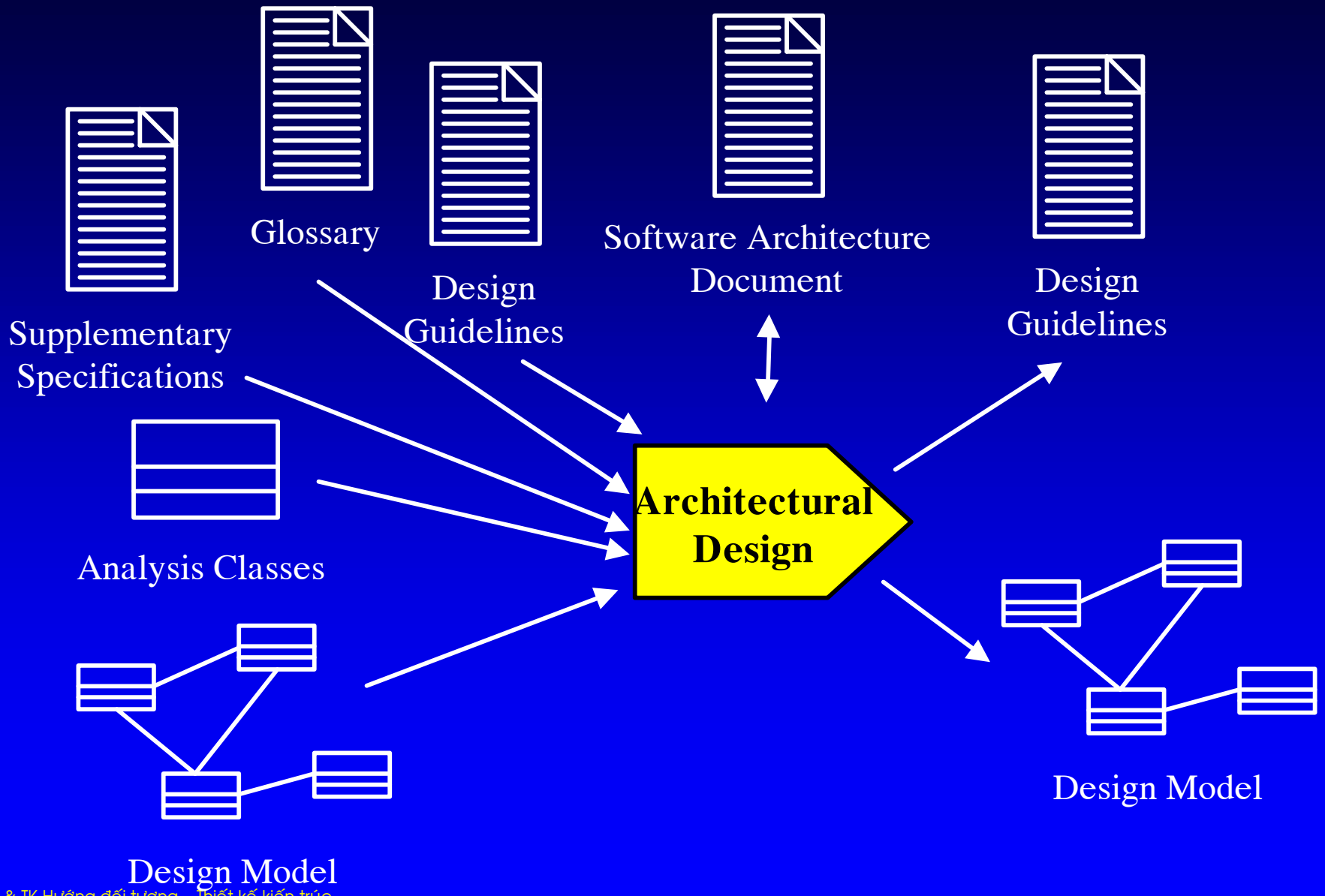
Mục tiêu:

- Tìm hiểu mục đích của công đoạn Thiết kế kiến trúc và thời điểm thực hiện công đoạn này
- Diễn giải về các cơ chế thiết kế và cài đặt và cách gán chúng từ các cơ chế phân tích
- Tìm hiểu về subsystems aà interfaces và vai trò của chúng trong kiến trúc hệ thống
- Mô tả quy trình xác định các interfaces và subsystems
- Tìm hiểu các lý lẽ và các cở sở hỗ trợ cho các quyết định về kiến trúc

Vị trí của Phân tích kiến trúc



Tổng quan về phân tích kiến trúc



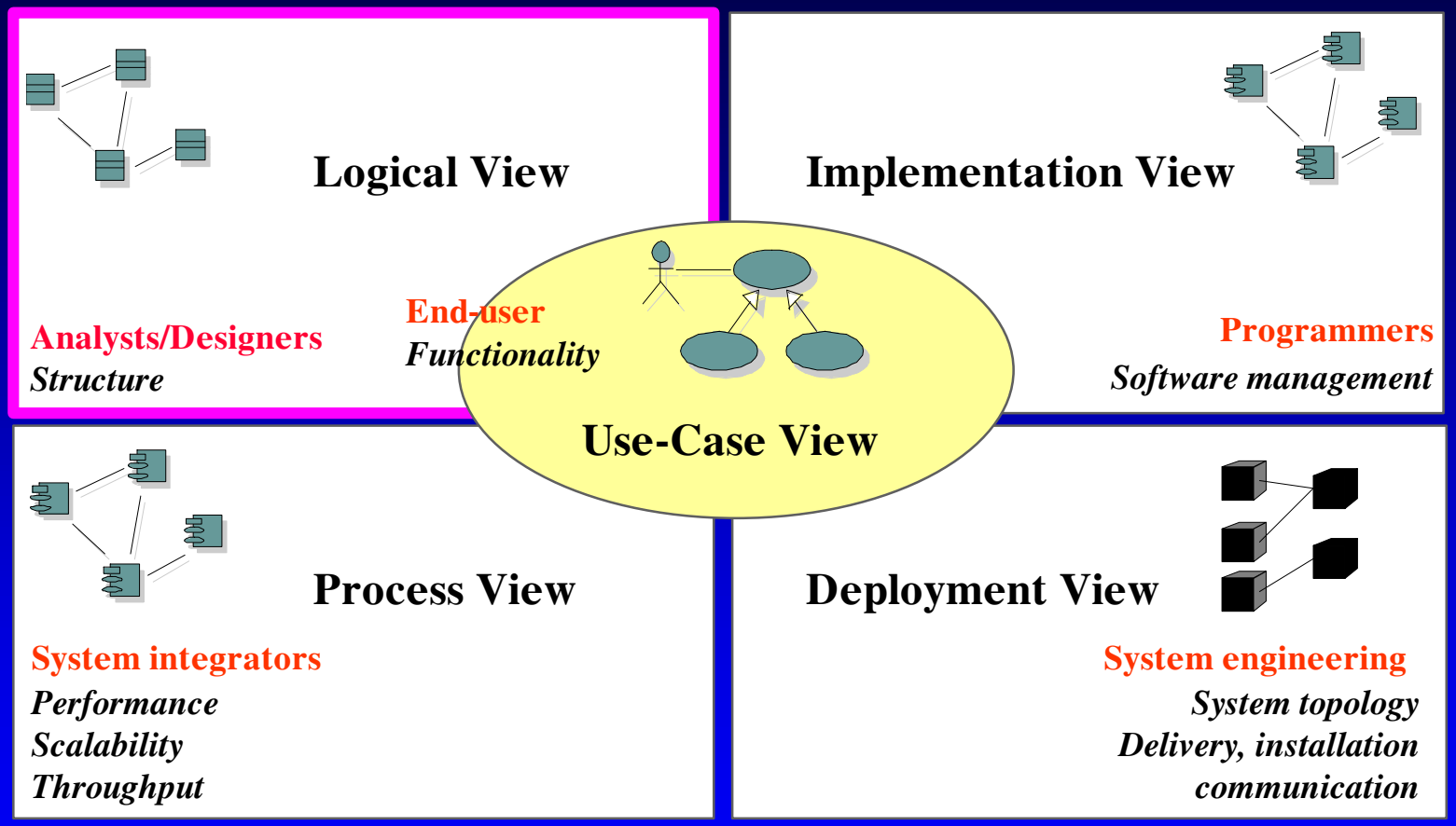
Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

Mô hình kiến trúc "4+1 View"

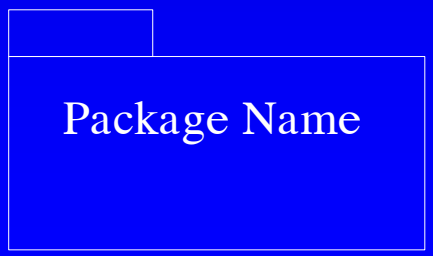


Logical View là phần có ý nghĩa về mặt kiến trúc của Design Model

Nhắc lại: Class và Package

- Thế nào là class?
  - Là mô tả của một tập các đối tượng cùng chia xẻ các trách nhiệm, mối quan hệ, các tác vụ, thuộc tính, và ngữ nghĩa.

- Thế nào là package?
  - Là một cơ chế dùng chung để nhóm các phần tử thành các nhóm
  - Là một phần tử của môhình có thể chứa bên trong các phần tử khác



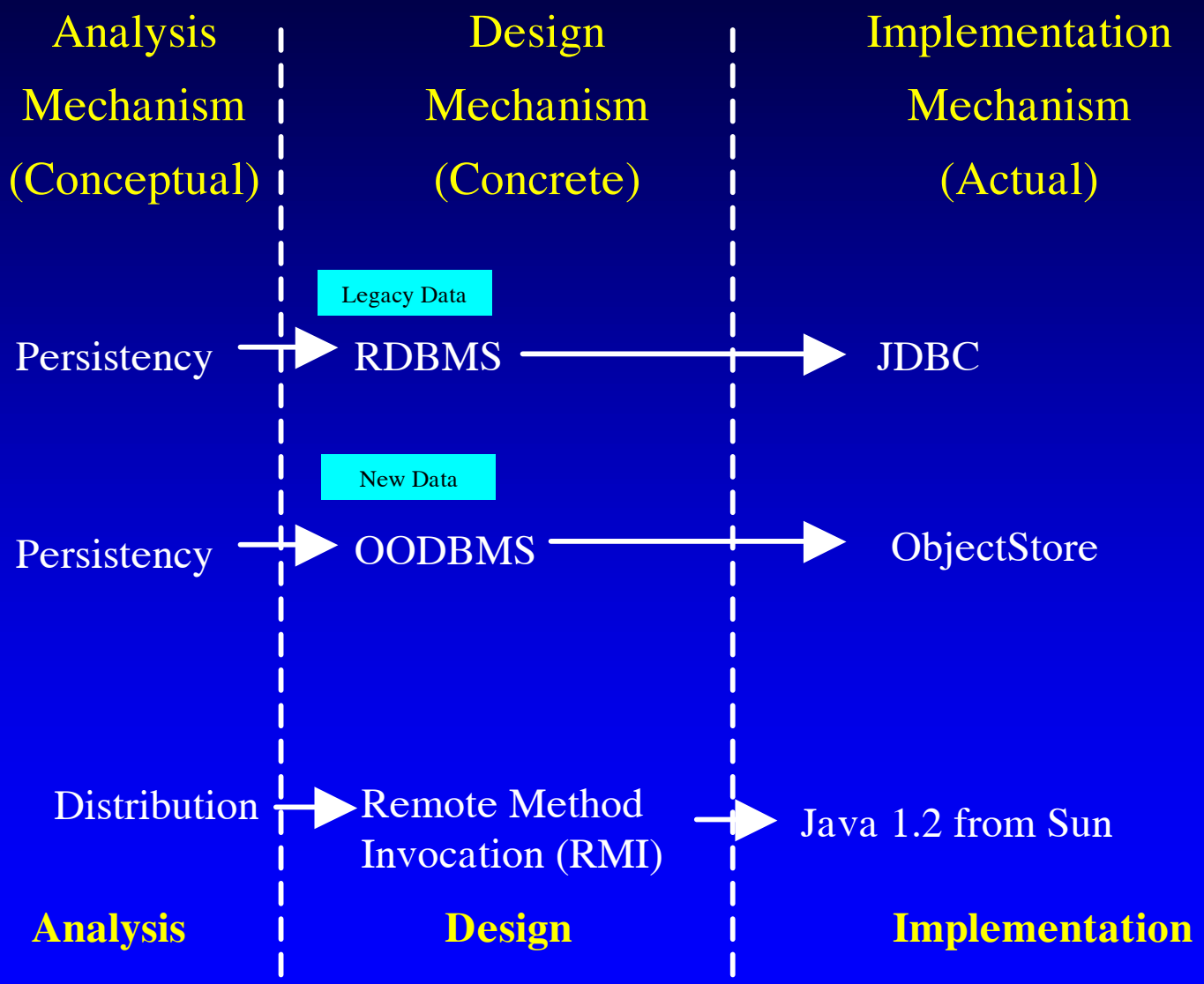
Các Global Package

- Toàn bộ package được sử dụng bởi tất cả các package khác
- Những package này được đánh dấu là **global**

Architectural Design Topics

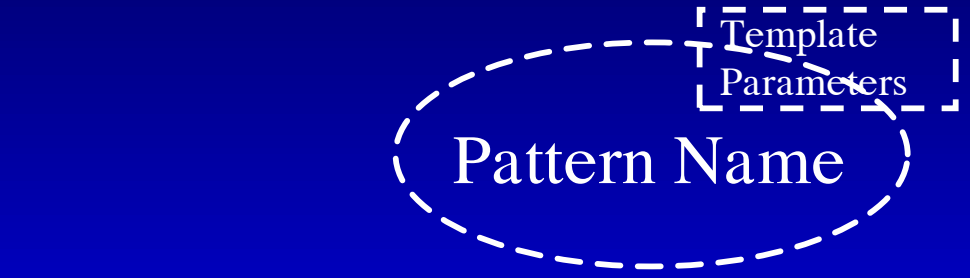
- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

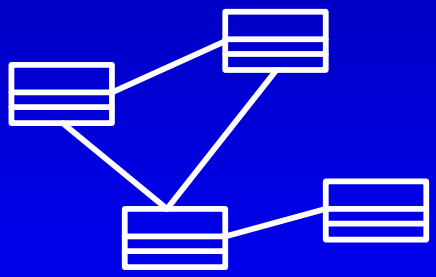
Các cơ chế thiết kế và cài đặt



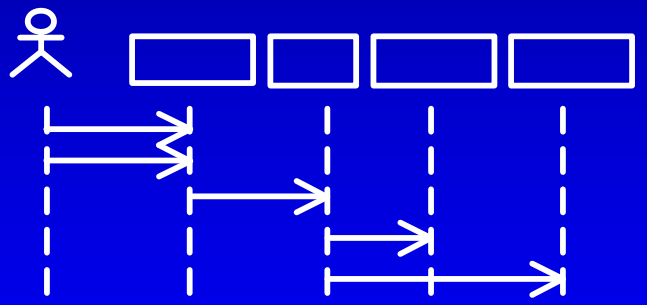
Documenting Architectural Mechanisms

- Các cơ chế kiến trúc có thể xem như các khuôn mẫu (pattern)





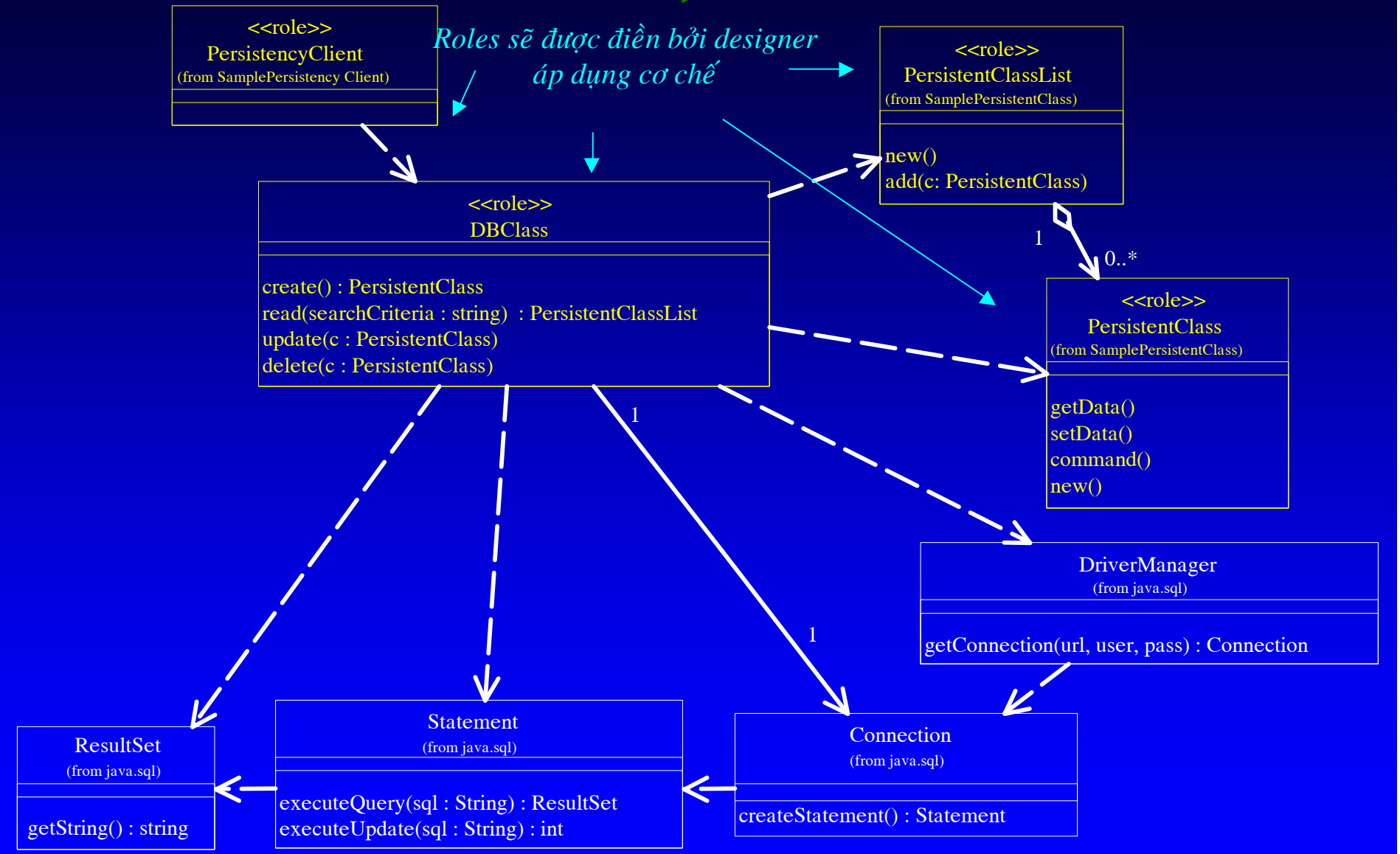
Structural Aspect



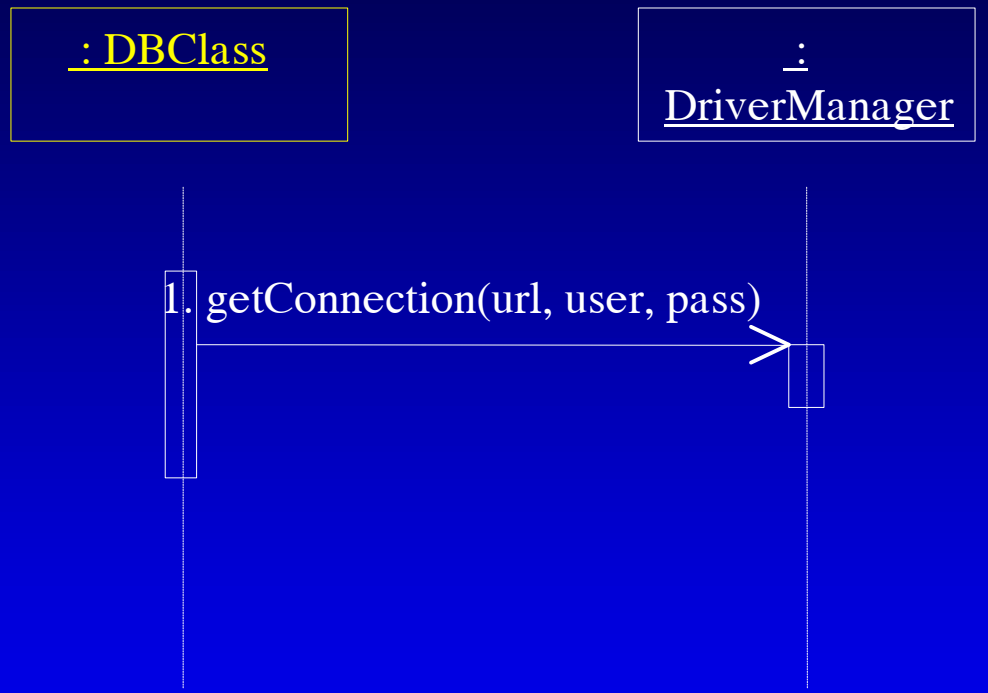
Behavioral Aspect

Được ghi nhận trong Design Guidelines

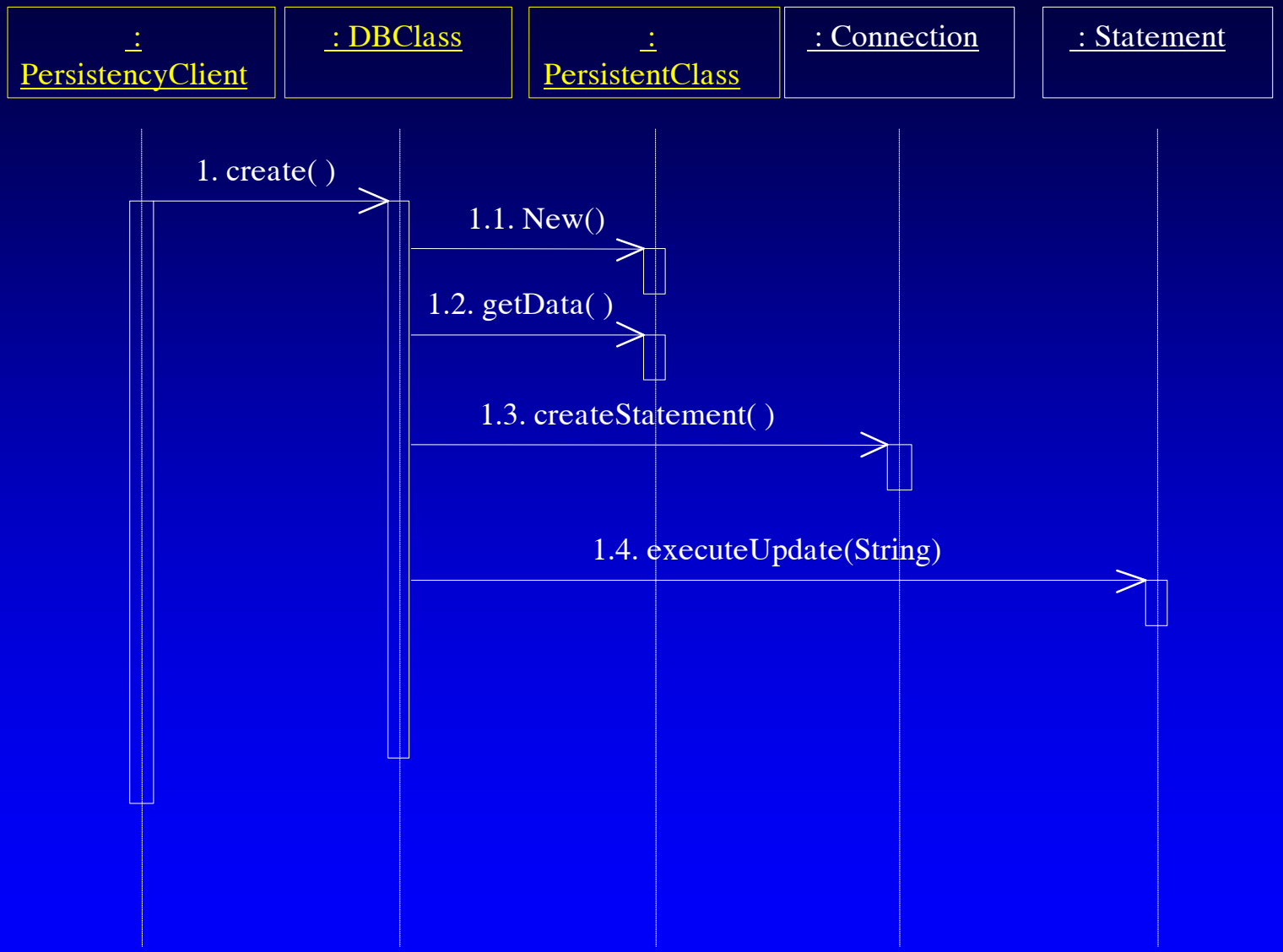
Ví dụ: Persistency: RDBMS: JDBC



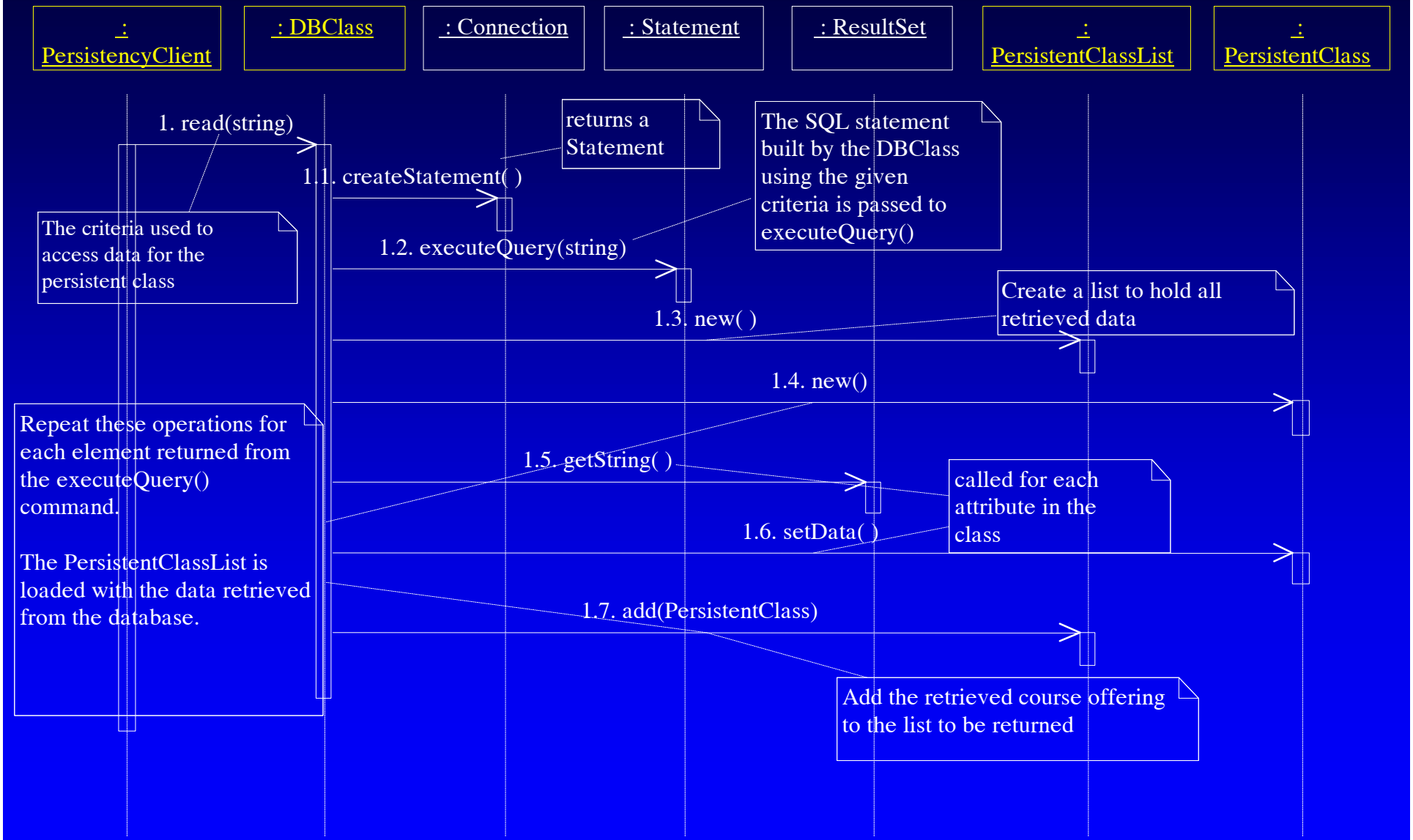
Ví dụ: Persistency: RDBMS: JDBC: Khởi tạo



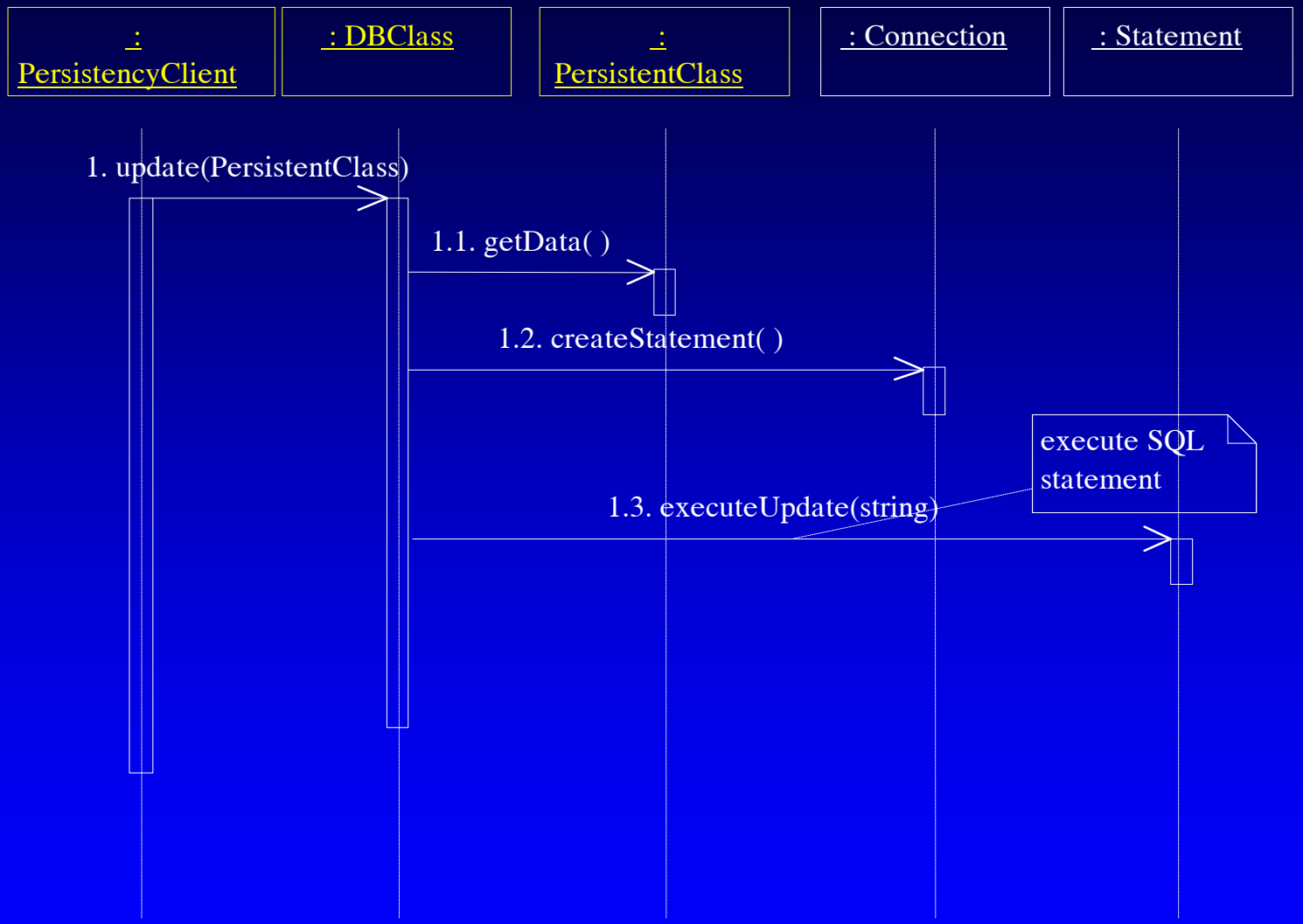
Ví dụ: Persistency: RDBMS: JDBC: Create



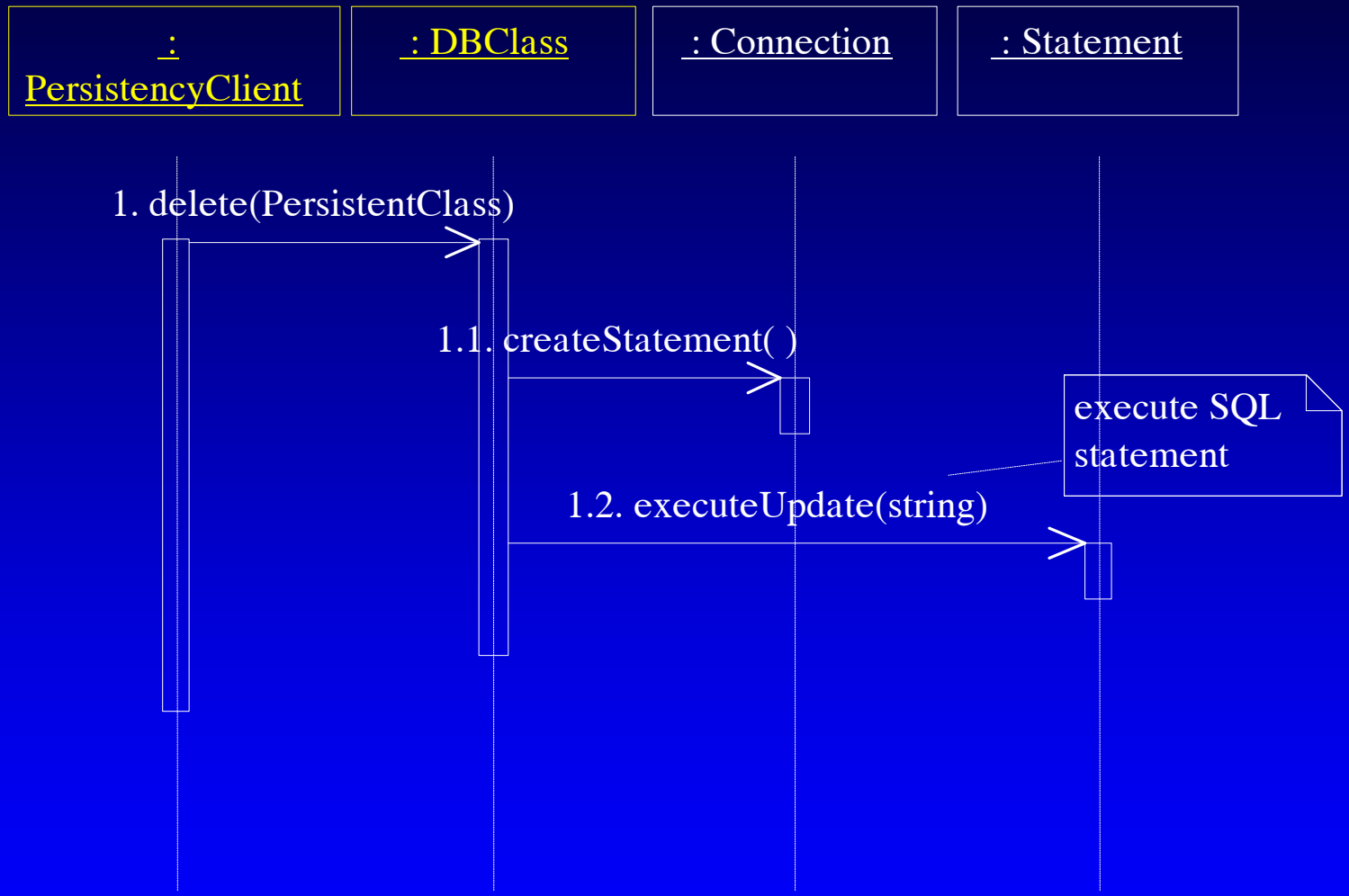
Ví dụ: Persistency: RDBMS: JDBC: Read



Ví dụ: Persistency: RDBMS: JDBC: Update



Ví dụ: Persistency: RDBMS: JDBC: Delete

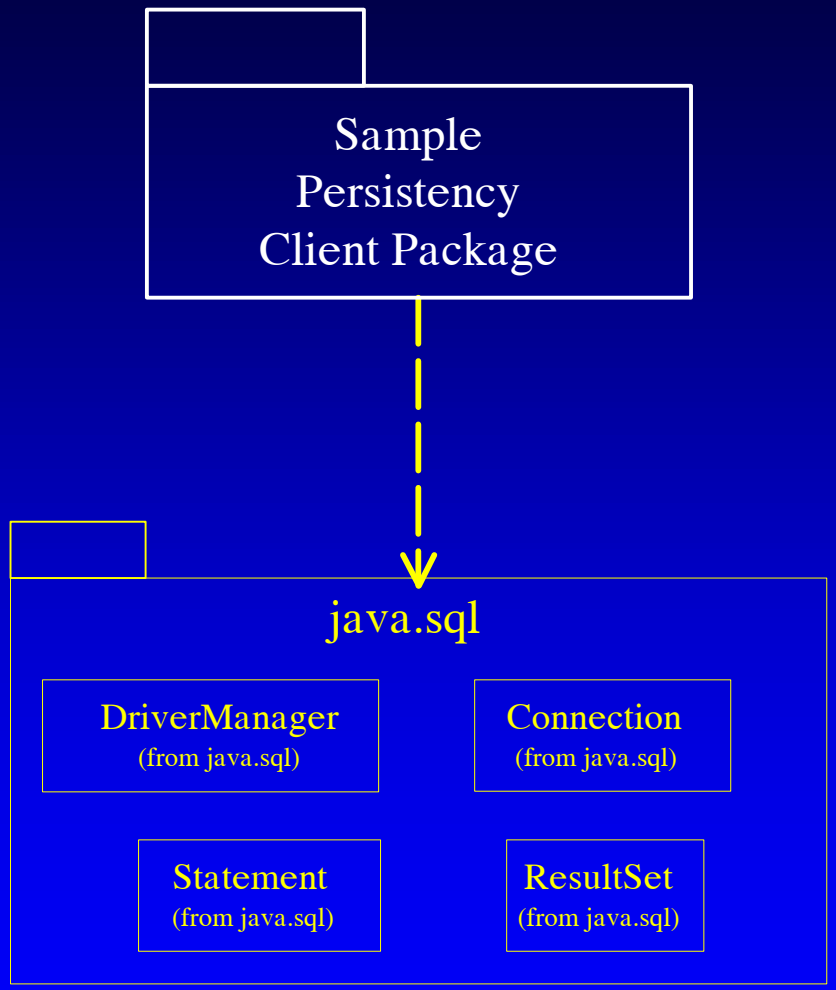


Tổng hợp JDBC: Các bước

- Cung cấp khả năng truy cập đến class libraries cần thiết để cài đặt JDBC
  - *Cung cấp java.sql package*
- Tạo các DBClass cần thiết
  - *Một DBClass / persistent class*
- Tích hợp các DBClass vào thiết kế
  - Xếp đặt vào package/layer
  - Thêm các mối quan hệ từ các persistency client
- Tạo/cập nhật interaction diagram để diễn tả:
  - Database initialization
  - Persistent class access: Create, Read, Update, Delete

Deferred

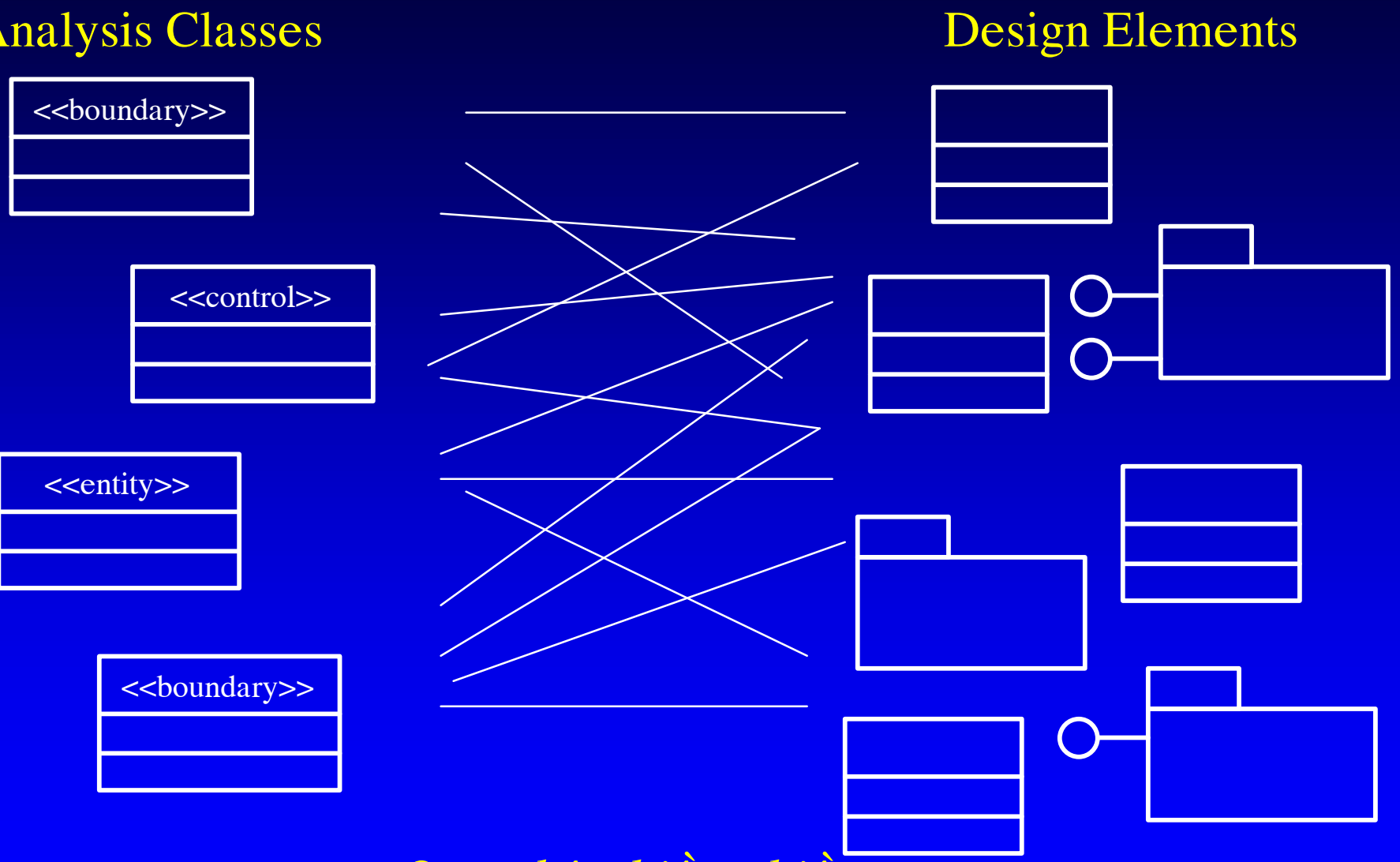
Ví dụ: Tổng hợp JDBC



Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

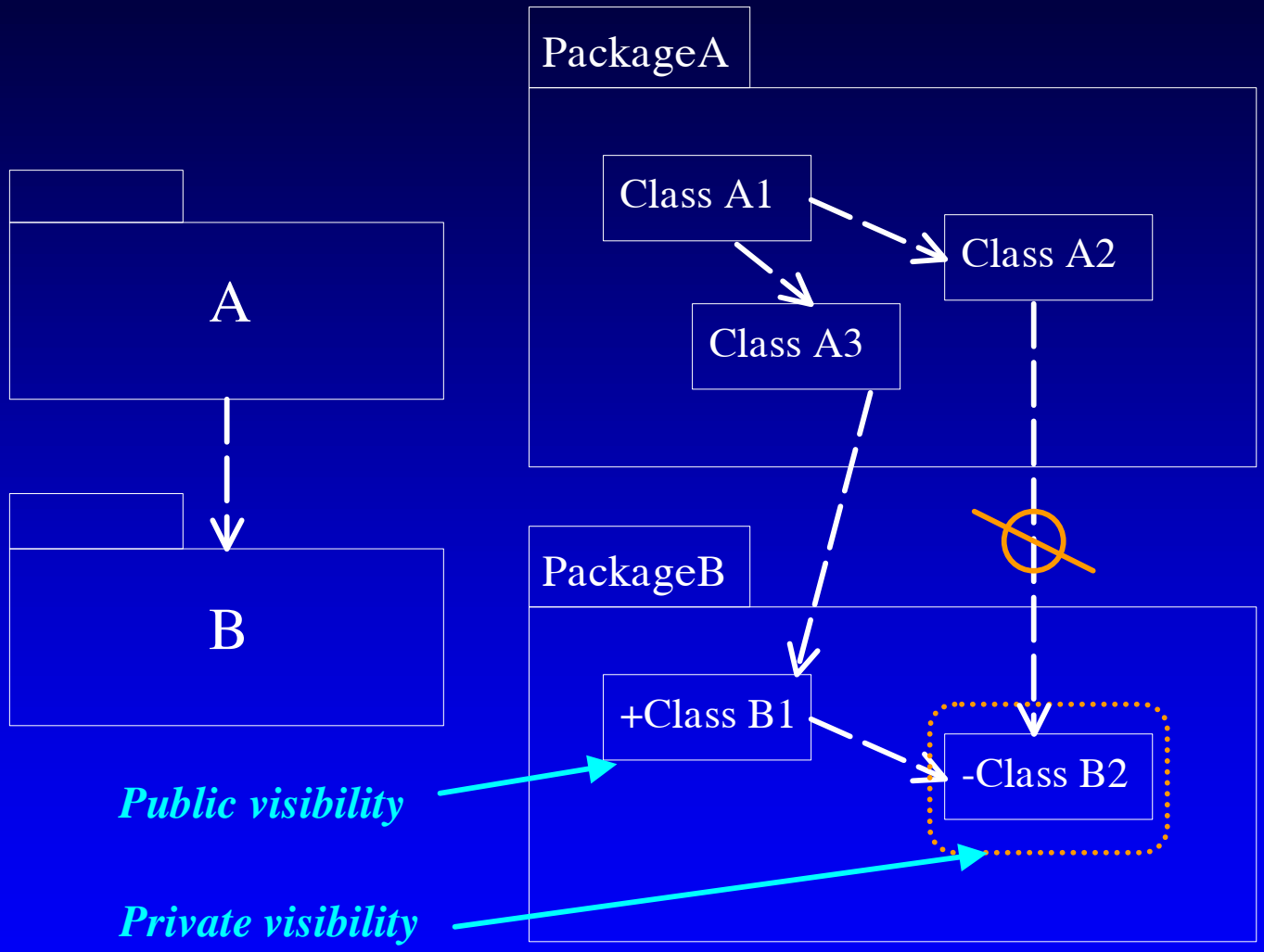
Từ Analysis Classes đến Design Elements



Xác định các Design Class

- Analysis class ánh xạ thẳng thành design class nếu:
  - Đơn giản
  - Biểu diễn một single logical abstraction
- Các analysis class phức tạp hơn có thể:
  - Tách thành nhiều class
  - Trở thành một package
  - Trở thành một subsystem (sẽ khảo sát sau)
  - Một tổ hợp bất kỳ …
- Các analysis class đơn giản có thể trở thành một design class

Các phụ thuộc Package: Tính khả kiến của các ptử

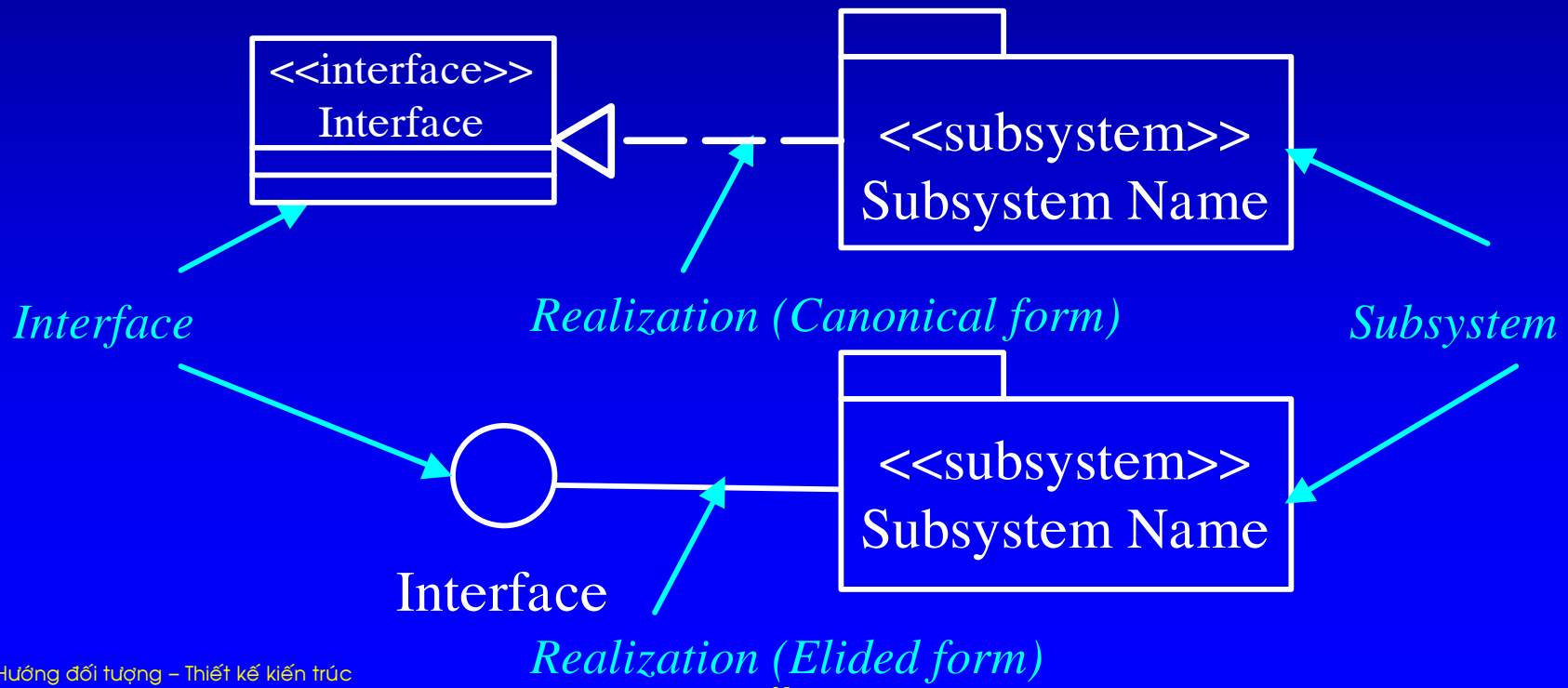


Chỉ các public class mới được tham chiếu từ bên ngoài package sở hữu nó

Nguyên lý OO : Encapsulation

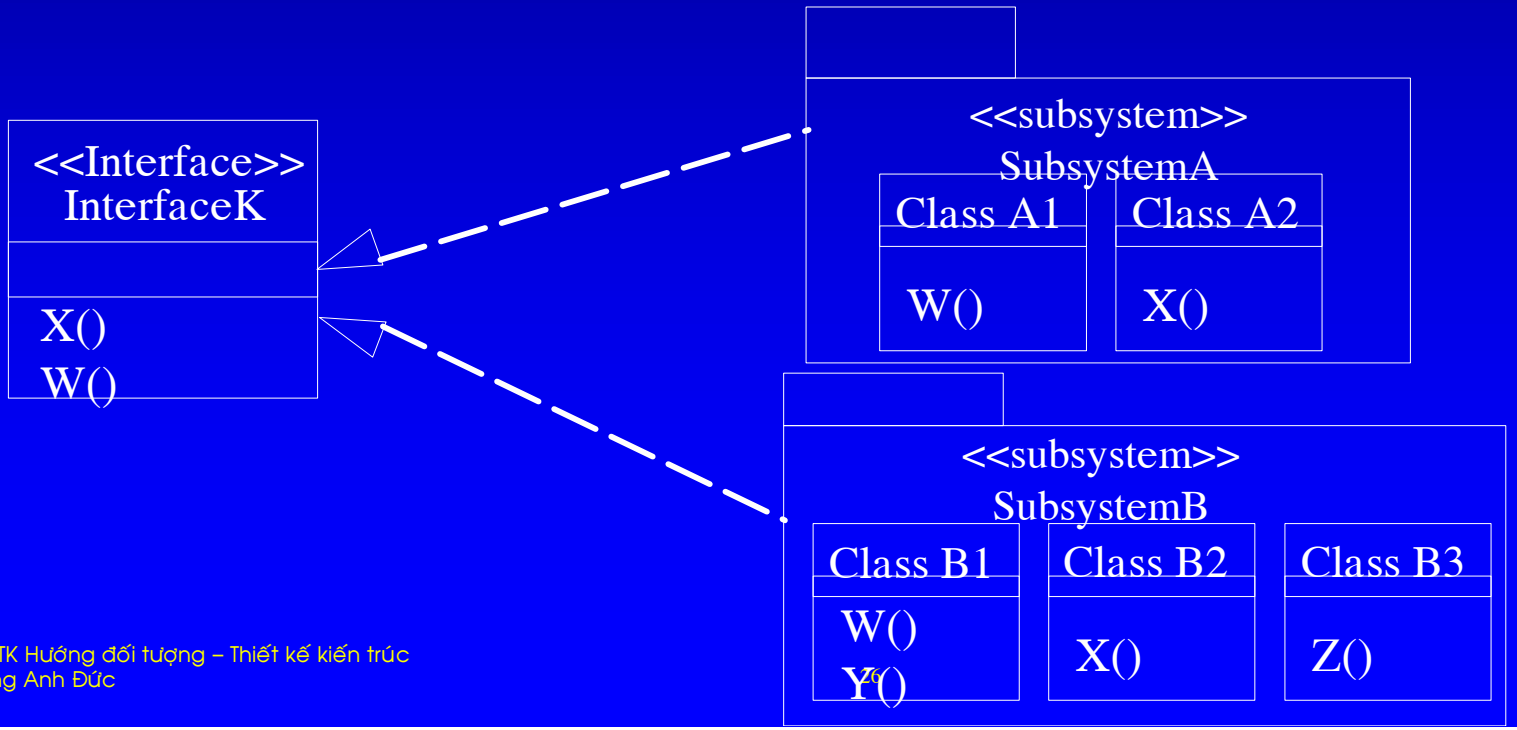
Nhắc lại: Subsystem và Interface

- Một dạng trung gian giữa package (có thể chứa các phần tử khác) và class (có hành vi)
- Hiện thực hoá 1 hoặc nhiều interface định nghĩa hành vi của nó



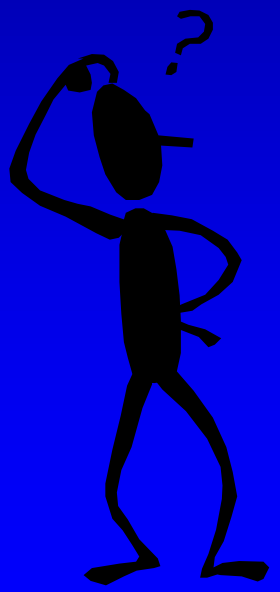
Subsystem và Interface (tt.)

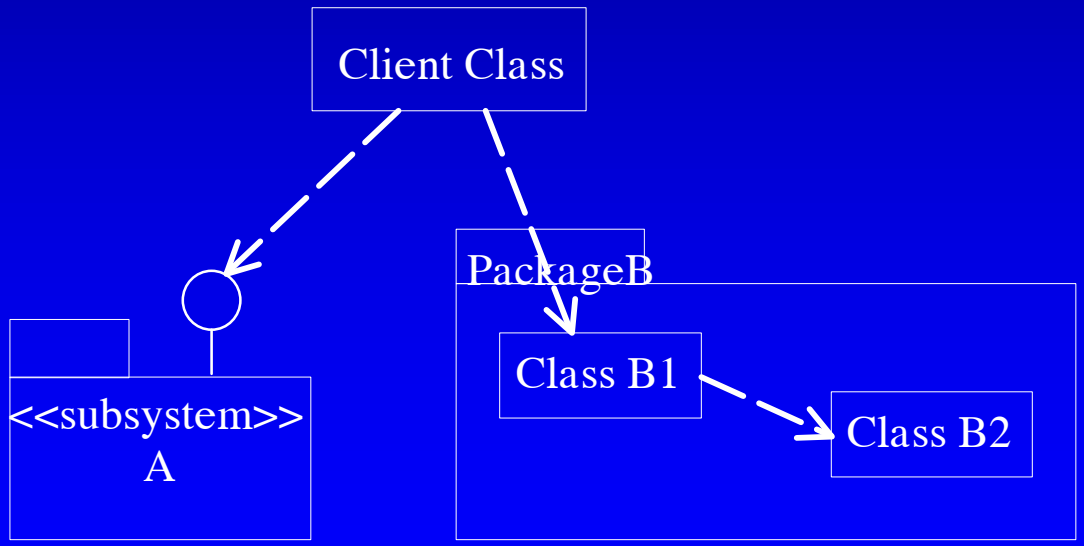
- Các Subsystem:
  - Hoàn toàn đóng gói hành vi
  - Thể hiện một khả năng hoàn toàn độc lập, với các interface rõ ràng (có tiềm năng tái sử dụng)
  - Mô hình hoá nhiều phương án cài đặt khác nhau



So sánh Package với Subsystem

- Subsystem cung cấp hành vi, package không
- Subsystem hoàn toàn đóng gói nội dung của nó, package thì không
- Subsystem dễ dàng được thay thế





Encapsulation là mấu chốt !

Cách dùng Subsystem

- Subsystem có thể dùng để chia system thành các phần độc lập về:
  - Thứ tự, cấu hình, hoặc vận chuyển
  - Phát triển, chừng nào mà interface còn chưa thay đổi
  - Triển khai trên các node tính toán phân tán
  - Thay đổi mà không phá vỡ các phần khác của system
- Subsystem còn có thể dùng để:
  - Phần chia system thành các đơn vị cung cấp độ bảo mật cao đối với các tài nguyên then chốt
  - Biểu diễn các sản phẩm có sẵn hoặc các system nằm ngoài bản thiết kế (chẳng hạn như các component)

Các Subsystem nâng cao mức độ trừu tượng

Các gợi ý giúp xác định các Subsystem

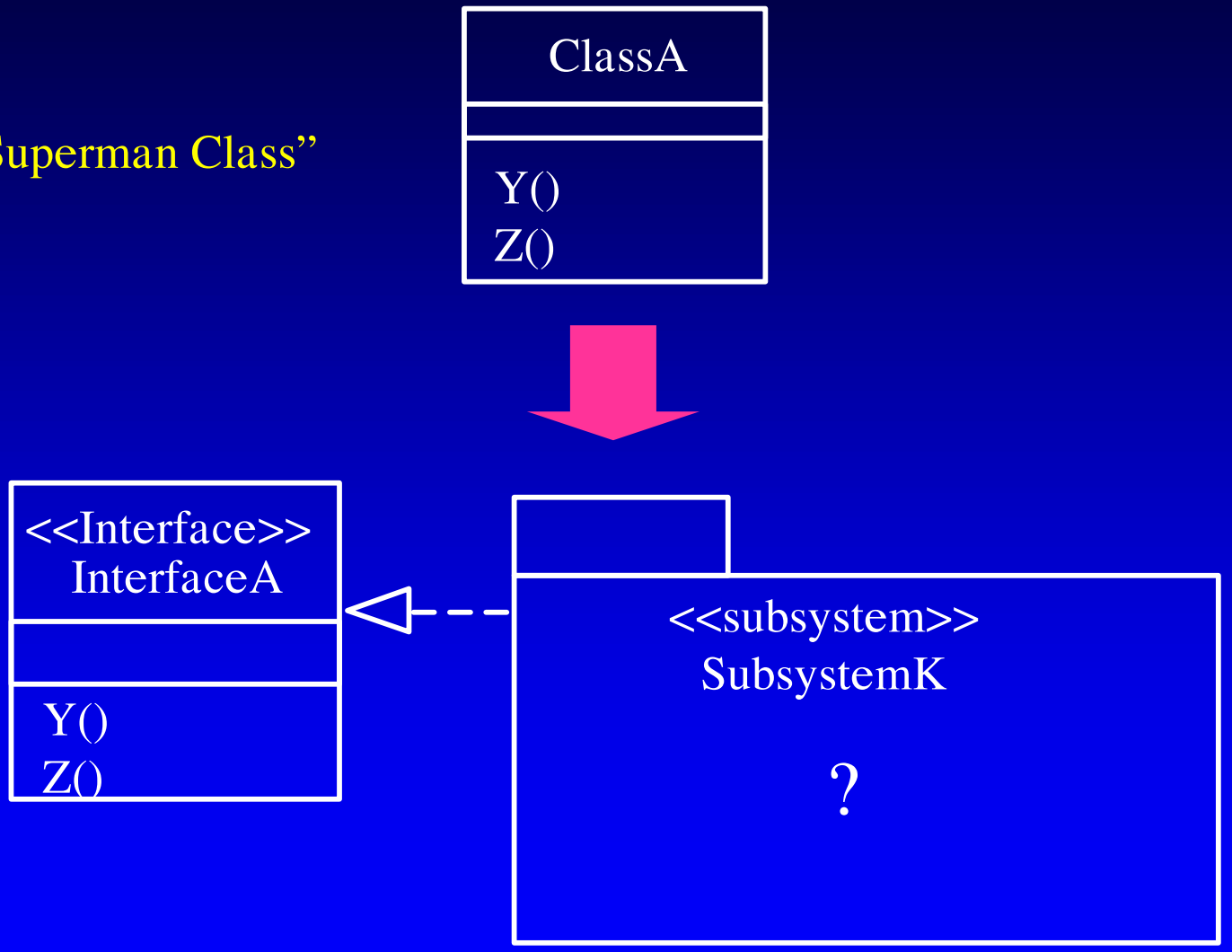
- Tìm kiếm sự cộng tác giữa các object
- Tìm kiếm sự tuỳ chọn
- Chú ý user interface của system
- Chú ý các Actor
- Tìm kiếm sự kết dính giữa các class
- Xem xét sự thay thế (các mức độ service)
- Xem xét sự phân bố
- Xem xét sự kém bền vững

Các Subsystem tiềm năng

- Các Analysis classe có thể tiến hoá thành các subsystem
  - Các Class cung cấp các dịch vụ và/hoặc các tiện ích trọn vẹn
  - Các Boundary class (user interface và external system interface)
- Các sản phẩm sẵn có hoặc các system nằm ngoài thiết kế
  - Communication software
  - Database access support
  - Các kiểu và cấu trúc dữ liệu
  - Các tiện ích dùng chung
  - Các sản phẩm ứng dụng đặc thù

Identifying Subsystems

"Superman Class"



Identifying Interfaces

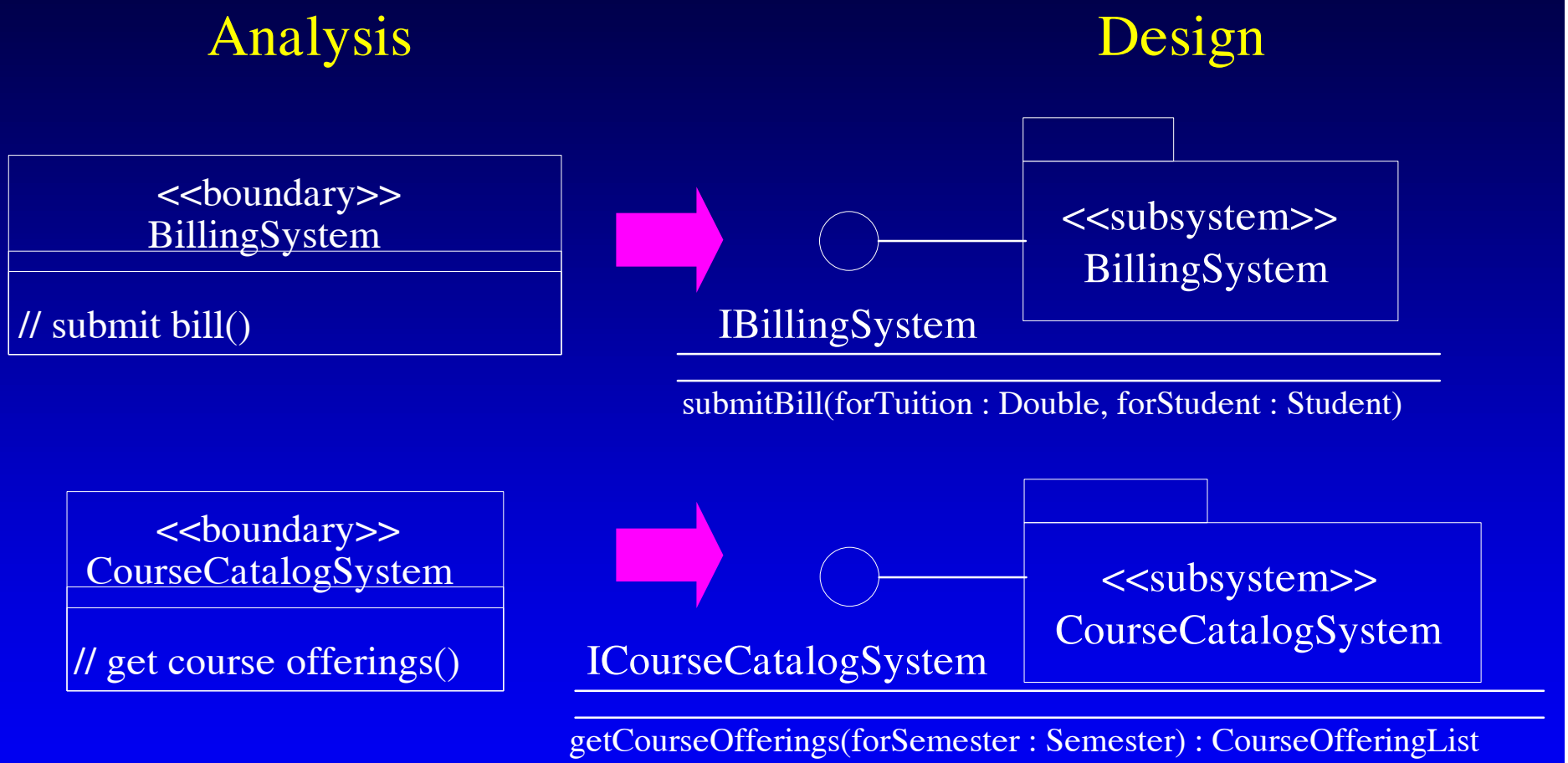
- Mục đích
  - Để xác định interface của các subsystem dựa trên nhiệmvụ của chúng
- Các bước
  - Xác định một tập các interface tiềm năng cho tất cả các subsystem
  - Tìm kiếm sự tương tự giữa các interface
  - Định nghĩa các phụ thuộc của interface
  - Ánh xạ các interface đến các subsystem
  - Định nghĩa hành vi được mô tả bởi interface
  - Đoán gói các interface

Các interface ổn định, thiết kế tốt là nền tảng cho một kiến trúc bền vững

Interface Guidelines

- Đặt tên cho Interface
  - Thể hnện vai trò trong system
- Mô tả Interface
  - Chuyển tải các nhiệm vụ
- Định nghĩa Operation
  - Tên phải phản ánh đúng kết quả của operation
  - Mô tả operation được thực hiện, tất cả các tham số và kết quả
- Interface documentation
  - Package các thông tin hỗ trợ: sequence aà state diagrams, kế hoạch kiểm chứng, …

Ví dụ: Các Design Subsystem

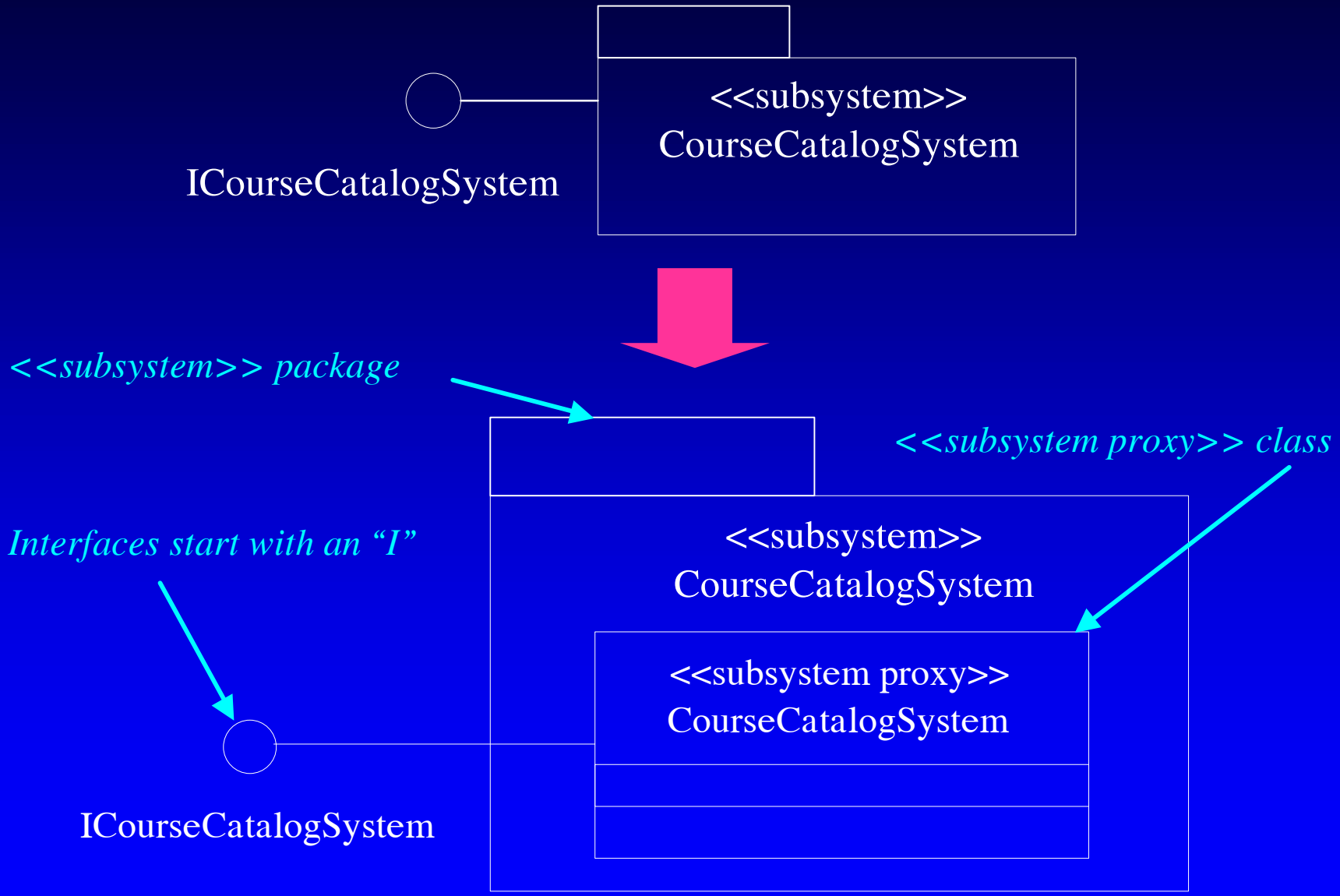


Tất cả các analysis class khác đều chuyển thành design class

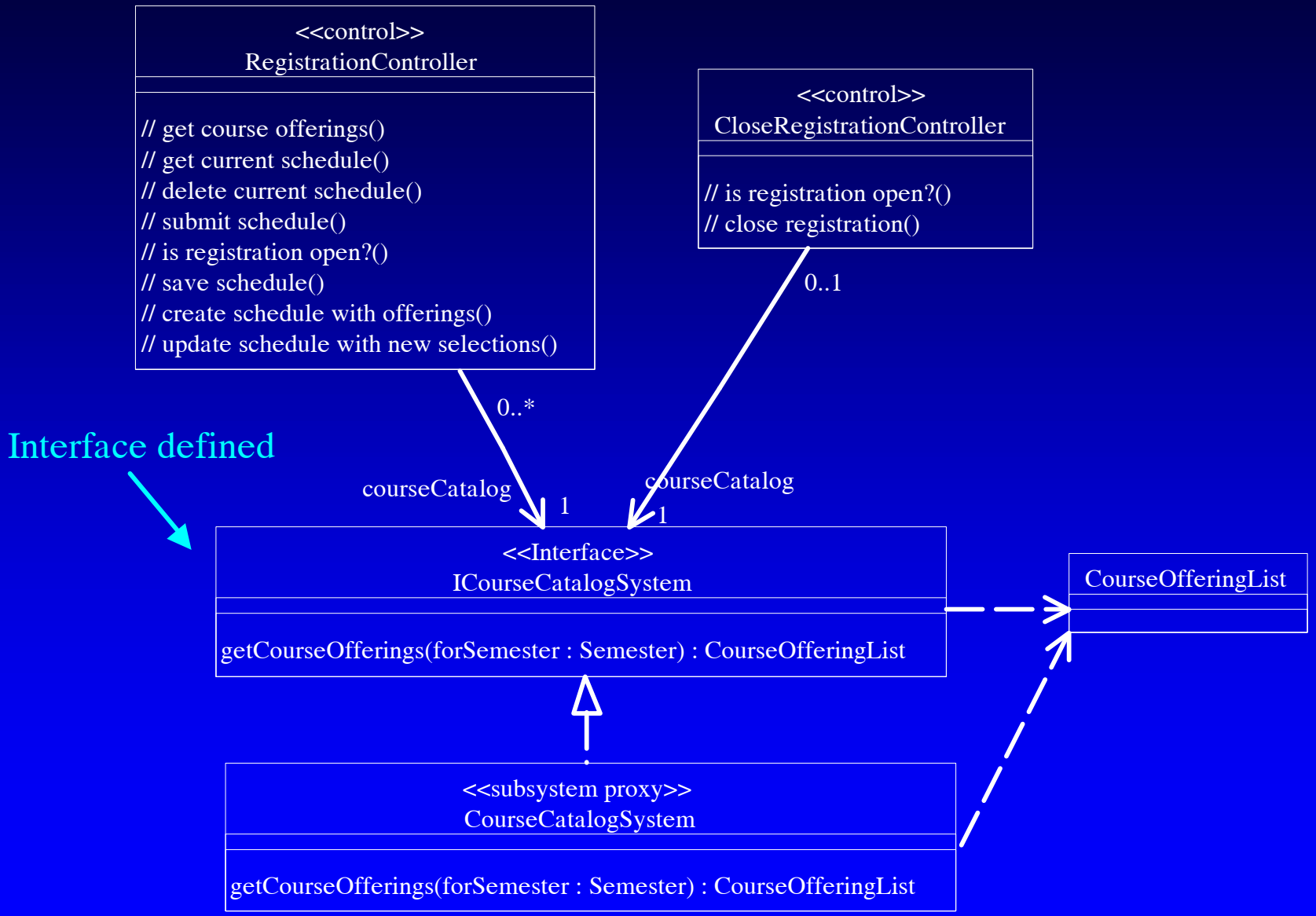
Via dụ: Analysis-Class-To-Design-Element Map

Analysis Class	Design Element
CourseCatalogSystem	CourseCatalogSystem Subsystem
BillingSystem	BillingSystem Subsystem
All other analysis classes map directly to design classes	

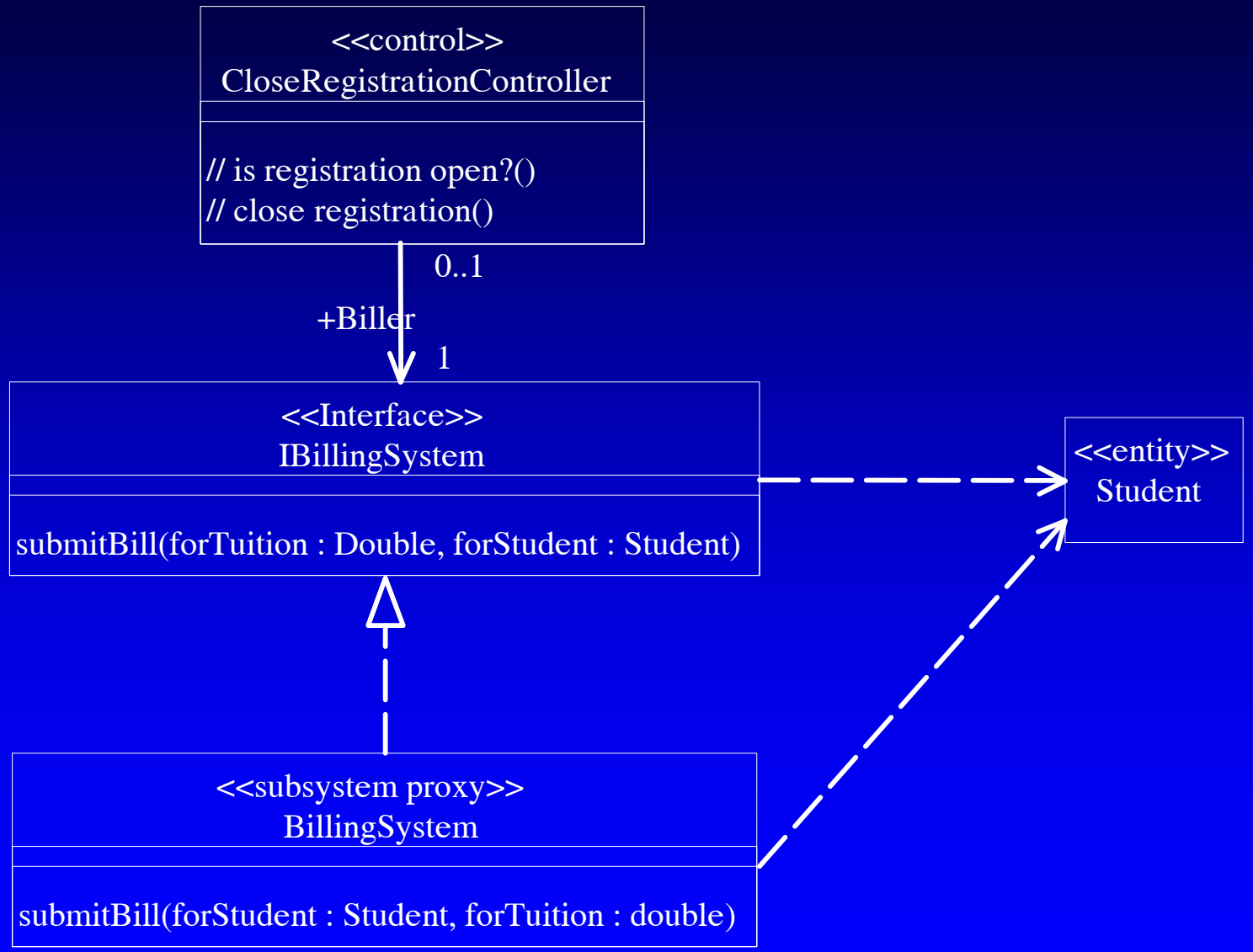
Qui ước mô hình hoá: Subsystem và Interface



Ví dụ: Subsystem Context: CourseCatalogSystem



Ví dụ: Subsystem Context: Billing System



Bài tập: Architectural Design, phần 1

- Cho biết các vấn đề sau:
  - Các analysis class và mối quan hệ của chúng

Bài tập: Architectural Design, phần 1 (tt.)

- Hãy xác định:
  - Các Subsystem, các interface và các quan hệ của chúng với các phần tử thiết kế khác
  - Ánh xạ các analysis class thành các phần tử thiết kế

(còn tiếp)

Bài tập: Architectural Design, Part 1 (tt.)

- Hãy xây dựng các lược đồ sau:
  - Với mỗi subsystem, xây dựng 1 subsystem context class diagram
  - Xây dựng bảng ánh xạ các analysis class thành các phần tử thiết kế (design element)

Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

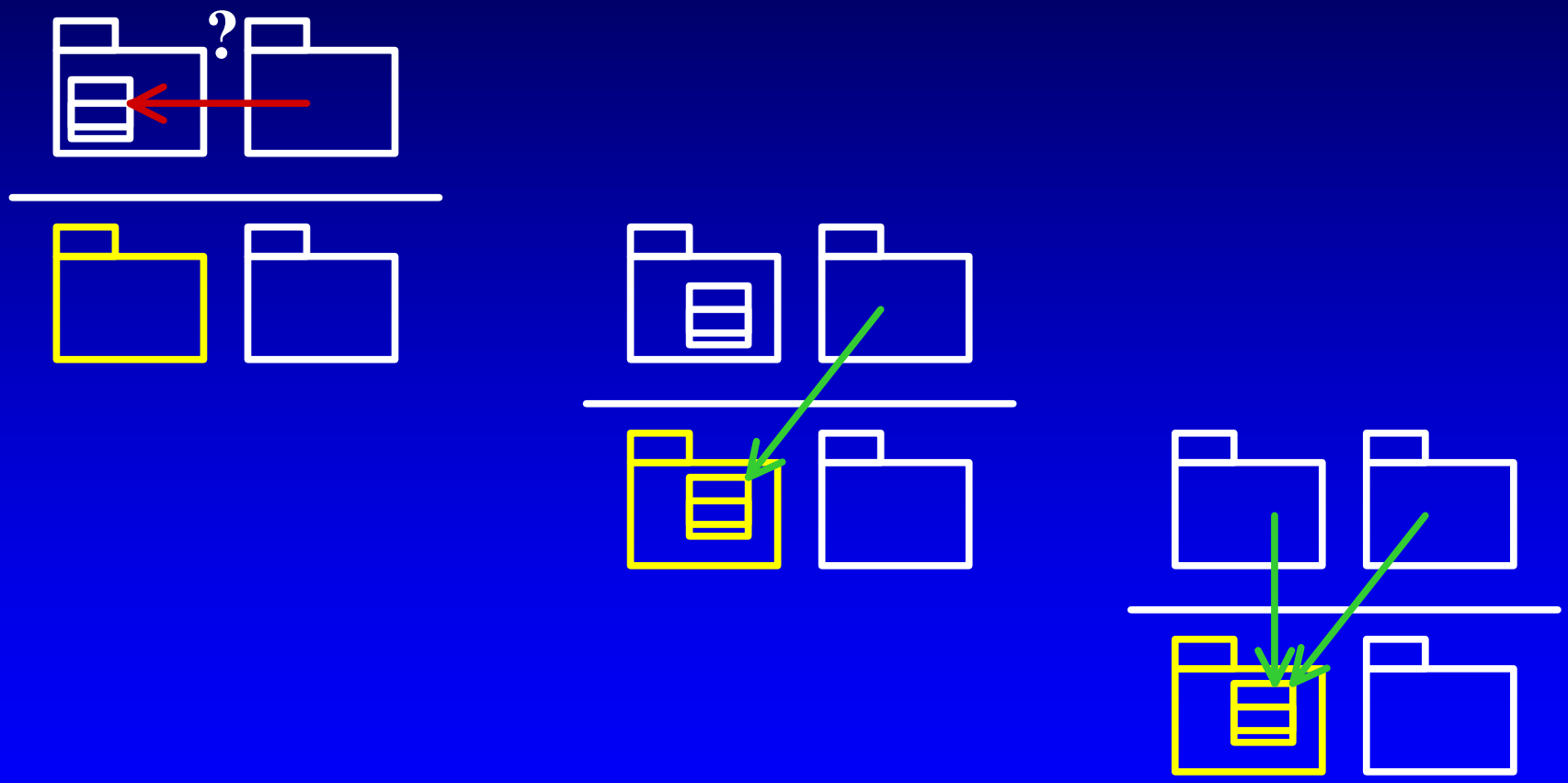
Xác định các khả năng dùng lại

- Mục đích
  - Để xác định nơi đâu có thể dùng lại các subsystem hay các component đã xây dựng dựa trên interface của chúng.
- Các bước
  - Tìm kiếm các interface tương tự nhau
  - Hiệu chỉnh các interface mới để phù hợp hơn
  - Thay thế các interface cần có bằng các interface có sẵn
  - Ánh xạ các subsystem cần có với các component có sẵn

Những cơ hội dùng lại

- Bên trong hệ thống đang xây dựng:
  - Nhận biết sự giống nhau giữa các package và các subsystem
- Bên ngoài hệ thống đang xây dựng:
  - Các component thương mại
  - Các component từ các ứng dụng đã xây dựng trước đó
  - Các component đã được **reverse engineered**

Cơ hội dùng lại ngay bên trong hệ thống

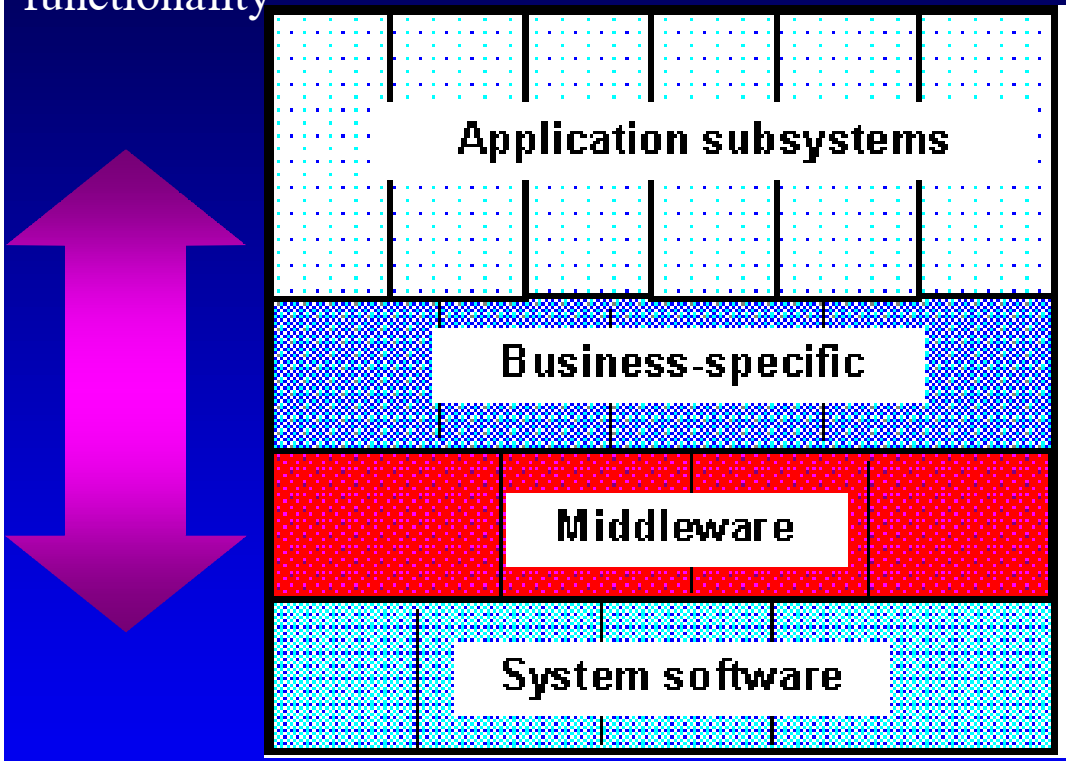


Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

Hướng tiếp cận phân lớp truyền thống

Specific functionality



Distinct application subsystem that make up an application - contains the value adding software developed by the organization.

Business specific - contains a number of reusable sybsystems specific to the type of business.

Middleware - offers subsystems for utility classes and platform-independent services for distributed object computing in heterogeneous environments and so on.

System software - contains the software for the actual infrastructure such as operating systems, interfaces to specific hardware, device drivers and so on.

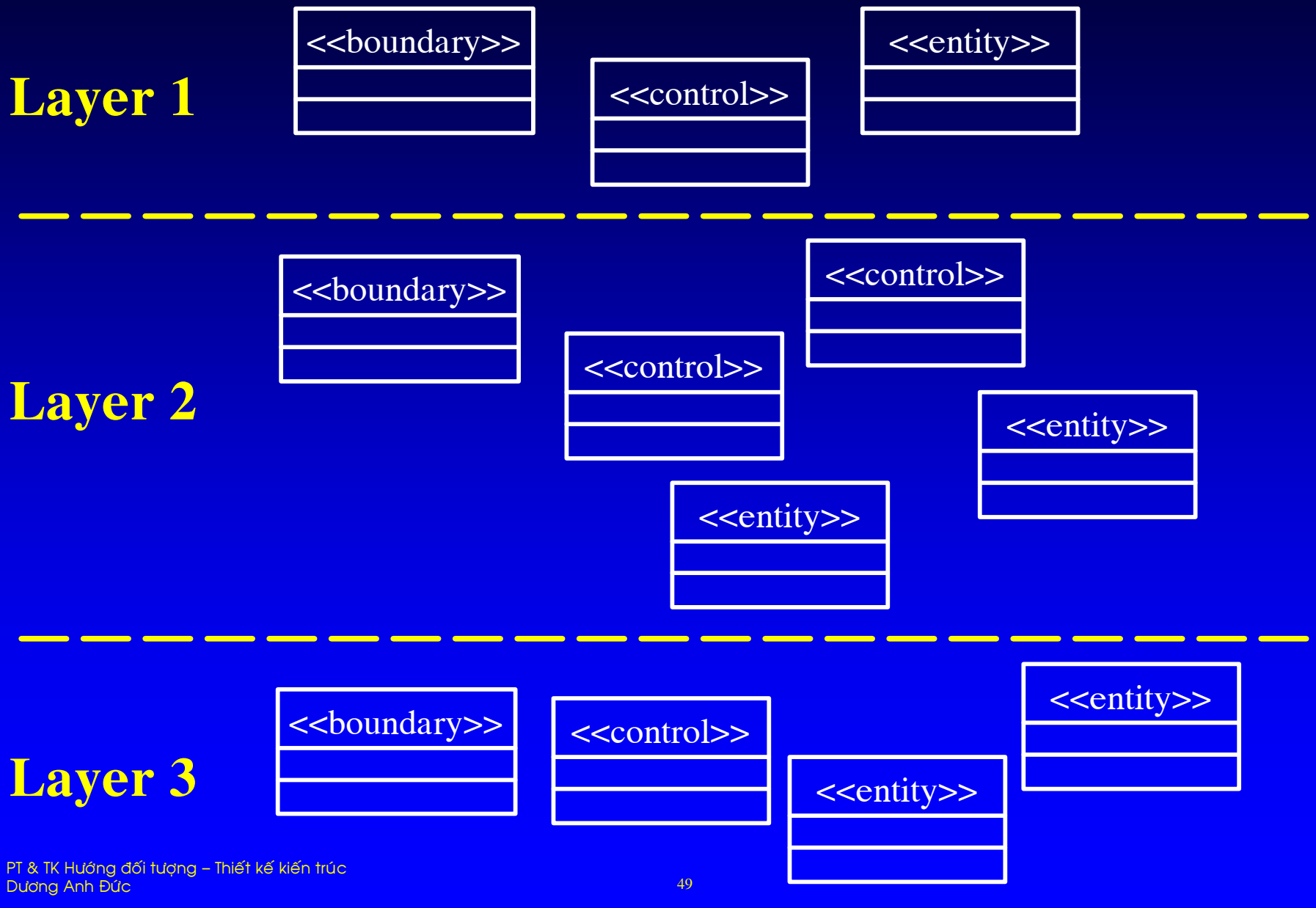
General functionality

Layering Guidelines

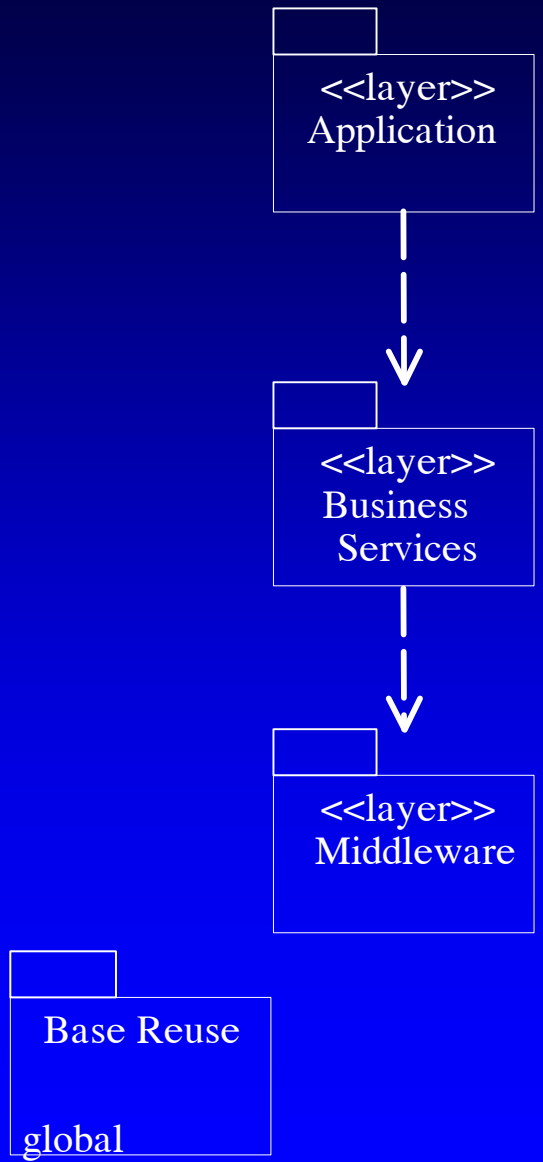
- Tính khả kiến
  - Chỉ có các phụ thuộc giữa layer hiện tại và layer kế
- Tính dễ thay đổi
  - Các layer ngoài bị thay đổi khi y/c đ/v HT thay đổi
  - Các layer trong bị thay đổi khi môi trường hoạt động thay đổi
- Tính tổng quát
  - Các phần tử có chức năng tổng quát ở các layer thấp
- Số lượng các layer
  - Hệ thống nhỏ: 3-4 layer
  - Hệ thống phức tạp: 5-7 layer

Mục đích là giảm sự chồng lắp và tăng khả năng bảo trì nâng cấp

Các Design Element và Kiến trúc



Ví dụ: Architectural Layers

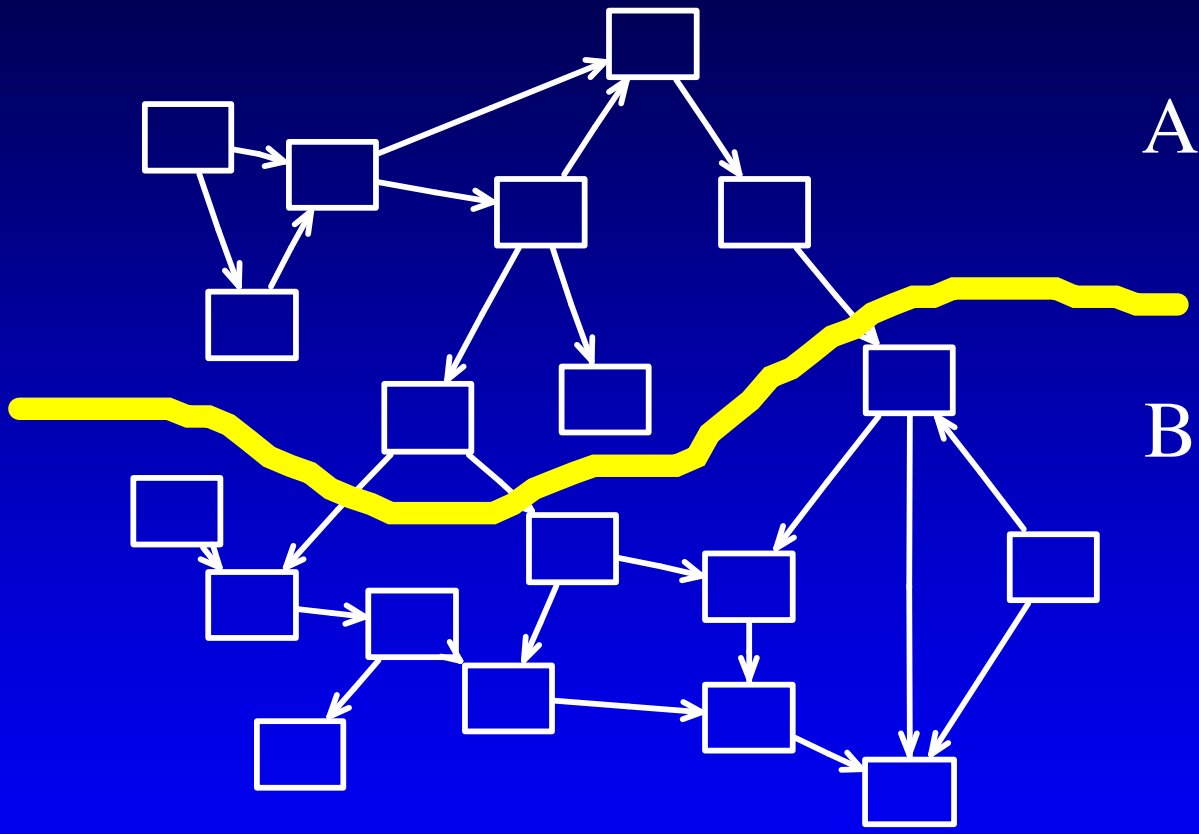


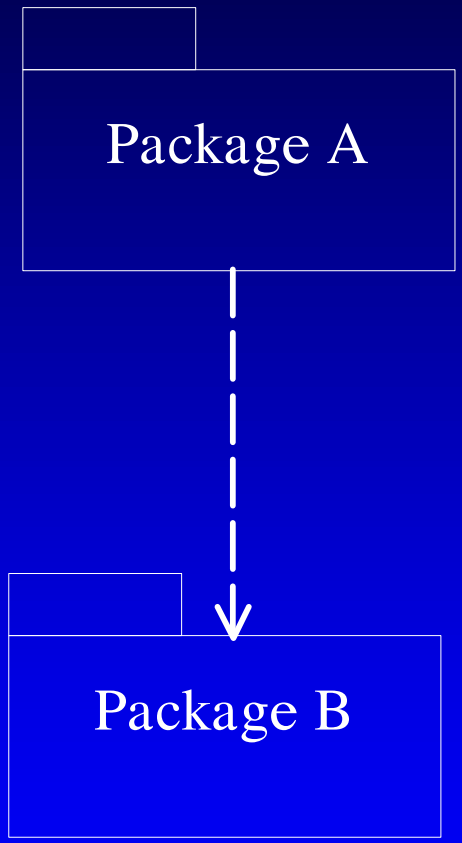
Các tiêu chuẩn phân chia

- Sự chồng lắp và kết dính
- Tổ chức của đơn vị sử dụng
- Năng lực và kỹ năng
- Sự phân bố của hệ thống
- Tính bảo mật
- Khả năng biến đổi

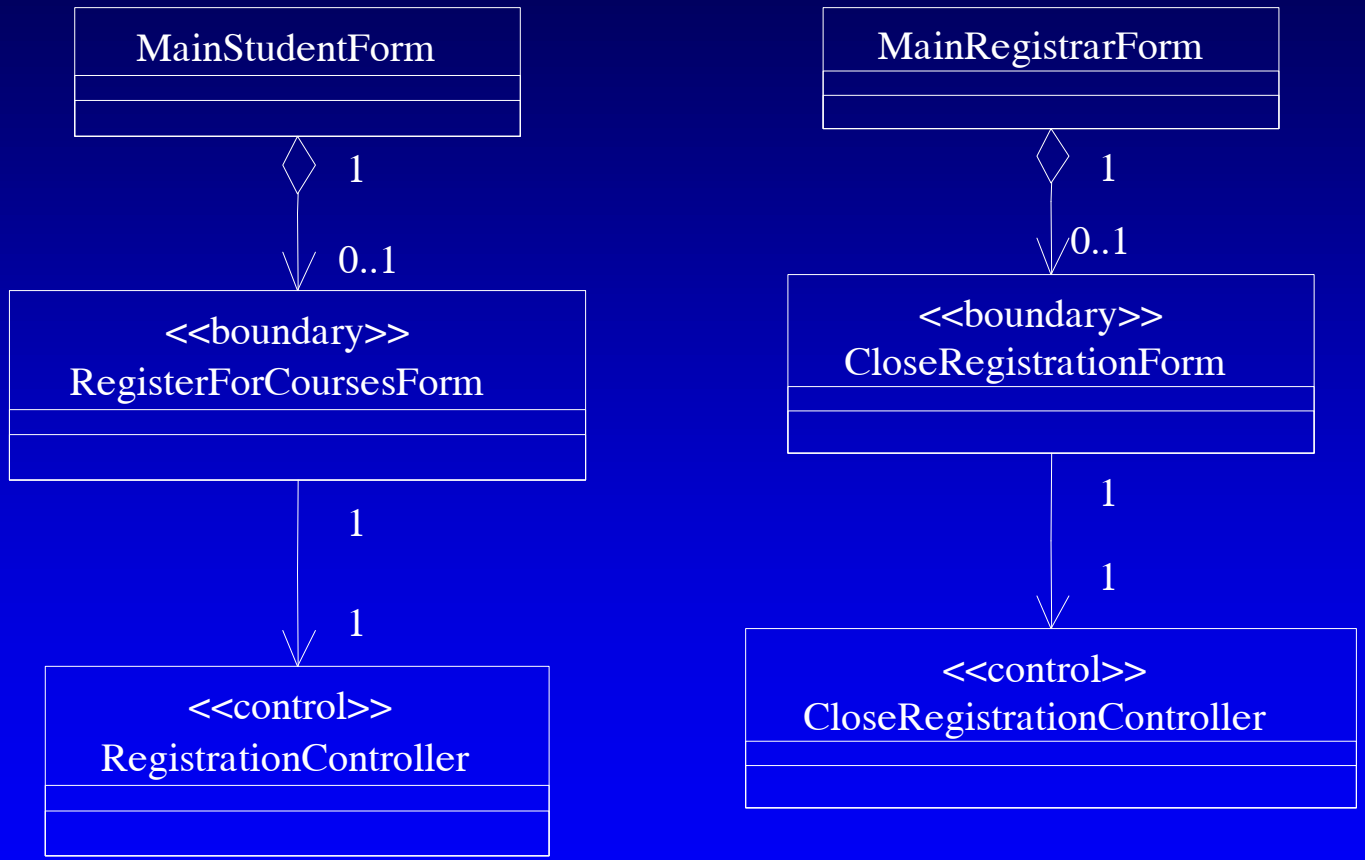
Hãy thử loại bỏ các phụ thuộc xoay vòng

Ví dụ: Partitioning

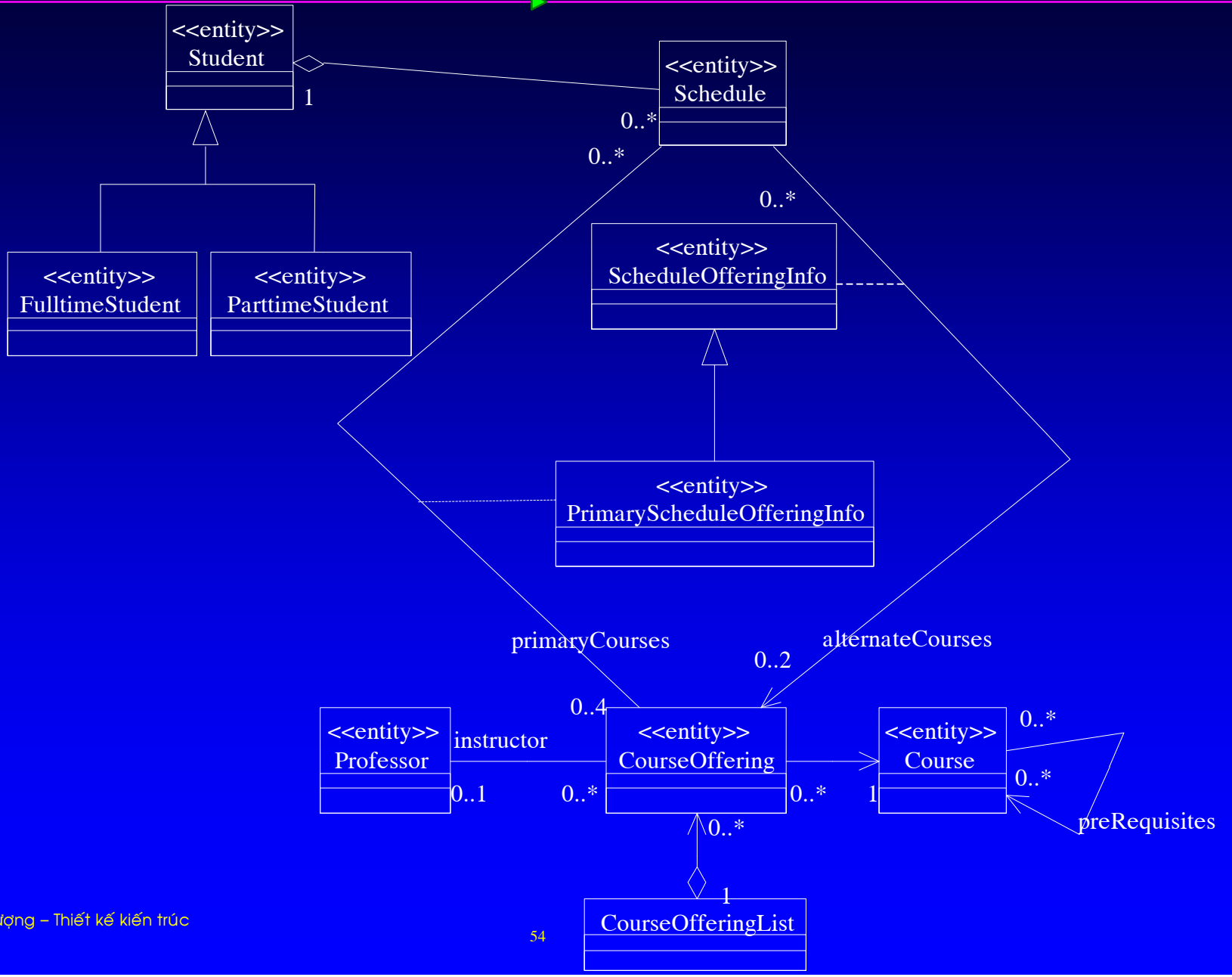




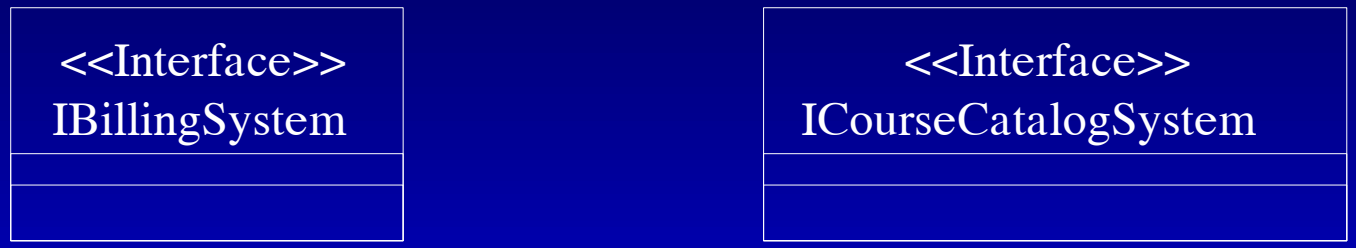
Ví dụ: Registration Package



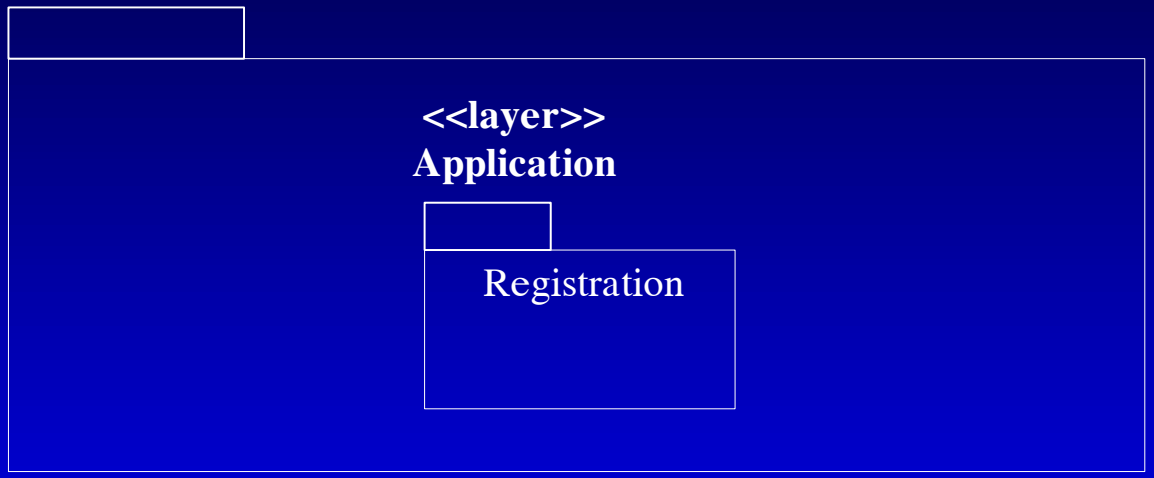
Vó dụ: University Artifacts Package



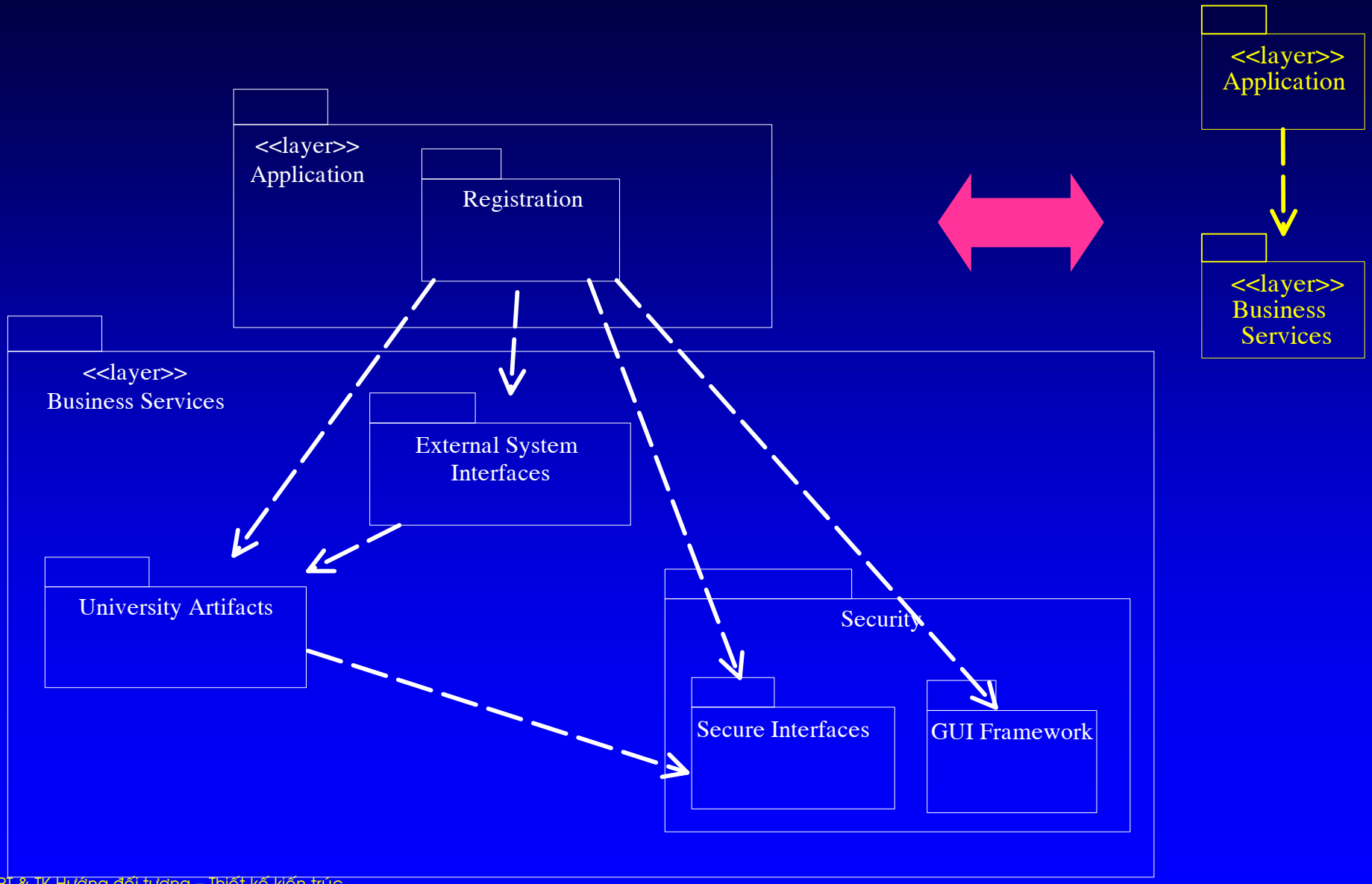
Ví dụ: External System Interfaces Package



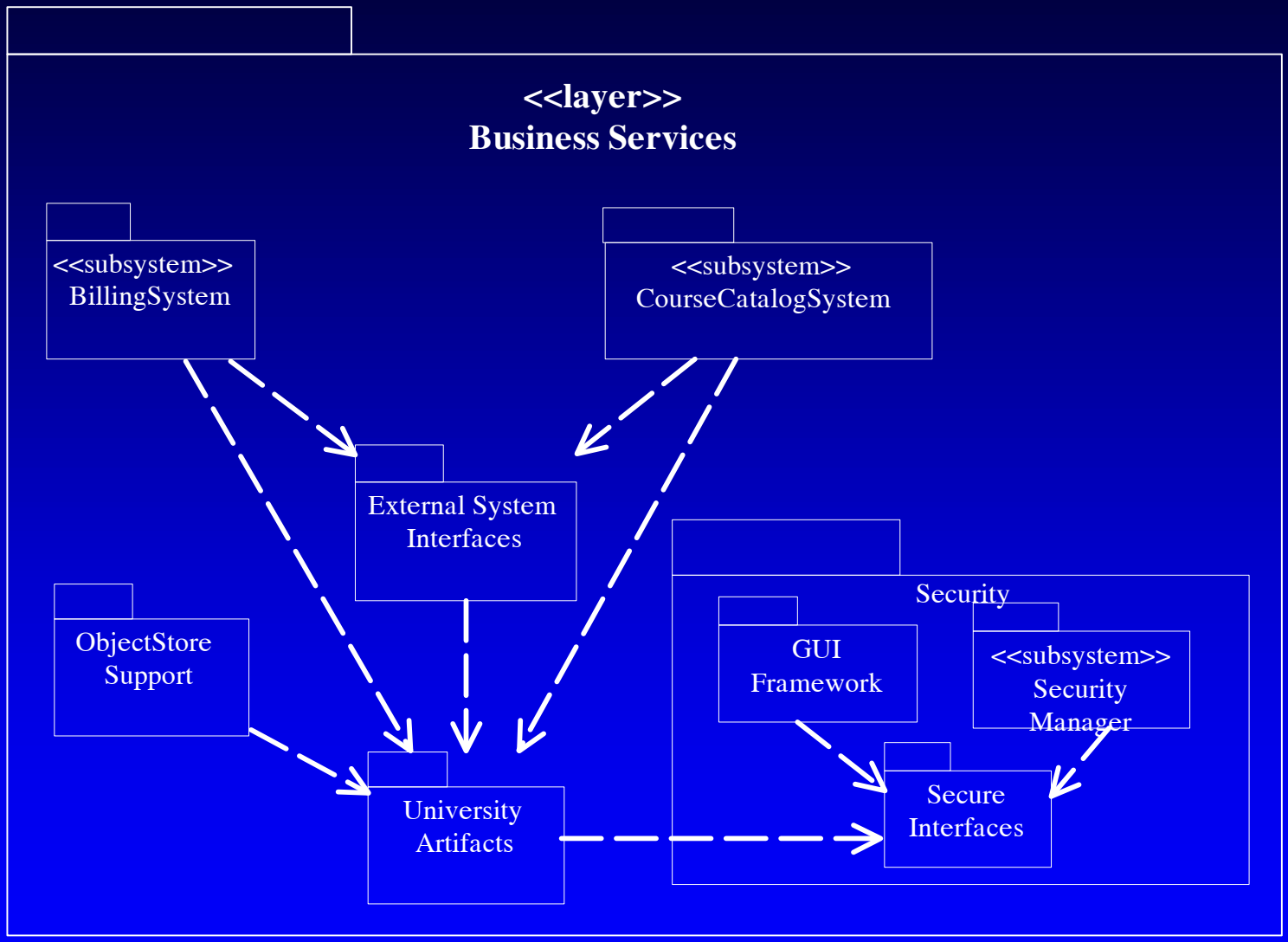
Ví dụ: Application Layer



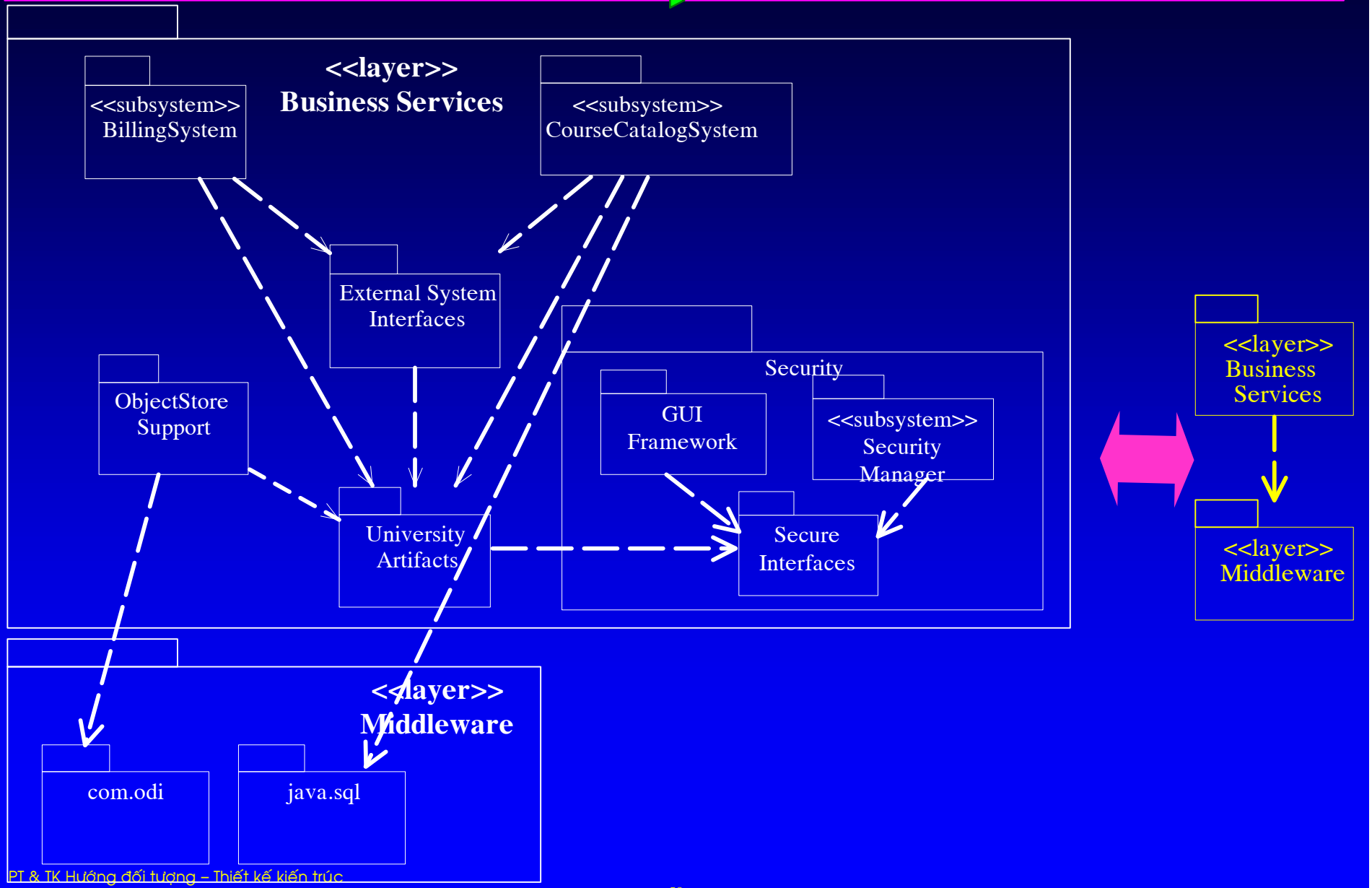
Ví dụ: Application Layer Context



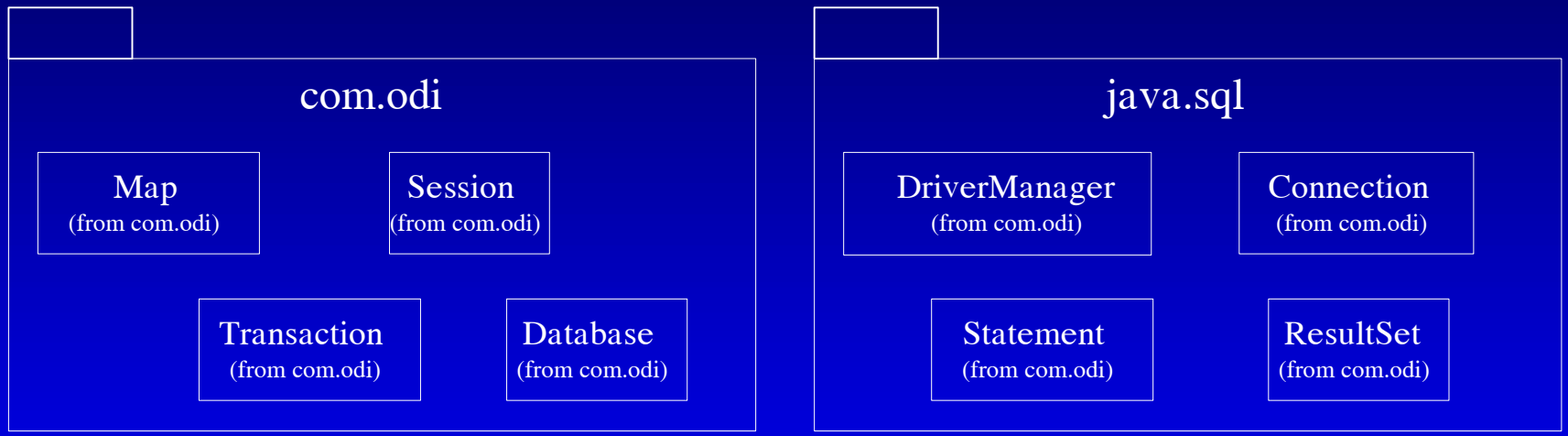
Ví dụ: Business Services Layer



Ví dụ: Business Services Layer Context



Ví dụ: Middleware Layer



Architectural Design Topics

- Các khái niệm then chốt
- Các cơ chế thiết kế và cài đặt
- Các Design Class và Subsystem
- Các khả năng tái sử dụng
- Tổ chức mô hình thiết kế
- Checkpoints

Checkpoints

- Tổng quát
  - Kiến trúc có cung cấp 1 bức tranh dề hiểu về những dịch vụ của các package khác nhau không?
  - Kiến trúc có cung cấp 1 bức tranh dề hiểu về các cơ chế không?
  - Bạn có thể tìn được lời giải có thể dùng rộng rãi hơn trong lãnh vực của ứng dụng không ?
- Layers
  - Có nhiều hơn 7 layers không?
- Subsystems
  - Việc phân chia thành các subsystem có logic và phù hợp với toàn bộ mô hình không?

Checkpoints (tt.)

- Packages
  - Tên của các package có dễ hiểu, có ý nghĩa không?
  - Mô tả về package có khớp với trách nhiệm của các class chứa bên trong không?
  - Sự phụ thuộc giữa các package có tương ứng với các quan hệ giữa các class chứa bên trong không?
  - Các class bên trong package có phù hợp với các tiêu chuẩn phân chia package không?
  - Có thể phân chia một package thành hai package?
  - Tỉ lệ giữa số package và số các classcó hợp lý?

Checkpoints (tt.)

- Các Class
  - Tên mỗi class có phản ánh đúng vai trò của nó ?
  - Liệu class có kết dính như một thể thống nhất ?
  - Toàn bộ các thành phần trong class có cần thiết cho use-case realizations?
  - Tên của role trong các aggregation và association có diễn tả chính xác mối quan hệ?
  - Các bản số trong mối quan hệ có chính xác?

Nhắc lại: Architectural Design

- Mục tiêu của Architectural Design là gì?
- Thiết kế và cài đặt là gì?
- Mechanisms? Cho ví dụ.
- Interface là gì?
- Subsystem là gì? Khác package chỗ nào?
- Subsystem dùng để làm gì và làm sao để xác định được nó?
- Phân lớp và chia nhóm như thế nào?

Bài tập: Architectural Design, Part 2

- Làm các việc sau:
  - Xây dựng layers, packages, và các phụ thuộc
  - Xây dựng các design elements (chẳng hạn, các class, subsystem, interface) và quan hệ của chúng

Bài tập: Architectural Design, Part 2 (tt.)

- Hãy xác định
  - Vị trí của các design element (như, subsystem và design class) trong kiến trúc (nghĩa là, xác định các package/layer chứa các design element)

(continued)

Bài tập: Architectural Design, Part 2 (tt.)

- Hãy xây dựng:
  - Bảng liệt kê các design element và các package "sở hữu" chúng