# Chương 7: Thiết kế kiến trúc

Nguyễn Hồng Hạnh, MSc
Bộ môn Công nghệ Phần mềm
Khoa Công nghệ thông tin – ĐH Xây Dựng
Email: hanhnh@huce.edu.vn





## Nội dung trình bày

1. Thiết kế hệ thống
2. Bài tập tổng hợp

# 1. Thiết kế hệ thống

1.1. Mục đích thiết kế hệ thống

1.2. Phân rã HT thành các HT con

1.3. Mô tả các thành phần vật lý của HT

1.4. Bố trí các thành phần khả thi vào các nút phần cứng





## 1.1 Mục đích thiết kế hệ thống

- Thiết kế hệ thống chính là thiết kế kiến trúc tổng thể của nó.
- Các thành phần tạo nên kiến trúc là gì phụ thuộc vào từng cách nhìn đối với hệ thống. Trong phần này, ta tiếp cận kiến trúc theo 3 góc nhìn (theo hệ con, theo thành phần phần mềm, theo các đơn vị phần cứng):
  - Phân rã hệ thống thành các hệ con (các gói).
  - Mô tả các thành phần vật lý của hệ thống.
  - Bố trí các thành phần khả thi vào các nút phần cứng.

# 1. Thiết kế hệ thống

1.1. Mục đích thiết kế hệ thống

1.2. Phân rã HT thành các HT con

1.3. Mô tả các thành phần vật lý của HT

1.4. Bố trí các thành phần khả thi vào các nút phần cứng



## 1.2. Phân rã HT thành các HT con

- Khái niệm về hệ con
  - Các lớp là những thực thể cấu trúc rất nhỏ so với một HT thực. Bởi vậy, khi số các lớp trong hệ thống đã lên tới hàng chục, ta nên gom các lớp liên quan với nhau thành từng nhóm gọi là các hệ con.
  - Hệ con (subsystem) là một sự gom nhóm lôgic các lớp có sự gắn kết bên trong mạnh và sự tương liên bên ngoài yếu.
  - Thuật ngữ hệ con được nhiều tác giả dùng, và G. Booch lại gọi là phạm trù (category), thực ra không phải là thuật ngữ chuẩn của UML. Trái lại, UML dùng thuật ngữ gói (package), cho nên ta cũng sẽ biểu diễn hệ con dưới dạng gói, mang theo khuôn dập <<subsystem>>.

## 1.2. Phân rã HT thành các HT con

- Nội dung của một hệ con (gồm các lớp và các mối liên quan giữa chúng) được UML 2.0 diễn tả trong một khung (frame), với một tựa đề viết trong một hình chữ nhật cắt góc theo khuôn dạng:

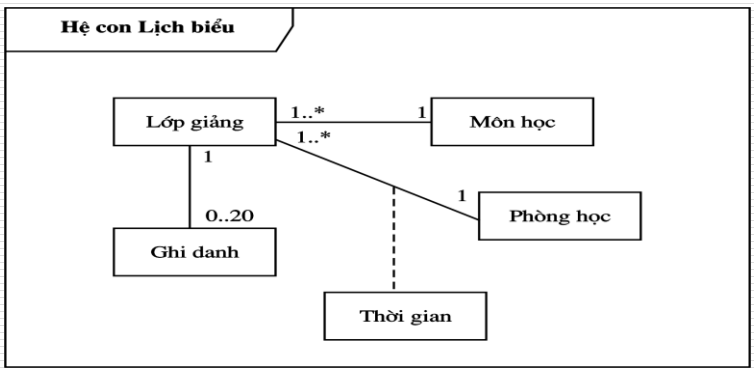
[<loại>] Tên [<tham số>]





## 1.2. Phân rã HT thành các HT con

- Số các lớp trong một hệ con không nên ít quá hay nhiều quá (thường thì có khoảng 10 lớp là vừa).




## 1.2. Phân rã HT thành các HT con

Sự gắn kết cao của các lớp trong cùng một hệ con thể hiện:

- *Về mục đích*: Chúng phải cung cấp các dịch vụ có cùng bản chất cho người dùng. Như vậy chúng phải thuộc vào cùng một lĩnh vực và đề cập một số thuật ngữ chung (chẳng hạn hệ con giao diện đề cập các thuật ngữ như: cửa sổ, thực đơn, nút nhấn,...).
- *Về xu thế phát triển*: Người ta tách các lớp bền vững với các lớp có nhiều khả năng thay đổi. Đặc biệt, thường tách các lớp nghiệp vụ với các lớp ứng dụng và xếp chúng vào các hệ con khác nhau.
- *Về ứng dụng các công nghệ*: Để tận dụng các dịch vụ công nghệ có sẵn, như các thư viện chương trình (lớp/thành phần), các GUI, các hệ quản trị cơ sở dữ liệu v.v..., ta thường tách các hệ con giao tiếp, hệ con quản trị dữ liệu ra khỏi phần lõi (ứng dụng và nghiệp vụ) của HT.





## 1.2. Phân rã HT thành các HT con

- Sự tương liên giữa các hệ con thể hiện ở mối liên quan phụ thuộc giữa chúng. Mà sự phụ thuộc giữa hai hệ con phản ánh các mối liên quan tĩnh (thừa kế, liên kết...) và các mối liên quan động (trao đổi thông điệp) giữa các lớp thuộc hai hệ con đó. Sự phụ thuộc giữa các hệ con phải càng đơn giản, lỏng lẻo thì càng tốt.
- Để đảm bảo tính tương liên yếu này, khi thành lập hệ con, áp dụng các quy tắc sau:
  - Các lớp thuộc vào cùng một phả hệ thừa kế nên được xếp vào cùng một hệ con.
  - Các lớp có mối liên quan kết nhập và hợp thành với nhau thường được xếp vào cùng một hệ con.
  - Các lớp cộng tác với nhau nhiều, trao đổi thông tin nhiều, thể hiện qua các biểu đồ tương tác, thì nên đặt chung vào một hệ con.
  - Nên tránh sự phụ thuộc vòng quanh giữa các lớp.

## 1.2. Phân rã HT thành các HT con

- Kiến trúc phân tầng
  - Một hệ con thường được định nghĩa bởi các dịch vụ mà nó cung cấp. Mối liên quan giữa một hệ con với phần còn lại của hệ thống có thể là ngang hàng hay là khách hàng/dịch vụ.
  - Trong mối liên quan ngang hàng (peer-to-peer) thì mỗi bên đều có thể truy cập các dịch vụ của bên kia. Bấy giờ sự giao tiếp không nhất thiết là ở dạng câu hỏi và trả lời liền sau đó, mà có thể là một sự giao tiếp loanh quanh, rất dễ dẫn tới những sai lỗi đáng tiếc về thiết kế.
  - Còn mối liên quan khách hàng/dịch vụ (client/server) thì đơn giản hơn: bên khách hàng gọi bên dịch vụ và bên dịch vụ thực hiện một dịch vụ theo yêu cầu và trả kết quả cho bên khách hàng. Bên khách hàng thì phải biết giao diện của bên dịch vụ, song bên dịch vụ thì không cần biết giao diện của bên khách hàng.





## 1.2. Phân rã HT thành các HT con

- Từ hai hình thức giao tiếp đó mà ta có hai cách để chia hệ thống thành các hệ con:
  - Tổ chức hệ thống thành các tầng theo chiều ngang, với mối quan hệ khách hàng/dịch vụ luôn luôn hướng từ tầng trên xuống (các) tầng dưới. Ví dụ của hệ thống phân thành tầng: hệ thống tạo cửa sổ trong giao diện người dùng của máy tính.
  - Tổ chức hệ thống thành lát theo chiều đứng, với quan hệ ngang hàng giữa các lát, tuy nhiên các lát là khá độc lập hoặc tương liên yếu với nhau. Ví dụ của hệ thống phân lát: một hệ điều hành, thường gồm các hệ con như là các hệ quản lý tệp, điều khiển thiết bị, quản lý sự kiện và ngắt...

## 1.2. Phân rã HT thành các HT con

- Rõ ràng là tổ chức phân tầng là đáng được ưu tiên hơn, vì nó mang lại nhiều ưu thế trong thiết kế, trong cài đặt cũng như trong sử dụng lại.
- Song đối với các hệ thống lớn thì ta thường phải phối hợp cả hai cách tổ chức phân tầng và phân lát, chẳng hạn phân hệ thống thành tầng, nhưng trong mỗi tầng thì lại phân thành lát.





## 1.2. Phân rã HT thành các HT con

- Khi thực hiện phân tầng, thì số tầng là tuỳ thuộc sự phức tạp của hệ thống:
  - Trong một hệ đơn giản, thì số tầng có thể chỉ là hai (2-tiers). Ví dụ: tầng khách hàng thì quản lý giao diện người dùng và các quá trình khai thác, còn tầng dịch vụ thì xử lý việc cất giữ các dữ liệu.
  - Trong một hệ phức tạp hơn, thì người ta tách tầng trên thành tầng giao diện - ứng dụng, và ở dưới nó là tầng nghiệp vụ (hay lĩnh vực), bền vững hơn và có nhiều khả năng sử dụng lại hơn. Vậy đó là một kiến trúc khách hàng/dịch vụ ba tầng (3-tiers).

## 1.2. Phân rã HT thành các HT con

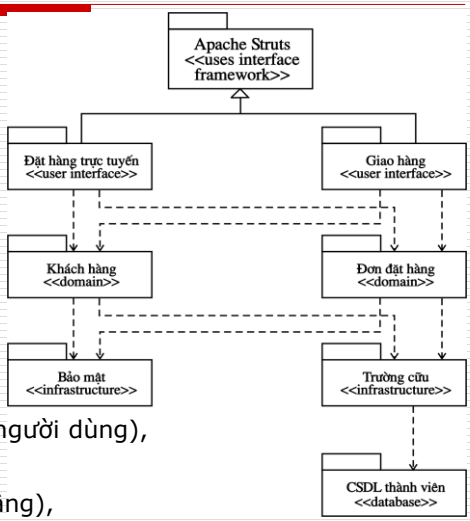
- Cuối cùng thì trong các hệ lớn, số tầng còn có thể nhiều hơn (n-tiers), mà điển hình là kiến trúc năm tầng, với các tầng kể từ trên xuống là:
  - **Tầng trình bày:** Chuyển các dữ liệu cho người dùng và biến đổi các hành động của người dùng thành các sự kiện vào của hệ thống.
  - **Tầng ứng dụng:** bao gồm các đối tượng điều khiển và dẫn dắt các quy luật của ứng dụng.
  - **Tầng nghiệp vụ:** bao gồm các đối tượng nghiệp vụ (hay lĩnh vực), cùng sự cài đặt các quy tắc quản lý chúng.
  - **Tầng truy cập dữ liệu:** phục hồi các đối tượng nghiệp vụ từ các phương tiện lưu trữ.
  - **Tầng lưu trữ dữ liệu:** bảo đảm sự lưu giữ lâu dài các dữ liệu.





## 1.2. Phân rã HT thành các HT con

- Một thí dụ về kiến trúc khách hàng/dịch vụ năm tầng cho như bên, trong đó mỗi gói (hệ con) đều có mang khuôn dập thích hợp, như là:
- <<user interface framework>> (khuôn khổ giao diện người dùng),




- <<user interface>> (giao diện người dùng),
- <<domain>> (lĩnh vực),
- <<infrastructure>> (cơ sở hạ tầng),
- <<database>> (cơ sở dữ liệu).

# 1. Thiết kế hệ thống

1.1. Mục đích

1.2. Phân rã HT thành các HT con

1.3. Mô tả các thành phần vật lý của HT

1.4. Bố trí các thành phần khả thi vào các nút phần cứng





## 1.3. Mô tả các thành phần vật lý của HT

- Thành phần và biểu đồ thành phần
  - Nếu như biểu đồ gói (hệ con) mà ta nói ở phần trên phản ánh cho góc nhìn về cấu trúc lôgic của hệ thống (ở mức cao so với biểu đồ lớp), thì biểu đồ thành phần, với các đơn nguyên trong đó là các thành phần, lại cho ta một cách nhìn về cấu trúc vật lý của hệ thống.
  - Chữ "vật lý" ở đây được hiểu theo nghĩa là sự mô tả hướng tới các sản phẩm phần mềm, là kết quả của sự cài đặt và thực sự tồn tại, chứ không phải là các sản phẩm lôgic, kết quả của quá trình phân tích. Tuy nhiên ở đây ta cũng chưa đề cập tới phần cứng, mặc dù tính vật lý của nó cũng là đương nhiên.

## 1.3. Mô tả các thành phần vật lý của HT

- UML định nghĩa thành phần (component) là một bộ phận vật lý và thay thế được của hệ thống, thích ứng và cung cấp sự thực hiện cho một tập các giao diện.
- Nói đơn giản hơn, thì thành phần là một cài đặt của một tập hợp các phần tử lôgic, như các lớp hay các hợp tác.





## 1.3. Mô tả các thành phần vật lý của HT

- Có ba loại thành phần:
  - Các thành phần triển khai (deployment components): Đó là các thành phần cần và đủ để tạo nên một hệ thống khả thi, như là các thư viện động (DLL) và các mã khả thi (executable). Định nghĩa thành phần của UML là đủ rộng để bao hàm các mô hình đối tượng kinh điển, như là COM+, CORBA, và Enterprise Java Beans, cũng như các mô hình đối tượng khác biệt như là các trang Web động, các bảng cơ sở dữ liệu, và các mã khả thi sử dụng những cơ chế truyền thông riêng.

## 1.3. Mô tả các thành phần vật lý của HT

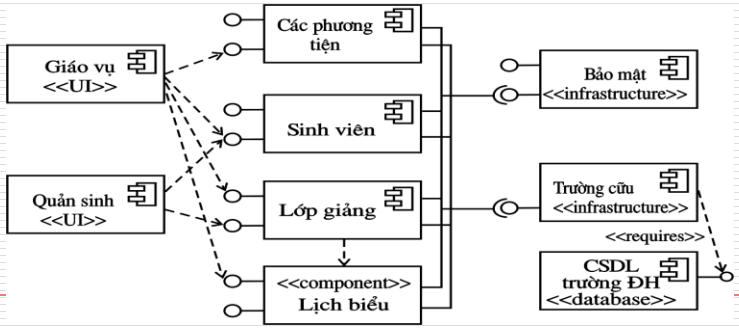
- Các thành phần sản phẩm làm việc (work product components): Đó là các thành phần có từ quá trình phát triển hệ thống, bao gồm các tệp mã nguồn, các tệp dữ liệu, từ đó mà ta đã tạo lập ra các thành phần triển khai. Các thành phần này không trực tiếp tham gia vào hệ thống thực thi, nhưng không có chúng thì không tạo được hệ thống thực thi.
- Các thành phần thực hiện (execution components): Đó là các thành phần được tạo nên như là một kết quả của một hệ thực hiện, chẳng hạn một đối tượng COM+, được cá thể hoá từ một DLL.





## 1.3. Mô tả các thành phần vật lý của HT

- Để tổ chức các thành phần lại với nhau, ta có hai cách:
  - Gom các thành phần vào các gói, nghĩa là đưa chúng vào các hệ con;
  - Thiết lập các mối liên quan phụ thuộc giữa chúng, và như thế ta có một biểu đồ thành phần.




## 1.3. Mô tả các thành phần vật lý của HT

- Mục đích mô hình hoá của biểu đồ thành phần
  - Như đã trình bày, có nhiều loại thành phần (ít nhất có ba loại chính là thành phần triển khai, thành phần sản phẩm làm việc và thành phần thực hiện),
  - Vậy biểu đồ thành phần lập ra phải có mục đích mô tả loại thành phần nào.



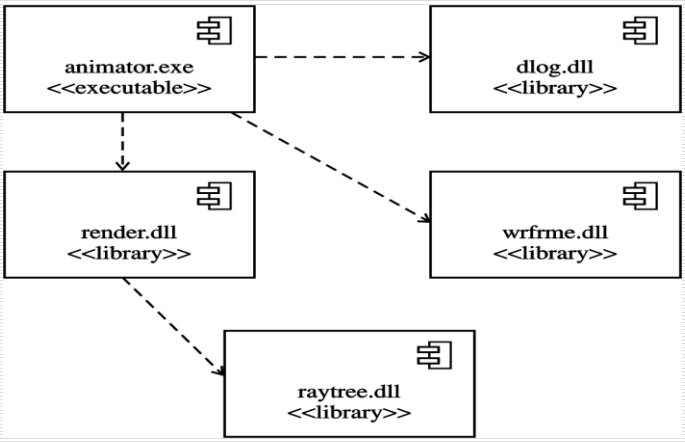


## 1.3. Mô tả các thành phần vật lý của HT

- Mô hình hoá các thành phần khả thi và thư viện: Có thể nói mục đích chính của việc sử dụng biểu đồ thành phần là để mô hình hoá các thành phần triển khai, tạo nên cái cài đặt của hệ thống.
  - Nếu ta muốn cài đặt một hệ thống chỉ gồm đúng một tệp khả thi (.EXE), thì ta chẳng cần dùng tới thành phần.
  - Ngược lại nếu hệ thống gồm nhiều tệp khả thi và liên kết với các thư viện đối tượng thì ta cần dùng biểu đồ thành phần để giúp ta hiển thị, đặc tả, thành lập và tư liệu hoá các quyết định của chúng ta đối với hệ thống vật lý.

## 1.3. Mô tả các thành phần vật lý của HT

- *Thí dụ*: Một biểu đồ các thành phần triển khai





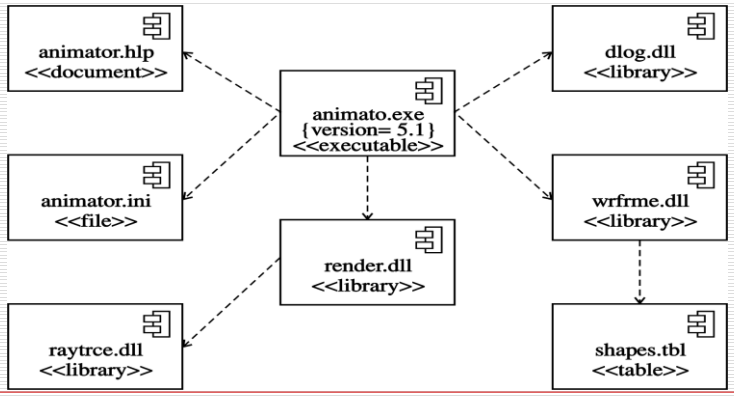



## 1.3. Mô tả các thành phần vật lý của HT

- Mô hình hoá các bảng, các tệp và các tư liệu: Bên cạnh các tệp khả thi và thư viện tạo nên phần chạy được của hệ thống, thì còn nhiều thành phần bố trí khác, gọi là các thành phần phụ trợ, cần cho việc cài đặt hệ thống.
  - Chẳng hạn để cài đặt, ta vẫn cần các tệp dữ liệu, các tư liệu trợ giúp, các scripts, các tệp log, các tệp khởi tạo, các tệp xếp chỗ hay gỡ bỏ. Mô hình hoá các thành phần này cũng là một phần quan trọng để diễn tả hình trạng của hệ thống.

## 1.3. Mô tả các thành phần vật lý của HT

- *Thí dụ*: Từ biểu đồ trên, thêm vào các bảng, các tệp và các tư liệu




27

27

## 1.3. Mô tả các thành phần vật lý của HT

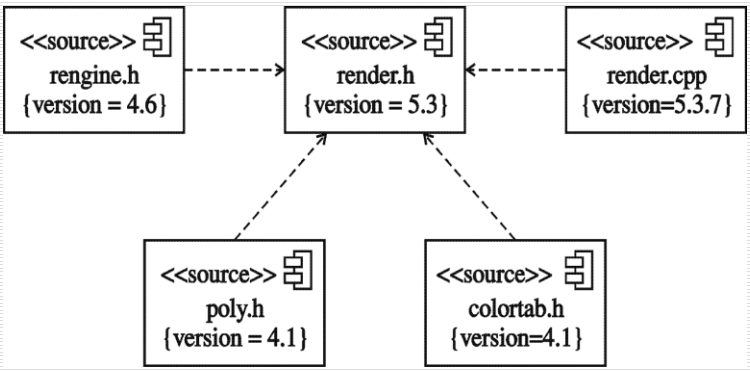
- Mô hình hoá mã nguồn: Mô hình hoá mã nguồn cũng là một mục đích của việc tạo lập biểu đồ thành phần.
- Các tệp nguồn dùng để chứa các chi tiết về các lớp, các giao diện, các hợp tác và các phần tử lôgic khác; chúng tạo nên một bước trung gian để tạo lập các thành phần vật lý, nhị phân (nhờ một công cụ nào đó).
- Các công cụ (chẳng hạn chương trình biên dịch) thường đòi hỏi các tệp nguồn phải được tổ chức theo một quy cách nhất định nào đó (thông thường là một hay hai tệp cho một lớp). Các mối liên quan phụ thuộc giữa các tệp này phản ánh một trật tự biên dịch.

28

28

## 1.3. Mô tả các thành phần vật lý của HT

- *Thí dụ*: Mô hình hoá các tệp nguồn dùng để xây dựng thư viện render.dll từ các thí dụ trước








## 1. Thiết kế hệ thống



## 1.4. Bố trí các thành phần khả thi vào các nút phần cứng

- Để bố trí các thành phần phần mềm lên các phần cứng, ta dùng các biểu đồ triển khai.
- Biểu đồ triển khai (deployment diagram) là một biểu đồ diễn tả sự bố trí các executable artifacts trên underlying platform. Nó gồm các nút và các kết nối giữa các nút đó.
- Một nút (node) là một phần tử vật lý tồn tại vào lúc chạy và biểu diễn cho một tài nguyên tính toán (computational resource), nói chung thì phải có ít nhất một chỗ nhớ, và thông thường thì có thêm khả năng xử lý. Một nút được biểu diễn bởi một hình hộp, có mang tên.
  - Nếu tên này không gạch dưới, thì nút thể hiện một lớp các tài nguyên,
  - Nếu tên được gạch dưới, thì nút thể hiện một cá thể tài nguyên.



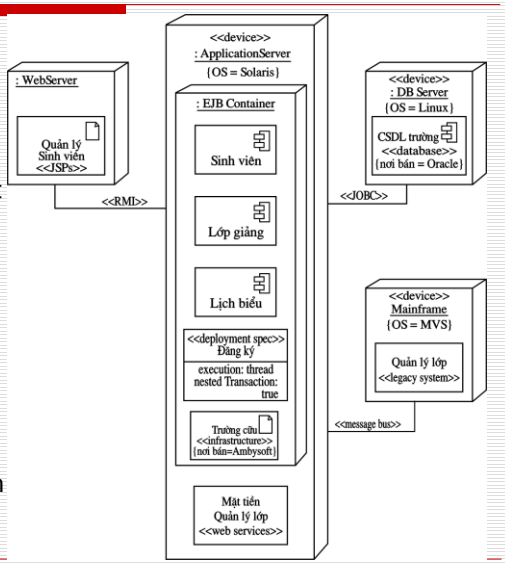


## 1.4. Bố trí các thành phần khả thi vào các nút phần cứng

- Như vậy có hai mức diễn tả của biểu đồ triển khai: mức lớp (tương tự một biểu đồ lớp) và mức cá thể (tương tự một biểu đồ đối tượng).
- Theo định nghĩa như trên, nút có thể là:
  - Một thiết bị (nút cứng), thường mang khuôn dập <<device>> (hoặc cụ thể hơn là processor, console, kiosk, printer,...);
  - Một môi trường thực hiện (nút mềm), thường mang khuôn dập <<execution env>>, như là EJB Container hay là J2EEServer.
- Hình sau đây cho một biểu đồ triển khai (ở mức cá thể) biểu diễn cho hình trạng vật lý của hệ thống thông tin về trường đại học.

## 1.4. Bố trí các thành phần khả thi vào các nút phần cứng

- Nút WebServer chưa có khuôn dập, đó là vì người phát triển hệ thống chưa có quyết định: nút đó có thể đơn giản là một loại phần mềm nào đó (chẳng hạn một Browser), hoặc là một thiết bị vật lý.
- Các nút có thể chứa các nút khác hoặc các sản vật phần mềm. Chẳng hạn nút ApplicationServer chứa nút EJBContainer (một nút mềm), và nút này lại chứa ba thành phần phần mềm, một đặc tả bố trí, và một sản vật phần mềm.








## 1.4. Bố trí các thành phần khả thi vào các nút phần cứng

- Các kết nối (connections) là các mối liên quan giao tiếp giữa các cặp nút, thể hiện về mặt vật lý bằng một đường truyền (như là một kết nối Ethernet, một đường truyền tuần tự hay một bus dùng chung).
- Mỗi kết nối hỗ trợ cho một hay nhiều giao thức truyền thông, mà ta cần chỉ rõ bằng các khuôn dập thích hợp. Bảng sau cho một số khuôn dập thường dùng cho các kết nối, cùng với ý nghĩa của chúng.

## 1.4. Bố trí các thành phần khả thi vào các nút phần cứng

| Khuôn dập | Ý nghĩa |
|---|---|
| Asynchronous | Một kết nối không đồng bộ, |
| HTTP | HyperText Transport Protocol, một giao thức Internet |
| JDBC | Java Database Connectivity, một Java API để truy cập CSDL |
| ODBC | Open Database Connectivity, một MicroSoft API để truy cập CSDL |
| RMI | Remote Method Invocation, một giao thức liên lạc của Java |
| RPC | Remote Procedure Call |
| Synchronous | Một kết nối đồng bộ, trong đó bên gửi chờ trả lời từ bên nhận |
| Web services | Liên lạc qua các giao thức web services như là SOAP và UDDI |



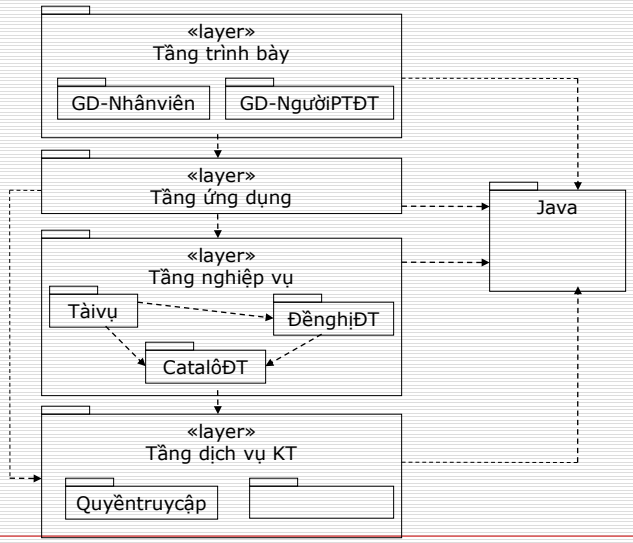


# Nội dung trình bày

1. Thiết kế hệ thống
2. Bài tập tổng hợp

## 2. Bài tập tổng hợp

*Đề xuất một kiến trúc phân tầng cho HT YCĐT*

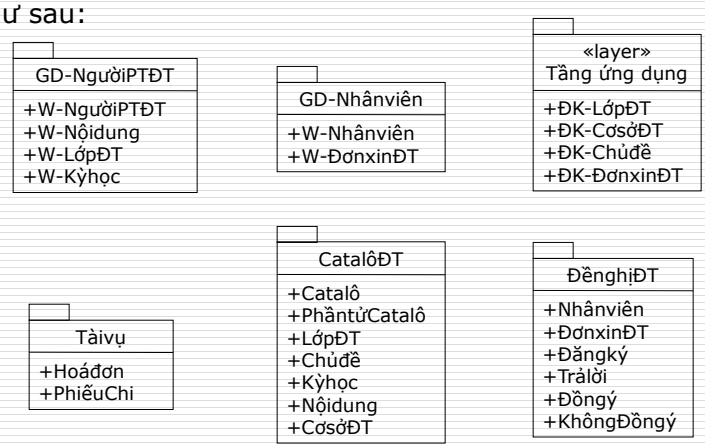



37

37

## 2. Bài tập tổng hợp

Trong kiến trúc phân tầng trên, thì gói java chứa các lớp cơ sở dùng chung cho mọi tầng, còn các gói con thì chứa các lớp như sau:

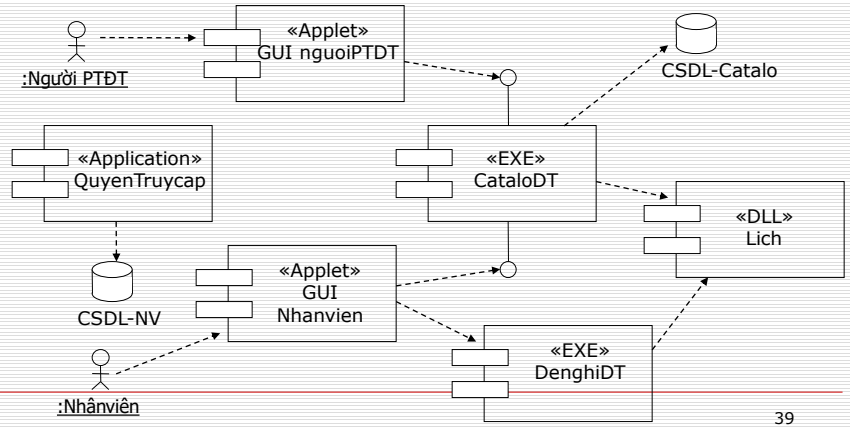



38

38

## 2. Bài tập tổng hợp

**Câu hỏi 27:** *Hãy đề xuất một BĐ thành phần cho hai vòng lặp đầu của HT. Chỉ cần đưa ra các thành phần không lớn lắm, đủ để biểu diễn cho một hợp tác giữa nhiều lớp. Đừng quên ngôn ngữ đích là Java.*

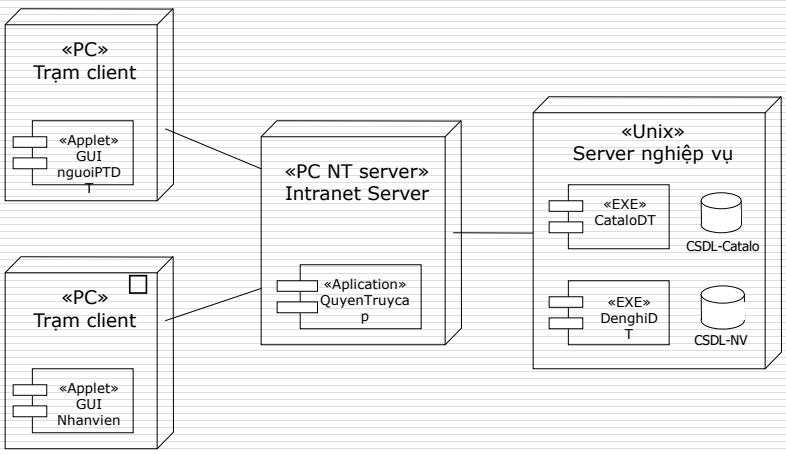



39

39

## 2. Bài tập tổng hợp

**Câu hỏi 28:** *Hãy đề xuất một BĐ bố trí cho hai vòng lặp đầu của HT.*




40

40

## Hỏi - đáp





## Lời hay ý đẹp

**"Tỏ ra hơn bạn, bạn sẽ thành thù của ta;**
**chịu nhường bạn, bạn sẽ liên kết với ta"**

**La Rochefoucauld**