Bài tập tuần 06

Thiết kế lớp (class design)

Mục tiêu

- Trong Thiết kế lớp, đối với mỗi usecase, chúng ta xây dựng sơ đồ lớp và thiết kế chi tiết cho mỗi lớp.
    - Mô tả các thuộc tính và hoạt động của lớp
    - Xây dựng biểu đồ lớp
    - Tìm hiểu thông tin chi tiết của một lớp cụ thể (ví dụ, những thao tác và lớp nào cần được thêm vào để hỗ trợ, và cách chúng cộng tác với nhau, trách nhiệm được phân bổ cho từng lớp).

Đánh giá

- Hoàn thành xác định các lớp thiết kế và xây dựng sơ đồ lớp

Phần I: Mô tả các thuộc tính và trách nhiệm của lớp

- Tinh chỉnh các lớp phân tích và biến chúng trở thành các lớp thiết kế bằng cách thêm các đặc điểm và thuộc tính đáp ứng cho chúng.

1. Mô tả các thuộc tính của lớp

Các thuộc tính được sử dụng để lưu trữ thông tin. Tên thuộc tính phải là một danh từ chỉ rõ thông tin mà thuộc tính đó nắm giữ. Mô tả của thuộc tính phải chỉ rõ thông tin nào sẽ được lưu trữ trong thuộc tính; điều này có thể là tùy chọn khi thông tin được lưu trữ có thể nhận biết rõ ràng qua tên thuộc tính.

Một số hướng dẫn để tìm thuộc tính (xuất phát yêu cầu, đặc tả ca sử dụng, bảng thuật ngữ)

- Thuộc tính / đặc điểm của các lớp đã xác định
- Thông tin được lưu giữ bởi các lớp xác định
- “Danh từ” không trở thành lớp
    - Thông tin chứa giá trị quan trọng
    - Thông tin duy nhất được "sở hữu" bởi một đối tượng
    - Thông tin không có hành vi

Mỗi thuộc tính được định nghĩa theo cú pháp như sau: **attributeName : Type = Default**

• Tên của thuộc tính, phải tuân theo các quy ước đặt tên của cả ngôn ngữ thực hiện và dự án.

• Kiểu của thuộc tính, sẽ là kiểu dữ liệu cơ bản được hỗ trợ bởi ngôn ngữ triển khai.

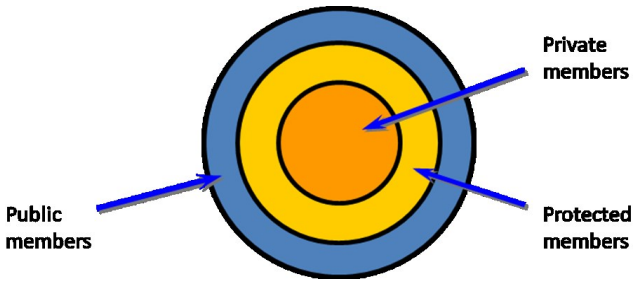
o Tuân theo các quy ước đặt tên của ngôn ngữ thực hiện và dự án

o Kiểu phải là kiểu dữ liệu cơ bản trong ngôn ngữ triển khai: Kiểu dữ liệu dựng sẵn, kiểu dữ liệu do người dùng xác định hoặc lớp do người dùng xác định

• Giá trị mặc định hoặc ban đầu của thuộc tính, được khởi tạo khi các thể hiện mới của lớp được tạo.

• Giới hạn truy cập của thuộc tính, sẽ nhận một trong các giá trị sau:

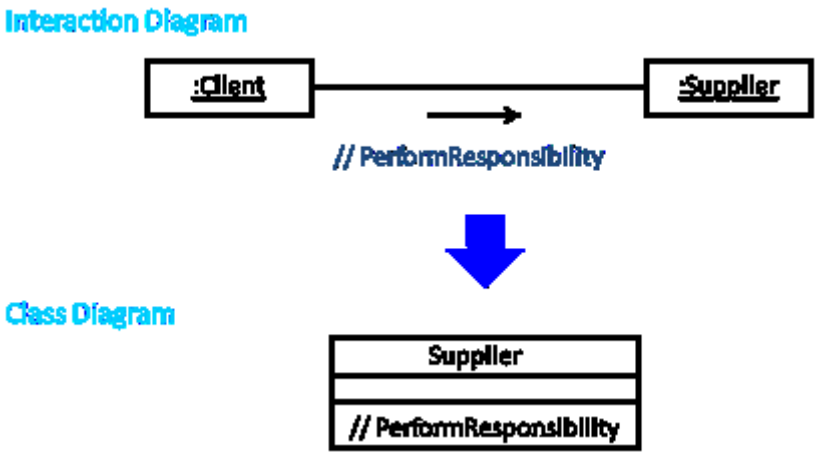
- Public (+): Thuộc tính có thể nhìn thấy cả bên trong và bên ngoài gói chứa lớp.
- Protected (#): Thuộc tính chỉ hiển thị với chính lớp đó, với các lớp con của nó hoặc với lớp bạn bè (phụ thuộc vào ngôn ngữ)
- Private (-): Thuộc tính chỉ hiển thị trong lớp hiện tại và bạn bè trong lớp.



Đối với các lớp thực thể, cần xác định thuộc tính là lưu giữ thông tin lâu dài (mặc định) hay tạm thời. Mặc dù bản thân lớp có thể là lớp lưu giữ thông tin, nhưng không phải tất cả các thuộc tính của lớp đều cần phải ổn định.

2. Mô tả các trách nhiệm của lớp

Các trách nhiệm bắt nguồn từ các thông điệp trên sơ đồ Tương tác. Đối với mỗi thông điệp, hãy kiểm tra lớp của đối tượng mà thông điệp được gửi đến. Nếu trách nhiệm chưa tồn tại, hãy tạo một trách nhiệm mới cung cấp hành vi được yêu cầu.

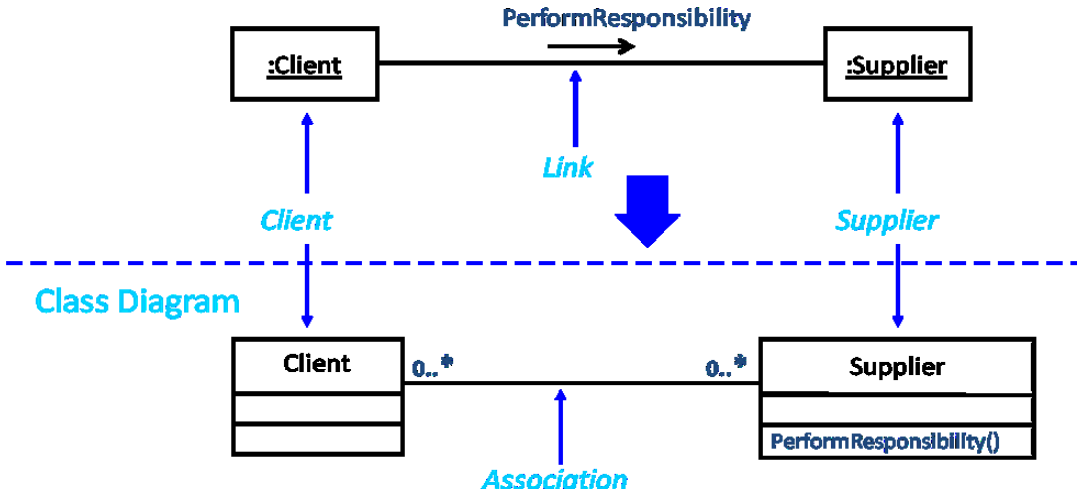


Phần II: Xây dựng biểu đồ lớp (class diagram)

Để tạo sơ đồ lớp, bạn phải tìm các mối quan hệ phụ thuộc, liên kết giữa các lớp thiết kế.

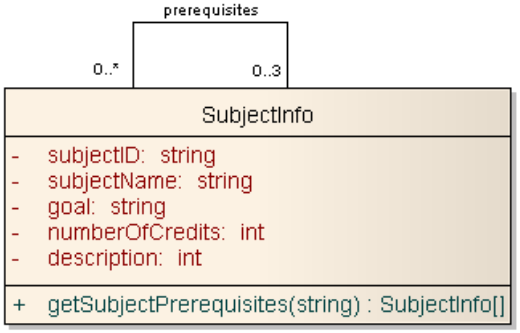
1. Xác định các quan hệ Associations

Associations thể hiện mối quan hệ cấu trúc giữa các đối tượng thuộc các lớp khác nhau; chúng kết nối các thể hiện của hai hoặc nhiều lớp với nhau trong một thời gian nào đó. Bạn có thể sử dụng **association** để cho thấy rằng các đối tượng biết về các đối tượng khác. Đôi khi, các đối tượng phải giữ các tham chiếu đến nhau để có thể tương tác; ví dụ, để gửi thông điệp cho nhau. Do đó, trong một số trường hợp, **associations** có thể xác định từ các mẫu tương tác trong Sơ đồ trình tự hoặc Sơ đồ giao tiếp.



Hầu hết các **association** đều đơn giản (tồn tại giữa hai lớp) và được vẽ dưới dạng đường thẳng nét liền nối các cặp ký hiệu lớp. Mối quan hệ bậc ba cũng có thể gặp phải. Đôi khi một lớp có một liên kết với chính nó. Điều này không nhất thiết có nghĩa là một cá thể của lớp đó có liên kết với chính nó; thông thường điều đó có nghĩa là một thể hiện của lớp có liên kết với các thể hiện khác của cùng một lớp.

Ví dụ, một đến ba môn học có thể là điều kiện tiên quyết của một hoặc nhiều môn học khác.



Bội số (Multiplicity) của quan hệ

Multiplicity trong các quan hệ **associations** giúp trả lời cho hai câu hỏi:

- Liên kết association là bắt buộc hay tuỳ chọn?
- Số lượng thực thể tối thiểu và tối đa có thể được liên kết với một thực thể khác?

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

Multiplicity cho phép bạn biết số lượng quan hệ giới hạn dưới và giới hạn trên mà một đối tượng nhất định có thể có với một đối tượng khác. Nhiều khi bạn không biết số lượng trường hợp tối đa có thể là bao nhiêu, và bạn sẽ sử dụng **"*"** để chỉ định rằng con số này là không xác định.

Câu hỏi quan trọng nhất mà multiplicity trả lời: Sự liên kết có bắt buộc không? Một số giới hạn dưới lớn hơn 0 cho biết rằng mối quan hệ là bắt buộc.

Vai trò (Role) của quan hệ

Mỗi đầu của một liên kết có một vai trò trong mối quan hệ với lớp ở đầu kia của liên kết. Vai trò chỉ định mặt mà một lớp thể hiện trên mỗi khía cạnh của liên kết. Vai trò phải có tên và tên vai trò ở các phía đối diện của liên kết phải là duy nhất.

- Tên vai trò phải là một danh từ chỉ vai trò của đối tượng liên kết trong mối quan hệ với đối tượng liên kết.
- Việc sử dụng tên liên kết và tên vai trò là loại trừ lẫn nhau: người ta sẽ không sử dụng cả tên liên kết và tên vai trò. Đối với mỗi liên kết, hãy quyết định cái nào truyền đạt nhiều thông tin hơn.
- Tên vai trò được đặt bên cạnh cuối dòng liên kết của lớp mà nó mô tả.

Trong trường hợp tự liên kết, tên vai trò là điều cần thiết để phân biệt mục đích cho association.

Aggregations và Compositions

Associations có hai loại đặc biệt là Aggregations và Compositions; Tuy nhiên, không khuyến khích sử dụng nhiều hai loại này.

Aggregations là một dạng liên kết mạnh hơn được sử dụng để mô hình hóa mối quan hệ toàn bộ giữa các phần tử của mô hình. Toàn bộ / tổng thể có một liên kết tổng hợp với các bộ phận cấu thành của nó.

- Một hình thoi rỗng được gắn vào phần cuối của một đường liên kết ở phía bên của tập hợp (toàn bộ) để biểu thị sự kết hợp. Vì tập hợp là một dạng liên kết đặc biệt, nên việc sử dụng multiplicity, vai trò, điều hướng, v.v. cũng giống như đối với **associations**.

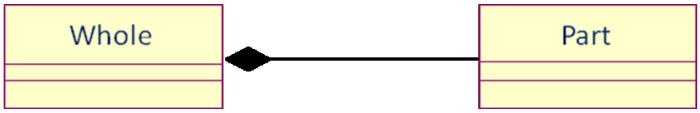
Những tình huống thích hợp sử dụng **aggregations**:

- Một đối tượng được cấu tạo về mặt vật lý từ các đối tượng khác (ví dụ, ô tô được cấu tạo về mặt vật lý bởi một động cơ và bốn bánh xe).
- Một đối tượng là một tập hợp logic của các đối tượng khác (ví dụ, một gia đình là một tập hợp của cha mẹ và con cái).
- Một vật thể có chứa các vật thể khác (ví dụ, một máy bay có chứa phi công).

Compositions là một dạng của **aggregation** với quyền sở hữu mạnh và thời gian tồn tại trùng hợp của lớp thành phần với lớp tổng thể. Lớp tổng thể "sở hữu" lớp thành phần và chịu trách nhiệm về việc tạo ra và phá hủy bộ phận đó. Lớp thành phần bị loại bỏ khi lớp toàn thể được loại bỏ.

Một hình thoi đặc được gắn vào cuối của một đường liên kết (ở "phía toàn thể") để chỉ quan hệ **Compositions**.

Trong một số trường hợp, **compositions** có thể được xác định sớm ngay trong giai đoạn Phân tích, nhưng thường thì phải đến Thiết kế thì các quyết định đó mới có thể được đưa ra một cách tự tin. Đó là lý do tại sao **compositions** được giới thiệu ở đây thay vì trong Phân tích ca sử dụng.

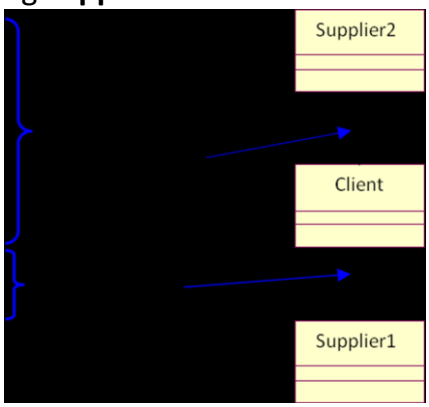


Compositions nên được sử dụng thay vì **Aggregations** "đơn giản" khi có sự phụ thuộc lẫn nhau mạnh mẽ giữa tổng thể và các bộ phận, trong đó định nghĩa về tổng thể không đầy đủ nếu không có các bộ phận. Ví dụ, một Đơn đặt hàng không có nghĩa nếu không có gì được đặt hàng.

2. Xác định các quan hệ Dependencies

Mối quan hệ phụ thuộc (**Dependencies**) biểu thị mối quan hệ ngữ nghĩa giữa các phần tử mô hình, trong đó sự thay đổi của **supplier** có thể gây ra sự thay đổi **client**.

Có bốn tùy chọn để tạo một lộ trình giao tiếp đến đối tượng **supplier**:



- Global: The supplier object is a global object.
- Parameter: The supplier object is a parameter to, or the return class of, an operation in the client object.
- Local: The supplier object is declared locally (that is, created temporarily during execution of an operation).
- Field: The supplier object is a data member in the client object.

Sự phụ thuộc là một loại lộ trình giao tiếp là một loại quan hệ nhất thời. Những điều này xảy ra khi khả năng hiển thị là toàn cục, tham số hoặc cục bộ.

Xem xét từng mối quan hệ kết hợp và xác định xem nó nên vẫn là một liên kết hay trở thành một mối quan hệ phụ thuộc. Liên kết và tổng hợp là mối quan hệ cấu trúc (khả năng hiển thị trường). Mối quan hệ kết hợp được thực hiện bởi các biến tồn tại trong phần thành viên dữ liệu của định nghĩa lớp. Bất kỳ mối quan hệ nào khác (toàn cầu, cục bộ và khả năng hiển thị tham số) đều là mối quan hệ phụ thuộc.

3. Xác định các quan hệ Generalization

Tổng quát hóa là mối quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và / hoặc hành vi của một hoặc nhiều lớp. Tổng quát hóa tinh chỉnh một hệ thống phân cấp của các trừu tượng trong đó một lớp con kế thừa từ một hoặc nhiều lớp cha. Khái quát hóa là một mối quan hệ "là một loại". Bạn luôn có thể nói rằng lớp tổng quát của bạn “là một loại” lớp cha.

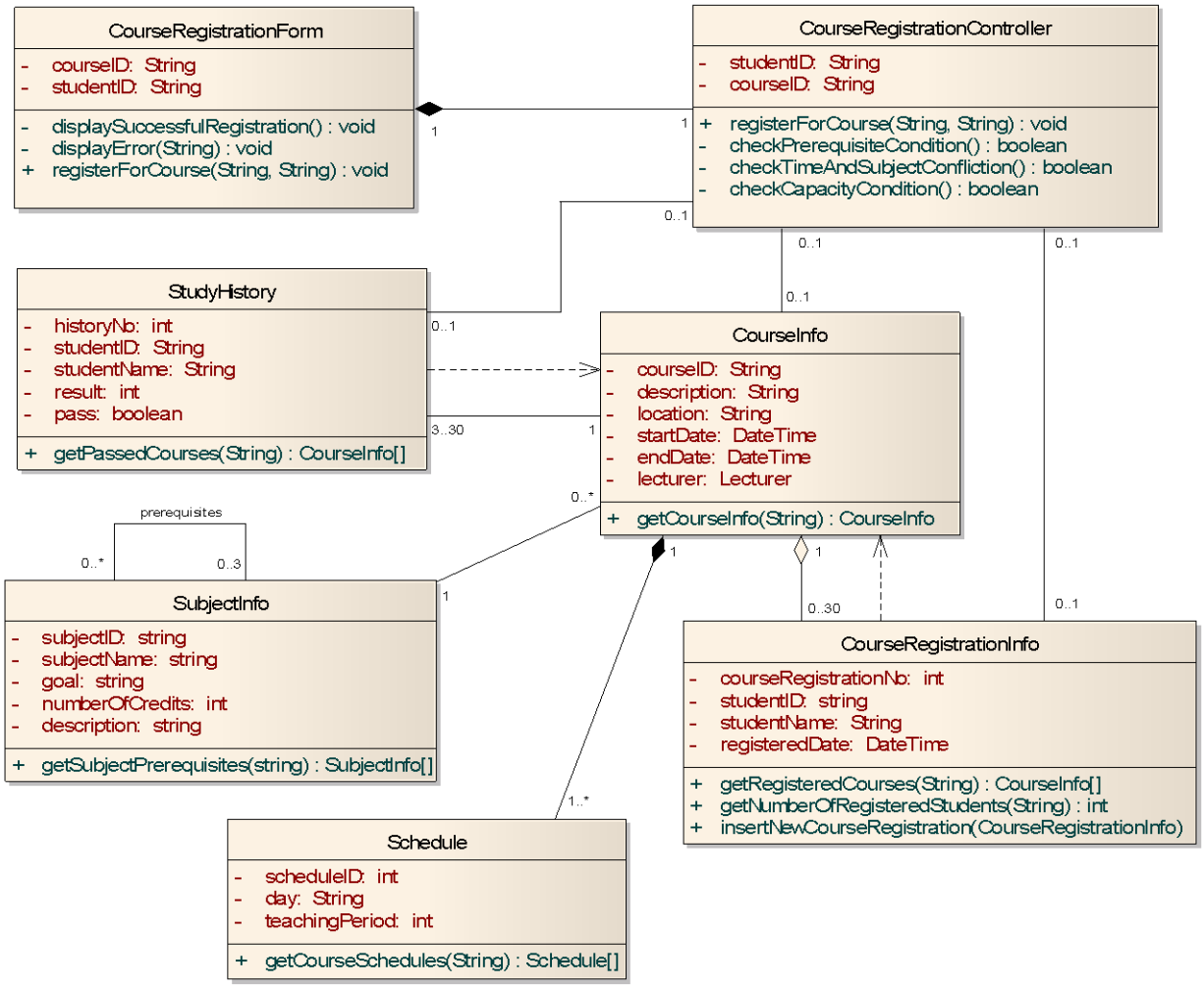
Các thuật ngữ “tổ tiên” và “hậu duệ” có thể được sử dụng thay cho “lớp cha” và “lớp con”.

Khi tổng quát được tìm thấy, một lớp cha chung được tạo ra để chứa các thuộc tính, liên kết, tập hợp và phép toán chung. Hành vi chung bị xóa khỏi các lớp để trở thành lớp con của lớp cha chung.

4. Xây dựng biểu đồ lớp (class diagram)

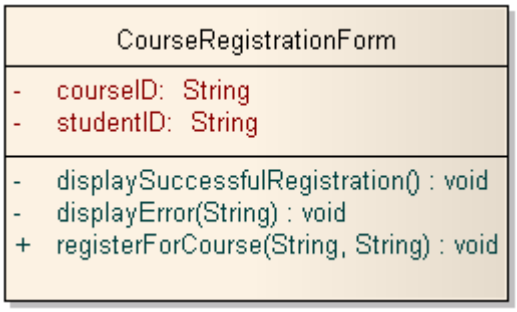
Đặt các lớp thiết kế và các mối quan hệ lớp với nhau để xây dựng một biểu đồ lớp cho từng usecase.

Sơ đồ lớp sau đây là kết quả của bước này cho usecase “Đăng ký khóa học”.



Phần III: Thiết kế chi tiết các lớp

CourseRegistrationForm



***** Attributes:**

studentID and courseID are passed when student choose a course to register. These attributes are used as parameters for registerForCourse method.

***** Methods:**

+ registerForCourse(studentID: String, courseID: String)

* Purpose: perform course registration for student studentID with course courseID

* Action:

New an instance of CourseRegistrationController

Call registerForCourse method of CourseRegistrationController object with studentID and courseID as parameter.

Catch error if any

Display successful registration

+ displayErrorMessage(message: char[])

Display error message according to message parameter

+ displaySuccessfulRegistration()

Display register successfully

HẾT