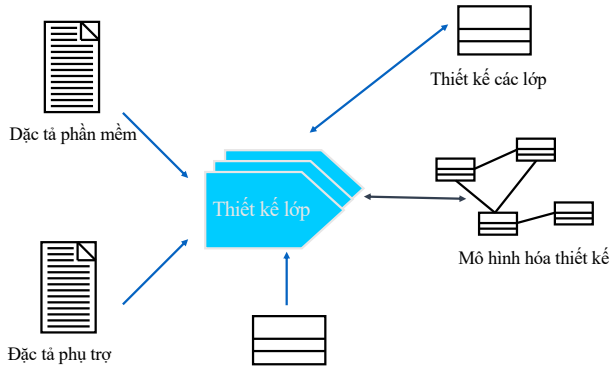
ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

THIẾT KẾ XÂY DỰNG PHẦN MỀM

Bài 6. Thiết kế lớp

1

Tổng quan về thiết kế lớp



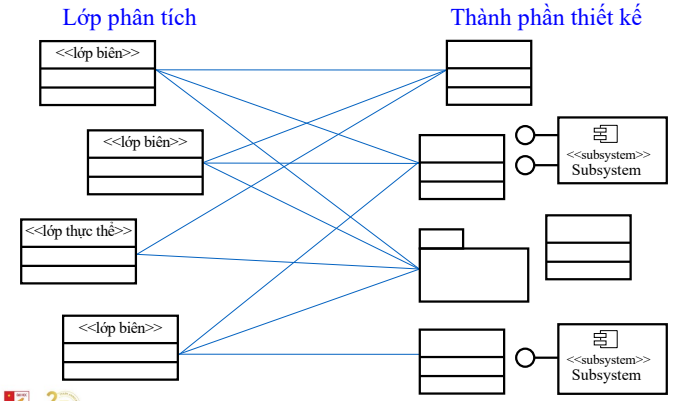
2

Nội dung

1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

3

Từ các lớp phân tích tới các thành phần thiết kế



Quan hệ ánh xạ nhiều-nhiều (n-n)

4

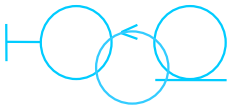
Nhận diện các lớp thiết kế

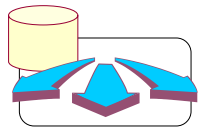
- Một lớp phân tích ánh xạ trực tiếp tới một lớp thiết kế nếu:
  - Đó là một lớp có cấu trúc đơn giản
  - Chỉ biểu diễn một sự logic trừu tượng
- Những lớp phân tích phức tạp hơn có thể
  - Chia nhỏ thành nhiều class
  - Chuyển thành package
  - Chuyển thành subsystem (thảo luận ở sau)
  - ...

5

Class Design Considerations

- Ký pháp
  - Boundary (lớp biên)
  - Entity (lớp thực thể)
  - Control (lớp điều khiển)
- Áp dụng các mẫu thiết kế (design pattern)





6

Bao nhiêu class là đủ?

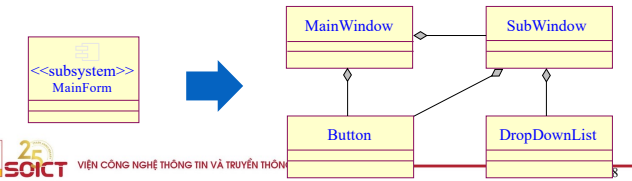
- Nhiều class đơn giản thì điều đó có nghĩa là
  - Sự bao quát hệ thống tổng thể bị hạn chế
  - Có khả năng tái sử dụng cao hơn
  - Dễ cài đặt
- Có ít class và các class đều phức tạp
  - Sự bao quát trong hệ thống tổng thể nhiều hơn
  - Khó tái sử dụng
  - Khó cài đặt

Một class nên có một mục đích nhất định
Một class chỉ nên làm một việc và làm tốt việc đó

7

Những chiến lược cho thiết kế các lớp biên

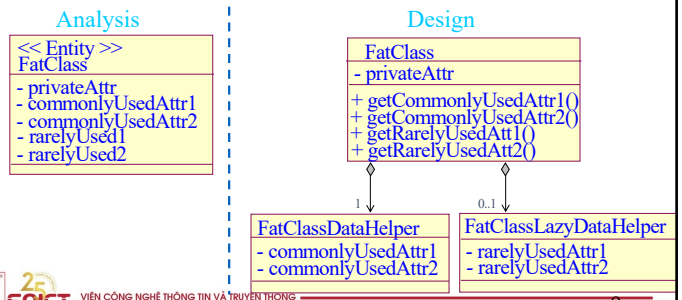
- Các lớp biên liên quan tới giao diện người dùng
  - Công cụ thiết kế và phát triển giao diện người dùng được sử dụng là gì?
  - Với công cụ phát triển giao diện người dùng đó thì có thể tạo ra được bao nhiêu phần trăm giao diện?
- Các lớp biên của những hệ thống ngoài (external subsystem)
  - Thông thường mô hình hóa thành subsystem



8

Những chiến lược để nhận diện ra các lớp điều khiển

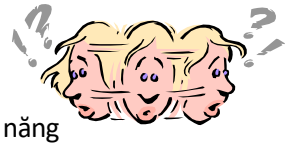
- Các lớp điều khiển thường mang tính chủ động và
- Những yêu cầu về hiệu năng có thể bắt buộc chúng ta phải tái cấu trúc (refactoring)



9

Những chiến lược thiết kế các lớp điều khiển

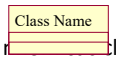
- Những vấn đề đặt ra với các lớp điều khiển?
  - Chúng thực sự cần thiết không?
  - Có nên chia nhỏ ra?
- Việc quyết định dựa vào?
  - Độ phức tạp
  - Khả năng thay đổi
  - Khả năng phân bổ và hiệu năng
  - Quản lý giao dịch

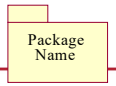


10

Review: Class and Package

- Class là gì?
  - Một bản mô tả của một tập các đối tượng có chung trách nhiệm, quan hệ, hành động, thuộc tính và ngữ nghĩa
- Package là gì?
  - Một công cụ có mục đích gom nhóm các class lại với nhau
  - Một thành phần mà có thể chứa các thành phần khác

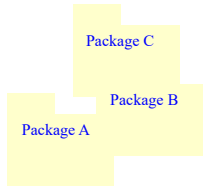




11

Nhóm các lớp thiết kế vào trong các Packages

- Việc gom nhóm các lớp vào trong một package có thể dựa vào các tiêu chí:
  - Các thành phần cấu hình
  - Nơi phân bổ tài nguyên cho các đội phát triển dự án
  - Phản ánh các thể loại người dùng
  - Biểu diễn các sản phẩm và dịch vụ có sẵn



12

Gợi ý trong việc gom nhóm các lớp biên

Nếu như giao diện hệ thống sẽ trải qua những sự thay đổi đáng kể

If it is **unlikely** the system interface will undergo considerable changes

Đặt các lớp biên và những package riêng biệt

Các lớp biên được gom nhóm với những lớp có chức năng liên quan

13

Gom nhóm các lớp có chức năng liên quan tới nhau

- Những tiêu chí để xem xét các lớp có liên quan về mặt chức năng hay không :
  - Sự thay đổi trong hành vi và cấu trúc của một class làm thay đổi các class khác
  - Xóa một class sẽ tác động tới những class khác
  - Hai đối tượng có sự trao đổi thông điệp qua lại nhiều hoặc có sự giao tiếp qua lại phức tạp
  - Một lớp biên có thể liên quan về mặt chức năng với lớp thực thể nếu chức năng của lớp biên đó là biểu diễn thực thể
  - Hai class tương tác hoặc chịu ảnh hưởng cùng bởi một tác nhân

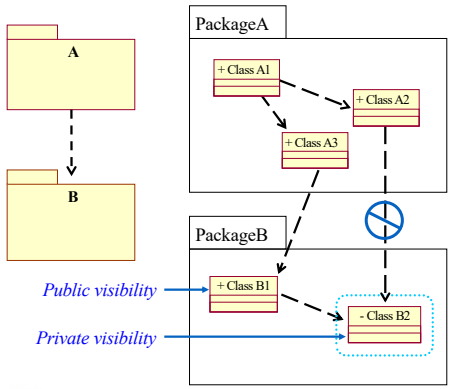
14

Gom nhóm các lớp có chức năng liên quan tới nhau (tiếp)

- Những tiêu chí để xem xét các lớp có liên quan về mặt chức năng hay không (tiếp):
  - Hai lớp có mối quan hệ với nhau
  - Một class khởi tạo đối tượng của class khác
- Những tiêu chí để không đặt hai class cùng chung một package:
  - Hai class liên quan tới các nhân khác nhau
  - Một class tùy chọn và một class mang tính bắt buộc

15

Sự phụ thuộc giữa các package



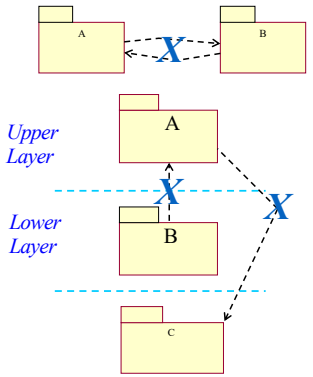
Chỉ những class được khai báo public thì mới có thể đọc tham chiếu từ bên ngoài package

Nguyên tắc bao đóng (Encapsulations)

16

Package Coupling: Gợi ý

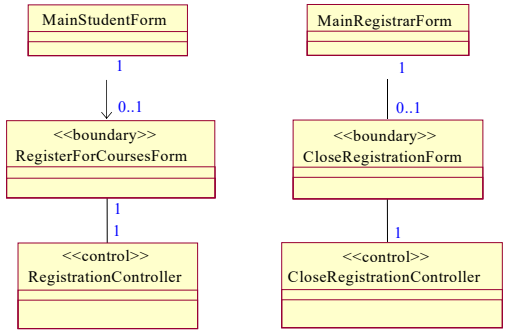
- Các package không nên phụ thuộc vòng
- Các package ở tầng dưới không nên phụ thuộc vào các package ở tần trên
- Về cơ bản sự phụ thuộc không nên nhảy cóc qua các tần trung gian



X = Coupling violation

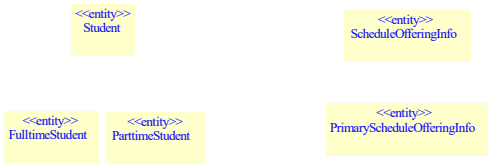
17

Ví dụ: Registration Package



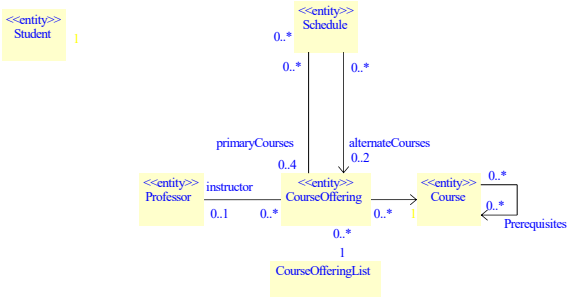
18

Ví dụ: University Artifacts Package: Generalization



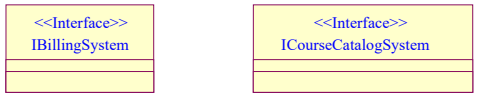
19

Ví dụ: University Artifacts Package: Associations



20

Ví dụ: External System Interfaces Package



21

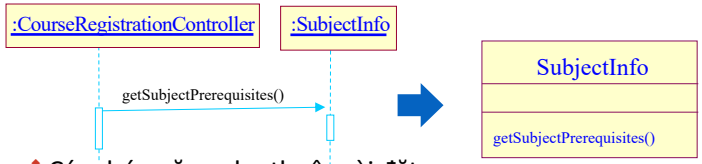
Nội dung

1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

22

2.1. Định nghĩa các thao tác

- Các thông điệp được hiển thị trong các sơ đồ tương tác



- Các chức năng phụ thuộc cài đặt
  - Chức năng quản lý
  - Cần cho việc sao chép các class
  - Cần cho việc kiểm thử chất lượng

23

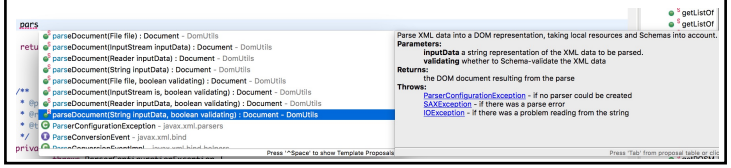
Name and Describe the Operations

- Create appropriate operation names
  - Indicate the outcome
  - Use client perspective
  - Are consistent across classes
- Define operation signatures
  - operationName([direction]parameter: class,..) : returnType
    - Direction is **in** (default), **out** or **inout**
    - Provide short description, including meaning of all parameters

24

Đặc tả cho chương trình

```
/**
  * Parse XML data into a DOM representation, taking local resources and Schemas
into account.
  * @param inputData a string representation of the XML data to be parsed.
  * @param validating whether to Schema-validate the XML data
  * @return the DOM document resulting from the parse
  * @throws ParserConfigurationException if no parser could be created
  * @throws SAXException if there was a parse error
  * @throws IOException if there was a problem reading from the string
*/
public static Document parseDocument(String inputData, boolean validating) throws
ParserConfigurationException, SAXException, IOException {
          //Change to UnicodeReader for utf-8
          ...
 }
```



25

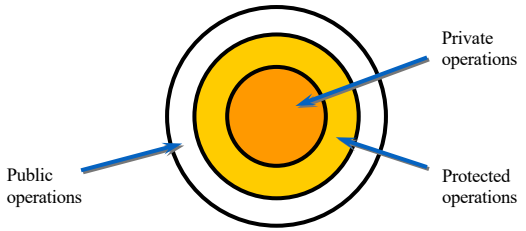
Hướng dẫn: Thiết kế chữ ký cho các thao tác (operation signatures)

- Khi thiết kế các chữ ký cho các thao tác ta cần quan tâm nếu như các tham số truyền vào là:
  - Tham chiếu hoặc tham trị
  - Bị thay đổi bởi operation đó
  - Tùy chọn (có thể có hoặc không)
  - Gán giá trị mặc định
  - Trong miền giá trị không hợp lệ
- Càng ít tham số thì càng tốt
- Truyền các đối tượng thay vì các bit dữ liệu

26

Tầm nhìn thao tác (Operation Visibility)

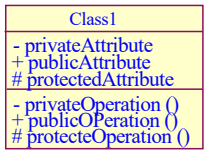
- Tầm nhìn được sử dụng để ép buộc sự bao đóng
- Có thể là public, protected, or private



27

Biểu thị tầm nhìn như thế nào?

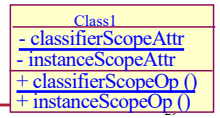
- Ta biểu thị tầm nhìn bởi các ký hiệu dưới đây:
  - \+ Public access
  - \# Protected access
  - \- Private access



28

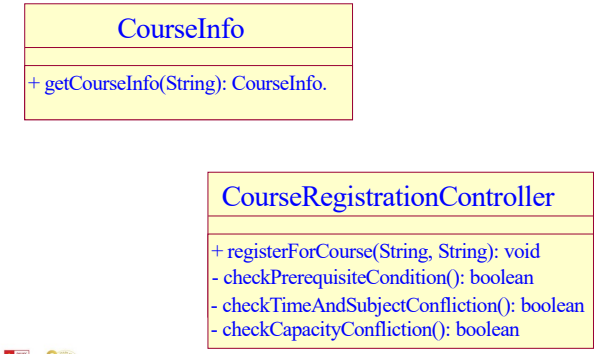
Phạm vi

- Nhận diện số lượng đối tượng thực thể của các thuộc tính/thao tác
  - Đối tượng thực thể: một đối tượng cho mỗi đối tượng lớp
  - Đối lượng lớp: một đối tượng cho tất cả các đối tượng lớp
- Phạm vi của lớp được biểu thị bởi dấu gạch dưới tên của các thuộc tính



29

Course Registration CS: Operations for CourseInfo. and CourseRegistrationController



30

2.2. Định nghĩa ra các phương thức

- Phương thức là gì ?
  - Mô tả cài đặt cho một thao tác
- Mục đích
  - Định nghĩa ra các phương diện đặc biệt trong việc cài đặt các thao tác
- Những điều cần quan tâm:
  - Thuật toán cụ thể
  - Những đối tượng và thao tác khác được sử dụng
  - Những thuộc tính và tham số được cài đặt và sử dụng như thế nào
  - How relationships are to be implemented and used

31

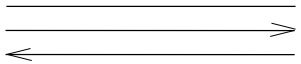
Nội dung

1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

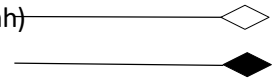
32

Mỗi quan hệ giữa các class

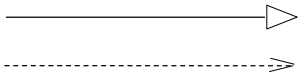
- Association (kết hợp)



- Aggregation (kết tập)
- Composition (hợp thành)



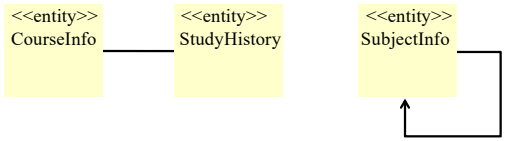
- Inheritance (Kế thừa)
- Dependence (Phụ thuộc)



33

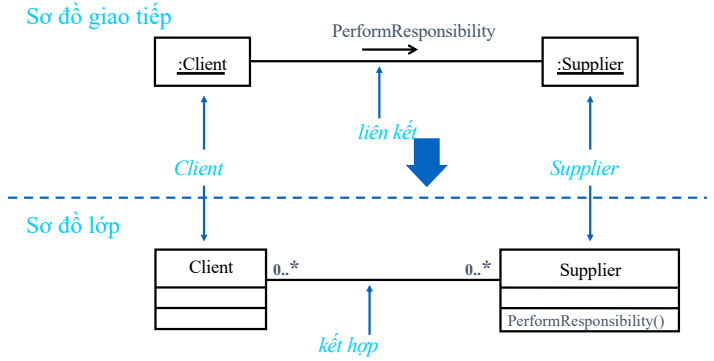
3.1. Association là gì?

- Mối quan hệ về mặt ngữ nghĩa giữa hai hoặc nhiều class thiết lập nên sự liên kết giữa các đối tượng tạo ra từ các class đó
- Một mối quan hệ về mặt cấu trúc, thể hiện rằng các đối tượng của một vật thì được liên kết tới đối tượng của vật khác



34

Tìm Association

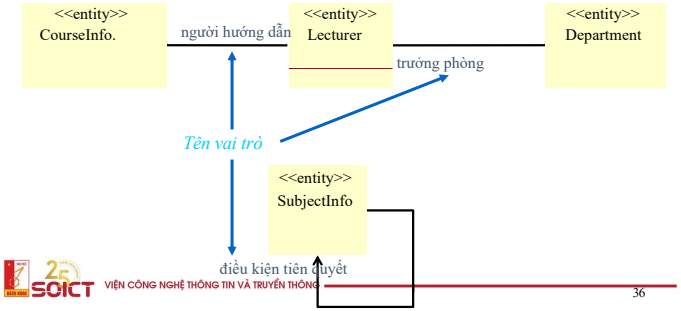


Mối quan hệ cho mỗi liên kết

35

3.1.1. Vai trò là gì?

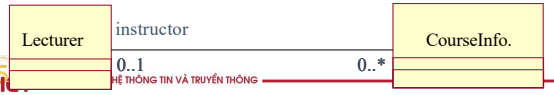
- Là bộ mặt mà class tham gia liên kết



36

3.1.2. Multiplicity là gì?

- Multiplicity là số lượng đối tượng của một class có liên quan tới MỘT đối tượng của class khác.
- Với mỗi liên kết, chúng ta sẽ có hai trọng số ở hai đầu liên kết.
  - Mỗi giảng viên có thể giảng dạy nhiều khóa học
  - Mỗi khóa hoc sẽ được giản dạy bởi một hoặc không có giảng viên nào.



37

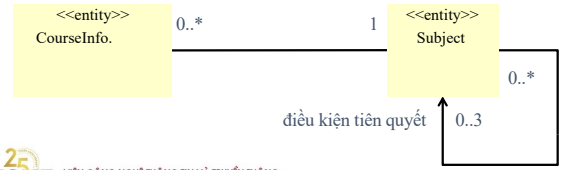
Các giá trị chỉ số

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

38

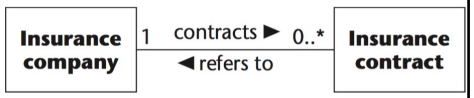
Multiplicity có ý nghĩa gì?

- Multiplicity trả lời 2 câu hỏi:
  - Liên kết là bắt buộc hay tùy chọn?
  - Số lượng đối tượng nhỏ nhất và lớn nhất có thể được liên kết tới một đối tượng khác?



39

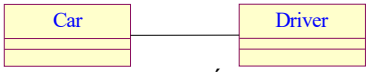
Cài đặt bằng Java



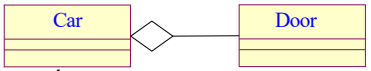

```
//InsuranceCompany.java file
public class InsuranceCompany
{
   // Many multiplicity can be implemented using Collection
   private List<InsuranceContract> contracts;


   /* Methods */
}
// InsuranceContract.java file
public class InsuranceContract
{
   private InsuranceCompany refers_to;

   /* Methods */
}
```

3.1.3. Các loại liên kết

- Association
  - use-a (sử dụng)
  - Các đối tượng của một class được kết hợp với các đối tượng của class khác



- Aggregation
  - có / là một phần
  - Liên kết mạnh, một đối tượng của một lớp tạo ra các đối tượng của lớp khác

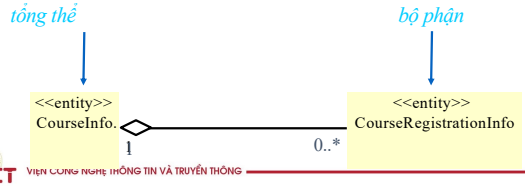


- Composition
  - Kết tập mạnh, đối tượng được hợp thành không thể được chia sẻ bởi các đối tượng khác và chết đi cùng với lớp kết tập nó
  - Chia sẻ vòng đời

41

Review: Kết tập là gì?

- Một dạng đặc biệt của kết hợp (association), mô hình hóa nên mối quan hệ tổng thể - bộ phận giữa một lớp kết tập-aggregate (the whole) và các phần của nó
  - Kết tập là mối quan hệ "is a part of" (là một phần của).
- Multiplicity được biểu diễn như các liên kết khác.



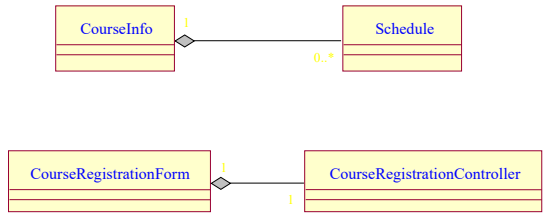
42

Review: Hợp thành là gì?

- Một dạng đặc biệt của kết tập cùng tính sở hữu mạnh và có cùng vòng đời với lớp kết tập nó.
- Lớp tổng thể (the whole) sở hữu lớp bộ phần và chịu trách nhiệm cho sự khởi tạo và phá hủy lớp thành phần.
  - Thành phần sẽ bị mất đi khi tổng thể bị mất đi.
  - Thành phần có thể bị bỏ đi trước khi tổng thể bị phá hủy.

43

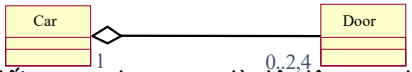
"Register for course" Use case



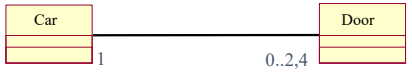
44

Dùng association hay aggregation?

- Nếu 2 đối tượng được gắn chặt bởi một mối quan hệ tổng thể - bộ phận
  - Sử dụng quan hệ aggregation.



- Nếu 2 đối tượng được xem là độc lập tuy nhiên vẫn có sự liên kết giữa chúng
  - Sử dụng quan hệ association.



Nếu nghi ngờ, hãy chọn association.

45

Aggregation – Cài đặt trong java

```
class Car {
    private List<Door> doors;
    Car(String name, List<Door> doors) {
        this.doors = doors;
    }

    public List<Door> getDoors() {
        return doors;
    }
}
```


46

Composition – Cài đặt trong java

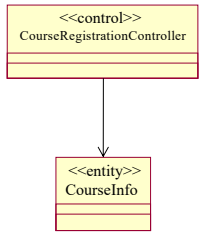
```
final class Car {
    // For a car to move, it need to have a engine.
    private final Engine engine; // Composition
    //private Engine engine;     // Aggregation

    Car(Engine engine) {
        this.engine = engine;
    }

    // car start moving by starting engine
    public void move() {
        //if(engine != null)
        {
            engine.work();
            System.out.println("Car is moving ");
        }
    }
```

```
class Engine {
    // starting an engine
    public void work() {
        System.out.println("Engine of car has been started ");
    }
}
```


47

3.1.4. Sự điều hướng

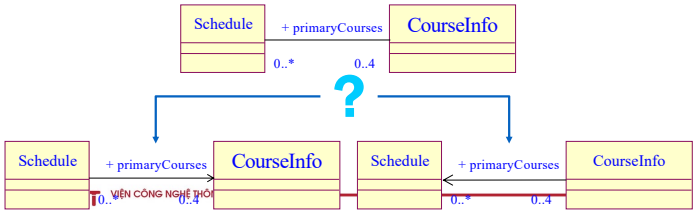
- Chỉ ra một class liên kết có thể điều hướng tới một class mục tiêu khi sử dụng association hay không



48

Sự điều hướng: Hướng nào thì thực sự cần thiết?

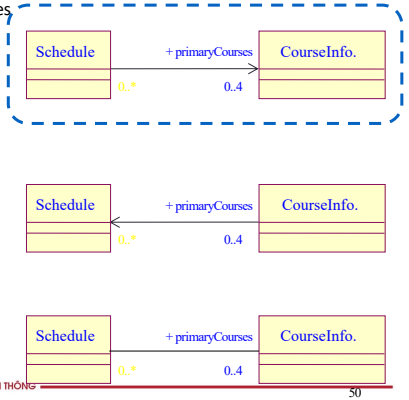
- Khai thác sơ đồ tương tác
- Thậm chỉ ngay cả khi cả hai hướng dường như được yêu cầu, một hướng cũng có thể thực hiện
  - Sự điều hướng theo một phía thường không phổ biến
  - Số lượng đối tượng của một class nhỏ



49

Ví dụ: Điều chỉnh hướng

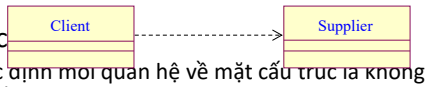
- Số lượng Schedules thì nhỏ hoặc không cần một danh sách Schedules để CourseInfo xuất hiện
- số lượng CourseInfo thì nhỏ, hoặc không cần một danh sách CourseInfo cho một Schedule
- Số lượng CourseInfo và Schedules thì không nhỏ
- Phải có khả năng điều hướng theo cả hai phía



50

3.2. Sự phụ thuộc

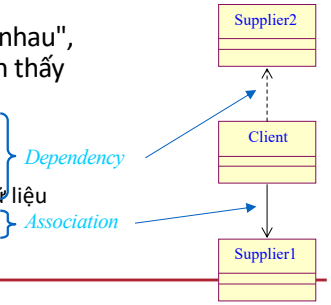
- Sự phụ thuộc là gì?
  - Một mối quan hệ giữa 2 đối tượng
- Mục đíc
  - Xác định mối quan hệ về mặt cấu trúc là không cần thiết
- Điều cần tìm kiếm :
  - Điều gì làm cho Supplier thấy được bởi Client



51

Dependencies vs. Associations

- Associations là những mối quan hệ cấu trúc
- Dependencies là những mối quan hệ không có cấu trúc
- Để các đối tượng "biết lẫn nhau", chúng phải có khả năng nhìn thấy
  - Tham chiếu biến cục bộ
  - Tham chiếu tham số
  - Tham chiếu toàn cục
  - Tham chiếu theo trường dữ liệu



52

Associations vs. Dependencies in Collaborations

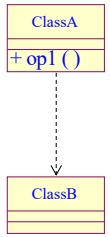
- Một đối tượng của associton là một liên kết
  - Tất cả các liên kết trở thành liên kết ngoại trừ global, local, parameter
  - Các mối quan hệ là phụ thuộc ngữ cảnh
- Dependencies là những liên kết tạm thời:
  - Giới hạn về mặt thời gian
  - Mối quan hệ độc lập về mặt ngữ cản
  - Một mối quan hệ tổng quát

Một dependency là một loại quan hệ thứ cấp, không cung cấp nhiều thông tin về quan hệ. Để xem chi tiết hơn cần kiểm tra sự giao tiếp giữa các class

53

3.2.1. Tầm nhìn biến cục bộ

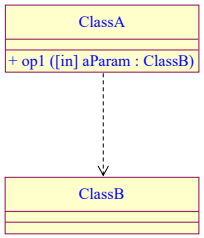
- op1() operation chứa một biến cục bộ với loại dữ liệu là ClassB



54

3.2.2 Tầm nhìn theo dạng tham số

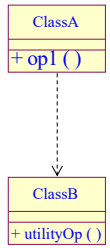
- Đối tượng của ClassB được truyền tới đối tượng của ClassA



55

3.2.3. Tầm nhìn toàn cục

- Đối tượng ClassUtility thì có thể nhìn được bởi vì nó là toàn cục



56

Nhận diện sự phụ thuộc

- Quan hệ bền vững — Association (tầm nhìn trường dữ liệu)
- Quan hệ tạm thời — Dependency
  - Nhiều đối tượng chia sẻ cùng đối tượng
    - Truyền đối tượng theo dạng tham số (parameter visibility)
    - Tạo một đối tượng toàn cục (global visibility)
  - Nhiều đối tượng không chia sẻ cùng đối tượng (local visibility)
- Thời gian khởi tạo và phá hủy hết bao lâu?
  - Chi phí cao? Sử dụngtầm nhìn trường, tham số hoặc toàn cục
  - Cố gắng tạo ra các mối quan hệ đơn giản nhất có thể

57

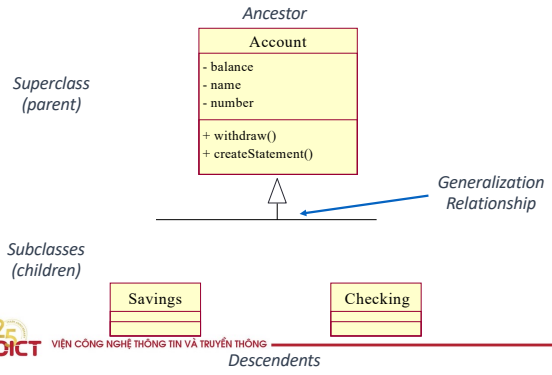
3.3. Tổng quát hóa

- Một mối quan hệ giữa các class nơi mà một class chia sẻ cấu trúc và/hoặc hành vi của một hoặc nhiều class.
- Định nghĩa một sự phân cấp tính trừu tượng nơi các class con kết thừa từ một hoặc nhiều class cha.
  - Đơn kế thừa
  - Đa kế thừa
- Là mối quan hệ "is a kind of" (là một loại của)

58

Ví dụ: Đơn kế thừa

- Một lớp kế thừa từ một lớp khác



59

Nội dung

1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

60

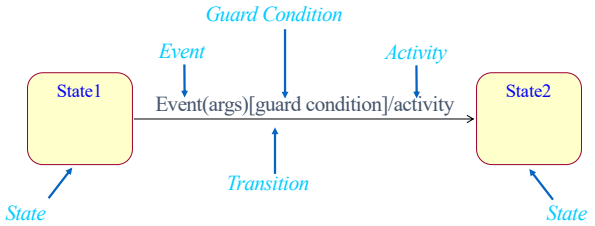
4. Định nghĩa các trạng thái

- Mục đích
  - Thiết kế cách trạng thái của đối tượng ảnh hưởng đến hành vi của nó
  - Thiết kế máy trạng thái để mô hình hóa hành vi này
- Những điều cần quan tâm:
  - Những đối tượng nào có những trạng thái quan trọng?
  - Làm thế nào để xác định được các trạng thái có thể của một đối tượng?
  - Làm thế nào để máy trạng thái ánh xạ tới phần còn lại của mô hình?

61

Máy trạng thái là gì?

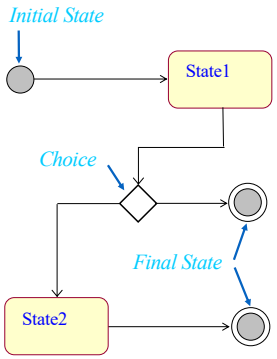
- Một sơ đồ có hướng của các trạng thái được liên kết với nhau bằng các dịch chuyển
- Mô tả lịch sử vòng đời của một đối tượng



62

Các trạng thái

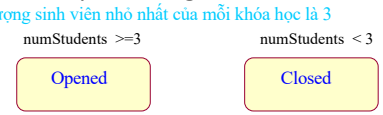
- Initial state (trạng thái khởi tạo)
  - Bắt đầu khi một đối tượng được tạo ra
  - Mang tính bắt buộc và chỉ có thể có duy nhất một trạng thái khởi tạo
- Choice
  - Việc đánh giá động của một chuỗi các điều kiện
  - Chỉ kích hoạt chuyển dịch đầu tiên
- Final state
  - Chỉ ra sự kết của vòng đời của đối tượng
  - Tùy chọn, có thể có nhiều hơn một trạng thái kết thúc



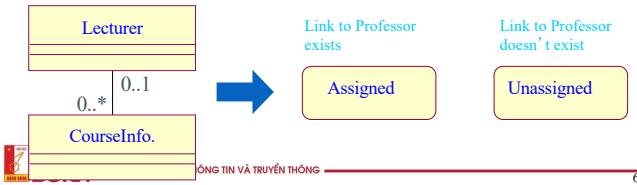
63

Nhận diện và định nghĩa các trạng thái

- Các thuộc tính quan trọng và các thuộc tính động



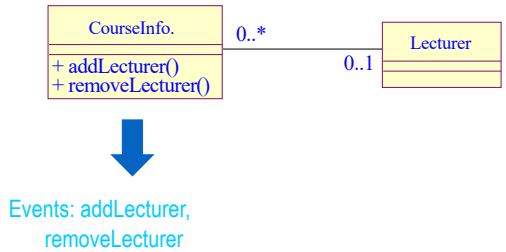
- Liên kết tồn tại và liên kết không tồn tại



64

Nhận diện các sự kiện

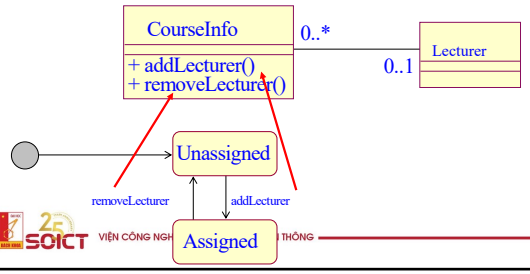
- Nhìn vào các thao tác của class, interface



65

Nhận diện sự chuyển dịch

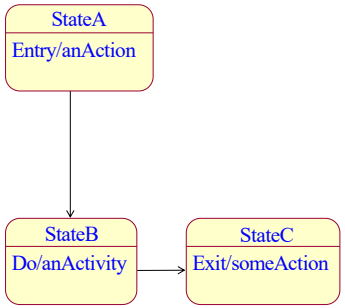
- Với mỗi trạng thái, xác định sự kiện nào gây ra sự chuyển dịch tới trạng thái đó, bao gồm các điều kiện khi cần thiết
- Các chuyển dịch mô tả những điều xảy khi có sự phải hồi lại với sự kiện nhận được



66

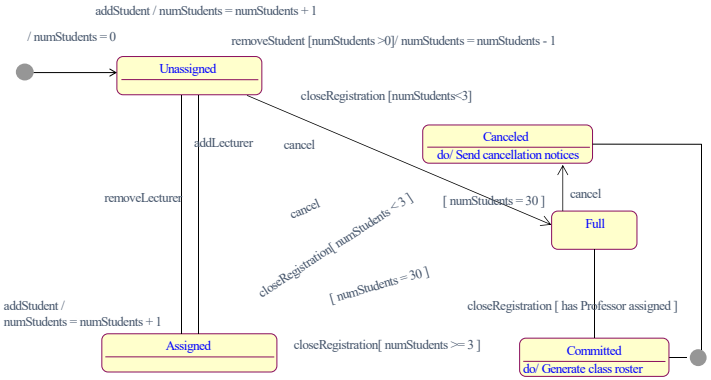
Thêm các hành động

- Đầu vào
  - Được thực thi khi trạng thái được tham gia
- Thực hiện
  - Quá trình thực thi diễn ra
- Thoát
  - Được thực thi khi trạng thái kết thúc



67

Ví dụ: máy trạng thái



68

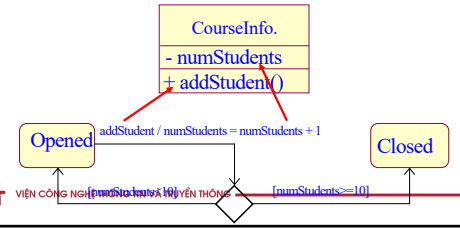
Đối tượng nào có những trạng thái quan trọng

- Những đối tượng mà vai trò của nó được phân biệt bởi các chuyển dịch
- Các usecase phức tạp và bị điều khiển bởi trạng thái
- Không cần thiết để mô hình hóa các đối tượng như:
  - Những đối tượng có sự ánh xạ đơn giản với cài đặt
  - Những đối tượng không bị điều khiển bởi trạng thái
  - Những đối tượng có duy nhất một trạng thái tính toán

69

Máy trạng thái ánh xạ tới phần còn lại của mô hình như thế nào?

- Các sự kiện có thể ánh xạ tới các thao tác
- Các phương thức nên được cập nhật cùng các thông tin trạng thái cụ thể
- Các trạng thái thường được biểu diễn sử dung các thuộc tính
  - Diều này đóng vai trò như đầu vào cho "*Định nghĩa các thuộc tính*" bước



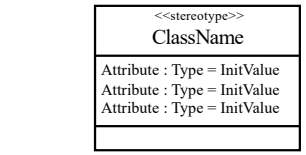
70

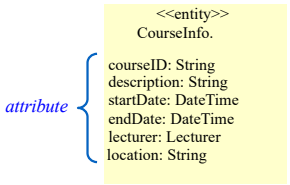
Nội dung

1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

71

Review: Thuộc tính là gì?





72

5.1. Nhận diện các thuộc tính

- Những đặc trưng của các class xác định
- Thông tin được lưu giữ bởi các class xác định
- Những "Danh từ" không tạo thành class
  - Những thông tin mà giá trị của nó không quan trọng
  - Những thông tin duy nhất được sở hữu bởi đối tượng
  - Những thông tin mà không có hành vi

73

5.1. Nhận diện các thuộc tính (2)

- Kiểm tra các mô tả về phương thức
- Kiểm tra các trạng thái
- Kiểm tra bất kỳ thông tin nào
- mà bản thân class đó cần duy trì

74

5.2. Biểu diễn các thuộc tính

- Mô tả tên, kiểu và giá trị mặc định tùy chọn
  - attributeName : Type = Default
- Tuân theo quy ước đặt tên của ngôn ngữ và dự án
- Kiểu nên là kiểu dữ liệu cơ bản trong ngôn ngữ cài đặt
  - dữ liệu có sẵn, dữ liệu người dùng định nghĩa, hoặc class người dùng định nghĩa
- Thiết lập phạm vi (tầm nhìn)
  - Public: + Private: -
    Protected: #

75

5.3. Những thuộc tính được dẫn xuất

- Thuộc tính dẫn xuất là gì?
  - Một thuộc tính mà giá trị của nó có thể được tính toán dựa và giá trị của các thuộc tính khác
- Sử dụng chúng khi nào?
  - Khi không có đủ thời gian để tính toán lại giá trị mỗi lần nó cần được sử dụng
  - Khi có sử đánh đổi về mặt hiệu năng giữa yêu cầu về hiệu năng và bộ nhớ

76

Ví dụ: Định nghĩa các thuộc tính

<<entity>>
Schedule

- scheduleID: int
- day: String
- teachingPeriod: int

<<boundary>>
CourseRegistrationForm

- courseID: String
- studentID: String

<<controll>>
CourseRegistrationController

- courseID: String
- studentID: String

<<entity>>
StudyHistory

- historyNo: int
- pass: boolean
- result: int
- studentID: String
- studentName: String

<<entity>>
CourseInfo.

- courseID: String
- description: String
- startDate: DateTime
- endDate: DateTime
- lecturer: Lecturer
- location: String

<<entity>>
SubjectInfo.

- subjectID: String
- subjectName: String
- goal: String
- description: String
- numberOfCredits: int

77

Nội dung

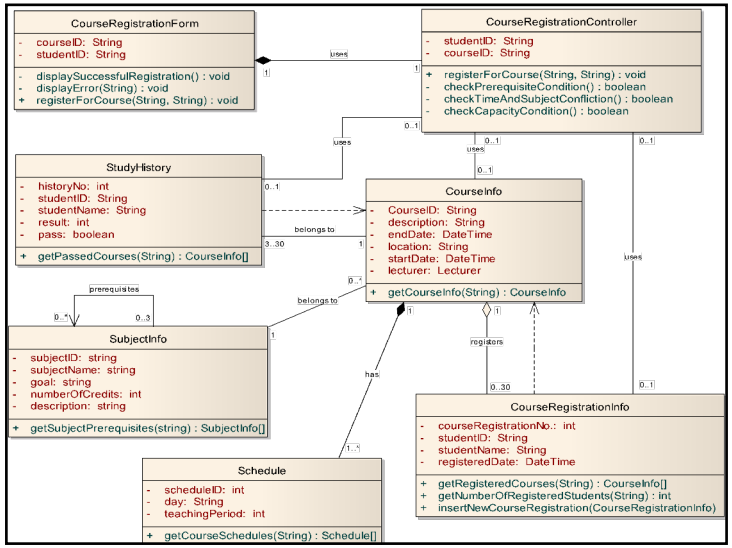
1. Tạo các lớp khởi tạo
2. Định nghĩa ra các thao tác/phương thức
3. Định nghĩa ra mối quan hệ giữa các lớp
4. Định nghĩa ra các trạng thái
5. Định nghĩa ra các thuộc tính
6. Sơ đồ lớp

78

6. Sơ đồ lớp

- Góc nhìn tĩnh của hệ thống
- Khi mô hình hóa góc nhìn tĩnh của hệ thống, sơ đồ lớp thương được sử dụng để mô hình theo một trong ba trường hợp sau
  - Từ vựng của một hệ thống
  - Sự hợp tác (Collaborations)
  - Một ngữ cảnh cơ sở dữ liệu logic

79



80

Review: Package là gì?

- Một công cụ với mục đích tổ chức các thành phần vào thành các nhóm.
- Một thành phần mô hình có thể chứa những thành phần khác.
- Một package có thể được sử dụng:
  - Để tổ chức mô hình phát triển
  - Nhưng một đơn vị quản lý phát triển

81

Review points: Lớp (class)

- Tên các class rõ ràng
- Sự trừu tượng được định nghĩa rõ ràng
- Thuộc tính/hành vi gắn với chức năng
- Thực hiện sự tổng quát hóa
- Tất cả các yêu cầu phần mềm được xử lý
- Những yêu cầu thì thống nhất cùng với máy trạng thái
- Vòng đời đối tượng được mô tả
- Lớp có hành vi được yêu cầu

82

Review points: Thao tác (operation)

- Operations dễ hiểu
- Mô tả trạng thái chính xác
- Những hành vi yêu cầu được đề xuất
- Các tham số được định nghĩa rõ ràng
- Những thông điệp được gắn hoàn toàn tới các thao tác
- Những đặc tả cài đặt cần chính xác
- Những chữ ký cần tuân thủ tiêu chuẩn
- Tất cả các operations cần thiết với Use-Case Realizations

83

Review points: Thuộc tính (attribute)

- Đơn khái niệm
- Tên mang tính mô tả
- Tất cả các thuộc tính cần thiết với Use-Case Realizations

84

Review points: Quan hệ (relationship)

- Tên các vai trò mang tính mô tả
- Trọng số giữa 2 đầu quan hệ cần chính xác

85

Câu hỏi?

86

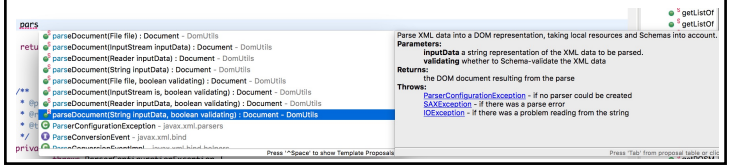
Thiết kế lớp

- Thiết kế các thuộc tính
  - Loại, mô tả
- Thiết kế thao tác
  - Chữ ký thao tác
  - Mục đích/mô tả của thao tác
  - Mục đích/mô tả của mỗi tham số
  - Mô tả giá trị trả về
  - Lỗi/Ngoại lệ (khi nào)
- Thiết kế phương thức
  - Thuật toán cụ thể
  - Sử dụng các tham số như thế nào

87

Đặc tả chương trình

```
/**
  * Parse XML data into a DOM representation, taking local resources and Schemas
into account.
  * @param inputData a string representation of the XML data to be parsed.
  * @param validating whether to Schema-validate the XML data
  * @return the DOM document resulting from the parse
  * @throws ParserConfigurationException if no parser could be created
  * @throws SAXException if there was a parse error
  * @throws IOException if there was a problem reading from the string
*/
public static Document parseDocument(String inputData, boolean validating) throws
ParserConfigurationException, SAXException, IOException {
          //Change to UnicodeReader for utf-8
          ...
 }
```



88