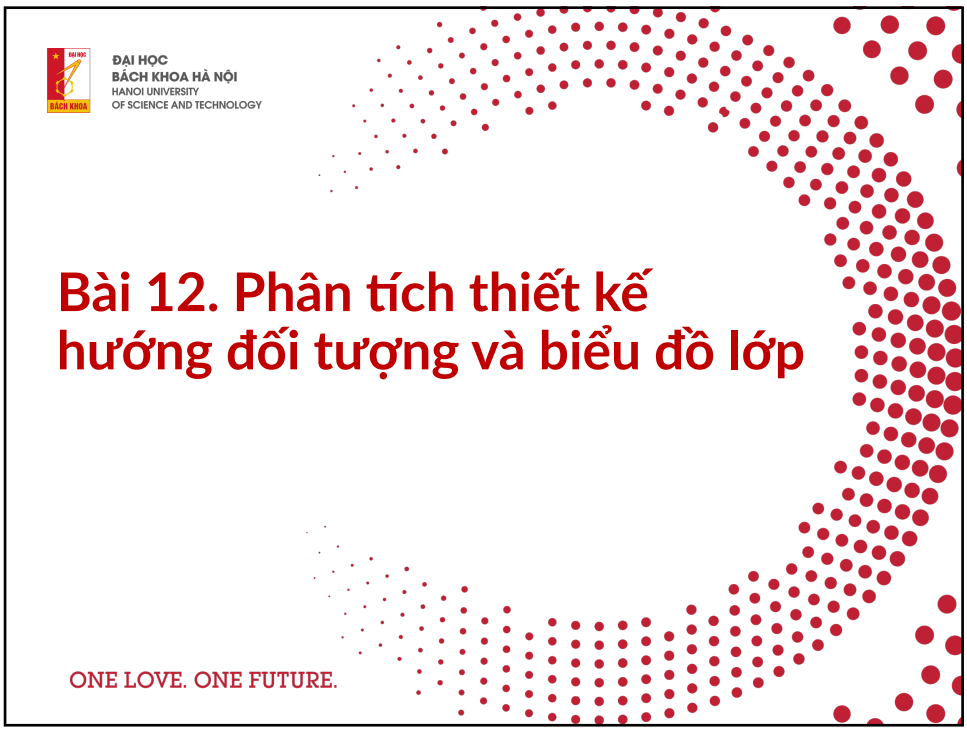

HUST
ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
ONE LOVE. ONE FUTURE.

1


ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY
Bài 12. Phân tích thiết kế hướng đối tượng và biểu đồ lớp
ONE LOVE. ONE FUTURE.

2

Nội dung

1. Phân tích thiết kế hướng đối tượng
2. Biểu đồ lớp
3. Quan hệ giữa các lớp
4. Ví dụ và bài tập

3

Nội dung

1. **Phân tích thiết kế hướng đối tượng**
2. Biểu đồ lớp
3. Quan hệ giữa các lớp
4. Ví dụ và bài tập

4

Tầm quan trọng của OOAD

- Nhiều người phát triển dự án
  - Cho rằng phần mềm chủ yếu được xây dựng bằng cách gõ "code" từ bàn phím
  - Không dành đủ thời gian cho quá trình phân tích và thiết kế phần mềm
- → Họ phải "cày bừa" để hoàn thành chương trình vì
  - Không hiểu hoặc hiểu sai yêu cầu
  - Giao tiếp với các thành viên không tốt
  - Không tích hợp được với module của đồng nghiệp…
- → Họ nhận ra rằng "Phân tích" và "Thiết kế" cần được coi trọng hơn, nhưng đã quá muộn

Tầm quan trọng của OOAD (2)

- Cần thiết lập một cơ chế hiệu quả để nắm bắt yêu cầu, phân tích thiết kế
- Cơ chế này phải như là một "ngôn ngữ thống nhất" giúp cho quá trình hợp tác hiệu quả giữa các thành viên trong nhóm phát triển phần mềm.
- → OOAD: Object Oriented Analysis and Design)

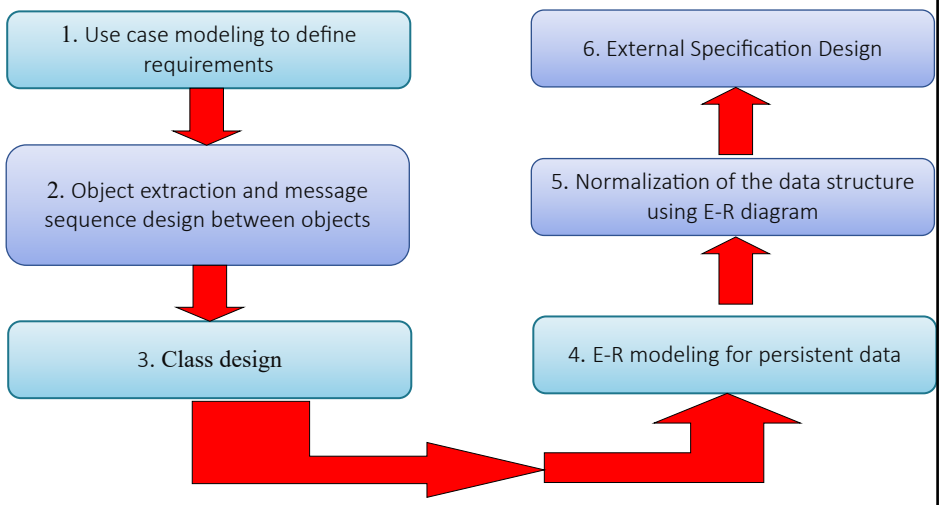
Mục đích của OOAD

- Chuyển các yêu cầu của bài toán thành một bản thiết kế của hệ thống sẽ được xây dựng
- Tập trung vào quá trình phân tích các YÊU CẦU của hệ thống và thiết kế các MÔ HÌNH cho hệ thống đó trước giai đoạn lập trình
- Được thực hiện nhằm đảm bảo mục đích và yêu cầu của hệ thống được ghi lại một cách hợp lý trước khi hệ thống được xây dựng
- Cung cấp cho người dùng, khách hàng, kỹ sư phân tích, thiết kế nhiều cái nhìn khác nhau về cùng một hệ thống

Phương pháp OOAD

- OOAD được chia thành 2 giai đoạn
  - Phân tích hướng đối tượng (OOA)
  - Thiết kế hướng đối tượng (OOD)
- OOA là giai đoạn nhằm tạo ra các mô hình cơ bản (mô hình khái niệm) của hệ thống dựa theo những gì khách hàng yêu cầu về hệ thống của họ
- OOD sẽ bổ sung thêm các thông tin thiết kế chi tiết cho các mô hình nói trên

Phương pháp OOAD (2)



9

OOA

- Xác định yêu cầu phần mềm
- Đặc tả yêu cầu phần mềm thông qua mô hình các đối tượng và tương tác giữa chúng
- Tạo được mô hình có các thành phần là đối tượng và khái niệm đời thực, dễ hiểu với người dùng
- Mô hình hóa các thực thể, giữ nguyên cấu trúc, quan hệ, hành vi giữa chúng

10

OOA (2)

- Ví dụ với 1 phòng bán ô tô:
  - Các thực thể:
    - Khách hàng
    - Người bán hàng
    - Phiếu đặt hàng
    - Phiếu (hoá đơn) thanh toán
    - Xe ô tô
  - Tương tác và quan hệ giữa các thực thể trên :
    - Người bán hàng dẫn khách hàng tham quan phòng trưng bày xe.
    - Khách hàng chọn một chiếc xe
    - Khách hàng viết phiếu đặt xe
    - Khách hàng trả tiền xe
    - Xe ô tô được giao đến cho khách hàng

OOD

- Thực thi các mô hình khái niệm là đầu ra của bước OOA
- Các khái niệm trong OOA được ánh xạ theo thành các lớp thực thi. Các ràng buộc, các giao diện được thiết kế. Kết quả là đặc tả chi tiết về hệ thống cần xây dựng, theo một công nghệ cụ thể được lựa chọn

OOD

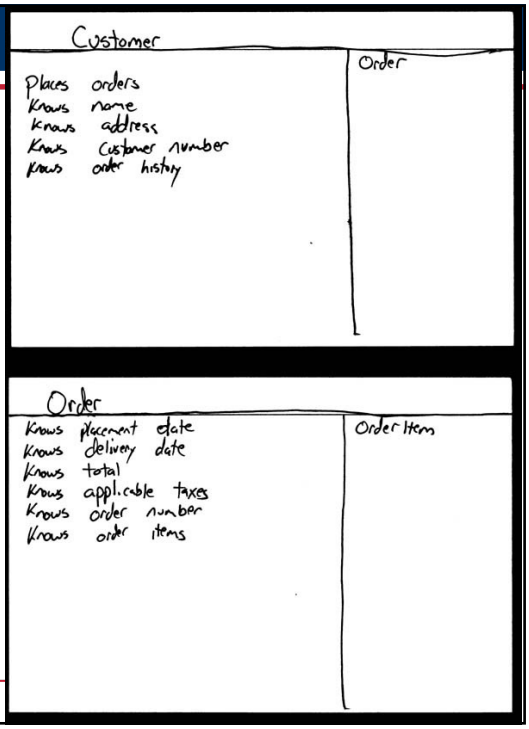
- Tổ chức chương trình thành các tập hợp đối tượng cộng tác
  - Mỗi đối tượng là thực thể của một lớp
- Thiết kế trên kết quả của OOA
  - Cải thiện, tối ưu hóa thêm
  - Thiết kế các
    - Phương thức (operations)
    - Thuộc tính (attributes)
    - Mối quan hệ giữa các lớp (classes)
  - Đưa ra các biểu đồ tĩnh và động
    - Tĩnh: biểu thị các lớp và đối tượng
    - Động: biểu thị tương tác giữa các lớp & phương thức hoạt động

Thiết kế biểu đồ lớp

- Mục tiêu: cần xác định các thành viên của mỗi lớp và quan hệ giữa các lớp
- Một trong các kỹ thuật được ứng dụng nhiều nhất là Thẻ Class-Responsibility-Collaboration (CRC) card.
- Mỗi thẻ thể hiện một lớp, trên thẻ chúng ta lưu lại các thông tin sau về các lớp:
  - 1. Tên của lớp. Thông thường người ta đặt tên lớp liên quan đến vai trò của lớp, chúng ta sẽ sử dụng lớp để làm gì.
  - 2. Trách nhiệm của lớp: lớp có thể làm gì. Thông thường các thông tin ở đây bao gồm tên của các hàm thành phần
  - 3. Tương tác của lớp: lớp này có thể tương tác được với những lớp nào khác

CRC Card

Class Name	
Responsibilities	Collaborators



15

Thiết kế đối tượng (1/2)

- Trong PT&TK hướng đối tượng người ta đã tổng kết 5 bước để thiết kế đối tượng:
  - Bước 1. Phát hiện đối tượng (Object discovery). Bước này được thực hiện ở giai đoạn phân tích chương trình.
  - Bước 2. Lắp ráp đối tượng (Object assembly). Bước tìm kiếm các đặc điểm của đối tượng để thêm vào các thuộc tính, các hàm thành phần cho đối tượng

16

Thiết kế đối tượng (2/2)

- Trong PT&TK hướng đối tượng người ta đã tổng kết 5 bước để thiết kế đối tượng:
  - Bước 3. Xây dựng hệ thống (System construction). Trong giai đoạn này chúng ta phát triển các đối tượng, xem xét các tương tác giữa các đối tượng để hình thành hệ thống hoạt động.
  - Bước 4. Mở rộng hệ thống (System extension). Khi chúng ta thêm vào các tính năng của hệ thống, cần thêm các lớp mới, các đối tượng mới và các tương tác giữa các đối tượng này với các đối tượng đã có trong hệ thống.
  - Bước 5. Tái sử dụng đối tượng (Object reuse). Đây là một trong những thử nghiệm quan trọng của các đối tượng và lớp trong thiết kế phần mềm. Chúng ta cần phải sử dụng lại các lớp và các đối tượng trong phần mềm (thông qua tính kế thừa và tương tác giữa các đối tượng)

Lưu ý (1/2)

- Một số điểm lưu ý khi phát triển các lớp
  - 1. Cần tạo ra lớp trước, sau đó mới nghĩ tới việc phát triển và hoàn thiện lớp trong quá trình giải quyết bài toán
  - 2. Khi phân tích hay phát triển các lớp không nên tập trung xác định tất cả thành viên một lớp, chúng ta sẽ biết rõ hơn khi phát triển hệ thống (learns as you go)
  - 3. Việc phát hiện ra các lớp cần thiết cho chương trình là một trong những nhiệm vụ chính của thiết kế hệ thống, nếu chúng ta đã có những lớp này (trong một thư viện lớp nào đó chẳng hạn) thì công việc sẽ dễ dàng hơn

Lưu ý (2/2)

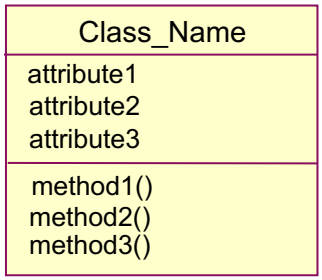
- Một số điểm lưu ý khi phát triển các lớp
  - 4. Khi lập trình cần tuân thủ theo các thiết kế đã làm. Không nên băn khoăn khi không sử dụng phương pháp lập trình truyền thống và thấy choáng ngợp trước số lượng lớn các đối tượng.
  - 5. Luôn giữ nguyên tắc: mọi vấn đề cần giải quyết theo phương án đơn giản nhất, không phức tạp hóa. Sử dụng nguyên lý của Occam Razor: *Lớp đơn giản nhất bao giờ cũng là lớp tốt nhất, hãy bắt đầu bằng những cái đơn giản và chúng ta sẽ kết thúc bằng những hệ thống phức tạp*

Nội dung

1. Phân tích thiết kế hướng đối tượng
2. **Biểu đồ lớp**
3. Quan hệ giữa các lớp
4. Ví dụ và bài tập

Lớp (Class)

- Sử dụng hình chữ nhật gồm 3 thành phần
  - Tên lớp
  - Các thuộc tính
  - Các phương thức



Biểu diễn thuộc tính

- Chỉ ra tên, kiểu và giá trị mặc định nếu có
  - `attributeName : Type = Default`
- Tuân theo quy ước đặt tên của ngôn ngữ cài đặt và của dự án.
- Kiểu (type) nên là kiểu dữ liệu cơ bản trong ngôn ngữ thực thi
  - Kiểu dữ liệu có sẵn, kiểu dữ liệu người dùng định nghĩa, hoặc lớp tự định nghĩa.

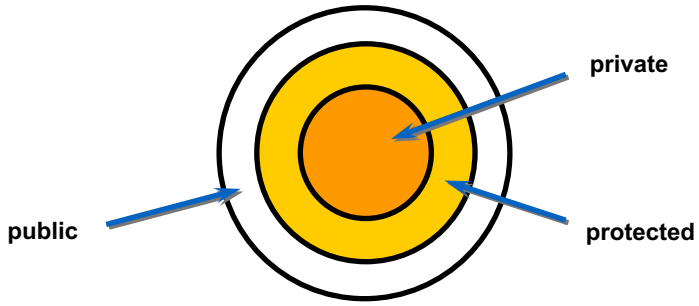
Mô tả phương thức

- Tên phương thức:
  - Mô tả kết quả
  - Sử dụng góc nhìn của đối tượng khách (client – đối tượng gọi)
  - Nhất quán giữa các lớp
- Chữ ký của phương thức:

  operationName([direction] parameter:class,...):returnType
  - Direction: **in** (mặc định), **out** hoặc **inout**

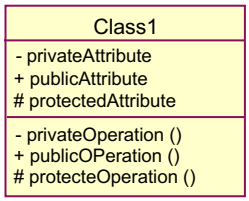
Phạm vi truy cập (Visibility)

- Phạm vi truy cập được sử dụng để thực hiện khả năng đóng gói



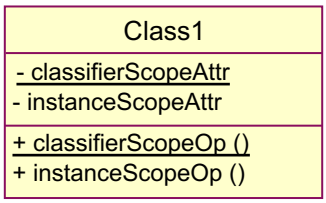
Phạm vi truy cập được biểu diễn như thế nào?

- Các ký hiệu sau được sử dụng:
  - \+ Public access
  - # Protected access
  - \- Private access



Phạm vi (Scope)

- Xác định số lượng thể hiện của thuộc tính/thao tác:
  - Instance: Một thể hiện cho mỗi thể hiện của mỗi lớp
  - Classifier: Một thể hiện cho tất cả các thể hiện của lớp
- Phạm vi Classifier được ký hiệu bằng cách gạch dưới tên thuộc tính/thao tác.



Ví dụ: Scope

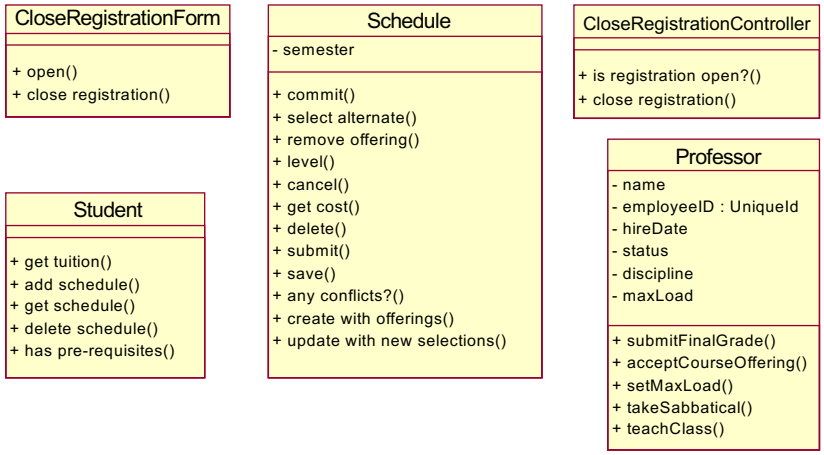
Student
- name - address - studentID - nextAvailID : int
+ addSchedule ([in] theSchedule : Schedule, [in] forSemester : Semester) + getSchedule ([in] forSemester : Semester) : Schedule + hasPrerequisites ([in] forCourseOffering : CourseOffering) : boolean # passed ([in] theCourseOffering : CourseOffering) : boolean + getNextAvailID () : int

Biểu đồ lớp là gì?

- Biểu đồ lớp chỉ ra sự tồn tại của các lớp và mối quan hệ giữa chúng trong bản thiết kế logic của một hệ thống
  - Chỉ ra cấu trúc tĩnh của mô hình như lớp, cấu trúc bên trong của chúng và mối quan hệ với các lớp khác.
  - Chỉ ra tất cả hoặc một phần cấu trúc lớp của một hệ thống.
  - Không đưa ra các thông tin tạm thời.
- Khung nhìn tĩnh của một hệ thống chủ yếu hỗ trợ các yêu cầu chức năng của hệ thống.

Biểu đồ lớp (Class Diagram – CD)

- Khung nhìn tĩnh của hệ thống

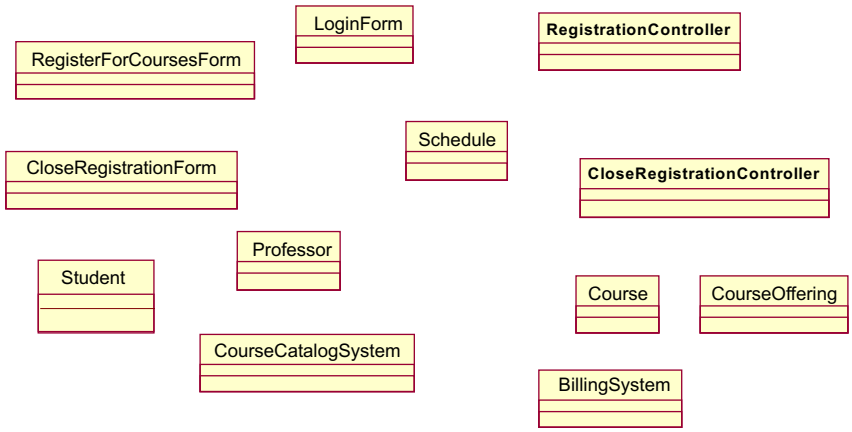


Khi nào sử dụng biểu đồ lớp?

- Từ vựng của hệ thống (Vocabulary)
  - Khi trừu tượng hóa một phần hoặc bên ngoài hoặc biên của hệ thống.
  - Chỉ ra kết quả trừu tượng hóa và trách nhiệm của chúng
- Cộng tác (Collaboration)
  - Nhóm các lớp và các thành phần khác làm việc cùng nhau để thực hiện một công việc nào đó.
- Lược đồ CSDL logic (Logical database schema)
  - Tương tự như bản thiết kế khái niệm cho CSDL
  - Chứa các đối tượng cần lưu trữ lâu dài tức là cần lưu trong CSDL

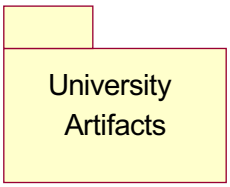
Ví dụ Biểu đồ lớp

- Có cách nào tốt hơn để tổ chức biểu đồ lớp?

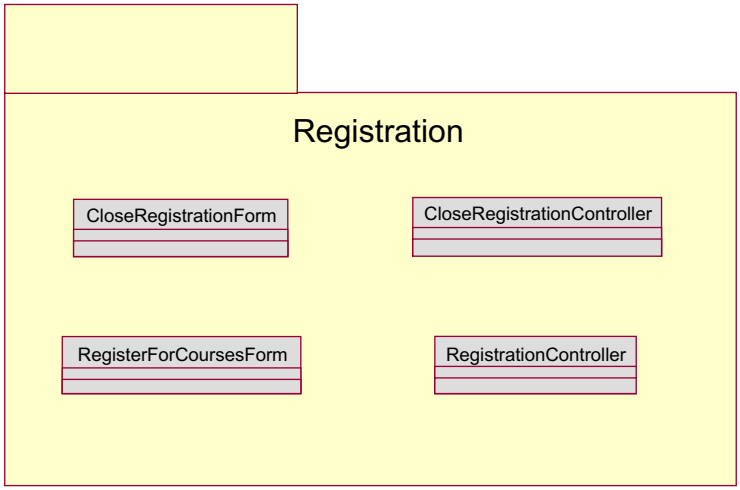


Gói (package)

- Một cơ chế chung để tổ chức các phần tử thành nhóm.
- Một phần tử trong mô hình có thể chứa các phần tử khác.



Ví dụ: Registration Package



33

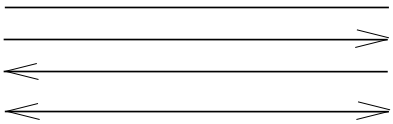
Nội dung

1. Phân tích thiết kế hướng đối tượng
2. Biểu đồ lớp
3. **Quan hệ giữa các lớp**
4. Ví dụ và bài tập

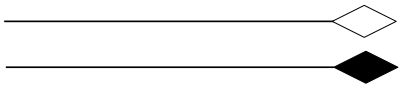
34

Class Relationships

- Association



  - Aggregation
    - Composition

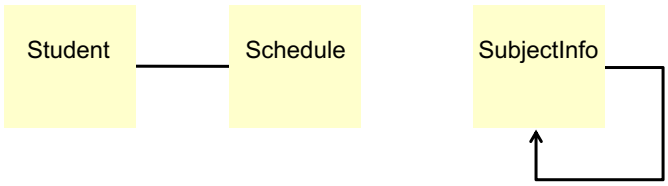


- Inheritance
- Dependency

35

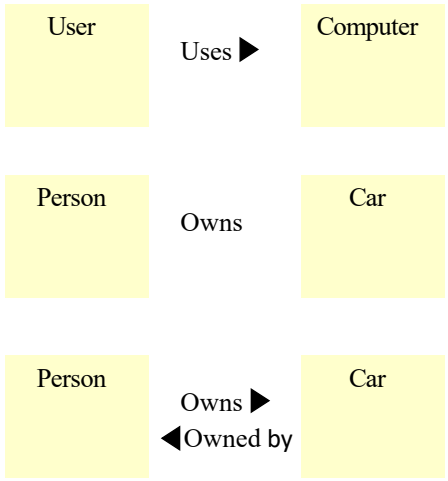
Liên kết (association) là gì?

- Mối liên hệ ngữ nghĩa giữa hai hay nhiều lớp chỉ ra sự liên kết giữa các thể hiện của chúng
- Mối quan hệ về mặt cấu trúc chỉ ra các đối tượng của lớp này có kết nối với các đối tượng của lớp khác.



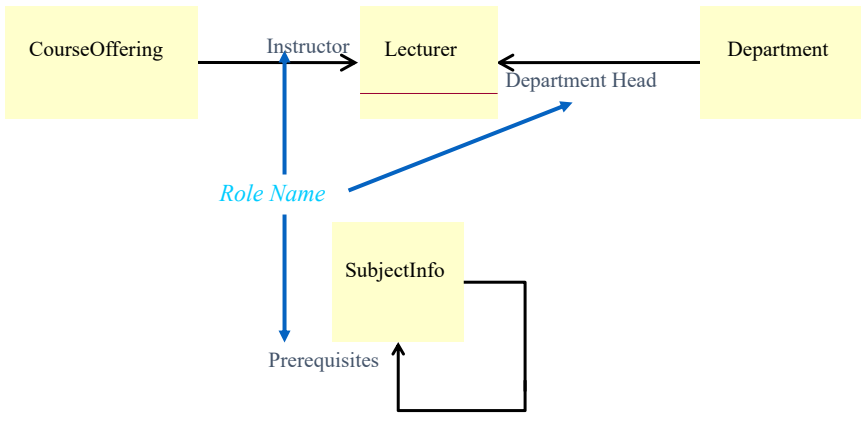
36

Tên và hướng của liên kết



37

Vai trò (role) trong liên kết



38

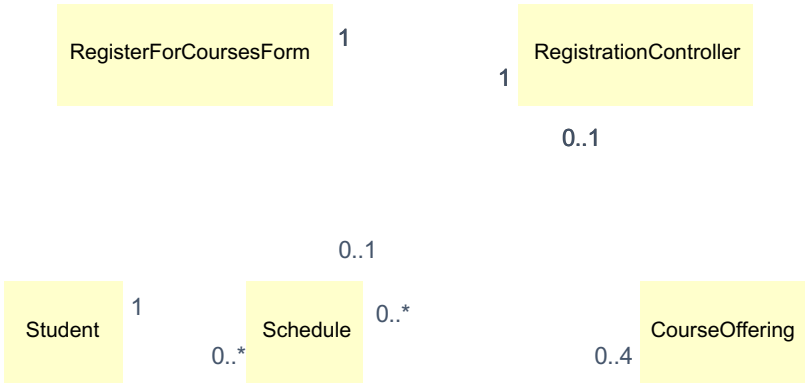
Bội số quan hệ (Multiplicity)

- Bội số quan hệ là số lượng thể hiện của một lớp liên quan tới MỘT thể hiện của lớp khác.
- Với mỗi liên kết, có hai bội số quan hệ cho hai đầu của liên kết.
  - Với mỗi đối tượng của Professor, có nhiều Course Offerings có thể được dạy.
  - Với mỗi đối tượng của Course Offering, có thể có 1 hoặc 0 Professor giảng dạy.

Biểu diễn bội số quan hệ

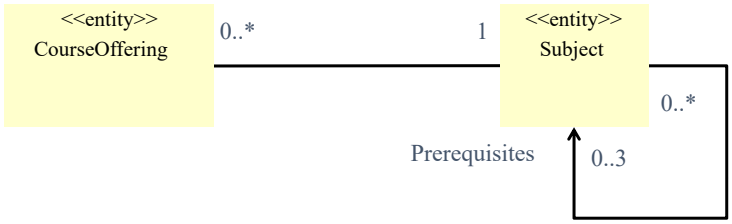
Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

Ví dụ về bội số quan hệ

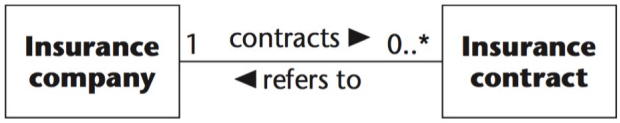


Ý nghĩa của bội số quan hệ

- Giúp trả lời 2 câu hỏi
  - Liên kết là bắt buộc hay tùy chọn?
  - Số lượng nhỏ nhất và lớn nhất các thể hiện của một lớp được liên kết với **một** thể hiện của lớp khác



Java implementation



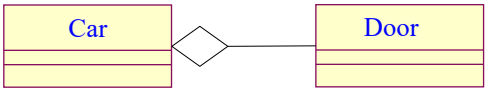

```
//InsuranceCompany.java file
public class InsuranceCompany
{
   // Many multiplicity can be implemented using Collection
    private List<InsuranceContract> contracts;

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

    /* Methods */
}
```


43

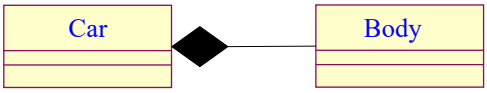
Các loại liên kết

- Association
  - use-a
  - Các đối tượng của một lớp liên kết với các đối tượng của lớp khác

- Aggregation
  - has-a/is-a-part
  - Liên kết mạnh-Strong association. Thể hiện của một lớp được tạo bởi (**made up**) các thể hiện của lớp khác



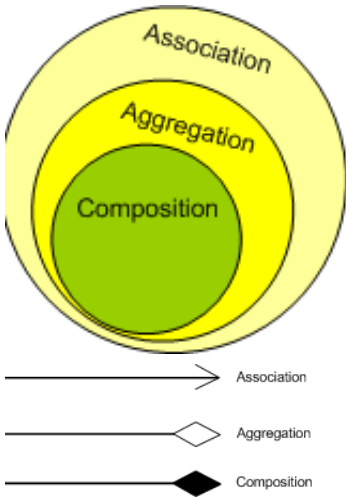
- Composition
  - Kết tập mạnh-Strong aggregation. Đối tượng bộ phận không thể được chia sẻ, và cùng bị hủy với đối tượng tổng thể
  - Share life-time



44

Association, Aggregation and Composition

- Liên kết (Association)
  - Sử dụng (use-a)
- Kết tập (Aggregation)
  - Strong association
  - has-a/is-a-part
- Hợp thành (Composition)
  - Strong aggregation
  - Share life-time



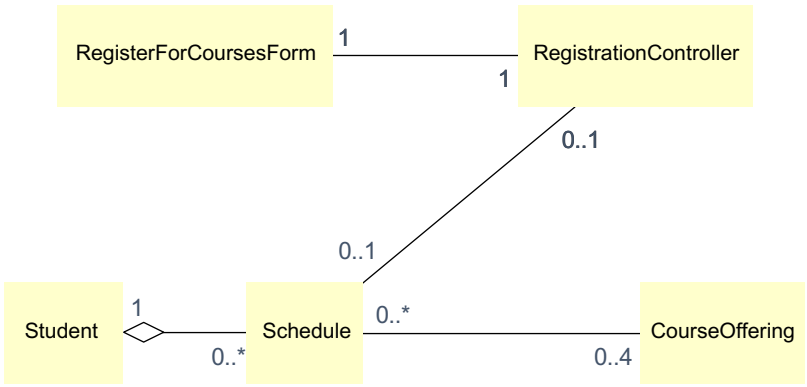
45

Kết tập (aggregation) là gì?

- Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.
  - Kết tập là mối quan hệ "là một phần" ("is a part-of").
- Bội số quan hệ được biểu diễn giống như các liên kết khác

46

Ví dụ về kết tập

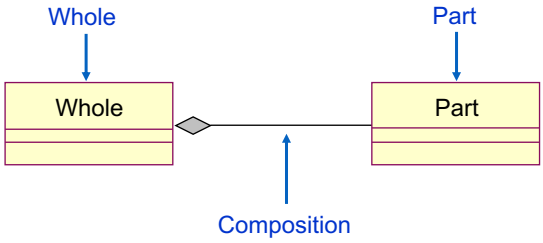


Aggregation – Java implementation

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

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

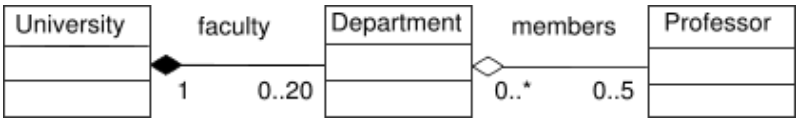
Hợp thành/Cấu thành (Composition) là gì?

- Một dạng của kết tập với quyền sở hữu mạnh và các vòng đời trùng khớp giữa hai lớp
  - Whole sở hữu Part, tạo và hủy Part.
  - Part bị bỏ đi khi Whole bị bỏ, Part không thể tồn tại nếu Whole không tồn tại.



Ví dụ – Aggregration vs. Composition

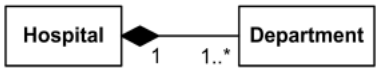
- University chứa (own) nhiều Department
- Mỗi Department có 1 số các Professor



- Nếu hủy University:
  - Các phòng ban cũng không còn tồn tại
  - Nhưng các Professor trong các Department vẫn còn tồn tại
- Dấu hiệu khác:
  - 1 professor có thể làm trong nhiều Department
  - 1 Department chỉ thuộc về 1 University

Ví dụ Composition

Folder could contain many files, while each File has exactly one Folder parent. If Folder is deleted, all contained Files are deleted as well.



Hospital has 1 or more Departments, and each Department belongs to exactly one Hospital. If Hospital is closed, so are all of its Departments.

Composition – Java implementation

```
final class Car {
   private final Engine engine;

   Car(EngineSpecs specs) {
      engine = new Engine(specs);
   }

   void move() {
      engine.work();
   }
}
```

Composition – Java implementation

```
class Person {
   private final Brain brain;

   Person(Brain humanBrain) {
      brain = humanBrain;
   }
}
```

```
Brain b = new Brain();
// or we have an instance of Brain in other scopes
// not exactly in this scope

Person p1 = new Person(b);
Person p2 = new Person(b);
```


53

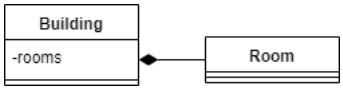
Composition – Java implementation

```
public class House {
   private final Room room;

   public House() {
      room = new Room();
   }
}
```

The child **cannot exist independent** of the parent

54

Composition – Java implementation




```
class Building {
   class Room {}
   private final List<Room> rooms;
   public Building() {
      rooms = new ArrayList<Room>();
      rooms.add(new Room());
      rooms.add(new Room());
   }
}
```

The objects' lifecycles are tied. If the whole is destroyed, the part will also be destroyed with it.

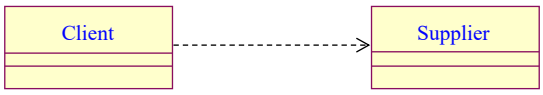
Note that doesn't mean that the whole can't exist without any of its parts. E.g., we can tear down all the walls inside a building, hence destroy the rooms. But the building will still exist

```
class Building {
   List<Room> rooms;

   interface Room {
      void doInRoom();
   }
   Room createAnonymousRoom() {
      return new Room() {
         @Override
         public void doInRoom() {}
      };
   }

   Room createInlineRoom() {
      class InlineRoom implements Room {
         @Override
         public void doInRoom() {}
      }
      return new InlineRoom();
   }

   Room createLambdaRoom() {
      return () -> {};
   }
   // …
}
```


55

Phụ thuộc - Dependency

- Là quan hệ giữa 2 đối tượng của 2 lớp

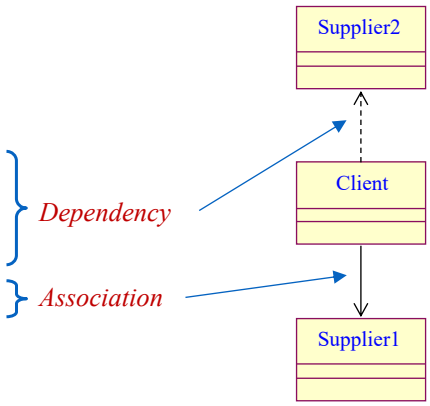


56

Dependencies vs. Associations

- Các đối tượng cần phải "biết nhau" để truyền thông điệp được cho nhau

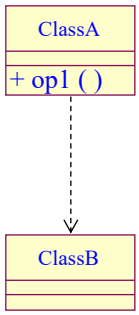
  - Local variable reference
  - Parameter reference
  - Global reference
  - Field reference



57

Local Variable Visibility

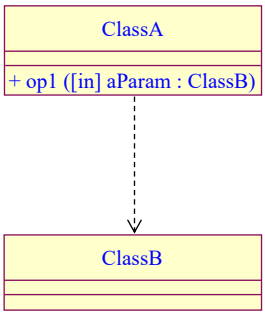
- Phương thức op1() tạo và sử dụng biến cục bộ tham chiếu tới đối tượng ClassB



58

Parameter Visibility

- Tham chiếu tới đối tượng lớp ClassB được truyền làm tham số trong phương thức op1 của lớp ClassA

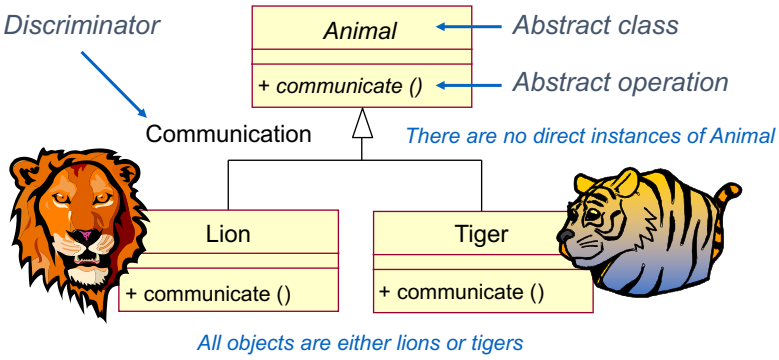


Tổng quát hóa (Generalization)

- Mối quan hệ giữa các lớp trong đó một lớp chia sẻ cấu trúc và/hoặc hành vi với một hoặc nhiều lớp khác
- Xác định sự phân cấp về mức độ trừu tượng hóa trong đó lớp con kế thừa từ một hoặc nhiều lớp cha
  - Đơn kế thừa (Single inheritance)
  - Đa kế thừa (Multiple inheritance)
- Là mối liên hệ "là một loại" ("is a kind of")

Lớp trừu tượng và lớp cụ thể (Abstract and Concrete Class)

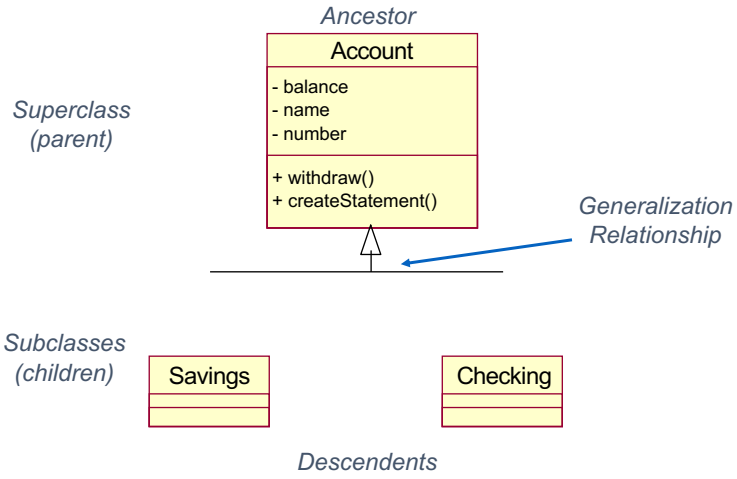
- Lớp trừu tượng không thể có đối tượng
  - Chứa phương thức trừu tượng
  - Chữ nghiêng
- Lớp cụ thể có thể có đối tượng



61

Ví dụ về Đơn kế thừa

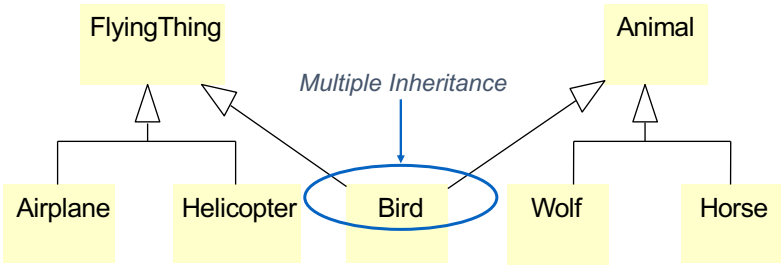
- Một lớp kế thừa từ MỘT lớp khác



62

Ví dụ về Đa kế thừa

- Một lớp có thể kế thừa từ nhiều lớp khác

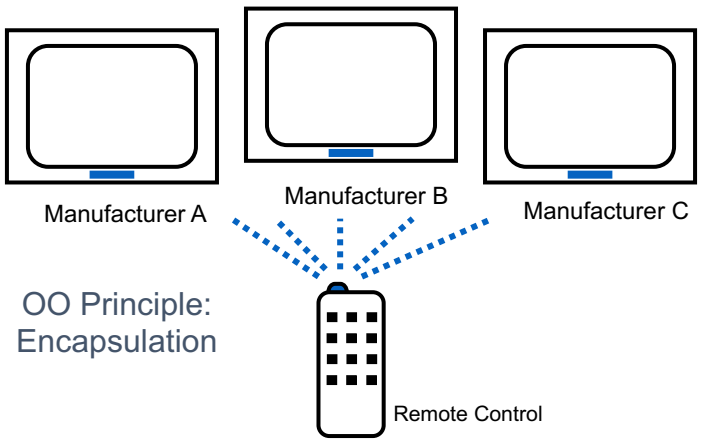


Sử dụng đa kế thừa chỉ khi cần thiết và luôn luôn phải cẩn thận!

63

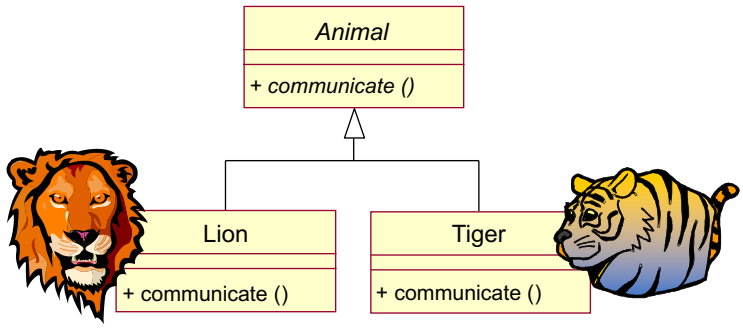
Đa hình (Polymorphism) là gì?

- Khả năng che giấu các thực thi khác nhau dưới một giao diện duy nhất.



64

Tổng quát hóa: Thực thi đa hình



Without Polymorphism

```
if animal = "Lion" then
      Lion communicate
else if animal = "Tiger" then
      Tiger communicate
end
```

With Polymorphism

```
Animal communicate
```

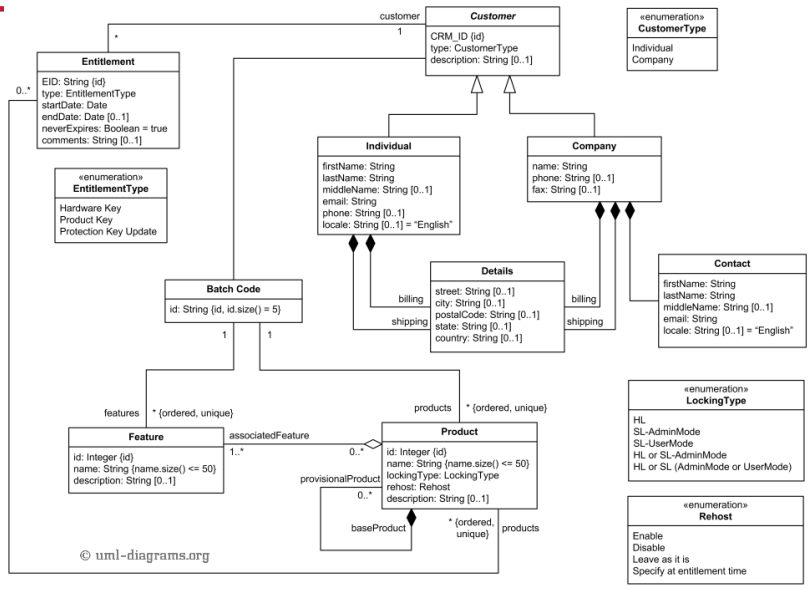

65

Nội dung

1. Phân tích thiết kế hướng đối tượng
2. Biểu đồ lớp
3. Quan hệ giữa các lớp
4. **Ví dụ và bài tập**

66

Ví dụ



https://www.uml-diagrams.org/software-licensing-domain-diagram-example.html

Bài tập 1

- **Xây dựng phần mềm quản lý đăng ký dạy và học ở trường đại học như sau:**
- Các sinh viên và giảng viên được lưu trữ thông tin vào trong phần mềm này với các nội dung về tên, tuổi, định danh cá nhân, mã số sinh viên hoặc mã cán bộ. Giảng viên còn có thông tin về cấp bậc (**level** với dạng số nguyên từ 1 trở đi); và mã số thuế (**tax**).
- Mỗi giảng viên có thể dạy nhiều lớp (**Course**) và có thể chủ nhiệm nhiều sinh viên. Mỗi sinh viên cũng đăng ký nhiều lớp học (**Course**), ứng với mỗi sinh viên có một bảng điểm (**Table**) và mỗi sinh viên được chủ nhiệm bởi một giảng viên.
- Bảng điểm của một sinh viên lưu trữ thông tin điểm của từng lớp học của sinh viên đăng ký.
- Hãy xây dựng biểu đồ lớp

Bài tập 2

- **Chương trình quản lý Thông tin trong một nhà ga được mô tả như sau:**
- Hoạt động chuyên chở trong nhà ga gồm nhiều đoàn tàu. Mỗi đoàn tàu có một số hiệu riêng, thông tin về ga đích đến của đoàn tàu và lịch trình chạy của đoàn tàu (giờ khởi hành và giờ dự kiến đến ga đích).
- Một đoàn tàu gồm nhiều toa tàu.Mỗi toa thuộc một trong hai loại toa chở khách hoặc toa chở hàng. Mỗi toa tàu có một số hiệu duy nhất và trọng lượng không tải tính bằng tấn (khi không chở khách hay hàng hoá). Mỗi toa chở khách còn có thông tin riêng về số lượng khách tối đa có thể chở.
- Khi tàu vào ga toa chở khách có thêm các hoạt động: thêm khách lên toa, bớt khách xuống toa.
- Thông tin về hành khách đi tàu gồm có họ tên, số chứng minh nhân dân, đoàn tàu và toa tàu mà họ mua vé. ga lên tàu và điểm xuống.
- Hãy xây dựng biểu đồ lớp

Bài tập 3

- **Một phần mềm Quản lý xe buýt tại bến xe được mô tả như sau:**
- Một xe buýt (**Bus)** chạy được tối đa 30 chuyến/ngày (**Trip**). Mỗi chuyến chứa tối đa 80 hành khách (**Person**). Hành khách được chia làm hai loại: hành khách mua vé theo từng lượt đi (**Customer**) và hành khách mua vé tháng (**Passenger**).
- *Tất cả các hành khách* đều được định danh bằng tên (**name**) và số chứng minh thư (**citizenCard**). Khách mua vé tháng có thêm thông tin mã vé ID.
- *Các xe buýt* có thông số về số lượng ghế ngồi (**numberOfSeats**) khác nhau. Trong lớp **Bus**, người ta xây dựng phương thức public **isEnableToLeaveStation(Trip t)**, trả về **true** nếu số hành khách trên chuyến xe buýt **t** bé hơn hoặc bằng 80% số ghế ngồi. Lớp **Bus** là lớp toàn thể, lớp **Trip** kết tập trong nó với tên vai trò là ***trips***.
- Các xe buýt có thông số để định danh, đây là một con số. Trong lớp **Trip**, người ta xây dựng phương thức public **availableSeats( )** trả về số lượng các ghế trống có trên chuyến xe.
- Người ta cài đặt trong lớp **Trip** phương thức mang tên **numberOf Passenger( )**, trả về số lượng các khách sử dụng vé tháng có trên chuyến xe.
- Hãy xây dựng biểu đồ lớp

71