Unit 4. Phân tích hệ thống bằng biểu đồ lớp

Teacher: Phạm Văn Hùng

Nội dung

1. Mô hình khái niệm - mô hình đối tượng
2. Xác định các lớp đối tượng
3. Các mối quan hệ giữa các lớp đối tượng
4. Biểu đồ lớp

1. Mô hình khái niệm – Mô hình đối tượng

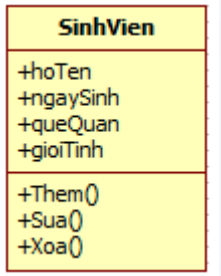
- **Đối tượng (Object):** Đối tượng là khái niệm cơ sở quan trọng nhất của cách tiếp cận hướng đối tượng, một đối tượng có thể tượng trưng cho 1 thực thể có thực trong thực tế (ví dụ một chiếc ô tô, hoặc một người cụ thể), hoặc có thể là 1 thực thể mang tính khái niệm (ví dụ một giao dịch trong ngân hang như rút tiền mặt, một đơn đặt hàng).
- **Lớp đối tượng (Class)**:Đối tượng là thể hiện, là một đại biểu của một lớp.

1. **Lớp** là một mô tả về một nhóm các đối tượng có những tính chất (thuộc tính) giống nhau, có chung các hành vi ứng xử (thao tác gần như nhau), có cùng mối liên quan với các đối tượng của các lớp khác và có chung ngữ nghĩa trong hệ thống.
2. **Lớp** chính là cơ chế được sử dụng để phân loại các đối tượng của một hệ thống. Lớp thường xuất hiện dưới dạng những danh từ chung trong các tài liệu mô tả bài toán hay trong các thảo luận với người sử dụng. Cũng như các đối tượng, lớp có thể là những nhóm thực thể có trong thế giới thực, cũng có những lớp là khái niệm trừu tượng và có những lớp được đưa vào trong thiết kế để phục vụ cho cài đặt hệ thống, v.v.

2. Xác định các lớp đối tượng

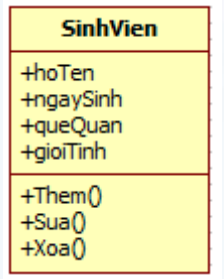
- Lớp và mối quan hệ của chúng có thể mô tả trong các biểu đồ lớp trong UML.
- Trong biểu đồ lớp, lớp được mô tả bằng một hình hộp chữ nhật, trong đó có tên của lớp, có thể có các thuộc tính và các hàm (phương thức)

1. Tên lớp được qui định viết bằng chữ không dấu và viết hoa chữ cái đầu tiên của các từ (Ví dụ: SinhVien, KhachHang)
2. Tên các thuộc tính (Attribute) thì cũng giống như tên lớp nhưng trừ chữ cái của từ đầu tiên không viết hoa (Ví dụ: hoTen, ngaySinh)
3. Tên các phương thức(Operation) (hàm) giống như tên thuộc tính nhưng có thêm ()



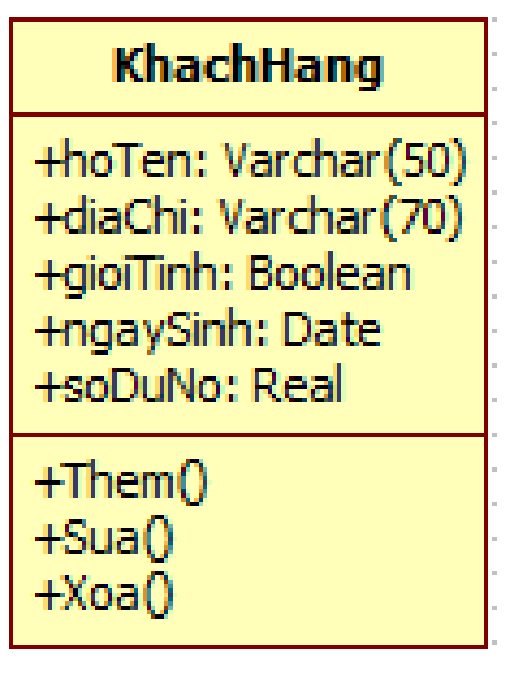
2. Xác định các lớp đối tượng

- Trong các phương thức (hàm) đứng trước tên phương thức là
  1. Dấu + hàm xác định tính công khai (public), mọi đối tượng trong hệ thống đều nhìn thấy được. Nghĩa là mọi đối tượng đều có thể truy nhập được vào dữ liệu công khai.
  2. Dấu # đứng trước tên thuộc tính, hàm xác định tính được bảo vệ (protected),chỉ những đối tượng có quan hệ kế thừa với nhau nhìn thấy được
  3. Dấu - đứng trước tên thuộc tính, hàm xác định tính sở hữu riêng (private), chỉ các đối tượng trong cùng lớp mới nhìn thấy được.



2. Xác định các lớp đối tượng

- Để khai báo mở rộng cho các thuộc tính (Attribute) có thể khai báo thêm kiểu dữ liệu:
  1. String: Chuỗi
  2. Char (n) hoặc Varchar: Ký tự
  3. Date: Ngày tháng
  4. Int: Kiểu số nguyen
  5. Real: Kiểu số thực
  6. Boolean: Kiểu Logic



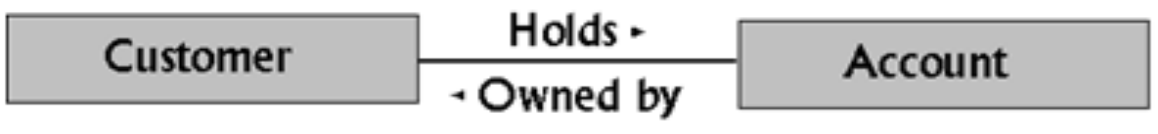
3. Các mối quan hệ giữa các lớp

- Trong UML có 4 mối quan hệ phổ biến
  a) Association
  b) Aggregation
  c) Composition
  d) Generalization

a. Association (quan hệ liên kết)

Khái niệm: Quan hệ Association (Liên kết hay liên hệ) được mô tả quan hệ giữa hai lớp với nhau, thể hiện chúng có liên quan với nhau, kết nối với nhau. Association thể hiện qua các quan hệ như "has: có", "Own: sở hữu" v.v...

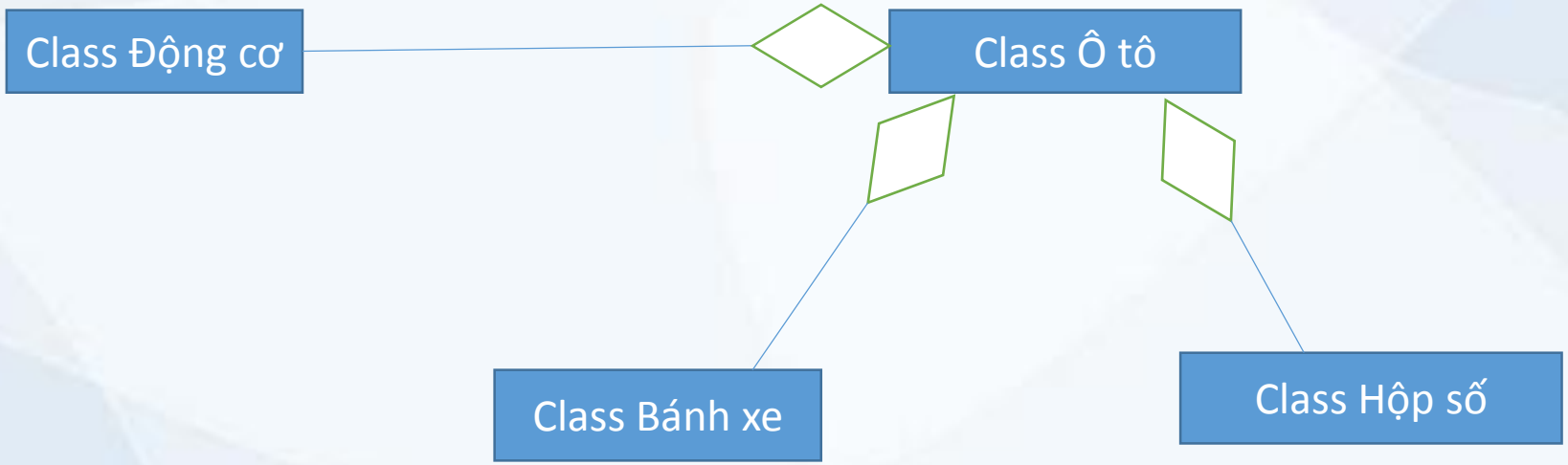
Chẳng hạn giữa 2 lớp Customer (Khách hàng) và lớp Account (Tài khoản)



- Giữa 2 object của 2 lớp có sự ghép cặp (thầy – trò (n-n) , khách hàng – hóa đơn (n-1)) . Tập hợp các kết nối cùng loại (cùng ý nghĩa) giữa các object của 2 lớp tạo thành mối liên kết association , quan hệ giữa 2 tập hợp (2 lớp)
- Là mỗi liên hệ giữa 2 lớp có role, role là tên vai trò của mối liên kết : vd như : của , cho , có , liên kết tới , trao đối với , ...
- Khi hủy một Object này không ảnh hưởng đến một Object khác

b. Aggregation (quan hệ kết tập hoặc thu nạp)

Khái niệm quan hệ kết tập: Kết tập (Aggregation) là một trường hợp đặc biệt của liên hệ (Association). Kết tập biểu thị rằng quan hệ giữa các lớp dựa trên nền tảng của nguyên tắc *"một tổng thể (whole) được tạo thành bởi các bộ phận (part)"*. Nó được sử dụng khi chúng ta muốn tạo nên một thực thể mới bằng cách tập hợp các thực thể tồn tại với nhau. Một ví dụ tiêu biểu của kết tập là chiếc xe ô tô gồm có bốn bánh xe, một động cơ, một khung gầm, một hộp số, v.v....

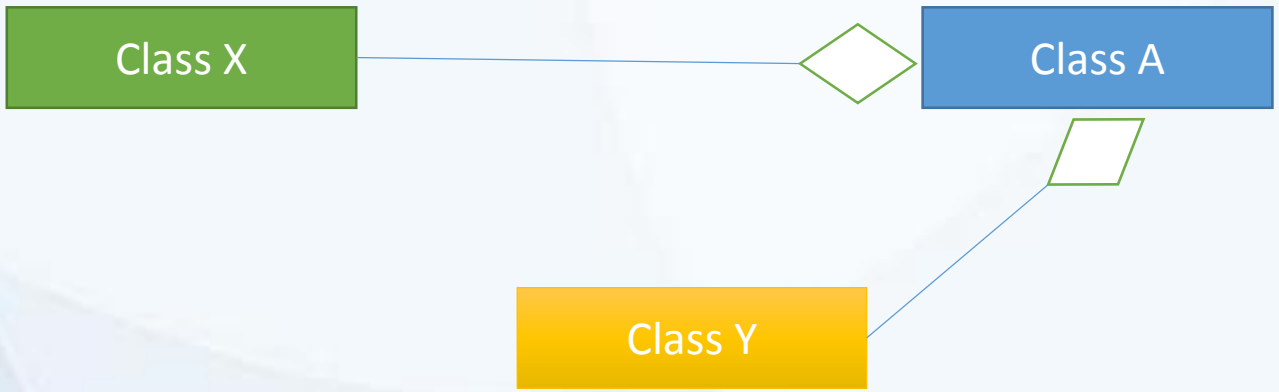


b. Aggregation (quan hệ kết tập hoặc thu nạp)

Ví dụ: Class A có thể được tạo bởi các Class X, Class Y, …; nghĩa là A có thể được tạo thành bởi nhiều X, Y, Z… kết hợp lại.

Lưu ý:

1. X, Y, Z có thể được tạo ra 1 cách độc lập, không cần phải bắt buộc tạo ra A
2. X, Y, Z có thể kết hợp để tạo ra một B (B khác A)
3. A có thể bị hủy nhưng X, Y, Z vẫn tồn tại



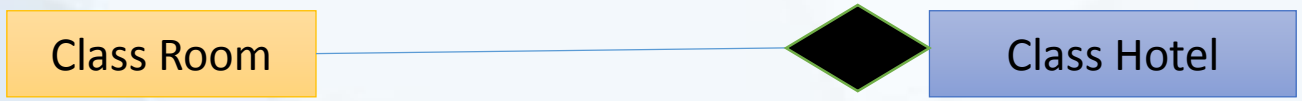
c. Composition (quan hệ hợp thành)

Composition tương tự như Aggregation nhưng khác là vòng đời của bên part (bộ phận) sẽ bị phụ thuộc và bên whole (Tổng thể)

Giả sử ta có 2 class là: Class Hotel và Class Room

- Trong class Hotel (Khách sạn) của 1 công ty kinh doanh dịch vụ khách sạn sẽ có nhiều thể hiện (instance) hay còn gọi là các đối tượng của lớp đó chính là các hotel , mỗi khách sạn sẽ có nhiều room (phòng)

- Khi hotel (whole) bị hủy thì room1, room2 (part) cũng bị hủy theo. Nói cách khác là vòng đời của bên part bị phụ thuộc vào vòng đời của bên whole trong part-whole relationship (quan hệ một phần trong toàn bộ)

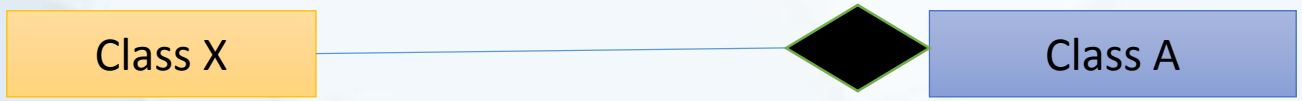


c. Composition (quan hệ hợp thành)

Ví dụ: Class A có thể được hợp thành từ Class X, Class Y, Class Z,…

Lưu ý:

1. X, Y, Z được tạo ra để kết hợp lại thành A
2. X, Y, Z không có thể kết hợp để tạo ra một B (B khác A), vì X, Y, Z hoàn toàn thuộc A
3. Khi A bị hủy thì X, Y, Z cũng bị hủy theo
4. Khi X, Y, Z bị hủy không ảnh hưởng đến A



d. Dependency (quan hệ phụ thuộc)

Quan hệ phụ thuộc nó mô tả Khi một class A phụ thuộc và một class B, những thay đổi ở B có thể ảnh hưởng đến A.

Chẳng hạn ta có:

- lớp Customer (Khách hàng) trong đó có thuộc tính IDcustomer (mã khách hàng) và 1 vài thuộc tính khác
- lớp Order (Đơn đặt hàng) nó có thuộc tính Idorder (mã đơn hàng) và cũng có thuộc tính Idcustomer (mã khách hàng)
- Nếu ban đầu bên lớp Customer ta thiết kế kiểu của Idcustomer là String và thiết lập mối quan hệ **Dependency** với lớp Order thì kiểu của Idcustomer là cũng là String, nếu ta đổi kiểu của Idcustomer bên lớp Customer thành Int thì Idcustomer bên lớp Order cũng bị đổi thành Int

d. Generalization (quan hệ khái quát/ thừa kế)

Generalization là quan hệ thừa kế được sử dụng rộng rãi trong lập trình hướng đối tượng, nó mô tả 1 lớp con (Child Class) được thừa kế các thuộc tính của 1 lớp cha (Parent Class)

Ví dụ: Ta có 1 lớp cha là lớp **Phong** và có thể có các lớp con là **PhongKT**, **PhongTC** các lớp con này thừa kế tất cả thuộc tính của lớp cha và nó có thể bổ sung them 1 số thuộc tính riêng