**Diagram UML**

Ngôn ngữ mô hình hóa thống nhất (tiếng Anh: Unified Modeling Language, viết tắt thành UML) là một ngôn ngữ mô hình gồm các ký hiệu đồ họa mà các phương pháp hướng đối tượng sử dụng để thiết kế các hệ thống thông tin một cách nhanh chóng.

1. **Class Diagram**
2. Định nghĩa

* Class diagram mô tả kiểu của các đối tượng trong hệ thống và các loại quan hệ khác nhau tồn tại giữa chúng.
* Miêu tả những attributes và behaviour một cách chi tiết
* Là một kỹ thuật mô hình hóa tồn tại ở tất cả các phương pháp phát triển hướng đối tượng.
* Biểu đồ hay dùng nhất trong UML và gần gũi nhất với các lập trình viên.
* Giúp các lập trình viên trao đổi với nhau và hiểu rõ ý tưởng của nhau.

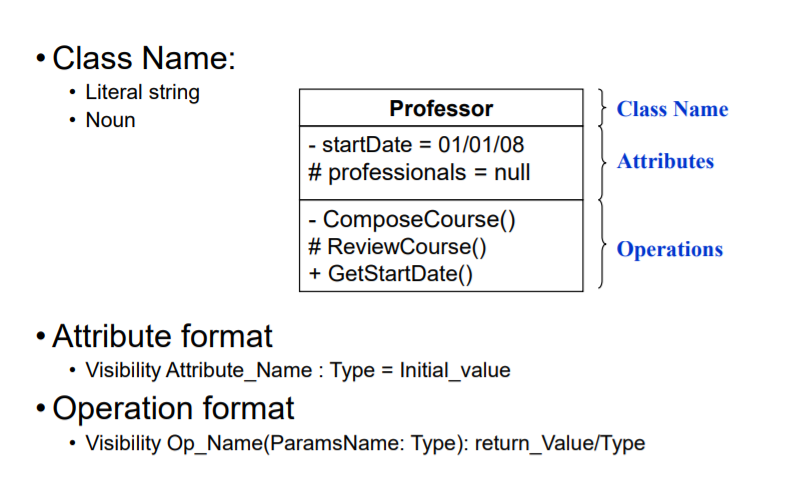
1. Các thành phần

Một Class diagram là một loại biểu đồ cấu trúc tĩnh mô tả cấu trúc của hệ thống bằng cách hiển thị:

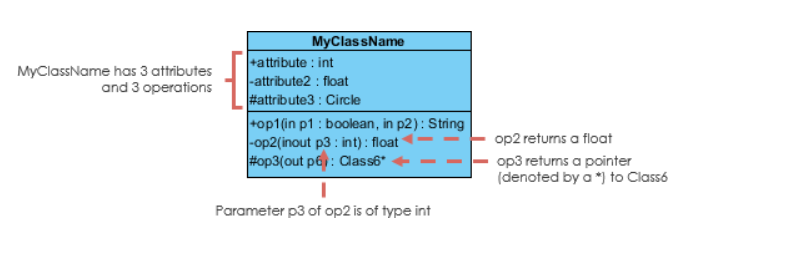
* + Các lớp
  + Các thuộc tính
  + Các thao tác (phương thức)
  + Các mối quan hệ giữa các đối tượng

1. Cú pháp đồ hoạ của biểu đồ lớp

Classes được biểu diễn bởi các hình chữ nhật có hiển thị class name và có lựa chọn hiển thị tên của các operations và attributes. Các ngăn được sử dụng để phân chia tên class, attributes và operations.

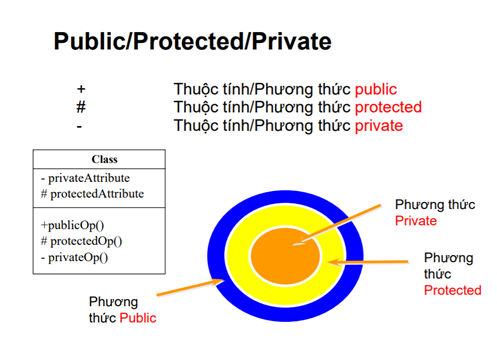


* 1. Tên lớp
  + Tên của lớp xuất hiện trong phân vùng đầu tiên.
  1. Thuộc tính
  + Các thuộc tính được hiển thị trong phân vùng thứ hai.
  + Loại thuộc tính được hiển thị sau dấu hai chấm.
  1. Phương thức
  + Các phương thức được hiển thị trong phân vùng thứ ba.
  + Kiểu trả về của một phương thức được hiển thị sau dấu hai chấm ở cuối chữ ký phương thức.
  + Kiểu trả về của các tham số phương thức được hiển thị sau dấu hai chấm theo sau tên tham số.



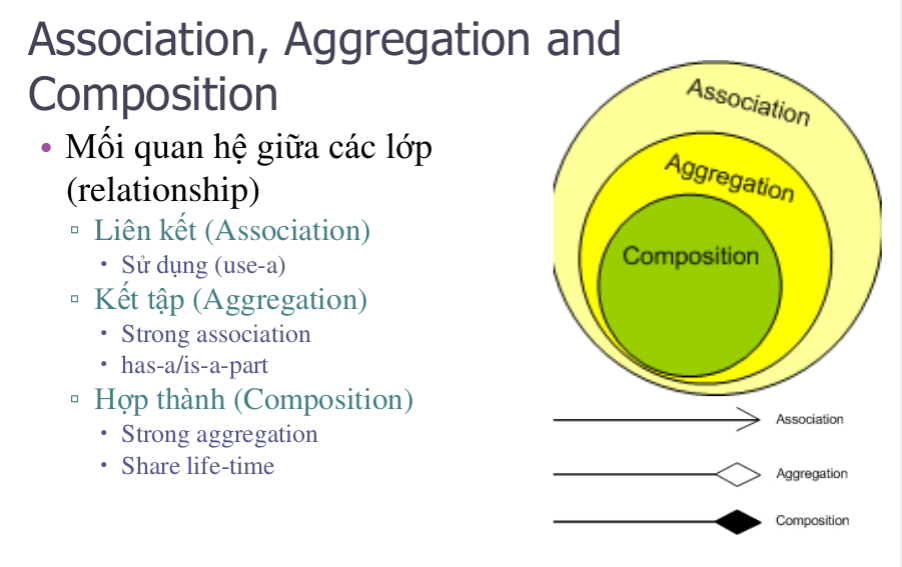
1. **Access Modifier**

* Sử dụng để đặc tả phạm vi truy cập cho các Attribute và Operation của 1 class (Cấp quyền cho các class khác sử dụng Attribute và Operation của class này).
* Sử dụng các kí hiệu



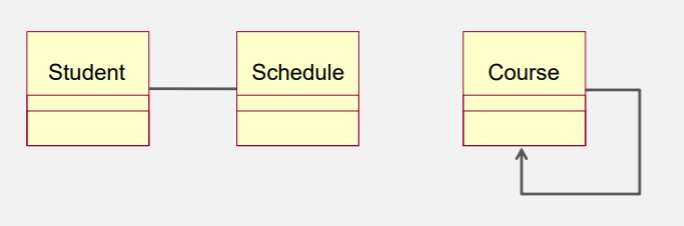
* + Private (–): Chỉ mình các đối tượng được tạo từ class này có thể sử dụng.
  + Public (+): Mọi đối tượng đều có thể sử dụng.
  + Protected (#): Chỉ các đối tượng được tạo từ class này và class kế thừa từ class này có thể sử dụng.

1. **Relationship trong Class Diagram**



1. Association (Liên kết)

* Liên kết là mối quan hệ giữa các lớp trong một Sơ đồ lớp UML. Chúng được biểu diễn bằng một đường liền nét giữa các lớp.
* Sự liên kết giữa 2 lớp khi mà không ai sở hữu ai. (liên kết ngang hàng)



1. Multiply trong Class Diagram (Bội số quan hệ)

* Là số lượng thể hiện của một lớp liên quan tới một thể hiện của lớp khác.

|  |  |
| --- | --- |
| **Biểu diễn** | **Ý nghĩa** |
|  | Unspecified |
| 1 | chính xác 1 |
| 0..\* | 0 hoặc nhiều |
| \* | 0 hoặc nhiều |
| 1..\* | 1 hoặc nhiều |
| 0..1 | 0 hoặc 1 |
| 2..4 | Khoảng xác định |
| 2, 4…6, 8 | Nhiều khoảng, |

* 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.



1. Agregation (Kết tập)

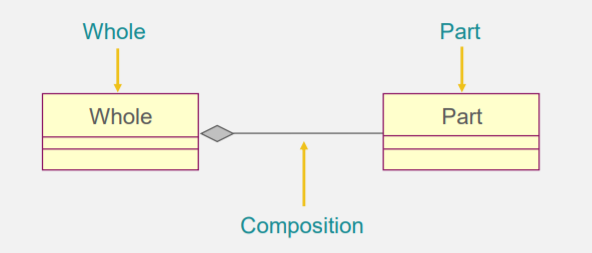
* Aggregation cũng giống như Association, nhưng khác là Aggregation có mối quan hệ sở hữu (ownership) giữa các instance
* Là một dạng đặc biệt của liên kết mô hình hóa mối quan hệ toàn thể-bộ phận (whole-part) giữa đối tượng toàn thể và các bộ phận của nó.
  + Kết tập là mối quan hệ “là một phần” (“is a part-of”).
* Biểu diễn:



* Ví dụ:
  + Điện thoại cần có pin. Pin là bên part, điện thoại là whole.
  + Vòng đời của part không phụ thuộc vào whole. Điện thoại cần có một cục pin (Pin) để hoạt động. Khi điện thoại hư có thể đem cục pin này sang điện thoại khác.

1. Compositon (Hợp thành)
   * + - ***Whole – part relationship:***
         * 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.

* Composition tương tự như Aggregation nhưng khác là vòng đời của part sẽ bị thụ thuộc vào whole
* Biểu diễn:



* Ví dụ: Khi hotel (whole) bị hủy thì room1, room2 (part) cũng bị hủy theo.