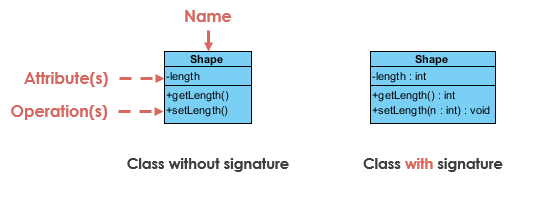
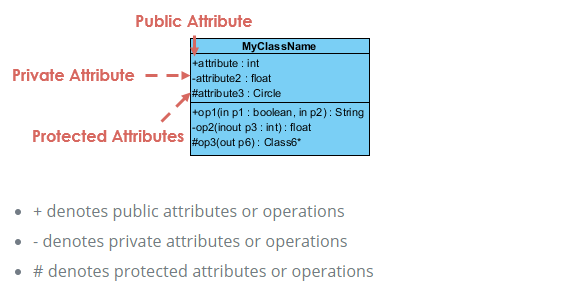
# UML Class Diagram

## UML Class Notation

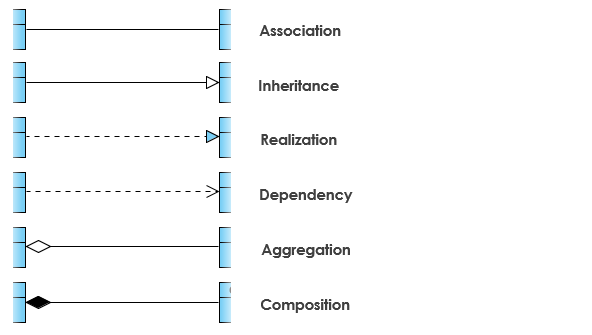
“A class” đại diện cho một nội dung cái mà gói “state (attribute)” và “behavior (operations)”. Mỗi “attribuite” có “a type”. Mỗi “operation” có một “signature”.





## Relationships between classes

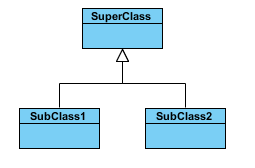
Một “class” có thể có một hoặc nhiều mối quan hệ với “classes” khác. Một mối quan hệ có thể là một trong những kiểu như sau:



### Inheritance (or Generalization)

“A generalization” là một mối quan hệ phân loại giữa một phân loại tổng quát hơn và một phân loại cụ thể hơn. Mỗi “instance” của “specific classifier” thì cũng gián tiến là “instance” của “general classifier”. Vì vậy, “the specific classifier” kế thừa các tính năng của “the more general classifier”.

* Thể hiện một “is-a” relationship
* Một abstract class name được thể hiện bởi chữ in nghiêng
* Subclass1 và Subclass2 là cụ thể hơn của SuperClass

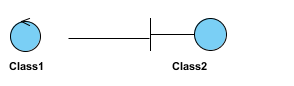


### Asociation

Là mối quan hệ giữa các class trong UML Class Diagram. Chúng được thể hiện bởi “a solid line” giữa các classes. Mối quan hệ thường được đặt tên bởi một động từ hoặc cụm động từ cái mà tham chiếu “the real world problem domain”.

### Simple Asociation

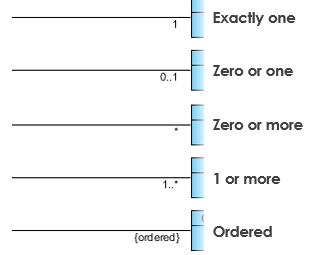
* Một kết cấu link giữa 2 class ngang bằng nhau
* Đây là một mối quan hệ giữa class 1 và class 2



### Cardinality

Cardinality được thể hiện dưới dạng:

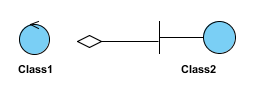
* One to one
* One to many
* Many to many



### Aggregation

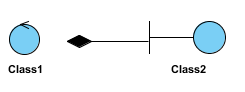
Là một kiểu đặc biệt của Association

* Nó thể hiện một “part of” relationship
* Class2 là một phần của Class1
* Nhiều instances của Class2 có thể có mối quan hệ với Class1
* Objects của Class1 và Class2 có lifetimes riêng biệt



### Composition

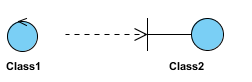
* Là một kiểu đặc biệt của Aggreation ở chỗ mà các phần bị phá hủy thì toàn bộ bị phá hủy
* Objects của Class2 sống và chết với Class1
* Class2 không thể đứng bởi chính bản thân nó.

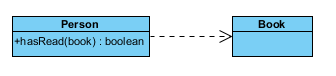


### Dependency

Một object của một class phải sử dụng một object của một class khác trong code hoặc method. Nếu object không được chứa bởi bất cứ trường nào thì nó chính là dependency relationship.

* Một kiểu đặc biệt của association
* Tồn tại giữa 2 classes nếu các thay đổi tới định nghĩa của một thằng sẽ là nguyên nhân thay đổi tới thằng khác
* Class1 phụ thuộc Class2





### Realization

Là mối quan hệ giữa “the blueprint class” và class thực thi chi tiết của nó. Nói một cách khác ta có thể hiểu là mối quan hệ giữa interface và implementing class

