LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Object Oriented Programming

Lập trình hướng đối tượng là gì?

- Là phương pháp thiết kế và phát triển phần mềm dựa trên kiến trúc lớp (class) và đối tượng (object)
- Đưa các đối tượng trong thế giới thực thành các đối tượng trong chương trình. Lấy đối tượng làm trung tâm
- Tập trung vào dữ liệu thay cho các hàm
- Chương trình được chia thành các đối tượng độc lập.
- Cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng.
- Dữ liệu được che giấu, bao bọc.
- Các đối tượng trao đổi với nhau thông qua các hàm

Ưu Điểm Lập trình hướng đối tượng là gì?

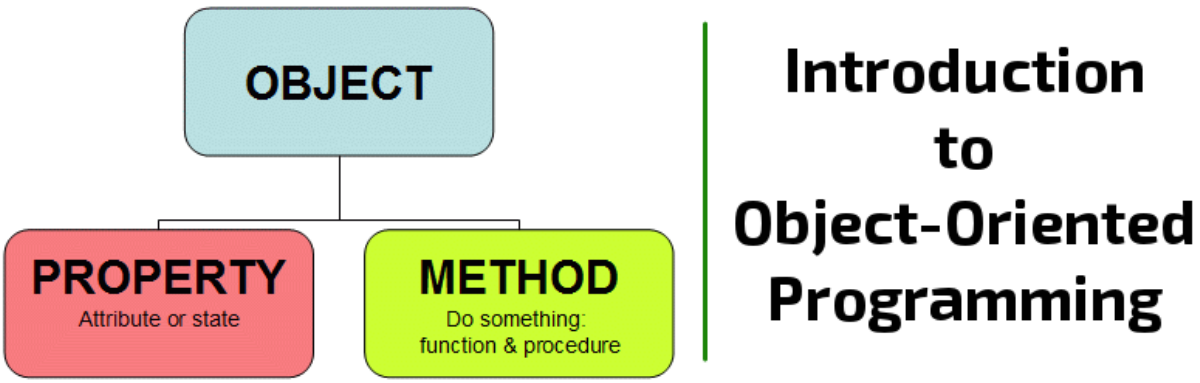
- Loại bỏ được những dư thừa, trùng lặp trong việc xây dựng ứng dụng.
- Cài đặt đối tượng giúp xúc tiến việc sử dụng lại, trao đổi giữa các đối tượng với nhau do đó sẽ giảm kích thước, thời gian xử lý,… thời gian phát triển hệ thống, tăng năng xuất lao động.
- Dễ bảo trì, nâng cấp, giảm lỗi.

Để hiểu rõ hơn ta cùng nhau phân tích chi tiết hơn ở những phần tiếp theo

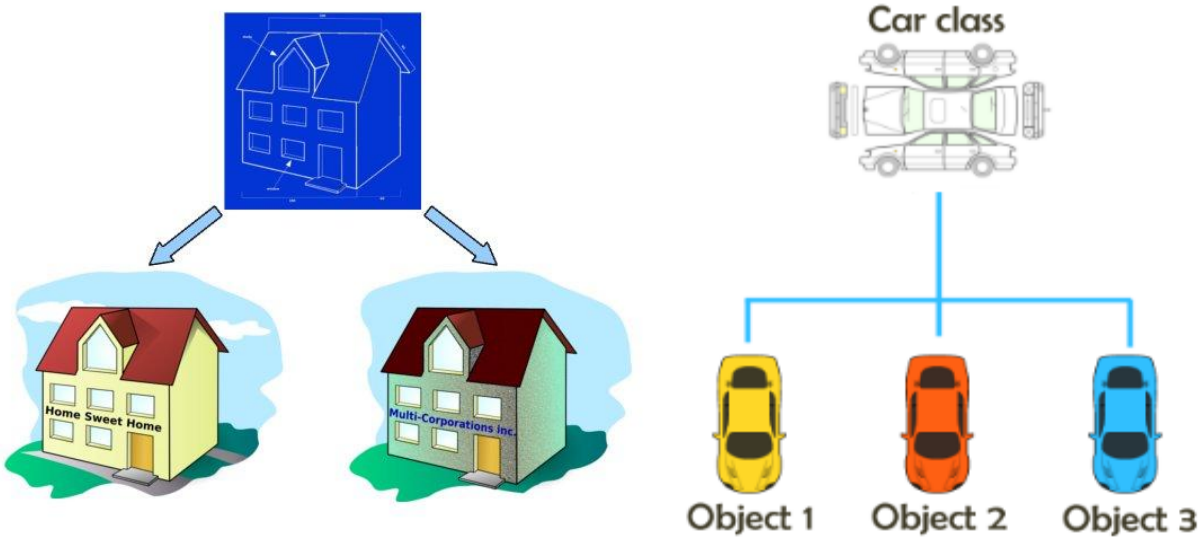
Đối tượng là gì?

Theo cách hiểu thông thường thì đối tượng có thể là người, là vật, là hiện tượng hay là bất cứ cái gì mà ta nhắm vào trong suy nghĩ trong hành động nghĩa là ta thấy được, sờ mó được… Như vậy, đối tượng là những thực thể tồn tại trong thế giới thực.
Khi đi sâu vào quan sát và phân tích thì một đối tượng trong thế giới thực chúng ta như con chó, con mèo, mùa xuân, mùa thu, mặt trời, mặt trăng, sông suối biển cả… đều có những tính chất và hành động riêng của nó.

Vậy nhìn nhận đối tượng trong phần mềm thì như thế nào?
Câu trả lời cũng không khác gì, chúng đều có những điểm tương đồng. Đối tượng trong phần mềm cũng có những tính chất và hành động hay gọi hoa mỹ hơn là các Thuộc tính (Properties) và Phương thức (Methods)



Class trong hướng đối tượng là gì?

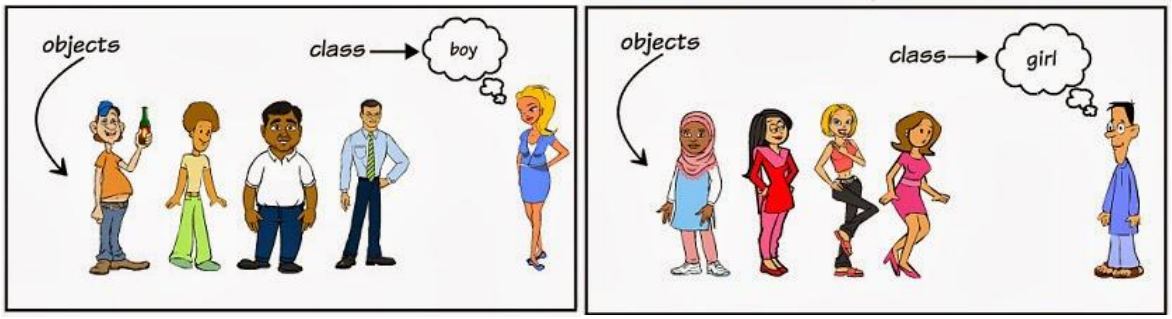


Nhìn vào 2 hình ví dụ trên ta thấy, một class cũng giống như một bản vẽ của một ngôi nhà hay của một chiếc xe.

Nếu đối tượng cụ thể và rõ ràng đến đâu thì lớp lại trừu tượng và chung chung đến đó, nó định nghĩa hình dạng ngôi nhà, chiếc xe trên giấy tờ, trên bản vẽ, những mối quan hệ giữa vị trí này đến vị trí kia, thiết bị này lắp ráp với thiết bị kia… được mô tả, xác định rõ và lên kế hoạch cụ thể, mặc dù ngôi nhà hay chiếc xe này chưa thực sự tồn tại.

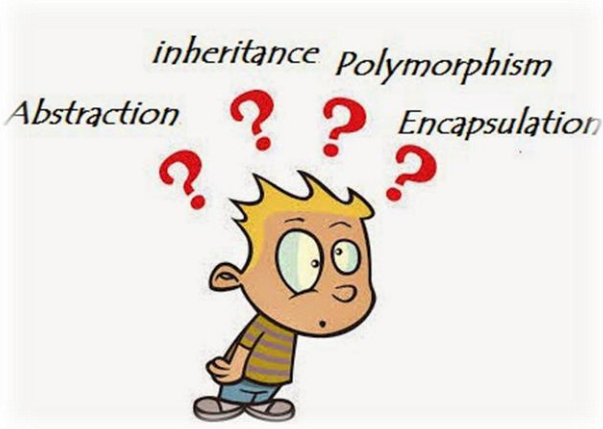
Một class hay một bản vẽ có được sử dụng chung để xây 1 hoặc nhiều ngôi nhà khác nhau, chúng đều có hình dạng, kích cỡ, vật liệu, chức năng… giống nhau như trong bản vẽ, tuy nhiên nội thất bên trong, cách trang trí bày bố thì lại là do từng hộ gia đình thiết kế.

Từ đó ta kết luận Class trong hướng đối tượng là một mô hình nhóm lại các đối tượng cùng loại



Các đặc tính của hướng đối tượng là gì?

Trong lập trình hướng đối tượng có bốn tính chất cơ bản bao gồm tính trừu tượng (*abstraction*), tính đóng gói (*encapsulation*), tính đa hình (*polymorphism*) và tính kế thừa (*inheritance*). Bốn tính chất này có thể hiểu cơ bản theo cách sau:



Đầu tiên ta tìm hiểu tính kế thừa (*inheritance*)

Chắc hẳn bạn đã từng nghe câu: “Cha nào con đấy” hay “Con nhà tông không giống lông cũng giống cánh” trong tiếng anh là “Like father, like son”. Nếu bạn hiểu được câu tục ngữ trên là coi như bạn đã hiểu được kế thừa trong hướng đối tượng.

Có lẽ đây là tính chất dễ hiểu và ngắn gon nhất trong hướng đối tượng. Thừa kế ở đây không chỉ hiểu hẹp ở mức thừa kế lại tài sản mà phải hiểu rằng đối tượng con đang kế thừa tính chất, tính cách của đối tượng cha. Đối tượng con có thể thừa kế một phần hoặc thừa kế hoàn toàn đối tượng cha cũng như có những thuộc tính và phương thức riêng của mình.

Vậy tính kế thừa mô phỏng lại việc con kế thừa một vài hoặc hoàn toàn các đặc điểm của cha trong khi vẫn có những đặc điểm riêng trong thế giới thực.

```java
package class_oopinjava;

public class Employee {

    float salary = 10000;
    String companyName ="OOP in Java";
}

```

```java
package class_oopinjava;

public class Programmer extends Employee {

    float bonus = 500;

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Programmer programmer = new Programmer();
        System.out.println("Programmer Company is: " +programmer.companyName);
        System.out.println("Programmer salary is: " +programmer.salary);
        System.out.println("Programmer bonus is: "+programmer.bonus);
    }

}

```

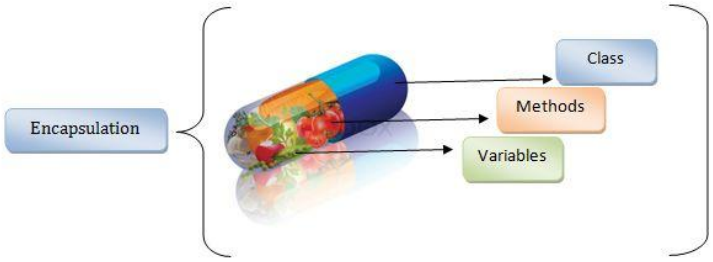
Kết luận:

- Cho phép xây dựng một lớp mới dựa trên các định nghĩa của một lớp đã có.

- Lớp đã có gọi là lớp Cha, lớp mới phát sinh gọi là lớp Con

- Lớp con kế thừa tất cả các thành phần của lớp Cha, có thể mở rộng các thành phần kế thừa và bổ sung thêm các thành phần mới.

Tính đóng gói là gì?



Tính đóng gói được thể hiện qua việc mỗi một đối tượng có một phạm vi tương tác cụ thể. Nếu trong C là khái niệm về namespace thì trong Java là khái niệm về package.

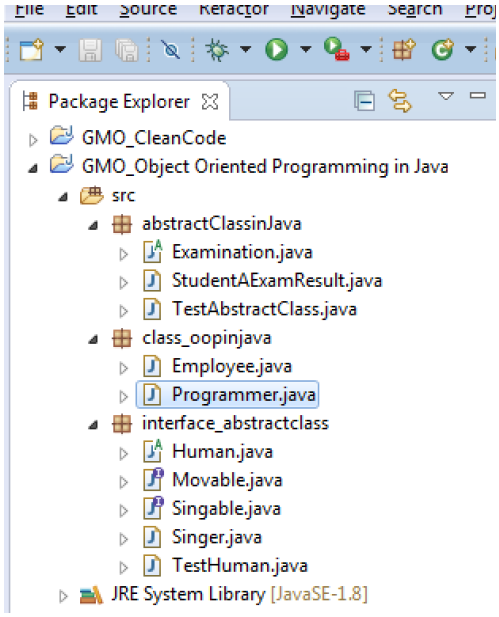
Đóng gói chính là sử dụng các Access Modifiers để che dấu và bảo vệ thông tin cho các trường dữ liệu hoặc phương thức (*xem phần access modifiers*). Việc sử dụng thuần thục nó là rất

quan trọng, vì nếu các thông tin về dữ liệu không được bảo vệ cẩn thận thì sẽ để lại lỗ hổng rất đáng lo ngại.

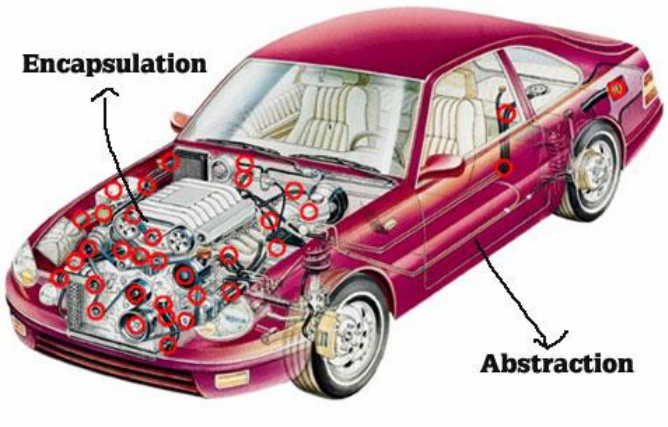
Mục đích chính của đóng gói là đảm bảo quyền truy cập trừ bên ngoài vào các thành phần bên trong của đối tượng. Ví dụ như như cái laptop bạn đang dùng đọc bài chia sẻ này, nếu nó không được bao bọc đóng gói vào một lớp vỏ nhôm vỏ nhựa để bảo vệ thì hôm nay bạn vui tay tháo đi một thanh Ram, mai buồn buồn tháo ổ cứng ra thì đảm bảo là đó không còn là một cái laptop bình thường được nữa, không những thế có thể bạn sẽ bị tê liệt vì điện giật.

Hoặc bạn tưởng tượng xem nếu như ra đường bạn thấy một cô gái xinh đẹp, và bạn không phải là bạn trai hay người thân thích gì với cô ấy, mà bạn đến bẹo má bá vai, thì bạn biết điều gì sẽ xảy ra với bạn rồi đúng không…

Như vậy, nếu không nằm trong một nhóm quan hệ nhất định bạn sẽ bị giới hạn những quyền nhất định.



Tính trừu tượng trong hướng đối tượng là gì?



Quan sát chiếc xe bạn sẽ hiểu được tính trừu tượng, cho dù bên trong chiếc xe đó là hàng tá những linh kiện được nối ráp với nhau một cách loàng ngoằng khó hiểu nhưng ta không qua tâm tới điều đó, không quan tâm nó được tạo ra cách nào hay nó hoạt động như thế nào, ta quan tâm nó màu gì, loại xe gì, đời nào, bao nhiêu chỗ, các tính năng gì…

Tác dụng lớn nhất của tính chất này là khi tương tác với một đối tượng bất kì bạn phải thiết kế sao cho chỉ quan tâm tới đầu vào (*input*) và đầu ra (*output*) thay vì phải tập trung tới xử lí (*process*) bên trong của đối tượng. Điều này giúp cho việc sử dụng lại phần code, tiện ích của các lập trình viên khác được dễ dàng hơn, giúp tiết kiệm công sức hơn. Đồng thời việc phân chia công việc cho nhiều người làm cũng dễ dàng hơn. Vậy tính trừu tượng đang mô phỏng lại cách mà các đối tượng trong thế giới thực tương tác với nhau.

Vậy tính chất thứ 4 trong hướng đối tượng là gì?

Đa hình nói một cách dễ hiểu hơn thì tính đa hình mô phỏng lại việc mỗi đối tượng ở trong những hoàn cảnh khác nhau lại có những hình thái khác nhau.

Ví dụ như một lập trình viên khi ở công ty thì mang những đặc điểm, hoạt động của một lập trình viên nhưng khi về nhà, tương tác với đối tượng ba mẹ của đối tượng lập trình viên thì lúc này đối tượng phải mang đặc điểm, hoạt động của một người con. Để mô phỏng lại việc này thì đối tượng lập trình viên sẽ cung cấp ra một danh sách các thuộc tính cũng như phương thức được xác định trước và trong mỗi một hoàn cảnh, ở một cương vị khác nhau thì lập trình viên lúc đó sẽ có những thể hiện khác nhau.

Nói nôm na đơn giản hơn là bạn có 3 lớp A,B,C kế thừa nhau và trong 3 lớp này có ba method cùng tên là show(); . Khi ta tạo mới đối tượng cho các lớp trên thì những đối tượng này gọi tới method show() nằm ở lớp nào thì thực thi chức năng phương thức ở lớp đó. Bởi vì chúng kế thừa nên sử dụng rõ tính đa hình này.

Trong ví dụ trên, chú bò sữa, Chú khỉ già, hay Bác voi bản đôn thì cũng có hành động là CallSpeak() xong dù gọi chung phương thức nhưng kết quả thì hoàn toàn khác nhau, con nào kêu theo tiếng con đó, không lẫn lộn vào nhau.

Tìm hiểu Abstract class và Interface

Trước khi hiểu về lớp trừu tượng, hãy hiểu về khái niệm trừu tượng trong Java trước *(xem lại tính trừu tượng trong java)*

Trừu tượng là việc hiện thị các chức năng quan trọng cho người dùng và ẩn đi chi tiết nội bộ thực hiện bên trong. Ví dụ như việc gửi tin nhắn cho người yêu hằng ngày, ta chỉ gõ tin nhắn và nhấn nút gửi, đợi vài giây sau là người ấy nhận được, mà chẳng ai quan tâm hay biết cách xử lý hay giao thức gửi đi bên trong.

Trừu tượng cho phép bạn xem cái đối tượng làm, thay vì ngồi xem nó làm như thế nào.

Abstract class hiểu đơn giản là một lớp trừu tượng, tự bản thân cái tên đã nói lên tất cả.

Vì là trừu tượng nên ở trong class đó, các phương thức chỉ được khai báo ở dạng khuôn mẫu (template), chung chung mà không được viết chi tiết. Việc cài đặt chi tiết các phương thức chỉ được thực hiện ở các lớp dẫn xuất (lớp con) kế thừa lớp

trừu tượng đó. Giống như việc muốn biết sms của ta được gửi đi như thế nào thì phải gặp mấy tên lập trình, hỏi nó cặn cẽ cách thức mã hóa, truyền tin, nhận tin, giải mã…

Như vậy, mỗi lớp con chỉ được mở rộng (extends) từ một lớp trừu tượng và nó không cho phép tạo instance (thể hiện) thuộc lớp đó, nghĩa là không khởi tạo bằng toán tử **new** như các lớp thông thường.

Interface được hiểu như là một bản hợp đồng. Các lớp hiện thực (Implement) interface này phải tuân theo bản hợp đồng. Điều này có nghĩa là bạn phải thực thi đầy đủ các phương thức, property, event, delegate, indexer đã được khai báo trong bản hợp đồng – Interface đó.

Nói cách khác, **interface** là tập các phương thức hoạt động được khai báo sẵn mà không cài đặt phần thân. Các lớp dẫn xuất có thể thực thi từ nhiều Interface và có nhiệm vụ bổ sung đầy đủ cách thức hoạt động của mình.

Vậy điểm chung và riêng của Interface và Abstract Class là gì?

Có thể hiểu Interface và Abstract class theo các hướng khác nhau nhưng mục đích cuối cùng thực chất chính là tránh được tình trạng code bị dư thừa, lặp đi lặp lại, hoặc thiếu xót các phương thức khi thực hiện trong một dự án lớn. Không cho phép tạo instance, đều hỗ trợ tính đa hình

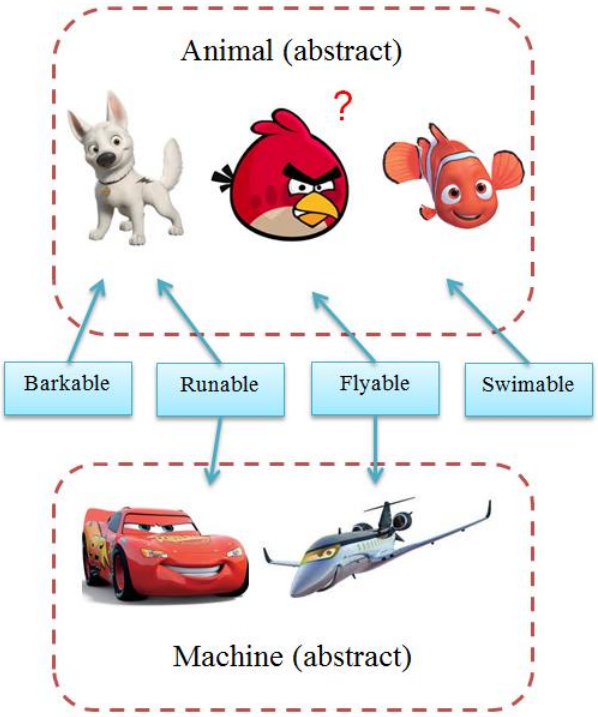
Abstract class	Interface
Lớp trừu tượng có thể có phương thức trừu tượng hoặc phương thức không trừu tượng	Phương thức trong Interface là phương thức trừu tượng
Có thể thực hiện phần thân của phương thức trong đó	Không cho phép thực hiện phần thân của phương thức
Được khai báo biến thành viên	Không được khai báo biến thành viên
Không ràng buộc, các method, biến thành viên có thể là public/ private/ protected, hoặc abstract	Tất cả phương thức đều là public
Sử dụng từ khóa "extends" để thừa kế	Sử dụng từ khóa "implements" để thừa kế
Có thể thừa kế từ 1 lớp abstract khác	Không thể thừa kế từ 1 interface khác
Có thể thừa kế từ 1 interface	Không thể thừa kế từ 1 lớp abtract

Không cho phép đa thừa kế.	Cho phép đa thừa kế
Có phương thức khởi tạo	Không có phương thức khởi tạo

Ví dụ:
- Abstract class ConVat có các lớp con Chim, Ca.
- Abstract class MayMoc có các lớp con MayBay, Thuyen
- Interface: iBay, iBoi, iChay.

=> MayBay, Chim sẽ có cùng Interface là iBay. Rõ ràng mặc dù MayBay, Chim có cùng cách thức hoạt động là bay nhưng chúng khác nhau về bản chất.

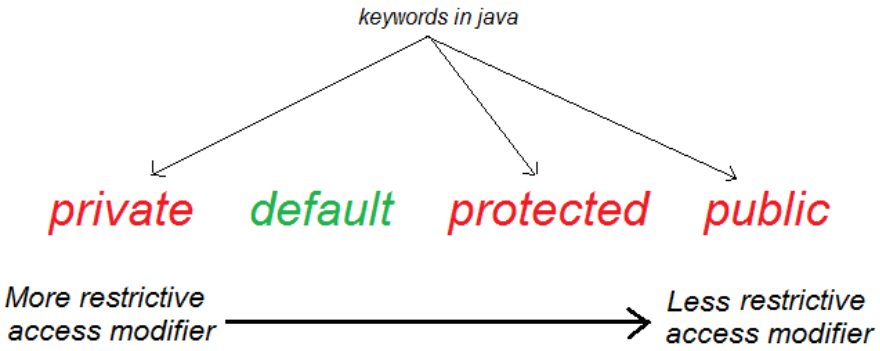
=> MayBay cũng có interface là iChay nhưng Chim không thể nào kế thừa thêm abstract class MayMoc



Access modifier

Có hai loại modifier trong java: access modifiers và non-access modifiers. Các access modifiers trong java xác định độ truy cập (Phạm vi) vào dữ liệu của của các trường, phương thức, cấu tử hoặc class.

Có 4 kiểu của java access modifiers:



Và có một vài non-access modifiers chẳng hạn static, abstract, synchronized, native, volatile, transient, v.v.. Trong tài liệu này chúng ta sẽ học về access modifier.

Visibility	Public	Protected	Default	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From a subclass in the same package	Yes	Yes *(Package, Inheritance)*	Yes *(Package)*	No
From a subclass outside the same package	Yes	Yes *(Inheritance)*	No	No
From any non-subclass class outside the package	Yes	No	No	No

Private access modifier chỉ cho phép truy cập trong nội bộ một class. Bạn không thể truy cập vào trường private ở bên ngoài class. Java sẽ thông báo lỗi tại thời điểm biên dịch class.

Phạm vi truy cập của access modifier mặc định là trong nội bộ package. Trong cùng một package bạn có thể truy cập vào các thành viên có access modifier mặc định. Và không được phép truy cập bên ngoài package, kể cả trong class con.

Protected access modifier có thể truy cập bên trong package, hoặc bên ngoài package nhưng phải thông qua tính kế thừa. Protected access modifier chỉ áp dụng cho field, method và constructor. Nó không thể áp dụng cho class (class, interface, enum, annotation).

Public access modifier là mạnh mẽ nhất và có thể truy cập ở mọi nơi. Nó có phạm vi truy cập rộng nhất so với các modifier khác.