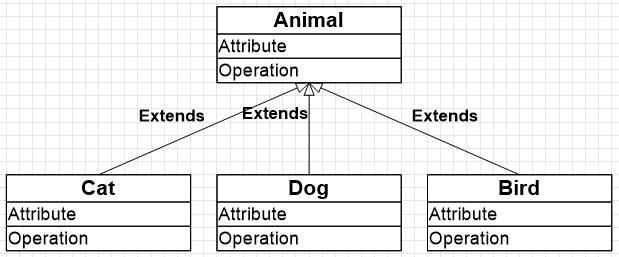
**Các khái niệm**

* Đây là đặc trưng thứ hai của ngôn ngữ lập trình hướng đối tượng Java(OOP).
* Kế thừa(inheritance) là một kĩ thuật trong Java cho phép một lớp kế thừa lại những tính năng(trường, phương thức) của một lớp khác.
* Ta còn gọi tính chất kế thừa là mở rộng từ một lớp(extend).
* Liên quan đến khái niệm kế thừa, ta có các khái niệm sau:
  + Lớp cha(super class, base class): là các lớp được lớp khác kế thừa.
  + Lớp con(sub class, derived class): là các lớp đi kế thừa lớp khác. Lớp con có thể bổ sung thêm các trường và các phương thức của riêng nó.
  + Tính tái sử dụng(reusability): kế thừa hỗ trợ tái sử dụng lại code. Thay vì tạo một lớp từ đầu với tất cả các trường, phương thức. Ta kế thừa từ một lớp đã có nào đó có chứa các tính năng ta cần. Do đó tái sử dụng lại các trường, phương thức của lớp đã có và chỉ cần bổ sung thêm các thông tin riêng.
  + Tính tổng quát hóa: lớp cha càng ở trên cao thì càng tổng quát, chứa càng ít thông tin nhưng là những thông tin chung nhất có ở tất cả các lớp con của nó.
  + Tính cụ thể hóa: lớp con càng ở phía dưới càng có những đặc trưng riêng không có ở lớp cha.
  + Quan hệ là một(is a): lớp con là một ví dụ khác cụ thể hơn của lớp cha. Ví dụ: mèo là động vật, cá là động vật, người là động vật,…
* Ví dụ hình ảnh kế thừa: Lớp Cat, Dog, Bird kế thừa từ Animal.

[](https://braniumacademy.net/wp-content/uploads/2020/08/vi-du-ke-thua-UML.png)

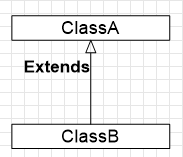
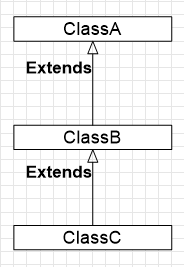
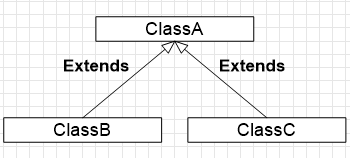
* Mục đích của kế thừa: Tái sử dụng lại code của các lớp khác. Do đó tiết kiệm thời gian, chi phí khi làm một sản phẩm nào đó.

**Những gì được kế thừa?**

* Khi một lớp con nào đó kế thừa một lớp cha, lớp con sẽ kế thừa tất cả các trường và phương thức có access modifier public và protected của lớp cha.
* Khi đó các thành phần được kế thừa sẽ được truy cập một cách trực tiếp từ trong lớp con mà không gặp bất cứ trở ngại nào. Giống như là lớp con tự định nghĩa ra các thành phần đó vậy.
* Các thành phần có access modifier là private default access của lớp cha chỉ có thể được truy cập trực tiếp từ lớp con nếu lớp con và lớp cha nằm trong cùng gói.
* Các thành phần private của lớp cha không được kế thừa bởi lớp con. Do đó đảm bảo tính đóng gói dữ liệu, thứ gì private thì của lớp nào thì chỉ có lớp đó mới có thể sử dụng trực tiếp.
* Các phương thức khởi tạo của lớp cha không được kế thừa bởi lớp con. Nhưng lớp con có thể truy cập đến phương thức khởi tạo của lớp cha qua keyword super.
* Vậy là ta đã rõ những gì được sử dụng lại ở lớp con trong mối quan hệ kế thừa của Java.

**Java chỉ hỗ trợ kế thừa đơn**

* Java chỉ hỗ trợ đơn kế thừa với lớp, tức là một lớp chỉ có thể có một lớp cha trực tiếp. Giống như mỗi người chỉ có 1 cha trực tiếp sinh ra mình vậy.
* Nếu muốn đa kế thừa, tức là một lớp kế thừa từ nhiều cha khác nhau thì ta sử dụng interface. Nội dung này sẽ trình bày cụ thể trong bài về interface.
* Vậy tại sao lại chỉ hỗ trợ đơn kế thừa? Lý do là khi đa kế thừa có thể xảy ra nhiều vấn đề phức tạp rối rắm. Chúng ta nên đơn giản hóa vấn đề chứ không nên phức tạp hóa nó. Ví dụ nếu đa kế thừa sẽ có thể xảy ra trường hợp cả hai lớp cha có cùng phương thức, cùng tham số, cùng kiểu trả về và thực hiện chức năng giống nhau->dư thừa và khó mà nhận diện được phương thức đang được sử dụng là từ lớp cha nào.
* Các dạng kế thừa với lớp:
  + Kế thừa đơn: 1 lớp con chỉ kế thừa từ 1 lớp cha và 1 lớp cha chỉ có 1 lớp con.
  + Kế thừa nhiều bậc(thế hệ): một lớp con có thể có nhiều lớp cha, ông, cụ, kị. Đồng thời lớp con đó lại là lớp cha, ông, cụ, kị của các lớp con cháu khác
  + Kế thừa phân cấp(phân nhánh): một lớp có thể là cha của nhiều hơn 1 lớp con.

[](https://braniumacademy.net/wp-content/uploads/2020/08/vi-du-single_inheritance.png)Ví dụ kế thừa đơn[](https://braniumacademy.net/wp-content/uploads/2020/08/vi-du-multilevel_inheritance.png)Ví dụ kế thừa nhiều bậc[](https://braniumacademy.net/wp-content/uploads/2020/08/vi-du-hierachical_inheritance.png)Ví dụ kế thừa phân cấp

**Khai báo kế thừa**

* Sử dụng keyword extends để thể hiện lớp A kế thừa B.
* Tổng quát:

**access non-access class A extends B {**

***// ...***

**}**

* Trong đó:
  + A là lớp con hay lớp dẫn xuất, lớp đi kế thừa lớp khác. Nó đứng trước keyword extends
  + Keyword extends là bắt buộc để thể hiện quan hệ kế thừa.
  + Sau keyword extends là B, chính là lớp cha, lớp cơ sở, lớp cho lớp khác kế thừa.
  + Trong thân của lớp con có thể chứa những khai báo riêng của chính nó mà không có ở lớp cha.

**Ví dụ minh họa**

* Ví dụ kế thừa đơn:

package lesson44;

public class **Animal** {

    private String name;

    private float weight;

    public void move() {

*// do something...*

}

    public void eat() {

*// do something...*

}

}

class **Cat extends Animal** {

    private String color;

    private String favoriteFood;

    public void catchMouse() {

*// do something...*

}

    public void washFace() {

*// do something...*

}

}

Ví dụ kế thừa nhiều thế hệ(bậc):

package lesson44;

public class **Person** {

    private String id;

    private String fullName;

    private void move() {

*// ...*

}

}

class **Student extends Person** {

    private String studentId;

    private float avgGrade;

    public void doHomeWork() {

*//...*

}

}

class **GraduatedStudent extends Student** {

    private String level;

    private float startingSalary;

    private int graduatedYear;

    public void workInOffice() {

*//...*

}

}

Ví dụ kế thừa phân cấp:

package lesson44;

public class **Animal** {

    private String name;

    private float weight;

    public void move() {

*// do something...*

}

    public void eat() {

*// do something...*

}

}

class **Cat extends Animal** {

    private String color;

    private String favoriteFood;

    public void catchMouse() {

*// do something...*

}

    public void washFace() {

*// do something...*

}

}

public class **Dog extends Animal** {

    private String species;

    public void bark() {

*//...*

}

    public void biteOther() {

*//...*

}

}