**1. Kế thừa là gì? Nó giúp ích gì trong lập trình hướng đối tượng?**

**Kế thừa (Inheritance)** là một cơ chế trong lập trình hướng đối tượng (OOP) cho phép một lớp (gọi là lớp con hoặc subclass) thừa hưởng các thuộc tính và phương thức từ một lớp khác (gọi là lớp cha hoặc superclass). Trong Java, kế thừa được thực hiện bằng cách sử dụng từ khóa extends.

**Cách hoạt động:**

* Lớp con có thể sử dụng lại mã từ lớp cha mà không cần viết lại.
* Lớp con có thể mở rộng hoặc ghi đè (override) các phương thức của lớp cha để phù hợp với nhu cầu cụ thể.

**Lợi ích trong lập trình hướng đối tượng:**

* **Tái sử dụng mã:** Giảm sự trùng lặp bằng cách cho phép các lớp con sử dụng lại mã từ lớp cha.
* **Tổ chức mã tốt hơn:** Tạo ra một hệ thống phân cấp (hierarchy) giữa các lớp, giúp mã dễ hiểu và quản lý.
* **Tính linh hoạt:** Lớp con có thể mở rộng hoặc thay đổi hành vi của lớp cha mà không cần sửa đổi lớp cha.
* **Hỗ trợ đa hình (Polymorphism):** Cho phép đối xử với các đối tượng của lớp con như các đối tượng của lớp cha, tăng tính linh hoạt trong thiết kế.

**2. Sự khác biệt giữa kế thừa và bao đóng (encapsulation) trong Java là gì?**

| **Tiêu chí** | **Kế thừa (Inheritance)** | **Đóng gói (Encapsulation)** |
| --- | --- | --- |
| **Định nghĩa** | Cho phép một lớp thừa hưởng thuộc tính/phương thức từ lớp khác. | Ẩn chi tiết triển khai và kiểm soát truy cập vào dữ liệu. |
| **Mục tiêu** | Tái sử dụng mã và thiết lập quan hệ giữa các lớp. | Bảo vệ dữ liệu và đảm bảo tính toàn vẹn của đối tượng. |
| **Cách thực hiện** | Sử dụng từ khóa extends để liên kết lớp con với lớp cha. | Sử dụng private cho thuộc tính và getter/setter công khai. |
| **Ví dụ** | Lớp Dog kế thừa từ lớp Animal. | Lớp Person có thuộc tính private String name với getter. |
| **Quan hệ** | Liên quan đến mối quan hệ "là một" (is-a). | Liên quan đến việc kiểm soát truy cập và ẩn dữ liệu. |

**Tóm lại:**

* **Kế thừa** tập trung vào việc chia sẻ và mở rộng mã giữa các lớp.
* **Đóng gói** tập trung vào việc bảo vệ dữ liệu và kiểm soát cách truy cập nó.

**3. Ví dụ ngắn giải thích cách sử dụng từ khóa extends**

class Animal {

protected String name;

public Animal(String name) {

this.name = name;

}

public void eat() {

System.out.println(name + " đang ăn.");

}

}

class Dog extends Animal {

public Dog(String name) {

super(name);

}

public void bark() {

System.out.println(name + " đang sủa: Gâu gâu!");

}

@Override

public void eat() {

System.out.println(name + " đang ăn xương.");

}

}

public class Main {

public static void main(String[] args) {

Dog dog = new Dog("Buddy");

dog.eat();

dog.bark();

Animal animal = new Animal("Leo");

animal.eat(); // "Leo đang ăn."

}

}

**4. Lợi ích và hạn chế khi sử dụng kế thừa trong Java**

**Lợi ích:**

1. **Tái sử dụng mã:** Không cần viết lại các thuộc tính/phương thức đã có trong lớp cha.
2. **Tính tổ chức:** Tạo cấu trúc phân cấp rõ ràng, dễ quản lý (ví dụ: Animal -> Dog, Cat).
3. **Hỗ trợ đa hình:** Cho phép sử dụng đối tượng lớp con như đối tượng lớp cha (ví dụ: Animal animal = new Dog();).
4. **Dễ mở rộng:** Có thể thêm chức năng mới trong lớp con mà không thay đổi lớp cha.

**Hạn chế:**

1. **Phụ thuộc chặt chẽ:** Lớp con phụ thuộc vào lớp cha, nếu lớp cha thay đổi thì lớp con có thể bị ảnh hưởng.
2. **Khó bảo trì:** Khi hệ thống phân cấp quá phức tạp (nhiều mức kế thừa), việc sửa đổi hoặc mở rộng mã trở nên khó khăn.
3. **Không linh hoạt bằng composition:** Trong một số trường hợp, sử dụng composition (tổ hợp) thay vì kế thừa sẽ tốt hơn để tránh ràng buộc cứng nhắc.
4. **Ghi đè không kiểm soát:** Nếu không cẩn thận, việc ghi đè phương thức có thể làm thay đổi hành vi không mong muốn.