**Kế Thừa (Inheritance) và Đa Hình (Polymorphism) trong Java**

Java hỗ trợ lập trình hướng đối tượng với các tính năng như **kế thừa** và **đa hình**, giúp xây dựng chương trình dễ bảo trì, mở rộng và tái sử dụng mã nguồn.

**1. Kế Thừa (Inheritance)**

**Kế thừa** là khả năng tạo một lớp mới dựa trên một lớp đã có sẵn. Lớp mới (lớp con) thừa hưởng tất cả các thuộc tính và phương thức của lớp hiện có (lớp cha), và có thể mở rộng hoặc thay đổi chúng.

**1.1. Từ khóa extends**

Khi một lớp kế thừa từ lớp khác, ta sử dụng từ khóa extends.

**Cú pháp:**

class LớpCha {

// Thuộc tính và phương thức của lớp cha

}

class LớpCon extends LớpCha {

// Lớp con có thể mở rộng thêm thuộc tính và phương thức

}

**1.2. Ví dụ Kế Thừa**

class Animal {

String name;

public void eat() {

System.out.println("The animal is eating.");

}

}

class Dog extends Animal {

public void bark() {

System.out.println("The dog is barking.");

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.name = "Buddy";

myDog.eat(); // Phương thức từ lớp cha

myDog.bark(); // Phương thức của lớp con

}

}

**Kết quả:**

The animal is eating.

The dog is barking.

* Lớp Dog kế thừa thuộc tính và phương thức của lớp Animal, đồng thời có thể thêm phương thức mới như bark().

**2. Đa Hình (Polymorphism)**

**Đa hình** cho phép một đối tượng có thể hành xử theo nhiều cách khác nhau, tùy thuộc vào việc nó đang được tham chiếu dưới lớp cha hay lớp con.

**2.1. Ghi Đè Phương Thức (Overriding)**

Ghi đè phương thức là khi lớp con định nghĩa lại phương thức đã có trong lớp cha với cùng tên, cùng kiểu trả về và tham số.

**Cú pháp:**

@Override

public void methodName() {

// Nội dung phương thức ghi đè

}

**Ví dụ:**

class Animal {

public void sound() {

System.out.println("The animal makes a sound.");

}

}

class Dog extends Animal {

@Override

public void sound() {

System.out.println("The dog barks.");

}

}

public class Main {

public static void main(String[] args) {

Animal myAnimal = new Animal();

Animal myDog = new Dog(); // Đa hình

myAnimal.sound(); // Gọi phương thức của lớp Animal

myDog.sound(); // Gọi phương thức ghi đè của lớp Dog

}

}

**Kết quả:**

The animal makes a sound.

The dog barks.

* Ở đây, đối tượng myDog là một thực thể của lớp Dog, nhưng được tham chiếu dưới kiểu Animal. Phương thức sound() được ghi đè ở lớp Dog sẽ được gọi khi sử dụng đối tượng myDog.

**3. Lớp Trừu Tượng và Giao Diện (Abstract Class và Interface)**

**3.1. Lớp Trừu Tượng (Abstract Class)**

Lớp trừu tượng là lớp không thể tạo đối tượng trực tiếp, mà phải được kế thừa bởi các lớp con. Nó có thể chứa các phương thức trừu tượng (chưa có phần thân) hoặc phương thức thông thường (đã có phần thân).

**Cú pháp:**

abstract class ClassName {

// Phương thức trừu tượng

public abstract void abstractMethod();

// Phương thức thông thường

public void regularMethod() {

// Nội dung phương thức

}

}

**Ví dụ:**

abstract class Animal {

public abstract void sound(); // Phương thức trừu tượng

public void sleep() {

System.out.println("The animal is sleeping.");

}

}

class Dog extends Animal {

@Override

public void sound() {

System.out.println("The dog barks.");

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.sound(); // Gọi phương thức trừu tượng được ghi đè

myDog.sleep(); // Gọi phương thức thông thường

}

}

**Kết quả:**

The dog barks.

The animal is sleeping.

**3.2. Giao Diện (Interface)**

**Interface** trong Java là một tập hợp các phương thức trừu tượng mà lớp thực thi giao diện đó phải định nghĩa lại. Java hỗ trợ đa kế thừa thông qua giao diện, tức là một lớp có thể thực thi nhiều giao diện.

**Cú pháp:**

interface InterfaceName {

void methodName(); // Phương thức trừu tượng

}

**Ví dụ:**

interface Animal {

void sound();

}

interface Pet {

void play();

}

class Dog implements Animal, Pet {

@Override

public void sound() {

System.out.println("The dog barks.");

}

@Override

public void play() {

System.out.println("The dog is playing.");

}

}

public class Main {

public static void main(String[] args) {

Dog myDog = new Dog();

myDog.sound(); // Gọi phương thức của giao diện Animal

myDog.play(); // Gọi phương thức của giao diện Pet

}

}

**Kết quả:**

The dog barks.

The dog is playing.

* Dog thực thi cả hai giao diện Animal và Pet, và phải định nghĩa lại tất cả các phương thức trong các giao diện này.

**Tóm Tắt**

* **Kế thừa (Inheritance)**: Lớp con sử dụng từ khóa extends để kế thừa thuộc tính và phương thức từ lớp cha.
* **Đa hình (Polymorphism)**: Lớp con có thể ghi đè (override) phương thức của lớp cha, và đối tượng có thể hành xử khác nhau dựa trên kiểu tham chiếu.
* **Lớp trừu tượng (Abstract class)**: Là lớp không thể tạo đối tượng, có thể chứa các phương thức trừu tượng hoặc thông thường. Lớp con kế thừa phải ghi đè phương thức trừu tượng.
* **Giao diện (Interface)**: Chỉ chứa các phương thức trừu tượng, và một lớp có thể thực thi nhiều giao diện.

Những tính năng này giúp lập trình hướng đối tượng trong Java trở nên mạnh mẽ và linh hoạt hơn trong việc thiết kế và mở rộng ứng dụng.