Bài: Kế thừa trong lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Kế thừa trong lập trình hướng đối tượng

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong bài trước, Kteam đã giải thích các bạn từ khóa THIS. Tiếp tục ở bài này, chúng ta sẽ tìm hiểu tính **Kế thừa trong lập trình hướng đối tượng**.

Nội dung

Để đọc hiểu bài này, tốt nhất các bạn nên có kiến thức cơ bản về các phần sau:

- CÁC BIẾN TRONG JAVA.
- CÁC KIỂU DỮ LIỆU TRONG JAVA.
- CÁC HẠNG TOÁN TỬ TRONG JAVA
- CẤU TRÚC RẼ NHÁNH TRONG JAVA
- VÒNG LẶP WHILE TRONG JAVA
- VÒNG LẶP FOR TRONG JAVA
- MẢNG TRONG JAVA
- VÒNG LẶP FOR-EACH TRONG JAVA
- VAI TRÒ BREAK, CONTINUE TRONG VÒNG LẶP JAVA
- SWITCH TRONG JAVA
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- CLASS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- CÁC LOẠI PHẠM VI TRUY CẬP TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TỪ KHÓA STATIC TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TỪ KHÓA THIS TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài này chúng ta sẽ tìm hiểu những vấn đề sau:

- Khái niệm kế thừa
- Khai báo và sử dụng kế thừa
- Chú ý về kế thừa

Khái niệm kế thừa

Kế thừa có nghĩa là thừa hưởng lại, ví dụ như tài sản của ba mẹ sẽ được giao lại cho con cái.

Kế thừa trong lập trình **(Inheritance)** có nghĩa là một lớp sẽ thừa hưởng lại những thuộc tính, phương thức từ lớp khác.

Việc sử dụng kế thừa nhằm tái sử dụng code đã viết trước đó, thuận tiện trong việc bảo trì và nâng cấp chương trình.

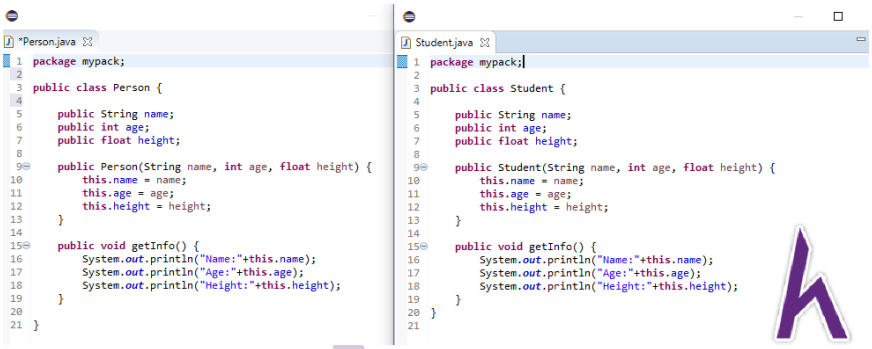
Khai báo và sử dụng kế thừa

Cú pháp:

```
class <tên lớp con> extends <tên lớp cha> {

}
```

Ví dụ: ta có 2 lớp **Person** và **Student** như sau



*Person.java

```
package mypack;

public class Person {

    public String name;
    public int age;
    public float height;

    public Person(String name, int age, float height) {
        this.name = name;
        this.age = age;
        this.height = height;
    }

    public void getInfo() {
        System.out.println("Name:"+this.name);
        System.out.println("Age:"+this.age);
        System.out.println("Height:"+this.height);
    }

}
```

Student.java

```
package mypack;

public class Student {

    public String name;
    public int age;
    public float height;

    public Student(String name, int age, float height) {
        this.name = name;
        this.age = age;
        this.height = height;
    }

    public void getInfo() {
        System.out.println("Name:"+this.name);
        System.out.println("Age:"+this.age);
        System.out.println("Height:"+this.height);
    }
}
```

Ta thấy 2 lớp **Person** và **Student** có chung thuộc tính và phương thức. Ta sẽ để nguyên code ở lớp **Person** lại:

:

```
public class Person {

        public String name;
        public int age;
        public float height;

        public Person(String name, int age, float height) {
                this.name = name;
                this.age = age;
                this.height = height;
        }

        public void getInfo() {
                System.out.println("Name:"+this.name);
                System.out.println("Age:"+this.age);
                System.out.println("Height:"+this.height);
        }
}
```

Bây giờ ta chỉ cần cho lớp Student kế thừa Person như sau:

:

```
public class Student extends Person{

        public Student(String name, int age, float height) {
                super(name, age, height);
        }
}
```

Trong phương thức khởi tạo Student, ta sẽ dùng từ khóa **super** để cho lớp con truy cập các những thứ liên quan đến lớp cha. Như ví dụ trên thì ta dùng **super()** để gọi phương thức khởi tạo lớp cha.

Tiếp theo, ta thử khởi tạo đối tượng Student và gọi phương thức **getInfo()**:

:

```
public class HelloWorld {

	public static void main(String[] args) {
		Student a = new Student("Chau", 21, 1.7f);
		a.getInfo();
	}
}
```

Theo kết quả, đối tượng a sử dụng được phương thức **getInfo()** từ lớp cha

```
<terminated> HelloWorld (1) [Jav
Name:Chau
Age:21
Height:1.7
```

Chú ý về kế thừa

Slogan đặc trưng kế thừa: "Cha có thì con có, con có chưa chắc cha đã có"

Tính chất kế thừa các ngôn ngữ lập trình hướng đối tượng đa số đều tương đồng với nhau về tính chất. Có thể các bạn không nhớ khái niệm và cú pháp, nhưng chỉ cần hiểu câu nói trên là bạn đã hiểu về kế thừa.

Ví dụ: Như ví dụ trước thì lớp Student kế thừa Person, ngoài những thuộc tính kế thừa ra, ta muốn thêm thuộc tính **universityName** cho Student

:

```
public class Student extends Person {

	public String universityName;

	public Student(String name, int age, float height, String universityName) {
		super(name, age, height);
		this.universityName = universityName;
	}
}
```

Như vậy theo đúng tính chất: lớp cha Person có name, age, height thì lớp con Student có. Lớp con Student có **universityName** thì lớp cha Person không có.

Tận dụng từ khóa super để bảo trì và nâng cấp code

Từ khóa super mục đích chính truy cập những phương thức của lớp cha. Trong việc phát triển phần mềm, ta cần nâng cấp chương trình. Việc tận dụng từ khóa super sẽ giúp ta vừa tận dụng những dòng code trước đó và viết tiếp code mới.

Ví dụ: ta thấy phương thức **getInfo()** chỉ trả về thông tin name, age, height. Bây giờ, ta sẽ nâng cấp phương thức có thể trả về thông tin universityName ở lớp Student

:

```
public class Student extends Person {

	public String universityName;

	public Student(String name, int age, float height, String universityName) {
		super(name, age, height);
		this.universityName = universityName;
	}

	public void getInfo() {
		super.getInfo();
		System.out.println("University Name:"+this.universityName);
	}

}
```

Ta thử khởi tạo để kiểm tra:

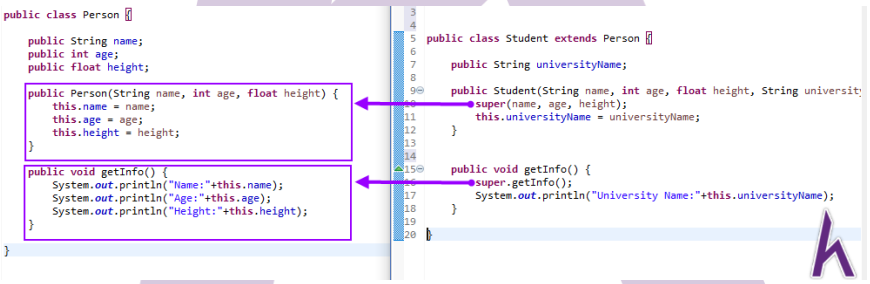
:

```
public class HelloWorld {

	public static void main(String[] args) {
		Student a = new Student("Chau", 21, 1.7f, "UTE");
		a.getInfo();
	}
}
```

```
Name:Chau
Age:21
Height:1.7
University Name:UTE
```

Flow của chương trình có thể hiểu như sau:



Kết

Như vậy chúng ta đã tìm hiểu kế thừa trong lập trình hướng đối tượng

Ở bài sau, Kteam sẽ giới thiệu đến bạn về SETTER VÀ GETTER TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".