Bài: Interface trong lập trình hướng đối tượng Java

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

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

Ở bài trước, chúng ta đã tìm hiểu về TÍNH TRỪU TƯỢNG trong lập trình hướng đối tượng. Hôm nay, Kteam sẽ giới thiệu cho các bạn về **interface** để biết về đa kế thừa trong Java.

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
- THỪA KẾ TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- SETTER & GETTER TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- OVERRIDING VÀ OVERLOADING TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
- TÍNH TRỪU TƯỢNG TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

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

- Interface là gì? Tại sao phải sử dụng?
- Khai báo và sử dụng interface

Interface là gì? Tại sao phải sử dụng?

Interface là một kiểu dữ liệu tham chiếu trong Java. Nó là tập hợp các phương thức **abstract** (trừu tượng). Khi một lớp kế thừa **interface**, thì nó sẽ kế thừa những phương thức **abstract** của **interface** đó.

Một số đặc điểm của interface:

- Không thể khởi tạo, nên không có phương thức khởi tạo.
- Tất cả các phương thức trong interface luôn ở dạng public abstract mà không cần khai báo.
- Các thuộc tính trong interface luôn ở dạng public static final mà không cần khai báo, yêu cầu phải có giá trị.

Mục đích của interface là để thay thế đa kế thừa lớp của những ngôn ngữ khác (ví dụ như C++, Python...). Ngoài ra, interface sẽ giúp đồng bộ và thống nhất trong việc phát triển hệ thống trao đổi thông tin.

Khai báo và sử dụng interface

Cú pháp:

```
interface <tên interface> {

    // Khai báo các thành phần bên trong interface

}
```

Bây giờ ta sẽ tạo ra interface **IStudy** giành riêng cho class **Student**, ta vẫn tạo file .java như mọi khi và viết chương trình như sau:

java:

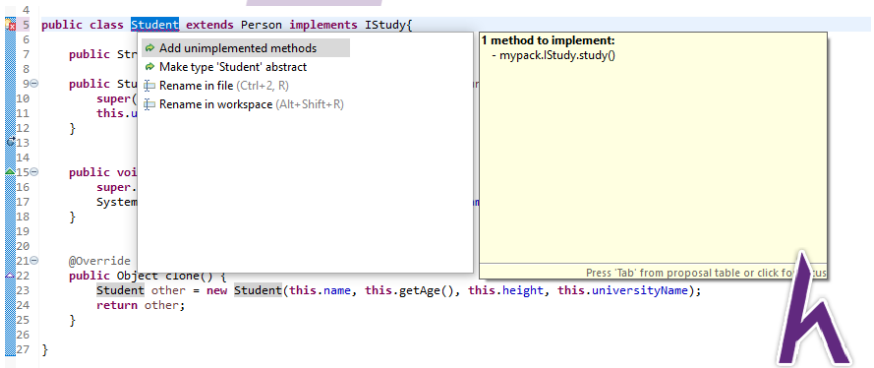
```
interface IStudy {
        void study();
}
```

Ta cho class **Student** kế thừa nó như sau:

java:

```
public class Student extends Person implements IStudy{
```

Nếu dùng Eclipse, bạn sẽ thấy IDE yêu cầu override lại phương thức **study()** của **IStudy** ngay:



Ta sẽ overriding, thêm đoạn chương trình trong lớp **Student** như sau:

java:

```
@Override
public void study() {
        // TODO Auto-generated method stub
        System.out.println(this.name+" is studing");
}
```

Một class có thể kế thừa nhiều interface, ta sẽ thử tạo thêm interface **ISpeak**:

java:

```
interface ISpeak {
        void speak();
}
```

Ta thêm interface **ISpeak** vào class **Student** bằng cách sau:

java:

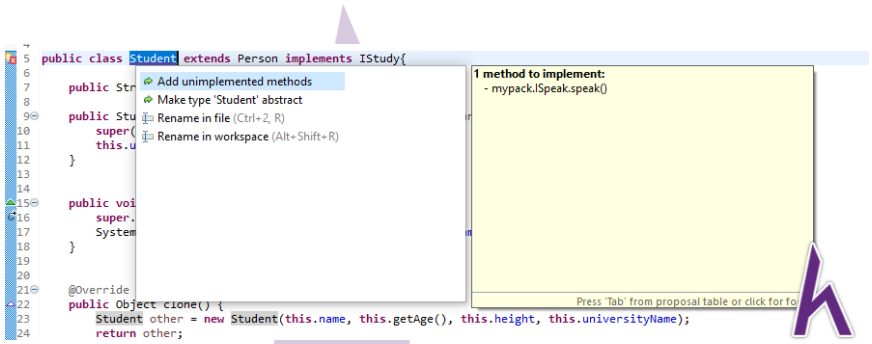
```
public class Student extends Person implements IStudy, ISpeak{
```

Hoặc, ta thử thêm **ISpeak** ở lớp cha Person:

java:

```
public abstract class Person implements ISpeak{
```

Bởi vì class **Person** là lớp ảo, nên **Person** không cần override phương thức **speak()**. Ngoài ra, **Student** là lớp con **Person**, nên mặc dù **Student** không kế thừa **ISpeak** trực tiếp nhưng vẫn phải override phương thức **speak()**. Ta sẽ thấy Eclipse yêu cầu khai báo:



Ta sẽ hoàn thiện lớp **Student** như sau:

java:

```
public class Student extends Person implements IStudy{

	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);
	}


	@Override
	public Object clone() {
		Student other = new Student(this.name, this.getAge(), this.height, this.universityName);
		return other;
	}


	@Override
	public void study() {
		// TODO Auto-generated method stub
		System.out.println(this.name+" is studing");
	}


	@Override
	public void speak() {
		// TODO Auto-generated method stub
		System.out.println(this.name+" is speaking");
	}
}
```

Kết

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

Ở bài sau, Kteam sẽ giới thiệu đến bạn về PHƯƠNG THỨC MAIN TRONG JAVA

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**".