Bài: Overriding và Overloading trong Java

Xem bài học trên website để ủng hộ Kteam: Overriding và Overloading trong 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

Trong bài trước, Kteam đã giới thiệu các bạn về SETTER & GETTER trong lập trình hướng đối tượng. Ở bài này, chúng ta sẽ tìm hiểu về **overriding và overloading 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

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

- Overriding là gì? Cách sử dụng
- Overloading là gì? Cách sử dụng

Overriding là gì? Cách sử dụng

Overriding (ghi đè) có nghĩa là có 2 phương thức giống nhau về tên và tham số truyền vào. Một phương thức ở lớp cha, còn cái kia ở lớp con. **Overriding** cho phép lớp con có thể thực hiện riêng biệt cho phương thức mà lớp cha đã cung cấp.

Ví dụ: Như trong bài trước ta tạo lớp **Student** kế thừa lớp **Person**. Với phương thức **getInfo** của lớp **Person** chỉ in được thông tin name, age, height trong khi lớp **Student** còn có thuộc tính **universityName**. Như vậy, ta sẽ overriding lại phương thức **getInfo**:

java:

```
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);
	}

}
```

Khi ta khai báo phương thức **getInfo** trong lớp **Student**, có nghĩa ta đang overrding. Và đối tượng thuộc lớp **Student** sẽ gọi phương thức **getInfo** từ lớp **Student** thay vì lớp **Person**.

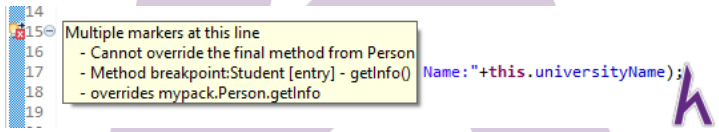
Cách chống Overriding

Nếu không muốn lớp con có thể **Overriding** lại phương thức nào đó, ta sẽ sử dụng từ khóa final

java:

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

Lớp con sẽ không thể Overriding được phương thức **getInfo()**



Overloading là gì? Cách sử dụng

Overloading là nhiều phương thức trong một lớp có chung tên nhưng khác tham số truyền vào

Ví dụ: với setter cho **thuộc tính age**, có thể người dùng truyền vào **tham số age** là kiểu int, kiểu byte, short hoặc long. Như vậy, ta sẽ Overloading nhiều phương thức setter cho **thuộc tính age** để đảm bảo.

java:

```
public class Person {

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

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

	public void setAge(int age) {
		if (age>=0 && age<=100 ) {
			this.age = age;
		}
	}

	public void setAge(byte age) {
		if (age>=0 && age<=100 ) {
			this.age = age;
		}
	}

	public void setAge(short age) {
		if (age>=0 && age<=100 ) {
			this.age = age;
		}
	}

	public void setAge(long age) {
		if (age>=0 && age<=100 ) {
			this.age = (int) age;
		}
	}

	public int getAge() {
		return this.age;
	}

	public Person clone() {
		Person other = new Person(this.name, this.age, this.height);
		return other;
	}

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

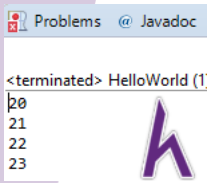
Giờ ta sẽ kiểm tra trong chương trình main:

java:

```
public class HelloWorld {

	public static void main(String[] args) {
		Person a = new Person("Chau", 21, 1.7f);
		byte b = 20;
		a.setAge(b);
		System.out.println(a.getAge());
		short c = 21;
		a.setAge(c);
		System.out.println(a.getAge());
		int d = 22;
		a.setAge(d);
		System.out.println(a.getAge());
		long e = 23;
		a.setAge(e);
		System.out.println(a.getAge());
	}
}
```

Như vậy, ứng với tham số kiểu dữ liệu nào, Java sẽ gọi phương thức liên quan mà không gặp rắc rối nào:



Kết

Như vậy chúng ta đã tìm hiểu overriding và overloading trong lập trình hướng đối tượng

Ở bài sau, Kteam sẽ giới thiệu đến bạn về TÍNH TRỪU TRƯỢNG 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**".