Bài: Setter và getter trong lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Setter và getter 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 giới thiệu các bạn về KẾ THỪA TRONG LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG. Ở bài này, chúng ta sẽ tìm hiểu lại 2 phương thức **setter và getter**.

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

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

- Setter và Getter là gì? Lý do sử dụng
- Cú pháp và sử dụng Setter và Getter
- Chú ý

Setter và Getter là gì? Lý do sử dụng

Setter và **Getter** là 2 phương thức sử dụng để cập nhật hoặc lấy ra giá trị thuộc tính, đặc biệt dành cho các thuộc tính ở phạm vi **private**.

Việc sử dụng Setter và Getter cần thiết trong việc kiểm soát những thuộc tính quan trọng mà ta thường được sử dụng và yêu cầu giá trị chính xác. Ví dụ thuộc tính age lưu tuổi con người, thực tế thì phạm vi tuổi là từ 0 đến 100, thì ta không thể cho chương trình lưu giá trị age âm hoặc quá 100 được.

Cú pháp và sử dụng Setter và Getter

Cú pháp:

Setter

```
public void set<tên thuộc tính> (<tham số giá trị mới>) {

    this. <tên thuộc tính> = <tham số giá trị mới>;

}
```

Getter

```
<kiểu dữ liệu thuộc tính> get<tên thuộc tính> () {

    return this. <tên thuộc tính>;

}
```

Ví dụ: ta sẽ tạo phương thức setter và getter cho thuộc tính age lớp Person

java:

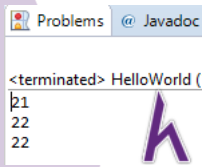
```
public class Person {

        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 int getAge() {
                return this.age;
        }

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

}
```

Vì ta đã chỉnh age sang private nên giờ chỉ có thể truy cập thông qua 2 phương thức này:

java:

```
public class HelloWorld {

	public static void main(String[] args) {
		Person a = new Person("Chau", 21, 1.7f);
		System.out.println(a.getAge());
		a.setAge(22);
		System.out.println(a.getAge());
		a.setAge(-5);
		System.out.println(a.getAge());
	}
}
```

Theo kết quả, giá trị age cuối cùng là 22



Chú ý

Khi đã dùng setter và getter thì thuộc tính nên để private

Vì setter và getter nhằm quản lý truy cập của thuộc tính, thì ta không nên để thuộc tính có thể truy cập dễ dàng, không nên để ở dạng public.

Hãy cẩn thận với kiểu dữ liệu tham chiếu

Ta tạo một class có thuộc tính là kiểu dữ liệu tham chiếu là một mảng như sau

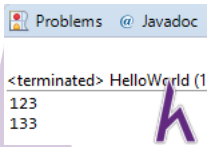
java:

```
public class Example {

	private int[] array;

	public void setArray (int[] array) {
		this.array = array;
	}

	public void displayArray() {
		for (int a : this.array) {
			System.out.print(a);
		}
		System.out.println();
	}

}
```

Ta hãy thử trường hợp này:

java:

```
public class HelloWorld {

	public static void main(String[] args) {
		Example a = new Example();
		int[] mang = {1,2,3};
		a.setArray(mang);
		a.displayArray();
		mang[1] = 3;
		a.displayArray();
	}
}
```



Như kết quả, lúc đầu ta tạo mảng mang ở chương trình main và gán nó cho thuộc tính **array** của lớp Example, như vậy **array** có giá trị là {1,2,3}. Tuy nhiêu, sau đó ta thử thay đổi giá trị một phần tử trong mang thì giá trị **array** cũng thay đổi theo.

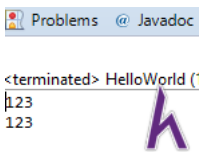
Lý do khi ta gán giá trị trong kiểu dữ liệu tham chiếu, bản chất ta gán giá trị vùng bộ nhớ lưu trữ. Có nghĩa lúc này **array** và mang đang ánh xạ chung một đối tượng trong bộ nhớ.

Cách xử lý là tạo một bộ nhớ khác và copy giá trị đó vào. Ta sẽ dùng phương thức **clone()**, đây là phương thức hỗ trợ của Java, nó sẽ tạo một bản sao rồi trả bản sao đó cho đối tượng được gán.

java:

```
public class Example {

	private int[] array;

	public void setArray (int[] array) {
		this.array = array.clone();
	}

	public void displayArray() {
		for (int a : this.array) {
			System.out.print(a);
		}
		System.out.println();
	}

}
```

Ta chạy thử thì kết quả thì nó hết sai:



Cũng tương tụ cho phương thức Getter, ta cũng return về bản sao của thuộc tính đó:

java:

```
public class Example {

	private int[] array;

	public void setArray (int[] array) {
		this.array = array.clone();
	}

	public int[] getArray() {
		return this.array.clone();
	}

	public void displayArray() {
		for (int a : this.array) {
			System.out.print(a);
		}
		System.out.println();
	}

}
```

> **Lưu ý:** Riêng với kiểu dữ liệu String, mặc dù là kiểu dữ liệu tham chiếu. Tuy nhiên, String có cơ chế là khi thay đổi giá trị thì nó bản chất nó đang tạo ra một đối tượng String mới. Vì vậy, như các kiểu dữ liệu nguyên thủy, bạn có thể làm phương thức getter và setter bình thường.

Tự tạo phương thức clone() cho lớp của mình

Như ở bài giải thích về hướng đối tượng, thì lớp cũng chính là kiểu dữ liệu do ta tự định nghĩa ra. Vì nó là kiểu dữ liệu tham chiếu, có những lúc lớp ta viết sẽ nằm trong tham số phương thức Getter và Setter.

Vậy ta thứ viết phương thức **clone()** trong lớp Person như sau:

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 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);
	}

}
```

Kết

Như vậy chúng ta đã tìm hiểu setter và getter trong lập trình hướng đối tượng

Ở bài sau, Kteam sẽ giới thiệu đến bạn về OVERRIDING VÀ OVERLOADING 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**".