Bài: Class trong lập trình hướng đối tượng

Xem bài học trên website để ủng hộ Kteam: Class 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 cho các bạn sơ qua về LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG. Bây giờ, Kteam sẽ giải thích các bạn rõ hơn về **Class 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

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

- Class là gì?
- Cú pháp khai báo class.
- Thuộc tính và phương thức trong hướng đối tượng.
- Phương thức khởi tạo

Class là gì?

Class (Lớp) là người dùng định nghĩa thiết kế cho hướng đối tượng. Nó đại diện cho những tập thuộc tính và phương thức chung cho tất cả các đối tượng của lớp này.

Nếu trong những ngôn ngữ thuần hướng đối tượng như **Java**, **C#**,.. thì Class chính là kiểu dữ liệu mà lập tình viên tự tạo ra.

Cú pháp khai báo Class

Cú pháp:

```
<Phạm vi truy cập> class <tên lớp> {

   <Phạm vi truy cập> <các thành phần của lớp>;

}
```

Trong đó:

- **<tên lớp>**: Là tên class do người tập trình tự tạo ra
- **<Phạm vi truy cập>**: gồm có những từ khóa public, protected, private,... sẽ được trình bày trong bài CÁC LOẠI PHẠM VI TRUY CẬP TRONG JAVA
- **<các thành phần của lớp>**: Là biến hoặc phương thức trong lớp.

Ví dụ:

:

```
public class Person {
        public String name;
        public int age;
        public float height;

        public void eat() {
                System.out.print("Person is eating");
        }
}
```

- Khái báo một class tên là **Person**.
- Class Person có các thuộc tính: name lưu tên, age lưu tuổi, height lưu chiều cao
- Class Person có phương thức eat: Khi được gọi sẽ in ra màn hình "**Person is eating**"
- Các từ khóa **public** sẽ giải thích ở bài sau.

Thuộc tính và phương thức trong hướng đối tượng

Thuộc tính

Thuộc tính là những thông tin riêng của mỗi đối tượng, ta có thể thấy nó như là những biến liên quan đến đối tượng đó.

Chúng ta cần phải thống nhất nhóm đối tượng cần có những thông tin cơ bản gì? Không thể có chuyện đối tượng bạn **A** có tên, tuổi, chiều cao; bạn **B** chỉ có tên, cân nặng, quê quán; Việc thông tin không thống nhất gây ra quản lý khó đảm bảo.

Đó là lý do ta phải khai báo các thuộc tính trong lớp để các đối tượng của lớp đó bắt buộc phải có thông tin lưu trữ các thuộc tính trên.

Phương thức

Đây là kiến thức khá mới mẻ trong loạt bài viết này. Phương thức trong hướng đối tượng là cách xử lý hành vi của đối tượng. Bản chất, trong phương thức sẽ chứa loạt code, khi ta gọi phương thức của đối tượng, những dòng code trong phương thức đó sẽ thực hiện.

Nếu các bạn đã từng học các ngôn ngữ lập trình hướng thủ tục, thì phương thức nó khá giống hàm. Tuy nhiên, phương thức khác hàm là phương thức phải khai báo trong lớp, còn hàm thì khai báo độc lập.

Cú pháp:

```
<Phạm vi truy cập> <từ khóa> <Kiểu dữ liệu trả về> <tên phương thức> ([Tham số]) {

    <Chương trình>

}
```

- **<Phạm vi truy cập>**: Phạm vi truy cập phương thức, sẽ nói bài sau.
- **<từ khóa>**: Gồm các từ khóa final, static,... sẽ nói ở những bài sau.
- **<Kiểu dữ liệu trả về>**: Ta có thể định nghĩa phương thức có trả về dữ liệu kiểu gì không. Như trả kiểu int, long, double hoặc tên một class nào đó, nếu không trả về gì ta chọn từ khóa **void**. Để trả dữ liệu, ta sẽ dùng từ khóa **return** trong phương thức.
- **<tên phương thức>**: Tên của phương thức
- **[Tham số]**: Là những tham số ta muốn truyền vào phương thức để thực hiện.

- **<Chương trình>**: Những dòng code thực hiện khi gọi tên phương thức.

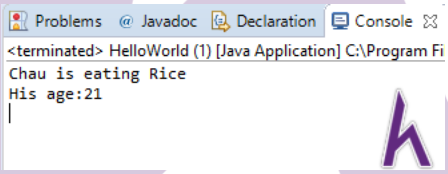
Ví dụ:

:

```
public class Person {
	public String name;
	public int age;
	public float height;

	public void eat(String foodName) {
		System.out.println(name + " is eating "+ foodName);
	}

	public int getAge() {
		return age;
	}
}
```

:

```
public class HelloWorld {
	public static void main(String[] args) {
		Person a = new Person();
		a.name = "Chau";
		a.age = 21;
		a.height = 1.7f;

		a.eat("Rice");
		int age = a.getAge();
		System.out.println("His age:"+age);
	}
}
```



Trong khái báo lớp **Person**. Ta có phương thức eat(), có tham số truyền vào là biến foodName kiểu **String**. Ta sẽ in ra màn hình tên của đối tượng **Person** ăn món có tên là foodName. Vì phương thức eat() không trả về giá trị gì, ta để từ khóa là **void**.

Ở phương thức getAge() là trả về tuổi của đối tượng. Vì vậy ta chọn từ khóa là int vì biến age trong class thuộc kiểu int. Rồi trong phương thức, ta **return** giá trị age của đối tượng về.

Trong chương trình main. Sau khi khởi tạo đối tượng và gán giá trị cho các thuộc tính. Ta thử chạy phương thức eat(), ta truyền vào giá trị "Rice" vào tham số. Như vậy, giá trị tham số foodName sẽ đại diện giá trị "Rice". Sau đó sẽ in ra màn hình là "**Chau is eating Rice**".

Ở phương thức getAge(), vì phương thức sẽ **return** giá trị thuộc tính age của đối tượng. Ta khai báo một biến age để lưu giá trị mà phương thức trả về. Sau đó in nó ra màn hình.

Phương thức khởi tạo

Các bạn sẽ thấy, khi chúng ta tạo ra một đối tượng ta sẽ viết Person **a = new Person();**. Thực ra, khi chúng đang gọi phương thức khởi tạo của đối tượng. Thường phương thức khởi tạo sẽ cùng tên với class.

Mặc định, phương thức khởi tạo tự có như trên. Tuy nhiên, ta thường cần chỉnh sửa phương thức khởi tạo cho hợp yêu cầu. Ví dụ như đoạn code này:

:

```
public class HelloWorld {
	public static void main(String[] args) {
		Person a = new Person();
		a.name = "Chau";
		a.age = 21;
		a.height = 1.7f;
	}
}
```

Sau khi khởi tạo, ta cần phải gán giá trị cho các thuộc tính của đối tượng a. Tuy nhiên, trong quá trình phát triển phần mềm, người sử dụng class chưa chắc hiểu hết toàn bộ code bên trong một class. Vì vậy, từ vấn đề trên, ta muốn mỗi khi khởi tạo một đối tượng Person thì cần cung cấp thông tin ngay trong phương thức khởi tạo.

Ta sẽ khai báo phương thức khởi tạo như sau:

:

```
public class Person {
	public String name;
	public int age;
	public float height;

	public Person(String name, int age, float height) {

	}

	public void eat(String foodName) {
		System.out.println(name + " is eating "+ foodName);
	}

	public int getAge() {
		return age;
	}
}
```

Như vậy trong phương thức khởi tạo yêu cầu cần 3 tham số truyền vào là **name**, **age**, **height**. Bây giờ, ta sẽ gán các giá trị của 3 tham số đó vào 3 thuộc tính đối tượng tương ứng.

Tuy nhiên, nơi ta đang viết là đang định nghĩa của lớp, không phải chương trình xử lý của đối tượng. Như vậy, làm sao gán giá trị cho 3 thuộc tính của đối tượng. Để làm được điều đó, ta sẽ sử dụng từ khóa **this** để ám chỉ đối tượng trong class. Ở những bài viết sau, sẽ giải thích rõ về từ khóa **this** hơn.

:

```
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 eat(String foodName) {
		System.out.println(name + " is eating "+ foodName);
	}

	public int getAge() {
		return age;
	}
}
```

Ta sẽ sửa làm chương trình main như sau:

:

```
public class HelloWorld {
	public static void main(String[] args) {
		Person a = new Person("Chau", 21, 1.7f);
		a.eat("Rice");
	}
}
```

- Ta sẽ khởi tạo Person **a = new Person("Chau", 21, 1.7f);**. Truyền các tham số vào để gán cho các thuộc tính của đối tượng a.
- Khi chạy phương thức khởi tạo, các thuộc tính của đối tượng a sẽ gán giá trị. Ta thấy sự tương đồng như sau:

Như vậy, ta đang rút gọn lại code. Khi khởi tạo đối tượng thì lập tức gán các giá trị và các đối tượng luôn, người lập trình sẽ giảm code gán giá trị (Việc giảm code như thế này cũng giảm được rủi ro trong lập trình). Và ta thấy từ khóa **this** chính là đại diện cho đối tượng **a**, và sau này ta tạo các đối tượng **b,c**,... thì this sẽ đại diện tương tự.

Kết

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

Ở bài sau, Kteam sẽ giới thiệu đến bạn về CÁC LOẠI PHẠM VI TRUY CẬP 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**".