# Bài: Tính trừu tượng trong lập trình hướng đối tượng với Java

*Xem bài học trên website để ủng hộ Kteam: Tính trừu tượng trong lập trình hướng đối tượng với 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 OVERIDING & OVERLOADING trong lập trình hướng đối tượng

Hôm nay, Kteam sẽ giới thiệu cho các bạn một tính chất quan trọng trong lập trình hướng đối tượng chung. Đó là **tính trừu tượng**, hay cố gắng tìm hiểu trong bài học này.

---

## 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

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

- Tính trừu tượng là gì?
- Trừu trượng trong lập trình hướng đối tượng là gì?
- Tính trừu tượng trong Java

---

## Tính trừu tượng là gì?

Mặc dù đây là bài viết lập trình, nhưng Kteam sẽ nói qua về ngôn ngữ học, rất nhiều người lập trình lâu năm đôi khi họ không thể hiểu bản chất từ **trừu tượng**

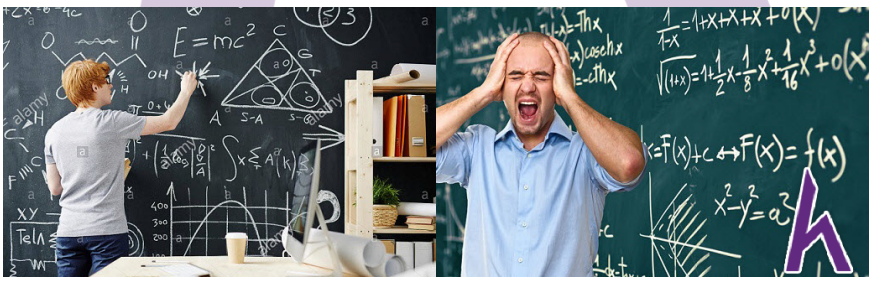
**Trừu tượng** là một từ Hán Việt: '**trừu**' nghĩa là rút ra, '**tượng**' có nghĩa là hình tượng, tượng trưng. Vậy theo nghĩa bóng, trừu tượng có nghĩa là rút ra một khái niệm từ những hình tượng cụ thể, tạo ra một ý niệm trong suy nghĩ con người.

Tính trừu tượng rất ý nghĩa không những lập trình nói riêng mà trong giáo dục nói chung. Khá bất ngờ tính trừu tượng đã áp dụng từ những học sinh lớp 1 và mẫu giáo. Kteam sẽ lấy ví dụ sau:


Nếu ai còn nhớ, đây chính là que tính được sử dụng khi học toán lớp 1 (Sau này Kteam không rõ giáo dục có sự thay đổi không, có thể đã dạy sớm hơn trong chương trình mẫu giáo). Việc học toán lúc đầu cho các em nhỏ cần phải hiểu về các con số và toán tử, có thể giải thích quy trình như sau:

- **Bước 1:** Cho các em nhỏ tập đếm dựa theo que tính để làm quen với con số, để cho tụi nhỏ hiểu ý nghĩa các con số như số 1 là một cái, số 2 là hai cái,...
- **Bước 2:** Khi làm quen được các con số, các thầy cô sẽ dạy cách tính toán: thường các cô để 5 cái bên trái, 5 cái bên phải và hỏi tổng là bao nhiêu. Các bạn đừng có nghĩ đa số tụi nhỏ sẽ biết 5+5=10 (tùy theo khả năng mỗi đứa). Chúng sẽ đếm bên trái xong rồi đếm bên phải, dần dần trong đầu sẽ trừu tượng ra phép cộng. Rồi thầy cô thử lấy ra một vài que tính, tụi nhóc cũng phải tập đếm lại và cũng trừu tượng ra phép trừ...

Như vậy, việc giáo dục toán que tính đang tạo trừu tượng toán học cho học sinh, dần dần các khái niệm trừu trượng của toán học cao hơn sẽ được dạy dần như nhân chia, lũy thừa, căn bậc, đạo hàm,... Thì tính trừu tượng toán càng cao, học toán càng giỏi. Nếu thời học sinh bạn sẽ thấy: đứa giỏi toán khi nhìn công thức nó sẽ ngồi ngẫm phân tích trong đầu rồi mới thử giải, có những đứa gặp công thức thì giải theo cách này đến cách khác mà chưa phân tích trước thì bản chất là đang thử sai và dựa vào may mắn, đứa yếu hơn thì nhìn công thức chỉ thấy rối đầu.




Ngay trong lập trình cũng vậy: Đứa giỏi lập trình thì khi có vấn đề thì sẽ ngồi ngẫm phân tích để đưa ra ý tưởng trước khi code, còn đứa không giỏi thì cứ nhảy vào code trong khi chưa rõ lý do rồi sau đó nghĩ mình đang viết cái gì.

## Tính trừu tượng trong lập trình hướng đối tượng là gì?

**Tính trừu tượng trong lập trình hướng đối tượng** là chỉ nêu ra vấn đề mà không hiển thị cụ thể, chỉ hiện thị tính năng thiết yếu đối với đối tượng người dùng mà không nói quy trình hoạt động. Ví dụ: như tạo ra tính năng gửi tin nhắn, ta chỉ cần hiểu là người dùng viết tin rồi nhấn gửi đi. Còn quy trình xử lý tin nhắn gửi như thế nào thì ta chưa đề cập đến.

Như vậy, tính trừu tượng là che giấu thông tin thực hiện từ người dùng, họ chỉ biết tính năng được cung cấp: Chỉ biết thông tin đối tượng thay vì cách nó sử dụng như thế nào. Nó có những ưu điểm sau:

- Cho phép lập trình viên bỏ qua những phức tạp trong đối tượng mà chỉ đưa ra những khái niệm phương thức và thuộc tính cần thiết. Ta sẽ dựa những khái niệm đó để viết ra, nâng cấp và bảo trì.
- Nó giúp ta tập trung cái cốt lõi đối tượng. Giúp người dùng không quên bản chất đối tượng đó làm gì.

---

# Tính trừu tượng trong Java

## Lớp trừu tượng

**Lớp trừu tượng** là lớp được khai báo mà không thể tạo ra đối tượng từ lớp đó. Ta sẽ tạo những lớp con kế thừa lớp trừu tượng.

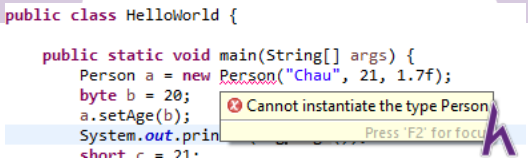
Mục đích lớp trừu tượng là tạo ra lớp chung cho những lớp có liên quan với nhau kế thừa. Ví dụ khi xây dựng phần mềm quản lý nhà trường: Những lớp sinh viên, giảng viên, cán bộ,... có những thuộc tính và phương thức chung như tên, năm sinh, quê quán,... thì ta sẽ tạo một lớp con người là lớp trừu tượng và những đặc điểm chung được để trong lớp con người. Khi phát triển chương trình, ta chỉ có thể tạo các đối tượng từ lớp con kế thừa lớp con người; không thể cho tạo đối tượng từ lớp con người được.

Để tạo lớp trừu tượng ta dùng từ khóa **abstract** trước từ khóa **class**. Ta sẽ dùng lớp Person từ những bài trước đó, biến nó thành lớp **abstract**:

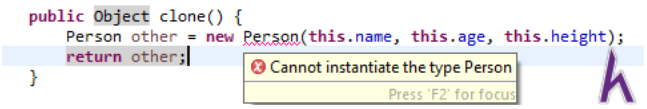
:

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

}
```

Khi ta thử khởi tạo đối tượng lớp Person, Eclipse sẽ cảnh báo lỗi:




Ngay cả phương thức Clone ta viết bên trong bài trước cũng cảnh báo (Code phía trên Kteam đã xóa phương thức đó).




Như vậy, chỉ có những lớp kế thừa lớp Person mới có thể sử dụng được.

## Phương thức trừu tượng

**Các phương thức trừu tượng** là là chỉ định nghĩa mà không có chương trình bên trong, lớp con kế thừa phải bắt buộc override nó lại để sử dụng. Phương thức trừu tượng có ý nghĩa định nghĩa phương thức bắt buộc phải có trong lớp con kế thừa.

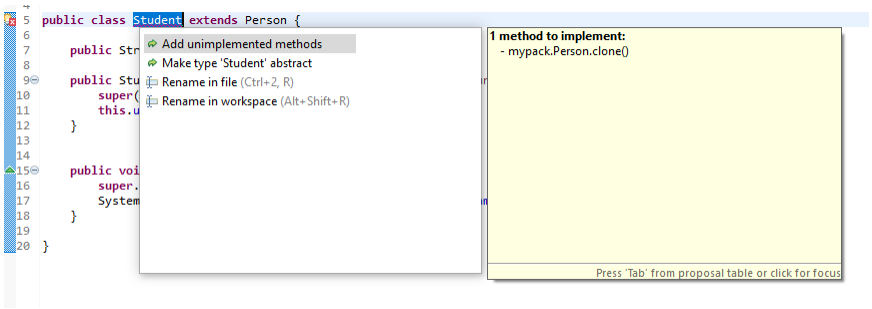
**Ví du:** Ta sẽ tạo phương thức trừu tượng **clone()** trong lớp Person để bắt các lớp con phải override lại.

:

```
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 abstract Object clone();

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

Ở phương thức **clone()**, ta cho phương thức trả từ khóa **Object** có nghĩa phương thức sẽ trả kiểu một đối tượng chung nào đó (như Person, Student, Example... ta gọi chung là **Object**).

Bây giờ tại lớp **Student**, Eclipse sẽ đưa ra cảnh báo phải override lại phương thức **clone()**




Ta sẽ override lại như sau:

:

```
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);
	}


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

}
```

### Chú ý

Trong dòng code

:

```
Student other = new Student(this.name, this.getAge(), this.height, this.universityName);
```

Vì thuộc tính **age** ở phạm vi **private** nên lớp con phải gọi this.**getAge()** để lấy giá trị. Theo Kteam thì nên cho các thuộc tính lớp trừu tượng ở dạng protected để lớp con kế thừa dễ dàng truy cập.

---

## Kết

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

Ở bài sau, Kteam sẽ giới thiệu đến bạn về INTERFACE 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**".