**Tính trừu tượng (Abstraction) trong Java**

**1. Khái niệm tính trừu tượng**

Tính trừu tượng trong Java là tính chất không thể hiện cụ thể mà chỉ nêu tên vấn đề. Đó là một quá trình che giấu các hoạt động bên trong và chỉ hiển thị những tính năng thiết yếu của đối tượng tới người dùng. Ví dụ: một người sử dụng điện thoại để gửi tin nhắn thì anh ta sẽ nhập nội dung tin nhắn, thông tin người nhận và ấn nút gửi. Khi anh ta bắt đầu gửi tin thì anh ấy không biết những gì diễn ra bên trong quá trình gửi mà chỉ biết được là kết quả của tin nhắn đã được gửi đến người nhận thành công hay chưa. Vì vậy trong ví dụ này, quá trình gửi tin nhắn đã được ẩn đi và chỉ hiển thị những chức năng mà người dùng cần đó là chức năng nhập nội dung tin nhắn, thông tin người nhận, kết quả gửi tin nhắn thành công hay thất bại. Đó chính là tính trừu tượng.

Ưu điểm khi sử dụng tính trừu tượng để lập trình:

* Tính trừu tượng cho phép các lập trình viên loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình, cải thiện khả năng bảo trì của hệ thống.
* Tính trừu tượng giúp chúng ta tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.
* Tính trừu tượng cung cấp nhiều tính năng mở rộng khi sử dụng kết hợp với tính đa hình và kế thừa trong lập trình hướng đối tượng.

Java trừu tượng hóa thông qua các lớp trừu tượng (Abstract class) và các giao diện (Interface). Trong bài này, tôi sẽ hướng dẫn các bạn tìm hiểu về lớp trừu tượng và phương thức trừu tượng, còn Interface tôi sẽ giới thiệu trong các bài sau.

**2. Phương thức trừu tượng (Abstract method) và lớp trừu tượng (Abstract class)**

### **2.1. Phương thức trừu tượng (Abstract method)**

Các phương thức chỉ có phần khai báo mà không có thân phương thức nằm trong cặp dấu {} và có một dấu chấm phẩy để kết thúc được gọi là phương thức trừu tượng. Để định nghĩa một phương thức là phương thức trừu tượng chúng ta sẽ sử dụng từ khóa abstract đứng trước tên phương thức.

Cú pháp khai báo phương thức trừu tượng:

**Cú pháp**

|  |  |
| --- | --- |
| 1 | [access\_modifier] abstract [kiểu\_trả\_về] [tên\_phương\_thức\_trừu\_tượng] [<đối\_số\_truyền\_vào>]; |

, trong đó:

* [access\_modifier] là phạm vi truy cập của phương thức trừu tượng. Phạm vi truy cập của phương thức trừu tượng tương tự như của các phương thức bình thường nhưng không được khai báo phạm vi truy cập là private, nếu để là private thì trình biên dịch sẽ báo lỗi.
* [kiểu\_trả\_về] là kiểu dữ liệu của phương thức.
* [tên\_phương\_thức\_trừu\_tượng] phải tuân theo quy tắc đặt tên phương thức (hàm) của Java.
* Phương thức này có thể có hoặc không có <đối\_số\_truyền\_vào>.

**Ví dụ**

|  |  |
| --- | --- |
| 1  2  3  4 | // khai báo 1 phương thức trừu tượng có tên là khaiBaoPhuongThucTruuTuong()  // phương thức này có phạm vi truy cập là public  // và không có đối số truyền vào  public abstract khaiBaoPhuongThucTruuTuong(); |

**Lưu ý:** Để sử dụng phương thức trừu tượng này, chúng ta cần phải ghi đè (override) nó trong lớp con kế thừa trực tiếp lớp khai báo phương thức này.

### **2.2. Lớp trừu tượng (Abstract class)**

* Lớp trừu tượng là lớp được khai báo với từ khóa abstract đứng trước tên của lớp.
* Nếu 1 lớp được khai báo là 1 lớp trừu tượng thì chúng ta không thể dùng trực tiếp nó để tạo ra đối tượng mà phải viết một lớp kế thừa của lớp trừu tượng đó.
* Lớp trừu tượng có thể có hoặc không có phương thức trừu tượng. Nếu một lớp có ít nhất 1 phương thức trừu tượng thì lớp đó phải được khai báo là lớp trừu tượng.
* Những lớp là lớp trừu tượng cũng không cần có phương thức khởi tạo.
* Một khi có một lớp nào đó kế thừa lớp trừu tượng thì lớp con đó bắt buộc phải override lại nội dung tất cả các phương thức trừu tượng có trong lớp đó.

Tóm lại, lớp trừu tượng là 1 lớp không thể khởi tạo đối tượng từ nó, nhưng nó lại ràng buộc các lớp con kế thừa trực tiếp nó phải có các phương thức trừu tượng của nó thông qua sự ghi đè (override) phương thức.

Để khai báo lớp trừu tượng, chúng ta sẽ sử dụng cú pháp sau:

**Cú pháp**

|  |  |
| --- | --- |
| 1  2  3 | [access\_modifier] abstract class TenClass {    } |

**AbstractClassDemo**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package vidu;    public abstract class AbstractClassDemo {      public void hienThi() {          System.out.println("Đây là lớp trừu tượng");      }        // khai báo 1 phương thức abstract có tên là hienThi()      // phương thức này không có thân phương thức      // và có kiểu trả về là void      public abstract void show();  } |

**3. Ví dụ**

Ví dụ dưới đây sẽ minh họa cách sử dụng tính trừu tượng trong Java.

**Animal.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | package vidu;    public abstract class Animal {      private String tiengKeu;        public abstract void hienThiTiengKeu();  } |

**Dog.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package vidu;    public class Dog extends Animal {        @Override      public void hienThiTiengKeu() {          System.out.println("Gâu");      }    } |

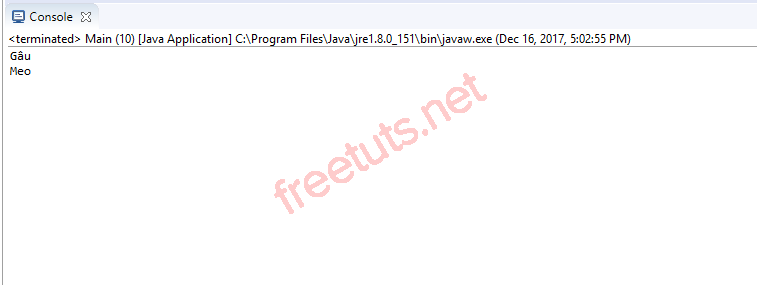
**Cat.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package vidu;    public class Cat extends Animal {        @Override      public void hienThiTiengKeu() {          System.out.println("Meo");      }    } |

**Main.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | package vidu;    public class Main {        public static void main(String[] args) {          Dog dog = new Dog();          dog.hienThiTiengKeu();            Cat cat = new Cat();          cat.hienThiTiengKeu();      }    } |

Kết quả sau khi biên dịch chương trình:



**Giải thích hoạt động của chương trình trên:**

Trong ví dụ trên, tôi tạo ra 3 lớp đối tượng là Animal, Dog và Cat, trong đó lớp Animal là lớp trừu tượng. Lớp Animal có một phương thức trừu tượng tên là hienThiTiengKeu() dùng để hiển thị tiếng kêu của từng loài động vật tương ứng. Lớp Dog và Cat là 2 lớp kế thừa của lớp Animal, 2 lớp này khi mới tạo ra sẽ có nội dung lớp như sau:

**Dog.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package vidu;    public class Dog extends Animal {        @Override      public void hienThiTiengKeu() {          // TODO Auto-generated method stub        }    } |

Các bạn thấy đoạn code trên khi lớp Dog kế thừa lớp Animal mới được tạo ra nó sẽ kế thừa lại phương thức trừu tượng của lớp cha thông qua sự ghi đè phương thức đó, cụ thể ở đây là phương thức hienThiTiengKeu(). Nếu chúng ta xóa bỏ phương thức này trong lớp Dog thì hệ thống sẽ báo lỗi, vì vậy chúng ta có thể rút ra kết luận: lớp cha có những phương thức trừu tượng nào thì lớp con cũng phải có phương thức trừu tượng đó (override) - đây chính là ý tưởng chính của tính trừu tượng.