Tính trừu tượng

Tính trừu tượng là một tính chất mà chỉ tập trung vào những tính năng của đối tượng và ẩn đi những thông tin không cần thiết. Tính chất này giúp bạn trọng tâm hơn vào những tính năng thay vì phải quan tâm tới cách mà nó được thực hiện. Ví dụ về tính trừu tượng trong đời sống:

Khi bạn đi rút tiền ở cây ATM thì bạn không cần quan tâm tới cách mà cây ATM hoạt động hay các thành phần có trong cây ATM, cái mà bạn quan tâm duy nhất đó là tính năng rút tiền. Trong trường hợp này các thông tin không cần thiết của cây ATM đã được ẩn đi, đó chính là tính trừu tượng.

Tương tự trong lập trình cũng vậy, khi sử dụng một đối tượng bạn chỉ cần quan tâm tới các phương thức cần thiết và chỉ cần biết phương thức đó được dùng để làm gì chứ không cần quan tâm tới source code và các phương thức khác. Để thực hiện tính trừu tượng trong Java thì bạn có thể sử dụng lớp trừ tượng (abstract class) và interface.

Lớp trừu tượng (abstract class)

Lớp trừu tượng là lớp được khai báo với từ khóa `abstract` và có một số đặc điểm sau:

1. Nếu một lớp được khai báo là lớp trừu tượng thì bạn không thể khởi tạo được đối tượng của lớp đó mà chỉ có thể khởi tạo được đối tượng của lớp con. Ví dụ:

```
2. //Khai báo lớp trừu tượng Animal
3. abstract class Animal{
4. }
5.
6. class Cat extends Animal{
7. }
8.
9. public class Entry {
10.         public static void main(String[] args) {
11.                 Animal a = new Cat();
12.         }
}
```

Chương trình trên sẽ không báo lỗi nhưng nếu bạn thay `Animal a = new Cat();` thành `Animal a = new Animal();` thì chương trình sẽ báo lỗi (do bạn không thể khởi tạo được đối tượng của lớp trừu tượng).

13. Lớp trừu tượng có thể có các phương thức trừu tượng. Phương thức trừu tượng là phương thức mà chỉ có phần khai báo, không có phần thân. Ví dụ:

```
14. // Khai báo lớp trừu tượng Animal
15. abstract class Animal {
```

```
        // Khai báo phương thức trừu tượng makeSound()
        public abstract void makeSound();
}
```

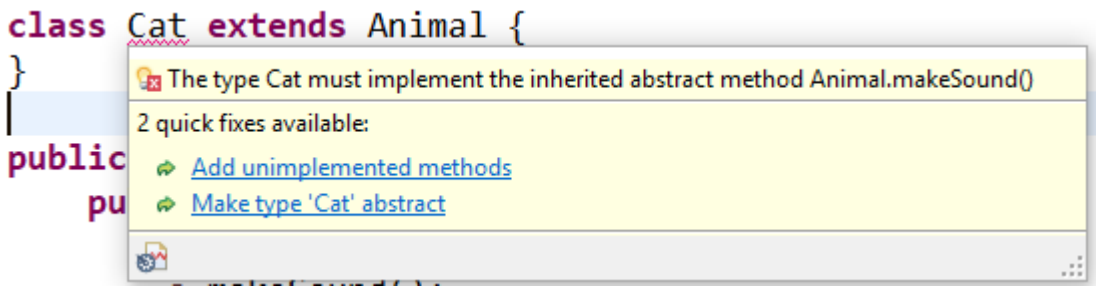
18. Nếu một lớp được kế thừa từ lớp trừu tượng thì lớp đó phải ghi đè tất cả các phương thức trừu tượng. Ví dụ:

```
abstract class Animal {
        public abstract void makeSound();
}

class Cat extends Animal {
        @Override
        public void makeSound() {
                System.out.println("meow meow");
        }
}

public class Entry {
        public static void main(String[] args) {
                Animal a = new Cat();
                a.makeSound();
        }
}
```

Kết quả khi chạy chương trình:

```
meow meow
```

Trong chương trình trên nếu lớp `Cat` không ghi đè phương thức `makeSound()` thì chương trình sẽ báo lỗi giống như sau:



Lớp trừu tượng thường được dùng để làm lớp cha cho các lớp có cùng bản chất. Việc sử dụng lớp trừu tượng và phương thức trừu tượng sẽ giúp bạn ẩn đi được cách cài đặt của phương thức. Tuy nhiên, bạn vẫn có thể khai báo các thuộc tính, các phương thức có phần thân nên việc sử dụng lớp trừu tượng sẽ không được coi là trừu tượng hoàn toàn. Để đạt được tính trừu tượng hoàn toàn bạn cần sử dụng tới `interface`.

interface

interface được dùng để lưu trữ các phương thức trừu tượng và các biến hằng số. Một số đặc điểm của interface:

1. Giống với lớp trừu tượng, bạn không thể khởi tạo được đối tượng của interface mà chỉ có thể khởi tạo được đối tượng của lớp được kế thừa từ interface.
2. Tất cả các phương thức trong interface đều được trình biên dịch hiểu là các phương thức trừu tượng và tất cả các biến trong interface đều được trình biên dịch hiểu là các hằng số. Ví dụ, nếu bạn khai báo `interface IAnimal` giống như sau:

```
3. interface IAnimal{
4.   int N = 4;
5.   void move();
6.   void sound();
```

```
}
```

Thì trình biên dịch sẽ hiểu giống như sau:

```
interface IAnimal{
  public static final int N = 4;
  public abstract void move();
  public abstract void sound();
}
```

7. Một lớp có thể kế thừa nhiều interface. Như bạn đã biết, Java là ngôn ngữ được thiết kế với mục đích đơn giản nên không hỗ trợ đa kế thừa với `class`, nhưng do bản chất `interface` chỉ chứa các phương thức rỗng nên Java cho phép một lớp kế thừa nhiều `interface`. Ví dụ:

```
8. interface IFlyable {
9.   void fly();
10. }
11.
12. interface IEatable {
13.         void eat();
14. }
15.
```

```
class Bird implements IFlyable, IEatable {
        @Override
        public void fly() {
                System.out.println("Bird flying");
        }

        @Override
        public void eat() {
                System.out.println("Bird eats");
        }
}

public class Entry {
        public static void main(String[] args) {
                Bird bird = new Bird();
                bird.eat();
                bird.fly();
        }
}
```

Kết quả khi chạy chương trình:

```
Bird eats
Bird flying
```

Lưu ý: để kế thừa interface bạn dùng từ khóa `implements` thay vì `extends`.

Khi nhìn vào interface thì thứ duy nhất mà bạn nhìn thấy đó là các phương thức trừu tượng (các tính năng), do đó sử dụng interface được coi là trừu tượng hoàn toàn. Trong thực tế, khi đi làm bạn sử dụng tới interface rất nhiều, với mỗi lớp bạn thường tạo ra 1 interface riêng để thể hiện các tính năng của lớp đó và giao tiếp với các đối tượng thông qua interface. Ví dụ lớp `Customer` sẽ có `interface ICustomer`, lớp `Employee` sẽ có interface là `IEmployee`, ...