Interface trong Java

Một Interface trong Java là một blueprint của một lớp. Nó có các hằng tĩnh và chỉ có các phương thức trừu tượng. Interface là một kỹ thuật để thu được tình trừu tượng hoàn toàn và đa kế thừa trong Java. Interface trong Java cũng biểu diễn mối quan hệ IS-A. Nó không thể được thuyết minh giống như lớp trừu tượng.

Ghi chú: Java Compiler thêm từ khóa public và abstract trước phương thức của interface và các từ khóa public, static và final trước các thành viên dữ liệu.

Nói cách khác, các trường của Interface là public, static và final theo mặc định và các phương thức là public và abstract.

Một Interface trong Java là một tập hợp các phương thức trừu tượng (abstract). Một class triển khai một interface, do đó kế thừa các phương thức abstract của interface.

Một interface không phải là một lớp. Viết một interface giống như viết một lớp, nhưng chúng có 2 định nghĩa khác nhau. Một lớp mô tả các thuộc tính và hành vi của một đối tượng. Một interface chứa các hành vi mà một class triển khai.

Trừ khi một lớp triển khai interface là lớp trừu tượng abstract, còn lại tất cả các phương thức của interface cần được định nghĩa trong class.

Một interface tương tự với một class bởi những điểm sau đây:

- Một interface có thể bao gồm bất cứ lượng phương thức nào.
- Một interface được viết trong một file với định dạng **.java**, với tên của interface cùng với tên của file.
- Bytecode của interface xuất hiện trong một **.class** file.
- Interface xuất hiện trong package, những bytecode file tương ứng phải ở trong cấu trúc thư mục có cùng tên package.

Mặc dù vậy, một interface khác với một class ở một số điểm sau đây, bao gồm:

- Bạn không thể khởi tạo một interface.
- Một interface không chứa bất cứ hàm contructor nào.

- Tất cả các phương thức của interface đều là abstract.
- Một interface không thể chứa một trường nào trừ các trường vừa static và final.
- Một interface không thể kế thừa từ lớp, nó được triển khai bởi một lớp.
- Một interface có thể kế thừa từ nhiều interface khác.

Ví dụ đơn giản về Interface trong Java

Trong ví dụ này, Printable Interface chỉ có một phương thức, trình triển khai của nó được cung cấp bởi lớp A.

```
interface printable{
void print();
}

class A6 implements printable{
public void print(){System.out.println("Hello");}

public static void main(String args[]){
A6 obj = new A6();
obj.print();
 }
}
```

Khi ghi đè các phương thức được định nghĩa trong interface, có một số qui tắc sau:

- Các checked exception không nên được khai báo trong phương thức implements, thay vào đó nó nên được khai báo trong phương thức interface hoặc các lớp phụ được khai báo bởi phương thức interface.
- Signature (ký số) của phương thức interface và kiểu trả về nên được duy trì khi ghi đè phương thức (overriding method).
- Một lớp triển khai chính nó có thể là abstract và vì thế các phương thức interface không cần được triển khai.

Khi triển khai interface, có vài quy tắc sau:

- Một lớp có thể triển khai một hoặc nhiều interface tại một thời điểm.
- Một lớp chỉ có thể kế thừa một lớp khác, nhưng được triển khai nhiều interface.
- Một interface có thể kế thừa từ một interface khác, tương tự cách một lớp có thể kế thừa lớp khác.

Đa kế thừa trong Java bởi Interface

Nếu một lớp triển khai đa kế thừa, hoặc một Interface kế thừa từ nhiều Interface thì đó là đa kế thừa.

```
interface Printable{
void print();
}

interface Showable{
void show();
}

class A7 implements Printable,Showable{

public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
A7 obj = new A7();
obj.print();
obj.show();
 }
}
```

Câu hỏi: Đa kế thừa không được hỗ trợ thông qua lớp trong Java nhưng là có thể bởi Interface, tại sao?

Như đa thảo luận trong chương về tính kế thừa, đa kế thừa không được hỗ trợ thông qua lớp. Nhưng nó được hỗ trợ bởi Interface bởi vì không có tính lưỡng nghĩa khi trình triển khai được cung cấp bởi lớp Implementation. Ví dụ:

```
interface Printable{
void print();
}
interface Showable{
void print();
}

class TestTnterface1 implements Printable,Showable{
public void print(){System.out.println("Hello");}
public static void main(String args[]){
TestTnterface1 obj = new TestTnterface1();
obj.print();
 }
}
```

Trong ví dụ trên, Printable và Showable interface có cùng các phương thức nhưng trình triển khai của nó được cung cấp bởi lớp TestInterface1, vì thế không có tính lưỡng nghĩa ở đây.

Kế thừa Interface trong Java

Một lớp triển khai Interface nhưng một Interface kế thừa từ Interface khác.

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
class Testinterface2 implements Showable{

public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}
```

```
public static void main(String args[]){
Testinterface2 obj = new Testinterface2();
obj.print();
obj.show();
 }
}
```

Marker (hay Tagging) Interface trong Java là gì?

Đó là một Interface mà không có thành viên nào. Ví dụ: Serializable, Cloneable, Remote, … Chúng được sử dụng để cung cấp một số thông tin thiết yếu tới JVM để mà JVM có thể thực hiện một số hoạt động hữu ích.

```
//Cach Serializable interface duoc viet?
public interface Serializable{
}
```

Có hai mục đích thiết kế chủ yếu của tagging interface là:

Tạo một cha chung: Như với EventListener interface, mà được kế thừa bởi hàng tá các interface khác trong Java API, bạn có thể sử dụng một tagging interface để tạo một cha chung cho một nhóm interface. Ví dụ, khi một interface kế thừa EventListener, thì JVM biết rằng interface cụ thể này đang được sử dụng trong một event.

Thêm một kiểu dữ liệu tới một class: Đó là khái niệm *tagging*. Một class mà triển khai một tagging interface không cần định nghĩa bất kỳ phương thức nào, nhưng class trở thành một kiểu interface thông qua tính đa hình (polymorphism).

Lồng Interface trong Java

Ghi chú: Một Interface có thể có Interface khác, đó là lồng Interface. Chúng ta sẽ tìm hiểu chi tiết trong chương về Lồng các lớp trong Java. Ví dụ:

```
interface printable{
 void print();
 interface MessagePrintable{
   void msg();
```

```
 }
}
```