|  |
| --- |
| Nguyễn Trung Thành - 20176874 |

**Câu hỏi:**

1. Tại sao chúng ta không thể override các phương thức thuộc về lớp (i.e. phương thức static trong Java)? Các bạn rút ra điều gì từ những phân tích đó?
2. So sánh Abstract Class và Interface. Java đã giải quyết các vấn đề về đa kế thừa (như trùng tên trong lớp cha, hoặc kế thừa lặp lại) như thế nào khi sử dụng Interface? Lấy ví dụ và phân tích (có thể dùng biểu đồ UML và mã nguồn để minh hoạ).

**Trả lời:**

**Câu 1:**

* Định nghĩa về Override: một lớp con có thể định nghĩa một phương thức có cùng tên với phương thức của lớp cha. Phương thức của lớp con này có cùng tên và chữ ký thì được gọi là Overriding.
* Xét các phương thức thuộc về lớp (ví dụ: phương thức static trong Java). Ví dụ về phương thức static thì việc gọi đến 1 phương thức tĩnh này luôn được xác định ngay tại thời điểm biên dịch (compile-time) chứ không phải trong runtime => Không xảy ra tính đa hình với các phương thức tĩnh được vì Method overriding được sử dụng để thu được tính đa hình tại runtime.  
  Hơn nữa, các phương thức này thuộc về lớp chứ không phải là của đối tượng, nên khi truy cập vào nó ta dùng chính lớp đó để truy cập.
* Rút ra kết luận:  
  + Đối với các phương thuộc về lớp ta không thể thực hiện override được. Nhưng, các phương thức như static có thể thực hiện overloading được (nghĩa là chúng ta có cùng tên với phương thức nhưng khác nhau về chữ ký).

+ Tính đa hình trong Java: gồm có đa hình lúc biên dịch (compile) và đa hình lúc thực thi (runtime).

* Đa hình lúc runtime: là quá trình mà phương thức đã được ghi đè trong thời gian thực thi chạy chương trình.

Method overriding được sử dụng để thu được tính đa hình tại runtime.

* Đa hình lúc compile:

Method overloading được sử dụng để thu được tính đa hình lúc compile.

+ Đối với các phương thức thuộc về lớp (như static) ta không sử dụng được từ khoá như this / super. Nhưng các từ khoá this /super có thể sử dụng trong Overload / Override.

**Câu 2:**

* Định nghĩa:

+ Abstract class: là một lớp (đặc biệt) mà chúng ta không thể tạo ra các đối tượng từ nó. Abstract class thường được xử dụng để định nghĩa “các khái niệm chung”.

+ Interface: là một bản thiết kế của một lớp (không phải là một class). Interface không chứa bất kỳ một method nào nhưng nó định nghĩa cấu trúc cho các class sử dụng nó.

* Phân biệt giữa Abstract Class và Interface:

|  |  |  |
| --- | --- | --- |
|  | Abstract Class | Interface |
| Từ khoá | abstract | interface |
| Class | Đúng | Không phải class |
| Tính trừu tượng | 0 tới 100% | 100% |
| Đa kế thừa | Không hỗ trợ tính đa kế thừa | Có hỗ trợ |
| Constructors | Có chứa | Không chứa |
| Data Member | Có chứa | Không chứa |
| Default implementation | Chứa cả incomplete (abstract) và complete member | Chỉ chứa các incomplete member |
| Access modifiers | Có thể xác định modifier | Mọi phương thức mặc định đều là public |

* Vấn đề về đa kế thừa trong JAVA.

+ Xét ví dụ sau: (Không sử dụng interface)

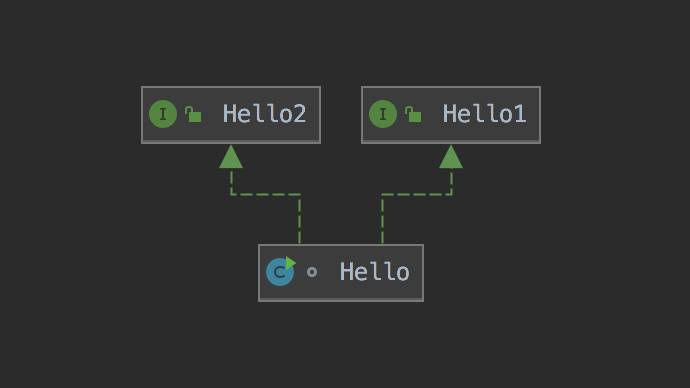
// Lớp cha thứ nhất  
public class Hello1 {  
 void hello() {  
 System.*out*.println("こんにちは");  
 }  
}

// Lớp cha thứ hai  
public class Hello2 {  
 void hello() {  
 System.*out*.println("Hello");  
 }  
}

class Hello extends Hello1, Hello2 {  
 public static void main(String args[]) {  
 Hello test = new Hello();  
 test.hello();  
 }  
}

Khi này, vấn đề xảy ra khi gọi method hello() trong class Hello sẽ không biết gọi method hello trong class Hello1 hay class Hello2 => Dẫn tới lỗi không thể thực hiện được.

* Sử dụng Interface: giúp giải quyết vấn đề trùng hợp tên method

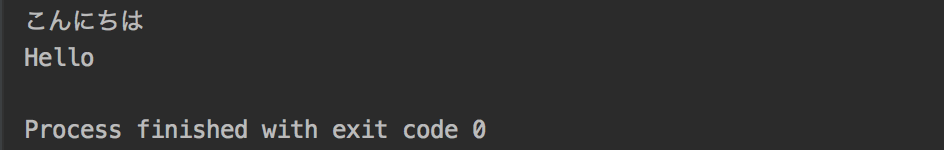


public interface Hello1 {  
 default void hello() {  
 System.*out*.println("こんにちは");  
 }  
}

public interface Hello2 {  
 default void hello() {  
 System.*out*.println("Hello");  
 }  
}

class Hello implements Hello1, Hello2 {  
  
 // Overriding default show method  
 public void hello()  
 {  
 // method of Hello1 interface  
 Hello1.super.hello();  
  
 // method of Hello2 interface  
 Hello2.super.hello();  
 }  
  
 public static void main(String args[]) {  
 Hello test = new Hello();  
 test.hello();  
 }  
}

* Kết quả chạy chương trình:



* Nhưng trong trường hợp nếu chúng ta loại bỏ “Overriding default show method” của class Hello sẽ xảy ra lỗi.
* Khi sử dụng Interface mà trong trường hợp cả 2 interface đều có cùng default method giống nhau về chữ ký thì implementing class cần chỉ ra rõ default method nào được sử dụng hoặc cần override default method.
* Kết luận: Sử dụng interface giúp chúng ta giải quyết các vấn đề về đa kế thừa (như trùng tên trong lớp cha, hoặc kế thừa lặp lại).