Nguyen Thanh Tam @nguyen.thanh.tam Follow
★ 473 16 23

Published Nov 21st, 2016 8:30 AM - 8 min read

Interface trong Java 8, giới thiệu Default Method và Static Method

Java Interface java 8

Một trong những thay đổi lớn nhất trong Java 8 là khái niệm về interface. Như chúng ta đã biết ở những phiên bản Java trước, interface chỉ cho phép chúng ta khai báo các phương thức bên trong nó. Nhưng trong Java 8 chúng ta sẽ có thêm 2 khái niệm mới

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

Thiết kế interface luôn là một công việc rất khó khăn, bởi vì khi chúng ta thay đổi các phương thức bên trong interface nó đòi hỏi phải thay đổi tất cả các class được implements từ nó. Một khi số lượng các class được implements từ interface phát triển nhiều lên thì đến mức độ nào đó interface có thể không mở rộng được nữa. Đây là lý do tại sao khi thiết kế một ứng dụng, hầu hết các framework cung cấp một base class, sau đó chúng ta sẽ extends và override lên các phương thức phù hợp với ứng dụng đang thực hiện. Bây giờ chúng ta sẽ tìm hiểu về phương thức default và static của interface, xem nó hoạt động thế nào.

Phương thức Default

Để tạo một phương thức default trong interface, chúng ta sẽ sử dụng từ khóa “default”. Ví dụ:

```
package com.example.java8;

public interface FirstInterface {

    void firstMethod(String string);

    default void log(String string){
```


Phương thức log(String str) chính là phương thức default của FirstInterface. Khi một class được implements từ FirstInterface nó **không bắt buộc** phải implement phương thức default. Tính năng này sẽ giúp chúng ta mở rộng các phương thức bổ sung phát sinh sau này mà không ảnh hưởng đến các class liên quan, chúng ta chỉ cần viết thêm các phương thức default trong interface.

Bây giờ chúng ta sẽ xem tiếp một ví dụ khác:

```
package com.example.java8;

public interface SecondInterface {

    void secondMethod();

    default void log(String str){
        System.out.println("This method is default implementation" + str);
    }

}
```

Như chúng ta biết rằng, Java không cho phép đa thừa kế đối với class bởi vì trình biên

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

+4

interface FirstInterface và SecondInterface, trình biên dịch sẽ ko biết chọn phương thức nào để thực hiện. Đa kế thừa là một việc rất bình thường trong Java, chúng ta thường bắt gặp vấn đề này trong các class Java core cũng như hầu hết các ứng dụng enterprise và framework. Để chắc chắn rằng, vấn đề này sẽ không xảy ra với interface, class phải implement các phương thức common default. Vì thế, nếu class được implements từ cả 2 interface trên, thì nó phải implement phương thức log() để trình biên dịch không ném ra lỗi.

```
package com.example.java8;
```

```java
public class MyClass implements FirstInterface, SecondInterface {

    @Override
    public void firstMethod() {
    }

    @Override
    public void secondMethod() {
    }

    @Override
    public void log(String string){
        System.out.println("MyClass logging::" + string);
    }
}
```


=> Những đặc điểm quan trọng về phương thức default trong interface:

1. Phương thức default giúp chúng ta mở rộng interface mà không phải lo ngại phá vỡ các class được implements từ nó.

2. Phương thức default giúp chúng ta tránh dùng các class tiện ích, ví dụ như tất cả phương thức của class Collections có thể được cung cấp ngay bên trong interface của nó.

3. Phương thức default giúp chúng ta tháo gỡ các class cơ sở (base class), chúng ta có thể tạo phương thức default và trong class được implement có thể chọn phương thức để override.

4. Một trong những lý do xuất hiện của phương thức default là để nâng cấp Collection API trong Java 8 hỗ trợ cho Lambda Expression.

5. Nếu bất kỳ class nào kế thừa những phương thức default giống nhau, thì nó sẽ không còn hiệu lực. Một điều tương tự, một phương thức default sẽ không thể override một phương thức từ java.lang.Object. Lý do rất đơn giản là bởi vì Object là base class của tất cả các class trong Java. Vì vậy nếu chúng ta có các phương thức

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

lý do tại sao chúng ta sẽ không có bất cứ phương thức default nào override các phương thức của class Object.

+4

6. Phương thức default cũng có thể được gọi là phương thức Defender (Defender

Methods) hay là phương thức Virtual mở rộng (Virtual extension methods)

Phương thức Static

Phương thức static cũng giống như phương thức default ngoại trừ việc nó không thể được override chúng trong class được implements.

Hãy xem ví dụ dưới đây:

```
package com.example.java8;

public interface ThirdInterface {

    default void print(String string) {
        if (!isNull(string))
            System.out.println("ThirdInterface Print::" + string);
    }

    static boolean isNull(String string) {
        System.out.println("Interface Null Check");
    }
}
```


Bây giờ sẽ xem class được implements có phương thức isNull()

```
package com.example.java8;

public class ThirdImpl implements ThirdInterface {

    public boolean isNull(String string) {
        System.out.println("Impl Null Check");

        return string == null ? true : false;
    }

    public static void main(String args[]){
        ThirdImpl obj = new ThirdImpl();
        obj.print("");
        obj.isNull("abc");
    }
}
```

Phương thức *isNull(String string)* là một phương thức đơn giản, nó không override phương thức của interface. Ví dụ nếu chúng ta thêm annotation @Override cho phương thức isNull(), trình biên dịch sẽ báo lỗi. Bây giờ chúng ta sẽ chạy ứng dụng và xem kết

```
Interface Null Check
Impl Null Check
```

Nếu chúng ta chuyển static thành default, thì kết quả như sau:

```
Impl Null Check
ThirdInterface Print::
Impl Null Check
```

Phương thức static chỉ hiển thị trong phương thức của interface, nếu chúng ta xóa phương thức isNull() trong class ThirdImpl, chúng ta sẽ không thể sử dụng nó cho object của ThirdImpl. Tuy nhiên, giống như các phương thức static khác, chúng ta có thể sử dụng phương thức static của interface thông qua tên của class. Ví dụ sau đây là cách sử dụng hợp lệ:

```
boolean result = MyData.isNull("abc");
```

`Những đặc điểm quan trọng về phương thức static trong interface:`

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

+4

2. Phương thức static rất hữu ích trong việc cung cấp các phương thức tiện ích, ví dụ như là kiểm tra null, sắp xếp tập hợp, v.v...
3. Phương thức static giúp chúng ta bảo mật, không cho phép class implements từ nó có thể override
4. Chúng ta không thể định nghĩa phương thức static của các phương thức thuộc class Object, chúng ta sẽ gặp lỗi "*This static method cannot hide the instance method from Object*". Điều này không cho phép trong Java, khi Object là base class cho tất cả các class và chúng ta không thể có một phương thức static và một phương thức khác cùng định dạng
5. Chúng ta có thể sử dụng phương thức static để bỏ đi những những phương thức dạng tiện ích như là Collections và làm cho tất cả các phương thức có thể liên lạc với interface, chúng ta sẽ dễ dàng tìm thấy và sử dụng những phương thức đó.

Functional Interfaces

Functional Interface là cách gọi khác của SAM(Single Abstract Method) interface, tức là những interface chỉ có duy nhất một method, ví dụ như Runnable, Callable, Comparator, ActionListener, ... Từ Java8, một annotation mới @FunctionalInterface cũng đã được giới thiệu để đánh dấu một interface nào đó là Functional Interface.

Related

Java String (P1)
Tran Anh Vu
5 min read
994 3 0 3

Interface in Java
Phan Man
1 min read
352 1 0 6

Sự tiến hóa của Interface trong Java kể từ JDK 1.7 và...
Do Ha Long
6 min read
244 1 2 4

Giới thiệu Default method trong Java 8
Nguyen Minh Tien
6 min read
1184 1 0 0

More from Nguyen Thanh Tam

String trong Java 11 có gì hot?

Tại sao phải dùng BigDecimal khi tính toán về...

Ngăn ngừa lãng phí bộ nhớ trong Java Collections như...

Sử dụng flyway library cho việc setup integration test

VIBLO Posts Questions Discussions Search Viblo Sign In/Sign up

Comments

Login to comment