Java 8 - Default method & Static method

1. Giới thiệu

Một trong những thay đổi lớn nhất trong Java 8 là khái niệm về interface. Như chúng ta đã biết từ Java 7 trở về 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 đối với interface là phương thức default (default methods) và phương thức static (static methods).

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 class cơ sở (base class), sau đó chúng ta sẽ mở rộng (extends) và ghi đè (override) lên các phương thức phù hợp với ứng dụng đang thực hiện.

2. Phương thức default

Để tạo một phương thức default trong interface, chúng ta sẽ sử dụng từ khóa "default".

Ví dụ: Interface1.java

```
private interface Defaulable {
    void method1(String str);
    // Interfaces now allow default methods, the implementer may or may not implement (override)
them.
    default String notRequired() {
        return "Default implementation";
    }
}

private static class DefaultableImpl implements Defaulable {
    @Override
    public void method1(String str) {
    }
}
```

```
private static class OverridableImpl implements Defaulable {
    @Override
    public void method1(String str) {
    }

    @Override
    public String notRequired() {
        return "Overridden implementation";
    }
}
```

Giải thích: Chúng ta xây dựng một interface là Defaulable với hai phương thức:

- void method1
- default String notRequired
  Khi xây dựng các lớp implements tới lớp **Defaulable**, chúng ta có thể không cần thực hiện Override phương thức **notRequired** (lớp **DefaultableImpl**) hoặc Override phương thức **notRequired** (lớp **OverridableImpl**)

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 của class Object được định nghĩa là phương thức default trong interface, nó sẽ không dùng được bởi vì các phương thức của Object luôn luôn được sử dụng. Đây lý do tại sao chúng ta sẽ không có bất cứ phương thức default nào override các phương thức của class Object.
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)

Chú ý:

- Phương thức default được thực hiện trên JVM mang lại hiệu quả và được hỗ trợ các chỉ dẫn byte code cho phương pháp gọi. Phương thức default cho phép các Java interface đã tồn

tại phát triển thêm mà không gây lỗi trong quá trình biên dịch. Ví dụ như bổ sung các phương thức vào interface **java.util.Collection: stream(), parallelStream(), forEach(), removeIf()**...
- Mặc dù vậy, phương thức default cũng cần được sử dụng một cách cẩn thận bởi nguyên nhân: Trước khi khai báo phương thức là default, cần chắc chắn là nó là cần thiết bị nó có thể gây ra sự nhập nhằng và biên dịch lỗi trong hệ thống phân cấp phức tạp.

3. 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.

Một tính năng thú vị cung cấp bởi Java 8 là interface có thể khai báo (và cung cấp implementation) các phương thức tĩnh. Dưới đây là một số ví dụ.

Ví dụ 1:

```
private interface DefaulableFactory {
    // Interfaces now allow static methods
    static Defaulable create( Supplier< Defaulable > supplier ) {
        return supplier.get();
    }
}
```

Main class:

```
public static void main( String[] args ) {
    Defaulable defaulable = DefaulableFactory.create( DefaultableImpl::new );
    System.out.println( defaulable.notRequired() );

    defaulable = DefaulableFactory.create( OverridableImpl::new );
    System.out.println( defaulable.notRequired() );
}
```

Kết quả:

```
Default implementation
Overridden implementation
```

Ví dụ 2:

`MyData.java`

```
public interface MyData {
    default void print(String str) {
        if (!isNull(str))
            System.out.println("MyData Print::" + str);
    }
    static boolean isNull(String str) {
        System.out.println("Interface Null Check");
        return str == null ? true : "".equals(str) ? true : false;
    }
}
```

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

`MyDataImpl.java`

```
public class MyDataImpl implements MyData {

    public boolean isNull(String str) {
        System.out.println("Impl Null Check");
        return str == null ? true : false;
    }

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

Phương thức `isNull(String str)` 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 quả:

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

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

```
Impl Null Check
MyData Print::
Impl Null Check
```

Phương thức static chỉ hiển thị trong phương thức của interface, nếu chúng ta xóa phương thức isNull() trong class `MyDataImpl`, chúng ta sẽ không thể sử dụng nó cho đối tượng (object) của `MyDataImpl`. 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:

1. Phương thức static là một thành phần của interface, chúng ta có thể sử dụng nó trong class được implements từ nó.
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 ...
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.

Revision #1
Created 29 September 2019 16:28:25 by Laptrinh.vn
Updated 19 October 2019 16:55:59 by Laptrinh.vn