Bài 53: Tính năng mới Java 8+

- ✓ Các phương thức default
- ✓ Các phương thức static
- ✓ Các phương thức private
- ✓ Ví dụ minh họa

Default methods

- Các phương thức default là các phương thức được khai báo với keyword default có phần thân hoàn chỉnh trong interface
- Các phương thức default được thêm vào Java từ phiên bản 8 nhằm bổ sung them tính năng cho interface
- Trước Java 8, các interface không thể chứa các phương thức được thực thi cụ thể. Tức interface chỉ có thể chứa abstract methods
- Việc thêm các default method sẽ không làm ảnh hưởng đến code cũ

Default methods

- Lý do là vì default method chứa phần thân hoàn chỉnh nên sẽ không bắt buộc các lớp implements nó phải override
- Các lớp implements interface có hai tùy chọn: override lại phương thức default hoặc không
- Ví dụ ban đầu ta có interface:

```
/**
 * interface chứa các phương thức có thể tính toán số học
 * hai số nguyên a, b
 */
public interface Calculable {
    int add(int a, int b); // cộng hai số

    int sub(int a, int b); // trừ hai số
}
```

Ví dụ

- Sau đó ta bổ sung thêm tính năng cho interface bằng default methods:

```
/**
 * interface chứa các phương thức có thể tính toán số học
 * hai số nguyên a, b
 */
public interface Calculable {
    int add(int a, int b); // cộng hai số

    int sub(int a, int b); // trừ hai số

    default float div(int a, int b) { // chia
        return 1.0f * a / b;
    }

    default int mul(int a, int b) { // nhân
        return a * b;
    }
}
```

Ví dụ

- Lớp implements interface này có thể override lại default methods:

```
class Calculator implements Calculable {

    @Override
    public float div(int a, int b) { // override lại
        if (b == 0) {
            System.out.println("Mẫu số phải khác 0");
            return Float.NaN;
        }
        return a * 1.0f / b;
    }
```

Kế thừa interface

- Khi kế thừa interface có phương thức default ta có ba tùy chọn:
  - Bỏ qua các phương thức default của interface cha
  - Khai báo lại các phương thức default của interface cha. Điều này làm phương thức default của interface cha trở thành phương thức abstract trong interface con
  - Định nghĩa lại phương thức default của interface cha. Điều này làm cho interface con override phương thức default của interface cha

Kế thừa interface

➢ Ví dụ:

```
public interface Calculable {
    int add(int a, int b); // cộng hai số

    int sub(int a, int b); // trừ hai số

    default float div(int a, int b) { // chia
        return 0;
    }

    default int mul(int a, int b) { // nhân
        return 1;
    }
}

interface OtherCalculable extends Calculable {
    @Override
    default float div(int a, int b) { // override
        return a * 1.0f / b;
    }

    int mul(int a, int b); // mul trở thành abstract method
}
```

Implements interface

- Khi một lớp implements nhiều interface có cùng phương thức default với cùng tham số, kiểu trả về thì khi sử dụng sẽ phải chỉ rõ default methods đó của interface nào
- Ví dụ có hai interface sau:

```
public interface Interface1 {
    default void show(String msg) {
        System.out.println("Gọi show() của Interface1: " + msg);
    }
}

interface Interface2 {
    default void show(String msg) {
        System.out.println("Gọi show() của Interface2: " + msg);
    }
}
```

Implements interface

➢ Khi lớp implements interface:

```
class Example implements Interface1, Interface2 {
    public static void main(String[] args) {
        Example example = new Example();
        var msg = "I'm a developer";
        example.show(msg);
    }

    @Override
    public void show(String msg) {
        Interface1.super.show(msg);
        Interface2.super.show(msg);
        System.out.println("Gọi show() của lớp Example: " + msg);
    }
}
```

Static methods

- Các phương thức static là các phương thức cố định đã định nghĩa sẵn thuộc sở hữu của interface
- Static methods được bổ sung vào interface từ Java 8
- Nếu interface có chứa một hành vi cố định nào đó và không muốn thay đổi ở các interface con hoặc lớp implements nó thì ta sử dụng static methods
- Static methods thường được sử dụng làm phương thức bổ trợ hoặc tiện ích

Ví dụ

- Interface sau định nghĩa hai static methods:

```
public interface Splitable {
    // phương thức tách và trả về mảng các từ trong chuỗi
    // với dấu hiệu phân tách là vị trí có dấu cách
    static String[] splitByWhitespace(String src) {
        if (src == null) {
            return null;
        }
        return src.split("\\s+");
    }

    // phương thức tách lấy tên của một người từ String cho trước
    static String getFirstName(String fullName) {
        if (fullName == null) {
            return null;
        }
        var index = fullName.indexOf(" ");
        return fullName.substring(0, index);
    }

    // một phương thức abstract
    String[] split(String src, String regex);
}
```

Private methods

- Là thành phần được thêm vào interface từ Java 9
- Chỉ sử dụng được nội bộ trong interface
- Làm tăng tính đóng, tái sử dụng code, làm helper methods
- Quy tắc sử dụng private methods:
  - Private methods không thể abstract
  - Chỉ được sử dụng nội bộ trong interface bởi các phương thức non-abstract
  - Phương thức private non-static không thể được sử dụng trong các phương thức static

Ví dụ

➢ Ví dụ các static method trong interface:

```
public interface Splitable {
    // phương thức private static
    private static String[] splitByWhitespace(String src) {
        if (src == null) {
            return null;
        }
        return src.split("\\s+");
    }

    // phương thức tách lấy tên của một người từ String cho trước
    static String getFirstName(String fullName) {
        if (fullName == null) {
            return null;
        }
        // gọi phương thức private static để tách các từ trong tên
        var words = splitByWhitespace(fullName);
        return words[0]; // giả định tên đứng đầu tiên trong họ tên
    }

}
```

Minh họa

- Thực hiện trong công cụ lập trình

Nội dung tiếp theo

Tự định nghĩa interface