Cập nhật tháng 8 năm 2024

[Bài đọc] Interface

Trong Java, **Interface** là một cấu trúc được sử dụng để định nghĩa các hành vi (method signatures) mà các lớp (classes) cụ thể phải tuân theo khi thực hiện (implement) nó. Một interface chỉ chứa các **phương thức trừu tượng** hoặc **các hằng số**. Nó hỗ trợ tính **đa kế thừa** trong Java, vì một lớp có thể triển khai (implement) nhiều interface cùng một lúc.

Đặc điểm của Interface

1. **Không thể chứa code thực thi (trước Java 8):**
   - Trước Java 8, các phương thức trong interface luôn là trừu tượng (abstract) và không thể chứa logic thực thi.
   - Từ Java 8, interface hỗ trợ thêm **phương thức mặc định (default method)** và **phương thức tĩnh (static method)** có phần thân.
2. **Các phương thức luôn là public và abstract (mặc định):**
   - Khi định nghĩa phương thức trong interface, bạn không cần ghi rõ từ khóa public abstract vì chúng được hiểu mặc định.
3. **Các thuộc tính luôn là public, static và final (mặc định):**
   - Các biến trong interface phải là hằng số (final) và có giá trị cố định ngay từ khi khai báo
4. **Không có constructor:**
   - Interface không thể khởi tạo đối tượng trực tiếp vì không chứa constructor
5. **Hỗ trợ đa kế thừa:**
   - Một lớp có thể triển khai nhiều interface bằng cách sử dụng từ khóa implements.
6. **Kế thừa giữa các interface:**
   - Một interface có thể kế thừa một hoặc nhiều interface khác bằng từ khóa extends.

Cú pháp cơ bản:

```java
interface InterfaceName {
    // Các hằng số
    int CONSTANT = 100;

    // Phương thức trừu tượng
    void abstractMethod();

    // Phương thức mặc định (Java 8+)
    default void defaultMethod() {
        System.out.println("Default method in interface.");
    }

    // Phương thức tĩnh (Java 8+)
    static void staticMethod() {
        System.out.println("Static method in interface.");
    }
}
```

Ví dụ sử dụng Interface:

1. **Định nghĩa và triển khai Interface**

```java
interface Animal {
    void eat();
    void sleep();
}

class Dog implements Animal {
    @Override
    public void eat() {
        System.out.println("Dog is eating.");
    }

    @Override
    public void sleep() {
        System.out.println("Dog is sleeping.");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        myDog.eat();
        myDog.sleep();
    }
}
```

2. **Interface với các phương thức mặc định và tĩnh**

```java
interface Vehicle {
    void move();

    default void stop() {
        System.out.println("Vehicle stopped.");
    }

    static void service() {
        System.out.println("Vehicle is being serviced.");
    }
}

class Car implements Vehicle {
    @Override
    public void move() {
        System.out.println("Car is moving.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car();
        car.move();
        car.stop();

        Vehicle.service(); // Gọi phương thức tĩnh
    }
}
```

3. Một lớp triển khai nhiều Interface

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

interface Scannable {
    void scan();
}

class PrinterScanner implements Printable, Scannable {
    @Override
    public void print() {
        System.out.println("Printing document...");
    }

    @Override
    public void scan() {
        System.out.println("Scanning document...");
    }
}

public class Main {
    public static void main(String[] args) {
        PrinterScanner device = new PrinterScanner();
        device.print();
        device.scan();
    }
}
```

Các thành phần chính của Interface:

1. **Phương thức trừu tượng (Abstract Method):**
   - Là các phương thức chỉ có phần khai báo, không có thân.
   - Tất cả các lớp triển khai interface đều phải cung cấp phần thân cho các phương thức này.
2. **Hằng số (Constants):**
   - Các thuộc tính trong interface mặc định là public static final.
   - Không thể thay đổi giá trị sau khi được gán.
3. **Phương thức mặc định (Default Method):**
   - Được thêm từ Java 8.
   - Có thể cung cấp một logic thực thi mặc định cho phương thức trong interface.
4. **Phương thức tĩnh (Static Method):**
   - Được thêm từ Java 8.
   - Có thể được gọi trực tiếp từ interface mà không cần thông qua lớp triển khai.
5. **Phương thức riêng tư (Private Method):**
   - Được thêm từ Java 9.

- Dùng để hỗ trợ các phương thức mặc định hoặc tĩnh bên trong interface mà không lộ ra bên ngoài.

Lợi ích của Interface

1. **Hỗ trợ đa kế thừa:**
   - Vì Java không hỗ trợ đa kế thừa giữa các lớp (class), interface cung cấp một cách giải quyết để đạt được mục tiêu này.
2. **Tính linh hoạt cao:**
   - Interface tách biệt hoàn toàn phần định nghĩa (signature) và phần thực thi, giúp thay đổi hoặc mở rộng dễ dàng.
3. **Hỗ trợ tính đa hình (Polymorphism):**
   - Các lớp triển khai interface có thể được sử dụng thông qua tham chiếu kiểu interface.
4. **Giúp đảm bảo tính trừu tượng:**
   - Interface chỉ tập trung vào việc định nghĩa hành vi, không can thiệp vào cách thực hiện.

Khi nào nên sử dụng Interface?

- Khi bạn muốn định nghĩa một **hợp đồng chung** cho các lớp không liên quan đến nhau.
- Khi bạn cần đảm bảo rằng tất cả các lớp tuân theo cùng một bộ phương thức.
- Khi bạn muốn thực hiện đa kế thừa.

Link tài nguyên đọc thêm: https://www.geeksforgeeks.org/interfaces-in-java/