Cập nhật tháng 8 năm 2024

[Bài Đọc] FunctionalInterface và biểu thức Lambda

1. **Functional Inteface**

- **Khái niệm:**
  - **Functional Interface** là một interface trong Java chỉ chứa **một phương thức trừu tượng duy nhất** (Single Abstract Method - SAM)
  - Từ Java 8, Functional Interface được sử dụng để hỗ trợ **biểu thức lambda**, giúp lập trình Java trở nên ngắn gọn và dễ đọc hơn
  - Được đánh dấu bằng annotation @FunctionalInterface (không bắt buộc nhưng nên dùng để đảm bảo interface tuân theo quy tắc của Functional Interface)
- **Đặc điểm chính:**
  - Chỉ được phép có **một phương thức trừu tượng.**
  - Có thể chứa các phương thức mặc định (default) và phương thức tĩnh (static).
  - Một số Functional Interface được Java cung cấp sẵn, như:
    - Runnable
    - Callable
    - Comparator
    - Consumer, Supplier, Predicate, Function (từ thư viện java.util.function)
- **Ví dụ:**

```
@FunctionalInterface
interface Greeting {
    void sayHello(String name);
}
```

2. Biểu thức Lambda

- Khái niệm:
  - **Lambda Expression** là một cách viết ngắn gọn để định nghĩa một phương thức trừu tượng của Functional Interface
  - Lambda cho phép chúng ta viết mã đơn giản hơn bằng cách loại bỏ cú pháp thừa, như định nghĩa lớp ẩn danh
- Cú pháp lambda:
  - Cú pháp lambda cơ bản:

```
(parameters) -> expression
```

- Hoặc khối lệnh:

```
(parameters) -> { statements; }
```

3. **Functional Interface và Lambda trong Thư viện Java**

- Java cung cấp nhiều Functional Interface tích hợp sẵn trong gói java.util.function. Dưới đây là một số loại phổ biến:
- **Predicate**
  - Dùng để kiểm tra điều kiện
  - Phương thức trừu tượng: boolean test(T t)
  - Ví dụ

```
Predicate<Integer> isEven = Integer num -> num % 2 == 0;
System.out.println(isEven.test( t: 4)); // Output: true
```

- **Consumer**
  - Dùng để thực hiện một hành động không trả về kết quả.
  - Phương thức trừu tượng: void accept(T t)
  - Ví dụ:

```
Consumer<String> print = String message -> System.out.println(message);
print.accept( t: "Hello, Java!"); // Output: Hello, Java!
```

- **Supplier**
  - Dùng để cung cấp một giá trị.
  - Phương thức trừu tượng: T get()
  - Ví dụ:

```
Supplier<Double> random = () -> Math.random();
System.out.println(random.get());
```

- **Function**
  - Dùng để chuyển đổi từ một kiểu này sang kiểu khác.
  - Phương thức trừu tượng: R apply(T t)
  - Ví dụ:

```
Function<String, Integer> length = String str -> str.length();
System.out.println(length.apply( t: "Hello")); // Output: 5
```

Link tài nguyên đọc thêm: https://www.baeldung.com/java-8-lambda-expressions-tips