

Họ và tên: Nguyễn Khánh Quy

MSSV: 21110282

## State Pattern

Mẫu **State** trong lập trình hướng đối tượng là một mẫu thiết kế hành vi được sử dụng để quản lý hành vi của một đối tượng khi trạng thái nội bộ của nó thay đổi.

Thay vì đưa ra các điều kiện và nhánh if-else để xác định hành vi dựa trên trạng thái hiện tại, mẫu **State** cho phép đối tượng chuyển đổi giữa các trạng thái và thực hiện hành vi tương ứng với mỗi trạng thái.

Dưới đây là một ví dụ về mẫu **State** áp dụng vào việc mô phỏng hành vi của một đèn giao thông trong ngôn ngữ lập trình Java. Trong ví dụ này, chúng ta sẽ thiết kế một hệ thống đơn giản mô tả quá trình hoạt động của một đèn giao thông và cách nó chuyển đổi giữa các trạng thái khác nhau (đỏ, vàng, xanh). Mỗi trạng thái của đèn giao thông sẽ có hành vi cụ thể, và mẫu **State** sẽ đảm bảo rằng đèn giao thông biết cách xử lý hành vi tương ứng với trạng thái hiện tại của nó.



## Bước 1: Xác định Interface (TrafficLightState):

Đầu tiên, chúng ta cần tạo một interface **TrafficLightState**. Đây là interface định nghĩa các phương thức chuyển đổi trạng thái và hành vi tương ứng cho mỗi trạng thái của đèn giao thông.

```
public interface TrafficLightState {  
  
    void handleRequest(TrafficLight light);  
  
}
```

## Bước 2: Tạo các lớp cụ thể (RedState, YellowState, và GreenState):

Mỗi lớp này đại diện cho một trạng thái cụ thể của đèn giao thông và triển khai các phương thức từ interface **TrafficLightState**.

Mỗi lớp này xác định hành vi cụ thể của đèn giao thông trong trạng thái tương ứng (ví dụ: hiển thị thông báo, chuyển đổi trạng thái, vv.).

```
public class RedState implements TrafficLightState {  
  
    @Override  
    public void handleRequest(TrafficLight light) {  
        System.out.println("Đèn giao thông màu đỏ. Dừng lại!");  
        light.setState(new GreenState());  
    }  
  
}
```

```
public class GreenState implements TrafficLightState {  
  
    @Override  
    public void handleRequest(TrafficLight light) {  
        System.out.println("Đèn giao thông màu xanh. Tiếp tục di chuyển!");  
        light.setState(new YellowState());  
    }  
  
}
```

```
public class YellowState implements TrafficLightState {

    @Override
    public void handleRequest(TrafficLight light) {
        System.out.println("Đèn giao thông màu vàng. Chuẩn bị dừng lại!");
        light.setState(new RedState());
    }

}
```

### Bước 3: Tạo lớp (TrafficLight):

Lớp này là lớp chứa trạng thái hiện tại của đèn giao thông và thực hiện việc chuyển đổi trạng thái dựa trên các yêu cầu từ bên ngoài (ví dụ: gọi phương thức **requestChange()**).

Lớp **TrafficLight** cũng cung cấp một phương thức để thay đổi trạng thái của đèn giao thông, và sau đó gọi các phương thức tương ứng của trạng thái mới để xử lý hành vi.

```
public class TrafficLight {

    private TrafficLightState state;

    public TrafficLight() {
        this.state = new RedState(); // Ban đầu, đèn giao thông ở trạng thái
        đỏ
    }

    public void setState(TrafficLightState state) {
        this.state = state;
    }

    public void requestChange() {
        state.handleRequest(this);
    }

}
```

#### Bước 4: Thử nghiệm:

Chúng ta tạo ra một đối tượng **TrafficLight** và sử dụng một vòng lặp vô hạn để liên tục yêu cầu thay đổi trạng thái của đèn giao thông bằng cách gọi phương thức **requestChange()**.

```
import java.util.Timer;
import java.util.TimerTask;

public class Main {

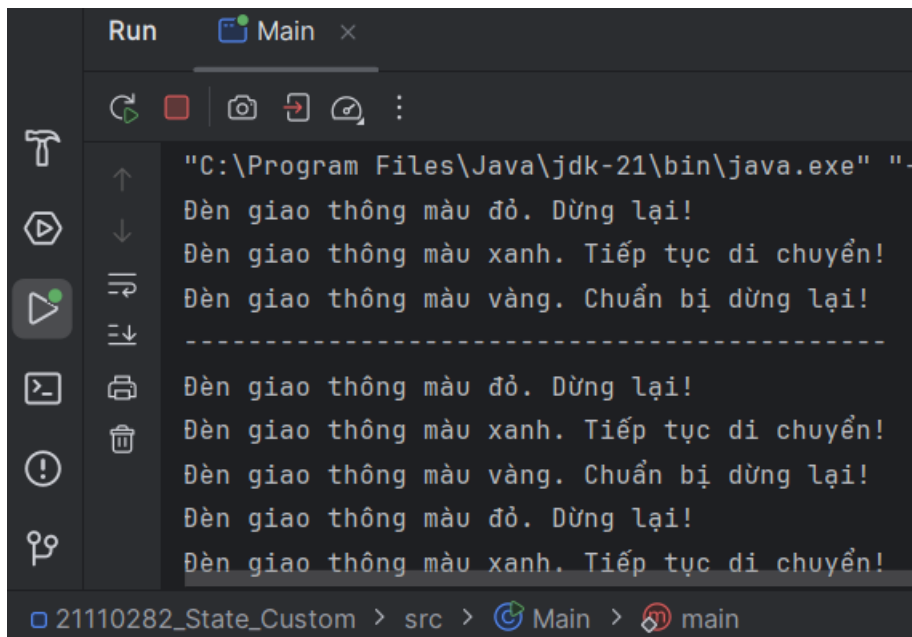
    public static void main(String[] args) {
        TrafficLight trafficLight = new TrafficLight();

        // Mô phỏng việc thay đổi trạng thái của đèn giao thông
        trafficLight.requestChange(); // Đỏ -> Xanh
        trafficLight.requestChange(); // Xanh -> Vàng
        trafficLight.requestChange(); // Vàng -> Đỏ

        System.out.println("-----");

        Timer timer = new Timer();
        timer.scheduleAtFixedRate(new TimerTask() {
            @Override
            public void run() {
                trafficLight.requestChange();
            }
        }, 0, 5000); // Delay 0 giây, và thực hiện sau mỗi 5 giây
    }
}
```

## Kết quả:



```
Run Main x
"C:\Program Files\Java\jdk-21\bin\java.exe" "-
Đèn giao thông màu đỏ. Dừng lại!
Đèn giao thông màu xanh. Tiếp tục di chuyển!
Đèn giao thông màu vàng. Chuẩn bị dừng lại!
-----
Đèn giao thông màu đỏ. Dừng lại!
Đèn giao thông màu xanh. Tiếp tục di chuyển!
Đèn giao thông màu vàng. Chuẩn bị dừng lại!
Đèn giao thông màu đỏ. Dừng lại!
Đèn giao thông màu xanh. Tiếp tục di chuyển!
21110282_State_Custom > src > Main > main
```

Trong ví dụ này, chúng ta triển khai mẫu thiết kế **State** để quản lý hành vi của một đối tượng khi trạng thái nội bộ của nó thay đổi. Điều này giúp tạo ra một cách cấu trúc linh hoạt và dễ dàng mở rộng để xử lý các trạng thái và hành vi phức tạp của đèn giao thông.