State Pattern

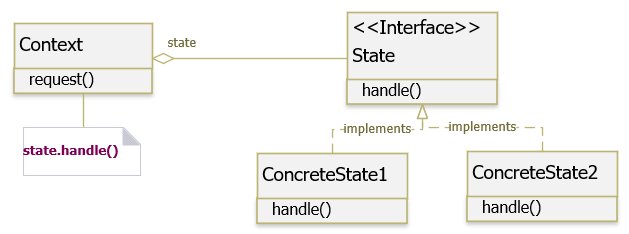
1. **Khái niêm State Pattern**

* State Pattern là một trong những Pattern thuộc nhóm hành vi (Behavior Pattern). Nó cho phép một đối tượng thay đổi hành vi của nó trong khi trạng thái nội bộ của nó thay đổi.
* Trong State pattern , chúng ta tạo các đối tượng đại diện cho các trạng thái khác nhau và một đối tượng bối cảnh mà hành vi của chúng thay đổi khi các thay đổi đối tượng trạng thái của nó.

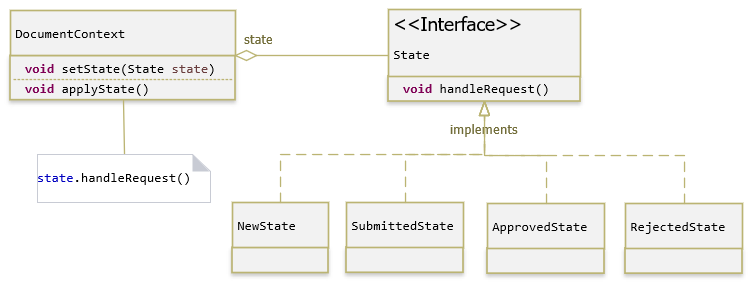
1. **Cấu trúc của State Pattern**

* Context:
  + Định nghĩa giao diện chính để giao tiếp với clients. Client không truy cập trực tiếp đến State của đối tượng.
  + Chứa một thể hiện của ConcreteState object tương ứng với trạng thái hiện tại của đối tượng.
* State:
  + Là một interface hoặc abstract class xác định các đặc tính cơ bản của tất cả các đối tượng ConcreteState. Chúng sẽ được sử dụng bởi đối tượng Context để truy cập chức năng có thể thay đổi.
* ConcreteState:
  + cài đặt các phương thức của State. Mỗi ConcreteState có thể thực hiện logic và hành vi của riêng nó tùy thuộc vào Context.

1. **Biểu đồ lớp của State Pattern**



1. **Ví dụ về State Pattern**
   * Giả sử chúng ta cần xây dựng một ứng dụng quản lý Document. Một Document có thể bao gồm các trạng thái: tạo mới (New), trình phê duyệt (Submitted), phê duyệt (Approved) và từ chối (Rejected).
   * Giả sử chúng ta muốn thêm một trạng thái mới như lưu nháp (Save Draft). Đơn giản chúng ta chỉ cần thêm vào enum một giá trị mới và thêm điều kiện xử lý trong switch-case. Tuy nhiên, nếu làm như vậy thì chúng ta đã vi phạm nguyên tắc Open/ Close. Mỗi khi có thêm một trạng thái mới chúng ta phải sửa nhiều nơi, code trong phương thức handleRequest() ngày càng nhiều và cần phải test lại toàn bộ app.
   * Bây giờ chúng ta áp dụng State Pattern cho ứng dụng trên:



* + Khi đó, nếu muốn thêm một trạng thái mới như lưu tạm (Save Draft), đơn giản tạo một class mới implements từ State mà không làm ảnh hưởng đến các State khác và Context.

1. **Lợi ích của State Pattern**
   * Đảm bảo nguyên tắc [Single responsibility principle (SRP)](https://gpcoder.com/4200-cac-nguyen-ly-thiet-ke-huong-doi-tuong/#Single_responsibility_principle_SRP) : tách biệt mỗi State tương ứng với 1 class riêng biệt.
   * Đảm bảo nguyên tắc [Open/Closed Principle (OCP)](https://gpcoder.com/4200-cac-nguyen-ly-thiet-ke-huong-doi-tuong/#Open-Closed_principle_OCP) : chúng ta có thể thêm một State mới mà không ảnh hưởng đến State khác hay Context hiện có.
   * Giữ hành vi cụ thể tương ứng với trạng thái.
   * Giúp chuyển trạng thái một cách rõ ràng.