Bài toán 1. Singleton, Factory (method, abstract)...

- Sinh viên tự nghĩ ra bài tập áp dụng 2 Design Pattern này.

A screenshot of a computer program

AI-generated content may be incorrect.

2. State, Strategy, Decorator

- State Design Pattern được áp dụng trong những trường hợp mà một đối tượng có thể có nhiều trạng thái khác nhau và hành vi của đối tượng thay đổi tùy thuộc vào trạng thái hiện tại của nó. Thay vì sử dụng các câu lệnh if-else hoặc switch-case phức tạp để kiểm tra và xử lý các trạng thái khác nhau, State Pattern giúp tách biệt mỗi trạng thái thành các lớp riêng biệt và cho phép đối tượng thay đổi trạng thái của mình một cách linh hoạt.

- Strategy Design Pattern được áp dụng trong trường hợp khi bạn có một thuật toán (hoặc một hành vi) mà có thể thay đổi được, và bạn muốn lựa chọn thuật toán đó tại thời điểm thực thi mà không cần thay đổi mã nguồn của đối tượng sử dụng thuật toán. Pattern này giúp tách biệt các thuật toán và cho phép chúng có thể thay đổi linh hoạt mà không làm ảnh hưởng đến các phần khác của hệ thống.

- Decorator Design Pattern được áp dụng trong những trường hợp khi bạn muốn mở rộng hoặc thay đổi hành vi của đối tượng mà không làm thay đổi mã nguồn của lớp đối tượng đó. Đây là một cách linh hoạt để "trang trí" hoặc "bọc" đối tượng với các chức năng bổ sung mà không cần thay đổi cấu trúc bên trong của đối tượng đó.

Cho các mô tả sau:

1. Tạo một hệ thống quản lý đơn hàng có các trạng thái như: Mới tạo, Đang xử lý, Đã giao, và Hủy.

Mỗi trạng thái sẽ có các hành vi khác nhau. Ví dụ:

Mới tạo: Kiểm tra thông tin đơn hàng.

Đang xử lý: Đóng gói và vận chuyển.

Đã giao: Cập nhật trạng thái đơn hàng là đã giao.

Hủy: Hủy đơn hàng và hoàn tiền.

Hãy dùng các Design Pattern: State, Strategy, Decorator viết bằng Java để xử lý mô phỏng trường hợp trên và đưa ra kết luận.