Cairo University
Faculty of Engineering
Computer Engineering Department

Software Engineering
Lab6 – Design Patterns 2

Objective: Notification System for a Streaming Platform

Imagine you’re building a notification system for a streaming platform. This platform allows users to follow certain content creators (e.g., musicians, vloggers) and receive notifications whenever those creators upload new content. Additionally, the platform has different notification channels (e.g., email, SMS, and push notifications) and subscription plans that control the types of notifications users can receive.

<u>Design Description</u>

1. Singleton Pattern:

- The platform has a NotificationManager that manages the dispatching of notifications to users. Since there should only be one instance of NotificationManager across the system, implement it as a Singleton.

2. Observer Pattern:

- Users should be able to subscribe to their favorite content creators. Each time a creator uploads new content, all followers should be notified.
- Use the Observer pattern here where:
  - ContentCreator is the subject, and it maintains a list of User observers.
  - User objects receive notifications when the creator they’re following posts new content.

3. Decorator Pattern:

- Each User has different notification channels (e.g., email, SMS, push notifications) based on their subscription plan. The base User class should be simple, with decorators that add the ability to send notifications via different channels.
- For example:
  - A basic user can only receive email notifications.
  - A premium user may receive notifications via email, SMS, and push notifications.
- Implement each channel as a Decorator that wraps the User object, so notifications can be added dynamically based on the user’s subscription plan.

Requirements

You are given a skeleton to begin with. You are required to:

Implement the Singleton Pattern for NotificationManager:

- Ensure that only one instance of NotificationManager is created. Add a get_instance() method to enforce this.

Implement the Observer Pattern:

- User should act as an observer for ContentCreator.
- Modify the ContentCreator class to notify followers (users) automatically by calling their update() method when new content is uploaded.

Implement the Decorator Pattern for Notification Channels:

- Implement different decorators (EmailNotificationDecorator, SMSNotificationDecorator, PushNotificationDecorator) that add notification capabilities to User objects based on the user's subscription.
- Each decorator should wrap a User and override the update() method to add specific behavior.

Expected Output

```
Alice is now following Music Channel
Bob is now following Music Channel
Alice is now following Vlog Channel
Music Channel has uploaded new content: New Song Release!
Alice received notification: Music Channel uploaded 'New Song Release!'
Email sent to Alice: Music Channel uploaded 'New Song Release!'
Bob received notification: Music Channel uploaded 'New Song Release!'
SMS sent to Bob: Music Channel uploaded 'New Song Release!'
Push notification sent to Bob: Music Channel uploaded 'New Song Release!'
Vlog Channel has uploaded new content: Day in My Life Vlog
Alice received notification: Vlog Channel uploaded 'Day in My Life Vlog'
```