**Exercise-5: Enhanced Decorator Pattern: Notification System with Receiver Details**

* **Introduction**

This document outlines the implementation of an enhanced Decorator Pattern for a flexible notification system. The system allows sending notifications through multiple channels (*Email, SMS, Slack*) and includes receiver details for more realistic notification handling.

* **Problem Statement**
* Develop a notification system that:
* Supports multiple notification channels (Email, SMS, Slack)
* Allows dynamic selection of notification channels
* Includes receiver details (name, email, phone number)
* Validates the availability of contact information before sending notifications
* Provides a user-friendly interface for sending notifications
* **Solution Overview**

The solution uses the Decorator Pattern to create a flexible notification system. Key components include:

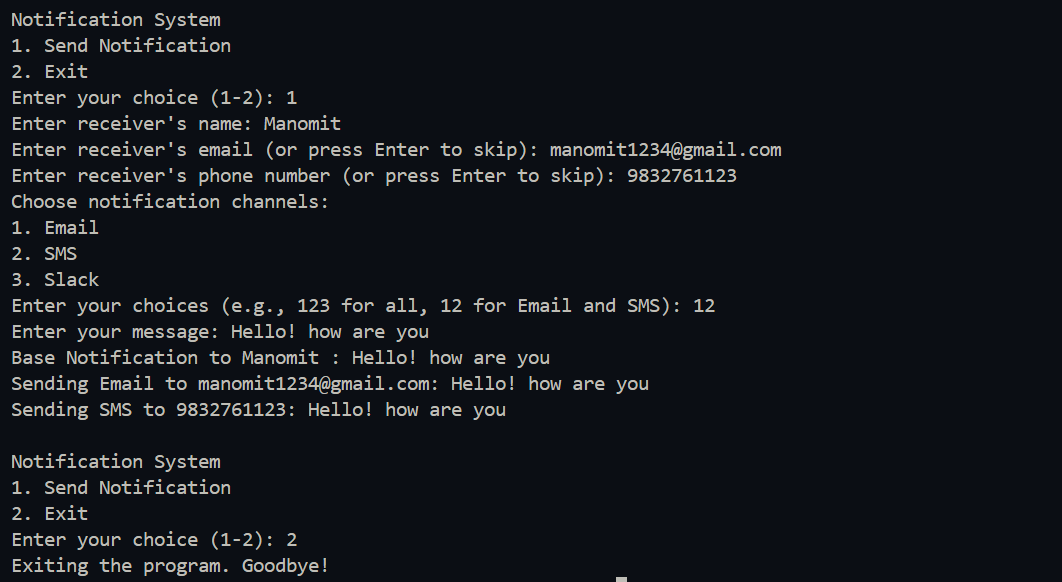
* A base notifier interface and implementation
* Decorator classes for each notification channel
* A Receiver class to encapsulate recipient details
* A test class for user interaction and demonstration
* **Implementation**

Link: code link

* **Explanation of the Enhanced Decorator Pattern**

The enhanced Decorator Pattern implementation in this notification system demonstrates several key concepts:

* **Core Functionality**: The *BaseNotifier* provides the basic notification functionality.
* **Decorators**: *EmailDecorator, SMSDecorator,* and *SlackDecorator* add specific channel notifications.
* **Dynamic Composition**: The client code can dynamically compose different notification channels based on user input.
* **Receiver Information**: The Receiver class encapsulates recipient details, allowing for more realistic notification handling.
* **Validation**: Each decorator checks for the availability of necessary contact information before sending a notification.
* **Open/Closed Principle**: New notification channels can be added by creating new decorators without modifying existing code.
* **Single Responsibility**: Each class has a single, well-defined responsibility, enhancing maintainability.
* **Output**



* **Conclusion**

This enhanced implementation of the Decorator Pattern showcases a flexible and extensible notification system. It demonstrates how the pattern can be adapted to handle complex, real-world scenarios while maintaining its core benefits of dynamic composition and extensibility. The addition of receiver details and contact information validation makes the system more practical and user-friendly.

The implementation successfully meets the requirements of supporting multiple notification channels, allowing dynamic selection, including receiver details, and providing a user-friendly interface. It serves as an excellent example of applying design patterns to solve real-world problems in software development.