**Exercise 10: Implementing the Decorator Pattern**

**Scenario:**

You are developing a notification system where notifications can be sent via multiple channels (e.g., Email, SMS). Use the Decorator Pattern to add functionalities dynamically.

1. Create a new Java Project:

* Create a new Java Project named DecoratorPatternExample.
* A Java project named DecoratorPatternExample is created in eclipse IDE.

2. Define Component Interface:

* Create an interface Notifier with a method send().

public interface Notifier {

void send(String message);

}

* Defines the send method.

3. Implement Concrete Component:

* Create a class **EmailNotifier** that implements Notifier.

public class EmailNotifier implements Notifier {

@Override

public void send(String message) {

System.out.println("Sending Email with message: " + message);

}

}

* Concrete implementation of Notifier.

4. Implement Decorator Classes:

* Create abstract decorator class **NotifierDecorator** that implements **Notifier** and holds a reference to a **Notifier** object.

public abstract class NotifierDecorator implements Notifier {

protected Notifier wrappedNotifier;

public NotifierDecorator(Notifier notifier) {

this.wrappedNotifier = notifier;

}

@Override

public void send(String message) {

wrappedNotifier.send(message);

}

}

* Abstract decorator class that implements Notifier and holds a reference to another Notifier.
* Create concrete decorator classes like SMSNotifierDecorator, SlackNotifierDecorator that extend NotifierDecorator.

public class SMSNotifierDecorator extends NotifierDecorator {

public SMSNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

System.out.println("Sending SMS with message: " + message);

}

}

public class SlackNotifierDecorator extends NotifierDecorator {

public SlackNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

System.out.println("Sending Slack message with message: " + message);

}

}

* SMSNotifierDecorator and SlackNotifierDecorator extend NotifierDecorator to add additional functionality.

5. Test the Decorator Implementation:

* Create a test class to demonstrate sending notifications via multiple channels using decorators.

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Notifier emailNotifier = new EmailNotifier();

Notifier smsNotifier = new SMSNotifierDecorator(emailNotifier);

Notifier slackNotifier = new SlackNotifierDecorator(smsNotifier);

System.out.println("Enter your message:");

String message = scanner.nextLine();

slackNotifier.send(message);

scanner.close();

}

}

* Demonstrates how to use decorators to send notifications via multiple channels.

Output:

