**Exercise 5: Implementing the Decorator Pattern**

**Code:**

// Component interface

interface Notifier {

void send(String message);

}

// Concrete Component

class EmailNotifier implements Notifier {

@Override

public void send(String message) {

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

}

}

// Abstract Decorator

abstract class NotifierDecorator implements Notifier {

protected Notifier wrappedNotifier;

public NotifierDecorator(Notifier notifier) {

this.wrappedNotifier = notifier;

}

@Override

public void send(String message) {

wrappedNotifier.send(message);

}

}

// Concrete Decorator: SMS

class SMSNotifierDecorator extends NotifierDecorator {

public SMSNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

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

}

}

// Concrete Decorator: Slack

class SlackNotifierDecorator extends NotifierDecorator {

public SlackNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

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

}

}

// Test class

public class DecoratorPatternExample {

public static void main(String[] args) {

// Simple email notification

Notifier emailNotifier = new EmailNotifier();

// Email + SMS

Notifier emailAndSMS = new SMSNotifierDecorator(emailNotifier);

// Email + SMS + Slack

Notifier emailSMSSlack = new SlackNotifierDecorator(emailAndSMS);

System.out.println("---- Sending Email only ----");

emailNotifier.send("Hello from Email");

System.out.println("\n---- Sending Email + SMS ----");

emailAndSMS.send("Hello from Email and SMS");

System.out.println("\n---- Sending Email + SMS + Slack ----");

emailSMSSlack.send("Hello from Email, SMS, and Slack");

}

}

**Output:**

