**Exercise 5: 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.

**Steps:**

**1. Create a New Java Project:**

* **Project Name**: DecoratorPatternExample
* Use your preferred IDE (like IntelliJ IDEA, Eclipse, or NetBeans) to create a new Java project.

**2. Define Component Interface:**

**Notifier Interface**:

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

public interface Notifier {

void send(String message);

}

**3. Implement Concrete Component:**

**EmailNotifier Class**:

* Create a class EmailNotifier that implements Notifier.

public class EmailNotifier implements Notifier {

@Override

public void send(String message) {

System.out.println("Sending email notification: " + message);

}

}

**4. Implement Decorator Classes:**

**NotifierDecorator Class**:

* Create an 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);

}

}

**Concrete Decorator Classes**:

* Create concrete decorator classes like SMSNotifierDecorator and SlackNotifierDecorator that extend NotifierDecorator.

public class SMSNotifierDecorator extends NotifierDecorator {

public SMSNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

sendSMS(message);

}

private void sendSMS(String message) {

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

}

}

public class SlackNotifierDecorator extends NotifierDecorator {

public SlackNotifierDecorator(Notifier notifier) {

super(notifier);

}

@Override

public void send(String message) {

super.send(message);

sendSlack(message);

}

private void sendSlack(String message) {

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

}

}

**5. Test the Decorator Implementation:**

**TestDecoratorPattern Class**:

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

public class TestDecoratorPattern {

public static void main(String[] args) {

// Create the base notifier (Email)

Notifier emailNotifier = new EmailNotifier();

// Decorate with SMS and Slack notifications

Notifier smsAndSlackNotifier = new SlackNotifierDecorator(

new SMSNotifierDecorator(emailNotifier));

// Send notifications via Email, SMS, and Slack

smsAndSlackNotifier.send("Your order has been shipped!");

// Send notification via only Email

emailNotifier.send("Your order has been delivered!");

}

}