**MODULE-1**

**DESIGN PATTERN AND PRINCIPLE**

SUPERSET ID:6407550

**Exercise 5: Implementing the Decorator Pattern**

**Notifier.java**

package decorator;

public interface Notifier {

void send(String message);

}

**EmailNotifier.java**

package decorator;

public class EmailNotifier implements Notifier {

@Override

public void send(String message) {

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

}

}

**NotifierDecorator.java**

package decorator;

public abstract class NotifierDecorator implements Notifier {

protected Notifier notifier;

public NotifierDecorator(Notifier notifier) {

this.notifier = notifier;

}

@Override

public void send(String message) {

notifier.send(message);

}

}

**SMSNotifierDecorator.java**

package decorator;

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: " + message);

}

}

**SlackNotifierDecorator.java**

package decorator;

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 message: " + message);

}

}

**NotificationTest.java**

package decorator;

public class NotificationTest {

public static void main(String[] args) {

Notifier emailNotifier = new EmailNotifier();

Notifier smsNotifier = new SMSNotifierDecorator(emailNotifier);

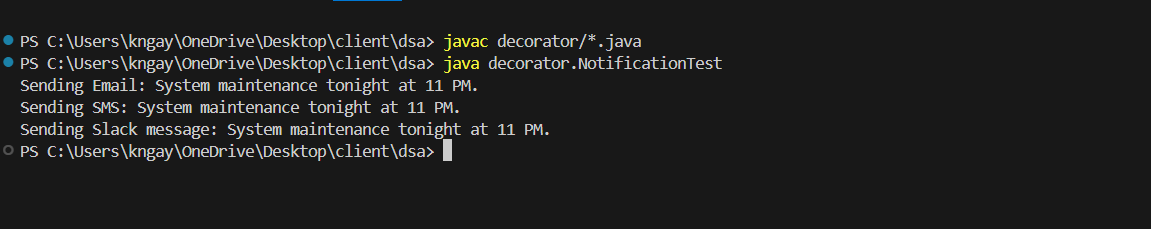
Notifier fullNotifier = new SlackNotifierDecorator(smsNotifier);

fullNotifier.send("System maintenance tonight at 11 PM.");

}

}

**OUTPUT:**

****