**DecoratorPatternExample**

**CODE :**

**Notifier.java**

public interface Notifier {

    void send(String message);

}

**EmailNotifier.java**

public class EmailNotifier implements Notifier {

    @Override

    public void send(String message) {

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

    }

}

**NotifierDecorator.java**

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);

    }

}

**SMSNotifierDecorator.java**

public class SMSNotifierDecorator extends NotifierDecorator {

    public SMSNotifierDecorator(Notifier notifier) {

        super(notifier);

}

  private void sendSMS(String message) {

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

    }

@Override

    public void send(String message) {

        super.send(message);

        sendSMS(message);         }

}

**SlackNotifierDecorator.java**

public class SlackNotifierDecorator extends NotifierDecorator {

    public SlackNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    private void sendSlack(String message) {

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

    }

    @Override

    public void send(String message) {

        super.send(message);

        sendSlack(message);

    }

}

**NotificationTest.java**

/public class NotificationTest {

    public static void main(String[] args) {

        Notifier notifier = new EmailNotifier();

        notifier = new SMSNotifierDecorator(notifier);

        notifier = new SlackNotifierDecorator(notifier);

        notifier.send("System maintenance at 2 AM.");

    }

}

**OUTPUT :**

