**Implementing the Decorator Pattern**

**DecoratorPatternExample.java**

interface Notifier {

    void send();

}

class EmailNotifier implements Notifier {

    @Override

    public void send() {

        System.out.println("Email Notification Sent!");

    }

}

abstract class NotifierDecorator implements Notifier {

    protected Notifier notifier;

    public NotifierDecorator(Notifier notifier) {

        this.notifier = notifier;

    }

    @Override

    public void send() {

        notifier.send();

    }

}

class SMSNotifierDecorator extends NotifierDecorator {

    public SMSNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    @Override

    public void send() {

        super.send();

        System.out.println("SMS Notification Sent!");

    }

}

class SlackNotifierDecorator extends NotifierDecorator {

    public SlackNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    @Override

    public void send() {

        super.send();

        System.out.println("Slack Notification Sent!");

    }

}

public class DecoratorPatternExample {

    public static void main(String[] args) {

        Notifier email = new EmailNotifier();

        Notifier emailAndSMS = new SMSNotifierDecorator(email);

        Notifier allNotifications = new SlackNotifierDecorator(emailAndSMS);

        System.out.println("Only Email Has Been Sent.");

        email.send();

        System.out.println("Email And SMS Has Been Sent.");

        emailAndSMS.send();

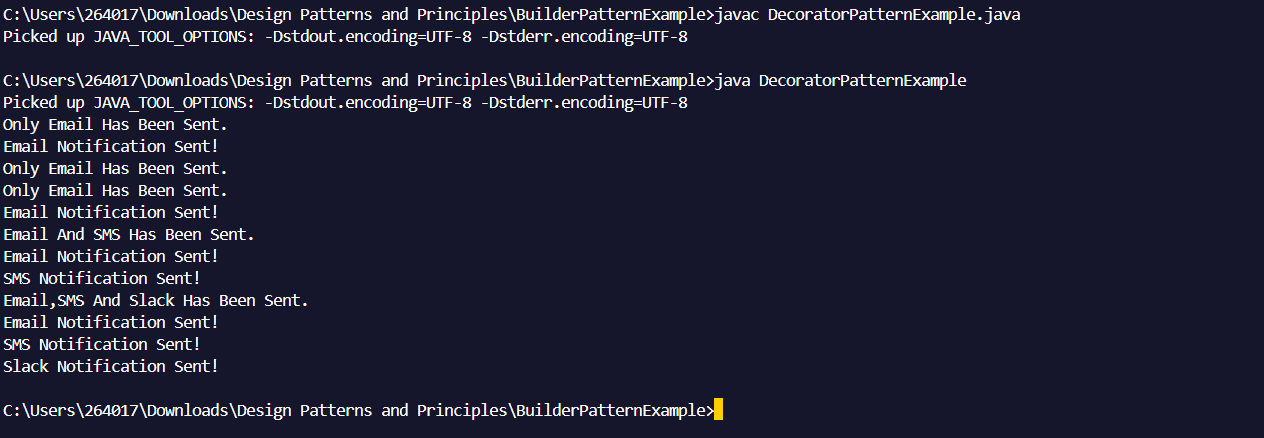
        System.out.println("Email,SMS And Slack Has Been Sent.");

        allNotifications.send();

    }

}

**Output:**

****