**Exercise 5: Implementing the Decorator Pattern**

**Program:**

package DesignPatterns.DecoratorPatternExample;

interface Notifier {

    void send(String message);

}

class EmailNotifier implements Notifier {

    public void send(String message) {

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

    }

}

abstract class NotifierDecorator implements Notifier {

    public Notifier notifier;

    public NotifierDecorator(Notifier notifier) {

        this.notifier = notifier;

    }

    public void send(String message) {

        notifier.send(message);

    }

}

class SMSNotifierDecorator extends NotifierDecorator {

    public SMSNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    public void send(String message) {

        super.send(message);

        sendSMS(message);

    }

    public void sendSMS(String message) {

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

    }

}

class SlackNotifierDecorator extends NotifierDecorator {

    public SlackNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    public void send(String message) {

        super.send(message);

        sendSlack(message);

    }

    public void sendSlack(String message) {

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

    }

}

public class Main {

    public static void main(String[] args) {

        Notifier notifier1 = new SlackNotifierDecorator(new EmailNotifier());

        notifier1.send("Got New Message");

        System.out.println(".......................");

        Notifier notifier2 = new SMSNotifierDecorator(new EmailNotifier());

        notifier2.send("Got New Message another");

        System.out.println(".......................");

        Notifier notifier3 = new SMSNotifierDecorator(new SlackNotifierDecorator(new EmailNotifier()));

        notifier3.send("Got New Message another about meeting");

    }

}

