**DECORATOR PATTERN**

interface Notifier {

void send(String message);

}

class EmailNotifier implements Notifier {

@Override

public void send(String message) {

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

}

}

abstract class NotifierDecorator implements Notifier {

protected Notifier notifier;

public NotifierDecorator(Notifier notifier) {

this.notifier = notifier;

}

@Override

public void send(String message) {

notifier.send(message);

}

}

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);

}

}

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);

}

}

public class Main {

public static void main(String[] args) {

Notifier baseNotifier = new EmailNotifier();

Notifier smsNotifier = new SMSNotifierDecorator(baseNotifier);

Notifier slackAndSmsNotifier = new SlackNotifierDecorator(smsNotifier);

slackAndSmsNotifier.send("System alert: CPU usage high!");

}

}

