**Exercise 5: Implementing the Decorator Pattern**

**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);

}

*@Override*

public void send(String message) {

super.send(message);

sendSMS(message);

}

private void sendSMS(String message) {

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

}

}

**SlackNotifierDecorator.java**

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);

}

}

**Main.java**

public class Main {

public static void main(String[] args) {

// Basic email notification

Notifier emailNotifier = new EmaiLNotifier();

System.***out***.println("Email Notification Only:");

emailNotifier.send("System maintenance tonight at 10 PM");

System.***out***.println("\nEmail + SMS Notification:");

Notifier smsDecorator = new SMSNotifierDecorator(new EmaiLNotifier());

smsDecorator.send("Server is down!");

System.***out***.println("\nEmail + SMS + Slack Notification:");

Notifier fullDecorator = new SlackNotifierDecorator(new SMSNotifierDecorator(new EmaiLNotifier()));

fullDecorator.send("Critical alert: Disk space full!");

}

}

**Output:**

