**Exercise 5: Implementing the Decorator Pattern**

**Notifier.java**

public interface Notifier {

    void send(String message);

}

**NotifierDecorator.java**

public abstract class NotifierDecorator implements Notifier {

    protected Notifier notifier;

    public NotifierDecorator(Notifier notifier) {

        this.notifier = notifier;

    }

    public void send(String message) {

        notifier.send(message);

    }

}

**SMSNotifierDecorator.java**

public class SMSNotifierDecorator extends NotifierDecorator {

    public SMSNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    public void send(String message) {

        super.send(message);

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

    }

}

**SlackNotifier.java**

public class SlackNotifierDecorator extends NotifierDecorator {

    public SlackNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    public void send(String message) {

        super.send(message);

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

    }

}

**Test.java**

public class Test {

        public static void main(String[] args) {

        Notifier base = new EmailNotifier();

        Notifier sms = new SMSNotifierDecorator(base);

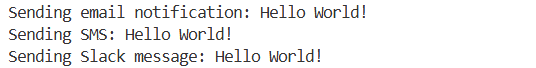
        Notifier slack = new SlackNotifierDecorator(sms);

        slack.send("Hello World!");

    }

}

**Output**

****