**Exercise 5: Implementing the Decorator Pattern**

**Source Code:**

**Notifier.java**

public interface Notifier {

void send(String msg);

}

**EmailNotifier.java**

public class EmailNotifier implements Notifier {

public void send(String msg) {

System.out.println("Email: " + msg);

}

}

**NotifierDecorator.java**

public abstract class NotifierDecorator implements Notifier {

protected final Notifier wrap;

protected NotifierDecorator(Notifier n) { wrap = n; }

}

**SMSNotifierDecorator.java**

public class SMSNotifierDecorator extends NotifierDecorator {

public SMSNotifierDecorator(Notifier n) { super(n); }

public void send(String msg) {

wrap.send(msg);

System.out.println("SMS: " + msg);

}

}

**SlackNotifierDecorator.java**

public class SlackNotifierDecorator extends NotifierDecorator {

public SlackNotifierDecorator(Notifier n) { super(n); }

public void send(String msg) {

wrap.send(msg);

System.out.println("Slack: " + msg);

}

}

**NotificationDemo.java**

public class NotificationDemo {

public static void main(String[] args) {

Notifier notifier = new SlackNotifierDecorator(

new SMSNotifierDecorator(

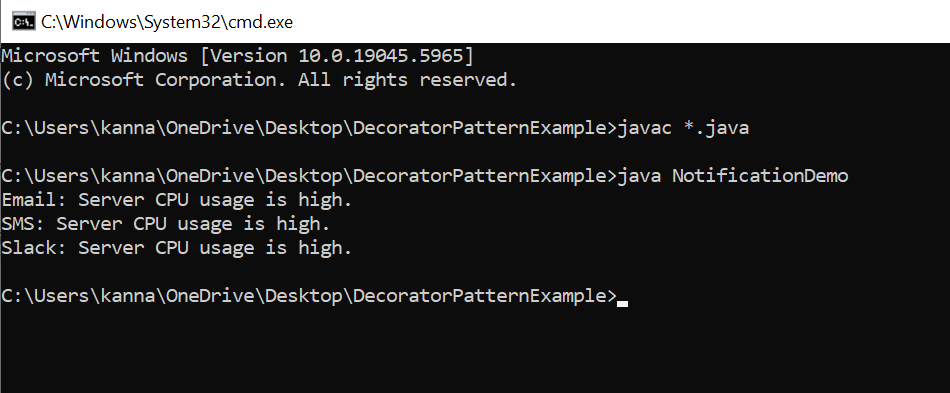
new EmailNotifier()));

notifier.send("Server CPU usage is high.");

}

}

**Output:**

****