**Exercise 5: Implementing the Decorator Pattern**

**Scenario:**

You are developing a notification system where notifications can be sent via multiple channels (e.g., Email, SMS). Use the Decorator Pattern to add functionalities dynamically.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **DecoratorPatternExample**.
2. **Define Component Interface:**
   * Create an interface **Notifier** with a method **send()**.
3. **Implement Concrete Component:**
   * Create a class **EmailNotifier** that implements Notifier.
4. **Implement Decorator Classes:**
   * Create abstract decorator class **NotifierDecorator** that implements **Notifier** and holds a reference to a **Notifier** object.
   * Create concrete decorator classes like **SMSNotifierDecorator**, **SlackNotifierDecorator** that extend **NotifierDecorator**.
5. **Test the Decorator Implementation:**
   * Create a test class to demonstrate sending notifications via multiple channels using decorators.

**CODE:-**

// DecoratorPatternExample.java

interface Notifier {

    void send(String message);

}

// Concrete Component

class EmailNotifier implements Notifier {

    public void send(String message) {

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

    }

}

// Abstract Decorator

abstract class NotifierDecorator implements Notifier {

    protected Notifier notifier;

    public NotifierDecorator(Notifier notifier) {

        this.notifier = notifier;

    }

    public void send(String message) {

        notifier.send(message);

    }

}

// Concrete Decorators

class SMSNotifierDecorator extends NotifierDecorator {

    public SMSNotifierDecorator(Notifier notifier) {

        super(notifier);

    }

    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);

    }

    public void send(String message) {

        super.send(message);

        sendSlack(message);

    }

    private void sendSlack(String message) {

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

    }

}

// Test class

public class DecoratorPatternExample {

    public static void main(String[] args) {

        // Basic email notifier

        Notifier emailNotifier = new EmailNotifier();

        // Add SMS and Slack functionalities

        Notifier smsNotifier = new SMSNotifierDecorator(emailNotifier);

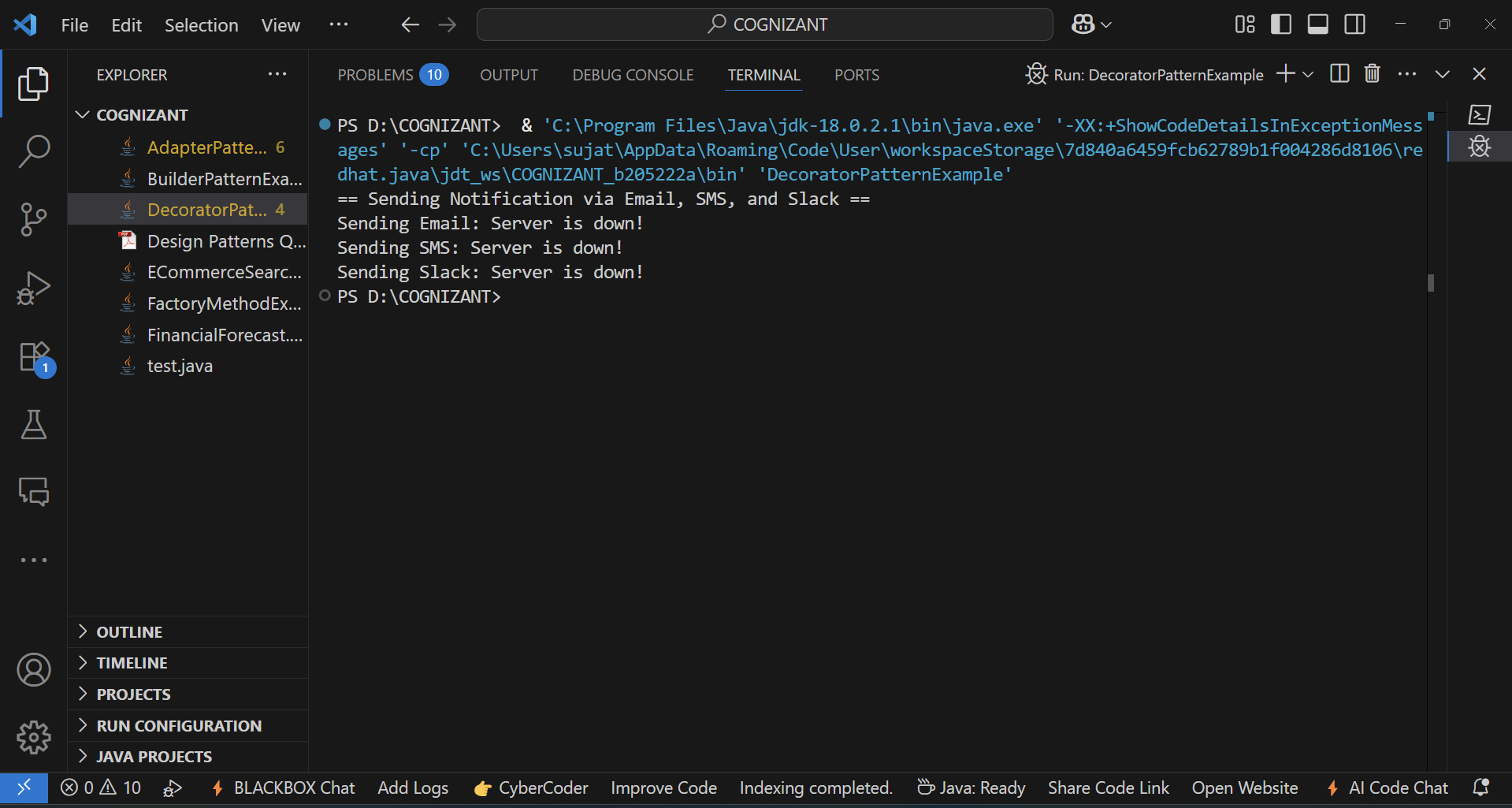
        Notifier slackAndSmsNotifier = new SlackNotifierDecorator(smsNotifier);

        System.out.println("== Sending Notification via Email, SMS, and Slack ==");

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

    }

}

**OUTPUT:-**