**Exercise 6: Implementing the Proxy Pattern**

**Scenario:**

You are developing an image viewer application that loads images from a remote server. Use the Proxy Pattern to add lazy initialization and caching.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **ProxyPatternExample**.
2. **Define Subject Interface:**
   * Create an interface Image with a method **display()**.
3. **Implement Real Subject Class:**
   * Create a class **RealImage** that implements Image and loads an image from a remote server.
4. **Implement Proxy Class:**
   * Create a class **ProxyImage** that implements Image and holds a reference to RealImage.
   * Implement lazy initialization and caching in **ProxyImage**.
5. **Test the Proxy Implementation:**
   * Create a test class to demonstrate the use of **ProxyImage** to load and display images.

**CODE:-**

// ProxyPatternExample.java

interface Image {

    void display();

}

class RealImage implements Image {

    private String filename;

    public RealImage(String filename) {

        this.filename = filename;

        loadFromRemoteServer();

    }

    private void loadFromRemoteServer() {

        System.out.println("Loading image from remote server: " + filename);

    }

    @Override

    public void display() {

        System.out.println("Displaying image: " + filename);

    }

}

class ProxyImage implements Image {

    private RealImage realImage;

    private String filename;

    public ProxyImage(String filename) {

        this.filename = filename;

    }

    @Override

    public void display() {

        if (realImage == null) {

            realImage = new RealImage(filename); // Lazy loading

        }

        realImage.display(); // Cached display

    }

}

public class ProxyPatternExample {

    public static void main(String[] args) {

        Image image1 = new ProxyImage("photo1.jpg");

        Image image2 = new ProxyImage("photo2.jpg");

        // First time - loads from server

        image1.display();

        System.out.println();

        // Second time - uses cached image

        image1.display();

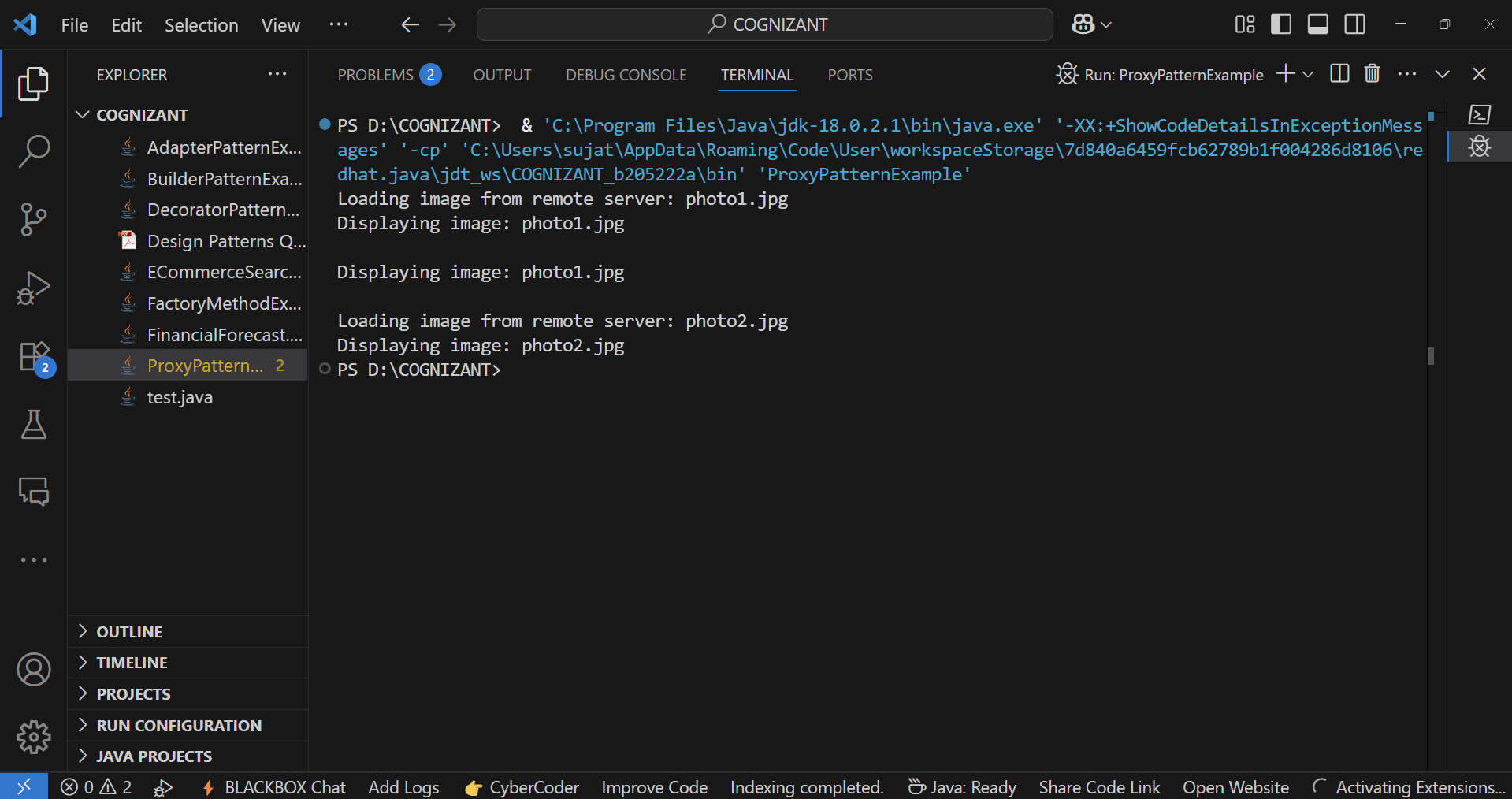
        System.out.println();

        // Loads second image

        image2.display();

    }

}

**OUTPUT:-**