**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.

1. Create a new Java Project:

* Create a new Java Project named ProxyPatternExample.
* A Java project named ProxyPatternExample is created in eclipse IDE.

2. Define Subject Interface:

* Create an interface **Image** with methods like **display()**.

public interface Image {

void display();

}

* The Image interface declares a single method display(), which will be implemented by both the real object and the proxy.

3. Implement Real Subject Class:

* Create a class **RealImage** that implements Image and loads an image from a remote server.

public class RealImage implements Image {

private String fileName;

public RealImage(String fileName) {

this.fileName = fileName;

loadImageFromDisk();

}

private void loadImageFromDisk() {

System.out.println("Loading " + fileName);

try {

Thread.sleep(2000); // simulate delay

} catch (InterruptedException e) {

e.printStackTrace();

}

}

@Override

public void display() {

System.out.println("Displaying " + fileName);

}

}

* The RealImage class implements the Image interface and represents the real object that performs the actual image loading and displaying operations.
* The constructor simulates loading an image from the disk, which takes time.

4. Implement Proxy Class:

* Create a class **ProxyImage** that implements Image and holds a reference to RealImage.
* Implement lazy initialization and caching in **ProxyImage**.

public 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);

}

realImage.display();

}

}

* The ProxyImage class implements the Image interface and holds a reference to a RealImage object.
* It performs lazy initialization by creating the RealImage object only when the display() method is called for the first time.

5. Test the Proxy Implementation:

* Create a test class to demonstrate the use of **ProxyImage** to load and display images.

import java.util.Scanner;

public class ProxyPatternTest {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the name of the first image file to display: ");

String fileName1 = scanner.nextLine();

Image image1 = new ProxyImage(fileName1);

System.out.print("Enter the name of the second image file to display: ");

String fileName2 = scanner.nextLine();

Image image2 = new ProxyImage(fileName2);

System.out.println("\nDisplaying the first image:");

image1.display();

System.out.println("\nDisplaying the first image again:");

image1.display();

System.out.println("\nDisplaying the second image:");

image2.display();

System.out.println("\nDisplaying the second image again:");

image2.display();

scanner.close();

}

}

* The ProxyPatternTest class is the client code that interacts with the proxy
* It allows the user to enter the names of the image files to be displayed.
* The proxy controls the loading of the images, ensuring they are loaded from disk only once.
* Output

