**Exercise 6: Implementing the Proxy Pattern**

**Code:**

// Define the Subject interface

interface Image {

void display();

}

// Real subject class that simulates loading an image from a remote server

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);

// Simulate delay

try {

Thread.sleep(1000); // 1 second delay to simulate loading

} catch (InterruptedException e) {

e.printStackTrace();

}

}

@Override

public void display() {

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

}

}

// Proxy class with lazy initialization and caching

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 initialization

} else {

System.out.println("Using cached image: " + fileName);

}

realImage.display();

}

}

// Test class

public class ProxyPatternExample {

public static void main(String[] args) {

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

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

System.out.println("\nFirst time displaying image1:");

image1.display(); // Loads from remote server

System.out.println("\nSecond time displaying image1:");

image1.display(); // Uses cached image

System.out.println("\nFirst time displaying image2:");

image2.display(); // Loads from remote server

System.out.println("\nSecond time displaying image2:");

image2.display(); // Uses cached image

}

}

**Output:**

