**Design Pattern And Principles**

**EXERCISE 6: Implementing the Proxy Pattern**

**Source Code**

// Main.java

// Step 2: Subject Interface

interface Image {

void display();

}

// Step 3: Real Subject Class

class RealImage implements Image {

private String filename;

public RealImage(String filename) {

this.filename = filename;

loadFromServer();

}

private void loadFromServer() {

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

}

public void display() {

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

}

}

// Step 4: Proxy Class

class ProxyImage implements Image {

private String filename;

private RealImage realImage;

public ProxyImage(String filename) {

this.filename = filename;

}

public void display() {

if (realImage == null) {

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

} else {

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

}

realImage.display();

}

}

// Step 5: Test Class

public class Main {

public static void main(String[] args) {

Image img1 = new ProxyImage("Maldives.png");

Image img2 = new ProxyImage("LightHouse.jpg");

System.out.println("\n--- First time display ---");

img1.display(); // loads from server

img2.display(); // loads from server

System.out.println("\n--- Second time display ---");

img1.display(); // uses cached image

img2.display(); // uses cached image

}

}

**OUTPUT**

****