**Exercise 6: Implementing the Proxy Pattern**

**Image.java**

**package** mypackage;

**public** **interface** Image {

**void** display();

}

**Implement Real Subject Class**

**RealImage.java**

**package** mypackage;

**public** **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);

}

**public** **void** display() {

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

}

}

**Implement Proxy Class**

**ProxyImage.java**

**package** mypackage;

**public** **class** ProxyImage **implements** Image {

**private** RealImage realImage;

**private** String fileName;

**public** ProxyImage(String fileName) {

**this**.fileName = fileName;

}

**public** **void** display() {

**if** (realImage == **null**) {

realImage = **new** RealImage(fileName); // Lazy initialization

} **else** {

System.***out***.println("Image already loaded from cache.");

}

realImage.display();

}

}

**Test the Proxy Implementation**

**TestProxyPattern.java**

**package** mypackage;

**public** **class** TestProxyPattern {

**public** **static** **void** main(String[] args) {

Image image = **new** ProxyImage("nature\_photo.jpg");

// Image will be loaded from remote server only once

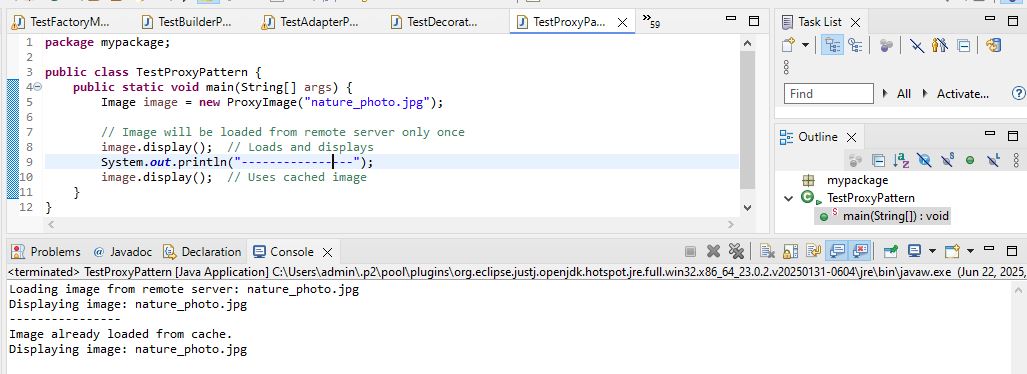
image.display(); // Loads and displays

System.***out***.println("----------------");

image.display(); // Uses cached image

}

}

  
We are developing an image viewer application that loads images from a remote server. Loading large images takes time and resources. To optimize performance, we apply the Proxy Pattern.

**1. Subject Interface**

The Image interface defines a common method display() which both the real image and proxy implement.

**2. Real Subject Class**

RealImage implements Image and loads the image from the remote server when it is instantiated.

* Actual loading happens inside the constructor.
* Displays image when display() is called.

**3. Proxy Class**

ProxyImage also implements Image but delays creation of RealImage until display() is first called (lazy initialization).

* Caches the loaded image to avoid repeated loading.

**4. Advantages of Proxy Pattern**

* Provides lazy loading to avoid unnecessary resource usage.
* Reduces memory usage and improves performance.
* Controls access to the real object.
* Can add additional functionality like access control, logging, and caching.

**5. Time Complexity**

* First call: O(1) for proxy call + time for real image loading.
* Subsequent calls: O(1), as image is loaded from cache.

**6. Real-life Applications**

* Virtual Proxies for large resource loading (images, videos, files).
* Remote Proxies to control access to remote objects (e.g., RPC, web services).
* Protection Proxies for access control and security.