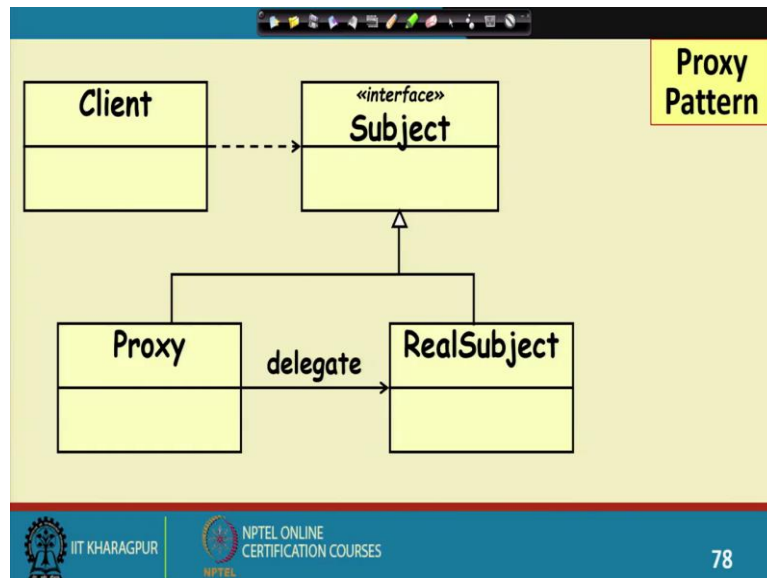


Object Oriented System Development using UML, Java and Patterns
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 56
Proxy Pattern II

Welcome to this lecture.

In the last lecture, we had looked at the Proxy Pattern, we said that it's a very useful pattern, very simple pattern and a proxy has many different uses in an application. We had identified seven or eight very different uses and then we were looking at the class structure of the proxy pattern. Now, let's continue from there and we will look at the different applications of the proxy.

(Refer Slide Time: 00:55)



In above slide class structure of proxy pattern is shown. The class structure itself is very simple. The subject interface is implemented by the proxy and the real subject and the proxy holds a reference to the real subject and delegates any method call that it cannot handle by itself.

(Refer Slide Time: 01:18)

Virtual Proxy: Why Stand-in?

1. The image is expensive to load
2. The complete image is not always necessary

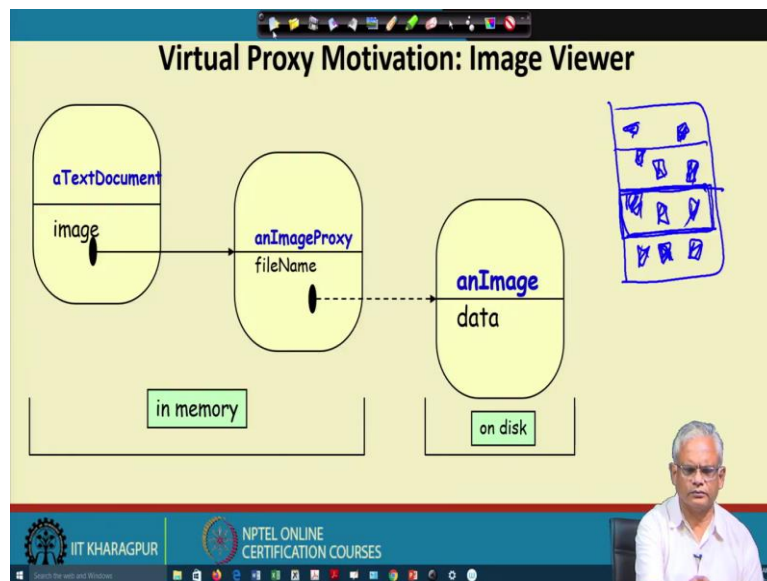
The diagram illustrates the concept of a virtual proxy. On the left, a large image of an elephant is shown within a hexagonal frame, labeled "2MB". An arrow labeled "lightweight" points to the right, where a smaller, pink hexagonal frame is shown, labeled "2KB".

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now let's look at the virtual proxy. The virtual proxy is a stand-in or it is a representative of the real object. The main motivation of the virtual proxy is to speed up an application. Many of the high-resolution images are very time consuming to load from the disk and display. If there are thousands of images like this, it will really slow down an application during it start.

The image may be 2 MB to load (in the above slide). But then the proxy by itself may be just 2 KB or less. In many applications as we will see now that the complete image is not always necessary, we will just show the proxy to the client or to the user and only when he is interested, he will click on the proxy and then the real image will be shown.

(Refer Slide Time: 02:53)



If we try to understand a text document has many images and initially, what will be shown in the text document as the proxy. The real image is a heavyweight object. The data for the image is residing on disk. If we are opening a text document having let say 500 high quality images, then once we load the text document, it will take long time and the user will feel, that the application is slow and for that reason, most of the text editors they use an image proxy where all the images are not loaded.

When the text document is opened, only those which are in the visible area are loaded and as the client moves across the document, those which come into the viewing area are loaded. If we can just represent a large text document. Let say this is a large text document (in the above slide draw with blue colour) and it has many different images embedded inside it which are high quality images. In the viewport or the window only, few images are viewable. This can be accommodated on one screen. For the ones which are not in the viewport or the screen, for those only the proxies exist and here, these are the real images that are loaded only when the client or the user sees this. If the client stops, this images changes and stops. These images are never fetched from the disk, the application appears to be fast and just run a small one. Imagine that it has some 200 pages, and just one page is loaded, the client or the user at any time might just look at only 3-4 pages. So, only those images are loaded and as the client changes the page and the proxy determine to which images need to visible and they load the real image from the disk.



(Refer Slide Time: 06:23)

Virtual Proxy example

- Images are stored and loaded separately from text
- If a RealImage is not yet loaded, a ProxyImage displays a grey rectangle in place of the image
- The client cannot tell whether it is dealing with a ProxyImage instead of a RealImage

```

classDiagram
    class Image {
        boundingBox()
        draw()
    }
    class ProxyImage {
        boundingBox()
        draw()
    }
    class RealImage {
        boundingBox()
        draw()
    }
    Image <|-- ProxyImage
    Image <|-- RealImage
    ProxyImage --> RealImage : realSubject
  
```

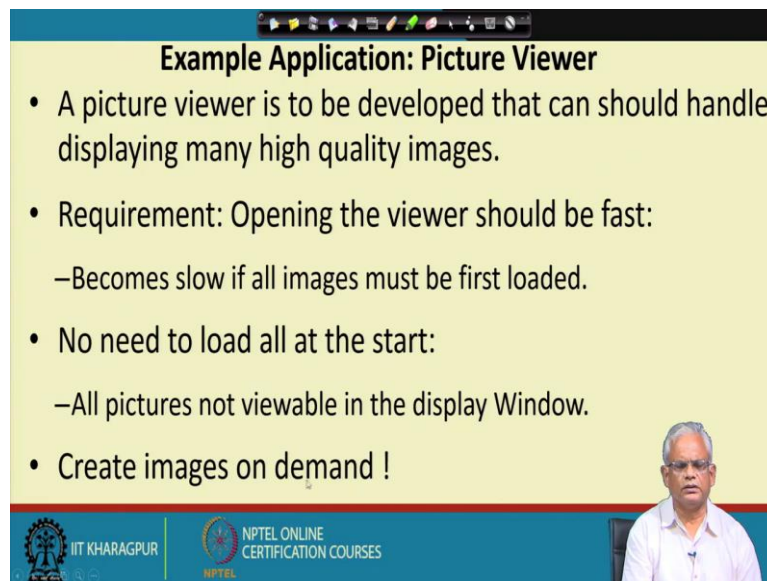



If we have to draw the class structure for the virtual proxy is the same one like simple proxy (in the above slide). The image is the interface. It has a boundingBox() and draw() methods. The boundingBox() method is necessary to determine whether it is visible unique part of the image becomes visible or the image needs to be loaded. The ProxyImage and the RealImage both implement the bounding box and the draw.

The proxy holds a reference to the real subject. If the image is not loaded and in the screen that is being viewed, then it loads that and any method invocation is passed to the real image. But if the real image is not loaded, just the proxy exists. The proxy may be a grey rectangle. Initially, as you change the page you might just find a grey rectangle until the proxy gets loaded, because the proxy may take some time for heavyweight objects.

You can see such a situation in many of the word processors as you change the page, you might find that initially, some of the images appear as grey boxes or grey rectangles and those are the proxy and they are trying to load the real image because it has become viewable and as soon as the real image is loaded, you can see the real image.

(Refer Slide Time: 08:45)



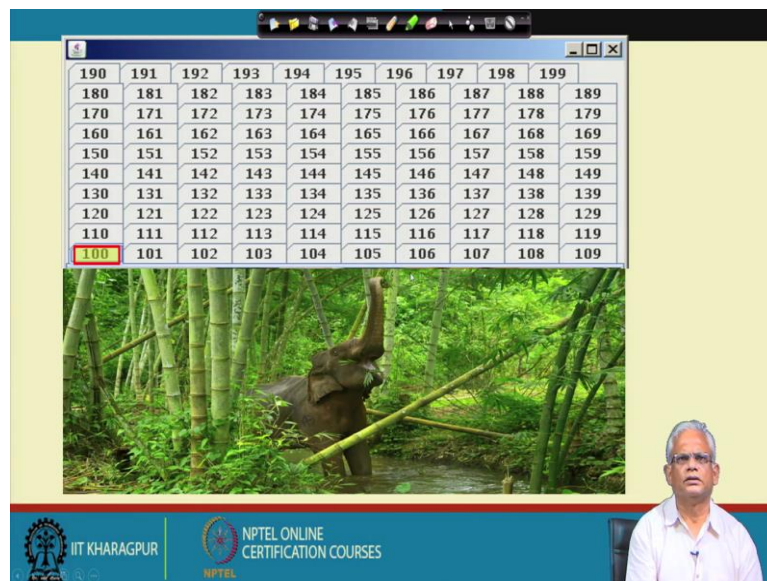
Example Application: Picture Viewer

- A picture viewer is to be developed that can should handle displaying many high quality images.
- Requirement: Opening the viewer should be fast:
 - Becomes slow if all images must be first loaded.
- No need to load all at the start:
 - All pictures not viewable in the display Window.
- Create images on demand !

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

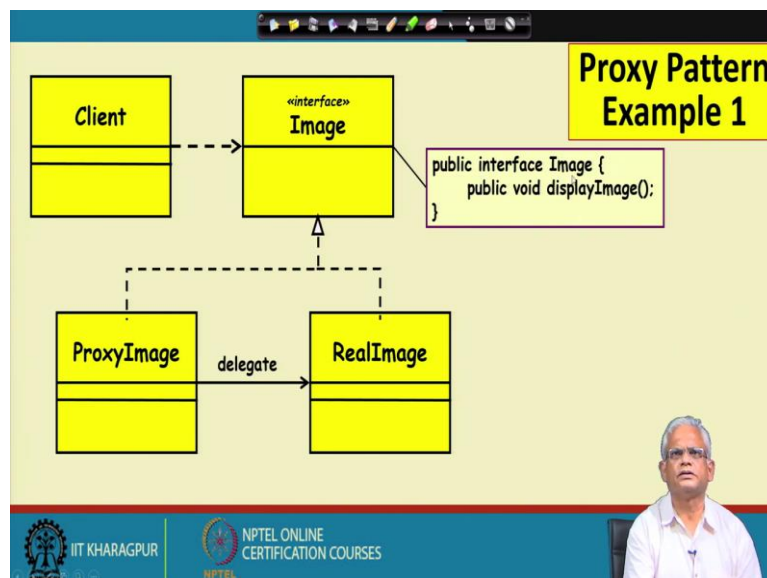
Now let's look at an application the picture viewer. This is a photo album where there is a photo album application where we can store many high-quality images maybe 10,000 images can be stored. In a very naive application where the proxy pattern is not used as you start the photo album application, all the images that have been stored in the photo album will get loaded and it will take 10 minutes to open such an application. But all the users want the application to open very fast and therefore only the proxy is shown we will see the picture in the next slide. We don't load all the images at the start and in any case, all the pictures are not viewable in the display window. As the client requests an image the image is loaded.

(Refer Slide Time: 10:07)



So, this is an example (in the above slide), these are all proxies, there are many pictures here and let us say one of the pictures is shown here (high quality picture number 100). Initially, we can see at most one picture at a time and therefore, the application starts only one of the pictures is set that is loaded and as the user selects one of these than a different picture will be loaded. Those are not loaded, the start loaded on demand. So, this is an example application of the virtual proxy.

(Refer Slide Time: 10:54)



Now let's look at the code for proxy pattern (in the above slide). The image interface implemented by the proxy image and real image. Proxy image holds a reference to the real image and delegates when necessary. This is the code for the interface Image:

```
public interface Image{  
  
public void displayImage();}
```

So, this is the code for the image, and this is the method – displayImage() and this method will be implemented whether real image and the proxy image.

(Refer Slide Time: 11:43)

• **Word Processor:**

- Suppose a text document contains lots of multimedia objects and yet should load fast
- Create proxies that represent large images, movies, etc. --- only load objects on demand as they become visible on the screen (only a small part of the document is visible at a time)

**Similar Uses:
Lazy Loading**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

In the virtual proxy, the main idea is lazy loading. In a word processor, the text document contains lots of multimedia objects and these objects can be images, movies, etc and these will be loaded only when required.

Initially, the proxies are shown and when this become visible, the client first time opens a page and the page becomes visible and if the client never looks at a page, those images are never fetched from the disk and never created and the overhead is never incurred. Let say we have a text document with 10000 pages and at one time the user looks at only 5-7 pages, then just imagine how much faster it will be with the proxy.

(Refer Slide Time: 13:01)

Lazy Loading

- A hybrid approach can be used:
 - **The proxy implements some operations itself.**
 - Create the real object only if the client invokes one of the operations it cannot perform
- A proxy stores necessary information to create the object on-the-fly:
 - file name, network address, etc.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

As we are mentioning, sometimes the proxy might implement some of the operations itself and here as the client invokes those methods which the proxy can answer, it does not delegate to the real object and the proxy stores the necessary information to create the server object on the fly. For example, the file name or the network address if it resides across the network.

(Refer Slide Time: 13:44)

Image Proxy: Code Example

```
classDiagram
    class DocumentEditor
    class Graphic {
        Draw()
        GetExtent()
        Store()
        Load()
    }
    class Image {
        imageImp
        extent
        Draw()
        GetExtent()
        Store()
        Load()
    }
    class ImageProxy {
        fileName
        extent
        Draw()
        GetExtent()
        Store()
        Load()
    }
    DocumentEditor ..> Graphic
    Graphic <|-- Image
    Graphic <|-- ImageProxy
    ImageProxy ..> Image
```

```
if(image == 0)
    image = LoadImage(fileName);
    Image.Draw();

if(image == 0)
    return(extent);
else
    return(image.GetExtent());
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

This is the code example (in the above slide). Let say we have this graphic, the document editor invokes the graphic depending on whether becomes visible and the image and the image proxy both implement the Graphic methods here – Draw(), GetExtent(), Store() and Load() and if the image is not loaded, then it loads the image and if the image ==0 it return the extent else return the result of the GetExtent method here.

(Refer Slide Time: 15:09)

```
interface Graphic {
    public void displayImage();
    public void loadImage();
}

class Image implements Graphic {
    private String filename;
    public Image(String filename) {
        this.filename = filename;
        System.out.println("Loading "+filename);
    }
    public void displayImage() {
        System.out.println("Displaying "+filename);
    }
}
```

**Image Proxy:
Code Example**

The diagram shows the Graphic interface with methods draw(), getExtent(), store(), and load(). The Image class implements Graphic, with a private attribute imageImp and extent, and methods draw(), getExtent(), store(), and load(). The ImageProxy class also implements Graphic, with a private attribute fileName and extent, and methods draw(), getExtent(), store(), and load(). Arrows indicate that Image and ImageProxy implement Graphic, and ImageProxy depends on Image.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now this is the rest of the code (in the above slide). Graphic is the interface here and this is the display image and load image are just written two of the methods of the Graphic interface. Class Image implements graphic. Image has the file name attribute and the file name is created in the constructor and then the displayImage() just prints the file name.

(Refer Slide Time: 15:55)

```
public class ImageProxy implements Graphic {
    private String filename;
    private Image image;

    public ImageProxy (String filename){
        this.filename = filename;
    }
    public void displayImage() {
        if (image == null) {
            image = new RealImage(filename); //load only on demand
        }
        image.displayImage();
    }
}
```

**Proxy
Pattern
Example 1**

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now the image proxy also implements Graphic and it has a reference to the image here as you can see. It has a reference to the real image and the image proxy it takes as argument the file name and stores the file name and then if the image has not been created and the display image is called on the proxy, then only if it is not created, it creates the real image and then calls the displayImage and called real image.


(Refer Slide Time: 17:04)

```
public class ProxyExample {
    public static void main(String[] args) {

        ArrayList<Image> images = new ArrayList<Image>();
        images.add(new ProxyImage("HiRes_10MB_Photo1"));
        images.add(new ProxyImage("HiRes_10MB_Photo2"));
        images.add(new ProxyImage("HiRes_10MB_Photo3"));

        images.get(0).displayImage();
        images.get(1).displayImage();
        images.get(0).loadImage();
    }
}
```

Proxy Example - -- Client




IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Now this is the client proxy example (in the above slide). It has many images, and an ArrayList maintain for that. Many images are added in the ArrayList. Here, three images, Photo1, Photo2, Photo3 are added and then on the ArrayList, we just access the objects at 0, 1 and 0 and then we can call the displayImage or the load image and these are just the proxy here. Just look at here, that actually the ArrayList is images, but then it contains only the proxy and only when needed, the actual image will be loaded and displayed.

(Refer Slide Time: 18:04)

Remote Proxy

- The proxy object:
 - Assembles data into a network message
 - Sends it to the remote object (Server)
- A remote receiver:
 - Receives this data,
 - Turns it into a local message that is sent to the remote object



```
graph LR; Client[Client] --> Proxy[Proxy]; Proxy --> Internet((Inter net)); Internet --> Server[Server];
```

The diagram illustrates the flow of data in a remote proxy setup. A Client sends a request to a Proxy. The Proxy then sends the request through the Internet to a Server. The Server processes the request and returns the data to the Proxy, which then sends it back to the Client.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now let's look at a second application is a remote proxy (in the above slide). Here, the proxy is a local sit in for the real server object, the client thinks that the server is available on the same machine. But actually, the only proxy is there on the same machine. The server is across the internet somewhere.

The client interacts with the proxy thinking that the real server is there in that machine, the client thinks that the proxy itself is the real server. But then, as the client requests the real proxy, the proxy tries to handle some of the requests if it can, otherwise it assembles the data or the request into a network message and then sends it to the remote object which is the server and then the server processes it returns it to the proxy, maybe compresses the reply, may be large data and then the proxy receives the compressed data and decompresses it and then returns it to the client

(Refer Slide Time: 19:28)

Remote Proxy

- **Stand-in for an object often needed because it is:**
 - Not locally available;
 - Instantiation and access are complex
 - Needs protected access for security.
- **The stand-in must:**
 - Have the same interface as the real object;
 - Handle as many messages as it can;
 - Delegate messages to the real object when necessary.

Diagram: Client → Proxy → Internet → Server

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Simplifies the application development that is the client code. It need not get bogged down with the server address, contacting the server, instantiating and so on. All these responsibilities are taken over by the proxy. The proxy also handles many messages. It can and does not unduly overload the server and also speeds up the response to the client. It delegates messages to the real object when necessary.

(Refer Slide Time: 20:13)

Remote Proxy Pattern Structure

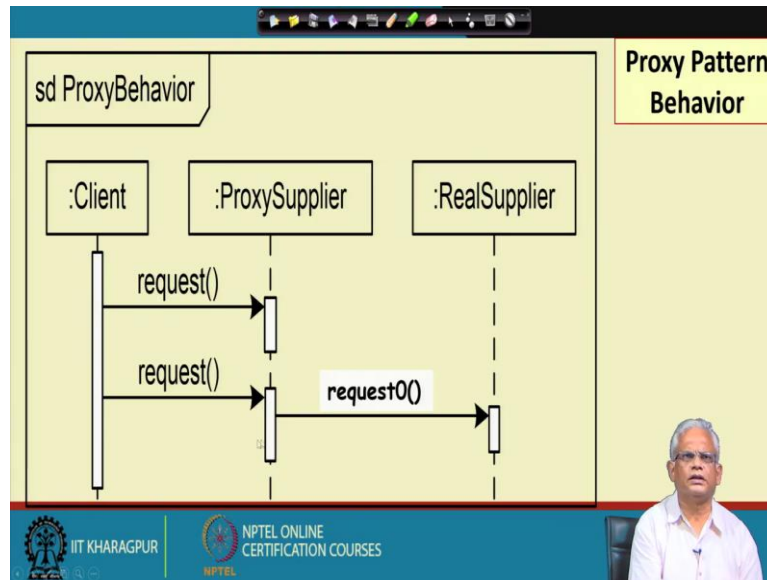
```
classDiagram
    class Supplier {
        <<interface>>
        request()
    }
    class Client
    class ProxySupplier
    class RealSupplier
    Client --> ProxySupplier
    ProxySupplier ..> Supplier
    ProxySupplier ..> RealSupplier
    ProxySupplier --> RealSupplier
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, this is the remote proxy structure (in the above slide), the server or the supplier interface and then implemented by the proxy and the real server. The client interacts only with the proxy, which is there on the same machine, the client creates the proxy, the proxy holds the reference to the real server, the address and the network and it can create the network over the

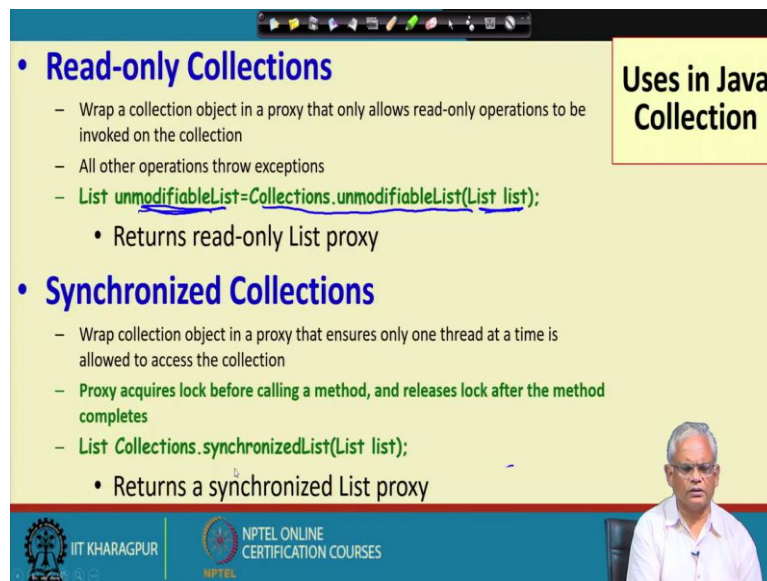
network if the real server does not exist, otherwise if the exists it just invokes the method when it cannot handle by itself.

(Refer Slide Time: 20:56)



If we look at the proxy behaviour in the form of a sequence diagram, the three classes that are involved: the client, the proxy and the real server. The client interacts only with the proxy and some of the requests handles by the block proxy here and does not even pass on to the real supplier and some requests it does some work here and then creates an appropriate method and the real supplier get the data, any further processing required might do and return the data to the client.

(Refer Slide Time: 21:44)



• Read-only Collections

- Wrap a collection object in a proxy that only allows read-only operations to be invoked on the collection
- All other operations throw exceptions
- List `unmodifiableList=Collections.unmodifiableList(List list)`;
 - Returns read-only List proxy

• Synchronized Collections

- Wrap collection object in a proxy that ensures only one thread at a time is allowed to access the collection
- Proxy acquires lock before calling a method, and releases lock after the method completes
- List `Collections.synchronizedList(List list)`;
 - Returns a synchronized List proxy

Uses in Java Collection

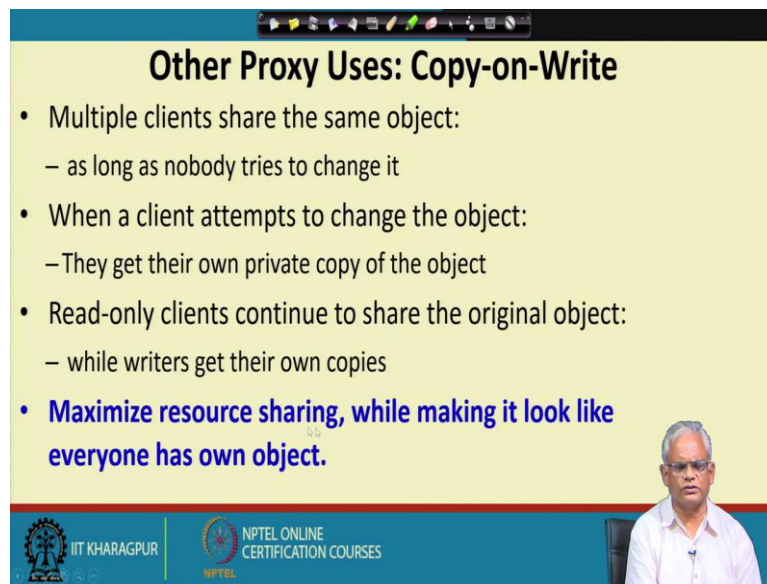
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

The Java makes use of the Java collection classes make use of the proxy pattern. For example, the read only collections or the synchronized collections. In the read only collections, when we want some collection object to be only read only, the clients cannot make any changes to this kind of data, we want some data to be only read only access to the client objects, they cannot modify it.

Then a proxy is given to the client the proxy is created as you can see here (in the above slide underlined). This is how the read only collections work: `Collections.unmodifiableList(List list)` and the argument given is the list, this is the ID of the actual list and `unmodifiableList` is the proxy, which holds the idea of the actual list and it has additional code here. If the client requests to write, it throws an exception. Otherwise, if there is a read request, it just passes it on to the list. The synchronized collection is also a proxy.

Here again, when many objects try to access an object, the proxy synchronizes the access it acquires a lock before calling the method and if the lock is not available, it does not make the call. The client objects wait and only when the lock is available, it calls the method and after the method results return it releases the lock and therefore, the server a sequential at any time. Only one of the clients can make a call to the server object and that discipline is enforced the synchronized collection and here the code is that the client deals with only the proxy. The list is given as an argument in the constructor the client deals with only the proxy.

(Refer Slide Time: 25:00)



Other Proxy Uses: Copy-on-Write

- Multiple clients share the same object:
 - as long as nobody tries to change it
- When a client attempts to change the object:
 - They get their own private copy of the object
- Read-only clients continue to share the original object:
 - while writers get their own copies
- **Maximize resource sharing, while making it look like everyone has own object.**

The slide features a small video inset of a man in a white shirt and glasses in the bottom right corner. At the bottom, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

There are other uses of proxy. Proxy has large number of users copy on write, multiple clients share the same object and therefore it is not necessary to create many instances of the object helps to speed up the application.

But if one of the client tries to modify the object, then a copy of that is created and the client who modifies uses the copy that has been specifically created for the that client. The other clients continue to say the same object and here it speeds up the application by maximizing resource sharing, while making it look like everyone has its own object.

Refer Slide Time: 25:56)

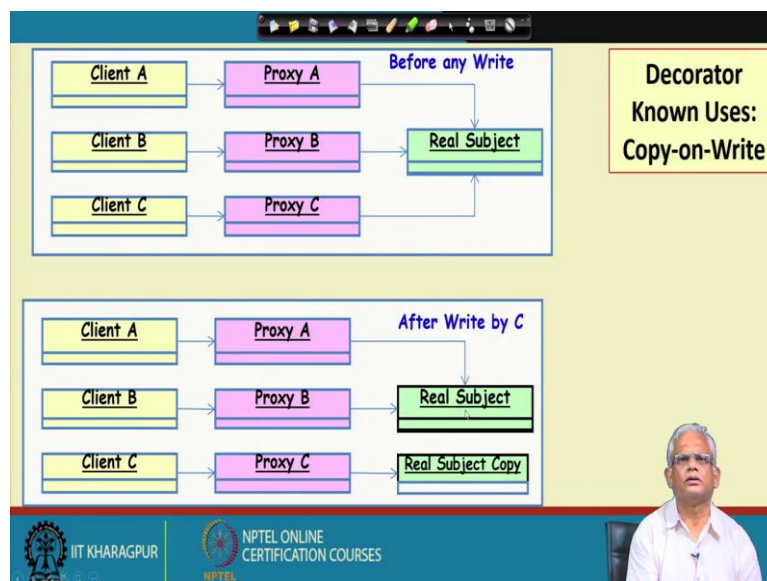
**Uses:
Copy-on-Write**

- Highly optimized String classes often use this approach
- To make this work:
 - Clients are given proxies rather than direct references to the object
- When a write operation occurs:
 - **A proxy makes a private copy of the object on-the-fly to insulate other clients from the changes**

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Optimized string classes use this approach. The clients are given proxies rather than direct references to the object and only when the client tries to write the proxy creates a copy of the string object and gives it to the client.

(Refer Slide Time: 26:24)



This is a representation of the copy and write (in the above slide). Just see here that before any right the clients are sharing the same object, they are just reading and through the proxy, they access the real subject, but when one of the client wants to write, it has got its own object. Here, Proxy C wants to write and therefore it has got its own object and the others are sharing the object.

(Refer Slide Time: 27:03)

Uses:
Reference
Counting

- Proxies maintain the reference count inside the object
- The last proxy to go away is responsible for deleting the object:
 - That is, when the reference count goes to 0, delete the object.

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

Similarly, reference counting(in the above slide). That is how many objects are now sharing an object. So here, the proxies maintain a reference, the reference count itself is inside the real subject, the proxies increase and decrease the reference count as a new proxy tries to access it first increases the count and as it disconnects, it decreases the count. If the proxy is already one, then before going it may delete the object.

(Refer Slide Time: 27:43)

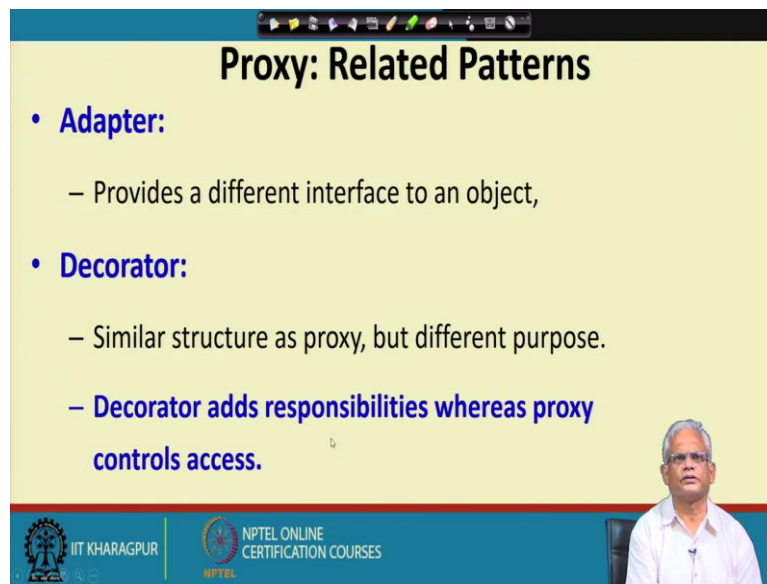
Proxy: Final Analysis

- A Proxy decouples clients from servers.
 - A Proxy introduces a level of indirection.
- **Proxy differs from Adapter in that it does not change the object's interface.**

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

The final analysis, we saw that proxy has many applications. It decouples the client from the server but then it introduces a level of indirection. It differs from the adapter, that it uses the same interface of the server object whereas the adapter. The proxy has the same interface as the real object.

(Refer Slide Time: 28:26)



The slide is titled "Proxy: Related Patterns" and is presented in a yellow-themed format. It lists two related patterns:

- **Adapter:**
 - Provides a different interface to an object,
- **Decorator:**
 - Similar structure as proxy, but different purpose.
 - **Decorator adds responsibilities whereas proxy controls access.**

The slide also features a small video inset of a man in a white shirt in the bottom right corner. At the bottom, there are logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES.

There are related pattern (in the above slide). The adapter provides different interface to an object as we have already seen. An adapter changes the interface to a different interface and we will look at the decorator pattern in the next lecture. It is also similar to the proxy pattern but for different purpose. Decorator is a very powerful pattern. Here you can add responsibilities, whereas the proxy controls access to the object. As we look at the decorator in the next lecture, this aspect will be clear.

We are already at the end of this lecture. We will stop here and continue in the next lecture. Thank you.