# Lecture 21 : Proxy Design Pattern

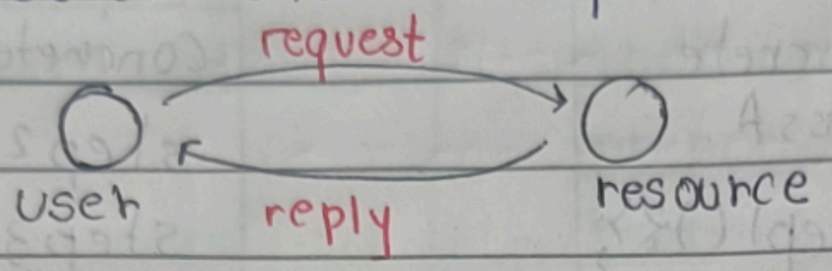
Page No.
Date

## # Introduction

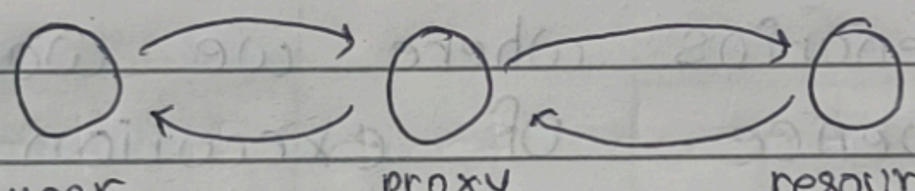
- It is also use case specific design pattern

## # What is Proxy Design Pattern

- let's assume user requests for resource and resource share response.




- Here we don't want direct communication btw user and resource, isliye humne ek object introduce kiya called proxy (proxy of resource)




- Ab user proxy ko request krega. Proxy validate karega request and share karega to resource & resource reply krega proxy ko & proxy share krega user ko.

## # Why Proxy is introduced?

1) Suppose humara resource ke pass critical data hai & hum nhi chahte ki usse koi bhi access kare directly toh humne proxy introduce kiya job validate & authenticate krega user ko then communicate krne dega.

2) To implement checks on data sent by user

3) Resource humara dusre server pr hai over internet but proxy hum locale object-banate joh communication ~~exis~~ krta hai with resource through connection

* Proxy behaves like resource user can't differentiate between proxy & resource & who they are talking.

# Types of Proxy.

a) Virtual Proxy
b) Protection Proxy
c) Remote Proxy

} core concept is same.

⇒ Proxy is representative of resource.

# Virtual Proxy

- let's understand this with help of an example of displaying an image

| ImageDisplay |
| --- |
| string path;<br>ImageDisplay(path)<br>{};<br>display(){}; |

```
ImageDisplay(path){
    this->path = path;
    //① Load from disc;
    //② Compression;
    //③ filter;
}
```

- Ab jab client display() call karega toh image display hoga according to path specified

- Pr load ~~kiya~~ krne ke pehle usse kuch task/ operation perform karne padege jo humne pass kiye hai constructor mein.

- If we pass all this operations in display()<> so it will take time to display image so we pass it in constructor.

Client code →

```
ImageDisplay *img = new ImageDisplay(b);
                img -> display();
```

↳ Client calls display method

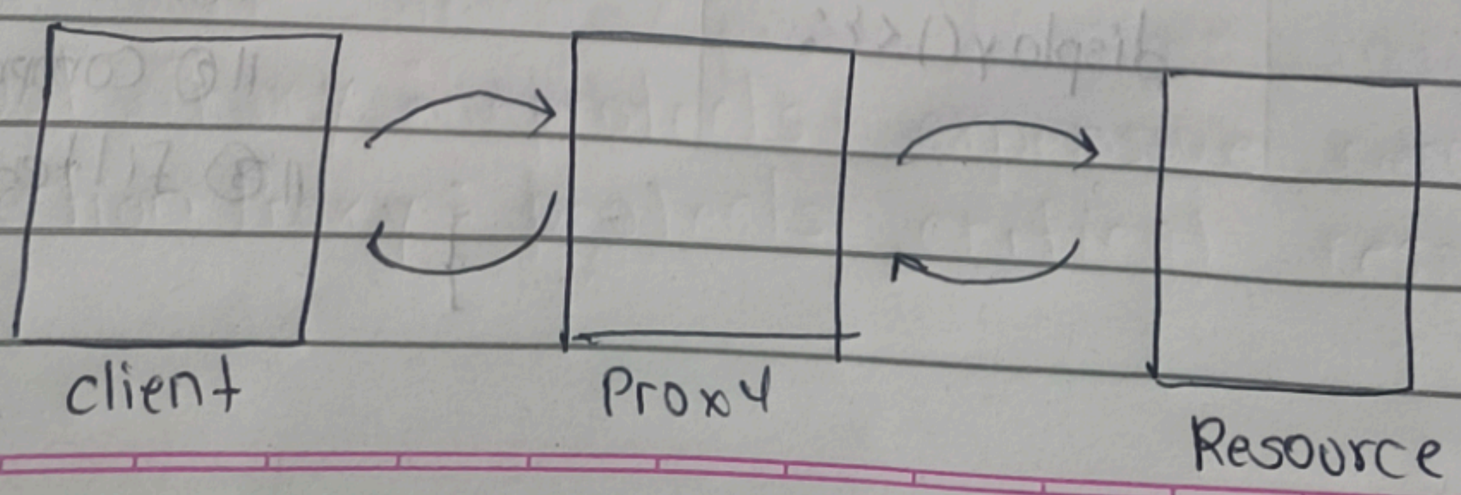
It will perform expensive operation

- Par what if client doesn't call display method?

↳ Object toh create hoga server par & expensive operation ke karan time bhi waste hoga.

- Yeh bss example hai but large applications mein bhi hota hai aisa.

⇒ Isliye we introduced proxy between client and resource.

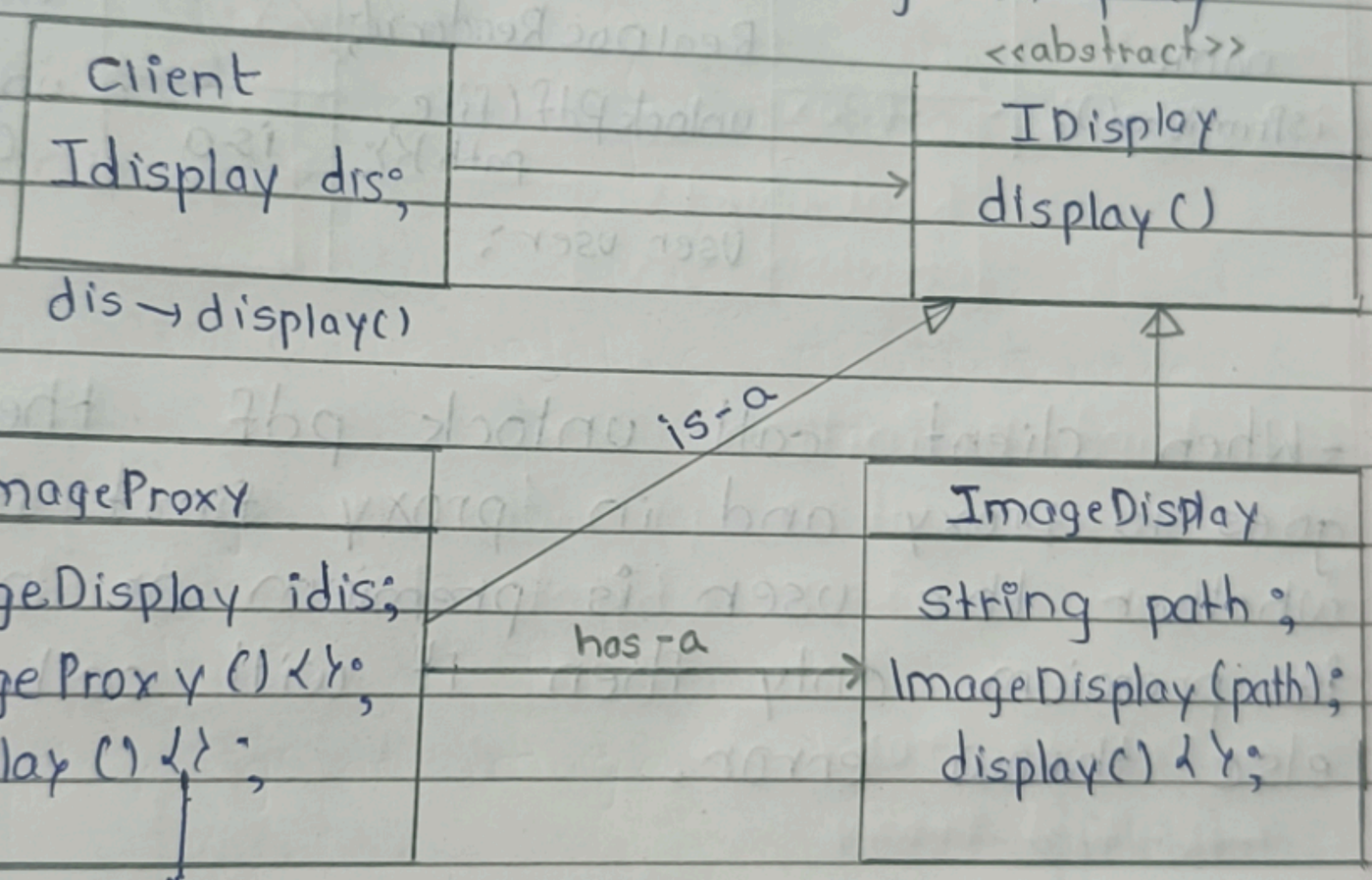


-Proxy introduce krne se ab - proxy refrence hold krega ImageDisplay ka and directly object nhi banayega. It will only create object when client calls display method.

# UML Diagram for Image Display

- Pehle hum superclass / interface banayege for ImageDisplay taaki future mein pdfdisplay, docDisplay bhi ho.

- Proxy will have reference of ImageDisplay & proxy will behave like Image Display

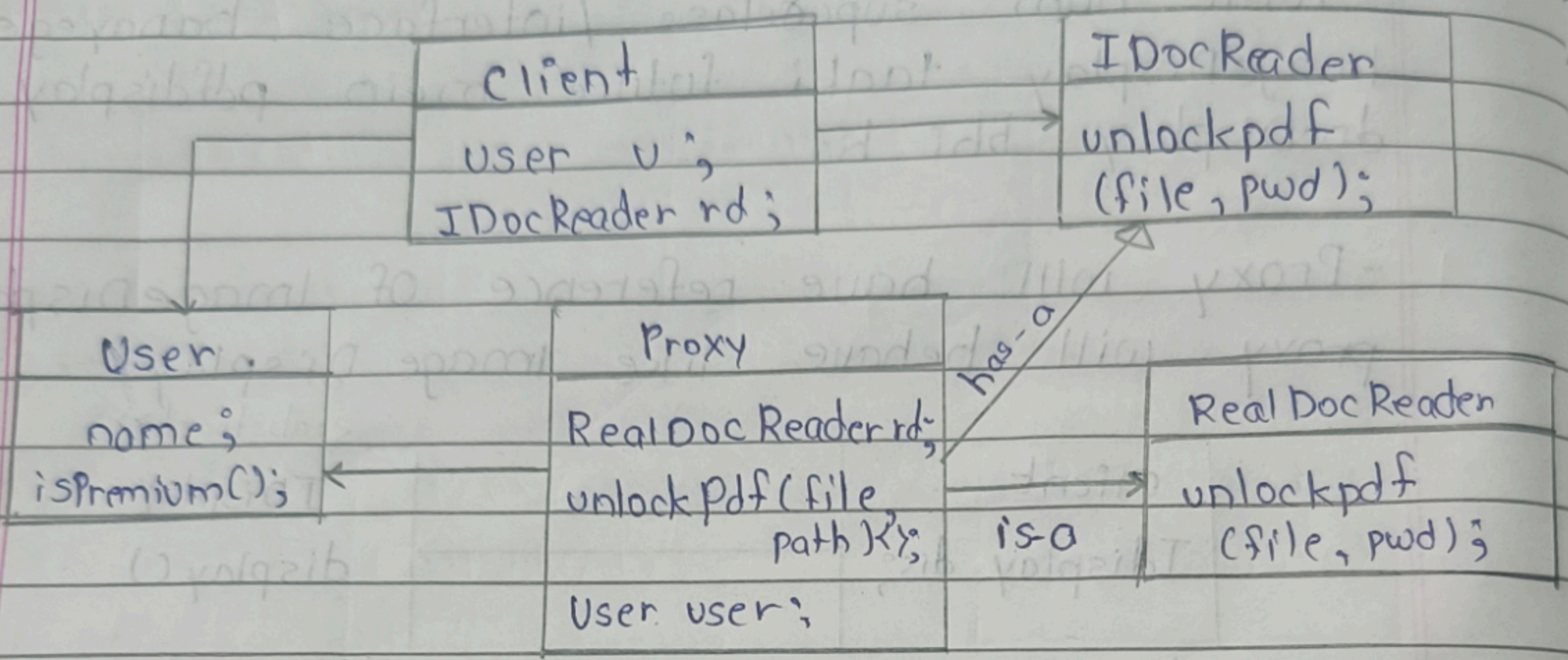



→ client call kreaga Idisplay ko, client ko nhi pata hoga that what we have passed ImageProxy or ImageDisplay but we will always pass ImageProxy

# Protection Proxy

- Let's understand with example of DocReader.

- Humne ek class baniya having feature to unlock pdf. But humme jab hai feature limit krna hai premium user ke liye, taaki har koi usse access naa kar paye.




- When client call unlock pdf then request goes to proxy and in proxy first it check whether the user is premium or not. If user is premium only then it will call unlockpdf() else throw error.

Conclusion

Protection proxy authenticate the user so that everybody can't access those resource over the internet.

- Virtual Proxy : protects expensive resources.
- Protection Proxy : protects critical resources.

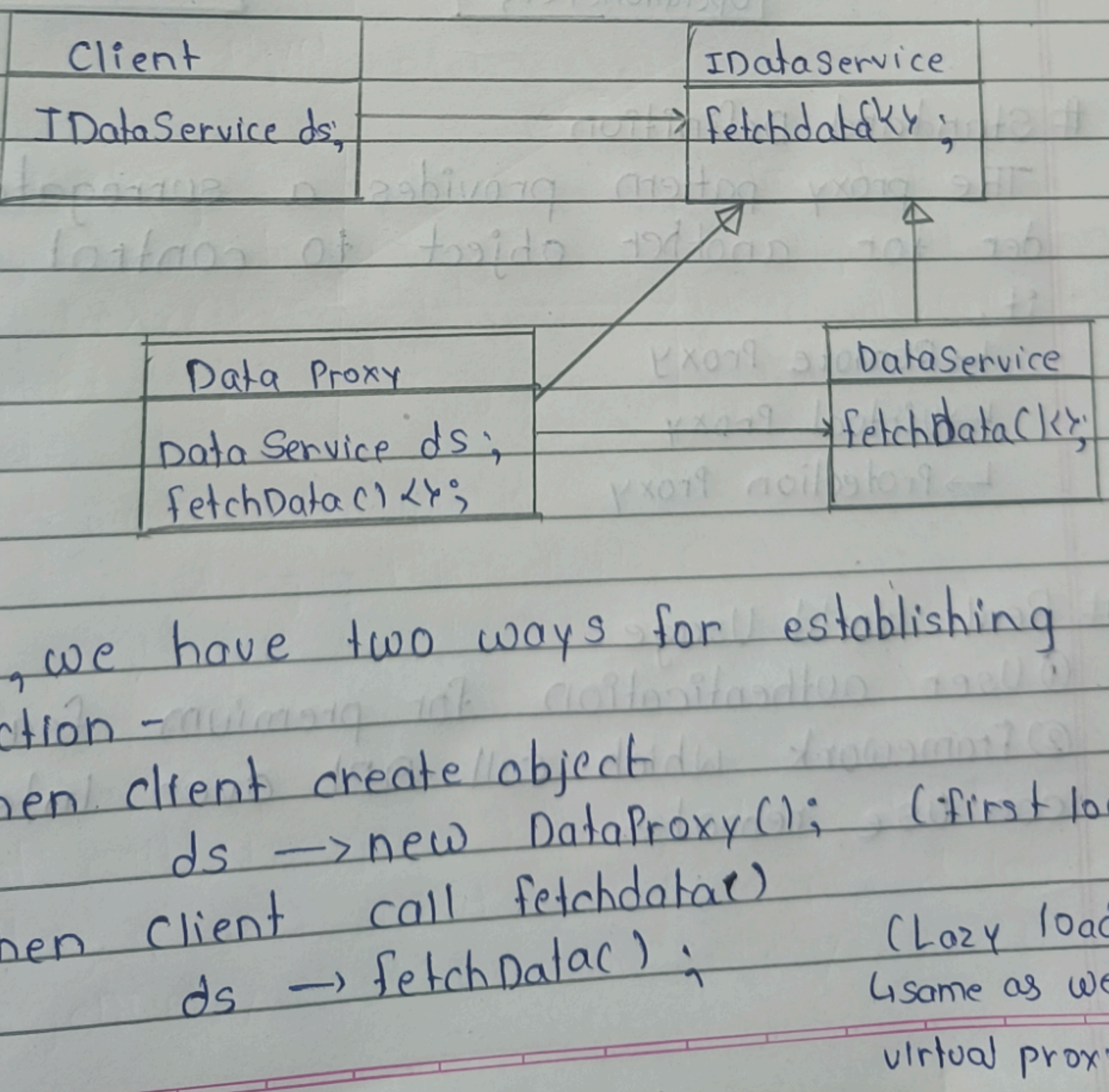
# Remote Proxy

- It become the representative of those object which exist on another server

- Here, we make a class DataService which exist on another server having method fetch(). and client want to use this method.

- If we don't introduce proxy here client should know about server and then establish connection to call this method.

- So, we introduce DataProxy bcz we don't want that client should know about server and all.




- Here, we have two ways for establishing connection -

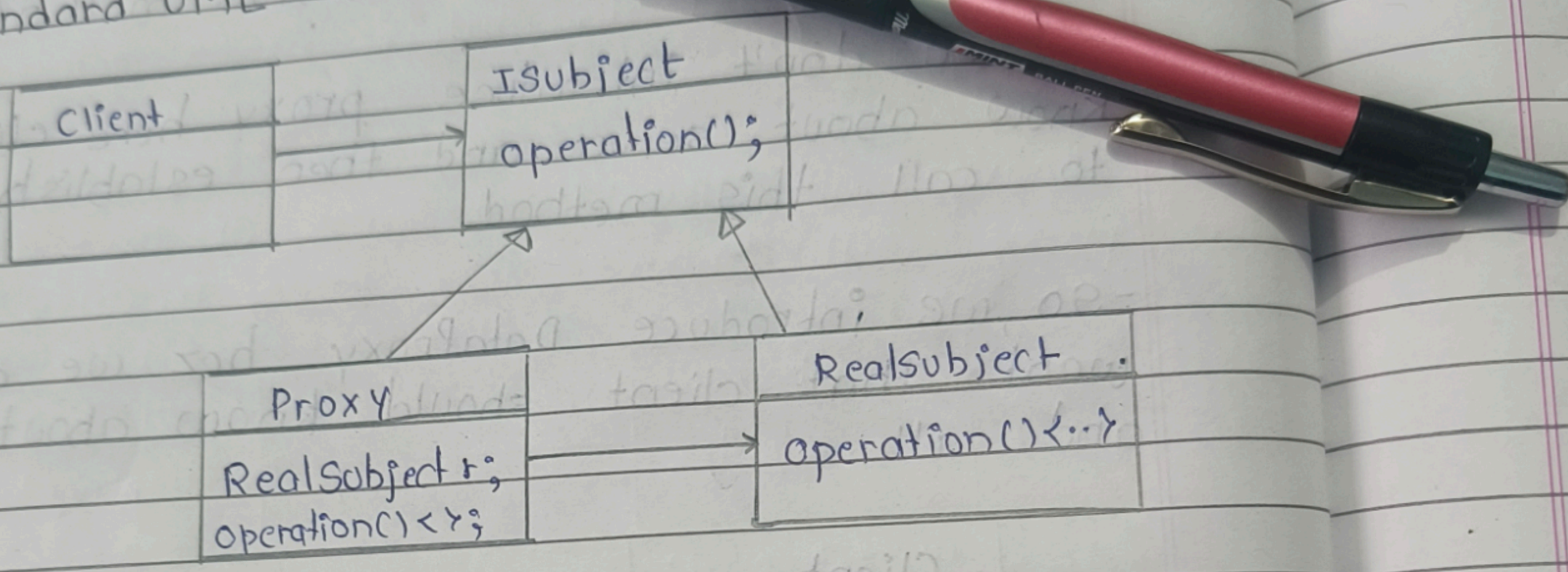
① When client create object
ds → new DataProxy(); (first loading)

② When client call fetchdata()
ds → fetchData(); (Lazy loading)
(same as we do in virtual proxy

- Here we follow lazy loading when client call method only then we establish connection.

Conclusion : Remote Proxy become the representative of that resource which exist somewhere else over the Internet

# Standard UML




# Standard Definition

The proxy pattern provides a surrogate or placeholder for another object to control access to it.

↳ Remote Proxy
↳ Virtual Proxy
↳ Protection Proxy

# Real World Usage

① User authentication for premium feature.
② Framework which call API over Internet so we can use remote.