SHRI RAMDEOBABA COLLEGE OF ENGINEERING AND MANAGEMENT, NAGPUR - 440013

DESIGN PATTERNS
V SEMESTER

COURSE COORDINATOR: RINA DAMDOO

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROXY DESIGN PATTERNS

- **Intent**

  Provide a surrogate or placeholder for another object to control access to it.

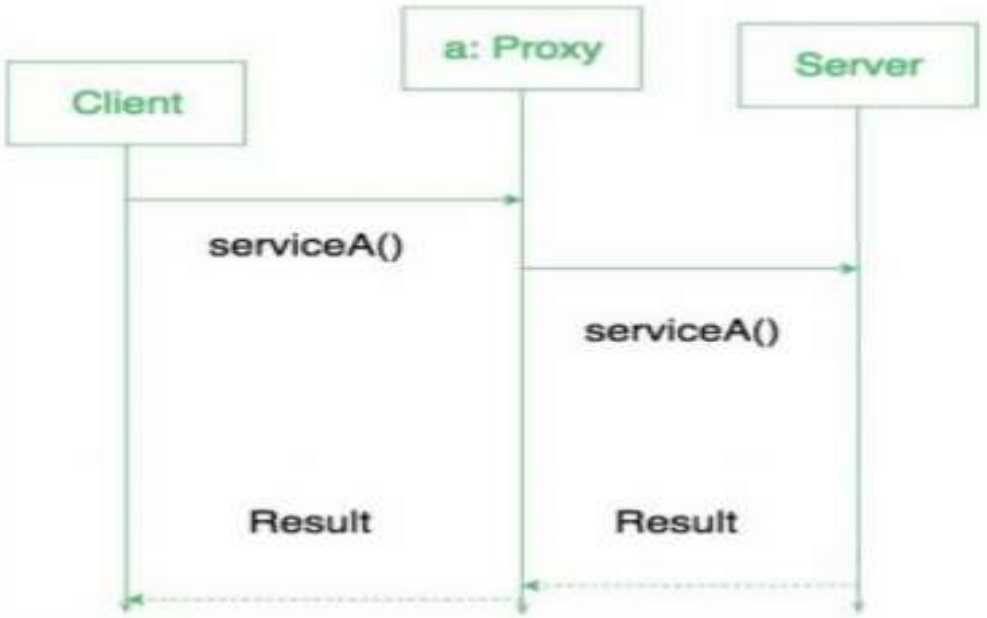
Also Known As: Surrogate

OBJECTIVE

- One reason for controlling access to an object is to defer the full cost of its creation and initialization until we actually need to use it.
- Consider a document editor that can embed graphical objects in a document. Some graphical objects, like large raster images, can be expensive to create.
- Opening a document should be fast, so we should avoid creating all the expensive objects at once when the document is opened. This isn't necessary anyway, because not all of these objects will be visible in the document at the same time.

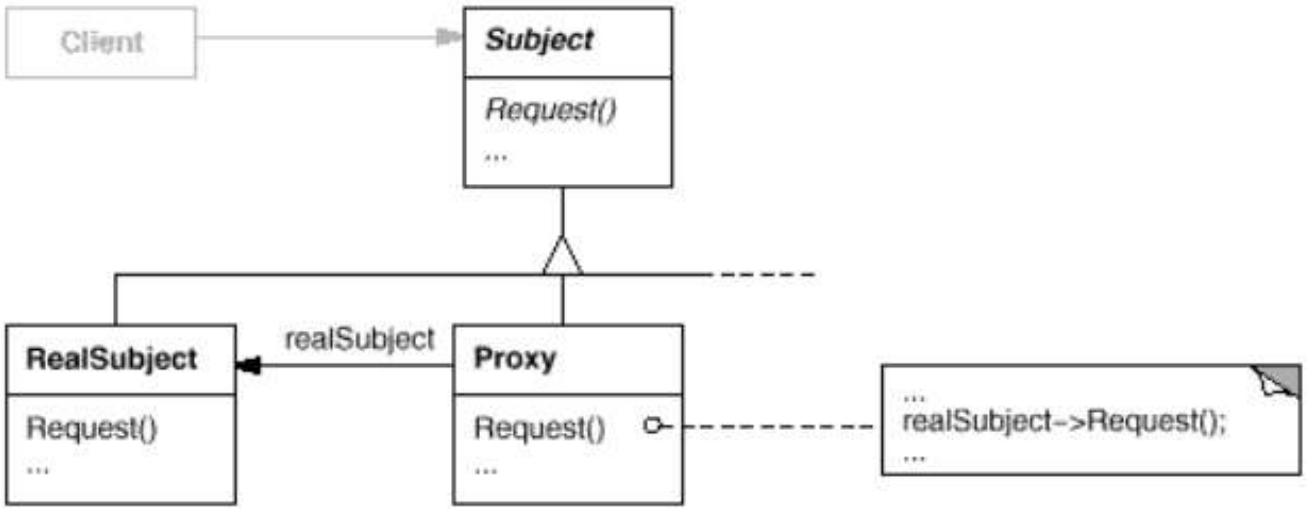
PROXY DESIGN PATTERNS

Problem Statement:
There can be numerous applications of PROXY DESIGN PATTERNS
We can discuss Client-Server connection through Proxy.

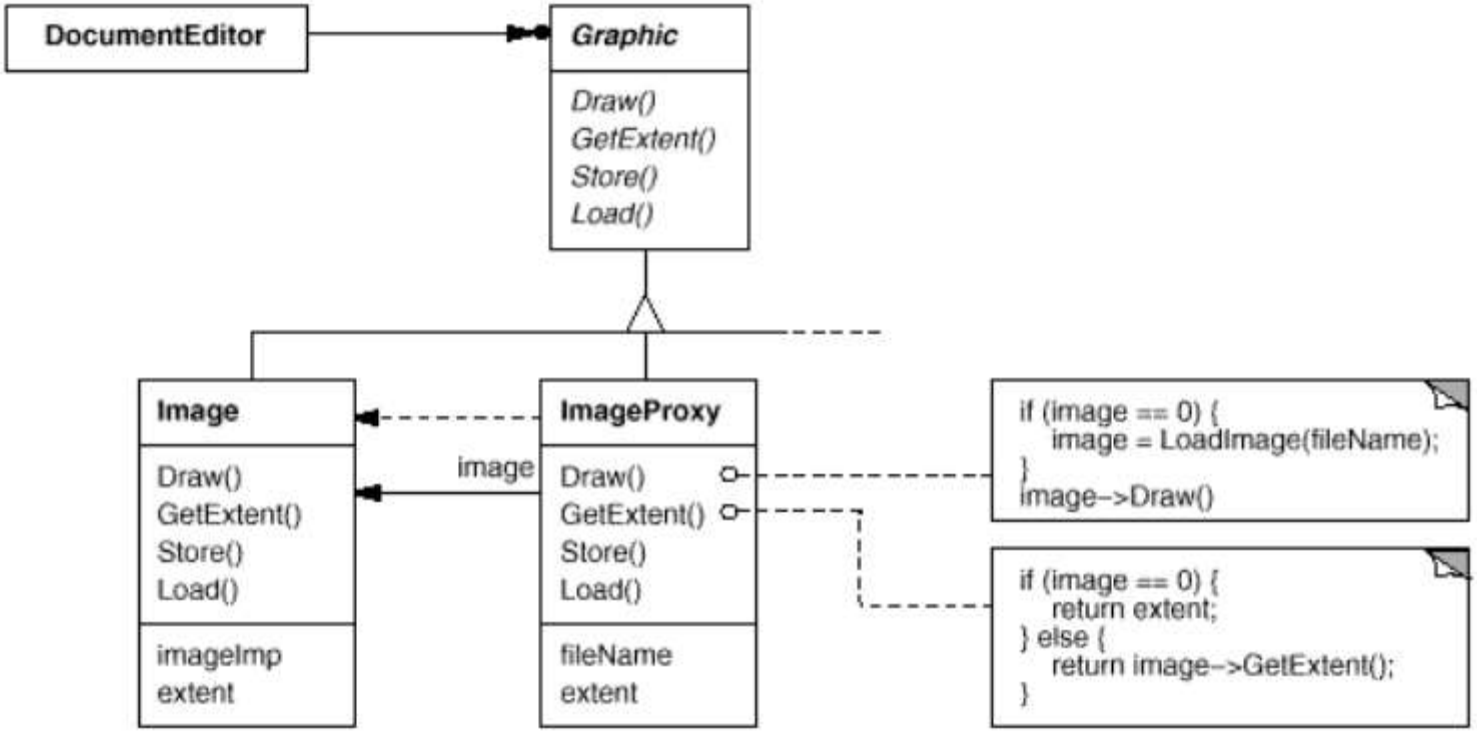


Proxy Design Patterns

Structure



MOTIVATION


DocumentEditor
Graphic
Draw()
GetExtent()
Store()
Load()
Image
Draw()
GetExtent()
Store()
Load()
imageImp
extent
image
ImageProxy
Draw()
GetExtent()
Store()
Load()
fileName
extent
if (image == 0) {
image = LoadImage(fileName);
}
image->Draw()
if (image == 0) {
return extent;
} else {
return image->GetExtent();
}

Implementation Variants

```
public class ProxyInternet implements Internet
{
    private Internet internet = new RealInternet();
    private static List<String> bannedSites;

    static
    {
        bannedSites = new ArrayList<String>();
        bannedSites.add("abc.com");
        bannedSites.add("def.com");
        bannedSites.add("ijk.com");
        bannedSites.add("lnm.com");
    }

    @Override
    public void connectTo(String serverhost) throws Exception
    {
        if(bannedSites.contains(serverhost.toLowerCase()))
        {
            throw new Exception("Access Denied");
        }

        internet.connectTo(serverhost);
    }

}
```

Reference: www.geeksforgeeks.org

APPLICABILITY

Use the Proxy pattern when:

- ????????what are different types of proxies ??????????

Proxies are generally divided into four types –

• **Remote proxy** – represent a remotely loctated object. To talk with remote objects, the client need to do additional work on communication over network. A proxy object does this communication on behalf of original object and client focuses on real task to do.

• **Virtual proxy** – delay the creation and initialization of expensive objects until needed, where the objects are created on demand. Hibernate created proxy entities are example of virtual proxies.

APPLICABILITY

- **Protection proxy** – help to implement security over original object. They may check for access rights before method invocations and allow or deny access based on the conclusion.

- **Smart Proxy –** performs additional housekeeping work when an object is accessed by a client. An example can be to check if the real object is locked before it is accessed to ensure that no other object can change it.

A **smart reference** is a replacement for a bare pointer that performs additional actions when an object is accessed. Typical uses include

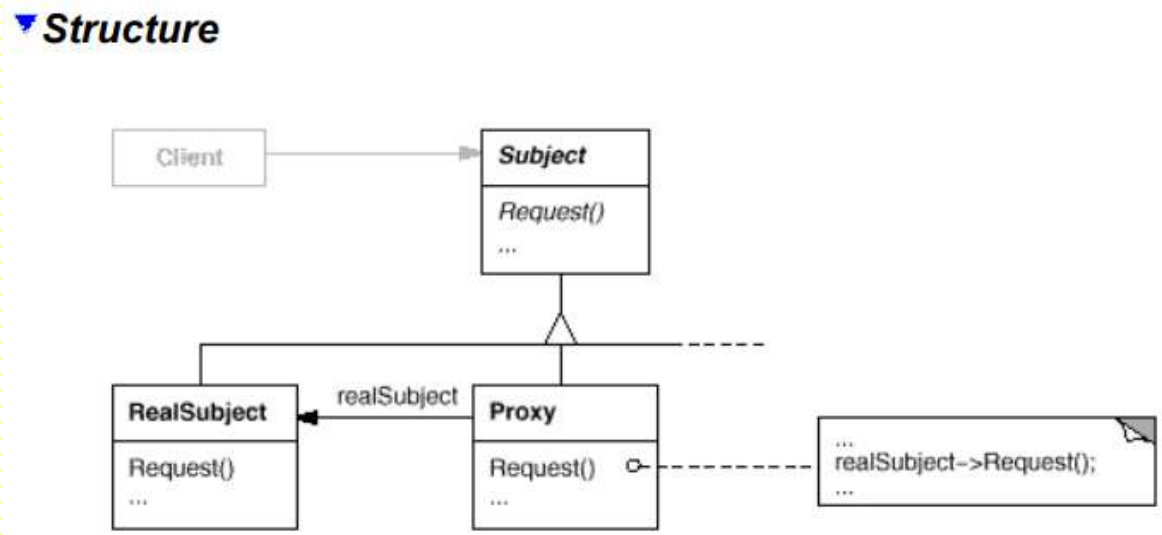
- counting the number of references to the real object so that it can be freed automatically when there are no more references (also called **smart pointers**.
- loading a persistent object into memory when it's first referenced.

PARTICIPANTS

Participant	Responsibility
Proxy	• maintains a reference that lets the proxy access the real subject. Proxy may refer to a Subject if the RealSubject and Subject interfaces are the same. • provides an interface identical to Subject's so that a proxy can by substituted for the real subject. • controls access to the real subject and may be responsible for creating and deleting it. • other responsibilities depend on the kind of proxy
Subject	• defines the common interface for RealSubject and Proxy so that a Proxy can be used anywhere a RealSubject is expected.
RealSubject	• defines the real object that the proxy represents.

COLLABORATIONS

•Proxy forwards requests to RealSubject when appropriate, depending on the kind of proxy.



CONSEQUENCES

- The Proxy pattern introduces a level of indirection when accessing an object.
- The additional indirection has many uses, depending on the kind of proxy:

    1. A remote proxy can hide the fact that an object resides in a different address space.
    2. A virtual proxy can perform optimizations such as creating an object on demand.
    3. Both protection proxies and smart references allow additional housekeeping tasks when an object is accessed.

IMPLEMENTATION

Implementation Consider the following issues when applying the Proxy pattern:

- **Few issues are language dependent (C++ and Smalltalk)**
- Proxy doesn't always have to know the type of real subject. If a Proxy class can deal with its subject solely through an abstract interface, then there's no need to make a Proxy class for each RealSubject class; the proxy can deal with all RealSubject classes uniformly. But if Proxies are going to instantiate RealSubjects (such as in a virtual proxy), then they have to know the concrete class.

Known Uses & Related Patterns

- **Proxy servers**
- **ATM Machines**
- **Credit/Debit cards**
- **Hibernate proxy**
- **Java Persistence API:** The JPA lazy loading mechanism can be implemented using Proxies

- **Adapter**
- **Decorator**