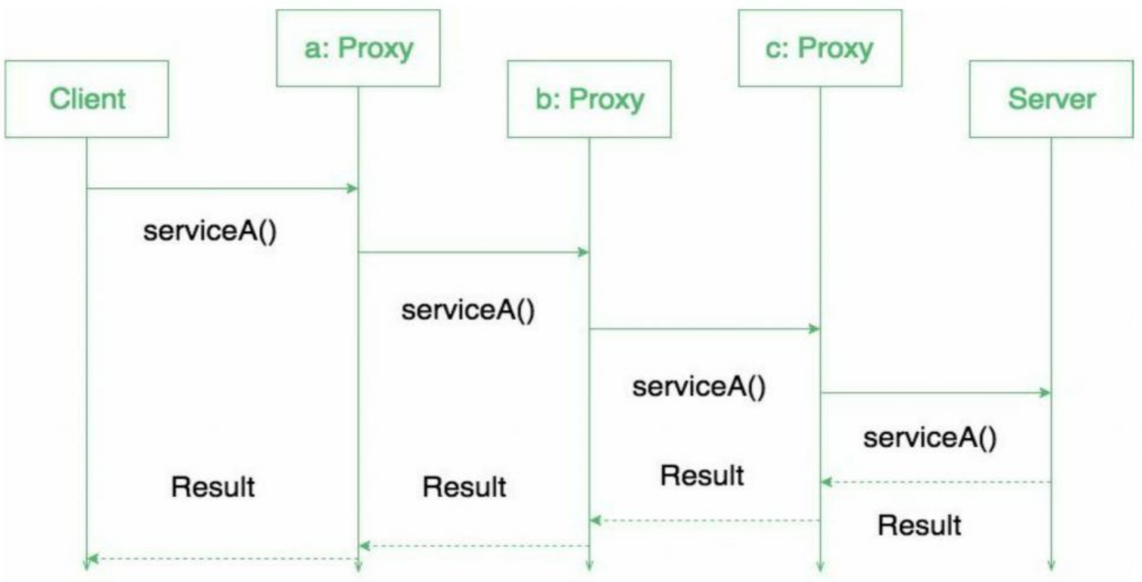
Proxy Design Pattern

Proxy means ‘in place of’, representing’ or ‘in place of’ or ‘on behalf of’ are literal meanings of proxy and that directly explains **Proxy Design Pattern**.

Proxies are also called surrogates, handles, and wrappers. They are closely related in structure, but not purpose, to Adapters and Decorators.

A real world example can be a cheque or credit card is a proxy for what is in our bank account. It can be used in place of cash, and provides a means of accessing that cash when required. And that’s exactly what the Proxy pattern does – “**Controls and manage access to the object they are protecting**“.

As in the decorator pattern, proxies can be chained together. The client, and each proxy, believes it is delegating messages to the real server:



When to use this pattern?

Proxy pattern is used when we need to create a wrapper to cover the main object’s complexity from the client.

Remote proxy:

They are responsible for representing the object located remotely. Talking to the real object might involve marshalling and unmarshalling of data and talking to the remote object. All that logic is encapsulated in these proxies and the client application need not worry about them.

Virtual proxy:

These proxies will provide some default and instant results if the real object is supposed to take some time to produce results. These proxies initiate the operation on real objects and provide a default result to the application. Once the real object is done, these proxies push the actual data to the client where it has provided dummy data earlier.

Protection proxy:

If an application does not have access to some resource then such proxies will talk to the objects in applications that have access to that resource and then get the result back.

Smart Proxy:

A smart proxy provides an additional layer of security by interposing specific actions when the object is accessed. 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.

Some Examples

A very simple real life scenario is our college internet, which restricts few site access. The proxy first checks the host you are connecting to, if it is not part of restricted site list, then it connects to the real internet. This example is based on Protection proxies.

Lets see how it works :

Interface of Internet

```
package com.saket.demo.proxy;

public interface Internet
{
	public void connectTo(String serverhost) throws Exception;
}
```

RealInternet.java

```
package com.saket.demo.proxy;

public class RealInternet implements Internet
{
	@Override
	public void connectTo(String serverhost)
	{
		System.out.println("Connecting to "+ serverhost);
	}
}
```

ProxyInternet.java

```
package com.saket.demo.proxy;

import java.util.ArrayList;
import java.util.List;


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);
	}

}
```

Client.java

```
package com.saket.demo.proxy;

public class Client
{
	public static void main (String[] args)
	{
		Internet internet = new ProxyInternet();
		try
		{
			internet.connectTo("geeksforgeeks.org");
			internet.connectTo("abc.com");
		}
		catch (Exception e)
		{
			System.out.println(e.getMessage());
		}
	}
}
```

As one of the site is mentioned in the banned sites, So

Running the program will give the output :

```
Connecting to geeksforgeeks.org
Access Denied
```

Benefits:

- One of the advantages of Proxy patterns is security.
- This pattern avoids duplication of objects which might be huge size and memory intensive. This in turn increases the performance of the application.
- The remote proxy also ensures security by installing the local code proxy (stub) in the client machine and then accessing the server with help of the remote code.

Drawbacks/Consequences:

This pattern introduces another layer of abstraction which sometimes may be an issue if the RealSubject code is accessed by some of the clients directly and some of them might access the Proxy classes. This might cause disparate behaviour.

Interesting points:

- There are few differences between the related patterns. Like Adapter pattern gives a different interface to its subject, while Proxy patterns provides the same interface from the original object but the decorator provides an enhanced interface. Decorator pattern adds additional behaviour at runtime.

- Proxy used in Java API: java.rmi.*;