Proxy Pattern

Simply, proxy means an object representing another object.

According to GoF, a Proxy Pattern **"provides the control for accessing the original object".**

So, we can perform many operations like hiding the information of original object, on demand loading etc.

RMI API uses proxy design pattern. Stub and Skeleton are two proxy objects used in RMI.

Advantage of Proxy Pattern

* It provides the protection to the original object from the outside world.

Usage of Proxy Pattern:

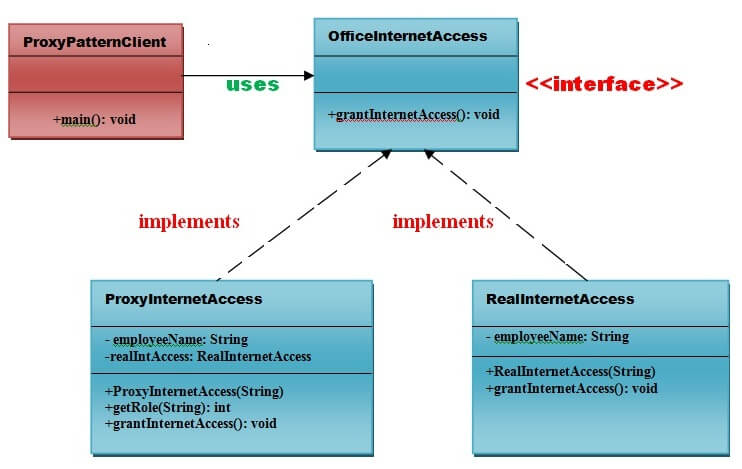
It is used:

* It can be used in **Virtual Proxy** scenario---Consider a situation where there is multiple database call to extract huge size image. Since this is an expensive operation so here we can use the proxy pattern which would create multiple proxies and point to the huge size memory consuming object for further processing. The real object gets created only when a client first requests/accesses the object and after that we can just refer to the proxy to reuse the object. This avoids duplication of the object and hence saving memory.
* It can be used in **Protective Proxy** scenario---It acts as an authorization layer to verify that whether the actual user has access the appropriate content or not. For example, a proxy server which provides restriction on internet access in office. Only the websites and contents which are valid will be allowed and the remaining ones will be blocked.
* It can be used in **Remote Proxy** scenario---A remote proxy can be thought about the stub in the RPC call. The remote proxy provides a local representation of the object which is present in the different address location. Another example can be providing interface for remote resources such as web service or REST resources.
* It can be used in **Smart Proxy** scenario---A smart proxy provides additional layer of security by interposing specific actions when the object is accessed. For example, to check whether the real object is locked or not before accessing it so that no other objects can change it.

Example of Proxy Pattern

Let's understand the example of proxy design pattern by the above UML diagram.

UML for Proxy Pattern:



Implementation of above UML:

Step 1

Create an **OfficeInternetAccess** interface.

1. **public** **interface** OfficeInternetAccess {
2. **public** **void** grantInternetAccess();
3. }

Step 2

Create a **RealInternetAccess** class that will implement **OfficeInternetAccess** interface for granting the permission to the specific employee.

*File: RealInternetAccess.java*

1. **public** **class** RealInternetAccess **implements** OfficeInternetAccess {
2. **private** String employeeName;
3. **public** RealInternetAccess(String empName) {
4. **this**.employeeName = empName;
5. }
6. @Override
7. **public** **void** grantInternetAccess() {
8. System.out.println("Internet Access granted for employee: "+ employeeName);
9. }
10. }

Step 3

Create a **ProxyInternetAccess** class that will implement **OfficeInternetAccess** interface for providing the object of **RealInternetAccess** class.

*File: ProxyInternetAccess.java*

1. **public** **class** ProxyInternetAccess **implements** OfficeInternetAccess {
2. **private** String employeeName;
3. **private** RealInternetAccess  realaccess;
4. **public** ProxyInternetAccess(String employeeName) {
5. **this**.employeeName = employeeName;
6. }
7. @Override
8. **public** **void** grantInternetAccess()
9. {
10. **if** (getRole(employeeName) > 4)
11. {
12. realaccess = **new** RealInternetAccess(employeeName);
13. realaccess.grantInternetAccess();
14. }
15. **else**
16. {
17. System.out.println("No Internet access granted. Your job level is below 5");
18. }
19. }
20. **public** **int** getRole(String emplName) {
21. // Check role from the database based on Name and designation
22. // return job level or job designation.
23. **return** 9;
24. }
25. }

Step 4

Now, Create a **ProxyPatternClient** class that can access the internet actually.

*File: ProxyPatternClient.java*

1. **public** **class** ProxyPatternClient {
2. **public** **static** **void** main(String[] args)
3. {
4. OfficeInternetAccess access = **new** ProxyInternetAccess("Ashwani Rajput");
5. access.grantInternetAccess();
6. }
7. }