The Proxy Pattern is a design pattern that falls under the structural design patterns category.

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

Proxy: It is an object that acts as an interface to another object, known as the "real object."

The proxy object controls access to the real object, allowing you to add additional functionality

such as lazy initialization, access control, logging, or caching without changing the real object's code.

Real Subject/Object: This is the object that the proxy represents.

The real subject/object is typically a resource-intensive object or an object located remotely, and the proxy controls access to it.

Client: The client interacts with the proxy object as if it were the real object.

The client doesn't need to know whether it's dealing with a proxy or the real object.

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

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

*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(serverhost + " : Access Denied");
 }

 internet.connectTo(serverhost);
 }

}

*public class* TestProxy {
 *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());
 }
 }
}

Connecting to geeksforgeeks.org

abc.com : Access Denied

Remote Proxy: It acts as a local representative for an object that resides in a different address space. This allows the client to interact with the remote object as if it were local.

Virtual Proxy: It creates expensive objects on demand. For example, when dealing with large images or documents, you might not want to load the entire object into memory until necessary. The proxy loads the object only when the client requests it.

Protection Proxy: It controls access to the real object based on access rights. For example, if an object should only be accessed by certain users or after certain conditions are met, the protection proxy enforces these rules before allowing access to the real object.

Logging Proxy: It adds logging functionality to keep track of the calls made to the real object.

Caching Proxy: It stores the results of expensive operations and returns the cached result when the same operation is requested again, instead of recalculating it.