🔒

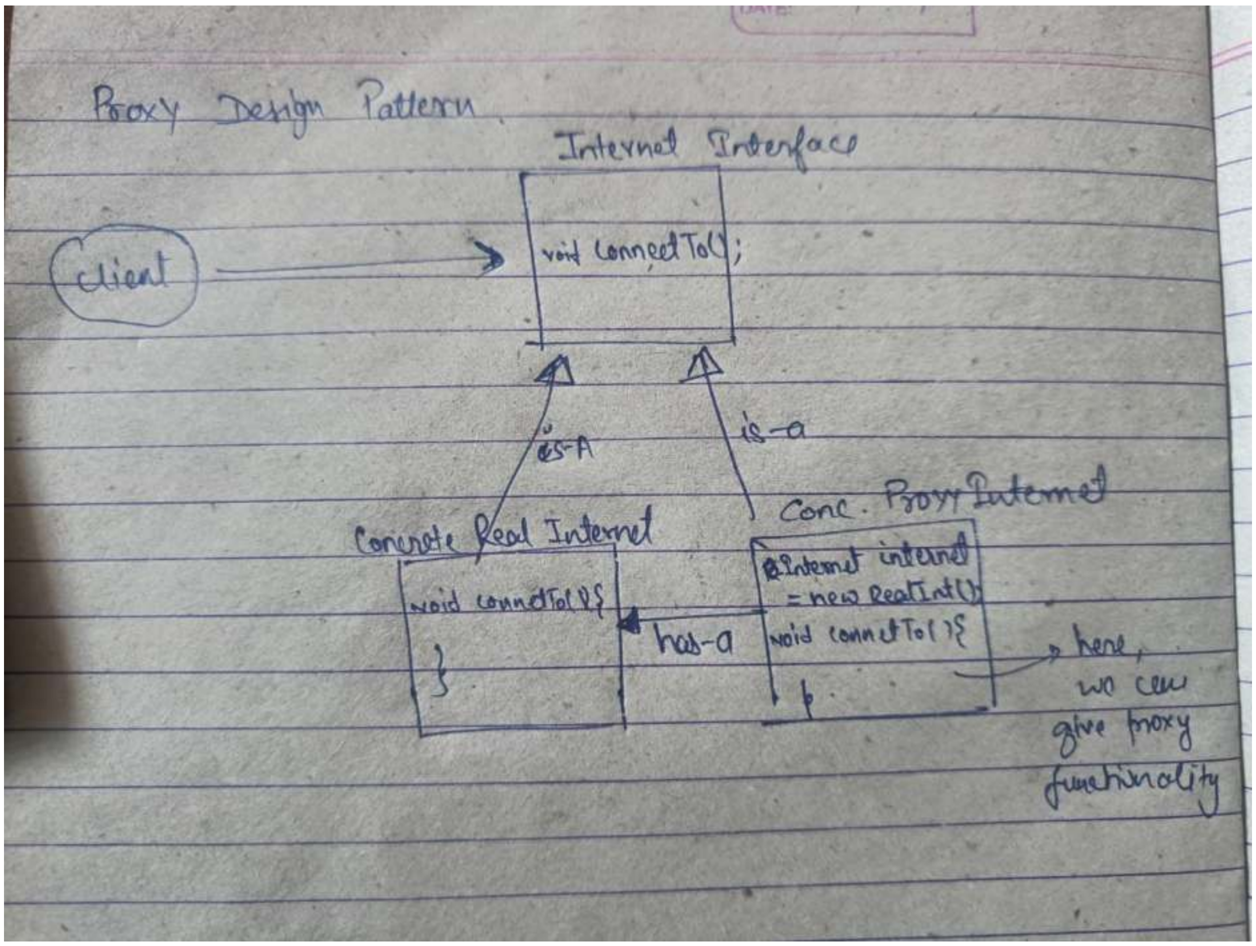
5. Proxy Design Pattern

Proxy design pattern is used to provide extra layer to our handler object to reduce it complexity or perform some security check.

Ex:

1. Network Proxy / Access Restriction.
2. Caching
3. Pre-Processing (like Middlewares) or Post-Processing

UML Diagram:



Code:

```
// Sachin Mahawar
#include <bits/stdc++.h>
using namespace std;

class Internet
{
public:
    virtual void connectTo(string hostname) = 0;
};

class RealInternet : public Internet
{
public:
    void connectTo(string hostname)
    {
        cout << "Connecting to " << hostname << "\n";
    }
};
class ProxyInternet : public Internet
{
    static vector<string> bannedSites;
    Internet *internet = new RealInternet();
```

```cpp
public:
    void connectTo(string hostname)
    {
        if (find(bannedSites.begin(), bannedSites.end(), hostname) == bannedSites.end())
        {
            internet->connectTo(hostname);
        }
        else
        {
            cout << "[" << hostname << "] Access denied!\n";
        }
    }
};

vector<string>
    ProxyInternet::bannedSites = {"abc.com", "google.com", "xyz.com"};

int main()
{
    Internet *internet = new ProxyInternet();
    vector<string> servers = {
        "google.com",
        "gsd.com",
        "abc.com",
    };

    for (auto &x : servers)
    {
        internet->connectTo(x);
    }

    return 0;
}
```