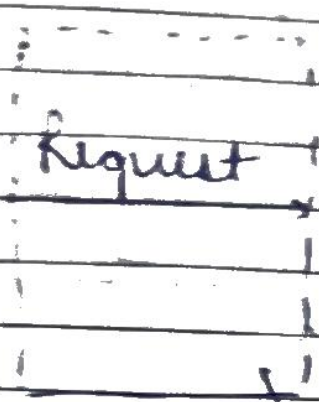
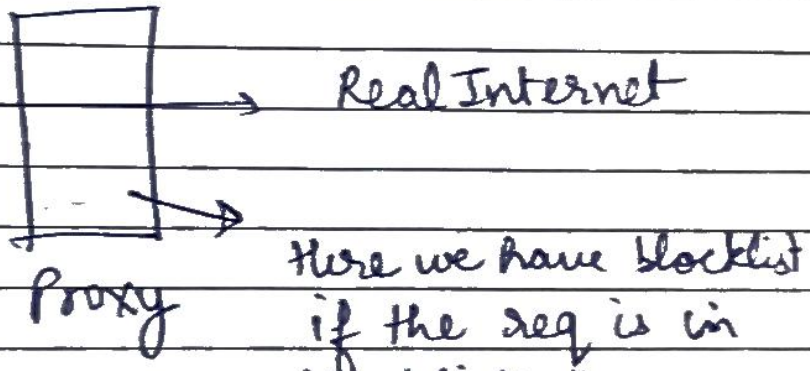
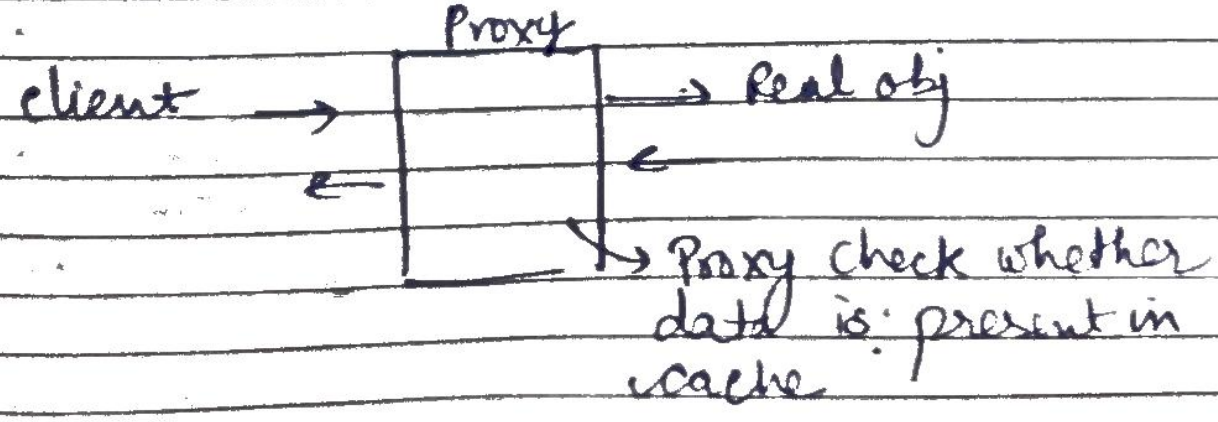
# PROXY DESIGN PATTERN




- UseCase :-
  1. Access Restriction.

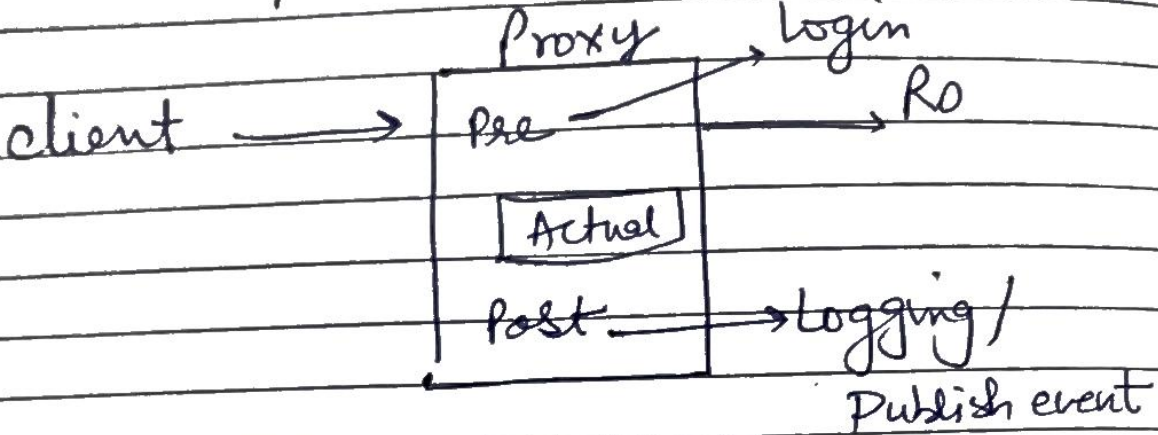



2. Used in caching



Page No.
Date / /

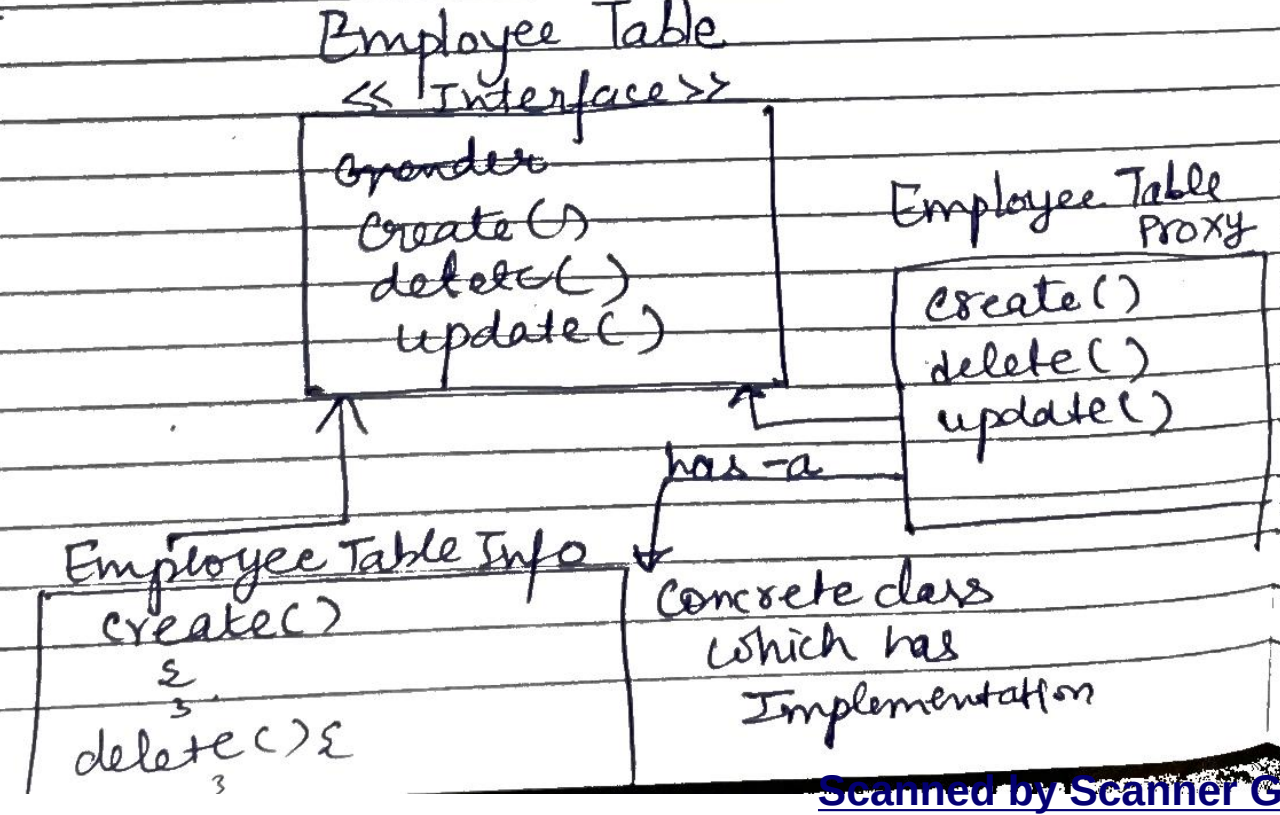
3. Used in Pre-processing and Post-Processing




→ Basically this pattern is useful when you want to add an extra layer of control over access to a real object.

→ Proxy act as intermediary, controlling access to real object.

Ex



Page No.
Date / /

Employee Table Proxy has an object of Employee Table concrete class

When a request for CRUD operations is done by client, first proxy will validate() it first and then after doing that it will call the actual function from concrete class

In Employee Proxy

```
void create(String client, EmployeeDo obj)
    throws Exception {
        if (client.equals('Admin')
        {
            employeeDBobj.create(client, obj);
        }
        throw new Exception("Access Denied");
}
```

Similarly we have implementations for delete, update functions.

- Here we can Also have multiple proxy chain

Client → Proxy 1 → Proxy 2 → Proxy 3 → so on

Chain of Responsibility Principle