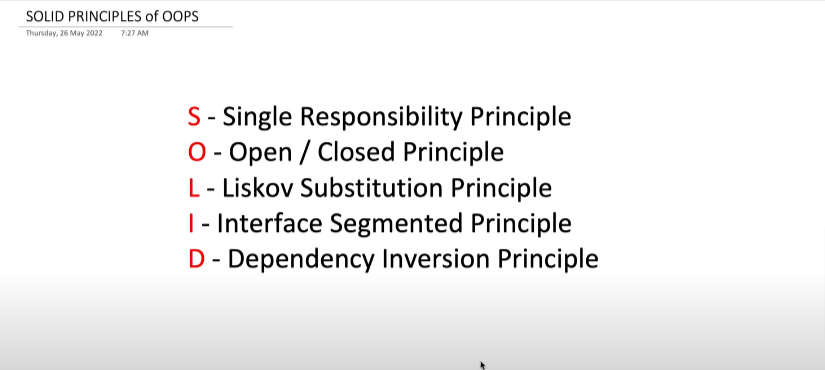
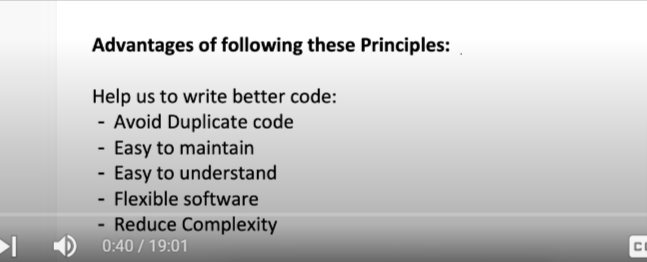
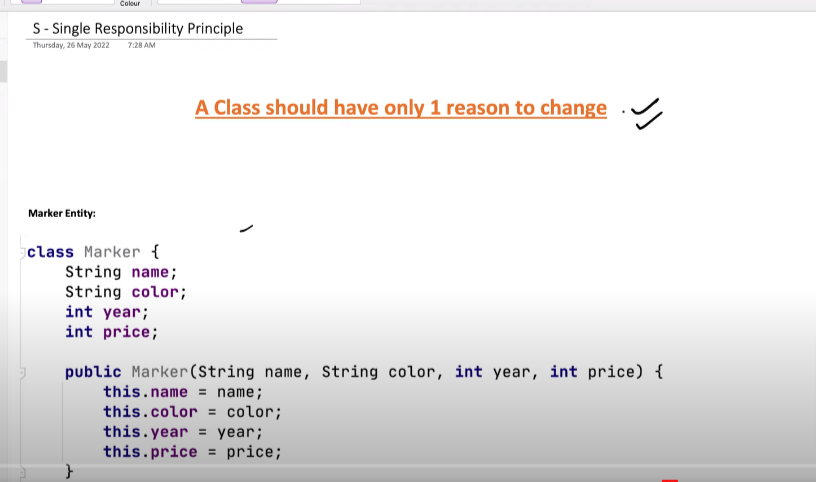
Low Level design

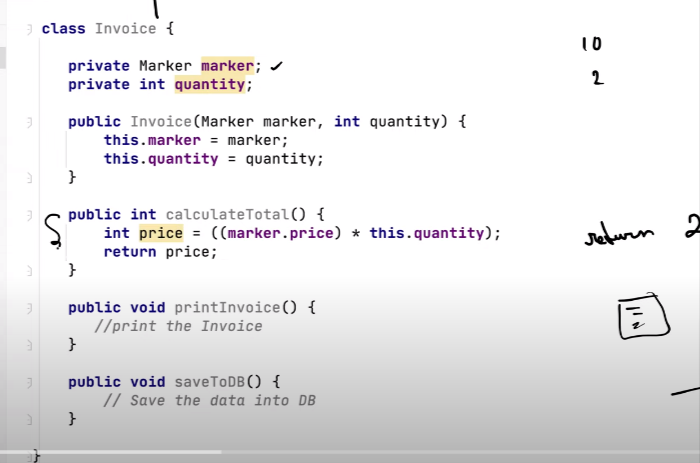
1.Solid principle of Opps



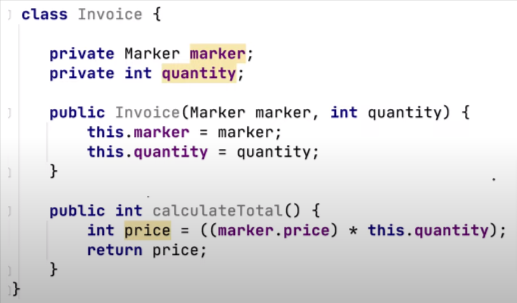
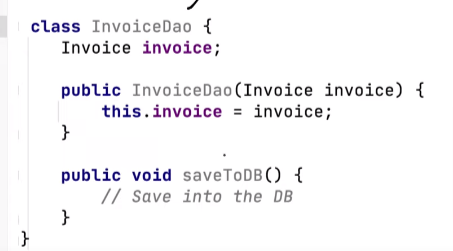
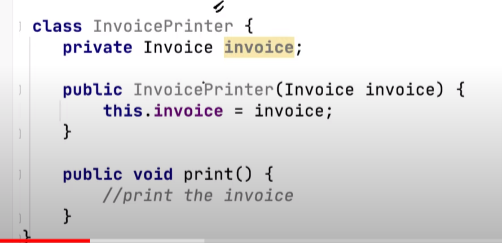


1.Single Resposibility principle

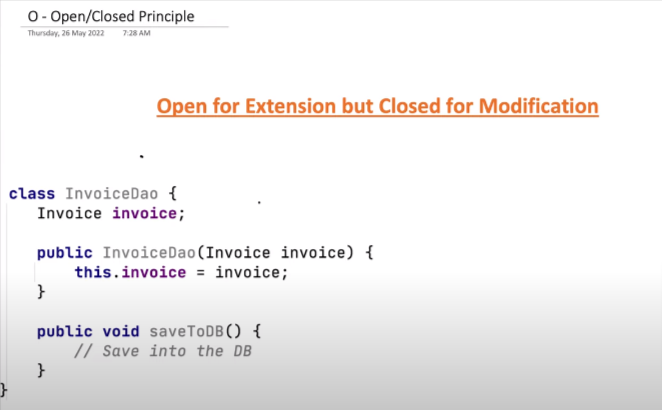




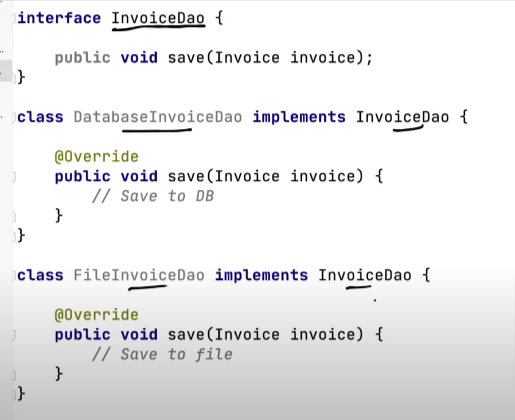
For Single responsibility principle to follow :

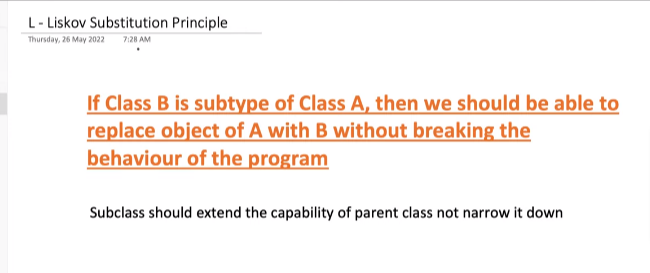
1.2.Open/Close principle

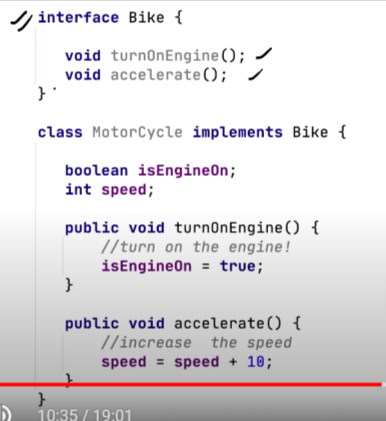
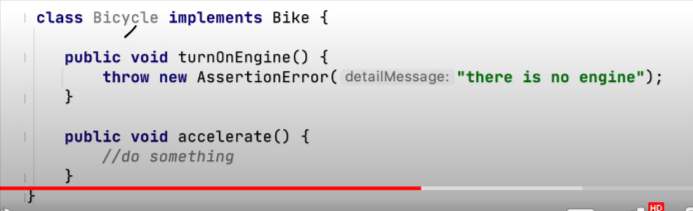


Means that : suppose we have a class InvoiceDoa initally it save invoice to database but we want to have method which will going to save into the file , in that case we will not going to create another method in invoiceDoa , what we do is we will going to create a interface using that interface we will going to implement our different method.

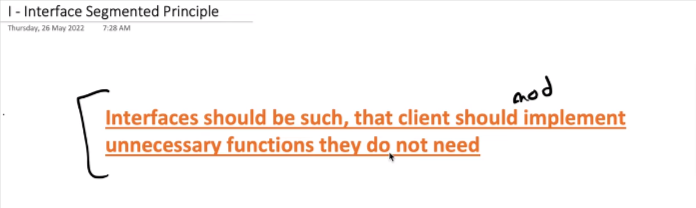


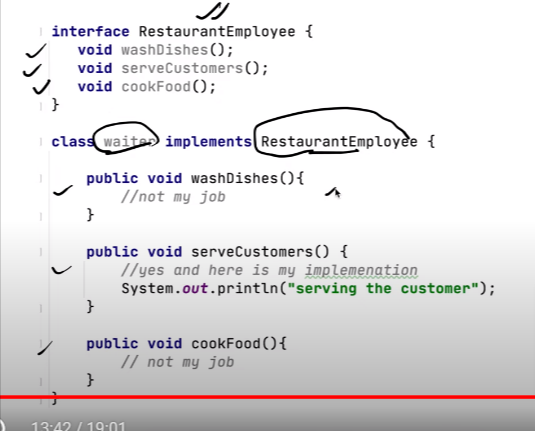
1.3 Liscove Substitution Principle

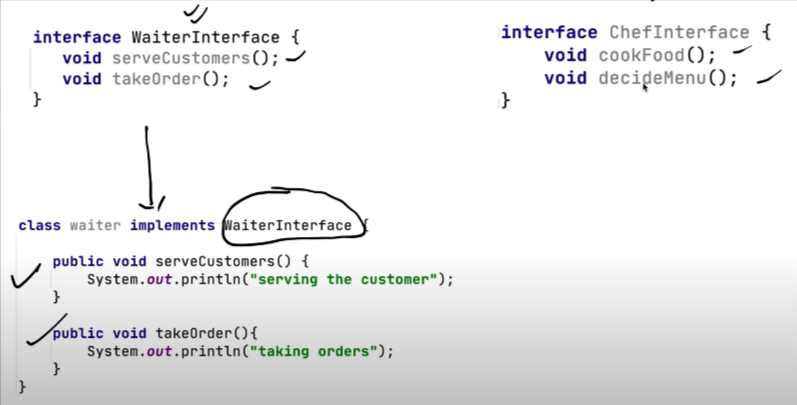


1.4Interface segmented class :

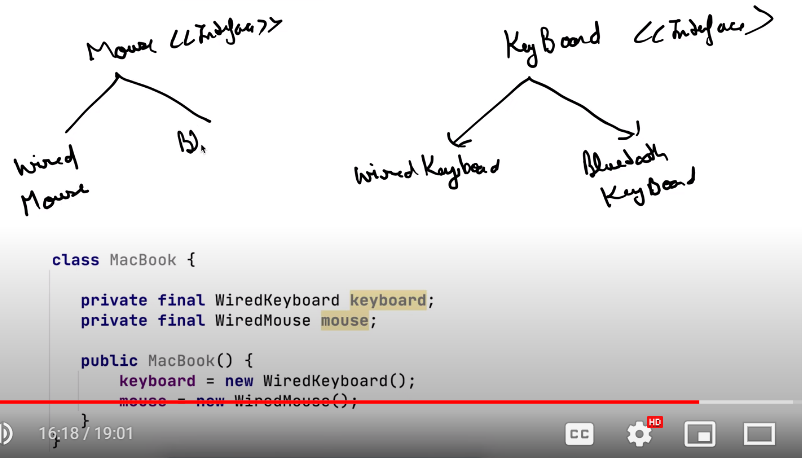


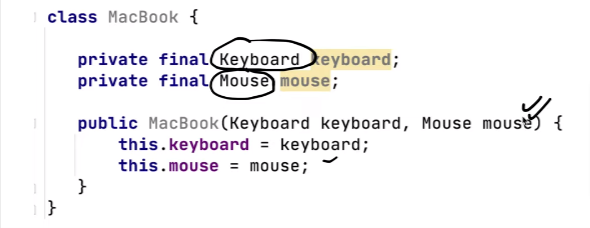




1.5 . Dependency inversion class







2.Design Pattern

Design pattern are the prinicple for the obect oraiented programming , with the help of design pattern we will be able to design scalable,mantainable,reusable software.

1.Strategy design pattern

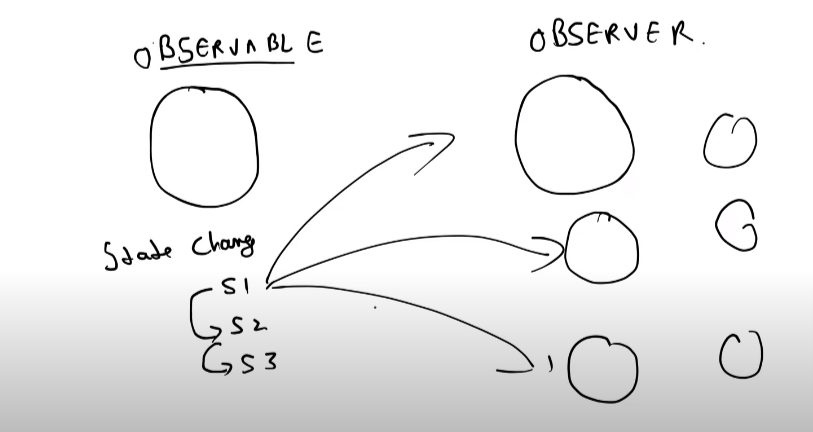
i) Suppose : we have lots of child parent classes , what happen is if child is inheriting mthod from the parent upto that it is good ,but if at the same level childs are using same property in that senario we have code reuseablity issue , to solve that problem is we use strategy desgin pattern .

ii)In stragey design pattern what we do is we create a interface then will going to create all the required concreate classes , and in the base class we will going to pass the interface by creating a constructor we called it as constructor injection.

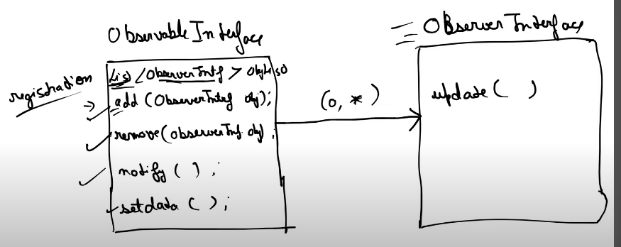
without strategy design pattern :

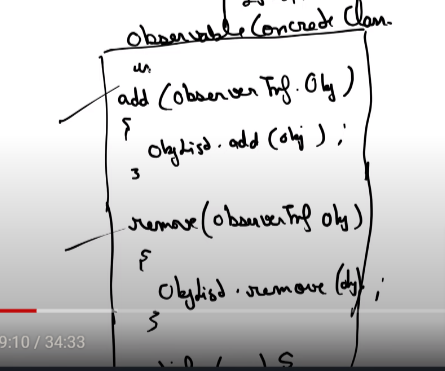
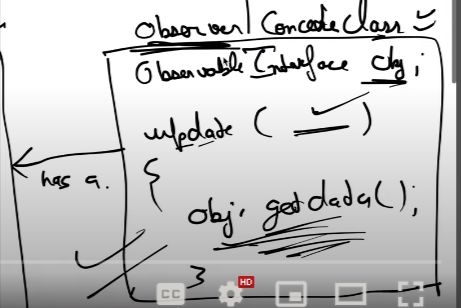
C:\System Design\Strategy\_Design\_Pattern\src\withStrategyPattern

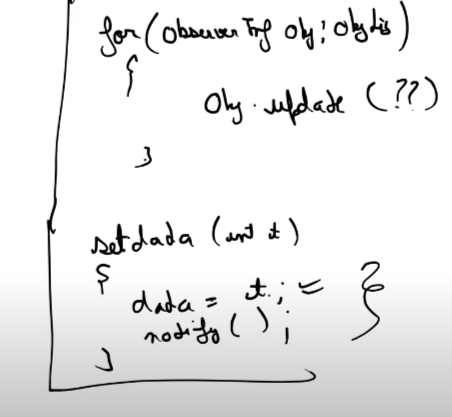
2.Observer Design pattern



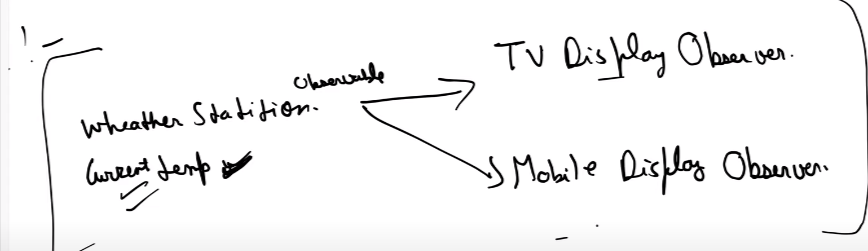
Implementation :





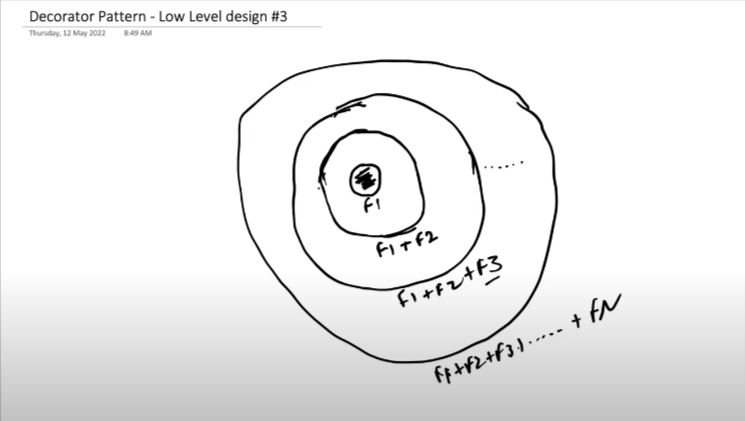
Example of observer design pattern :



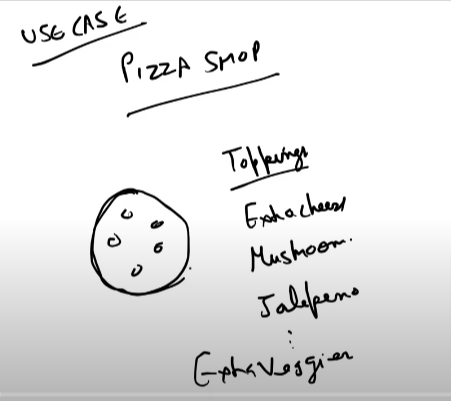
Ex of observal design pattern : design a system to notfiy all the subsuriber that the product is avilable in amozen

Link: C:\System Design\Observer\_Design\_Pattern

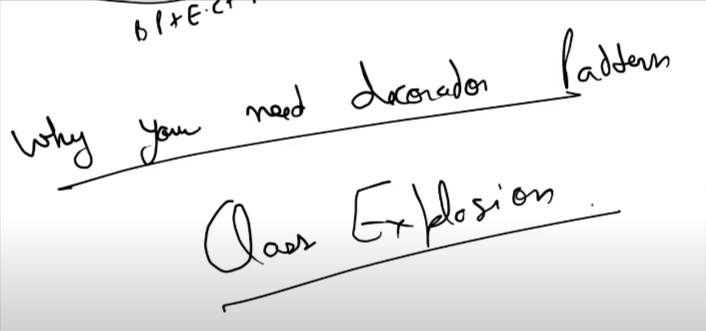
3.Decorator pattern



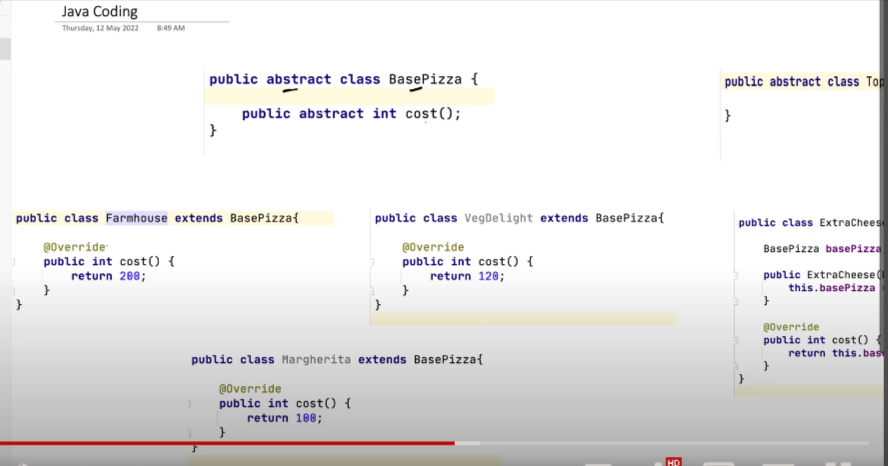
3.2.use case of decorator design pattern :

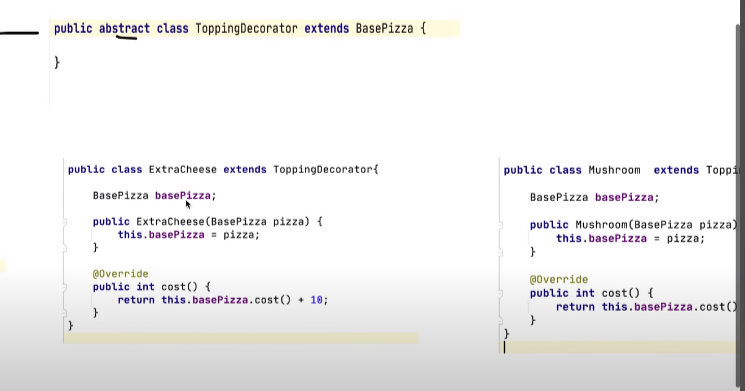


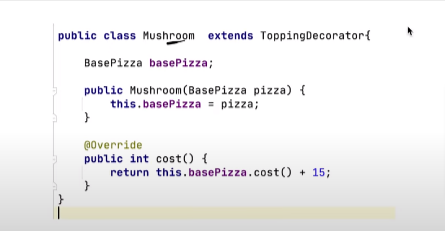
3.3Why do we need decorator design pattern : to avoid class explosion

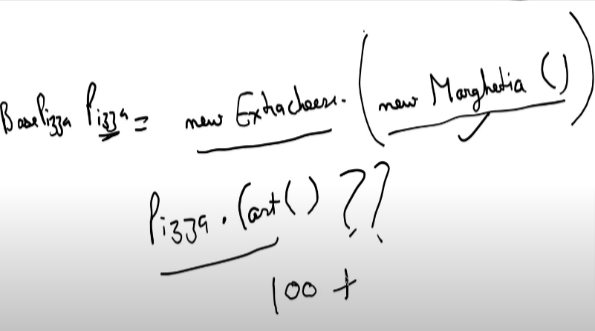
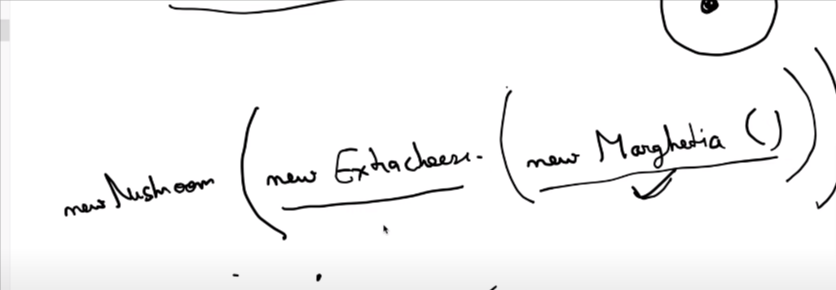


Implementation :



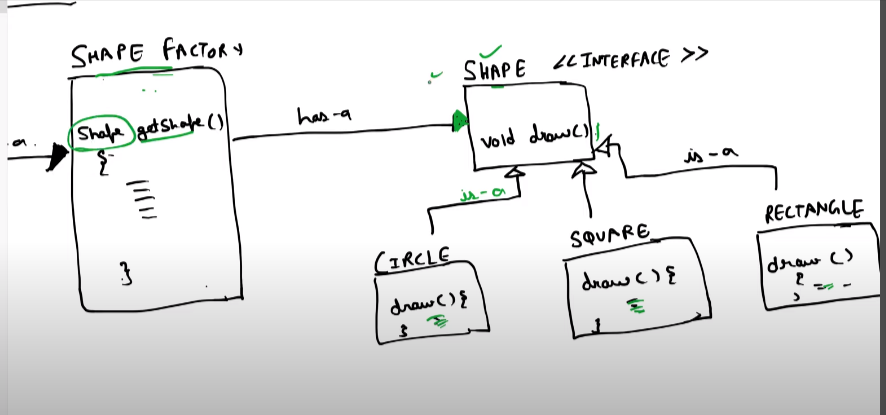
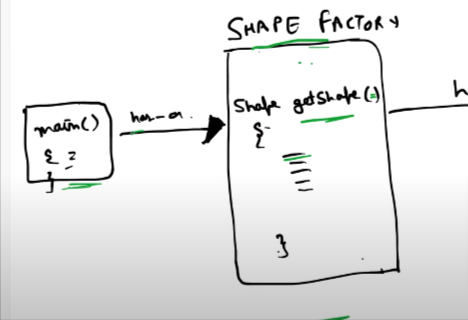


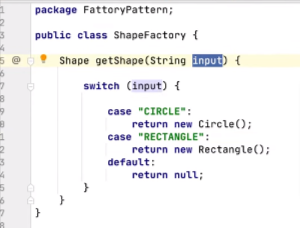
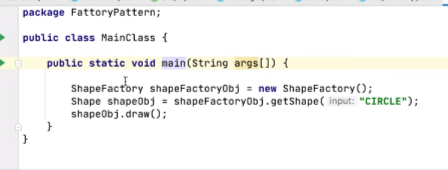


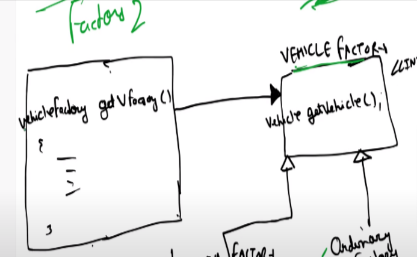
4.Factory vs abstract Factory pattern

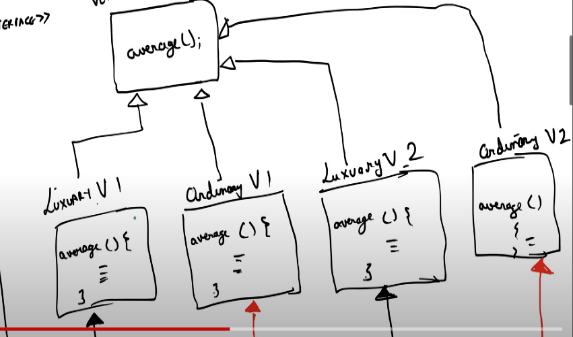
When we have to create object conditionaly then we use factory pattern.

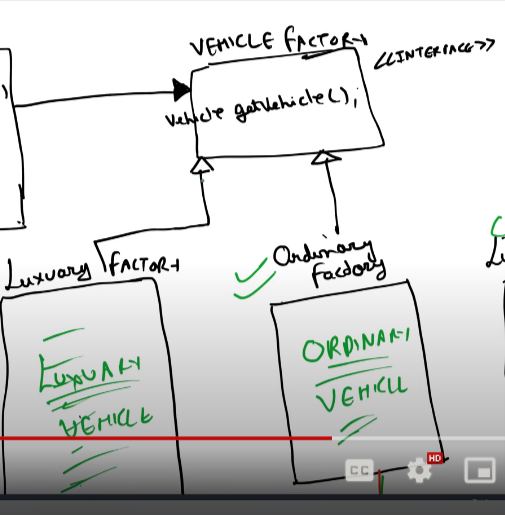


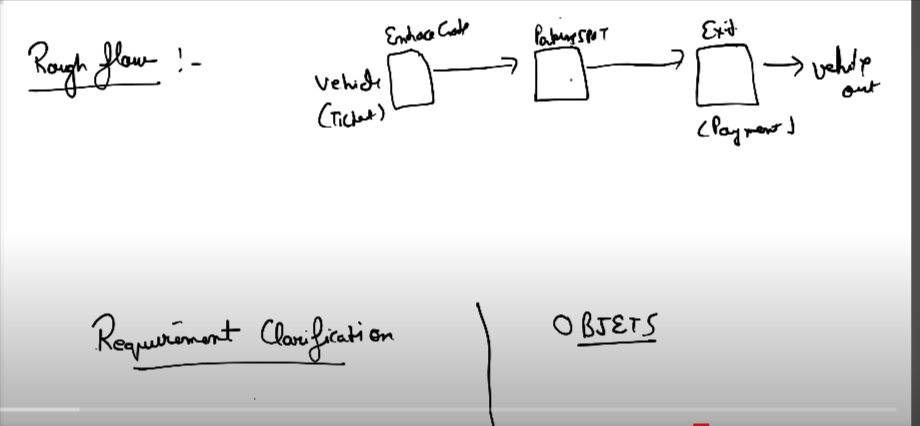
Abstract Factory : when we have to create factory of factory then we will going to use abstract factory desion pattern

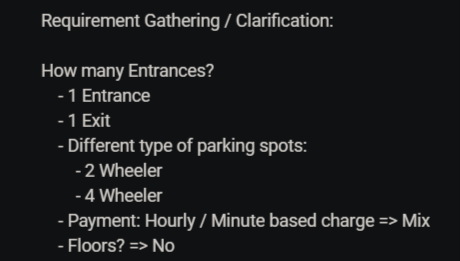
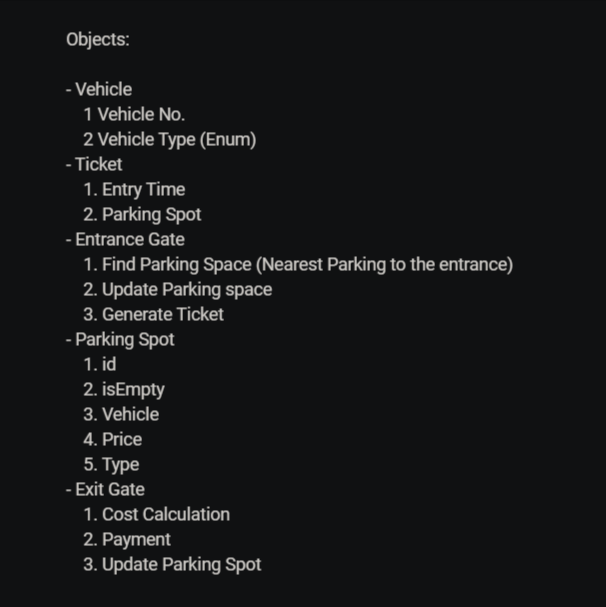


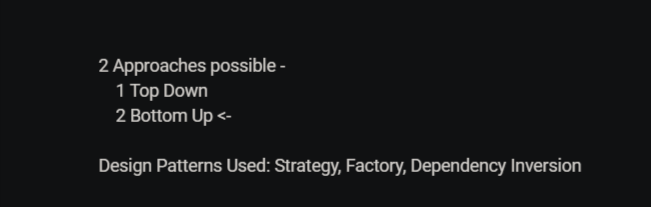




Parking lot low level desing :





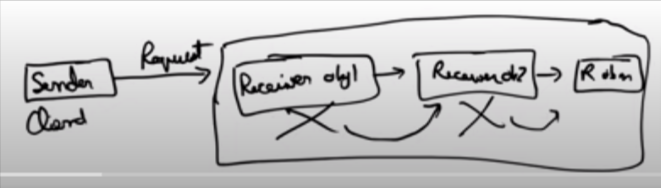




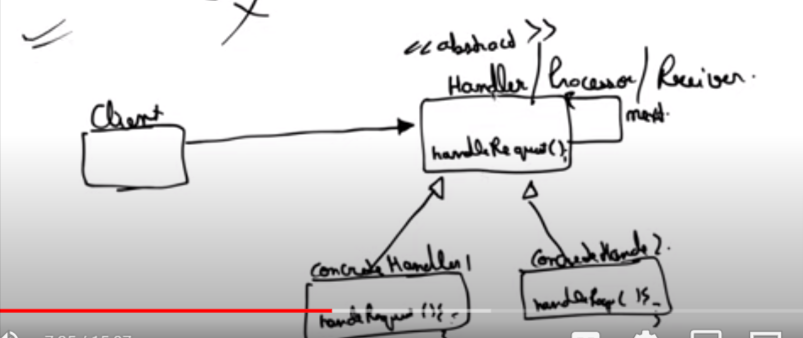


5.Chain of Responsibility Design Pattern :

Application : Atm/ Vendor Machine , Design Loggers

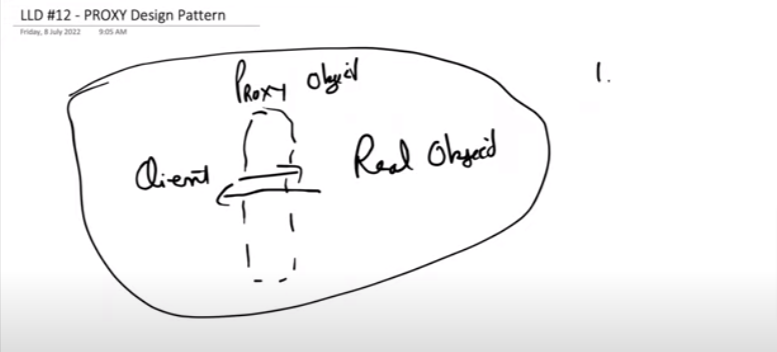


In Chain of Resposiblility design pattern we will going to have Cleint which will going to send a Request and we will going to have a chain of Reciver . so Reciver obj what it will going to do is it will going to take the request for the client and will going to check wheter it will be able to full fill the request or not it not then it will going to send the request to the next reciver object and so on , if it will be able to full fill the request then it will going to it will going to send the response.



Eg :Design a Logger System

6.Proxy Design Pattern



Application : i)user asessing to any resource

ii)caching

7)state Design pattern

Note : this design pattern is used when the operation is happening state wise .

Eg :vending machine