Design Pattern

These are proven solutions to common design problems.

* They are the best Practices / template for coding.
* They tell you how to structure your classes and object.

WHY → 1) Reusability
2) maintainability
3) Scalability
4) loose coupling
5) interview

Without Pattern	With Pattern
- messy	- clean
- tight coupling	- flexible structure

① Creational Design Pattern : How to create the object. (5)

- Factory method
- Abstract factory method
- Builder
- Prototype
- Singleton

② Structural Design Pattern : Combining of classes and object organizing (7)

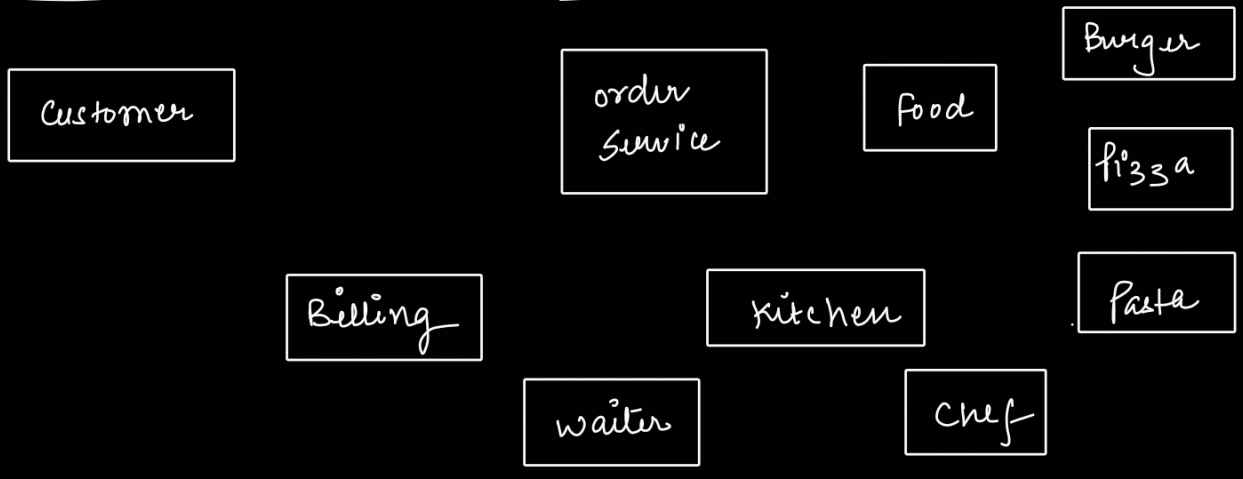
- Adaptor
- Bridge
- composite
- Decorator

- Facade
- Flyweight
- Proxy

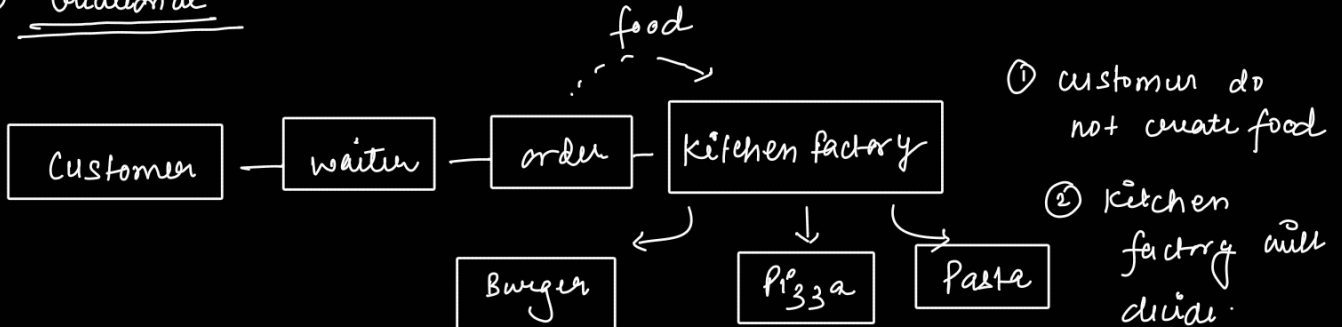
③ Behavioural Design Pattern : Communication between object. Collaboration (11)

- Chain of Responsibility
- command
- iterator
- mediator
- memento
- observer
- state
- Strategy
- template method
- visitor
- interpreter

Example — Restraunt System



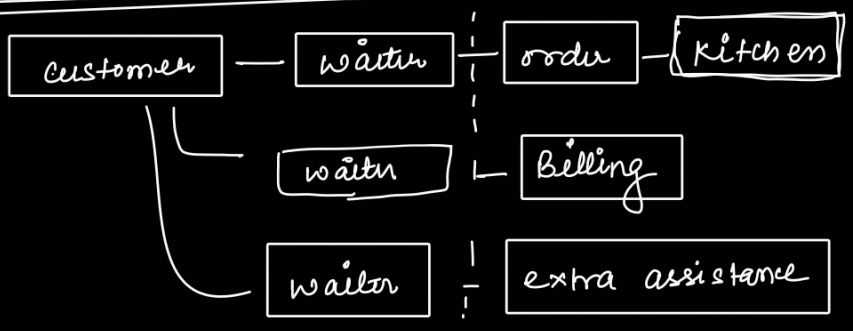
① Creational



① customer do not create food

② Kitchen factory will decide.

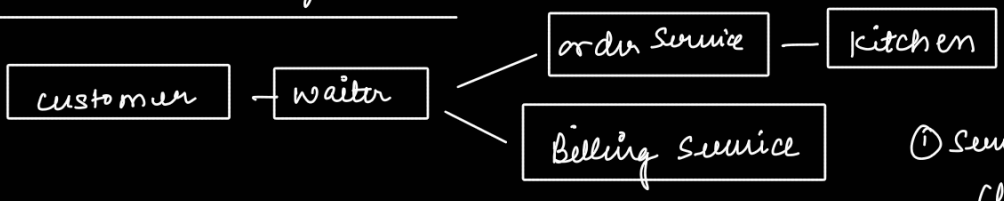
② Structural Design Pattern



① customer talks to waiter only

② hide complexity

③ Behavioural Design Pattern



① Service have clear responsibility

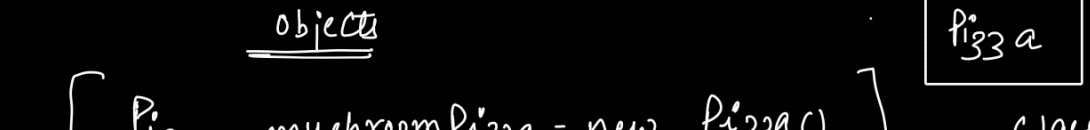
② Easy to add new behaviour.

SOLID

1) S → Single Responsibility - one class - one job
2) O → Open Closed Principle - Extend, dont modify
3) L → Liskov Substitution - child should Replace Parent
4) I → Interface Segregation - small, specific interfaces
5) D → Dependency Inversion - Depends on abstraction

Creational Design Pattern → Singleton Design Pattern

objects



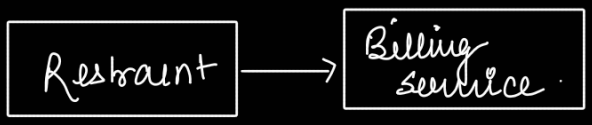
```
Pizza  mushroomPizza = new Pizza()

Pizza  chickenPizza = new Pizza()
```

→ we are making multiple objects in our code.

Restraunt example

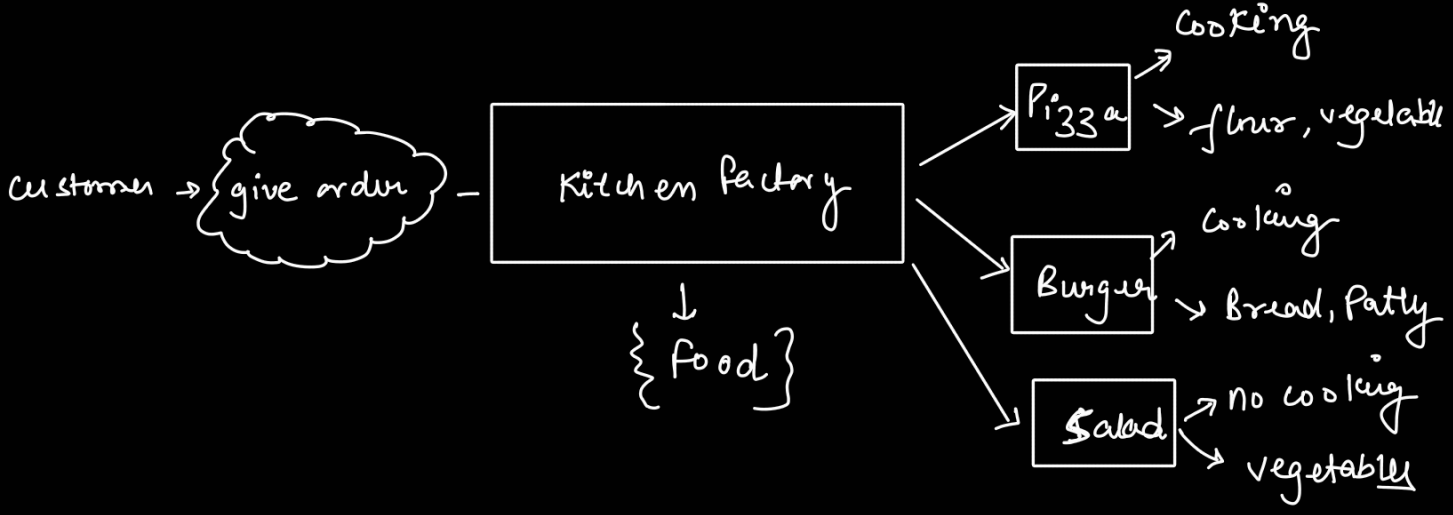
① Billing service that tell total sum at the end of the day.



Real life example

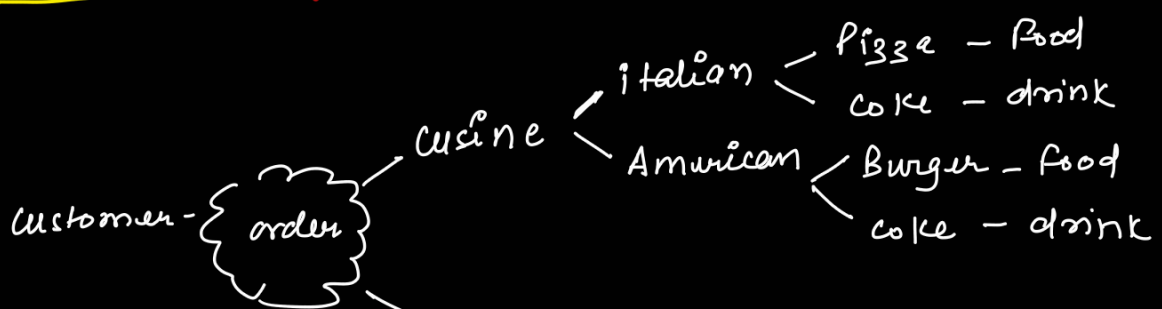
① DB connection one in whole project

Creational Design Pattern → Factory Design Pattern



→ So Basis on the Parameter we will decide which object we have to create.

Creational Design Pattern - Abstract factory Design Pattern

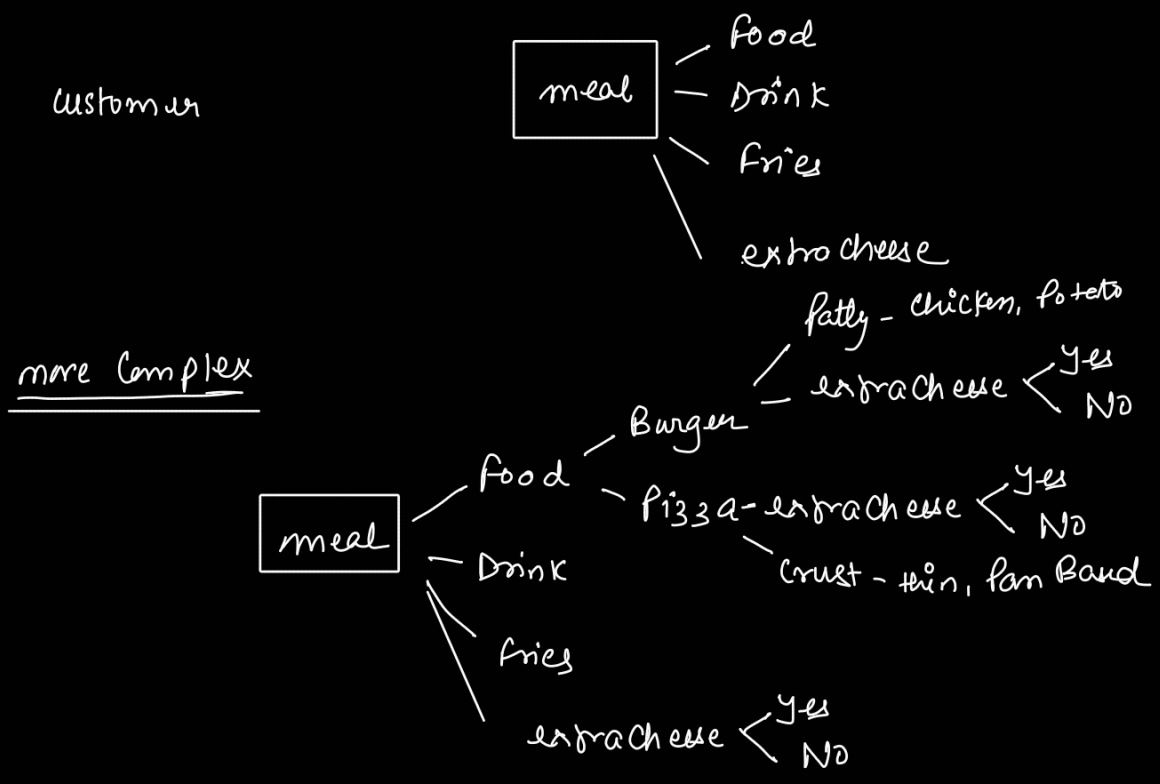


Simple - Pizza - Food
Burger - Food
Coke - Drink

Creational Design Pattern – Builder Design Pattern

Pattern that is helpful in creating complex objects

× Step by step ×



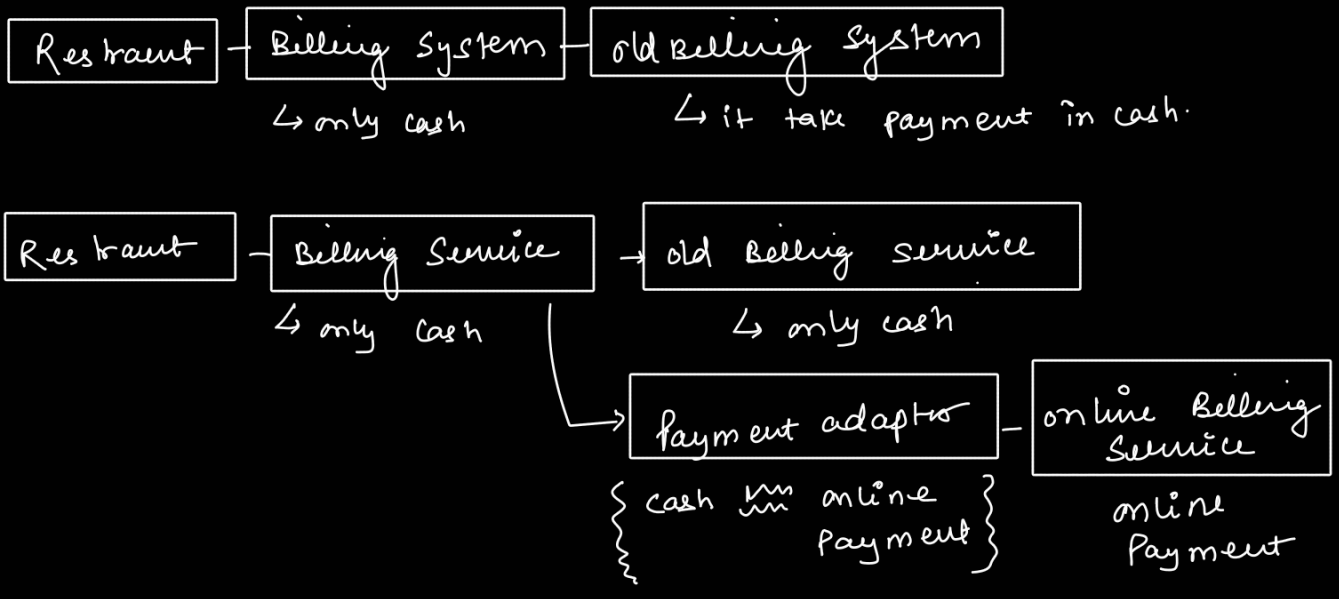
Creational Design Pattern - Prototype Design Pattern

Please repeat my order,

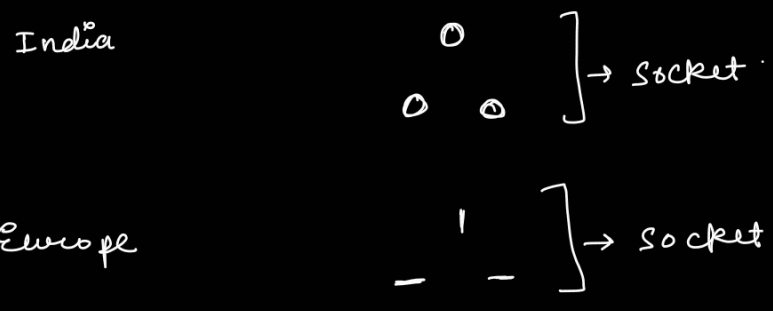
correct way to do is, just copy past [meal 1].

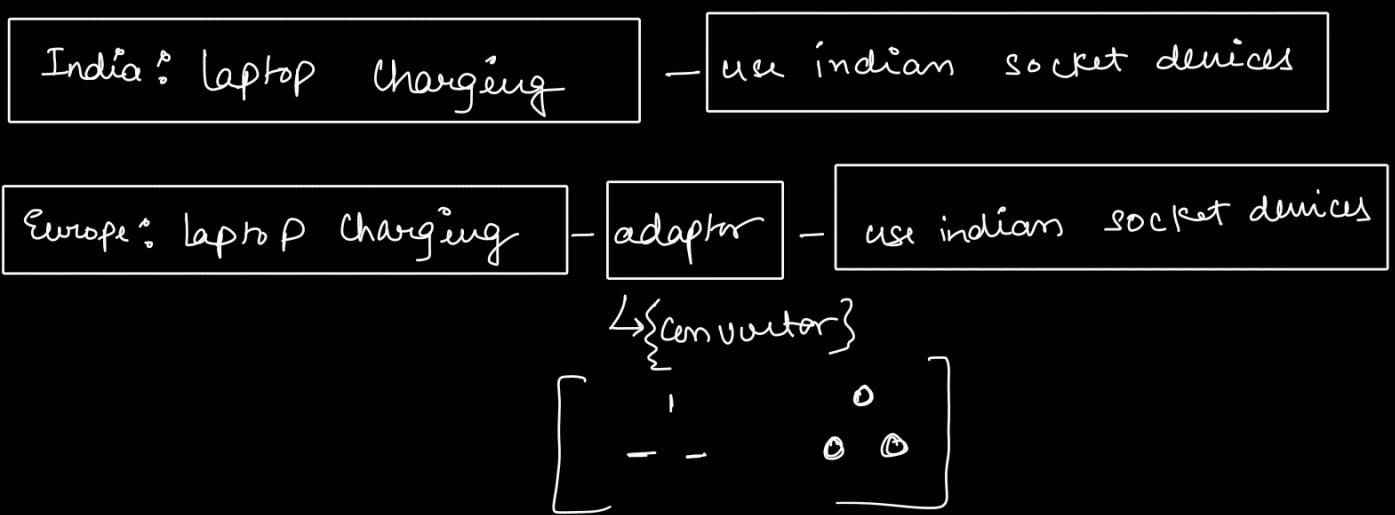
again making meals - completely - X

Structural Design Pattern - Adaptor Design Pattern

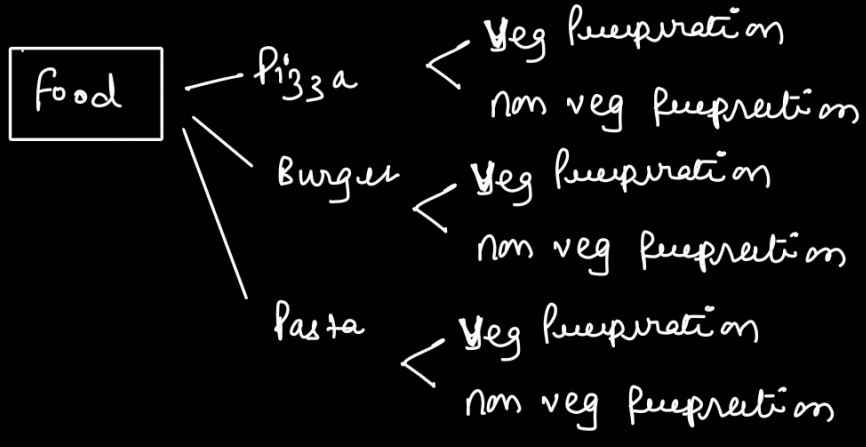


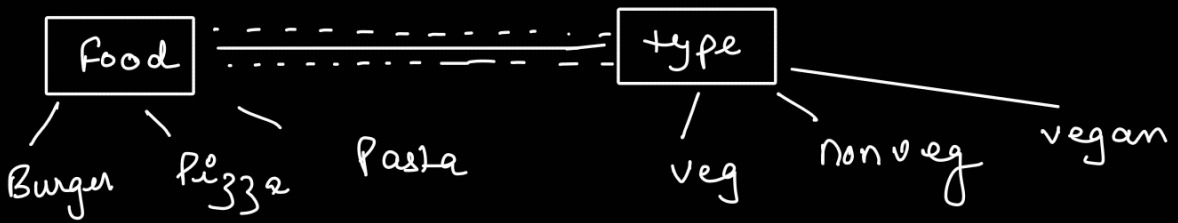
example 2





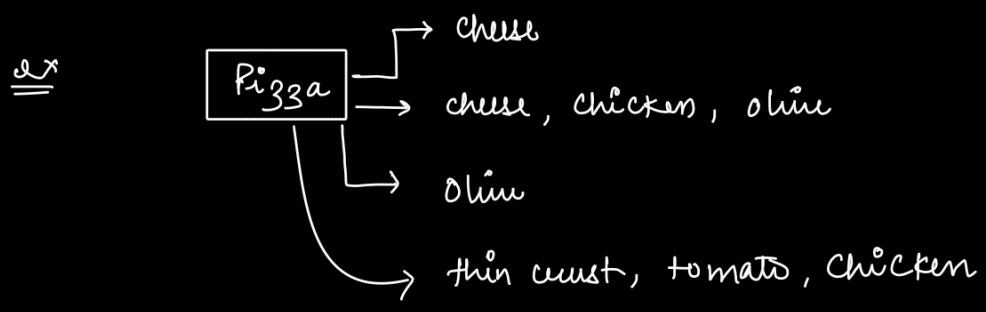
Structural Design Pattern - Bridge Design Pattern





Structural Design Pattern - Decorator Design Pattern

Add new behaviour to an object without dynamically without modifing its class.



Pizza = 200
cheese = 30
chicken = 40
olive = 30

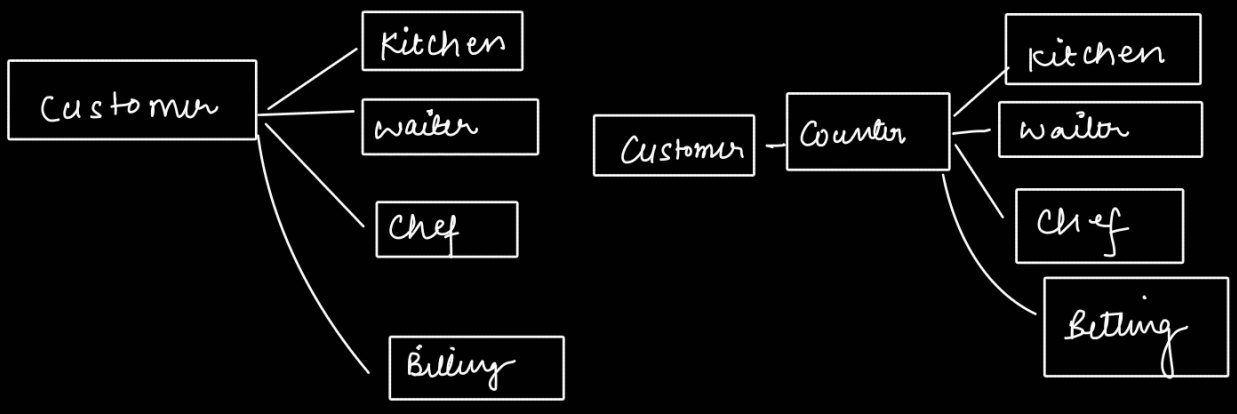
So,

Pizza + cheese = 230

Pizza + cheese + chicken + olive = 300

Pizza + olive = 230

Structural Design Pattern - Facade Design Pattern



Fascade Pattern Provide Simple interface to a complex system.

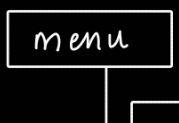
Structural Design Pattern - Proxy Design Pattern

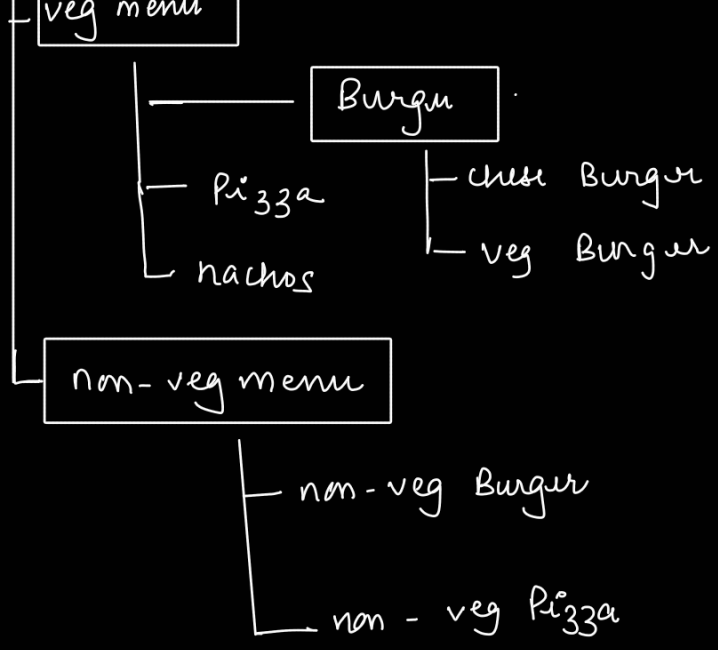
Proxy control access to multiple real services.

Fascade Pattern Provide Simple interface to a complex system.

Structural Design Pattern : Composite Design Pattern

Hierarchy





Structural Design Pattern – Flyweight Design Pattern

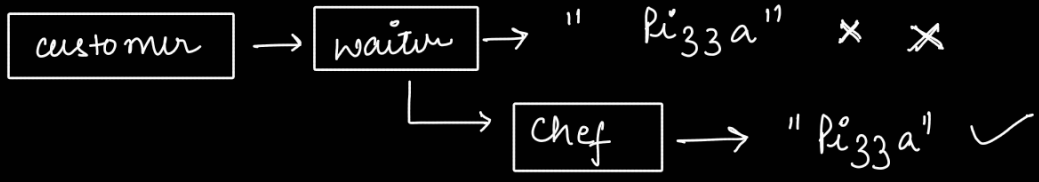
	extrencic state	intrinsic state
Customer 1	Table – 4 : Size – 7 :	Pizza("thin crust", "tomato")
Customer 2	Table – 2 : Size – 10 :	Pizza("thin crust", "tomato")
Customer 3	Table – 7 : Size – 12 :	Pizza("thin crust", "tomato")

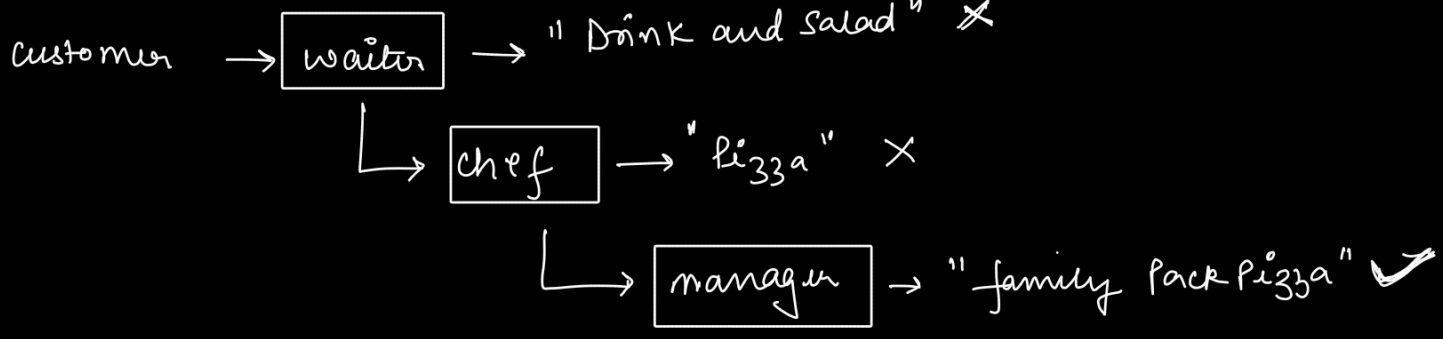
→ common
→ caching

When to use :
→ if we have to so many objects
→ if object creation take so much time

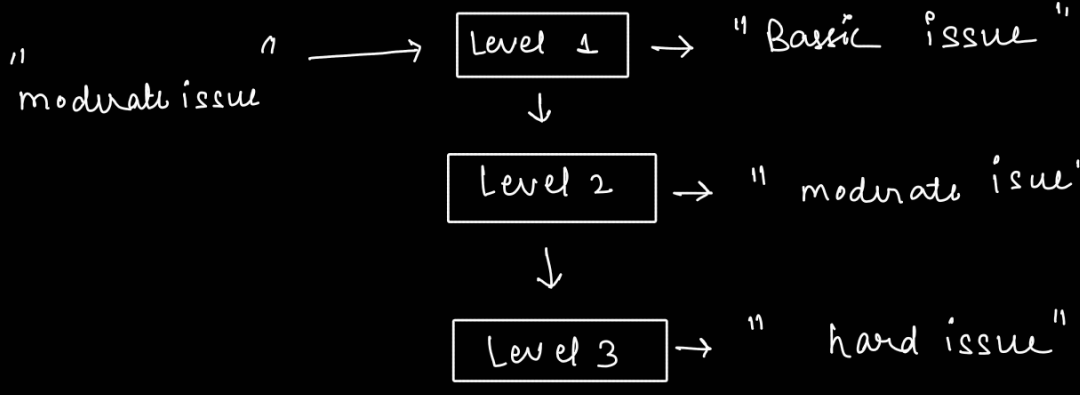
Behavioural Design Pattern – Chain of Responsibility

Customer → waiter → "drink and salad"

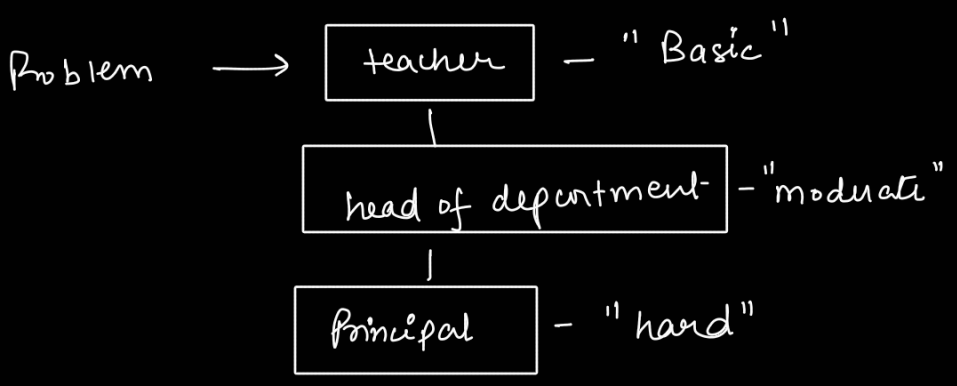




Example



Example: School :

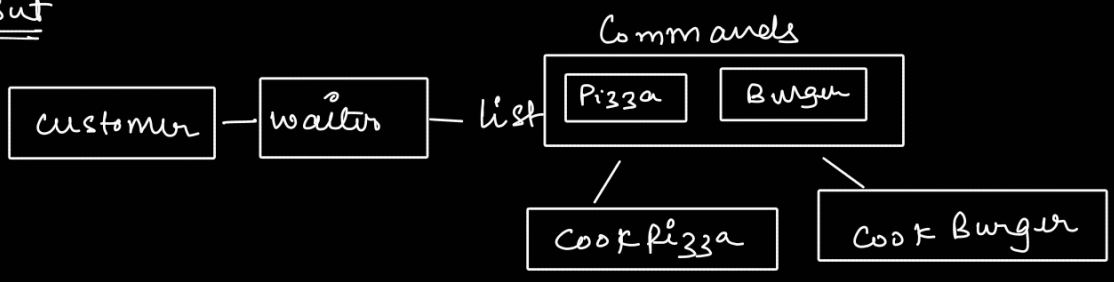


Behavioural Design Pattern - Command Design Pattern

Customer - waiter - Pizza

Customer - waiter - Burger

But



Behavioural Design Pattern - Iterator Design Pattern

Customer want to view menu step by step :

menu items : Pizza, nachos, Burger

Example :
- Pizza
- nachos
- Burger

menu items :- Veg menu
- Pizza
- Burger
  - veggie Burger
  - Cheese Burger

non-veg
- chicken Burger
- chicken Pizza

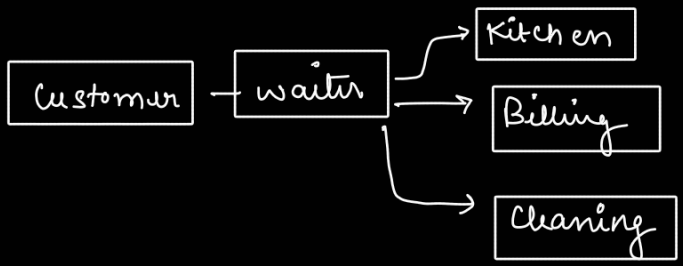
[dfs]

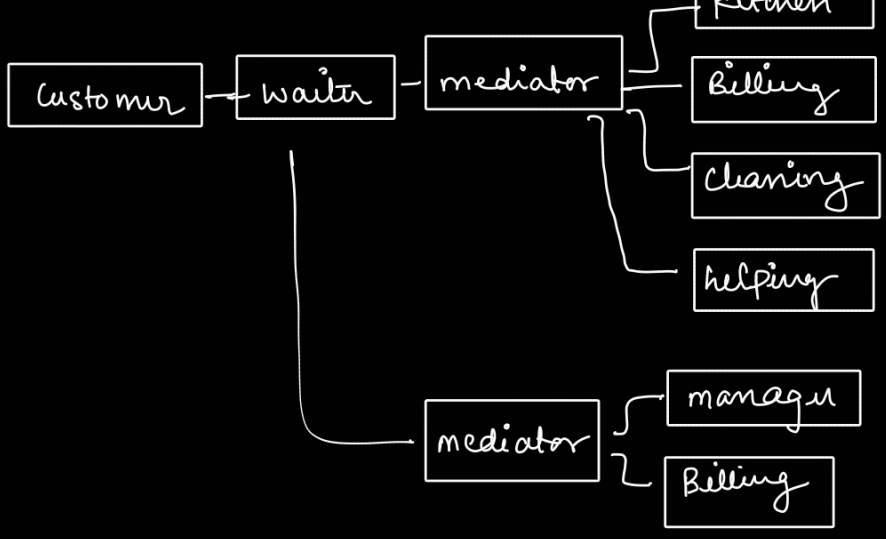
[BFS]

[spm - Cheese Burger not available]

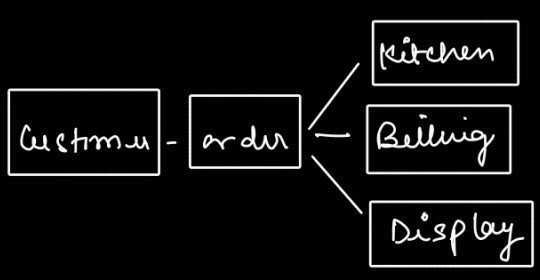
→ customer just need menu

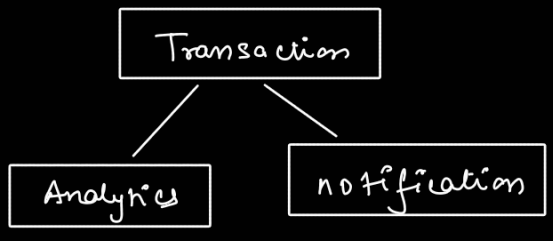
Behavioural Design Pattern → Mediator Design Pattern





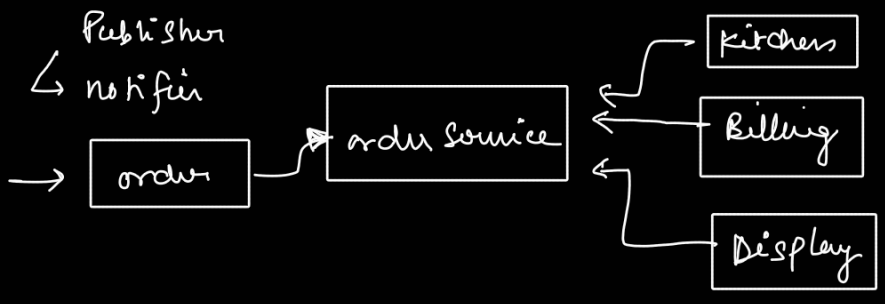
Behavioural Design Pattern → Observer Design Pattern





how to achieve

Subscriber



Behavioural Design Pattern → Interpreter Design Pattern

→ Customer – "Pizza and Burger"

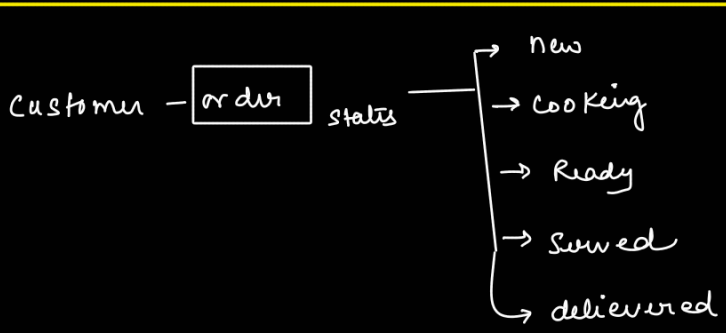
"Burger and Pizza and coke"

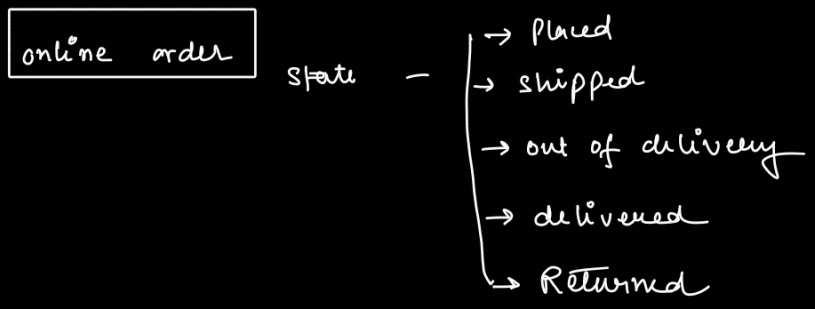
"(Burger or Pizza) and coke"

→ Real life use case → BODMAS = (10+2)*30

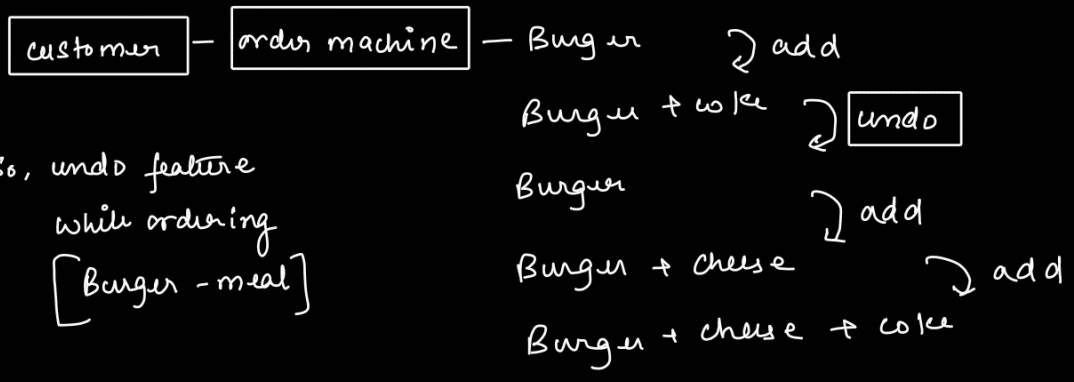
→ Sql → select * from Data where id=" " or id=" "

Behavioural Design Pattern → State Design Pattern



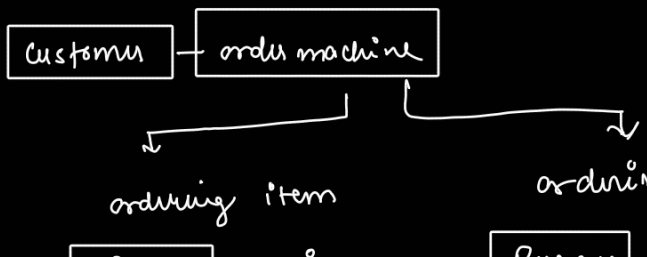


Behavioural Design Pattern → Memento Design Pattern



So, undo feature while ordering [Burger - meal]

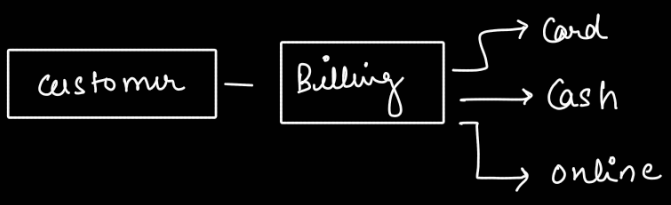
Behavioural Design Pattern → Template Design Pattern



Pizza — cooking
serving item
bill generation

Burger — cooking
serving item
bill generation

Behavioural Design Pattern → Strategy Design Pattern



Behavioural Design Pattern → Visitor Design Pattern