Agenda

Structural

→ Decorator

Behavioural

→ Observer

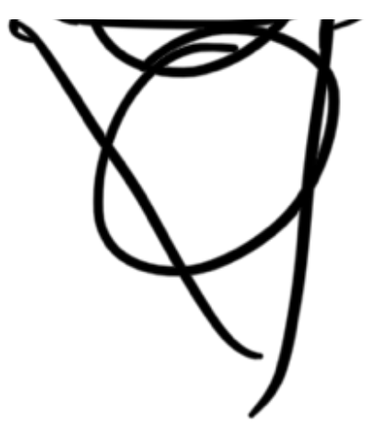
→ Strategy

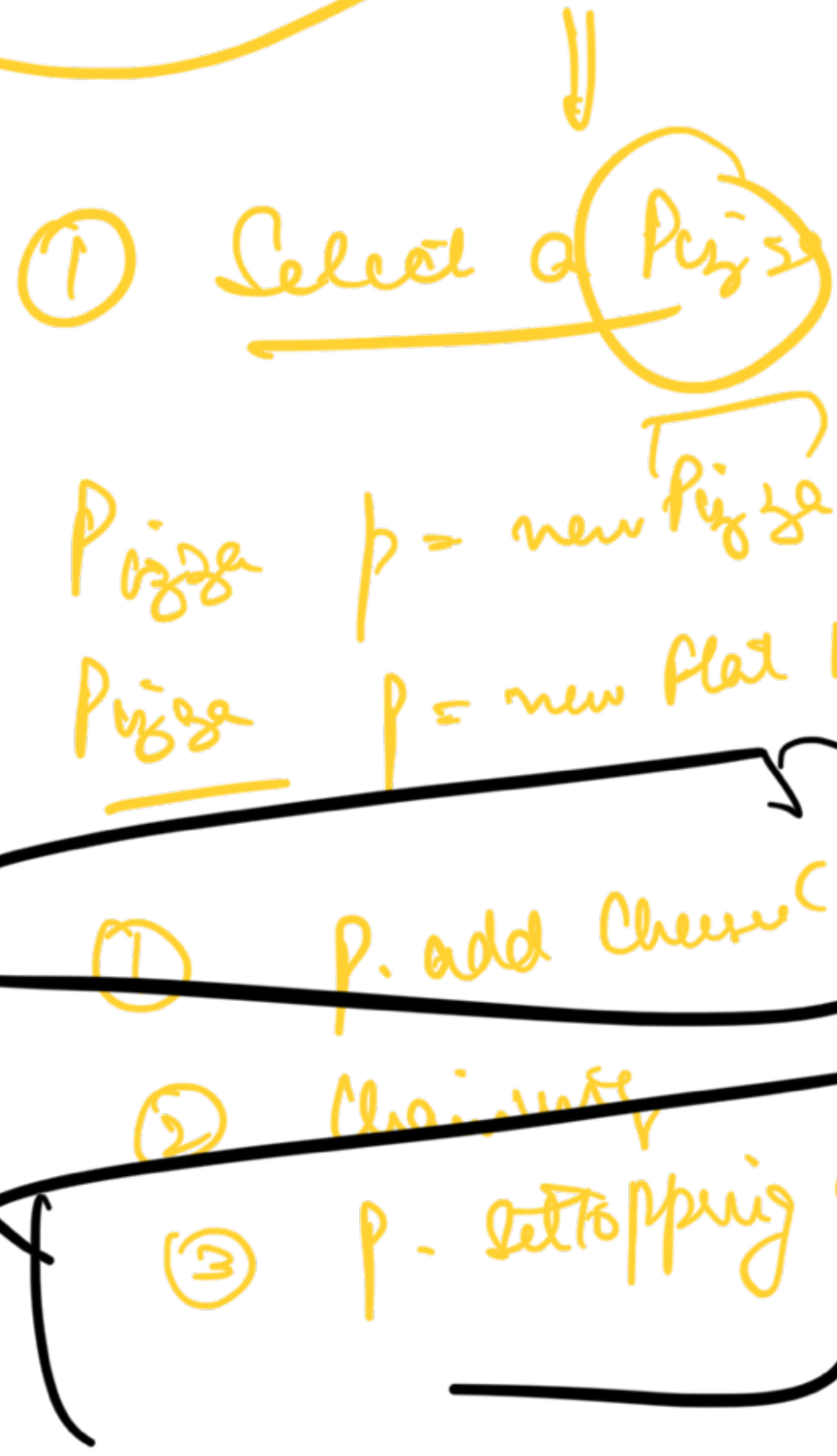
Creational

→ Prototype

Start at 9:10PM

10:48PM




① Select a Pizza
Pizza p = new Pizza
① p.addCheese();
② Choose
③ p.setToppings(

interface Pizza {

  setTopping()
  getBase()
  addBase()

}

Margherita implements Pizza {

interface Toppings {

add

farmhouse

Pizza

List <Base>

Client {

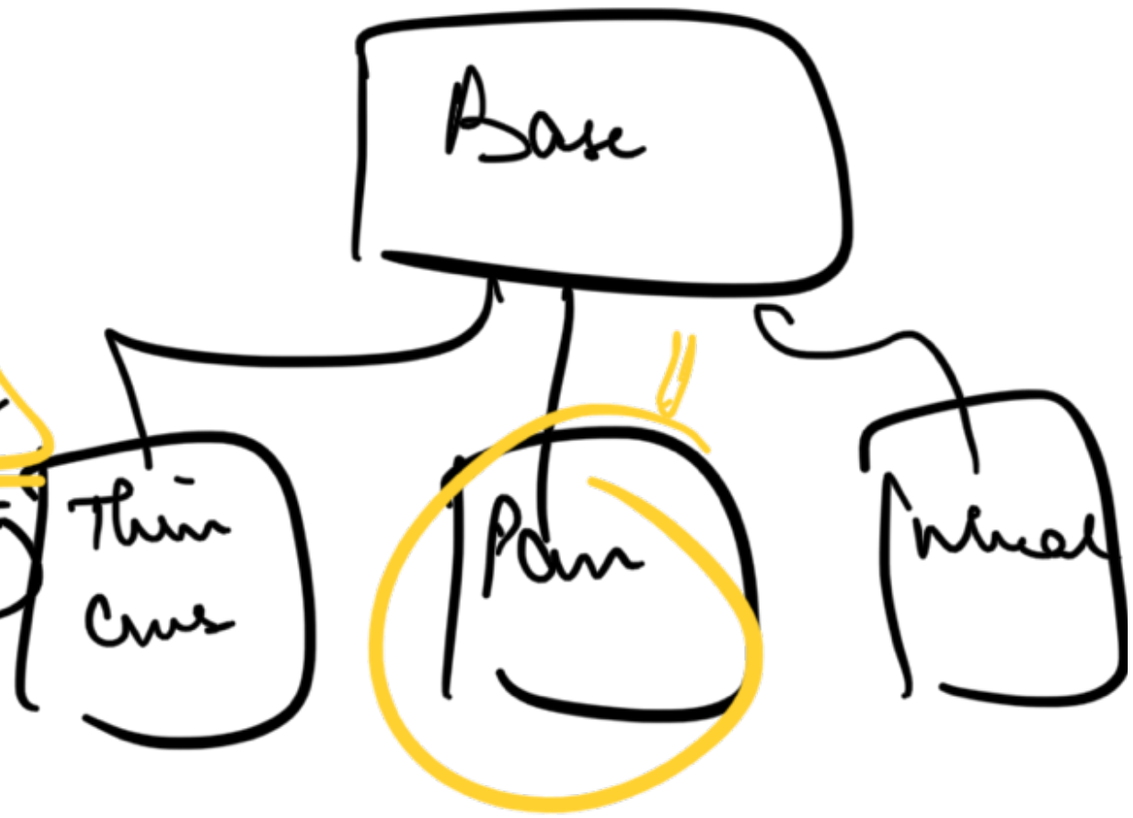
psvm() {

Pizza p = New Pizza

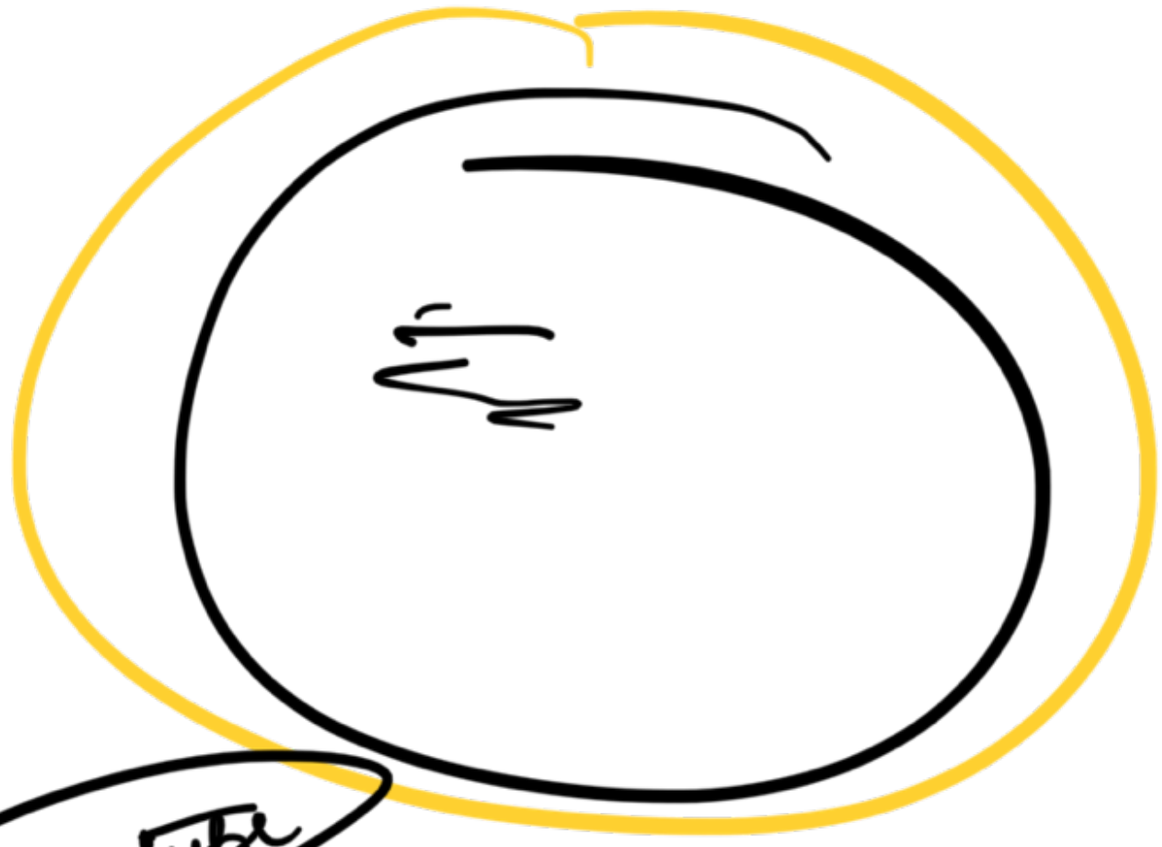
p.SetBase(newItem);

List<Topping> toppings =

p.setToppings(toppings);



}




arg- extc
narg —
formon :
① Create a copy
② add on that copy

Decorator Design Pattern

⇒ Pizza

① define a Central

② define the set of Methods you are about

① cost Calculation ( ) ⇐

② description ( ) ⇐

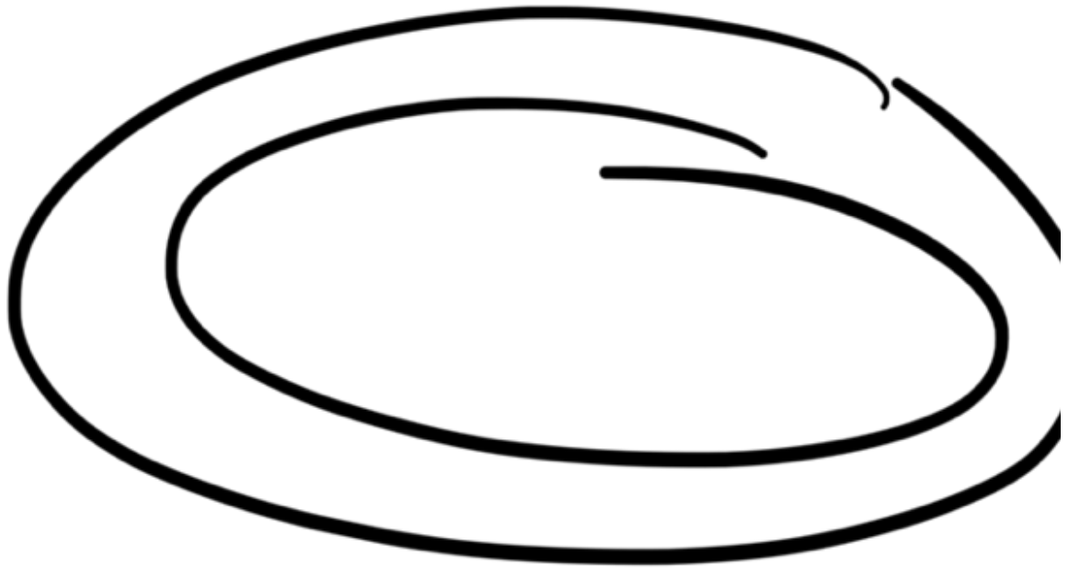
```
interface Pizza {
    int getCost();
    String getDescription();
}
```

③ There are 3 kinds of objects (Entities)

① Base (That can't be added on an existing stuff)

② Addon: (That can only be added)

③ Addon + Base



④ For each type of entity make a class that implements the Base interface

Base →

```
Base() {
```

Addon → Paneer (Pizza ₹) 9

η

Addon → Both type of construction

interface Pizza {

int getCost();

String getDescription()

}

Thin Crust Base

Pan Base

Soft Base

Cheese Burst Base

⇒ Start making a Pizza

class Thin Crust Base implements Pizza {

desc() ⇒ return "Thin Crust Base"

cost() ⇒ return 80;

}

Add ON Entities

Paneer

class Paneer Topping implements Pizza {
Pizza p;
Paneer Topping (Pizza p) {
this.p = p;
}

cost() → p.cost() + 50;

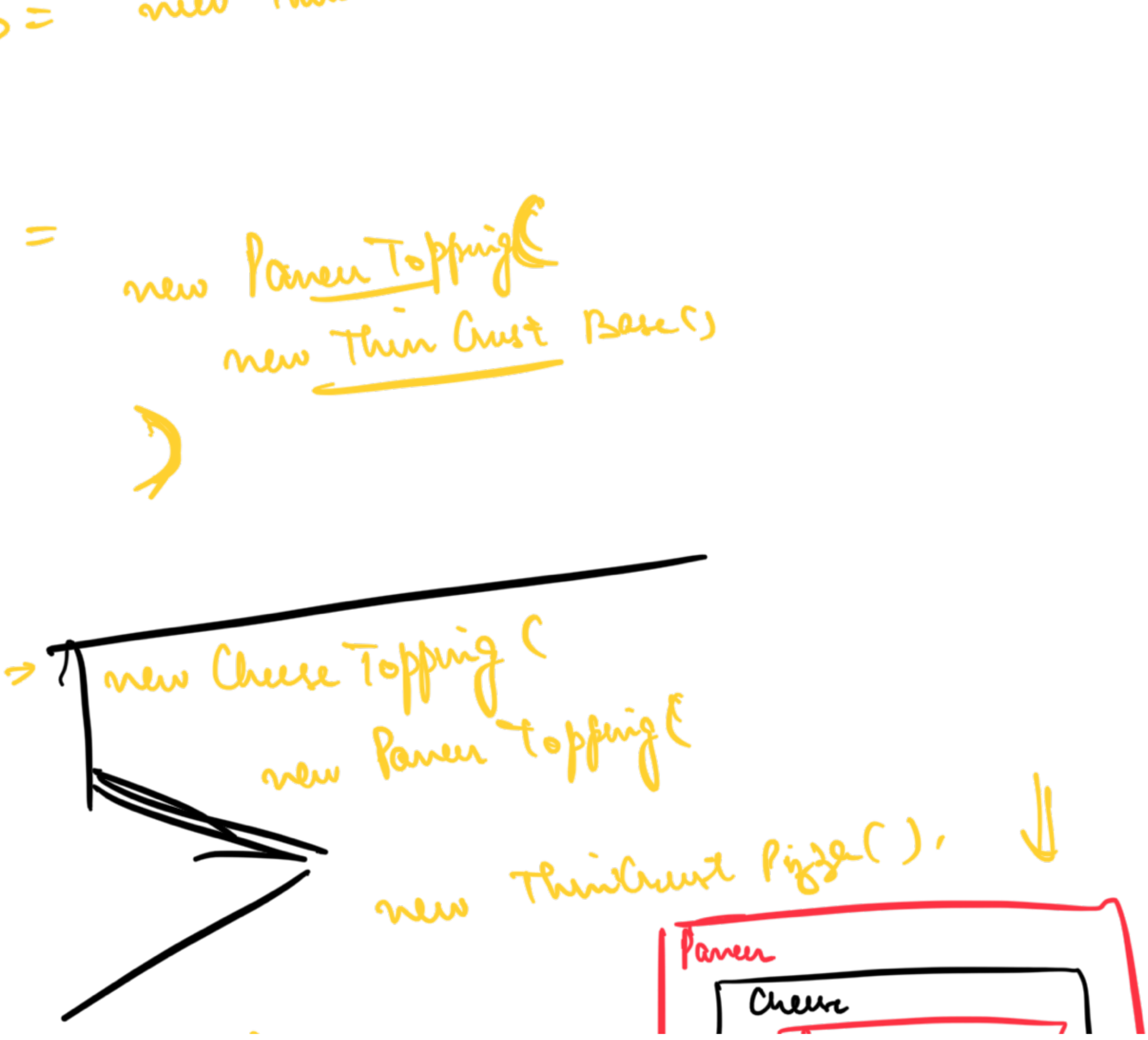
desc() → p.desc() + " Paneer"

}

Pizza p = new Thin Crust Base()

or

= new PaneerTopping(
new Thin Crust Base())



Paneer

Base

Pizza p = new Paneer Topping (
new Cheese Topping (
new Paneer Topping (
new Crust Pizza ( ) ;
)
)
)

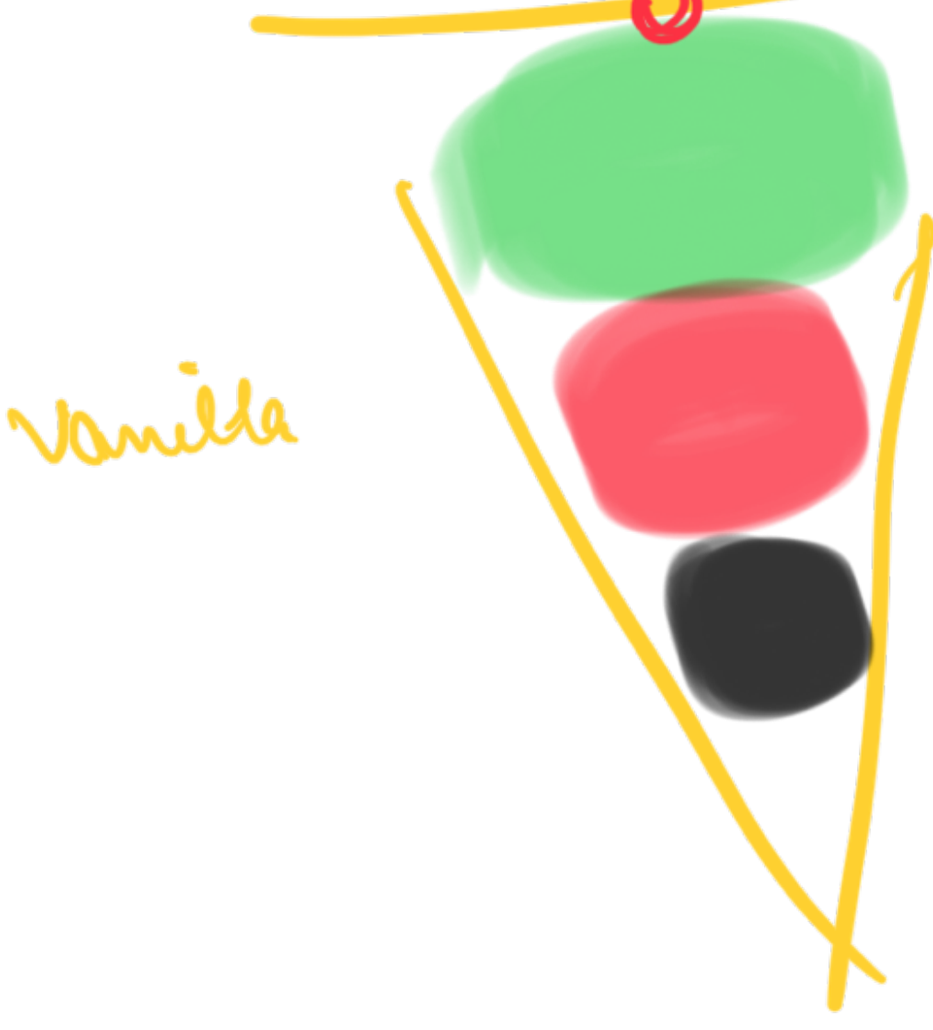
p.add()

p - es se

f. description (

Ice Cream Cone

Emoji



Default interface ⇒ Ice Cream

1 Base ...

2 Base Entities +

3 Addon Entities +

4 Base + Addon +

X

Vanilla Scoop

Choco Scoop

V Cone

C Cone

O Cone

Base: Can make an interface (Pizza / Ice Cream) in isolation

... the interface is base.

Starting pt. of the ...

When we can add to the property of a class.

When we don't add a new functionality

HTML

Margin/ Padding.

decorate an HTML element with
a margin

interface Element

Base ⇒ Button, Checkbox, TextA

Addons ⇒ Padding, Margin, Border

@RestController
@Controller
Controller { =

REST API
Command Line Controller
RestController
Controller

createUser()

registerUser()

login()

}

Add to a behaviour at runtime
without changing existing cls
anything / uses | logic

REST Controller

REST API

Controller

boolean supp RBS ?

⇒ Controller UC = new RestController(

new UserController()

)

// Do REST related method

UC

oc.

⇓

delegate to normal co

⇓

Application a = new Application()

⇑

Application a = new Secure App"(

⇓

new App"();

→ Create an object in a understandable

manne

⇒ add properties / behaviours without modifying code

OBSERVER DP

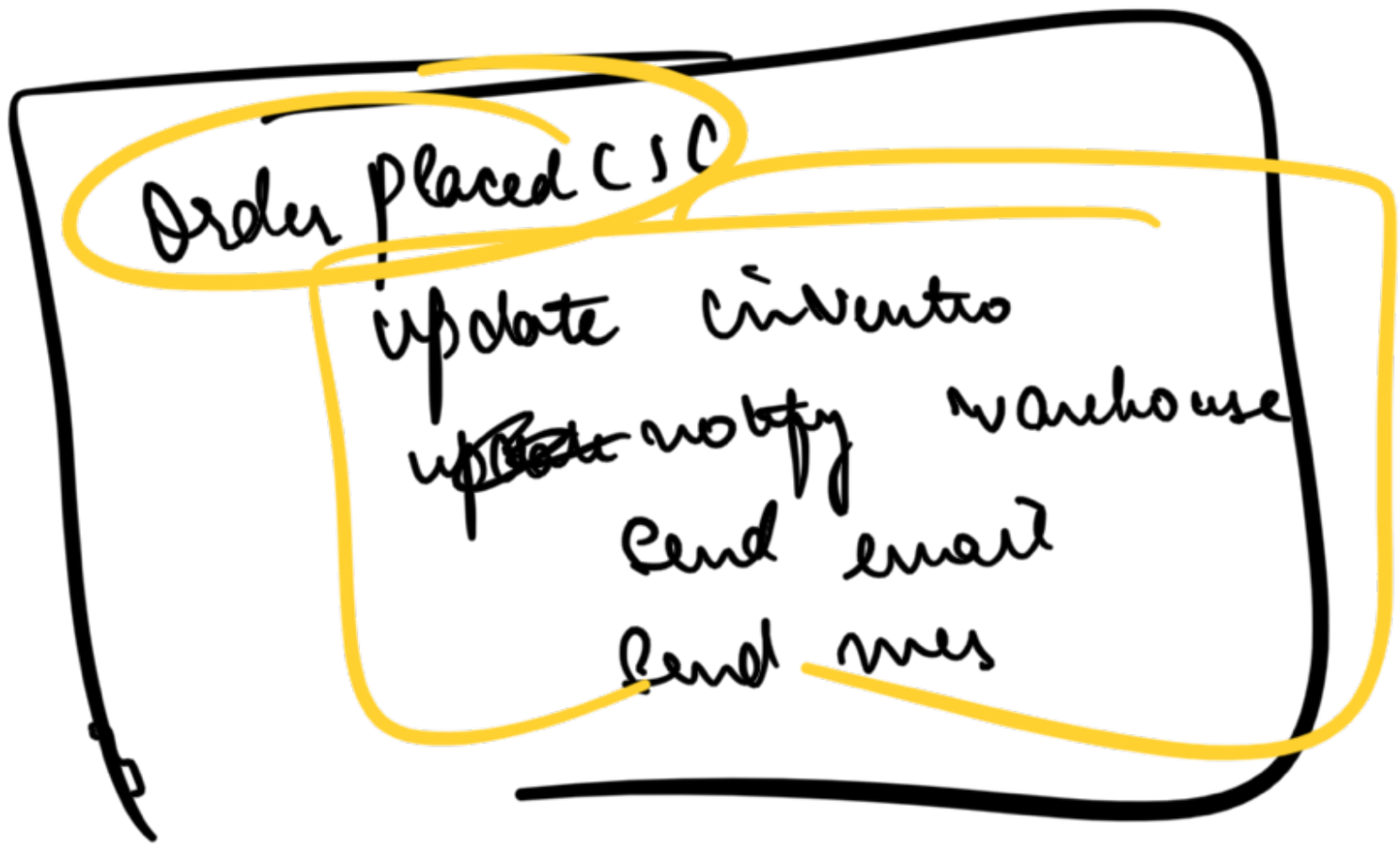
Code

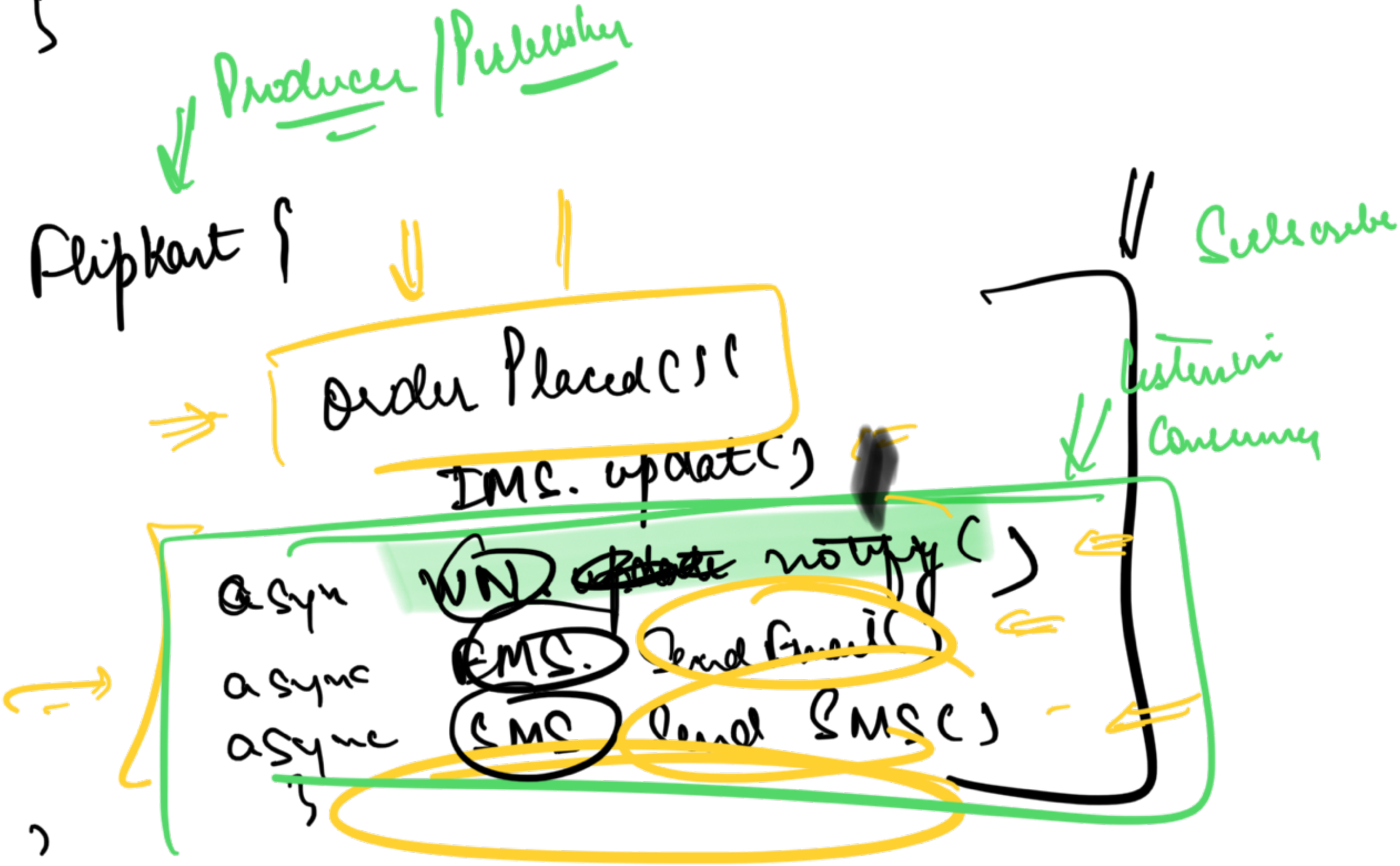
User Sign up Event ()
Send OTP
Send verify email

start RYC process

update our DB

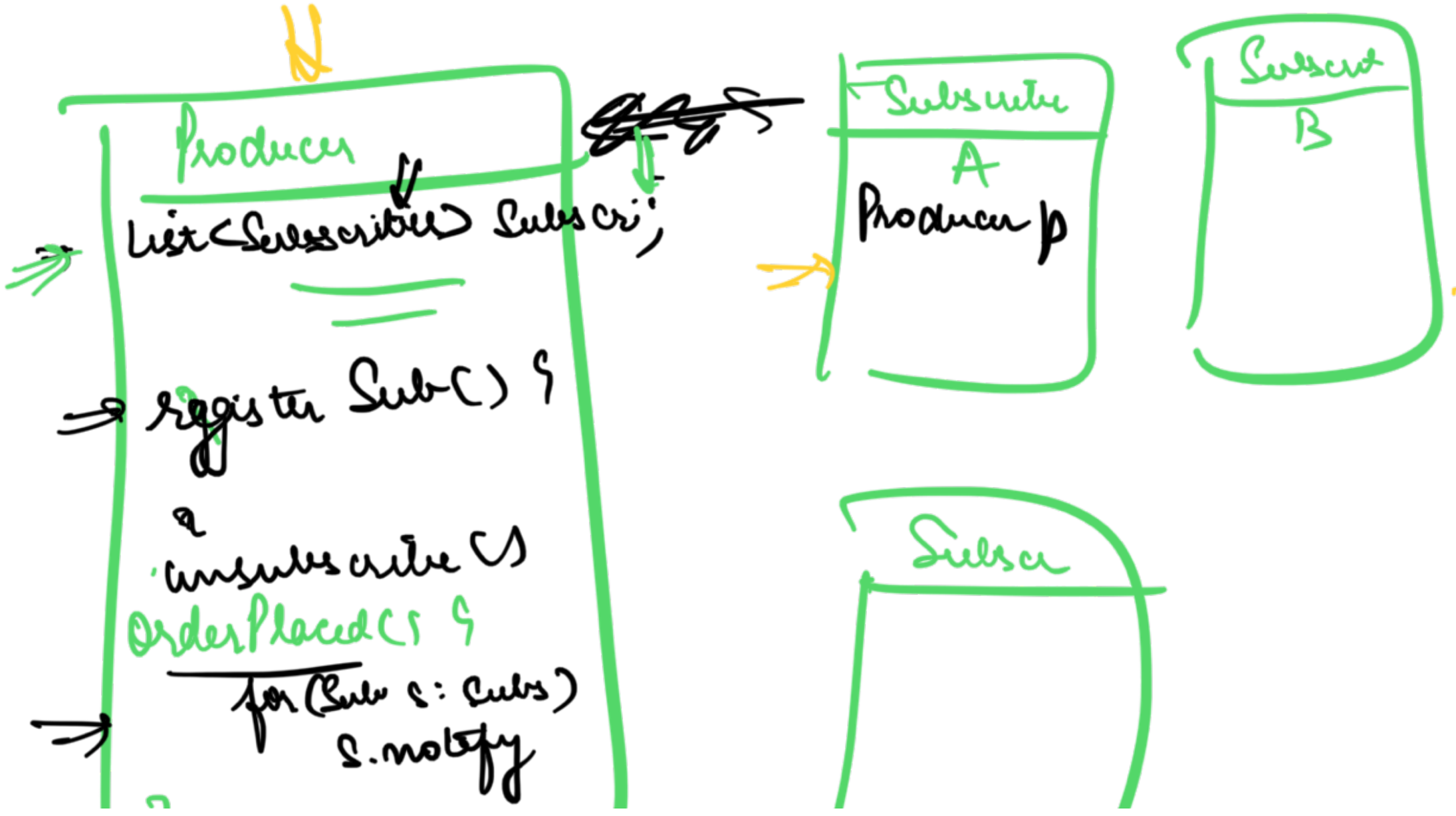
Flipkart ?





Tight coupling b/w the owners and the

Flipkart


Producer
List<Subscriber> Subscri;
registerSub()
unsubscribe()
OrderPlaced()
for(Sub s: subs)
s.notify
Subscriber
A
Producer p
Subscri
B
Subsc

Subscriber

notify() throws Except

Removed Tight Coupling

Observers

<< Subscriber >>

Producer


notify()
EMS
notify()
SMS

producing of
an event

an event