Structural Design Pattern :-

code structure and implementation details.

Agenda

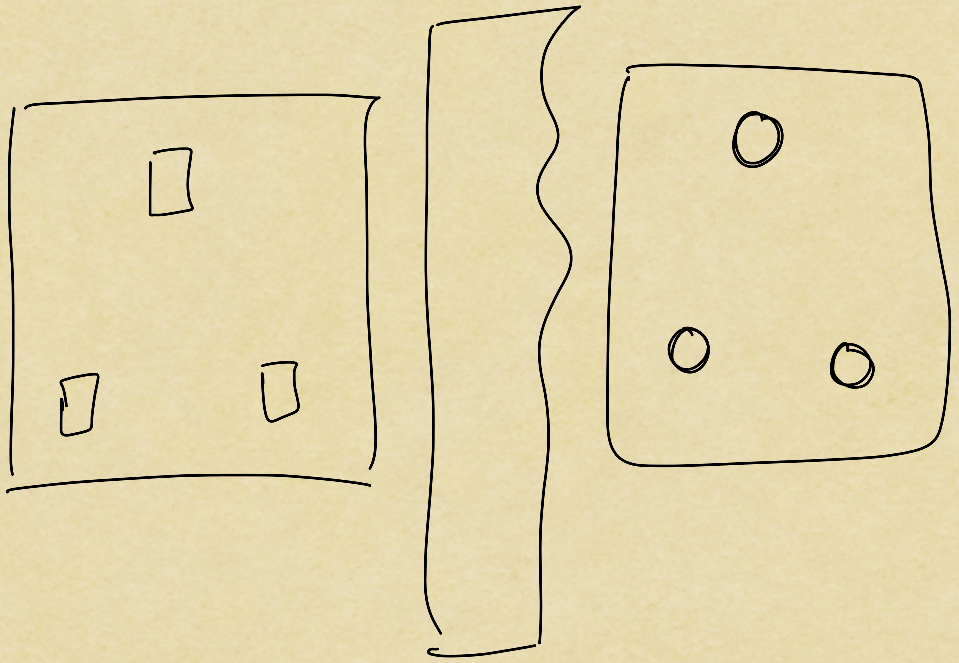
i) Adapter Design Pattern

ii) Facade Design Pattern

i) Adapter Design Pattern :-

Adapter ⇒ bridge / connector b/w two entities

⇒ international power adapter :-



Intermediary layer that connects / transforms one to another

example ⇒ Apple

only provides type C
↓
adapters
- ethernet
- HDMI
- pendrives / USB.
- power cable

assume

Apple says will provide all ports ⇒ type C
- type A
- HDMI
- ethernet
- power cable

→ bulky
→ more code to support the hardware
→ expensive

↓

not a good idea

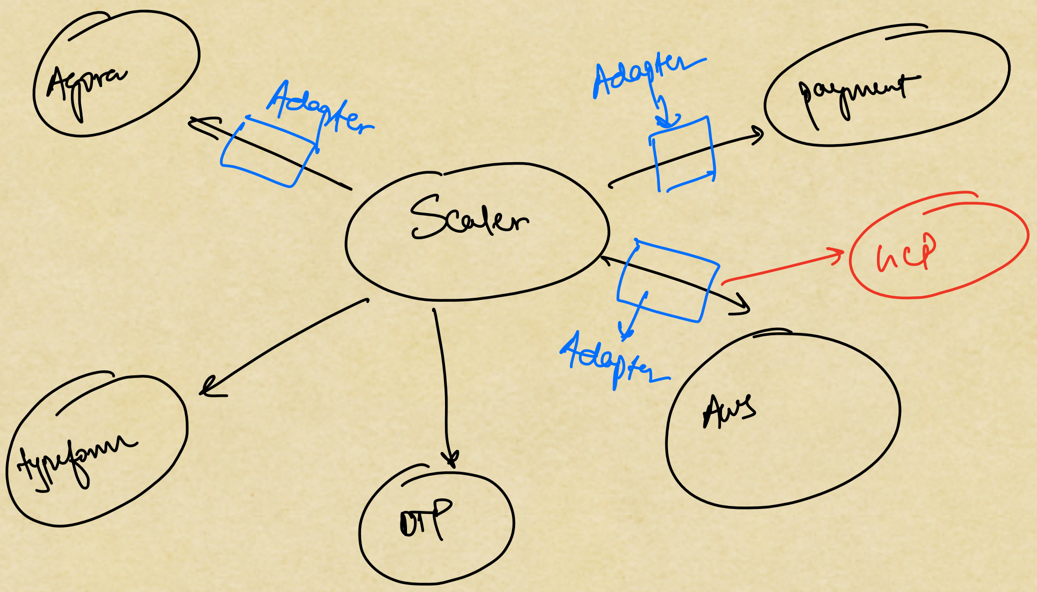
Apple developers

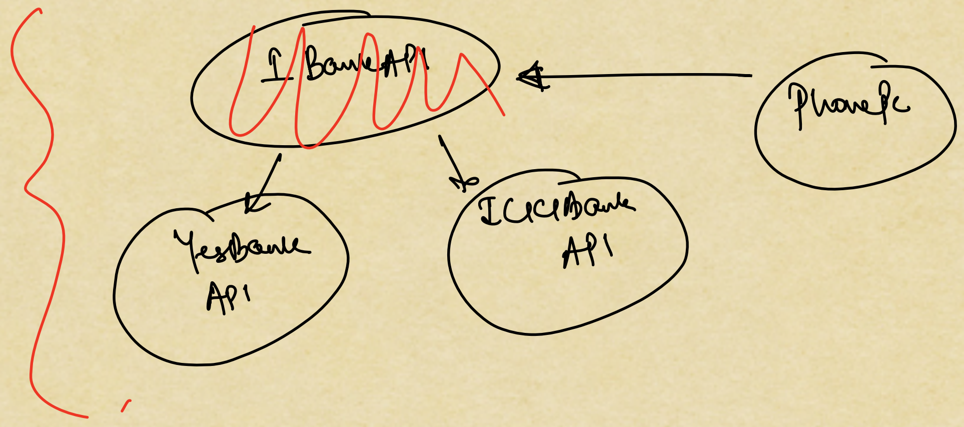
↓

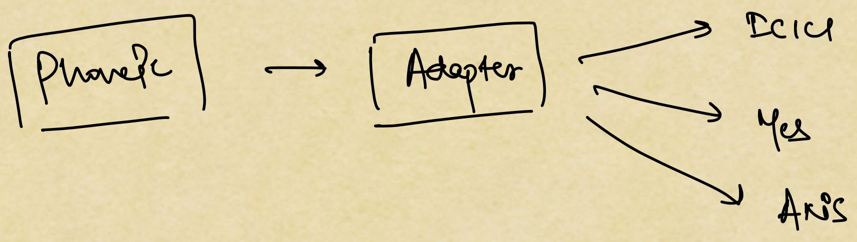
only support typeC and let adapters take care of conversion from (*) to type C.

↓

typeA, HDMI etc





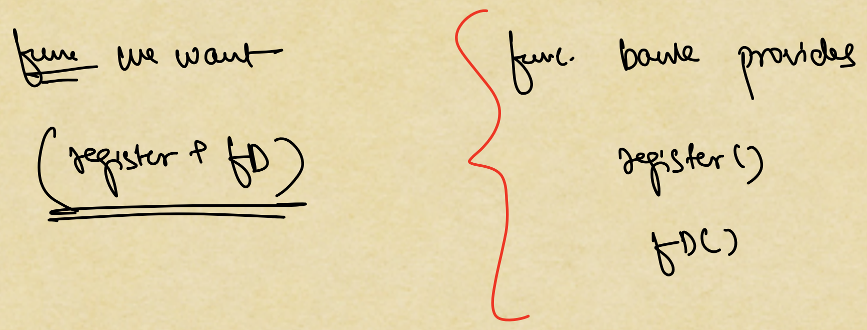


→ Adapter Design Pattern ensures that our codebase remains maintainable when we are switching from any 3rd party dependency / lib / system

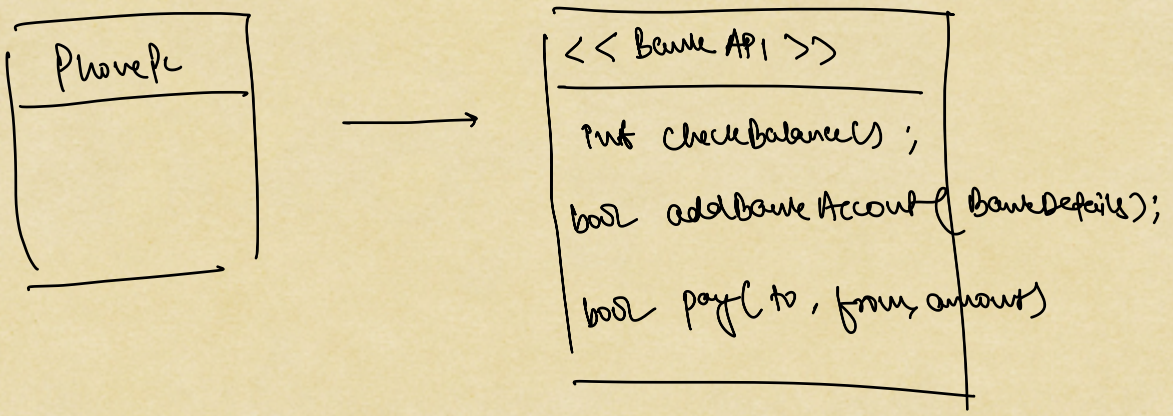
* whenever you are using any 3rd Party dependencies, make sure that you always connect to the dependency via adapter.

⇒ How to use adapter:

→ Think about what functionality we want to gain from the 3rd party

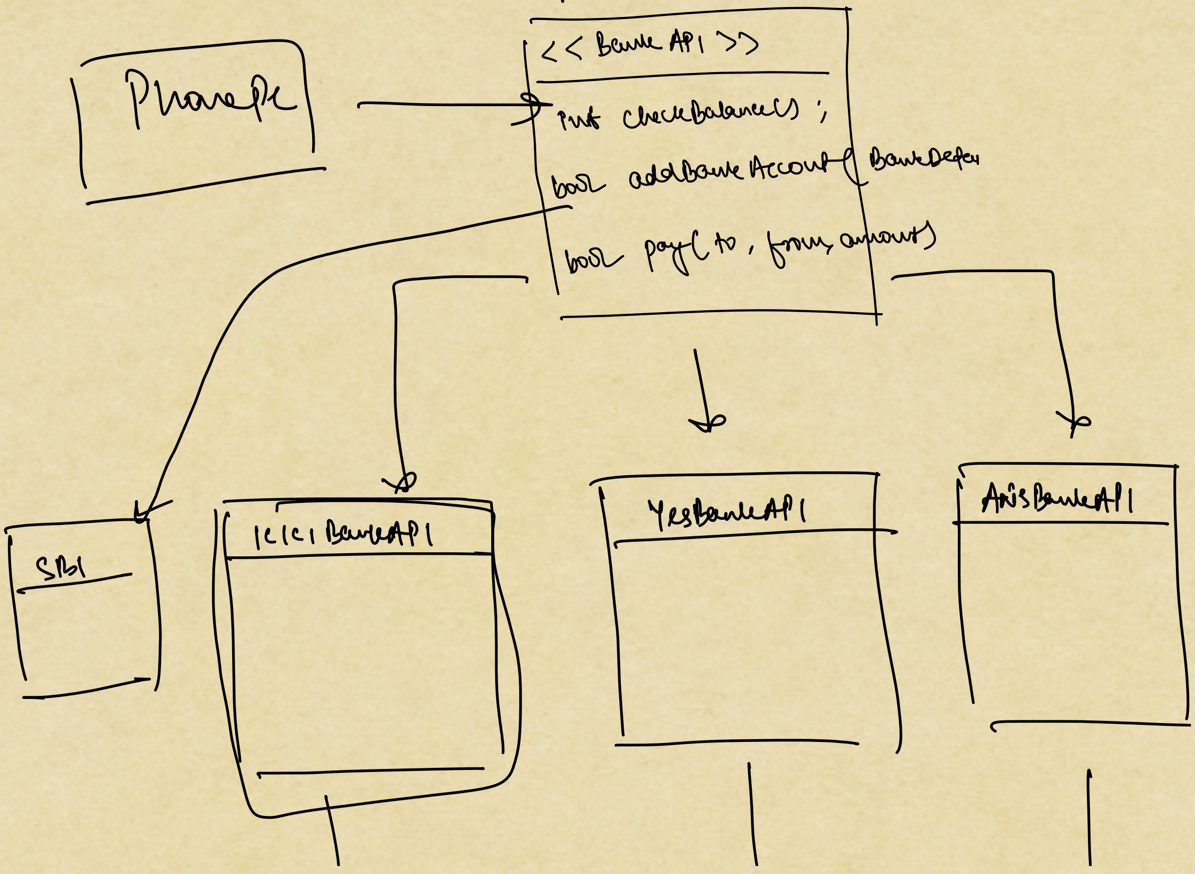


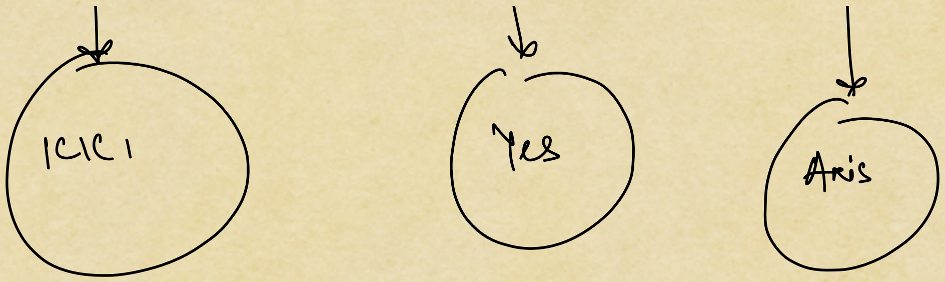
⇒ Create an interface having the methods that will do the actions that we want :-

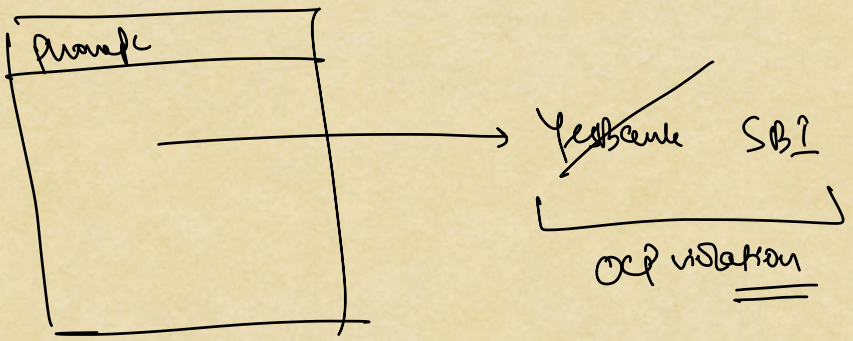


PhonePe
<< BankAPI >>
int checkBalance() ;
bool addBankAccount(BankDetails);
bool pay(to , from, amount)

⇒ Start creating impl. classes that implement the interface by talking to 3rd party dependencies :-



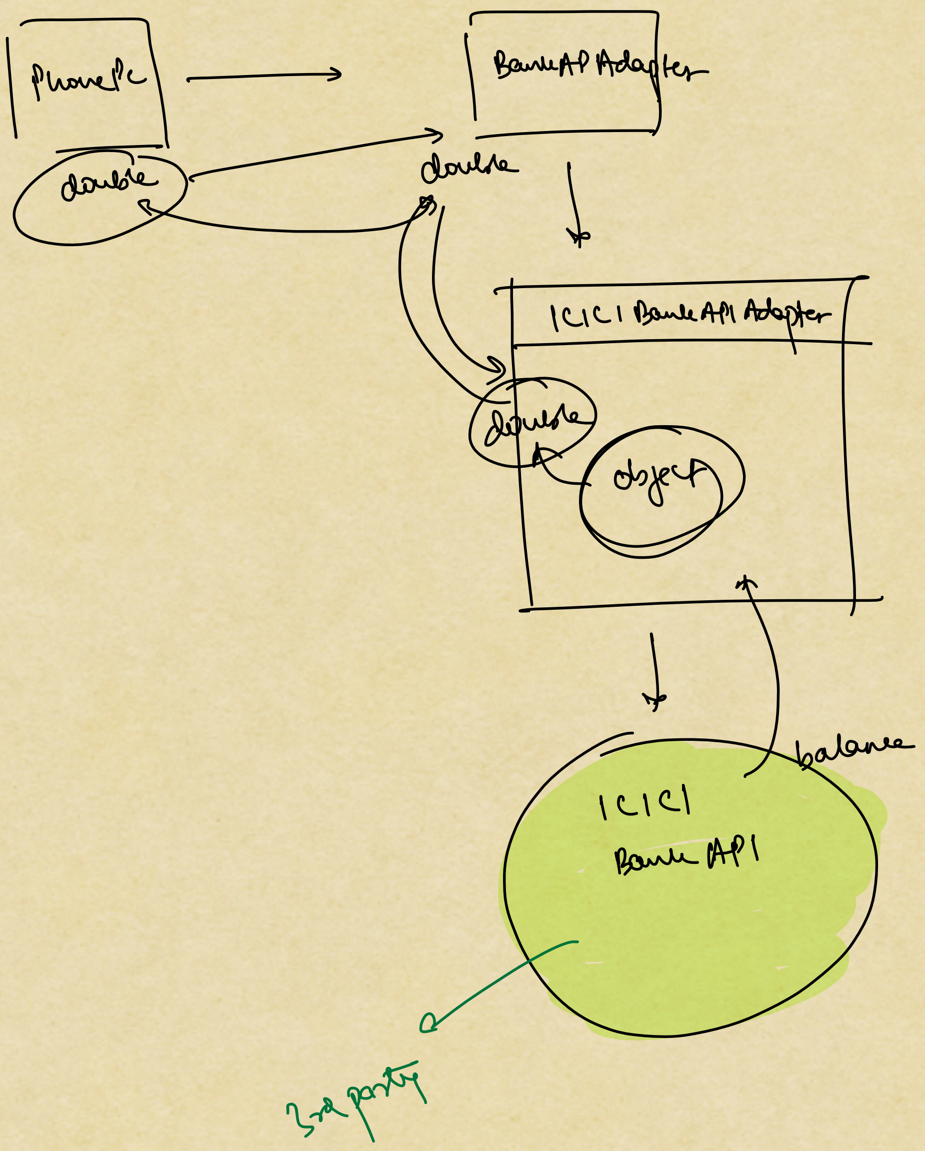




Calendly

↓

Google | teams | zoom etc

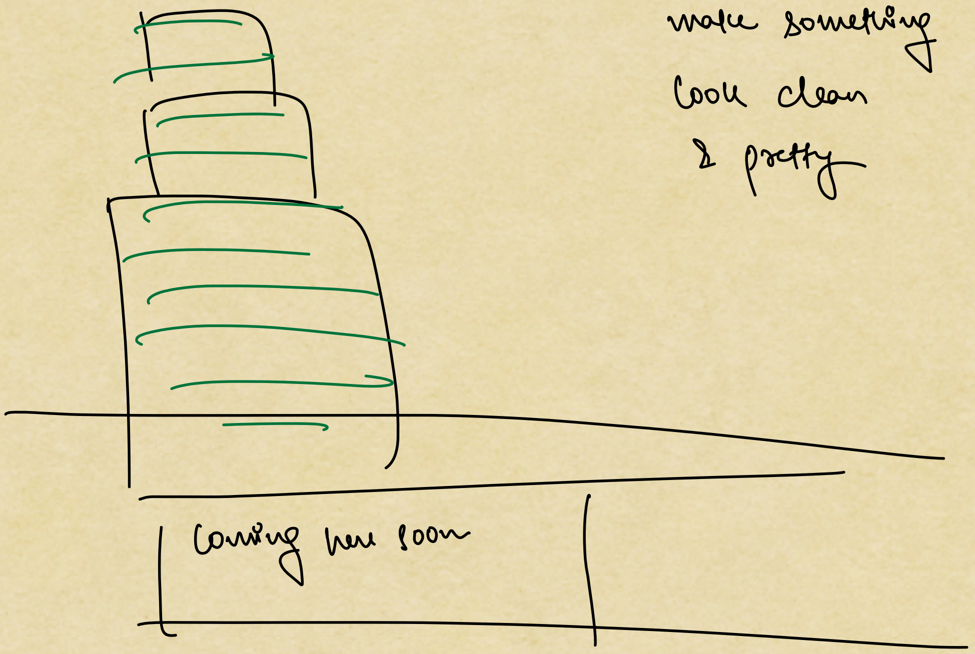

PhonePe
BankAPI Adapter
double
double
ICICI BankAPI Adapter
double
object
balance
ICICI
Bank API
3rd party

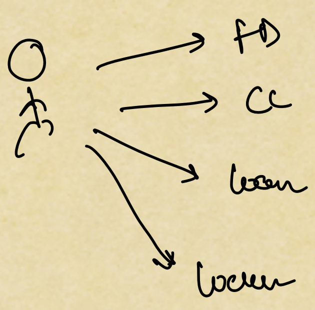
Facade Design Pattern:

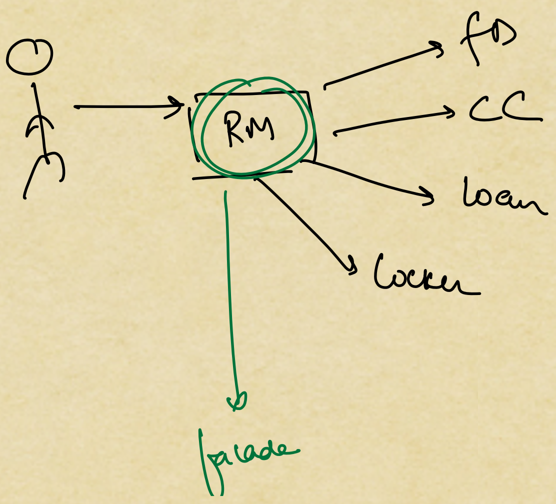
Facade ⇒ boundary | appearance | hidden

↓

make something look clean & pretty

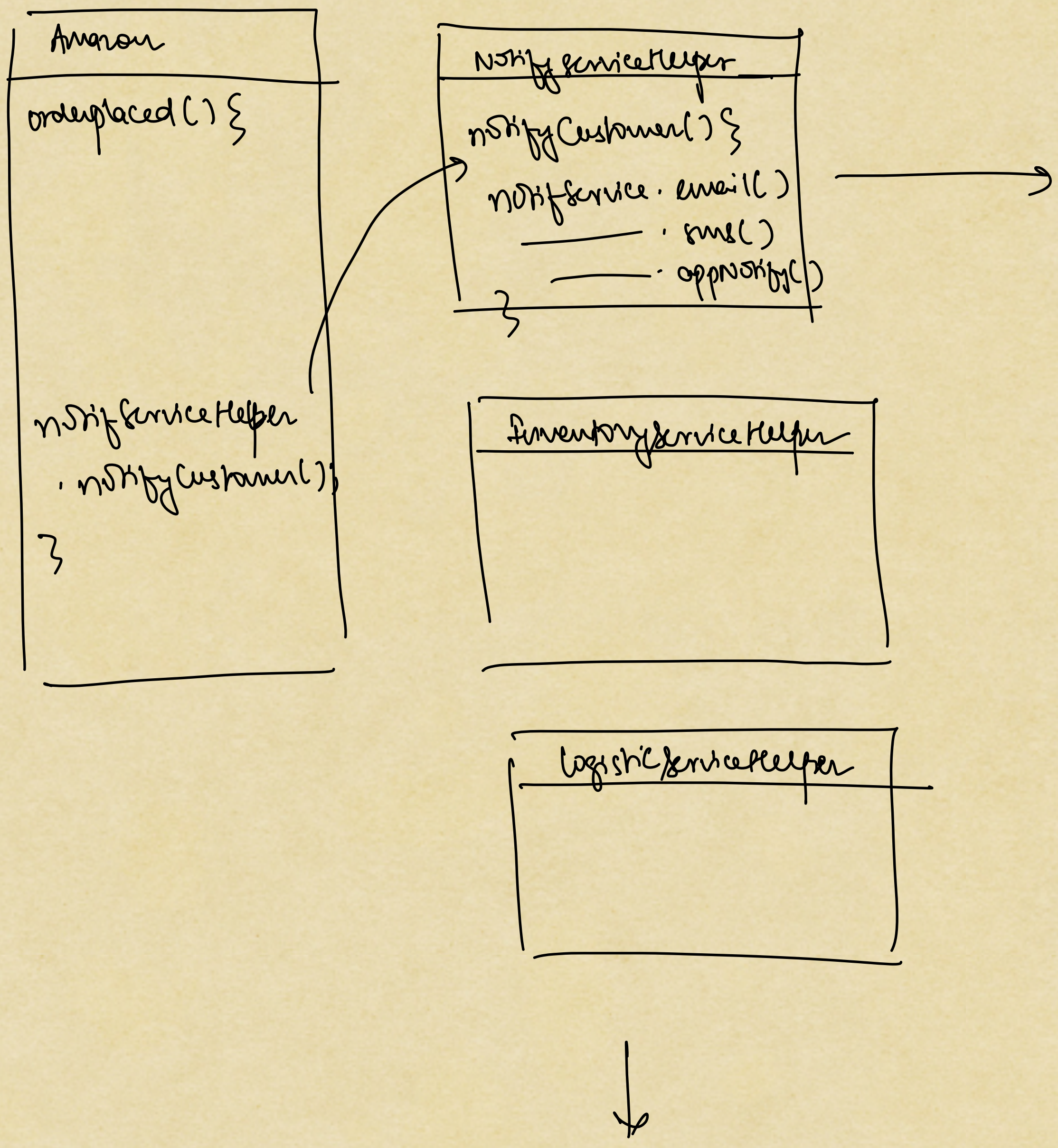






```
AmazonOrderService

placeOrder() {
    userService.validateUsr();
  ✓ invSer.checkStock();
  ✓ paymentService.doPayment();
  ✓ paymentService.generateInvoice();
  ✓ invSer.updateStock();
  ✓ logisticService.generateLink()
  ✓ logisticService.update();
  ✓ invSer.processOrder();
  ✓ notifyService.email();
  ✓ notifyService.sms();
  ✓ notifyService.appNotify();
}
```



```
orderplaced() {



notifServiceHelper
 .notifyCustomer();
}
```

```
notifyCustomer() {
 notifService.email()
 ________ .sms()
 ________ .appNotify()
}
```

AmazonOrderService

```
placeOrder() {
userService.validateUser();
InvServiceHelper.process();
paymentServiceHelper.process();
logisticServiceHelper.process();

notificationSerHelper.
                notify();
```

Code => HW