

# LLD - Decorator and Facade

# Agenda

- Flyweight registry  
+ Impl.
  - Decoration
  - Facade
- } 2 hrs

---

## Flyweight

- Scalability
  - Objects
- 

02   
03 

- out of memory

- Fly weight   
intrinsic - does not change  
extrinsic  
↳ change!

Bullet {  
x, y, z } extrinsic  
speed  
image } intrinsic  
bullet type }  
}



Flyweight  
(i attribute)

①

Bullet & {  
image  
bullet type  
weight

②

Flying Bullet

{  
 $x, y, z, i$

Bullet  $b_j$

10,000  $\Rightarrow$  Bullet



Image

Bullet  $\rightarrow$  9mm

45mm flying Bullet

weight

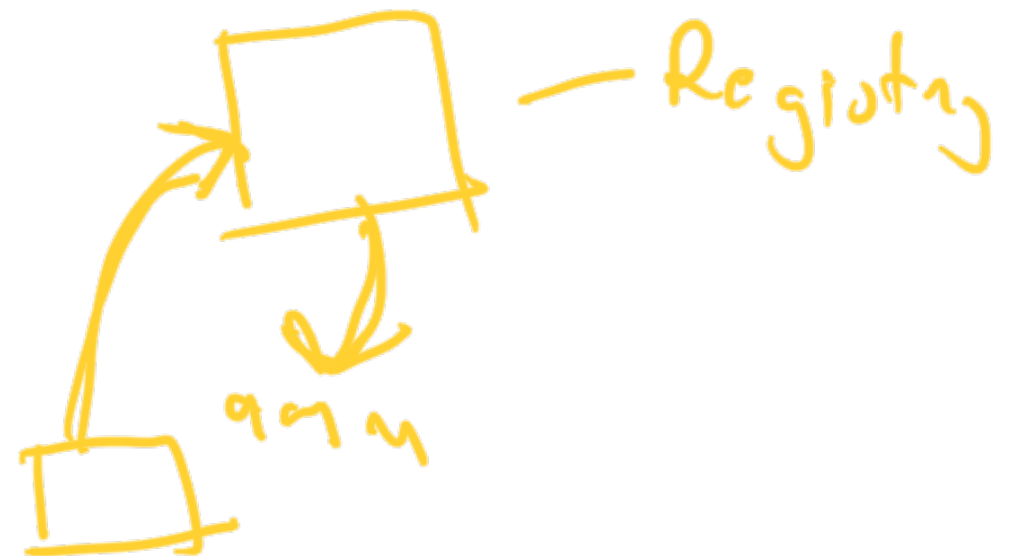
no dir

3 { 9mm Bullet  
45 mm Bullet  
50 mm Bullet }

100 mm  
200 m

Registry 2

add



registu  
fetch

get

Bullet Registry {

→ Map < Type, Bullet >

add Bullet (Type, Bullet)<sup>type</sup>

Bullet get (Type)

}

Application

→ add

→ get

→ fly weight → yes

---

Deconator

---

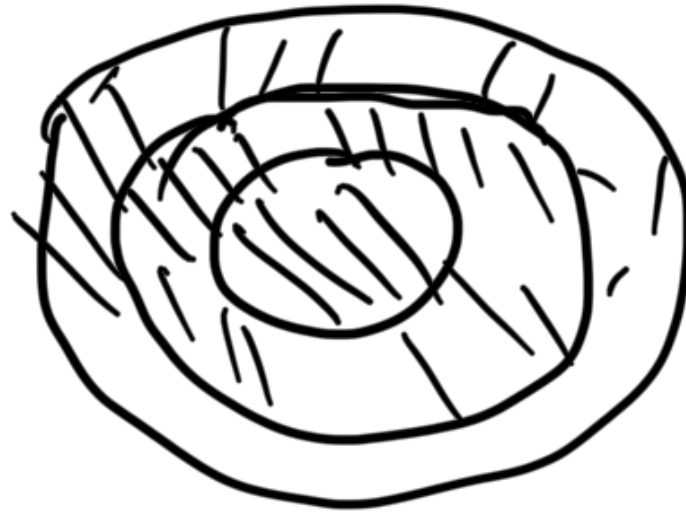
→ FE

→ react - router

with Router()

---

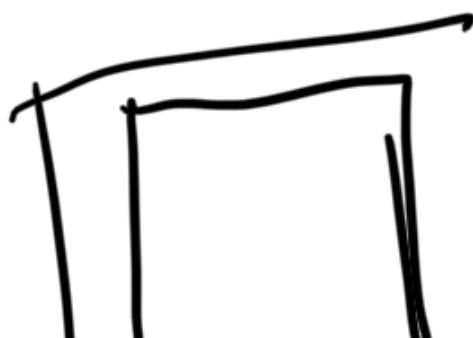
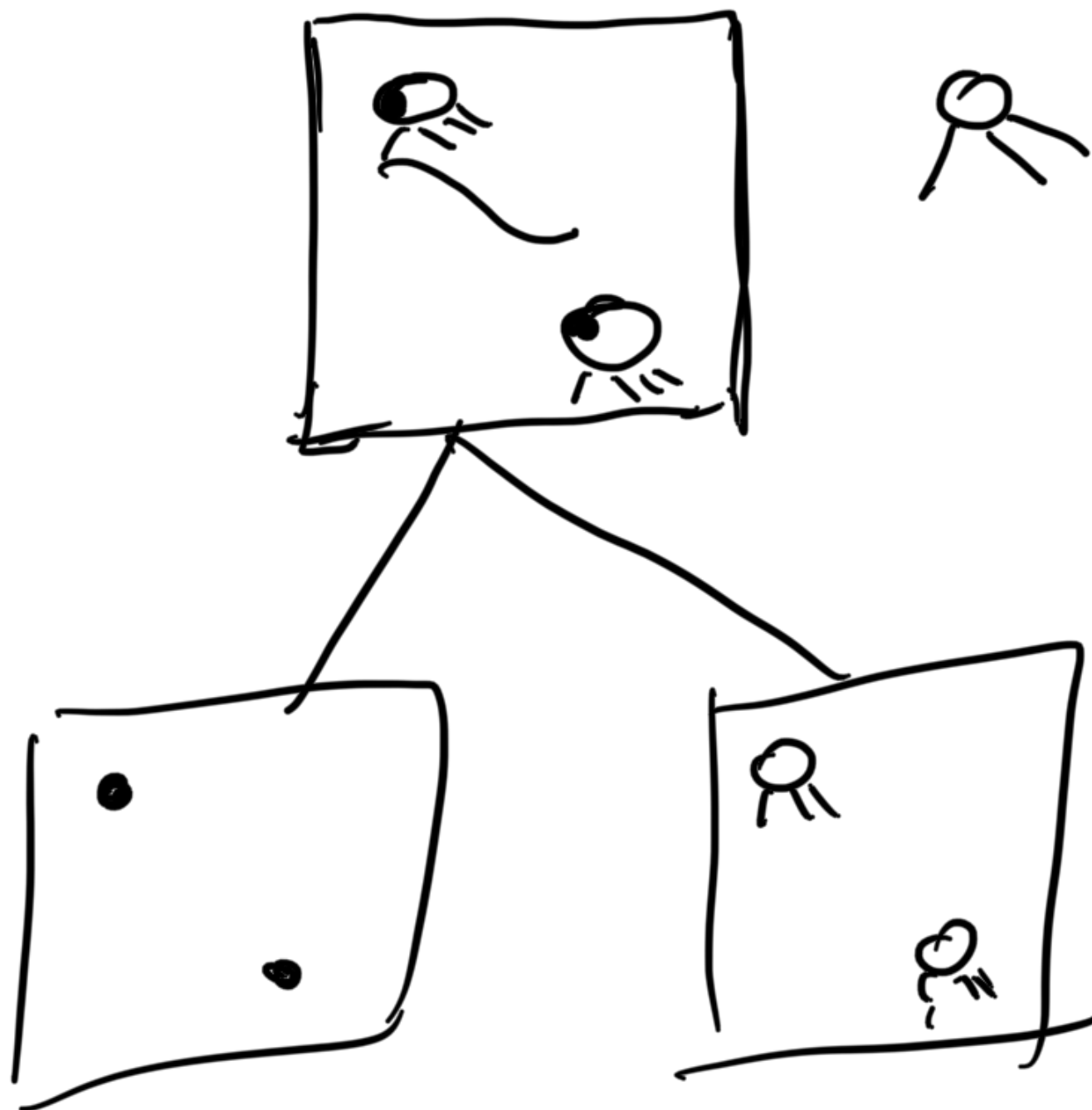
Deconator



→ with Router  
→ with Header }

Map





Map

↓ — with Spider

new Map()

withSpid. (new Map)

---

Emails

— Communication

PhoneService()

EmailService S

1-7-2017

Send { String encil, String  
message }

}

Encil, text  
↑      ↑

Communication {  
send { target, message }

}

↑

↑

Email  
Service

Phone /  
Service

---

Dongseon → Email + text + slack

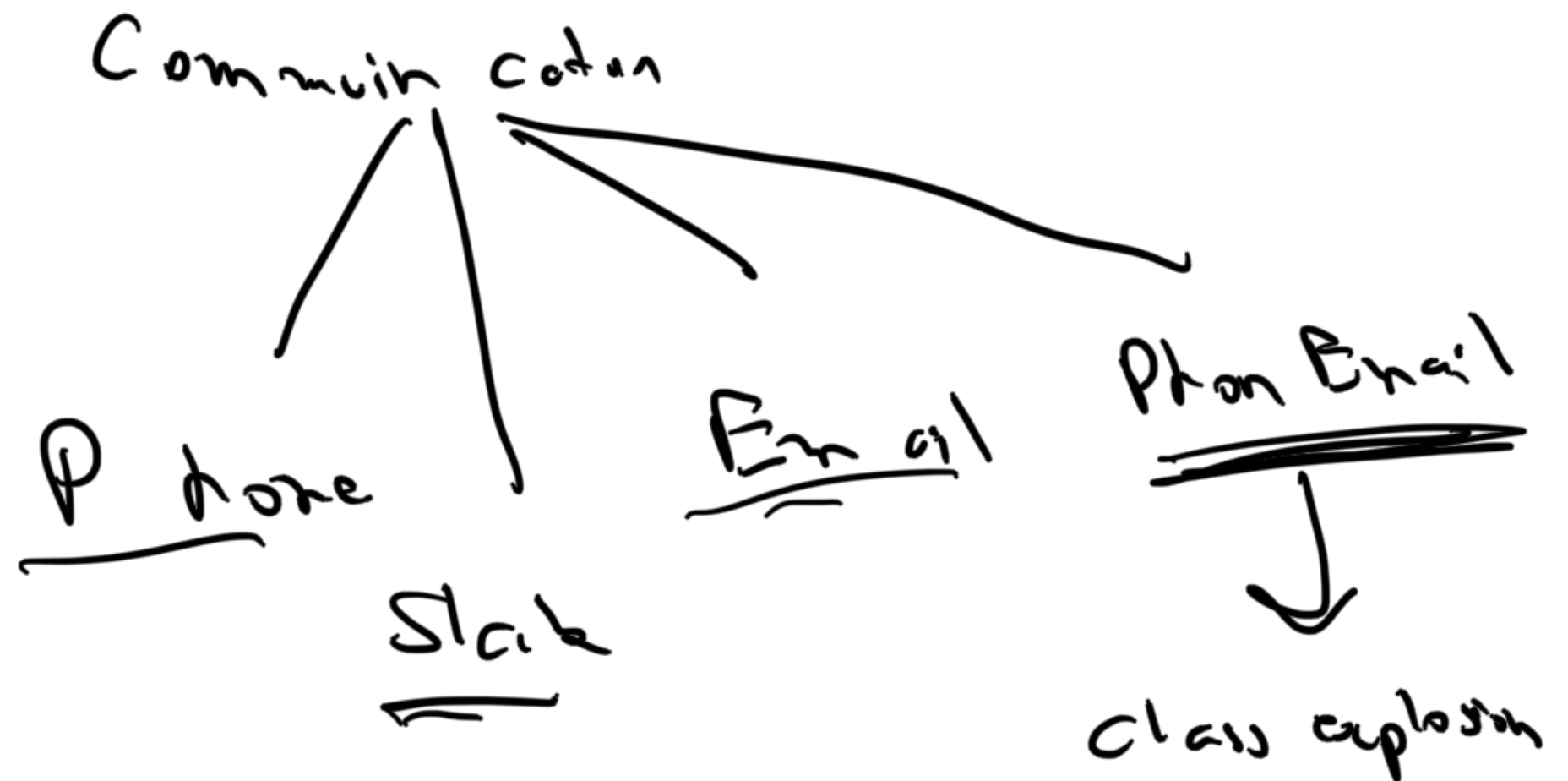
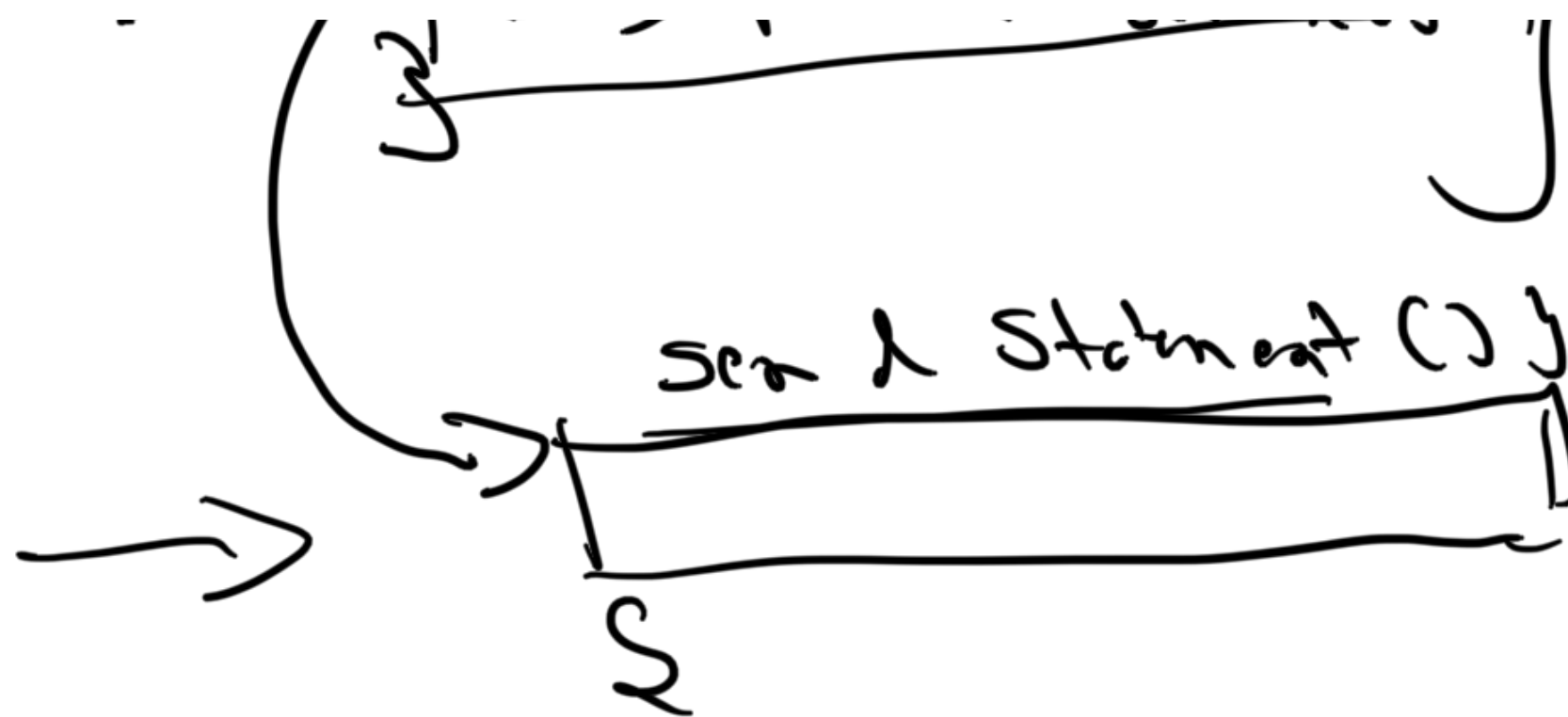
→ OTP

→ Credit statement → Email + text

Send OTP ( )

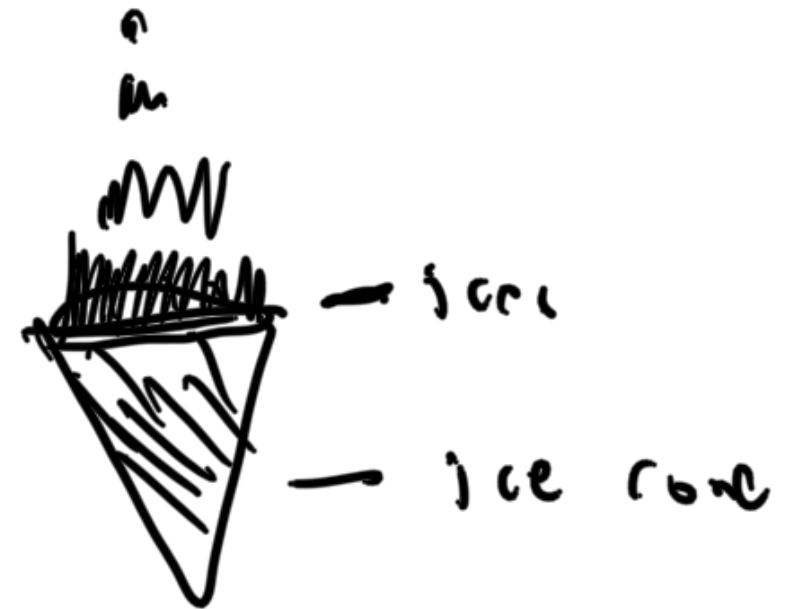
→ email. Send ( )

→ phone. send ( )




Tie behaviour to hierarchy

Composition - Recursion



→ extend functionality without



Below the text "without", there are two small, simplified diagrams of ice cream cones. The one on the left is a simple triangle with a wavy line at the top. The one on the right is a similar triangle with a wavy line at the top and a small circle above it.

modifying my class



Deconstructor

①

Destructor

→ read()

→ write()

}

(2)

Con create class

File DataSource imp. Database {  
read();  
write();

}

(3)

Create a decorator abstract class

DatabaseDecorator implements Database {  
→ Database db  
→ read();  
→ write()



↓         

① De constructors implement main interface

② Composition - Database in my Account.

④ Concrete Accounts

Encryption Decryption. extend BDFS

need ( ) {

String obj = db.need ( );

```
} return decrypt(obj)
```

```
write() {  
    input
```

```
String encrypted = encrypt(input);  
return db.write(encrypted);  
}
```

---

Compression Decondon ext. Db {

```
read() {
```

```
String object = db.read();
```

```
return decompress(object);
```

```

}
write ( input ) {
}
db. write ( compress ( object ) )
}

```

---

① Database db = nc ~ File D S W;

nc ~ Compressed ( db ) →

nc ~ Encrypted ( db ) →

db

en = new Encrypted (db)  
new Compressed (en)

---

- ① Common interface → Database
- ② { Concrete subclass → basic behaviour  
→ File Data source
- ③ { Abstract decorator  
→ implement Database  
→

reference → U at db src

---

④ Con create Acc on db

→ Compression

extend Base Data 2

① need

② modify

---

db = new File Database ( )

en = new En (ab);

comp = new comp (~~en~~)

comp.write("Hello")



comp.write("Hello")

en.write ("Hello")



en.write("Hello")

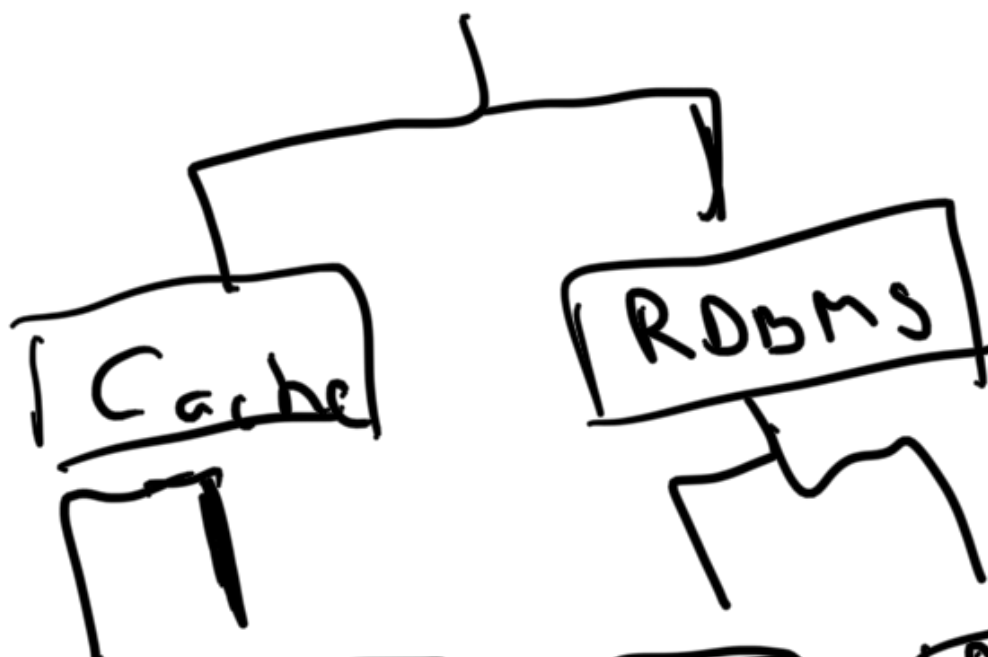
en.write

EC "Hello"

## Abstract F

- Create objects
- family  
subclasses

Database



## Decorator

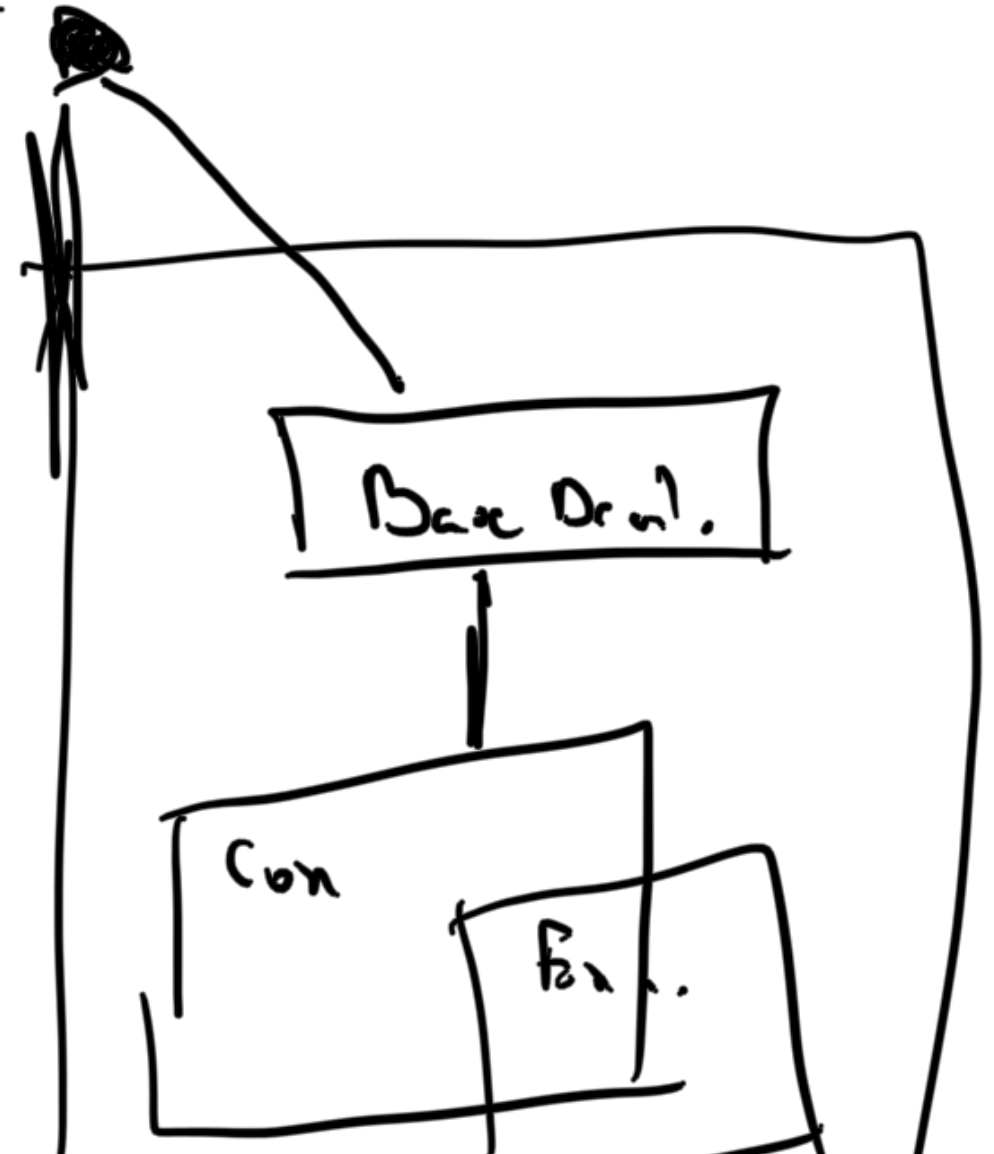
- add functionality
- How to add functionality  
w/o modifying it



Redis Dynamp Msg Pos. | tic behavior to

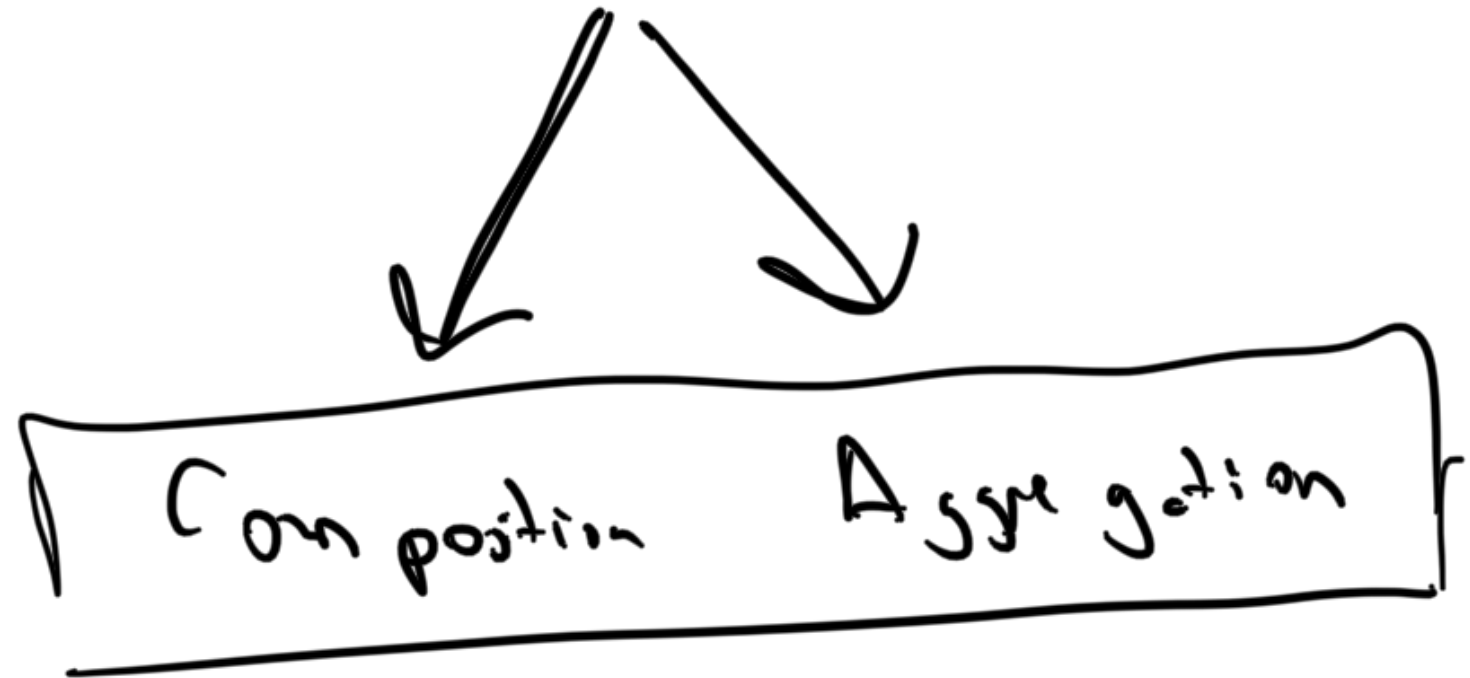
Priority  
E + C

Database





Inheritance — Association



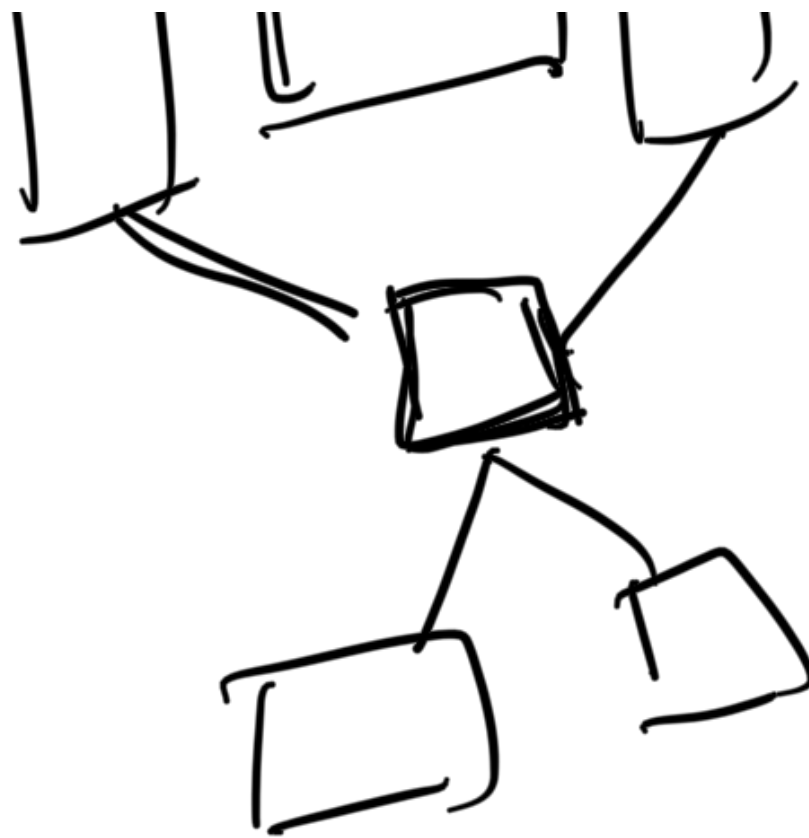
Adapter — interoperable



Direct

Decoupled





A. A.

compose

External  
+ Adapter

Decorator  
Product