LLD - Adapter and flyweight

Tax Calculator

Old

New

type < old
new

```
calculateTax(Salary){
    if(salary < 10){
        tax = 10% of salary
    }
}
```

Factory - Set of related classes

Old

New

Condition — regime

Condition / multiple subclasses,

Abstract factory

- subclasses
- family of subclasses

family of products

→ factory of factories

India , new region

Indian

We want India.

factory — products / related classes
— on basis of a condition
you want to create
diff. instances

abstract → family of products

factory → hierarchy

Agenda — Structural design p.

→ Adapter

→ Flyweight

→ Decorator

→ Facade

Structural

↳ struct me

↳ interact

- SOLID

↳ loose coupling

↳ tight cohesion

Adapter

incompatible interfaces

adapter allows us to work with incompatible interfaces.

[ ] users - JSON

① Change from JSON to XML? X

② Create new API with xml content

| users - json

| users - xml ✗

③ JSon

| user format=xml

Java → JSon → xml

Java → JSon

→ xml

Payment Gateway

- Razorpay
- Cashfree
- PayU
- Stripe
- PayPal

adopt th.

Razorpay API

↳ RazorpayJS

- generateLink
- makePayment(email, amount)
- checkStatus

↳ OK, Failed

makeOrder() {

razorpayAPI.makePayment(email,

an ent)

}

PayU {

→ createLink() ;

→ Create Payment (email, entity) ;

⇒ checkStatus()

↳ success ,

FAILURE

}

1. ~~Different methods~~

2. Different payment

3. Different return values

- Razorpay API →
- PayU API →

① Create common interface - adapters.

interface PaymentProvider {
→ linkCO;
→ do (Payment Request)

→ Lombok checkObj;

DTO - Builder

|

CommonDTO

PaymentRequest {

[String name;
String email;
Double amount;]

PaymentStatus {

OK , ERROR

}

(2) concrete interface classes

API → RazorPay API

API → PayU API

RazorPay API {

link() {

return response. } delegate

```
pay(PaymentRequest) {
    return rp.makePayment(
        req.email, req.name,
        req.amount
    )
}

check(id) {
    return rp.checkStatus(id)
}

toRPEnum(): PaymentStatus {
    switch
```

if else

Flyweight

extensibility principle

→ game development

→ Scalability → RAM

Publ

(x,y,z) (x,y,z)
size direction
u Send u Send ia

- Player
- Map
- Gun
- Trees
-> Bullets

bullet {

Double x; - e

Double y; - e

Double z; - e

-> Double weight; - int

-> String colour; - int

byte[] image - int

.09 , .55 , .78

Bullet Type type = int

}

$4 \times 8 + 8 + 8 + 1\,KB$

$= 32 + 16\,B$

$\approx 1\,KB = 1$ bullet

$200 \times 200 \times 1$

$40000 \approx 40\,MB$

$\approx$ 2000 x 2000 x 1

4 0000 KB

4 MB

bullets

flyweight

extrinsic → changing

intrinsic → unchanging

→ State

→ behavior.

intrinsic

↳ does not change

Class

Extrinsic

Intrinsic

(Flyweight)

Bullet

Bullet
(flyweight)

- image
- weight
- type

FlyingBullet

- x, y, z
- speed

Impl.

(1) Create flyweight

Bullet 의 image

→ intrinsic

weight
type
color

② Create extrinsic state object

FlyingBullet {
  x, y, z
  • speed
  • bullet bullet
}

③ Compose bullet w FB

FlyingBullet {
  ...

Bullet bullet

J

Figs

10,000 → 10,000

4.8 × 10^5 B

480 KB

Bullets

1

1 KB

481 KB

4KB →