After Mid

TOPIC NAME: Design Pattern DAY: TIME: DATE: / /

- Best practices of Obj oriented SDEs.
- Solution to general problems.
- Obtained by trail, error over time.

Usage :

① common platform for developing
  + easily connect with other devs — pick up in another project
  |ex : use singleton pattern - download data - obj same everywhere

② Best practice + solution + help learn fast

Types of Design pattern

23 patterns in ③ catagories :

① creational ② structural ③ behavioral

extra J2EE

① Creational pattern : ② create obj hiding create logic

unhide : car = new car(- - -)

↳ but this is hidden here.

Adv : advantage of flexibility of creat obj


GOODLUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

② structural Pattern :

+ inheritance

③ Behavioral pattern ;

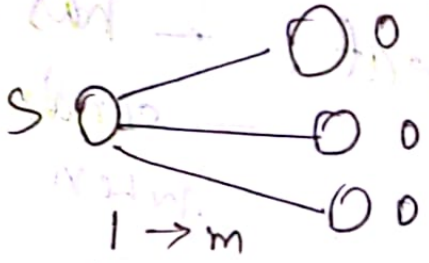
+ communicate between obj

ex: Acc – customer

cash give, take communicate

## Observer Pattern

- one to many rln between objs
- + when change sub → observers obs are notified




usis – text – students

ex: class subject :

① add observer ② remove observer

③ Update ④ state = new ( )

↳ notify observers

notify when a new fan added

TOPIC NAME: _____ DAY: _____

TIME: DATE: / /

observer = fan example

class observer

① add sub ② remove sub

③ update subject

ex: f1. addclub(c1) → subject

f2. add club(c2)

f2. add club(c2)

→ f1 will be notified when f2 joins c2

Participants:

① subject: ← has observers

— sends noti to observers when state changes

② Observer: - gets notified of changes

concreate sub: sub class of sub
↳ film club, 470 teacher

concreate observer: sub class of obs
↳ 470 students

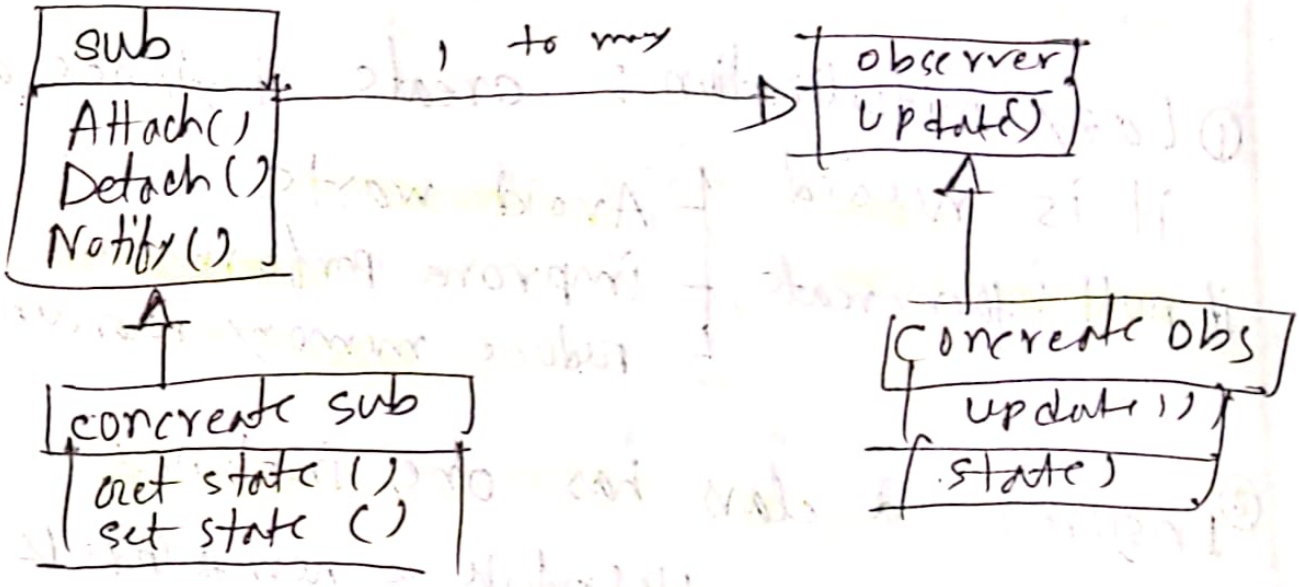


GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

structure




concrete observer extends to observer

film for to fan { → add cub
remove fan

So concreate uses the obs's operation by extending to its superclass

Same for cellb and concreate cedeb

key : obs always looks for change

concerned w object update

name
problem
solution

TOPIC NAME : _____ DAY: _____

TIME: DATE: / /

# Singleton Pattern

① Lazy initialization : create instance after it is needed

if null : then create
- Avoid waste
- improve performance
- reduce memory requirment

② Ensures a class has one instance.

Helpdesk = new Helpdesk ()

↓ instance ↓ class call

Reason :

i) incorrect programme if multiple thread

ii) overuse resource : database connection

↳ once is enough then use it for others

Creational pattern : indirecty create obj hidden

(101) One obj for 1 printer then 100 comps use it

"GOODLUCK"

"GOOD LUCK"

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Rather than creating same objs in multiple class — create 1 obj for multiple class

① Helpdesk class – private
tservice()
if no instance → public class get
h = new helpdesk()
return h
static

② student class
↳ call get

③ Teacher
↳ call get

first call obj created, other times same obj returned.

main class
class
Singleton → create instance

other class → access

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

# Adapter Pattern

client req → Adapter (convert req to compatible) → Adaptee external incompatible

- structural pattern :
  - concerned w how class and obj compose to form larger structures
- Intention : Adapter makes client interface compatible w existing system

Adapter (wrapper)
↳ cause wrapping of it

- class (inheritance)
- obj (obj creation)

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Ex —

Existing system :

ctg Pizza :

① sausage ()

② bread ()

main interface :
of client

Pizza :

① topping ()

② bun ()

adapter

ctg Adapter :

① topping {
sausage () }

② bun {
bread () }

Participants :

① target : client interface ex Pizza

② client : who call the methods

③ Adaptee : existing code the needs to be adapt

④ Adapter : Adapts adaptee

TOPIC NAME : _____ DAY: _____

TIME: DATE: / /

# Object Adapter

take obj of adapter class and call it in adapter.

Adapter : → obj use to call methods

ctg = ctgPizza()

① topp { ctg.sausage() }

:

class

Pro:

① Override adapter's methods

Con: Can't get subclass methods. only parent and this.

obj

cons

① Can't override methods

Pro: gets parent, sub class objects methods.

GOOD LUCK™

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

# Software testing

- evaluate if system satisfies requirements

Testers : software tester, Developer
leader, manager
end user

Error - fault - failure

① Classification (source of info)

① black box test (input output test
Send user - no code knowledge
- functional -

② White box - - structural, programme based
↳ devs - knows all codes

③ Grey - mixer

② classification:

TOPIC NAME: _____ DAY: _____

TIME: DATE: / /

## Non functional test

① Performance — load taking capacity

② Usability — friendly interface ฯลฯ

③ security

④ Portability — all device

functional :

① Unit test :

↳ Develops do befor giving to test team

↳ isolate each pant and check rea

② Integration test :

↳ combine units and see if work together

↳ Bottom up : low to High

↳ Top down : High to low

③ System testing :

↳ test as a whole

↳ quality

↳ specialized test team

GOOD LUCK™

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Non func :

Performance test :

speed, capacity, stable, scale

Load testing :

APPly max load ← access
input data manipulate

↳ behavior at peak test

Stress test :

test in abnormal situation

↳ take away resources

↳ overload

↙

break point find

shut down network database etc

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

# Test Driven Development

## TDD

① write test then design → xp agile

Steps :
1. Add test
2. Run test
3. change code
4. Run test
5. finalize

---

## Unit testing

- Automated testing
  - tests small code
    - ex: methods, funcs
  - tmarrow scope : at a time focus on small chunk

white box testing

GOOD LUCK™

GOOD LUCK™

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

Steps : (white box)

① Setup test

② each test → test method
                ├ statements test

③ Execute + result

④ match result

Test

Manual
Users

Auto
dev team

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

# Fixed Test Myths

① Testing is not too expensive.

Myth : It is expensive.

- early testing saves cost
- No testing - complications
  - ↳ later cost high

② Testing is not time consuming

- fixing error is

③ Test can be started before fully developed

↳ only in waterfall not

④ Complete test is not possible.

- many paths (if, else) so not all test possible

⑤ Tested software is not bug free

⑥ Defects missed are not due to testers.

- test strategy
- T, cost, resr

GOOD LUCK

GOOD LUCK

TOPIC NAME : _____ DAY : _____

TIME : DATE : / /

⑦ Tester not responsible for quality of product

⑧ Test Automation — test case generate auto to
↳ shouldn't be used always
+ need manual testing first - then test automate after stable
+ can't be used if rear keep changing.

⑨ Anyone can't test software.
+ only ~~white~~ black box test can do

DAY: TIME: DATE: / / TOPIC NAME:

| Black | Grey | White |
|---|---|---|
| 1. Internal working unknown. | 1. Some knowledge | 1. Full knowledge. |
| 2 Closed box test - data driven, functional - | 2. Translucent test | 2. Clear box test Structural, code based |
| 3. Performed by: end user, tester, developer. | 3. → same | 3. tester, developers |
| 4. - based on external expectations | 4. based on database and diagaflow diagrams | 4. base on internal code |
| 5. test time | 5. Partly time | 5. max time |
| 6. No algorithm test | 6. No algorithm test | 6. Algo test |
| 7. trial and error method test | 7. Same if known ← | 7. Data domain and internal boundary test |

GOOD LUCK

GOOD LUCK