1. Good Evening
2. We begin at 9:10 pm
3. Topic → Factories

Agenda

- Factory Design Patterns
  - → Factory Method (9)
  - → Abstract Factory (C)
  - → Simple Factory (OCP violation)

Factory Method [Inheritance]

Story: A userservice which will save user's details to the db

UserService 2

createUser(): ✓

getUsers(): ✓

updateUser(): ✓

deleteUser(): ✓

UserService (Client)

createUser(User user)

Database db = new Database("MySQL");

db.execute("_____");

}

}

String type

Database(String type) {
this.type = type;
}

Database {

Query getQuery() {

}

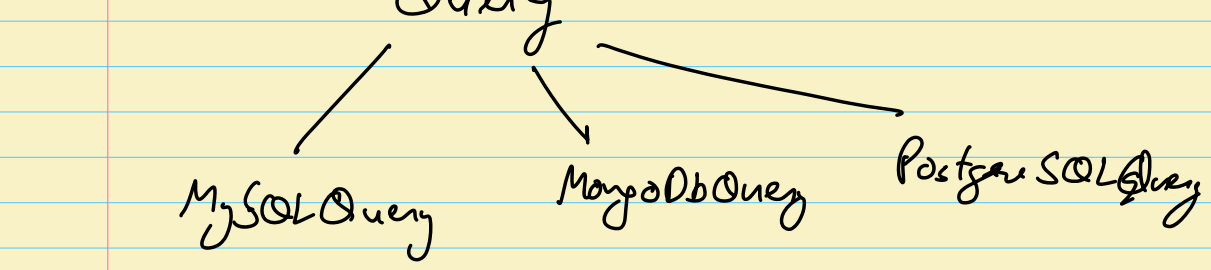
execute(String queryText) {

Query query = getQuery();

query.execute();

}

}



V1

Database {

Query getQuery() {

if (type == "MongoDb") {

return new MongoQuery();

}

else if (type == "MySQL") {

return new MySQLQuery();

}

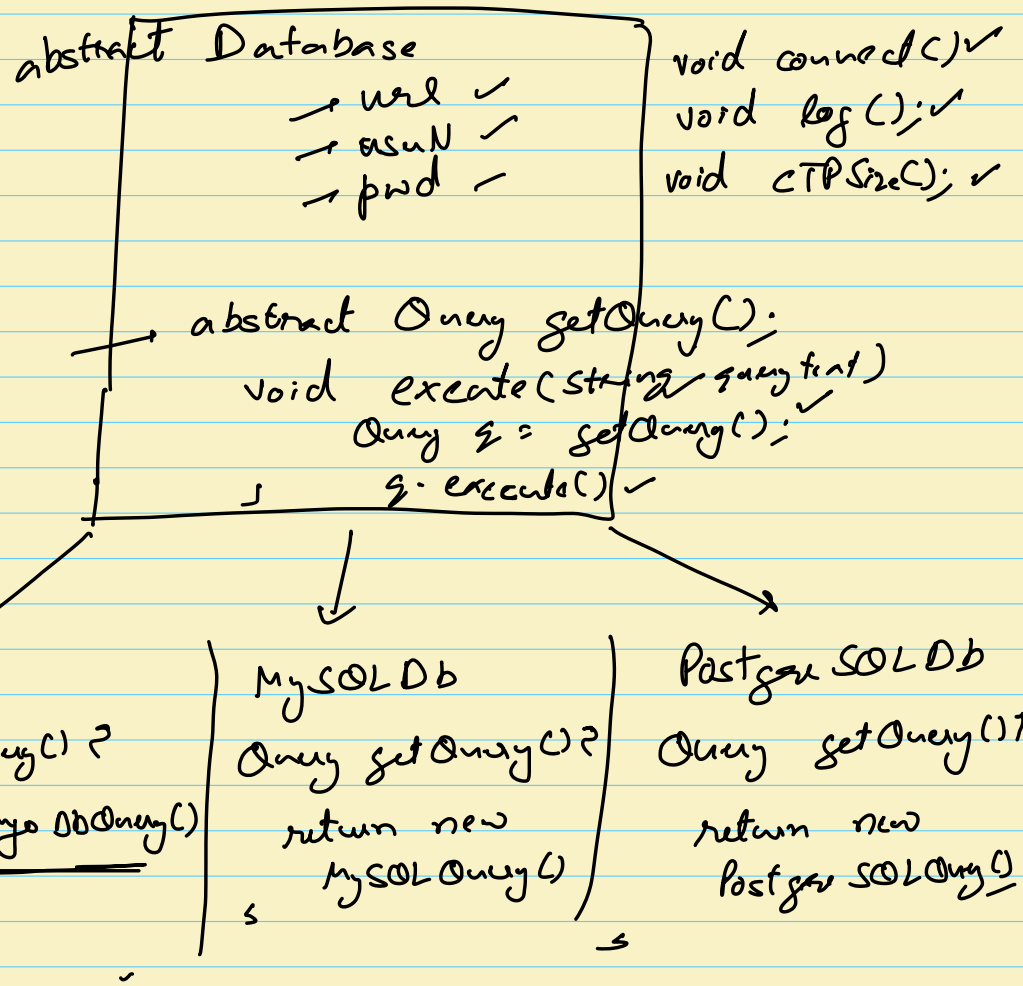
}

OCP violation

```
void execute (String text) {
    Query q = setQuery();
    q.execute(text);
}
}
```

Problem → OCP violation

Version 2



UserService

createUser() ?

{

Example

abstract class A ?

a void fun1()

void fun2() {
// work ✓
fun1(); ✓
// work -
}

{

{

class B : A ?

void fun1() ?

{ sout("fun1 in B")

}

}

class C : A

void fun1() ?

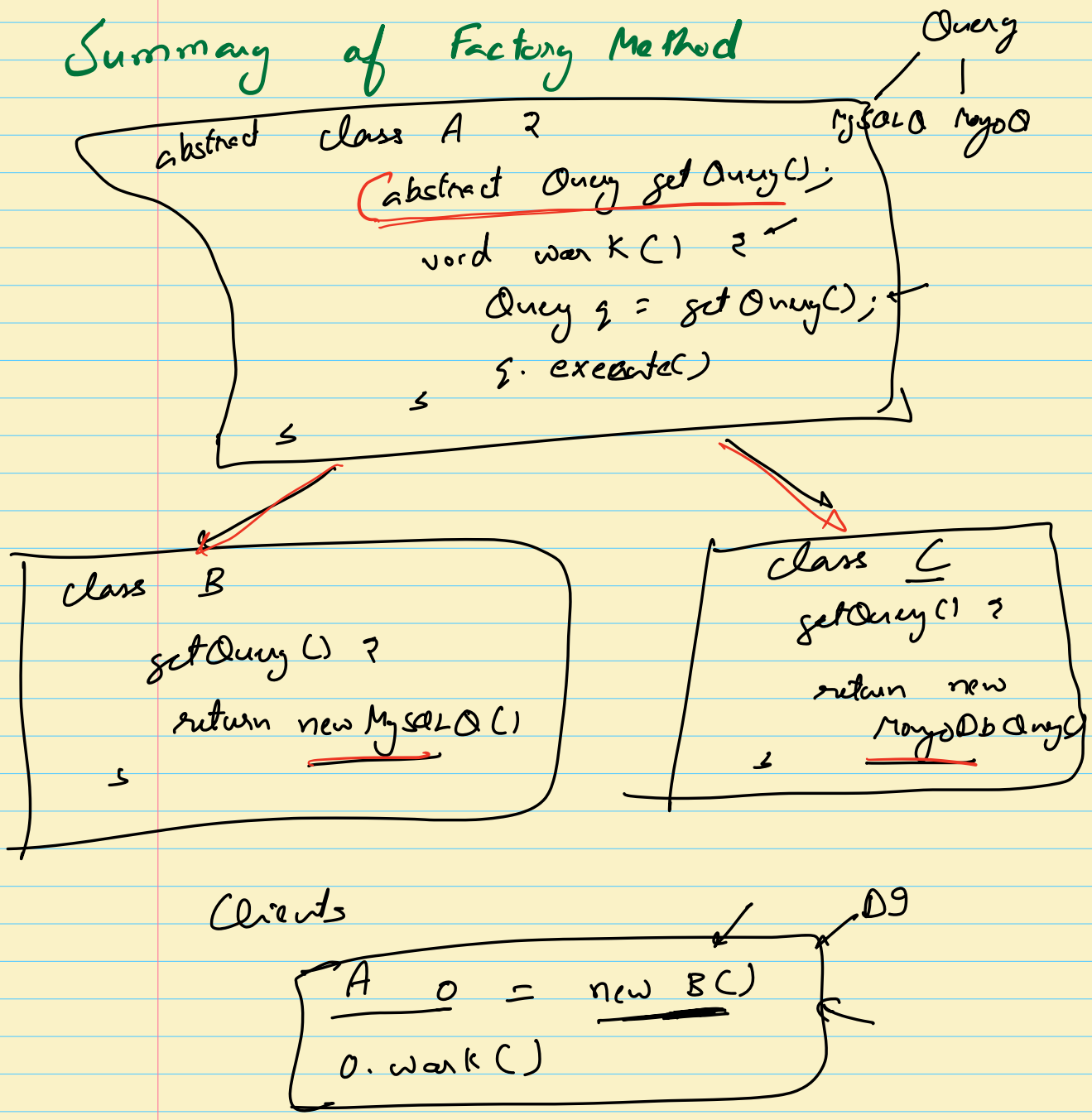
sout("fun1 in C")

{

{

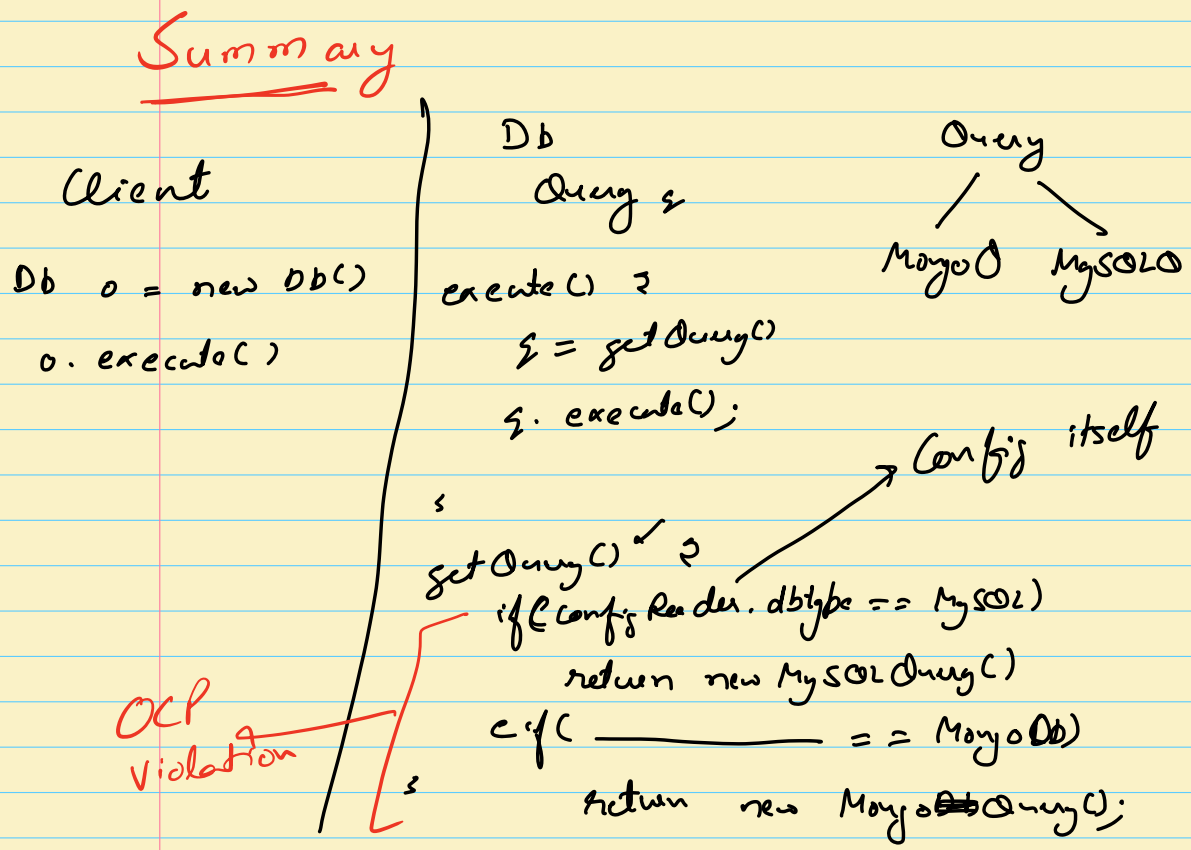
```
A o1 = new C();
o1.fun2();
```

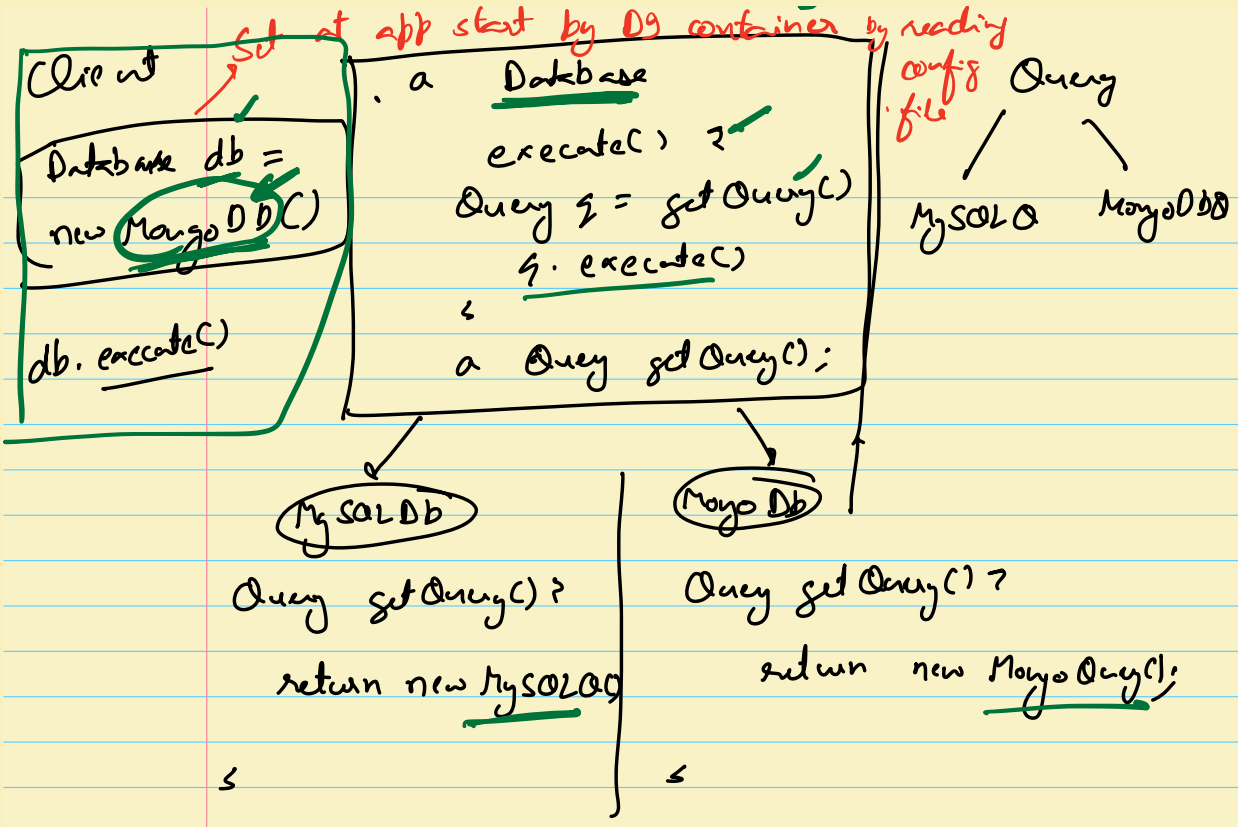
Summary of Factory Method

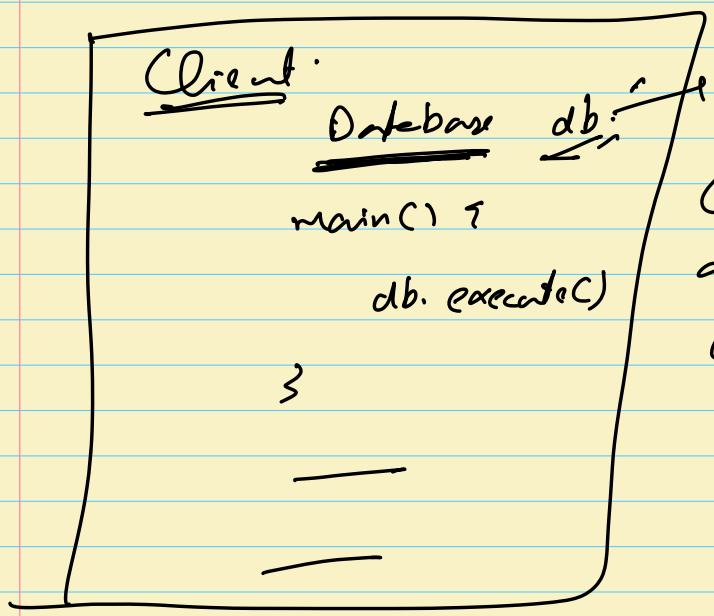


Client will instantiate A with appropriate instance at app-start using configs files & dependency injection.

Summary



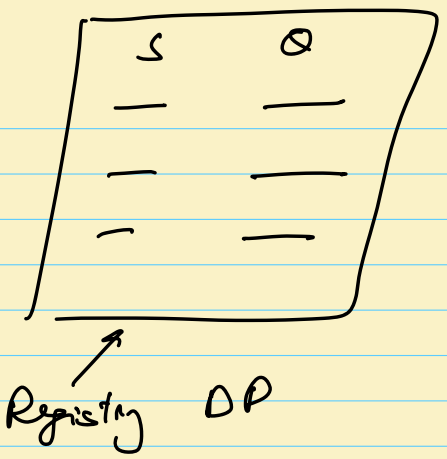




DI container, config files

{ < interface = "Database" impl = "MongoDB" />

- Factory Method
- Abstract Factory
- Simple Factory

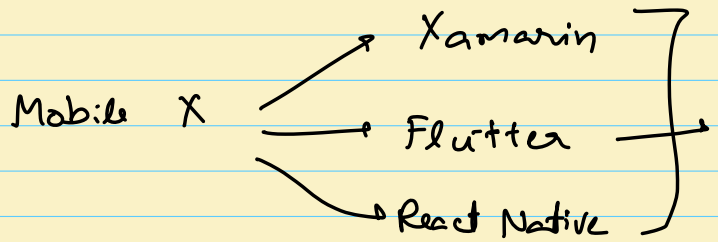


Break → 10:34 to 10:44

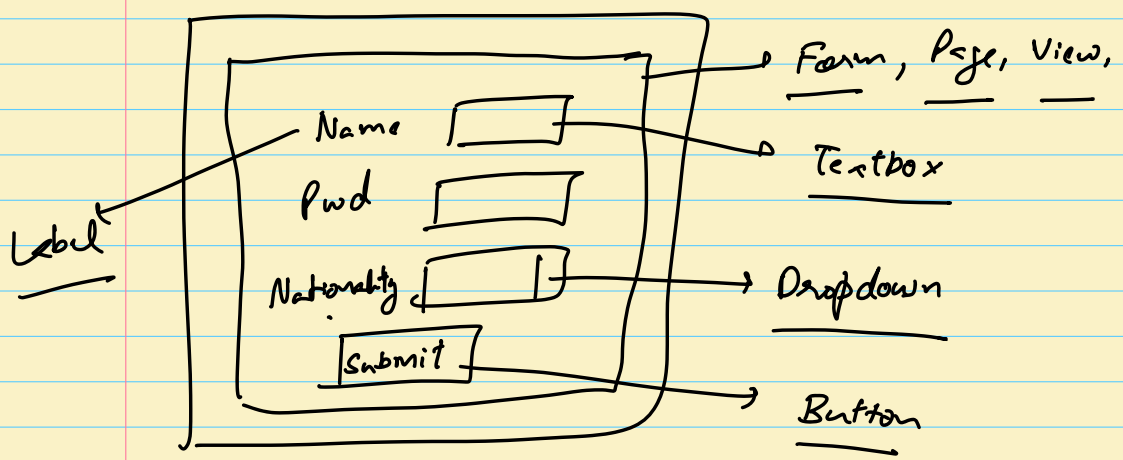
- Abstract Factory → 15
- FM vs AF → 15
- Practical / Simple factory → 10 → Not a SoF
  OCP violation

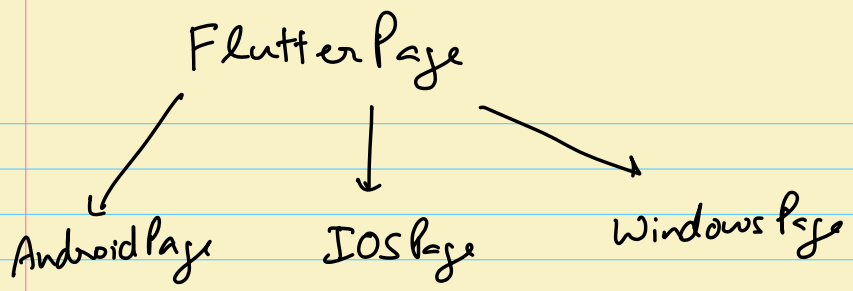
Story → 1. FM
2. Problem
3. Abstract Factory
4. Differences.

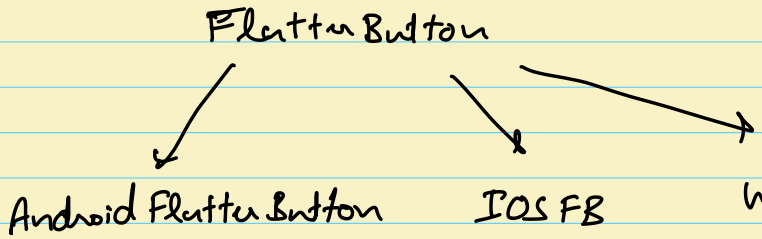
X Platform development



Story → [Wanting a cross platform framework like Flutter]







abstract FlutterPage ?

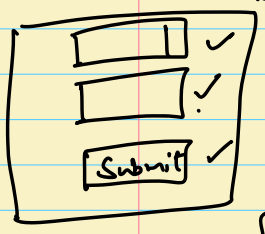
setRefreshRate();

setBackgroundColor();

FlutterButton b

FlutterTextBox t

FlutterDD ddl

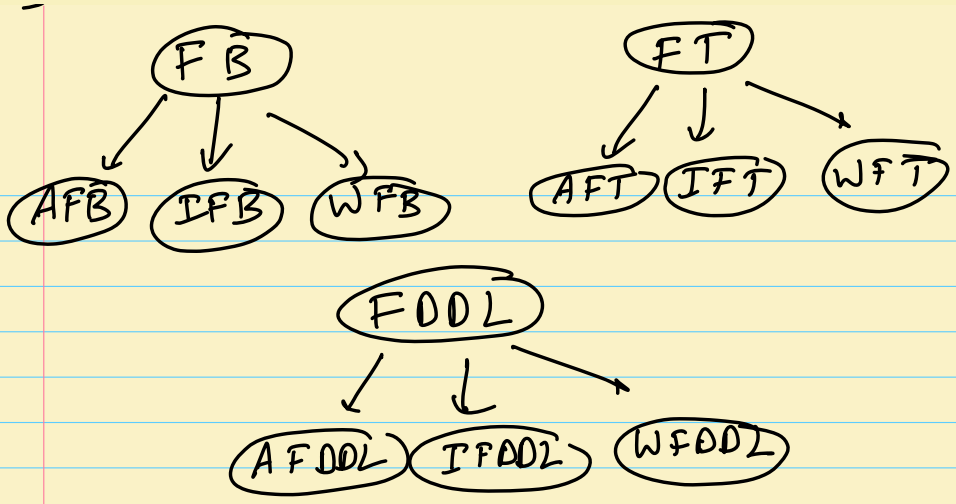


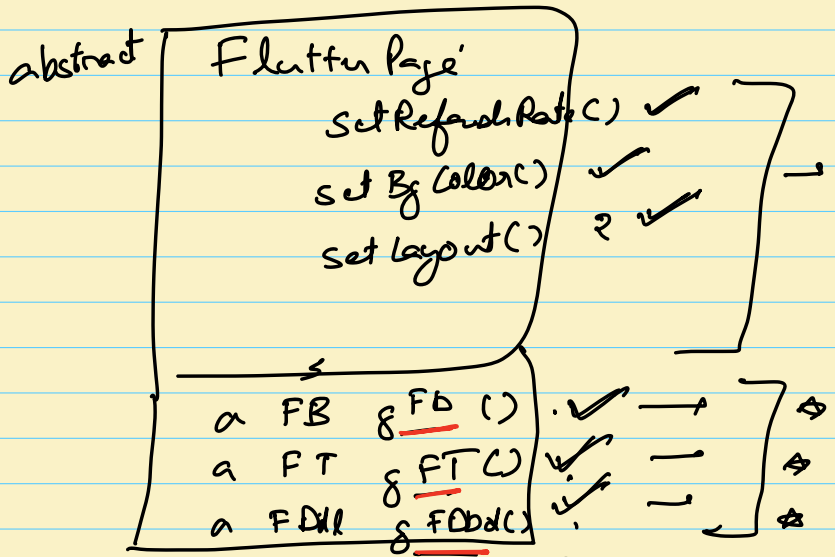

```
void setLayout() ?
    b = setFlutterBtn();
    b.setPos(100, 100)
    t = setFlutterTextbox()
    t.setPos(—, —)
    t.setSize(50, 250)
    ddl = setFlutterDD();
    ddl.setOptions(—)
    ddl.setSize()
}
```

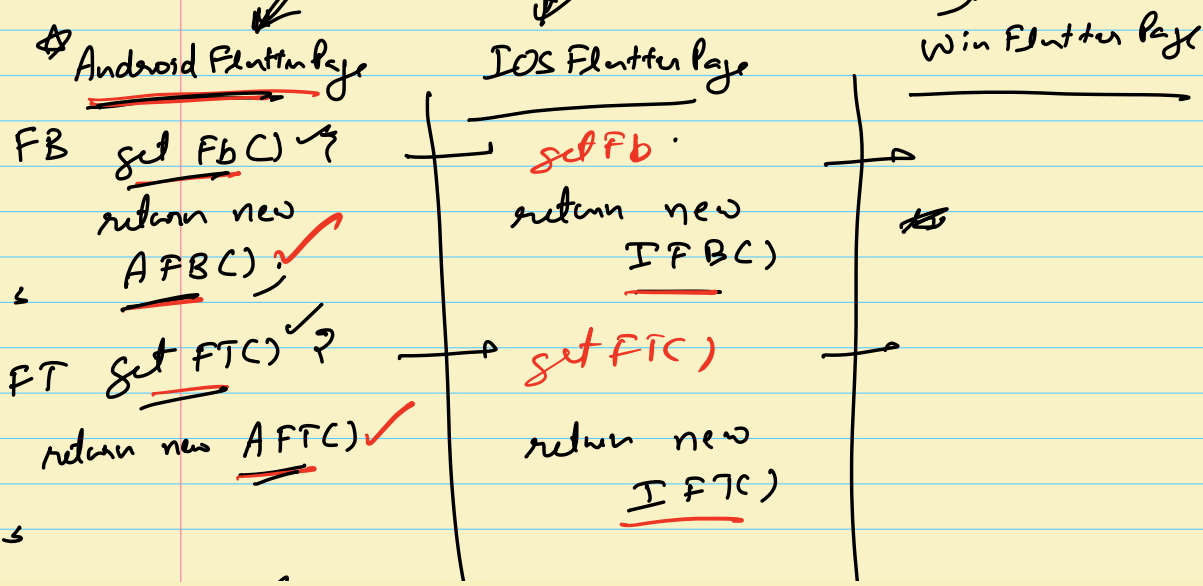
a FB setFlutterBtn();

a FT setFlutterTxt();

a FDD setFlutterDdl();


FB
AFB
IFB
WFB
FT
AFT
IFT
WFT
FDDL
AFDDL
IFDDL
WFDDL


abstract
FlutterPage
setRefreshRate()
setBgColor()
setLayout()
a FB & Fb()
a FT & FT()
a FDDL & FDDd()


Android FlutterPage
IOS FlutterPage
Win FlutterPage
FB setFb()
return new AFB()
FT setFT()
return new AFT()
setFb
return new IFB()
setFT()
return new IFT()

FDDd getFDDd()?
return new AFDDd();

return new IFDDd()

```
class AppStart {
  FlutterPage fp;
  main() {
    if (platform == "Android") {
      fp = new AndroidFlutterPage();
    } else if (platform == "IOS") {
      fp = new IOSFlutterPage();
    } else ——
    fp.setLayout();
  }
```

Factory Method implementation

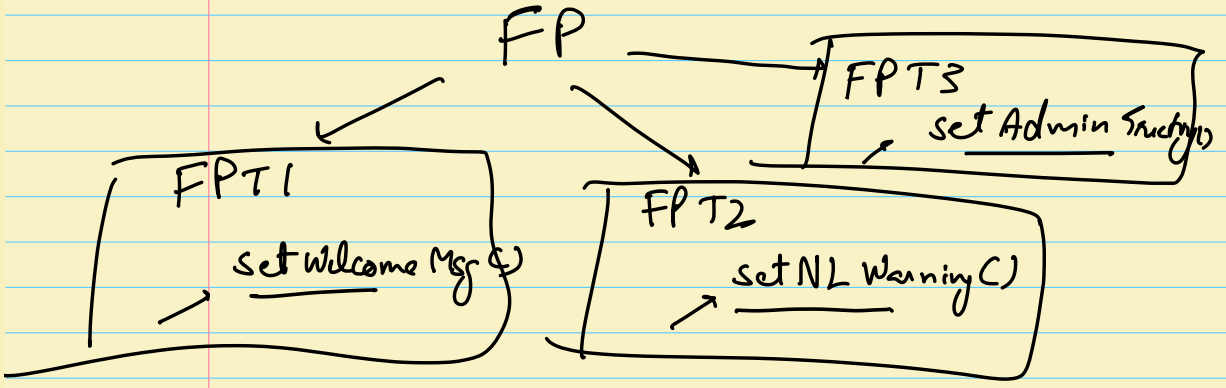
→ Factory Method design pattern depend on inheritance to provide concrete implementations of dependencies.

FP → FB, FT, FDd

AFP	IFP	WFP
AFB	IFB	WFB
AFT	IFT	WFT
AFddl	IFDdl	WFDdl

☆ Problem → We might want to inherit FlutterPage class for other reasons also, besides providing implementation of factory methods

example.



Problem → Class Explosion

Solution → Abstract Factory

→ In previous case we had two reasons to inherit FlutterPage

- To provide factory imp (3 classes)
- To provide different Master Pages for diff kind of users → NonLogin User
  - Login User
    - Admin

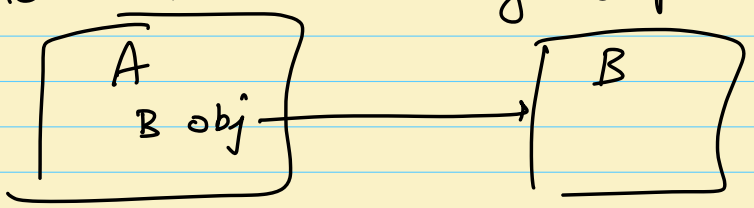
[3 cl]

Class reqd = 3 * 3 = 9

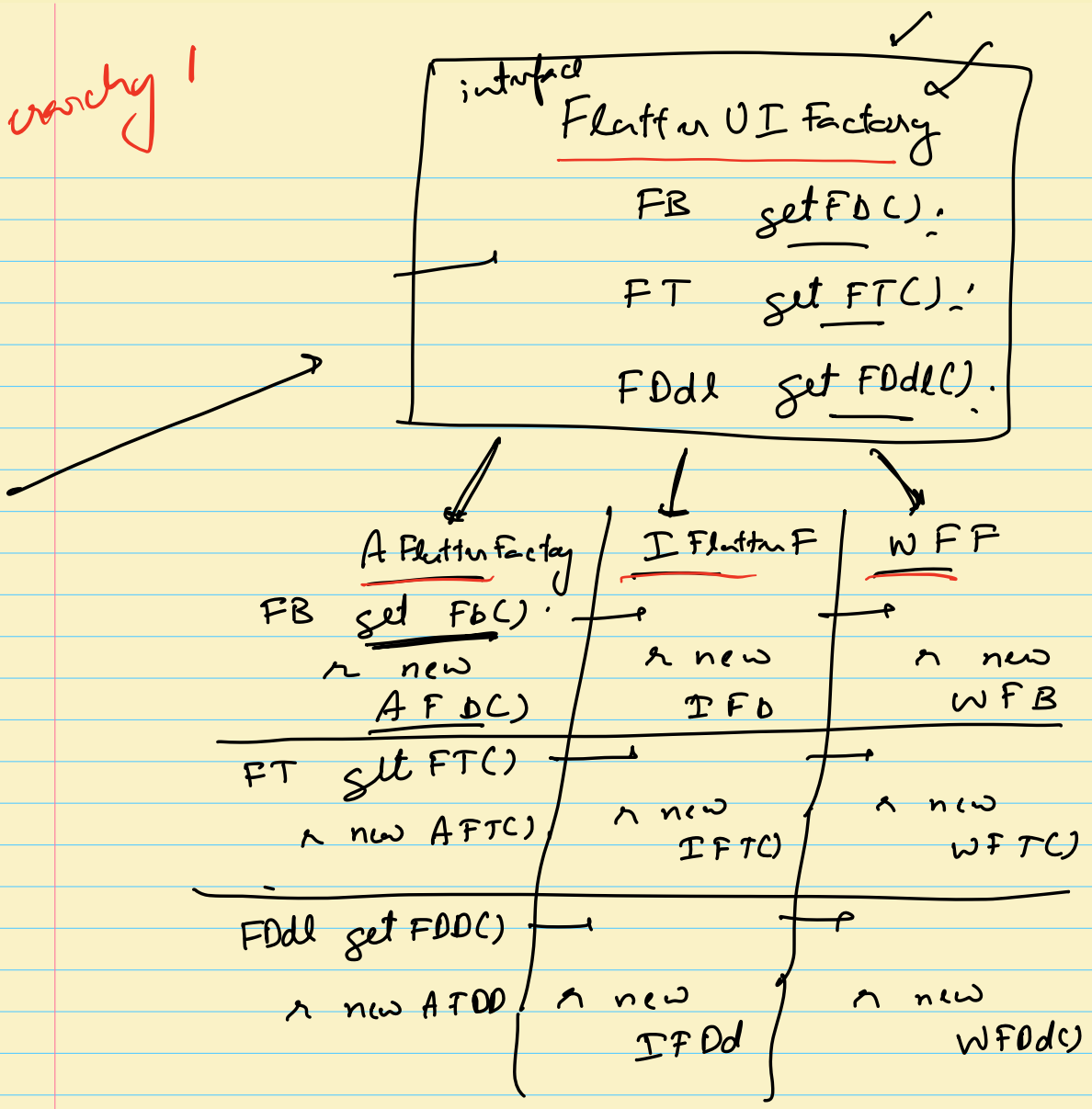
Solution → Create two separate hierarchies

1. For platforms
2. Diff. users

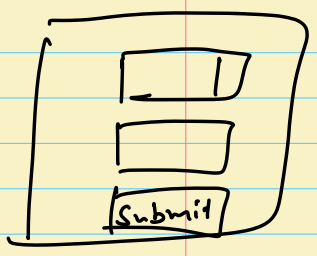
Combine the two using composition



Hierarchy !


interface
Flutter UI Factory
FB setFB():
FT setFT():
FDdl setFDdl().
AFlutterFactory
IFlutterF
WFF
FB setFb()
new AFD()
new IFD
new WFB
FT setFT()
new AFTC()
new IFTC()
new WFTC()
FDdl setFDD()
new AFDD()
new IFDd
new WFDd()

Hierarchy 2



Not abstract now

FlutterPage ∃ FButton b
FTextbox t
FDdl ddl

setRBC()
setBgC()
setLogout()

b = uifactory.setFBC();
b.setPos(-,-)
b.setText("Submit")
t = uifactory.setFTC();
t.setPos(-,-)
ddl = uifactory.setFDdlC();
ddl.setPos(-,-);

}

- pvt. FlutterUIfactory uifactory;
- public void setUIFactory(FlutterUIFactory ff)?

{ uifactory = ff; }

class AppStart {

FlutterPage fp = new FlutterPage();

main() {

if (platform == "Android") {

fp.setUIFactory(new AndroidUIFactory());

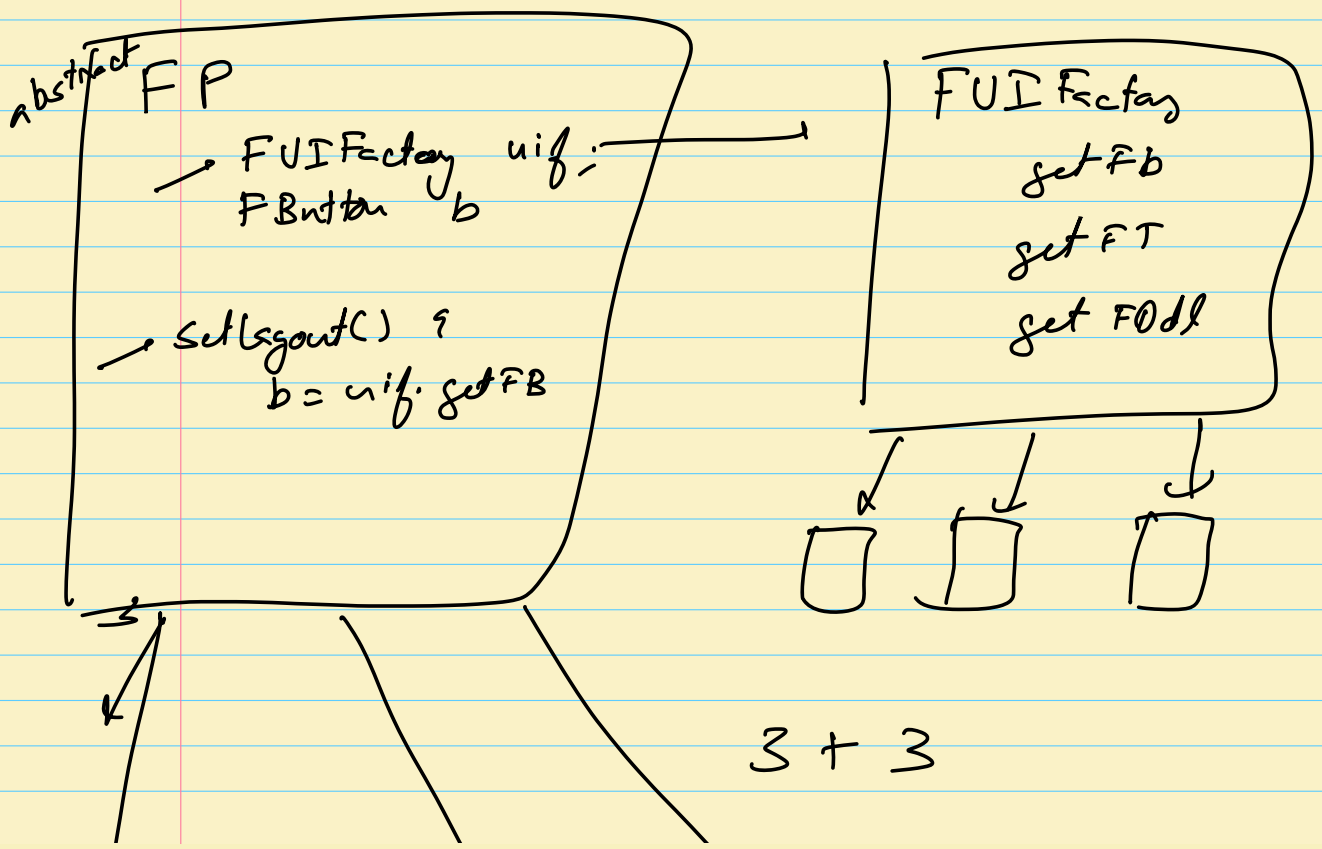
} if (platform == "win") {

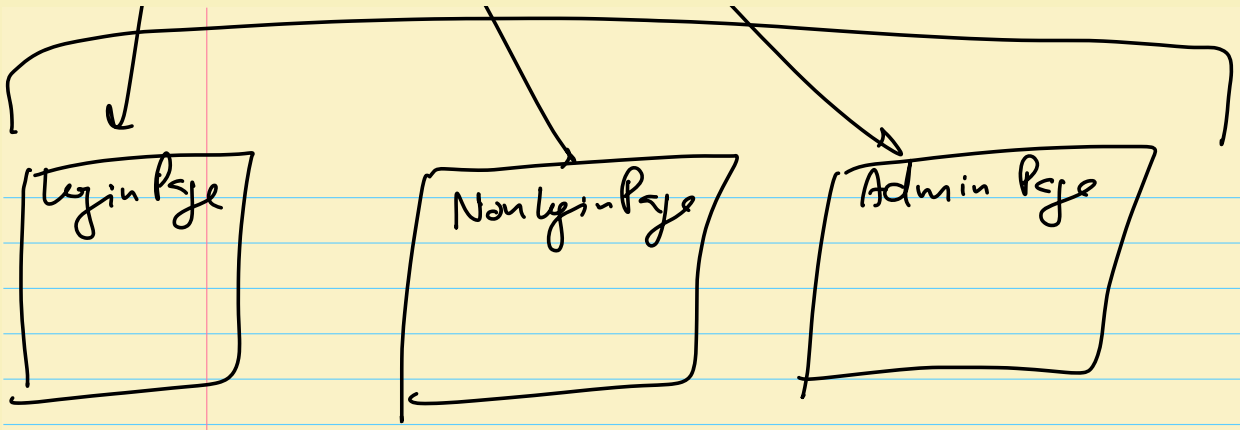
fp.setUIFactory(new WUIFactory());

}

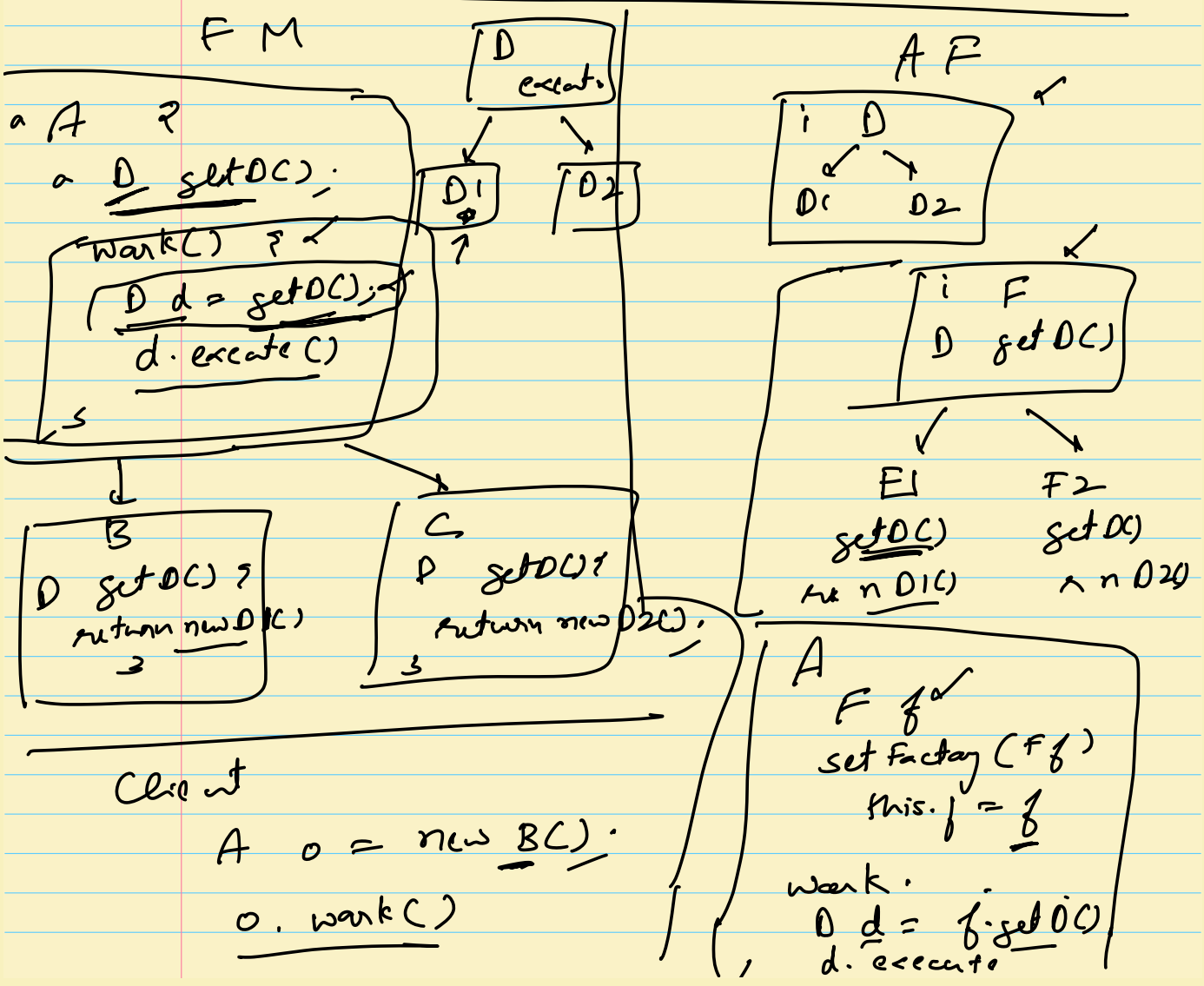
fp.setLayout();

}





Difference btw FM vs AF



Client

```
A o = new A()
o.setFactory(new F1C());
o.work()
```