Lect 5 : SOLID Design Principle

Page No.
Date 16 05 25

Problems in Software Development without Design Principles -

↳ Maintainability : Integrating new feature ~~shall~~ can be challenging, requiring expensive code modification & potentially introducing new bugs.

↳ Readability : The codebase is difficult for engineers to understand

↳ Too much bugs

SOLID Introduction

↳ Introduced by Robert C Martin in 2000 (though not needed, but credits required)

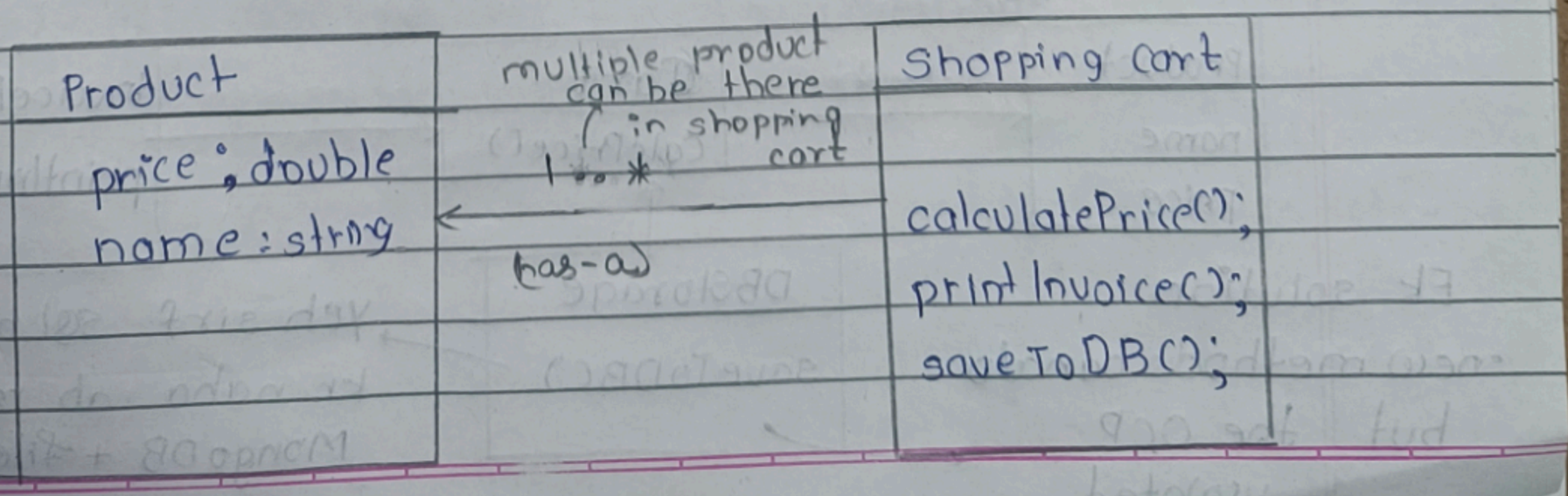
S : Single ~~Represent~~ Responsibility Principle (SRP)
O : Open / closed Principle (OCP)
L : Liskov substitution Principle (LSP)
I : Interface Segregation Principle (ISP)
D : Dependency Inversion Principle (DIP)

1) S : Single Responsibility Principle (SRP) → Ex : TV Remote

↳ A class should have only one reason to change.

↳ A class should do only one thing → single responsibility

i.e. a class should not do multiple operation.

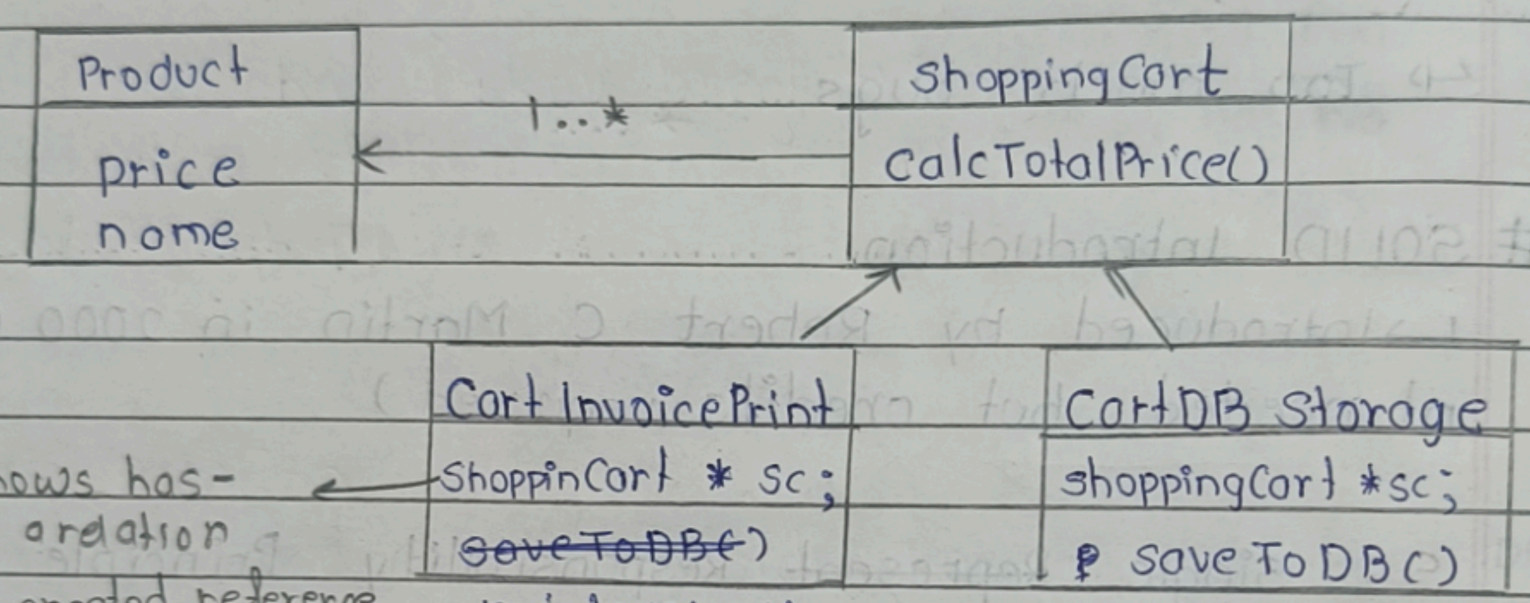


Confusion Abt SRP: Ek class ek hi kaam mtlb no. of methods ek hi hone chahiye aisa nhi but jitne bhi methods hai unhone milkr ekhi task krna chahiye.

Ab problem kya hai?
Agar mujhe DB/printInvoice/calculatePrice ko change krna hai toh class mein multiple reason ke wajah se ho rhe (we need ek hi reason se class mein changes ho).

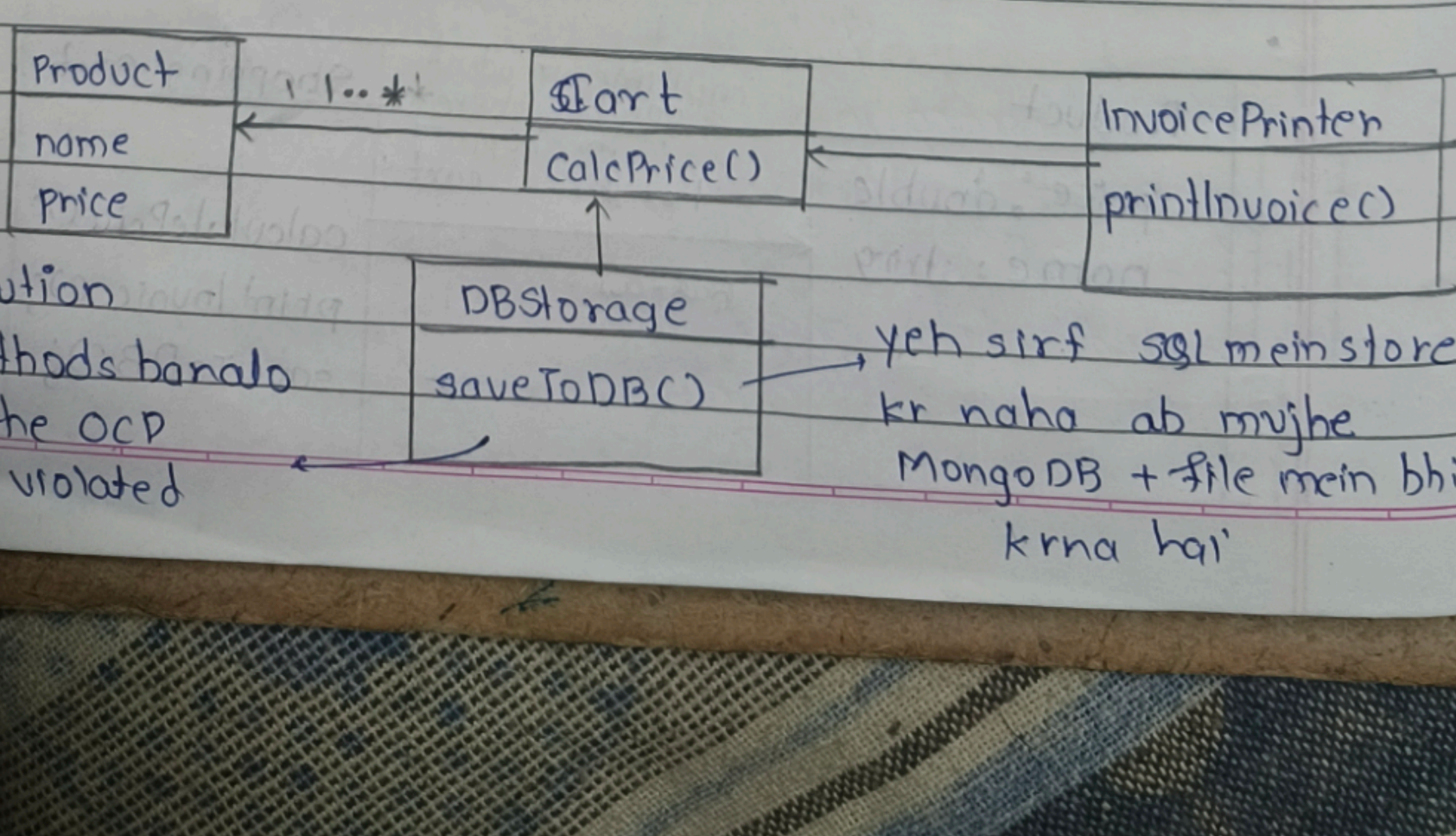
To resolve it we will use <u>composition</u>

Solution



code in VS

2) O : Open Close Principle
↳ A class should be open for <u>extension</u> ↳ new feature add krna (using new class)
but close for <u>modification</u> ⇓ purane class mein jaake code chnge krna



Ek solution
• new methods banalo
but the OCP violated

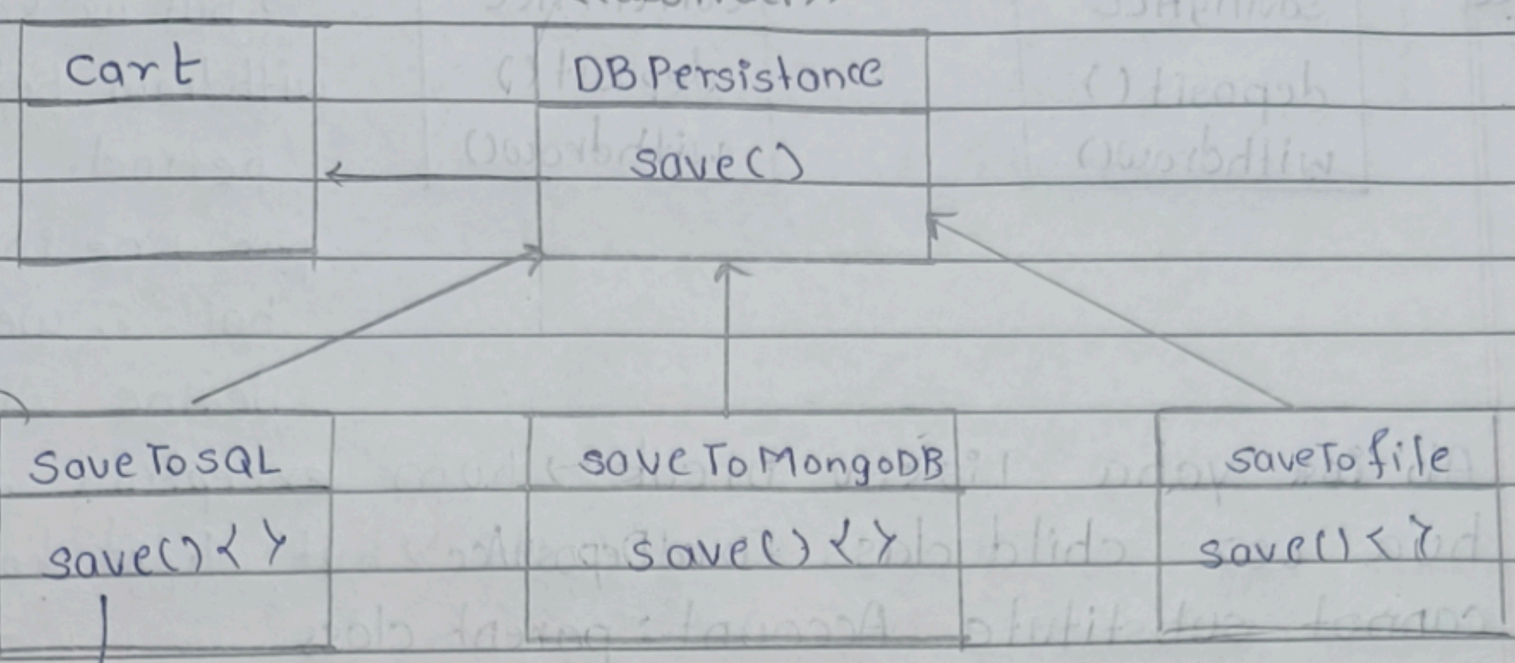
yeh sirf SQL mein store kr naha ab mujhe MongoDB + file mein bhi krna hai

Inheritance
ko woh
naa ki
To resol
Polymor
Solution
Cart
concrete class
SaveTo
save()
yha
hum
Ab aga
ek co
3) L : Lisk
→ subcl
classes
Yeh
jagah
na h
ho to
Jaha bhi mein
A ke methods
ko call kru uski
jgh hum esa
bhi likh sku.

ko woh humesha expand kre child naa ki narrow ho.

To resolve it we will use Abstraction, Inheritance, Polymorphism

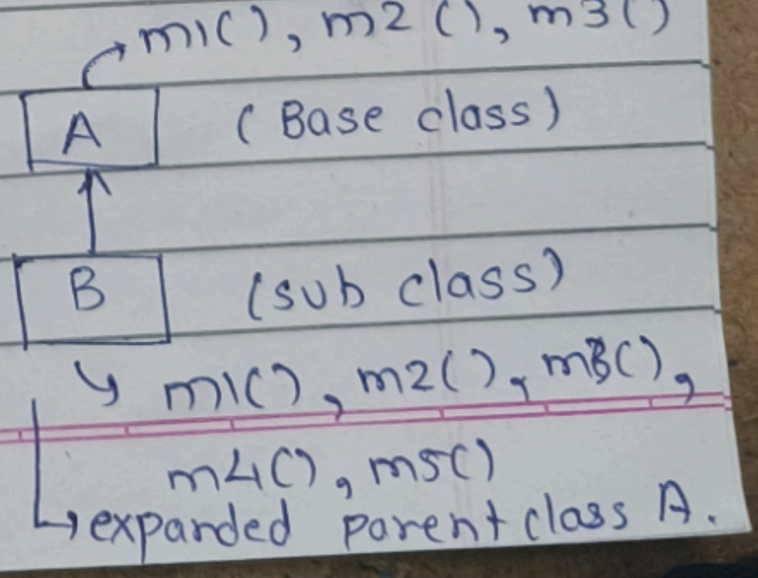
Solution



Ab agar aur kbi humme store krna hai toh bass ek concrete class bna dege & done.

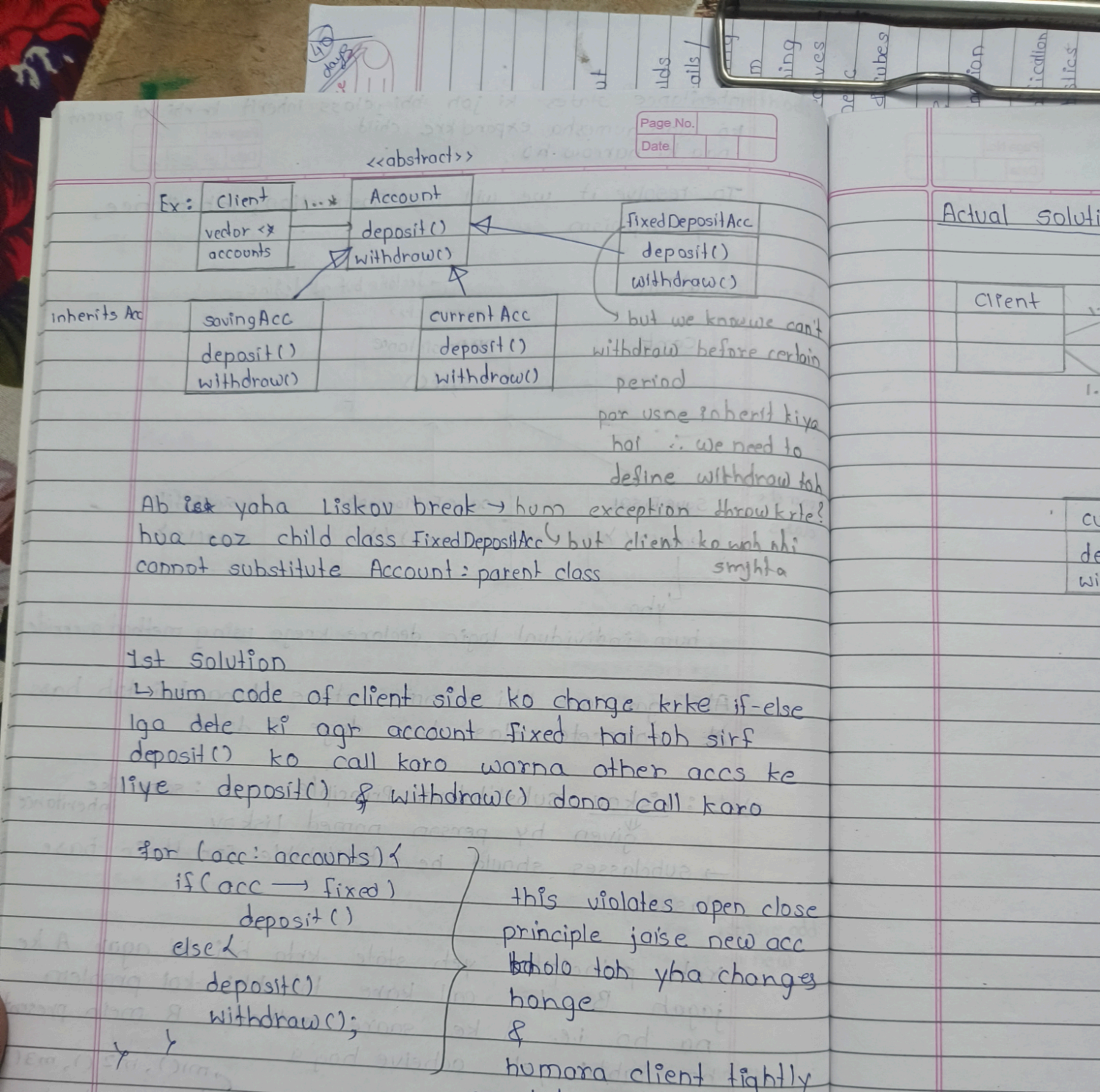
3) L : Liskov substitution Principle : same like inheritance
⇓ given by person named Liskov
→ subclasses should be substitutable for their base classes

Yeh principle yeh state krta hai ki agar A ke jagah B ko call kare toh bhi koi problem na ho i.e. A ke saare methods B mein present ho tabhi yeh acheive hoga



Jaha bhi mein A ke methods ko call kru uski jgh humesha B hi likh sku.

(client)


<<abstract>>
Ex:
Client
1..*
Account
vector <*
accounts
deposit()
withdraw()
FixedDepositAcc
deposit()
withdraw()
inherits Acc
savingAcc
deposit()
withdraw()
currentAcc
deposit()
withdraw()
but we know we can't withdraw before certain period
par usne inherit kiya hai ∴ we need to define withdraw toh
Ab yaha Liskov break → hum exception throw krte? hua coz child class FixedDepositAcc but client ko woh nhi cannot substitute Account: parent class smjhta
1st solution
↳ hum code of client side ko change krke if-else lga dete ki agr account fixed hai toh sirf deposit() ko call karo warna other accs ke liye deposit() & withdraw() dono call karo
for (acc: accounts){
if(acc → fixed)
deposit()
else{
deposit()
withdraw();
}}
this violates open close principle jaise new acc holo toh yha changes honge
&
humara client tightly coupled ho gaya in classes se naa ki abstract class se coz usse pata chahiye ab ki uska acc konse type ka hai
Actual soluti
Client

Page No.
Date

Actual solution