Low Level Design 1) SOLID Principles Advantages Avoid duplicate code > Flexible Software Easy to maintain sunderstand 2 Redue Complexity 5 -> Single Responsibility A class should how only 1 reason to change (means a class should have only one responsibility)

class Markon [] String names String colors interryeur in a int price; public Marker (String name) String color , int year, int prie)

[this name = name; this color = colori this year = years this price = price; Class Invoice Private Marker marker; // Invoice Thas a Marker private int Quantity; public Invoice (Marker marker sint quantity) this marker = markers this quantity = quantity; public void int calculate Total () ? int price = (marker.price + this guartity) return pricos /// vijeta ///

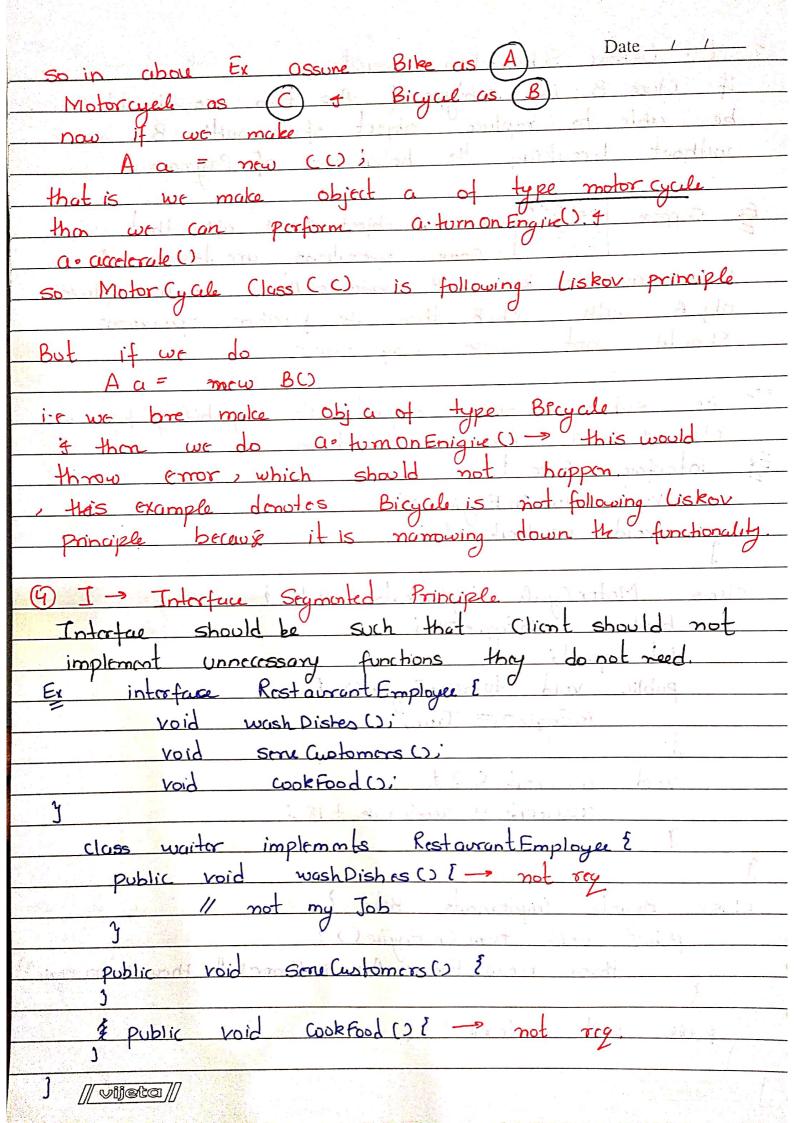
해 있다. 그런 마이트 보고 있는 것이 되었다. 그 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은 사람들은	
	Date/_ /
public void printInvoice () [// print invoice	a gura a
1/ print invoice	a Administration
17 D when the Miles	who house
public boid Sove ToDBC) & building	at parts
public boid Sove ToDBC)	*
// Sau data mile	morning the Fig.
J spends of mour f year and hundre	south A me
=> now suppose in calculate Total we want to GDP & discount than the logic will change	o add
=> now suppose the logic will change	right?!
GDP F BECOOK O	point?
=> similarly if we want to change how the is printed then also logic is changed	print Invoice()
is printed then also logic is changed	o dair
=> Same with Save To DB() logic is changed	if we went
some other thing.	
=> so in total we have 3 reasons to the Single Responsibility (s) says a class sho	inge, But
Single Responsibilty (S) suys a class sho	ould have
only one I reason to Change.	
N. T. d	
Now Invoice class is not following Single Re so to correct it make diff classes for	sponsible
function such that Each Class is ha	win- cont.
1 responsibility to Change	d d
	1 h
Ex class Invoice {	4
private Markor markori	ř
private int quantity	i who
Dolla Invola Marker merker 2 int auc	onhity)
[constructor.	A MEY
Public int calculate Total () ? Logic	1.
//_vijeta//_)	

	Date 7 / Date
	Class Invoice Duta Accessinger
	Invoice invoices
	public Invoice Duta Accesslayer (Invoice invoice) { this invoice = invoice i J
	this invoice = invoice is I
	Coldmaniford wastern
7	Public void SONTODB () 1
	Milogic for Save into DB
	Total Total State Total
	Class Invoice Printer 1
	private Invoice invoices
N. Control of the Con	Public Travoice Printer (Invoice invoice)
	this invoice = invoice > 1
	public void print() [
70	4 print invoile
3	- Justin and the state of the s
	> Now we can see each class has single responsibility
Sign of the same o	
	to Change
	2 So it is usabil like if T want to shance
	So it is useful like if I want to change calculation logic I need only Invoice class to
d'an	Change is the second of the se
	Change in the state of the stat
2	0 → Open / Close Principle
	Open for Extension But Close for Modification
	> Now about we have a class Invoice Dutu Acess Layor
9	
	sou To File also so what can we do? We
9	can read describe to the Class.
•	Class Invoice Data Access layer ?
	// Constructor
	Public void Squ ToDB 2 3
	[vijeta] public boid sau to File []

=> But is it following (0), Ans is No. Date
as it says that a class is open for Extensions
means (it come be extended Cor Inherited)
but cannot be modified (mouns cannot add
more functionality).
=> So What we can do is make another class
that Extends Invoice Duta Acess layer to some
to File.
to the second of
=> Solution :
interface Invoice Data Acess Layer & marie invoice);
public void sale (Invoice invoice)
1 Whoma had while
Il sound from six
D Class Artubase Invoice DAL implements Invoice Data Acrosloyer
5
Management Que Overnder 1 santo Mana me mos me will co
Public void soul (Invoice invoice)
1 // Sau to DB J
2) I would at topal I to add Dison is to see
and the district the state of t
@ Class File Invoice DA1 implements Invoice Date Acres Layor
1 @ Overnde
public void som (Invoice invoice)
Sale of file of the sale of th
The season of the season of the season with the
=> so in future if another function comes like
Sau To xyz we we simply extend it & write
Logic.
toyli.

//vijeta//

(8) L-> Liskou Substitution Principle Date_/_
if Class B is subtype of A, then we should
be able to replace object of A with B
with it has loss the behaviour of Program
without breaking the behaviour of Program
Ex Suppose ObjA=A we have object A & after that
Some operations are done on A
Now as we know B extends A so if we replace
obj A with objB than the further operations
Should not cause any error.
=> i.c subclass should extend the capability of parent
class of not narrow it down.
Ex interface Bike [
void turn on Engire ():
void accelerate ();
class Motor Cyale implements Bike 1
boolean is Engire ON;
int speed;
public void tumona Engire O. [is Engire On True,
is Engire On= True, is a ration of the
La transport of your field
void accelarate () [() limited
ciccelerale = acrolorale + 183
I de l'action de la material est de material este de la material d
James James Committee Comm
class Bicycle implements Bike 2
public void torn On Engire () [
public void turn On Engile () [throw new Error Assortion Error (" those is no ongin's
Public void andronk () (
1 // northeaten ///



가게 좋아 하는 게 마리마이 이 마음이 들어가 되었다. 이 아이에 아이에 아이는 이 아이에 가는 사람이 가는 것이 되었다. 그리고 함께 보다 생각하였다. [1882년] 1982년 - 1982년
In above Example we can see Date
that weiter has to implement all functions of
Interface, But wart does a warter works Food?
or clean Dishes? No right? so why to
implement unnecessary functions. this is what
(I) says in solid principle and had stone
- 18 2 Wird
=> Solution: is to breakDown Interfaces into seperate
interfale more servered thereford broaded I down more silded
interface Waiter Interface ?
void Serve Customos (),
void take Order ().
J
interfae Chef Interfae ?
void cook Food ();
void decideMonu();
J
Public Waiter implements Waiter Interface []
6 D -> Dependency Inversion Principle
-> Class should depend on Interfaces rather than
Concrete Classes
En Interface Mouse Interface keybourd
Chieses. Chieses.
wired Mouse Bluetoth mare. Wired Kaybarrd Bluetooth.
Class Mac Book {
private final Wired key Boad Responds
private final Wired key Board keyboards: private final wired Mour mouses:
public MacBook U L
Reubourd = new Wiredkarboard (2) I this is not
Reyboard = new Wired ky Board (); I this is not move = mew Wired Move (); preferred.
I // caz it might happen we change keybourd to
3 Bluetooth key board, but we will not be able

=>	so it is adviced to do with constructor Da	ite
	njecton. sich alle franchene al auf men	- retratect
-	Todas of Sid working specimens	to see the see
	Private final Keyboard keyboards	(J)
	Private final keyboard keyboard: Private final Move move: Puss Like this Public MacBook (kayboard keyboard: Move move 1 this keyboard = keyboard: this is this move = move: It is dyna	no de de la constante de la co
- i	this keyboard = keyboard if this is	not burdeodod
	This moux - mouses It is eggin	J. W.
		Al Nebertier
172		hull Live
	L. I saturdad rodines of an exempt - In	AND WISH .
10-87		