Unit - IV :— object Oriented Design process and Design Axioms, Corollaries, Design Patterns, Designing classes: Object Oriented Design Philosophy, UML Object Constraint Language, Designing classes: The Process, Class visibility, Refining Attributes, Designing Methods and Protocols, Packages & Managing classes, View layer: Designing Interface objects, Designing View layer classes, Macro & Micro Level Interface Design Process.

✱ **The OOD Process & Design Axioms :—** After the analysis phase the conceptual model is developed further into an object oriented Model, using obj. Oriented design.

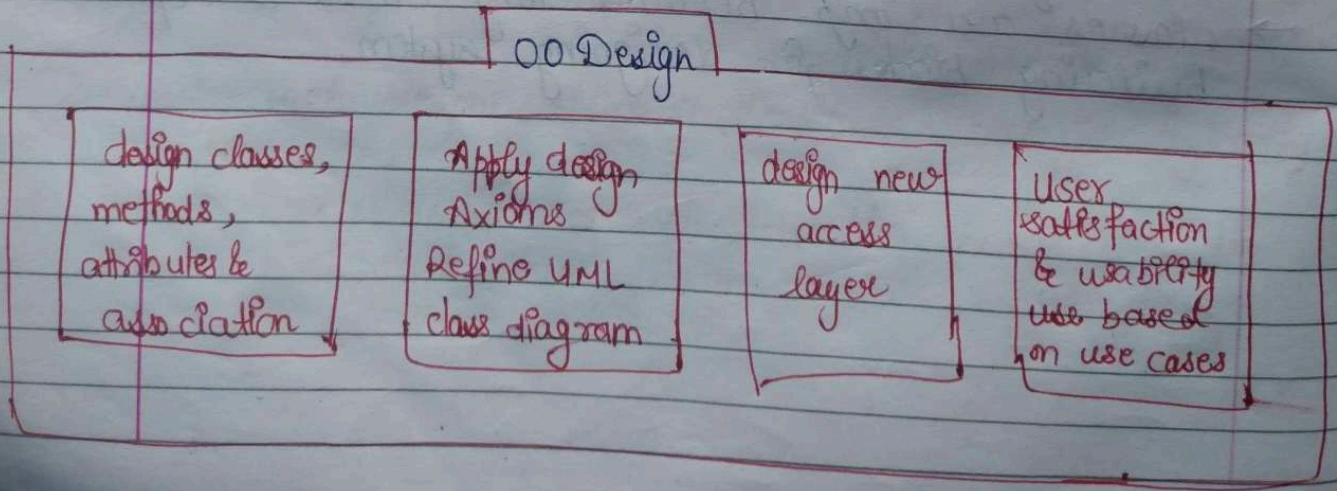
→ Attributes, Methods, & association are identify in analysis phase.

→ New classes will be introduce to store intermidate result.

→ During the design phase, we elaborate the model into logical entities.

→ The Goal here is to design classes that we need to implement the system.

→ In design phase we focus on to view & access classes (how to maintain the info.)



Axioms:- is a fundamental truth that always is observed to be valid & for which there is no counter Eg:- or exception.

⇒ Axioms are of 2 types :-

① Axiom 1:- The independence Axiom :- It maintain the independence of components.
In this each component must satisfy its requirement without affecting other component. In this each component perform its task independently.

② Axiom 2:- The info. Axiom :- Minimize the info. content of the design. It mainly concerned with simplicity. In OOS to minimize complexity, use inheritance & system build in classes.

③ Corollaries :- A corollary is a proposition that follows from an axiom or from another proposition that has already been proven.

diagram → back.

① Corollary 1:- uncoupled design with less info. content:-
→ main goal here is to minimize the objects.
→ min. info. need be passed b/w objects.

② Corollary 2:- Single purpose :- → Each class must have a single, clearly defined purpose.
→ Each class must be provide only one service.
→ Each method should be of moderate size, no more than page.

③ Corollary 3:- Large no. of simpler classes, Reusability:-

→ keep classes simple allows reusability.
→ There are benefit in having large no. of simpler classes, because the chance of reusing smaller classes in other project is high.

④ Corollary 4:- Strong Mapping:-
→ Strong mapping link classes identified during analysis & class designed during design phase.
→ The analyst identifies object's type & inheritance & think about event that change the state of object.

⑤ Corollary 5:- Standardization:-
→ To reuse classes, we must have good understanding of classes.
→ They must be easily searched, based on user's criteria.

⑥ Corollary 6:- designing with inheritance:-
→ When we implement class, we need to determine ancestors, what attribute it will have & what msg it will understand.
→ Ideally, one has to choose inheritance to minimize the amount of program instruction.

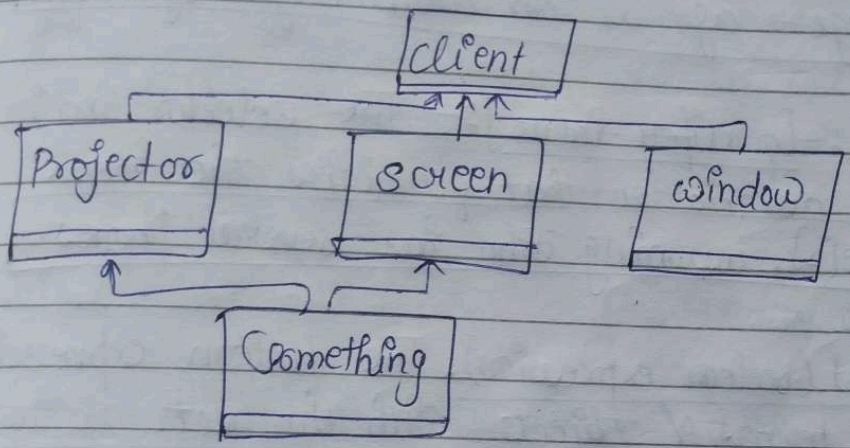
⊛ design Patterns:- → It is general repeatable soln. to a commonly occuring problem in s/w design.
→ These are the devices that allow system to share knowledge about their design.
→ Documenting pattern is one way that allow reuse & possibly sharing info.ⁿ.

DATE
PAGE

↳ Classes :- There can be any no. of classes but atleast 4 or more classes are required.

→ Adv. :- Adding interface layer in module.

→ Disadv. :- We loose some func^n in lower level of classes.



(*) Designing classes :- once we are ready to identify our class & interaction blw other class then we are ready to Design.

→ OOD requires that you think in terms of classes. In design app^n. Classes work together to provide functionality we desire.

→ Our goal is we reuse class libries.

→ First we make a set of class, each class have an expertise & each expertise work together in useful ways.

(A) OOD Philosophy :- PDF

(B) UML object Constraint Language :- → UML is a graphical lang. with set of rules & semantics.

→ The rules & semantics of UML are expressed in English in form known as object constraint Lag.

→ Obj. Constraint Lang. (OCL) is a specification lang. that uses simple logic for specifying the properties of the system.

(*) The expression are meant to work on sets of values when applicable:—

(1) Item selector :— The selector is the name of an attribute in the item. The result is value of attribute.
Eg:- John.age → age is an attribute of obj John.

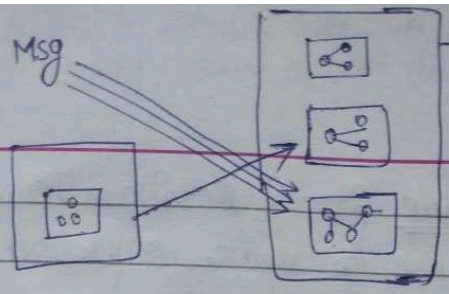
(2) Item selector :-[qualifer-Value]:— The selector indicates a qualified association that qualifies the item.
Eg:- John.phone[2], assuming John has several phones.

(3) set:- select (boolean expression). The boolean expression is written in terms of objects with the set.
Eg:- Company.employ → salary > 3000
This represent employees with salary over $30,000

(C) Designing classes: The Process:—

(1) Apply design Axioms to design classes, their attributes, methods, associations, structure & protocols.
(2) Refine & Complete state UML class diag by adding details to that diag.
(3) Refine attributes.
(4) design methods & protocols by utilizing UML activity diagram to represent method's algo.
(5) Refine association b/w classes (if require)
(6) Refine the class hierarchy & design with inheritence.
(7) Iterate & Refine.

(D) Class visibility:— designing well defined Public, Private and Protected Protocol.



DATE
PAGE

⊙ Private Protocol :- (visibility) : A class having set of methods that it uses internally, message to itself. In this msg that normally should not be send from other obj.
The class itself uses a method.

⊙ Public Protocol :- (visibility) :- accesible by all classes.

⊙ Protected Protocol : (Visibility) :- subclass can also use the methods in addition to itself.

* ENCAPSULATION LEAKAGE :- lack of well defined protocol can manifest itself as encapsulation Leakage.
⊙ The Problem of Encapsulation Leakage arives when class internal implementation are disclosed through interface.

⊙ Refining Attributes :- → The main goal of this activity is to refine existing attributes or add attributes that can evaluate system into implementation.
→ In analysis phase, the name of attribute was sufficient, however in designing phase detailed info. is added to the modes.
Types of attributes :-
⊙ Single value attributes :- → has only 1 value.
→ when chage state of obj changes, the changes are reflected in value of attributes.
→ Attribute represent state of obj.
→ Eg :- Name, address, salary.

DATE
PAGE

⊙ Multi-value attribute :— having more values.

Eg:→ A person may have one or more bank accounts.
→ joint account.
→ so an account has zero to many instance connection to person.

(F) Designing Methods & Protocols :→ We already discussed that obj. oriented programming revolves around the concept of objects & classes.
→ Classes are the blueprints & objects are the real world entity.
→ A class can provide several types of methods:—
① Constructor :— the obj. having same no as that of classname.
② Destructor :— The method destroy's the instance, or they destroys the obj.'s that are of no need.
③ Conversion Method :— used to convert the values from one unit of measure to another, it basically involved typecasting & type conversion.
④ Copy Method :— that copies the contents of one instance to another instance.
⑤ Attribute set :— They are used to set the values of one or more attributes or set name.
⑥ Attribute get :— that are used to return values of one or more attribute.
⑦ I/O method :— The method that provides or receive data to or from a device.
⑧ Domain specific :— The method specific to appl".

(G) Packages & Managing Class :— PDF.

(viii) User Interface design:- user की need को satisfy करना The main goal of UI is to display & obtain needed info. in an accessible, efficient manner.
→ A well defined UI has visual appeal that motivates users to use appln.

DATE
PAGE

→ login page

(24) View Layer:- designing interface objects.

- After the analysis is complete, we can start designing the user interface for the objects & determining how these objects are presented to the user.
- The design of software interface, max than anything else, how user interacts & ∴ experince the appln.
- It is imp. for a design to provide users the info. they need & clearly tell them how to successfully complete a task.

1. View layer obj. responsible for 2 major aspects of appln:-

a. Input:- responding to user interaction. The UI must be designed to translate an action by the user.

b. Output:- displaying or printing business objects. This layer must ft. the best possible of business obj. for the user.

2. The process of designing new layer:-

Three layer arch. used for softw. development namely (1) Access layer, (2) View layer & user-interface/ business layer.

(25) Designing View Layer classes:-

(1) The process of designing view layer classes is divided into 4 major activities:-

a. Macro level UI design process:- uper-case से, Identifying or use cases like ATM. view layer objects.

d. Micro " " " activities:-

(i) designing the view layer objects by applying design axioms & corollaries.

(ii) Prototyping the view layer interface.

3. Testing usability & user satisfaction.

4. Refining & Iteracting the design.

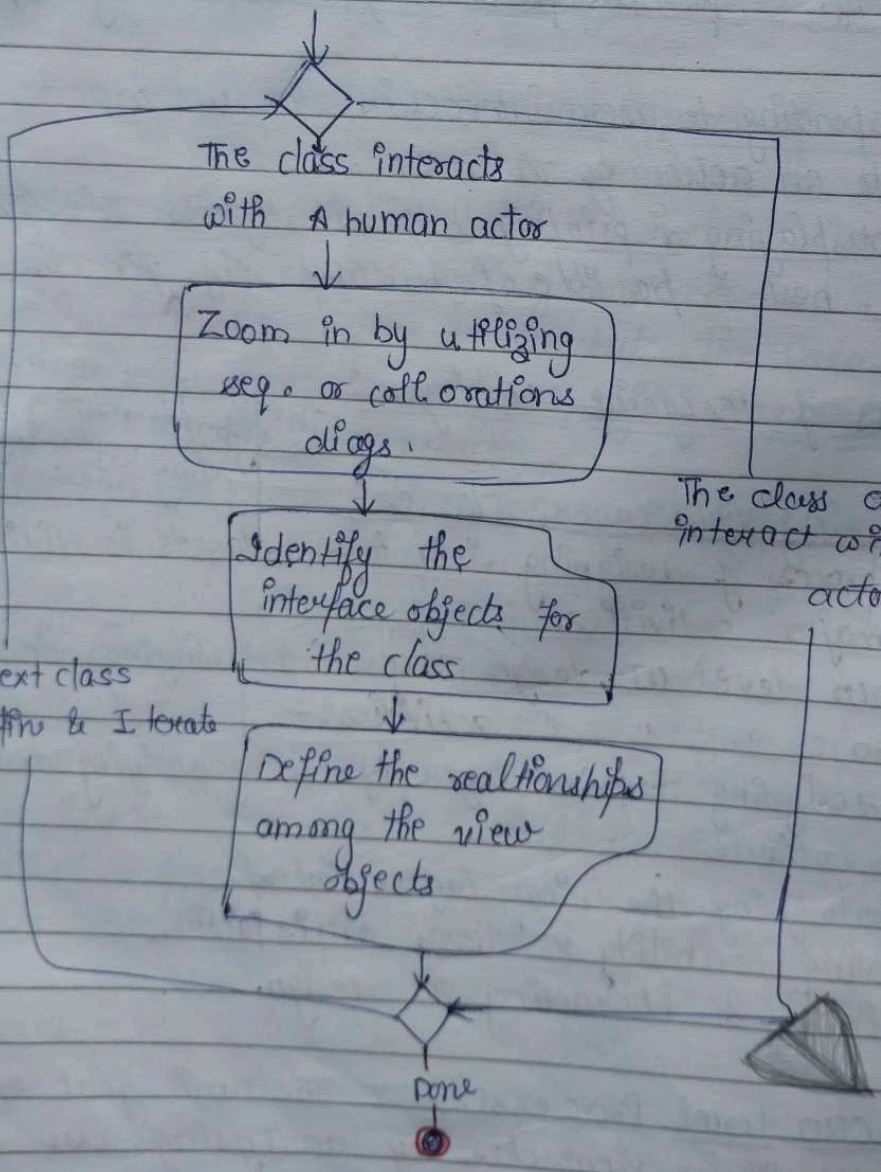
(A) MACRO level Processes:- The main goal of this level is identifying view class by analysing use cases.

→ The interface obj. handles all comm. with actor but processes no business rules or obj. storage activities.

→ The view layer macro process consists of 2 steps:—

1. Determine if class interacts with a human actor.
   a. Identify view (interface) obj.'s for class.
   b. Define relationships among the view (interface) obj.'s.
2. Iterate & refine.

⇒ Adv. user को or user friendly interface दे पा रहे हैं।
(by errors को minimize करके



(A) UI Design:— To view UI design as a creativity process is constituents are the following:—
(a) A curious & Imaginative mind.
(b) A broad background & fundamental knowledge of existing tools & methods
(c) deal with uncertainty & ambiguity
(d) site or appln should not be crashed.

(B) MICRO Level Process:— internal structure पर work करती है।
The micro level process of designing view objects has 2 steps.
1. for every interface object identified in macro UI design process, apply micro - level UI design rules & corollaries to develop the UI.
2. Iterate & Refine.

(1) UI Design Rule I:— (Applⁿ of corollary-2)
→ Making the interface simple. The following factors should be considered while evaluating the impact.:—
(i) Adding more & more features to the applⁿ will make it complex & its performance, stability, maintenance & support costs will be affected.
(ii) It is harder to fix a design problem after release of product because user may adapt or even become dependent on a older applⁿ or design.
(iii) Make applⁿ as simple as u can. so that user could understand that.
(iv) Add small-small extensions to the applⁿ code do not necessarily have a proportional effect in a UI.

(2) UI Design Rule II:— Making the interface Transparent & Natural (Applⁿ of corollary-4).
→ It implies that there should be strong mapping b/w user's view of doing things & UI classes.
→ A goal of UI design is to make user interaction with computer as simple & natural as possible.

⊛ UI Design Rule III :—

Allowing users to be in control of s/w.

- ∝ Making the inface forgiving. (forgot password, easy but security को खतरा नहीं करना चाहिए)
- ∝ " " " Visual. (GIF, vedos, images भी होनी चाहिए)
- ∝ Provide immediate feedback
- ∝ Avoid modes.
- ∝ Make the interface consistent. (all the pages should take equal similarly speed)

⊛ Purpose of View Layer Interface :—

UI can employ one or more windows. Windows commonly are used for the following purposes :—

⇒ **Forms & data entry windows :—** provide access to data that users can retrieve display & change in the appl^n.

⇒ **Dialog Boxes :—** user intract with appl^n. typical feature is ok button.

⇒ **Appl^n windows (main windows) :—** It contains the entire appl^n with which users can interact.