OOAD

1st class

I. Introduction to OML

Importance of modelling.

Principle of modelling.

Object Oriented modelling.

Conceptual model of the OML

Architecture and s/w development life Cycle.

II Basis Structural modelling (mid-1)

1) Classes, relationships & Common mechanisms & diagrams

2) Advance structural modelling

Advance Classes; Advance relationships, interfaces, types and rolles, packages.

III Class & Object diagram

Terms, Concepts, modelling techniques for class and object diagram.

IV Basis Behavioral modelling

Interactions, Interactions diagram, Use cases, Use Case diagram, Activity diagram.

Scanned by CamScanner

V Advanced Behavioral Modelling

Events, Signals, State Machines, process & ~~other~~ threads, time and Space, State chat diagrams.

VI Architecture Modelling

Component, Deployement, Component diagram & Deployement Diagram, Case Study.

Author Name

Greedy booch
Hands erik.
} text book 90%.

Why we use Objects ?

To work on project and Essentially applications.

Project: Any real time applications.

OOAD
1) Requirement Gathering
2) planning
3) Analysis
4) Design ⟹ Visual Representation Diagram → Symbol use.
5) Implementation
6) Coding
7) Testing

Scanned by CamScanner

Objected Oriented Analysis & pattern :— 2nd class

Four accepts.

Identity, Behavior, Inheritance, polymophism

> object is a real time entity.
> Blue print of object is called class.
> there Extracting objects one to another object.
> polymorphism an object can have many objects.

Object Oriented Development :—

Model : Simplification of reality. (or) Blue print of reality.

we have 3 types of models

→ class Model
→ State Model
→ Interaction Model

class Model : which will be having static Structure in nature.

→ class diagram.
→ Object diagram.

State Model : It undergo different levels over time.

→ state chat diagram
→ activity diagram

Interaction Model : Having intraction b/w users & developes.

→ UseCase diagram
→ Sequence diagram
→ Collaboration diagram.

Scanned by CamScanner

Why Model ?

→ Better Understanding

→ Visualizing the System Architecture.

→ For Specifying the Structure.

→ Communication → is a major part.

Why modelling ?

1) We build the Models to Communicate with Structure and behavior of the system.

2) We Build the Models to Visualize & Control System Architecture.

3) We Build the Models to better understanding object of the system.

4) we Build the Models to manages the risk.

Importance of Modelling :—

1) Models help's us to Vidulaize a system as it is or as we want.

2) Models helps us to Specifies the structual (or) behavior ~~System~~ structure.

3) Models gives us a template that guides us in Constructing a system.

4) Models Documents the Decisions what we made.

Scanned by CamScanner

Principle of Modelling :—

1) The choice of model that has performed influence how the problem is attacked and how the solution is shaped.

2) Every model may be expressed at different levels of abstraction.

3) Best models are connected to reality.

4) No single model is sufficient for constructing the system.

UML :—

Unified Modeling language but not programming.

1) Visualizing
2) Specifying
3) Constructing
4) Documenting

→ Constructing nothing but engineering

two type of engg 1) forward engg ♀ – code

2) Reverse engg code – ♀ ↓ Diagram.

→ Documenting nothing but respresent prototypes, decisions, plan, design, ideas, diagrams, artifacts.

Scanned by CamScanner

UML Uses :—

1) Banking Sectors
2) Enteprises [nothing but Companies]
3) Financial Application
4) Medical S/w
5) CAD applications
6) Scientific applications
7) Financial Applications.

History of UML :—

Uml is Developed by Grady Booch, Ivar Jackobson, and Ranbaugh.

→ Grady Booch Method Concentate an Constructing of system
→ Ivar Jackobson Method OOSE (Object Oriented S/w engg)
→ Ranbaugh Method Concentrate an OMT (Object modeling technique).

1991 → UML is introduce.
1994 → 1st version of UML is realesed.
1970 → They Stated resesch.

Scanned by CamScanner

→ UML work started GRE RD Labarities

→ In 1994 0.8 version is realsed but this version not support object Oriented Concept.

→ later in 1996 0.9 version was released in 1997 UML 1.0 was released.

→ In 2004 UML 2.0 versions was released.

Raya, National rose → S/w of UML.

3rd class

Conceptual Model UML

I. Things :-

Structural things :- they are 7 kinds of ST.

This are the nouns of UML and discribes the Static view part of a system. which means the Structural representation of things is Specified.

1) class :

A class is a set of objects that can share common attributes and Operations.

class is resp by.

class name
Attribute
Operations

Eg Bank Acc

Attribute : B_id, B_IFSC code.

Operation : Deposit ()
withdraw ()
transfer ()

Scanned by CamScanner

Conceptual model of UML

Basis Building Blocks

Things

1) Structural things (static view)
   a) class
   b) Interface
   c) Collaboration
   d) Usecase
   e) Component
   f) Node
   g) Active classes.
2) Behavioral things
   a) Interaction
   b) State Machine
3) Grouping things
   a) Package
4) Annotational things
   a) Note

Relationships

1) Dependency
2) Association (Aggregation)
3) Generalization)
4) Realization.

Diagrams

1) class diagrams
2) Object diagrams
3) Usecase diagrams.
4) Sequence diagrams } Intraction diagrams [dynamic view]
5) Collaboration diagrams }
6) State chat diagrams } Dynamic view
7) Activity diagrams }
8) Component diagrams } Static view
9) Deployment diagrams. }

kinds of
- class – actors, signals, utilities
- Activity class – processes & threads
- Component – applications, documents, files, libraries, pages, tables.

Rules

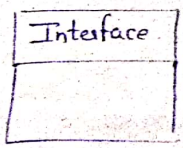
1) Name
2) Scope
3) Visibility
4) Integrity
5) Execution

Common Mechanisms

1) Specification
2) Adornments
3) Common Division
   a) Classes & Objects
   b) Component & Interface
4) Extensible Mechanism
   a) Stereotypes
   b) Tagged Values
   c) Constraints.

2) Interface :-

Interface is a set of Operations that are Used to Specify the Service of a class. Interface resp by



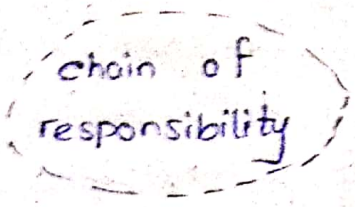
1 Spelling

Scanned by CamScanner

3) Collaboration :-

It is an interaction b/w objects that Combine work together to provide a Co-operative behaviour of a system. It is resp by

(or)

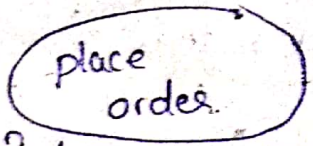


← ellispe

4) Usecase :-

It is provides the behaviour of an Actor. and result has to be obtained it is reps by

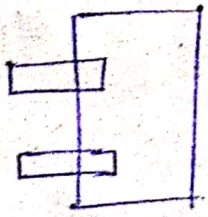
5) Activity class :-



The class whose objects own 1 or more processes or threads.

6) Component :-

It is a physical (or) replaceable part of the system that provides Set of interfaces. it is resp as.



Eg :- database, Pass Book.

retangle with tab

7) Node :-

A Node is a physical part of a System that Exist at runtime envirinment. it is resp with cube.

Scanned by CamScanner

Cube

Eg printer in ATM Machine.

II. Behavioral things :—

Behavioural things respresents the functionality of a object. & Specify dynamic view of an System.

1) Interaction :—

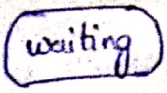
display →

→ Triangle with Solid arrow (symbol) ➡

A set of msgs Exchangs among the objects into acheive the Specific task. is called interaction.

2) State Machine :—

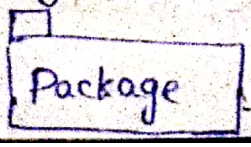
It is a Sequence of states that represents the life time of an object in response to the events



III Grouping things :—

Gathering both Structural things & Behavioral things into a Single Package.

Package : A package is a General purpose Mechansism Used to Organize group of things together.



Scanned by CamScanner

IV Annotational things :—

This are explanatitory part of System & this things respresents Constraints, Comments, Decriptions.

Note: A Note is a Symbole Used for rendoring Constraints and Comments to the Specific thing in UML model.

[dog eared symbol]

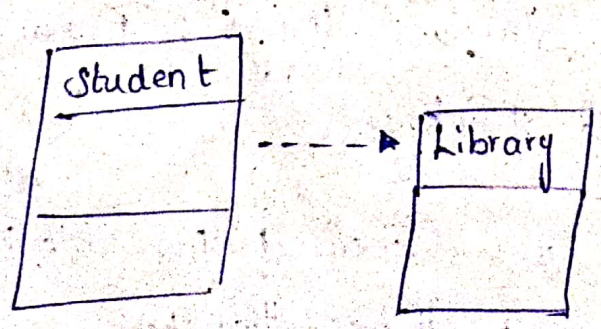
II Relationships :— 4 kinds of R.

1) Dependency
4) ~~Relationship~~ Realization
3) ~~Dependency~~ Generalization.
2) Association

1) Dependency: One thing Depence on the another thing.

[The element which depends on other element should be placed in left side always]

Eg



[Student should depends on the library for books]

(Dependent) (Independent)

Scanned by CamScanner

Symbol

Dependent element ----▸ [hallow head] independent element.

Depend symbol should be in left side.

Relationship :-

Relationship shows how elements are associated with each other and this associate on discribes the functionality of an application.

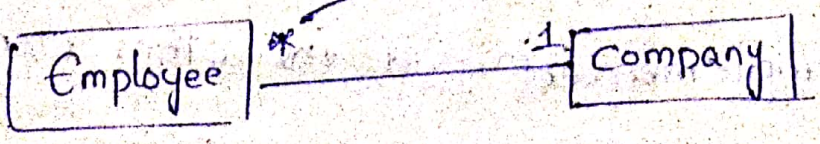
Dependency :-

Dependency is a symmatic Relationship b/w two things in which change in one element will effect the other element.

2) Association :-

Association is Basically a set of links that Connects the elements of UML model. It also describes how many object are taking place in this relationship. It resp by with a Multiplicity.

Eg:



Employee is associated with Company.

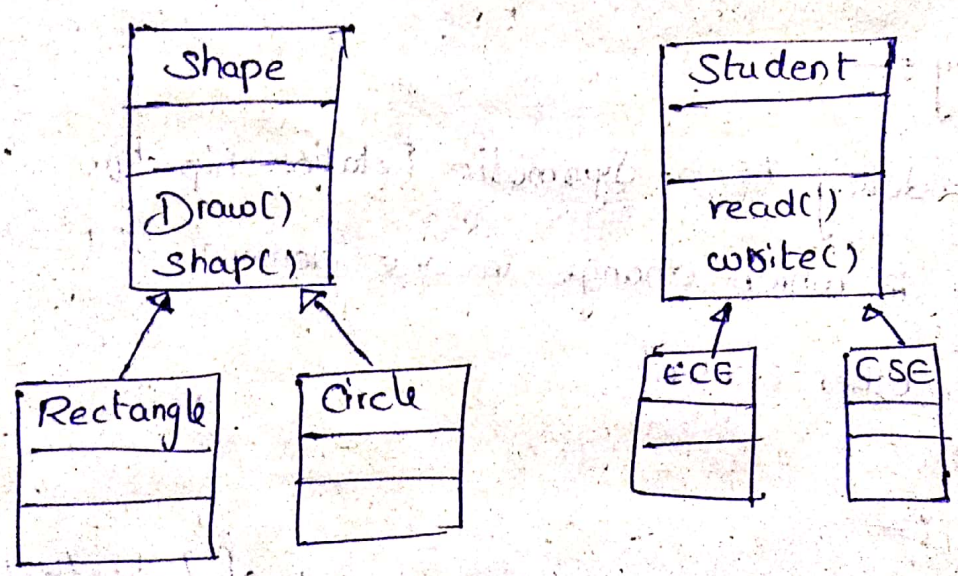
Scanned by CamScanner

3) Generalization :—

Generalization can be defined as a relationship which connects a specilized element with a Generalized element.

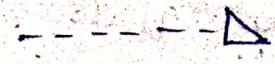
It basically describes the inheritance relation-ship in the world of object (parent & child relationship].

Eg



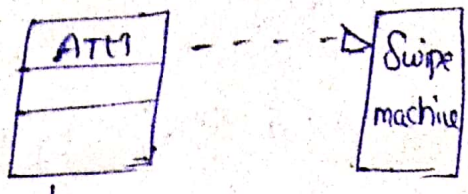
4) Realization

Realization can be defined as a semantic relationship in which two elements are connected. One element describes some responsibilities which are not implemented and the other element implements them.



Scanned by CamScanner

eg



if it is't working

III Diagrams

1) Class diagram — (Static)
2) Object diagram
3) Use-case diagram
4) Sequence diagram } → Interaction diagrams.
5) Colloboration diagram
6) Statechat diagram
7) Activity diagram
8) Component diagram
9) Deployement diagram.

1) Class diagram : [Static]

It Consists of classes, interfaces, collaboration, and their relationships.

It can address the static design view of a system and active classes are also involved in the class diagram.

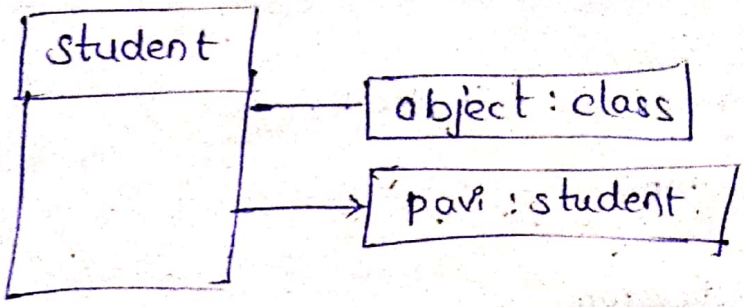
It is static point of view

Scanned by CamScanner

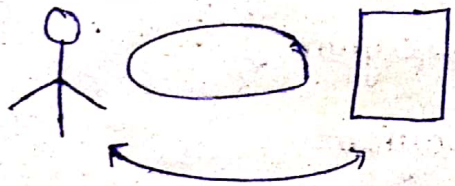
2) Object diagram :- [static]

→ It consists of objects and their relationships.

→ It can address the static view and can be defined as static snapshot of a class.



3) Use-Case diagram :- [static]



→ It Consists of set of usecases, actors & their relationships.

→ This is mainly used for representing the behaviour of the system.

Interaction diagram : [Dynamic]

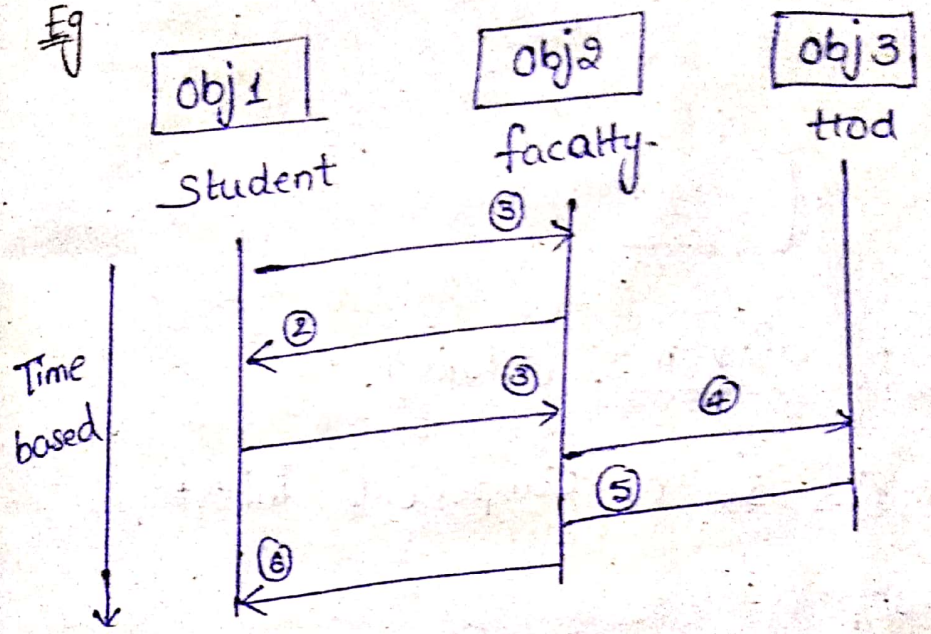
→ It consists of set of interactions and objects.

Scanned by CamScanner

4) Sequence diagram :-

→ It is a kind of interaction which emphasizes on time model msg among the objects.

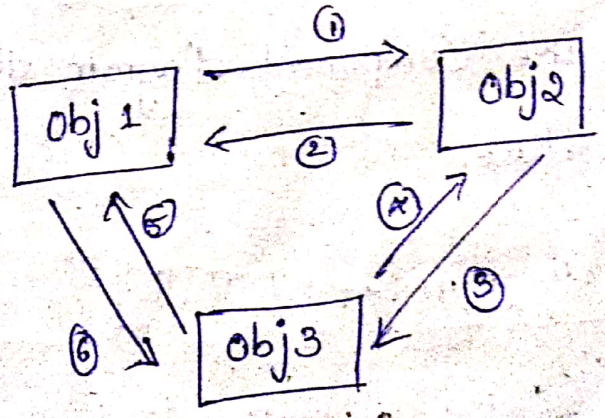
Eg



5) Collaboration diagram:

[It is not based on time].

Eg.



7) Activity diagram [Dynamic]

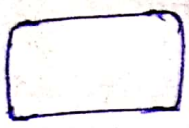
It is Special kind of state chat diagram that Consists Set of activities in response to the events.

Scanned by CamScanner

6) State-chat diagram :- [Dynamic]

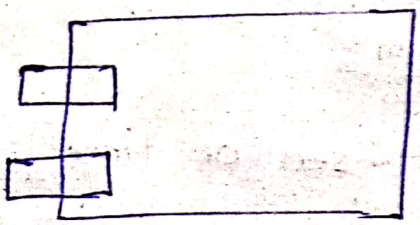
It Consists, of states in a state-machine during the life-time of an object. Set

It also a kind of class diagram and addresses the dynamic view of System.



8) Component diagram: [static]

→ It Contains Set of Components, interfaces and relationship.

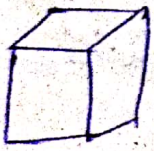


It should consist an intesface.

* When we use the Component diagram differently we use the interface. ×

9) Deployment diagram :- [Static]

It Consists of Set of nodes that exists at runtime envisonment. and the set of Components live on them.



Scanned by CamScanner

Rules of UML :-

→ It describes the vocabulary parts of a system.

→ A well formed model is one that is semantically self - Consistent.

1) Name : What you can call things, relationship, diagrams.

2) Scope : The content that gives specific meaning to a name.

3) Visibility : How those names can be seen and used by others.

4) Integrity : How things properly and consistently relating to each other.

5) Execution : What it means to run or simulate a dynamic model.

I
- class - rectangle
- interface - circle
- collaboration - ellipse
- Usecase - ellipse with solid lines.
- Activity class - heavy lines
- Component - rectangle with tab.
- Nodes - cube

II
- Interaction → directed line
- State machine → rounded rectangle

III
- package → tabbed folder

IV
- Note → retangle with a dog-eared corner.

Scanned by CamScanner

Structural things → nouns of UML

Behavioral things → dynamic parts of UML

Grouping things → Organizational part of UML

Annotational things → Explanatory parts of UML

1) Things are abstractions that 1st class citizen in a model.

2) These things are basic object-oriented building blocks of the UML.

3) These Relationships are the basic relational building blocks of the UML.

2nd

Dependency - dashed line ---->

Association - Solid line 0..1 *

Generalizations - solid line with hollow arrowhead pointing to the parent ——▷.

Realization - A cross b/w generalization & dependency relationship. ---▷

Scanned by CamScanner

4) A diagram in graphical presentation of a set of elements.

most often rendered as a connected graph of vertices & arcs.

3rd Address

1) class - static design view
2) object - static design view or process view
3) usecase - static usecase view.
4) Interaction - dynamic view
5) collaboration - -----
6) state chart - dynamic view
7) Acitivity - dynamic view
8) Component - static implementation view
9) Deployment - static deployment view

WAT-2 Result

1) B
2) B
3) D
4) D
5) D
6) A
7) A
8) C
9) B
10) C

Scanned by CamScanner

4) Extensible Mechanisms: Any language (UML) will have some mechanisms to ~~obtained~~ extend its capabilities such as Syntax and Semantics.

There are 3 types of Mechanism.

1) Stereotypes 2) Tagged Values 3) Constraints

Stereotypes: It is represents a new elements to a model.

<< >> (angulas brac)

Tagged Values: It is represents new attributes. { }.

~~Boundar~~ Constraints: Constrains represents the boundary of the entire model. { }

Eg: aplicable only for 2 months.

Common Mechanisms:

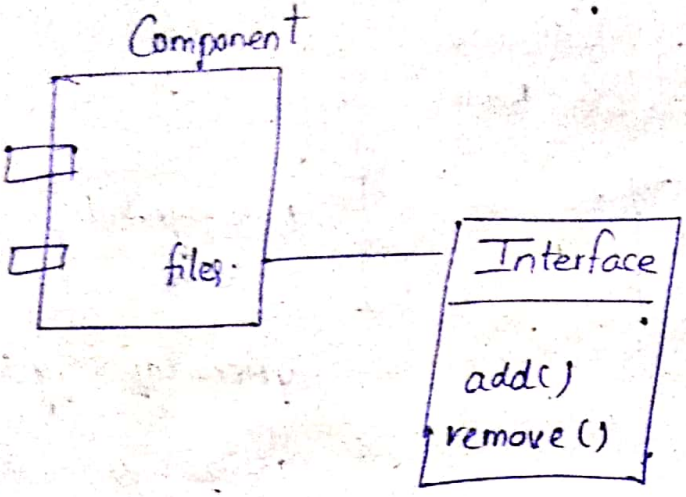
1) Specifications : Things that determine the model.
2) Adornments : xtra added element, optimal element.
3) Common division
4) Extensible mechanism

Scanned by CamScanner

3) Common-divisions :

1) Classes & objects

2) Component & Interfaces.



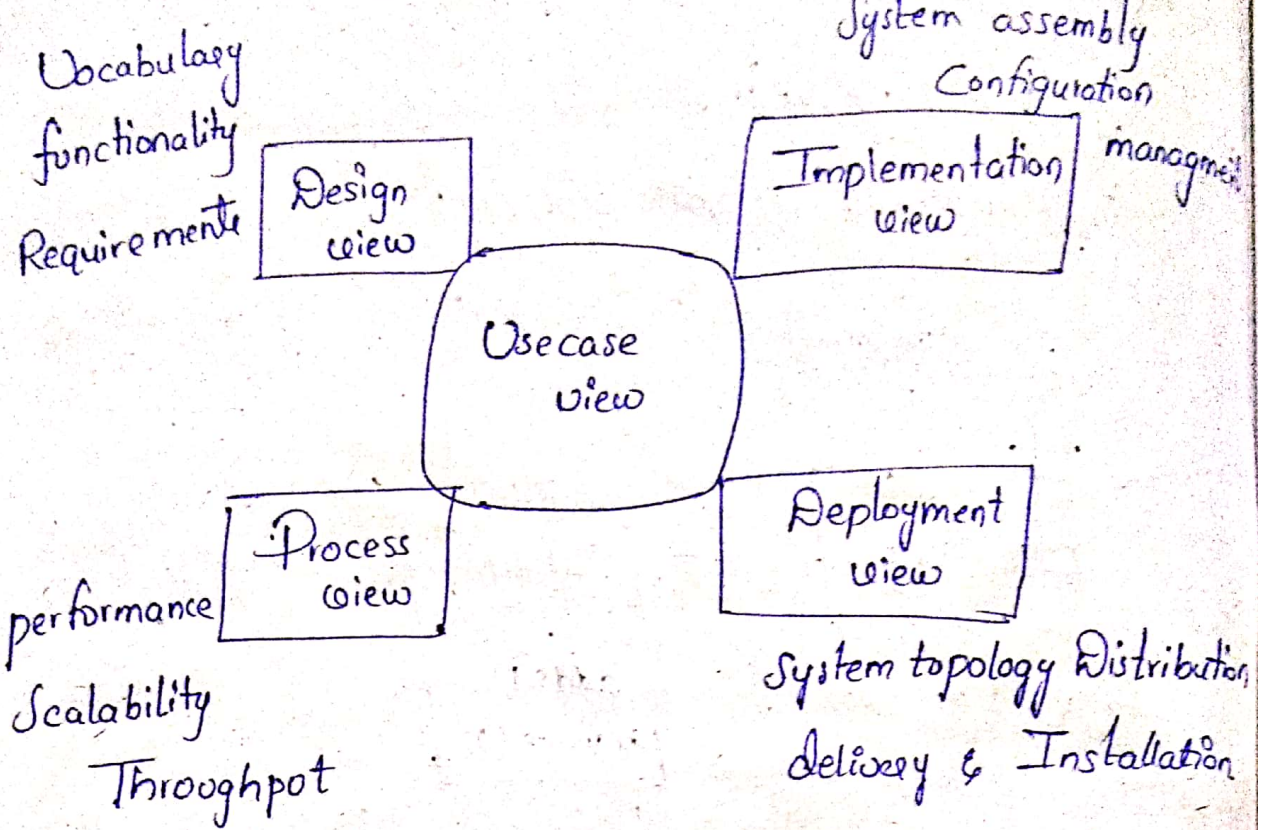
UML Architecture :-

→ Any real world system is used by different users whether it may be developers, testers, business people, analyst & many more.

→ Before desiging a system the architecture is made with different perspectives in mind.

→ The most important part into visualize the system from perspectives of different viewers.

Scanned by CamScanner



Use-Case View:

→ It is describe behavior of a system by its end users, analyst & testers.

→ It shapes the system architecture.

→ The static aspects are Captured in Use-Case diagram.

→ The dynamic aspect are Captured in interaction and activity diagram.

Design View:

→ It is describe classes, interfaces, & Collaboration that form voc.

Scanned by CamScanner

Process View:

→ It is Support to functional requirement.

→ The static aspects are Captured in class & object diagrams.

→ The dynamic aspects are Captured in interaction, state chart and activity diagram.

Process View

→ It is decribes threads & processes that form System Concurrency & Synchronization mechanism.

→ It addresses the performance, Scalability & throughput.

→ The static aspects are Captured in class & Object diagrams.

→ The dynamic aspects are Captured in interaction, state chart & activity diagram.

Implementation View

→ It describe the Component & files that are used to assemble.

→ It addresses Configuration managment.

Scanned by CamScanner

→ Static aspects are Captured in Component diagrams.

→ The dynamic aspects are captured in interaction, State chart & activity diagram.

⇒ Deployment View:

→ It encompasses the nodes that form System's Hardware topology on which the System executes.

→ It addresses distribution, delivery and installation of the parts.

→ Static aspects are deployement diagram.

→ Dynamic aspects are Captured in interaction, State chart & activity diagram.

Scanned by CamScanner

Software Development Life Cycle.

UML is largely process independent. To get most benefit from UML it should Consider a process.

1) Use-Case driven
2) Architecture Centric driven
3) Iterative and incremental driven

1) Use-Case driven:-

It means that usecases are used as primary artifacts for establishing the desired behavior of the system, for verifying & for validating the system Architecture, for testing and for Communicating among shakeholders of object.

2) Architecture Centric Driven:-

It means that System Architecture is used as primary artifact for Conceptualizing, Constructing, managing and development.

Scanned by CamScanner

3) Iterative and Incremental Driven:—

It is a process that involves a stream of releases and Continuous integration of System architecture to produce the release.

There are 4 phases.

1) Inception 2) Elaboration 3) Construction 4) Tansistion

Inception:

This is the first phases of process when the idea for development is brought up and well founded for entering into next phase.

Elaboration:

→ During this phase, the project team is expected to Capture a healthy majority of system requirements

→ The primary goal is to address risk factors.

Construction:

→ It involves the executable architecture baseline into a Complete working system.

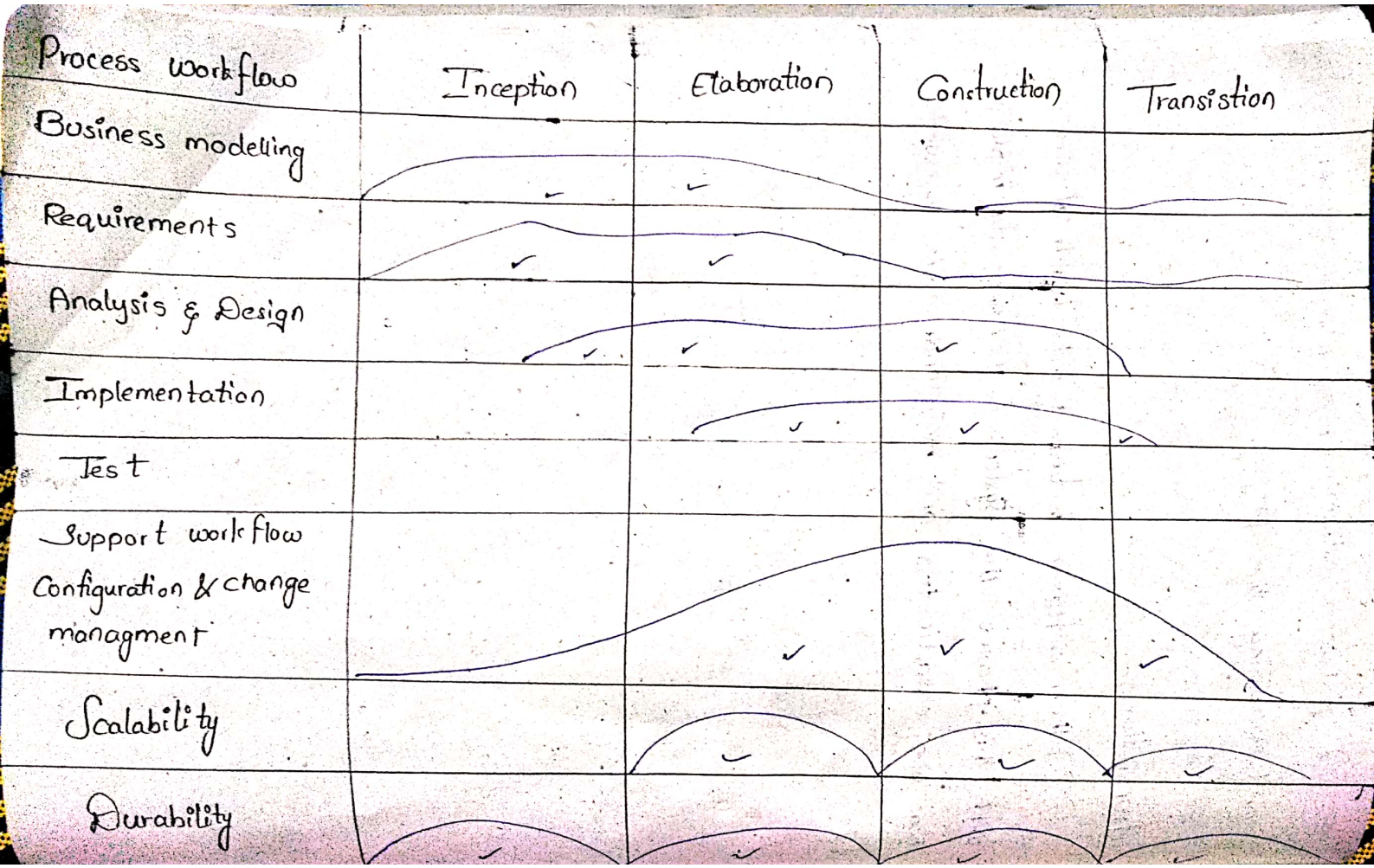
Scanned by CamScanner

→ The goal of Construction is to Complete all phases and examining the project.

Transistion:—

When software is turned into hands of user with this phases the System is Continuously improved bugs are easadicated and maintenance is observed.

Scanned by CamScanner



Scanned by CamScanner

Unit-2

Basic Structural Modelling

UML diagram types

Structural diagrams

1) Class diagrams
2) object diagrams
3) Component diagrams
4) Deployment diagrams

Behavioural diagrams

1) state-chat diagrams
2) Usecase diagrams
3) Activity diagrams
4) Sequence diagrams
5) Collaboration diagrams.

→ class diagram is most oftenly used.

UML modelling types

- Strutural modelling
- Architectural modelling
- Behavioural modelling

Strutural modelling → Basic SM, Advanced SM.

Basic SM

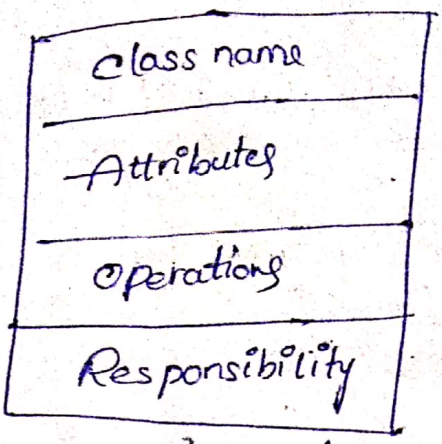
1) classes
2) Relationship
3) Diagrams
4) Common mechanisms.

Scanned by CamScanner

Classes

1) Name
2) Attribute
3) Operations
4) Responsibility
5) Organization of Attributes / operations
6) Other features

SN: 2, 3, 4
PN: 5, 6



Operations - Add(), remove()

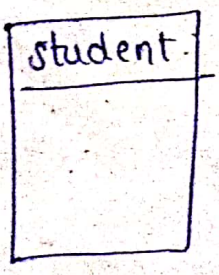
SN → Simple name

PN - path name.

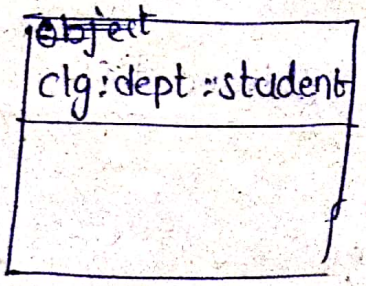
Classes : classes is an abstraction of UML model. classes hides the data and implements the operations. These operations bare implemented by interfaces. classes specifies the structural aspects of a model. Each and every class is represented with some terms and concepts.

Name represents two types

Simple name



Path name



Attributes

s.no : integer = "567";

sname : char = "abc";

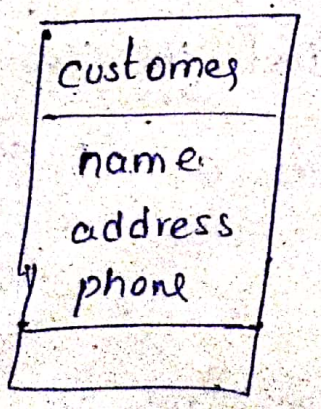
Scanned by CamScanner

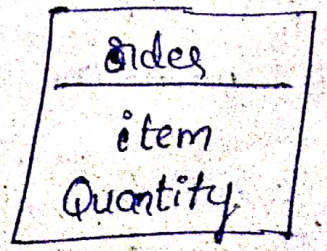
Common Modelling Techniques for classes :—

1) Modelling Vocabulary of a System.
2) Modelling distribution of responsibilities.
3) Modelling Non-Software things.
4) Modelling Primitive datatypes.

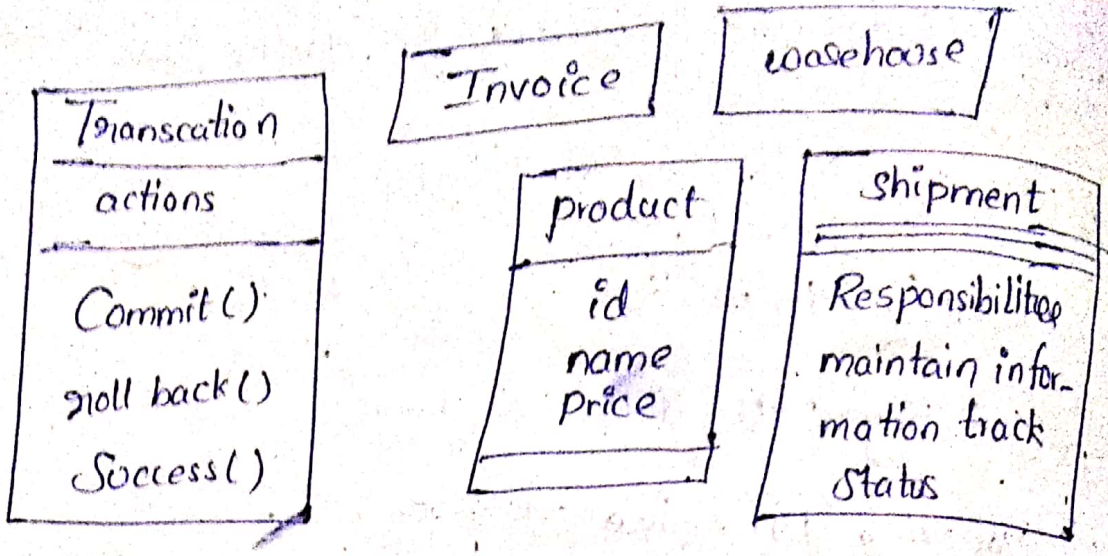
Modelling Vocabulary of a System :—

1) Identify those things that user uses to describe the problem are Solutions.
2) Use CRC cards and usecase base analysis to help in finding the abstractions.
3) For each abstraction identify set of responsibility and made sure that each classes is well defined first there is good balance of responsibilities among all your classes.
4) Provide the attributes & Operations that are needed to carry out those responsibilities for each class.





Scanned by CamScanner



CRC: Class Responsibility Collaboration.

Modelling Distribution of Responsibility :-

1) Identify set of classes that work together to carry out some type of behaviour

2) Identify set of responsibilities for each of those classes.

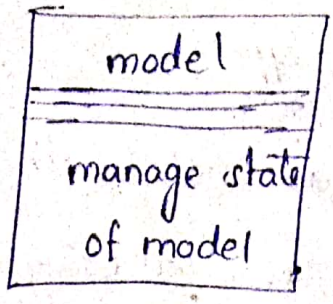
3) Look at the set of classes as whole class.

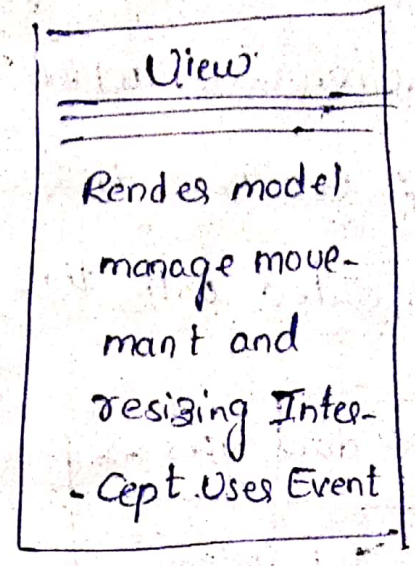
4) Split the classes that have two many responsibities into smaller abstractions.

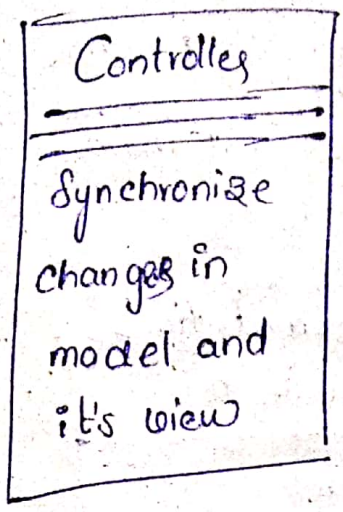
5) Collapse tiny classes into ~~types~~ larger responsibility so that each abstraction stands on its own.

6) Consider the ways in which the classes Collaborate with one another and redistribute their responsibility.

Scanned by CamScanner





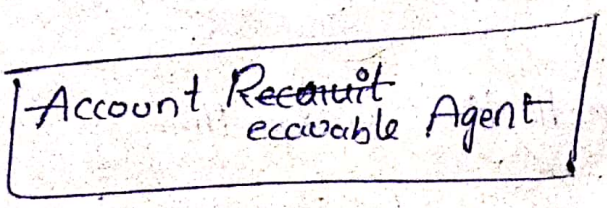


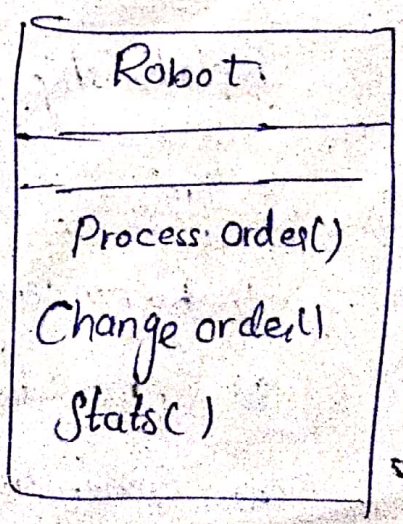
Modelling Non-Software things :—

1) Model that thing that you are abstracting as a class.

2) If you want ~~distribution~~ distiguish that thing for UML. Create new building blocks by using streotype.

3) If the thing your modelling is some kind of h/w that itself contain s/w.



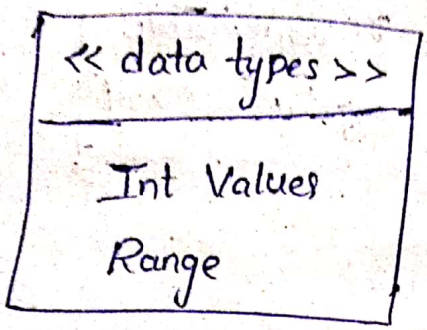


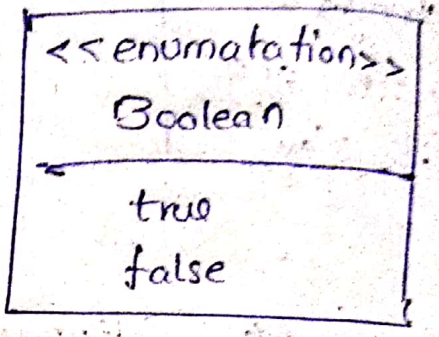
Scanned by CamScanner

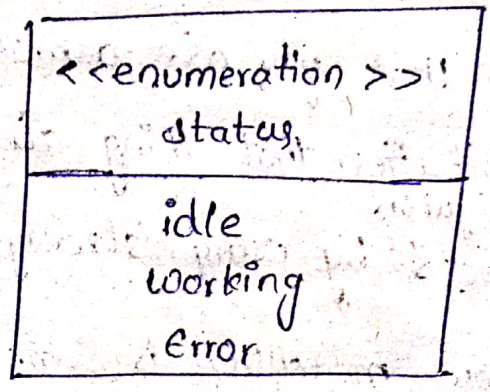
Modelling Primitive Data types :—

1) Model the things you're abstracting as a class, enumerations. which is rounded using class notation.

2) If you need to specify some range of values associated with this type then use Constraints.







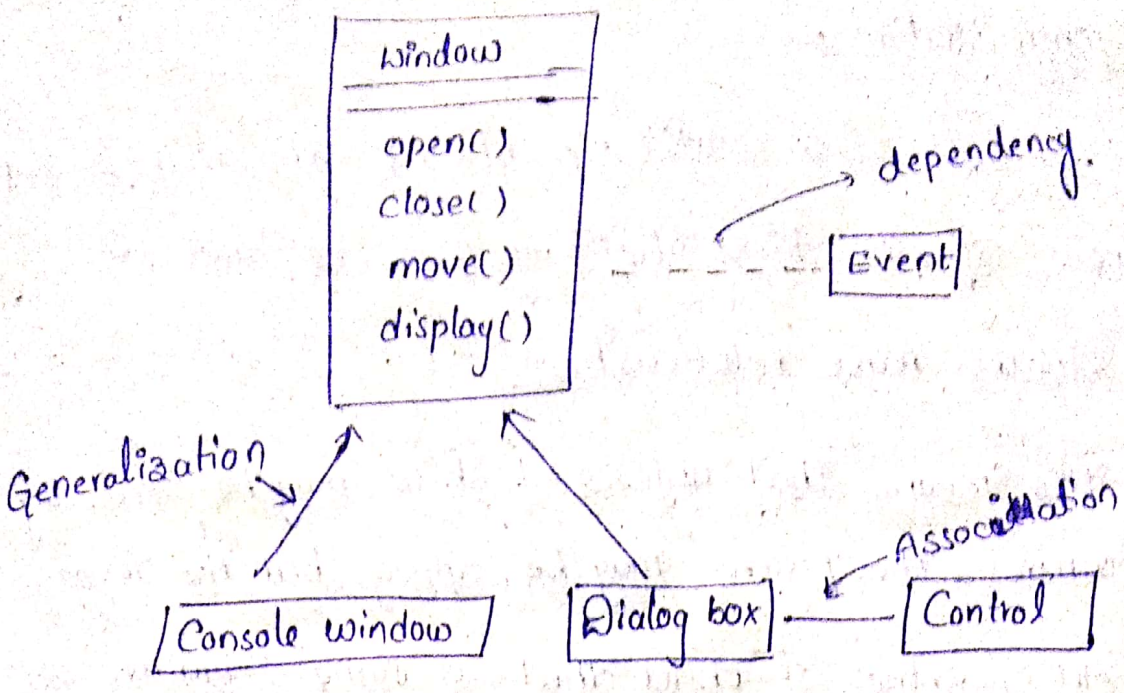
Relationship

Relationship a collection among things.

3 types

1) Dependency
2) Association
3) Generalization

Scanned by CamScanner

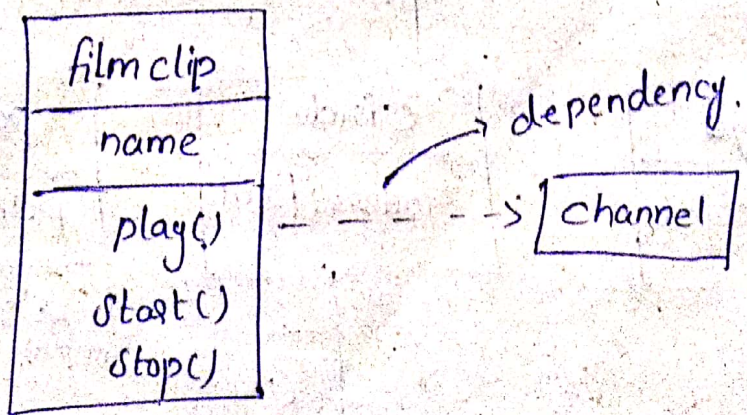


Console window & terminal

Terms & Concepts:

Dependency: Dependency states that change in specification of one thing may effect another thing that uses it.

→ this is used in Context of classes for show that one class uses another classes as an argument in the signature on operations.



Scanned by CamScanner

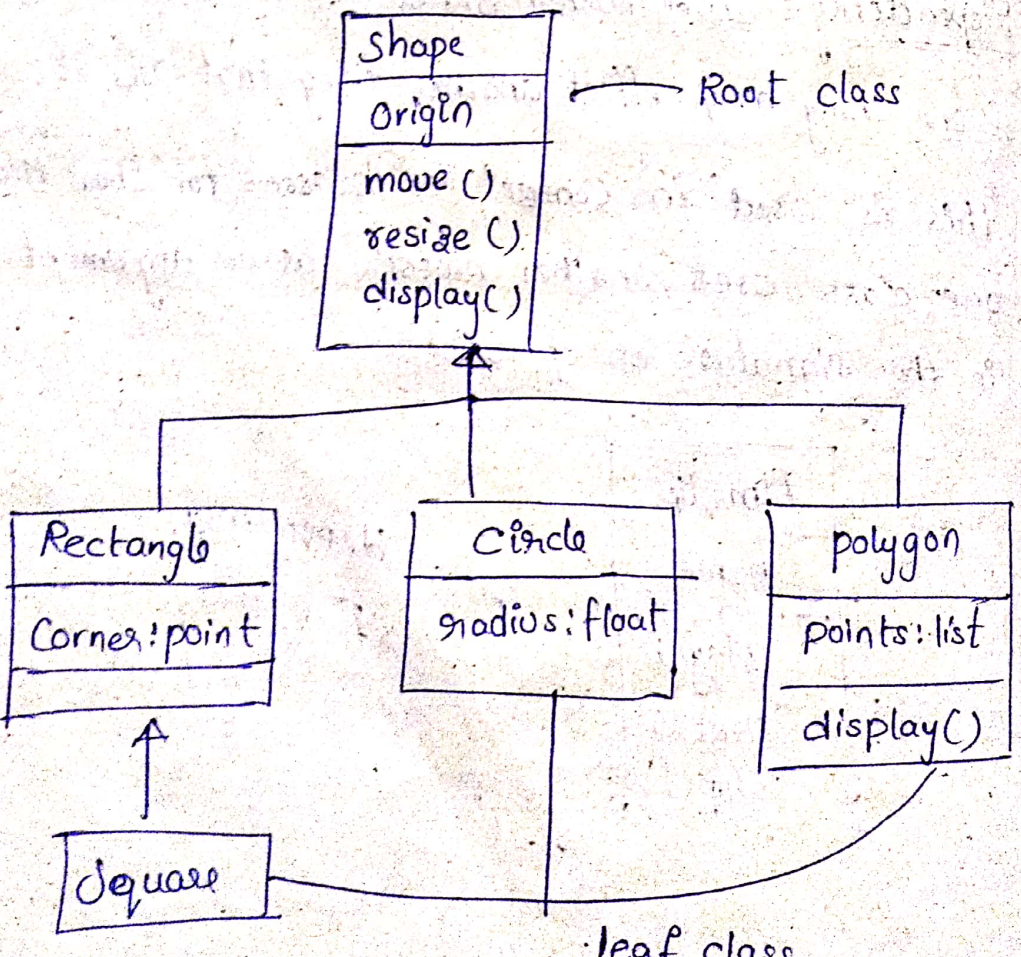
Generalization :-

→ It is a relationship blw general thing is and a more specific kind of thing. (Its is kind of ~~Another name~~ relationship).

→ Its means that objects of child may be used any where, the parent may be appear & but, not the reverse.

→ An operation of child that as some signature as an operation overrides parent operations, this is called polymorphism.

→ A class may have zero (or) more parents.



Scanned by CamScanner

→ A class with one parent is called single inheritence.

→ A class that has no parent are called "parent class" (or) "root class".

→ A class with multiple parents is called "multiple inheritance".

Association :-

→ It is the structural relationship that specifies objects of the thing can be connected to objects of another thing.

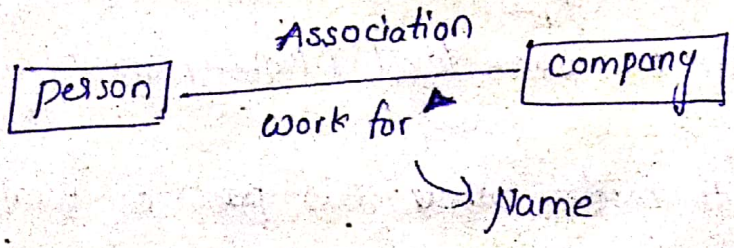
→ We have 4 types of associations.

1) Name
2) Role
3) Multiplicity
4) Aggregation

} 4 adornments.

5) other features

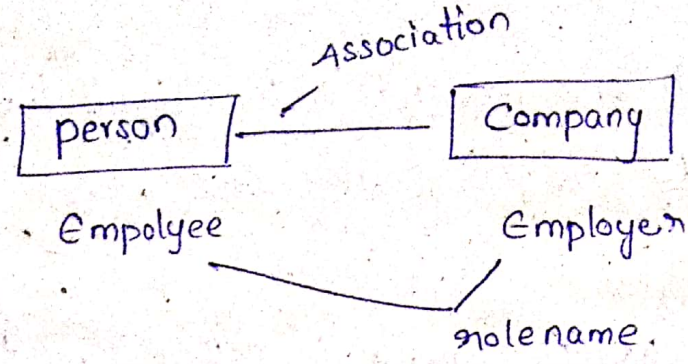
Name: Its describes nature of relationships.



Role: When a class participate in an association, it has a specific role that plays in the relationships.

→ A Role is just the face of the class at near end of Association presents to the class at another end of association.

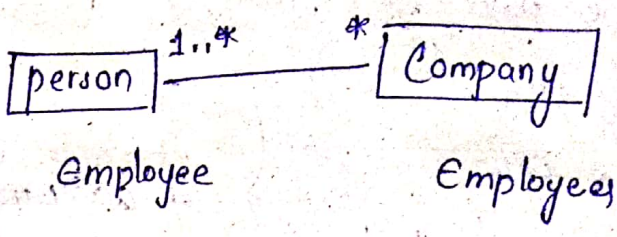
Scanned by CamScanner



Multiplicity : it represent structural relationship among objects.

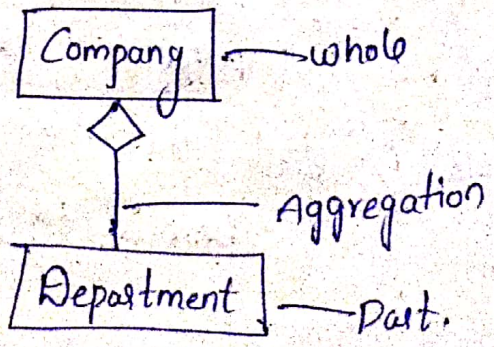
→ Its importance for you to state, how many object maybe Connected across an instance of an association.

→ *
→ 1
→ 0..1
→ 0..*
→ 1..*
→ Exact number



Aggregation: → A plain association b/w 2 classes repsented structural relationship/ reprentation. → In that both classes are Conceptualy at the same level (Nobody one imp then other one).

→ when you want to model whole (or) part of relationship in which 1 represents large thing and other respresent smallest thing.



Scanned by CamScanner

other features

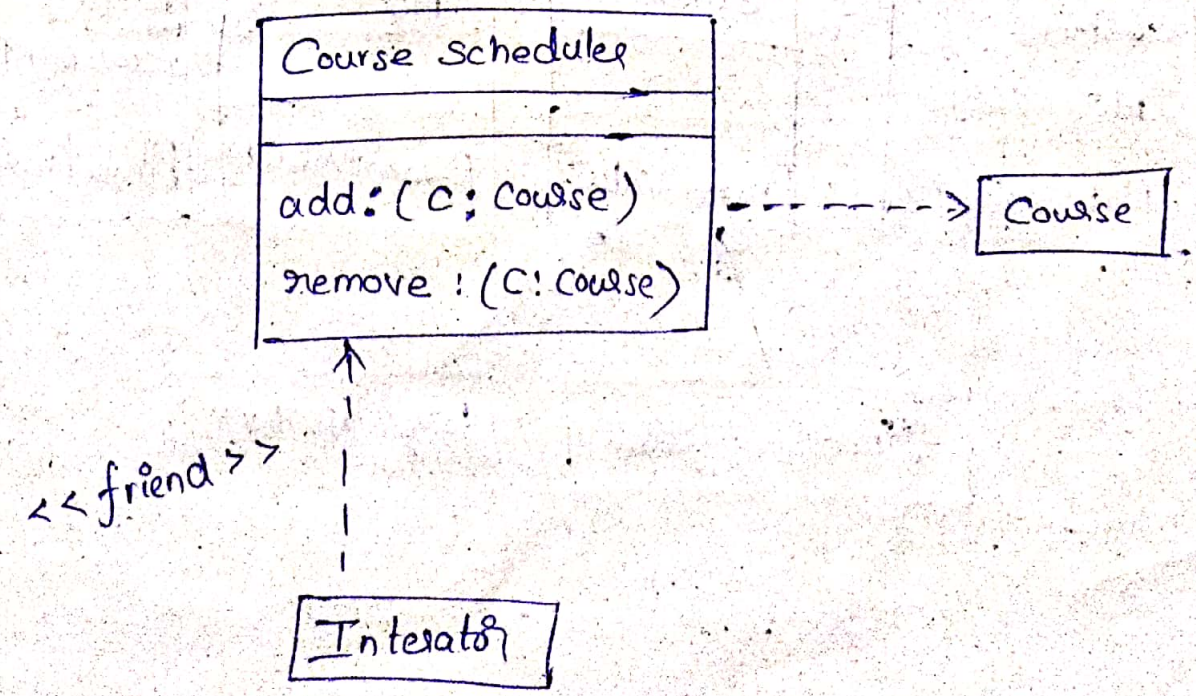
1) Links
2) transitions
3) Composite aggregation
5) discriminant
6) Association class.
7) Navigation.

01/02/19

Common Modelling technique for relationship :—

1) Modelling Simple Dependency
2) Modelling Single Inheritance
3) Modelling Structural Relationships.

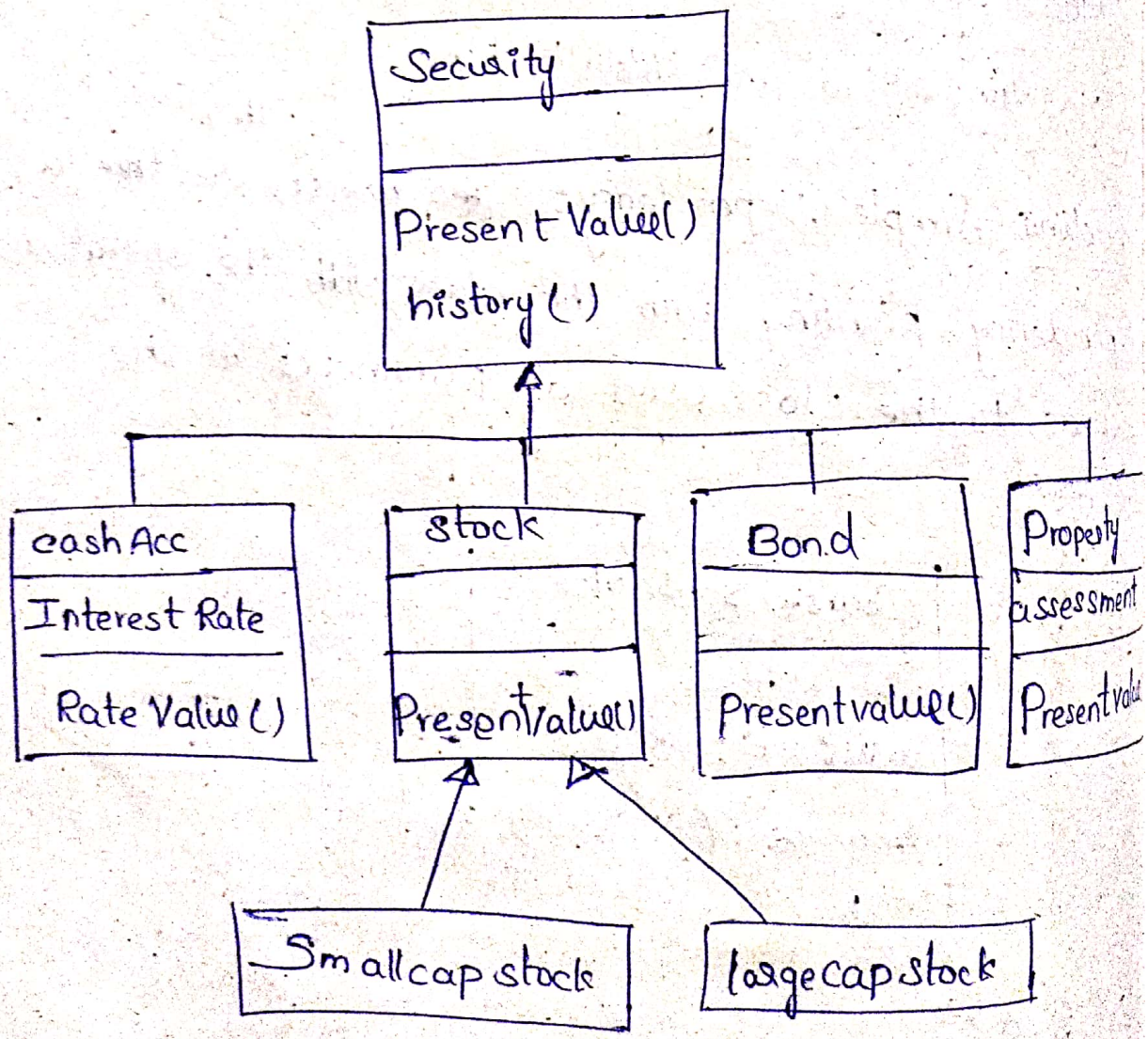
Modelling Simple Dependency: Two model x create this Dependency pointing from the class with the operations ~~used as~~ to the class used as parameter in the operations.



Scanned by CamScanner

Modelling Single Inheritance : To mode this inheritance given set of classes. look for reprensibility, attributes and operations that are common two or more classes.

→ ~~Elements~~ Elevate Common responsibility, attributes and operations to a more general class inheritance from the more general class. By ~~replacing~~ a generalazation relationship that is drawn from Specilized class to its more general parent.



Scanned by CamScanner

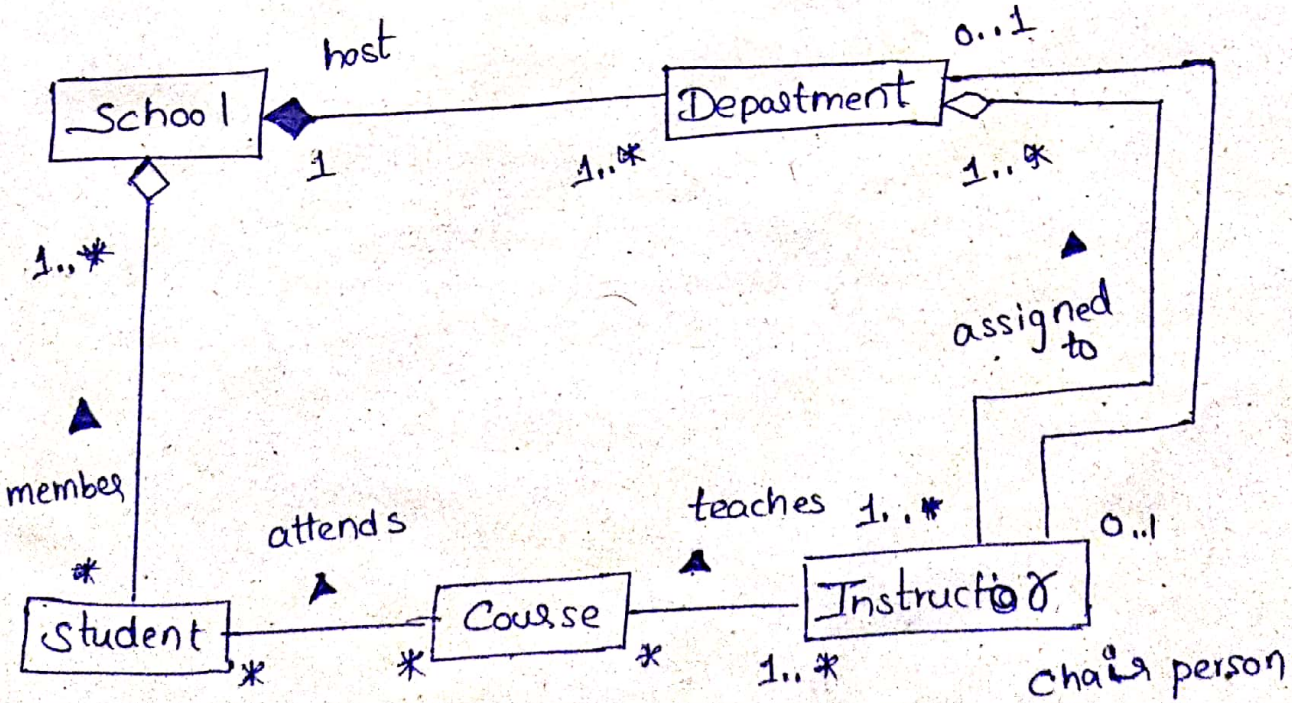
Modelling Structural Relationship: Dependency, Generalization are one sided because the don't have knowledge on child classes. 2) In case of association relationship modelling press of one another.

3) To model structural relationship

→ For each pair of classes if you ~~object~~ need to navigate from one object to another object specify an association b/w them.

→ This called "data driven view of Association".

→ For each pair of classes of objects of one class need to interact with objects of another class as parameters to an operations. Specify association



b/w them. This is called "behaviour driven View of Association".

Scanned by CamScanner

→ For each of this association specify a multiplicity as well as role names.

→ If one of the classes in an association is structually (or) Organizationally a whole. Compose to classes at the other end that look like small part. This is called "Aggregation by adorning the Association at end of the whole".

Scanned by CamScanner