RBB.

UNIT-I : OBJECT ORIENTED MODELING AND DESIGN.

8 hours **Introduction, Modeling Concepts, Class Modeling:** Introduction to Object oriented (OO) development, OO themes, OO modeling history. Modeling as Design Techniques: Modeling abstraction; The three models.

Class Modeling: Object and class concepts; Links and association concepts; Generalization and Inheritance. Introduction to association and aggregation.

INTRODUCTION

Note: Intention of this subject (object oriented modeling and design) is to learn how to apply object oriented concepts to all the stages of the software development life cycle.

* Object oriented modeling and design is a way of thinking about problems using models organized around real-world concepts.

* Object : fundamental construct which combines
  - → Data structure : Organizing and storing of data.
  - → Behaviour : functionalities to perform operations (Methods).

1 What is OO development?

Definition OO: OO means that we organize s/w as a collection of discrete objects that incorporate both Ds & behaviour.

9 Explain the four aspects of OO with example?

There are 4 aspects (characteristics) require:

- Classification
- Inheritance
- Polymorphism
- Identity

① Identity: Identity means that data is quantized into discrete, distinguishable entities called objects.

Eg: for objects: personal computer, bicycle, queen in chess etc.

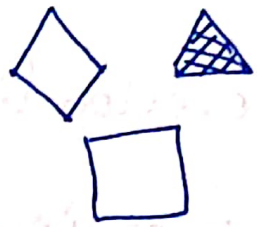
* Object can be concrete (file in file system) or conceptual (scheduling policy in multiprocessing os)
* Each object has its own inherent identity i.e two objects are distinct even if all their attributes values are identical.
* In a programming Languages → object is referenced to perform operations.

② Classification: means objects with same Ds (attribute) and behavior (operations) are grouped into a class.

Eg: Paragraph, monitor, chess piece

- Each object is an instance of its class.

• fig below shows objects and classes: Each class describes a possibly infinite set of individual obj.



abstract into →

Polygon class:
Attributes - Vertices
- border class
→ fill color
Operations: - draw
- Erase
- move

Eg Bycycle class:
Attributes :- frame size, wheel size, No of gears, material.
Operations :- shift, move, repair.

③ Inheritance: It is sharing of attributes & operations (features) among classes based on a hierarchical relationship. A super class has general information that sub classes refine and elaborate.

Eg Window (superclass) - has general info.
Scrolling (Adds scroll bar & offset).
fixed (subclasses) - Refine & elobrate

④ Polymorphism: Same operation may behave differently for different classes.
Eg: move operation: Pawn & Queen is different in a chess game.

• An operation is a procedure that an object performs.

Eg: Right Justify, display, more etc.

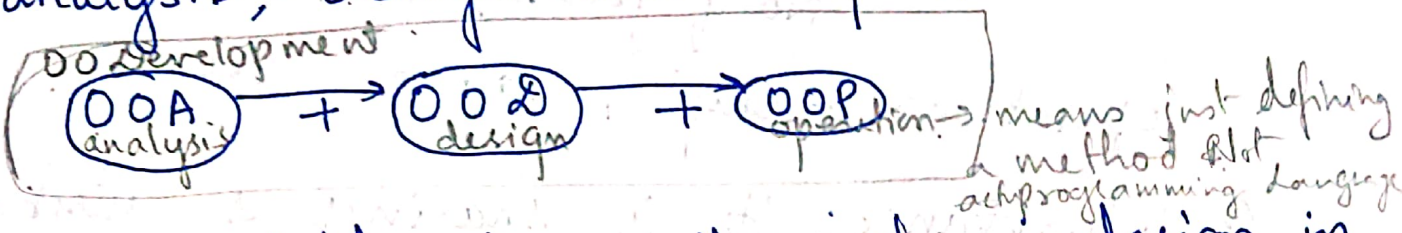
* Method: An implementation of an operation by a specific class.

Ex for polymorphism: Method overloading (static)
Method overriding (dynamic)

2. What is OO Development? Explain its methodology & its theory (10m).

What Is OO Development?

* Development refers to the s/w life cycle: analysis, design and implementation.



*OOMD: Addresses the clean design in a precise notation, facilities the entire s/w life cycle contains.

Modeling Concepts, Not Implementation:

* The essence of OO Development is the identification and organization of application concepts, rather than their final representation in a programming Language.
* It's a conceptual process independent of Programming languages.
* OO development is fundamentally a way of thinking & not a programming technique.
* It can serve as a medium for specification, analysis documentation & interfacing.

OO Methodology: Explain the stages of OO s/w development methodology.

- Here we present a process for OO development and a graphical notation for representing OO concepts.
- The process consist of building a model of an application and then adding details to it during design.

The methodology has the following stages.

1) System Conception: Business analyst (reader) or users formulating tentative requirements.

2) Analysis: Problems statements are rarely correct or complete. The analyst scrutinizes and rigorously restates the requirements from the system conception by constructing models. The analysis model is a consise precise abstraction of what the desired system must do not how it will be done.

(for ex: Coffee vending m/c — Put coin, you should get coffee)

The analysis model has two parts:

Analysis model → Domain model.
Analysis model → Application model.

Domain model: A description of the real-world objects reflected within the system.

Application model: A description of parts of the application system.

Eg:→ In case of stock broker application-

Domain objects may include - stock, bond, trade & commission.

Appln. objects might control the execution of trades & present the results.

development team sits to gather & develop at high level

3.) System design: The system architecture for solving the application problem.
Policies, more detailed portions of design
Make tentative resource allocation.

4.) Class design: The class designer adds details to the analysis model in accordance with the system design strategy. Here focus is on the data structure and algorithms needed to implement each class.

5.) Implementation: Translate the classes & relationships developed during class design into a particular programming language, database or hardware.

Three models:
We use three kinds of models to describe a system from different view points

1.) Class Model:- It describes the static structure of the objects in the system and their relationships.

static

* Class model contains class diagram – a graph whose nodes are classes and arcs are relationships among the classes.

2) static structure – id the structure of its objects & their relationships to each other at a single moment in time

dynamic
mouse

2) State Model :- describes the aspects of an object that change over time.
The state model specifies implements control with state diagrams.
A state diagram is a graph whose nodes are states and whose arcs are transition between states caused by events.

3) Interaction Model :- describes how the object in a system cooperate to achieve broader results.
The interaction diagram starts with use case, that are then elaborated with sequence diagram & activity diagram.

* Use case → focuses on functionality of a system ie what the system does for user.
* Sequence diagram → shows how the object interact and time sequence of their interaction.
* Activity diagram → elaborates important processing steps.

OO Themes.

1.) Abstraction → focuses on only essential aspects ie focus on only what is an object & does, before deciding how to implement it.
2.) Encapsulation → object internal implementation details are hidden from other object.
3.) Combining Data and Behavior
4.) Sharing
5.) Emphasis on the Essence of an Object → sum of their seperate effects
6.) Synergy → interaction of two or more substance produce a combine effect greater than

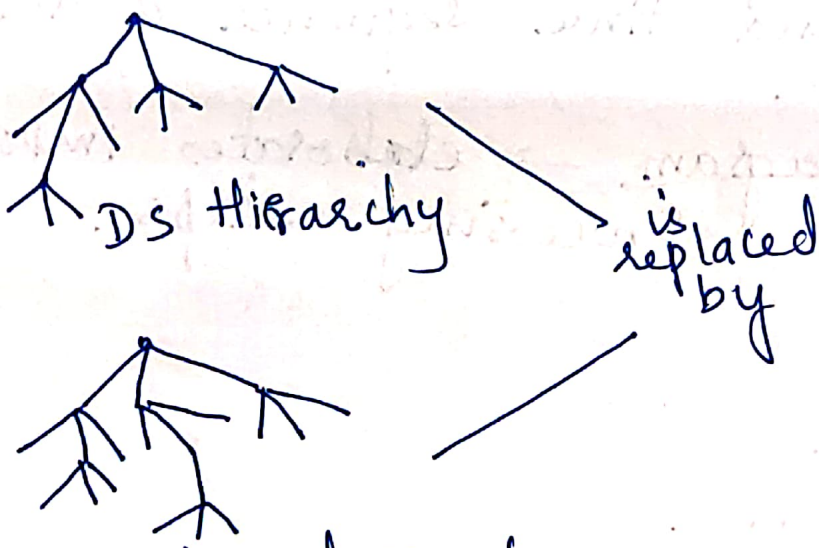
A good model captures the crucial aspects of a problem & omits others.

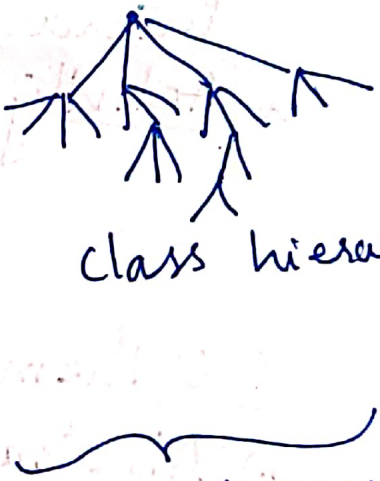
1) Abstraction ⇒ focuses on essential aspects of an application, while ignoring details ie. focusing only on what an object is and does, before desiding how to implement it.

2) Encapsulation (Information hiding) :- It separates the external aspects of an object (that are accesible to other objects) from the internal implementation details (that bare hidden from other objects)

* Encapsulation prevents portions of a program from becoming so interdependent that a small change has massive ripple effects.

3) Combining data and behavior:





* In OO system the data structure hierarchy matches the operation hierarchy.

4.) Sharing: OO techniques provide sharing at different levels.

* Sharing via inheritance is one of the main advantage.

* OO development not only lets you share information within an applicatim, but also offers the prospect of reusing designs & code on future projects.

5.) Emphasis on the essence of an objed. (what is the ds used rather than how it is implemented.)

* OO techonology focues on what an object is. rather than how it is used.

* OO development places a greater emphasis on data structure and lesser emphasis on procedure.

6.) Synergy (Interaction or co-operation of two or more organizations, substances to produce a combine effect greater than the sum of their seperate effects)

* Identity, classification, polymorphism and inheritance characterize OO languages.

* Each of these concepts can be used in isolatim, but together they complement each other synergistically.

1.5 OO Modeling History

What are models? Discuss their classification?

Ch-2

Modeling as a design Technique.

A model is an abstraction of something (system) for the purpose of understanding it, before building it.

Modeling: Designers build many kind of models for various purposes before constructing things. Models serve several purposes -

① * **Testing a physical entity before building it:** Engineers test scale models and improve their dynamics. ex:- Engineers test scale models of airplane, car & boats etc in wind tunnels & water tanks to improve their dynamics. Computation permit the simulation of many physical structures without the need to build physical models.

Test it for its efficiency

② * **Communication with customers:** Architects and product designers build models to show their customers, so that demo models show some or all the external behavior of a system.

③ * **Visualization:** Storyboards of movies, TV shows and advertisements let writers see how there ideas flow and they can modify unwanted transitions, unnecessary segments befor detailed writing begins.

Adv:- gives a clear idea of what is required.

Disadv:- more time is consumed to get a perfect o/p

④ * Reduction of complexity:

→ The main reason for modeling is to understand the systems that are too complex in understanding

→ Models reduce complexity by separating out a small number of important things to do with ↑ at a time.

Abstraction:

Def: "Abstraction is the selective examination of certain aspects of problem".

* Goal of Abstraction is → highlight the aspects that are more important and ignore which are less important.

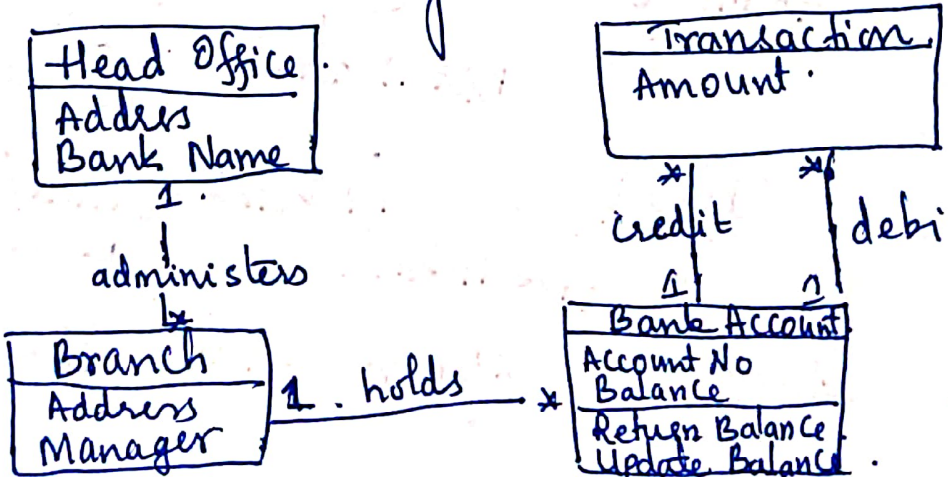
Explain three kinds of models which separates a system into distinct views.

2.3 The Three Models.

- Three Models are related but different View point. Each have its own important features.

1. Class model: class diagram expresses class model

- It describes the "static structure" of the objects in a system and their relationships



Eg: identify relationship attributes & operations

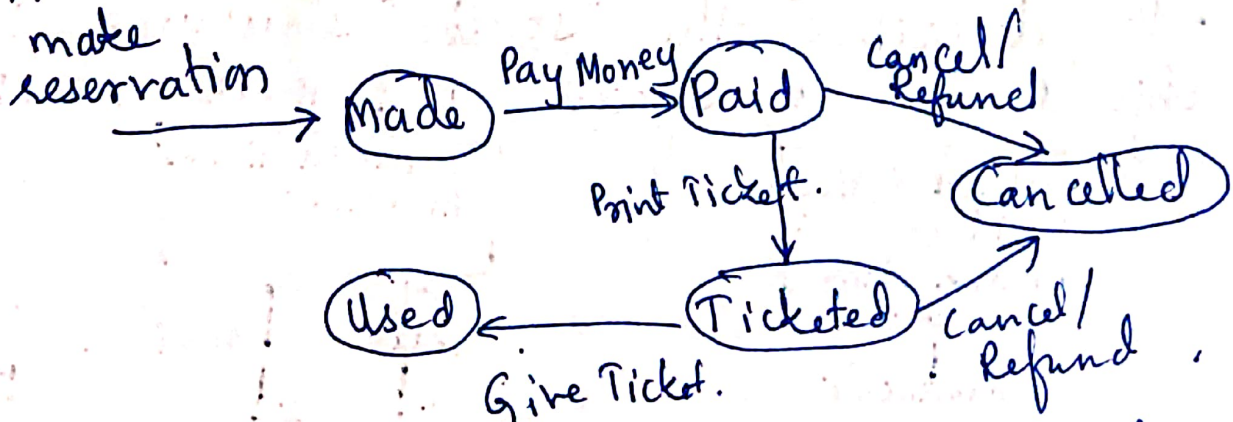
Goals

1. Model provides context for the state and interaction models.
2. Captures important concepts of an application from the real world.
3. Class diagrams express the class model
4. Generalization lets classes share structure and behavior, and association relate class.
5. Classes define the attribute values carried by each object and the operations that each object performs of undergoes.

2. State Model.

- Describes those aspects of objects concerned with time and the sequencing of operations.

Ex.



State diagram express state Model.

- Each diagram shows → state & event sequence.
- Action + events —becomes→ operations on objects in class model.
- References between state diagram → interaction in interaction model.

3] Interaction Model:

- Describes interaction between objects.
- Use cases, sequence diagrams and activity diagrams document the interaction model.

Use cases → describes interaction between system & outside actors,

Sequence diagram → show the object interact & the time sequence of their interaction.

Activity diagrams → shows the flow of control among the processing steps of computation.

4] Relationships Among the Models:

- Each models describes one aspect of the system but contain reference to other models.

Object Oriented Modeling History.

In 1991 @ GE R&D (company) led to development of the Object Modeling Technique (OMT).
(Problems arised due to similar notation.)

↓

In 1994 Jim Rambaugh Joined IBM & he started working with Grady Booch. Ivar Jacobson also joined with both. (1995)
Here they started consolidating various notations

↓

1996 Object Management Group (OMG) issued request for proposals for standard OO modeling notation.
ie UML.

↓

1997 OMG Accepted the Unified Modeling Language (UML).

↓

2001 OMG members revised UML by adding missing features.

↓

In 2004 UML 2.0 revision is approved
This book is based on 2.0 version of UML

Tom

Class Modeling:

A class model captures the static structure of a system by characterizing the objects in the system, the relationships between the objects and the attributes and operations for each class of objects.

Object and Class Concepts

① Objects ④ Values & Attributes
② Classes ⑤ Operation & Methods.
③ Class diagram

Purpose of class modeling is to describe objects.

1. **Objects:** An object is a concept, abstraction or thing with identity that has meaning for an application.

- All objects have identity and are distinguishable
  Ex:- Two apples with the same color, texture & shape are still individual apples.
- The term identity means objects are distinguished
  Ex:- Identical twins are two distinct persons, even though they may look the same.

* An object has 3 characteristics: state, behaviour & a unique identification. or

2. **Classes:**

→ An object is an instance of a class.

→ A class describes a group of objects with the same properties (attributes) behaviour (operations), kinds of relationships & semantics.

Ex: Person → class, company, process are all classes.

Name Birthdate work at a job → work for a job.

~~Ex: Persons company~~

3 Class diagrams

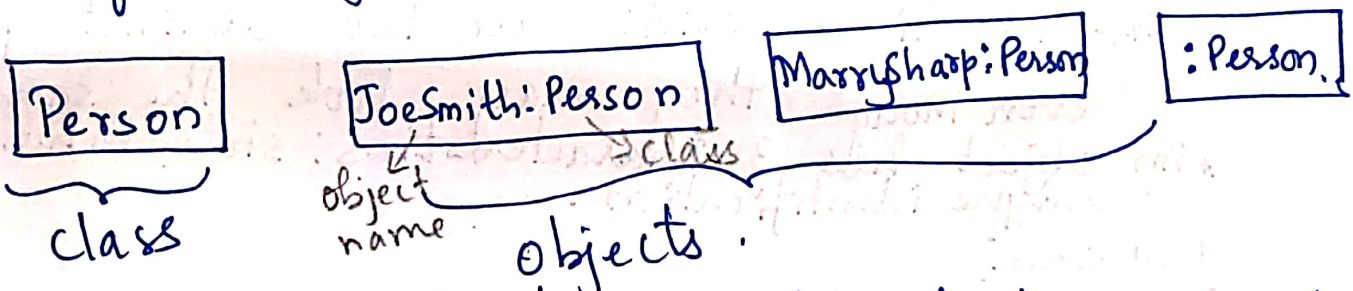
There are 2 kinds of structure < class diagrams / object diagrams

Class diagrams: provide a graphic notation for modeling classes and their relationships, thereby describing possible objects.

object diagram: shows individual objects and their relationships.

*object diagrams are useful for documenting test cases and discussing examples.

*class diagrams are useful for both abstract modeling and for designing actual prgms.



A class and objects. Objects and classes are the focus of class modeling.

Conventions used (UML)

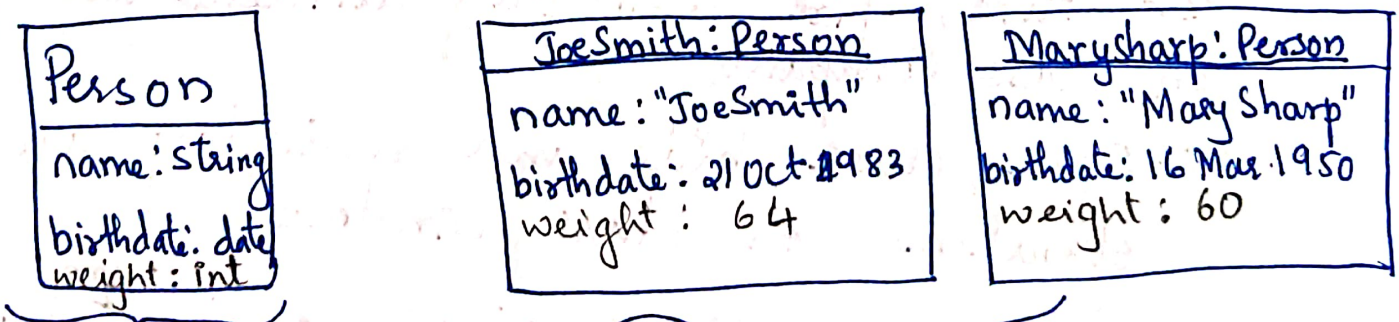
- UML symbol for both classes and object is box.
- Object are modeled using box with object name followed by colon followed by class name
- Use boldface to list class name, center the name in the box and capitalize the first letter.
- To sum together multiword names (such as JoeSmith) separate the words with intervening capital letters.

Values and Attributes

Value: A value is a piece of data.

Attribute: An attribute is a named property of a class that describes a value held by each object of the class.

following analogy holds: Object is to class as value is to attributes.



Class with Attributes — Object with values.

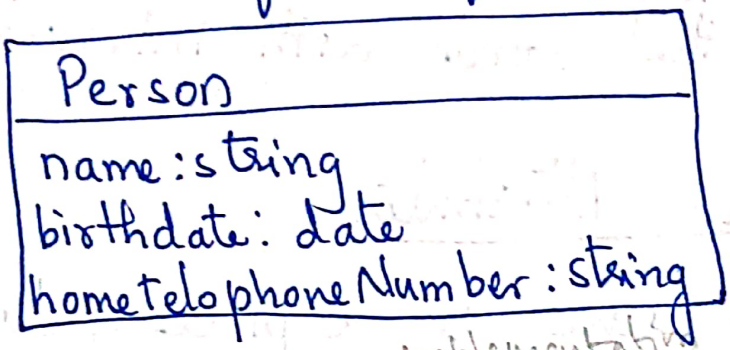
Ex:- Attributes: Name, birthdate, weight.
Values: JoeSmith, 21 oct 1983, 64.

The above fig shows modeling notation

Conventions used (UML)

- List attributes in the 2nd compartment of the class box. Optional details may follow each attribute
- A colon precedes the type, an equal sign precedes default value.
- ~~Alter~~ Show attribute name in regular face, left align the name in the box and use small case for the first letter.

Eg:-



⑤ Operations and Methods

Operations: An operations is a function or procedure that may be applied to or by objects in a class.

Ex:- Hire, fire and pay dividend are operations on class Company.

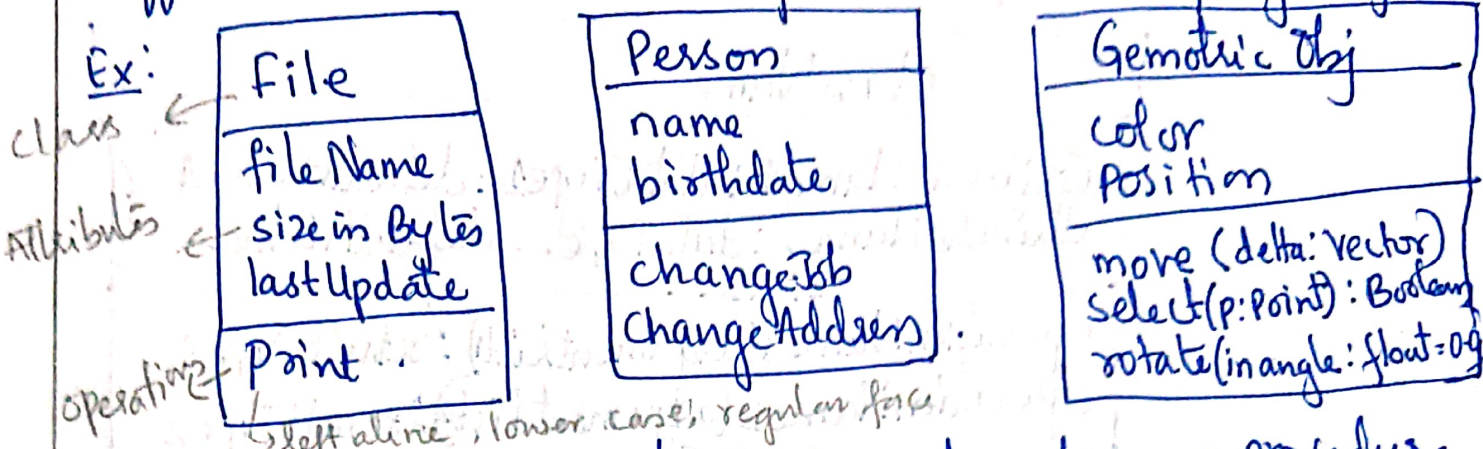
Ex:- Open, close, hide and redisplay are the operations on class window.

Methods: A method is the implementation of an operation for a class.

for Ex:- In class file, print is an operation, you could implement different methods to print ASCII files, print binary files, or print jpge pictures.

The same operations may be apply to many different classes. Such operations is polymorphic

Ex:



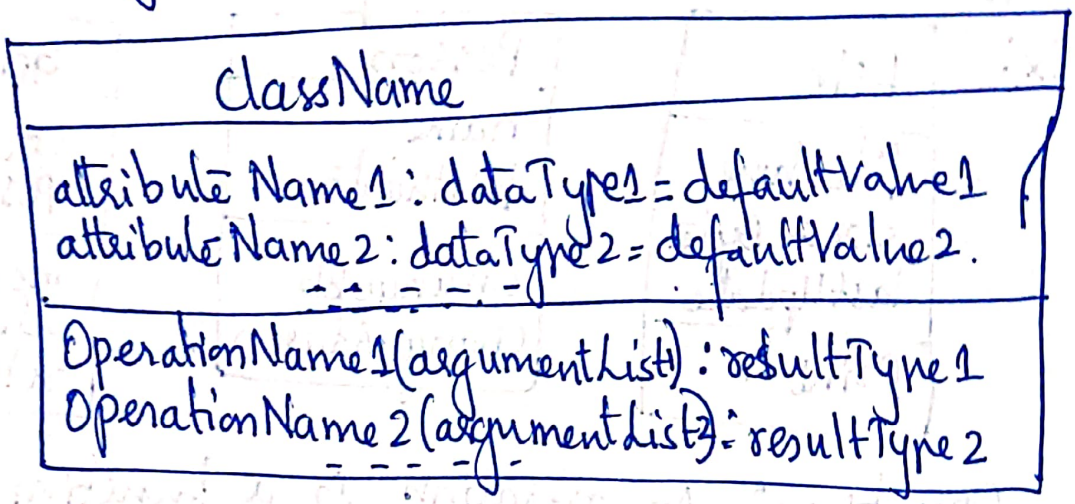
Operations: An operation is a function or procedure that may be applied to or by objects in a class.

UML conventions used:-

- List operations in 3rd compartment of class box.
- List operation name in regular face, left align and use lower case for first letter.
- Optional details like argument list and return type may follow each operation name.
- Paranthesis enclosed an argument list, commas separate the arguments.

Sucess

Summary of Notation for Classes.



* A box represent a class an may have as many as three compartments.
* Above fig shows summary of modeling notations for classes

Notation for an argument of an operation:

direction argumentName : type = defaultValue

* The direction, type and defaultValue are optional.
* Direction may be in, out or inout.
* The default value is used if no argument is supplied for the argument.

Link and Association Concepts.

Links and associations are the means for establishing relationships among objects and classes.

- Links and Associations.
- Multiplicity..
- Association End Names
- Ordering
- Bags and Sequences.
- Association Classes.
- Qualified associations.

Links and Associations.

* A **link** is a physical or conceptual connection among objects.

Ex:- Jde Smith Works for Simplex Company

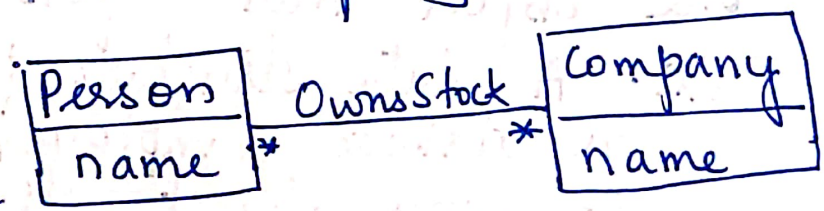
or Person Works for a company.

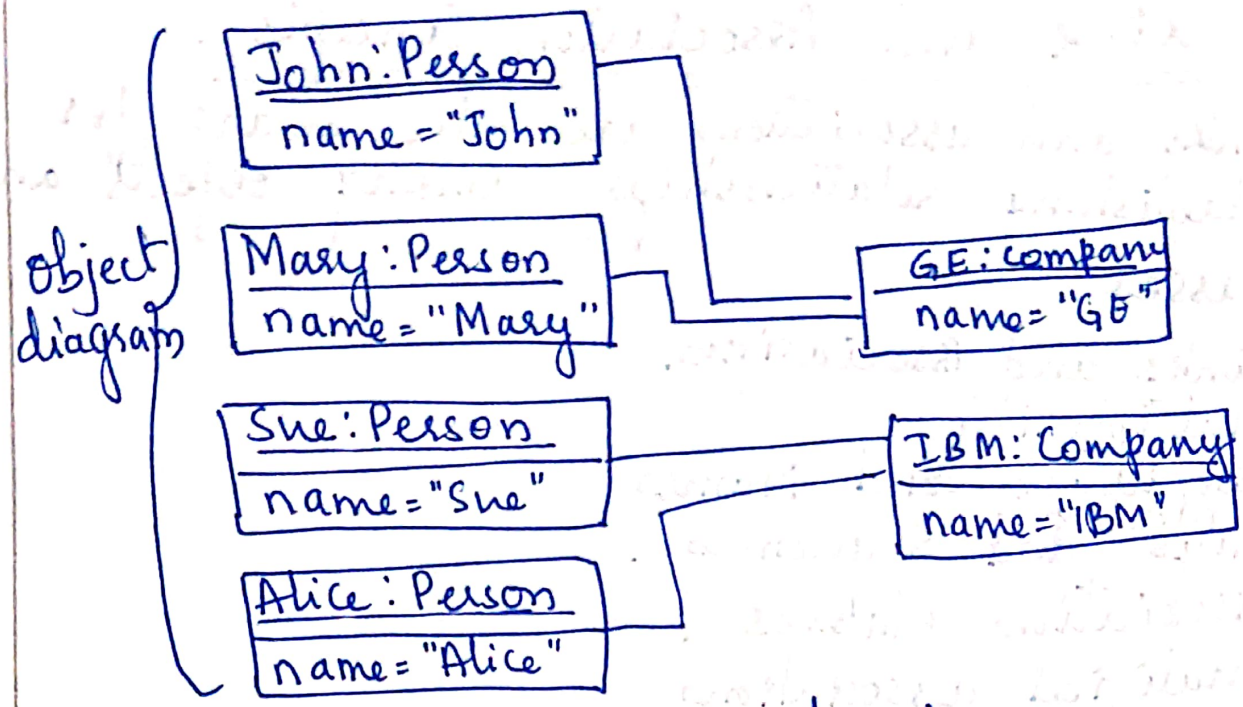
* A link is in an instance of an association

Association: An association is a description of a group of links with common structure and common semantics.

Ex: Person Worksfor a Company

class diagram





Many-to-many association.

Conventions used (UML):

- UML notation for a link is a line between objects, a line may consist of several line segments
  - If the link has the name, it is underlined.
  - Association connects related classes & it is also denoted by a line.
- Show link and association names in italics.

reference: A reference is an attribute in one object that refers to another object.

Ex:- A data structure for Person might contain a attribute employer that refers to a Company object, & a company object might contains an attribute employees that refers to a set of Person objets.

Multiplicity

Multiplicity: specifies the number of instance of one class that may relate to a single instance of an associated class.

→ Multiplicity constrains the number of related objects.

→ Assigns multiplicity to an association end.

UML Conventions:

- UML diagrams explicitly lists multiplicity at the ends of association lines.
- UML specifies multiplicity with an interval "1" (Exactly one), "1...*" (one or more), or "3...5" (three to five), "*" (zero or more).

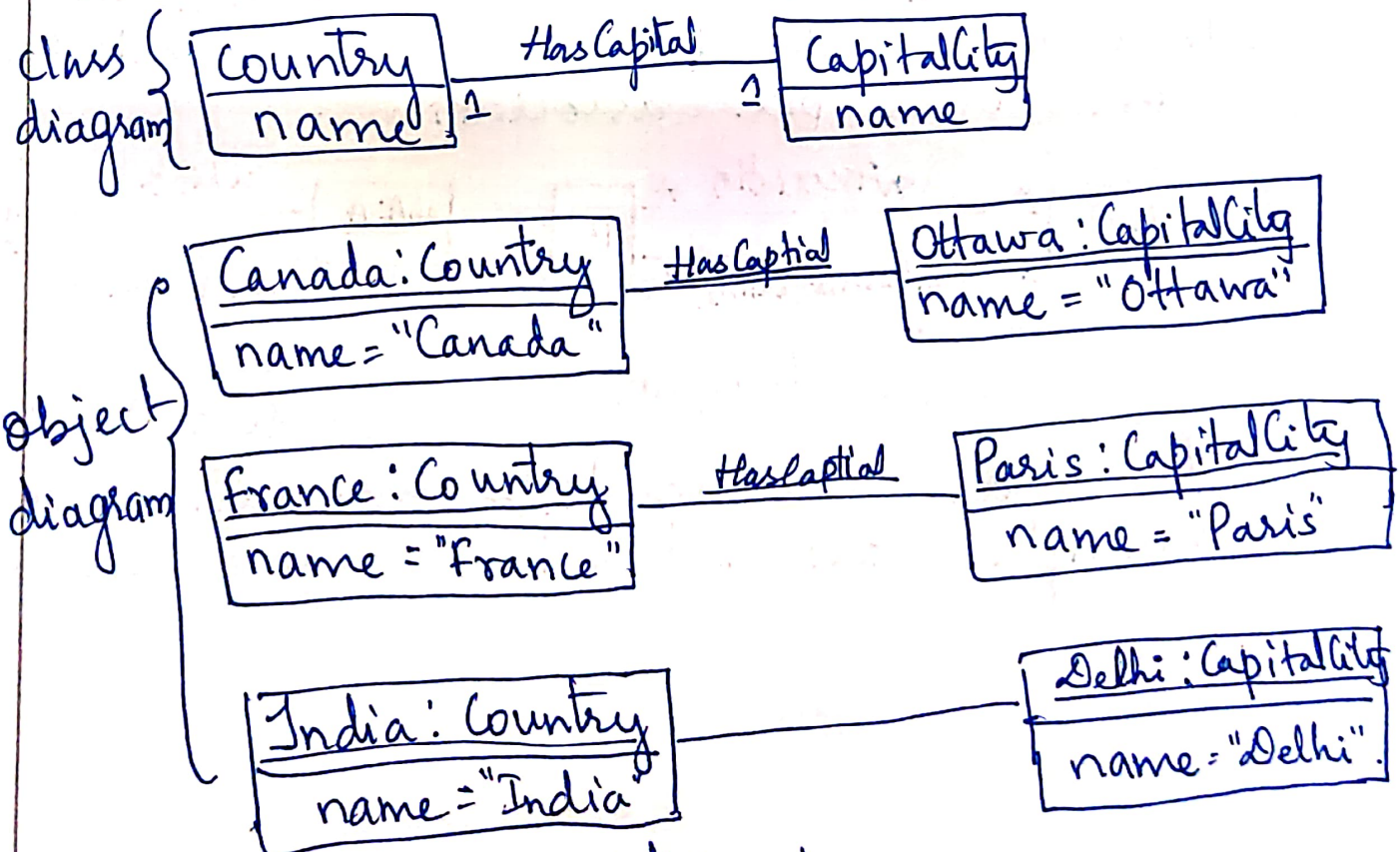


fig: One-to-One association.

Zero or one multiplicity.

→ One workstation may have zero or one console window to receive general error messages.

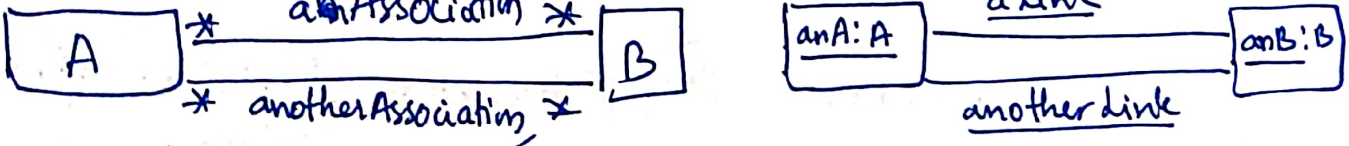
→ A multiplicity of "many" specifies that an object may be associated with multiple objects, but however, for each association there is at most one link between a given pair of objects.

Association Vs link.

Class diagram

object diagram

→ A pair of object can be instantiated at most one per association.



class diagram

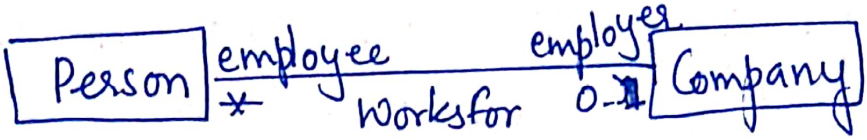
object diagram.

→ You can use multiple associations to model multiple links between the same objects.

Association End Names.

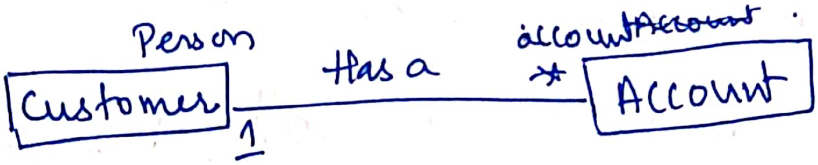
→ Each end of an association can have a name.

for Eg: Person and company participation in association Worksfor.



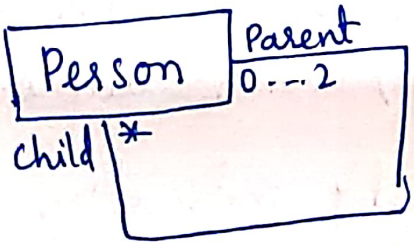
employee	employer
JoeSmith	IBM
MaryBow	Simplex
Petter	TCS

Ex:

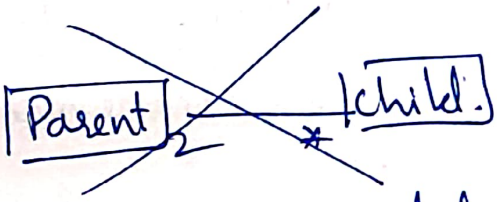


→ Use association end names to model multiple references to the same class as shown below.

Ex:

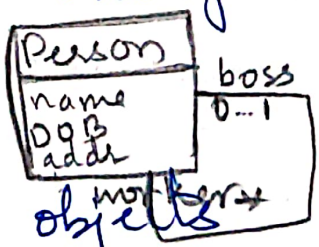


correct model.



wrong model

Ex:-

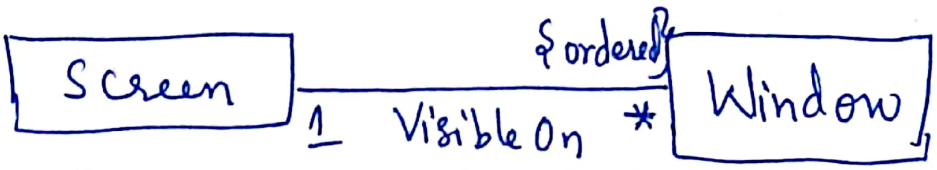


Ordering.

→ Indicate an ordered set of objects

→ for example the below figure shows a workstation screen containing a number of overlapping windows. Each window on a screen occurs at most once. ∴ the windows have an explicit order, so only the top

most window is visible at any point on the screen.



ordering the object for an association end.

→ You can indicate an ordered set of objects by writing "{ordered}" next to the appropriate association end.

Bags and Sequences.

Permits multiple links for a pair of objects

Bag: A bag is a collection of elements (objects) with duplicates allowed.

Sequence: A sequence is an ordered collection of elements (objects) with duplicates allowed.

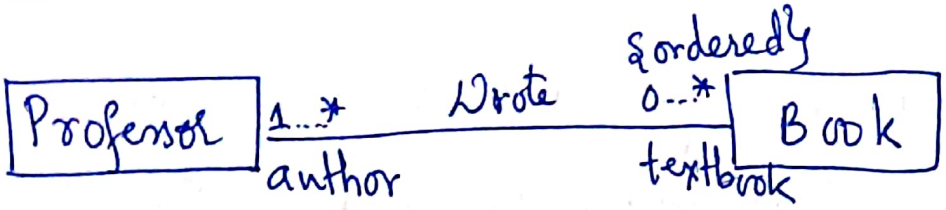
→ Like the {ordered} indication, {bag} and {sequence} are permitted only for binary associations.

Ex-

→ An example of sequence

→ An itinerary (a planned route) is a sequence of airports and the same airport can be visited more than once.

Ex



One or more professor has written zero or more textbooks which are in order.

Association classes.

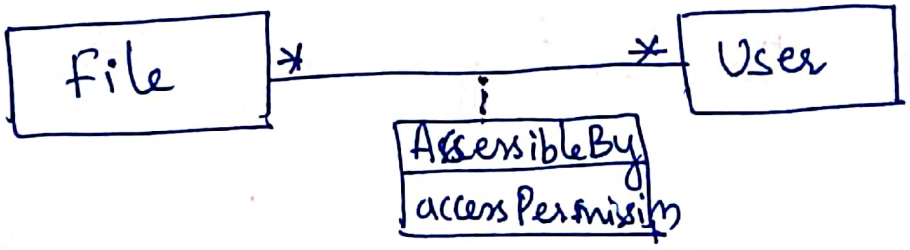
* Objects We know that objects of a class with attributes, Illy We can describe links of an association with attributes.

Association class: An association class is an association that is also a class.

objects → class.
links → association.

Ex:
→ Like a class, an association can have attributes and operations and participate in associations.

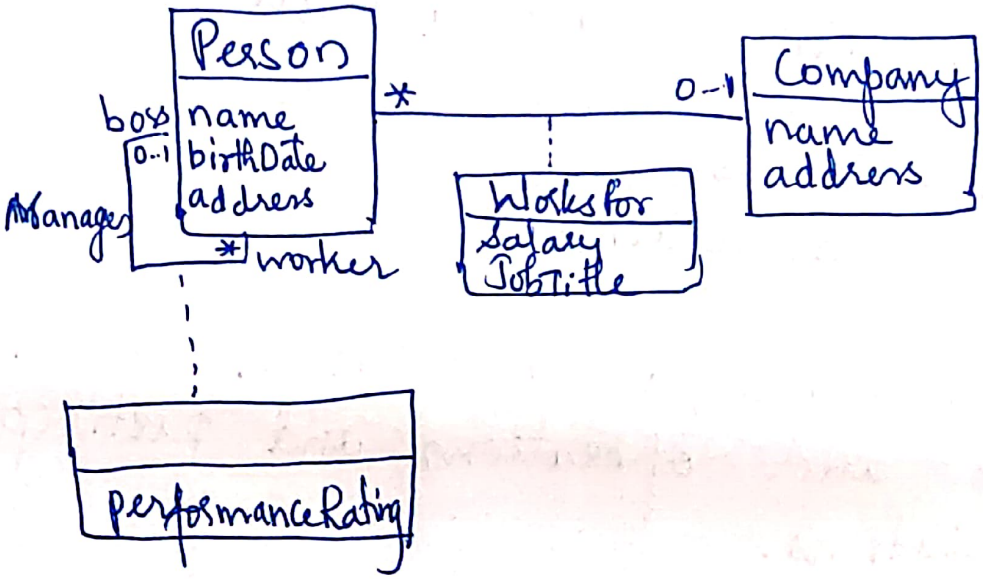
for Ex:- access Permission is an attribute of AccessibleBy association class.



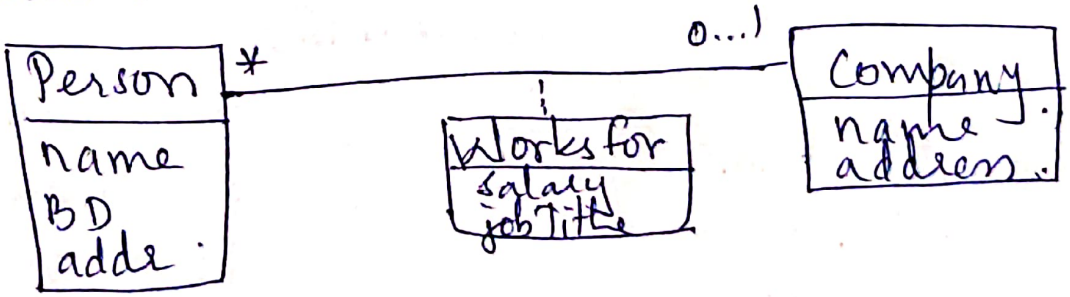
/etc/termcap	read	John Doe
/etc/termcap	read-write	Mary Brown
/usr/doe/.login	read-write	John Doe

→UML notation for association class is a box (a class box) attached to the association by a dashed line.

Ex:- Below fig presents attributes for two one-to-many relationships. Each person working for a company receives a salary and has job title. The boss evaluates the performance of each worker. Attributes may also occur for one-to-one association.



Proper use of association classes: Do not fold attributes of an association into a class.



Qualified Associations.

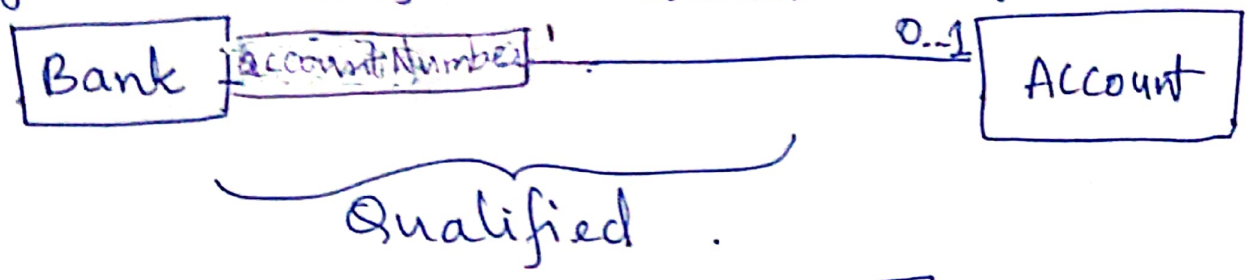
→ A Qualified association is an association in which an attribute called the qualifier disambiguates the objects for a "many" association ends. It is possible to define a qualifiers for one-to-many and many-to-many associations.

→ A qualifier selects among the target objects, reducing the effective multiplicity from * many-to-many association.

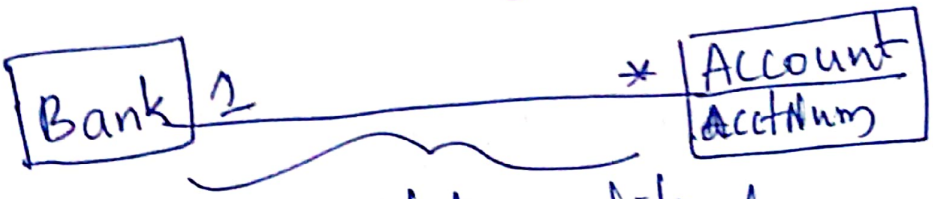
Ex 1:→ Qualifier for n association one to many multiplicity

A bank services multiple accounts. An account belongs to single bank.

Within the context of a bank, the Account Number specifies a unique account. Bank and Account are classes, and Account Number is a qualifier. Qualification reduces effective multiplicity of this association from one-to many to 1-to-1. Fig shows Qualified association.



Qualified.



Not qualified

→ Qualification increases the precision (accuracy) of a model.

UML notation for Qualifier.

→ A notation for a qualifier is a small box on the end of association line near the source class.

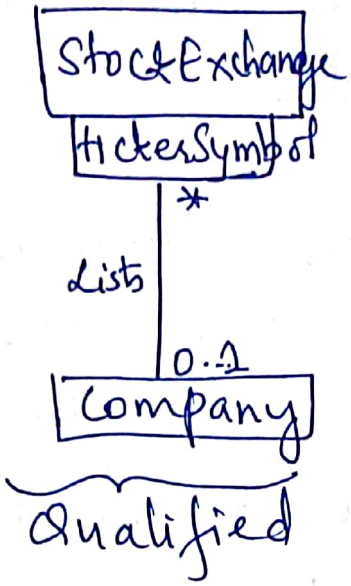
→ The source class + qualifier yields the target class.

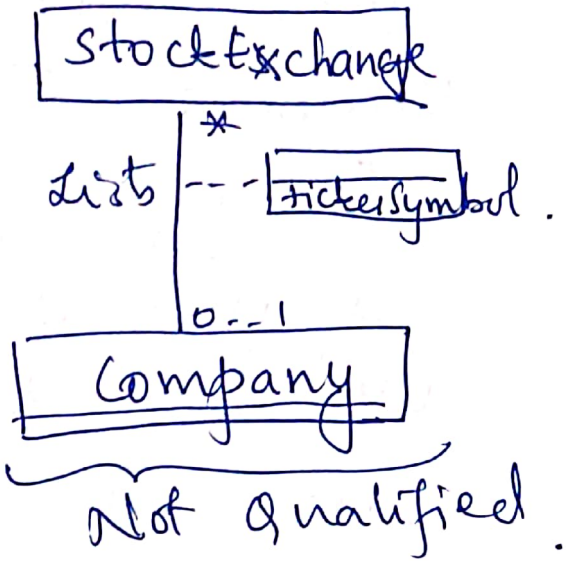
→ In above fig Bank + accountNumber yields an Account.

∴

Ex 2: A stock exchange lists many companies. However it list only one company with a given ticker symbol. A company may be listed on many stock exchanges, possibly different symbols.

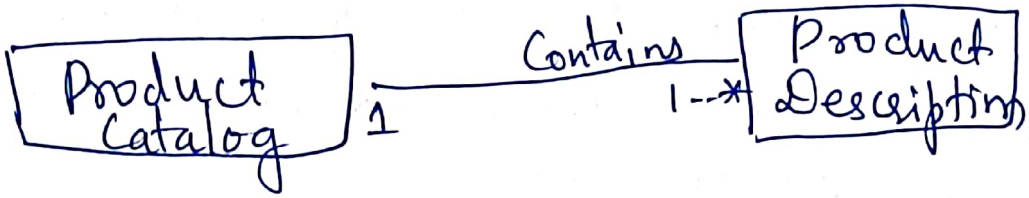
(ticker symbol: English letter symbol representing specific assets listed on stock exchange)

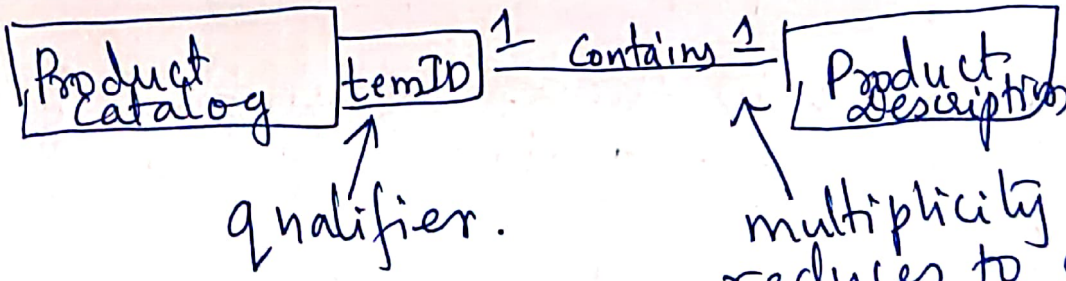




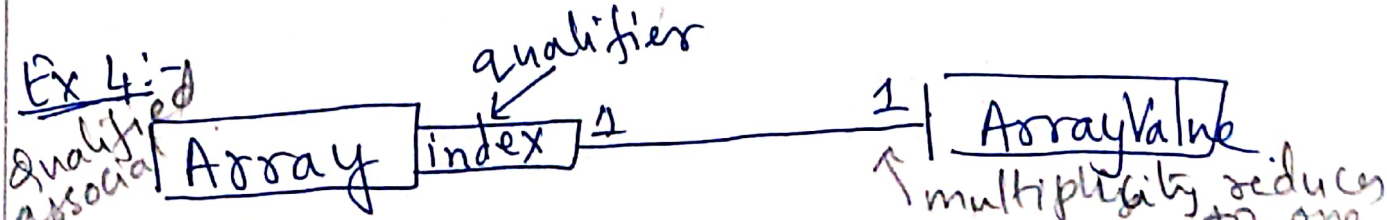
Ex 3:-

a)





Ex 4:-



Generalization and Inheritance.

Generalization: is a relationship between a class (the superclass) and one or more variations of the class (the subclass). Generalization organizes classes by their similarities and differences, structuring the description of objects.

→ Superclass holds common attributes, operations and associations; the subclass adds specific attributes, operations and associations.

→ Each subclass is said to inherit the features of its superclass.

→ There can be multiple levels of generalization.

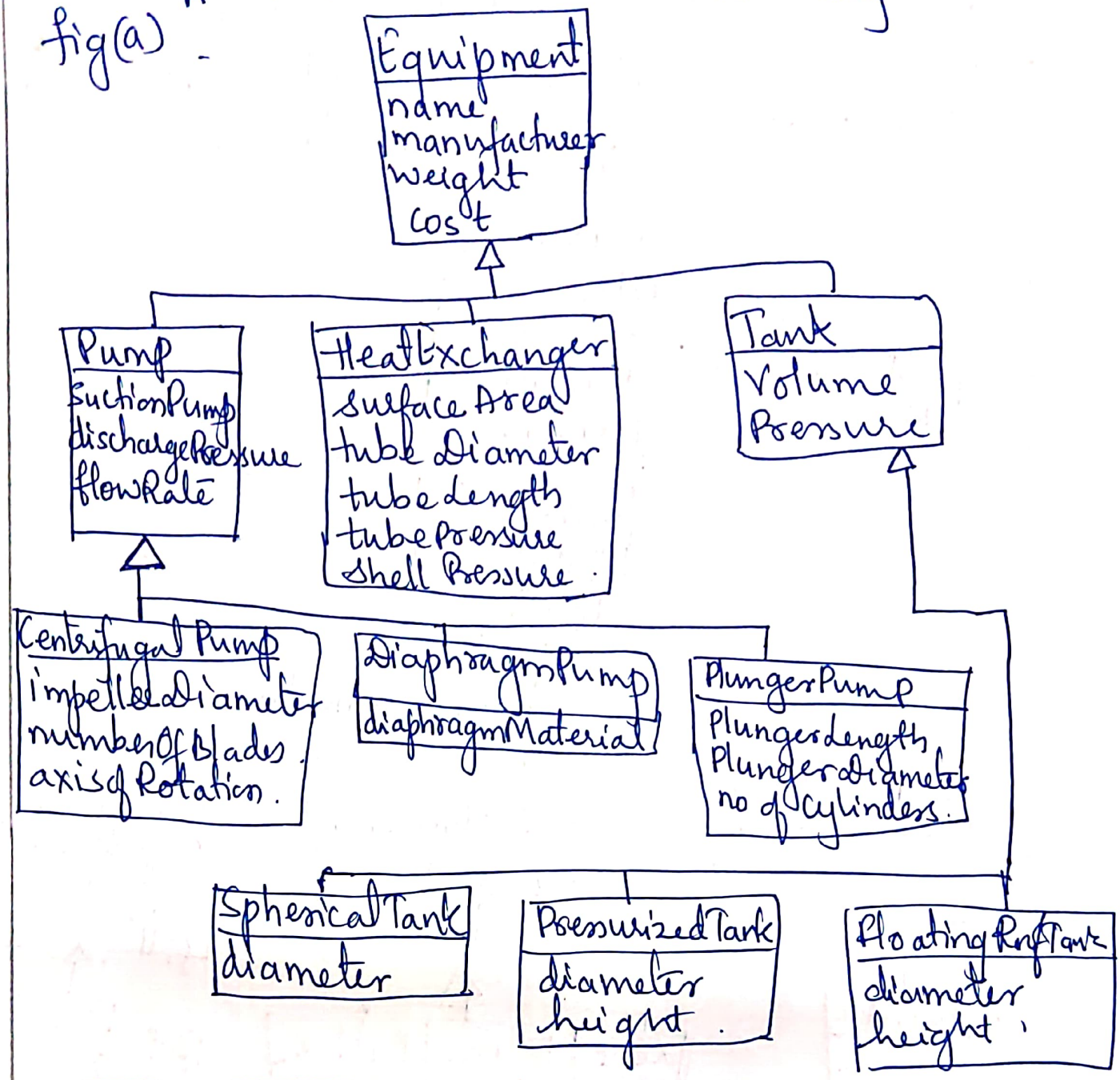
Ex:- fig (a) shows example of generalization for equipment.

Ex:- fig (b) shows example of generalization for graphic figures.

UML convention used:

Use large hollow arrowhead to denote generalization. The arrowhead points to superclass.

fig(a) - A multilevel inheritance hierarchy with instances.

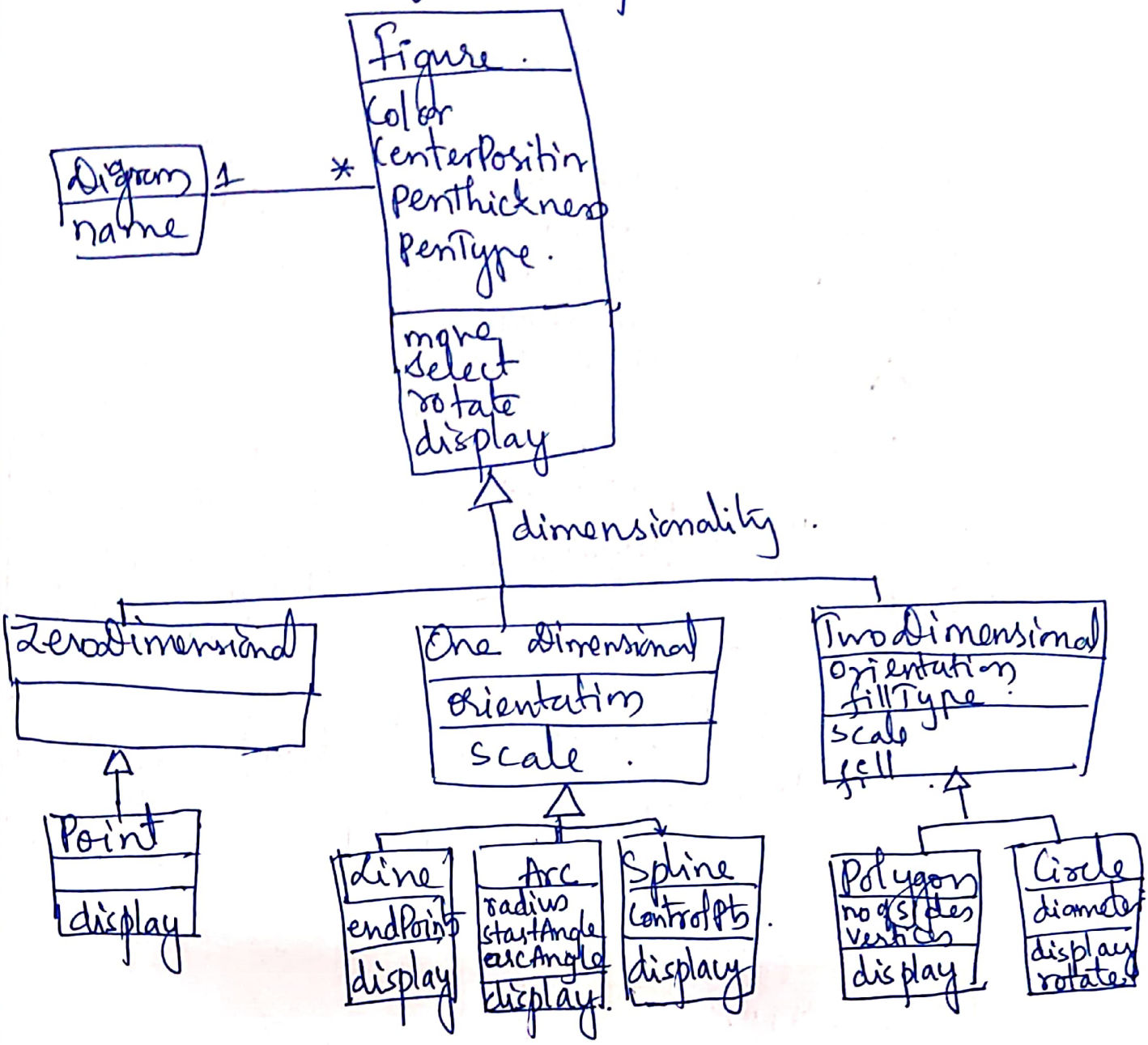


P101 : DiaphragmPump
name = "P101"
manufacturer = "Simplex
weight = 100kg
cost = $5000
Suction Pres = 1.1 atm
discharge Pres = 3.3atm
flowRate = 30l/hr
diaphragmMatt = Teflon

instance

fig(b). Inheritance for graphic figures.

→ Each subclass inherits the attributes, operations, and associations of its superclasses.



→ In above fig, move, select, rotate and display are operations that all subclasses inherit.

→ 'scale' applies to one-dimensional and 2-dimensional fig.

→ "fill" applies only to two dimensional figures.

Use of generalization: Generalization has 3 purposes –

1. To support polymorphism: You can call an operation at the superclass level, and OO language compiler automatically resolves the call to the method that matches the calling object's class.
2. To structure the description of objects: Refers to frame a taxonomy and organizing objects on the basis of their similarities and differences.
3. To enable reuse of code: Reuse is more productive than repeatedly writing code from scratch.

Note: The terms generalization, specialization, and inheritance all refer to aspects of same idea.

Overriding Features.

A subclass may override a superclass feature by defining a feature with the same name. The overriding feature (subclass feature) refines and replaces the overridden feature (superclass feature).

Why overriding feature?

→ To specify behaviours that depends on subclass

→ To tighten the specification of a feature.
→ To improve performance.

In above fig(b) (previous page) each leaf subclass had overridden 'display' feature.