OOS

Basic Concepts of Object-Oriented —

- Classes
- Objects
- Data Abstraction
- Data Encapsulation.
- Inheritence
- Polymorphism
- Message passing

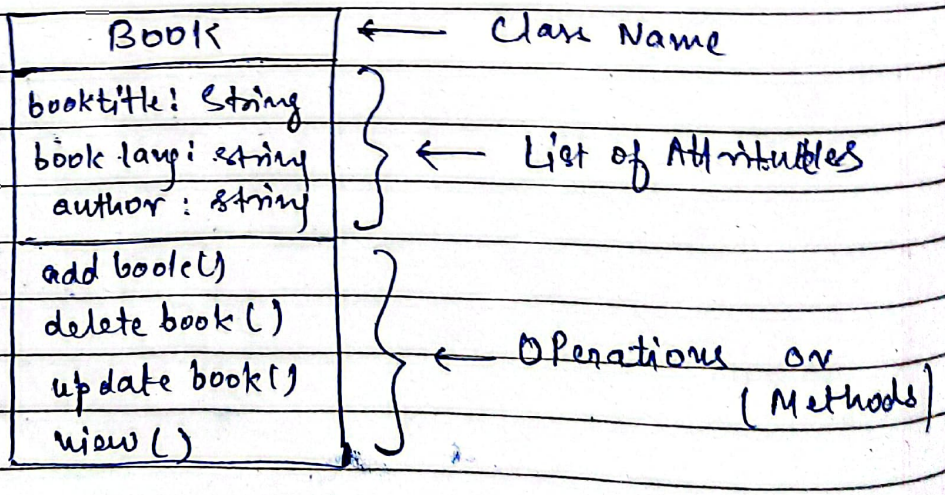
} These are - OOP features

★) **Classes** → These are the user defined data types on which objects are created

or,

Objects with similar properties and methods are grouped together to form a class.

Class Diagram — UML Notation.



Objects — (& their properties)

→ Run time entities that may represent a person, place, bank account or any item.

→ Each object contains data & code to manipulate the data.

→ Data represents attributes of that object & function represents the behaviour of object

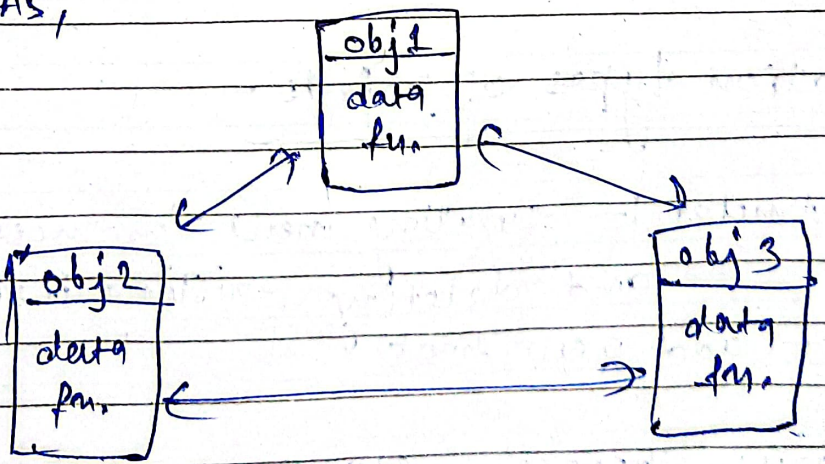
So,

objects < Attributes → data members
functions → code.

→ Object take up space in memory & have an associated add like structure in C.

→ When a program is executed the object interact by sending messages to one another.

As,



Ques ->) Characteristics of Objects -

-> All objects have attributes
Ex -) Student : Name
Roll no.
Address
Year
Department

-> All objects have a state
Ex -) Ticket : reserved, waiting list,
) Student : Present, Absent.

-> All objects have set of OPERATIONS which can be performed on them.
Operations determine object behaviour.

Ex -) Admit Student
) Cancle ticket

Ques ->) Operation types of objects -

-> Constructor :- creating new instances of class and deleting existing instance of cla
Ex - add new book

-> Query :- Accessing state without changing st , has no side effects.
Ex - find book name.

-> Update :- changes values of one or more attribu
Ex - change ~~author~~ book name
Implementation of operation on objects called methods

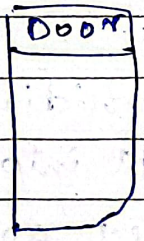
Ques 1) Difference between class & object

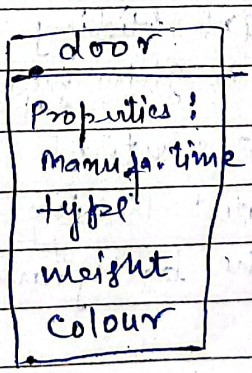
Class	Object
→ class is a datatype	→ object is an instance of class
→ It generates objects.	→ It gives life to class
→ Does not occupy memory location	→ It occupies memory location.
→ Cannot be manipulated.	→ Can be manipulated.

☆ Data Abstraction

It refers to the act of representing essential features without including the background details or explanations.

Ex- Suppose a class Door.





Here, we are not giving the big details of the door.

∴ Class uses the concept of data Abstraction
∴ they are also known as Abstract data types.

4) Encapsulation :-

→ The wrapping of data and functions into a single unit is known as encapsulation.

→ It is also known as information hiding concept.

As,
The data is not ~~accessed~~ accessible to the outside world and only those fn. which are wrapped in the class can access it.

→ Information hiding allows improvement of methods used by objects. without affecting other parts of a system.

Ques-) Difference b/w Abstraction & Encapsulation

Abstraction	Encapsulation
→ ~~Hiding essential~~ Hiding unwanted data & giving relevant data	
or → Representing essential features without ~~hiding~~ including the background details.	→ Means hiding the code and data into single unit to protect data from out world.
→ What the object does	→ How the object does.
→ Outer layout used in terms of design.	→ Inner layout, used in terms of implementation
Ex- outlook of mbile phone like, keypads, screen.	Ex- Inner implementation details of mbile phone As, how screen & keypad connected and work.

Date :
Page :

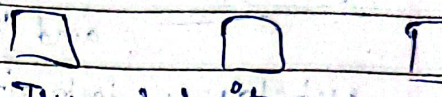
Inheritence —

base class Super class

Parent class

derived class

1) Single level
2) Multilevel "
3) Hierarchical "



Child class
Sub class

Time inherit property from super class.

→ New classes are created from current classes by using the idea of inheritence.

diff^n

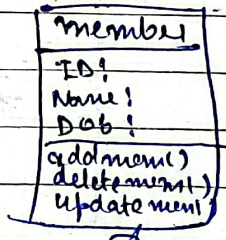
→ Each derived class inherits the attributes of its base class and this processes is known as inheritence

or,

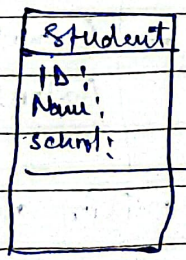
→ Inheritence is the process by which object of one class acquires the properties of object of another class.

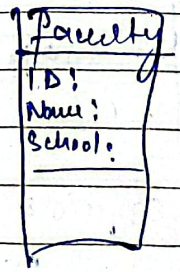
Ex →

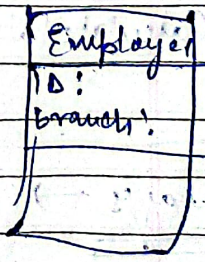
Base class →



derived class







Time inherits the properties of base class.

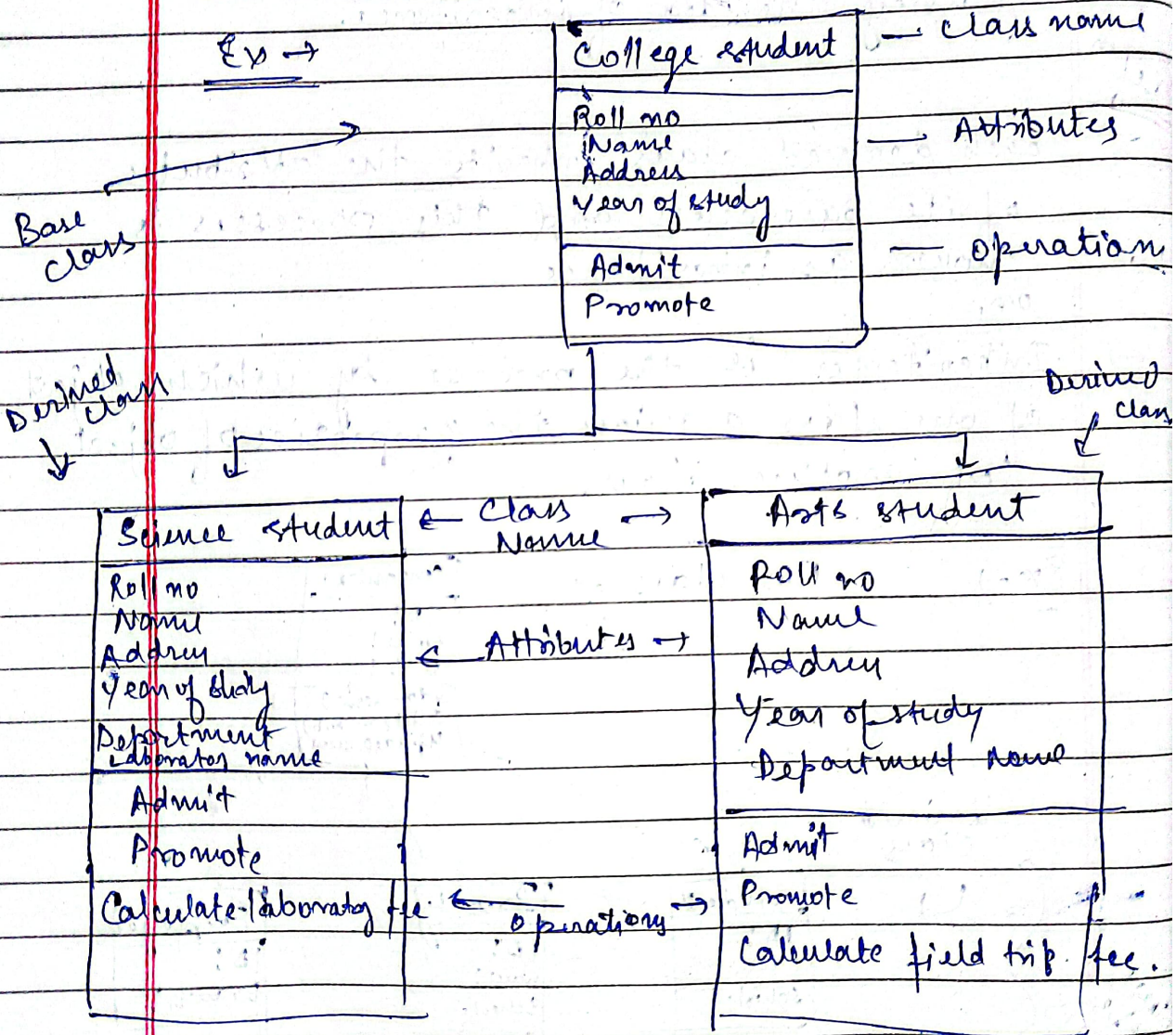
→ Reusability → we can add additional features to an existing class without modifying it.

→ Inheritence allows both generalization and specialization in modelling.

Specialisation — given student class, art students and science student are two subclasses, subclasses inherit properties of parent and in addition may have their own special attributes and operations.

Generalisation

Ex →



Advantages → a) It expresses commonality among classes/obj
b) Allows code reusability
c) highlights relationships.
d) help in code organisation

Date :
Page :

☆) Polymorphism :-

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritence.

→ In real world, the same operations may have different meaning in diffrent situations

Ex→ Let Base class called Aimal that has method called animalsound().
Derived class of Animals could be Dogs, Cats, and Birds, they also have their ~~fun~~ own implementation of animal sound.

Ex→ operation of Addition.
3 + 4 = 7 // Adding
Rama + Krishna = RamaKrishna // Concate-nating

This are concepts of Polymorphism :-

a) Operator Overloading - The process of making an operator to exhibit diff. behaviour in diff. instances.

b) Function Overloading - Using a single fn. name to perform diff. types of tasks.

" It involves designing the object classes & the rel^n b/w the classes.

A) Object Oriented Design —

. This OOD has 2 stages —

Conceptual Design
(high-level design)
• use class and interaction dig.

Detailed Design.
(low-level design)
• use state m/c dig.

(1) Conceptual design —

• recognize all the classes required to build the system.
• responsibilities are allocated to each class.
• In high level design, we use
the class diagram => analyze the rel^n b/w class
interaction " => shows the flow of event

(2) Detailed design —

• All & operations are allocated to each class based on their interaction dig.

• The state m/c diag. is developed to explain the next. Details of design is called as low level design.

Terms in OOD
• Instance Var. → Data, all state that are specific to an instance
• Instance method → Operations are specific to
• Class variable → Data are not specific to an
• class method → Operations are not specific
• Constructor → Special method that create & initialize (for a class.

Object Oriented Modelling

→ The object oriented modelling is a way of constructor visual models based on real world objects.

→ Modelling helps in understanding the problems develop proper documents, and producing well designed program.

→ Modelling producing well understood requirements, robust designs, high quality & maintainable systems.

Object oriented Modelling has 4 stages

(1) Analysis (2) System Design
(3) Object design (4) Implementation or Model.

(1) Analysis ⇒ Problem statement
Three model present ← Object Model
— Dynamic "
— Functional "

Object Model captures Static object
Dynamic " " behaviour "
Functional " represents the functional object of sys.

(2) System design ⇒ high level design

(3) Object design ⇒ objects in details

(4) Implementation ⇒ Objects are implemented i.e. code.

A) Importance of Model

For this we have to know why Model?
So basically,

→ to Analyse the problem - domain.

Advantages → •) Simplify reality
•) Capture requirements
•) Visualize the system in its entirety
•) Specify the structure / behaviour of the system.

→ Design the solution
•) document the sol'n in terms of its structure, behaviour etc.

A) Principles of Model.

(1) Choose your model ~~first~~ Well -
→ The choice of model profoundly impacts the analysis of the problem & the design of the sol'n.

(2) Every Model may be expressed at diff. levels of precision -
→ The same model can be scale up (or down) to different granularities.

(3) The best models are connected to reality -
→ Simplify the model, but don't hide imp. details.

(4) No single model is sufficient →
→ A set of model is needed to solve any non-trivial system.

Process of Object Modelling -

1) recognize obj and grouped into classes
↓
2) relationship b/w the classes
↓
3) User object model diagram is generated
↓
4) Define Attributes of user obj.
↓
5) Define the operations need to perform on classes.

Process of Object Oriented design -

1) A solution design is created from requirement or system sequence diagram.

2) Objects are identified and grouped into classes on behalf of similarity in attribute characteristics.

3) Class hierarchy and rel^n among them is defined.

4) Application framework is defined.

OOad

↓

☆) Object Oriented Methodologies:-

(1) Obj. oriented analysis by Coad and Yourdon
(2) Obj. oriented design by Grady booch.
(3) Obj. oriented Modeling techniques by James Rumbaugh
(4) Object Oriented s/w Engineering (OOSE) by Jacobson

In detail —

(1) Coad and Yordon Methodology → (Based on OO Analysis)

use for

— Identification of Classes and Objects.

~~— Identification of structures.~~

~~— Def" of subject, attributes & methods~~

investigate app domain and sys environment.

— Identification of structure

is-a rel", whole-part rel"

— Identification of subject, attributes & methods

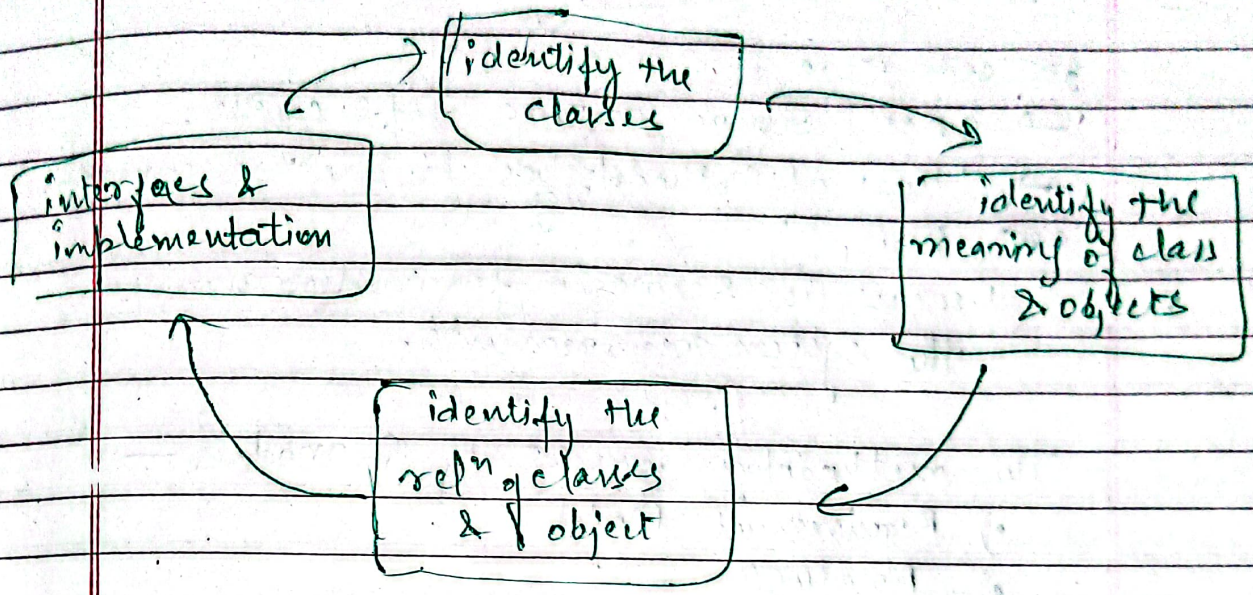
(2) Booch Methodology — (Based on OO design)

Approaches to s/w improvement process.
It can be done in two diff. points

(1) Micro improvement (2) Macro improvement

Date :
Page :

① Micro development process — It is like a S/W Architecture



② Macro - Development Process -

It ~~predicting~~ has to identify
→ Basic needs of S/W (conceptualization)
→ Analysis ~~(It has to develop)~~
→ Design
→ Development cycle of the details from 1 end to another
→ Maintenance i.e. Administration to the post development

③ Object Modelling techniques (OMT) (by James Rambaugh)

Rambaugh developed a technique, focus on Analysis, design and implementation of the system.

OMT consists of 4 phases — Explain briefly.
→ Analysis
→ System design
→ Object design
→ Implementation

(4) Jacobson Methodology (Also known as OOSE)

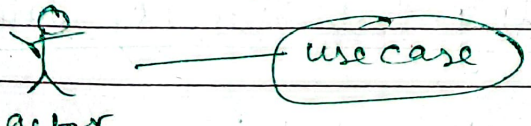
→ OOSE covers entire life cycle.
→ Heart of Methodologies is usecase concept.
here,
Usecase is scenario for understanding the system requirements.

This methodology consist of 5 models —
- Requirement Model
- Analysis "
- Design "
- Implementation "
- Test "

1) Requirement Modelling —

→ UML is a standard lang. for OO s/w design.
→ Use Case Modeling
defines s/w fn. requirement in term of use case and actors.

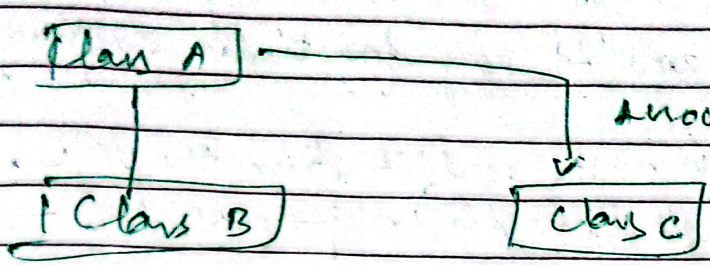
UML notation for usecase digg.



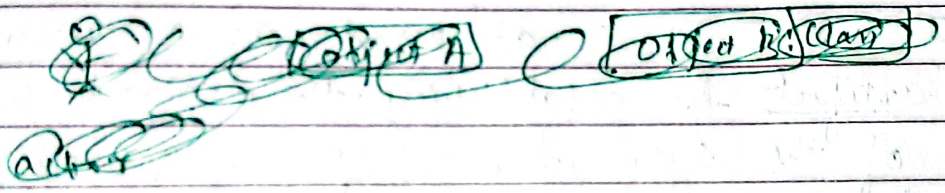
2) Analysis Modeling

Consists of static & dynamic Model
↓ ↓
View of sys that does not change with time
View of sys. that changes with time.

→ Static modeling - Define structural rel^n b/w classes

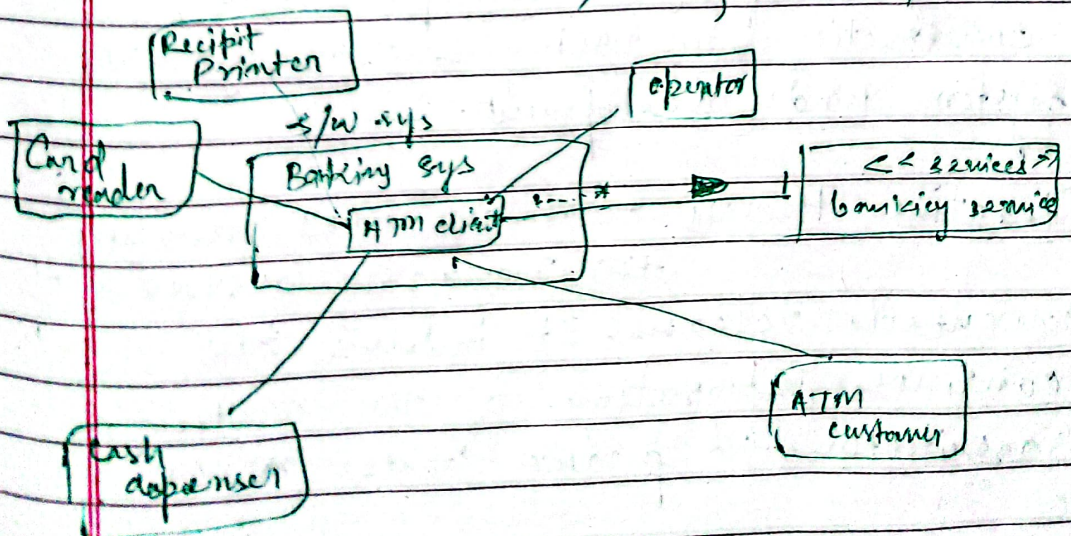


→ Dynamic Modelig → Define sequence of object communicating with each other using sequence dig.



•) Design Modelling → Develops overall s/w architecture

•) Design OO s/w ~~des~~ architecture



Software Development Life cycle (SDLC)

The UML is largely process-independent.
It is not tied to any particular software Development life cycle.

⇒ To get the most benefit from the UML, it should consider a process i.e. —

- Use case driven
- Architecture - Centric
- Iterative & incremental

- Use case driven — means that use cases are used as a primary artifact for establishing the desired behaviour of the sys., for verifying & validating the system architecture for testing and for communicating among the stakeholders of the project

- Architecture Centric :- means that a systems architecture is used as a primary artifact for conceptualizing, constructing, managing & evolving the system under development.

- Iterative Process - is one that involves managing a stream of executable releases. It involves the continous integration of the system's architecture to produce these releases.

This use case driven, architecture - centric and iterative process can be broken into phases.

Phase is a span of time b/w two major milestone of the process.

There are 4 phases in SDLC →

a) Inception → ~~(starting phase)~~ (Idea)
b) Elaboration → (~~Creation team, sys~~ list)
c) Construction → complete
d) Transition → maintainance.

(1) Inception — is the first phase of the process, when the idea for the development is brought up to the point of being at least internally sufficiently well-played to warrant entering into the elaboration phase

(2) Elaboration — During elaboration phase, the project team is expected to capture a healthy majority of the system reqn ~~the primary goal is to~~

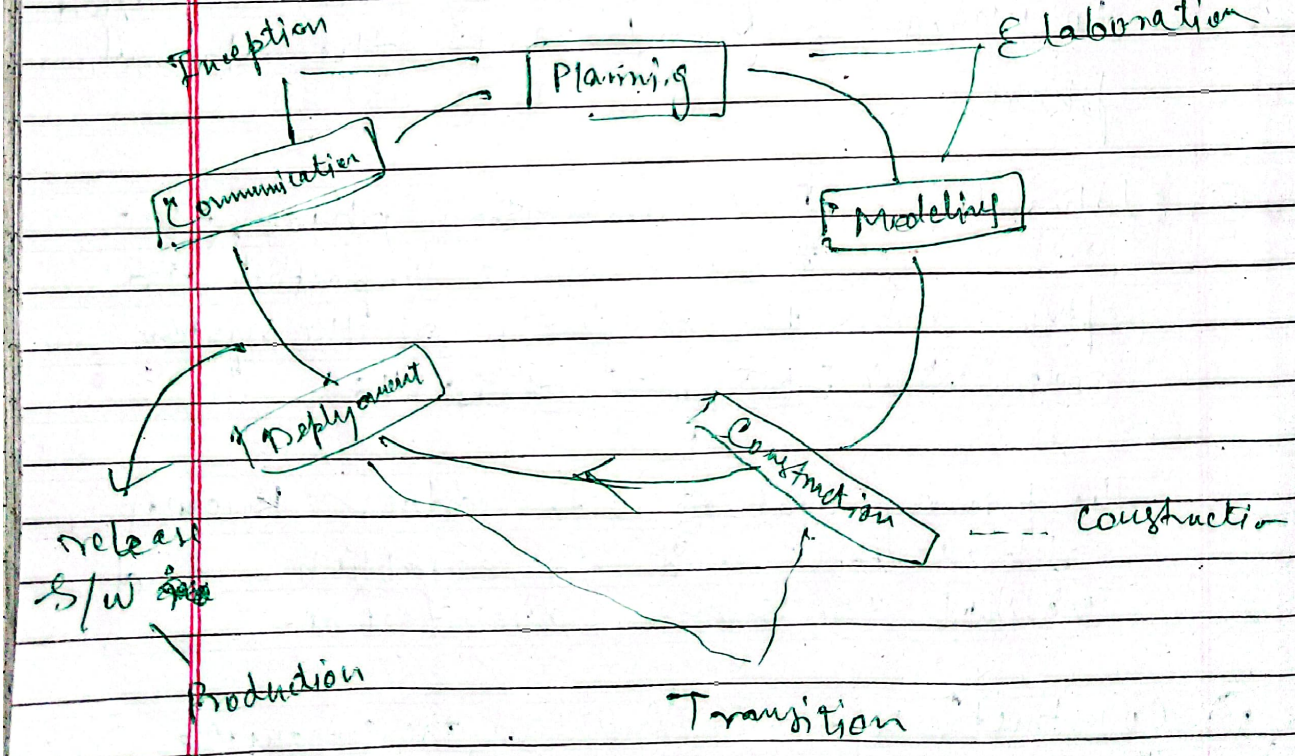
The primary goal is to address known risk factors and to establish and validate the system architecture.

(3) Construction — It involves the executable arch. baseline into a complete ~~software~~ working system.
The goal is to complete all reqⁿ, analysis and design.

(4) Transition — is the fourth phase of the process, when the s/w is turned into the hands of the user community.
During this phase, the system is continuously improved, bugs and eradicated and features that didn't make an earlier release are added.

☆) Introduction to Unified Process —

It has 5 phases — Inception, Elaboration, Construction, ~~Production~~, Transition, Production.



A) UML (~~Conceptual Model~~ Unified Modeling Language)

→ UML may be used to visualize, specify, construct and document the artifacts of a software intensive system.

Conceptual model of UML

To understand UML, You need to form a conceptual model of language, and this requires 3 major elements —

1. UML's Basic Building Block
2. The rules
3. Mechanism

1. Building Blocks of UML —

(i) Things
(ii) Relationship
(iii) Diagrams.

(i) Things — are the most imp. building block of UML.

These are mostly static parts of the model, representing elements that are either conceptual or physical.

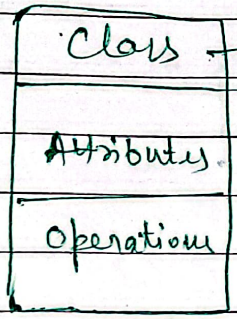
There are 4 types of Things in UML —

a) Structural things
b) Behavioural things
c) Grouping things
d) Annotational things

a) Structural things — define the static part of the model. They represent the physical & conceptual elements.

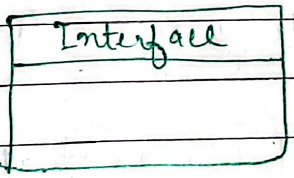
Following are brief description of structural things —

(i) Class :
Class represent a set of object having similar responsibility.

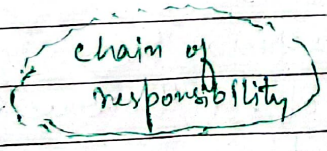


→ Write class name here in bold.

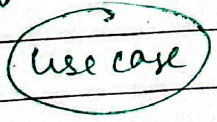
(ii) Interface :
defines a set of operation which specify the responsibility of the class.



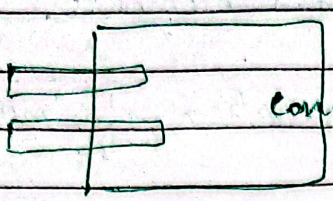
(iii) Collaboration :
Collaboration define an interaction b/w elements



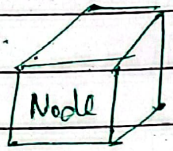
(iv) Usecase – represent a set of actions performs by a system for specific goal



(v) Component :- It describes the physical part of the system



(vi) Node :-
A node can be defined as a physical element that exist at runtime.



b) Behavioural things —
These are dynamic part of UML models.
It consist of two kind —
(i) Interaction → display message
(ii) State m/c → Waiting

Interaction — Behaviour that consist of a group of message exchange among elements to accomplish a specific task.

State m/c — It defines the sequence of state an object throws in response to events, events are external factor responsible for state.

c) Grouping things —

It can be defined as a mechanism to group elements of a UML model together.
There is only one grouping things available i.e. Package.

(i) Package: is only one grouping thing available for gathering structural & behavioural things.

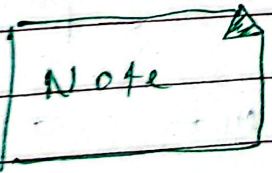
d) Annotational things —

It can be defined as a mechanism to capture remarks, description & comments of UML model elements.

Explanatory part of UML model

There is only one kind of Annotational thing i.e. Note.

Note Represented By



(iii) Relationship —

Relⁿ is another most imp. building block of UML. It shows how the elements are associated with each other & this association describe the functionality of an application.

There are 4 kinds of Relⁿ avl. —

1) Dependency
2) Association
3) Generalization
4) Realization

1) Dependency — It is the relⁿ b/w two things in which change in one element affects the other.

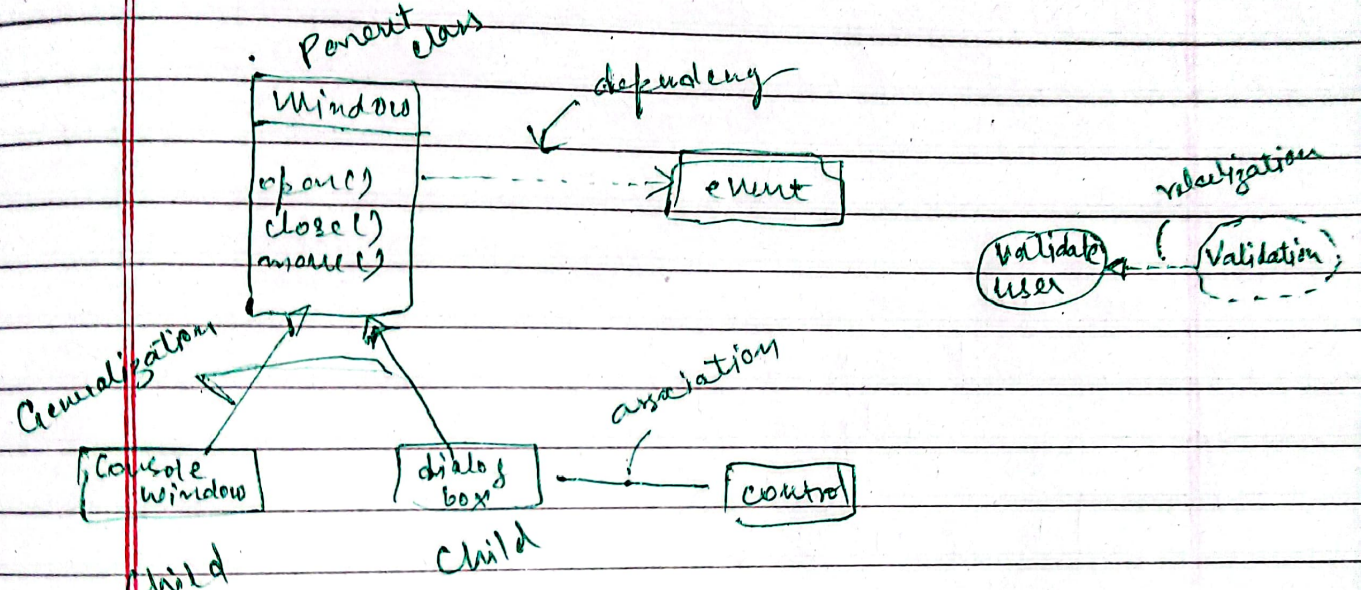
2) Association — It is basically a set of links that connect the elements of a UML model it also describes how many objects are taking part in that relⁿ

3) Generalization —

It can be defined as a relⁿ which connects a specialized element with a generalized element. It basically describes the ~~inheri~~ inheritence relⁿship in the world of obj.

4) Realization – It can be defined as a relⁿ in which two elements are connected one element describe some responsibility which is not implemented and the other one implements them. This relⁿ exists in case of interface.

Ex →



Modelling Vs Design