Rohan Nyati

Object Oriented Analysis & Design

Lab

DFD (Context diagram)

This is the virtual representation of flow of information

Components :-

1) External Entity (Input to the system) → Librarian → Consumes data produced by the system

2) Process → Search-book → circle

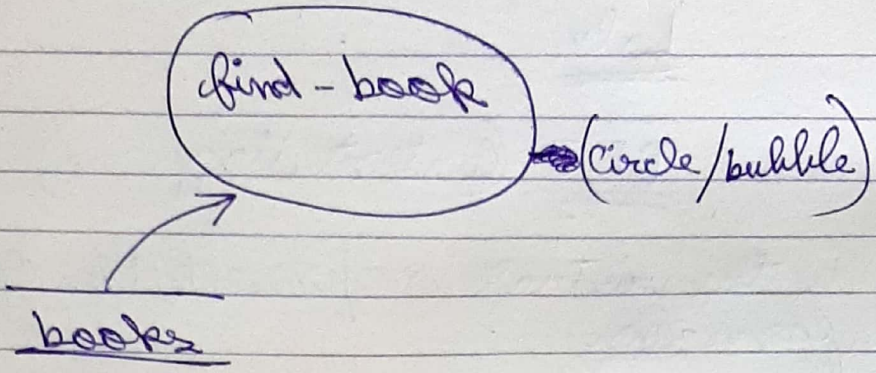
3) Data Flow Symbol → Book_name
   (1) Data produced by the external entities / entity and used by a function or process.
   (2) Data produced by the process & consumed by the another user / external entity

4) Data Store → Represent a logical file
   a logical file can be :
   (1) Data structure
   (2) Physical Store (like disc) → A physical file on disc

Books

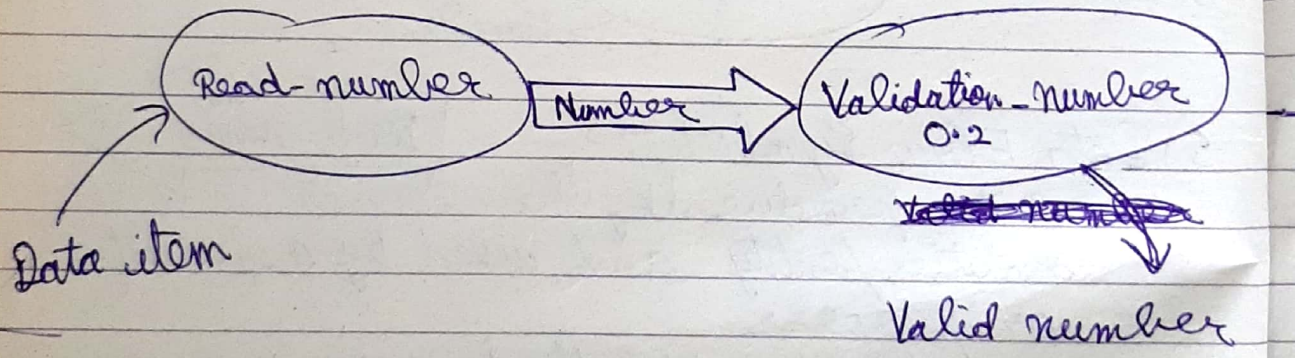
- Each data store is connected to a process.

• Each data store is connected to a process

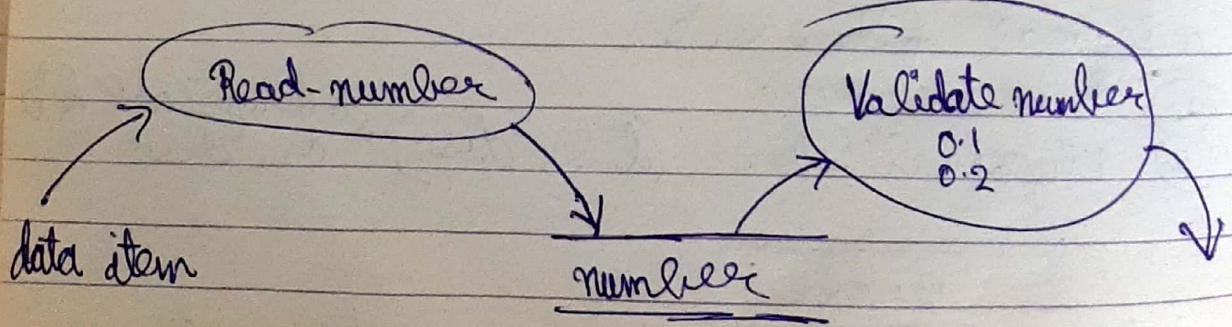


Synchronous & Asynchronous operation

* If 2 processes are connected directly without data store, by a data flow arrow so it is called "Synchronous".



* If 2 processes are connected by a data store so, it is called Asynchronous

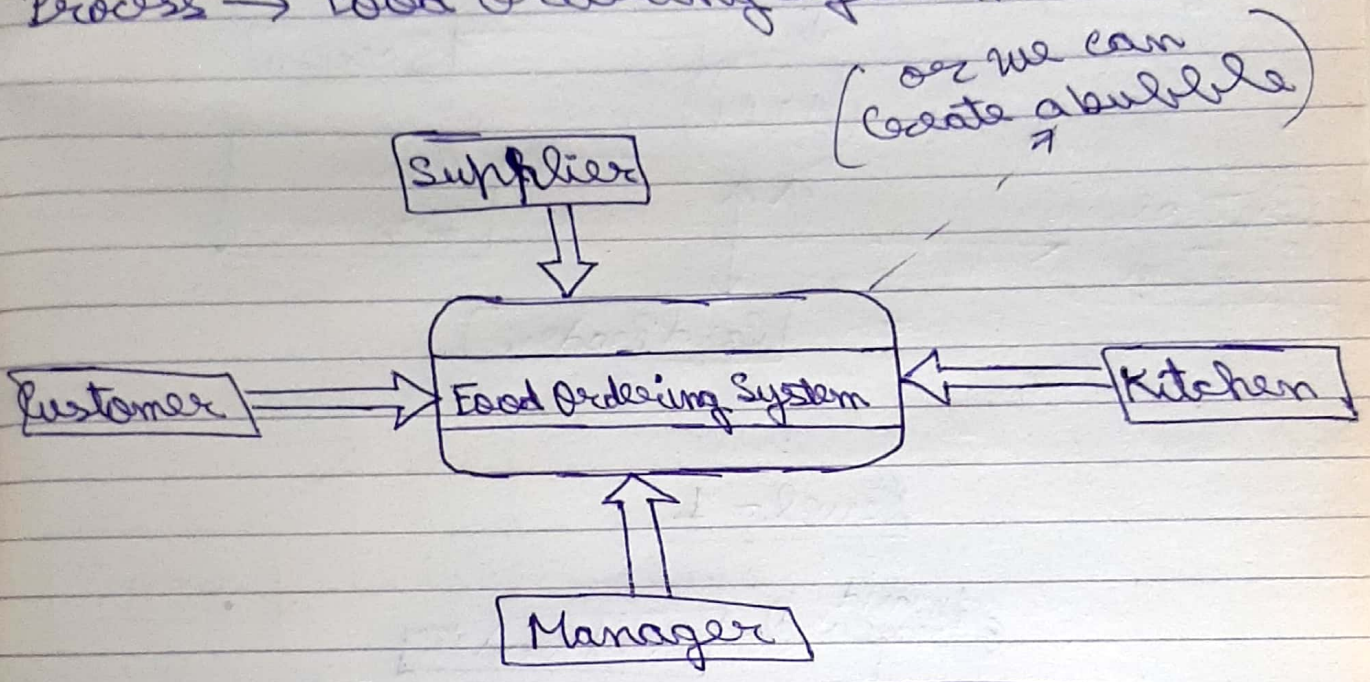


Online Food Order System

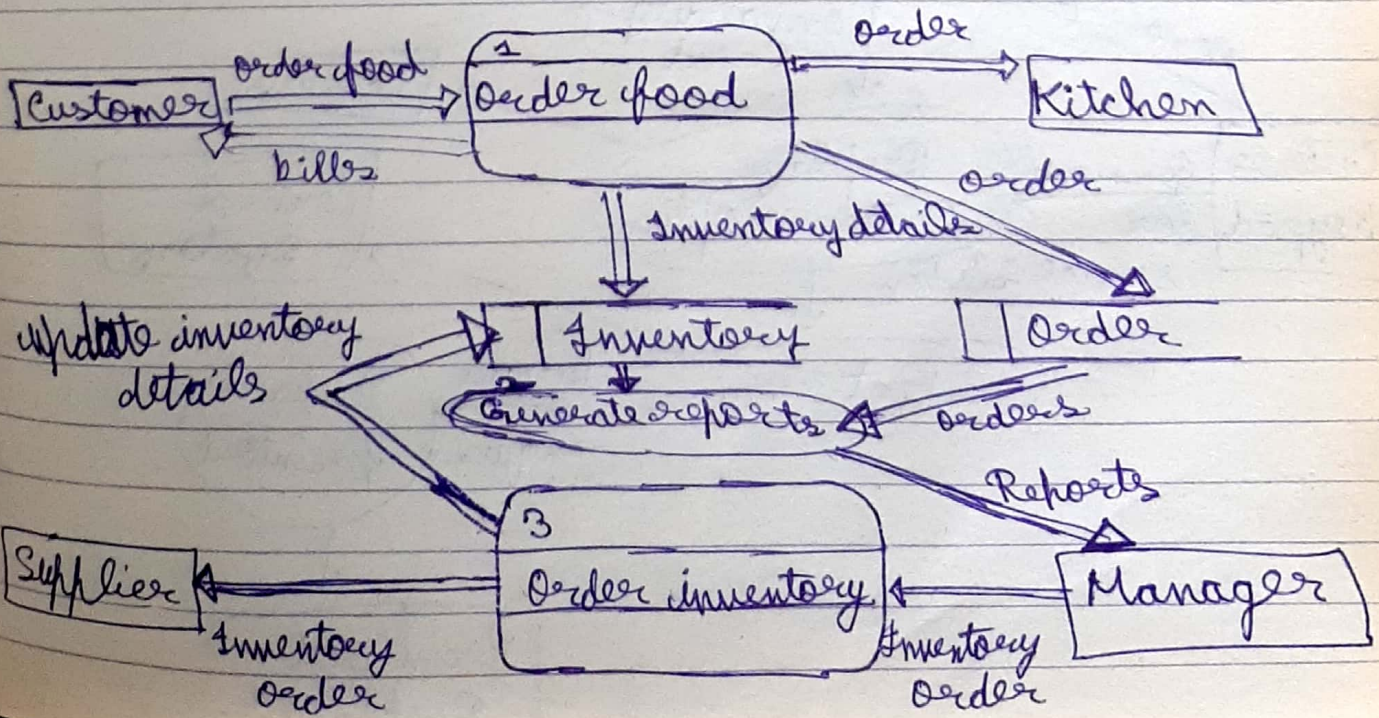
Level 0 diagram (for only one process) (most abstract level)

Entities → Customer, Manager, Kitchen, Supplier

Process → Food Ordering System



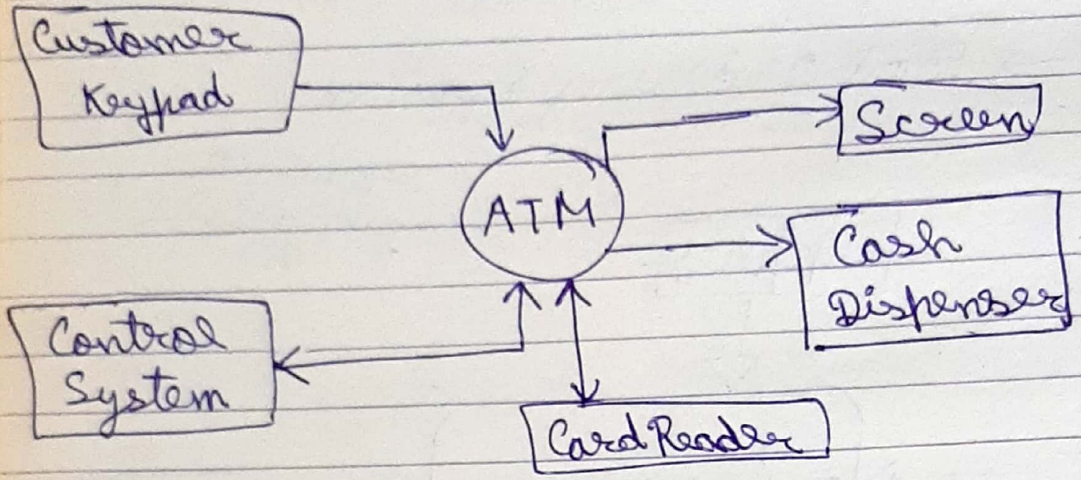
Level-1 diagram (3 process + one data store)



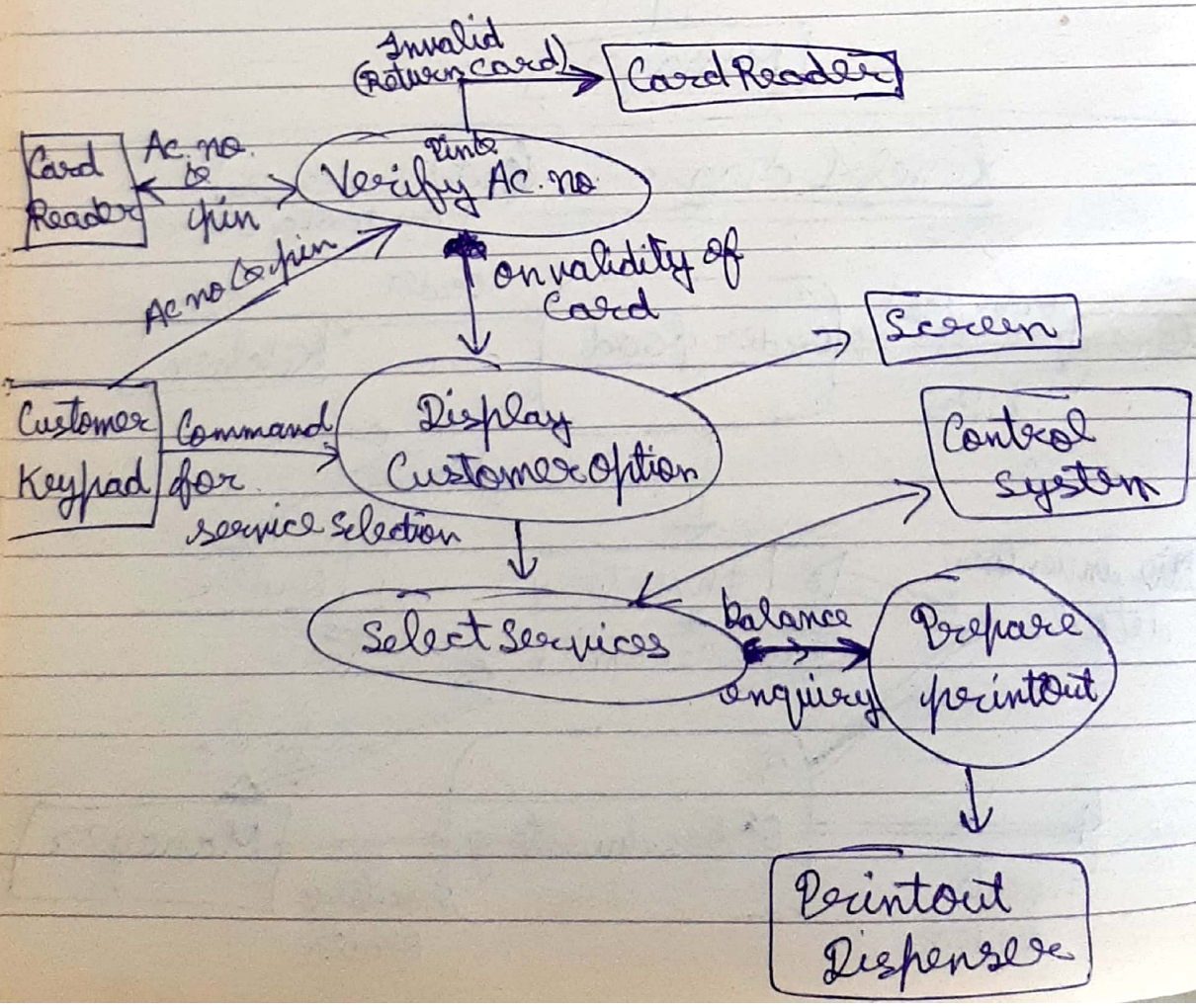
① For ATM Counter → For DFD

② ER → Dia → E-Commerse management

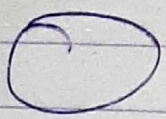
Level-0

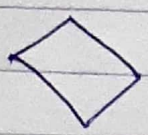


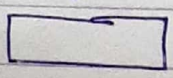
Level-1



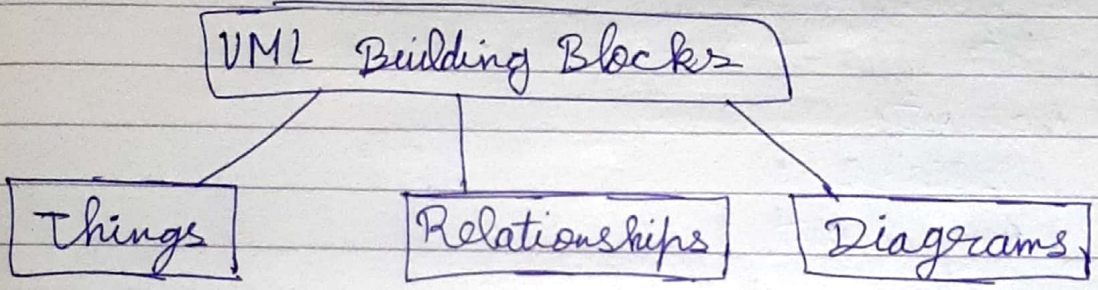
E R - diag.

 → Attribute

 → Relationship

 → Entity

Building Blocks of the UML



1) Things

(1) Structural → (i) Class
Collaboration
Use case
Active Classes
Components
Nodes

(2) Behavioral → Interaction
State Machine

(3) Grouping → Packages

(4) Annotational → Notes

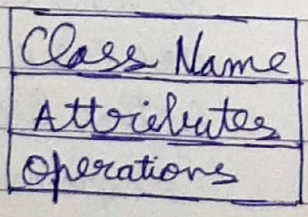
2) Relationships

1. Generalization
2. Dependency
3. Association

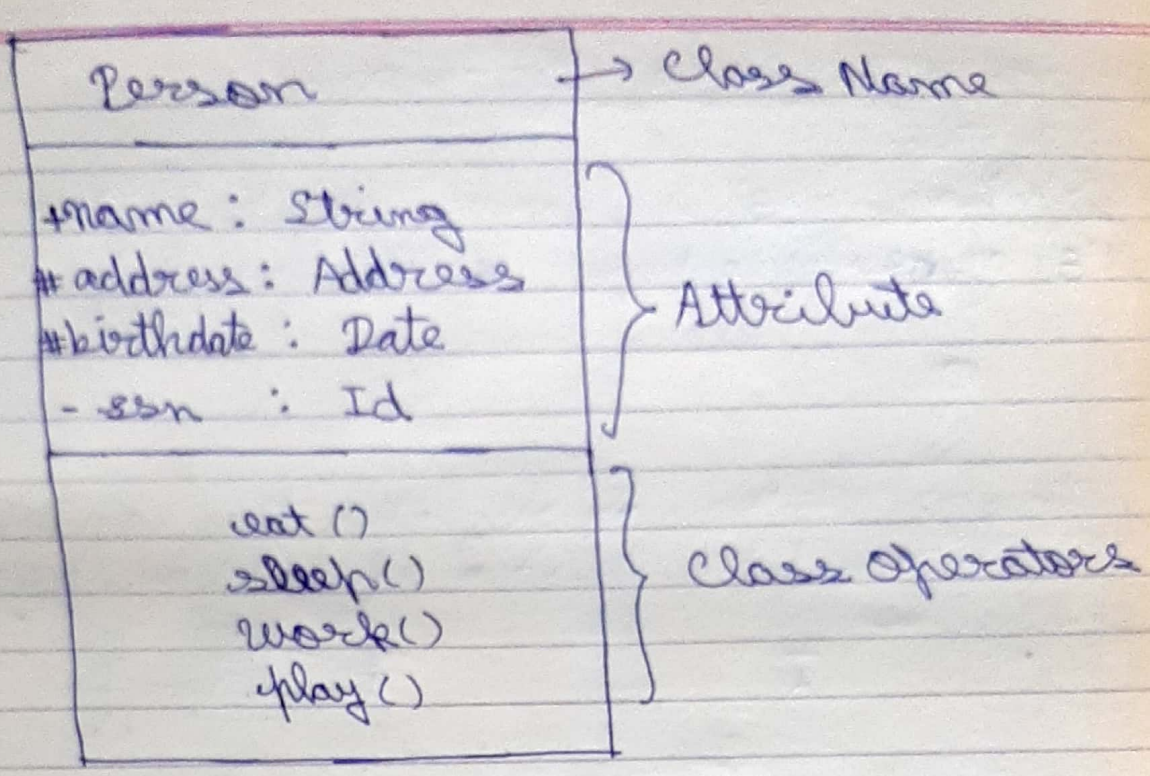
3) Diagrams

1. Object Diagram
2. Class "
3. Sequence "
4. Collaboration "
5. State Chart "
6. Activity "
7. Usecase "
8. Component "
9. Deployment "

Classes in UML

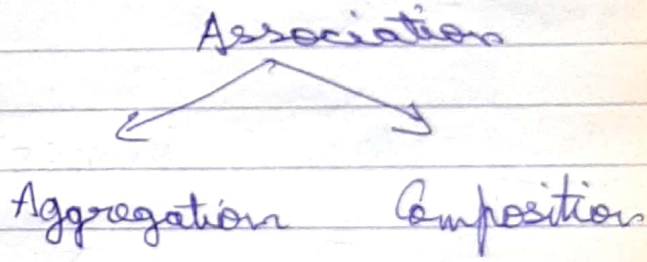


eg:



Class Attributes

+	public
#	protected
-	private
/	derived



Relationships

1) Aggregation → "is part of" ◇—

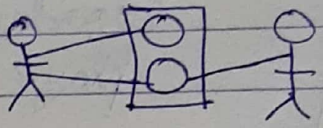
2) Composition → "is entirely made of" ◆—

3) Dependency → "uses temporarily"

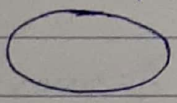
USE Case Diagram

1) System
2) Actors 3) Use Case 4) Connector
5) generalization 6) Relationship

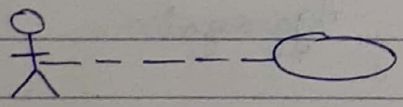
Actors → external object(s) this can be kept outside of the boundary

Symbol ⇒  

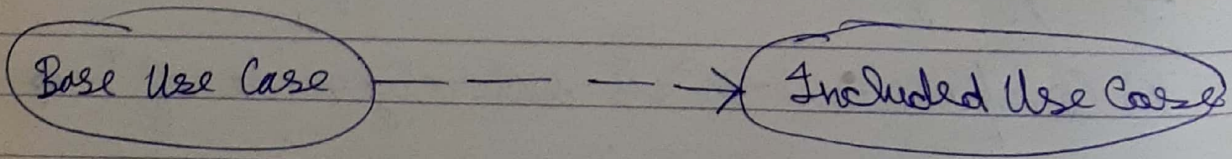
Use Case → Represents some capability

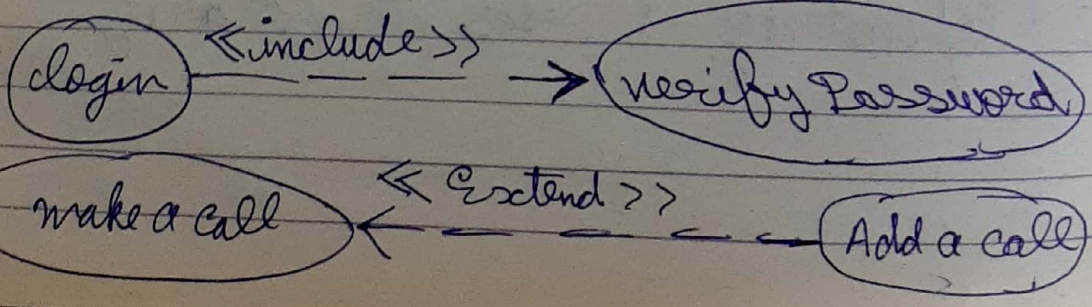
Symbol ⇒ 

Connector → Connectivity b/w users & use case



Primary Actor → Initiates
Secondary Actor → Response to Primary Actor



Eg: 

Star UML

→ Add Diagram → Use case diagram → use case Subject (Shopping app) → Use Case (Order) → Use Case (cancel) → Use Case (Return) → Use Case (login) → Actor (outside) (customer)

↓ make link b/w customer & order, cancel, Return (using Association)

↓

include b/w order & login (include)

&

include " " Return & login

" " cancel & login

↓

Use Case (Status) → Actor (Seller)

Use Case (Report)

↓

link Status & Report with seller

↓

Actor (Bank admin) → Connect with login

Include → When a use case is included means it must be included in the real time situation & without it flow will break.

Extend → when a usecase is extended to a base means it may or may not be included in real time situation & without it flow may be as usual

Activity Diagram

Swimlane Diagram

Scholarship incharge | Teacher | Accounts officer

starting point

Action or Activity state

Summit Student details

fork

Check Attendence

Check fees details

Yes >75% No

Yes paid ? No

Accepted

Rejected

Accepted

Rejected

merge

Join

Both Accepted ?

Yes No

Grant

Don't Grant

⊙ ← End node

Activity Diagram ↓ Behaviour Diagram

To illustrate flow of control

1st find → initial & final State

Join → It will be used to support the concurrent activities converging into one.

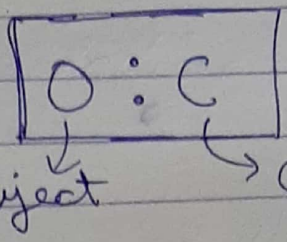
For Join notation, it will have two or more incoming edges and one outgoing edge

Fork → When a main activity or base activity is splitted into 2 or more parallel or simultaneous activities.

Behavioural Diagram

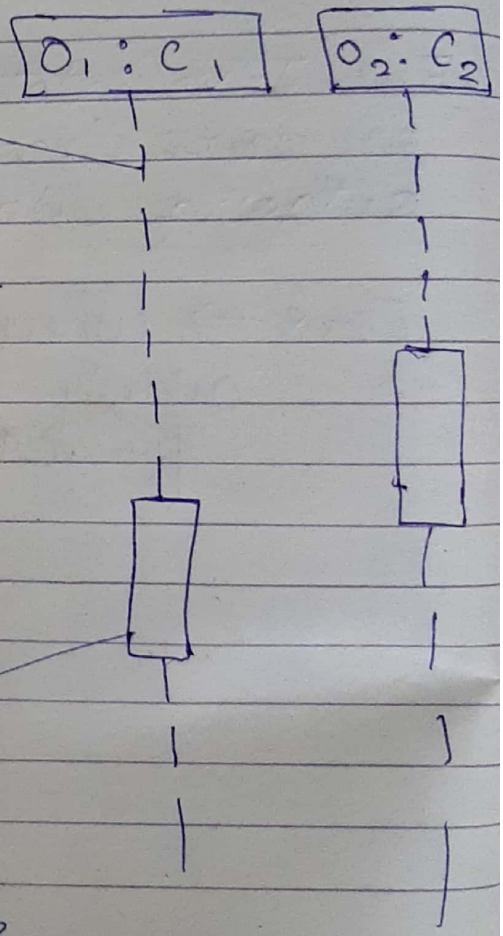
Sequence Diagram

① Object



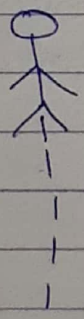
Eg. Shri : student

② Lifeline (dotted line)



no two lifelines can overlap each other

③ Actor

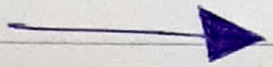


④ Activation bar:

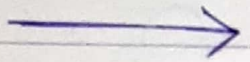
It shows a period/duration for which our object is active, our object is operational, it executes something

5. Messages

① Synchronous : sender waits for reply from receiver, after which he can forward his interaction



② Asynchronous : here he does not wait

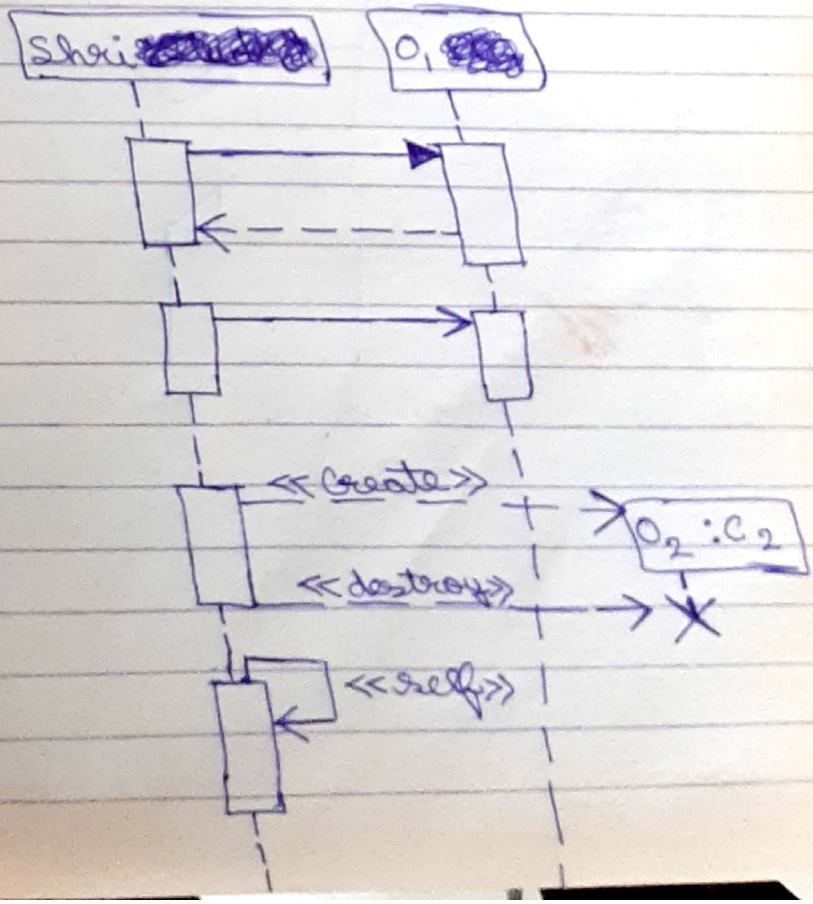


③ Return : Receiver se sender ke taraf jo reply aya

④ Create :

⑤ delete

⑥ Self



Component Diagram

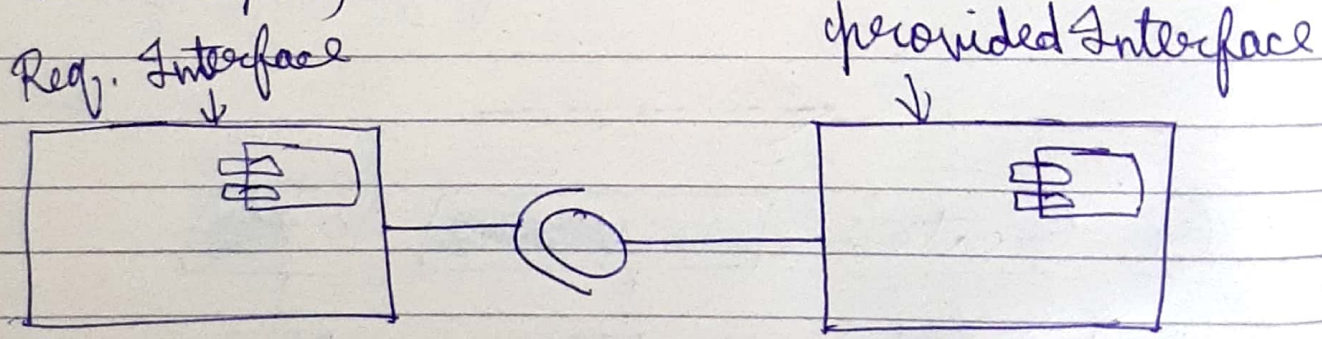
→ To model physical parts of a System.

→ A component is an executable piece of a system whose implementation details are hidden.

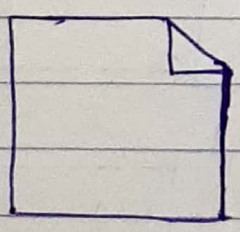
→ An Interface (small circle on a stick) it describes a group of operations used (required) and created (provided) by components

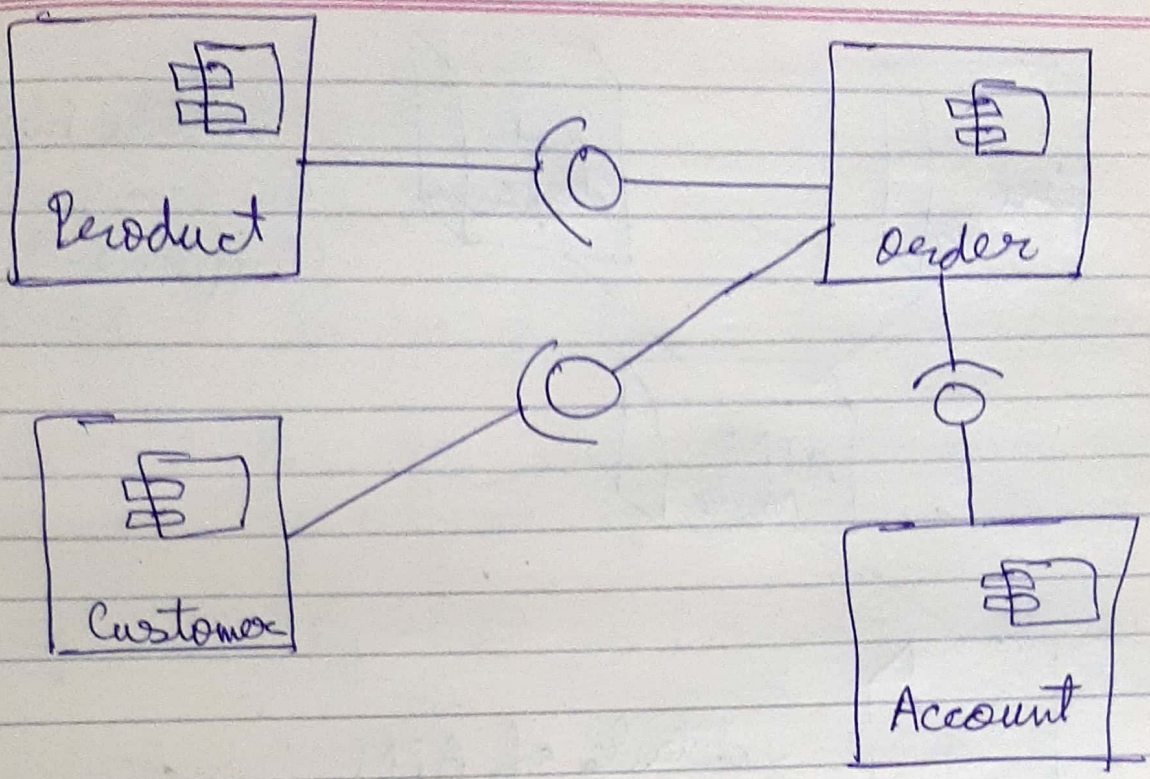
provided Interface →

Required Interface → (req. user input)



Artifact →



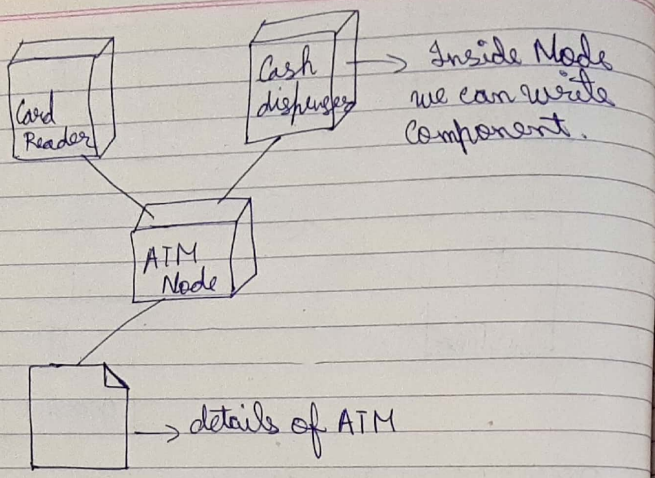


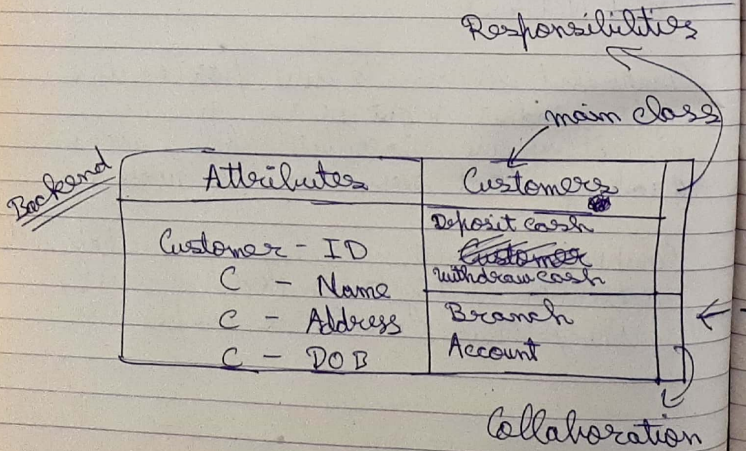
⇒ Diff. b/w Component & Node :-

Node Represents the physical part of the System, for Instance server, printer and Network.

Component represents any part of the system and it might be the physical aspects like file, Executable file, document, package that reside on the node.

Components are things that participate in the execution of a system. Nodes are the things that execute the components.





CRC Card

Class | Responsibilities | Collaboration

Class: A class describes the behaviour of a set of objects of the same kind.

Responsibility → It is basically, these are the knowledge that a class maintains & the services that it provides.

Collaborations → A collaborator is a class whose services are required to fullfill the responsibilities.