UNIT - V

Object Oriented Analysis (OOA)

→ What is object oriented Analysis ?

Object oriented Analysis (OOA) is a process of discovery where development team understands and models the requirement of the system. In OOA requirements are organised as objects. It integrates all parts, process and data. But in others or traditional structural analysis both process and data are considered independently. They use flow chart / structure charts for process and ER diagrams for data.

But in OOA some advance models are used. The common models are : Use cases, object models. Use cases describe pictures or overview for standard domain functions that the system must achieved. Object models describe the names, class relations, operations, and properties of the main objects. User interface prototypes can also be created for better understanding.

Object oriented analysis (OOA) begins by looking at the problem domain (the area of expertise or application that needs to analyze in order to solve a problem).

procedures as in structured analysis.

It aims is to produce a conceptual model of the information that exists in the area being analyzed. For the analysis there are a variety of sources. It can be formal document, a written requirement statement, interviews with stakeholders / other interested parties, other methods etc. The final result of object oriented analysis will appear in the form of a conceptual model that describes what the system is functionality required to do.

→ Functionalities of OOA

The core activities in OOA are given below

- Find the objects
- Organise the objects by creating object model diagram.
- Explain how the objects communicate with each others
- Set the characteristics or behaviour of the objects.
- Set the internal of objects.

→ Advantages of OOA.

The OOA provides better performance. Some common advantages of OOA are given below.

- It focuses on data rather than the procedures as in structured analysis.
- The objectives of encapsulation and data hiding help the developer to develop the systems that cannot be tampered by other parts of the system.
- It allows effective software complexity management by the virtue of modularity.
- It can be upgraded from small to large system easily.

→ Structured Analysis Vs Object Oriented Analysis

Some differences between Structured Analysis vs object oriented Analysis are given below:

- Structured analysis treats process and data as seperate components. Where, OO analysis combines process and data into single component which called object.
- Structured Analysis does not support reusability of code. So, the development time and cost is naturally high. But

Oo analysis support code re-usability which reduce the development time and cost.

~~The Object Oriented~~

Data Modelling :

→ What is data modeling ?

Data modeling is the process of creating a data model for the data to be stored in Database. This data model is conceptual representation of

- Data objects
- The associations between different data objects.
- The rules.

Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data models ensure consistency in naming conventions, default values, semantics, security while ensuring quality of the data.

Data model emphasizes on what data is needed and how it should be organized instead of what operations need to be performed on the data. Data model is like

architect's building plan which helps to build a conceptual model and set the relationship between data items.

The two types of Data Models techniques are:

1. Entity Relationship (E-R) model
2. UML (Unified Modeling language)

→ Why use Data Models?

The primary goal of data model are:

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to ~~certain~~ creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model

is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintanance cheaper and faster.

→ Types of Data Models:

1. Conceptual: This data model defines what the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organise, scope and define business concepts and rules.

2. Logical: Defines How the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analyst. The purpose is to developed technical map of rules and data structures.

3. Physical: This data model describes how the system will be implemented by using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

→ Advantages of Data Model

- The main goal of designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship b/w tables, primary and foreign keys, and stored procedures.
- Data Model helps business to communicate within and across organisations.
- Data Model helps to documents
- Help to recognize correct sources of data to populate the model.

→ Disadvantages of Data Model:

- To developer data model are should know physical data stored characteristics.
- This is a navigational system produces complex application development, management Thus, it requires a knowledge of the ~~biological~~ biographical truth.
- Even smaller change made in structure require modification in the entire

application.
- There is no set data ~~implementatio~~ manipulation language in DBMS.

→ Conclusion

- Data modeling is the process of developing data model for the data to be stored in Database.
- Data models ensure consistency in naming conventions, default values, semantics, securit while ensuring quality of the data.
- Data model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- There are three types of conceptual, logical and physical.
- The main aim of conceptual model is to establish the entities, their attributes, and their relationships.
- Logical data model defines the structure of the data elements and set the relationships b/w them.
- A physical data model describes the database specific implementation of the data model.
- The main goal of a designing data model is to make certain that data objects

offered by the functional team are represented accurately.

- The biggest drawback is that even smaller change made in structure require modification in the entire application.

Object Oriented Analysis and Design (OOAD)

Traditional System Analysis and Design ?

The systems Development life cycle (SDLC) or the structured system analysis & Design (SSAD) is a frame work of activities and task that need to be accompalised to develop an information system. This approach is also known as top-down design, madular programming, and stepwise refinement. In this approach, every problem is divided into smaller sub-problems. The solution of overall sub-problem are then combined to solve the overall problem. The process of implementing a structured design is called structured programming.

→ Object Oriented Analysis & Design

Object oriented analysis & design (OOAD) is a technological approach to analyze, design a software system or business by using object Oriented (OO) concept. Object

(10)

Oriented Analysis (OOA) is the investigation of objects. Object oriented Design (OOD) is the relationships of identified objects.

The most important purpose of OO analysis is to identify the objects of a system that have to be implemented. This analysis can also perform for an existing system. An efficient analysis is the only possible when we think in a way where objects can be identified. After identifying the objects the relationships b/w them are identified and finally the design is produced.

→ Purpose of OO analysis and Design:

We can summarize the purpose of OO analysis and design in the following way

- Identifying the objects of a system.
- Identify their relationships
- Generate a design which can be converted into applications using OO languages.

In software development life cycle we can apply and implement OO concepts by following three steps. The steps can be labeled as :-

OO Andysis → OO Design → OO implementation

The first phase is oo analysis. In oo analysis the most important purpose to identify objects and describes them in a proper way. If the objects are identified efficiently then the next task of design will be easy. The objects should be identified with their responsibilities. The responsibilities are the functions or jobs performed by the object. We know every object have to perform same responsibilities. When these responsibilities are collaberates accuratedy the purpose of the system is fulfilled.

The second phase is oo design. This phase highlights on the requirements and their fulfillment or completion. In this phase all the objects are collaberated according to their projected relationship. After the completion of relationships the design is completed.

The third phase is oo implementation. In this phase the above design is implemented using oo languages like C#, Java, C++ etc.

(2) → Advantages of Object-Oriented Analysis and Design:

Benefits of object oriented Analysis and Design are given below:

- It is easy to understand
- It is easy to maintain. Due to its maintainability OOAD is becaming more popular day by day.
- It provides re-usability.
- It reduce the development time & cost.
- It improves the quality of the system due to program reuse.

→ Disadvantages of OO Analysis & Design:

Drawbacks of object oriented Analysis and Design are given below:

- In OOAD, all time it is not easy to determine all the necessary clauses and objects required for a system.
- Most of our project development teams are familiar with traditional analysis & design. The OOAD offers a new kind of project management. That's why it may be difficult to complete a solution within estimated time and budget.
- Without an explicit reuse procedure this methadalogy do not lead to succesful

reuse on a large scale.

→ Differences b/w structured and object Oriented Analysis and Design:

A comparision between traditional structured analysis & design and object oriented analysis and design is given below:

Object oriented:

1. It is data oreinted.
2. Its main focus is data.
3. There is no seperation of the systems data and procedures. Data and processes are encapsulated into objects.
4. Its risk is low.
5. It breaks down the system date through the use of use-cases.

Structured

1. It is process oriented.
2. Its main focus is process.
3. There is a seperation of the systems of data and procedures.
4. Its risk is high.
5. It breaks down the system data through the use of Data Flow Diagram (DFD).

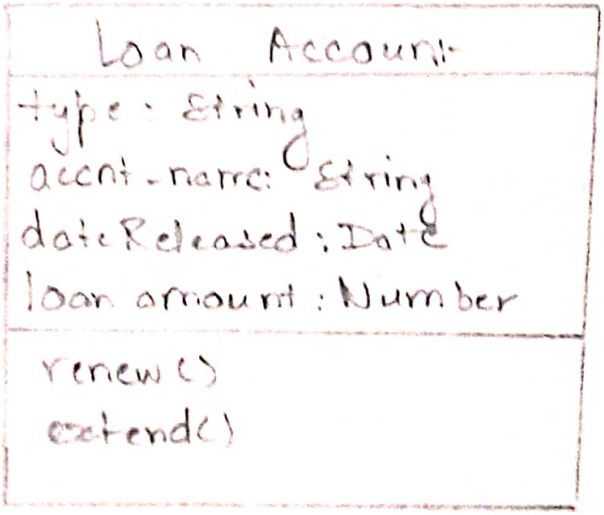
14

Class Diagrams

→ What are class Diagrams?

Class diagrams are main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

The following figure is an example of a simple class.



In the example, a class called 'loan Account' is depicted. Classes in class diagrams are represented by boxes that are partitioned into three.

1. The top partition contains the name of the class.
2. The middle part contains the class's attributes
3. The bottom partition shows the possible

operations that are associated with the class.

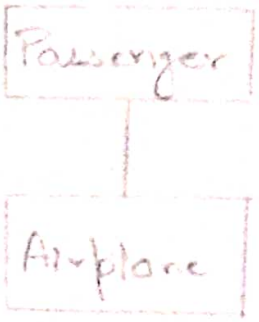
The example shows how a class can encapsulate all the relevant data of a particular object in a very systematic and clear way. A class diagram is a collection of class similar to the one above.

→ Relationships in Class Diagram:

Classes are interrelated to each other in specific ways. In particular, relationships in class diagrams include different types of logical connections. The following are such types of logical connections that are possible in UML

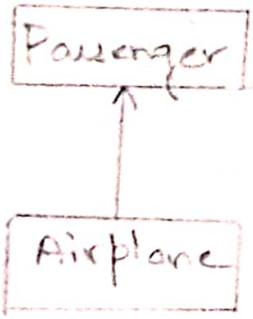
- Association
- Directed Association
- Reflexive Association
- Multiplicity
- Aggregation
- Composition
- Inheritance / Generalization
- Realization

(16) Association:



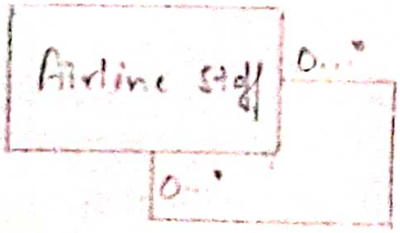
Association is a broad term that encompasses just about any logical connection or relationship t/a clauses. For example, passenger and airline may be linked as above:

Direct Association:



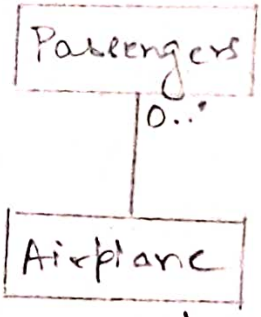
It refers to a directional relationship represented by a line with an arrowhead. The arrowhead depicts container-contained directional flow.

Reflexive Association:



This occurs when a class may have multiple functions or responsibilities. For example, a staff member working in an airline may be a pilot, aviation engineer, a ticket dispatcher, a guard, or a maintenance crew. If the maintenance crew member is managed by the aviation engineer there could be a managed by relationship in two instances of the same class.

Multiplicity:

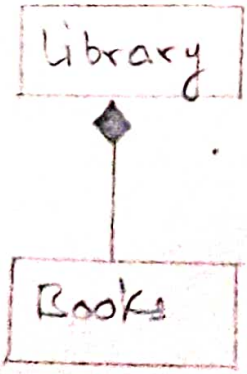


It is the active logical association when the cardinality of a class in relation to another is being depicted. For example, one fleet may include multiple airplanes, while one commercial airplane may contain zero to many passengers. The notation 0..* in the diagram means 'zero to many'.

(18) Aggregation

It refers to the formation of a particular class as a result of one class being aggregated or built as a collection. For example, the class 'library' is made up of one or more books, among other materials. In aggregation, the contained classes are not strongly dependent on the lifecycle of the container. In the same example, books will remain so even when the library is dissolved. To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.

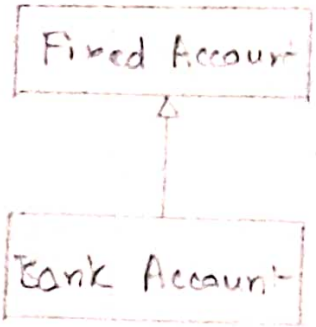
Composition



It is very similar to the aggregation relationship with the only difference being its key purpose of emphasizing the dependence of the

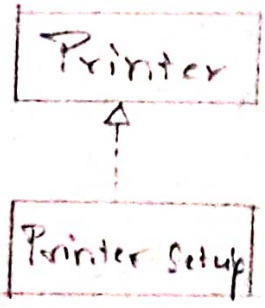
contained class to the life cycle of the container class. That is, the contained class will be obliterated when the container class is destroyed. To shaw a compasition use a directional line connecting two classes, with a filled diarrond shape adjacent to the container class and the directional arraw to the carntained class.

Inheritance / Generalization



It refers to a type of relationship wherein one associated class is a child of anather by virtue of assuming the same functionalities of the parent class. In other words, the child class is a specific type of the parent class. To shaw inheritance, a solid line from the child class to parent class is draw using unfilled arrawhead.

Realization



(20) It denotes the implementation of the functionality defined in one class by another another class. To show relationship, a broken line with an unfilled solid arrowhead is drawn from the class that defines functionality of the class that implements the function.

Unified Modelling Language (UML)

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering.

UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behaviour and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design and analysis. The object management Group (OMG) adopted Unified Modeling Language as a standard in 1997. Its been managed by OMG ever since. International organisation for Standardisation published UML as approved in 2005.

Do we really need UML?

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So UML becomes essential to communicate with non-programmers essential requirements, functionalities, and processes of the system.
- A lot of time is saved down the line when teams are able to visualize processes, User interactions and static structure of the system.

...n to UML (Unified Modeling Language):

The UML is a standard language
...g software. blue prints. The UML
...nguage for.

...ing
...ing
...cting
...cting

...ding blocks →

...L building ~~do~~ blocks. The building
UML can be defined as.

...nship
...

...ings are the most important building
of UML.

...can be:

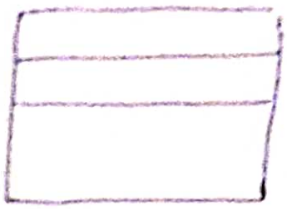
...ctural
...vioral
...bing
...ational

Structural things :→

Class :-

It is set of objects, with similar structure, behaviour.

It is represented by a rectangle.



Interface :-

An interface is a specified for the externally-visible operations of a class, component, or other classifier without specification of internal structure. It is represented by a circle.

Relations :→

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

Final state :-

A final state represents the last or "final" state of the enclosing composite state.

Symbol →

Junction Point :-

Junction point chains together transition into a single run-to-completion path. Each complete path involving a junction is logically independent and only one such path fires at one time.

Symbol →

Transition :→ A transition is a directed relationship b/w a source state vertex and a target state vertex. It may be a of a compound transition.

symbol :-

Activity diagram :-

It represents the different activities in the system

Action state:

An action state represents the execution of an atomic action, typically the invocation of an operation

An action state is a simple state with an entry action whose only exit transition is triggered by the implicit event of completing of the entry action.

Symbol

Sub-activity state:

A sub-activity state represents the execution of a non-atomic sequence of steps that has some duration; that is, internally it consists of a set of actions and possibly waiting for events

Association :-

An association is a structural relationship that specifies the relation between two objects when they are at the same level

It can be represented as follows:

Direct Association :-

Links a semantic association between two classes in the UML diagram

Aggregation :-

- Links a semantic association b/w two classes in the UML diagram
- Aggregation is used in class diagram

Composition :-

- Links a semantic association b/w two classes in UML diagram.
- Composition is used in class diagram

Generalisation :-

Generalisation is a specification relationship in which objects of the specified elements are substitutable for object of the generalization element

Dependency :-

A dependency is a semantic relationship in which if there is any change occured in one object that may effect other object

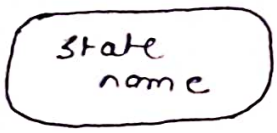
Realization :-

Realization is a specified tool that can be represented by providing a relationship with classifier.

State Chart Diagrams :-

State :- A state is a condition during the life of an object or an interaction during which it satisfies some condition, performs some action, or waits for some event. It is represented by a rounded rectangle.

Symbol :-

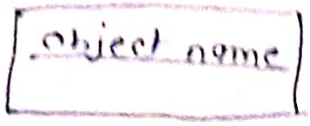


Sub machine state :- A submachine state is a syntactical convenience that facilitate reuse and modularity.

initial state :-

It represent the starting point in a region of a state machine. It has a single outgoing transition to the default state of enclosing region, and has no incoming transitions.

object → It is an instance of a class

symbol :- 

Stimulus :-

A stimulus is a Communication between two instances that Conveys information with the expectation that action will ensue.

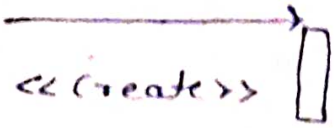
symbol 

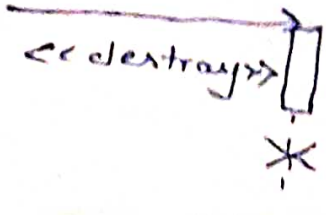
It can be annotated by a name. It has a property as action kind.

call: 

send : 

Return : 

create : 

Destroy : 

Interaction Diagrams:

An interaction diagram shares the same common properties as all other diagrams. It differs in its contents.

- object
- Links
- messages

It includes two diagram
- sequence and collaboration

Sequence Diagrams:-

sequence diagrams have two features that distinguish them from collaboration diagrams.

① object's life time
② The focus of control.

Collaboration Diagrams:-

Collaboration diagrams have two features that distinguish them from sequence diagrams.

① path
② The sequence number.

Class Diagrams :-

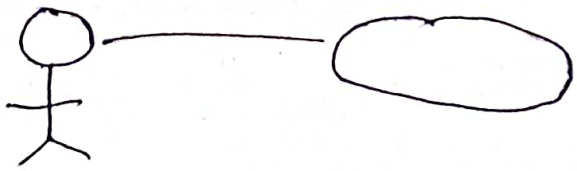
A class diagram is that which represents a set of classes, interfaces and collaboration and their relationships, graphically a class diagram is a collection of vertices and arcs.

It consist of three Compartments.

Uses : A class diagram is used to model the static design view of a system.

Use Case diagrams :-

A use case diagram shares the common properties as all diagram. It distinguishes in the contents of use cases, actors, dependency and generalization relationships.



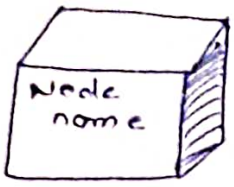
Actor

Uses :- A usecase diagram is used to model the static design view of a system.

Deployment Diagram :-

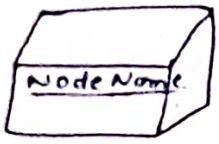
Node :-

A node is a run-time physical object that represents a computational resource generally having at least a memory

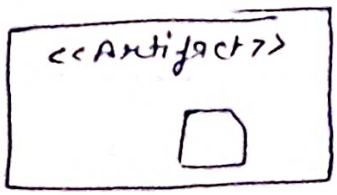


Node instance :-

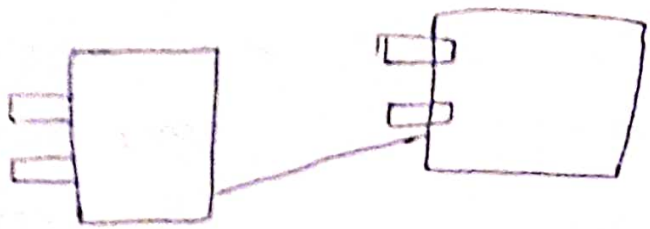
A node instance is an instance of node



Artifact :- An artifact represents a physical piece of information that is used or produced by a software development process.



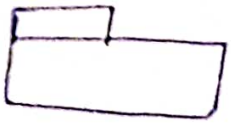
Component Diagrams :-



Package :-

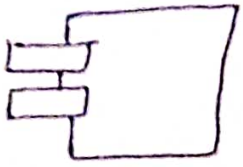
A package is a grouping of model elements package themselves may be nested within other package.

All kinds of UML model elements can be organised into packages

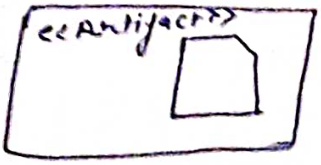


Component -

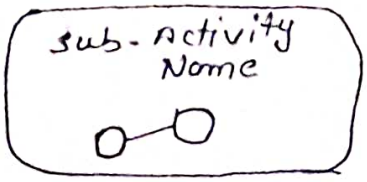
A Component represents a replaceable part of a system that encapsulates implementation and exposes a set of interfaces.



Artifact :- An Artifact represents a physical piece of information that is used and produced by a software development process



Symbol :-



Architecture of UML :-

UML plays an important role in defining different perspectives of a system. These perspectives are :

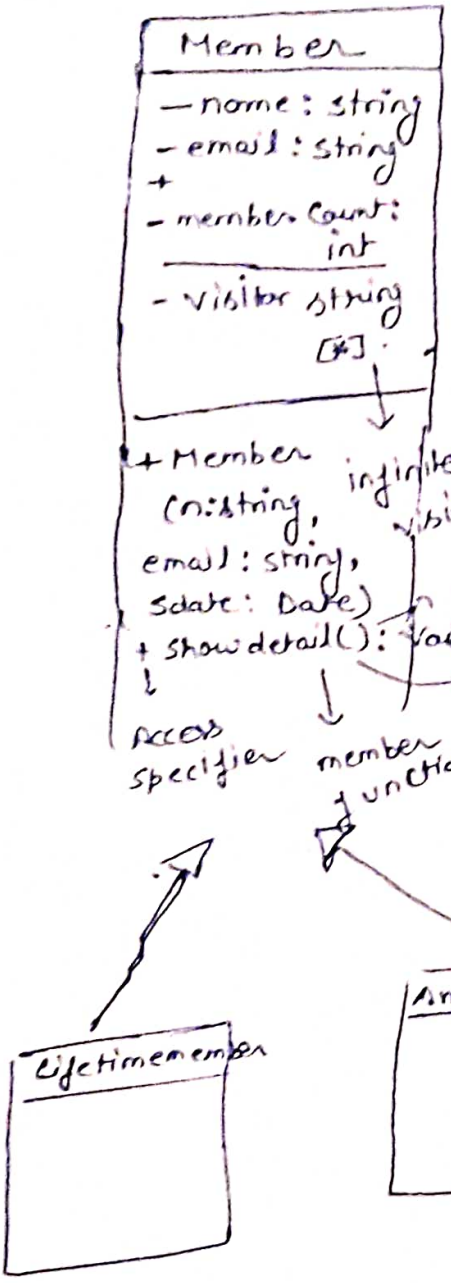
1. Design
2. Implementation
3. process
4. Deployment

Design of a system Consists of classes, interfaces and collaboration.

Process defines the flow of the system. So the same elements as used in design are also used to support this perspective.

Deployment represents the physical nodes of the system that forms the hardware

class Diagram :-



' data types :-

int, string, char, float

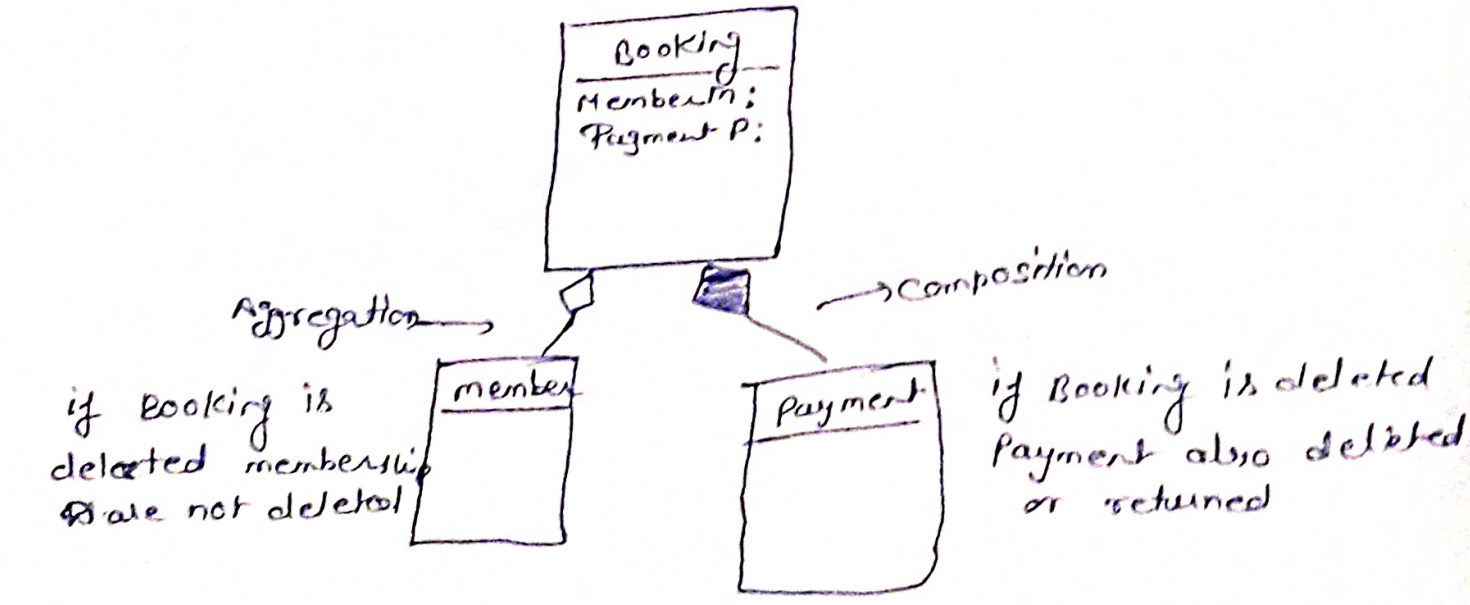
Access specifier :-

+ public
- private
#+ protected → class and child class
~ package → within the class or within the class of the package

Modifiers / Constraints :-

① static → by underline
② Read only → can not be change
③ unique

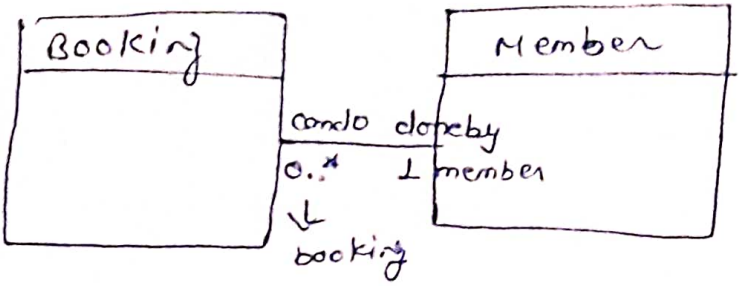
these are unique



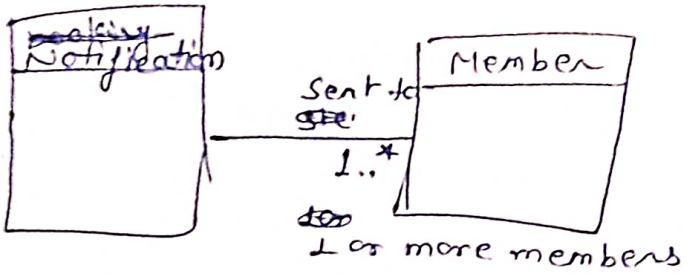
if Booking is deleted membership are not deleted

if Booking is deleted Payment also deleted or returned

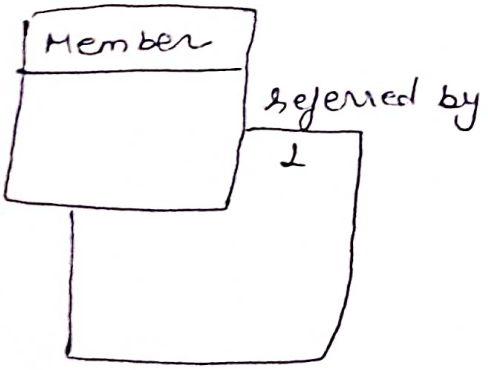
Bidirectional association



Unidirectional association:-



Reflexive Association

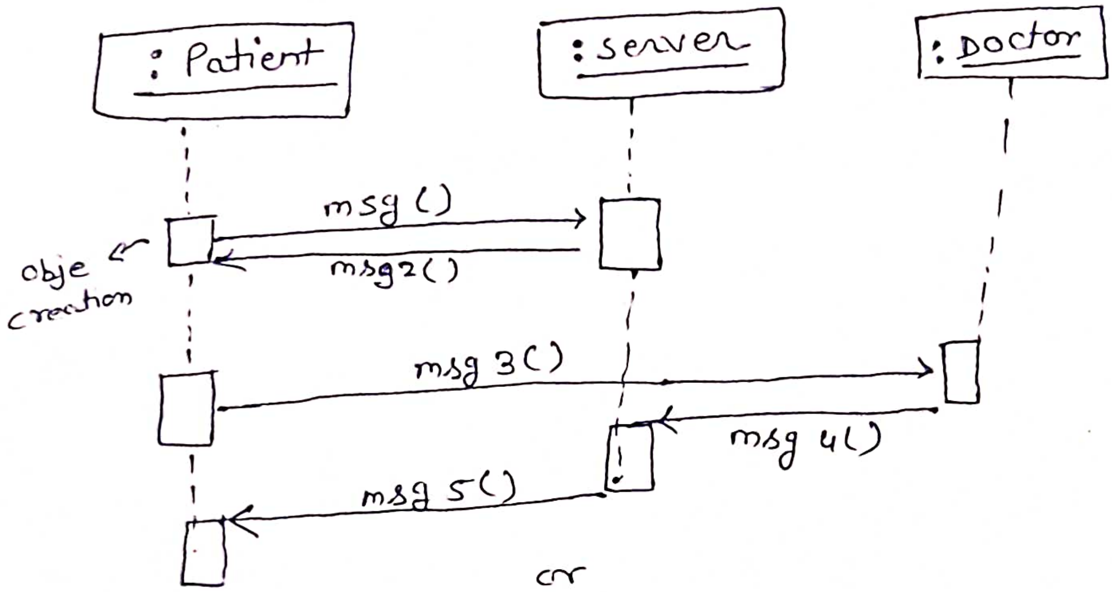


sequence diagram

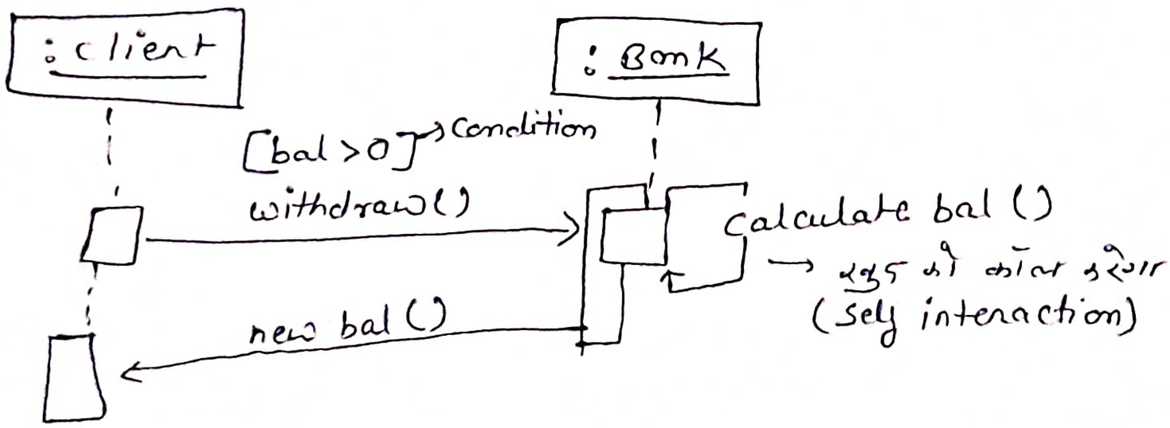
How object can communication with each other with message

① life line

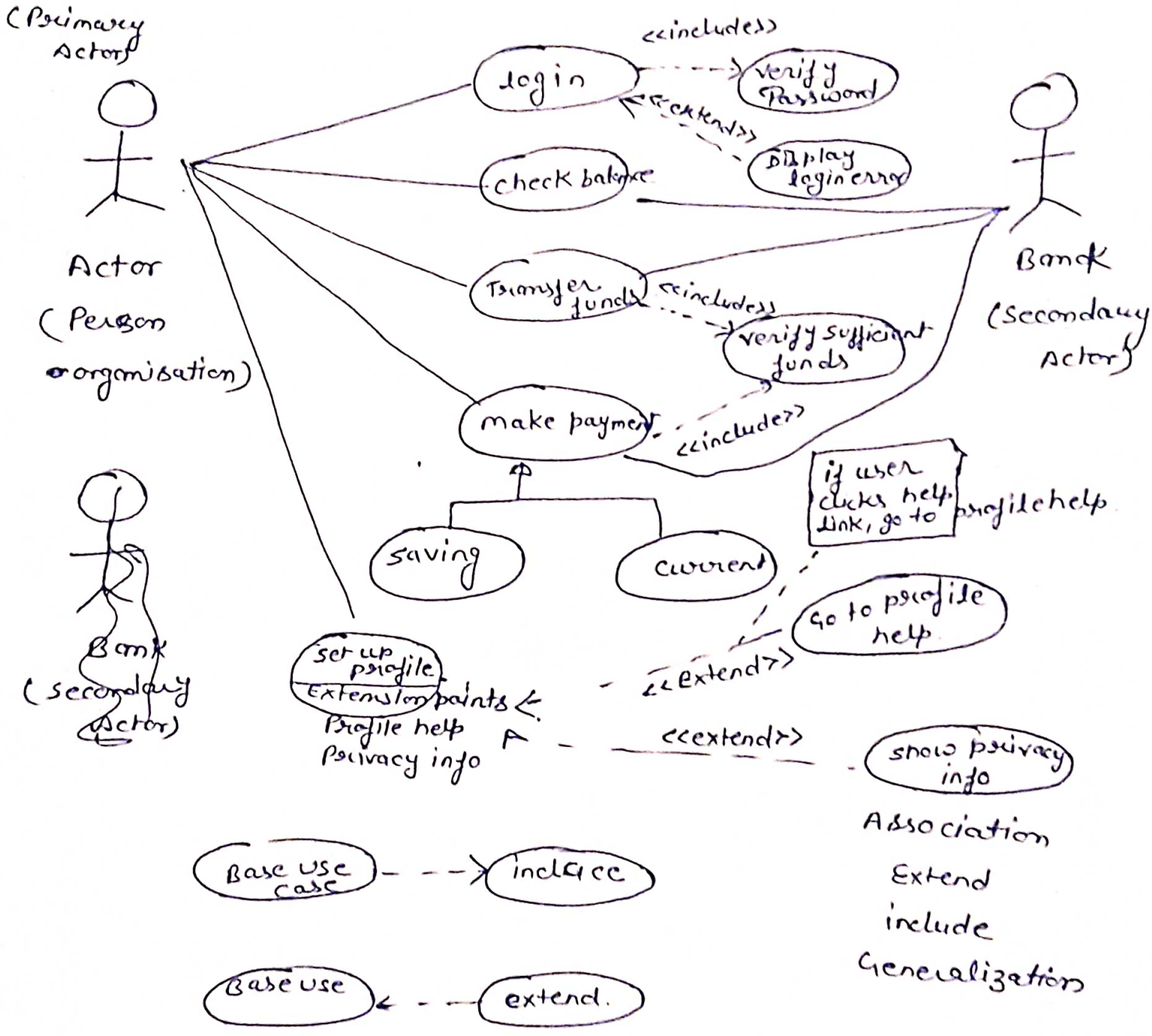
| object1: class | | obj2 : class



or



Banking App


(Primary Actor)
Actor
(Person or organisation)
login
<<includes>>
verify Password
<<extend>>
Display login error
check balance
Transfer funds
<<includes>>
verify sufficient funds
make payment
<<include>>
Bank
(secondary Actor)
saving
current
if user clicks help link, go to profilehelp
go to profile help
<<extend>>
Bank
(secondary Actor)
set up profile
Extension points
Profile help
Privacy info
<<extend>>
show privacy info
Association
Extend
include
Generalization
Base use case
include
Base use
extend.

use case diagram.

 - Actor - role play (system) or gr. of people

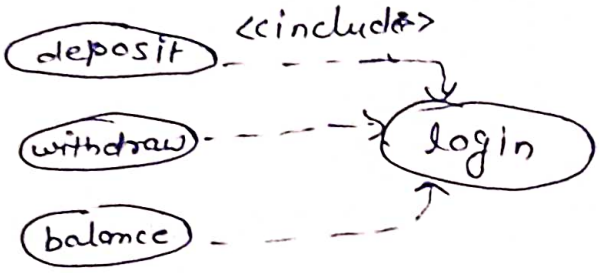
 - use case → capability

 - connector → interaction

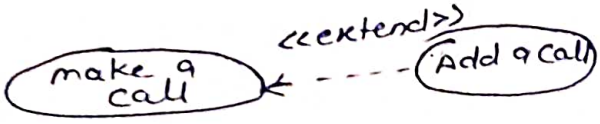
 - generalization - for dividation

 - stereotype - (relationship)

① <<include>> - Implicit fun. (use all time)



② <<extend>>



(not necessary all time)

Shoping App

① 

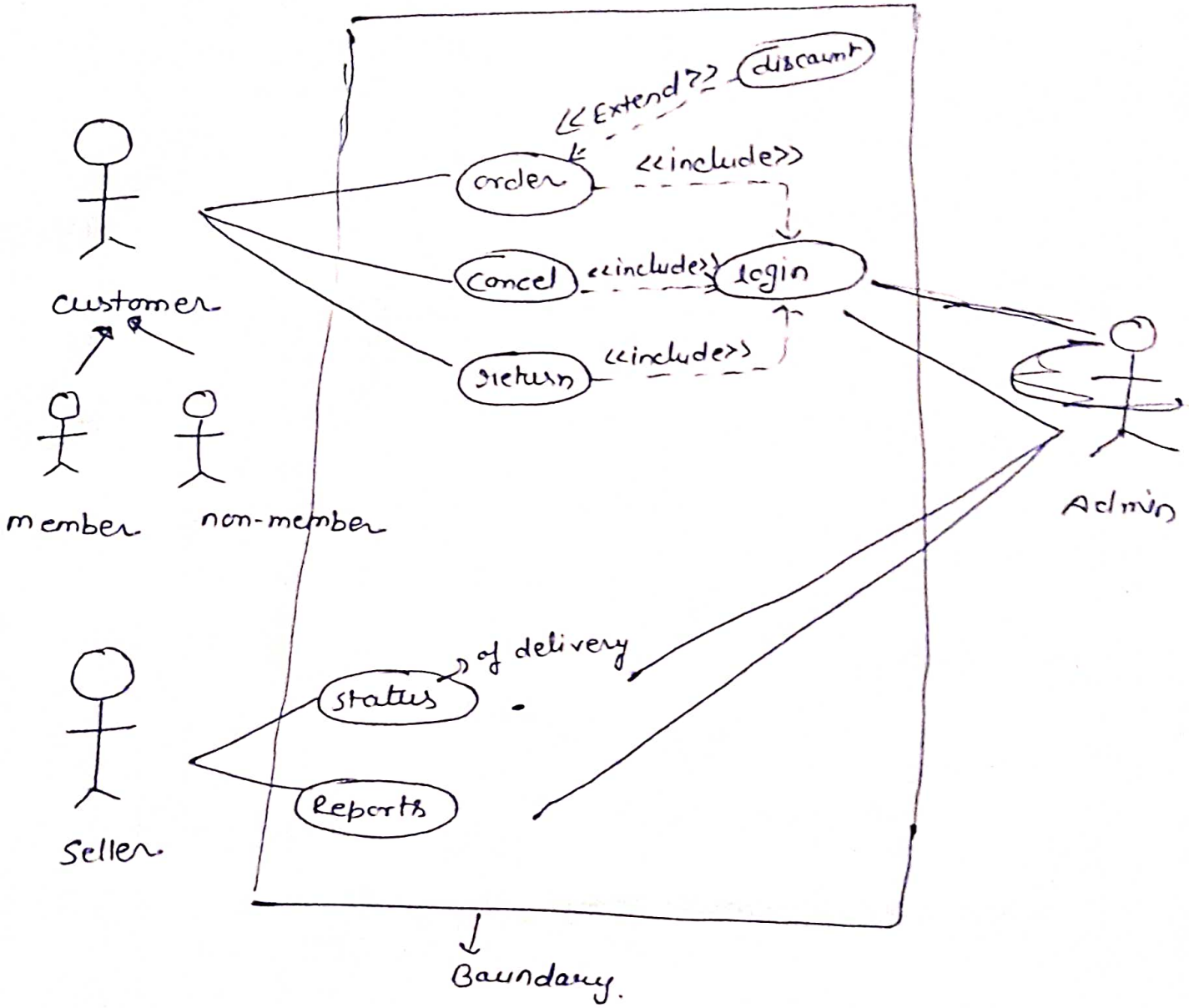
② <<include>>

③ <<extend>> ~~exclude~~

④ use-case

⑤ should be 2 actors.

⑥ system boundary



Use cases are in Boundary.

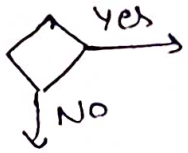
<< extend>> is optional

Activity Diagram

Activity diagram describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity.

start

Activity

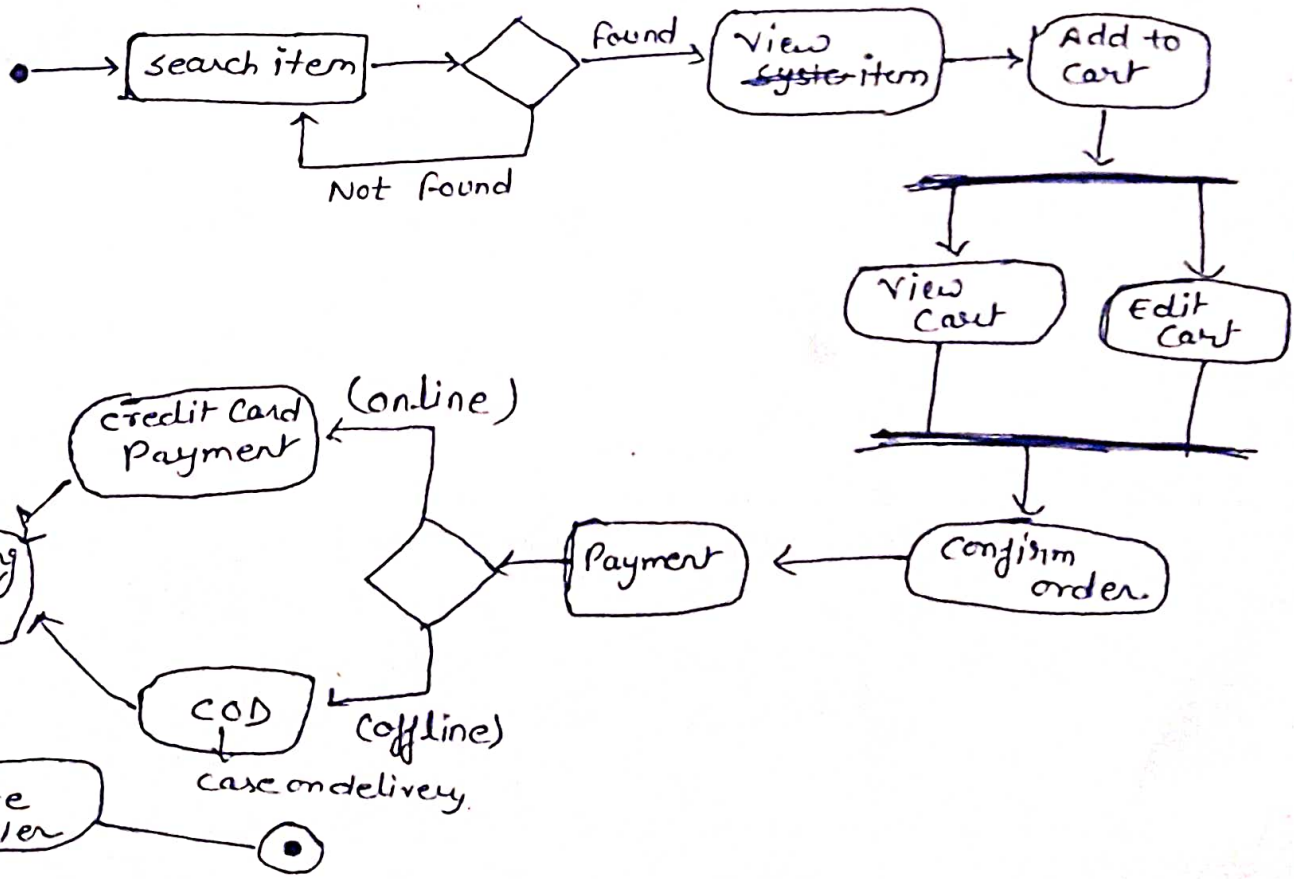


Condition

Parallel activity

end

Online shoping system



Library management