1 Definition of Object Oriented Paradigm:-

"Object oriented or Object Orientation is a s/w engg. concept in which concepts are represented as "objects" (a variable, a data structure or a function)". It can refers to:

Ref:- R4

1) Object-Oriented analysis & design
2) Object-Oriented design
3) Object-Oriented modeling
4) Object-oriented Programming
5) Object-Oriented Software Engineering
6) Object-oriented User Interface.

Ref:- R4

2 Object Oriented Concepts:-

Object oriented concepts include various features of OO approach, which inclu

Name of Lecturer: Abhishek Jain

1) Class

2) objects

3) Abstraction

4) Encapsulation

5) Inheritance

6) Polymorphism

1) Class :- A class is a collection of similar type of objects. once a class is defined, any no. of objects can be created which belong to that class. A class is a blueprint, a Prototype, that defines the variables & the methods common to all objects of a certain kind.

2) Objects :- An object is an instance of a class. Objects may be a variable, a data structure or a function.

3) Abstraction :- Data abstraction refers to, providing only essential information to the outside world & hiding their background details i.e.

Name of Lecturer : .....

to represent the needed information in Program without ... Presenting the details.

4) Encapsulation :- Storing data & functions in a single unit (class) is encapsulation. Data cannot be accessible to the outside world & only those functions which are stored in the class can access it.

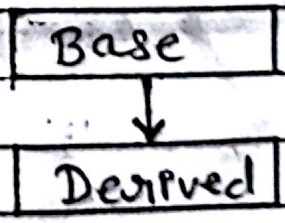
5) Inheritance :- It is the Process by which one object acquires the properties of another ... object.

OR

It is the Process by which object of one class can acquires the properties of object of another class.

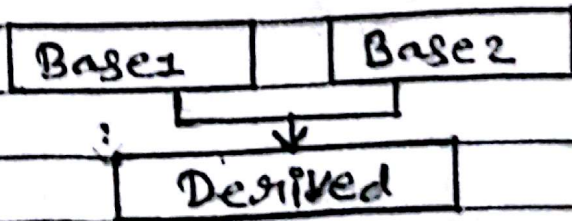
Types of Inheritances :

→ Single Inheritances :- In this one derived class inherits from only one base class.



→ Multiple Inheritance :-

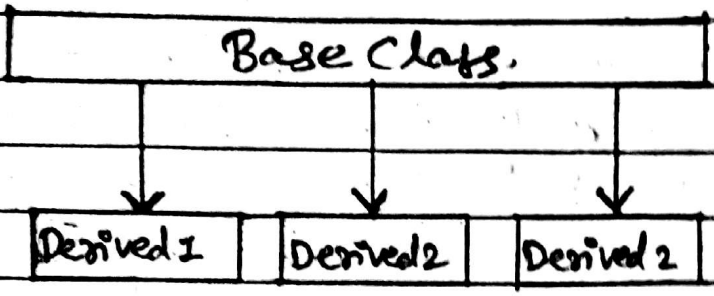
In this a single derived class is inherit from two or more than two base class



Name of Lecturer : Abhishek Jain

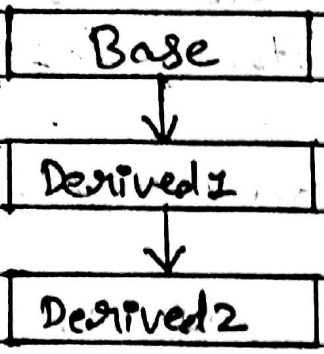
→ **Hierarchical Inheritance :-**

In this a multiple derived classes inherits from a single base class.

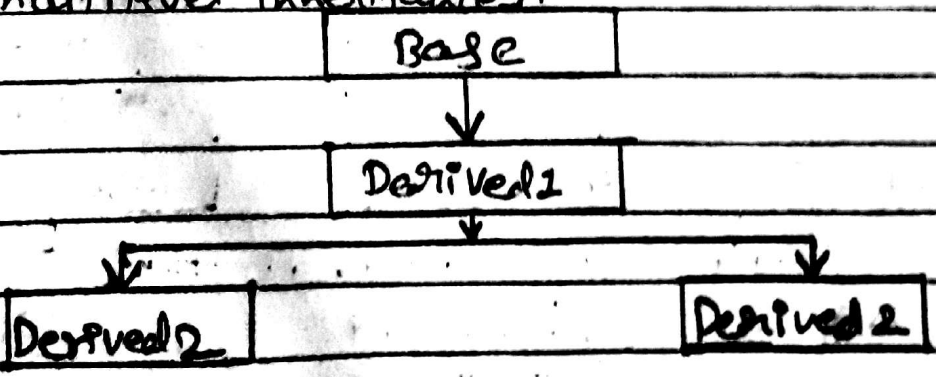


→ **Multilevel Inheritence :-**

When a derived class is created from another derived class.



→ **Hybrid Inheritance :-** It is a combination of single, hierarchical & multilevel inheritance.



6) **Polymorphism:-** Polymorphism means the ability to take more than one form. An operation may behave differently in different instances. The behaviour depends on the data types used in the operation.

Ref. :- R5, R6, R7

3] Object Oriented Analysis :- (OOA)

Definition of Analysis :-

"Analysis emphasizes an investigation of the Problem & requirements, rather than a solution".

Definition of OOA :-

"In OOA, there is an emphases on finding & describing the objects in the Problem domain

Conventional Vs OOA :-

→ Structured Analysis treats Processes & data as separate components. Whereas OOA combines data & Process that acts on the data into objects.

→ In Structured analysis, Analysis is done on

the basis of DFD's, Decision Table/TREE & ER (Entity Relationship) Analysis.

→ OOA is done on the basis of USE CASE MODEL, & Object Model i.e. by UML Modeling

→ Object Model
  → Find classes & class relations.
  → Object relations:
    1. Sequence Diagram
    2. Collaboration Diagram
    3. State Machine Diagram.

Different OOA Models :-

1) Booch Method
2) Rumbaugh Method
3) Jacobson Method
4) Coad & Yourdon Method
5) Wirfs-Brock Method.

1) BOOCH Method :- Strongly emphasizes the Iterative Process & the creativity of the developer as essential components in OO design.

Name of Lecturer : .....

→ The method is more a set of heuristics (a set of rules intended to increase the probability of solving some problem) & no strict baselines or order of work exists.

2) RUMBAUGH Method :-

Rumbaugh & his team developed 3 Object modeling techniques (OMT) for analysis, System design, object level design & implementation.

The three models are;

→ Object Model :- Representation of objects, classes etc.
→ Dynamic Model :- Objects & System behavior.
→ Functional Model :- Information flow.

3) JACOBSON Method :-

→ It is also called as OOSE (object oriented s/w Engg.).

→ Traditional Methods treat functions & data as separate. Such approach often leads to Problem during Maintenance.

→ OOSE do not separate functions & data, but views them as an integrated Whole.

Name of Lecturer : Abhishek Jain.

4) COAD AND YOURDON Method:-

→ The idea in this method is to extend the model with respect to processes, human interfaces & DBMS issues.

→ The method consists of following 5 steps:

→ Finding Class & object.

→ Identifying structures.

→ Identifying subjects.

→ Defining attributes.

→ Defining services.

5) WIRFS-BROCK Method:-

→ Wirfs-brock, do not make a clear distinction between analysis & design tasks.

→ Rather a continuous process that begins with the assessment of a customer specification & ends with design is proposed.

Ref:- R8

Name of Lecturer: .....

4] Domain Analysis :-

The objective of domain analysis is to define a set of classes (objects) that are encountered throughout an application domain. These can then reused in many applications.

Domain Analysis Process :-

S/w domain analysis is the identification, analysis & specification of common req's. from a specific application domain, typically for reuse on multiple projects within that application domain.

Ref. :- R8

5] Object Oriented Analysis Process (Modeling) :-

OO Analysis Process is used together basic customer requirements & then define an analysis model for an object-oriented system.

OO Analysis Process are;

1) Use-Cases :- Use-cases model the system from the end-user's point of view.

Name of Lecturer : Abhishek Jain

→ To define the functional & operational requirements of the system.

→ To Provide a clear & unambiguous description of how the end-user & the system interact with one another.

2) Class-Responsibility-Collaborator Modeling

→ Once basic usage scenarios have been developed for the system, it is time to identify candidate classes & indicate their responsibilities & collaborations called class-responsibility-collaborator (CRC) modeling.

→ It Provides a simple means for identifying & organizing the classes that are relevant to system or Product requirements.

3) Classification & Assembly Structures:-

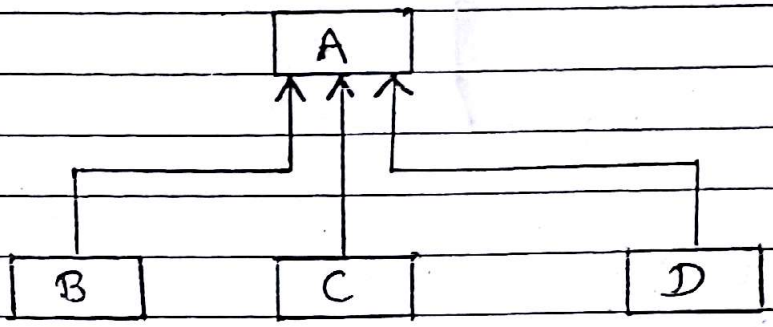
Once classes & objects have been identified, now the analyst begins to focus on classification structure. Each object is considered as a generalization then as a specialization.

Specialization is an entity set which

Name of Lecturer : .....

may include subgroupings of entities that are distinct in some way from other entities in the set.

Generalization is a multiple entity sets are synthesized into higher level entity set on the basis of common features.



Classification Structure Notation

→ The generalization class A is further refined into a set of specializations that are B, C & D. The attributes & operation of class A are inherited by the specializations of the class.

4) Defining Subjects:-

A subject is nothing more than a reference or pointer to more detail in the analysis model. Subject references are generally created for any structure that has more than five or six objects.

Name of Lecturer : Abhishek Jain

5) Instance Connections & Message Paths :-

→ An instance connection is a modeling notation that defines a specific relationship b/w instances of an object.

→ There are some steps to be followed for instance connections :

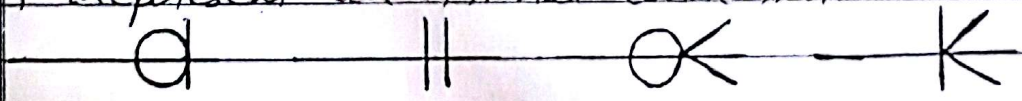
→ Step 1 :- Objects are connected by the simple lines.

Step 2 :- Once the lines have been established, each end is evaluated to determine whether a single (1:1) connection exists or a multiple connection (1 : Many) exists.

For (1:1) connection, the line is annotated with a single bar;

A (1:Many) connection is annotated with a 3-Pronged fork symbol

Finally we add another vertical bar if the connection is mandatory & a circle to represent an optional connection.



0:1 1:1 0:Many 1:Many

Connection Notations

Name of Lecturer :

Step 3:- The resultant representation is made in which all objects relationships are made by single lines & also required connections 1:1, 1:many etc. are made on them.

Message Paths :- Message paths are also known as Message Connections. They are shown by dotted arrows among the objects in which arrow implies the interchange of messages in the model.

Ref :- R8

5.6 Object - Relationship Model :-

The CRC (Class Responsibility - Collaborator) Modeling first establishes the first elements of class & object relationships so the next step is to define those collaborator classes that aid in achieving each responsibility. This establishes the "connection" blw classes.

The Object relationship model can be divided in three steps.

1) Using the CRC index cards, a nlw of collaborator objects can be drawn.

2) Reviewing the CRC Model index cards,

responsibilities & collaborators are evaluated & each unlabelled connection line is named.

3) Once the named relationships have been established, each end is evaluated to determine carefully.

Ref :- BS

[5.7] Data Modeling :-

Definition :- "Data modeling in S/w Engg. is the process of creating a data model for an information system by applying formal data modeling techniques".

Data modeling is already covered in Unit-3rd Point No :- 25, 26, 27, 28, 29

[5.8] Difference Between Data Modeling & OOA

1) Both approaches use the term "object". In data modeling, it is referred as data object whereas in OO approach it is simply an object.

2) Both approaches describe relationships

Name of Lecturer :

b/w objects, but data modeling doesnot concern itself with how these relationships are achieved.

3) Data modeling basically models the data, without concerning about the processing which transforms the data. Analysis modeling focuses on processing also when modeling the data.

Ref. :- R8

5.9 Object Oriented Design Concepts :-

In this review Object-Oriented terminology & introduce a few additional concepts that are relevant to design. Some concepts are

1) Objects, Operations & Messages.
2) Design Issues
3) Classes, Instances & Inheritance
4) Object Descriptions.

1) Objects, Operations & Messages :-

To accomplish OO design, we must establish a mechanism for

Name of Lecturer : Abhishek Jain

1) The representation of Data structure.

2) The specification process.

3) The invocation procedure.

An object might be machines, commands, files, displays, alphanumeric things etc.

When an object is mapped into its OO realizations, it consist of a private data structure & processes, called operations, that may legitimately transform the data structure.

Operations contain control & procedural constructs that may be invoked by a message request to the object to perform one of its operations.

The object also has a shared part i.e. its interface.

Message moves across the interface & specify what operation on the object is desired, but not how the operation is to be performed.

The object that recieves a message

Name of Lecturer : .....

determines how the requested operation is to be implemented.

2) **Design Issues :-**

Bertrand Meyer suggests five criteria for judging a design method's ability to achieve modularity & relates these to the design.

i) **Decomposibility :-** In this the designer can decompose a large problem into subproblems that are easier to solve.

ii) **Composibility :-** This design method ensures that Program Components (modules) are design & built, can be reused to create other systems.

iii) **Understandability :-** In this a program component can be understood without reference to other information or other modules

iv) **Continuity :-** The ability to make small changes in a Program & have these changes adjust themselves with corresponding changes in just one or a very few modules.

v) **Protection :-** An architectural characteristic that will reduce the propagation of

Name of Lecturer: Abhishek Jain

side effects if an error does occur in a given module.

3) Classes, Instances & Inheritance:-

→ Many objects in the real world have reasonably similar characteristics & perform reasonably similar operations

→ Real-world objects are categorized in a same way.

→ All objects are members of a larger class & inherit the properties that have been defined for that class.

→ A class is a set of objects that each has the same characteristics.

→ An individual object is therefore an instance of a larger class

4) Object Description:-

→ A design description of an object (an instance of a class or subclass) can take one of two forms

i) A protocol description that establishes

Name of Lecturer: _____

the interface of an object by defining each message that the object can receive & the related operation that the object performs, when it receives the message.

ii) An implementation description that shows implementation details for each operation implied by a message that is passed by an object.

Implementation details includes information about the internal data structure & procedural details that describes operations.

An implementation description is comprised of the following information.

a) A specification of the object's name & reference to a class.

b) A specification of private data structure with an indication of data items & types.

c) A procedural description of each operation or, alternatively pointers do such procedural descriptions.

Basic Picture

Analysis Model		Design Model
classes	⟷	objects
attributes	⟷	data structures
methods	⟷	algorithms
relationships	⟷	messaging
behavior	⟷	control.

[Ref:- R9]

[5.10] **Object-Oriented Design Methods:-**

→ Both Abott & Booch stated that OOD begins with a natural language (i.e. - English) description of the solution for the s/w realization of real world problem.

→ Later Schaler & Mellor & Coad & Yourdon introduced a more comprehensive notation to support this approach & argued that this activity is more properly characterized as analysis.

Name of Lecturer :

→ By identifying classes & objects, data abstractions are created.

→ By coupling operations to data, modules are specified & a structure for the S/W is established.

Early OOD approaches follows the following steps:

i) Define the Problem.

ii) Develop the informal strategy for the S/W realization of the real-world problem domain.

iii) Formalize the strategy by using the following substeps:

a) Identify objects & their attributes.

b) Identify operations that may be applied to objects.

c) Establish interfaces by showing the relationship b/w objects & operations.

iv) Reapply steps (ii), (iii), & (iv) recursively. All four steps should performed during S/W req. analysis.

Name of Lecturer: Abhishek Jain

5) Refine the work done during OOA, looking for subclasses, message characteristics, & other elaboration of detail.

6) Represent the data structure(s) associated with object attributes.

7) Represent the procedural detail associated with such operation.

[Ref:- 10]

5.11 Unified Approach:-

OO concepts were introduced much earlier than UML. So at that time there were no standard methodologies to organize & consolidate the OO development. At that point of time UML came into Picture.

A unified approach evolved from the combined efforts of Booch, Rumbaugh & Jacobson. This unified approach is known as UML.

→ The main aim to develop UML is to define a general purpose modeling language which all S/w developers

can use & also it needs to be made simple to understand & use. Ref:- R10

5.12 Introduction To Unified Modeling Language (UML):-

Definition:- "The UML is a standard language for writing S/w blueprints".

→ UML may be used for specifying, visualizing, constructing & documenting the artifacts of S/w systems.

→ UML is a Pictorial language used to make S/w blueprints.

→ UML is not a Programming language but tools can be used to generate code in various languages using UML diagrams.

→ UML is used for Problem Simplification & reuse.

→ UML diagrams are not only made for developers but also for business users, common People & anybody interested to understand the system.

→ At the conclusion the goal of UML can be defined as a simple modeling mechanism to model all Possible Practical systems in today's Complex environment. Ref:- R12

Name of Lecturer : Abhishek Jain

5.13 Conceptual Model of UML :-

UML requires three major elements to build its conceptual model ; it includes :

1) UML Building Blocks.
2) Rules to connect the building blocks.
3) Common mechanisms that apply throughout UML.

Ref :- R1

5.14 Building Blocks in UML :-

To build the blocks in UML for an application, things (objects) are defined & the relationships among them are defined; At last complete UML digram is made.

The building blocks of UML can be defined as :

1) Things
2) Relationships
3) UML Diagrams.

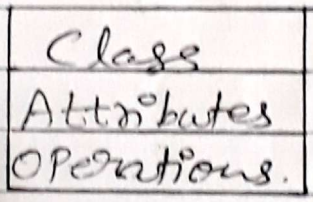
Name of Lecturer : .....

1) **Things:-** Things are most important building blocks of UML. Things can be:

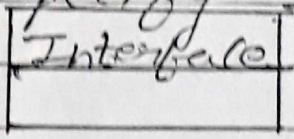
i) Structural.
ii) Behavioral.
iii) Grouping.
iv) Annotational.

i) **Structural:-** The structural things define the static part of the model. They represent physical & conceptual elements. Following are the brief descriptions of the structural things.

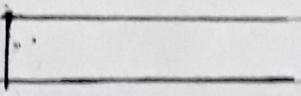
→ **Class:-** Class represents set of objects having similar responsibilities.



→ **Interface:-** Interface defines a set of operations which specify the responsibility of a class.

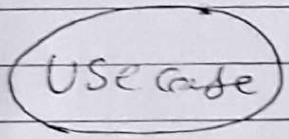


→ **Collaborations:-** Collaboration defines interaction blw elements.

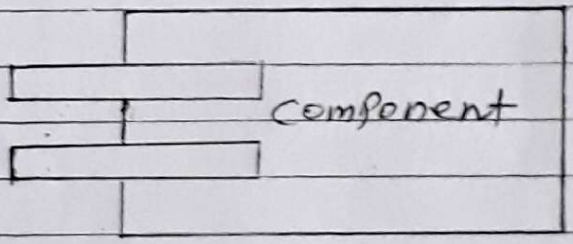


Name of Lecturer : Abhishek Jain.

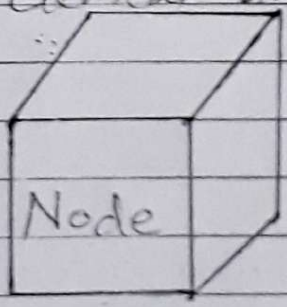
→ Use Case :- Use case represents a set of actions performed by a system for a specific goal.



→ Component :- Component describes physical part of a system.



→ Node :- A node can be defined as a physical element that exist at given time.



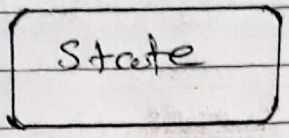
ii) Behavioral Things :- A behavioral things consists of the dynamic parts of UML model. Following are the behavioral things.

→ Interaction :- It is defined as a behavior

that consist of a group of messages exchanged among elements to accomplish a specific task.

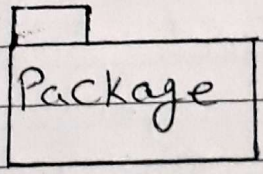
Message →

→ **State Machine :-** A state machine shows behavior that specifies the sequence of states of an object.



iii) **Grouping Things :-** Grouping things can be defined as a mechanism to group elements of a UML model together. There is only grouping thing available:

→ **Package :-** Package is the only one grouping thing available for gathering structural & behavioral things.



iv) **Annotational Things :-** It can be defined as a mechanism to capture remarks, descriptions & comments of UML model elements. Note is the only one Annotational thing available.

Name of Lecturer : Abhishek Jain

are used to make it a complete one.

→ UML includes the following nine diagrams & the details are described below;

1) Class Diagram
2) Object Diagram
3) Use case Diagram
4) Sequence Diagram
5) Collaboration Diagram
6) Activity Diagram
7) Statechart Diagram
8) Deployment Diagram
9) Component Diagram.

Ref. :- R11

Lecture No. .....

5.15 Class Diagram in UML :-

→ The class diagram is a static diagram. It represents the static view of an application.

→ Class diagram is not only used for visualizing, describing & documenting different aspects of a system but also for constructing executable code of the s/w application.

→ The class diagram describes the attributes & operations of a class & also the constraints imposed on the system.

→ The class diagram shows a collection of classes, interfaces, associations, collaborations & constraints. It is also known as a structural diagram.

→ Purpose :-

→ Analysis & design of the static view of an application.

→ Describe responsibilities of a system.

→ Base for component & deployment diagrams.

→ Forward & reverse engg.

Name of Lecturer : Abhishek Jain

→ How to draw a class Diagram:-

i) Place the name of the class in the first partition (centered, bolded & Capitalized). List the attributes in the second Partition & write operations in to the third.

CLASSNAME
attribute
operation.

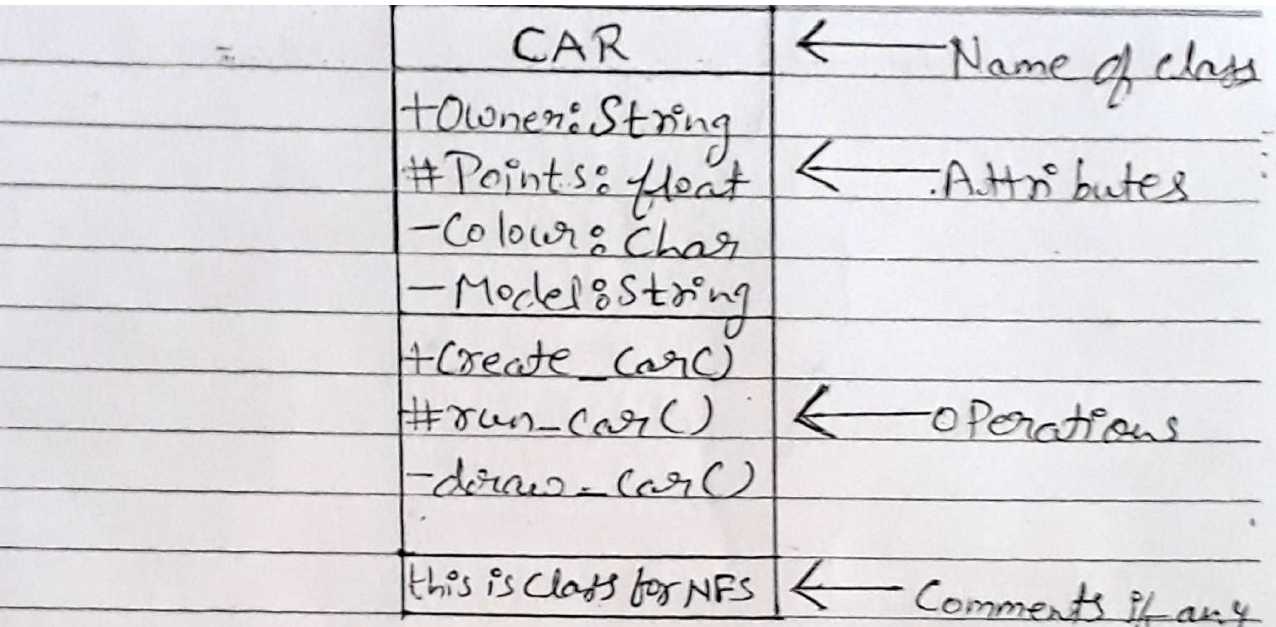
ii) The class name is a compulsory field & rest are optional field.

iii) The attributes are mentioned text. The Public attributes start with '+' sign, Protected with '#' sign & the Private attributes with '−' sign.

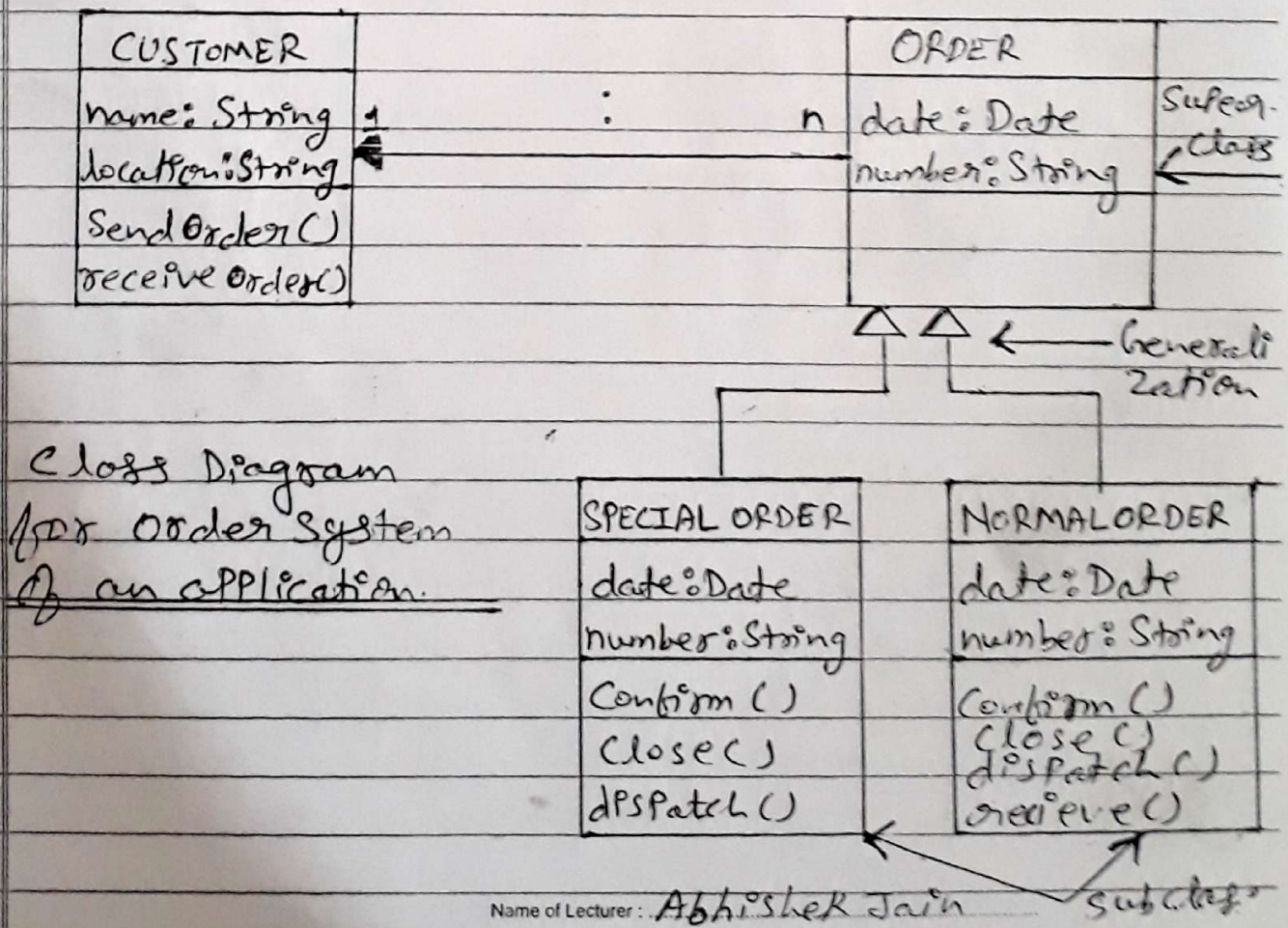
iv) The operations are also represented in a similar way. Like the signs +, #, & − are used for Public, Protected & Private operations & functions.

v) Lastly, there is an optional field for writing components & comments.

Name of Lecturer : .....



Representation of a class car.



Class Diagram for Order System of an application.

Name of Lecturer : Abhishek Jain

→ Now the following is an example of an order system of an application.

→ First of all Order & Customer are identified as the two elements & they have one to many relationship because a customer can have multiple orders.

→ We would keep order class is an abstract class & it has two concrete classes (inheritence relationship) special Order & Normal order.

→ The two inherited classes have all the properties as the order class. In addition they have additional functions like dispatch() & receive().

Ref:- R12

5-16 Object Diagram in UML :-

→ Object diagrams are derived from class diagrams, so object diagrams are dependent upon class diagrams

→ Object diagrams represent an instance of a class diagram.

Name of Lecturer : .....

→ The basic concepts are similar for class diagrams & object diagrams.

→ Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.

→ Object diagrams are used to render a set of objects & their relationships as an instance.

Purpose :- The difference b/w a class diagram & an object diagram is that the class diagram represents an abstract diagram or model consisting of classes & their relationships. But an object diagram represents an instance at a particular moment which is concrete in nature.

The Purpose of object diagram is:

→ Forward & reverse engg.

→ Object relationship of a system.

→ Static view of an interaction.

→ Understand Object behavior & their relationships from Practical Perspectives.

Name of Lecturer : Abhishek Jain

How to draw an Object Diagram:-

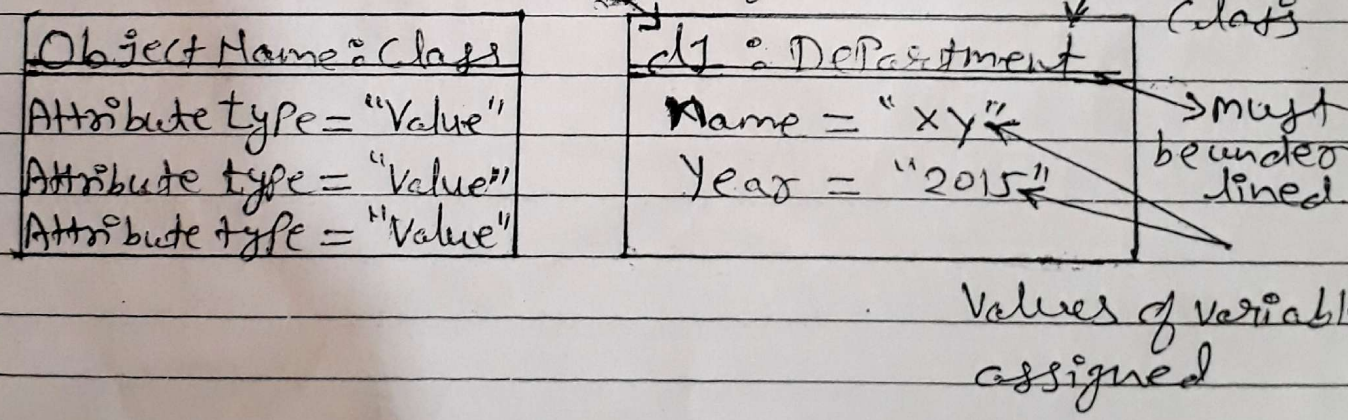
i) Representation of a class & an object is same :. except for the fact that instead of just name of the object, the object diagram contains both name of the object & the name of the class to which it belongs.

Object name : class

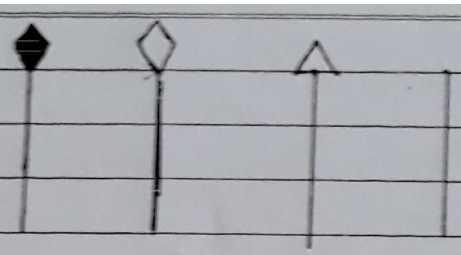
→ must be underlined

ii) Instead of the term association, as in class diagram, Object diagram used the term link. Rest of the conventions are same

iii) As with class diagram, you can list object attributes in a separate compartment, here in Object diagram object attributes must have values assigned to them

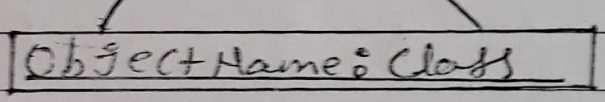


iv) Links are instances of associations. You can draw link using the lines used in class diagrams

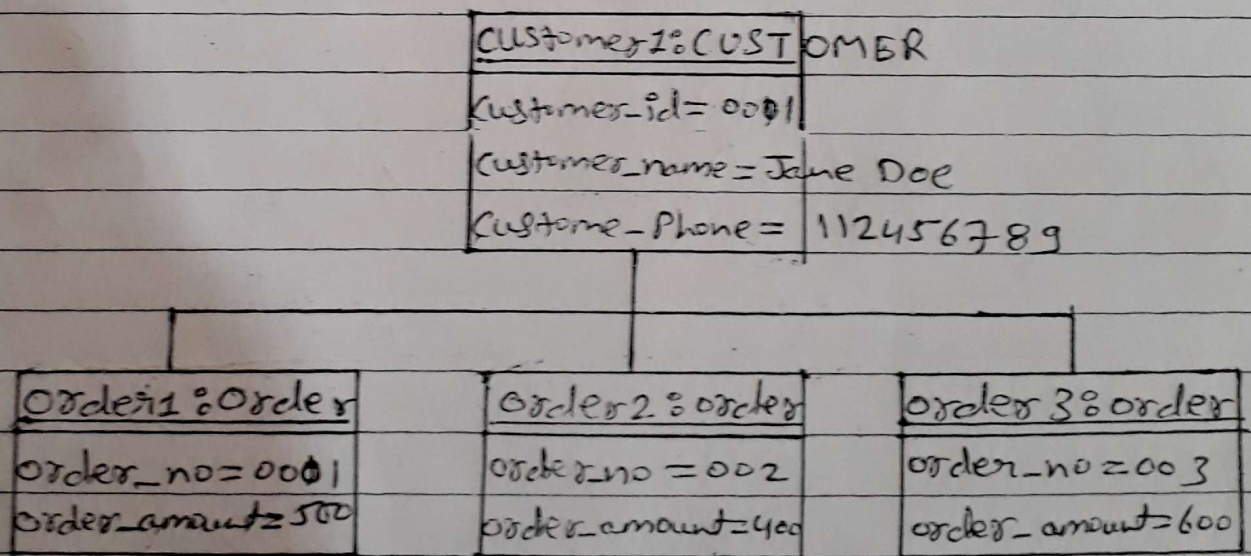


Links

vi) Object that can fulfill more than one role can be self linked. For ex: If A is an administrative assistant, also fulfilled the role of a marketing assistant, & the two positions are linked, A's instance of the two classes will be self-linked.



Self linked.



Object diagram of Customer Order

Ref :- R13

Name of Lecturer Abhishek Jain

5.17 Use Case Diagram in UML :-

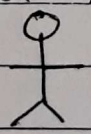
→ Use case diagrams model the dynamic aspects or behavior of the system.

→ A use case diagram depicts:

→ Use cases are the description of a set of sequences of actions, that a system performs yielding an observable result of value to an actor.

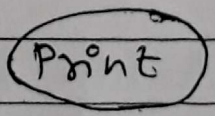
→ Use case diagrams model the functionality of a system using actors & use cases. Use cases are services or functions provided by the system to its users.

→ Actors are the set of roles that users of use cases play when interacting with the use cases.



Actor.

→ Use cases are represented by ovals. Labels with ovals represents the functions of the system.



→ Actors are the users of a system. They represent who will access the functions of the system.

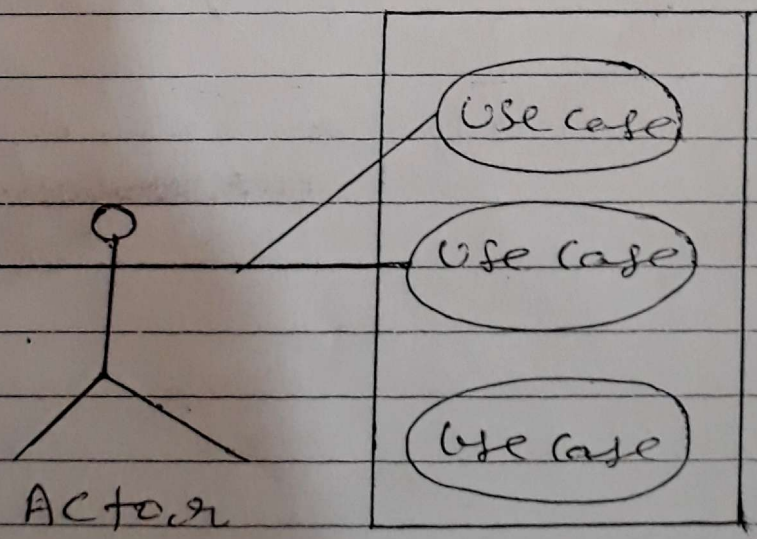
→ Relationship blw an actor & a use case is illustrated with a single line. For relationships among use cases, use arrows labelled "uses" or "extends" or "includes".

→ A "uses" relationship indicates that one use case is needed by another in order to perform a task.

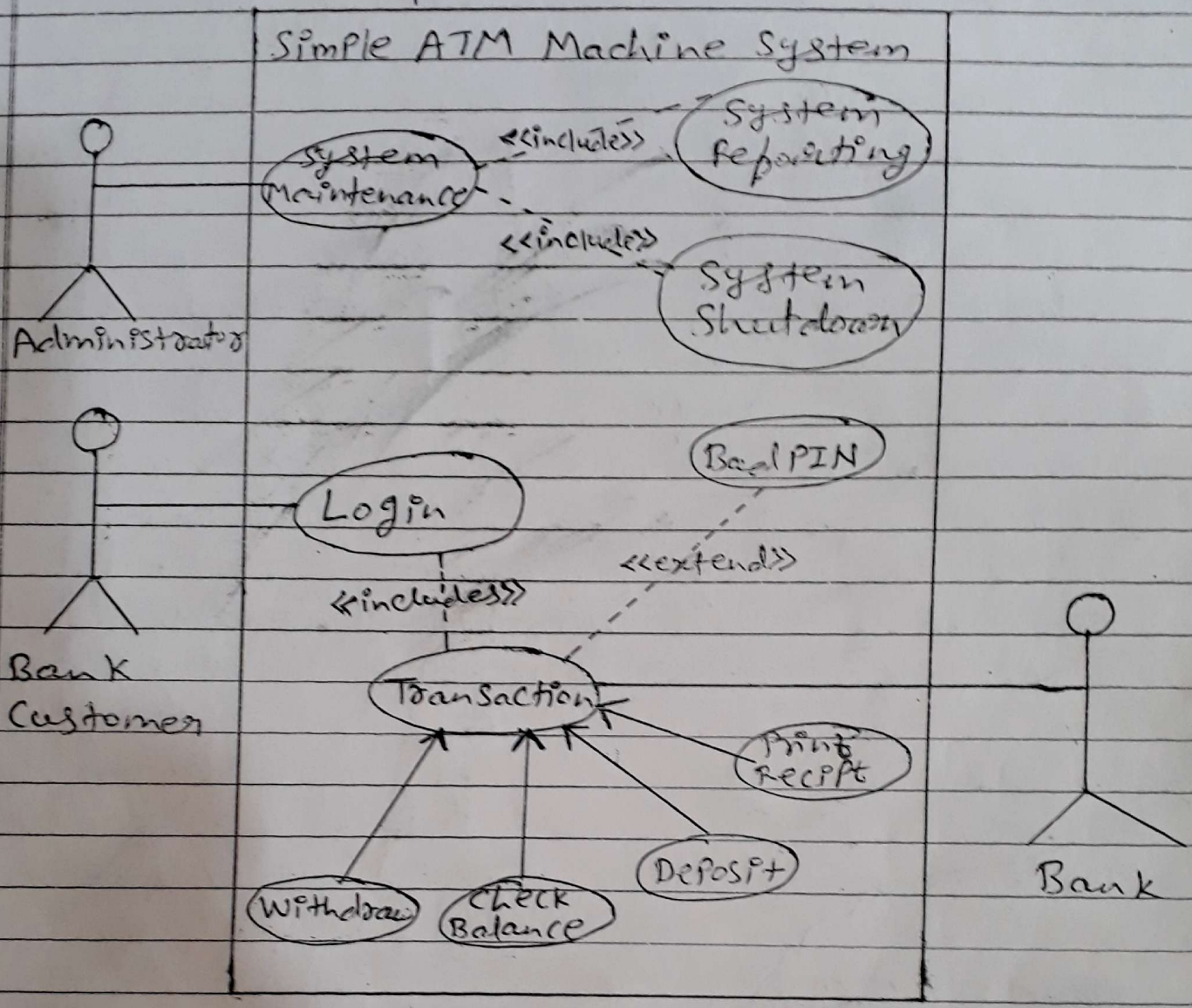
→ An "extends" relationship indicates alternative options under a certain use case.

→ An "include" relationship indicates a particular use case includes one or more than one use case.

→ Draw system boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.



Name of Lecturer : Abhishek Jain



ATM Machine System

Ref:- R14, R15

5-18 Sequence Diagram in UML :-

→ A sequence diagram is an interaction diagram that shows how processes operate with one another & in what order.

→ It represents a Message Sequence chart.

→ A sequence diagram shows object interactions arranged in time sequence.

→ It depicts the sequence of messages exchanged b/w the objects needed to carry out the functionality of the scenarios.

→ Sequence diagram models the dynamic aspects of a system.

→ Sequence diagrams are sometimes called event diagrams or event scenarios.

Basic sequence diagram symbols & Notations.

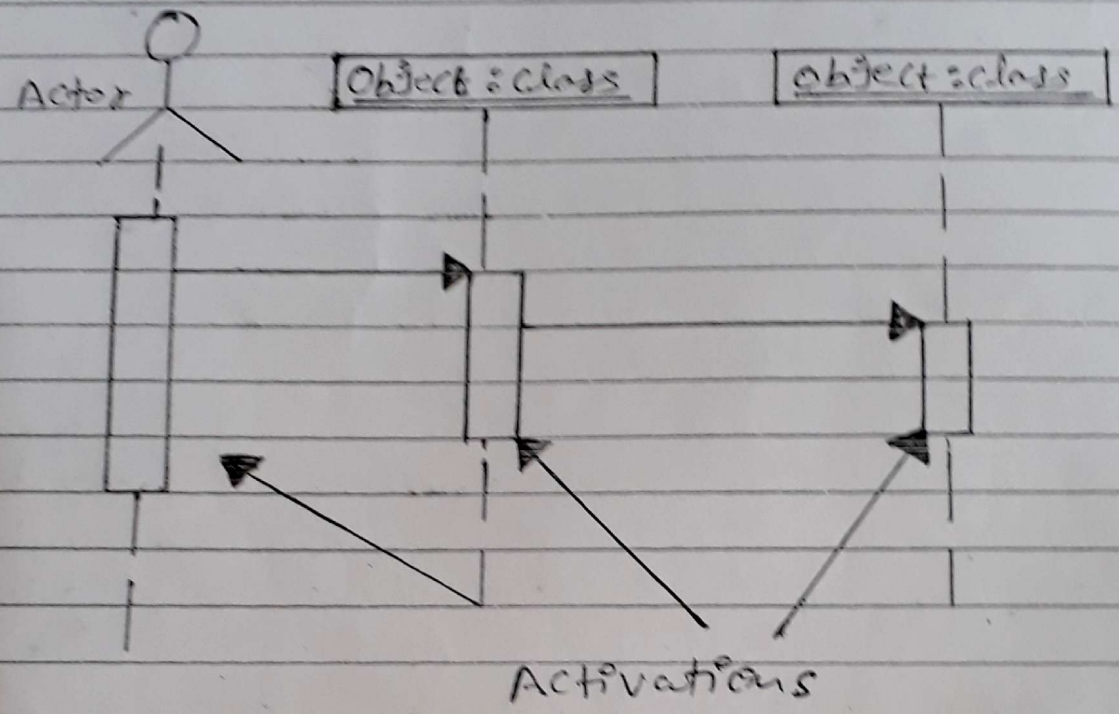
→ Class Roles :- Class roles describes the way an object will behave in context.

Object : Class

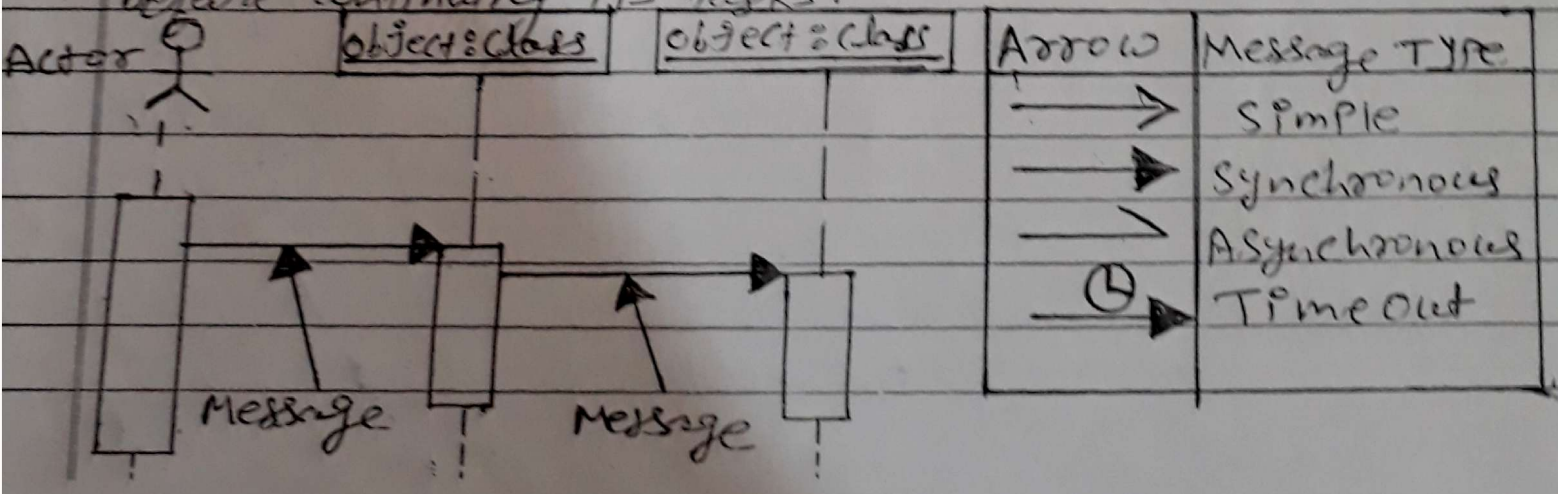
→ It don't list object attributes.

Name of Lecturer : Abhishek Jain.

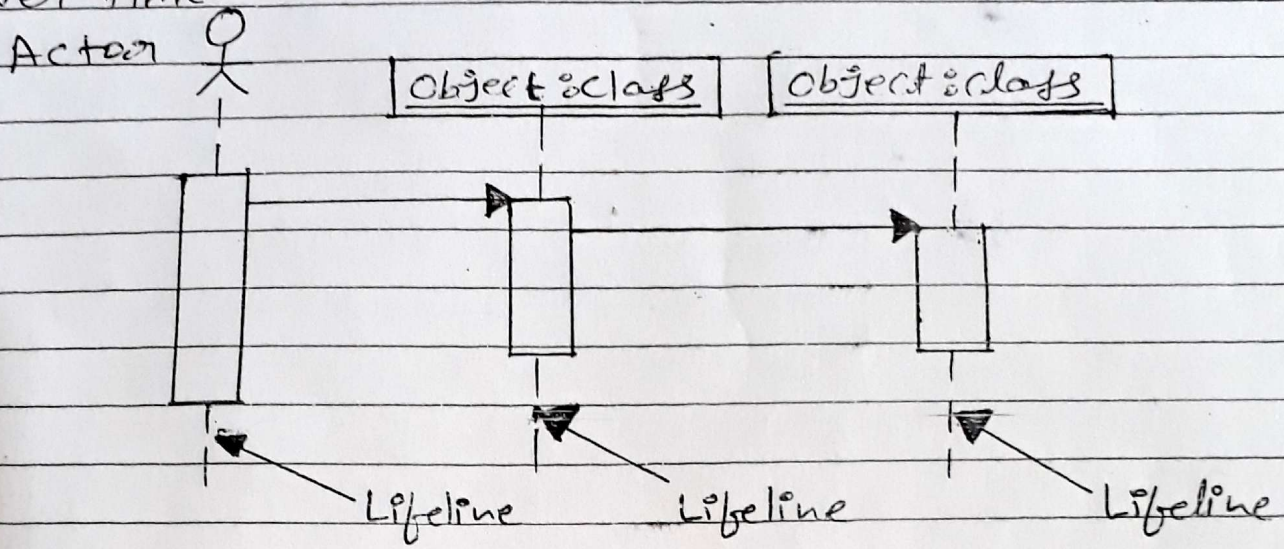
→ **Activation :-** Activation boxes represent the time an object needs to complete a task.



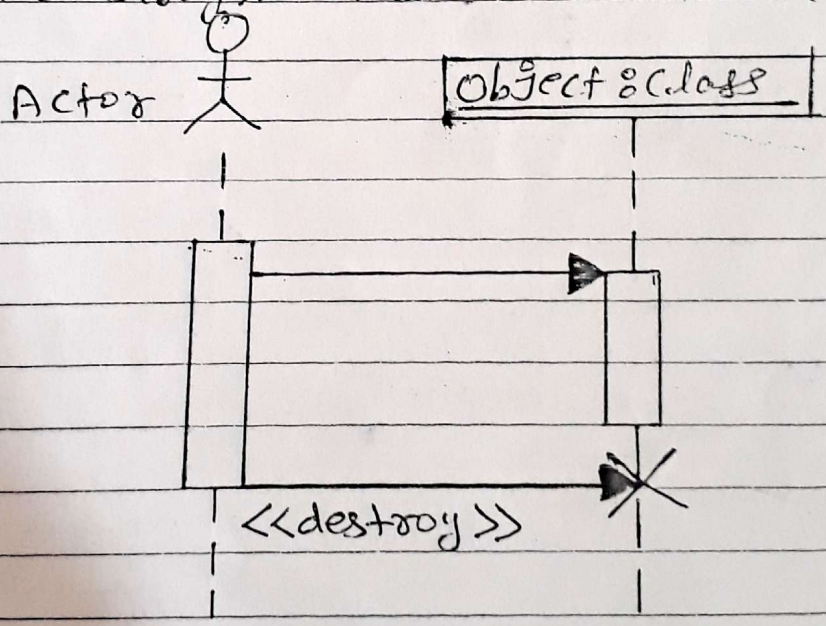
→ **Messages :-** Messages are arrows that represent communication b/w objects. use half arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



→ **Lifelines:-** Lifelines are vertical dashed lines that indicate the object's Presence over time.



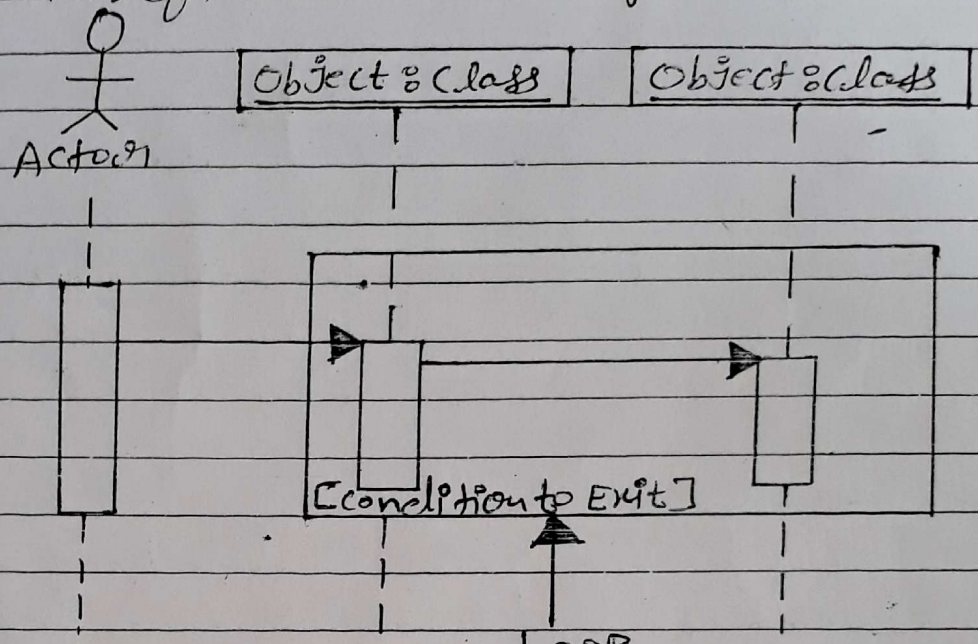
→ **Destroying objects:-** Objects can be terminated (destroyed) early using an arrow labelled "<<destroy>>" that point to an X.



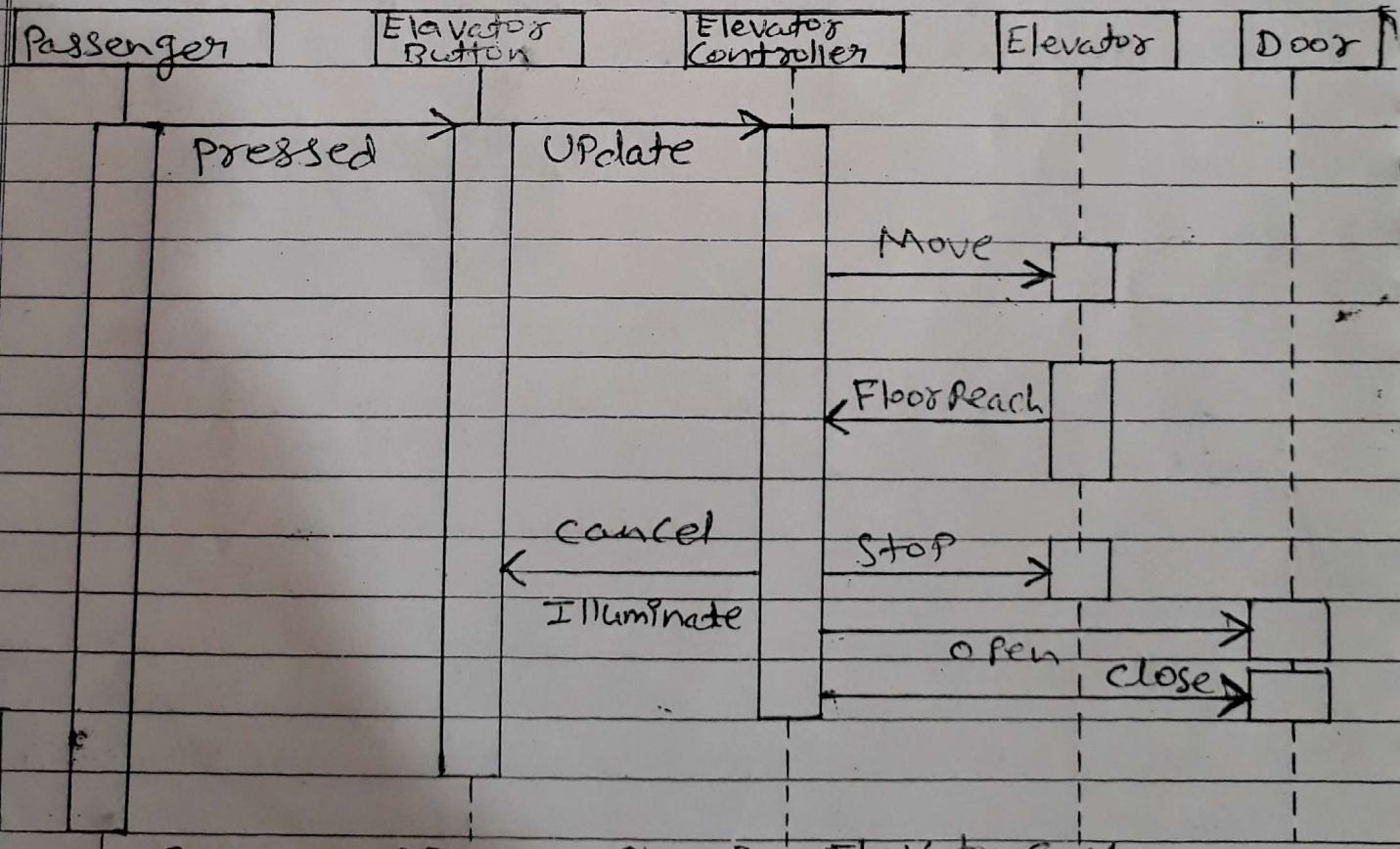
→ **Loops:-** A repetition or loop within a sequence diagram is depicted as a rectangle.

Name of Lecturer : Abhishek Jain

Place the condition for exiting the loop at the bottom left corner in Square brackets [].



Examples -



e/g:-
16, R17

Sequence Diagram Showing Elevator System

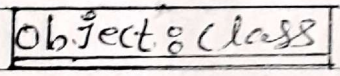
[5.19] **Collaboration Diagram in UML :-**

→ A collaboration diagram describes interactions among objects in terms of sequenced messages.

→ Collaboration diagrams represent a combination of information taken from class, sequence & use case diagrams describing both the static structure & dynamic behavior of a system.

→ Basic collaboration Diagram symbols & Notations.

→ **Class Roles :-** Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.



→ **Association/Relationship Roles :-**

→ A link connecting the associated objects. We can draw association roles using simple lines.

→ **Messages :-** Unlike sequence diagrams, Collaboration diagrams do not have an explicit way to denote time. Instead of denoting time, a number messages is used for

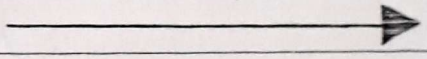
execution.

→ Sequence number can become nested Some times. For example, nested messages under the first message are labeled 1.1, 1.2, 1.3 & so on.

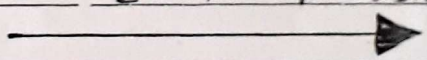
→ The condition for a message is usually Placed in Square brackets immidiately following the sequence number.

→ Use a * after the sequence number to indicate a loop.

1.4 [condition] : message name



1.4* [loop expression] : message name



→ In collaboration diagram the focus is on message passed b/w objects.

Collaboration diagrams are represented as follows

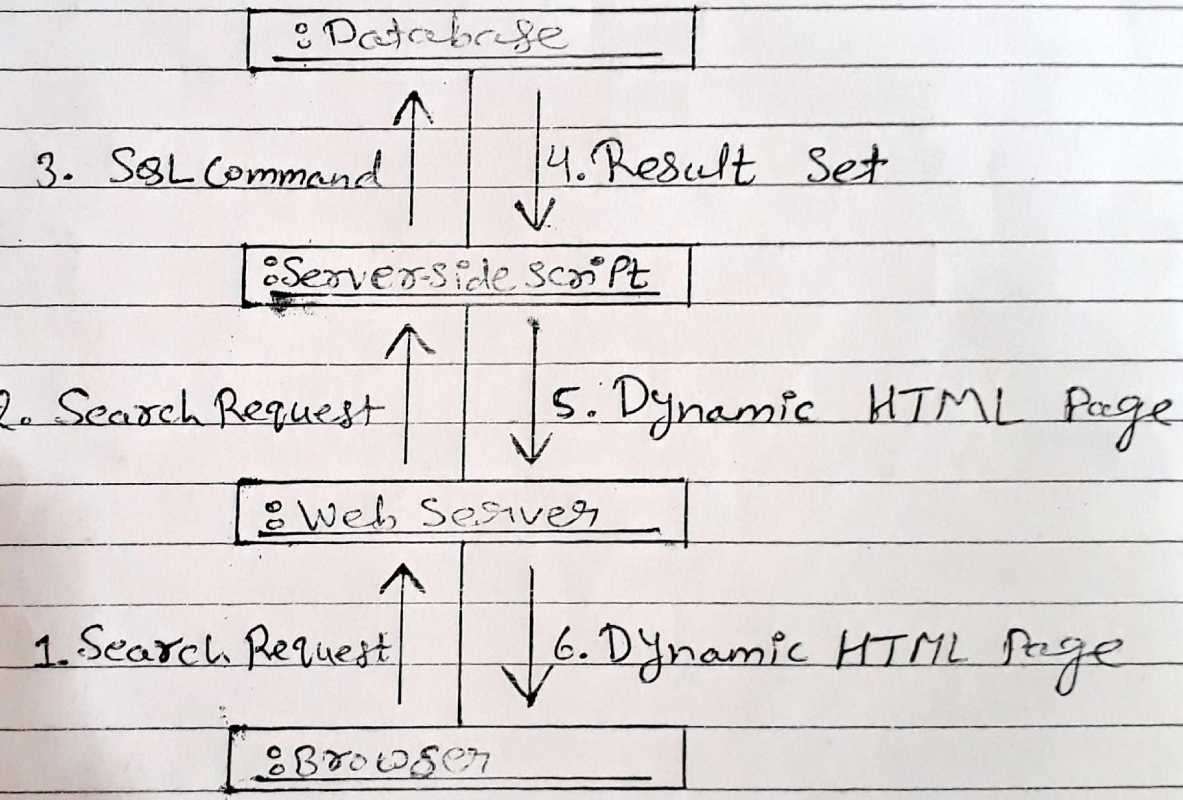
i) Objects that Participate in the interaction are Placed as vertices in a graph

ii) Links that connect these objects are rendered as arcs of the diagram

iii) The links are supplemented with messages that object sends & recieve.

iv) There is a sequence no. to indicate time ordering of a message.

v) The message is prefixed with a number



Collaboration Diagram Showing the flow from DB to Server

Ref.:- R18, R19, R20

Name of Lecturer : Abhishek Jain

5.20 Activity Diagram in UML :-

→ Activity diagram also describes the dynamic aspects of the system.

→ AD is basically a flow to represent the flow from one activity to another activity.

→ The activity can be described as an operation of the system.

→ The flow can be sequential, branched or concurrent.

→ Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to construct the executable system.

→ The only missing thing in activity diagram is the message part. Activity diagram does not show any message from one activity to another.

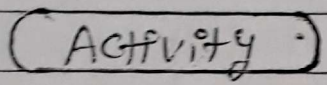
→ Although the diagram looks like a flow chart but it is not.

→ The difference b/w interaction diagram & activity diagram is that interaction diagram focuses on the flow of control from object to object whereas activity diagrams focuses on the flow of control from

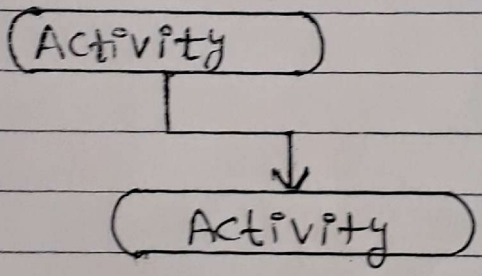
activity to activity.

→ **Basic Activity Diagram Symbols & Notations:**

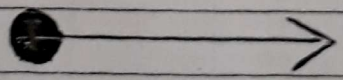
→ **Action States:-** Action States represent the noninterruptable actions of objects. We can draw an action state by using a rectangle with rounded corners.



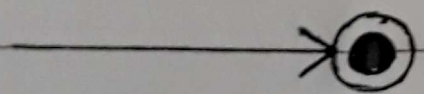
→ **Action Flow:-** Action flow arrows illustrate the relationships among action states.



→ **Initial State:-** A filled circle followed by an arrow represents the initial action state.

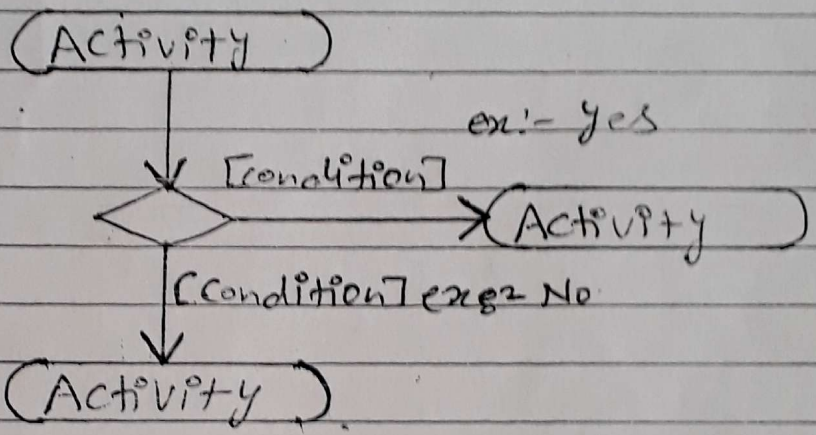


→ **Final State:-** An arrow pointing to a filled circle nested inside another circle represents the final action state.

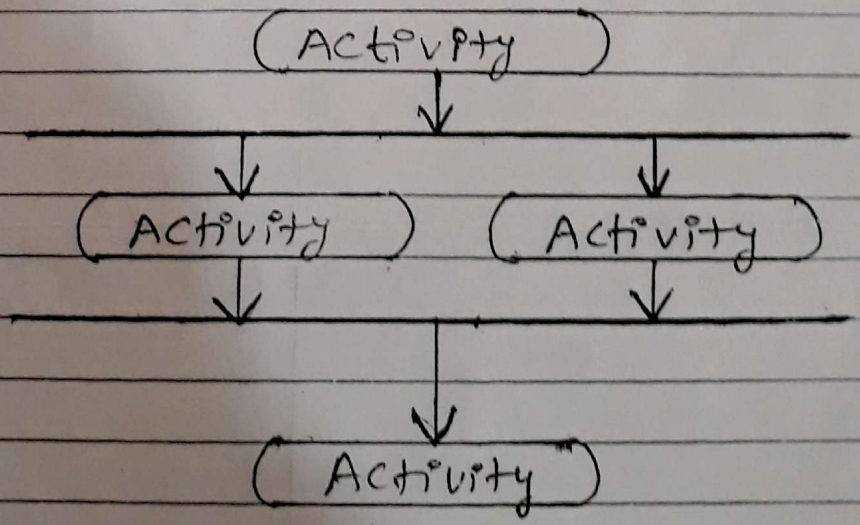


Name of Lecturer Abhishek Jain

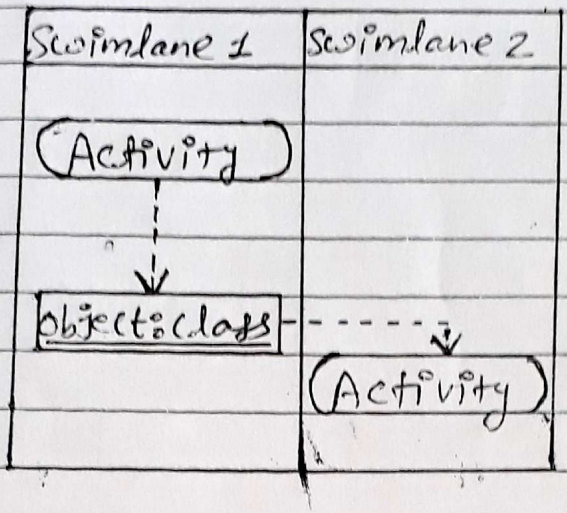
→ Branching :- A diamond represents a decision with alternate paths. The outgoing alternates should be labeled with a condition or guard expression. Iteration is also possible. We can also label one of the paths "else".



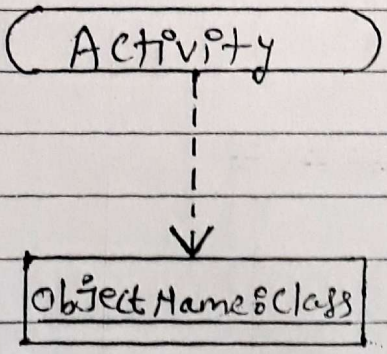
→ Synchronization :- A synchronization bar helps illustrate Parallel transitions. Synchronization is also called forking & Joining.



→ **Swimlanes :-** Swimlanes group related activities into one column.

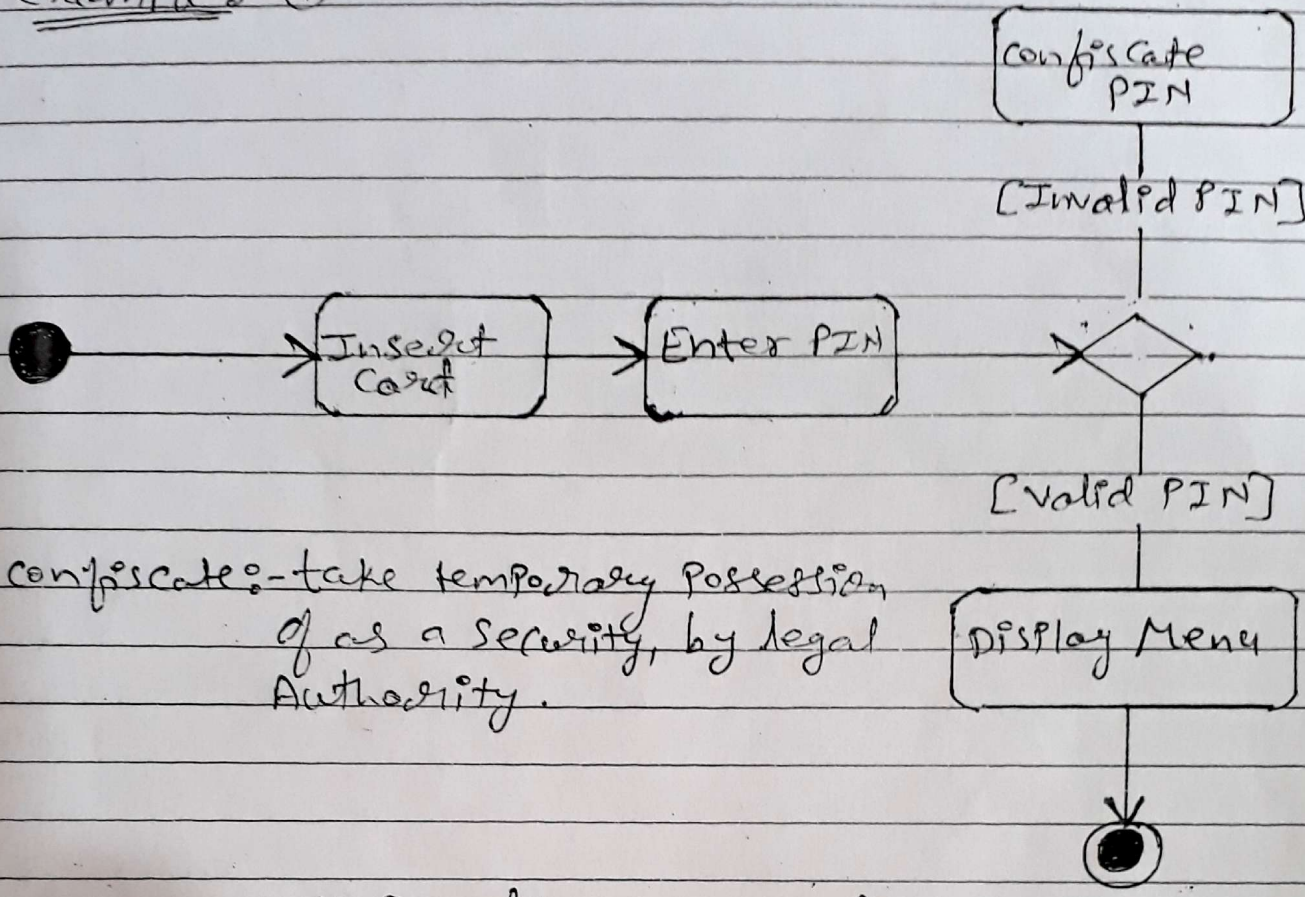


→ **Object Flow :-** Object flow refers to the creation & modification of objects by activities. An object flow arrow from an activity to an object means that the activity creates or influence the object. An object flow arrow from an object to an action or activity indicates that the action state uses the object.



Name of Lecturer : Abhishek Jain

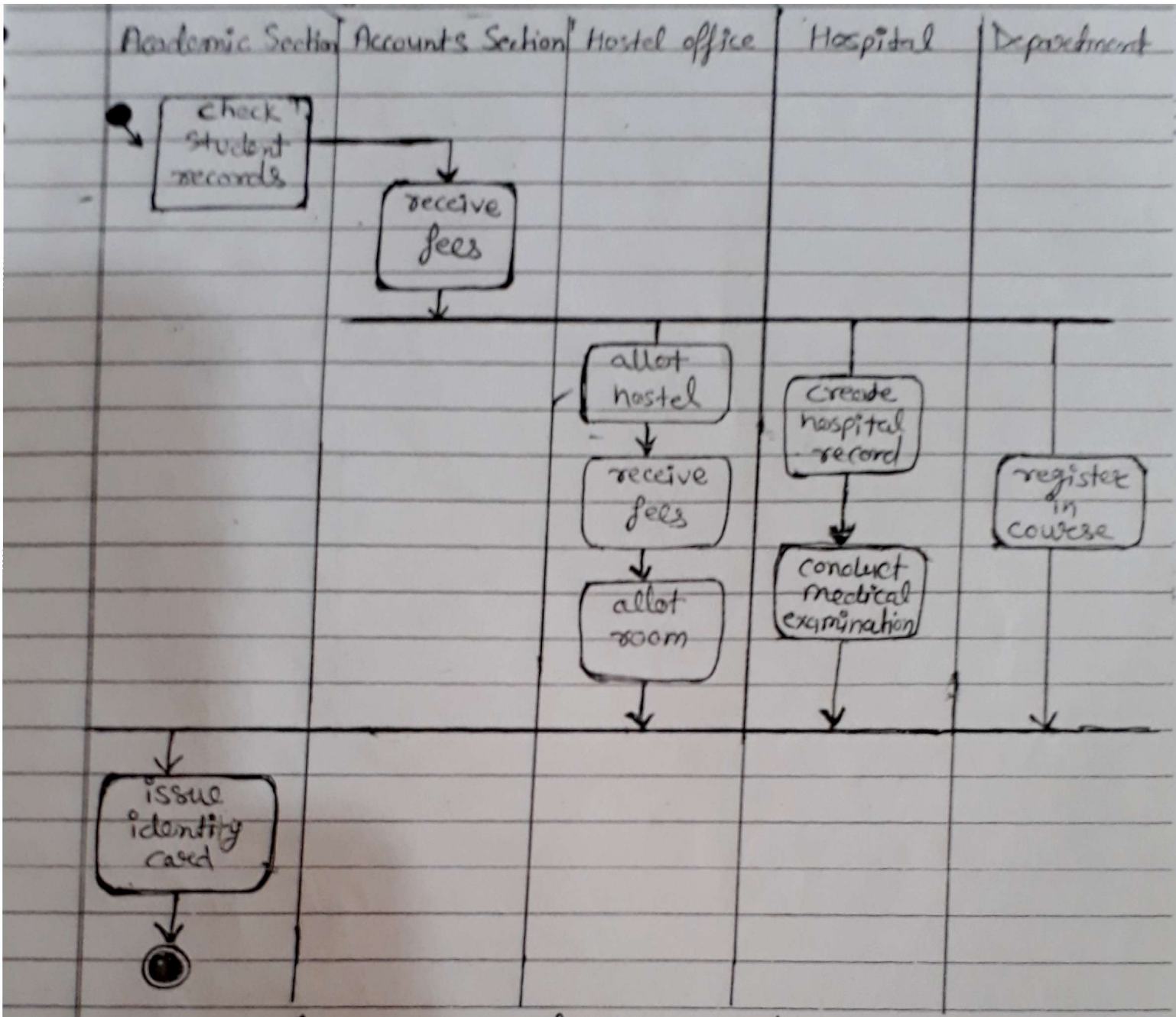
example:- ①



Saving confiscate:- take temporary possession of as a security, by legal Authority.

Activity diagram of ATM Processing

example:- ② The Student admission process in IIT is shown as an activity diagram. This shows the Part Played by different components of the institute in the admission Procedure. After the fees are recieved at the account section, parallel activities start at the hostel Office, hospital & the Department. After all these activities are completed (this synchronization is represented as a horizontal line), the identity card can be issued to a student by the academic section.



Activity diagram for student admission procedure at IIT

Ref :- R21, R22, R23

Name of Lecturer : Abhishek Jain

5.21 StateChart Diagram in UML :-

→ State chart diagram depicts the dynamic behaviour of a system.

→ State Chart diagram describes a State Machine.

→ A state chart diagram defines states it is used to model lifetime of an Object.

→ State chart diagram defines different states of an object during its lifetime. (the flow of control from one state to another state).

→ States are changed by events.

→ State Chart diagram are useful to model reactive systems.

→ Reactive systems can be defined as a System that responds to external or internal events.

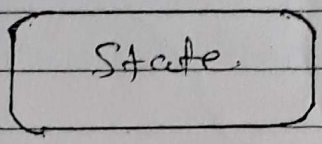
→ States are defined as a condition in which an object exists & it changes when some event is triggered.

→ State chart diagram is to model life time of an object from creation to termination.

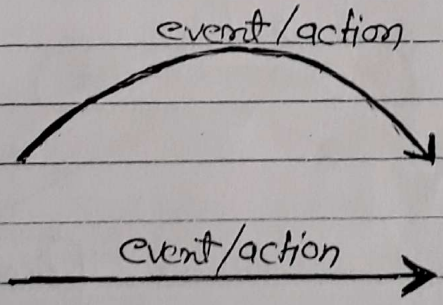
→ State chart diagram shows flow of control from state to state. Whereas in activity diagram, flow of control is shown from activity to activity.

→ Basic StateChart Diagram Symbols & Notations :-

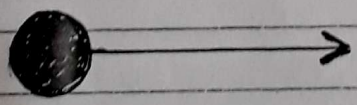
→ States :- States represent situation during the life of an object. We can illustrate a statechart diagram by using a rectangle with rounded corners.



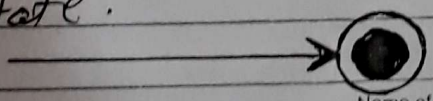
→ Transition :- A solid arrow represents the path b/w different states of an object. Label the transition with the event that triggered it & the action that results from it.



→ Initial State :- A filled circle followed by an arrow represents the objects initial state.



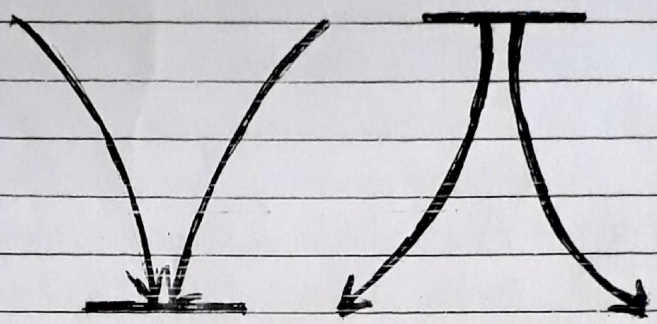
→ Final State :- An arrow pointing to a circle nested inside another circle represents the objects final state.



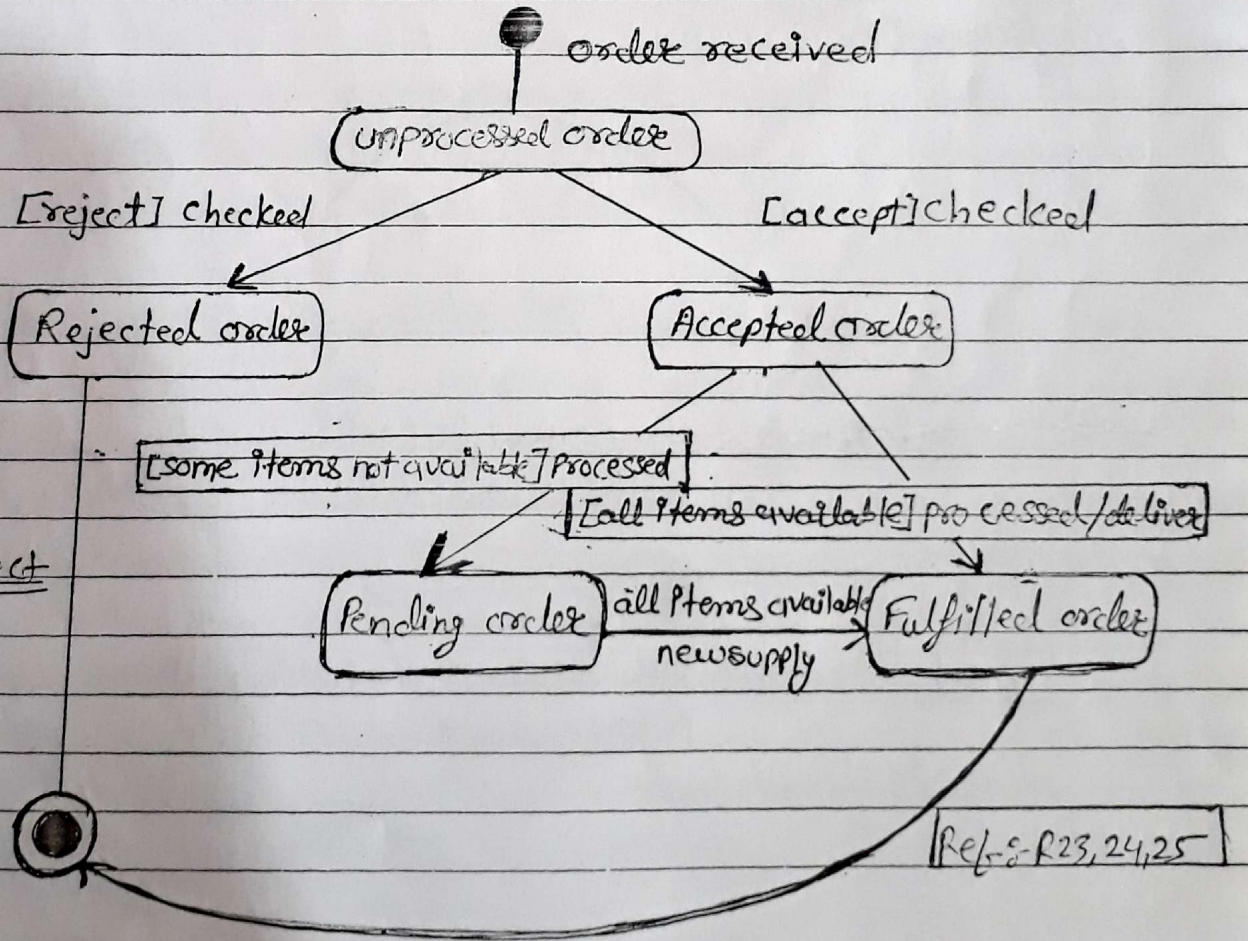
Name of Lecturer : Abhishek Jain

→ Synchronization & Splitting of control :-

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states.



Synchronization Splitting of control



ate Chart
'agram for
order object

5.22 Deployment Diagram in UML :-

→ Deployment diagrams are used to describe the static deployment view of a system.

→ Deployment diagrams describes the Physical resources of a system including nodes, components, & connections.

→ Deployment diagrams are used for describing the H/w components where S/w components are deployed.

→ Component diagrams are used to describe the components & deployment diagrams shows how they are deployed in H/w.

→ This diagram model the Topology of the H/w on which the system executes. (Topology means the configuration of a communication network).

→ It is used to Model Embedded, Clint/server & Distributed sys.

→ Basic Deployment Diagram Symbols & Notations :-

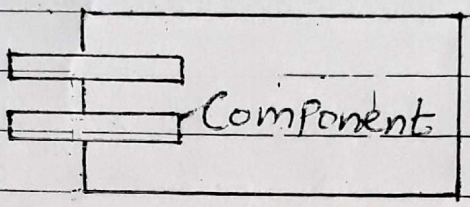
→ Component :- A component represents a modular part of a system that encapsulates its contents & whose manifestation (a clear appearance) is replaceable within its environment.

→ A component defines its behavior in terms of

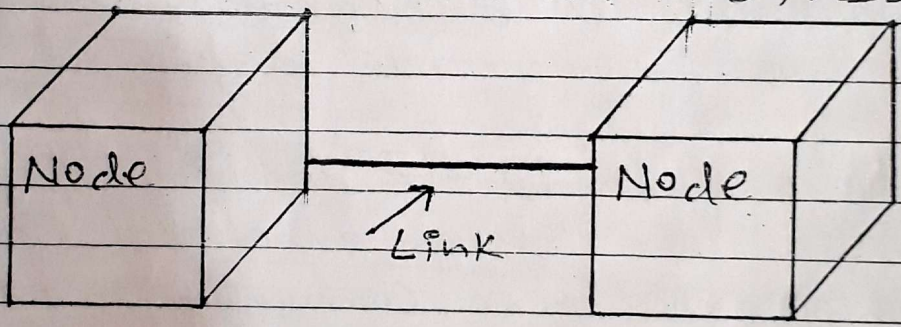
Name of Lecturer Abhishek Jain

Provided & required interfaces. Components are the S/W elements that are deployed to the H/W System.

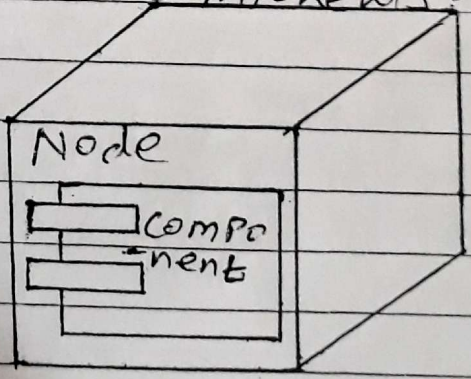
→ S/W comomponents are; (ex:- web applicatⁿ, database).



→ Link /Connection:- Association refers to Physical connection b/w nodes. different nodes are connected (ex:- JDBC, REST, RMI)



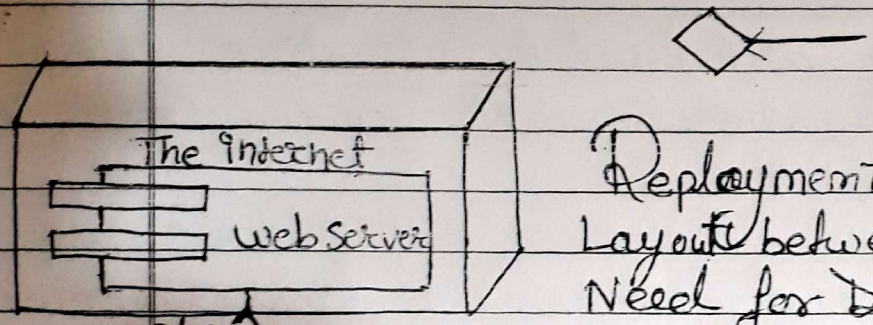
→ Node:- A node is a Physical resource that executes code components. Nodes are the H/W elements of the deployment system. Anything that Performs work in the system. (web server, database server, all server).



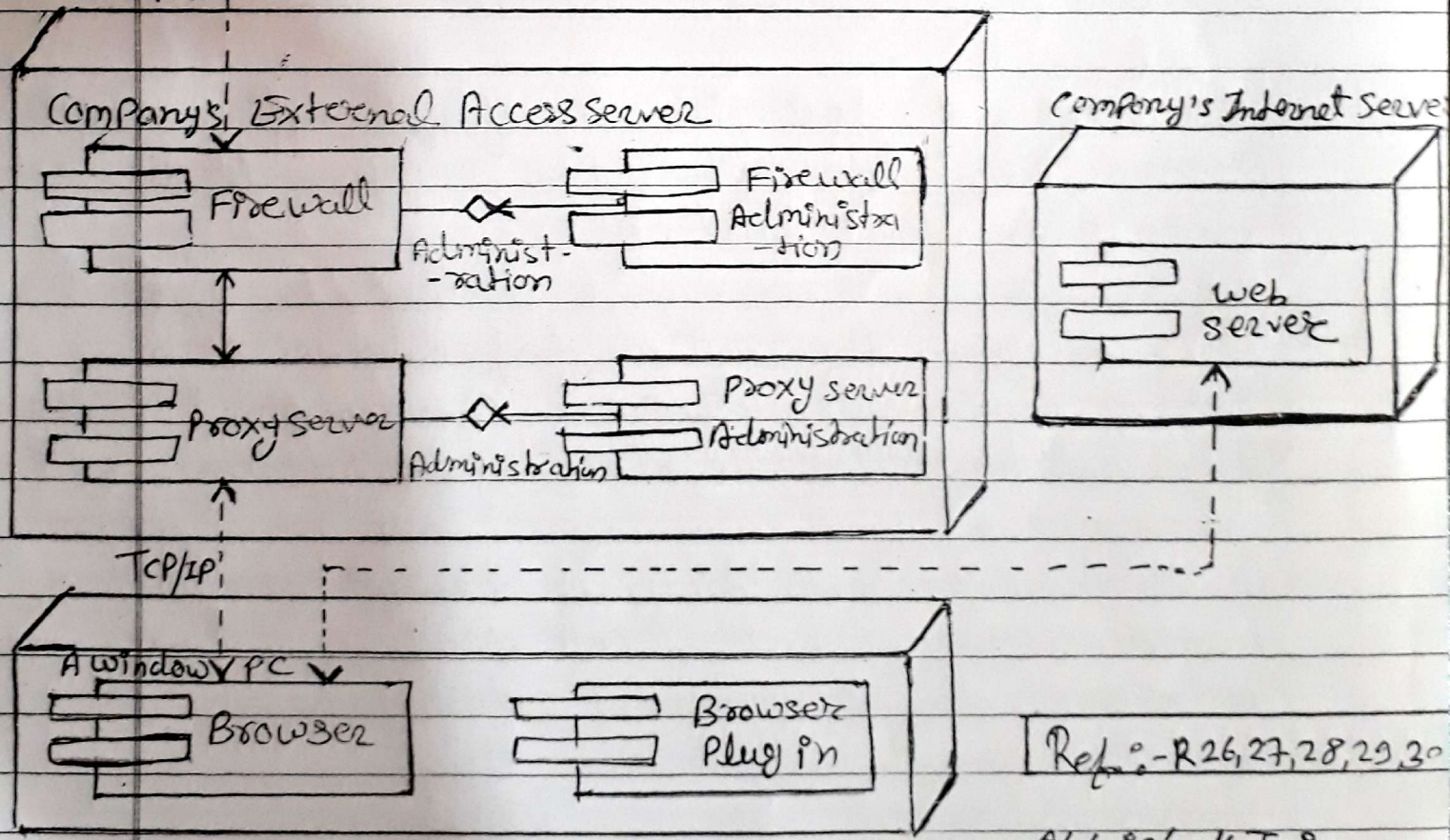
→ Dependencies:- Exists b/w components & can be specified by utilizing predefined or user-defined conventions.

- - - - - - - - - - →

→ Associations:- It is used to display communi-cation relation b/w components.



Replayment Diagram showing TCP/I Layout between A PC and AS Server Need for Deployment Diagrams.



Ref:- R26, 27, 28, 29, 30

Name of Lecturer: Abhishek Jain

5:23 Component Diagram in UML :-

→ Component diagrams are used to model physical aspects of a system.

→ Physical aspects are the elements like executables, libraries, files, documents etc. which resides in a node.

→ Component diagram are used to visualize the organization & relationships among components in a system.

→ These diagrams are also used to make executable systems.

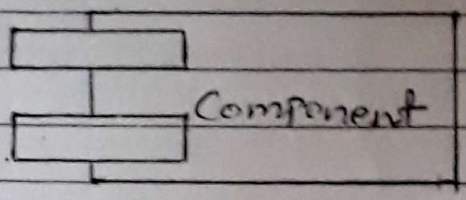
→ It does not describe the functionality of the system but it describes the components used to make those functionalities.

→ This diagram shows the dependencies among a set of components, where a component is the physical & replaceable part of a system.

→ A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.

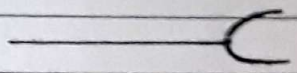
→ Basic Component Diagram Symbols & Notations:-

→ Component :- A component is a physical building block of the system. It is represented as a rectangle with tabs.

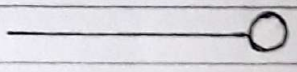


→ Interface :- Interface are of two types.

(a) Required Interface :-



(b) Provided Interface :-

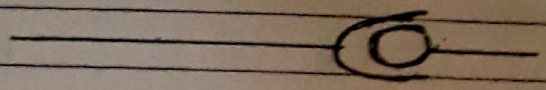


→ Interface describes a group of operations used or created by components.

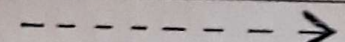
→ To depict the fact that the provided interface is used, or the required interface provided, by another element use the Assembly connector.

→ Assembly connector :- An assembly connector bridges a component's required interface (Component 1) with the provided interface

of another Component (Component 2), typically in a Component diagram.

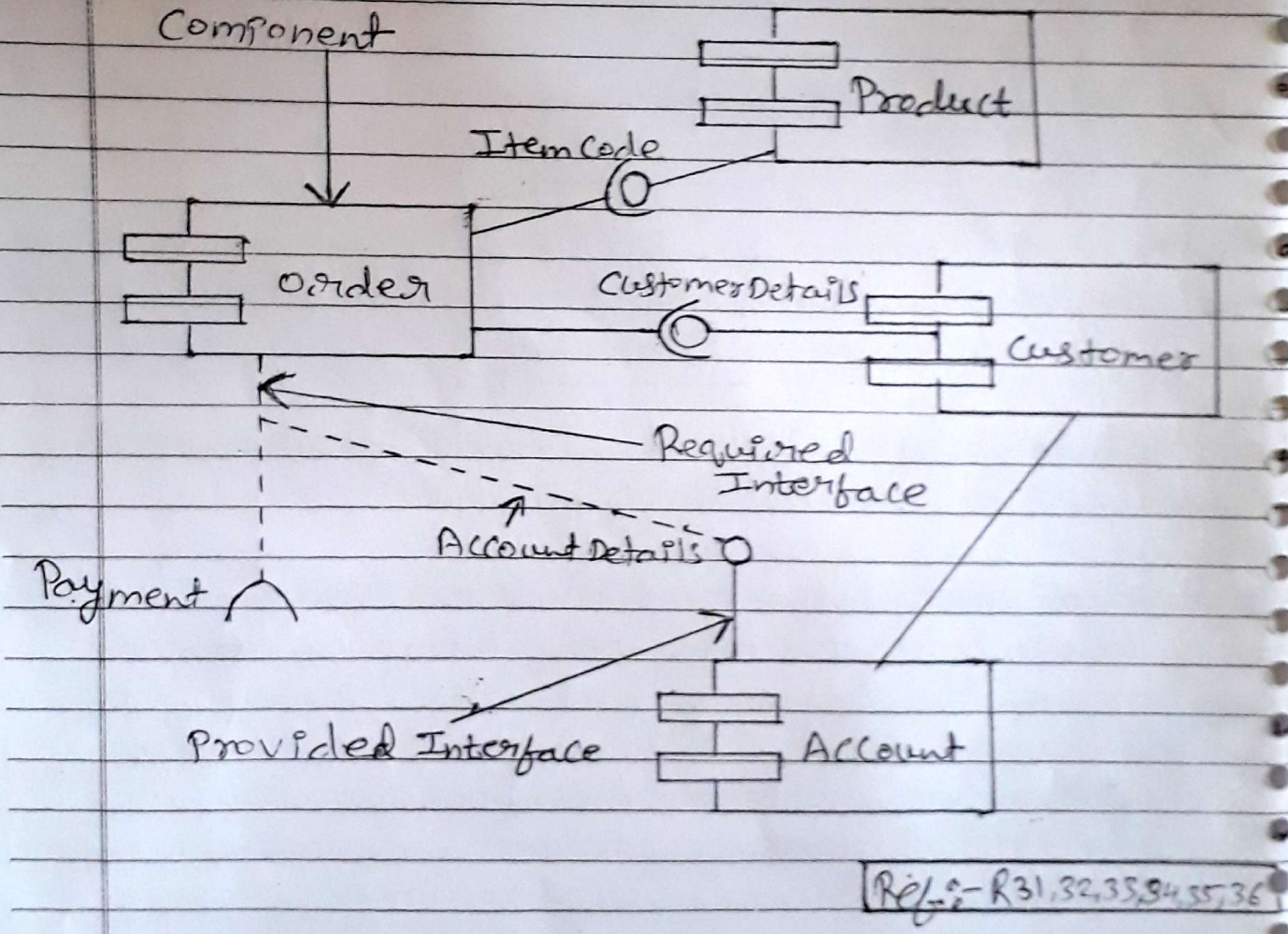


→ **Dependencies:-** Draw dependencies among Components using dashed arrows.



Dependency.

ex:-



Ref:- R31, 32, 33, 34, 35, 36

5.24 Difference b/w OOA & OOD :-

OOA :- Object Oriented Analysis :-

→ OOA looks at the Problem domain, with the aim of producing a conceptual model of the information that exists in the area being analyzed.

→ The Purpose of OOA is to develop Solution models that Satisfy the customer requirement.

→ It represent classes, Objects & interaction b/w objects.

→ The result of OOA is a description of What the System is functionally required to do, in the form of a conceptual model.

→ That will typically be presented by UML diagrams (represent analysis details).

→ Use case diagram & class diagram are used to model the OOA using UML.

→ The Purpose of OOA is to develop a model that describes computer s/w at it works to satisfy a set of customer defined requirements.

Name of Lecturer : Abhishek Jain

→ Object Oriented Design - (OOD)

→ The OOD Converts the OOA model into a design or logical solution model.

→ This serves an outline for S/w construction.

→ OOD Supports following OO concepts such as Abstraction, Information Hiding, Functional Independence, Modularity.

→ Design is the initial step in moving towards from the Problem domain. to solution domain.

→ A detailed design include specification of all the classes with its attributes, detailed interface.

→ The purpose of design is to specify a working solution that can be easily translated in to a Programming language code.

Ref. :- P37, P38