UML & UML DIAGRAMS [INTRODUCTION]

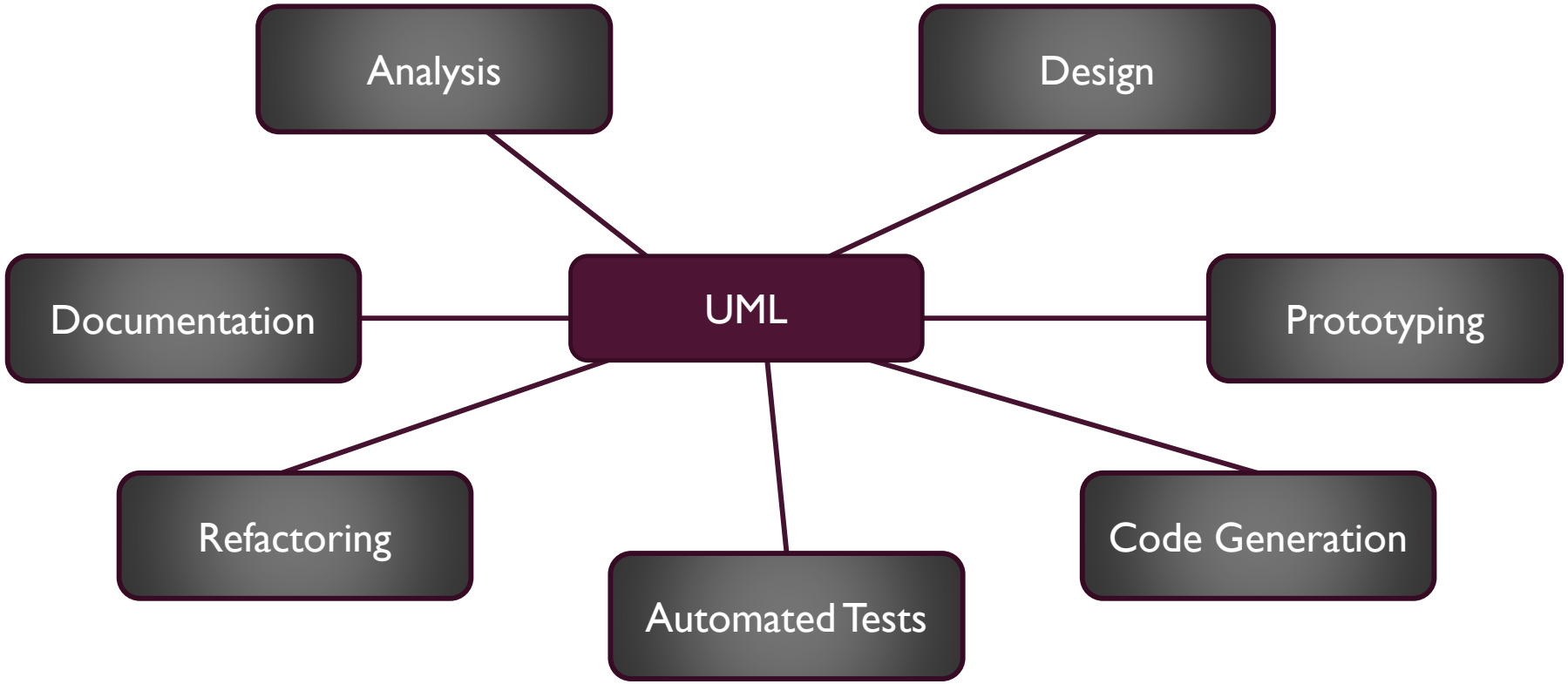
Ashita Agrawal

TABLE OF CONTENTS

INTRODUCTION TO UML DIAGRAM.

- UML stands for **U**nified **M**odelling **L**anguage.
- UML is a standard language for specifying, visualizing, constructing, and documenting a system in which software represents the most significant part.
- UML is different from the other common programming languages like C++, Java, COBOL etc.
- UML is a pictorial language used to make software blue prints.
- UML can serve as a central notation for software development process. Using UML helps project teams communicate, explore potential designs, and validate the architectural designs of software.
- UML diagrams are made using notation of things and relationships.

INTRODUCTION TO UML DIAGRAM.



A CONCEPTUAL MODEL OF UML

Conceptual model: A conceptual model can be defined as a model which is made of concepts and their relationships.

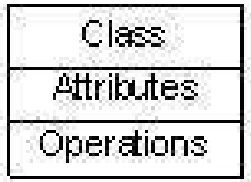
- As UML describes the real time systems it is very important to make a conceptual model and then proceed gradually. Conceptual model of UML can be better understood by learning the following three major elements :
  - *UML building blocks*
  - *Rules to connect the building blocks*
  - *Common mechanisms of UML*

BUILDING BLOCKS OF UML

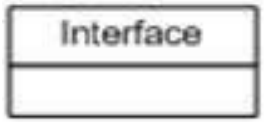
- The building blocks of UML can be defined as:
  - Things
  - Relationships
  - Diagrams
- Things: Things are the most important building blocks of UML. Things can be:
  - Structural
  - Behavioral
  - Grouping
  - Annotational

STRUCTURAL THINGS

- The **Structural things** define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.
- **Class:** Class represents set of objects having similar responsibilities.

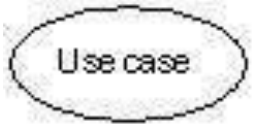


- **Interface:** Interface defines a set of operations which specify the responsibility of a class.

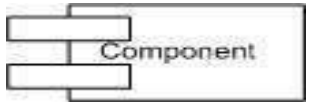


- **Collaboration:** Collaboration defines interaction between element

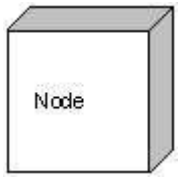
- **Use case:** Use case represents a set of actions performed by a system for a specific goal.



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

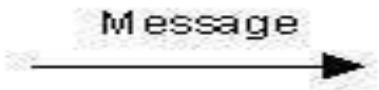


- **Node:** A node can be defined as a physical element that exists at run time.



BEHAVIORAL THING

- A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:
  - Interaction: Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.



  - State machine: State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change.

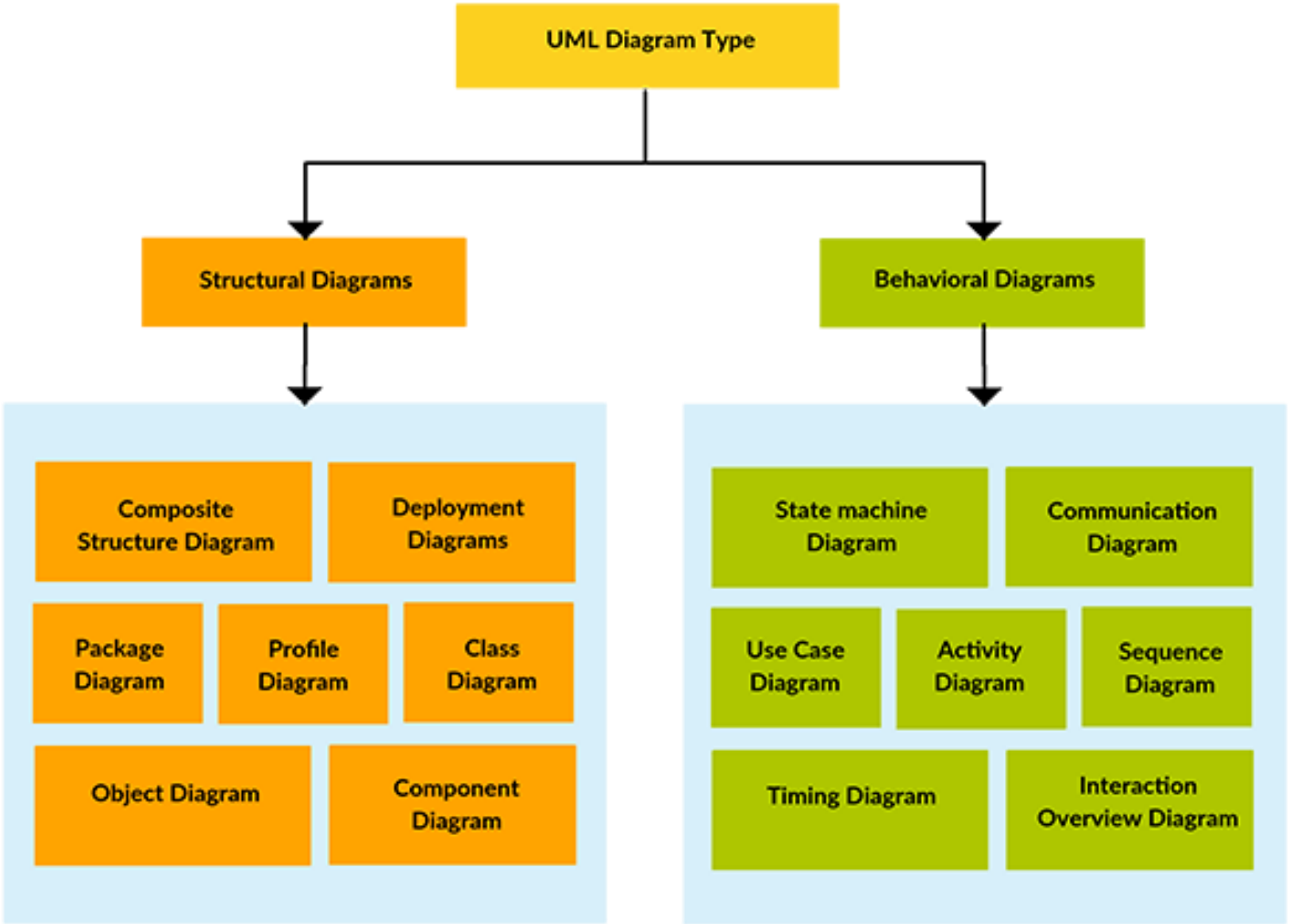
RELATIONSHIP

- Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application.
- There are four kinds of relationships available.
  - Dependency: Dependency is a relationship between two things in which change in one element also affects the other one.
  - Association: Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.
  - Generalization: Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.
  - Realization: Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them. This relationship exists in case of interfaces.

UML DIAGRAMS

- There are 9 diagrams in UML that can be used to model a system at different points of time in software life cycle of a system.
- They are:

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Activity diagram
7. State diagram
8. Deployment diagram
9. Component diagram

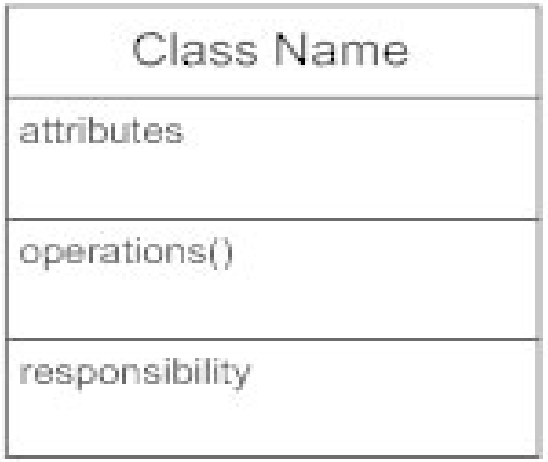

UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagrams
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
State machine Diagram
Communication Diagram
Use Case Diagram
Activity Diagram
Sequence Diagram
Timing Diagram
Interaction Overview Diagram

CLASS DIAGRAM

- A Class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.
- A class has three parts; name at the top, attributes in the middle and operations/methods at the bottom.
- The functionality provided by the class are termed "methods" of the class.
- Attribute uniquely identify the class.
- The class diagram is a static diagram. It represents the static view of an application.

Purpose of the class diagram can be summarized as:

- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.



Class

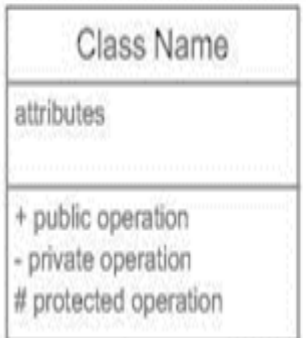
HOW TO DRAW CLASS DIAGRAM?

- Class diagrams have lot of properties to consider while drawing but here the diagram will be considered from a top level view.

The following points should be remembered while drawing a class diagram:

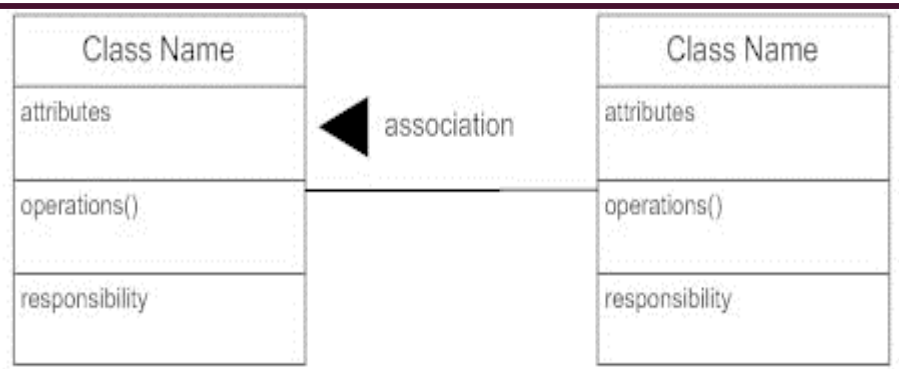
- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified.
- For each class minimum number of properties should be specified. Because unnecessary properties will make the diagram complicated.
- Use notes when ever required to describe some aspect of the diagram. Because at the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

NOTATIONS USED IN CLASS DIAGRAM

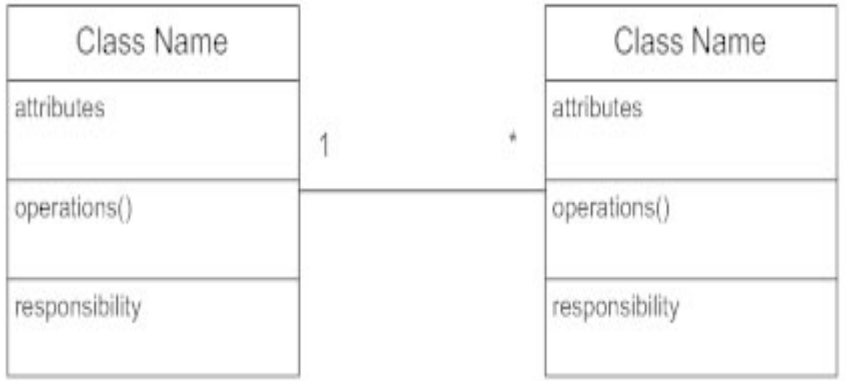


Visibility

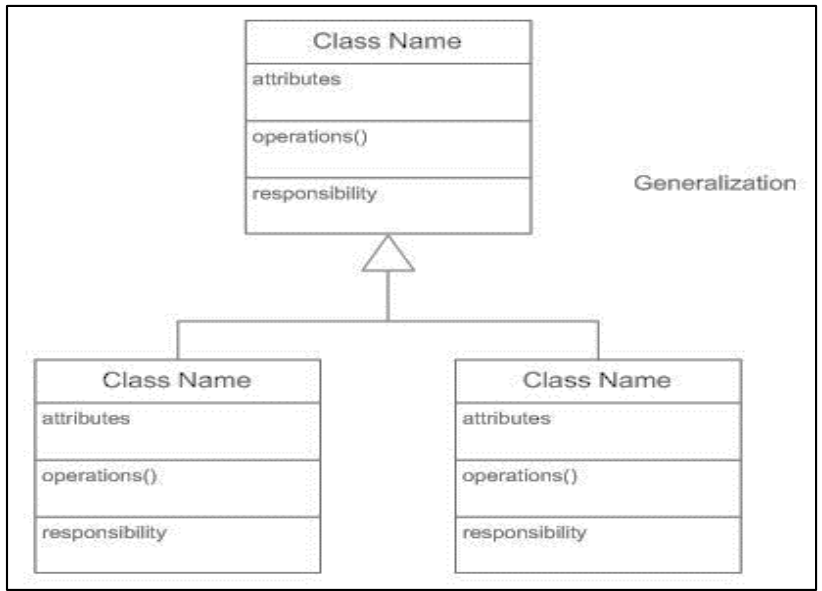
Marker	Visibility
+	public
-	private
#	protected
~	package

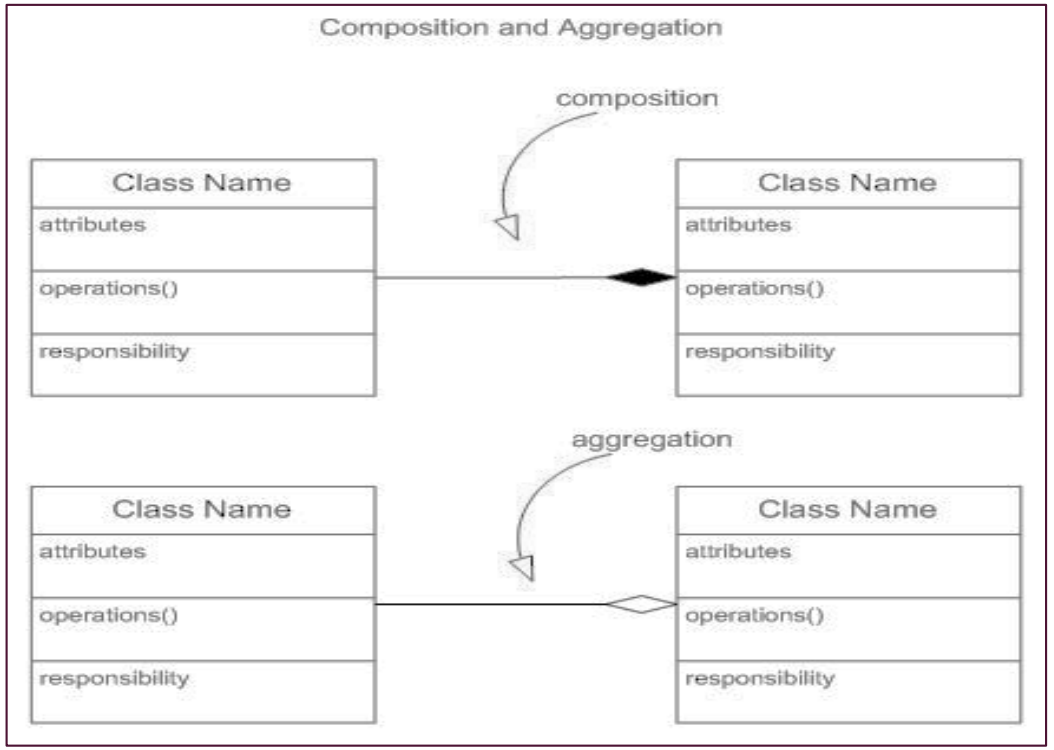


Multiplicity

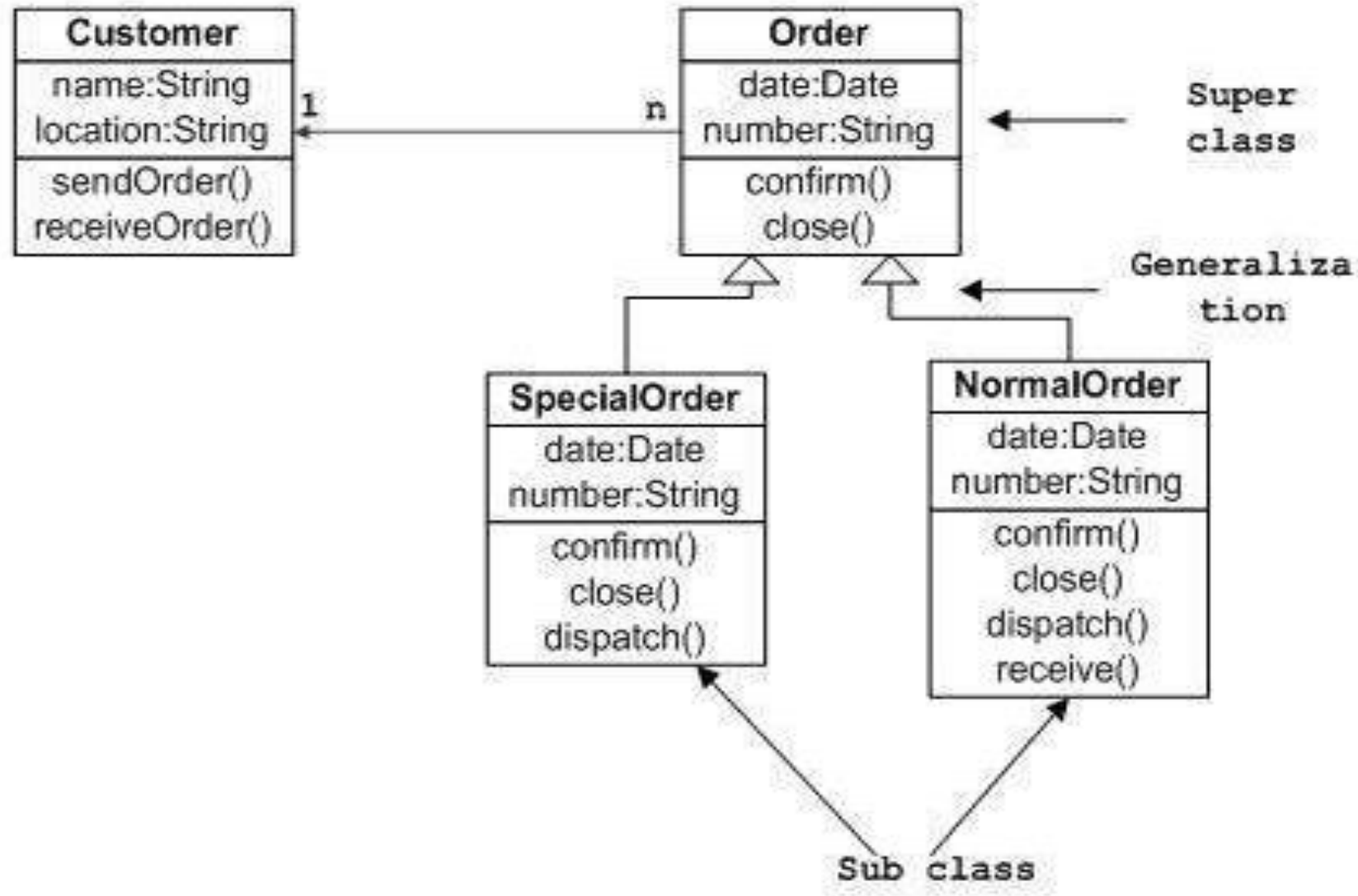


Indicator	Meaning
0..1	Zero or one
1	One only
0..*	0 or more
1..* *	1 or more
n	Only n (where n > 1)
0..n	Zero to n (where n >1)
1..n	One to n (where n > 1)

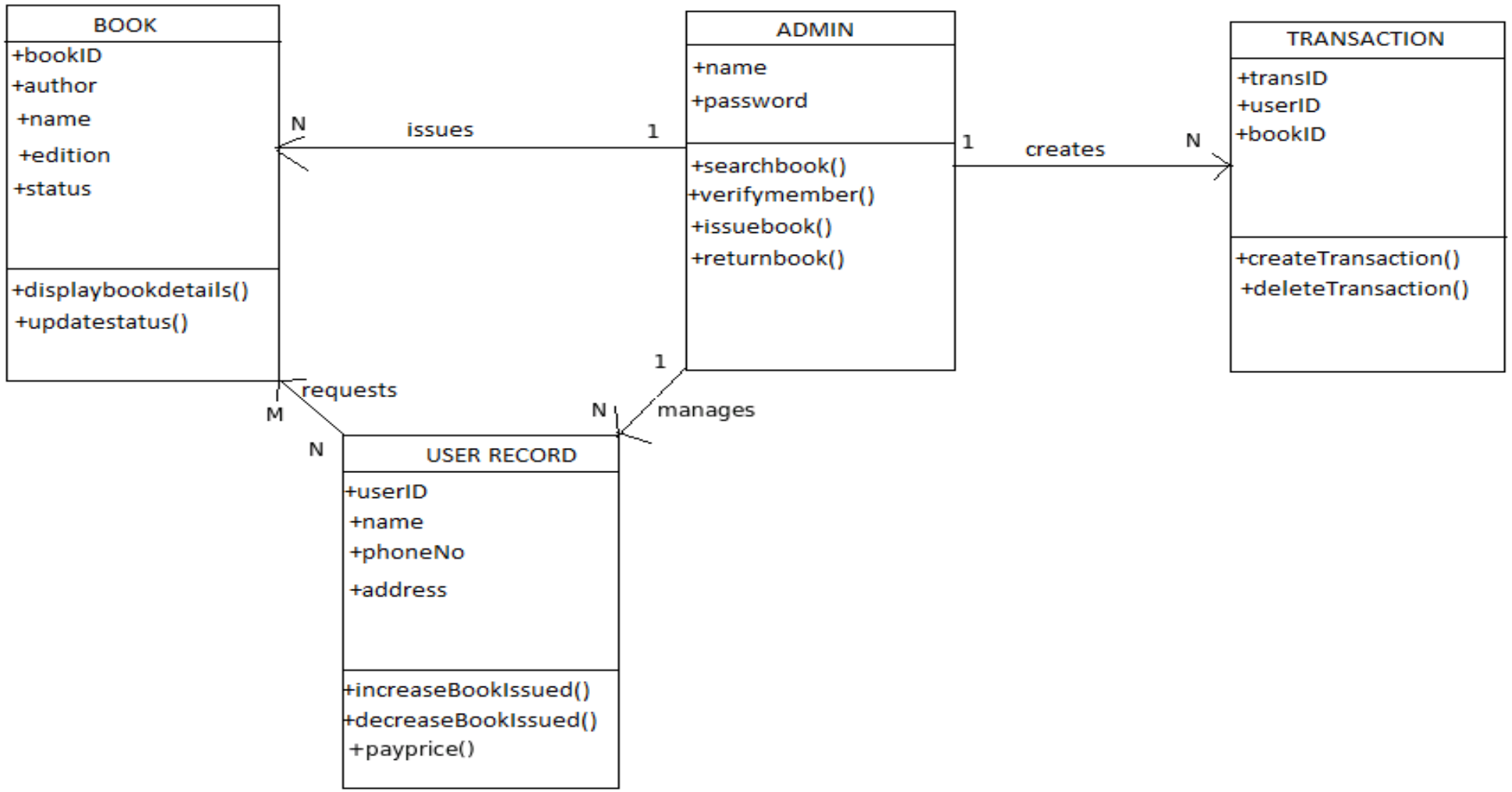




SAMPLE CLASS DIAGRAM OF ORDER MANAGEMENT SYSTEM



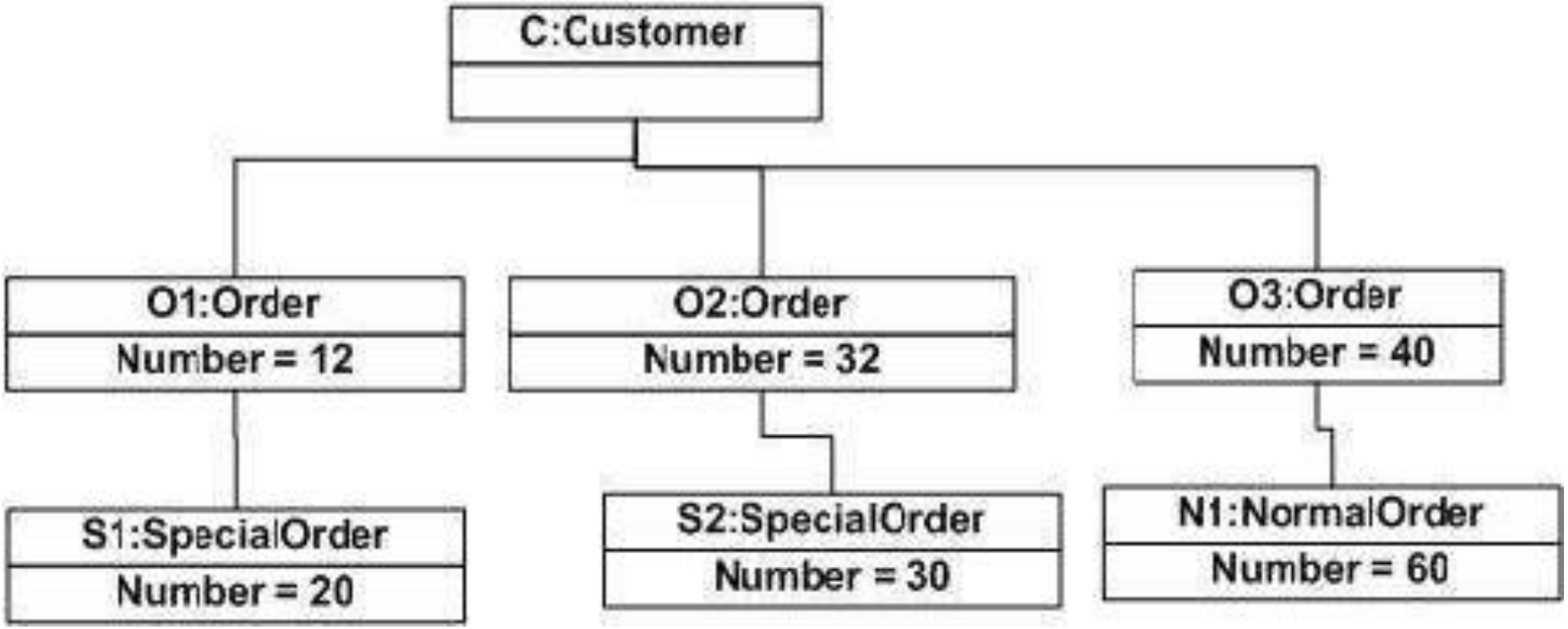
CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



OBJECT DIAGRAM

- Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.
- It is a special kind of class diagram. An object is an instance of a class.
- The object diagram captures the sate of different classes in system and their relationships or associations at a given point of time.
- In a brief, object diagrams are used for:
  - Making the prototype of a system.
  - Reverse engineering.
  - Modelling complex data structures.
  - Understanding the system from practical perspective.

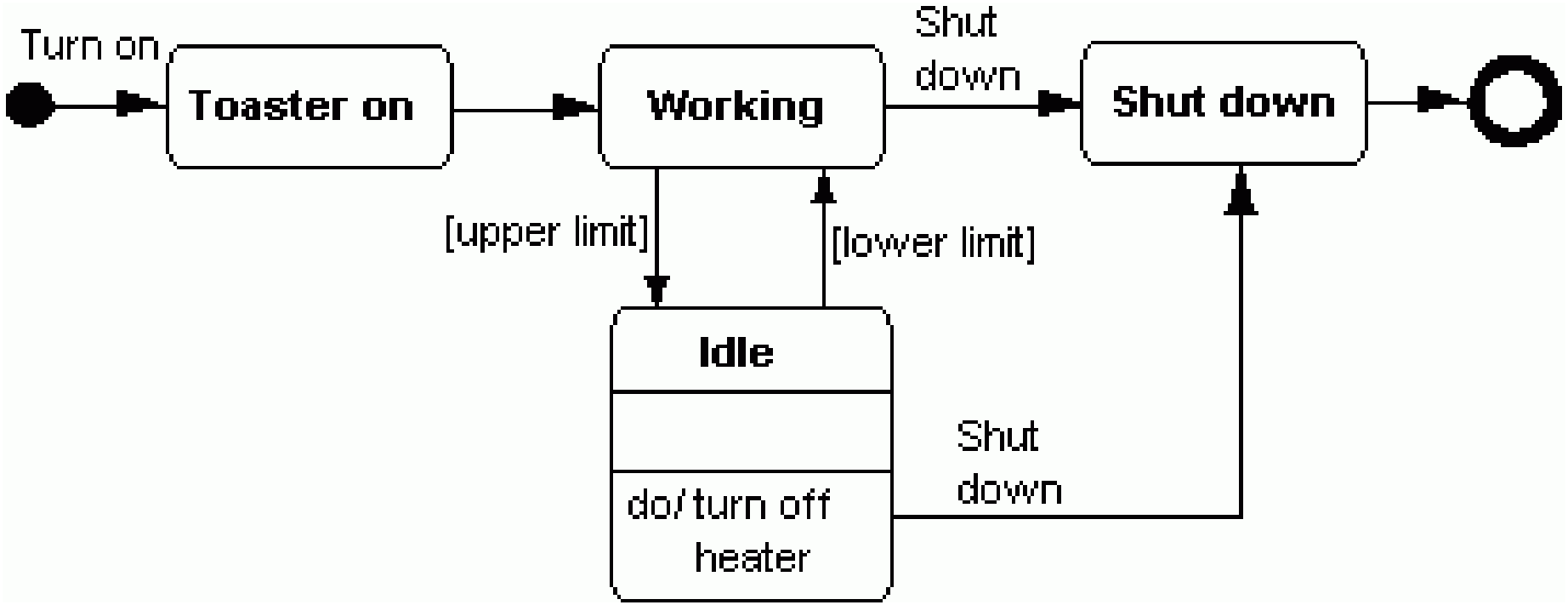
OBJECT DIAGRAM OF ORDER MANAGEMENT SYSTEM



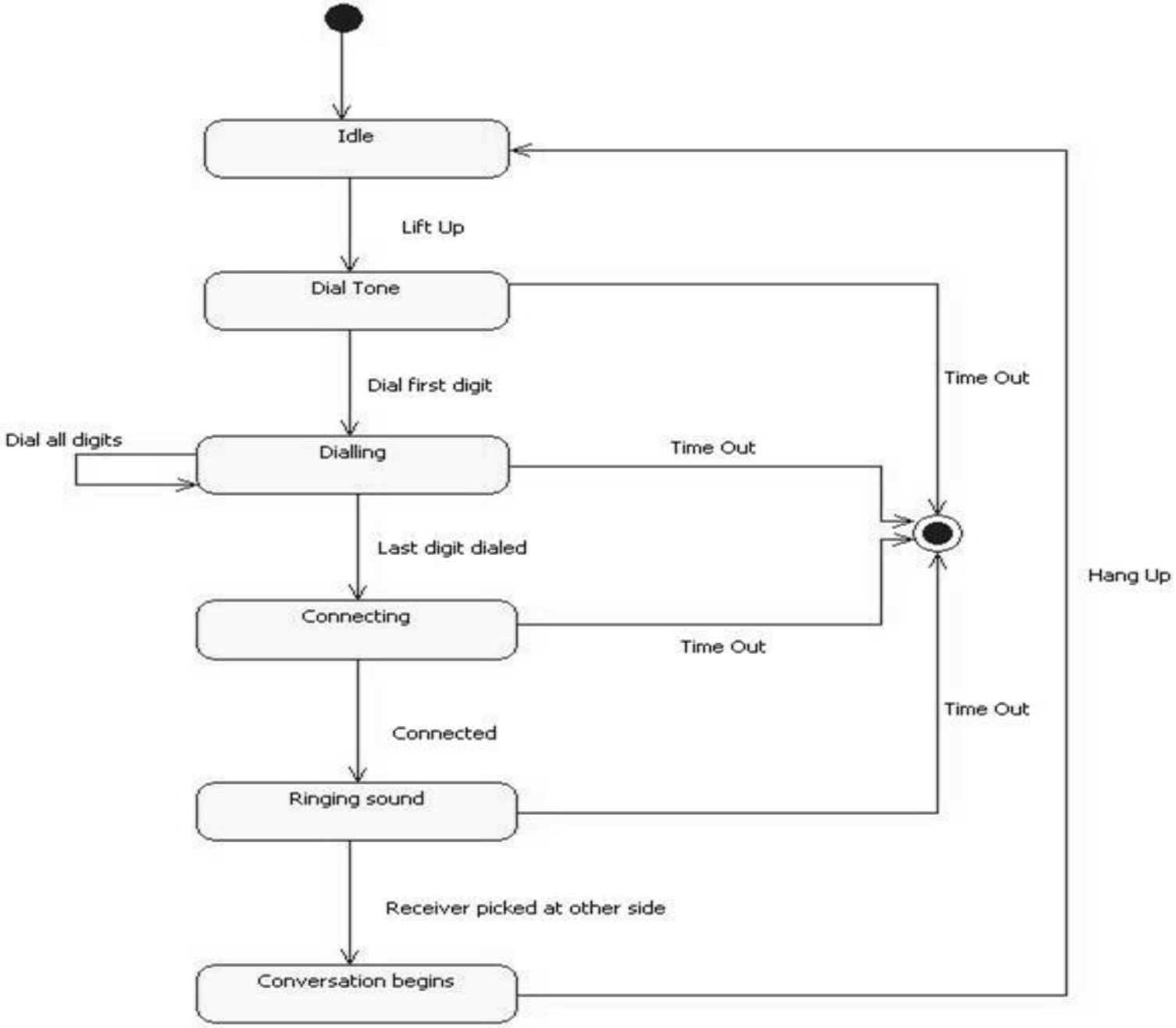
STATE DIAGRAM

- As the name suggests, it describes different states of a component in a system. The states are specific to a component/object of a system.
- Objects in the system change status in response to events. Used to model dynamic nature of a system.
- State diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered.
- Following are the main purposes of using State diagrams:
  - To model dynamic aspect of a system.
  - To model life time of a reactive system.
  - To describe different states of an object during its life time.
  - Define a state machine to model states of an object.
- Before drawing a State diagram we must have clarified the following points:
  - Identify important objects to be analyzed.
  - Identify the states & Identify the events.

STATE DIAGRAM FOR TOASTER



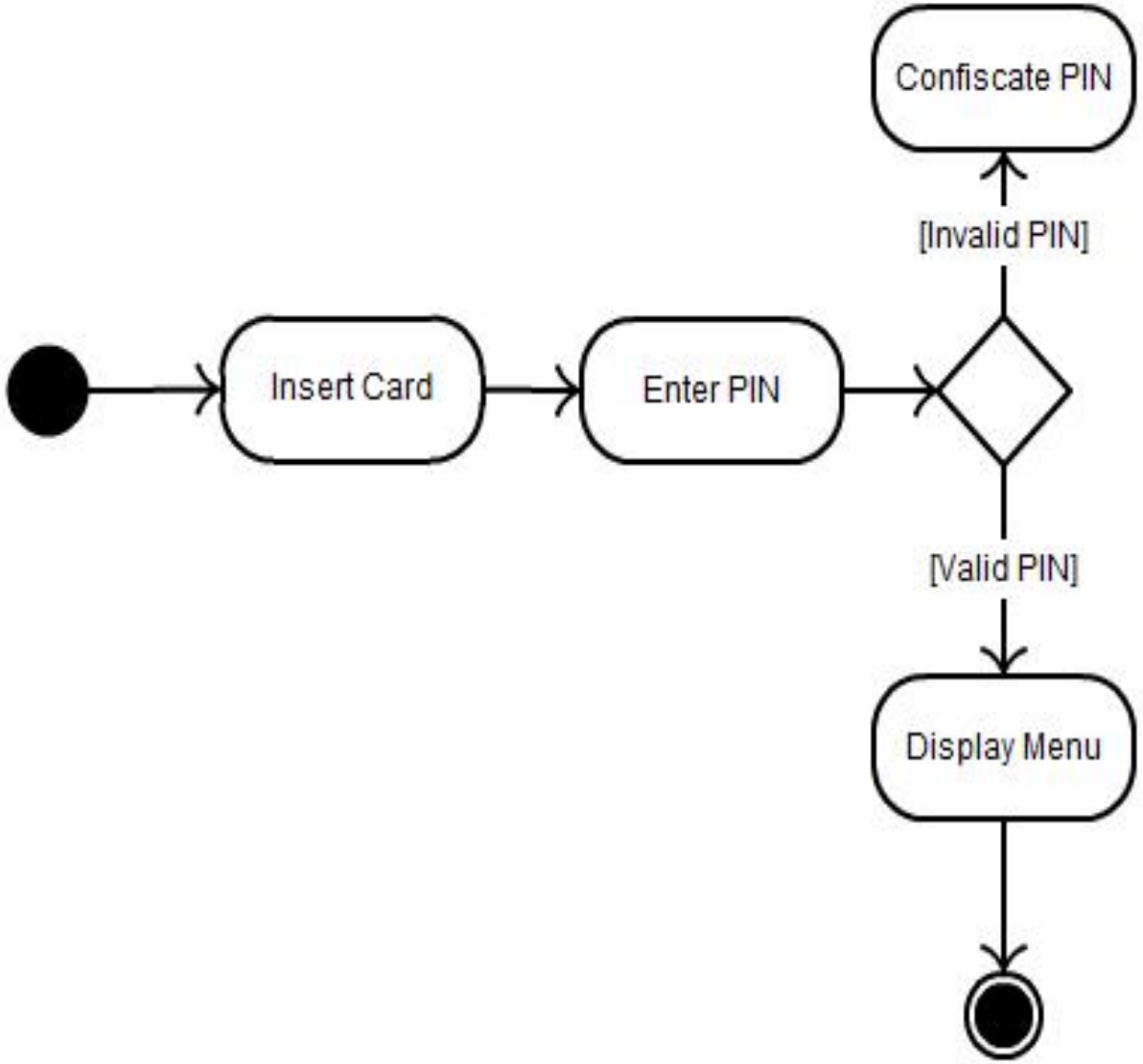
STATE DIAGRAM FOR PHONE



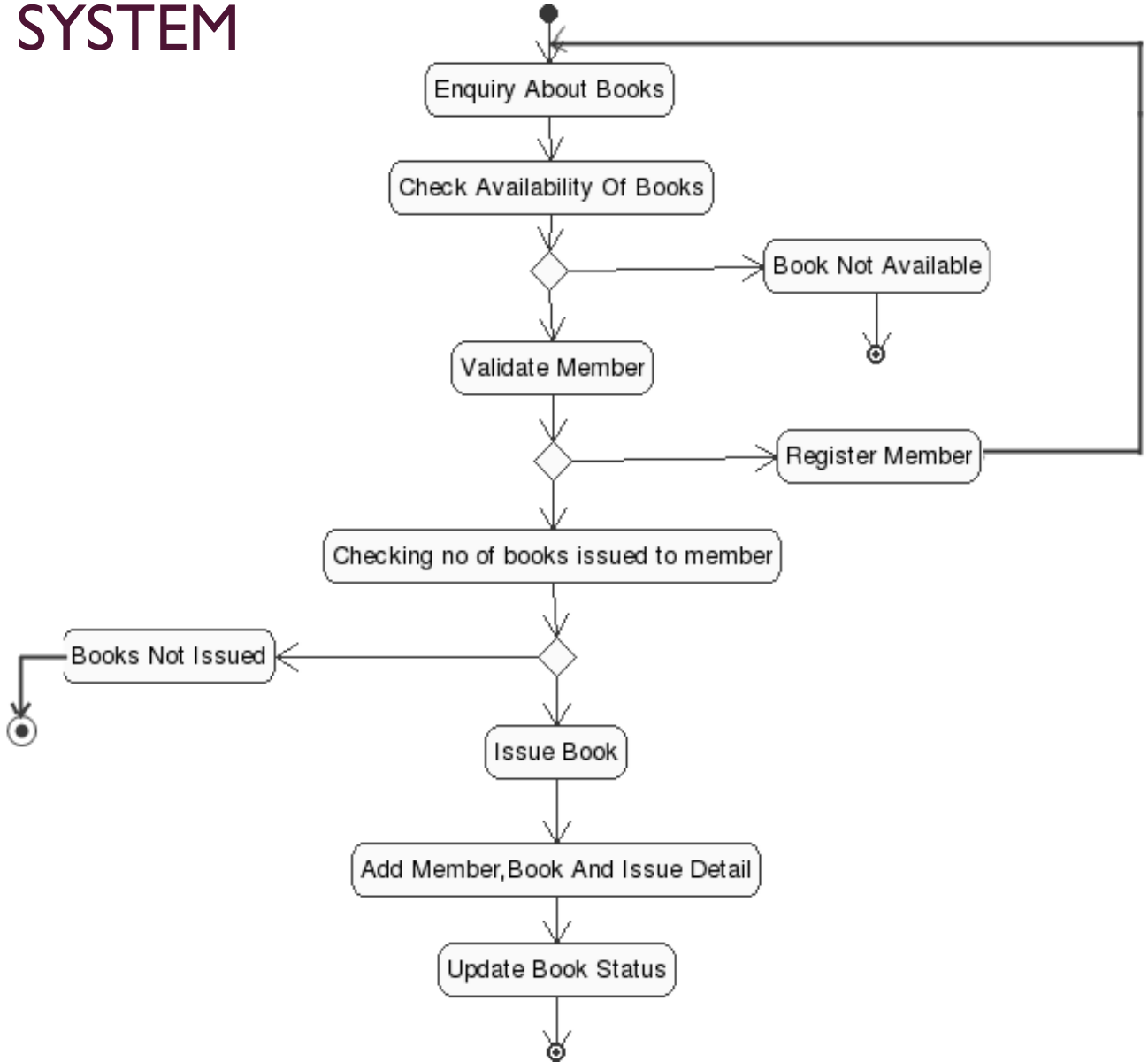
ACTIVITY DIAGRAM

- Activity diagram is another important diagram in UML to describe dynamic aspects of the system.
- Activity diagram is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system.
- So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements like fork, join etc.
- Purposes can be described as:
  - Draw the activity flow of a system.
  - Describe the sequence from one activity to another.
  - Describe the parallel, branched and concurrent flow of the system.
- Before drawing an activity diagram we should identify the following elements:
  - Activities
  - Association
  - Conditions
  - Constraints

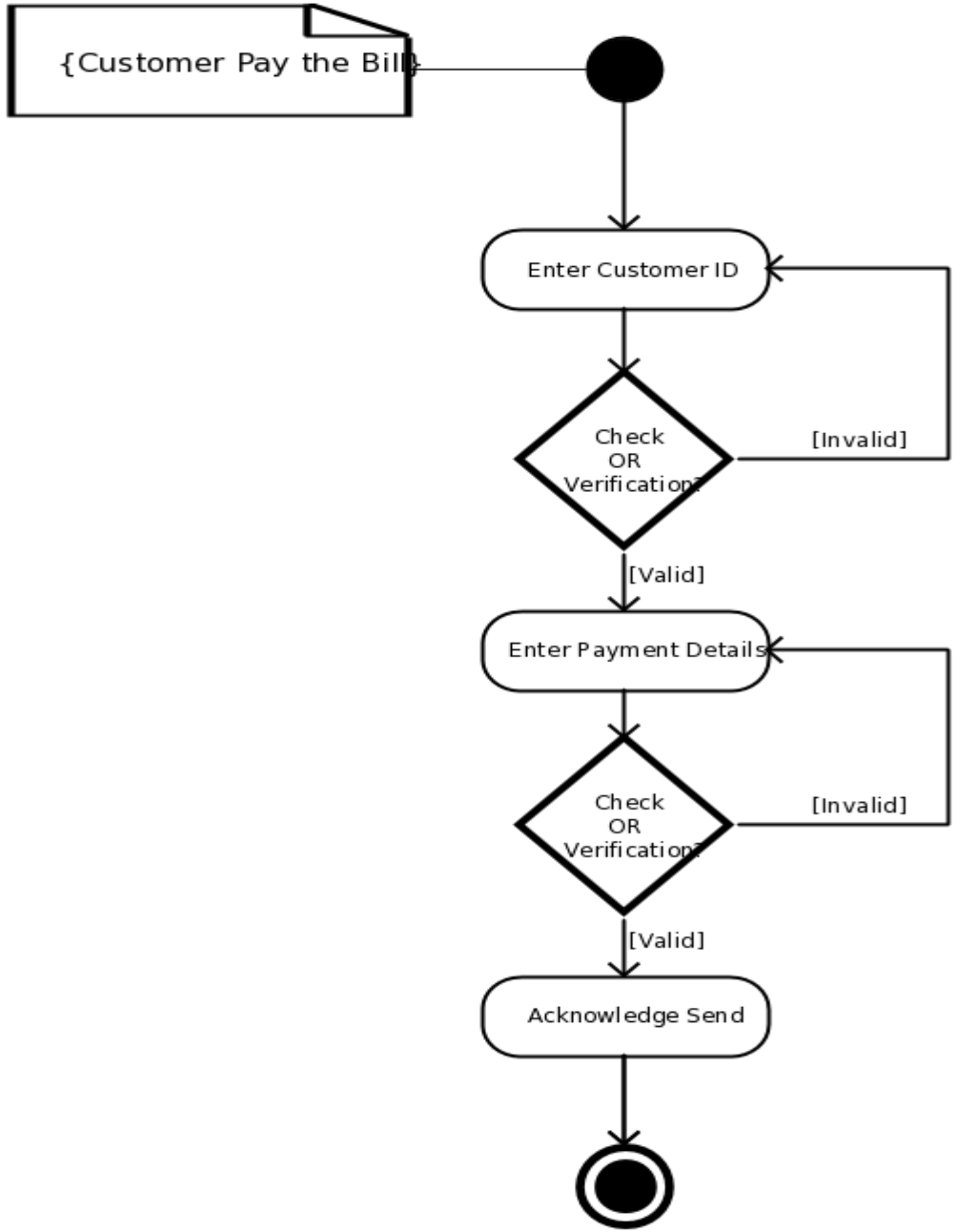
ACTIVITY DIAGRAM FOR ATM



ACTIVITY DIAGRAM FOR (ISSUE BOOK) ONLINE LIBRARY MANAGEMENT SYSTEM



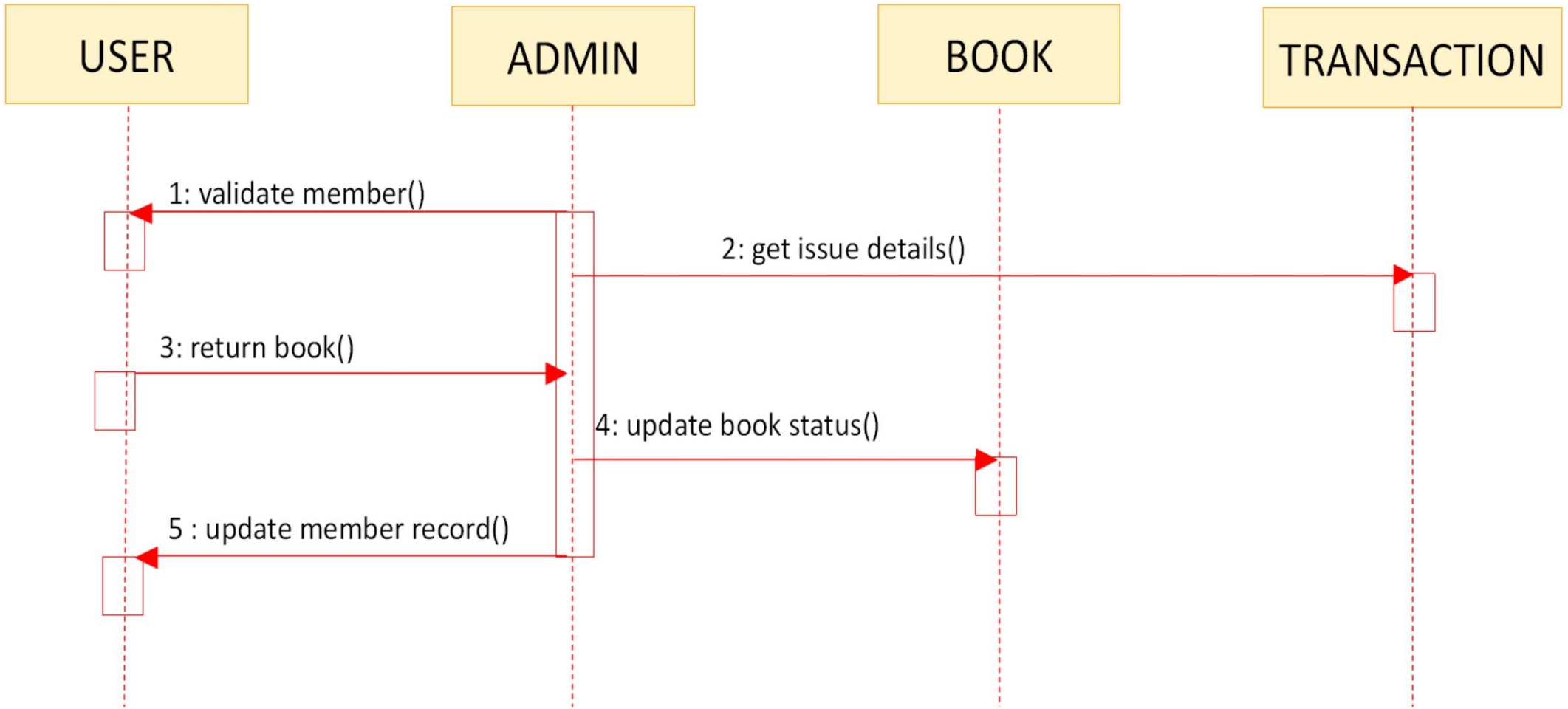
ACTIVITY DIAGRAM FOR ONLINE ELECTRICITY BILL PAYMENT



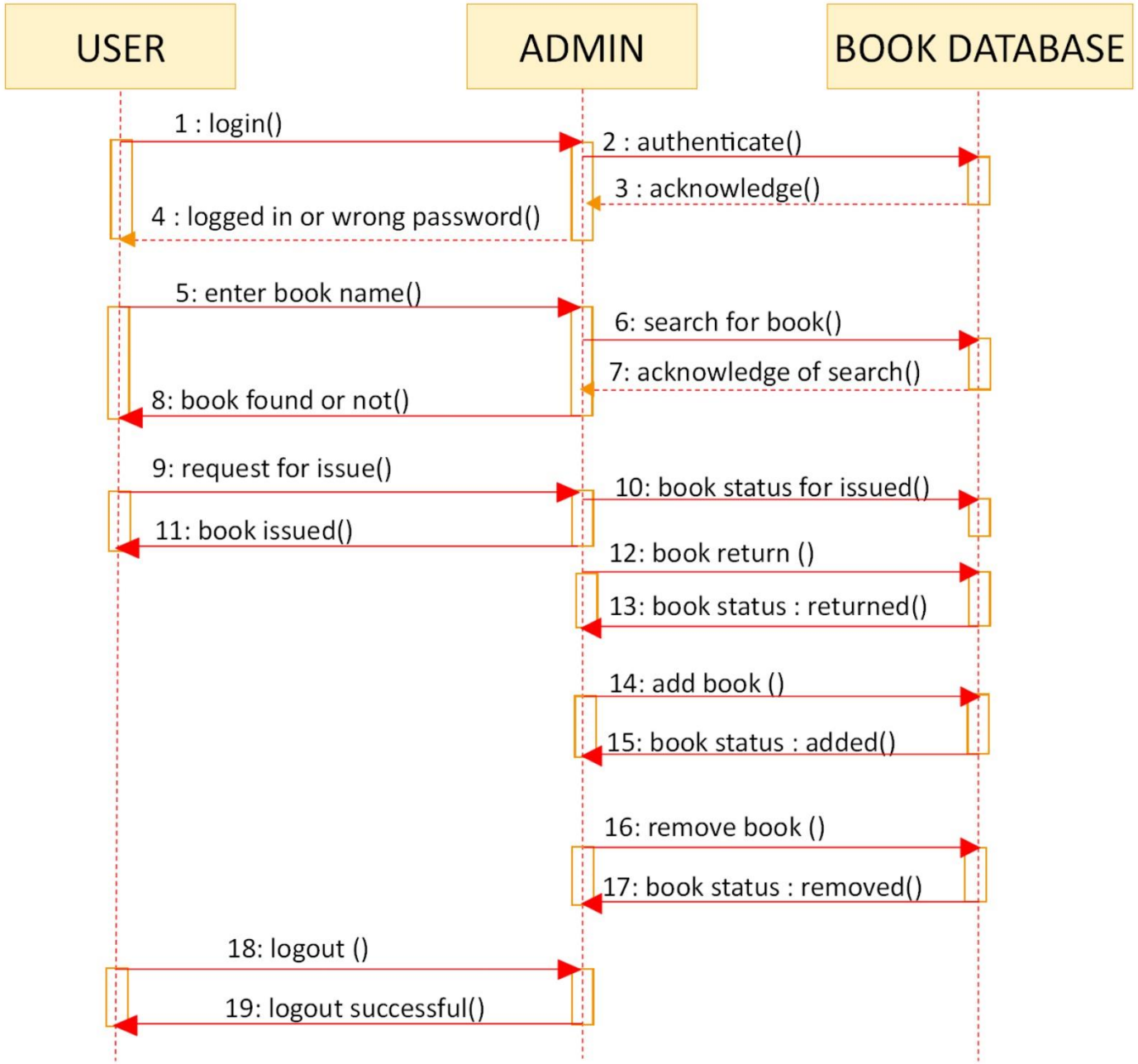
SEQUENCE DIAGRAM

- Sequence diagram is used to describe some type of interactions among the different elements/objects in the system. Important aspect of this is that it is time-ordered.
- Purposes of sequence diagram can be describes as:
  - To capture dynamic behaviour of a system.
  - To describe the message flow in the system.
  - To describe structural organization of the objects.
  - To describe interaction among objects.
- The following things are to identified clearly before drawing the interaction diagram:
  - Objects taking part in the interaction.
  - Message flows among the objects.
  - The sequence in which the messages are flowing.
  - Object organization

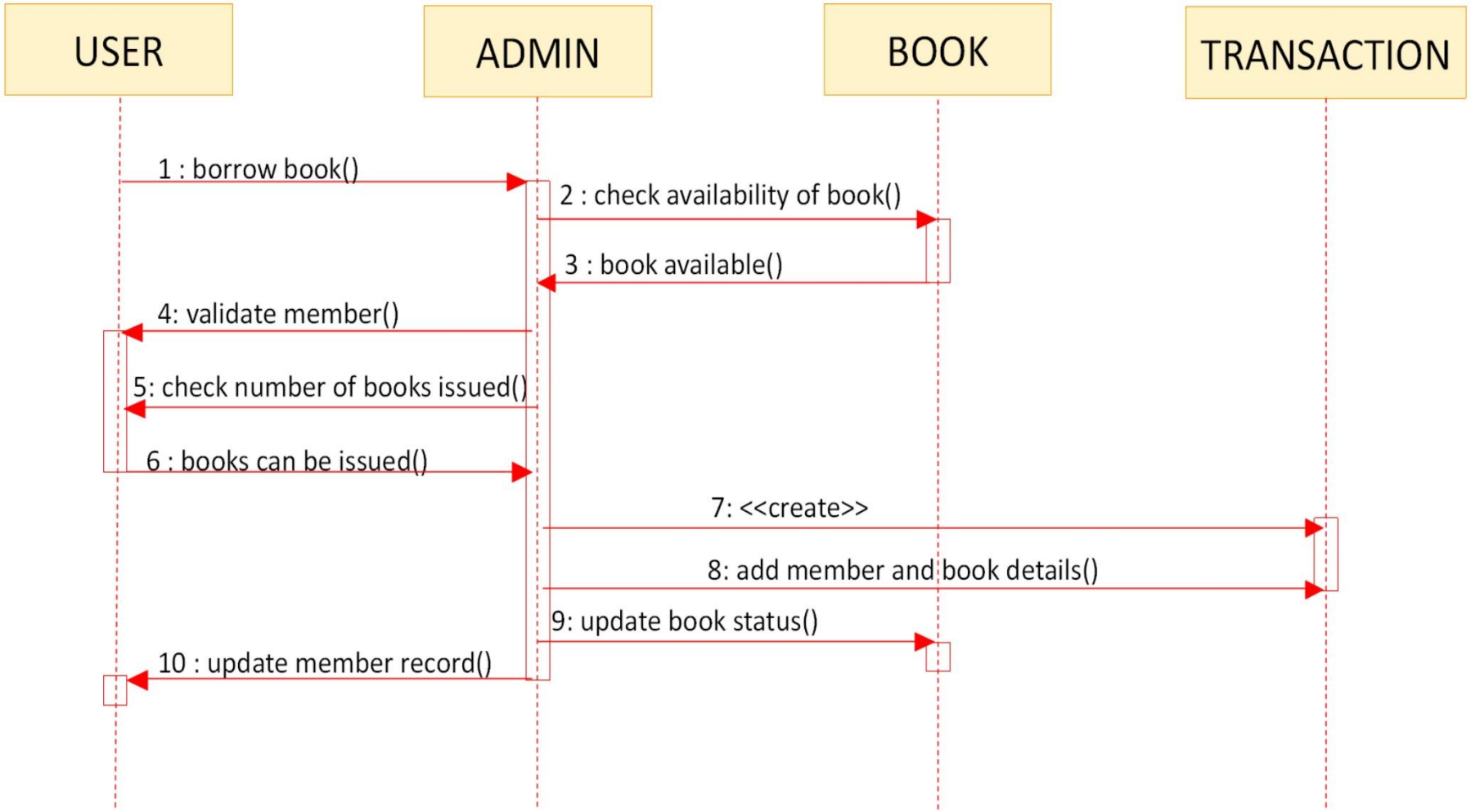
Uml sequence diagram for return book



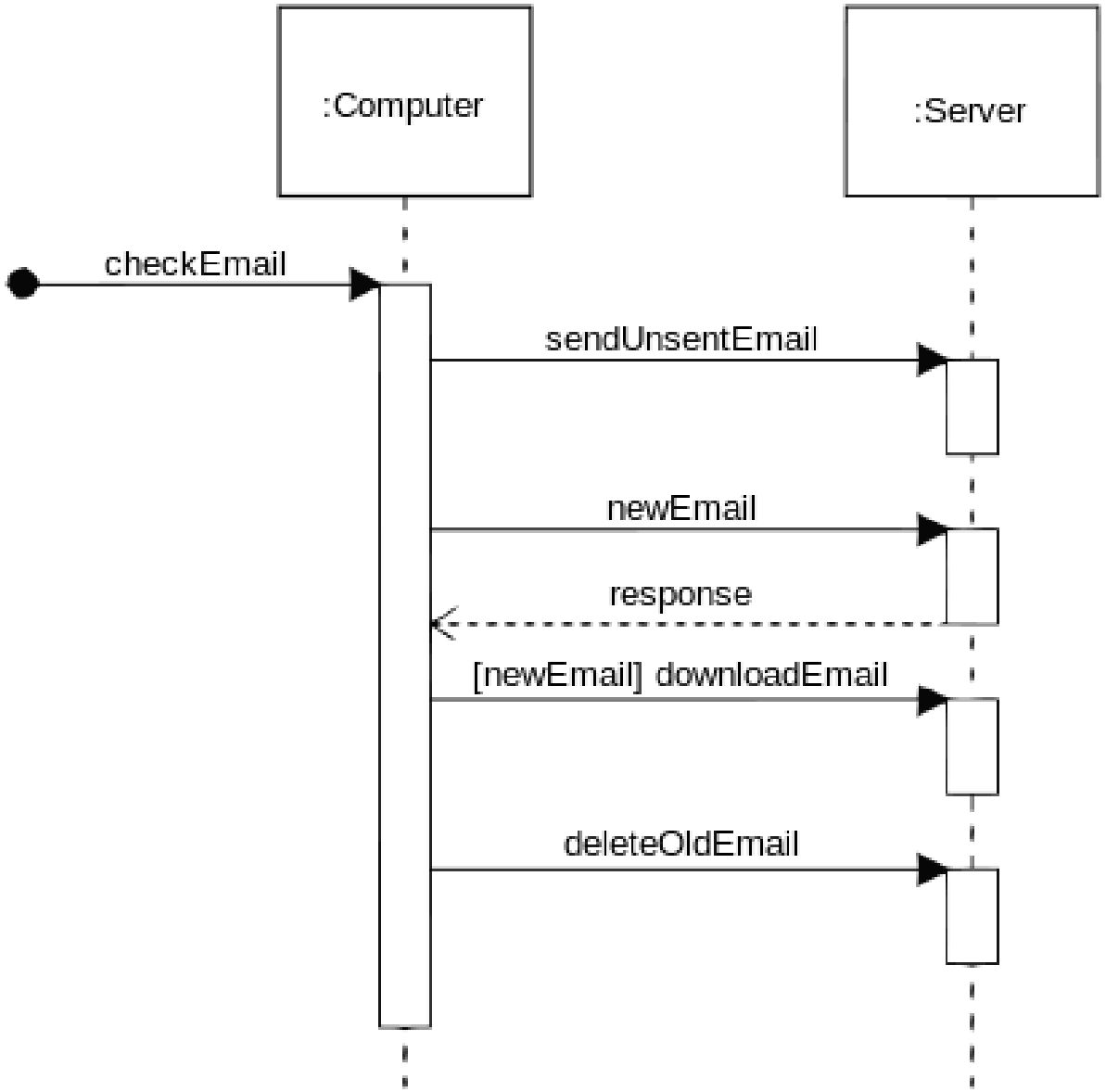
Uml sequence diagram for entire database



ACTIVITY DIAGRAM FOR ISSUE BOOK



SEQUENCE DIAGRAM OF E-MAIL MESSAGE SEQUENCE



COMPONENT DIAGRAM

- Component diagrams are different in terms of nature and behaviour. 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. So component diagrams are used to visualize the organization and relationships among components in a system.
- Purpose of the component diagram can be summarized as:

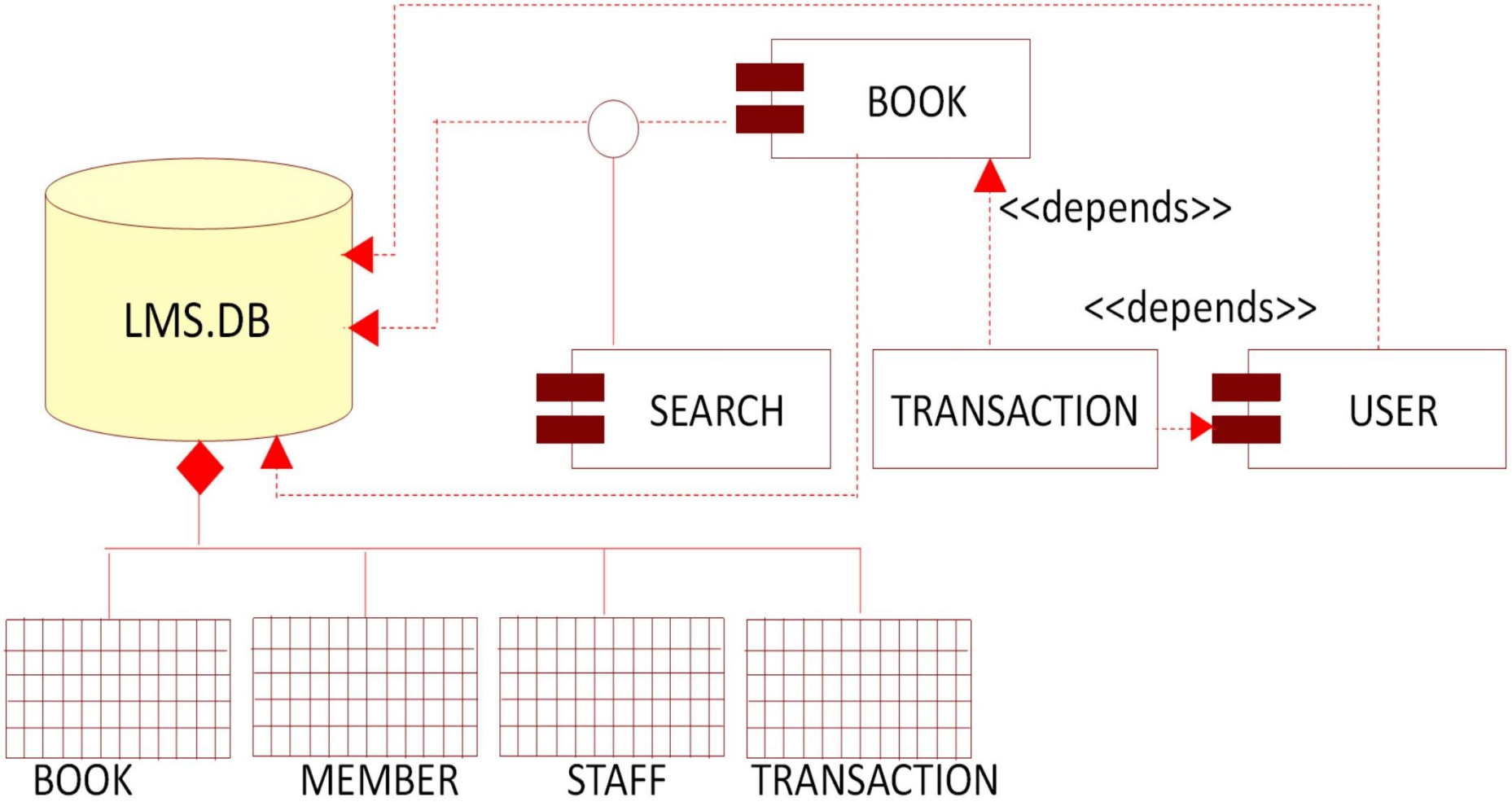
  1.)Visualize the components of a system. 2.)Construct executables by using forward and reverse engineering.

  3.)Describe the organization and relationships of the components.
- A single component diagram cannot represent the entire system but a collection of diagrams are used to represent the whole.
- Before drawing a component diagram the following artifacts are to be identified clearly:

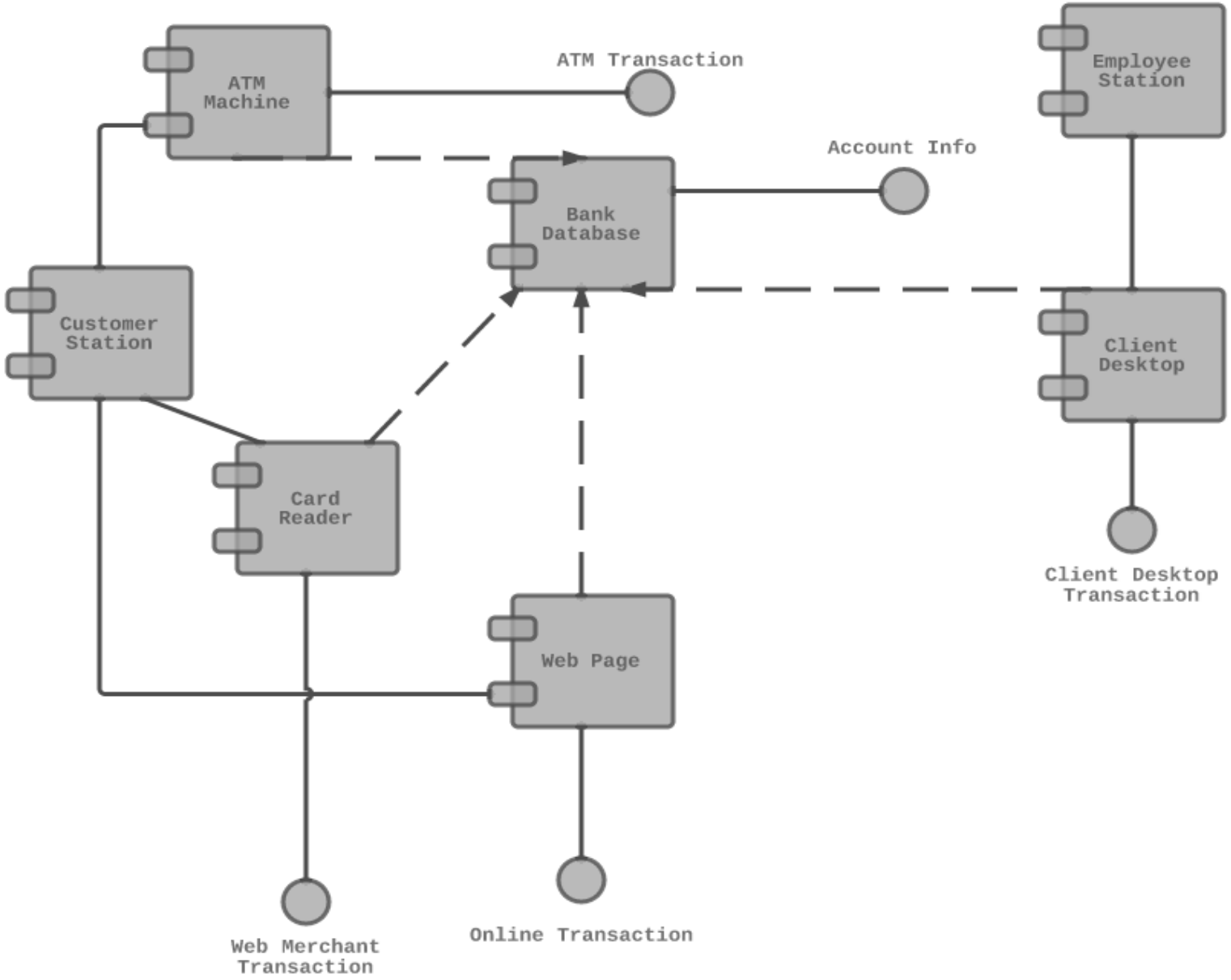
  1.) Files used in the system. 2.)Libraries and other artifacts relevant to the application.

  3.)Relationships among the artifacts.

COMPONENT DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



COMPONENT DIAGRAM FOR ATM



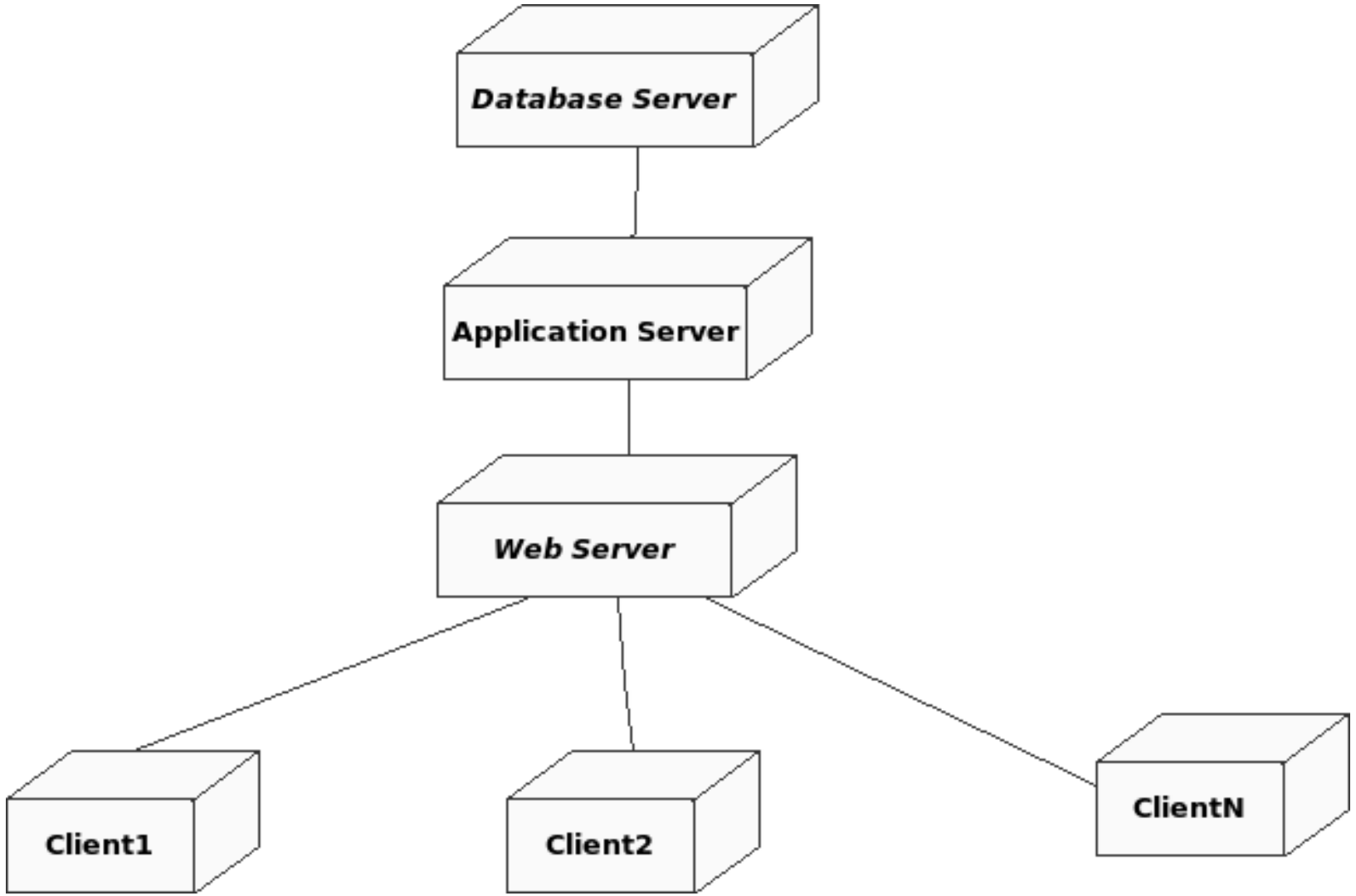
DEPLOYMENT DIAGRAM

- Deployment diagrams are used to visualize the topology of the physical components of a system where the software components are deployed. So deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.
- The purpose of deployment diagrams can be described as:
  - Visualize hardware topology of a system.
  - Describe the hardware components used to deploy software components.
  - Describe runtime processing nodes.
- Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important because it controls the following parameters:

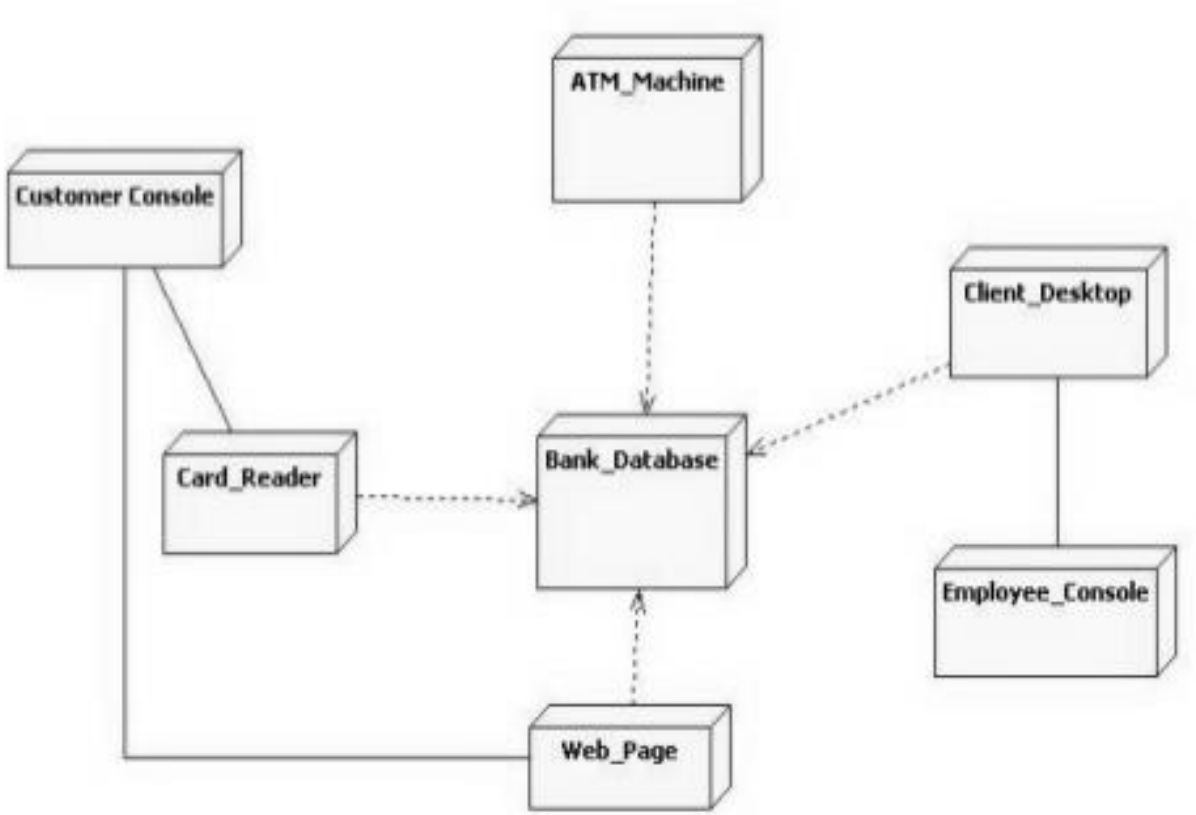
  >Performance >Scalability >Maintainability >Portability
- So before drawing a deployment diagram the following artifacts should be identified:

  >Nodes >Relationships among nodes

DEPLOYMENT DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM



Deployment for ATM Machine



USE CASE DIAGRAM

- Use case diagrams are consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.
- the purposes of use case diagrams can be as follows:
  - Used to gather requirements of a system.
  - Used to get an outside view of a system.
  - Identify external and internal factors influencing the system.
  - Show the interacting among the requirements are actors.
- When we are planning to draw an use case diagram we should have the following items identified.
  - Functionalities to be represented as an use case
  - Actors
  - Relationships among the use cases and actors.

USE CASE DIAGRAM

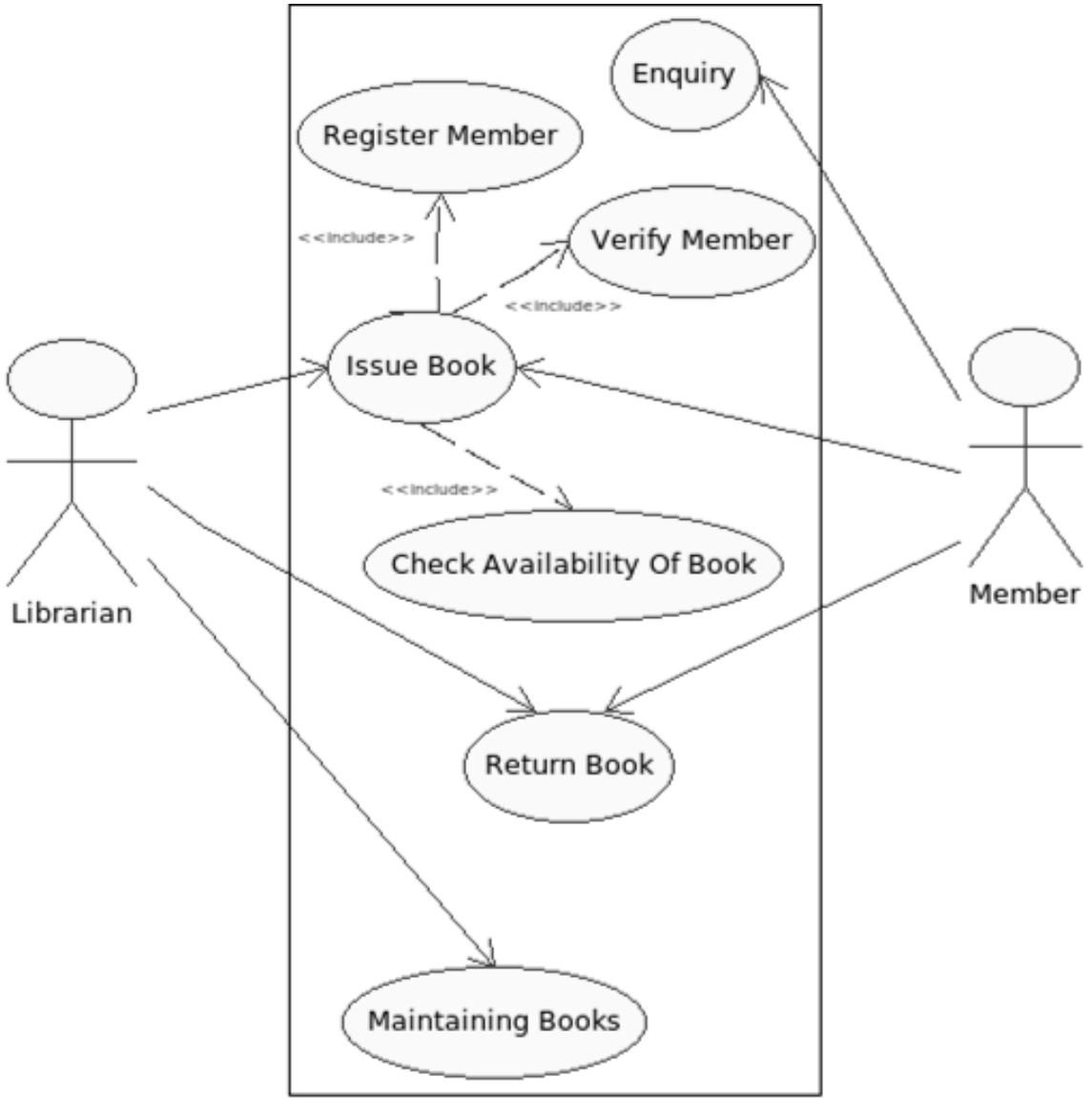
follow the following guidelines to draw an efficient use case diagram.

- The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.
- Use note when ever required to clarify some important points.

The following are the places where use case diagrams are used:

- Requirement analysis and high level design.
- Model the context of a system.
- Reverse engineering.
- Forward engineering

USE CASE DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM




Thank you!