

## K2 Level -Unit 2 Basic Structural modelling and behavioural modeling

<b>Question:1</b>	Draw Sequence diagram for issuing books from library.
<b>Solution: 1 (With step wise marking)</b>	<p>The diagram models various steps in book issuing process.</p> <pre> sequenceDiagram     participant Librarian     participant Books     participant MemberRecord     participant Transaction      Librarian-&gt;&gt;Books: check availability of book()     Books--&gt;&gt;Librarian: book available ()     Librarian-&gt;&gt;MemberRecord: validate member ()     Librarian-&gt;&gt;MemberRecord: check no of book issue()     MemberRecord--&gt;&gt;Librarian: book can be issued ()     Librarian-&gt;&gt;Transaction: create     Librarian-&gt;&gt;Transaction: add member and book details ()     Librarian-&gt;&gt;Books: update book status ()     Librarian-&gt;&gt;MemberRecord: update member record ()         </pre>

<b>Question:2</b>	Draw class diagram for ATM system and sequence diagram for withdrawing money from ATM.
<b>Solution: 2 (With step wise marking)</b>	<p><b>Class diagram-</b></p> <p>The top rectangle has the class's name; the middle rectangle contains the class's properties; and the bottom rectangle contains the class's methods, commonly known as operations.</p>

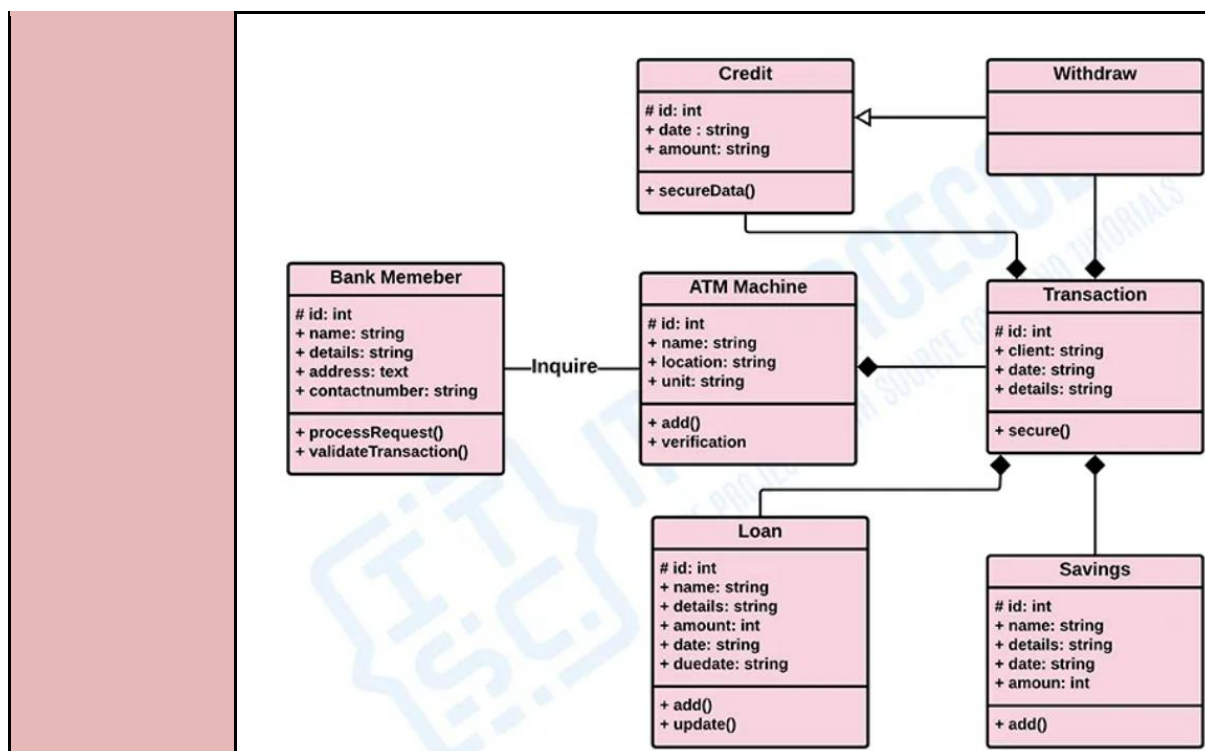


Table 1

<b>Question:3</b>	Describe importance of an Use-Case diagram. Draw a use case diagram for student admission system.
<b>Solution: 3 (With step wise marking)</b>	<p><b><u>Uses and Importance of use case diagrams-</u></b></p> <p>Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system.</p> <p>Use-case diagrams are helpful in the following situations:</p> <ul style="list-style-type: none"> <li>• Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.</li> <li>• While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.</li> <li>• During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.</li> <li>• During the testing phase, you can use use-case diagrams to identify tests for the system.</li> </ul> <p><b><u>Use case diagram for Admission system</u></b></p>

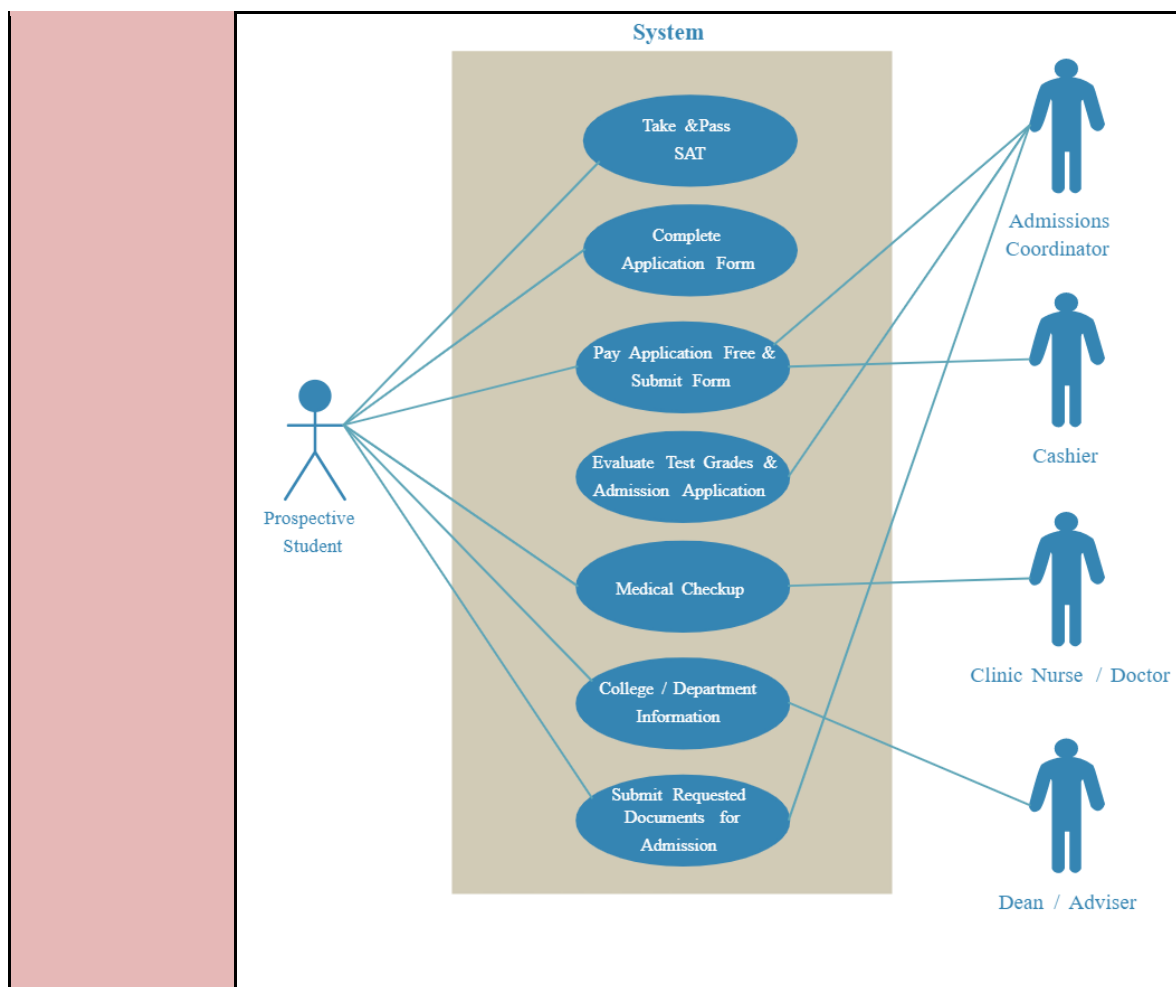


Table 2

<b>Question:4</b>	Differentiate between behaviour diagram and structure diagram.
<b>Solution: 4 (With step wise marking)</b>	<p><b>Structure Diagrams</b> – Used to model the static structure of a system, for example- class diagram, package diagram, object diagram, deployment diagram etc.</p> <p><b>Behavior diagram</b> – Used to model the dynamic change in the system over time. They are used to model and construct the functionality of a system. So, a behavior diagram simply guides us through the functionality of the system using Use case diagrams, Interaction diagrams, Activity diagrams and State diagrams.</p>

Table 3

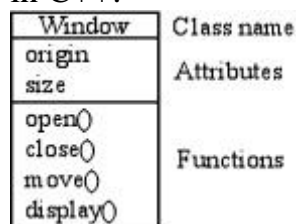
<b>Question:5</b>	Describe Structural and Behavioral things in Conceptual Model of UML.
<b>Solution: 5 (With step wise marking)</b>	<p>There are three building blocks in UML, things ,relationships and diagrams. The types of things are described as below:-</p> <ul style="list-style-type: none"> <li>Structural things -- The nouns of UML models. These represent elements that are conceptual or physical. There are seven kinds of structural things: Class, Interface,</li> </ul>

Collaboration, Use Case, Active Class, Component, and Node. (See the examples below.)

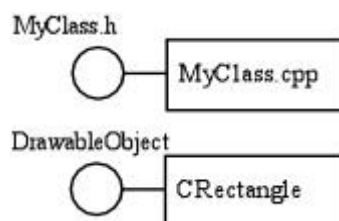
- Behavioral things -- Dynamic parts of UML models. The verbs which represent behavior over time and space. There are two kinds of behavioral things: Interaction, and State Machine. (See the examples below.)
- Grouping things -- Organizational parts of UML. These are boxes into which models can be decomposed. There is only one kind of grouping thing, the Package. (See the examples below.)
- Annotational things -- Explanatory parts of UML. Used to describe, illuminate, and remark any element of a model. There is only one kind of annotational thing, the note.

#### Seven types of structural things-

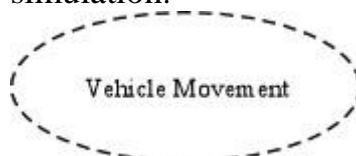
1. **Class** - An object with defined attributes and operations. A class in UML is very much like a class in C++.



2. **Interface** - A collection of functions that specify a service of a class or component, i.e. externally visible behavior of that class.

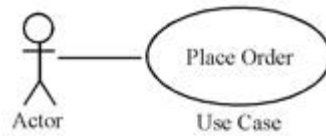


3. **Collaboration** - A larger pattern of behaviors and actions. Example: All classes and behaviors that create the modeling of a moving tank in a simulation.

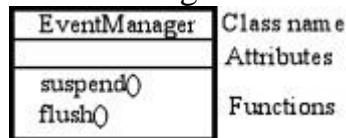


4. **Use Case** - A sequence of actions that a system performs that yields an observable result. Used to

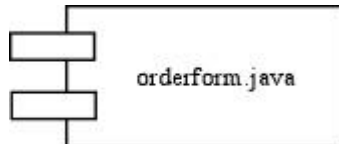
structure behavior in a model. Is realized by a collaboration.



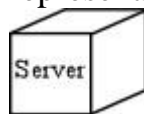
5. **Active Class** - Like a class but its represents behavior that runs concurrent with other behaviors, i.e. threading.



6. **Component** - A physical and replaceable part of a system that implements a number of interfaces.  
Example: a set of classes, interfaces, and collaborations.



7. **Node** - A physical element existing at run time and represents a resource.



### Behavioral things-

1. **Interaction** - A behavior made up of a set of messages exchanged among a set of objects in a particular context to accomplish a specific purpose.
2. **State Machine** - A behavior that specifies the sequences of states an object or interaction goes through during its' lifetime in response to events

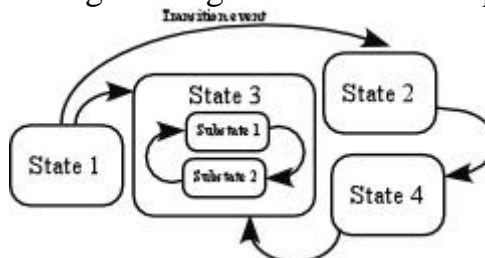


Table 4

### Question:6

Draw a UML class diagram to capture the following situation: "Every student is enrolled in a course. Each student may be enrolled in a set of

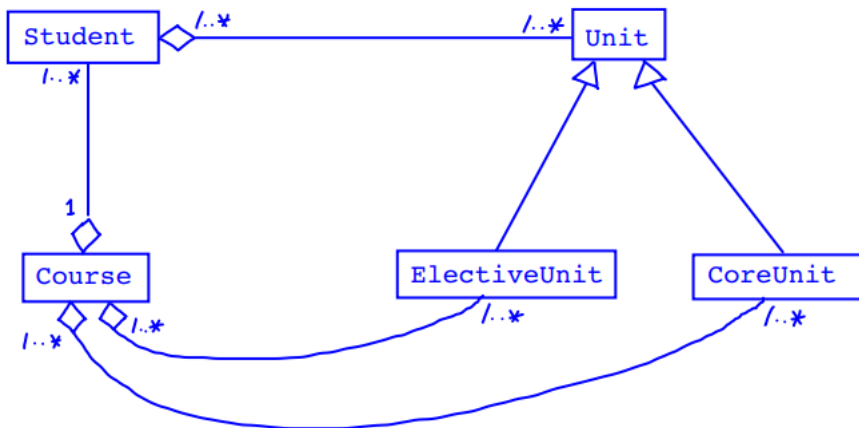
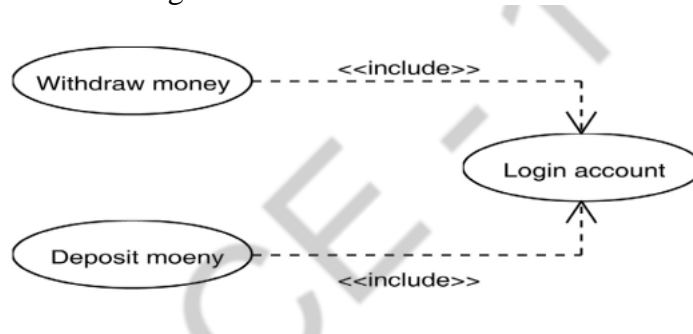
	units. Some units are core units for one or more courses and some units are elective units for one or more courses.”
<b>Solution: 6 (With step wise marking)</b>	 <pre> classDiagram     Student "1..*" -- "1..*" Unit     Student "1..*" -- "1" Course     Student "1..*" -- "1..*" ElectiveUnit     Unit &lt; -- ElectiveUnit     Unit &lt; -- CoreUnit     Course "1..*" -- "1..*" ElectiveUnit     Course "1..*" -- "1..*" CoreUnit     </pre> <p>a Course must have at least one Student enrolled in it and, as every student is enrolled in a Course, the relationship between Course and Student is 1-to-many. Assuming that elective units can be shared by different courses, between Course and ElectiveUnit is a many-to-many relationship (Similarly between Course and CoreUnit).</p>

Table 5

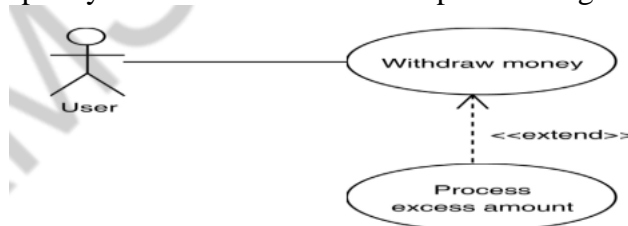
<b>Question:7</b>	<p>Identify the actors and the objects in the following scenario to register a patient in a hospital management system:</p> <p>The administrator enters the patient’s name, address, date of birth and emergency contact details into the system. If the patient has only public health insurance, the administrator enters the patient’s medicare number, and the system verifies this with government health database. If the patient also has private health insurance, then the administrator enters also the patient’s private health insurance details, and the system verifies these details with the private health insurance system. When these details are verified as correct, the system saves the patient's details and confirms the registration.</p>
<b>Solution: 7 (With step wise marking)</b>	<p>Actors: Administrator, Government Health Database, Private Health Insurance System (The last two are external systems)</p> <p>Objects: Patient Administrator Address EmergencyContact PublicHealthInsurance PrivateHealthInsurance Registration</p>
<b>Question:8</b>	Differentiate between include and exclude relationships.
<b>Solution: 8 (With step wise marking)</b>	<p>The &lt;&lt;include&gt;&gt; Relationship</p> <ul style="list-style-type: none"> <li>• Include relationships are used when two or more use cases share some common portion in a flow of events</li> <li>• This common portion is then grouped and extracted to form an inclusion use case for sharing among two or more use cases</li> </ul>

- Most use cases in the ATM system example, such as Withdraw Money, Deposit Money or Check Balance, share the inclusion use case Login Account
- case Login Account



#### The <<exclude>> Relationship

- In UML modeling, you can use an extend relationship to specify that one use case (extension) extends the behavior of another use case (base)
- This type of relationship reveals details about a system or application that are typically hidden in a use case
- The extend relationship specifies that the incorporation of the extension use case is dependent on what happens when the base use case executes
- The extension use case owns the extend relationship. You can specify several extend relationships for a single base use case



<b>Question:9</b>	Discuss role of UML in OO design.
<b>Solution: 9 (With step wise marking)</b>	<p>UML is a modeling language used to model software and non-software systems. Although UML is used for non-software systems, the emphasis is on modeling OO software applications. Most of the UML diagrams discussed so far are used to model different aspects such as static, dynamic, etc. Now whatever be the aspect, the artifacts are nothing but objects.</p> <p>If we look into class diagram, object diagram, collaboration diagram, interaction diagrams all would basically be designed based on the objects.</p> <p>Hence, the relation between OO design and UML is very important to understand. The OO design is transformed into UML diagrams according to the requirement. Before understanding the UML in detail, the OO concept should be learned properly. Once the OO analysis and</p>

	design is done, the next step is very easy. The input from OO analysis and design is the input to UML diagrams.
--	---

<b>Question:10</b>	When to use collaboration diagram?
<b>Solution: 10 (With step wise marking)</b>	The collaborations are used when it is essential to depict the relationship between the object. Both the sequence and collaboration diagrams represent the same information, but the way of portraying it quite different. The collaboration diagrams are best suited for analyzing use cases.

<b>Question:11</b>	Explain the purpose of state-chart diagram? Describe fork and join in UML.
<b>Solution: 11 (With step wise marking)</b>	<p><b><u>State-chart diagram-</u></b>  Statechart diagram <b>describes the flow of control from one state to another state</b>. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.</p> <p>Before drawing a Statechart diagram we should clarify the following points –</p> <ul style="list-style-type: none"> <li>• Identify the important objects to be analyzed.</li> <li>• Identify the states.</li> <li>• Identify the events.</li> </ul> <p><b>Example- state diagram for an online order</b></p> <pre> stateDiagram-v2     [*] --&gt; UnprocessedOrder : Order Received     UnprocessedOrder --&gt; RejectedOrder : [reject] Checked     UnprocessedOrder --&gt; AcceptedOrder : [accept] Checked     RejectedOrder --&gt; JoinNode     AcceptedOrder --&gt; PendingOrder : [some items not available] Processed     AcceptedOrder --&gt; FulfilledOrder : [some items available] Processed/deliver     PendingOrder --&gt; FulfilledOrder : [all items available new supply]     FulfilledOrder --&gt; JoinNode     JoinNode --&gt; [*]   </pre> <p><b>A Fork notation in a UML Activity Diagram</b> is a control node that splits a flow into multiple concurrent flows. This will have one incoming edge and multiple outgoing edges.</p>



**A join node** is a control node that synchronizes multiple flows. This will have multiple incoming edges and one outgoing edge.

Fork vertices in the UML Statechart Diagram serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices. The segments outgoing from a fork vertex must not have guards or triggers. Join vertices serve to merge several transitions emanating from source vertices in different orthogonal regions.



**Question:12**

Write about events and signals in behavioral modeling.

### Introduction to the Events and Signals

- The different things that happen are called events
- Many events occur at the same time and at unexpected times
- Events represent important occurrences that have a time (the time when the event happened) and space (where the event happened)
- Events are used to model stimulus that can make a state machine to change its state
- Events can be of two types -synchronous or asynchronous
- In a synchronous event the sender waits for an action from the receiver on the event. For example, pressing a button on the mouse, a phone call, etc.
- There is no waiting in an asynchronous event. For example, sending an e-mail, letter, delivery of newspaper, etc.

**Solution: 12**  
(With step wise marking)

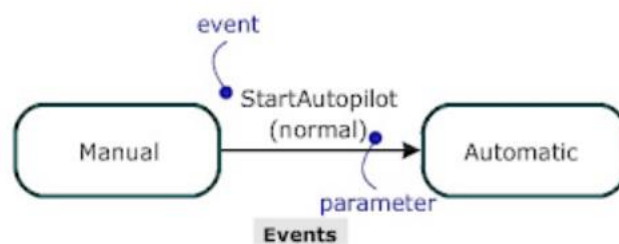


Table 6

<b>Question:13</b>	Illustrate swimlanes and object flow in Activity Diagram.
<b>Solution: 13 (With step wise marking)</b>	<p><b>Swimlanes-</b> Swimlanes are used to show which activities are performed by which organisation in the activity diagram. The lanes are boundaries are drawn and the activities of a particular organisation are drawn in the same lane as that of the organisation. Swimlanes have to be ordered in a Logical Manner.</p> <p><b>Activity</b> In activity diagrams one single activity, In our context, an activity represents a business process. Actions, control elements (decision, branching, merge, start, end, etc.), and objects are essential parts of an activity.</p> <p>These elements are connected to each other with so-called edges. The connected actions and control elements make up the control flow, which can also be called the <i>flow</i>.</p> <p><b>Object flow-</b> The object flow describes the flow of objects and data within activities. Edges can be labeled with a name (close to the arrow): The object flow in an activity diagram shows the path of one or more business objects between the various activities.</p>

<b>Question:14</b>	Discuss dependency defined between packages in a system.
<b>Solution: 14 (With step wise marking)</b>	<p>Two types of dependencies defined between packages:</p> <ul style="list-style-type: none"> <li>• package import</li> <li>• package merge</li> </ul> <p>A <i>package import</i> is "a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace. By default, an unlabeled dependency between two packages is interpreted as a package import relationship.</p> <p>A <i>package merge</i> is "a directed relationship between two packages, that indicates that the contents of the two packages are to be combined. It is very similar to Generalisation in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both.</p>