MAHARASHTRA STATE BOARD OF TECHNICAL EDUCATION
(Autonomous)
(ISO/IEC - 27001 - 2005 Certified)

MODEL ANSWER

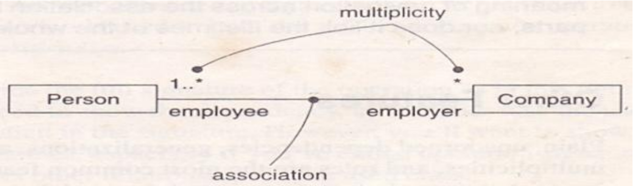
SUMMER – 2018 EXAMINATION

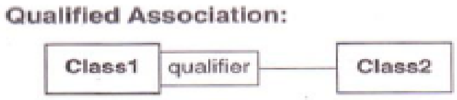
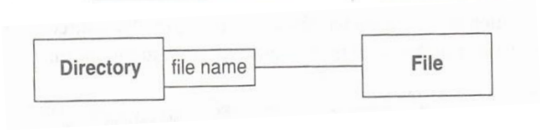
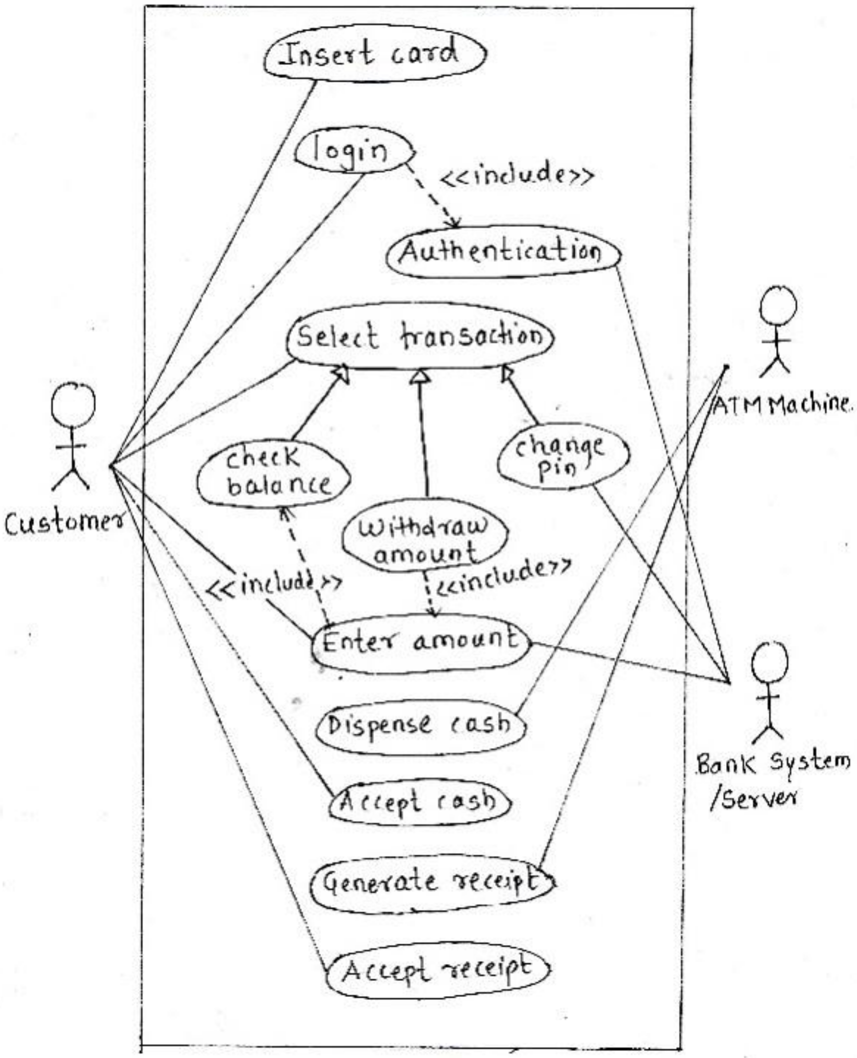
Subject: Object Oriented Modeling & Design **Subject Code: 17630**

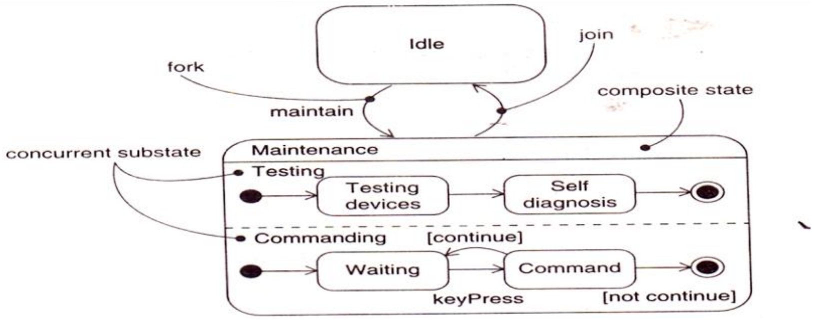
Important Instructions to examiners:

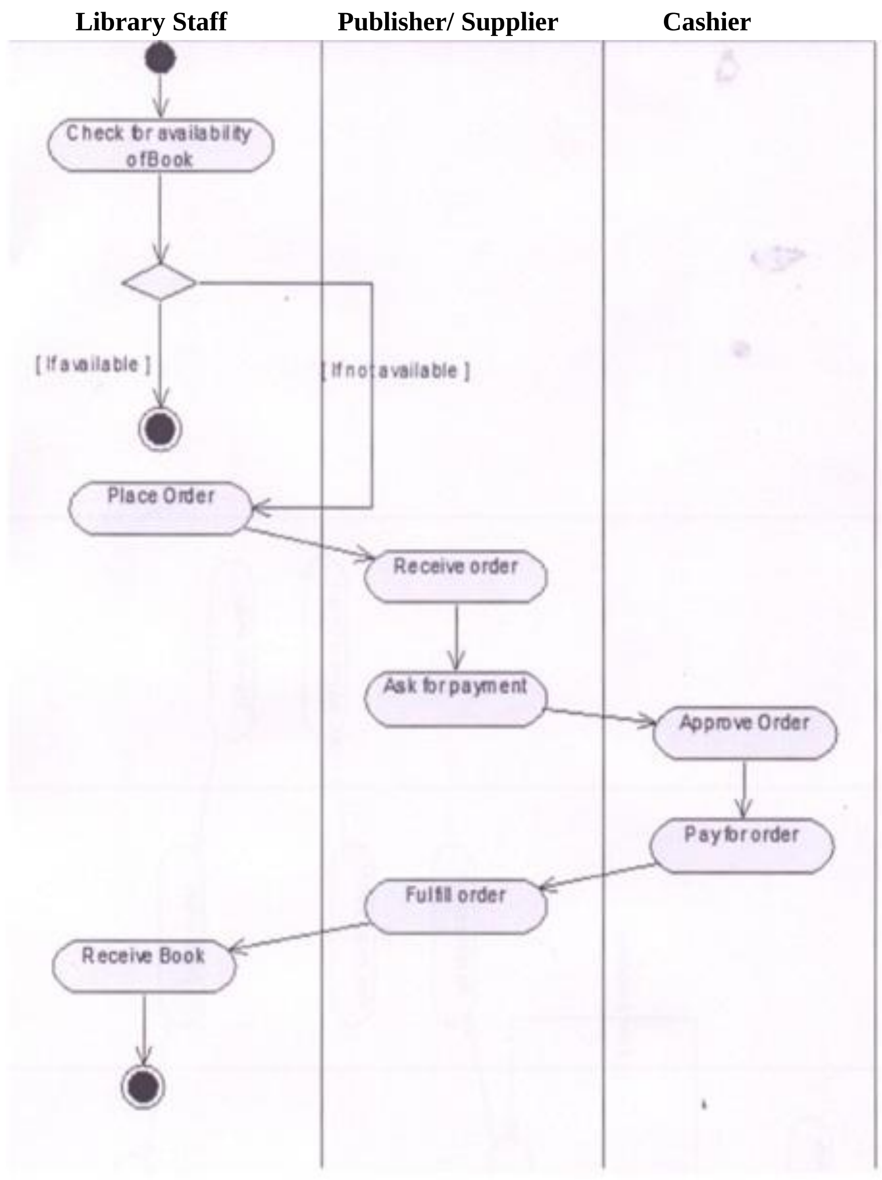
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

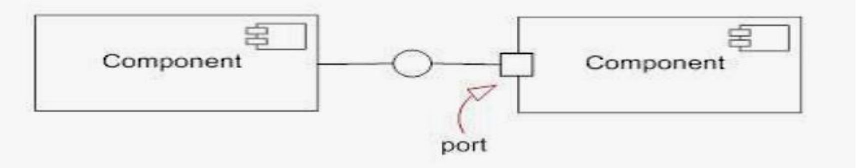
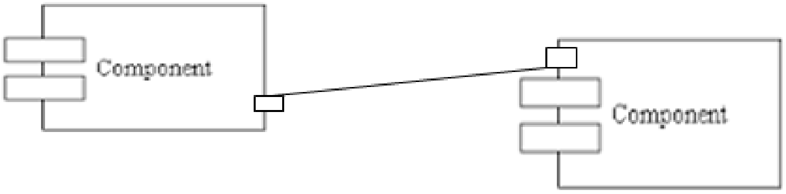
Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	 **(a)** **Ans.**	**Attempt any FIVE of the following:** **Explain object modeling technique (OMT) by Raumbaugh.** **Object Modeling Technique (OMT) by Rambaugh includes four stages:** 1. **Analysis**: - Starting from a statement of the problem, the analyst builds a model of the real-world situation showing its important properties. The analyst works with the requestor to understand the problem statement. The analysis model is a concise, precise abstraction of what the desired system must do, not how it will be done. A good model can be understood and criticized by application experts who are not programmers. The analysis model does not contain any implementation details. 2. **System Design:** - System designer makes high level decisions about the overall architecture. During system design, the target system is organized into subsystems based on both the analysis structure and the proposed architecture. The system designer decides what performance characteristics to optimize, choose a	**20** **4M** ***4 stages of OMT- 1M each***

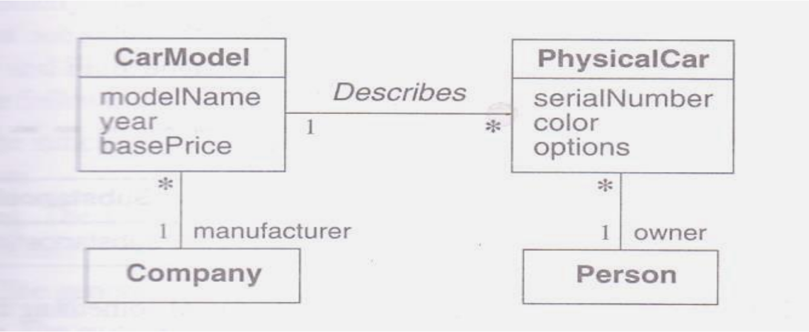
		strategy of attacking the problem and make tentative resource allocations. 3. **Object Design:** - The object designer builds a design model based on the analysis model but contains implementation details. The designer adds details to the design model in accordance with the strategy established during system design. The focus of object design is the data structures and algorithms needed to implement each class. 4. **Implementation**: - The objects, classes and relationships developed during object design are finally translated into a particular programming language, database or hardware implementation.	
	(b) **Ans.**	**Define multiplicity and qualified association with appropriate example.** **Multiplicity: -**Multiplicity specifies the number of instances of one class that mayrelate to a single instance of an associated class. The UML specifies multiplicity with following notations: - ”1” exactly one - “1…*” One or more - “3-5” three to five - “0..1” zero to one - “2,4,18” two, four or eighteen - “*” denotes “many”. **Example:**  multiplicity Person 1..* employee employer * Company association **Qualified association:** Qualified association specifies relation between two object classes and a qualifier. The qualifier is a special attribute that reduces the effective multiplicity of an association. The qualifier distinguishes among the set of objects at the many end of an association. A qualifier is drawn as a small box on the end of the association line near the class it qualifies.	**4M** ***Definition1M*** ***Any one example 1M*** ***Definition1M***

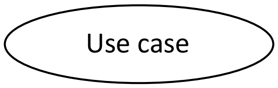
		Notation:  **Example:**  	*Any one example 1M*
	(c) **Ans.**	**Draw use case diagram for ATM system.** ***(Note: Any relevantdiagramshall be considered).***  Insert card login <<include>> Authentication Select transaction check balance Withdraw amount change pin <<include>> <<include>> Enter amount Dispense cash Accept cash Generate receipt Accept receipt Customer ATM Machine. Bank System /Server	**4M** ***Correct diagram 4M***

	(d) **Ans.**	**Describe the concept of concurrent state diagram.** Concurrent state diagram shows a set of independent behaviors of an object. Concurrent sub states are independent and can execute in parallel. A state may be divided into regions containing sub-states that exist and execute concurrently. UML shows concurrency within an object by partitioning the composite state into regions with dashed lines.  Idle join fork composite state maintain concurrent substate Maintenance Testing Testing devices Self diagnosis Commanding [continue] Waiting Command keyPress [not continue] **Example:-** In the above example, concurrent sub states are shown. Maintenance is a composite state. It is decomposed into two concurrent sub-states as testing and commanding. Each of these concurrent sub-states is further decomposed into sequential sub-states. When control passes from Idle to Maintenance state, control then forks to two concurrent flows. Execution of these two concurrent sub-states continues parallel in the system. Each nested state machine reaches its final state. If one concurrent sub state reaches its final state before the other, then control in that sub-state waits at its final state. When both nested state reaches their final state, control from the two concurrent sub-states joins back into one flow.	**4M** ***Relevant description 4M***
	(e) **Ans.**	**Draw activity diagram to purchase books from publisher in Library Management System.** ***(Note: Any relevant diagram shall be considered).***	**4M**

		 Library Staff Publisher/ Supplier Cashier Check for availability of Book [If available] [If not available] Place Order Receive order Ask for payment Approve Order Pay for order Fulfill order Receive Book	***Correct diagram 4M***
	(f) **Ans.**	**Describe use of port and connector in component diagram with example.** **Port:** A port is used to specify an interaction point through which a component can communicate with its environment, other components	**4M** ***Use 1M***

		or with its internal parts. Ports are represented using a square along the edge of a component. A port is often used to help expose required and provided interfaces of a component.  **Connector:** A connector is used to show a link that specifies communication between two or more classifiers. Connectors are of two types- 1. Delegation connector: A component realizes or uses an interface. A component can have internal parts. A part of a component can realize or use an interface. To show a connection among internal parts of a component and interface, delegation connector is used. 2. Assembly connector: It is a connector between two or more parts or ports on parts that defines services provided by parts for other parts.  	***Example 1M*** ***Use 1M*** ***Example 1M***
2.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **State and describe four principles of modelling.** **Principles of modeling are as follows:** 1. "The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped". This means choose your correct model as per the requirement of problem statement. Wrong model will mislead you, causing to focus on irrelevant issues. 2. "Every model may be expressed at different levels of precision:" This means all the user and developers both may visualize a system at different levels of details at different time. 3. "The best models are connected to reality". This means that the	**16** **4M** ***Each principle 1M***

		model must have things that are practically possible. They must satisfy the real word scenarios. 4. “No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models:” This means you need to have use case view, design view, process view, implementation view and development view. Each of these views may have structural as well as behavioral aspects. Together these views represent a system.	
	(b) **Ans.**	**Describe metadata with appropriate example.** **Metadata:** It is a data that describes other data. For example a class definition is a metadata. UML models are also referred as metadata as they describe the things required for the application. Many real world applications have metadata such as parts, catalogues, blue-prints and dictionaries.  CarModel modelName year basePrice Describes 1 * PhysicalCar serialNumber color options * 1 manufacturer Company * 1 owner Person In above example, car model has a model name, year, base price. A physical car has a serial no, color, options. A car model describes many physical car and stores common data about them. A car model is referred as metadata which relates to the data of physical care. A class descriptor object contains feature and they can have their own classes which are known as meta classes.	**4M** ***Description 2M*** ***Any one example 2M***
	(c) **Ans.**	**State and describe any four notations used in use-case diagram.** **Notations used in use case diagram are:** **1. Use case:** Use case is the description of set of sequences of actions. It is graphically represented as an ellipse and labeled with the name of the use case. Use case represents an action performed by a system. **Notation:**	**4M** ***Any four notation 1M each***

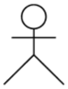


2. Actor:

An actor represents a coherent set of roles that users of use case can play while interacting with use cases. An actor represents a role that a human, hardware device or another system plays when it communicates with the system.

It is represented with the stickman notation.

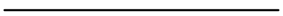
Notation:



3. Communication Line:

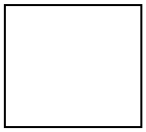
A Communication line is a connection between an actor and use case. It indicates that both are communicating with each other. Communication line is represented with a solid line.

Notation:

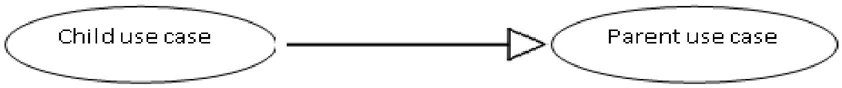
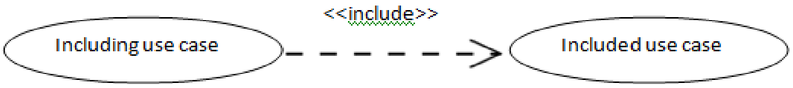
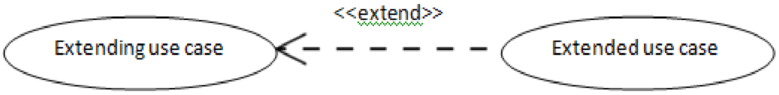


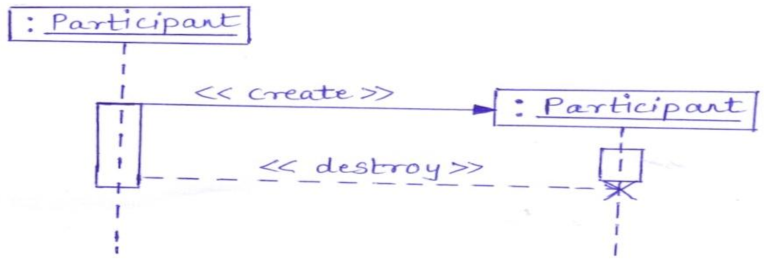
4. System Boundary: System boundary specifies the scope of an application in order to specify functionality. It indicates what the system includes and what it omits. System boundary groups together logically related things. It separates use cases and actors involved in the system. System boundary is shown with a box in a use case diagram.

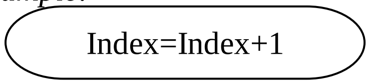
Notation:



5. Generalization: Generalization is used to show the relationship between two use cases. In this relationship the child use case inherits the behavior and meaning of parent use case. It is represented with the solid line with a large hollow triangle as an arrowhead. Arrow head indicates direction of generalization.

		 6. **Include Relationship:** An include relationship is the directed relationship between two use cases. Including a use case requires forceful execution of included use case. **Notation:**  **7. Extend Relationship:** An extend relationship is adirected relationship between two use cases that specifies extra actions in a system. Extend relationship specifies optional behavior for extending use case. **Notation:**  	
	(d) **Ans.**	**Describe create and destroy message used in sequence diagram with example.** **Create message:** Objects can be created according to the requirement of the system in between the processing of the system because they are not required for the entire duration of the sequence diagrams interaction. If an object does not exist at the beginning of a sequence diagram then it must be created in the system. The UML shows creation by placing the object notation at the head of the arrow for the message call that creates an object. **Destroy message:** An object can destroy itself or it can be destroyed by other objects of the sequence diagram because those objects may not be further required during the system. If the object is destroyed by itself then "X" is placed at the tail of the line and arrow head is towards another object to which it passes the control. If the object is	**4M** ***Description 1M each*** ***Example 1M each***

		destroyed by another object then a destroy message is send by another object from the system. In this case the large “X” is placed at the head of the return arrow. ***Example:-***  : Participant << create >> : Participant << destroy >>	
	(e) **Ans.**	**Write difference between action node and activity node in activity diagram.** **Action node:** The executable, atomic computations such as sending a signal to an object, creating or destroying object are called as action nodes. Action nodes cannot be decomposed. *For example*: an expression for calculating gross salary, entering amount for withdrawal cannot be decomposed. *Example:*- Action node Index=Index+1 **Activity node**: an activity is an ongoing non-atomic execution within an activity diagram. Activity results in action. Activity node can be further decomposed in multiple activities. Activity states arenot atomic that means they may be interrupted and they may take some time duration to complete. Activity node is a composite of flow control made up of other activity nodes and action nodes. *Example:*-Activity state Process Bill **OR**	**4M**

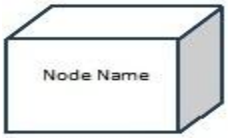
Action node	Activity node
The executable, atomic computations such as sending a signal to an object, creating or destroying object are called as action node.	An ongoing non-atomic computation within an activity diagram is called as Activity node.
Action node cannot be decomposed.	Activity node can be further decomposed in multiple activities.
It gives immediate result. Takes less time for execution.	It may take some time duration to complete the execution.
Example:-  Action node in the above example performs action as increment of value Index.	*Example:-.* Process Bill Activity node in the above example can be further divided into multiple activities or actions.

Any 4 differences 1M each

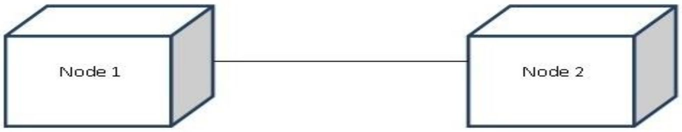
(f) **State and describe any four notations used in deployment diagram.** 4M

Ans. **Notations:-**

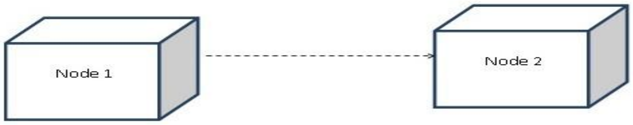
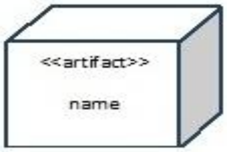
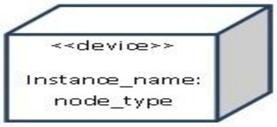
1. **Node:** A node is physical element that exists at runtime & represents a computation resource with some memory and processing capability. Nodes can be a server, printer, cash dispenser etc…



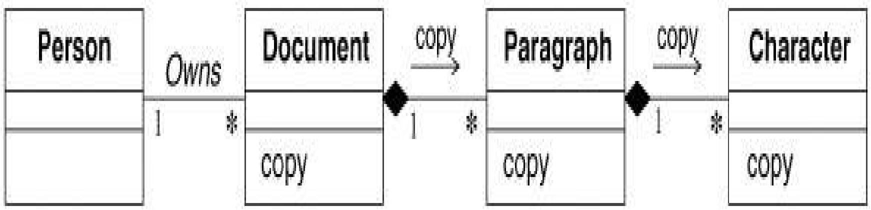
2. **Communication line-Association:** Communication line is used to connect 2 nodes or nodes with other devices. Communication lines specify 2 types of relationship for connecting to either a node or to the component. It is shown with a solid line.

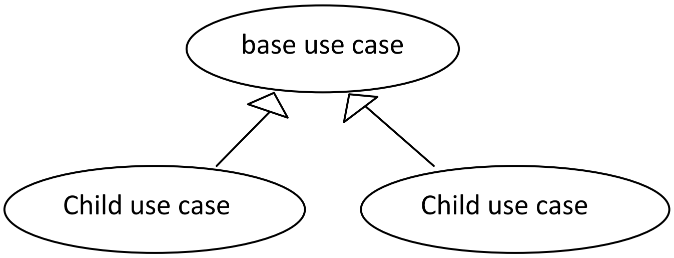
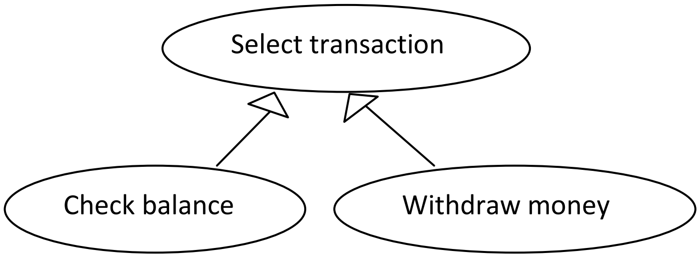


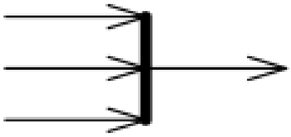
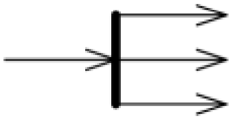
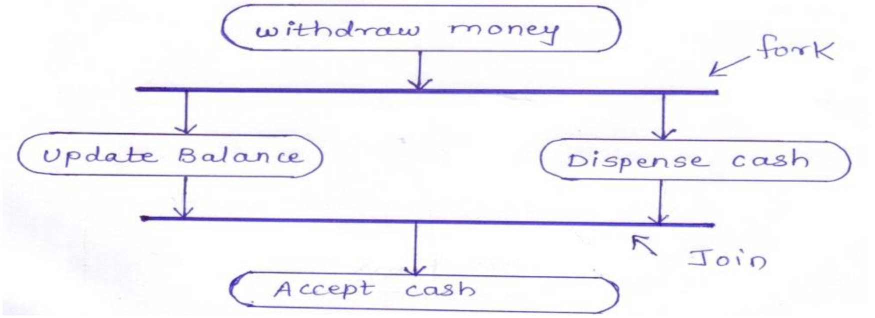
Any 4 notations 1M each

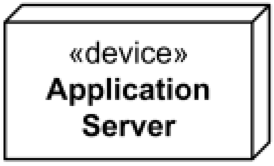
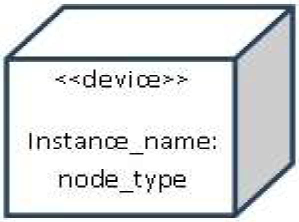
		3. Communication line-dependency: It is used to show relationship between node and a component. A component is placed inside the node to provide processing capability to the node. A node depends on the component. Dependency is shown with dashed line and a arrow head. It connects node with the component arrow head points towards component.  **4. Artifact:** Artifacts are physical file that execute or are used by software of the system. Artifacts includes: 1. Executable files such as .exe or .jar files 2. Library files such as .dll files 3. Source files such as .java or .cpp files 4. Configuration files that are used by software at runtime in specific format such as .xml or .txt  **Node instance:** Instance of a node means two or more nodes of similar node type. In diagram there can be more than one nodes with same properties and structure each node with similar structure is referred as instance of a node. Each instance has its unique identity  	
3.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **List and describe object oriented themes.** 1. Abstraction 2. Encapsulation 3. Combining Data& behaviour 4. Sharing 5. Emphasis on the essence of object	**16** **4M** ***List*** ***(Any 3)*** ***1M***

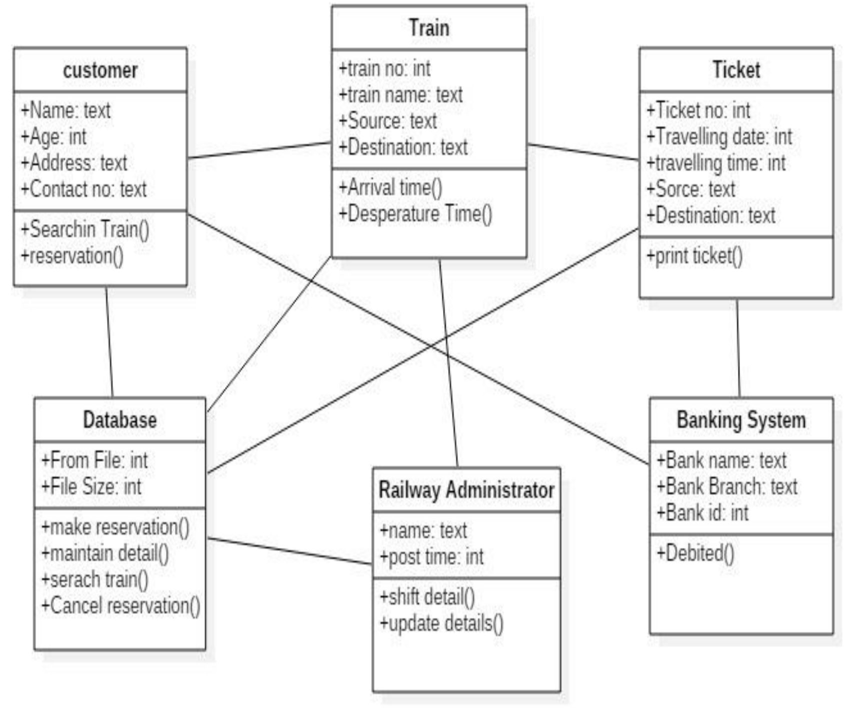
		6. Synergy **1. Abstraction: -** It means focusing on the essential aspects of an entity while ignoring its details. This means focusing on what an object is and does, before deciding how it should be implemented. Use of abstraction during analysis means dealing only with application domain concepts, not making decisions before problem is understood. **2. Encapsulation: -** It means information hiding. It consists of separating the external aspects of an object, which are accessible to other objects, from internal implementation details of the object, which are hidden from other objects. It prevents program from becoming so interdependent that a small change has massive ripple effect. It gives the ability to combine data structure and behaviour in a single entity. **3. Combining Data and Behavior: -** The burden of calling code for data execution and operations separately can be minimized by combining data properties and behavioural properties of an entity together. In object oriented program data structure and procedure is defined in single class definition. **4. Sharing:** Object Oriented technologies promote sharing at different levels. Inheritance of both data structure and behavior lets subclasses share common code. This sharing via inheritance is one of the main advantages of Object Oriented languages. Object Oriented Development not only lets you share information within an application but also offers the prospect of reusing designs and code on future projects. **5. Emphasis on the essence of object:** Object Oriented Technology stresses what an object is, rather than how it is used. The uses of an object depend on the details of the application and often change during development. **6. Synergy:** Identity, classification, polymorphism and inheritance characterize Object Oriented languages. Each of these concepts can be used in isolation but together they complement each other synergistically.	***Any three themes 1M each***
	(b) **Ans.**	**With suitable example describe propagation of operation.** **Propagation of Operation:** Propagation (Also called Triggering) is the automatic application of an operation to a network of objects when the operation is applied to some starting object. Propagation of operations to parts is often a	**4M** ***Description 2M***

		good indicator of propagation. Propagation is very well applicable to aggregated objects, operation carried on whole eventually changes the states of sub objects. ***Example***: A person owns multiple documents. Each document consists of paragraphs to characters. Copying a paragraph copies all the characters in it. The operation does not propagate in the reverse direction; a paragraph can be copied without copying the whole document. Similarly, copying a document copies the owner link but does not generate a copy of the person which is owner. ***Example***:  Person Owns 1 * Document copy copy 1 * Paragraph copy copy 1 * Character copy	***Example 2M***
	(c) Ans.	**Describe synchronous and asynchronous messages used in sequence diagram. Give notations.** **Synchronous Message:** A synchronous message requires a response before the interaction can continue. It's usually drawn using a line with a solid arrowhead pointing from one object to another. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. ⟶ Synchronous Message **Asynchronous Message:** Asynchronous messages don't need a reply for interaction to continue. Like synchronous messages, they are drawn with an arrow connecting two lifelines; however, the arrowhead is usually open and there's no return message depicted. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response. → Asynchronous Message	4M ***Synchronous messages 2M*** ***Asynchronous messages 2M***

	(d) **Ans.**	**Describe the concept of use-case generalization with example.** Generalization is used to show parent. Child relationship among the use-cases. Parent use case represents general behavior of a system. The child use case is a specialized behavior of parent use case with additional processing step. E.g. Select transaction is a base use case and withdraw money, check balance, change pin are sub use cases. Each sub use case defines parent use case with specialized behavior. **Notation:** Generalization is shown with a solid line and hollow triangle placed near the base use case.  **Example**  The child use case is depend on the structure of base use case. The child use case add special behavior to the parent use case by inserting segments of behavior into the inherited behavior. If parent use case is an abstract use case then the child use case redefines it by adding the segments into it.	**4M** ***Description 2M*** ***Example 2M***
	(e) **Ans.**	**Describe joining and forking in an activity diagram.** **Joining:** A joining may represent the synchronization of two or more concurrent flows of control. A joining can have two or more incoming transitions and one outgoing transition. Above the joining,	**4M** ***Joining 2M***

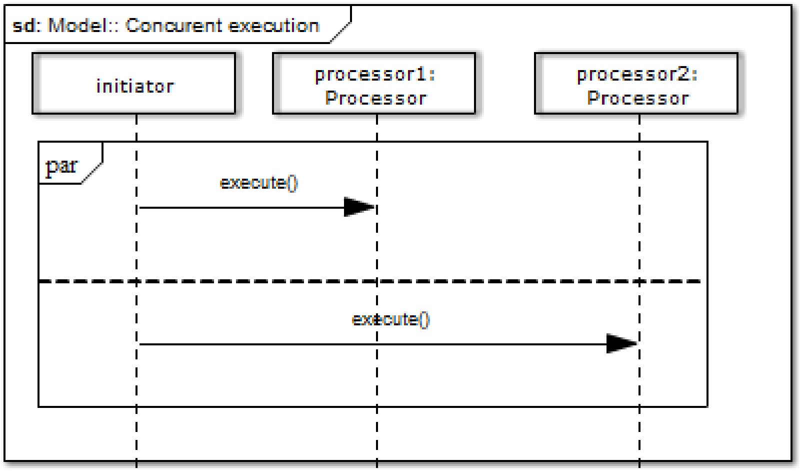
		the activities associated with each of these paths continue in parallel. At the joining, the concurrent flows synchronize, meaning that each waits until all incoming flow have reached the joining, at which point one flow of control continues below the joining. The notation for a joining is a line segment with several activity edges entering it, and only one edge leaving it.  **Forking:** A forking can represent the splitting of a single flow of control into two or more concurrent flows of control. A forking can have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below the forking, the activities associated with each of these paths continue in parallel i.e. concurrently. The notation for a forking is a line segment with a single activity edge entering it, and two or more edges leaving it.  **Example:**  withdraw money fork Update Balance Dispense cash Join Accept cash	***Forking 2M***
	(f) **Ans.**	**Describe the concept of node instance used in deployment diagram. Also write it's notation.** A node is a physical element that exists at run time and represents a computational resource with some memory and processing capability. Instance of node can be hardware device or execution	**4M** ***Description 3M***

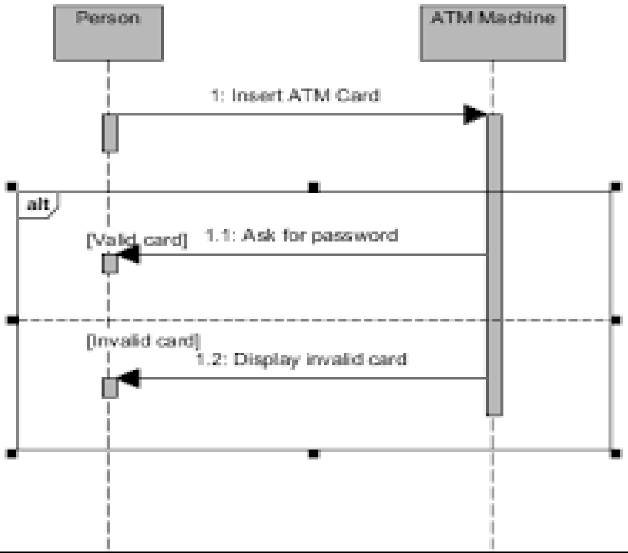
		environment.Hardware device can be server, printer etc. Execution environment can be a node that offers an executionenvironment for specific types of components that are deployed on it in the form of executable.  **OR** Node instance: Instance of a node means two or more nodes of similar node type. In diagram there can be more than one node with same properties and structure each node with similar structure is referred as instance of a node. Each instance has its unique identity.  	***Notation 1M***
4.	**(a)** **Ans.**	**Attempt any TWO of the following:** **Define the terms link and association. Also draw a class diagram for railway reservation system.** ***(Note: Any other relevant diagram shall be considered).*** **Links:** A Link is the basic relationship among objects. It is used in object diagrams. Helps inunderstanding the relationship between objects, with data values and multiplicity diagram. **Association:**An association represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes. An association with three links is called a ternary association. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.	**16** **8M** ***Definition 1M Each***

<table>
<tr>
<td></td>
<td></td>
<td>


</td>
<td><em>Diagram 6M</em></td>
</tr>
<tr>
<td></td>
<td><strong>(b)</strong>
<strong>Ans.</strong></td>
<td><strong>Describe with suitable example, parallel and conditional structured control.</strong>
A sequence of messages is fine for showing a single, linear sequence, but often we need to show conditionals and loops. Sometimes we want to show concurrent execution of multiple sequences. This kind of high-level control can be shown using structured control operators in sequence diagrams.

A control operator is shown as a rectangular region within the sequence diagram. It has a tag text label inside a small pentagon in the upper left cornerto tell what kind of a control operator it is. The operator applies to the lifelines that cross it. This is considered the body of the operator. If a lifeline does not apply to the operator, it may be interrupted at the top of the control operator and resumed at the bottom.</td>
<td><strong>8M</strong></td>
</tr>
</table>

		Parallel execution: The tag is par. The body of the control operator is divided into multiple subregions by horizontal dashed lines. Each subregion represents a parallel (concurrent) computation. In most cases, each subregion involves different lifelines. When the control operator is entered, all of the subregions execute concurrently. The execution of the messages in each subregion is sequential, but the relative order of messages in parallel subregions is completely arbitrary. This construct should not be used if the different computations interact. There are very many real-world situations that decompose into independent, parallel activities, however, so this is a very useful operator.  sd: Model:: Concurent execution initiator processor1: Processor processor2: Processor par execute() execute()	***Parallel Structured Control 4M***
		Conditional execution: The tag is alt. The body of the control operator is divided into multiple subregions by horizontal dashed lines. Each subregion represents one branch of a conditional. Each subregion has a guard condition. If the guard condition for a subregion is true, the subregion is executed. However, at most one subregion may be executed; if more than one guard condition is true, the choice of subregion is nondeterministic and could vary from execution to execution. If no guard condition is true, then control continues past the control operator. One subregion may have a special guard condition [else]; this subregion is executed if none of the other guard conditions are true.	***Conditional Structured Control 4M***



(c) **Describe any four notations used in state diagram. Also draw a state diagram for Hospital Management System.** **8M**

(Note: Any other relevant diagram shall be considered).

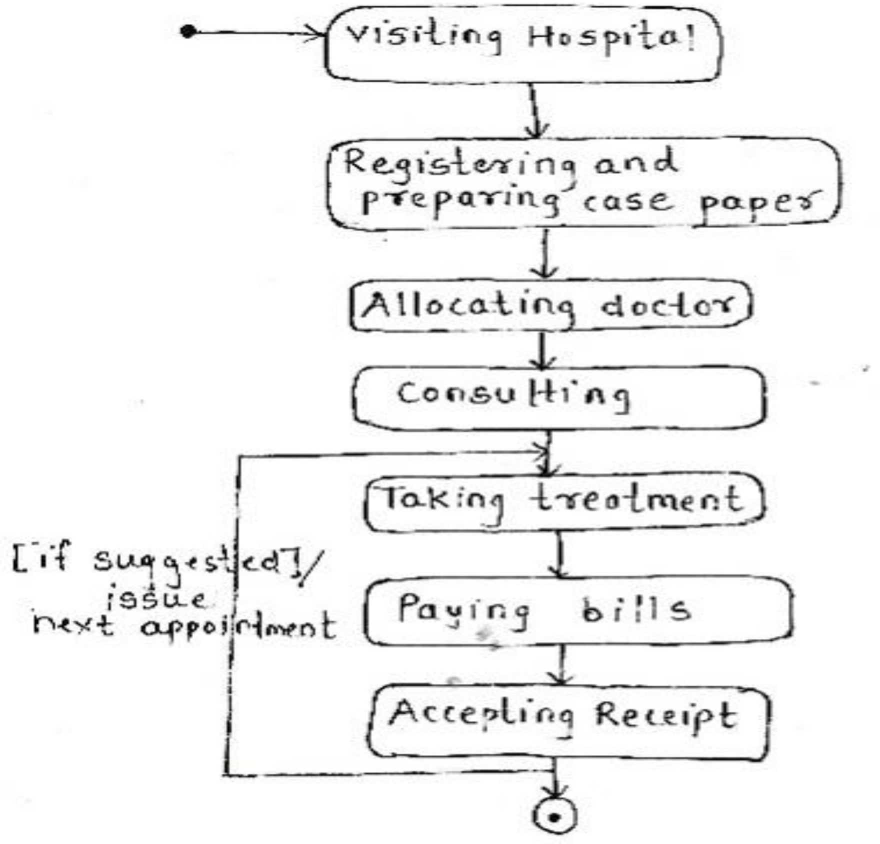
Ans. **State chart diagram notations:**

Sr. No.	**Name**	**Symbol**	**Description**
1	State	▭	A state is a condition or a situation in the life of an object during which it satisfies some conditions, performssome activity or waits for some events. It is represented with a rounded rectangle. Name of the state is written inside the rectangle.
2	Initial State	●	It indicates the default starting place of the state diagram. An initial state is represented as filled circle.
3	Final State	◉	Final state indicates end of the execution of the system. It is represented as a filled black circle surrounded by an unfilled circle.

Any 4 Notations 1M each

4	Transition	→	A transition is a relationship between two states. It indicates that an object in the first state performs someaction and enters in the second state when a specific event occurs. Transition is represented with a directed line.
5	Event	Event →	An event is the specification of a significant occurrence that has location in time and space. An event can be a signal or a call to a function. An event is indicated with text written above or below transition line.
6	Action	Event / Action →	An action is an executable computation. Action may include operation calls, the creation and destruction of another object or sending of a signal to an object. It is indicated with text written below or above the transition line associated with an event separated by slash.

State Diagram for Hospital Management System:

		 Visiting Hospital Registering and preparing case paper Allocating doctor Consulting Taking treatment Paying bills Accepting Receipt [if suggested]/ issue next appointment	***State Diagram 4M***
5.	**(a)** **Ans.**	**Attempt any FOUR of the following:** **Explain Software Development Life Cycle (SDLC) of UML.** The UML is largely process independent i.e. it is not any particular software development life cycle. However, to get the most benefit from the UML, you should consider a process that is: Use case driven Architecture centric Iterative and incremental **1. Inception** It is the first phase of the process, when the seed idea for the development is brought up. The following are typical goals for the Inception phase. ➢ Establish a justification or business case for the project ➢ Establish the project scope and boundary conditions ➢ Outline the use cases and key requirements that will drive the design trade offs ➢ Identify risks	**16** **4M** ***Description of software development life cycle with four correct phases 4M***

- Prepare a preliminary project schedule and cost estimate

2. Elaboration

It is the second phase of the process, when the product vision and its architecture are defined. In this phase, the system's requirements are articulated, prioritized and base lined. The primary goals of Elaboration are to address known risk factors and to establish and validate the system architecture.

3. Construction

It is the third phase of the process, when the software is brought from an executable architectural baseline to being ready to be transitioned to the user community. Construction is the largest phase in the project. In this phase the remainder of the system is built on the foundation laid in Elaboration.

4. Transition

It is the fourth phase of the process, when the software is turned into the hands of the user community. In this phase the system is deployed to the target users.The Transition phase also includes system conversions and user training.The Transition phase also includes system conversions and user training.

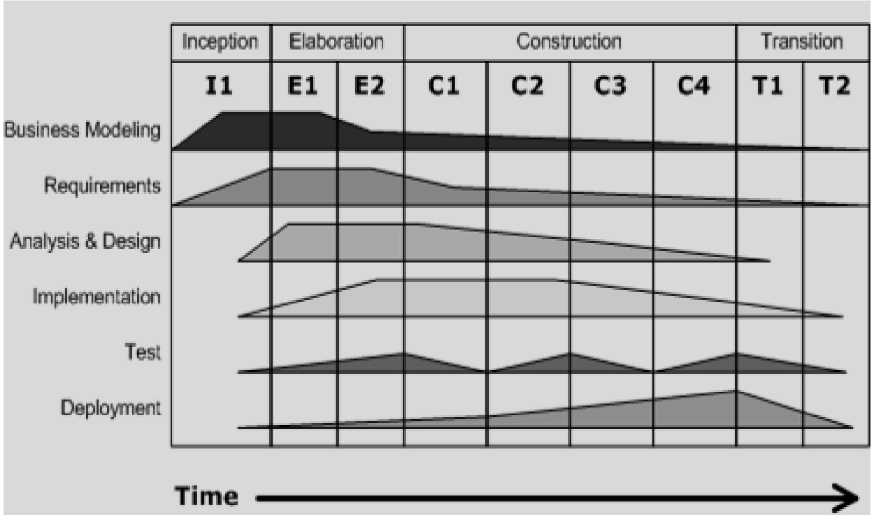
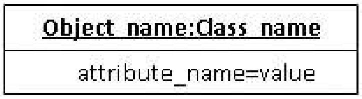
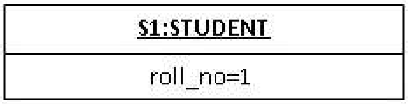
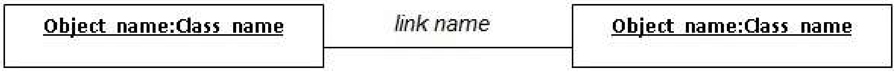
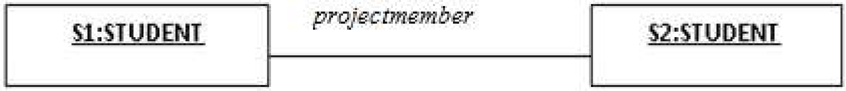
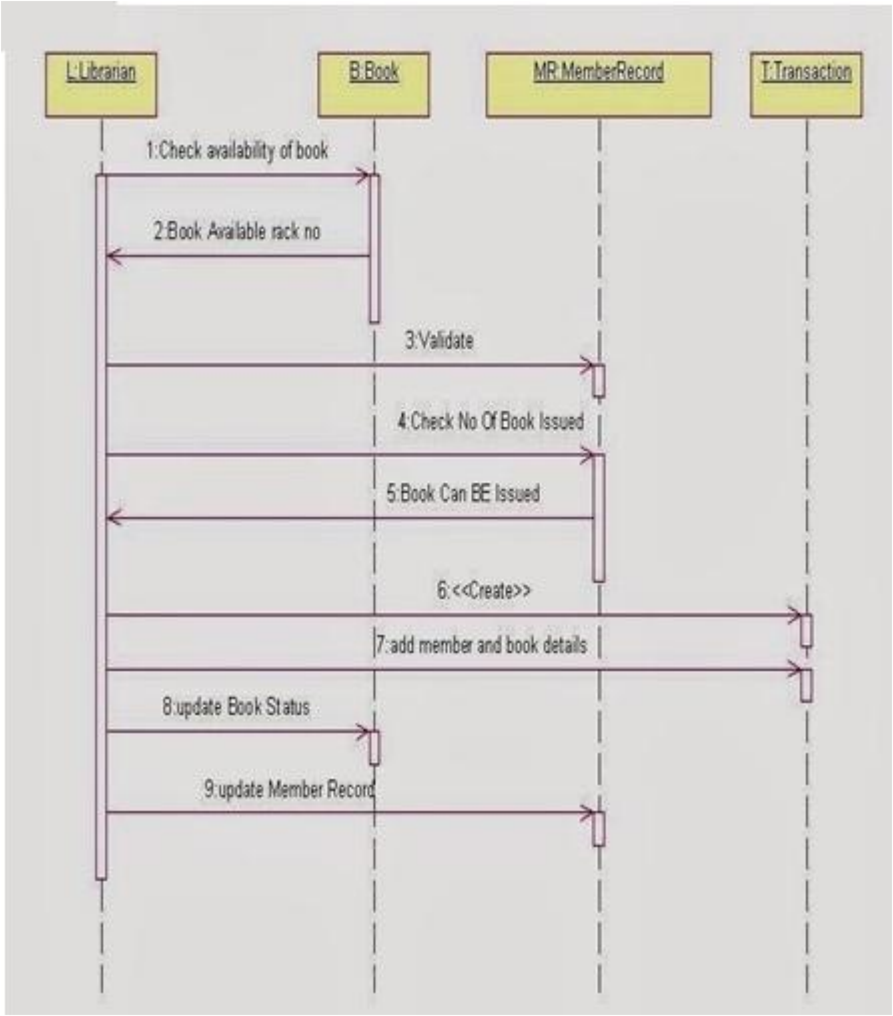
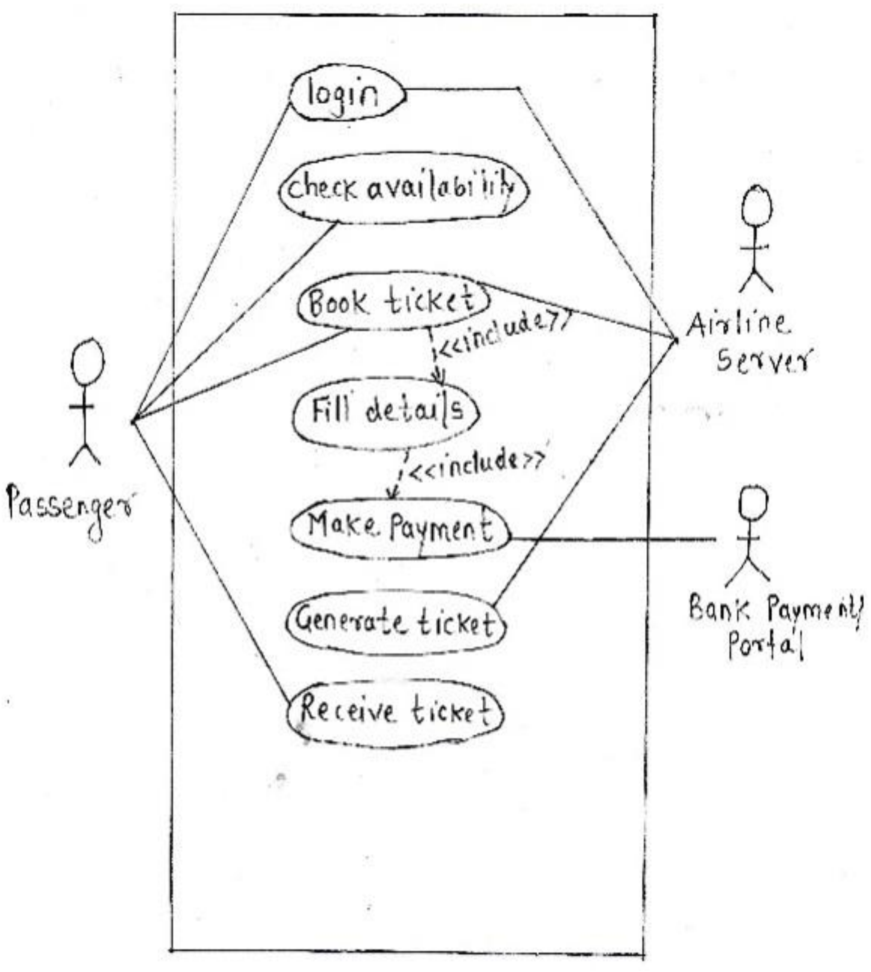


Fig: Software Development LifeCycle

	(b) **Ans.**	**State and Describe notations used in object diagram.** **Notations used in object diagram:** 1. Object 2. Link **1. Object: -** An object is a concept, abstraction or thing that has meaning for an application. Object is basic run time entity. In UML object is represented with a box including its name followed by a colon and class name. Object and class name both are written in bold face with underline. Object can have attributes. Attributes are specified in the second part of the block. Attribute name is followed by value. Notation  Example  **2. Link-** It is physical or conceptual connection among objects. It is used to show relationship among objects. It is represented with a solid line connecting two objects. Name of the link is written in italic form above line.  Object name:Class name link name Object name:Class name ***Example:-*** 	**4M** ***State any two Notations 1M each*** ***Explanation - 1M each***
	(c) **Ans.**	**Draw a sequence diagram for issuing a book from library.** ***(Note: Any relevant diagram shall be considered).***	**4M**

		 L:Librarian B:Book MR:MemberRecord T:Transaction 1:Check availability of book 2:Book Available rack no 3:Validate 4:Check No Of Book Issued 5:Book Can BE Issued 6:<<Create>> 7:add member and book details 8:update Book Status 9:update Member Record	***Correct sequence diagram for issuing a book from library 4M***
	(d) **Ans.**	**Draw the use-case diagram for online Airline Reservation System.** ***(Note: Any relevant diagram shall be considered).***	**4M**

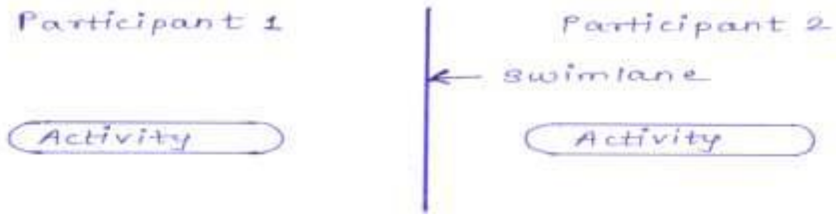
		 login check availability Book ticket <<include>> Fill details <<include>> Make Payment Generate ticket Receive ticket Passenger Airline Server Bank Payment Portal	***Correct use case diagram for online Airline reservation 4M***
	(e) **Ans.**	**Describe the importance of swim lanes in activity diagram.** ***(Note: Any relevant explanation and example shall be considered).*** **Swim lanes:** • Activity diagrams provide an ability to clarify which actor performs which activity. • A swimlane diagram (also sometime called a cross-functional diagram) documents the steps or activities of a process flow or workflow. More specifically, a swimlane diagram groups these activities into swimlanes which are horizontal or vertical columns that contain all of the activities which fit into the category represented by that swim lane.	**4M** ***Explanation 3M***

- Swimlanes can represent many categories of information such as actors which perform the activities (i.e., role or department), the stage of the process in which the activity takes place, or whatever else the creator of the document feels should be emphasized and communicated by the swimlane diagram. The term swimlane was adopted due to the visual similarity between the horizontal rows of the diagram to that of the swimlanes found within a swiming pool.

Following are the Importance of Swimlane.

1. Swim lane diagrams are used for information flows that involve different separate entities that are not necessarily working in a linear sequence.
2. It is used for administrative processes as, for example, order processing, part development, marketing, etc.
3. Each lane represents a different entity. An entity is usually associated with a certain function.
4. Hence, it could be a department, a subgroup of a department, an office, individual people, or it could even be larger than a department as, for example, a plant, a site, the customer, or the suppliers.

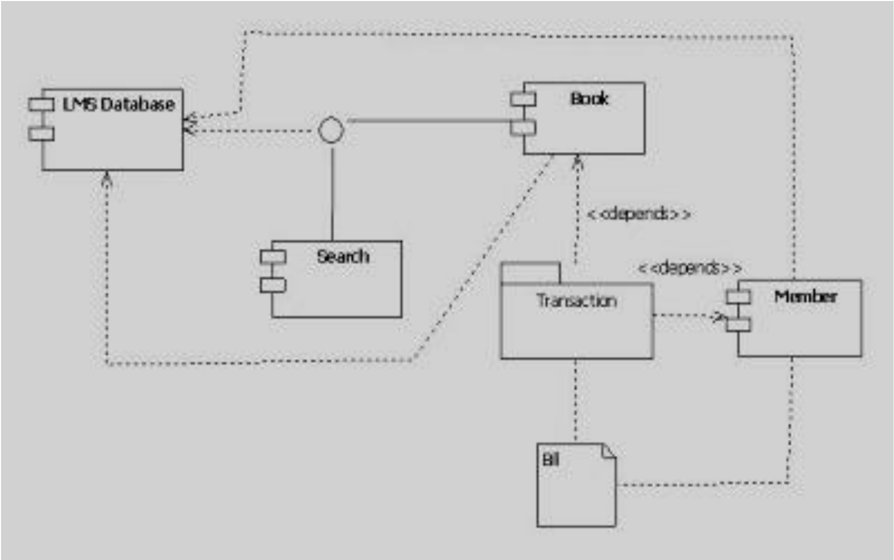
Example:



Example 1M

(f) **Draw Component diagram for Library Management System.** **4M**
(Note: Any relevant diagram shall be considered)

Ans.

		 LMS Database Book Search <<depends>> Transaction <<depends>> Member Bil	***Correct component diagram for library management system 4M***
6.	 **(a)** **Ans.**	**Attempt any FOUR of the following:** **Describe three models of UML.** **Following are the three models of UML:** Three models of OO methodology: 1. Class Model 2. State Model 3. Interaction Model **1. Class Model:** The class model describes the structure of objects in a system, their identity, their relationships to other objects, their attributes and their operations. The class model provides context for the state and interaction models. **2. State Model:** The state model describes those aspects of objects concerned with time and the sequencing of operations also events that mark changes, states that define the context for the events, and the organization of events and states. The state model captures control, the aspect of a system that describes the sequences of operations that occur, without regard for what the operations do, what they operate on, or how they are implemented.	**16** **4M** ***Description 4M***

3. Interaction Model:

The interaction model describes interaction between objects i.e. how individual objects collaborate to achieve the behavior of the system as a whole. The state and interaction models describe different aspects of behavior. Use cases, sequence diagrams and activity diagrams document the interaction model. Use cases document major themes for interaction between the system and outside actors.

(b) Ans.

List and classify various UML diagrams. **4M**

UML Diagrams are classified into Two major category as follows:

1. Structure Diagram

a. Class Diagram
b. Object Diagram
c. Deployment Diagram
d. Component Diagram

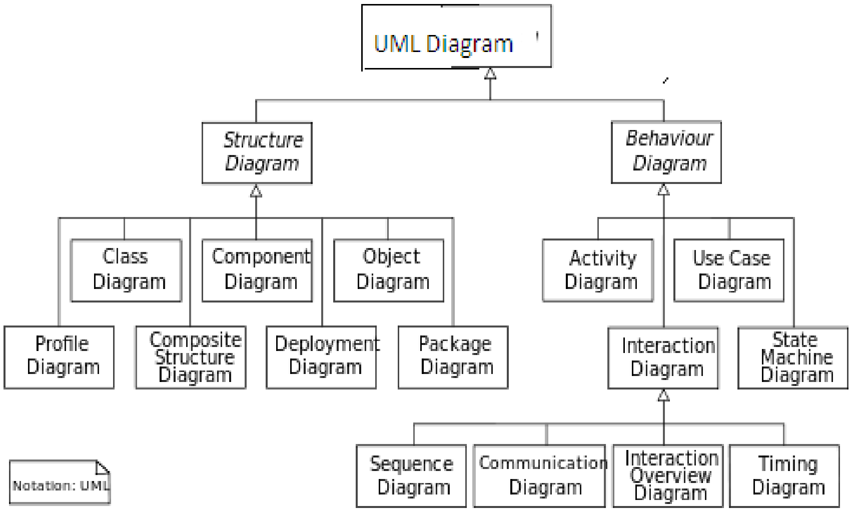
List 2M

2. Behavior Diagram

a. Activity Diagram
b. Use case Diagram
c. State Machine Diagram
d. Interaction Diagram
-Sequence Diagram
- Collaboration Diagram

Classification 2M

OR



(c) Ans.

Describe with example the concept of constraints on link. **4M**

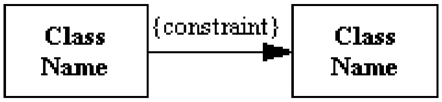
Constraints:

- A constraint represents some condition, restriction or assertion related to some element (that owns the constraint) or several elements.
- Constraint is usually specified by a Boolean expression which must evaluate to a true or false.
- Constraint must be satisfied (i.e. evaluated to true) by a correct design of the system.
- Constraints are commonly used for various elements on class diagrams.
- Constraints are functional relation between entities of an object model.
- Entity includes objects, classes, attributes, links and associations.
- A constraint restricts the values that entities can assume.
- Simple constraints may be placed in object models and complex may be in functional model.

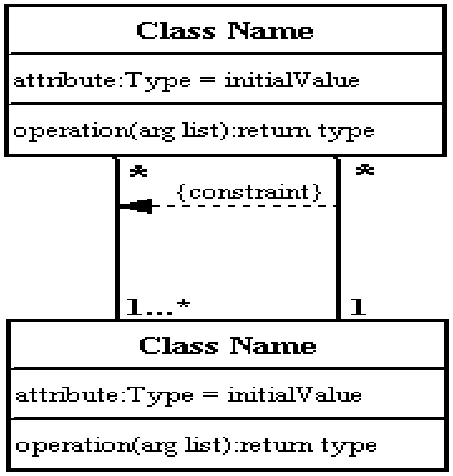
Description 3M

Example:

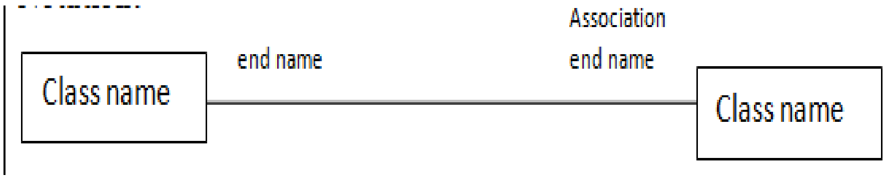
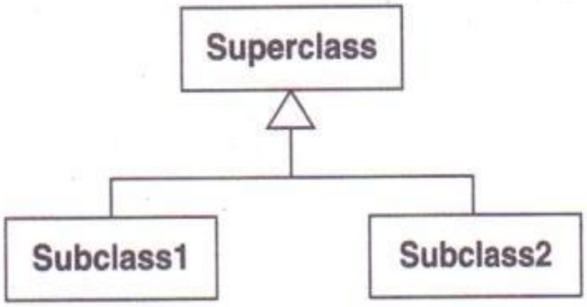
Constraint:



Symbol


Class Name
attribute:Type = initialValue
operation(arg list):return type
*
*
{constraint}
1...*
1
Class Name
attribute:Type = initialValue
operation(arg list):return type

Example 1M

	(d)	**Describe following terms with Notations:** **(i) Association end names** **(ii) Generalization**	**4M**
	Ans.	**(i) Association end names:** An association end names is a name that uniquely identifies one end of an association. It specifies a role of an object of a class which it plays in the association. An association end names is written next to the association line near the class that plays the role.	***Description1M***
		Notation:  Class name end name Association end name Class name	***Notation 1M***
		(ii) Generalization: Generalization is a relationship between a class and one or more derived classes of it. Generalization organizes classes by their similarities & differences, structuring the description of objects. The class being derived is called a super class and its derived classes are called as sub classes. Each sub class is said to "inherit" the features of super class. This property is called as inheritance .The super class holds common attributes, operations and association; the subclasses can add specific attributes, operations &associations.	***Description 1M***
		Notation: A large hollow arrow head denote generalization. The arrow head points to the super class. 	***Notation 1M***

(e) **Describe the concept of decision making and branching in activity diagram.** **4M**

Ans. **Decision Making and Branching:**

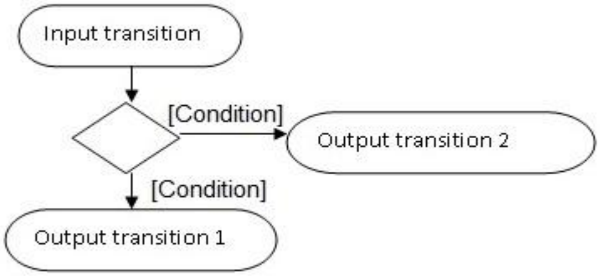
In an activity diagram, branching is used to show alternate path depending on the result of Boolean expression. In a system, some application processing may require flow of control based on Boolean expression. A branch may have one incoming transition and two or more outgoing transitions. On each outgoing transition, we place a Boolean expression, which is evaluated only once on entering the branch. Branching contains a decision box that holds Boolean expression. Depending on result of expression one of the branches is executed.

Explanations 3M

Notation:-

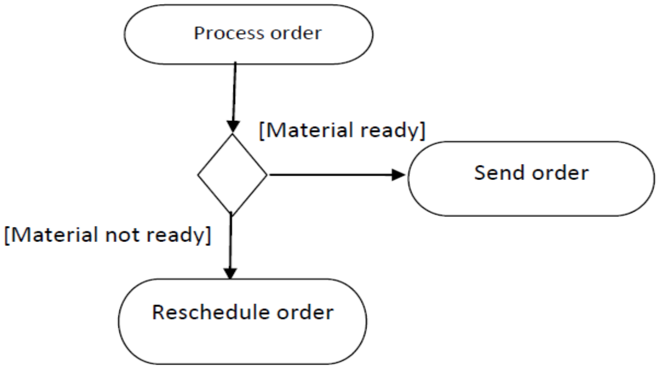
Diamond Shape is used for Decision and branches are represented by lines. The condition written in diamond is the decision criteria. Lines representing branches has guard condition with it.

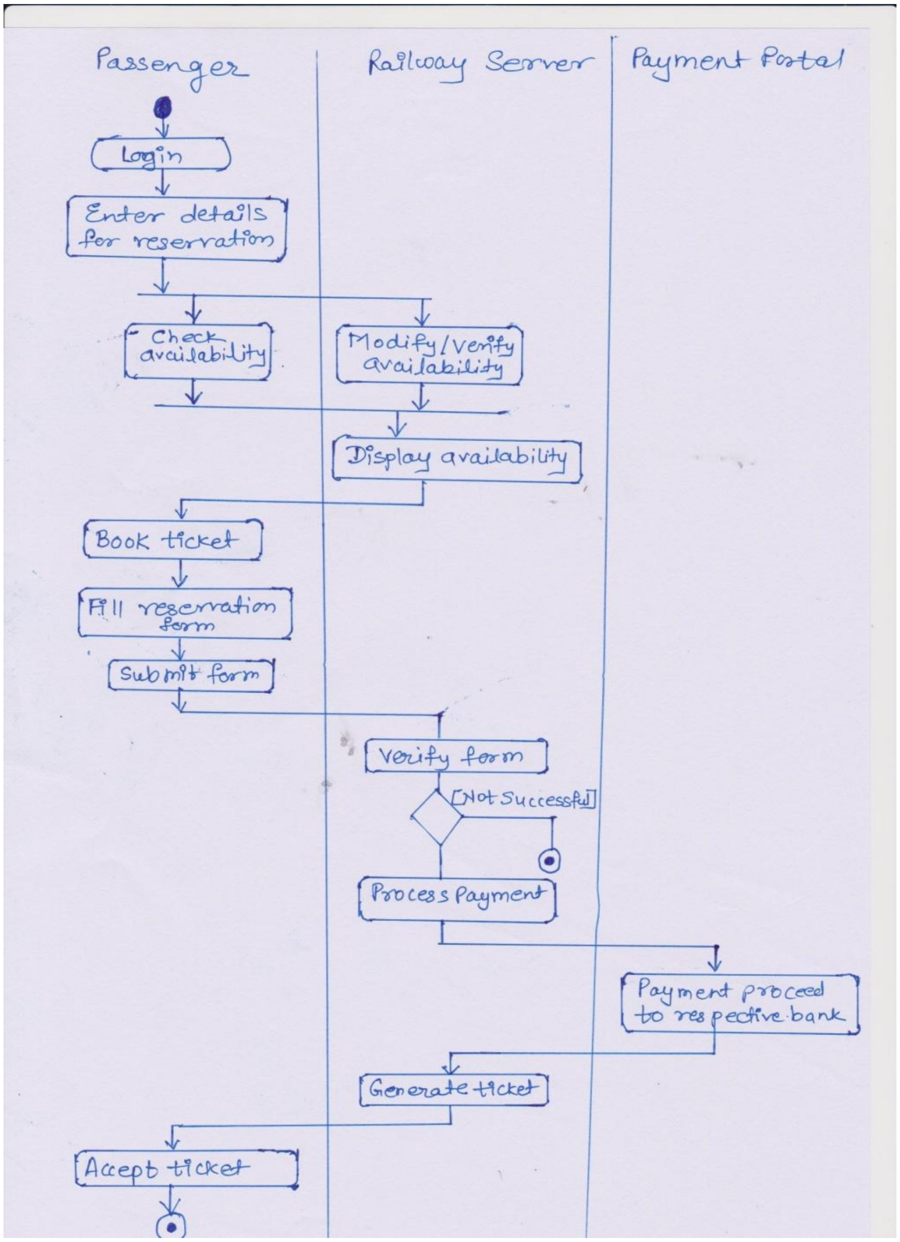
Notation:



Notation or example 1M

Example:



<table>
<tr>
<td></td>
<td><b>(f)</b>
<b>Ans.</b></td>
<td><b>Draw activity diagram for online booking of railway ticket.</b>
<b><i>(Note: Any relevant diagram shall be considered)</i></b>


</td>
<td><b>4M</b>
<b><i>Correct activity diagram for online booking of railway ticket 4M</i></b></td>
</tr>
</table>