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

MODEL ANSWER

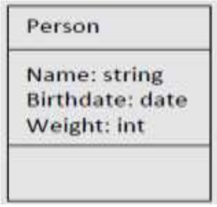
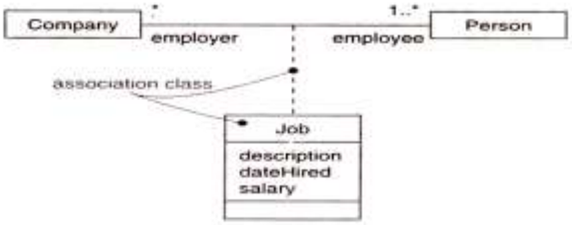
SUMMER - 2017 EXAMINATION

Subject: Object Oriented Modeling and 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.		**Attempt any FIVE of the following:**	**20**
	(a) Ans.	**Explain Object Modelling Technique (OMT) by Rambaugh.** **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 strategy of attacking the problem and make tentative resource allocations.	**4M** ***Four stages-1M each***

		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 object classes and relationships developed during object design are finally translated into a particular programming language, database or hardware implementation.	
	(b) Ans.	**Explain: (i) Attributes (ii) Link attribute with reference to class and object.** **(i) Attributes**: - An attributes is a named property of a class that describes a value held by each object of the class. A class may have any number of attributes or no attributes at all. An attribute represents some property of the thing that is shared by all the objects of that class. Attributes are listed in the second part of Class Box. Each attribute name may be followed by optional details. Each attribute name is unique within a class. For Example: Class Person has attributes Name, Birthdate and weight. Name is string, Birthdate is Date and Weight is integer.  **(ii) Link attributes: -** Link attribute specify properties of link/association between two classes/objects. For example: In the below example company and person has a link/association between them which has link attributes as description, datehired and salary. These attributes are placed inside the association class job. Association class is linked to association line with dashed line.  Company * employer 1..* employee Person association class Job description dateHired salary	**4M** ***Explanation of attribute 2M*** ***Explanation of Link attribute 2M***

(c) **Explain <<include>> and <<extend>> relationships in use case diagram**. **4M**

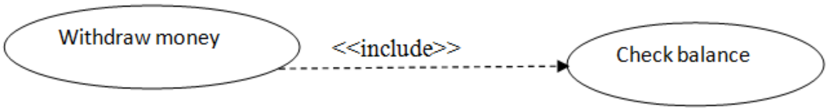
Ans.

<<include>> relationships: Include relationship is used to include one use case within the behavior sequence of another use case.

- An include relationship between use cases means that the base case explicitly incorporates the behavior of another use case at a location specified in the base.
- The include use case never stand alone. When an actor initiates any base use case then base use case executes included use case.
- An include relationship as a dependency can be render with stereotyped as include. To specify the location in a flow of events in which the base use case includes the behavior of another, write include followed by the name of the use case.
- Arrow is placed near the included use case. Arrow is directed from base use case to included use case.

Explanation of include 2M

Example:

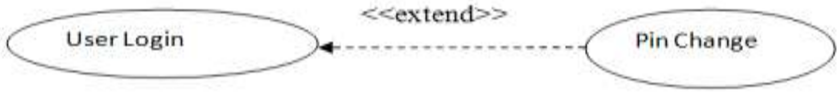


<<extend>> relationships: It adds incremental behavior of an use case.

- A extend relationship between use cases means that the base use case implicitly incorporates the behavior of another use case at a location specified indirectly by the extending use case.
- The extended use case adds itself to the base use case. Most of the time, an extend relationship has a condition attached to it. The extended use case executes only when the condition is true.
- The base use case may stand alone, but under certain conditions, its behavior may be extended by behavior of another use case.
- An extend relationship as a dependency can be render with stereotyped as extend. Arrow is directed from extended use case towards base use case.

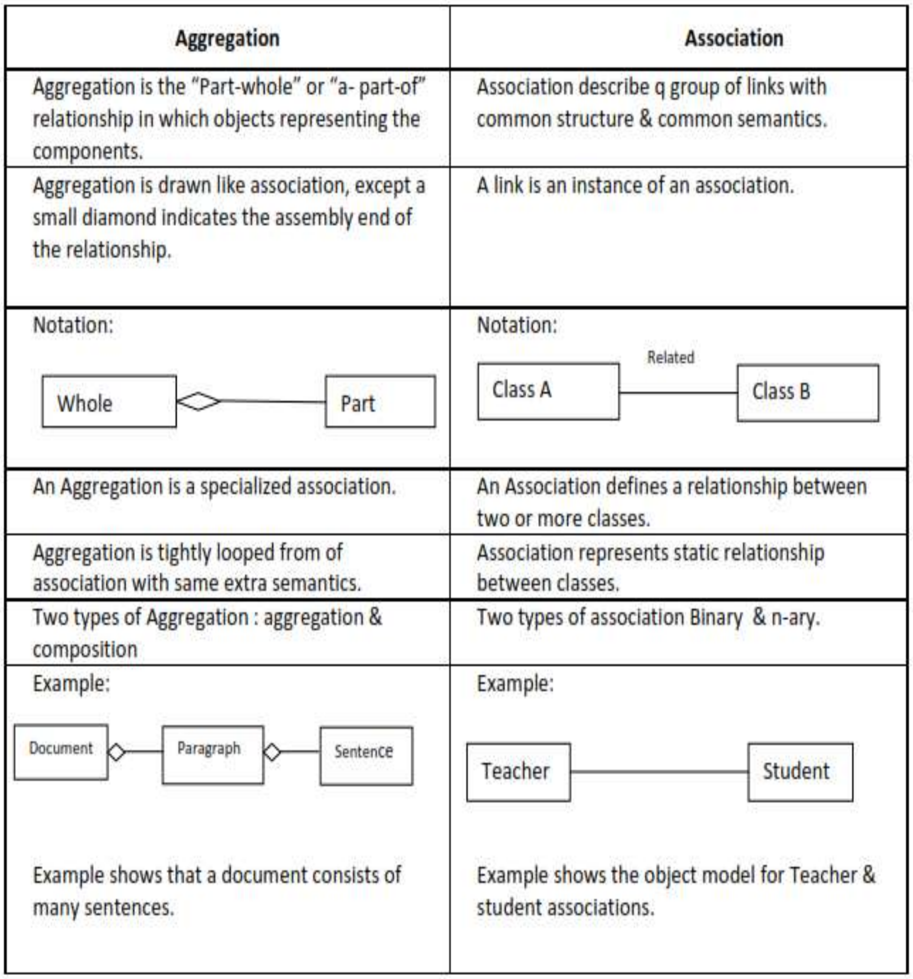
Explanation of extend 2M

Example:



(d) Ans.

Differentiate between aggregation and association. **4M**



Aggregation	Association
Aggregation is the “Part-whole” or “a- part-of” relationship in which objects representing the components.	Association describe q group of links with common structure & common semantics.
Aggregation is drawn like association, except a small diamond indicates the assembly end of the relationship.	A link is an instance of an association.
Notation: Whole ◇—— Part	Notation: Class A —— Related —— Class B
An Aggregation is a specialized association.	An Association defines a relationship between two or more classes.
Aggregation is tightly looped from of association with same extra semantics.	Association represents static relationship between classes.
Two types of Aggregation : aggregation & composition	Two types of association Binary & n-ary.
Example: Document ◇—— Paragraph ◇—— Sentence Example shows that a document consists of many sentences.	Example: Teacher —— Student Example shows the object model for Teacher & student associations.

Any four relevant difference 1M each

(e) Ans.

Explain branching in activity diagram. **4M**

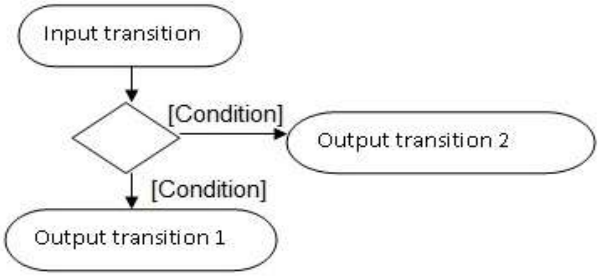
In an activity diagram, branching is used to show alternate path depending on the result of Boolean expression. In a system, some application processing may require flow of control based on Boolean expression.

A branch may have one incoming transition and two or more outgoing transitions. The outgoing transitions are evaluated only when a branch is executed. Branching contains a decision box that holds Boolean expression. Depending on result of expression one of the branches is executed.

Explanation 2M

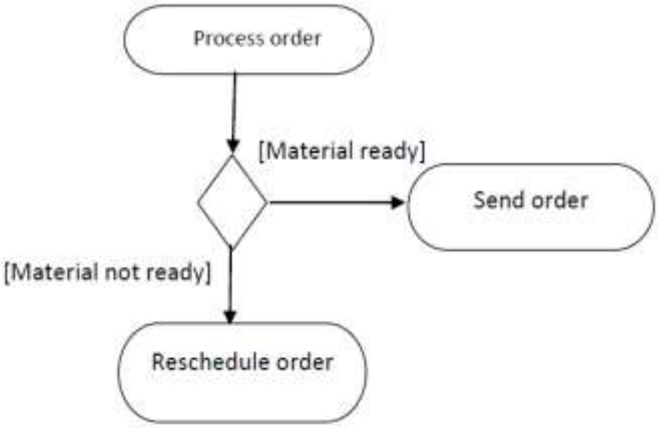
Notation:-

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.



Example:

example 2M



(f) **Define node. Draw any two notations, state two uses of deployment diagram.** **4M**

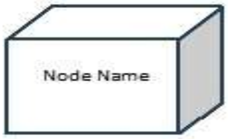
Ans. **Node:**

Nodes are an important building block in modeling the physical aspects of a system. A node is a **physical element** that exists at run time and represents a computational resource, generally having at least some memory and often processing capability. Node can be used to model the topology of the hardware on which the system executes. A node typically represents a processor or a device on which artifacts may be deployed.

Definition 1M

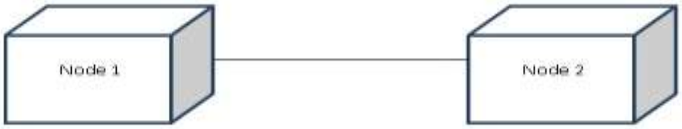
Notations:-

1. Node

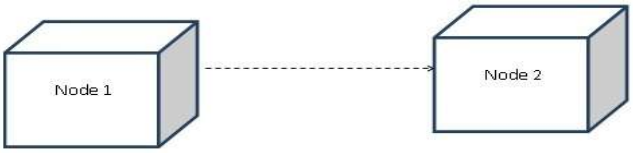


Any two notation 1M each

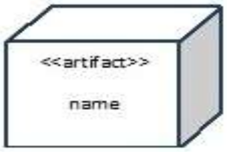
2. Communication line-Association



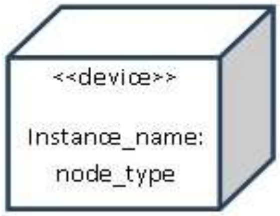
3. Communication line-dependency



4. Artifact



5. Node instance



Uses of deployment diagrams can be described as follows:

- To model the hardware topology of a system.
- To model embedded system.
- To model hardware details for a client/server system.
- To model hardware details of a distributed application.
- Forward and reverse engineering.

Any two uses 1M

(g) Ans. **Explain with diagram create and destroy messages.** ***4M***

Create message:

1. 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.
2. If an object does not exist at the beginning of a sequence diagram then it must be created in the system.
3. The UML shows creation by placing the object notation at the head of the arrow for the message call that creates an object.

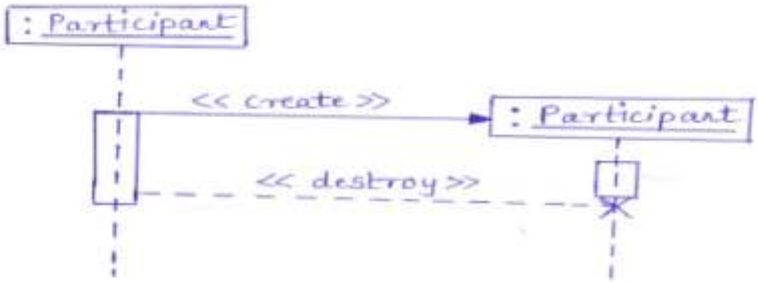
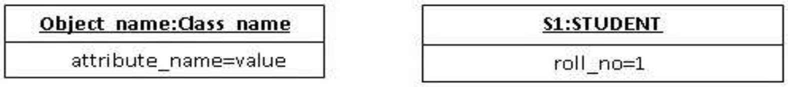
Explanation with diagram of create 2M

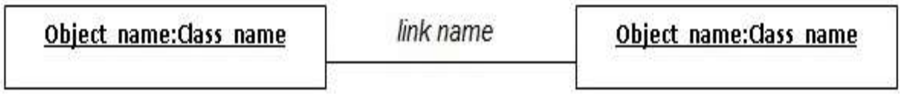
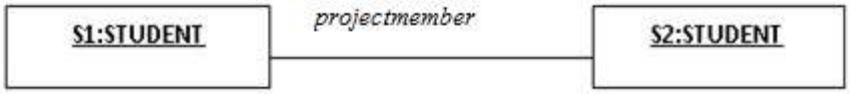
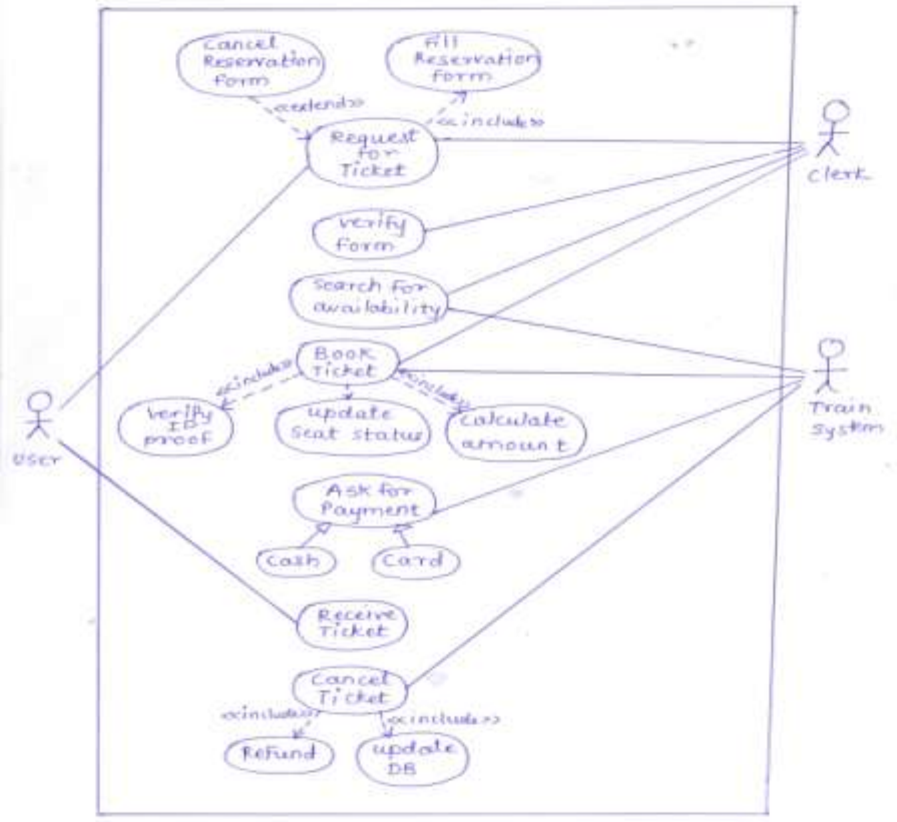
Destroy message:

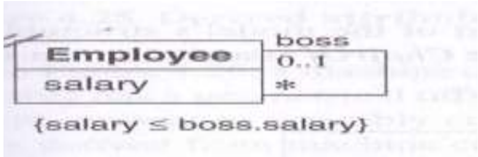
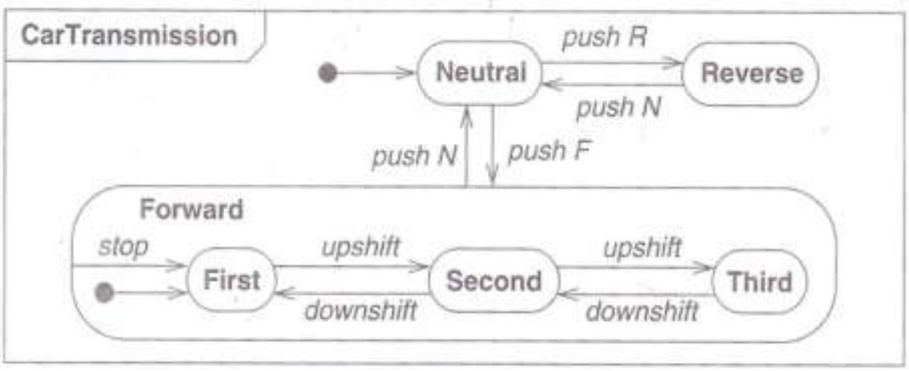
1. An object can destroy itself or it can be destroyed by other objects of the sequence diagram because those objects may not further require during the system.
2. If the object is destroyed by itself then "X" is placed at the tail of the line and arrow head is towards another object to which it passes the control.
3. If the object is 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.

Explanation with diagram of destroy 2M

Example:-

		 : Participant << create >> : Participant << destroy >>	
2.	**(a)** Ans.	**Attempt any FOUR of the following:** **Give any four principles of modelling.** **Principles of modelling 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. 2. Every model may be expressed at different levels of precision. 3. The best models are connected to reality. 4. No single model is sufficient. Every nontrivial system is best approached through a small set of nearly independent models.	**16** **4M** ***Four principles 1M each***
	(b) Ans.	**Draw and explain notations used for object diagram.** **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  Object name:Class name attribute_name=value S1:STUDENT roll_no=1 **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.	**4M** ***Any Two notation - Description and notation 2M each***

		 Object name:Class name link name Object name:Class name *Example:-* 	
	(c) Ans.	**Draw usecase diagram for railway reservation system.** ***(Note: Any other usecase diagram for railway reservation system with correct notations shall be considered).***  Cancel Reservation Form All Reservation form <<extend>> <<include>> Request for Ticket Clerk verify form Search for availability Book Ticket <<include>> <<include>> verify ID proof update seat status calculate amount User Train System Ask for Payment cash Card Receive Ticket Cancel Ticket <<include>> <<include>> Refund update DB	**4M** ***Usecase diagram for railway reservation system 4M***

	(d) Ans.	**Define constraints. How they are applied? Give example.** ***(Note: Any example showing constraint shall be considered)*** **Definition:** Constraint is a Boolean condition involving model elements such as objects, classes, attributes, links and associations. A constraint restricts the values that entities/elements can assume. **Notation:** Constraint is shown as text written inside curly bracket. {Constraint} *Example:* - Constraints on Objects:  Employee salary boss 0..1 * {salary ≤ boss.salary} In the above example a constraint is specified as salary<=boss.salary inside curly brackets. It restricts the value of attribute salary of employees with respect salary of boss.	**4M** ***Definition 2M*** ***Description of any one application with example 2M***
	(e) Ans.	**Explain substate with example.** Composite state has sub states. A Sub state can be sequential or concurrent. Sequential sub states include states of things that change with the help of transition in a particular sequence. Concurrent sub states are the states that executes in parallel. They are independent of each other. In nested state diagram, each of a state can have sub state and sub states receive the outgoing transition from of its composite state.  CarTransmission push R Neutral Reverse push N push N push F Forward stop First upshift Second upshift Third downshift downshift	**4M** ***Explanation 2M*** ***Example 2M***

The above state diagram shows automatic transition with composite state. Automatic transition can be done in reverse, neutral or forward direction. If it is in forward then it can be in first, second or third gear. Forward is a composite state that has three sub states as first, second and third. Selecting N in any forward gear shifts a transition from forward to neutral. Select F in neutral state causes a transition to forward state. The sub state first is the default initial state.

(f) **Describe conceptual model of UML with neat diagram.** **4M**

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

Ans. Conceptual model of UML consist of basic building blocks, the rules that dictate how those building blocks may put together and some common mechanisms that apply throughout the UML.

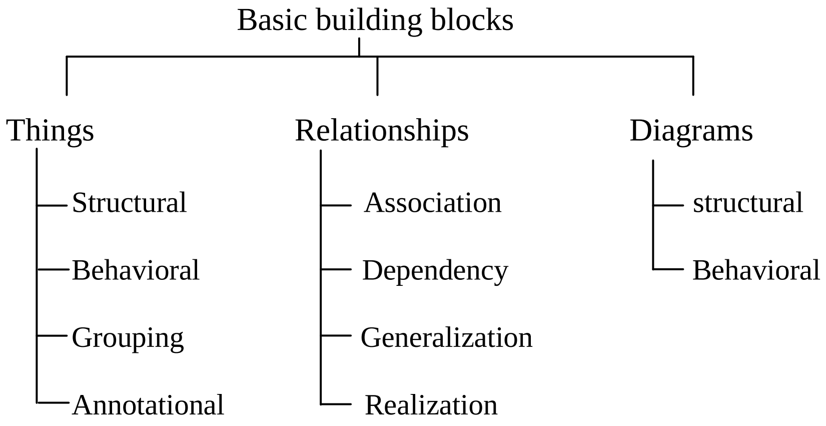


Diagram 1M

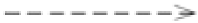
Things: - things are the abstractions that are first citizens in a model.

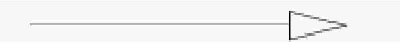
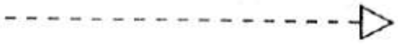
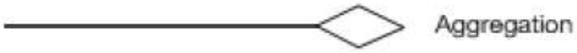
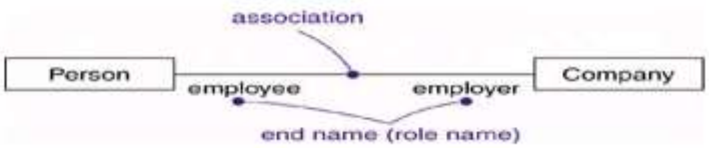
1. Structural things: They are nouns of UML model. These are the static parts that represent elements that are either conceptual or physical. It includes class, interface, collaboration, use case, active class, component, and node.
2. Behavioral things: They are dynamic parts of UML model. It includes interaction, state machine.
3. Grouping things: They are the organizing parts of UML model. It includes package.
4. Annotational things: They are the explanatory parts of UML model. It includes notes.

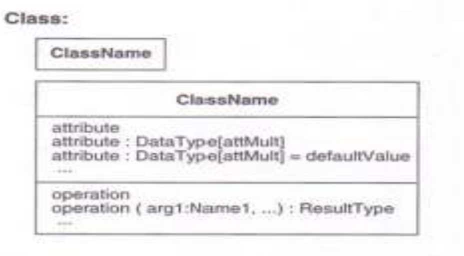
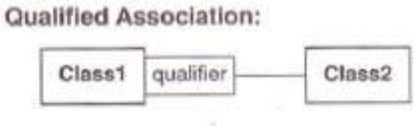
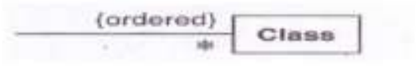
Relevant explanation 3M

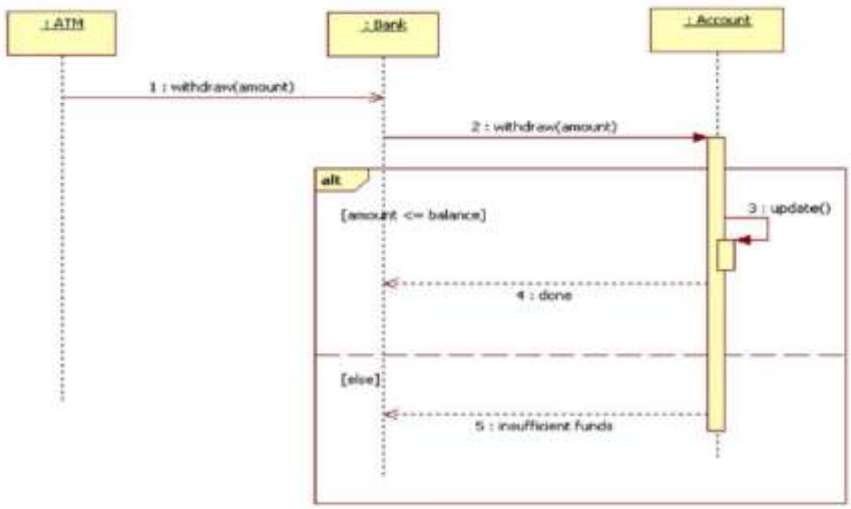
Relationships: The relationships are basic building blocks of the UML.

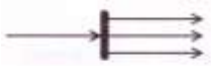
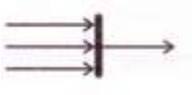
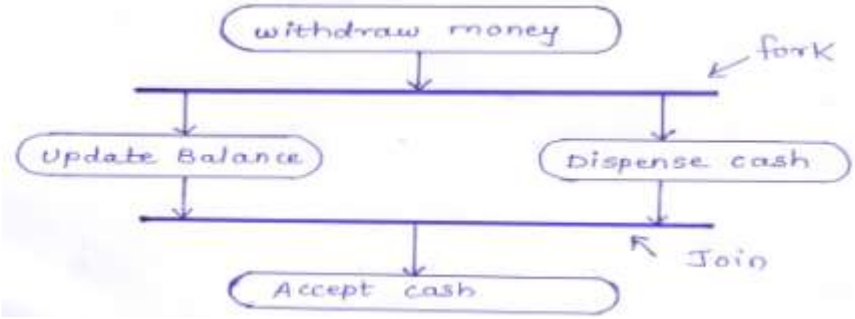
1. Association: It is structural relationship that describes a set of links among objects.
2. Dependency: It is a semantic relationship between two things in which

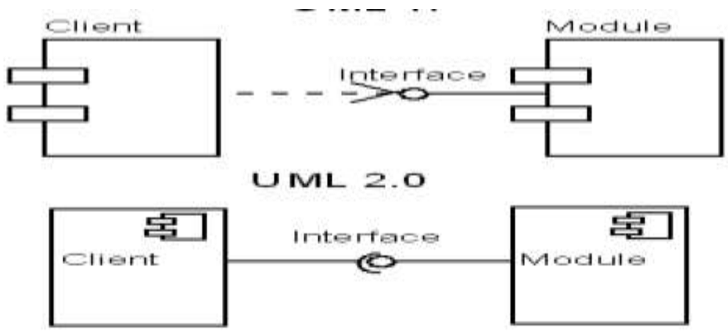
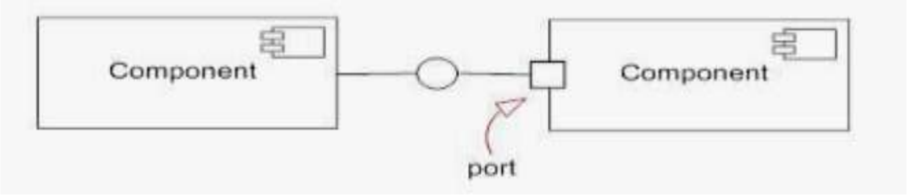
		change to one thing may affect the semantics of the other thing. 3. Generalization: It is a relationship in which objects of the specialized element are substitutable for objects of the generalized element. 4. Realization: It is a semantic relationship between classifiers, wherein one classifier specifies a contract that another classifier guarantees to carry out. Diagrams: 1. Structural diagrams: It is used to visualize, specify, construct and document the static aspects of a system. It includes class, object, and component and deployment diagram. 2. Behavioral diagrams: It is used to visualize, specify, construct and document the static aspects of a system. It includes use case, sequence, collaboration, state chart and activity diagram.	
3.	**(a)** Ans.	**Attempt any FOUR of the following:** **Explain different relationships in UML.** **Relationships in the UML:** **1. Dependency:** A dependency relationship specifies that a change in the specification of one thing may affect another thing that uses it. Graphically, a dependency is rendered as a dashed line, directed to the thing that is dependent on. *Notation:*  **2. Association:** An association is a structural relationship, specifying that objects of one thing are connected to objects of another. When an association connects two classes you can navigate from one object of one class to an object of another class and vice versa. Graphically an association is represented as a solid line connecting more than one class. *Notation:* **3. Generalization:** It is a relationship between a general thing (called the super class or parent) and a more specific kind of that thing (called the subclass or child). With generalization relationship a child can inherit all the structure and behavior of parent. A child may add new structure or behavior or it may modify the behavior of the parent. Graphically it is represented as a solid line with hollow triangle placed near to the super	**16** **4M** ***Any 4 Relation ships 1M each***

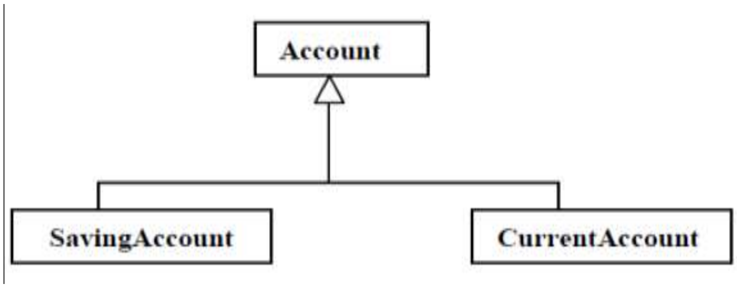
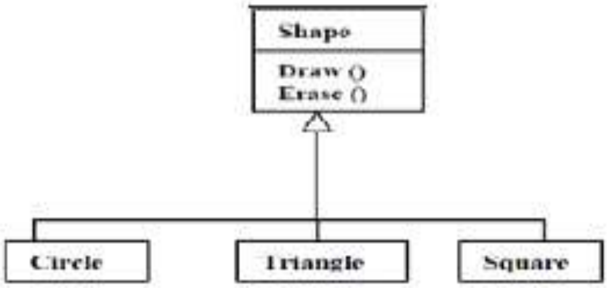
		class. *Notation:*  **4. Realization:** A realization is a schematic relationship between classifiers in which one classifier specifies a contract that another classifier guarantees to carry out. Graphically a realization is represented as dashed directed line with the hollow triangle pointing to the classifier that specifies the contract. *Notation*  **5. Aggregation:** Aggregation is a part whole relationship where an aggregate class connects multiple subclasses which are part of aggregate class. It is represented with a diamond towards aggregate class. *Notation*  	
	(b) Ans.	**Define following terms with notations :** **(i) Role names** **(ii) Class** **(iii) Qualified association** **(iv) Ordered association** **(i) Role names:** A role name 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. A role name is written next to the association line near the class that plays the role. *Example:-*  association Person employee employer Company end name (role name)	**4M** ***Explanation of terms with notation 1M each***

		(ii) Class: A Class is a group of objects with similar properties (attributes), common behavior (operation), common relationship to other objects and common semantics. *Example:* fruit, student, employee etc. *Notation:*  Class: ClassName ClassName attribute attribute : DataType[attMult] attribute : DataType[attMult] = defaultValue ... operation operation (arg1:Name1, ...) : ResultType ... **(iii) 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. *Notation:*  **(iv) Ordered association:** Usually the objects on the "many" side of an association have no explicit order, and can be regarded as a set. Sometimes the objects on the many side of an association have order. Writing {ordered} next to the multiplicity dot indicates an ordered set of objects of an association. Notation: 	
	(c) Ans.	**Explain structured control in sequence diagram for conditional execution.** **Structured control in sequence diagram:** It defines statements or group of statements in a diagram which	4M

		determines the sequence of execution of other instructions or statements. **Conditional execution:** 1. In sequence diagram conditional statements are used to check the condition. 2. Alt operator is used to indicate the condition in sequence diagram. The body of the control operator is divided into multiple sub regions by horizontal dashed lines. 3. Each sub region represents one branch of a condition. Each sub region has a guard condition. If the guard condition for a sub region is true, then that sub region is executed. 4. At most one sub region may be executed. 5. One sub region may have the special guard condition [else]. This sub region is executed if none of the other guard conditions are true. *Example:*  : ATM : Bank : Account 1 : withdraw(amount) 2 : withdraw(amount) alt [amount <= balance] 3 : update() 4 : done [else] 5 : insufficient funds In the above example alt operator is used to show condition as 'amount >=balance'. If the condition is true then, update operation will be perform on account object. If the condition is false then, else region executes if a message as insufficient funds from account object to bank object.	***Explanation 2M*** ***Example 2M***
	(d) Ans.	**Explain forking and joining with diagram.** **Fork in Activity diagram:** A fork in activity diagram represents the splitting of a single flow of	**4M**

		control into two or more concurrent flows of control. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. A synchronization bar is used to connect incoming and outgoing transitions. *Notation*:  **Join in Activity diagram:** A join in Activity diagram represents the synchronization of two or more concurrent flows of control. A join may have two or more incoming transitions and one outgoing transition. Above the join, activities associated with each of these paths continue in parallel. At the join, the concurrent flows synchronize means each flow waits until all incoming flows complete their execution. A synchronization bar is used to connect incoming and outgoing transitions. *Notation:*  *Example:*  withdraw money fork Update Balance Dispense cash Join Accept cash	***Explanation of forking with example 2M*** ***Explanation of joining with example 2M***
	(e) Ans.	**Explain concept of interface and ports.** **Interfaces:** A component can be connected with other components through interfaces. An interface is a collection of operations that are used to specify services of components. A component can provide an interface or can use services of a component. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface.	**4M** ***Explanation of interface 2M***

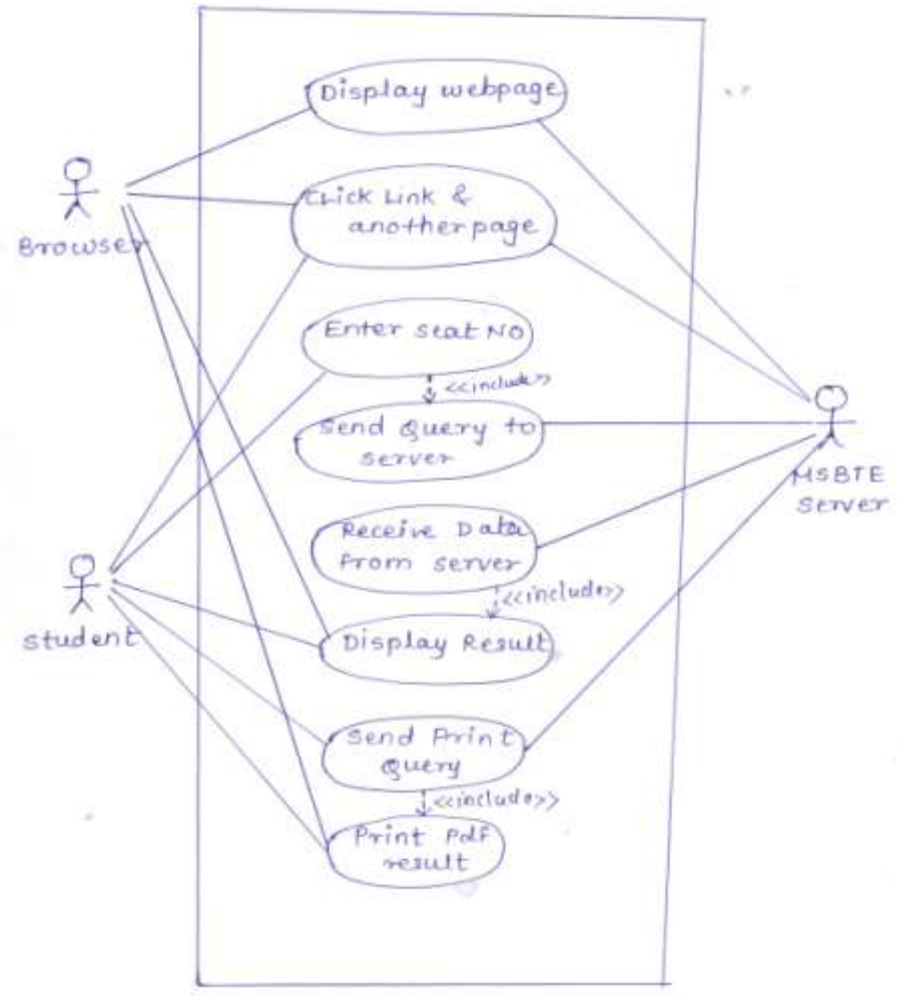
		 Client Module Interface UML 2.0 Client Interface Module **Ports:** A port specifies an interaction point through which a component can communicate with its environment, other components or with its internal parts. Ports are represented using a square along the edge of a component. A port is often used to help expose required and provided interfaces of a component. 	***Explanation of ports 2M***
	(f) Ans.	**Explain generalization and inheritance.** **Generalization:** It is also referred as 'is-a' relationship. It is relationship between a class (super class) and one or more variations of the class (sub classes).It organizes classes by their similarities and differences, structuring the description of objects. The super class holds common attributes, operations and association. The subclasses add specific attributes, operations and associations. Each sub class inherits the features of its super class. *Notation:* A large hollow arrowhead is used to show generalization. The arrowhead points towards the super class. *Example:-*	**4M** ***Explanation of generalization 2M***

		 **Inheritance:** It is the mechanism of inheriting features of super class in its subclass. Inheritance provides reusability of code where code declared for super class can be used by its sub class. *Example:*  Shape Draw () Erase () Circle Triangle Square	***Explanation of inheritance 2M***
4.	**(a)** Ans.	**Attempt any TWO of the following:** **State any two types of actors used in usecase diagram. Draw a neat usecase diagram for printing result from MSBTE website.** An actor specifies a role played by a user or any other system that interacts with the subject. Types of actors used in use case diagram: 1) Primary/principle Actor: People who use the main system functions are refereed as primary or principle actors. Example: in ATM system primary actor is customer 2) Secondary Actor: People who perform administrative or maintenance task are referred as secondary actors. It provides a service to the system under design. Example: in ATM system a person in charge of loading money into the system is a secondary actor.	**16** **8M** ***Any two types of actors 2M each***

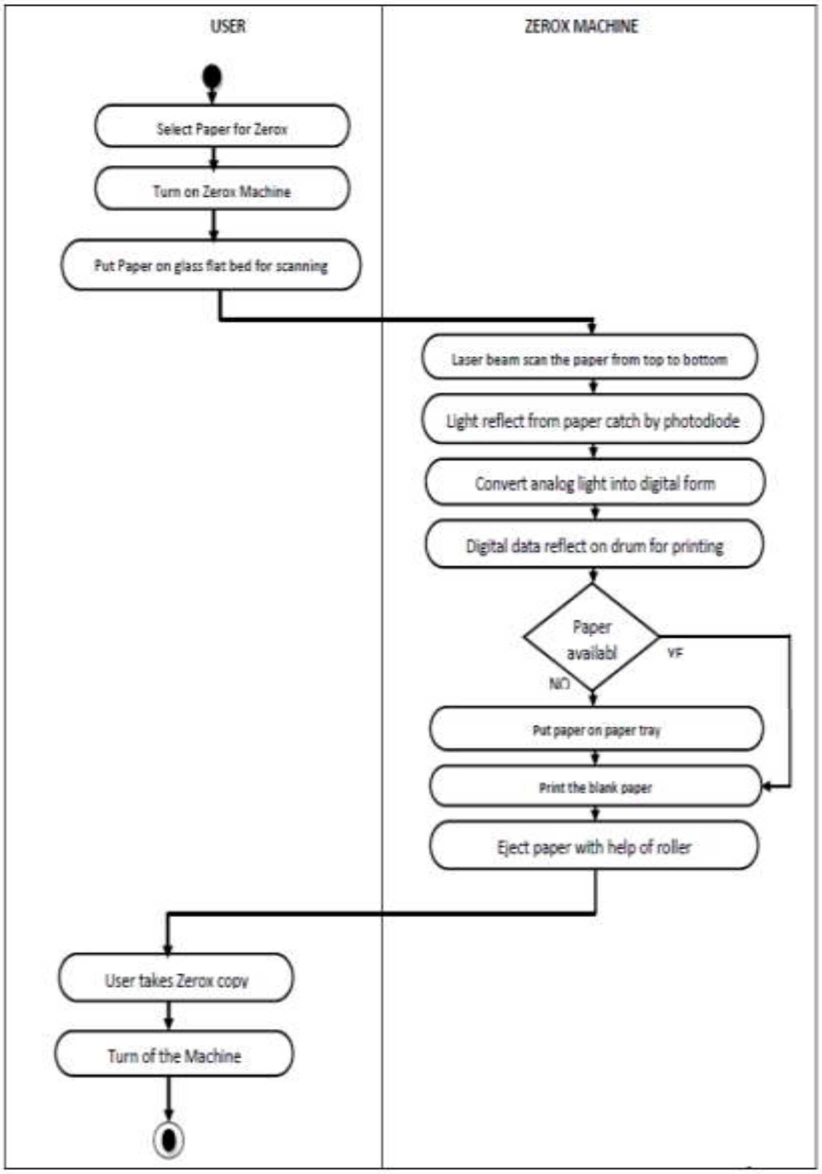
3) External actor: The hardware devices which are required as apart of application domain and must be used are referred as external actors.
   Example: in ATM system printer is an external actor.

4) Other system actor: The other system with which the system must interact referred as other system actors.
   Example: in ATM system bank network system is another system actor.

Use case diagram for printing result from MSBTE website.



Any relevant use case diagram 4M

	(b) Ans.	**Draw activity diagram for making photocopies from Xerox machine.** ***(Note: Any other activity diagram for making photocopies with correct notations shall be considered).***  USER ZEROX MACHINE Select Paper for Zerox Turn on Zerox Machine Put Paper on glass flat bed for scanning Laser beam scan the paper from top to bottom Light reflect from paper catch by photodiode Convert analog light into digital form Digital data reflect on drum for printing Paper availabl YE NO Put paper on paper tray Print the blank paper Eject paper with help of roller User takes Zerox copy Turn of the Machine	**8M** ***Any relevant diagram 8M***

(c) Explain importance of synchronous and asynchronous messages in sequence diagram. Draw sequence diagram for student admission in your institute. **8M**

(Note: Any other sequence diagram for student admission in your institute with correct notations shall be considered.)

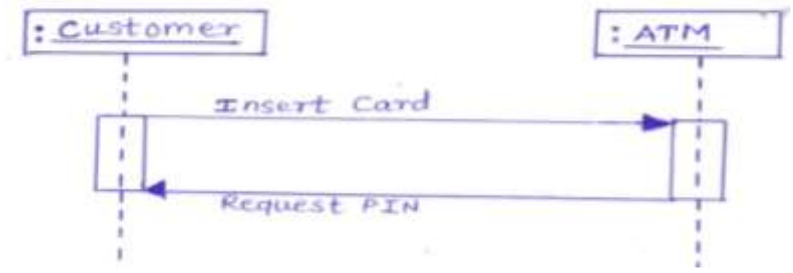
Ans. **Synchronous Messages:**

A synchronous message is used when sender sends a message to receiver and waits until the receiver processes the sent message. The receiver sends a return message to the sender and then sender continues its processing. It is important when sender needs acknowledgement of sent message from the receiver.

It is shown with a directed line from sender to receiver. The line ends at receiver with closed and filled arrow head pointing towards receiver.

Explanation of Synchronous messages 2M

Example:-



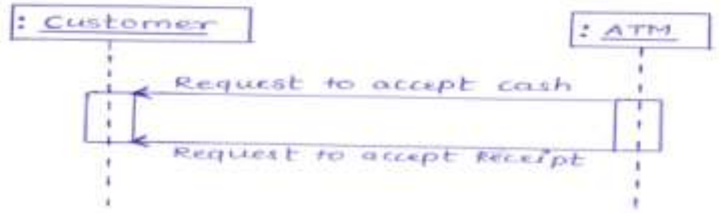
Asynchronous Messages:

An asynchronous message is used when the sender does not wait for the receiver to finish processing the message. A sender continues sending messages to other receiving objects without considering reply from receiver. It is important when sender does not need acknowledgement of sent message from the receiver.

It is shown with a directed line from sender to receiver. The line ends with open arrow head pointing towards receiver.

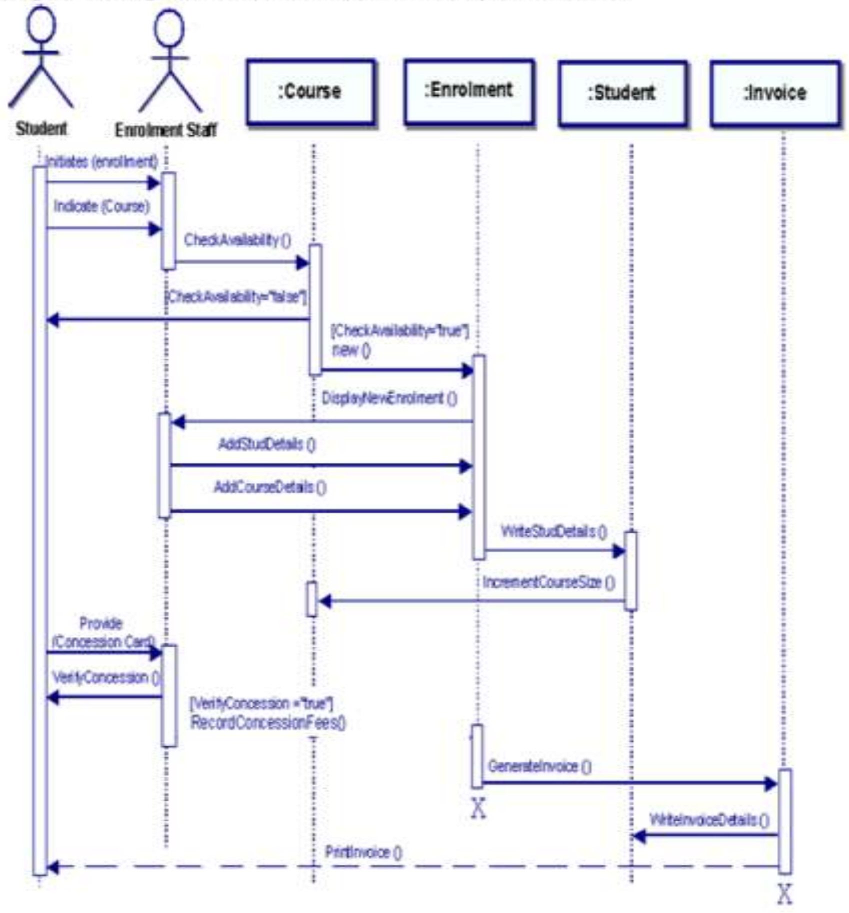
Explanation of Asynchronous message 2M

Example:-



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

Sequence diagram for student admission in your institute:



</td>
<td><i><b>Sequence diagram for student admission in your institute with correct notations 4M</b></i></td>
</tr>
<tr>
<td><b>5.</b></td>
<td>
<b>(a)</b>
Ans.</td>
<td>

Attempt any FOUR of the following:
List and classify various UML diagrams.
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
2. Behavior Diagram

</td>
<td><b>16</b>
<b>4M</b>

<i><b>List 2M</b></i>

<i><b>Classification 2M</b></i></td>
</tr>
</table>

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

(b) Ans.

Describe propagation of operation with diagram and example. **4M**

Propagation of Operation:

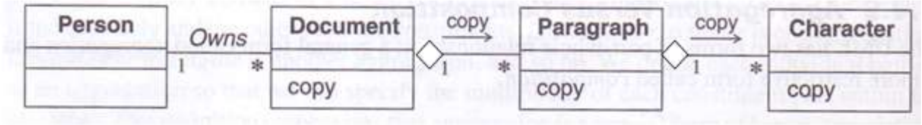
Propagation is also called as triggering. It is the automatic application of an operation to a network of objects when the operation is applied to some starting object.

Example: Moving an aggregate object moves its parts along with it. So the operation move applied to aggregate object propagates move operation to the parts.

Propagation of operation is shown with a small arrow indicating direction of propagation above association line.

Explanation 2M

Example/diagram:



Above diagram shows an example of propagation. A person owns multiple documents. Each document consists of paragraph that in turn consists of characters. The copy operation propagates from documents to 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.

Example with diagram 2M

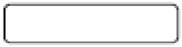
(c) Ans.

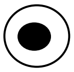
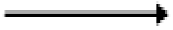
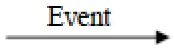
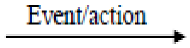
Draw and state notations for state diagram. **4M**

State chart diagram:

A state chart diagram shows flow of control from one state to another state.

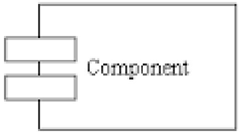
Notations:

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, performs some activity or waits for some events. It is represented with a rounded rectangle. Name of the state is written inside the rectangle.	***Any four notation 1M each***
		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.	
		4	Transition		A transition is a relationship between two states. It indicates that an object in the first state performs some action and enters in the second state when a specific event occurs. Transition is represented with a directed line.	
		5	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		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.	

	(d) Ans.	**Explain different notations used for component diagram.** **1. Component:** A component is a physical and replaceable part of the system that provides or uses set of interfaces. A component is shown as a rectangle with tabs. A component has name that distinguish it from other components. Name of the component is written as a text inside the	**4M**

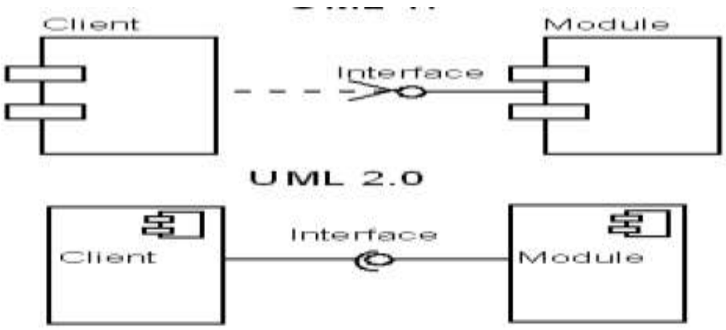
rectangle.



Any four notations 1M each

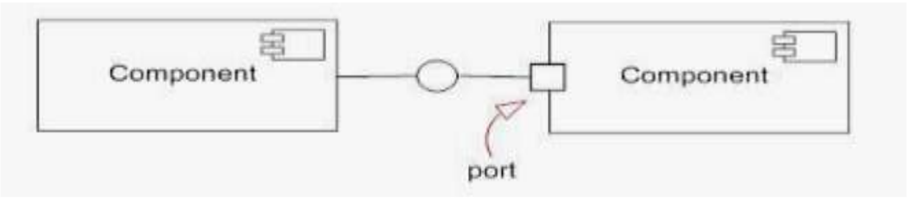
2. **Interfaces:**

A component can be connected with other components through interfaces. An interface is a collection of operations that are used to specify services of components. A component can provide an interface or can use services of a component. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface.



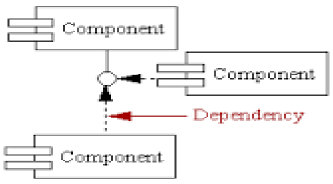
3. **Port:**

A port specifies an interaction point through which a component can communicate with its environment, other components 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.



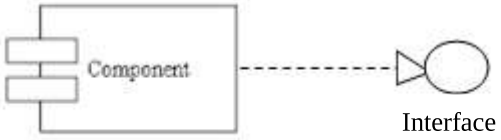
4. Dependencies

A dependency exists between two elements. Changes to the definition of one element may cause changes to the other. It is represented as dashed line with an arrow.



5. Realization

A component realizes an interface by providing service through interface. It is indicated with a dashed line and a hollow arrow head.

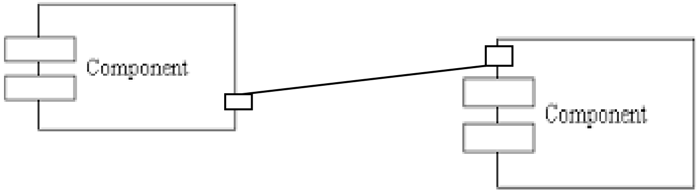


6. Connector

It is a link that specifies communication between two or more classifiers.

Connectors are of two types-

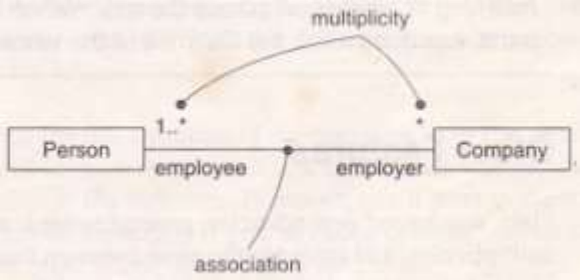
1. Delegation connector: a component realizes or uses an interface. A component can have internal parts. A part of a component can realize or use an interface. To show a connection among internal parts of a component and interface delegation connector is used.
2. Assembly connector: it is a connector between two or more parts or ports on parts that defines services provided by parts for other parts.

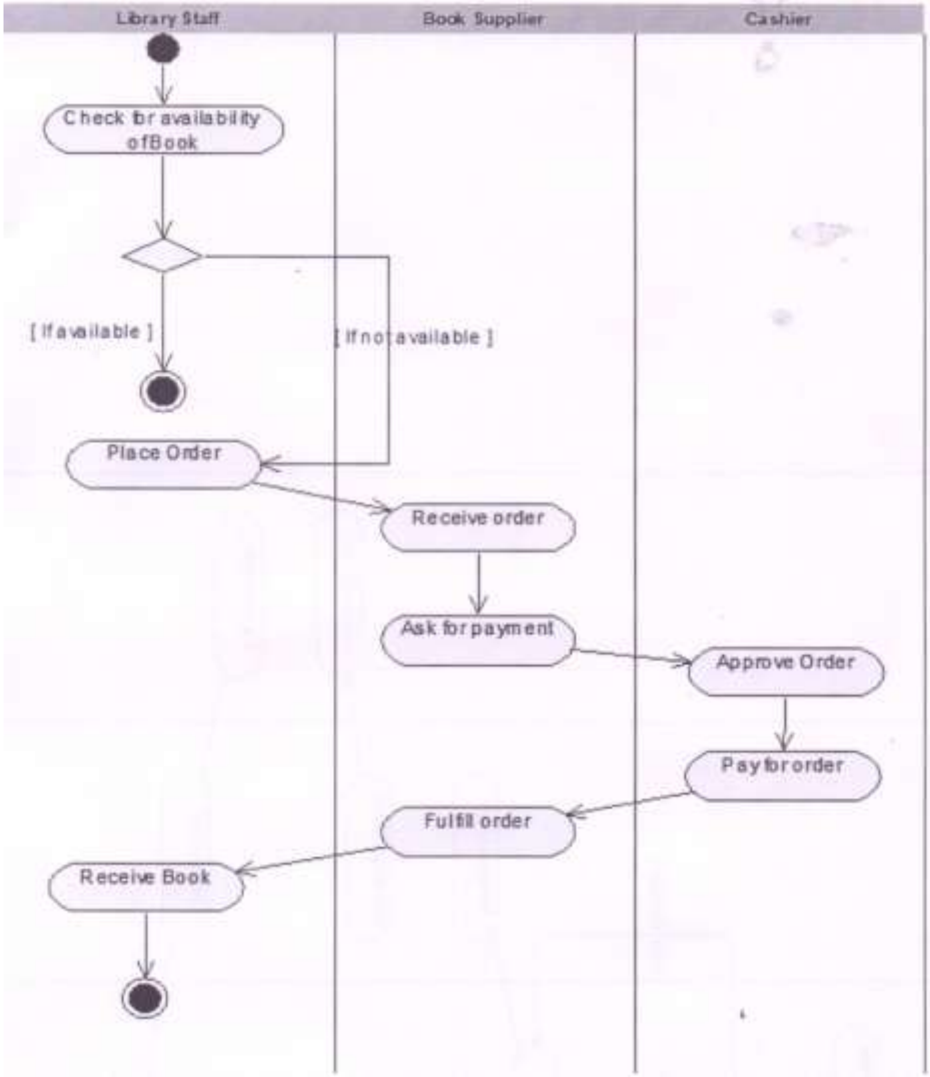


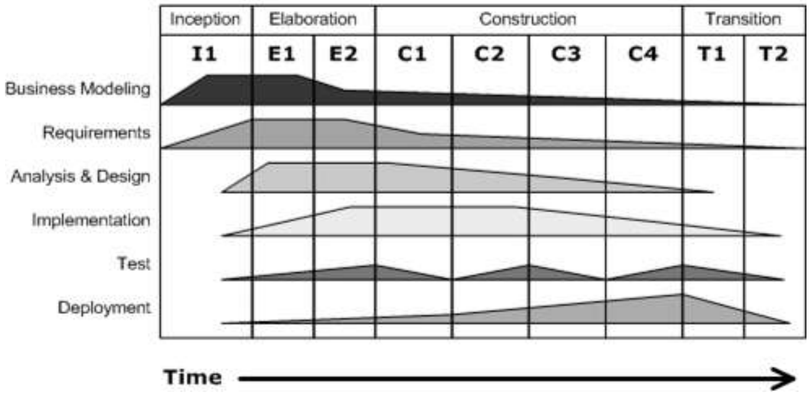
(e) Ans. **Explain multiplicity with example.** **4M**

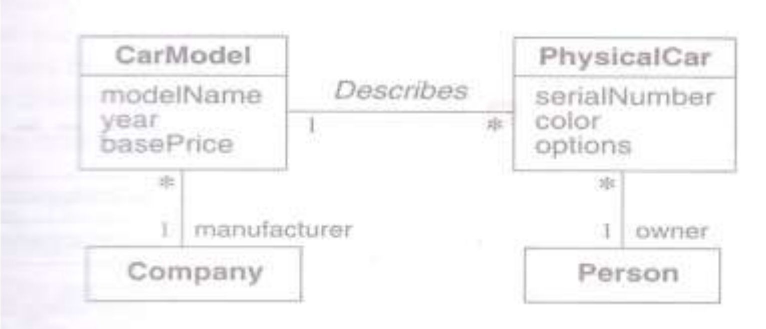
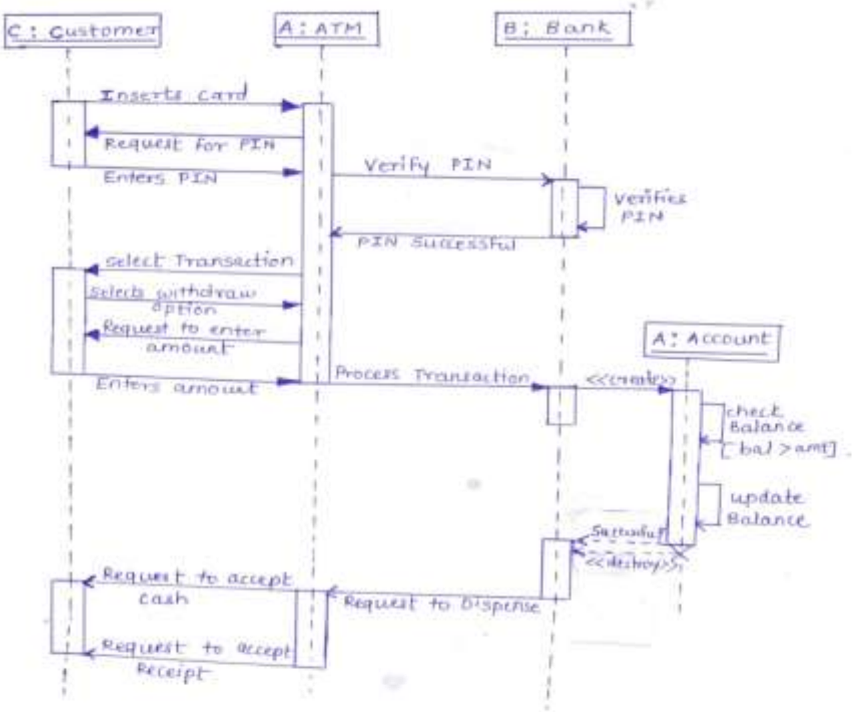
<u>Multiplicity</u>

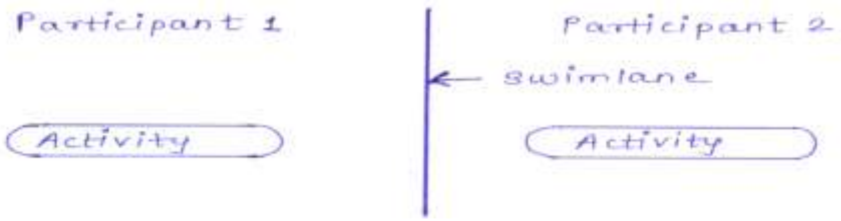
Multiplicity specifies the number of instances of one class that may

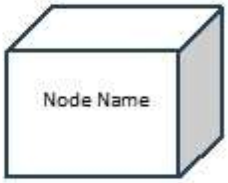
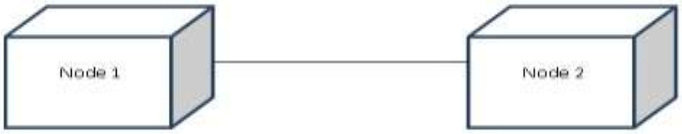
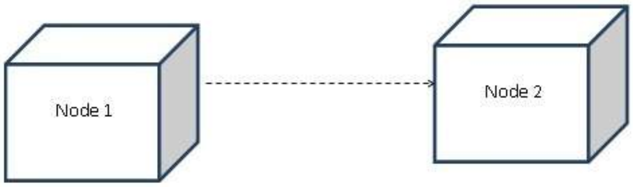
		relate to a single instance of an associated class. The UML specifies multiplicity as follows: **Notations:** 1. ”1” exactly one 2. “1…*” One or more 3. “3-5” three to five 4. 0..1 zero to one 5. “2,4,18” two, four or eighteen 6. Symbol * denotes “many”. **Example :**  multiplicity 1..* * Person employee employer Company association In the above example one or more persons may work in many companies.	***Explanation 2M*** ***Example 2M***
	(f) Ans.	**Draw an activity diagram to purchase books from supplier in library.** ***(Note: Any other activity diagram showing purchase book from supplier in library with correct notations shall be considered).*** **Activity diagram to purchase book from supplier in library:**	**4M**

<table>
<tr><td></td><td></td><td>
</td><td><i>Activity diagram for purchase book from supplier in library with correct notations
4M</i></td></tr>
<tr><td><b>6.</b></td><td><b>(a)</b>
Ans.</td><td><b>Attempt any FOUR of the following:</b>
<b>Describe unified software development life cycle.</b></td><td><b>16</b>
<b>4M</b></td></tr>
</table>

		 Inception Elaboration Construction Transition I1 E1 E2 C1 C2 C3 C4 T1 T2 Business Modeling Requirements Analysis & Design Implementation Test Deployment Time **Fig: Software development lifecycle (SDLC)** **There are four phases of SDLC:** **1) Inception:** It is first phase of SDLC where the idea for the development of a system is finalized. **2) Elaboration:** It is second phase of process where the product vision and its architecture is defined. In this phase the systems requirements are collected with respect to vision statement, evaluation criteria, functional and non-functional behavior and testing of the processes. **3) Construction:** It is a third phase of the SDLC where the project is implemented to handover to the user community the systems requirements and its evaluation criteria are constantly reexamined against the project needs in this phase the resources are allocated to reduce risk in the project after implementation. **4) Transition:** It is a fourth phase of process where the software is handed over to user community in this phase the software is continuously in improvement the bugs are detected and solved and if required new released of the project are developed.	***Any four stages 1M each***
	(b) Ans.	**Describe metadata. Explain 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 s parts, catalogues, blue-prints and dictionaries.	**4M** ***Description of metadata 2M***

		 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 and manufacturer a physical car has a serial no,, color, options and owner. 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.	***Example 2M***
	(c) Ans.	**Draw sequence diagram for ATM session of withdraw money.** ***(Note: Any other sequence diagram showing ATM -money withdrawal with correct notations shall be considered).*** **ATM -money withdrawal class diagram**  C : Customer A : ATM B ; Bank Inserts card Request for PIN Enters PIN Verify PIN Verifies PIN PIN Sucessful select Transaction Selects withdraw option Request to enter amount Enters amount Process Transaction A : Account <<create>> check Balance [bal > amt] update Balance Successful <<destroy>> Request to accept cash Request to Dispense Request to accept Receipt	**4M** ***ATM session of withdra w money 4M***

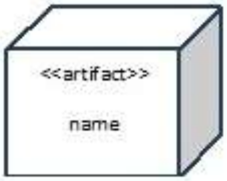
	(d) Ans.	**Explain swim lane in activity diagram with one example.** **Swim lane:** 1. In activity diagram, activity states are partitioned into groups. Each group represents the entity responsible for those activities. Each group is called as swim lane because visually each group is divided by a line from its neighbor. 2. A swim lane specifies a locus of activities. 3. Each swim lane has a unique name within its diagram. It represents some real world entity. 4. Each swim lane represents a high level responsibility for part of the overall activity of an activity diagram and each swim lane may eventually be implemented by one or more classes. 5. In an activity diagram partitioned into swim lane, every activity belongs to exactly ones swim lane, but transitions may cross lanes. **Example:**  Participant 1 Participant 2 swimlane Activity Activity In above diagram two groups are shown were first group is for participannt1 and second group is for participant2. A swim lanes separate the activities of each participants in its own group.	**4M** ***Explanation 2M*** ***Diagram /example 2M***

	(e) Ans.	**Describe notations used for deployment diagram.** **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. Association is used to show relationship between 2 nodes. It is shown with a solid line.  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.  	**4M** ***Any four notations 1M each***

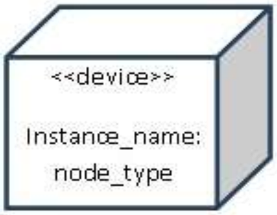
4. Artifact: Artifacts are physical file that execute or are used by software of the system.

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



5. 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.



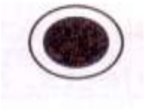
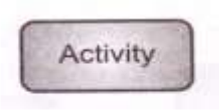
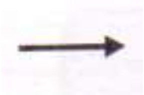
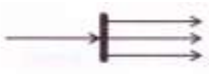
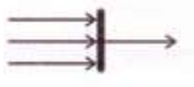
(f) Ans.

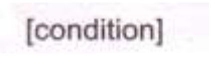
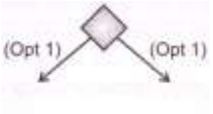
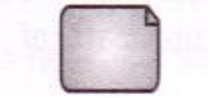
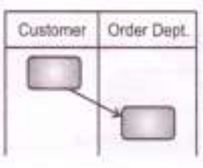
Draw and state notations used to draw activity diagram. **4M**

Notations of Activity Diagram:

Name	Symbol	Description
1.Start/Initial Nodes	●	It shows the starting point of the activity diagram. An initial or start node is depicted by a filled circle with black color.

Any four notations 1M each

2. Final/Exit Node		It shows the exit point of the activity diagram. An activity diagram can have zero or more activity final nodes. Final node is rendered as two concentric circles with filled inner circle.
4. Activity		Activity is parameterized behavior represented as coordinated flow of actions. An activity is the process being modeled, such as washing a car. An activity is a set if actions.
5. Transition /Edge/Path		The flow of the activity is shown using arrowed lines called edges or paths. The arrowhead on an activity edge shows the direction of flow from one action to the next. A line going into a node is called an incoming edge, and a line exiting a node is called an outgoing edge.
6. Fork Node		It is used to show the parallel of concurrent actions. Steps that occur at the same time are said to occur concurrently or in parallel. Fork has single incoming flow and multiple outing flows.
7. Join Node		The join means that all incoming actions must finish before the flow can proceed past the join. Join has multiple incoming flows

					and single outing flow.		
			8. Condition		Condition text is placed next to a decision marker to let we know under what condition an activity flow should split off in that direction.		
			9. Decision/ Branch		A marker shaped like a diamond is the standard symbol for a decision. There are always at least two paths coming out of a decision and the condition text lets we know which options are mutually exclusive.		
			10. Note		A note is used to render comments, constraints etc. of an UML element.		
			11. Swimlane		We use partitions to show which participant is responsible for which actions. Partitions divide the diagram into columns or rows (depending on the orientation of your activity diagram) and contain actions that are carried out by a responsible group. The columns or rows are sometimes referred to as swimlanes.		