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.1) a) Attempt any THREE of the following: 12

i) Describe any two features of OO (Object Oriented).
(2 marks for each feature, any 2 features can be considered)

Ans: 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 behavior in a single entity.

3) Sharing

OO techniques promote sharing at several different levels. The sharing of a code using inheritance is one of the main advantages of Object Oriented technique. It not only allows information to be shared within an application, but also offers the prospects of reusing designs and code on future projects.

4) Combining Data and Behavior

The burden of calling code for data execution and operations separately can be minimized by combining data properties and behavioral properties of an entity together. In object oriented program data structure and procedure is defined in single class definition.

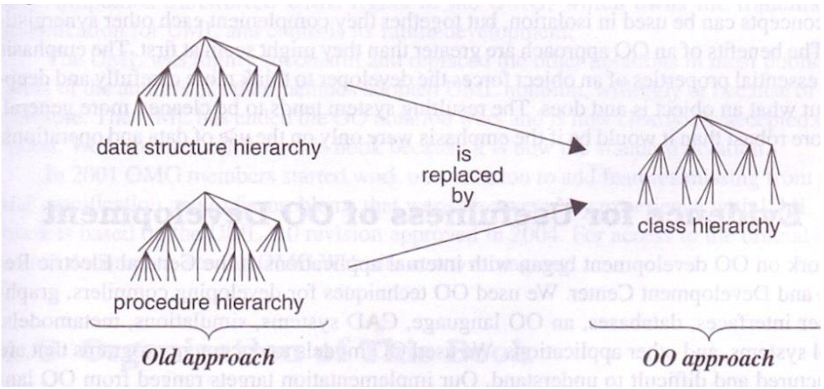


Fig: Combining data and behavior

ii) State any four principles of modelling.
(Each principle 1 mark)

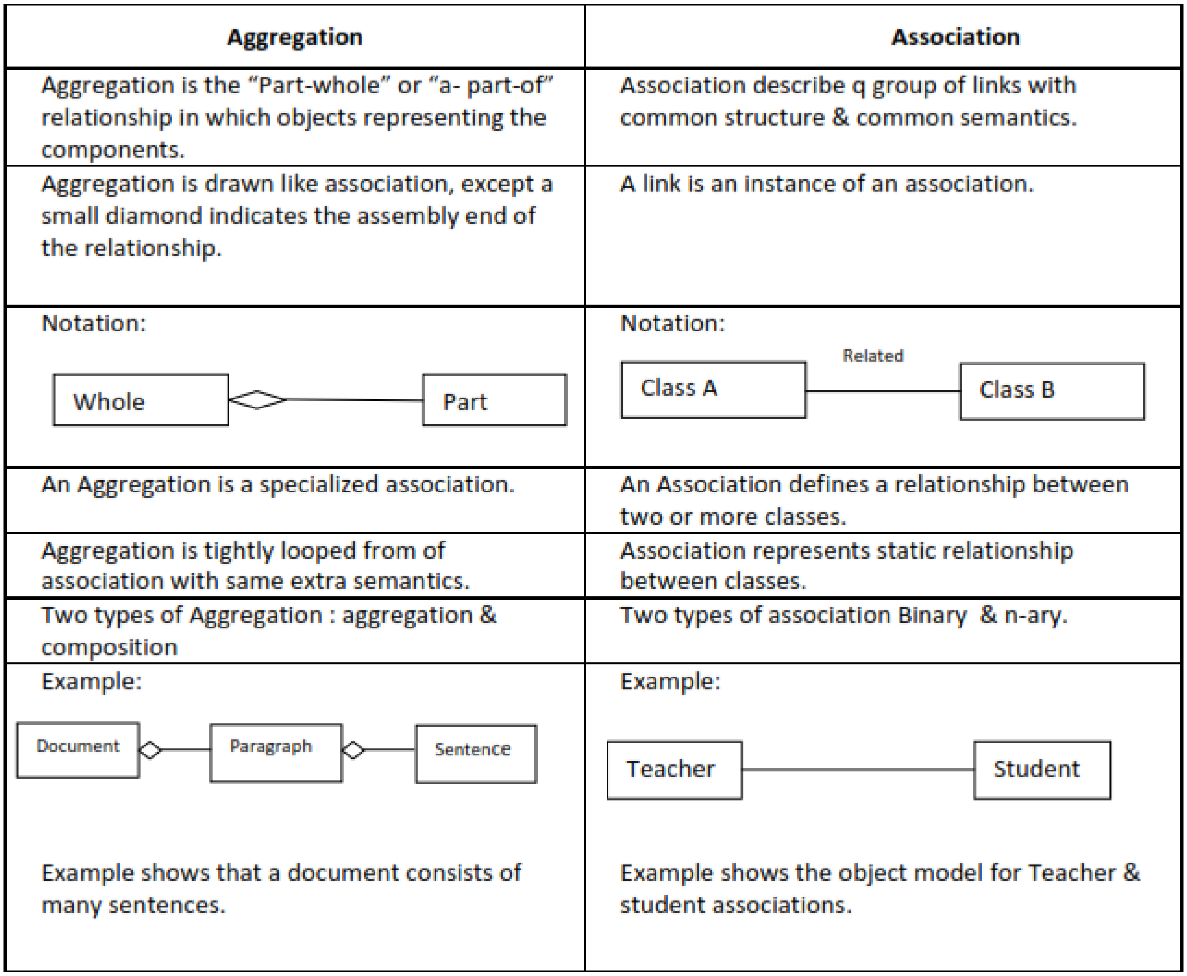
Ans: Modelling principles are as follows:

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".**
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.

iii) Differentiate between Aggregation and Association.

(Any 4 relevant points, 1mark to each point)

Ans:



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: Related Class A 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.

iv) Define following terms with notations:

1) Object

2) Class

3) Qualified association

4) Ordered association

(Definition and Notation1 Mark each)

Ans: 1) Object

-Objects are separate, distinguishable entities.

- They are basic run time entities.

- They are also referred as instances

-Objects can be defined as a concept, abstraction or thing with well defined solution forthe specific problem.

- Objects serve two purposes:
  i) They prompt understanding of real world.
  ii) Provides a practical basis for computer implementation.

Example: mango, swati

Notation

objectName:ClassName

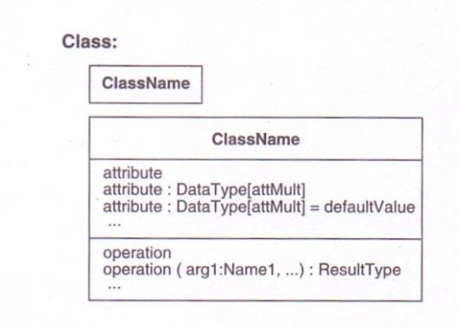
2) Class

-A Class is a group of objects with similar properties (attributes), common behavior(operation), common relationship to other objects and common semantics.

- A class is a collection of objects of similar types.
- Once class has been defined we can create any no. of objects belonging to the same class.

Example: fruit, student

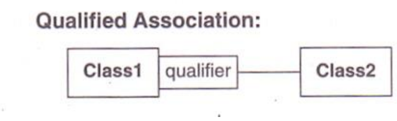
Notation


Class:
ClassName
ClassName
attribute
attribute : DataType[attMult]
attribute : DataType[attMult] = defaultValue
...
operation
operation (arg1:Name1, ...) : ResultType
...

3) 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.

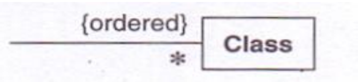
Notation



4) 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.

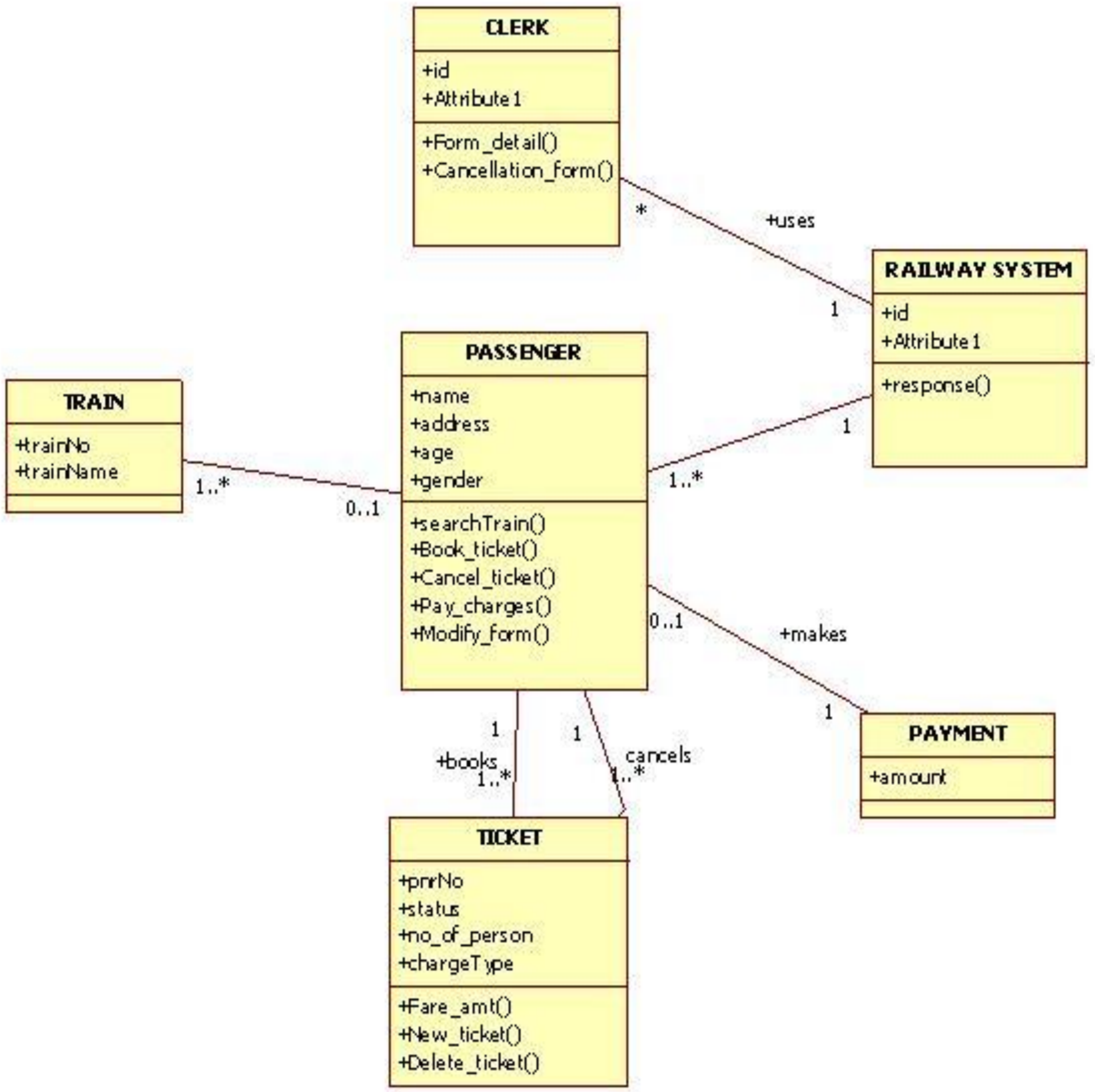
Notation:



Q.1) b) Attempt any ONE of the following: **6**

i) Draw neat class diagram for Railway Reservation System.
{ consider: Railway_details, Passenger_details, booking, Admin, route, etc }
(Diagram with correct notations -6M)
Note: any other class diagram showing railway reservation system shall be considered.

Ans:



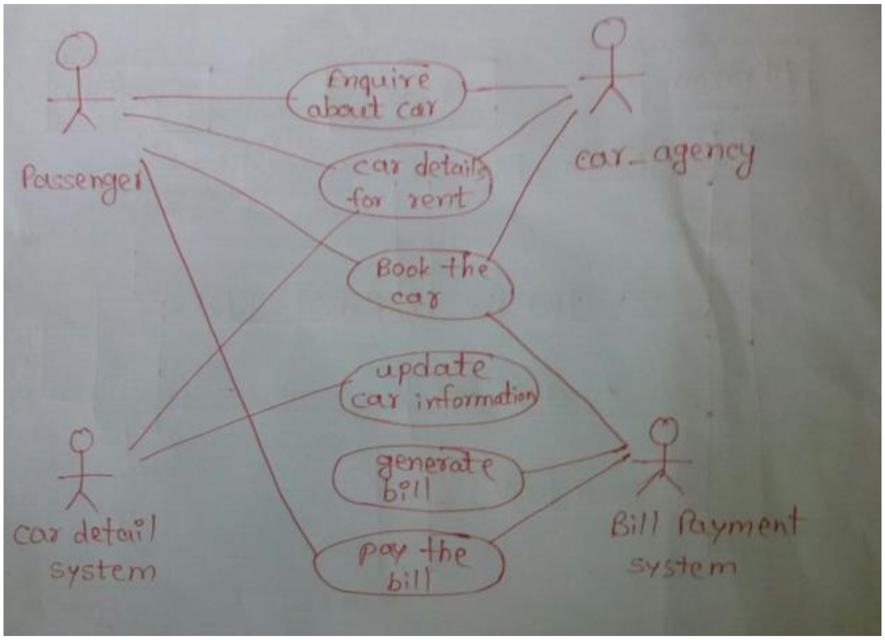
ii) State any two types of actors used in use case diagram.
Draw a neat use case diagram for car rental system
{ considercar_details, car_agency, passenger, Bill_payment }
(Two actors-Each actor 1M, Use case diagram with correct notation -4 M)
Note: any other use case diagram showing car rental system shall be considered.

Ans: Two types of actors used in use case diagram are:

1) Primary or Principle Actor: The primary actor of a use case is the stakeholder that calls on the system to deliver one of its services. It has a goal with respect to the system – one that can be satisfied by its operation. The primary actor is often, but not always, the actor who triggers the use case.

2) Supporting or Secondary Actor: A supporting actor in a use case in an external actor that provides a service to the system under design. It might be a high-speed printer, a web service, or humans that have to do some research and get back to us.

Use case diagram for car rental system:



Q.2) Attempt any TWO of the following: 16

a) Explain following types of relationships with notation:
(i) Realization
(ii) Generalization

(iii) Dependency

(iv) Association

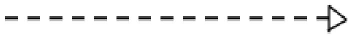
(For each relationship- Explanation 1M, Notation 1M)

Ans: i) Realization

It is a semantic relationship between classifiers, wherein one classifier classifies a contract that another classifier guarantees to carry out. Realization relationships encounters in two places:

1. Between interfaces and the classes or components that realize them
2. Between use cases and the collaborations that realize them.

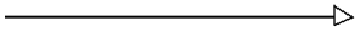
Graphically, a realization relationship is rendered as a cross between a generalization and a dependency relationship, as in figure below:



ii) Generalization

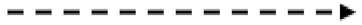
Generalization: it is a specialization / generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element (the parent).

In this way, the child shares the structure and the behavior of the parent. Graphically, a generalization relationship is rendered as a solid line with ahollow arrowhead pointing to the parent, as in figure below:



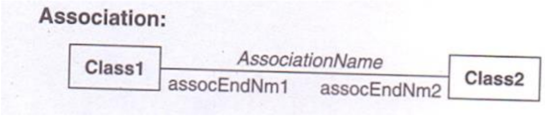
iii) Dependency

It is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing. Graphically, a dependency is rendered as a dashed line, possibly directed, and occasionally including a label, as in figure below:



iv) Association

It is a structural relationship that describes a set of links, a link being a connection among objects. Aggregation is a special kind of association, representing astructural relationship between a whole and its parts. Graphically, an association isrendered as a solid line, possibly directed, occasionally including a label, and often containing other adornments, such as multiplicity and role names, as in figure below:



b) State and explain notations used to draw class diagram. Draw a neat class diagram for ATM – money withdrawal.
(Any 4 class diagram notations- each 1M, ATM -money withdrawal class diagram with correct notation 4M)
Note: any other class diagram showing ATM -money withdrawal shall be considered.

Ans:

Class diagrams describe the static structure of a system or how it is structured rather than how it behaves.

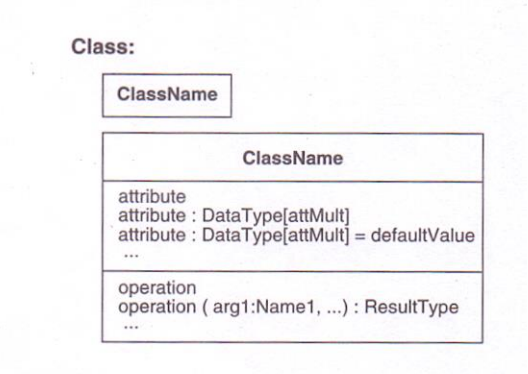
These diagrams contain the following element with their symbol.

1) Class– It represents entity with common characteristics or features. These features can include attributes, operations and associations.

The symbol used to denote class is rectangle with three compartments.

First contains the name of the class, second contains the attributes of the class and third contains the operations of that class.

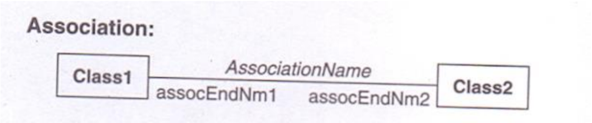
Notation:


Class:
ClassName
ClassName
attribute
attribute : DataType[attMult]
attribute : DataType[attMult] = defaultValue
...
operation
operation (arg1:Name1, ...) : ResultType
...

2) Association – It represents relationships that relate to two or more other classes where the relationships have common characteristics.

The symbol used to denote association is solid line.

Notation:

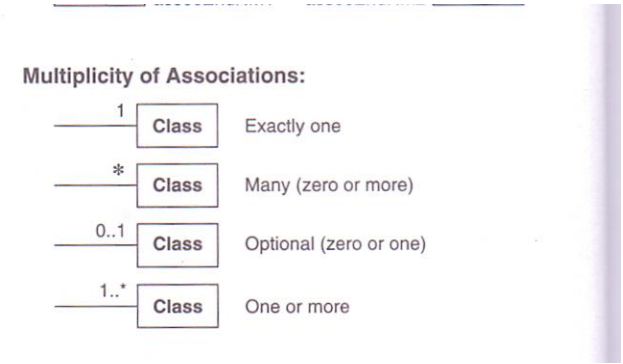


3) Multiplicity -Multiplicity in an association specifies how many objects participate in a relationship. Multiplicity decides the number of related objects. Multiplicity is generally explained as "one" or "many," but in general it is a subset of the non-negative integers.

Indicator and Meaning

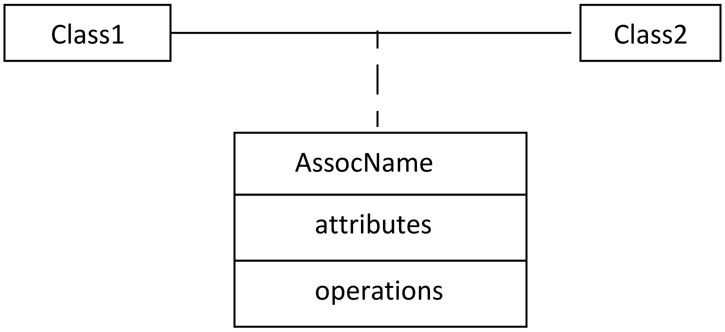
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only n (where n > 1)
0..n	Zero to n (where n > 1)
1..n	One to n where n > 1)

Notation:



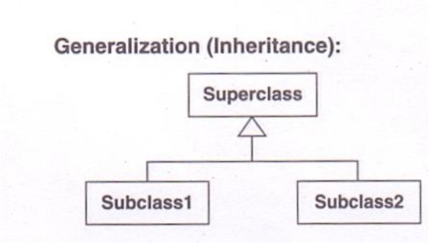
4) Association Class: An association class is an attribute of an association. It is also a class.

Notation:



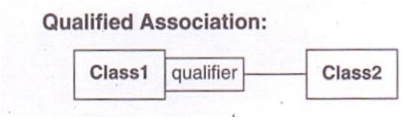
5) Generalization: It is a relationship between a class (superclass) and its derived classes (subclasses).

Notation:

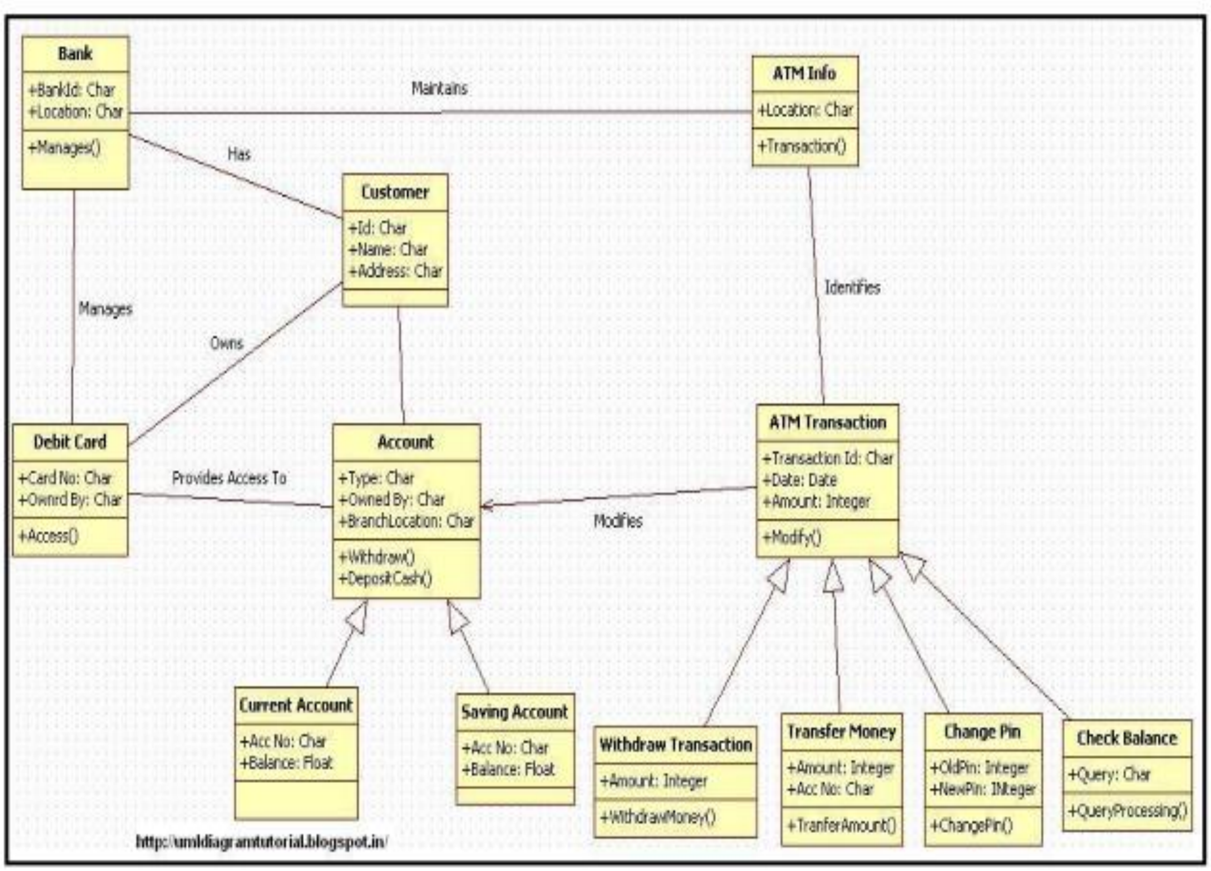


6) **Qualified association:** It uses the special attribute qualifier which reduces the effective multiplicity of an association.

Notation:



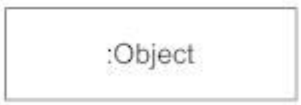
ATM -money withdrawal class diagram



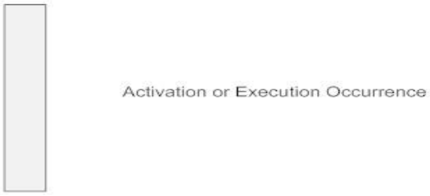
c) **State and explain notations used to draw sequence diagram.**
Draw a neat diagram for library management system – Book Issue.
(Any 4 sequence diagram notations: each 1M, library management system-book issue sequence diagram with correct notations 4M)
Note: Any other sequence diagram showing library management system-book issue shall be considered.

Ans: Notations:

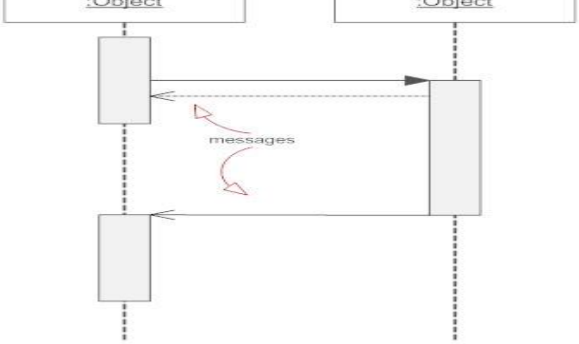
1) Object: Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.



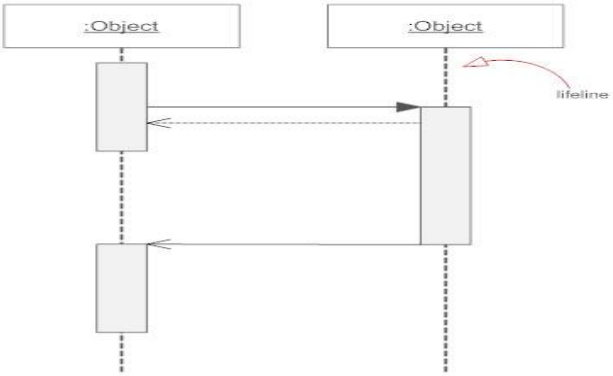
2) Activation or Execution Occurrence: Activation boxes represent the time an object needs to complete a task. When an object is busy executing a process or waiting for a reply message, use a thin gray rectangle placed vertically on its lifeline.



3) Messages: Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks.



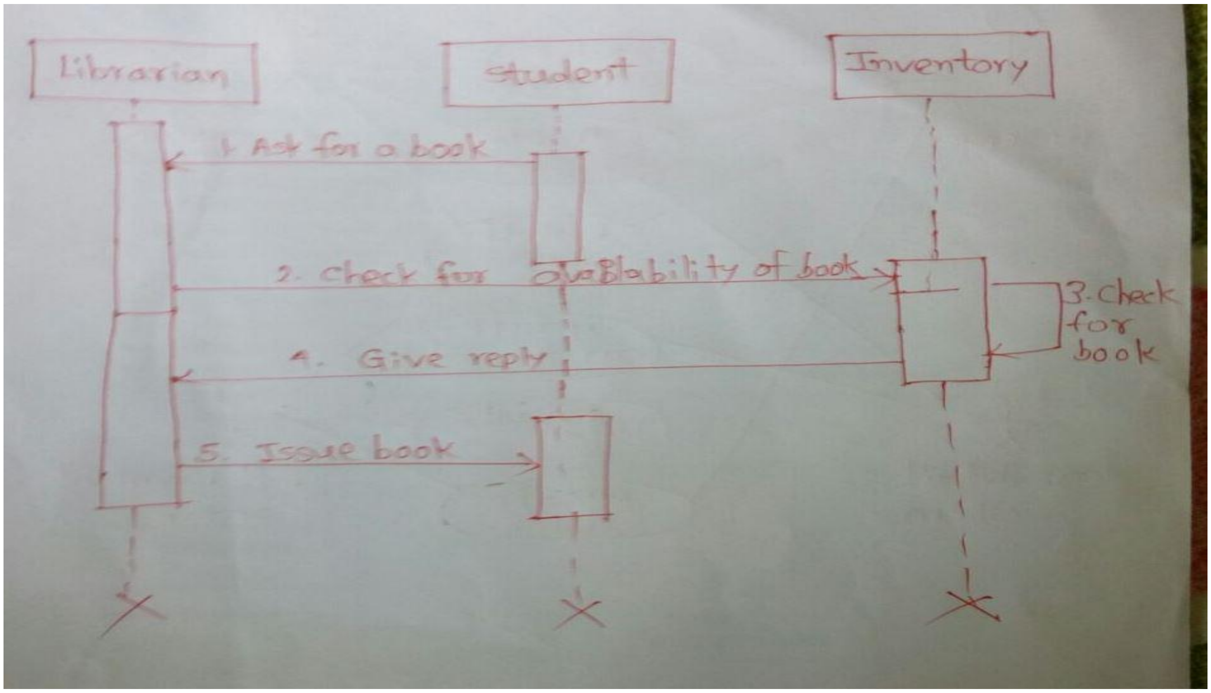
4) **Lifelines:** Lifelines are vertical dashed lines that indicate the object's presence over time.



5) Destroying Objects: Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X. This object is removed from memory. When that object's lifeline ends, you can place an X at the end of its lifeline to denote a destruction occurrence.

6) Loops: A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].

Library management system- Book Issue sequence diagram:



Q.3) Attempt any FOUR of the following: 16

a) List three models and explain functional model in detail.
(List: 2M, Explanation of Interaction Model/ Functional Model: 2M)
(Note: Explanation of Functional (E scheme) or Interaction Model (G scheme) shall be considered).

Ans: (According to E scheme syllabus)

Three Models:

1. Object Model
2. Dynamic Model
3. Functional Model

Functional model:

1. It describes the data value transformations within a system.
2. The functional model contains data flow diagram. It represents computations.
3. A data flow diagram is a graph whose nodes are processes and arcs are data flows.

Example of Data Flow Diagram

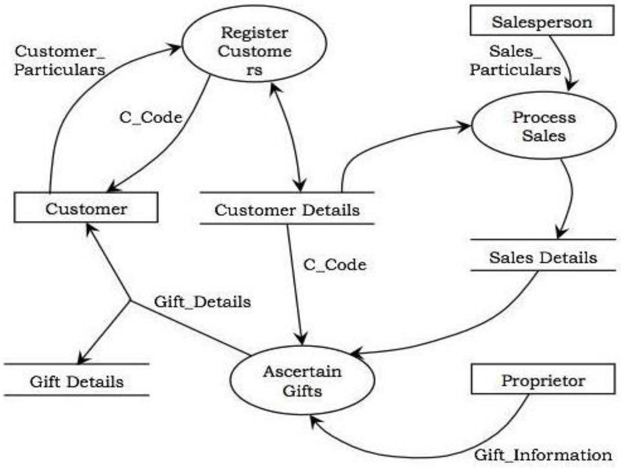


Fig: DFD for Data Warehouse

OR

(According to G scheme syllabus)

Three Models:

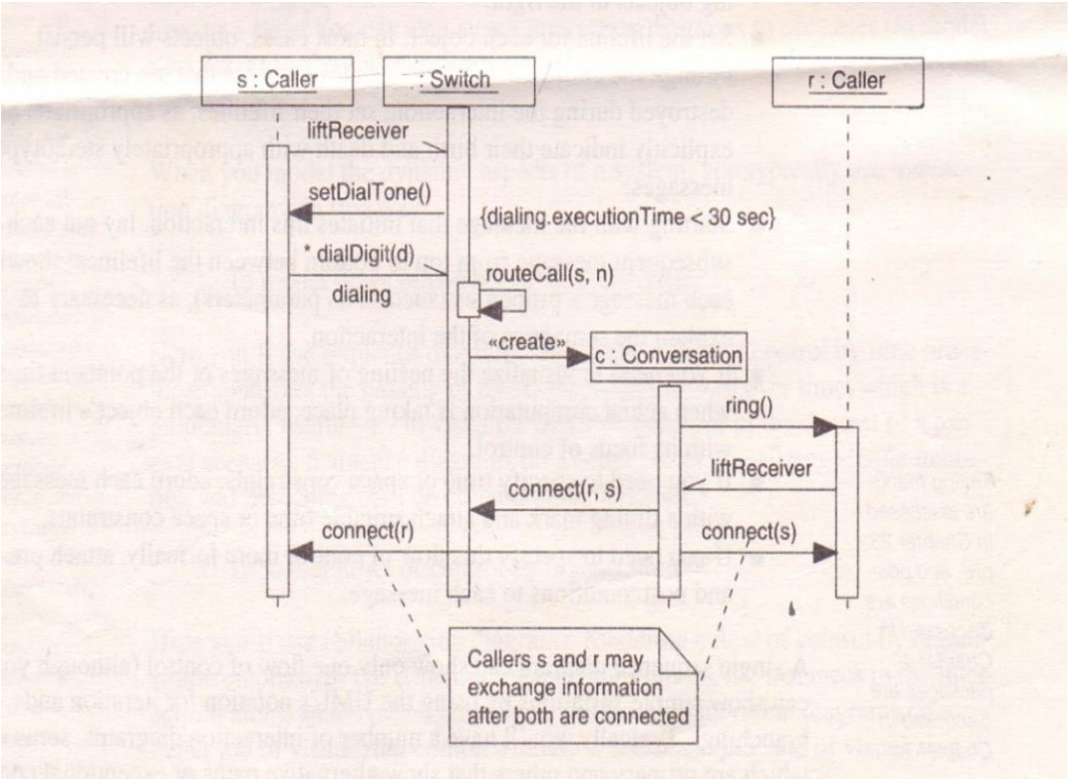
1. Class Model
2. State Model
3. Interaction Model

Interaction Model

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

4. Sequence diagrams show the objects that interact and the time sequence of their interactions.
5. Activity diagrams show the flow of control among the processing steps of a computation.

Example of Interaction model:

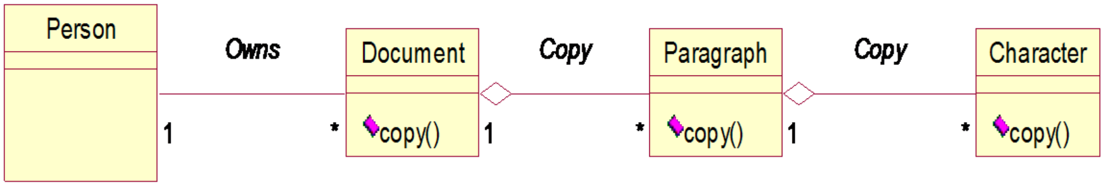


b) Describe propagation of operation with diagram and example.
(Explanation 2M, Example with diagram 2M)

Ans: Propagation of Operation:

1. Propagation is also called as triggering.
2. It is the automatic application of an operation to a network of objects when the operation is applied to some starting object.
3. As moving aggregate moves its parts, the move operation propagates to the parts.
4. Propagation of operations to parts is good indicator of aggregation.

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. Copying a document copies the owner link but does not spawn a copy of the person who is owner.

c) Describe Join and Fork operations in activity diagram with example.
(Explanation with example of Join 2M, Explanation with example of Fork 2M)
Note: Any other example showing join and fork operations in activity diagram shall be considered

Ans: Join in Activity diagram:

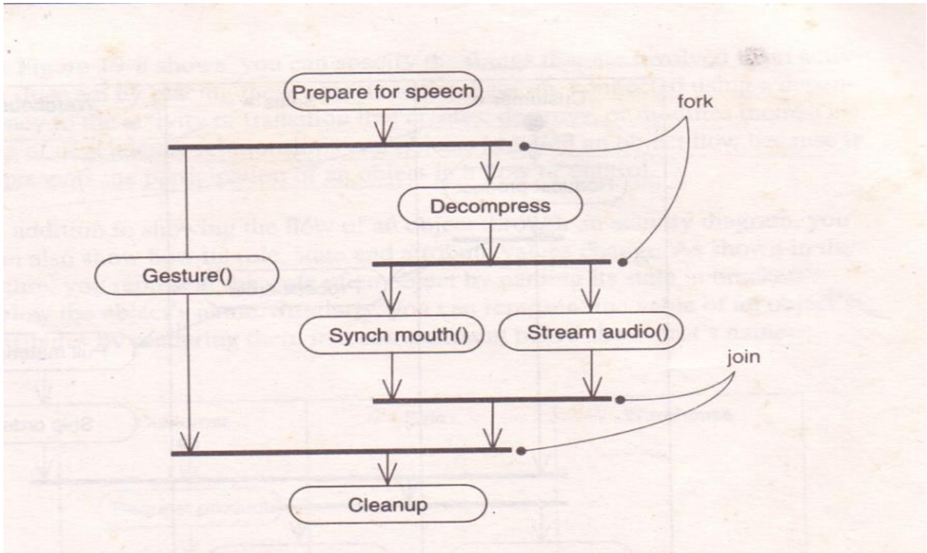
1. A join in Activity diagram represents the synchronization of two or more concurrent flows of control.
2. A join may have two or more incoming transitions and one outgoing transition.
3. Above the join, activities associated with each of these paths continue in parallel.
4. At the join, the concurrent flows synchronize, means each waits until all incoming flows have reached at the join, at which point one flow of control continues on below the join.

Fork in Activity diagram:

1. A fork in activity diagram represents the splitting of a single flow of control into two or more concurrent flows of control.

2. A fork may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control.
3. Below the fork, the activities associated with each of these paths continue in parallel.

Example:



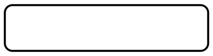
In the above example there are two forks and two joins. Joins and forks should balance, meaning that the number of flows that leave a fork should match the number of flows that enter its corresponding join.

d) Draw and state notations for the state chart diagram.
(Any four notation: each 1M)

Ans: 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.

2	Initial state	●	It indicates the default starting place of t5he state chart 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	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.

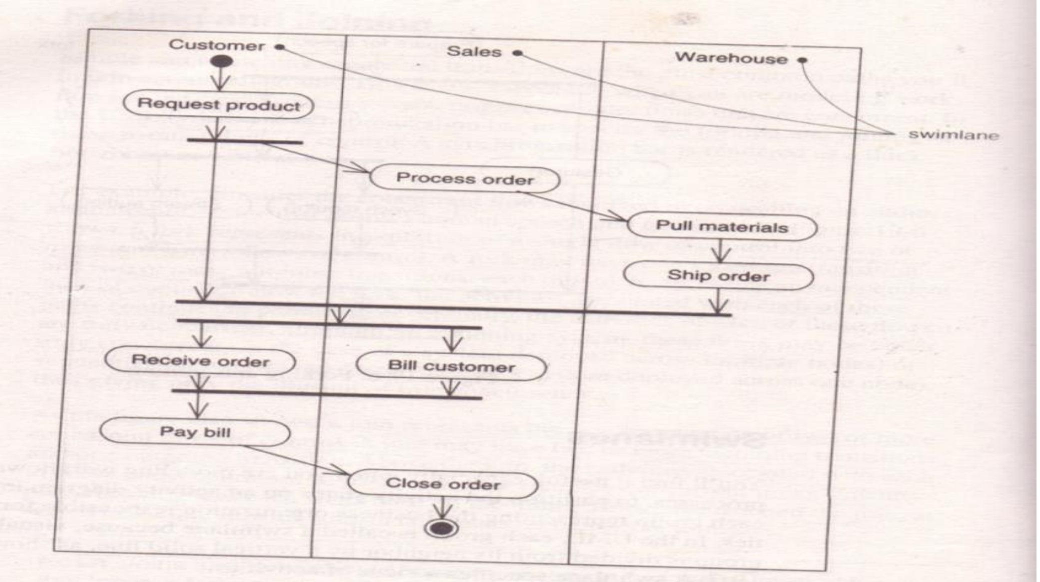
e) Explain the swim lane activity diagram with one example.
(Explanation 2M, Diagram/example 2M)

Ans: 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 on swim lane, but transitions may cross lanes.

Example:



In above diagram we can observe three groups (entities): Customer, sales and Warehouse Customer performs request product, receive order and pay bill activity.

Q.4) a) Attempt any THREE of the following: 12

i) Explain how UML diagrams are classified.
(Explanation of any 4 diagrams, each 1M)

Ans:

UML diagram is the graphical presentation of a set of elements. It may contain any combination of things and relationships.

UML includes nine diagrams:

1. Class diagram
2. Object diagram
3. Use case diagram
4. Sequence diagram
5. Collaboration diagram
6. Statechart diagram
7. Activity diagram

8. Component diagram
9. Deployment diagram

1. **Class diagram**
   It shows a set of classes, interfaces and collaborations and their relationships. Class diagram addresses the static design view of a system.
2. **Object diagram**
   An object diagram shows a set of objects and their relationships. Object diagram addresses the static design view of a system.
3. **Use case diagram**
   A use case diagram shows a set of use cases and actors and their relationships. Use case diagram addresses the static design view of a system.
4. **Sequence & Collaboration diagram**
   These are the kinds of interaction diagrams. An interaction diagram shows an interaction, consisting of a set of objects and their relationships including messages. Interaction diagram addresses the dynamic view of a system.
5. **Statechart diagram**
   A statecahrt diagram shows a state machine, consisting of states, transitions, events and activities. Statechart diagram addresses the dynamic view of a system.
6. **Activity diagram**
   An activity diagram is a special kind of a statechart diagram that shows the flow from activity to activity within the system. Activity diagram addresses the dynamic view of a system.
7. **Component diagram**
   A component diagram shows the organization and dependencies among a set of components. Component diagram addresses the static implementation view of a system.
8. **Deployment diagram**
   A deployment diagram shows the configuration of run-time processing nodes and the components that live on them. Deployment diagram addresses the static deployment view of architecture.

ii) Describe association classes and inheritance with neat diagrams.
(Explanation of Association Class with diagram: 2M, Explanation of Inheritance with diagram: 2M)
(Note: Any other suitable example shall be considered)

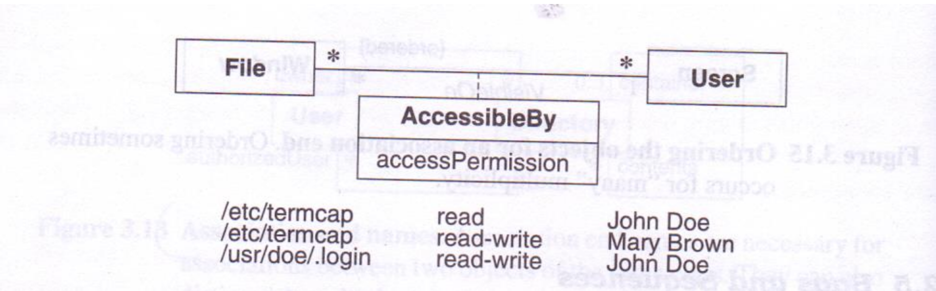
Ans: Association Class:

1. An association class is an attribute of an association that is also a class.
2. Like the links of an association, the instances of an association class derive identity from instances of the constitute classes.
3. An association class can have attributes and operations and participate in associations.

4. The UML notation for an association class is a class box attached to the association by dashed line.

Example:

5.
6.
7.
8.

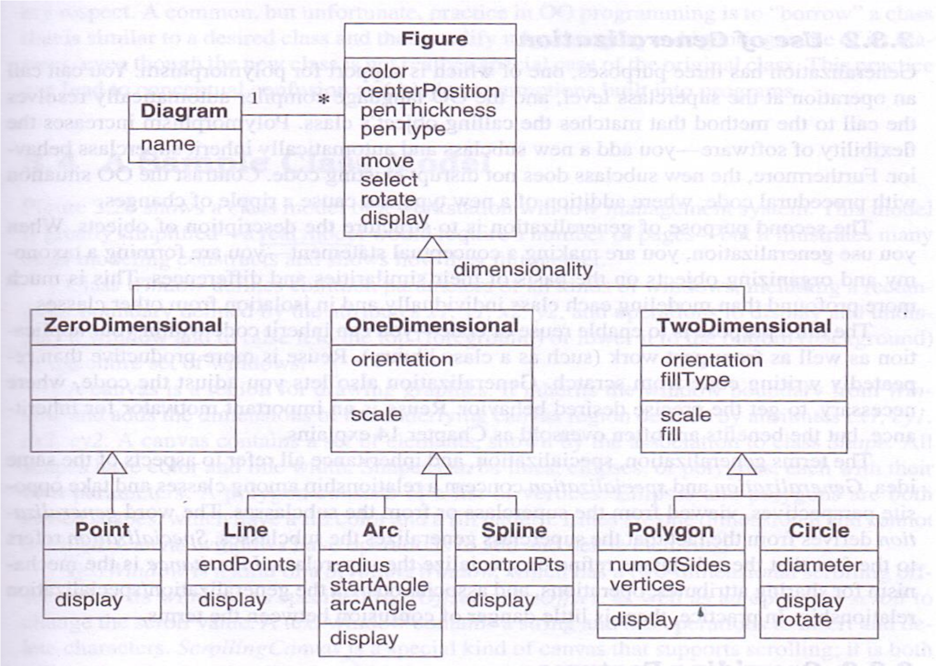


In the above example Accessible By is an association class and accessPermission is an attribute of an AccessibleBy class. The sample data at the bottom of the example shows the value for each link.

Inheritance:

1. Inheritance is a relationship between a class and one or more derived classes of it.
2. The class being derived is called as superclass and its derived classes are called as subclasses.
3. Inheritance is also called as is-a relationship.
4. In inheritance each instance of a subclass is an instance of the superclass as well.

Example

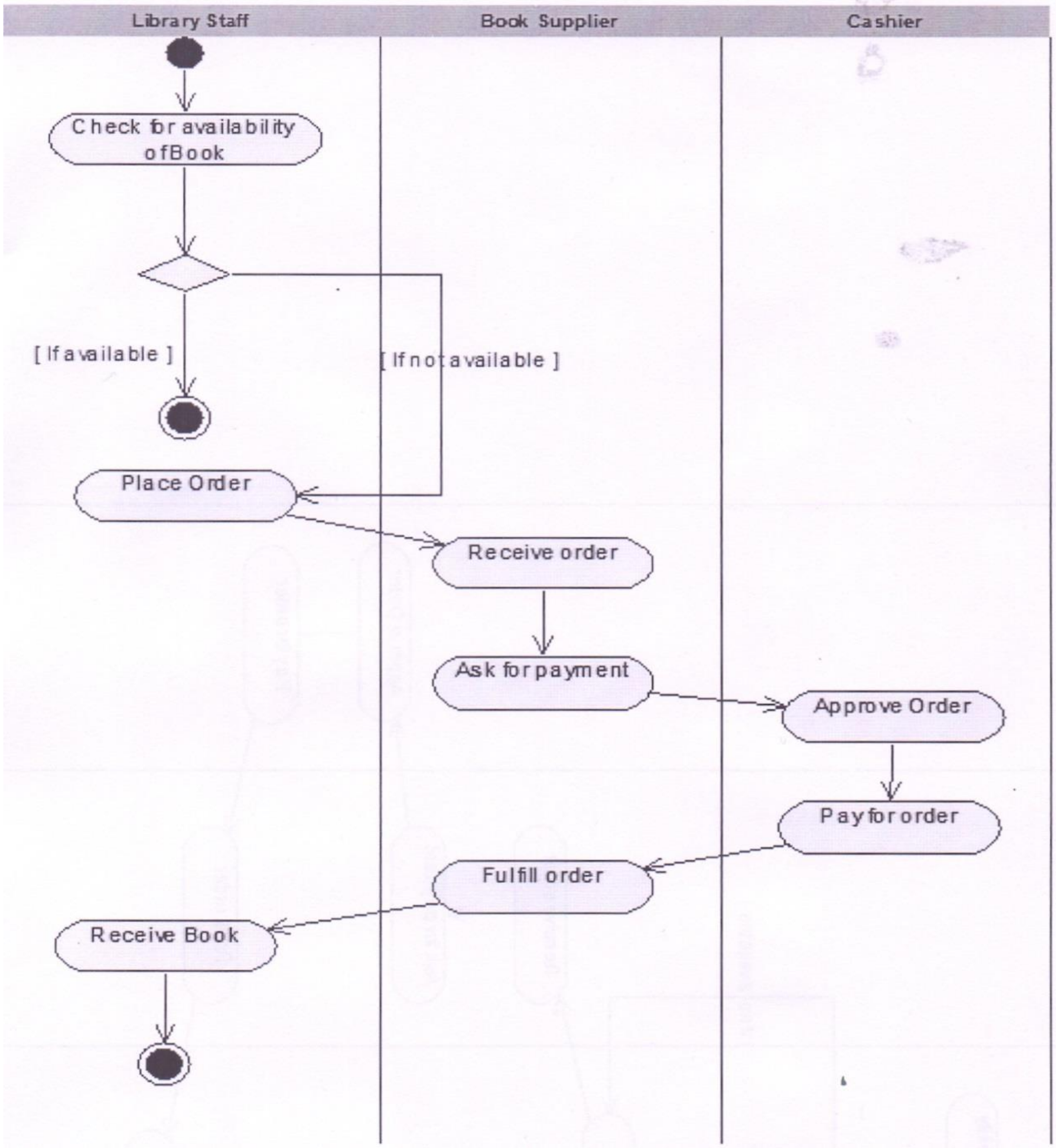


iii) Draw activity diagram to purchase books from supplier in library.

(Activity diagram for purchase book from supplier in library with correct notations 4M)

(Note: Any other activity diagram showing purchase book from supplier in library shall be considered.)

Ans: Activity diagram to purchase book from supplier in library

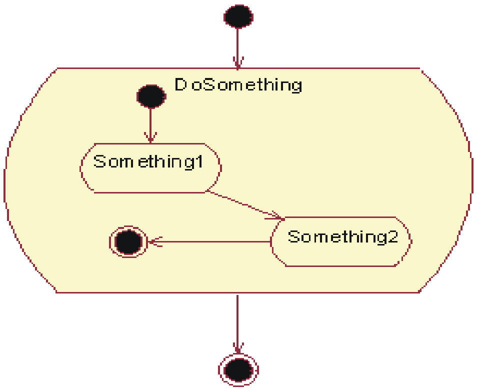


iv) What is importance of nested activity diagram? (Explanation 2M, suitable example 2M)

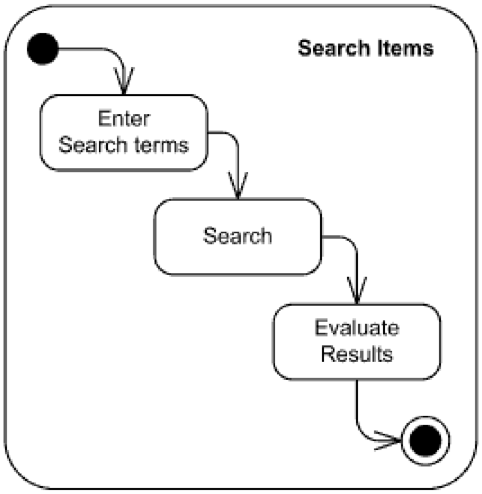
Ans: Nested Activity diagram

1. Nested activity diagram contains an activity diagram inside an activity.
2. Nested activity state may reference another activity diagram that shows the internal structure of the activity state.
3. It shows the subgraph(subactivity) inside of the activity state which refers to another diagram. Subactivity can be reused as independent activity.
4. Nested activity diagram gives the result of its activities to the activity in which it resides.

Structure:



Example:



Q.4) b) Attempt any ONE of the following: 6

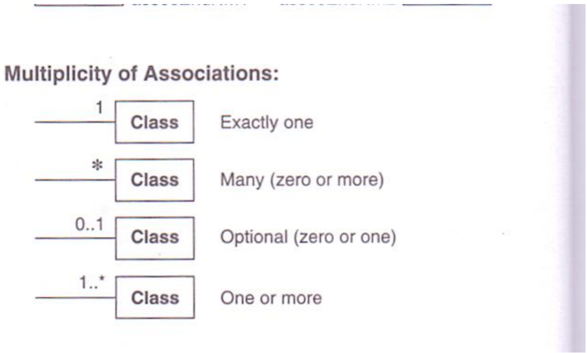
i) Explain multiplicity with all its notations.
(Explanation of multiplicity with example 4M, any 4 notation: 1/2M each).

Ans: Multiplicity

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 1:

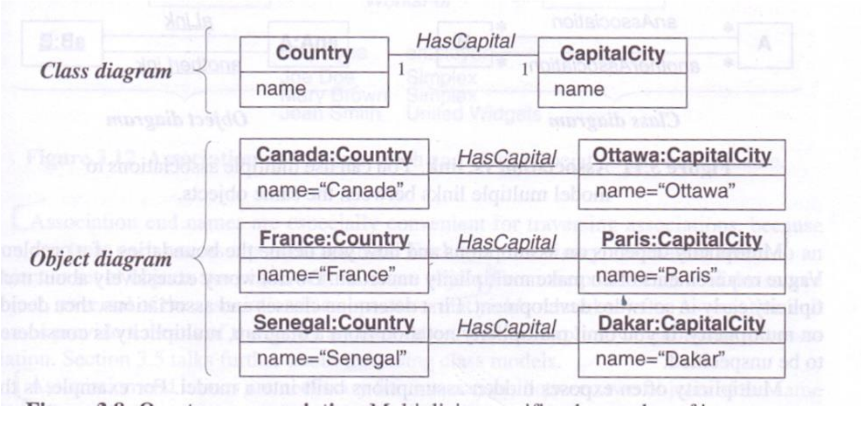


Fig: One-to-one association

Example 2:

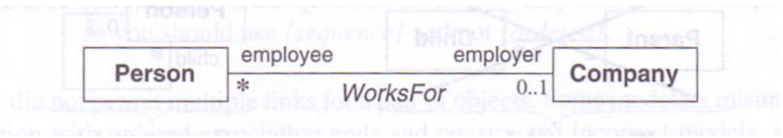


Fig: One to many association

ii) Explain structure control in sequence diagram for conditional and loop execution.

(Conditional Execution 3M, Loop Structure 3M OR Any Suitable Explanation 6M)

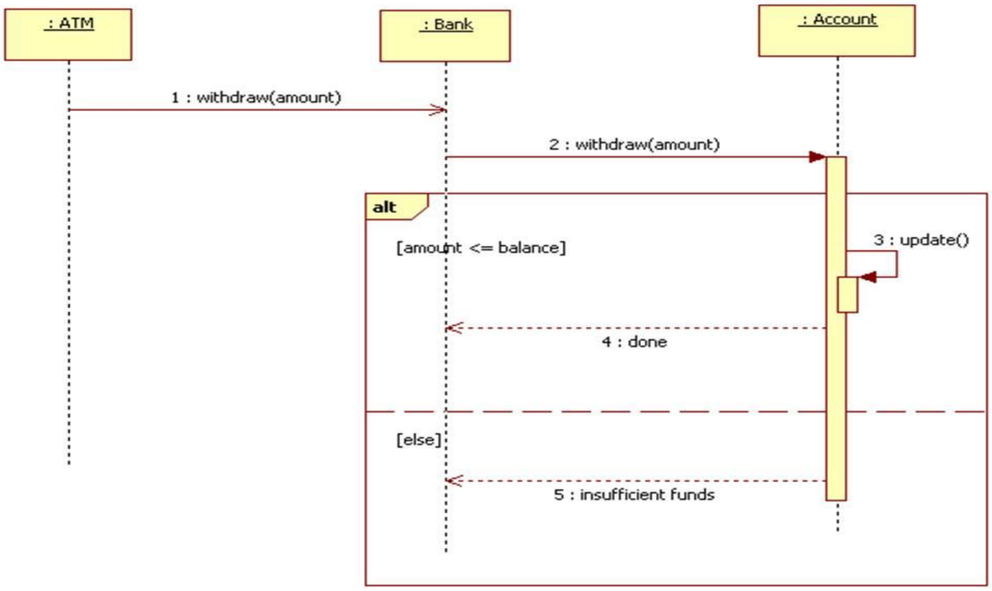
Ans: Structure control in sequence diagram

It defines statements or group of statements in a diagram which determines the sequence of execution of other instructions or statements.

Conditional execution

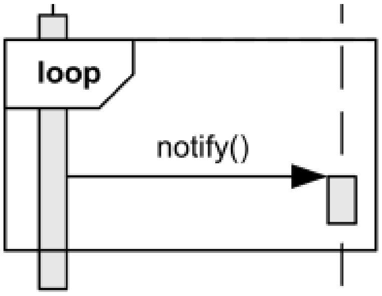
1. In sequence diagram conditional statements are used to check the condition.
2. Alt operator is used to check the condition in sequence diagram.
   The body of the control operator is divided into multiple subregions by horizontal dashed lines.
3. Each subregion represents one branch of a conditional. Each subregion has a guard condition. If the guard condition for a subregion is true, the subregion is executed.
4. 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.
5. One subregion may have the special guard condition [else]. This subregion is executed if none of the other guard conditions are true.

Example

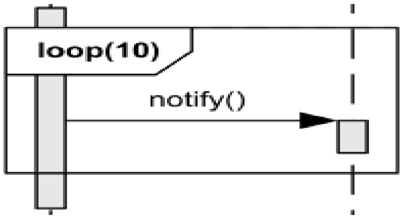


Loop

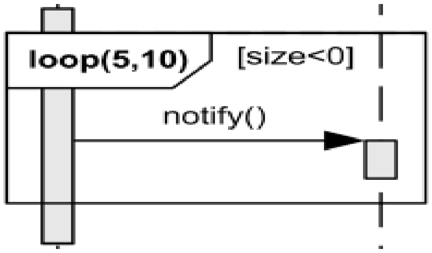
1. The interaction operator loop means that the combined fragment represents a loop.
2. The loop operand will be repeated a number of times.
3. The loop construct represents a recursive application of the seq operator where the loop operand is sequenced after the result of earlier iterations.
4. Loop could be controlled by either or both iteration bounds and a guard.
5. Loop operand could have iteration bounds which may include a lower and an upper number of iterations of the loop.
6. If loop has no bounds specified, it means potentially infinite loop with zero as lower bound and infinite upper bound.



7. If only min-int is specified, it means that upper bound is equal to the lower bound, and loop will be executed exactly the specified number of times.



8. If min and max bounds for the loop are specified then loop will exactly execute that number of times.



Q.5) Attempt any TWO of the following: 16

a) Explain importance of Synchronous and Asynchronous messages in sequence diagram. Draw sequence diagram for student admission in your institute.
(Explanation of Synchronous messages -2M, Asynchronous messages-2 Marks; sequence diagram for student admission in your institute with correct notations- 4M)
(Note: any other sequence diagram for student admission in your institute shall be considered.)

Ans: Synchronous Messages:

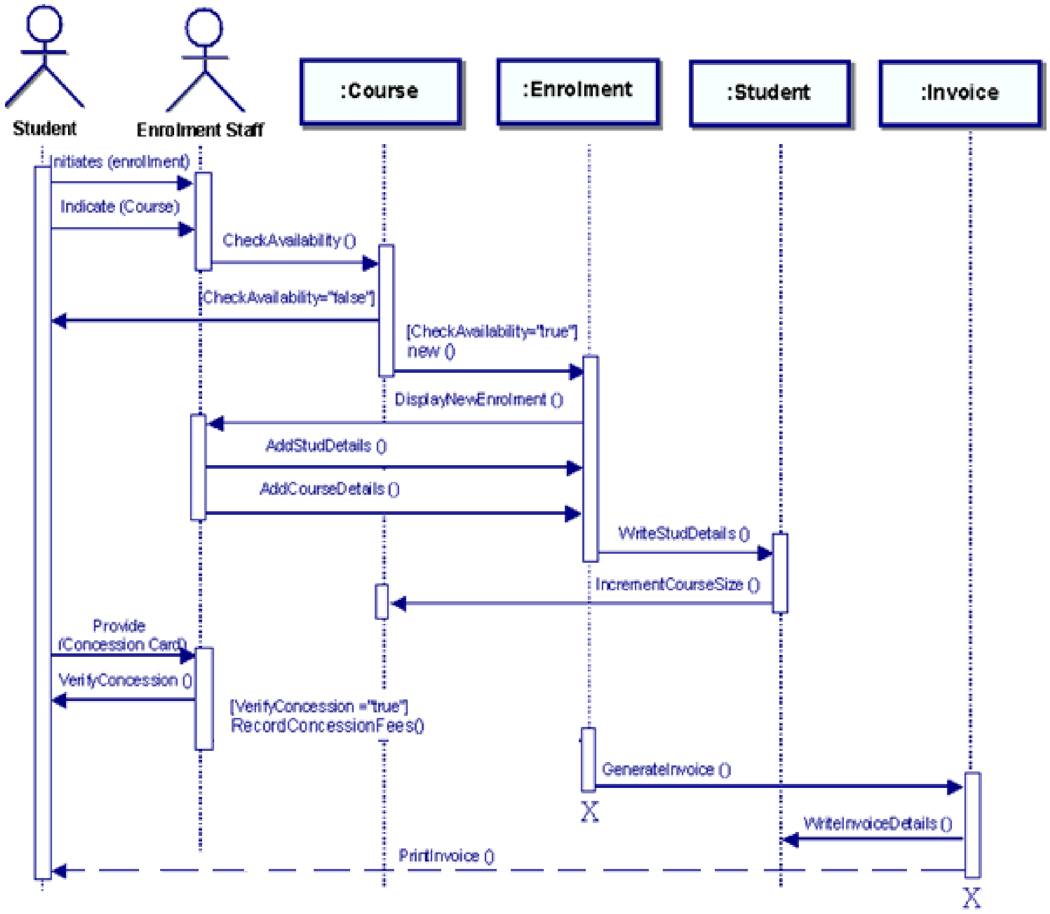
- A synchronous message is invoked when the Message Caller waits for the Message Receiver to return from the message invocation.
- A synchronous message is used when the sender waits until the receiver has finished processing the messages, only then does the caller continue.
- Most method calls in object-oriented programming languages are synchronous.
- A closed and filled arrowhead signifies that the message is sent synchronously. For example in given diagram transaction between enrolment staff and enrolment method is shown with such arrows.
- The rectangles on the lifeline are called activation and indicate that an object is responding to a message. It start when the message is received and ends when the object is done handling the message.
- When messages are used to represent method calls, each activation corresponds to the period during which an activation record for its call is present on the call stack.

- If you show that the receiver has finished processing the message and returns control to the sender, draw a dashed arrow from receiver to sender, optionally, a value that the receiver returns to the sender can be placed near the return arrow. For example Printinvoice() is finishing message shown with dotted line.

Asynchronous Messages:

- With an asynchronous message, the sender does not wait for the receiver to finish processing the message, it continues immediately.
- Message sent to a receiver in another process or calls that start a new thread are examples of asynchronous messages.
- An open overhead is used to indicate that a message is sent asynchronously.
  E.g. RecordConcessionFees() message is given diagram is an asynchronous message.

Sequence diagram for student admission in your institute:

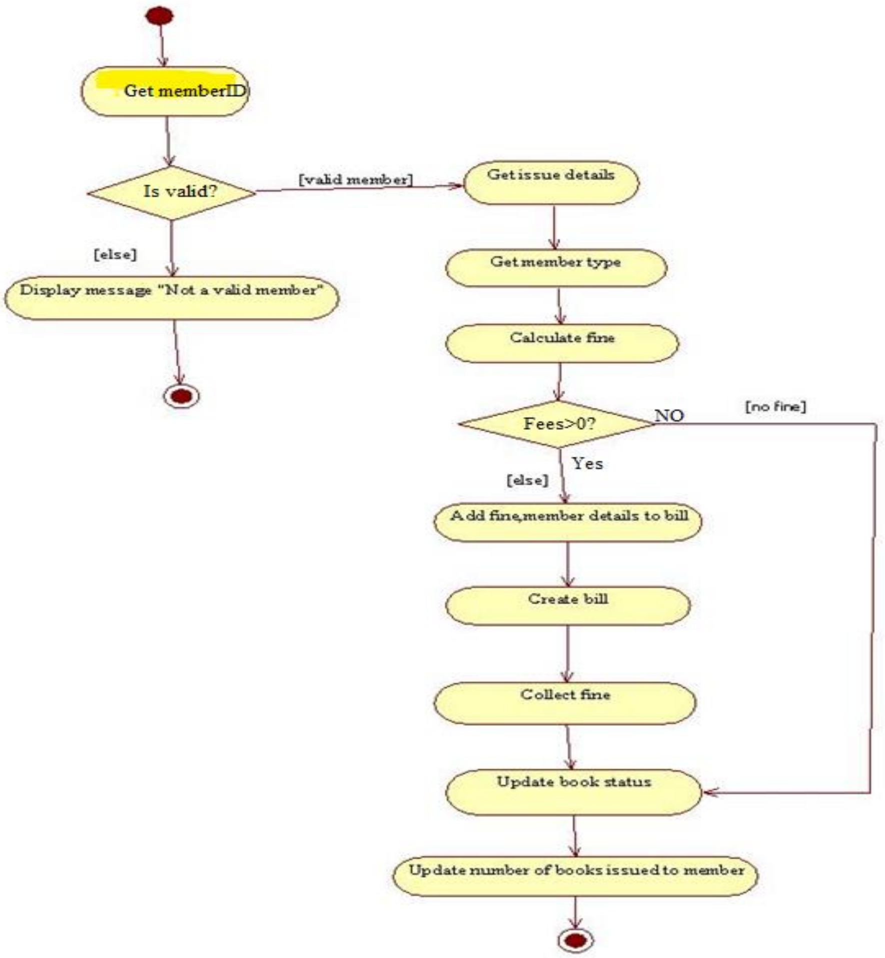


b) Explain decision making and branching in activity diagram.
Draw activity diagram for book return in library management system.
(Explanation of Decision Making and Branching-4 Marks; activity diagram for book return in library management system with correct notations-4 Marks)
Note: Any other activity diagram for book return in library management system shall be considered.

Ans: Decision Making and Branching:

- In an activity diagram, Diamond Shape is used for Decision and branches are represented by lines. The condition written in diamond is the decision criteria.
- A branch may have one incoming transition and two or more outgoing ones.
- On each outgoing transition, we place a Boolean expression, which is evaluated only once on entering the branch.
- We can use the keyword else to mark one outgoing transition, representing the path taken if no other guard expression evaluate to true.
- When two paths of control merge back together, we can also use a diamond symbol with two input arrow and one output arrow. No guards are necessary on merge.
- We can achieve iteration by using one action that sets the values of an iterator, another action that increments the iterator, and a branch that evaluates if the iteration is finished.
- The UML include node types for loops, but these may often be expressed more easily in text than in graphics.

Activity diagram for book return in library management system



c) **What is component? Is component replaceable? Explain why.**
Draw all notations necessary to draw component diagram.
(Defining component-2 Marks, explanation of component replacement-2Marks;Any Four Notations: 1 Mark each)

Ans: Component:

- A **component** is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. A component is rendered as a rectangle with tabs.
- A component has its behavior defined in terms of **provided interfaces** and **required interfaces** (potentially exposed via **ports**).
- Larger pieces of a system’s functionality may be assembled by reusing components as parts in an encompassing component or assembly of components, and wiring together their required and provided interfaces.

- A component is modeled throughout the development life cycle and successively refined into deployment and run-time. A component may be manifested by one or more artifacts.
- **Indirectly instantiated component** is defined at design time but does not exist as addressable object at execution time.
- Runtime behavior of the component and its ports is defined by the runtime behavior of classifiers or parts realizing it. Several standard stereotypes assume this attribute, e.g., **«specification»**,**«focus»**, **«subsystem»**.

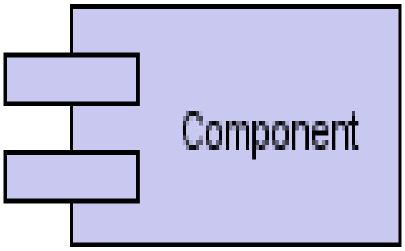
Is Component Replaceable?

- A component is replaceable. A component is substitutable-it is possible to replace a component with another that conforms to the same interface.
- A mechanism of inserting or replacing a component to form a run time system is transparent to the component user and is enabled by object models that require little or no intervening transformation or by tools that automate the mechanism.

Notations of Component Diagram:

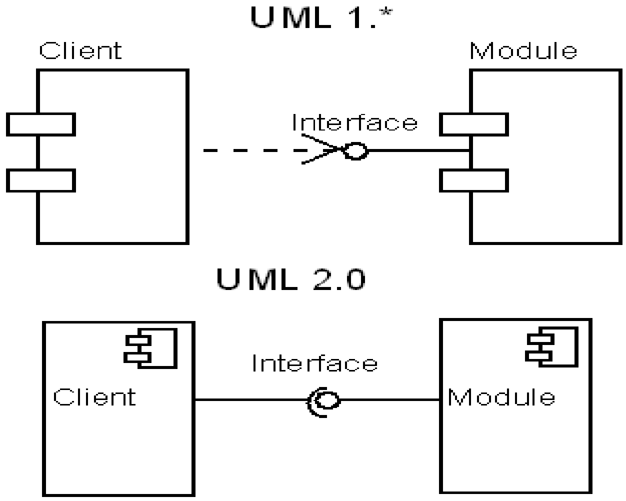
1. **Component**

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with tabs, usually including only its name. A component icon is a rectangle with two smaller rectangles jutting out from left-hand side. This symbol is visual stereotype.



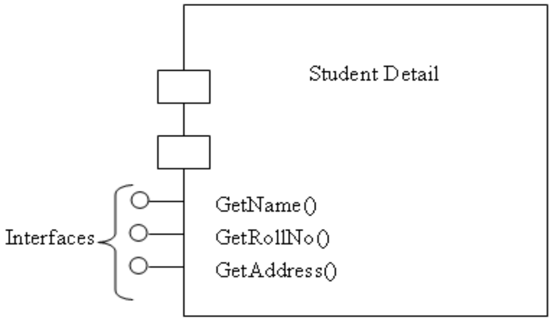
2. Interface

A component can be connected with other components through interfaces. An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.



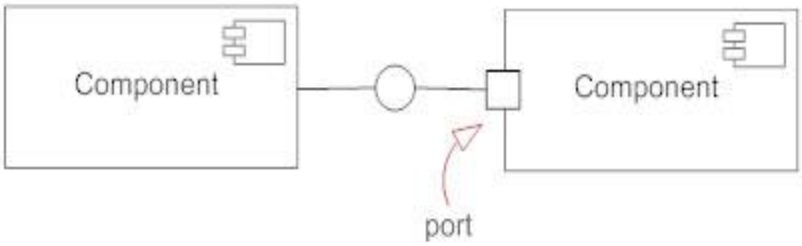
3.Dependencies

A dependency exists between two elements if changes to the definition of one element may cause changes to the other. It represented as dotted arrow as shown in figure.



4. Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



Q.6) Attempt any FOUR of the following: **16**

**a) Describe <<include>> and <<extend>> relationships in use case diagram.
(Explanation of <<include>> relationship -2 Marks; Explanation of <<extend>>relationship-2 Marks)**

Ans: Include relationship is use case of indicates direct incorporation of one use case to another. Whereas extend relationship is indirect incorporation.

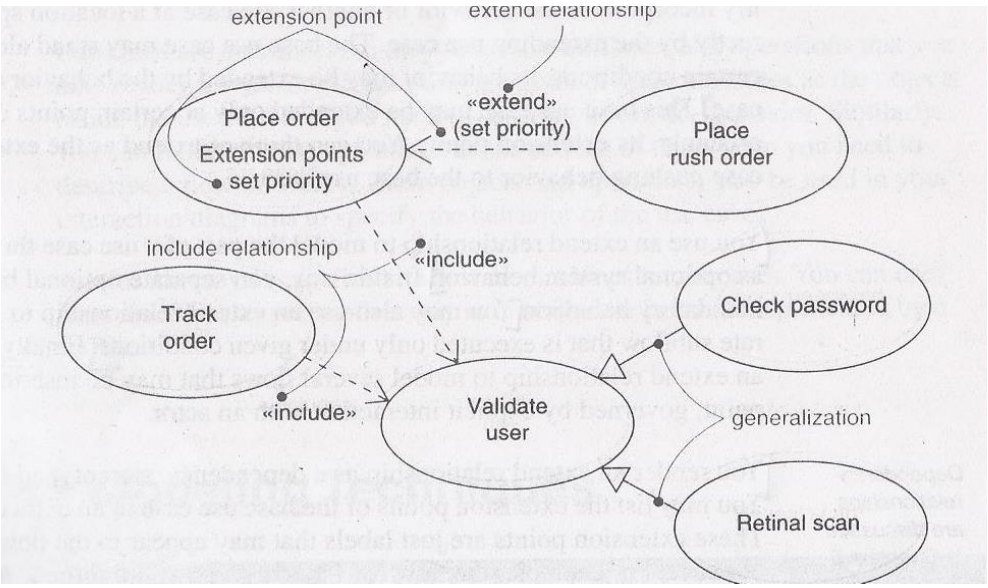
<<include>> relationships:

- 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, but is only instantiated as part of some larger base that include it.
- 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, simply write include followed by the name of the use case.

<<extend>> relationships:

- 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 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.

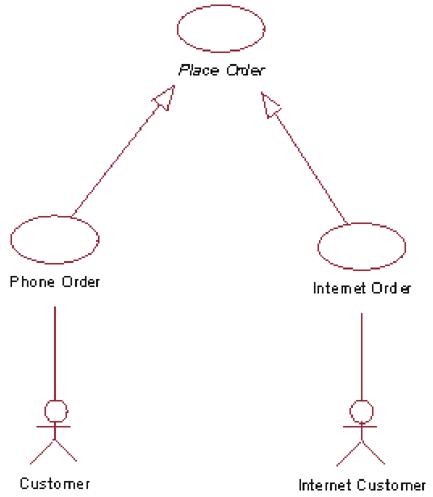
Example:-



b) Describe what is use case generalization in detail with example.
(Explanation with example-4 Marks)
(Note: Students may right any appropriate example to show use case generalization. Keywords are parent use case, child use case, sharing specifications, abstract)

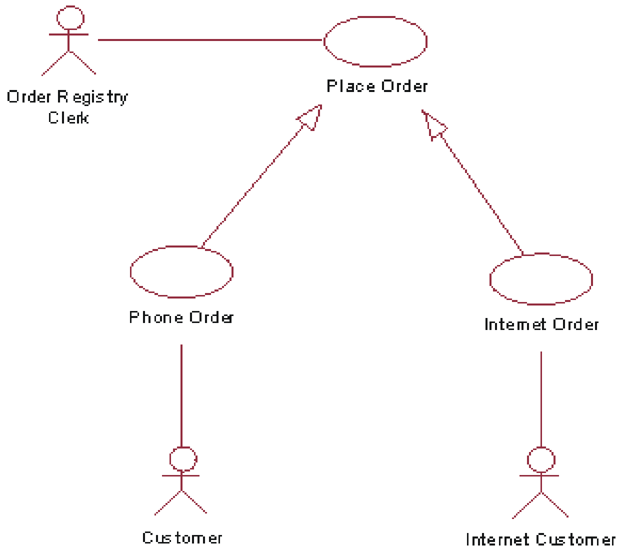
Ans:

- Generalization is used when you find two or more use cases that have commonalities in behavior, structure, and purpose. When this happens, you can describe the shared parts in a new use case which is child use case.
For e.g.: Place order is main system use case which may get order from phone or internet, so phone order and internet order are child use cases.



The use cases Phone Order and Internet Order are specializations of the abstract use case Place Order.

- In an Order Management system, the use cases Phone Order and Internet Order share a lot in structure and behavior. A general use case Place Order is defined where that structure and common behavior is defined. The abstract use case Place Order need not be complete in itself, but it provides a general behavioral framework that the child use cases can then make complete.
- The parent use case is not always abstract.
Example:

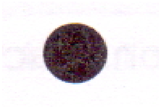
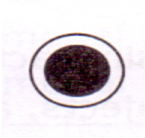
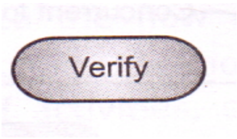
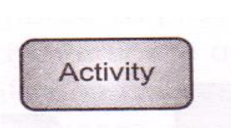
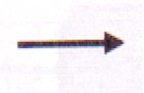
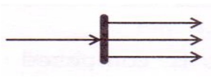
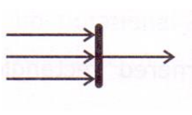


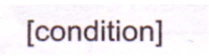
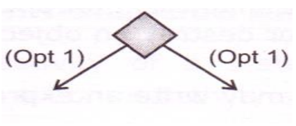
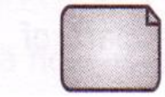
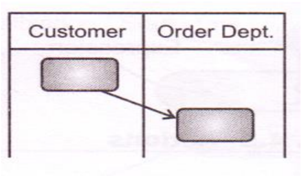
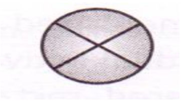
- The child use cases can add behavior to the structure that the parent use case provides, and also modify behavior in the parent.
- The child use case is dependent on the structure of the parent use case. The child use case may add additional behavior to the parent by inserting segments of behavior into the inherited behavior, or by declaring include- and extend-relationships to the child use case.
- The child may modify behavior segments inherited from the parent, although it must be done with care so that the intent of the parent is preserved. The structure of the parent use case is preserved by the child. This means that all behavior segments, described as steps or subflows of the parent’s flow of events, must still exist, but the contents of these behavior segments may be modified by the child.
- If the parent is an abstract use case, it may have behavior segments that are incomplete. The child must then complete those behavior segments and make them meaningful to the actor.
- A parent use case need not have a relationship to an actor if it is an abstract use case.
- If two child use cases are specializing the same parent (or base), the specializations are independent of one another, meaning they are executed in separate use-case instances. This is unlike the extend- or include-relationships, where several additions implicitly or explicitly modify one use-case instance executing the same base use case.
- Both use-case-generalization and include can be reuse behavior among use cases in the model.

c) **Draw and state notations used to draw activity diagram.**

(Any four notation: Each 1 Mark)

Ans: Notations of Activity Diagram:

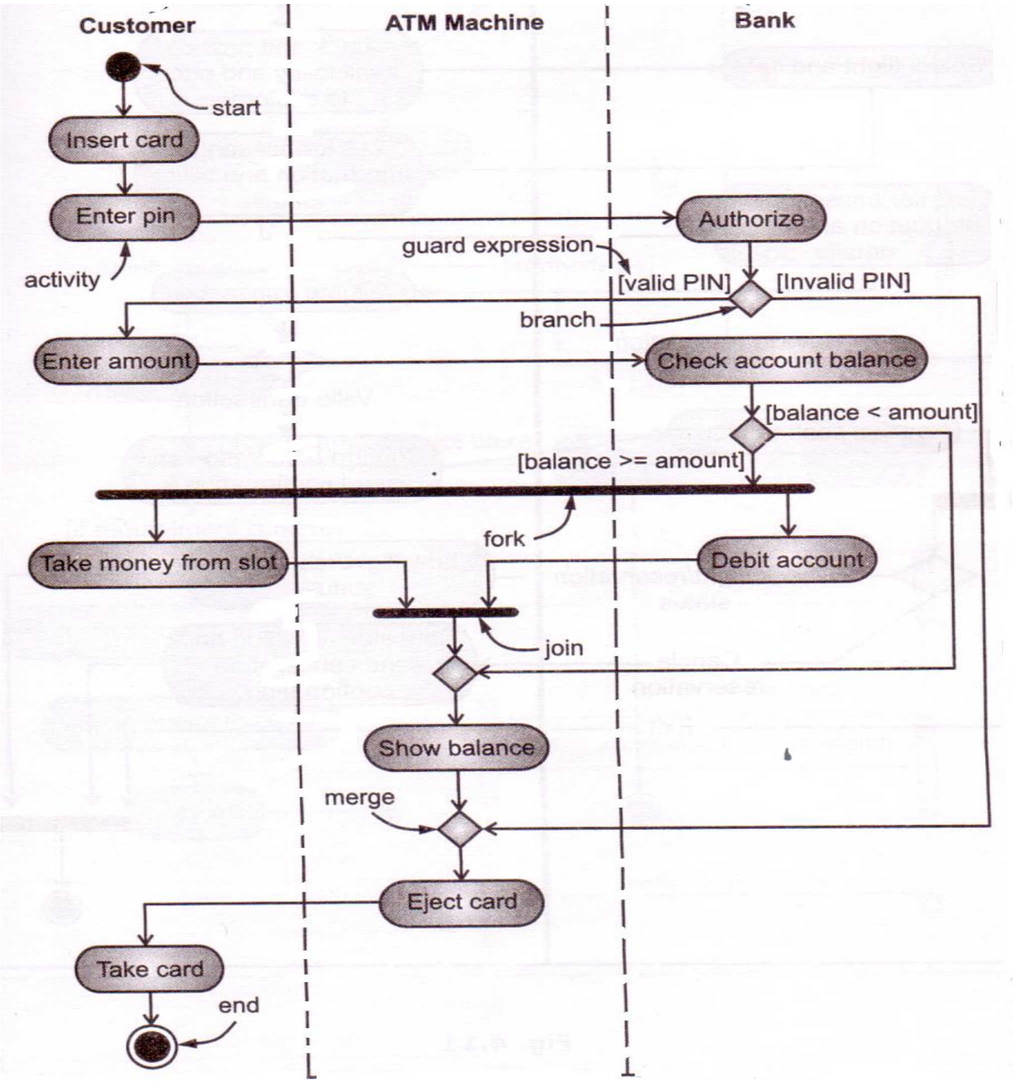
Name	Symbol	Description
1.Start/Initial Nodes		It shows the starting point of the activity diagram. An initial or start node is depicted by a filled circle with black color.
2. Final/Exit Node		It shows the exit point of the activity diagram. An activity diagram can have zero or more activity final nodes. Final node is rendered as two concentric circles with filled inner circle.
3. Action		Actions are active steps in the completion of a process. Actions are denoted by rounded rectangles. Action is smallest unit of work which cannot be divided into further tasks.
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.
12. Flow Final Node		A flow end node terminates its own path not the whole activity. The flow final node is depicted as a circle with a cross inside.

d) Draw an activity diagram to print receipt of one transaction in ATM.
(Activity Diagram to print receipt of one transaction in ATM with correct notations-4 Marks)
Note: Any other Activity Diagram to print receipt of one transaction in ATM shall be considered.

Ans:

“Withdraw money from a bank account through an ATM.” The three involved classes (people, etc.) of the activity are Customer, ATM, and Bank. The process begins at the back start circle at the top and ends at the concentric white/black stop circles at the bottom. The activities are rounded rectangles.

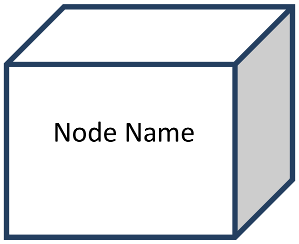


e) **Define node. Draw notation. Explain use of deployment diagram.**
(Definition of node -1M, notation-1 Marks; Use of Deployment diagram-2 Marks)

Ans: Node:

- Nodes, just like artifacts, are an important building block in modelling 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.

Notation:



Usage 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.