USN	1	P	E			M	C	A		

PESIT Bangalore South Campus

Hosur Road, 1km before Electronic City, Bengaluru -560100

Department of Master of Computer Applications

INTERNAL ASSESSMENT TEST II

Date : 23/09/19	**Max Marks:** 40
Subject & Code: Object Oriented Modeling & Design (17MCA51)	**Section:** V MCA
Name of Faculty: Ms. Richa Sharma	**Time:** 8:30-10:00 AM

Note: ***Answer FIVE full questions. Select one question from each part.***

	Part I	
Q 1	Explain the concept of aggregation with the help of example.	**8**
	OR	
Q 2	Discuss Procedural Sequence Models in detail with the help of example.	**8**
	Part II	
Q3	Discuss Activity Models in detail with suitable example.	**8**
	OR	
Q4	Explain Multiple Inheritance along with Multiple Classification & Workarounds.	**8**
	Part III	
Q 5	a) Discuss Use Case Relationships. b) Discuss various types of events.	**8**
	OR	
Q6	Explain Sequence Models in detail with the help of "Online Stock Broker" example.	**8**
	Part IV	
Q 7	a) Discuss Transitions & Conditions with suitable example. b) Draw Use Case diagram for Vending Machine. Also explain the guidelines for the same.	**8**
	OR	
Q 8	Explain the concept of Concurrency in State Diagrams along with "Car Aggregation" example.	**8**
	Part V	
Q 9	Discuss Use Case Summary and Use Case Description for Vending Machine Application.	**8**
	OR	
Q 10	Write short notes on a) Abstract Classes. b) Derived Data.	**8**

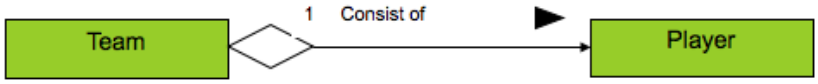
Solution Set

PART-I

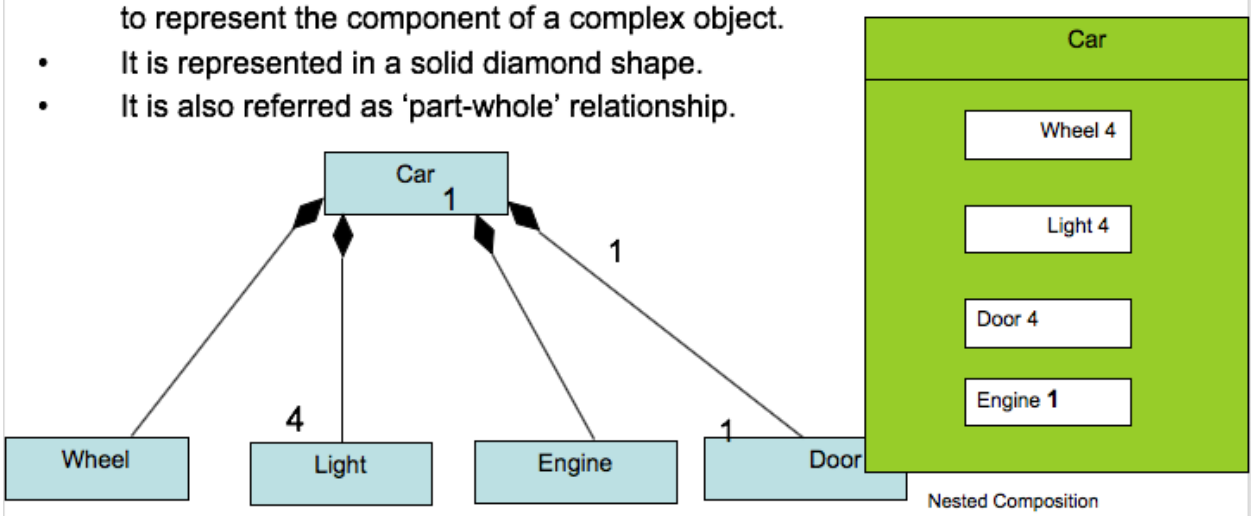
1. **Explain the concept of aggregation with the help of example.**

Aggregation & Composition (a part of):

- Aggregation is a form of association.
- A hollow diamond is attached to the end of the path to indicate aggregation.



- Composition known as 'a part of' is a form of aggregation with strong ownership to represent the component of a complex object.
- It is represented in a solid diamond shape.
- It is also referred as 'part-whole' relationship.



Aggregation --- Association

- Aggregation is a strong form of association in which aggregate objects is made of constituent parts of the aggregate.
- It is a special form of association, not an independent concept.
- It is a kind of association in which an aggregate object is made of constituent parts.
- If two objects are tightly bound by a part-whole relationship, it is aggregation. If the two objects are independent, it is association

Aggregation --- composition

- UML has two forms of part-whole relationships
- A general form called aggregation and more restrictive form called composition
- Composition is a form of aggregation with two additional constraints.
- A constituent part can belong to at most one assemble.
- Constituent part has coincident lifetime with assembly.
- Operations can be propagated across aggregation and compositions

owns

- Person---------document----------paragraph--------character

2. **Discuss Procedural Sequence Models in detail with the help of example.**

Horizontal dimension represent different objects- objects in real world. Ex: card reader, ATM screen

Vertical dimension represent the time.

Vertical line is called the object's lifeline

Object's lifeline represents the object's existence during the interaction.

A message is drawn between the lifelines of two objects to show that the objects communicate

Each message represents one object making a function call of another. Each message is labeled with message name.

Label can also include argument and some control info and show self-deligation, a message that an object sends to itself.

Sequence diagram with passive objects

Sequence diagram with transient objects

****Sequence diagrams with passive objects**

- With procedural code all objects are not constantly active.
- Most objects are passive and do not have their own threads of control
- Passive object is not activated until it has been called.
- Once the execution of an operation completes and control returns to the caller, the passive object becomes inactive.
- Thin rectangle is the activation and the entire period during which the object exists is called the lifeline of an object.

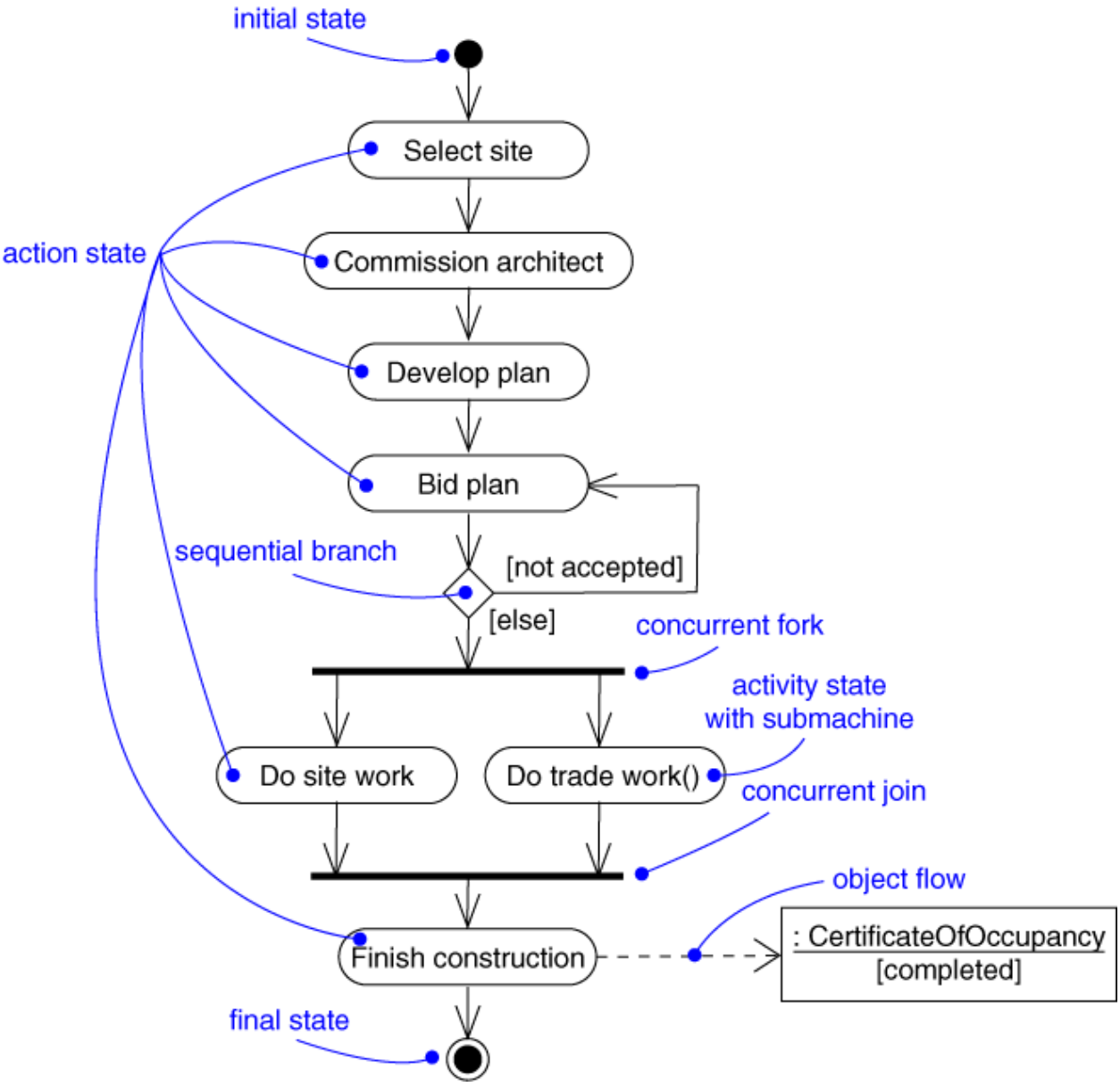
PART-II

3. **Discuss Activity Models in detail with suitable example.**

UML Activity Diagram:

- An activity diagram shows the sequence of steps that make up a complex process such as an algorithm or workflow.
- Activity diagram is a special case of state machine in which the states are activities representing the performance of operations.
- Activities, branches, initiation and termination, concurrent activities and executable activity diagrams

- An outgoing solid arrow attached to an activity symbol indicates a transition triggered by the completion of the activity.
- Activity diagram express a decision when condition are used to indicate different possible transition that depend on Boolean conditions of container object.



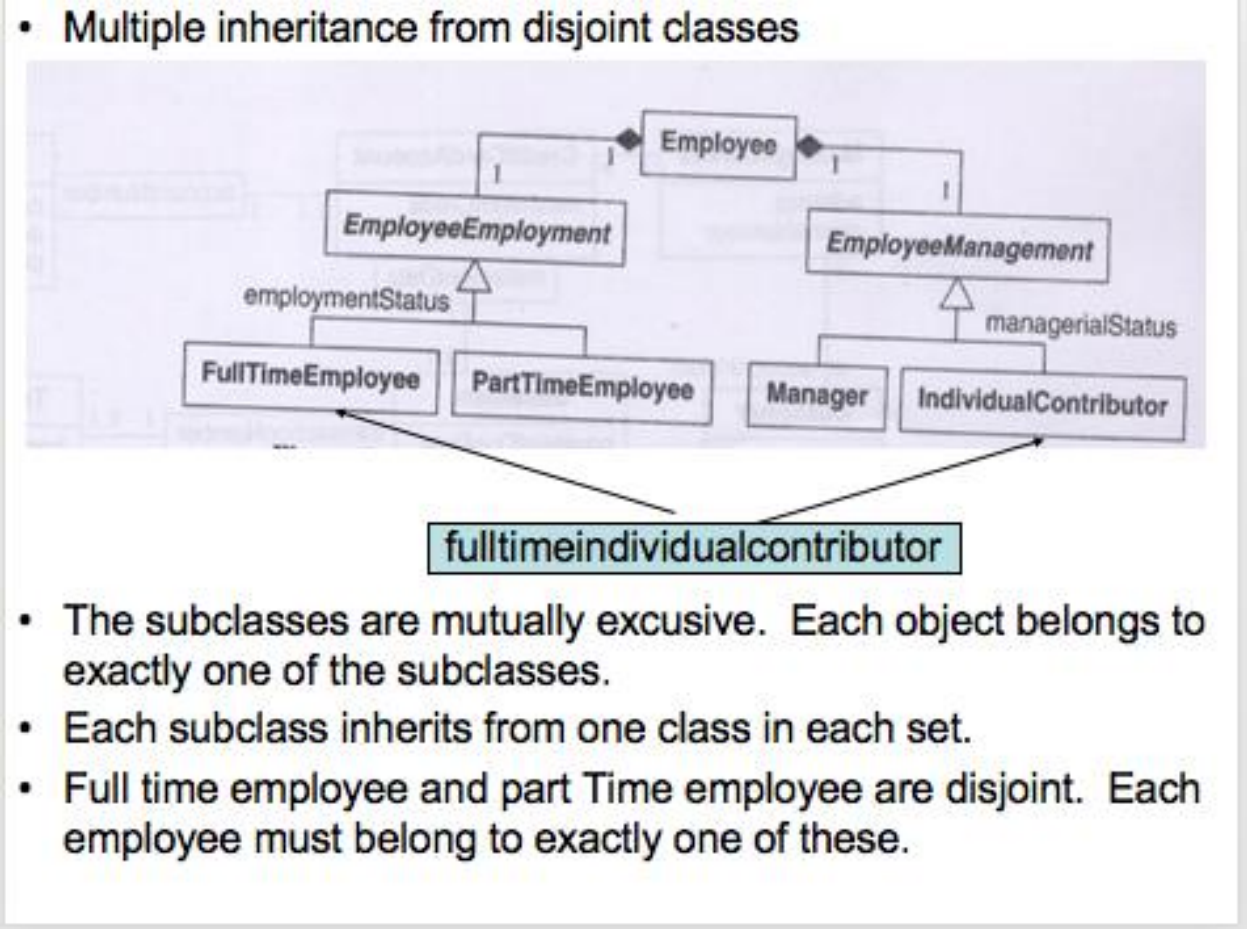
Guidelines for activity models:

- Don't misuse activity diagrams
- Level diagrams
- Be careful with branches and conditions
- Be careful with concurrent activities
- Consider executable activity diagrams

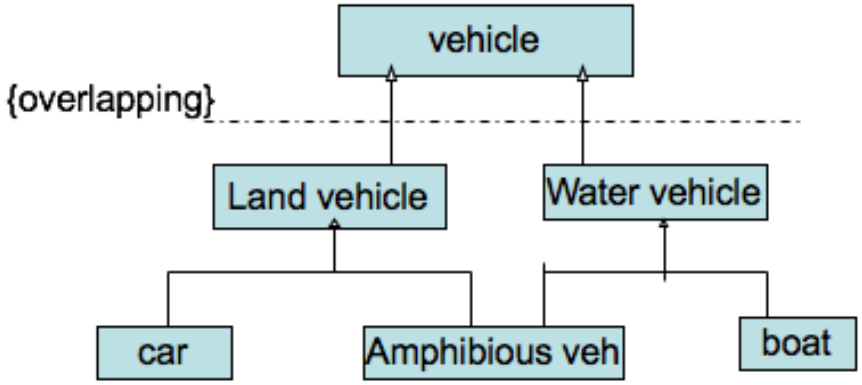
4. **Explain Multiple Inheritance along with Multiple Classification & Workarounds.**

Multiple Inheritance

- Multiple inheritance permits a class to have more than one superclass and to inherit features from all parents.
- Each generalization should cover a single aspect.
- Multiple generalization should use if a class can be refined on several distinct and independent aspects.
- A subclass inherits a feature from the same ancestor class.
- **Kinds of multiple inheritance**
- Multiple inheritance form disjoint classes
- Multiple inheritance from overlapping classes
- Multiple classification

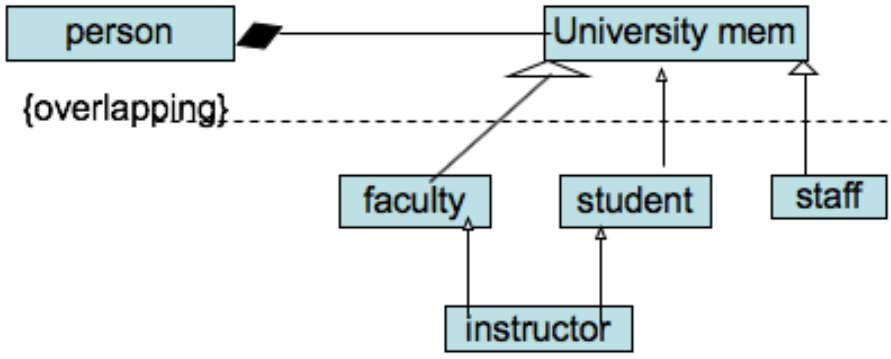


- Multiple inheritance from overlapping classes



- Overlapping means that the subclasses can share some objects. An object may belong to more than one subclass.
- Landvehicle and watervehicle overlap, b'cos vehicle may belong to more than one subclasses.

- Multiple classification



An instance of a class is inherently an instance of all ancestors of the class. Ex: an instructor could be both faculty and student. But if we have another Professor taking class at US. There is no class to describe the combination. Hence person is treated as an object composed of multiple unv mem object. This workaround replaces inheritance with deligation.

Workarounds

- Dealing with multiple inheritance is an implementation issue.
- Some restructuring technique is called delegation.
- Delegation is an implementation mechanism by which an object forwards an operation to another object for execution.

Delegation(allocation) using compositon of parts- recast a superclass with multiple independent generalizations as a composition in which each constituent part replaces a generalization.

- Inheritance of operations across the composition is not automatic. The composite must catch operations and delgate them to appropriate part.
- Need not create the various combinations as explicit classes.
- All combinations of subclasses form the different generalizations are possible. Ex: 4.18

Inherit the most important class and delegate the rest- preserves identity and inheritance across the most impt generalization. Ex:4.19

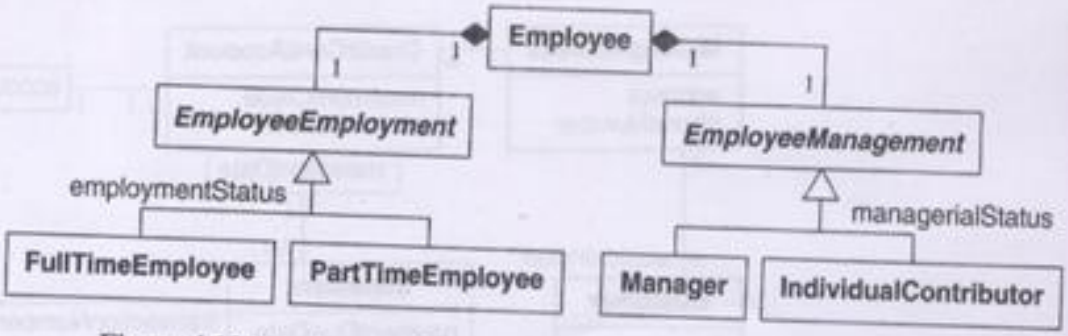


Figure 4.18 Workaround for multiple inheritance—delegation

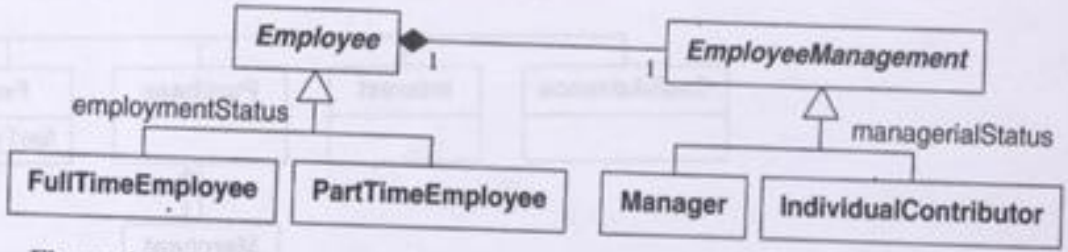


Figure 4.19 Workaround for multiple inheritance—inheritance and delegation

Need not create the various combinations as fulltimeindividualcont As explicit classes. All combinations of subclasses from different generalizations are possible.

- Nested generalization
- Factor on one generalization first, then the other
- Fulltime empl and part timeempl add two subclasses for managers and individual contributor

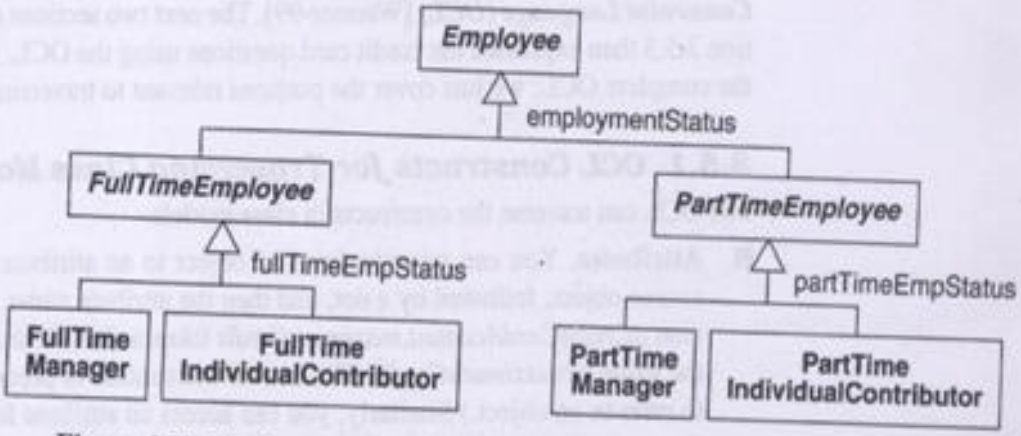


Figure 4.20 Workaround for multiple inheritance—nested generalization

Several issues to consider when selecting the best workaround.

- Superclasses of equal importance-if a subclass has several superclasses, all of equal improtance, use delgation and preserve symmetry in the model. 4.18
- Dominant superclass-if one superclass clearly dominates and the others are less important, preserve inheritance through this path , 4.19 or 4.20
- Few subclasses- if no. of combinations are small, consider nested. 4.20
- Sequencing generalization sets- if u use nested, factor on the most important criterion first, next second and so forth
- Large quantities of code- try to avoid nested.
- Identity- consider the importance of maintaining strict identity. Only nested generalization preserves this 4.20

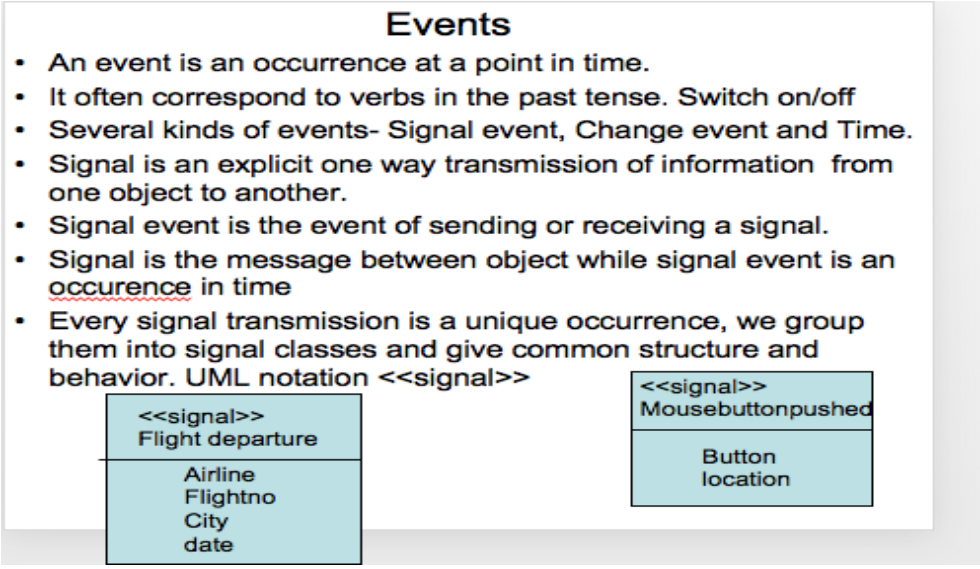
PART-III

5. (a) Discuss Use Case Relationships.

1. **The Include relationship: <<include>>: Relationship in use case diagram:**
2. **Include relationship:** It incorporates one usecase within the behavior sequence of another use case.
3. It is like a subroutine.
4. Keyword << include>>
5. An included usecase is inserted at a specific location within the behavior sequence of the larger use case.
6. Like subroutine is called from a specific location within another subroutine
7. The include relationship implies that the included behavior is a necessary part of a configured system.

- **The Extend relationship: <<extend>>:** Adds incremental behavior to a use case.
- It is like an include relationship, in which the extension adds itself to the base rather than the base explicitly incorporating to extension.
- Keyword <<extend>>
- The extend relationship connects an extension use case to a base use case.
- The base use case must be a valid use case in the absence of any extensions.

(b) Discuss various types of events



Change event

- Change event is an event that is caused by the satisfaction of a boolean expression.
- Whenever the expression changes from false to true the event happens
- UML notation for change event is keyword **when**
- **Ex**
- when (room temp < heating set point)
- When (room temp > cooling set point)

Time event

- Time event is an event caused by the occurrence of an absolute time .
- Uml notation **after**
- **Ex**
- when (date = january 1, 2009)
- After(10 seconds

6. Explain Sequence Models in detail with the help of "Online Stock Broker" example.

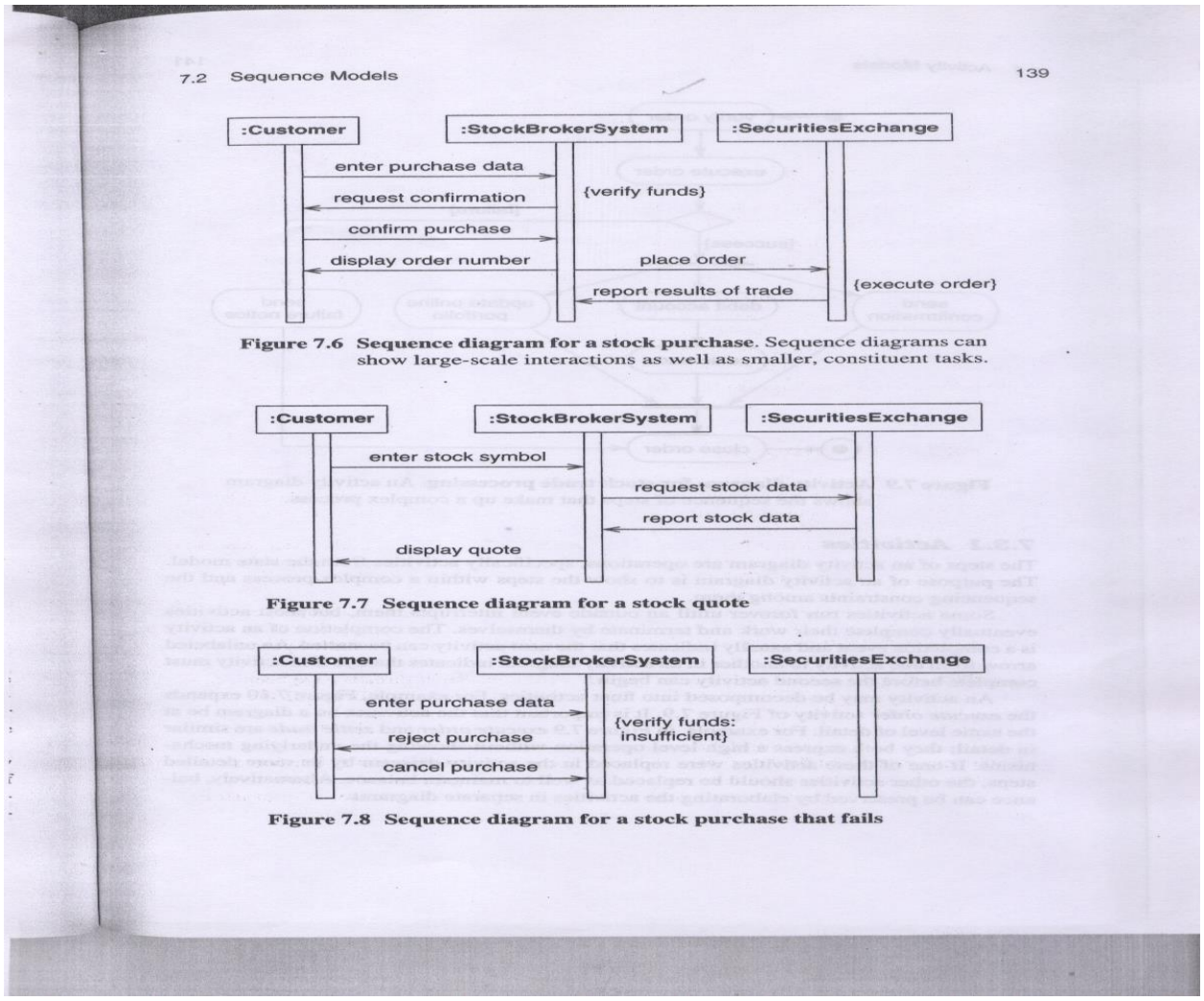
Sequence Diagram: -very simple, visual appeal & overall flow of system

Sequence diagram has two dimensions are,

- Vertical Dimension
- Horizontal Dimension
- Horizontal dimension represent different objects- objects in real world. Ex: card reader, ATM screen

- Vertical dimension represent the time.
- Vertical line is called the object's lifeline
- Object's lifeline represents the object's existence during the interaction.
- A message is drawn between the lifelines of two objects to show that the objects communicate
- Each message represents one object making a function call of another.
- Each message is labeled with message name.

- Label can also include argument and some control info and show self-deligation, a message that an object sends to itself.
- Analysts see the flow of processing in the sequence diag
- Developers see objects that need to be developed and operations for those objects.
- QA see the details of the process and develop test cases based on the processing.

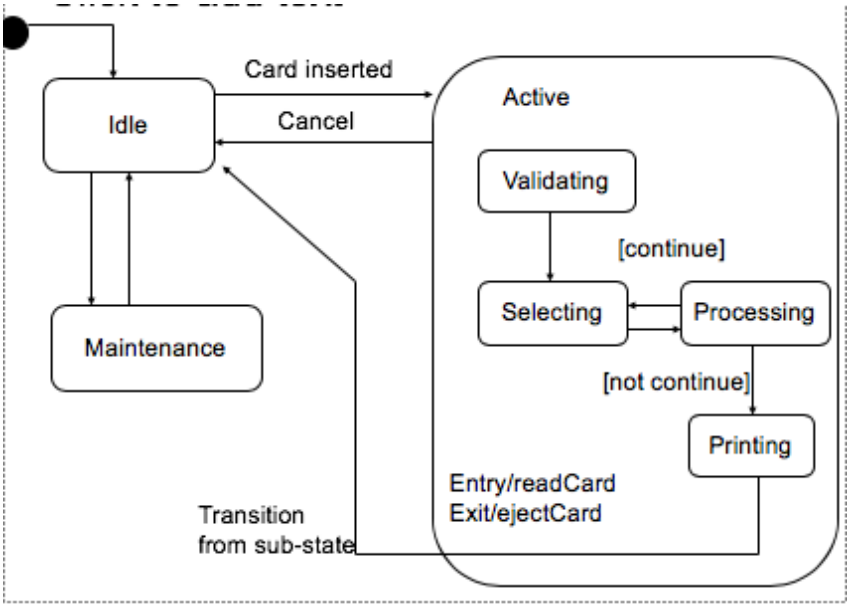


PART-IV

7. a) Discuss Transitions & Conditions with suitable example.

Transitions and conditions

- Transition is an instantaneous change from one state to another. Ex : phone.
- The transition is said to fire upon the change from the source state to the target state.
- Transitions fires when its event occurs. An event may cause multiple objects to transition.
- Guard condition is a boolean expression that must be true in order for a transition to occur.
- Guard transition fires when its event occurs, but only if the guard condition is true.
- A guard condition is checked only once at the time the event occurs and then fires.


Card inserted
Idle
Cancel
Active
Validating
[continue]
Selecting
Processing
Maintenance
[not continue]
Printing
Entry/readCard
Exit/ejectCard
Transition
from sub-state