CMR
INSTITUTE OF
TECHNOLOGY

USN | 1 | C | | | | | | | | |

Internal Assessment Test 2 – October 2019

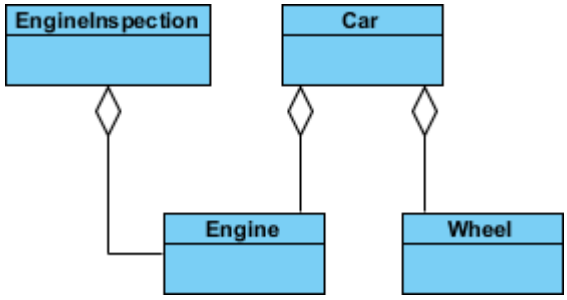
Sub:	Object Oriented Modeling and Design Patterns							Code:	17MCA51
Date:	12-10-19	Duration:	90 mins	Max Marks:	50	Sem:	V A & B	Branch:	MCA

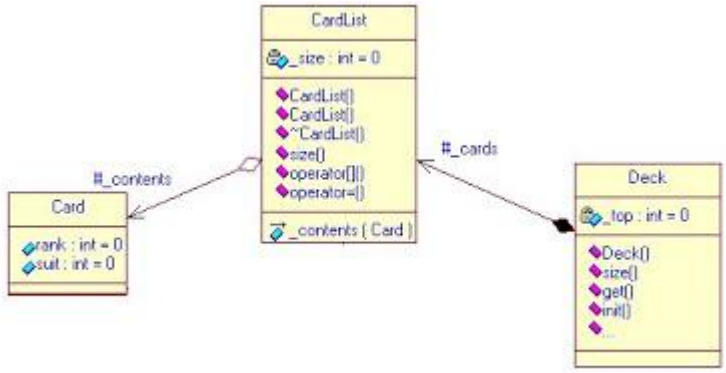
Note: Solution for Total Marks: 50

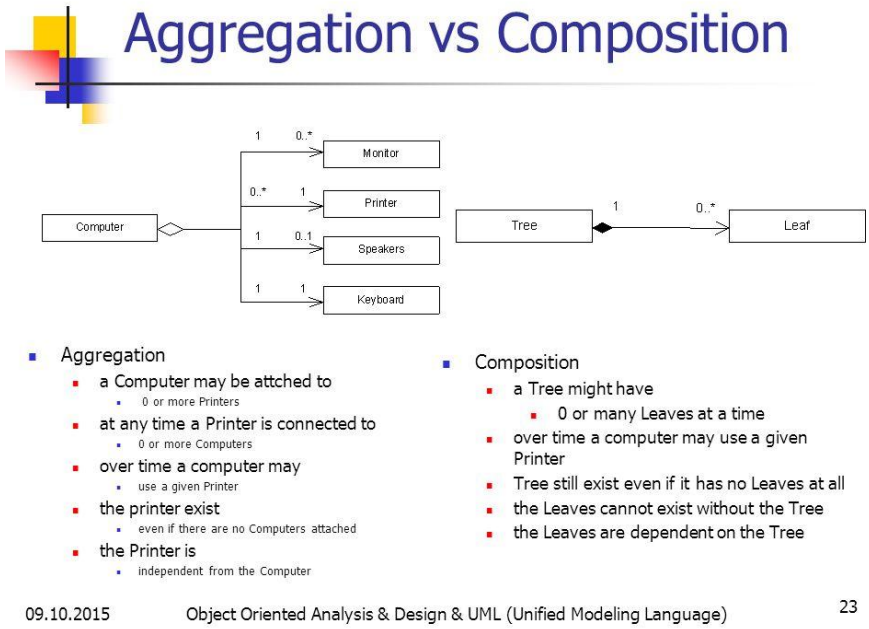
		Marks	OBE CO	OBE RBT
	PART-1			
1.	**What is aggregation and composition? Give their respective UML notations, with an example?**	10	CO2	L2

Aggregation implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.

Composition implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.







2. **What is an event? Explain different types of events, with an example.** 10 CO2 L2

An event is an occurrence at a point in time, such as user depresses left button or flight 123 departs from Chicago. Two events that are casually unrelated are said to be concurrent.

There are several kinds of events. The most common are the signal event, the change event and the time event.

Different kind of events:

1. Signal Event
2. Change Event
3. Time Event

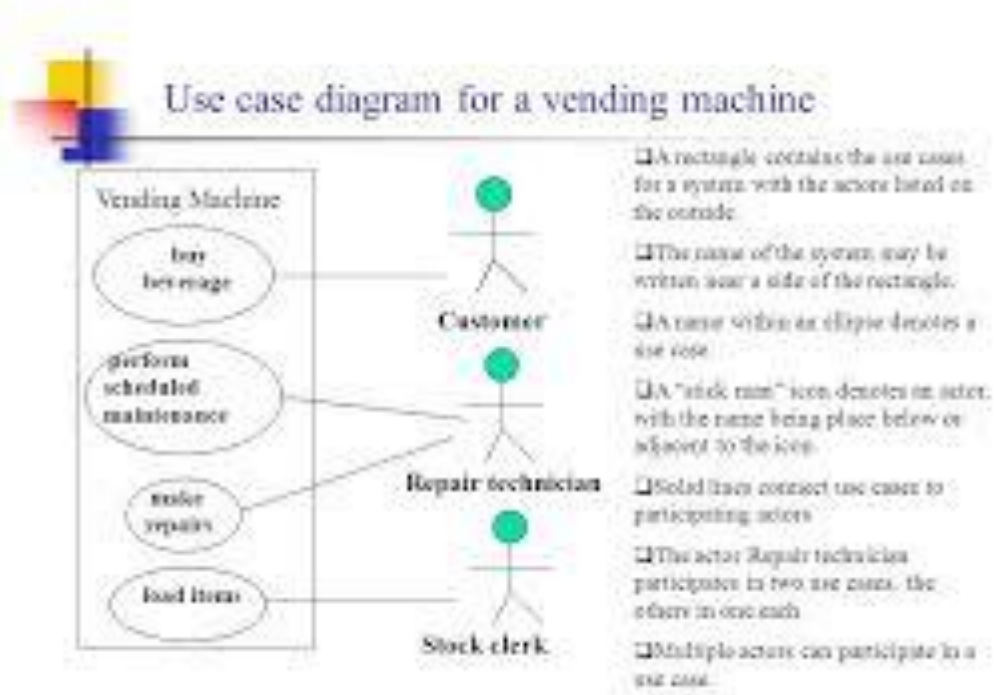
Signal Event: A signal is an explicit one-way transmission of information from one object to another.

Change Event: A change event is an event that is caused by the satisfaction of a boolean expression.

Time Event: A time event is an event caused by the occurrence of an absolute time or the elapse of a time interval.

PART-II

3. **Draw the use-case diagram, for vending machine. What are the guidelines needed to be followed while drawing use-case models.** 10 CO3 L3



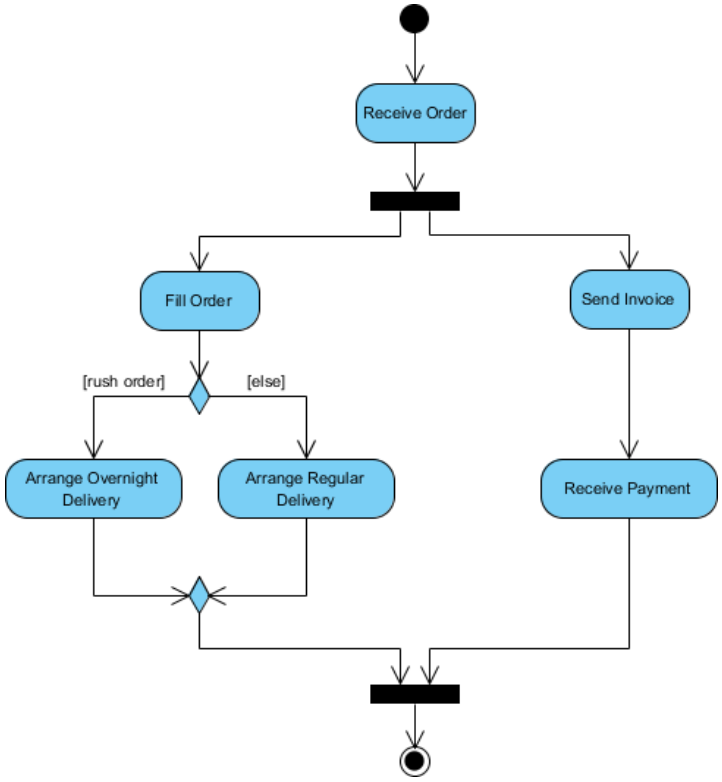
Guidelines for use case models

Determine the system boundary
Ensure that actors are focused
Each use case must provide value to user
Relate use case and actor
Remember that use cases are informal
Use cases can be structure

4. a **Explain activity diagram, with the UML notations. Given an example.** 5 CO3 L2

An activity diagram shows the sequence of steps that make up a complex process, such as an algorithm or workflow.

An activity diagram can show both sequential and concurrent flow of control.



b. **Mention the guidelines for activity models.** 5 CO3 L2

Don’t misuse activity diagrams

Level diagrams

Be careful with branches and conditions

Be careful with concurrent activities

Consider executable activity diagrams

PART-III

5 **Discuss the significance of state diagram and draw the state diagram for telephone line system.** 10 CO3 L3

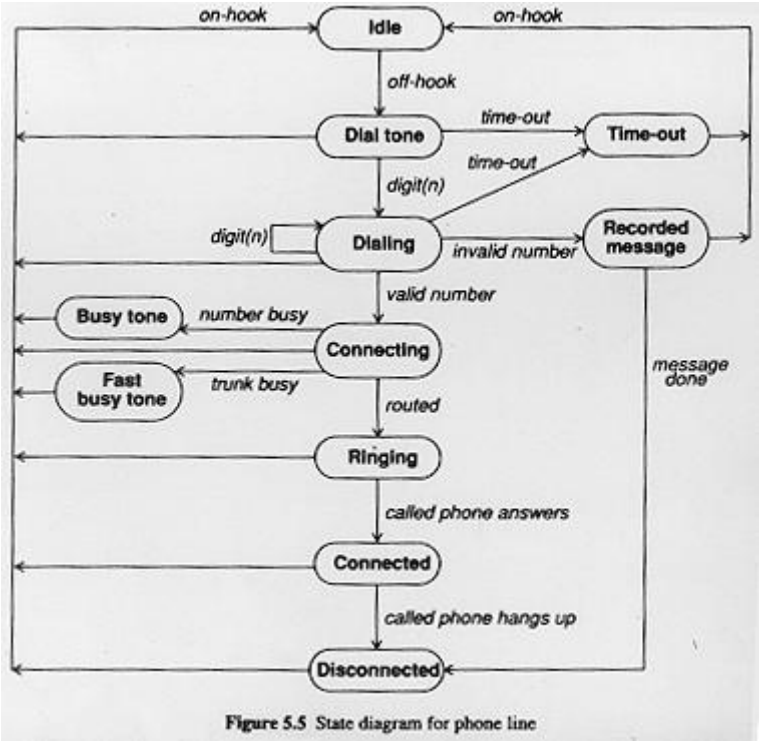
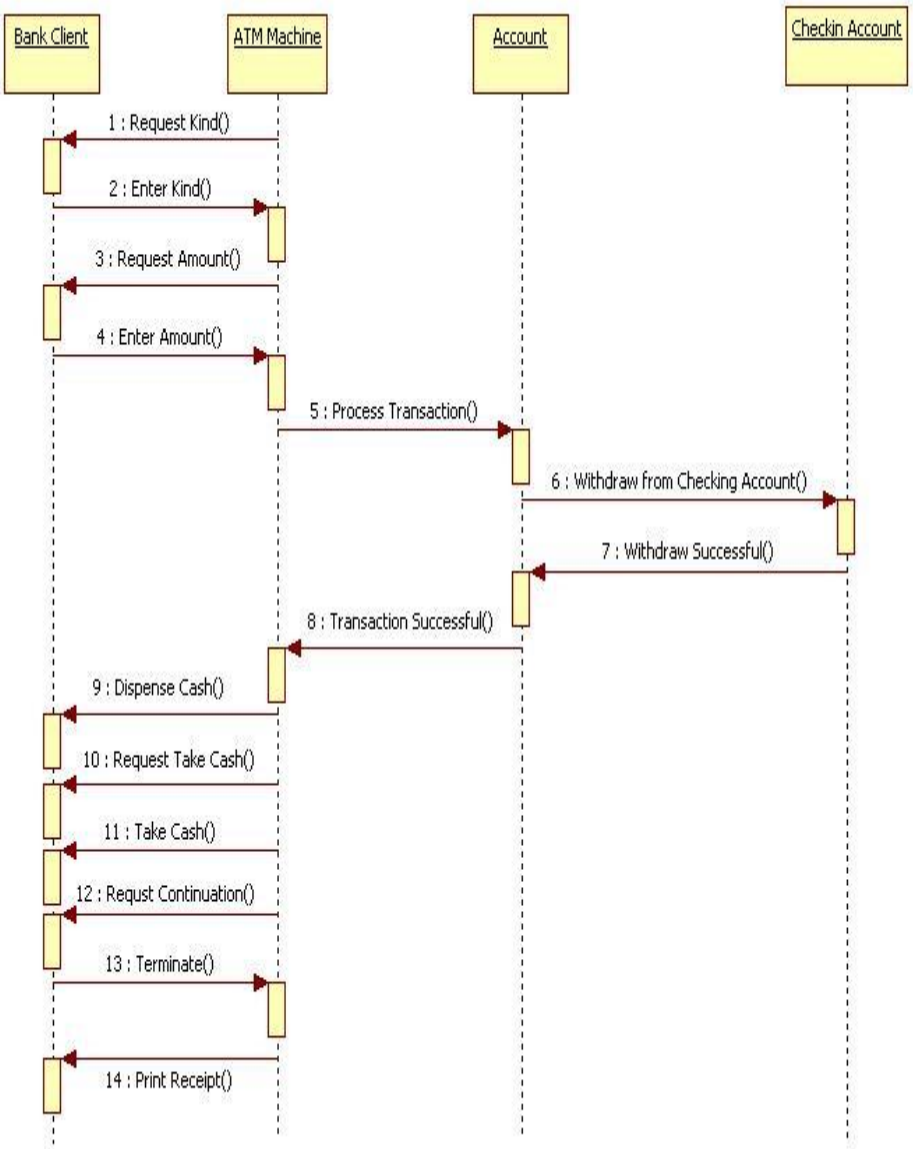


Figure 5.5 State diagram for phone line

6	**Write guidelines for sequence model and draw the sequence diagram for cash withdrawal process structure in ATM system**.	10	CO3	L3



Sequence Models

Prepare at least one scenario per use case

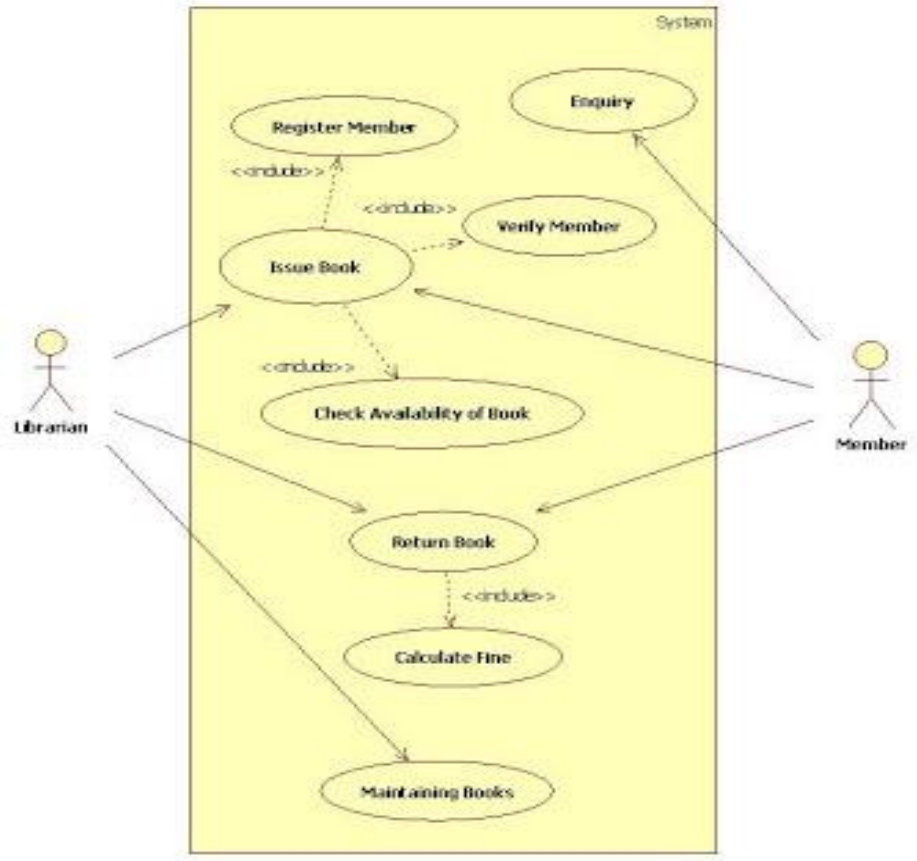
Abstract the scenarios into sequence diagrams

Divide complex interactions

Prepare a sequence diagram for each error condition

PART-IV

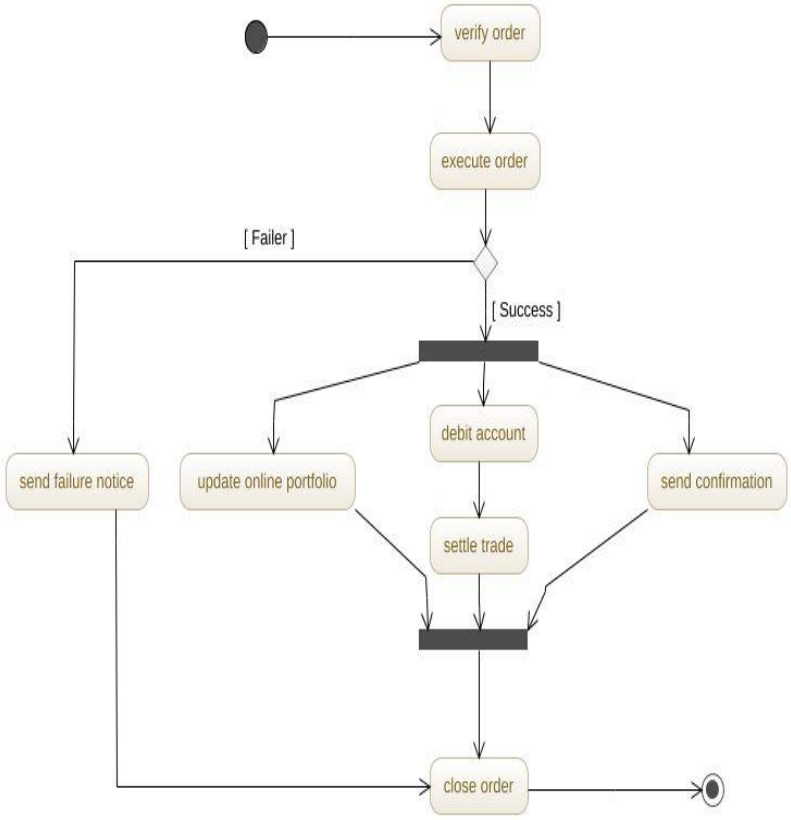
7	**Draw the use case diagram for library management system and describe the guidelines of use case diagram**	10	CO3	L3



Guidelines for use case models

Determine the system boundary
Ensure that actors are focused
Each use case must provide value to user
Relate use case and actor
Remember that use cases are informal
Use cases can be structure

8	**Draw an activity diagram for stock trade processing.**	10	CO3	L3



PART-V

9 **Explain the Whole part design pattern in detail** 10 CO5 L4

The Whole-Part design pattern helps with the aggregation of components that together form a semantic unit. An aggregate component, the Whole, encapsulates it constituent components, the Parts, organizes their collaboration, and provides a common interface to its functionality. Direct access to the Parts is not possible.

Context:

Implementing aggregate objects.

Problem:

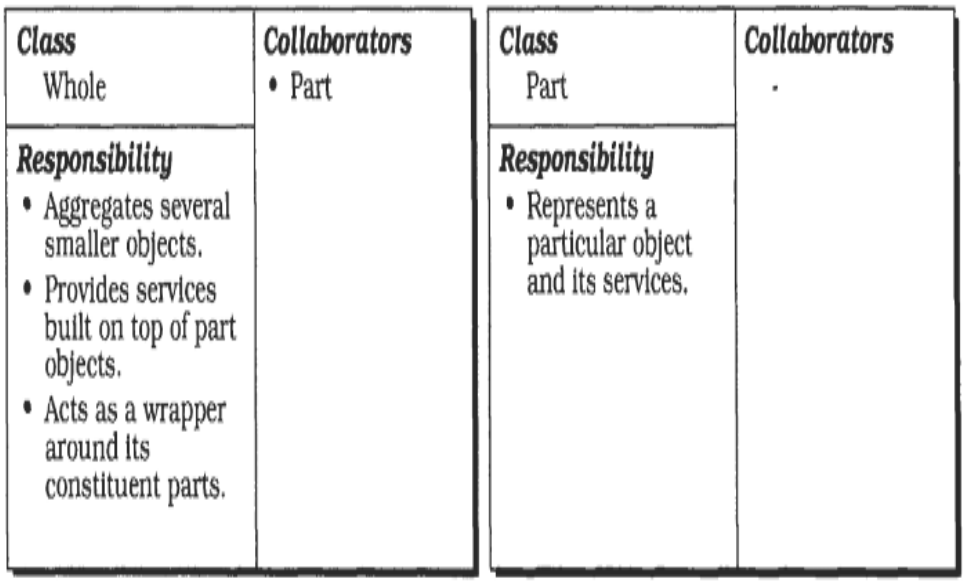
In almost every software system objects that are composed of other objects exist.
For example, consider a molecule object in a chemical simulation system.
In this example, a molecule object would have attributes such as its chemical properties, and methods, such as rotation.

These attributes and methods refer to the molecule as a semantic unit, and not to the individual atoms of which it is composed.
The combination of parts makes new behavior emerge known as emergent behavior.

Solution:

- Introduce a component that encapsulates smaller objects & prevents clients from accessing these constituent parts directly.
- Define an interface for the aggregate that is the only means of access to the functionality of the encapsulated objects.
- Allow the aggregate to appear as a semantic unit

Structure:



10 **Briefly write about the structure and dynamics of Command Processor** 10 CO5 L4

Abstract command component defines the interface of all command objects.

A minimum of the interface is to provide a procedure to execute a command.

Additional services require further interface procedure for all command objects.

Each user function we derive a command component form the abstract command.

Class

- Abstract Command

Responsibility

- Defines a uniform interface to execute commands
- Extends the interface for services of the command processor, such as undo and logging

Collaborators

- -

Class

- Command

Responsibility

- Encapsulates a function request
- Implements interface of abstract command

- Uses suppliers to perform a request

Collaborators

- Supplier

Controller represents the interface of the application, it accepts request such as "paste text" and creates the corresponding command objects

Command objects then go to Command Processor for processing

In essense, Controller is like a server waiting for request to map to Command objects

Command Processor manages command objects, schedule them and starts their execution, it is independent of specific commands because it only uses the abstract command interface

Supplier components provide most of the functionality required to execute concrete commands

When an undo mechanism is required, a supplier usually provides a means to save and restore its internal state

Class

- Controller

Responsibility

- Accepts service requests
- Translates requests into commands
- Transfers commands to command processor

Collaborators

- Command Processor
- Command

Class

- Command Processor

Responsibility

- Activates command execution
- Maintains commands objects
- Provides additional services related to command execution

Collaborators

- Abstract Command

Dynamics

Implement an undo mechanism after Capitalizing some text

The controller accepts the request from the user within its event loop and creates a 'capitalize' command object

The controller transfers the new command object to the command processor for execution and further handling

The command processor activates the execution of the command and stores it for later undo

The capitalize command retrieves the currently-selected text from its supplier, stores the text and its position in the document, and asks the supplier to actually capitalize the selection

After accepting an undo request, the controller transfers his request to the command processor. The command processor invokes the undo procedure of the most recent command

The capitalize command resets the supplier to the previous state, by replacing the saved text in its original position

If no further activity is required or possible of the command, the command processor deletes the command object.