Modeling as a Design Technique

A model is an abstraction of something for the purpose of understanding it before building it. Because a model omits nonessential details, it is easier to manipulate than the original entity. Abstraction is fundamental human capability that permits us to deal with complexity. Engineers, artists, and craftsmen have built models for thousands of years to try out designs before executing them. Development of hardware and software systems is no exception. To build complex systems, the developer must abstract different views of the system. Build models using precise notations, verify that the models satisfy the requirements of the system, and gradually add details to transform the model into an implementation

Three type Models

We model a system from three related but different viewpoints, each capturing important aspects of the system, but all required for a complete description. The class model represents the static, structural, "data" aspects of a system. The state model represents the temporal, behavioral, "control" aspects of a system. The interaction model represents the collaboration of individual objects, the "interaction" aspects of a system.

A typical software procedure incorporates all three aspects:
It uses data structures(Class Model), it sequences operations in time (state model), and it passes data and control among objects(interaction model). Each model contain references to entities in other models.

For example, class model attaches operations to classes, while the state and interaction models elaborate the operations.

The three kinds of models separate a system into distant views. The different models are not completely independent- A system is more than a collection of independent parts-but each model can be examined and understood by itself to a large extent.

Interaction Model

Interaction can be modeled at different levels of abstraction.

1) Use case Diagram
2) Sequence diagram
3) Activity diagram

Use Case Model

What is a Use Case?

A formal way of representing how a business system interacts with its environment .Illustrates the activities that are performed by the users of the system. A scenario-based technique in the UML. A sequence of actions a system performs that yields a valuable result for a particular actor.

What is an Actor?

A user or outside system that interacts with the system being designed in order to obtain some value from that interaction.

Use Cases describe scenarios that describe the interaction between users of the system (the actor) and the system itself.

Use case diagrams describe what a system does from the standpoint of an external observer. The emphasis is on what a system does rather than how. Use case diagrams are closely connected to scenarios. A scenario is an example of what happens when someone interacts with the system.

How to Draw Use Case Diagram?

Step 1 Identify the actors

As we read the scenario, define those people or systems that are going to interact with the scenario.

An Actor is outside or external the system.

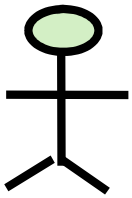
It can be a:

• Human

• Peripheral device (hardware)

• External system or subsystem

• Time or time-based event



Represented by stick figure

Step 2 Identify the use case

Each use case in a use case diagram describes one and only one function in which users interact with the system

May contain several “paths” that a user can take while interacting with the system • Each path is referred to as a scenario

Labelled using a descriptive verb-noun phrase

Represented by an oval

Use Case - Relationships

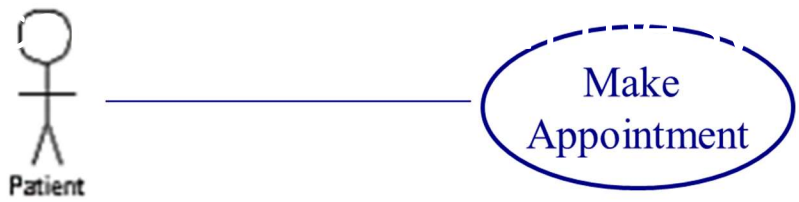
Relationships

Represent communication between actor and use case

-

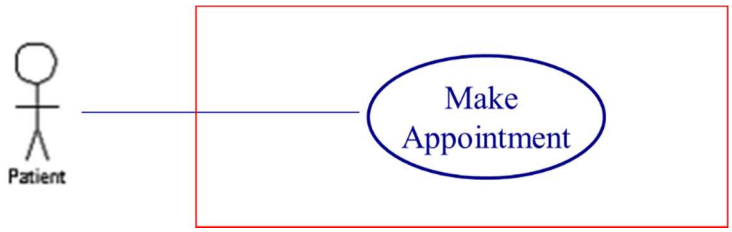
Depicted by line or double-headed arrow line

Also called association relationship



Boundary

A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.



Other Types of Relationships for Use Cases

1) Generalization
2) Include
3) Extend

Generalization Relationship

Represented by a line and a hollow arrow

From child to parent

Include Relationship

Represents the inclusion of the functionality of one use case within another

Arrow is drawn from the base use case to the used use case

Write << include >> above arrowhead line

Extend relationship

Represents the extension of the use case to include optional functionality

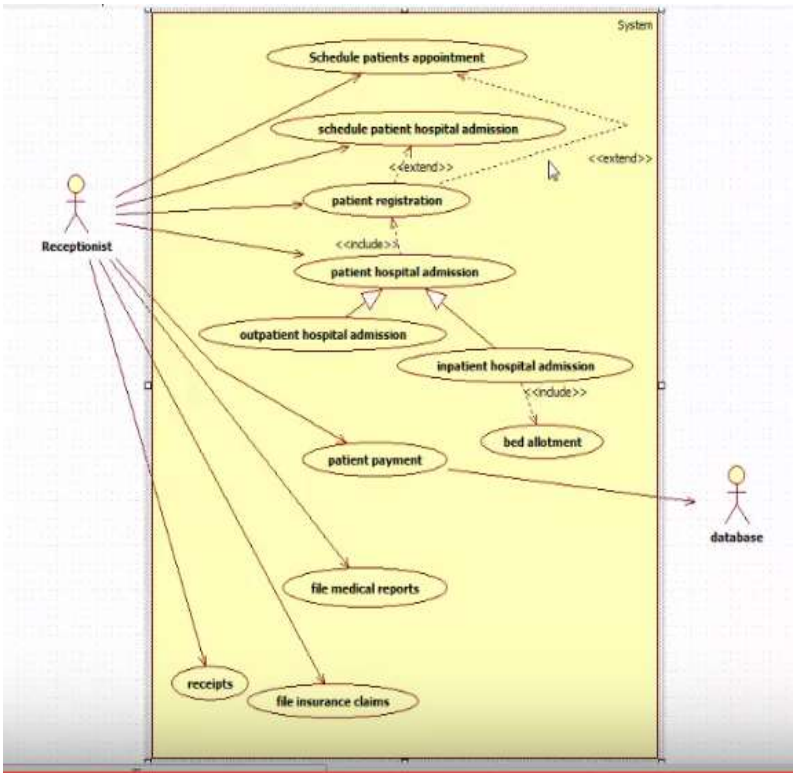
Arrow is drawn from the extension use case to the base use case

Write << extend >> above arrowhead line

Practical No. 1

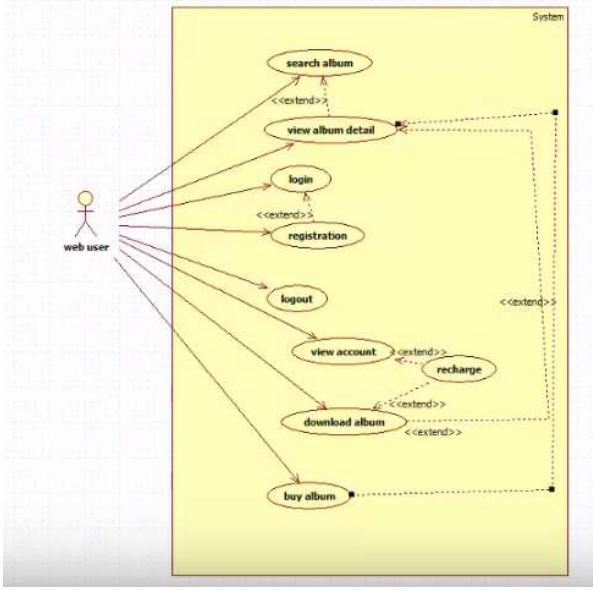
Draw a **use-case diagram** using StarUML for the "Hospital Reception Subsystem" explained below:

"Hospital Reception Subsystem" supports some of the many job duties of hospital receptionist. Receptionist schedule patient's appointments with the doctor and also schedule patient hospital admission. If doctor is available and admission to the hospital is possible then receptionist can extend the service to patient registration by collecting the patient information on patient arrival or over the phone. Patient registration is an integral part of patient Hospital Administration use case. Hospital administration use case is further generalized into outpatient hospital admission and inpatient hospital administration. Note that for the patient that will stay in the hospital, he or she should have a bed allotted in a ward. Receptionists might also receive patient's payments, record the min a database and provide receipts, file insurance claims and medical reports.



Practical No. 2

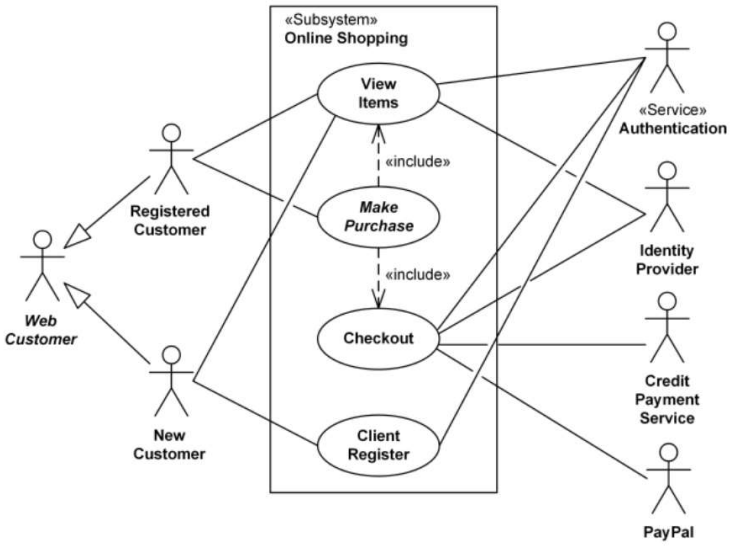
Draw a use-case diagram using staruml for the "music portal system" depicted below. The following narration describes some of the use cases for "**Music Portal System**". This system has **web user** as its main actor. The web user can perform first level uses cases namely **SearchAlbum, login, logout** and **ViewAccount**. **Registration** use case extends login i.e. if the user doesn't have a login and wishes to create a new one, he or she can register to get a login. Moreover the **ViewAlbumDetail** use case is extending the SearchAlbum use case. Further the ViewAlbumDetail use case is extended by two more services viz**. DownloadAlbum** and **BuyAlbum**. Note that to download or to buy an album the user must be a registered member. Moreover the buyalbum and ViewAccount is further extended with R**echarge** use case**.**



Practical No. 3

Draw a **use-case** diagram using StarUML for the scenario given below.
Web Customer actor uses some web site to make purchases online. Top level **use cases** are **View Items**, **Make Purchase** and **Client Register**. View Items use case could be used by customer as to level use case if customer only wants to find and see some products. This use case could also be used as a part of Make Purchase use case. Client Register use case allows customer to register on the web site, for example to get some coupons or be invited to private sales. Note that **Checkout** use case is **included use case** not available by itself - checkout is part of making purchase. Except for the **Web Customer** actor there are several other actors which will be described below with detailed use cases. **View Items** use case is **extended** by several optional use cases - customer may search for items, browse catalog, view items recommended for him/her, add items to shopping cart or wish list. All these use cases are extending use cases because they provide some optional functions allowing customer to find item. **Customer Authentication** use
Case is **included** in **View Recommended Items** and **Add to Wish List** because both require customer to be authenticated. At the same time, item could be added to the shopping cart without user authentication. **Checkout** use case includes several required uses cases. Web customer should be authenticated. It could be done through user login page, user authentication cookie ("Remember me") or Single Sign-On (SSO). Web site authentication service is used in all these use cases, while SSO also requires participation of external identity provider. **Checkout** use case also includes **Payment** use case which could be done either by using credit card and external credit payment service or with PayPal.

(A) TopLevel

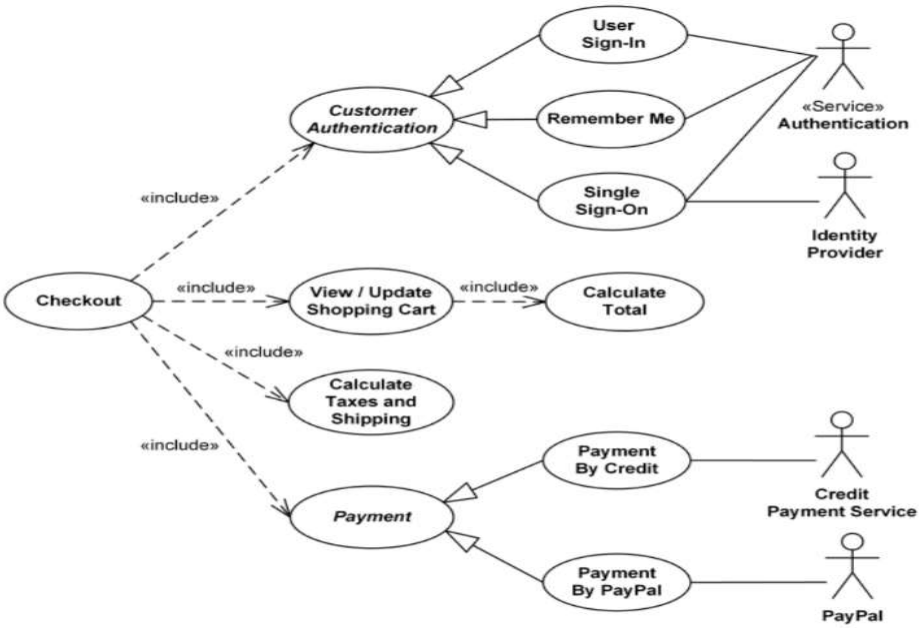


Online shopping UML use case diagram example - top level use cases.

(A) ViewItems

Online shopping UML use case diagram example - view items use case.

(B) Checkout, authentication and payment use cases.



Online shopping UML use case diagram example - checkout, authentication and payment use cases.

Practical No. 4

DRAW THE ACTIVITY DIAGRAM FOR THE GIVEN PROBLEM OF USE CASE.

DESCRIPTION OF THE EXAMINATION PAPER PREPARATION SUPPORT SYSTEM.

Use case name: submit question

Participant: lecturer

Entry conditions:

1. The question is ready and stored in a file
2. The lecturer is assigned to the module

Exit conditions:

1. The file is uploaded to the system
2. The module leader is notified of the availability of the question
3. The event is logged by the system

Flow of Events:

1. The lecturer logs into the system by entering his/her username and password;
2. The system checks the username and password;
3. The system displays the list of modules of which he/she is the lecturer, module leader and/or internal examiner;
4. The lecturer selects a module and his/her role in the module as a lecturer;
5. The system prompts the user to enter the file name and location on his/her computer, and Additional information if any;

Exceptional conditions and alternative flow of events:

When the username and password is not correct:

3.1: display error message, go back to step 1;

When the lecturer is not listed on the module:

4.1: quit the system;

Special requirements:

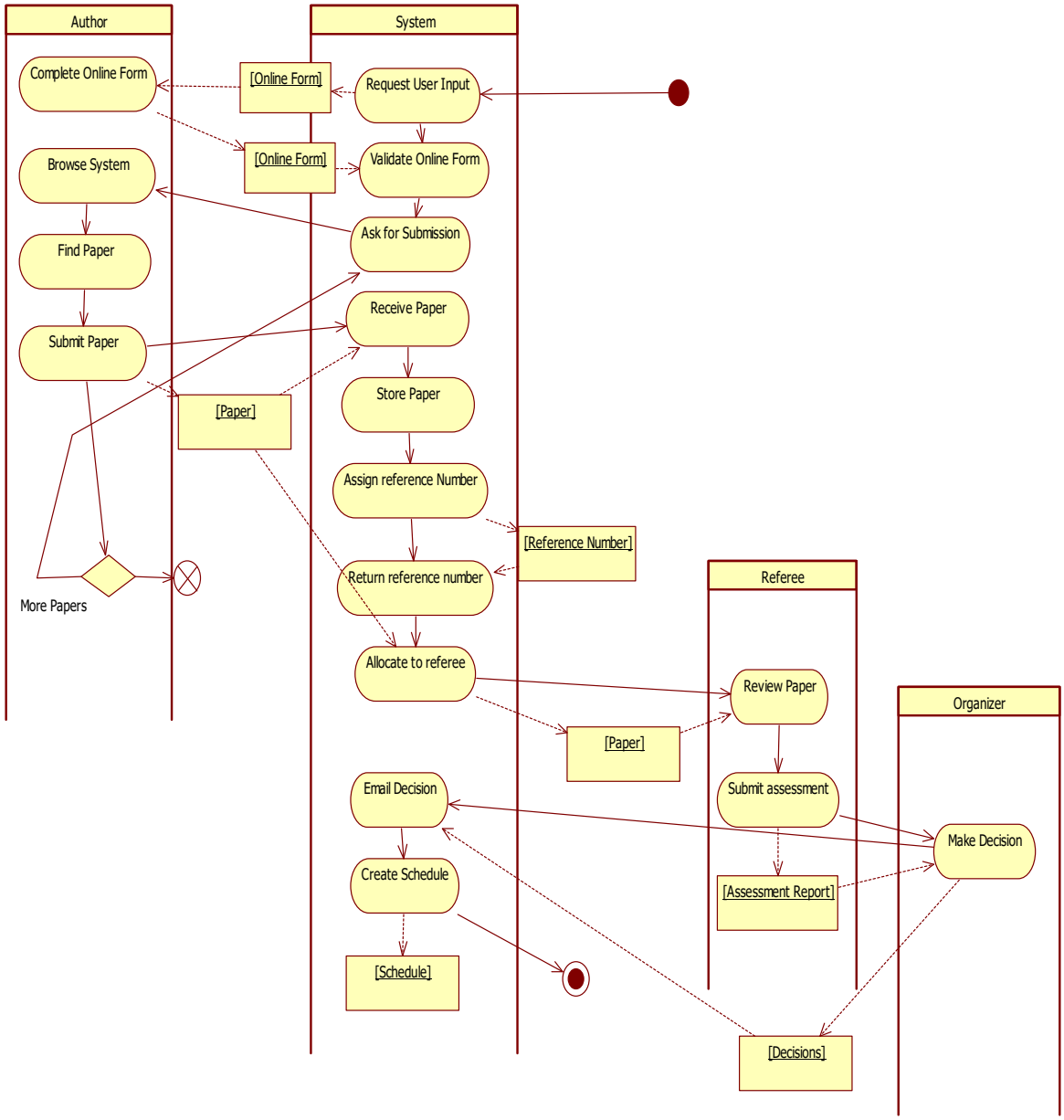
1. The file should be encrypted when transmitted from lecturer's computer to the server
2. The notification of success in uploading the file should be within 20 seconds
3. The event should be recorded in a log file to contain the following information:

a) Name of the lecturer,

b) Date and time of the event,

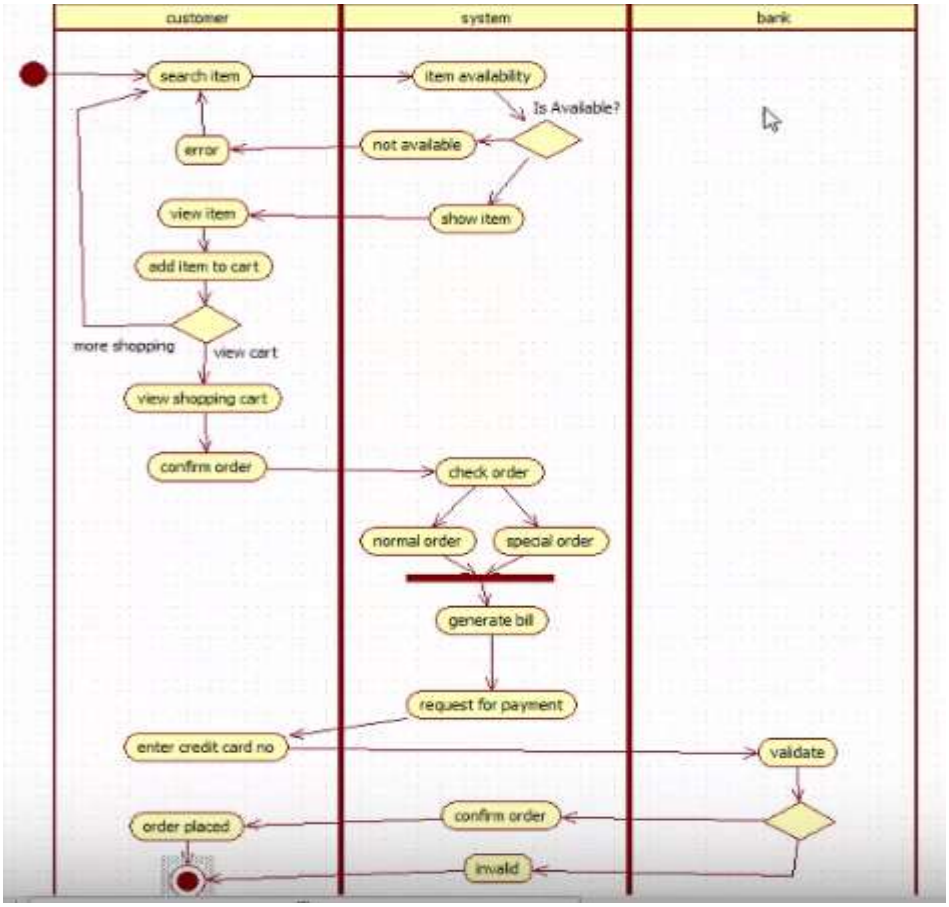
c) The name of the event (upload exam question).

d) The file on the server that stores the questions.


Author
System
Complete Online Form
[Online Form]
Request User Input
[Online Form]
Validate Online Form
Browse System
Ask for Submission
Find Paper
Receive Paper
Submit Paper
Store Paper
[Paper]
Assign reference Number
[Reference Number]
Return reference number
More Papers
Referee
Allocate to referee
Review Paper
Organizer
[Paper]
Email Decision
Submit assessment
Make Decision
Create Schedule
[Assessment Report]
[Schedule]
[Decisions]

Practical No. 5

Derive an activity diagram using staruml from the case given below on "order processing subsystem". Web **Customer** uses some web site to make purchases online. Where customer can search item, View item, **add item to the cart, place order and make payment.** For placing an order customer first searches the required items from the system. As and when the customer finds the item available in the system he starts adding item to the chart. The System provides facility to the Customer to add any number of items to the chart. Customer can also view his shopping chart containing items. Ones the customer finishes his shopping he can place the order by requesting system to confirm the order. The system will then check whether the order is normal order or any special order and according to that the system will generate the bill and request for payment. After getting the bill the customer can make payment. The bank will validate the credit card number. If the credit card number is valid the system will confirm the order. Otherwise the process will get terminated.



Practical No. 6

Draw an **activity diagram** using StarUML from the narrative text on **"Bank ATM Machine For Withdrawing Cash"**.

Summary:

An automated teller machine (ATM) or the automatic banking machine (ABM) is a banking subsystem that provides bank customers with access to financial transactions in a public space without the need for a cashier, clerk or bank teller.

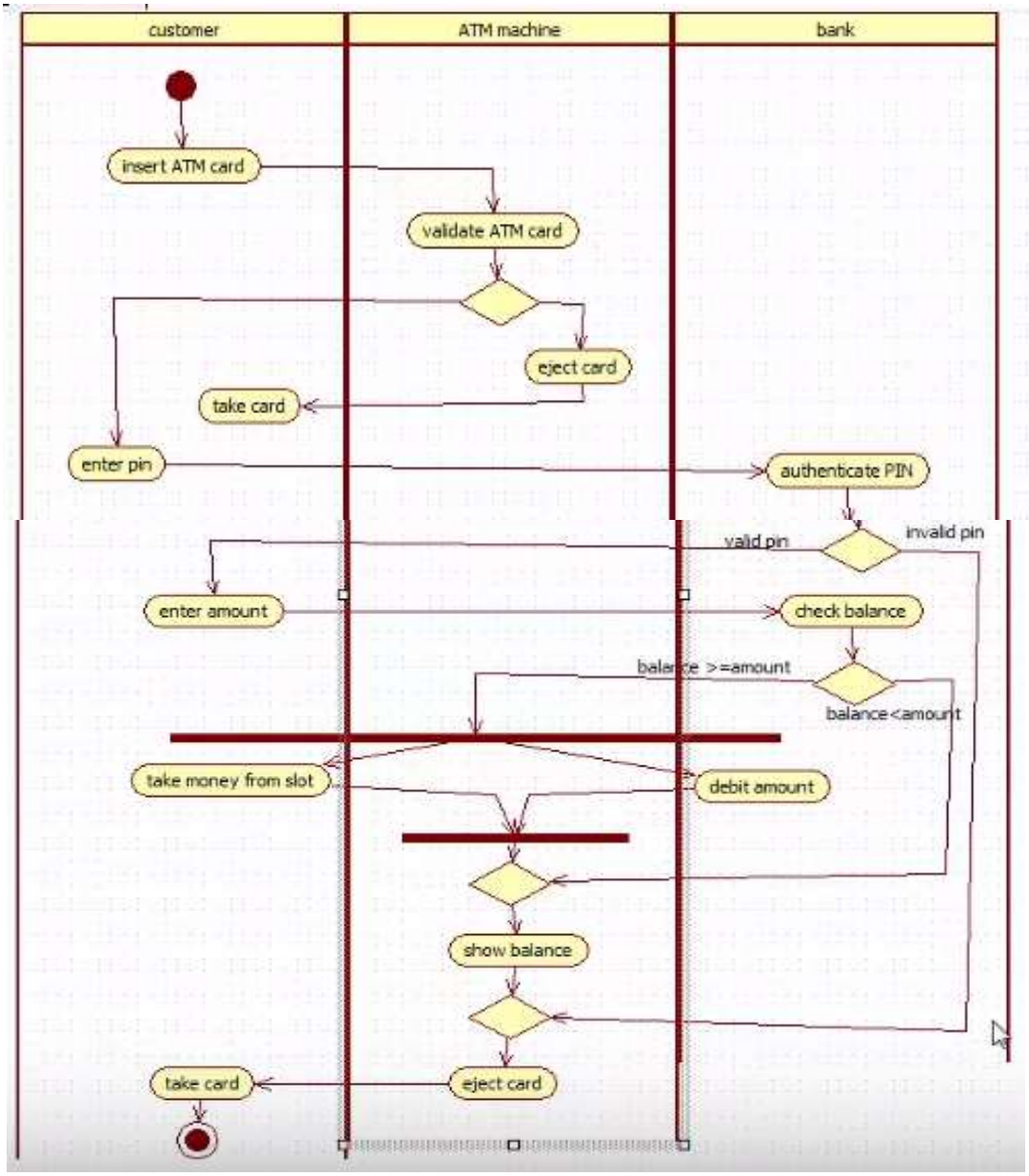
Customer uses bank ATM to check balances of his/her bank deposit funds, withdraw cash and/or transfer funds which generalization alternative of ATM transaction use case.

FOR WITHDRAWING CASH

On most bank ATMs, the customer is authenticated by inserting a plastic ATM card and entering a personal identification number (PIN). Bank will than authenticate the customer's pin number. Only authenticated customer can request the system for withdrawing money while the unauthenticated customer will get back his ATM card as the system willreject the card.

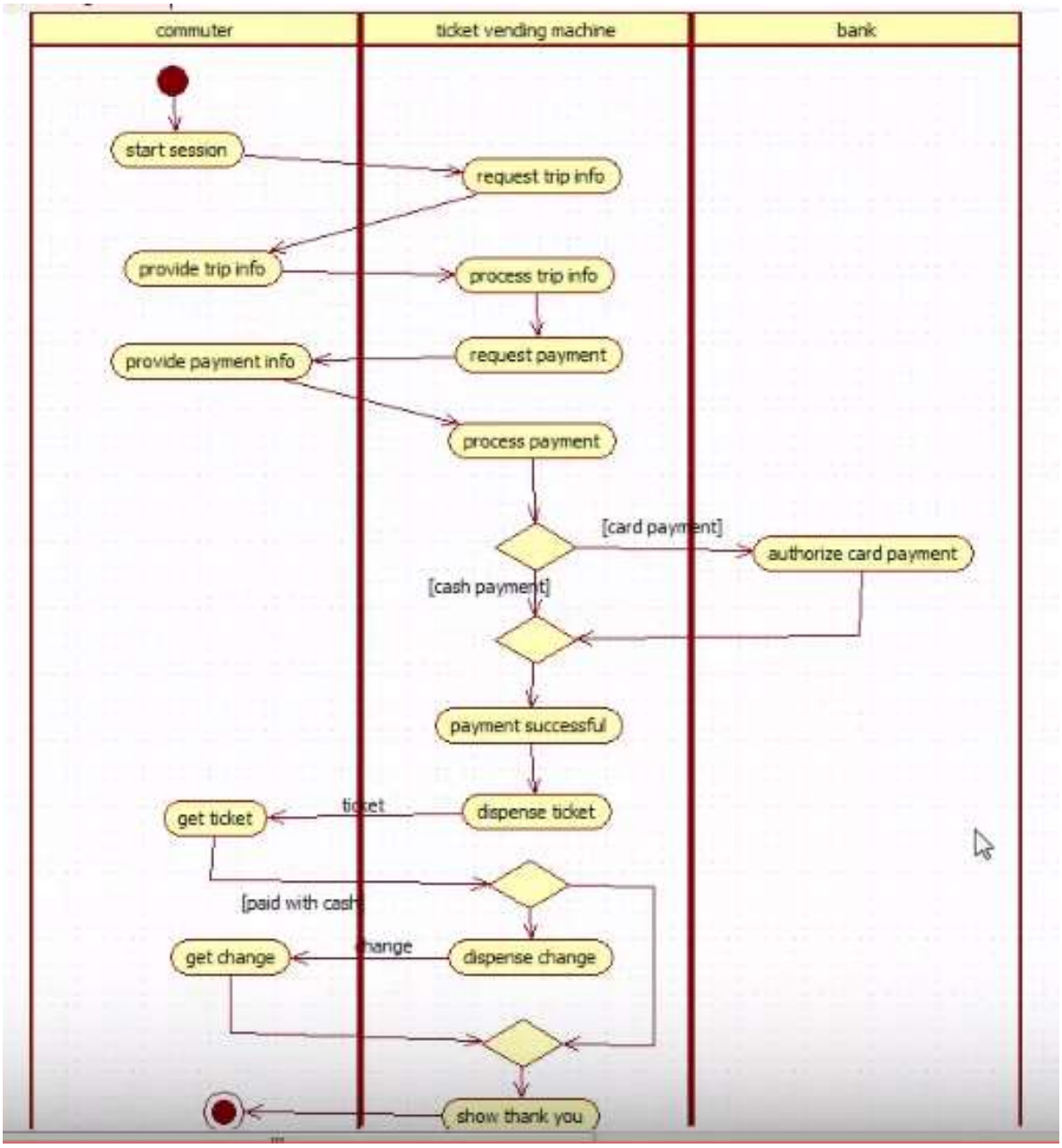
Then the system will request the authenticated customer to enter the amount be de withdrawn. The bank will check the balance amount of the customer if it is sufficient bank will provide the requested amount to the customer and debit the respective amount from the balance. The customerwill collect or take the amount from the slot. In case of insufficient amount the system will show the balance and reject the card. At the end of all the process the customer will take back his ATM card.

Answer :-


customer
ATM machine
bank
insert ATM card
validate ATM card
eject card
take card
enter pin
authenticate PIN
valid pin
invalid pin
enter amount
check balance
balance >=amount
balance<amount
take money from slot
debit amount
show balance
take card
eject card

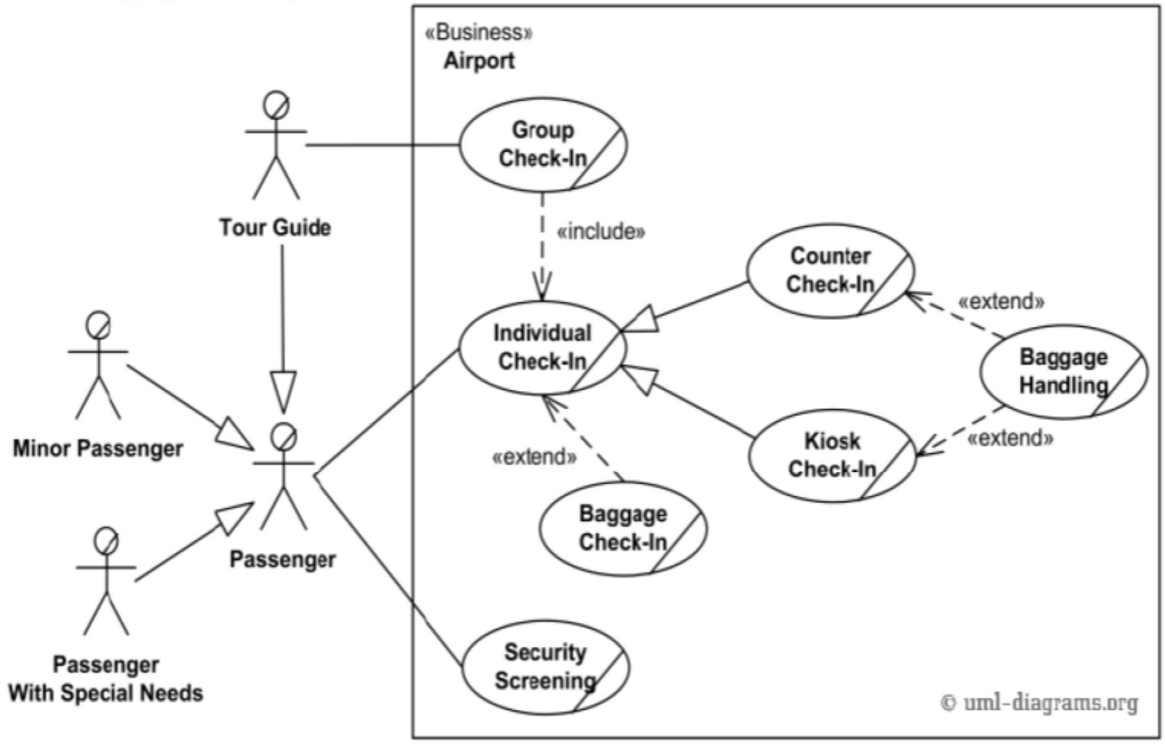
Practical No. 7

Draw an activity diagram using from the narrative text on **"ticket vending machine"** The scenario provided below describes the behavior of the purchase ticket use case. Activity is **started** by Commuter actor who needs to buy a ticket. Ticket vending machine will **request trip information** from Commuter. The commuter will **provide trip information** to the machine. Based on the provided trip information, the ticket vending machine will **process the trip information** and calculate payment due by **requesting payment**. Commuter will **provide payment** information to the machine. The machine will **process the payment** on the basis of payment by cash or credit or debit card. If payment by card was selected by Commuter, another actor, Bank will participate in the activity by authorizing the payment. After payment is complete, ticket is dispensed to the Commuter. And the commuter will get the ticket. Cash payment might result in some **change due**, so the change is dispensed to the Commuter in this case by the machine. Ticket vending machine will show some "Thank You" screen at the end of all the activity.

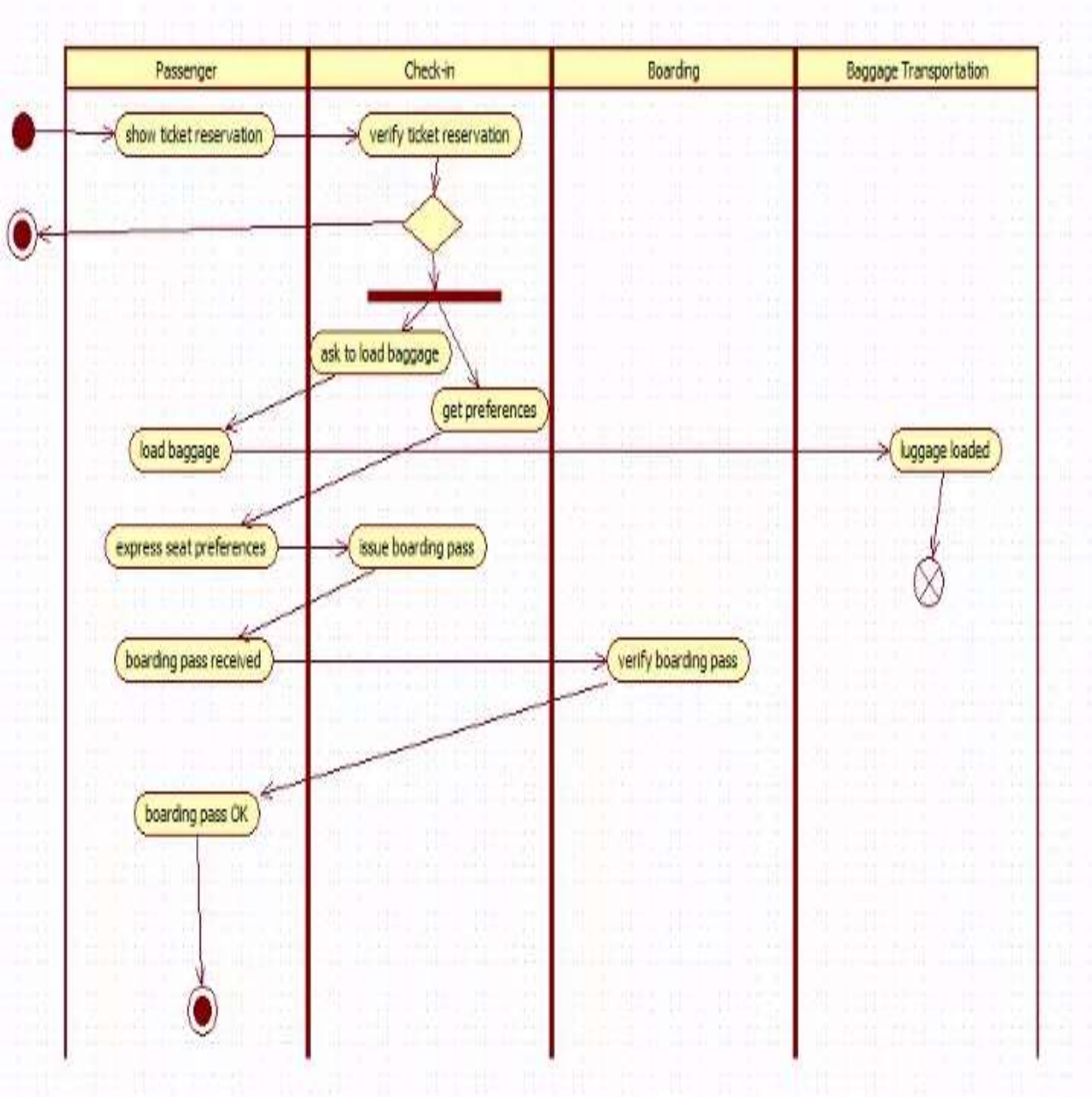

commuter
ticket vending machine
bank
start session
request trip info
provide trip info
process trip info
request payment
provide payment info
process payment
[card payment]
authorize card payment
[cash payment]
payment successful
dispense ticket
get ticket
[paid with cash]
dispense change
get change
show thank you

Practical No. 8

Draw an activity diagram using StarUML for the Airport check-in and security screening system depicted below.

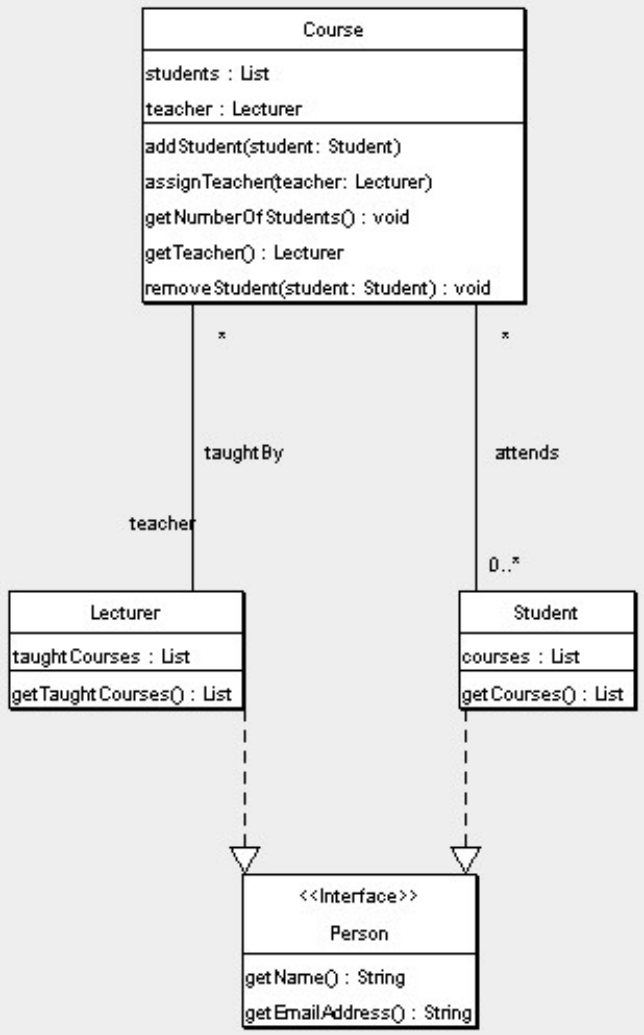


Answer:-


Passenger
Check-in
Boarding
Baggage Transportation
show ticket reservation
verify ticket reservation
ask to load baggage
get preferences
load baggage
luggage loaded
express seat preferences
issue boarding pass
boarding pass received
verify boarding pass
boarding pass OK

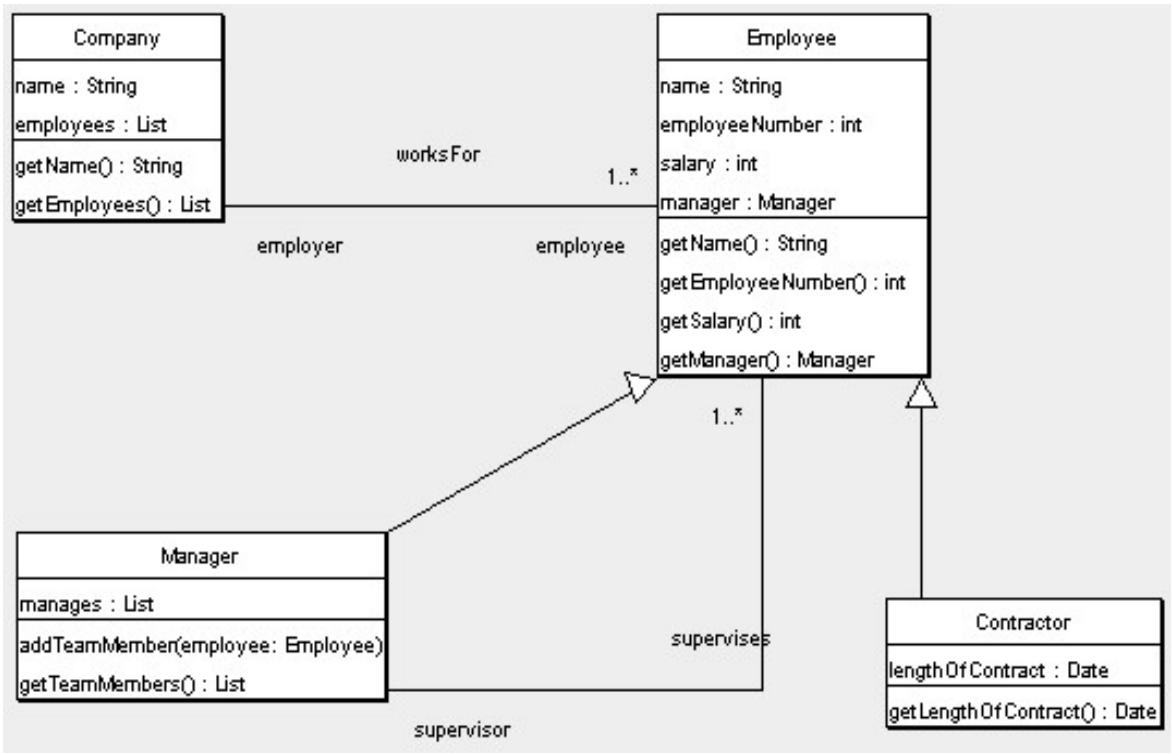
Practical No. 9

Draw a **class diagram** using StarUML for the scenario given below.
This is an example that **models University Courses**. Assume three classes' such as course, lecturer, student and an interface person.Each course objects maintains a list of student on that course and lecturer who has been assigned to teach that course. The courseobject has behavior that allows adding and removing student to and from course, assigning the teacher and getting a list of currentlyassigned student and currently assigned teacher. A teacher may teach several courses but a course only has a single teacher .A lecturer object maintains a list of courses that it teaches, course isattended by 0 or more student and student may attend multiple courses. A person interface will have getName() and getEmailAddress () methods both lecturer and student are shown tobe the type of person.



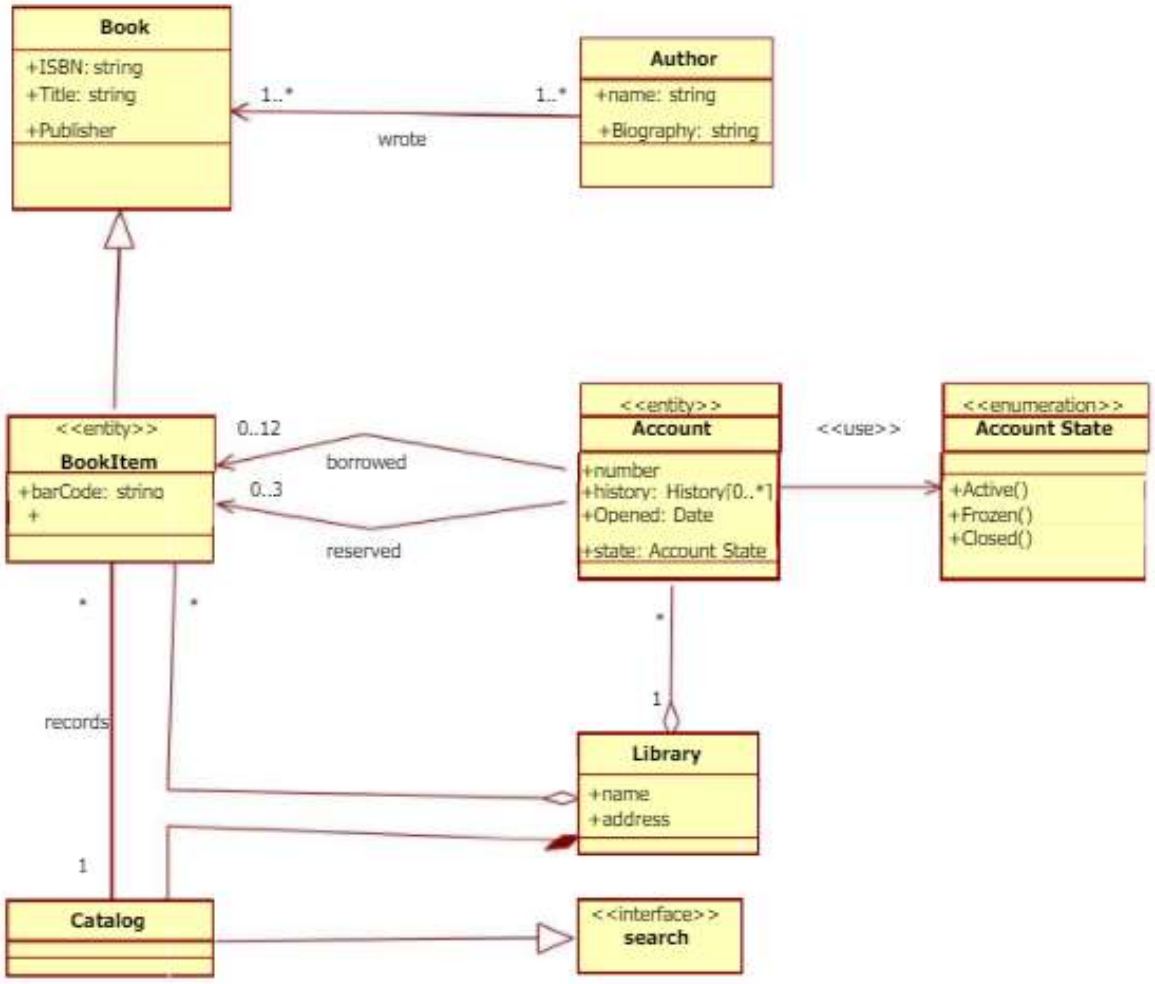
Practical No. 10

Draw a **class diagram** using StarUML for the scenario given below.
This scenario is from system that models companies for a payroll or reporting system. Company object has properties such as name and employees_list and getName and getEmployees as its behavior. Employee object includes employee no, name, salary and manager as its properties getName (), getEmplyoeeNo () ,getSalary() getManager() as its methods. getManager() accepts object of manager. Company may have one or more employees. A manager object keeps manages as list property and add TeamMember(employee_list) and getTeamMember() as itsbehaviors. One or more employee can be managed by manager objects. Some employees are contractual employees who are within a lieu of a contractor object. A contractor object may have length_of _contract as its property and getLength() as its behavior.



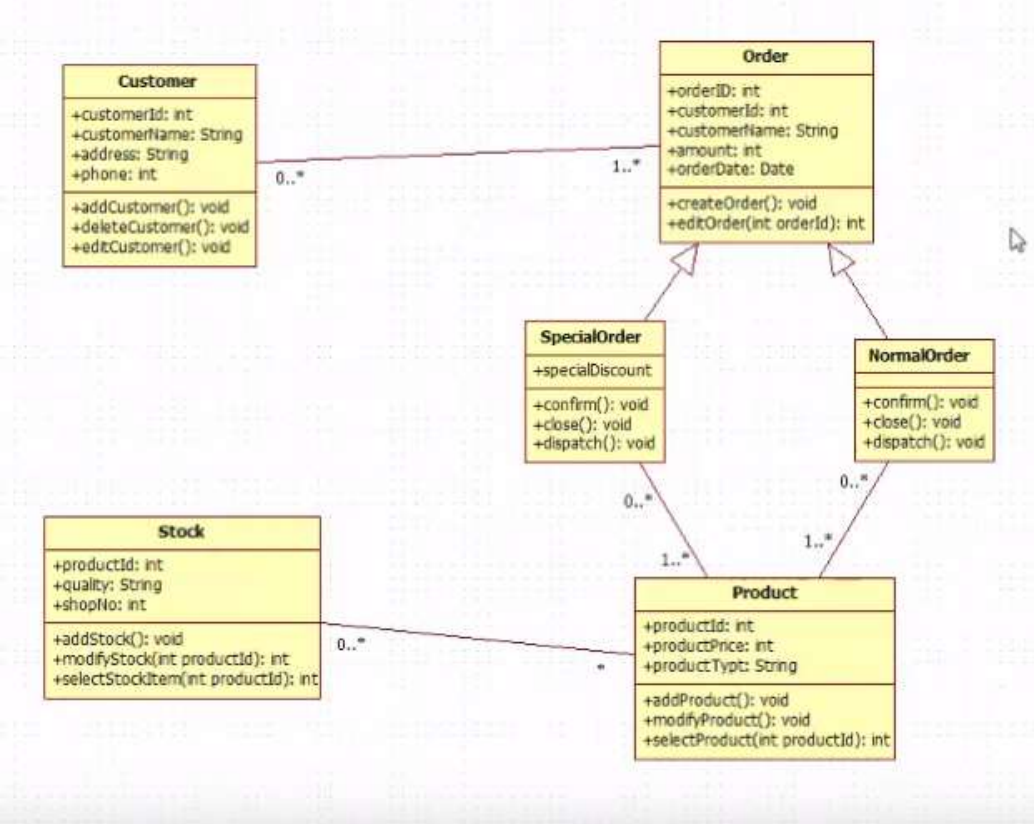
Practical No. 10

Create a **class diagram** (Use StarUML) for "library management" using the classes with their attributes and operation given below. Also set the appropriate relationship between the classes using the relationship tools from the toolbox following the overview of the system given below. Overview of the system:- It has a class "Book". Book has authors so it has an "Author" class. In order to collect book information it has "BookItem" class which uses some of the properties from book class. It needs an account for reserving book by the user so it has an "Account class." In account class there is an attribute named state which uses an enumeration named "AccountState". It also has a class "Library" to manage the account, user and the books. It has a user class to manage the user detail that has an account in the library and he can borrow and return books to library. The system also has an interface "Search" where the user searches the book he needed from the "Catalog" class.


Book
+ISBN: string
+Title: string
+Publisher
1..*
1..*
wrote
Author
+name: string
+Biography: string
<<entity>>
BookItem
+barCode: string
+
0..12
borrowed
0..3
reserved
<<entity>>
Account
+number
+history: History[0..*]
+Opened: Date
+state: Account State
<<use>>
<<enumeration>>
Account State
+Active()
+Frozen()
+Closed()
*
*
records
1
*
1
Library
+name
+address
1
Catalog
<<interface>>
search

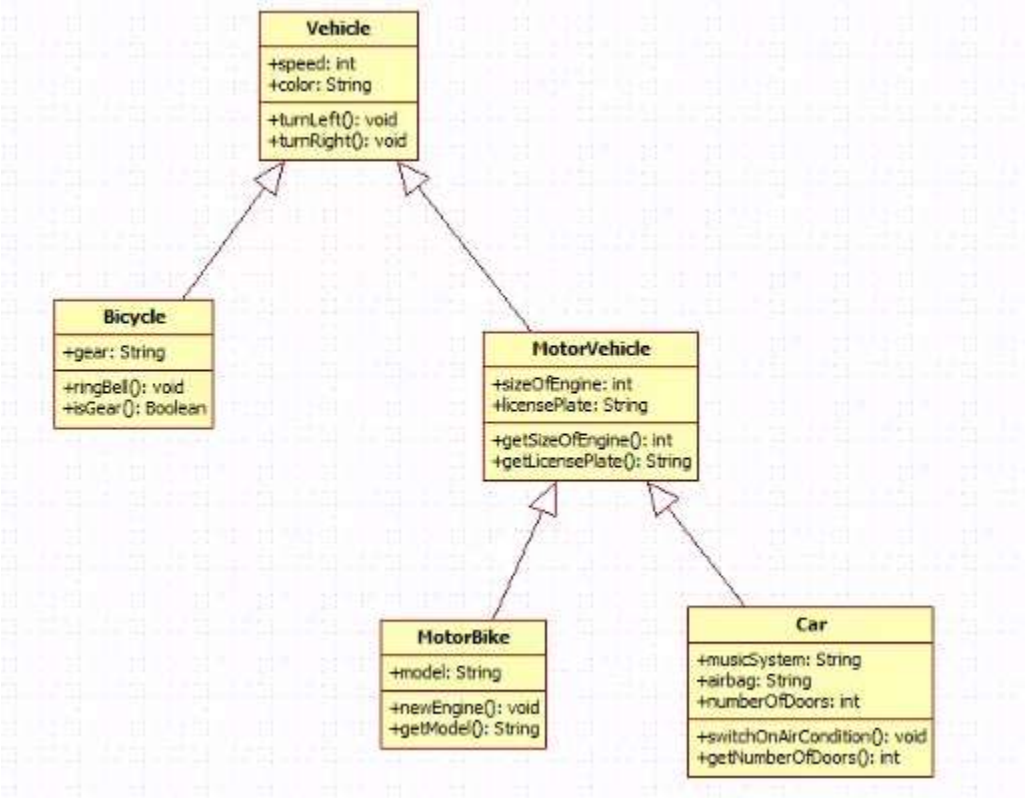
Practical No. 11

Draw a **class diagram** using StarUML for the scenario given below: This is an example that models **"ORDER MANAGEMENT"**. The Customer object has properties such as CustomerId,CustomerName, Address and Phone and methods such as AddCustomer(),DeleteCustomer() and EditCustomer(). Orderobject includes OrderId, CustomerId, CustomerName, ProductId, Amount and OrderDate as its property and CreateOrder() and EditOrder(OrderId) as its behavior. A customer can place one or many orders. Further there are SpecialOrder object andNormalOrder object which have same methods CreateOrder(), confirm(), close(), dispatch() whereas the SpecialOrder object alsohas one property named SpecialDiscount. **Special Order and Normal Order** objects are both kinds of order and are thereforeshown to inherit from order entity. Moreover the system also has Product entity having attributes such as ProductId, ProductPrice, ProductType and methods such as AddProduct(), ModifyProduct() and SelectProduct(ProductId). Stock object has properties likeProductId, Quality and ShopNo and behavior such as addStock(), ModifyStock(ProductId) and slectStockItem(ProductId). Note that specialOrder and NormalOrder has 1 or more product whereas stock has many products.



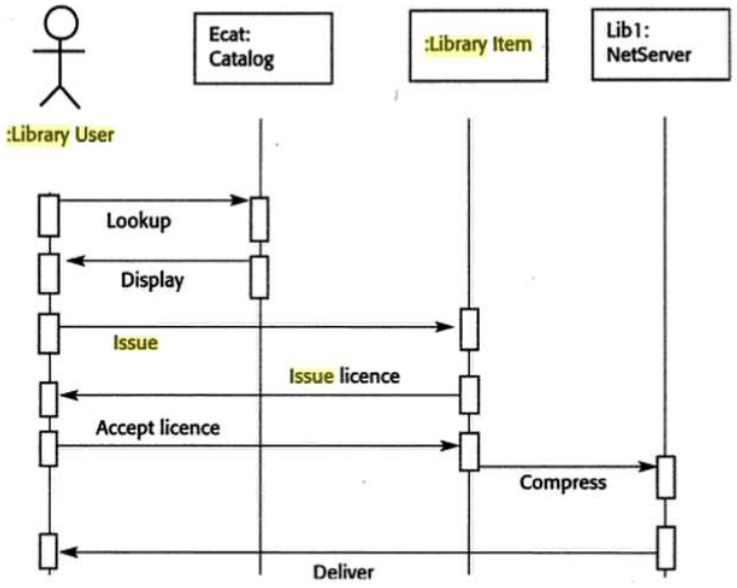
Practical No. 12

Draw a **class diagram** using StarUML for the scenario givenbelow.
This scenario shows an inheritance hierarchy of a series of classes and their subclasses. It's for an imaginary application that mustmodel different kinds of vehicles such as bicycles, motor bike and cars. All Vehicles have some common attributes (speed and color)and common behavior (turnLeft, turnRight). Bicycle and MotorVehicle are both kinds of Vehicle and are therefore shown toinherit from Vehicle. To put another way, Vehicle is the superclass of both Bicycle and MotorVehicle. In our model MotorVehicleshave engines and license plates. Attributes have been added accordingly, along with some behavior that allows us to examinethose attributes. MotorVehicles is the base class of both MotorBike and Car; therefore these classes not only inherit the speed and colorproperties from Vehicle, but also the additional attributes and behavior from MotorVehicle. Both MotorBike and Car haveadditional attributes and behavior which are specific to those kinds of object.



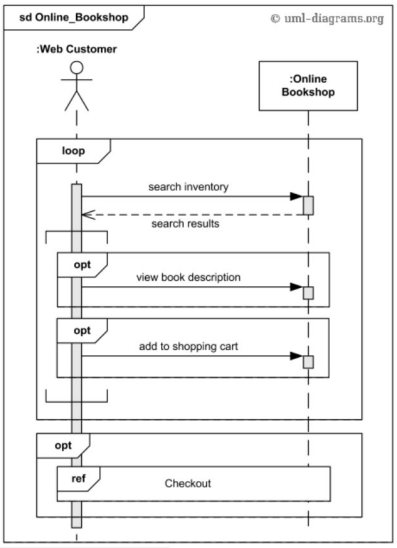
Practical No. 13

Draw a **sequence diagram** for the following scenario, the library user accesses the catalogue to see whether the item required is available electronically; if it is, the user requests the electronic issue of that item. For copyright reasons, this must be licensed so there is a transaction between the item and the user where the license is agreed. The item to be issued is then sent to a network server object for compression before being sent to the library user.



Practical No. 14

Construct a high level **sequence** diagram for online bookshop. Online customer can search book catalog, view description of a selected book, add book to shopping cart, do checkout.



Practical No. 15

Draw a **Sequence** diagram for the following scenario which shows interactions involved in using LIBSYS for downloading and printing an article. There are four objects of classes- Article, Form, Workspace and Printer--involved in this interaction. Essentially, a user request for an article triggers a request for a copyright form. Once the user has completed the form, the article is downloaded and sent to the printer. Once printing is complete, the article is deleted from the LIBSYS workspace.