The Unified Modeling Language (UML)

- It is a standardized general-purpose graphical language for modeling object-oriented software.

- This was developed in 1990’s by Object Management Group.

- It combines the ideas of Rumbaugh, Booch and Jacobson and hence the name ‘Unified’ modeling language.

- Programmers, software architects, and analysts use modeling languages such as UML to graphically describe the design of a software.

- The UML defines a variety of diagrams such as class diagrams, use-case diagrams, interaction diagram, statechart diagrams, activity diagrams etc.

- This language is sufficiently general to be used in all software engineering domains.

UML Class Diagrams

- It gives an overview of a system by showing its classes and the relationships among them.

- These diagrams show the static structure of the model.
- The main symbols shown on a class diagram are:

  Classes

  Associations

  Attributes

  Operations

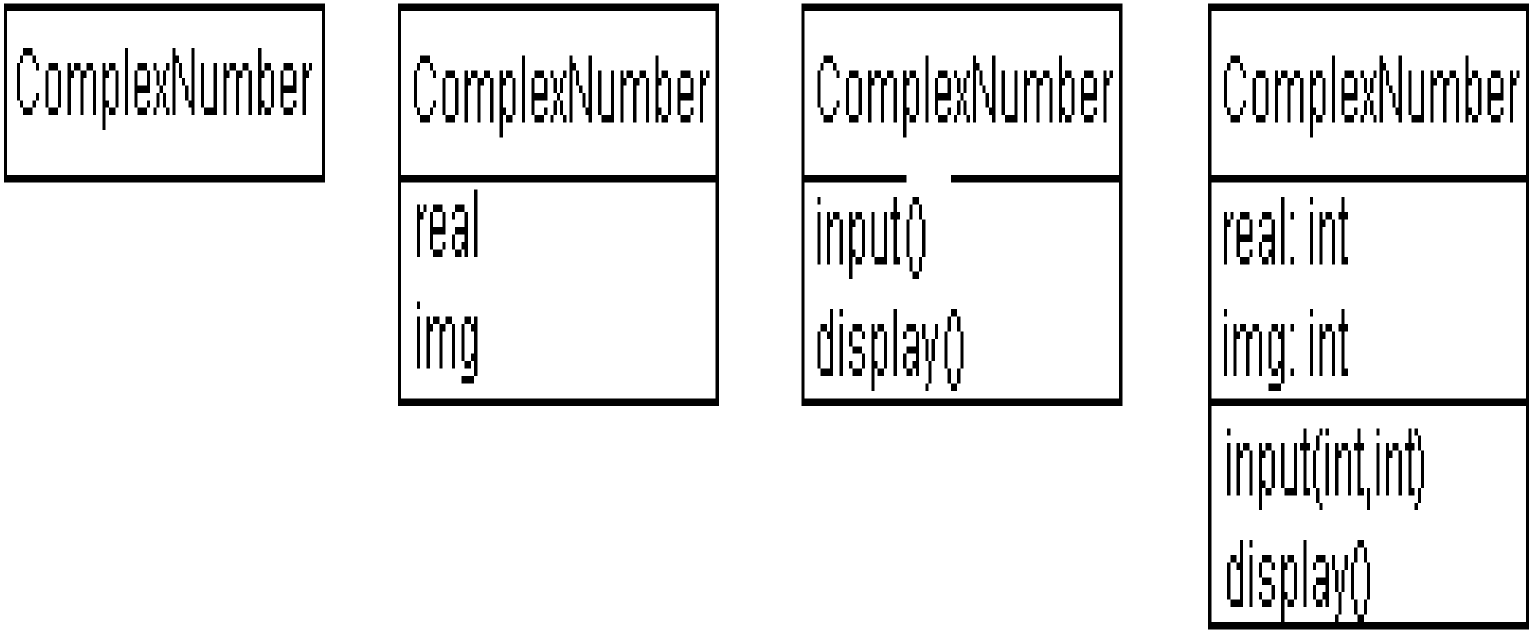
  Generalisation

Representation of a Class

- A class is represented as a box with the name of the class inside.

- This box can have three compartments first for representing the name of the class, second for the attributes and third for operations.

- These compartments may be omitted to simplify the diagrams.



ComplexNumber

ComplexNumber

real

img

ComplexNumber

input()

display()

ComplexNumber

real: int

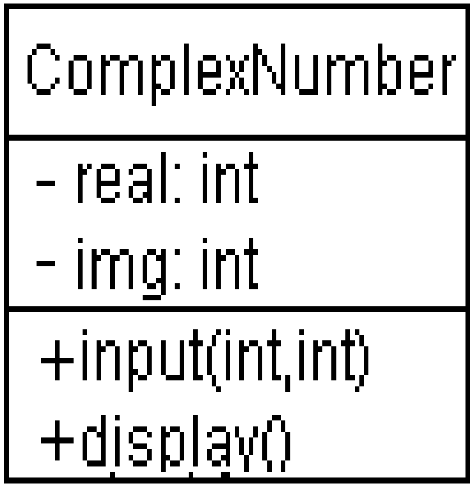
img: int

input(int,int)

display()

A class representing various levels of details

- **The visibility of the attributes and operations of a class can also be represented in a class notation using -, # and + symbols. The hyphen (-) means private, the pound sign (#) means protected, and plus (+) means public (see the following figure).**


ComplexNumber

- real: int
- img: int

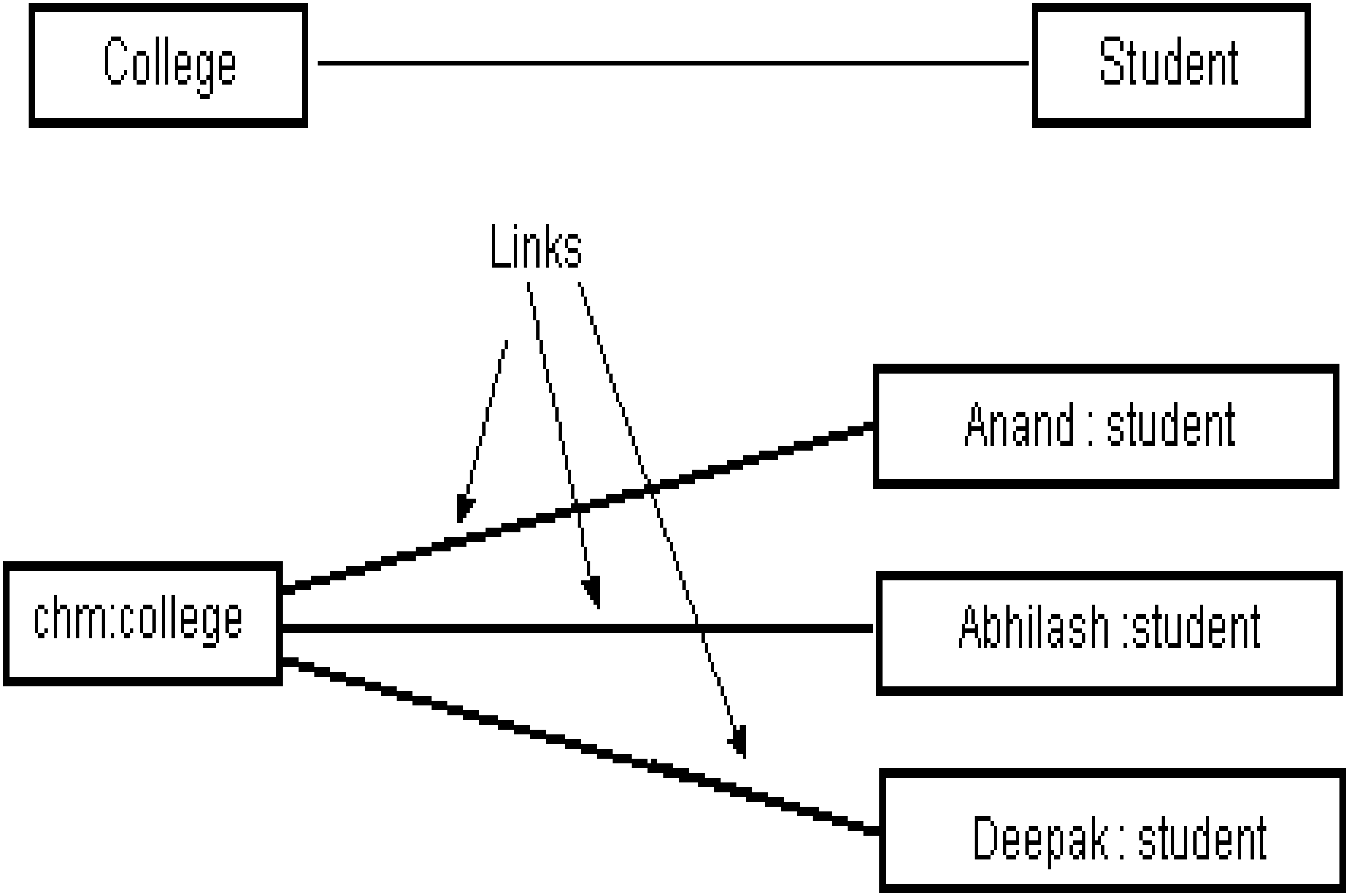
+input(int,int)
+display()

Association

- **An association is a relationship between two classes and is shown by a solid line between two classes.**

- **The instance of an association is known as a link.**

- **Therefore an association is a group of links with common structure and common semantics.**


College
Student
Links
Anand : student
chm:college
Abhilash :student
Deepak : student

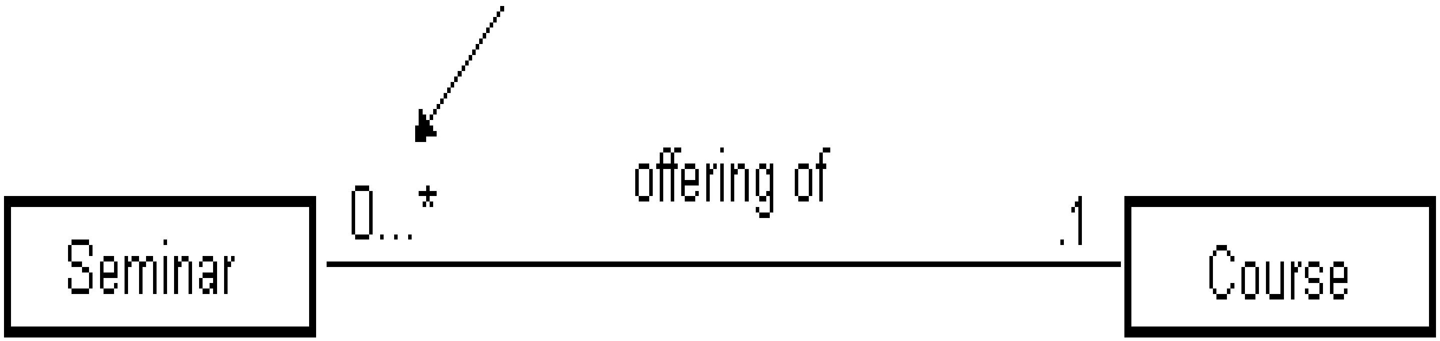
Multiplicity

- An association also represents multiplicity or cardinality.
- The multiplicity indicates how many objects of the class at one end of the association can be linked to a single instance of the class at the other end of the association.
- There are three types of multiplicity across an association.
- One-to-one
- One-to-many
- Many-to-many

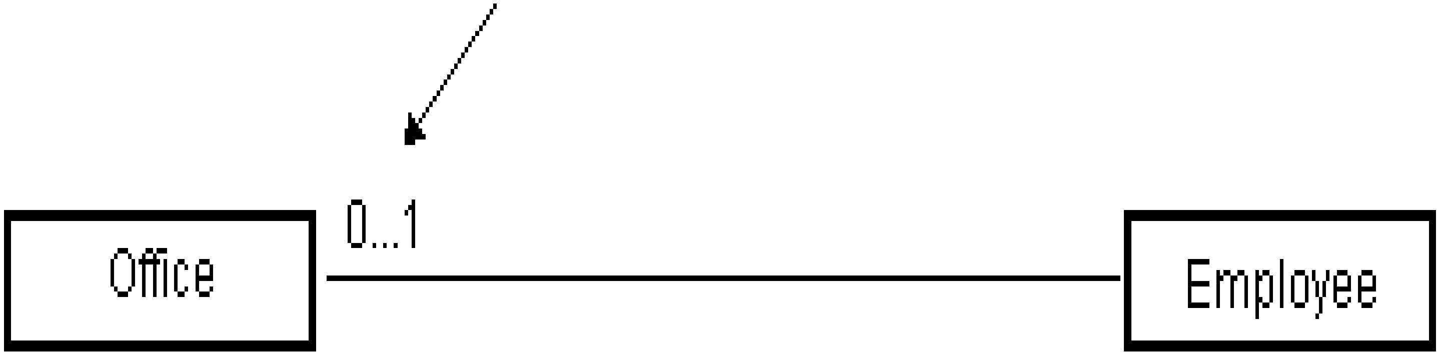
- **The multiplicity is represented as: lowerbound ... upperbound**
- **Eg:**

1	**means exactly one**
0...1	**zero or one**
*****	**from zero to any positive integer**
0...*	**from zero to any positive integer**
1...*	**from one to any positive integer**

many seminars offered for one course



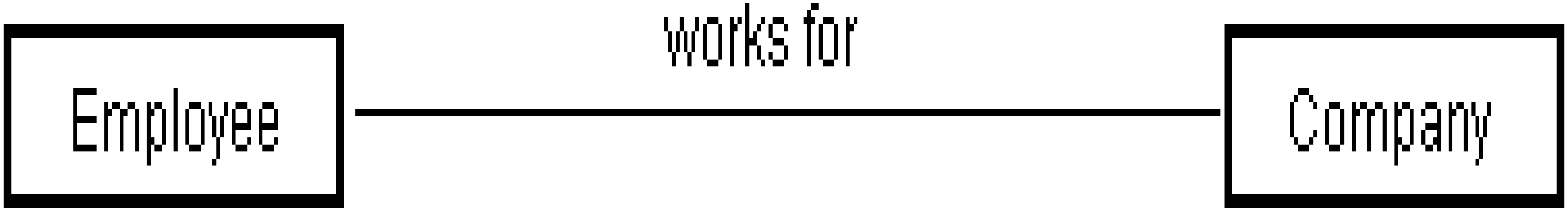
zero or one office is assigned to one employee



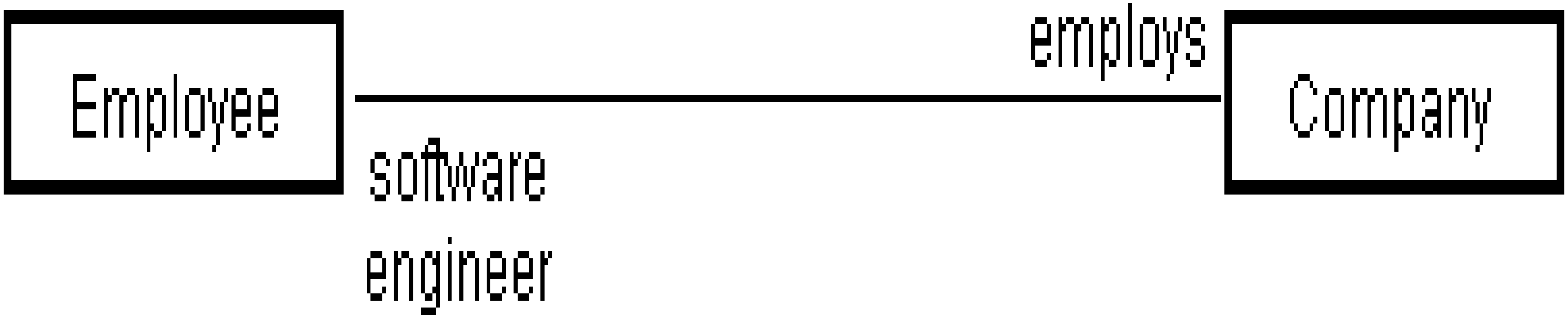
Labeled Association

- An association can be labeled by placing an association name in the middle of the association or by placing the role name either or both ends of the association.

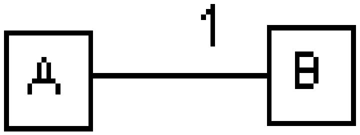
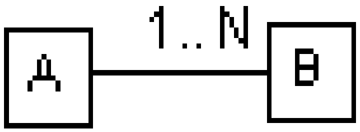
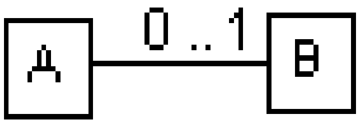
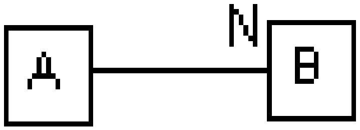
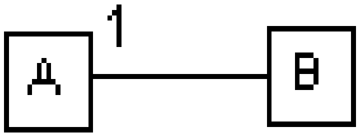
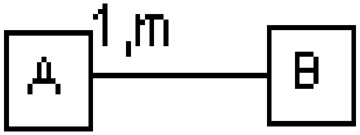
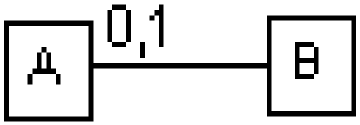
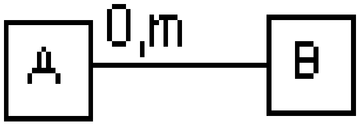
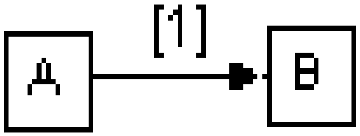
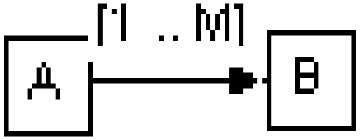
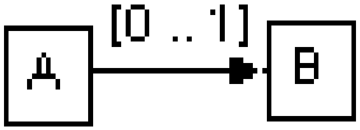
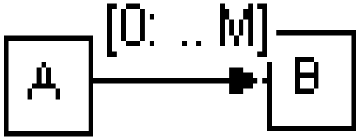
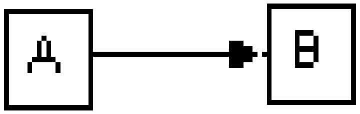
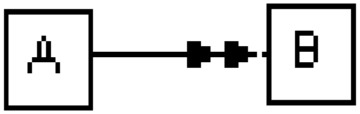
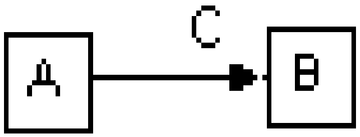
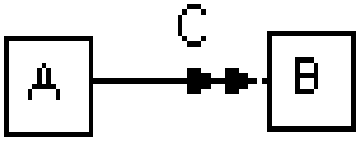
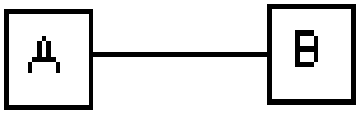
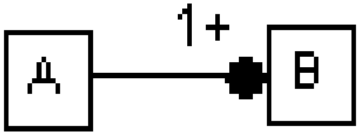
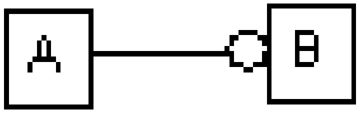
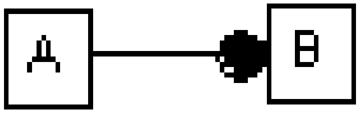
- If no association name or role name is specified, then the default association name ‘has’ is assumed.



association with association name

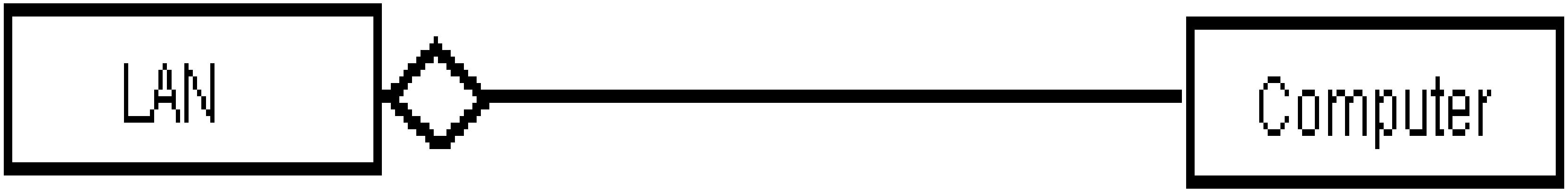


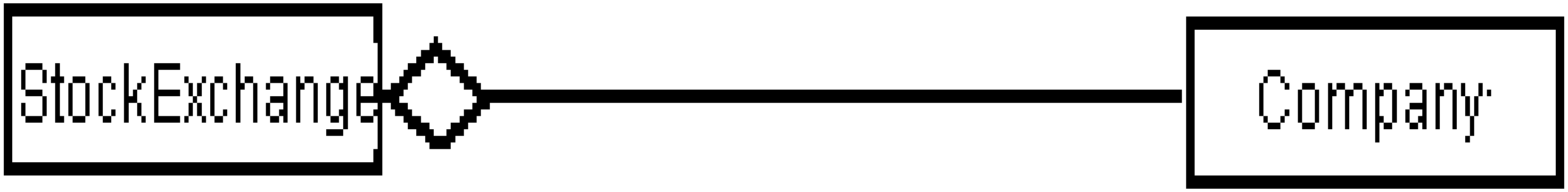
association with role name

Reading Left To Right	An A is always Associated with one B	An A is always Associated with one or many of B	An A is always Associated with zero or one of B	An A is always Associated with zero, one or many of B
Booch				
Coad/ Yourdon				
Jacobson				
Shlaer/ Mellor				
Rumbaugh				

Aggregations

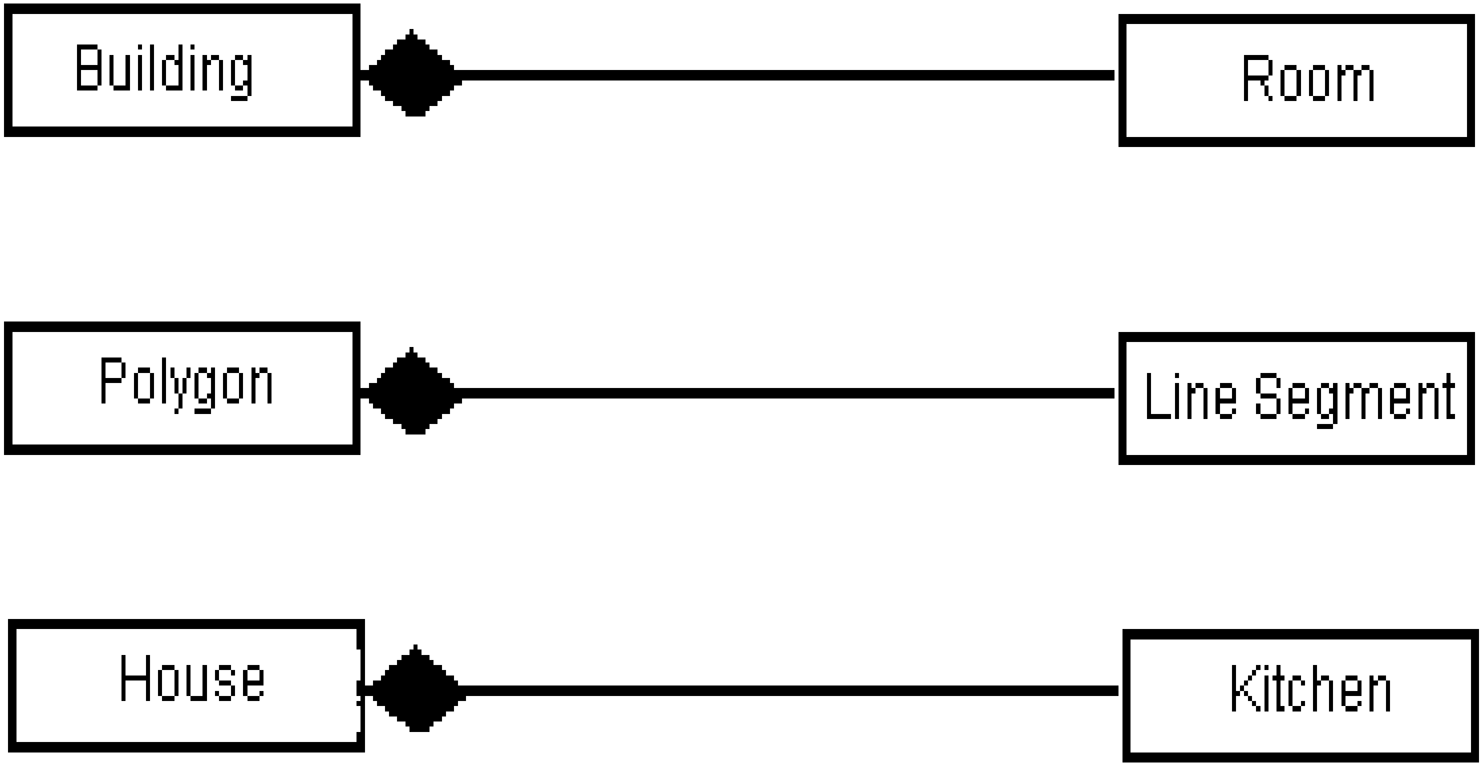
- Aggregation is an association in which one class belongs to a collection.
- Example :- Order has a collection of OrderDetails.
- It is represented by a diamond symbol placed next to the aggregate.
- Aggregations are special associations that represent a ‘has a’ or a ‘whole/part’ relationship among peers.


LAN
Computer


StockExchange
Company

Composition.

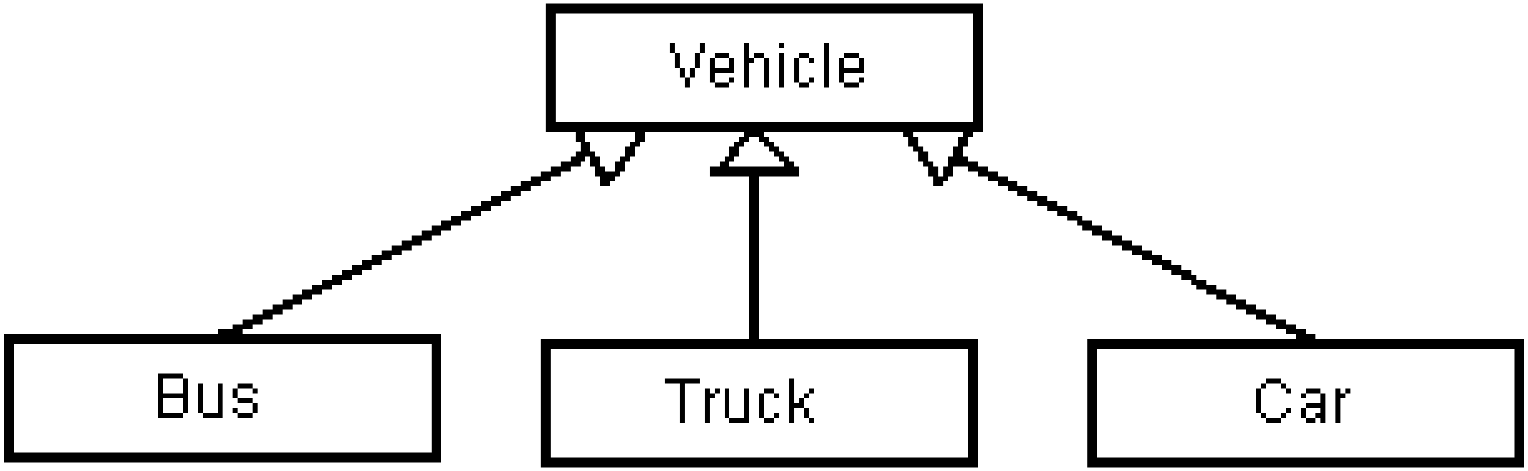
- Sometimes an aggregation relation may be a strong aggregation.
- The parts cannot have a life of their own.
- It means that if the aggregate is destroyed, then the parts also destroyed.
- A strong aggregation is also known as a composition.
- A strong aggregation is represented by a solid diamond symbol.


Building
Room
Polygon
Line Segment
House
Kitchen

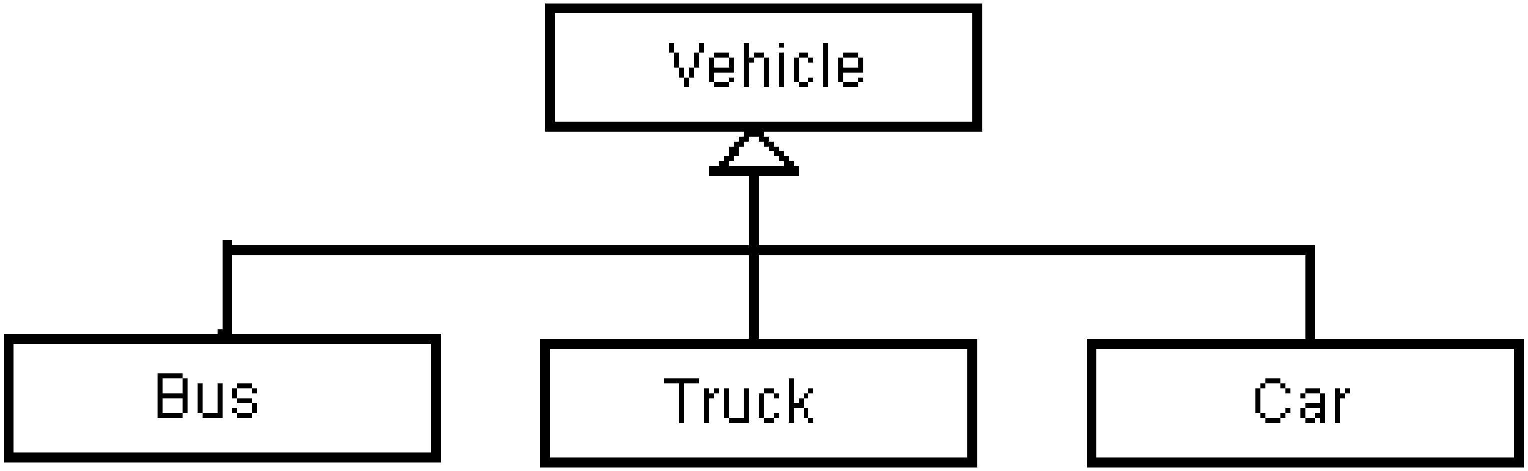
Generalization

- The generalization is the relationship between a more general class and a more specific class.

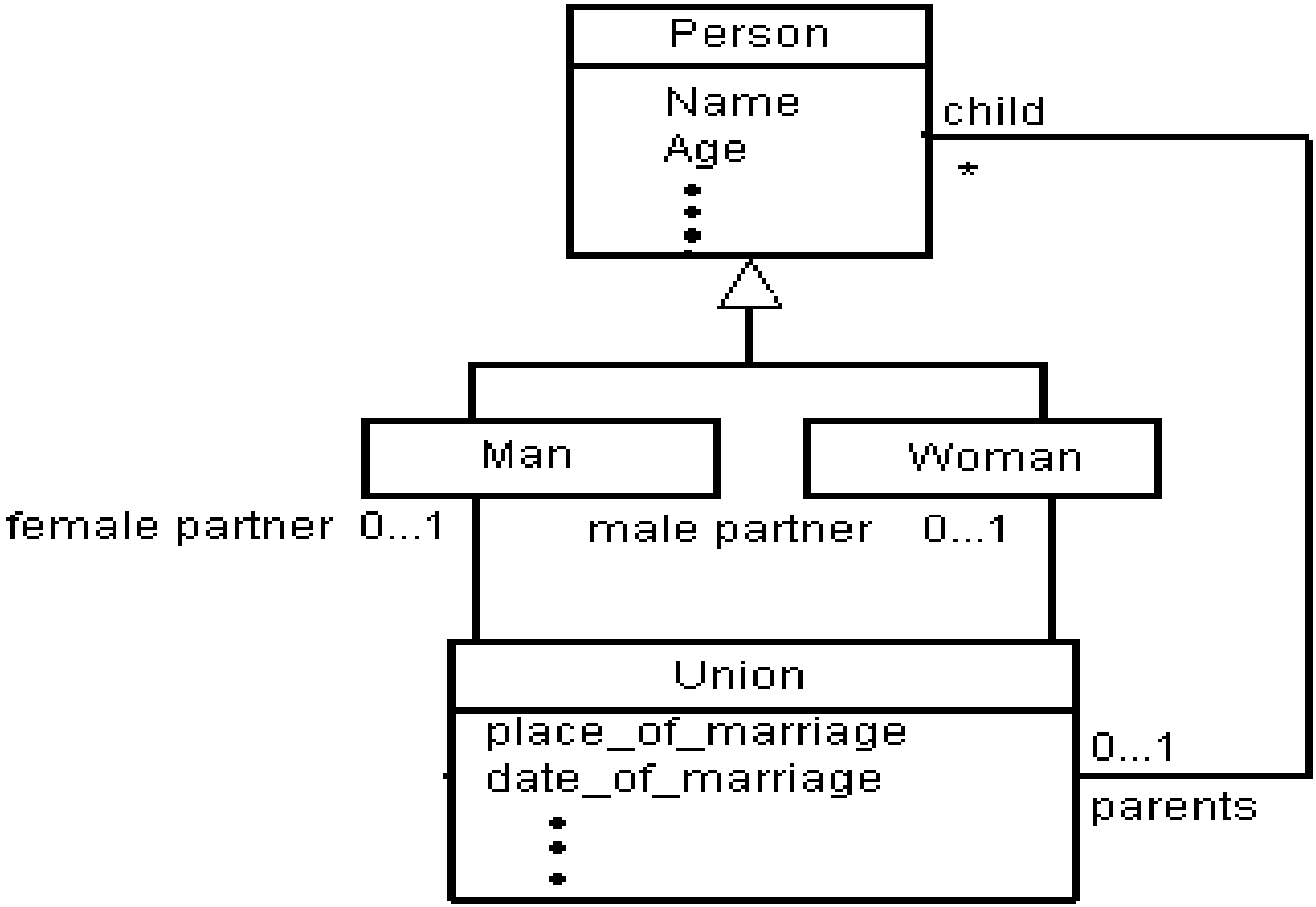
- It is represented by a small triangle pointing to the general class. They must follow the “is a” rule.



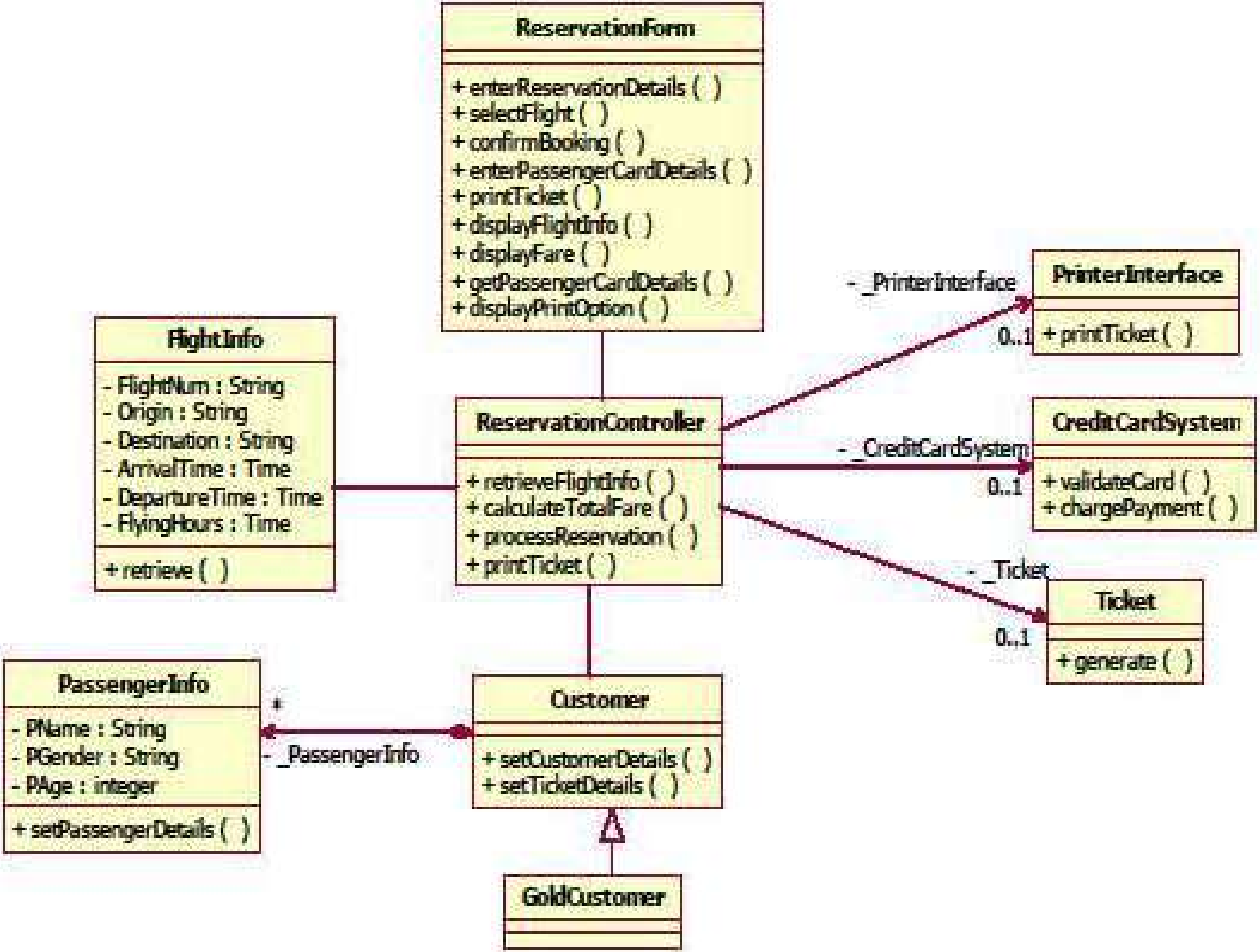
generalization seperate target style



generalization shared   target style



Genealogical class diagram


ReservationForm
+ enterReservationDetails ( )
+ selectFlight ( )
+ confirmBooking ( )
+ enterPassengerCardDetails ( )
+ printTicket ( )
+ displayFlightInfo ( )
+ displayFare ( )
+ getPassengerCardDetails ( )
+ displayPrintOption ( )
FlightInfo
- FlightNum : String
- Origin : String
- Destination : String
- ArrivalTime : Time
- DepartureTime : Time
- FlyingHours : Time
+ retrieve ( )
ReservationController
+ retrieveFlightInfo ( )
+ calculateTotalFare ( )
+ processReservation ( )
+ printTicket ( )
- _PrinterInterface
0..1
PrinterInterface
+ printTicket ( )
- _CreditCardSystem
0..1
CreditCardSystem
+ validateCard ( )
+ chargePayment ( )
- _Ticket
0..1
Ticket
+ generate ( )
PassengerInfo
- PName : String
- PGender : String
- PAge : integer
+ setPassengerDetails ( )
*
- _PassengerInfo
Customer
+ setCustomerDetails ( )
+ setTicketDetails ( )
GoldCustomer

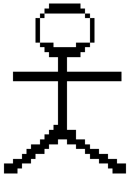
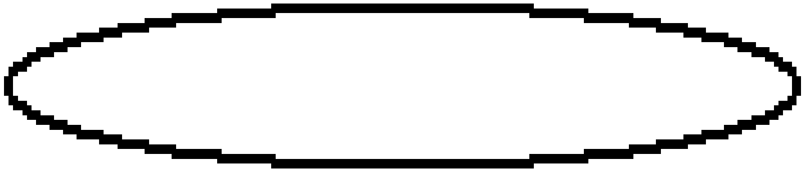
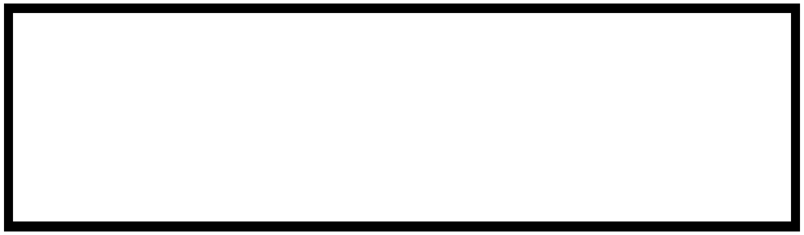
Use-Cases

- A use case is a typical sequence of actions that an actor performs in order to complete a given task.

- It describes the behavior of the system from a users point of view by using actions and reactions.

- An actor is role that a user or some other system plays when interact with your system.

Symbols Used in Use Cases

Actors	
Use Cases	
System	

Before drawing a use case diagram, a scenario must be created.

The steps involved in creating a scenario

- Give a short descriptive name to the use case
- List the actor or actors who can perform this use case
- Explain the goals of the actor.
- Specify preconditions
- Summery-summaries what occurs as the actor or actors perform the use case.
- List related use cases
- Steps-describe the steps of the use case using a two-column format.
- Specify post condition

Use case for leaving a particular automated Car Parking system

Actor

Car Drivers

Goals

To leave the parking lot after having paid the amount due.

Preconditions

The driver must have entered the car park with his or her car; and must have picked up a ticket upon entry.

Summary

When a driver wishes to exit the car park, he or she must bring his or her car to the exit barrier and interact with a machine to pay the amount due.

Related use case

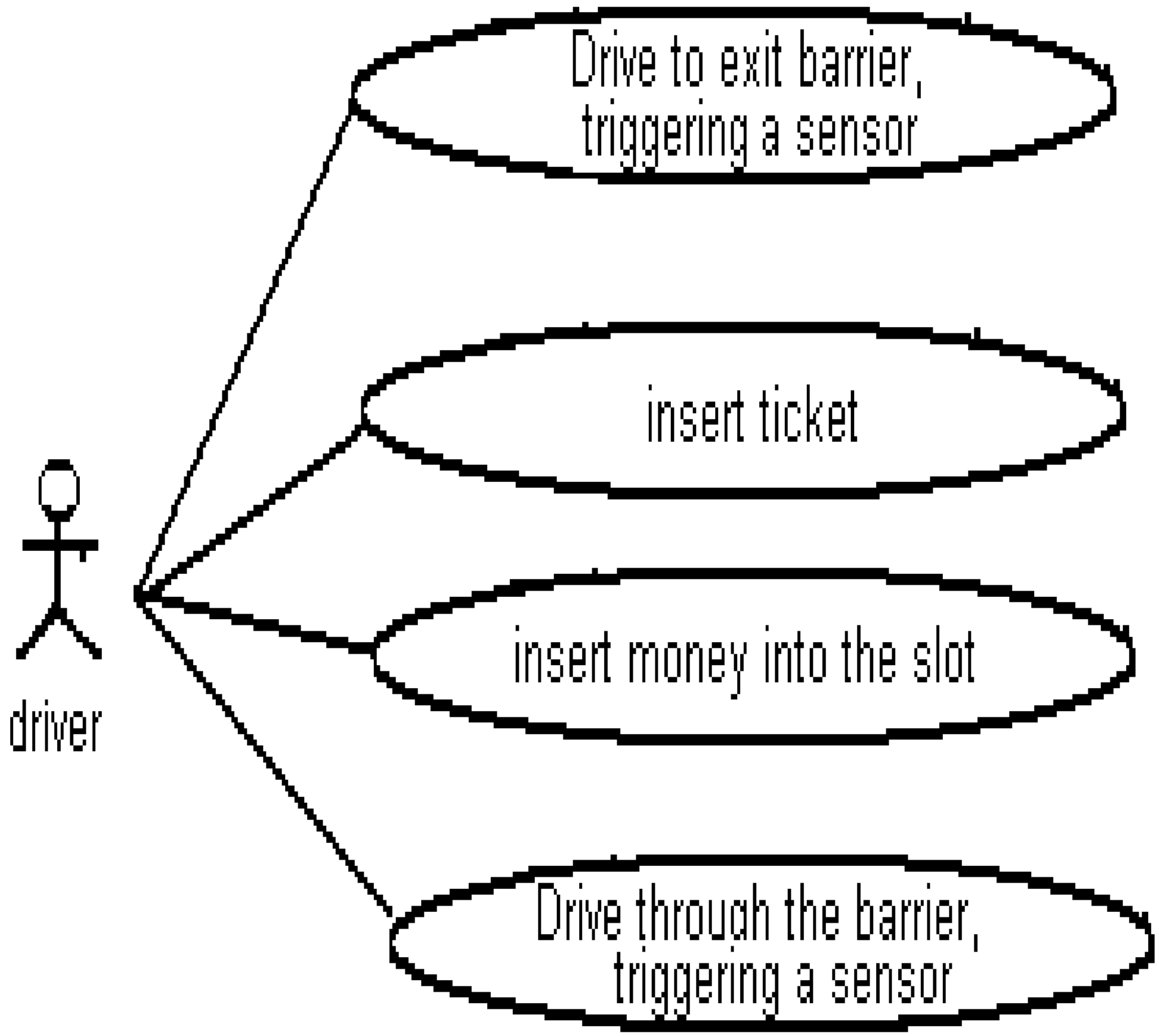
Exit car park by paying using a debit card.

Steps

Actor actions	**System Responses**
Drive to exit barrier	Detect presence of a
car.	
	Ask driver to
insert	card.
Insert ticket	Display amount due.
Insert money into the slot	Return any change and
ask	
	the driver to take
the	
	change(if any).
	Raise
Barrier.	
Drive through barrier	Lower barrier.

Post condition

A car exited from the car parking lot.


Drive to exit barrier,
triggering a sensor
insert ticket
insert money into the slot
Drive through the barrier,
triggering a sensor
driver

Points to note:

- The actors are not Navin and Altaf which are names of people.
- Note that an actor is a Role.
- And the role of actor in the previous example is that of a driver.

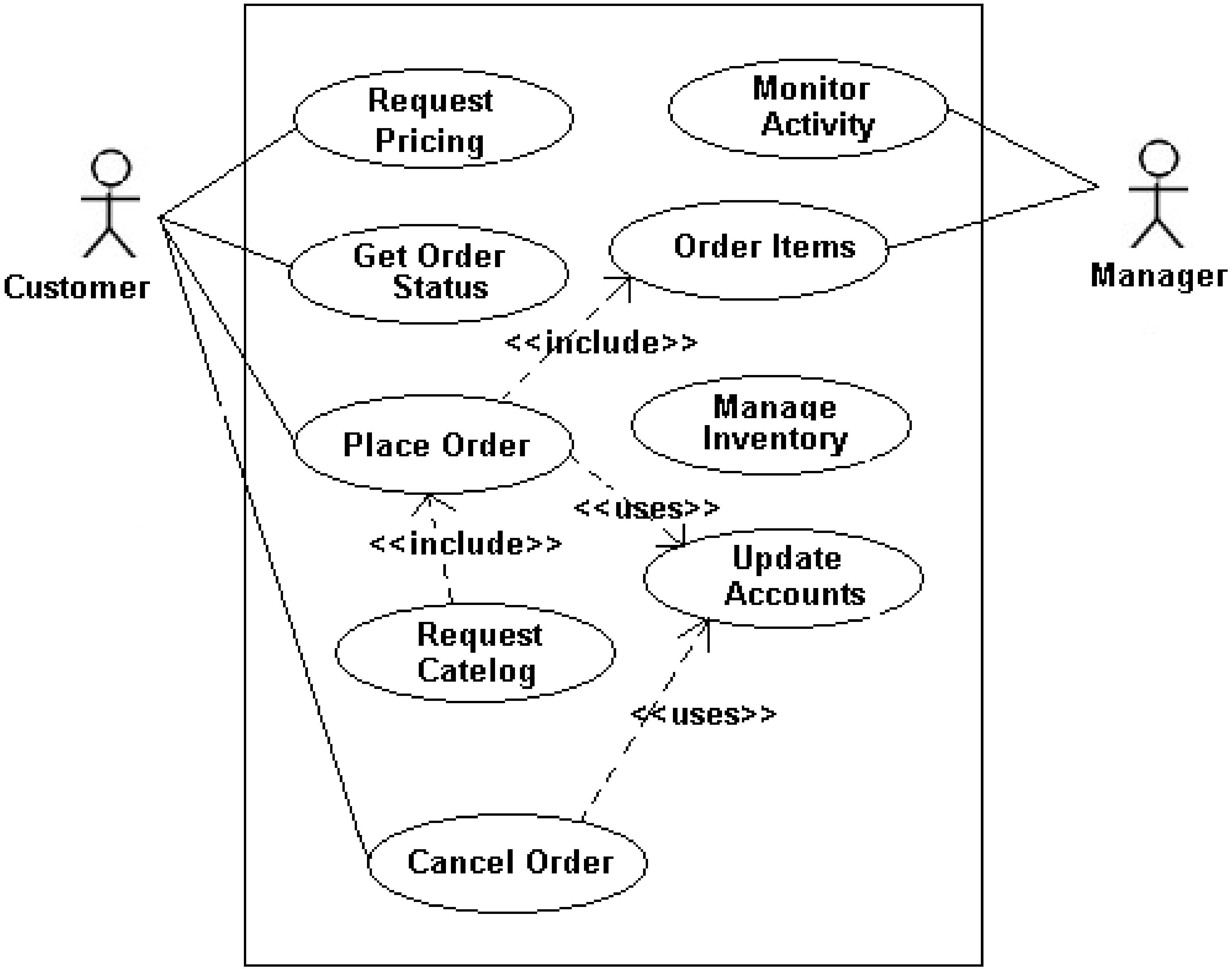
<<extends>>

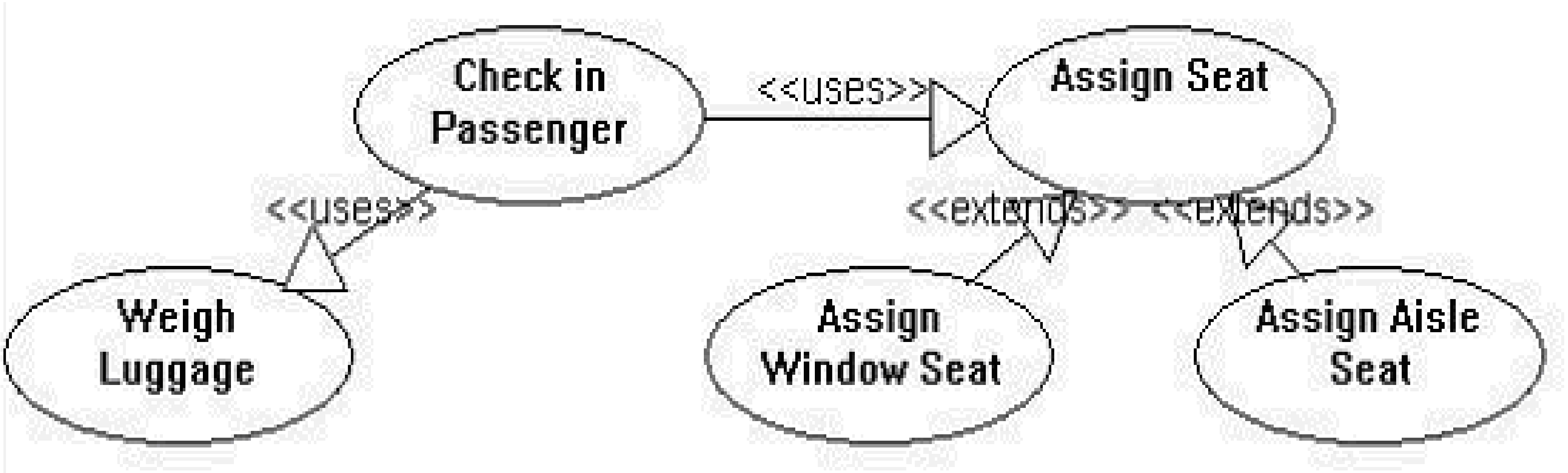
- “Y extends X" indicates that “Y" is a task for the same type as “X", but “Y" is a special, more specific case of doing “X".

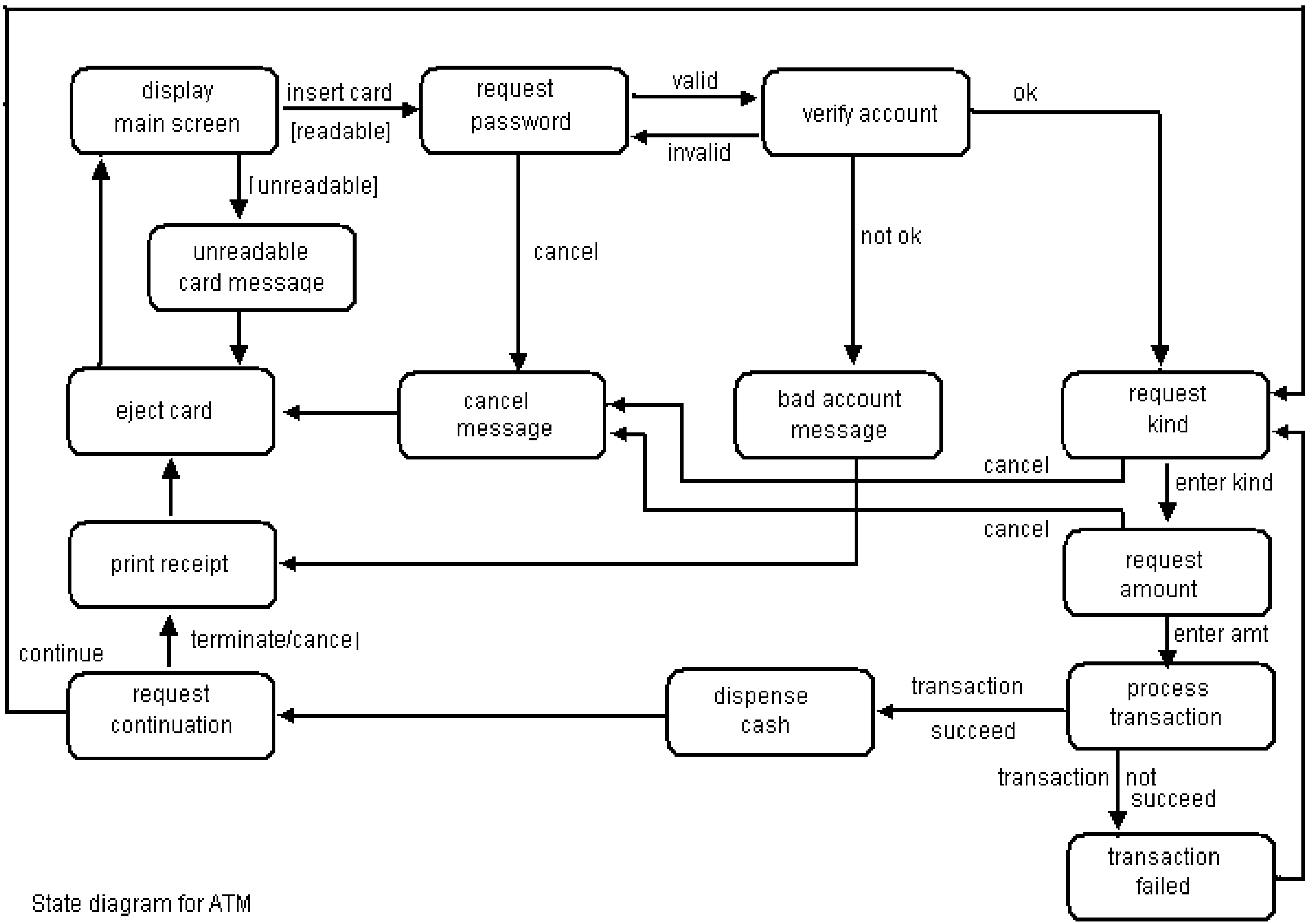
- That is, doing Y is a lot like doing X, but Y has a few extra processes to it that go above and beyond the things that must be done in order to complete X.

<<uses>> **or <<include>>**

- **"X uses Y" indicates that the task "X" has a subtask "Y"; that is, in the process of completing task "X", task "Y" will be completed at least once.**


Request Pricing
Monitor Activity
Customer
Get Order Status
Order Items
Manager
<<include>>
Manage Inventory
Place Order
<<uses>>
<<include>>
Update Accounts
Request Catelog
<<uses>>
Cancel Order


Check in Passenger
<<uses>>
Assign Seat
<<uses>>
Weigh Luggage
<<extends>>
<<extends>>
Assign Window Seat
Assign Aisle Seat



State diagram for ATM

Transport Management System

The following are the activities of a receptionist in a Bus Reservation System. Construct a use case diagram from the these activities.

Check Record For Seat Availability

Cancel Booking

Confirm Booking

Cancel Fee

Request Bill

Accept Pay

Print Bill

Library System

The following are the activities of a borrower in a Library System. Construct a use case diagram from the these activities.

Borrow Books

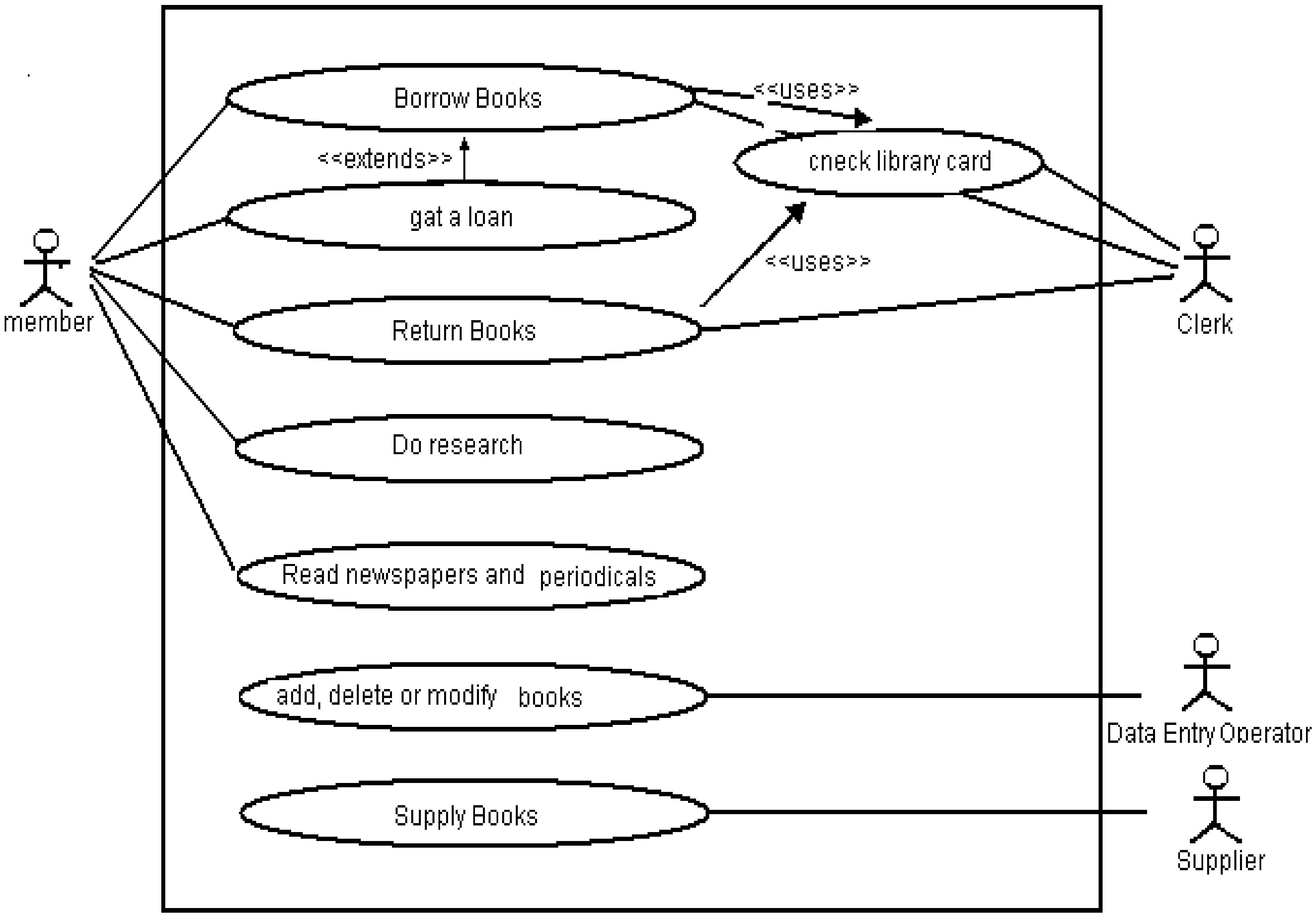
Do Research

Reading Newspapers and periodicals

Get Loan

Check Library Card

Return Books


Borrow Books
<<uses>>
<<extends>>
cneck library card
gat a loan
<<uses>>
member
Clerk
Return Books
Do research
Read newspapers and periodicals
add, delete or modify books
Data Entry Operator
Supply Books
Supplier

Activity Diagrams

Activity Diagrams

- It resembles a flowchart.

- It illustrates the dynamic nature of a system by modeling the flow of control from activity to activity.

- **In an activity diagram, most transitions are caused by internal events.**

- **Activity diagrams are used to**

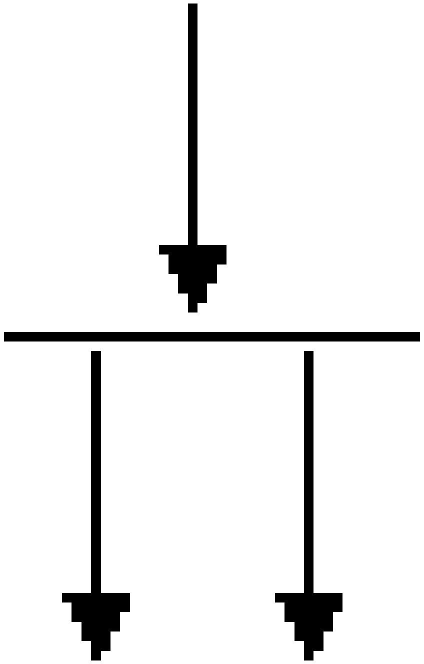
  1)model business process

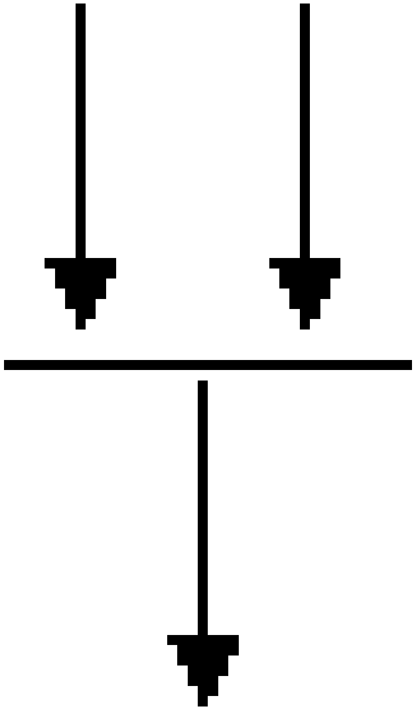
  2)to model internal operation of a use case.

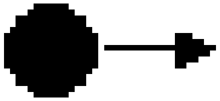
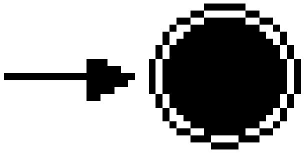
  3)to model work flows and computations.

Rounded rectangles 	Represents activity

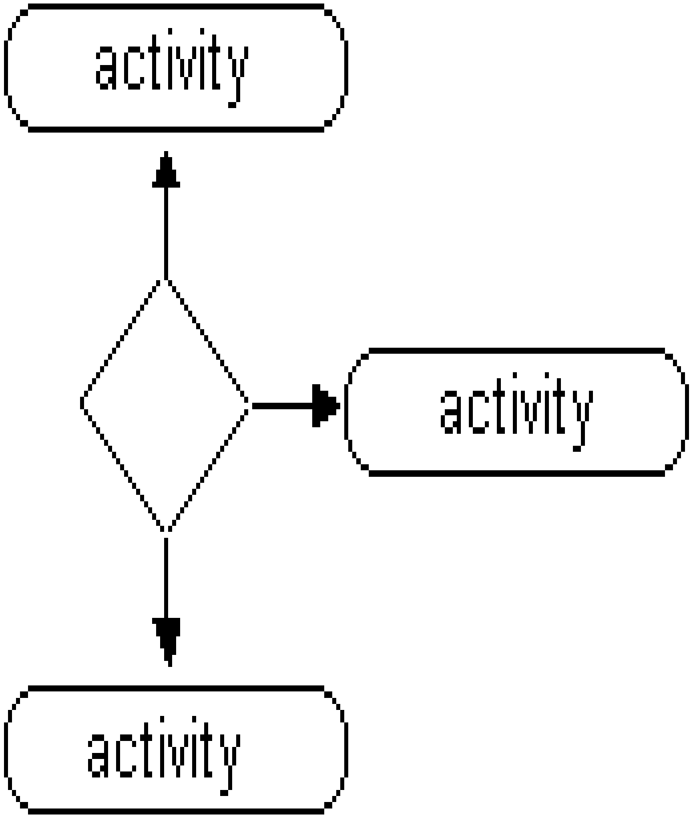
Arrows ↓	Simple completion of transitions

Fork (It has one incoming transition and multiple outgoing transitions). 	Fork represents the splitting of a single flow of control into two or more concurrent flows.

Join (Multiple arrows entering a heavy synchronization bar) 	Join represents the Synchronization of two or more flows of control into one sequential flow of control

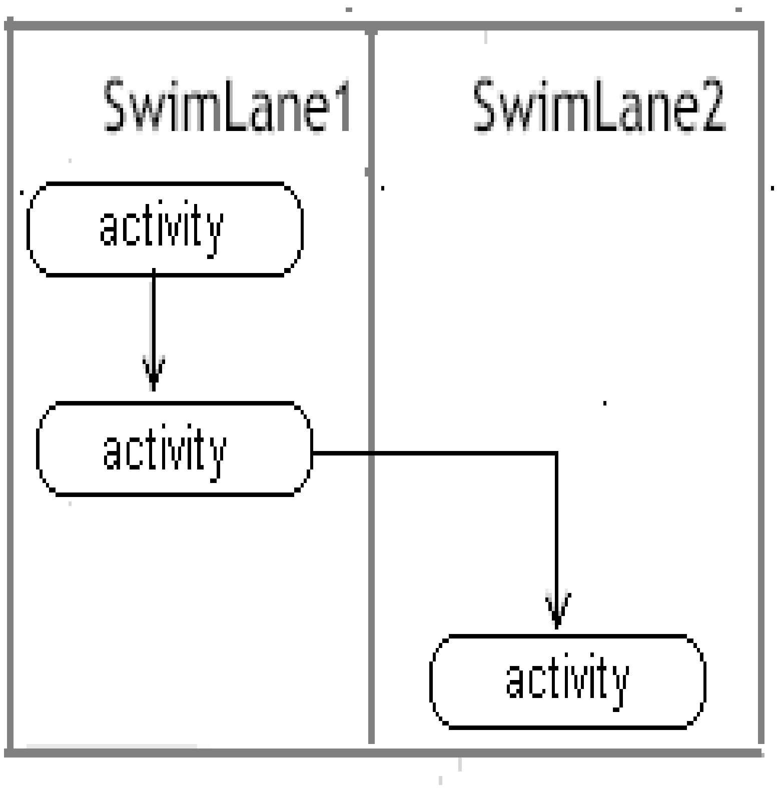
Start state 	Beginning of the state.
End state 	End of the state.

Alternate path

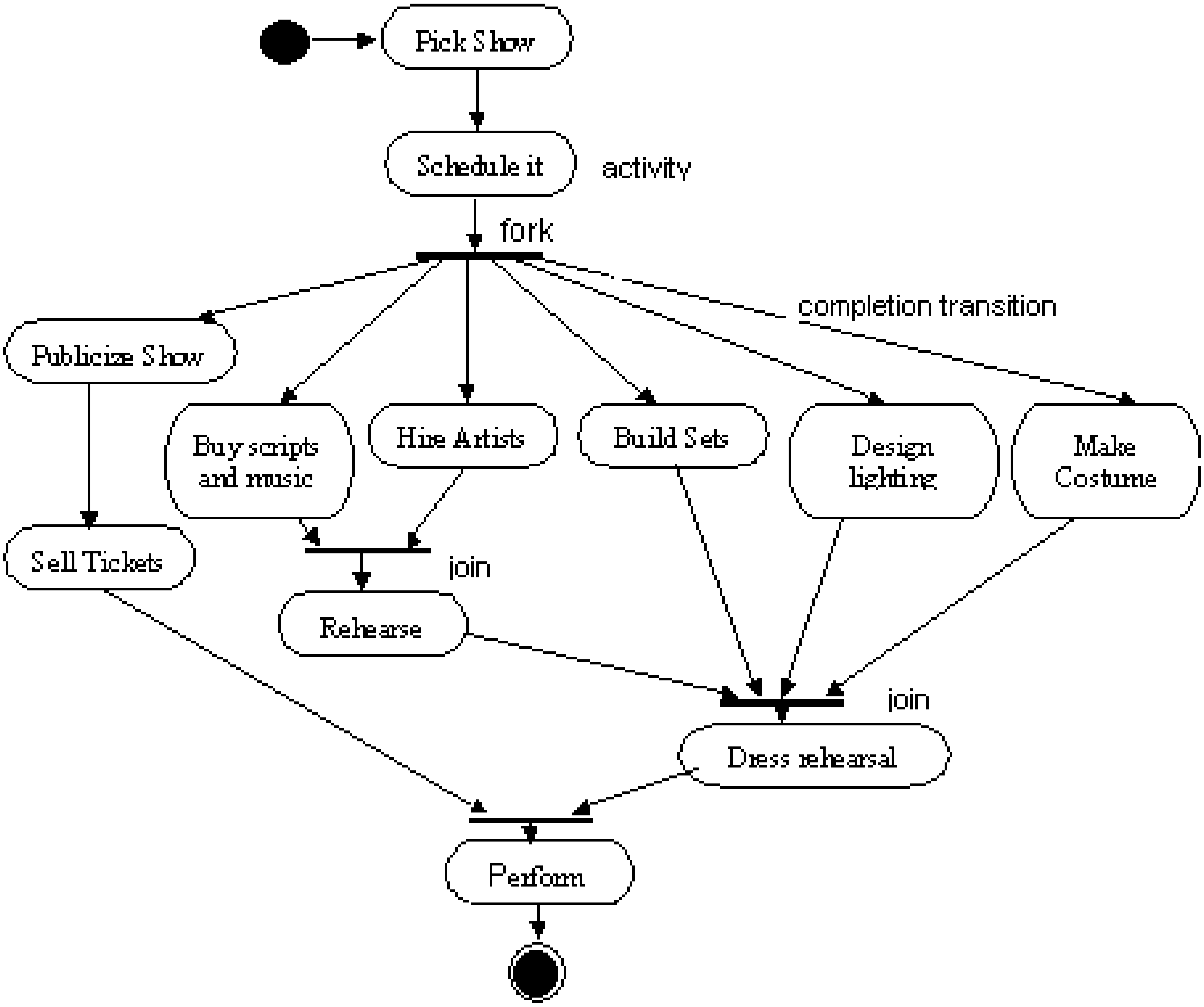


Diamond represents a decision with alternate path.

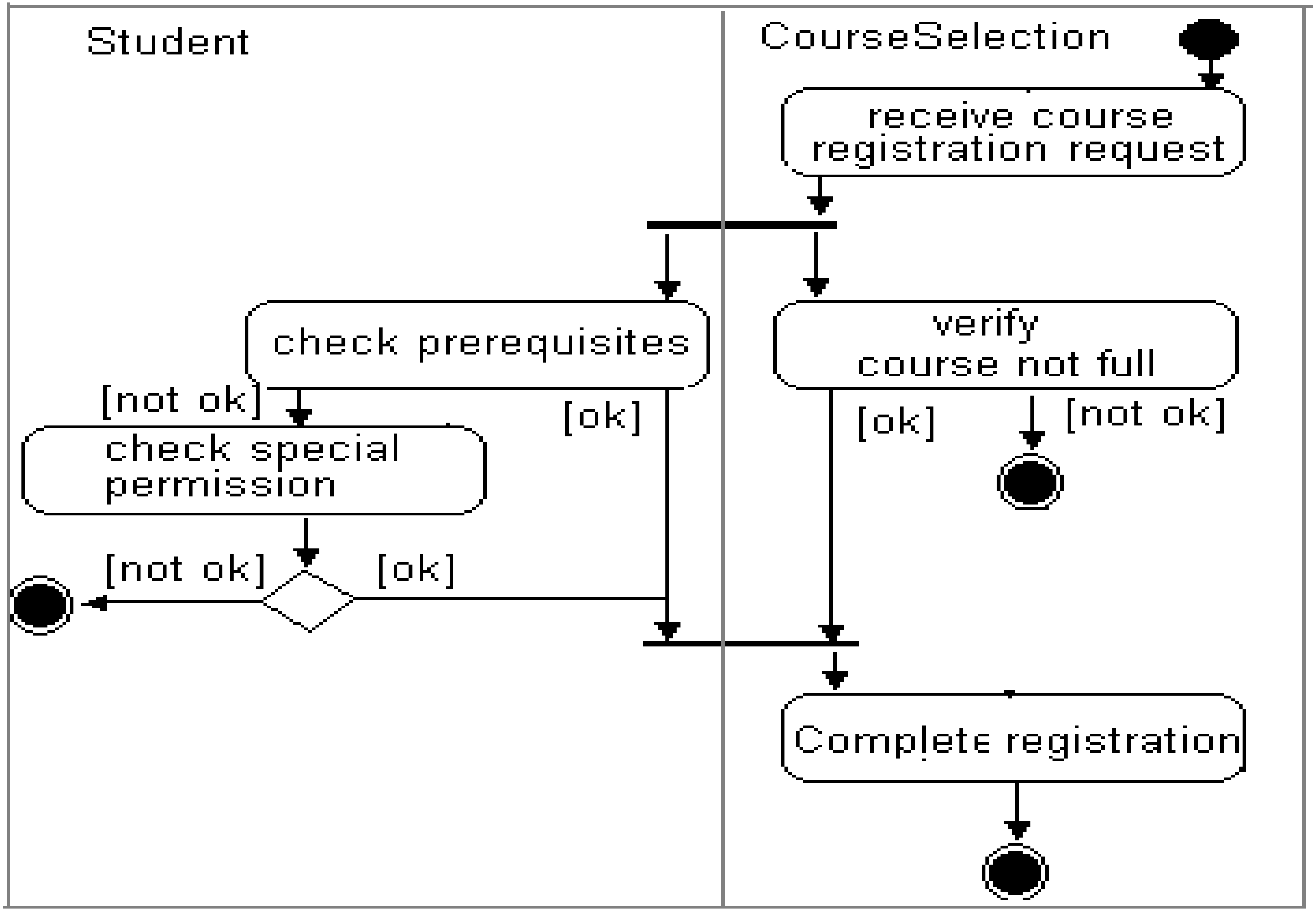
Swim lines

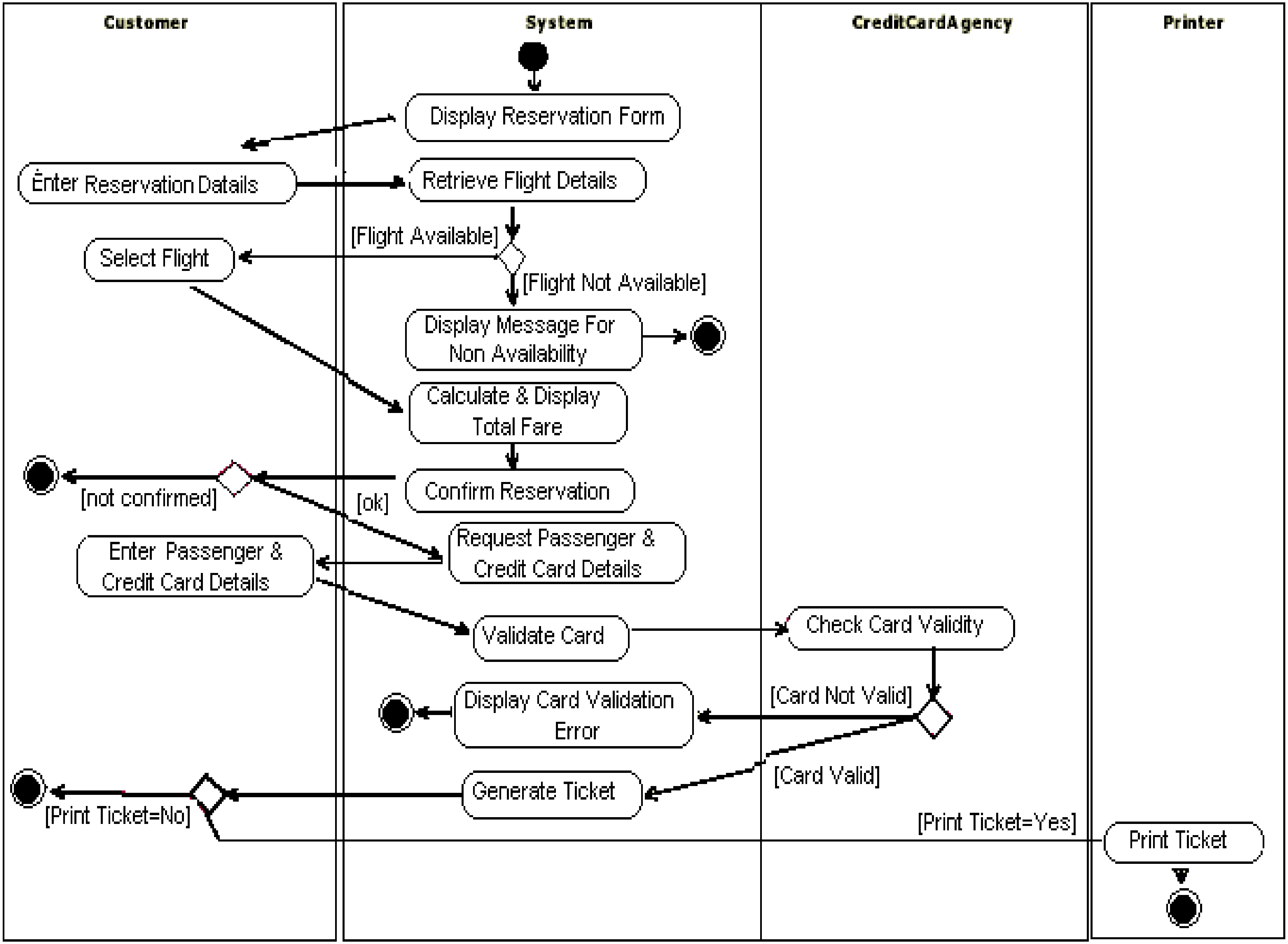


Swim lanes are used for grouping the related activities in to columns.



Activity diagram for a mounting a show


Student
CourseSelection
receive course registration request
check prerequisites
verify course not full
[not ok]
[ok]
[ok]
[not ok]
check special permission
[not ok]
[ok]
Complete registration


Customer
System
CreditCardAgency
Printer
Display Reservation Form
Ėnter Reservation Datails
Retrieve Flight Details
[Flight Available]
Select Flight
[Flight Not Available]
Display Message For Non Availability
Calculate & Display Total Fare
Confirm Reservation
[not confirmed]
[ok]
Request Passenger & Credit Card Details
Enter Passenger & Credit Card Details
Validate Card
Check Card Validity
[Card Not Valid]
Display Card Validation Error
[Card Valid]
Generate Ticket
[Print Ticket=No]
[Print Ticket=Yes]
Print Ticket