SOFTWARE ENGINEERING
(Week-4)

USAMA MUSHARAF

MS-CS (Software Engineering)

LECTURER (Department of Computer Science)

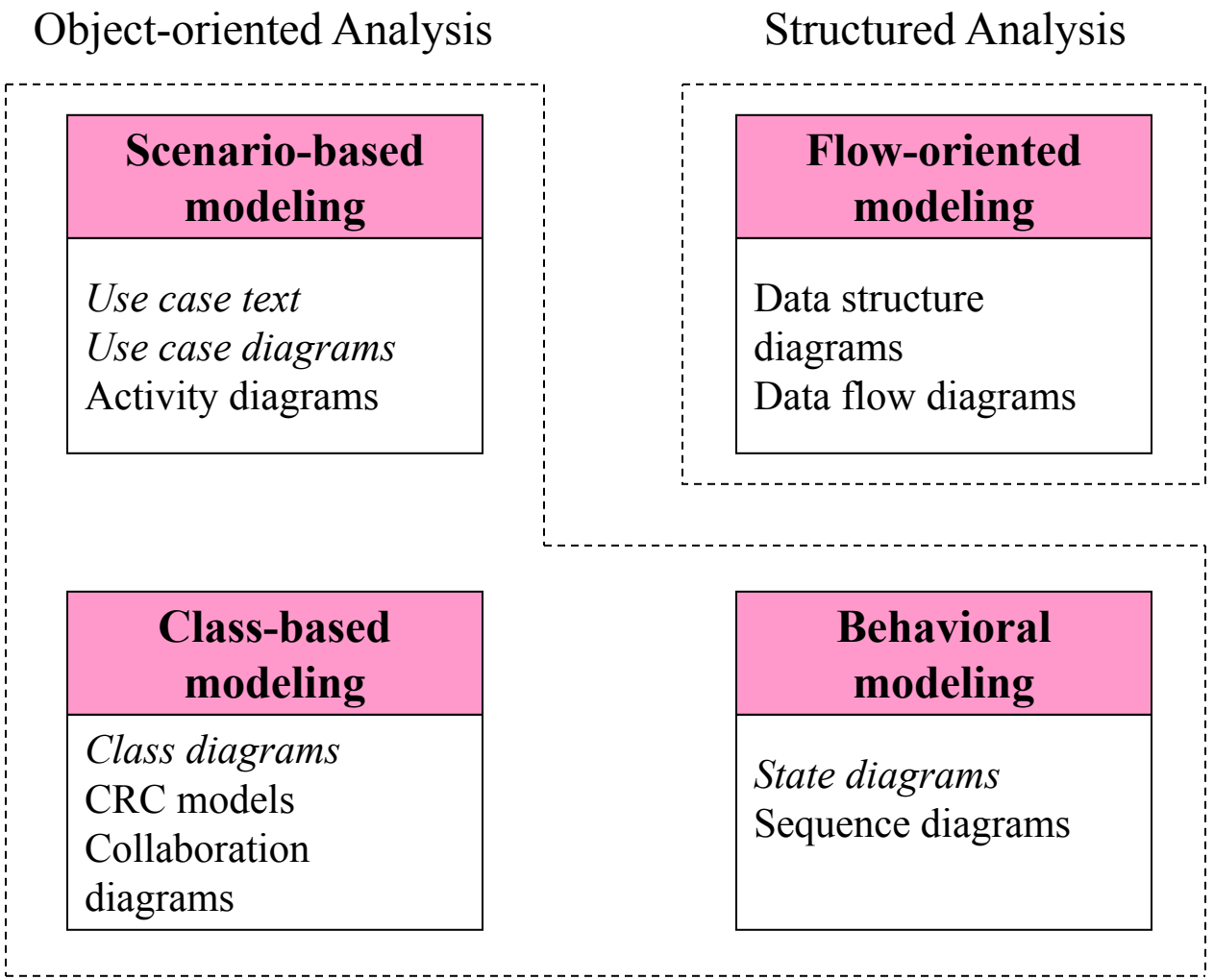
FAST-NUCES PESHAWAR

AGENDA OF WEEK # 4

- Requirement Analysis (Cont…)
- Object Oriented Analysis
- Modeling with UML (Static and Dynamic Models)
- Discussion on Assignment # 2

ANALYSIS MODELING (CONT…)

ELEMENTS OF THE ANALYSIS MODEL



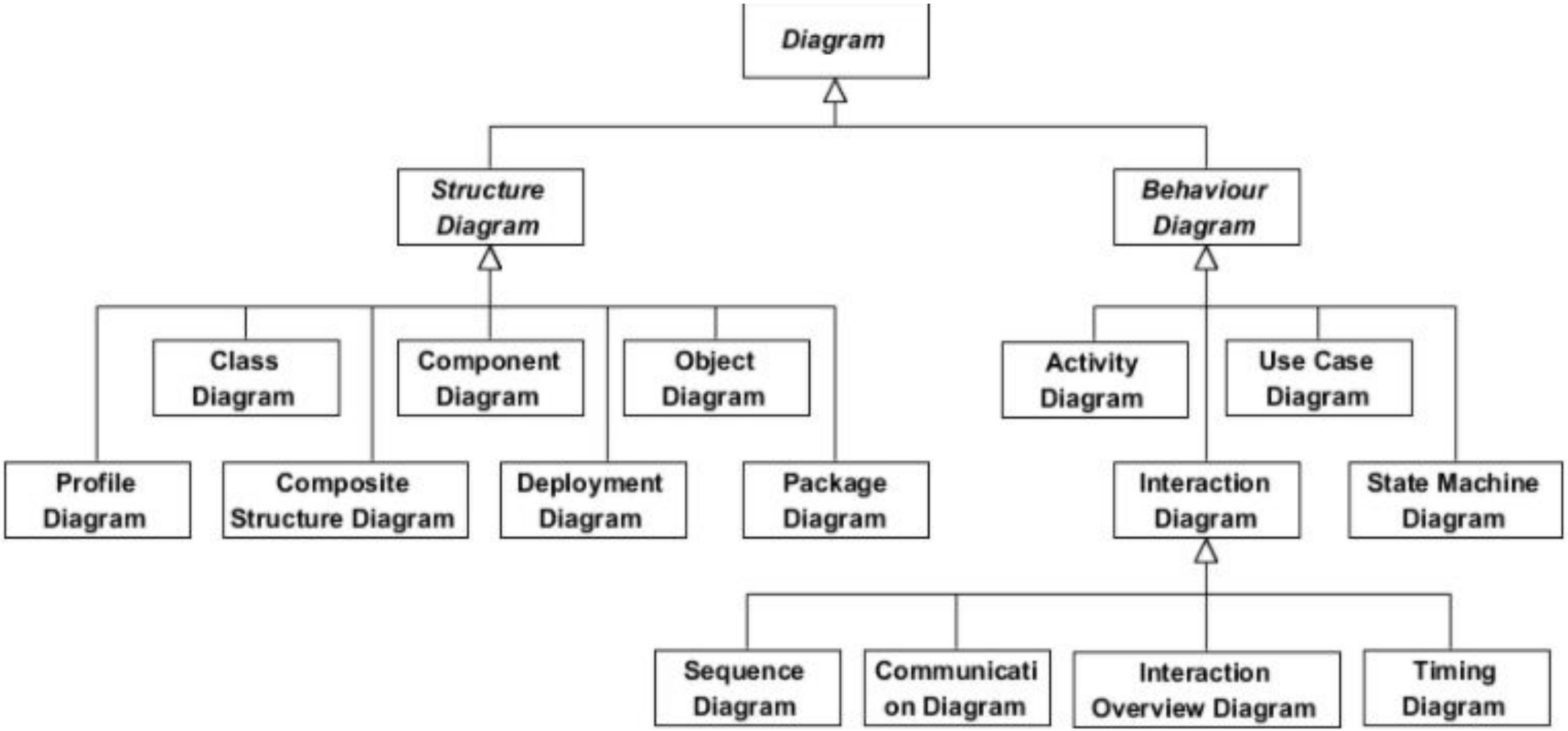
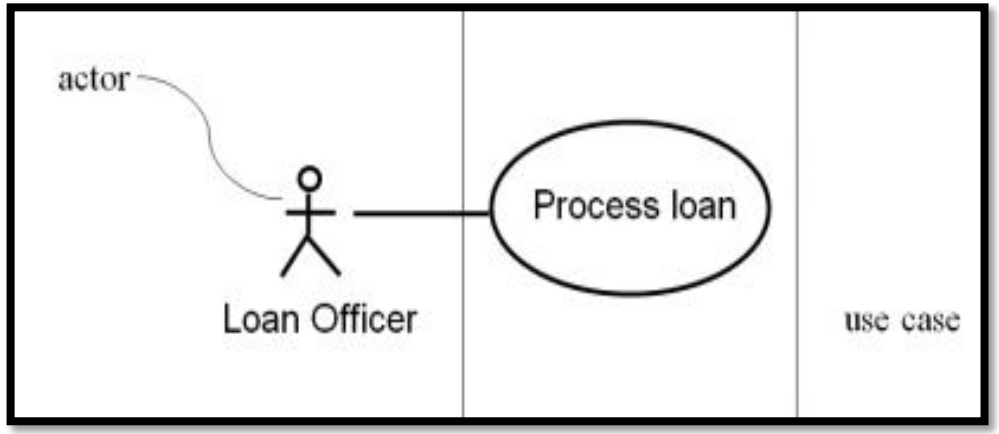


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communicati on Diagram
Interaction Overview Diagram
Timing Diagram

SCENARIO-BASED MODELING

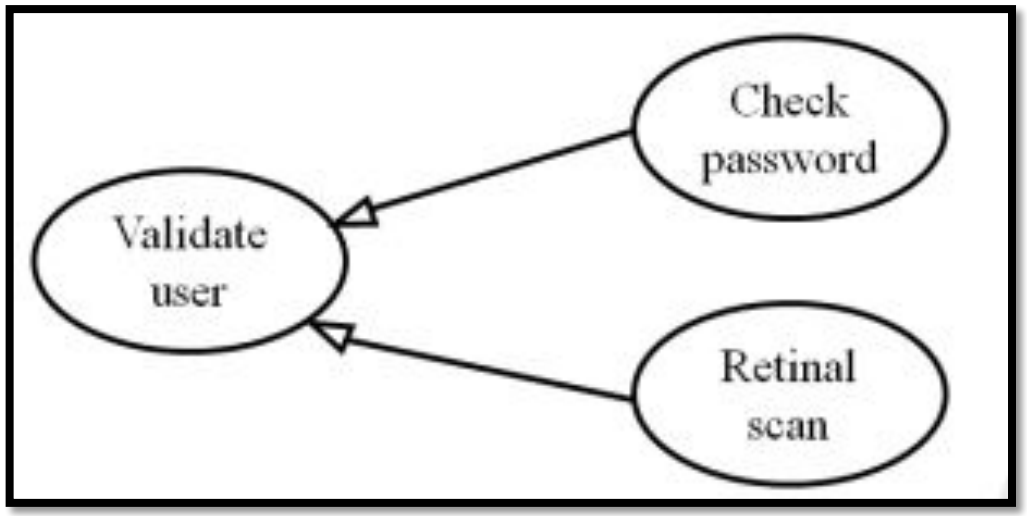
DEVELOPING USE CASES

- – Define the set of actors that will be involved in the story
  - Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
  - Actors are anything that communicate with the system or product and that are external to the system itself



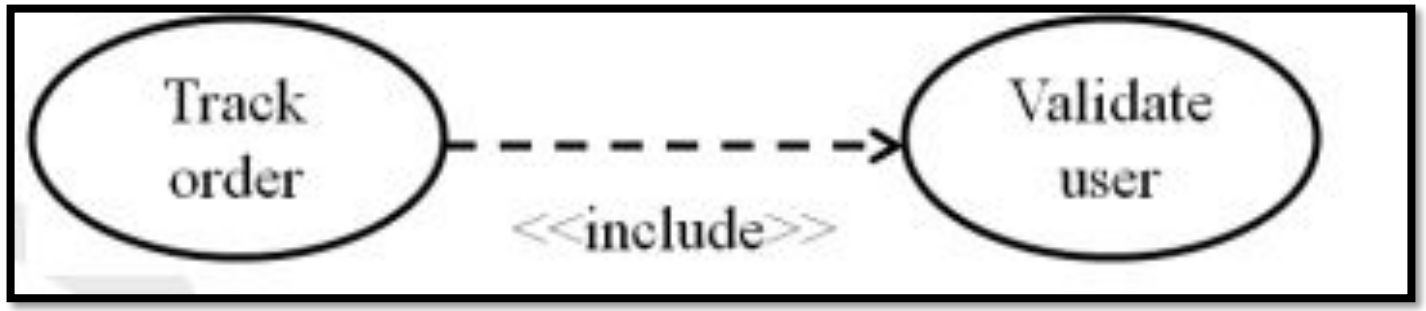
SPECIALIZED USE CASES

You may have two specialized children of this use case (Check password and Retinal scan).



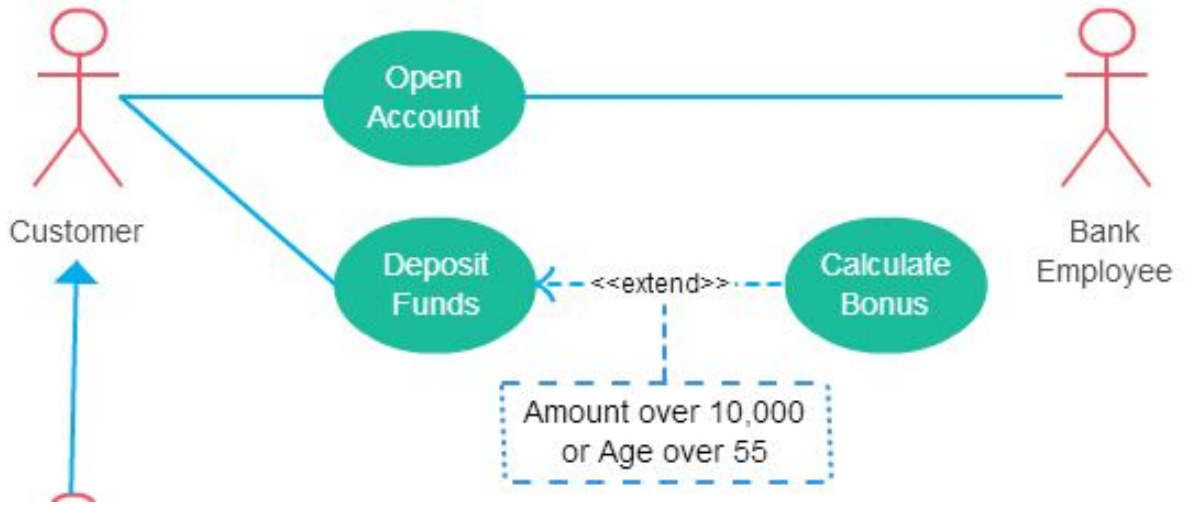
INCLUDE

- Include relationship is used to avoid describing the same flow of events several times, by putting the common behavior in a use case of its own
- This is an example of dependency

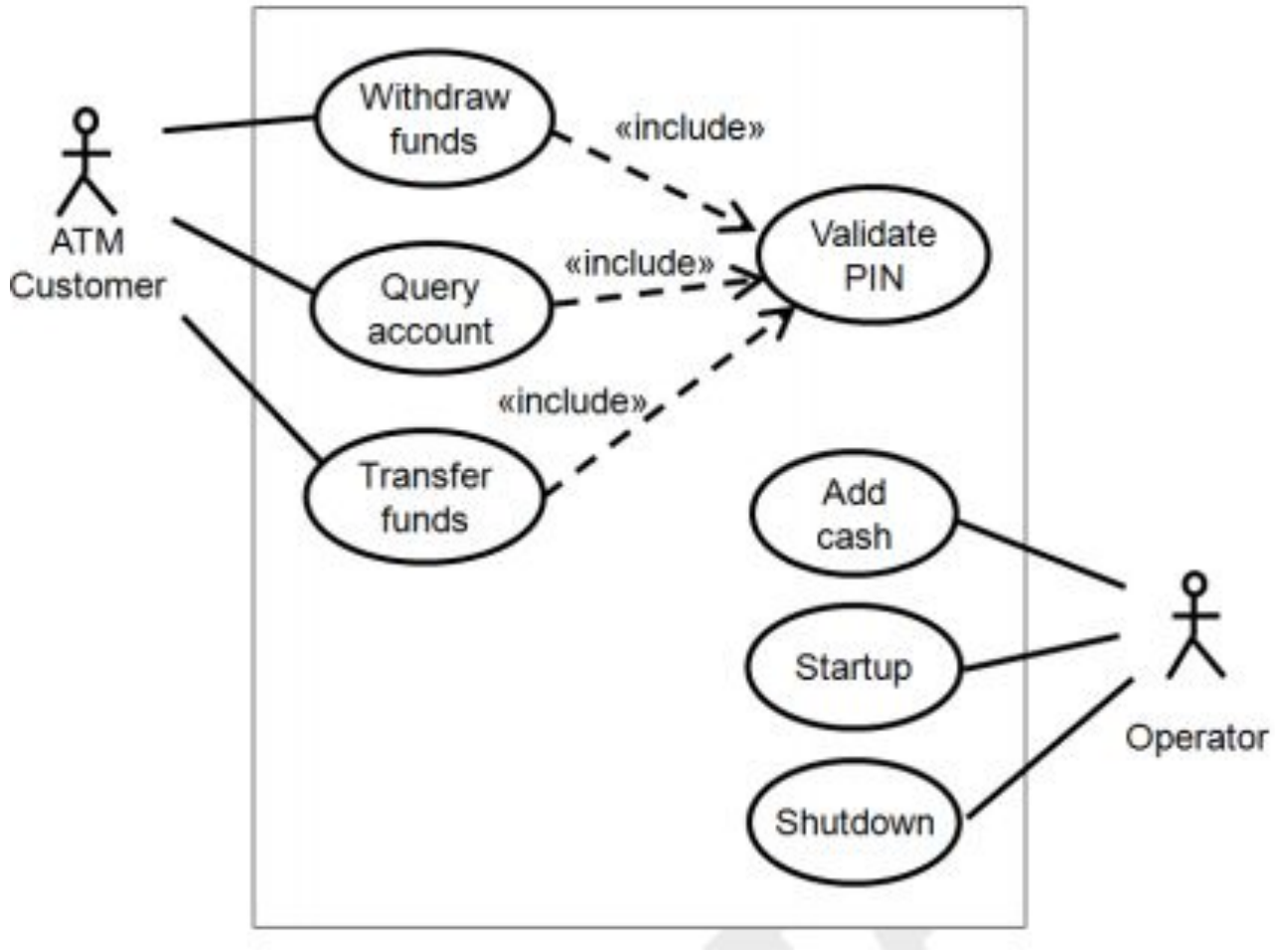


EXTEND

- **The extending use case is usually optional** and can be triggered conditionally. In the diagram, you can see that the extending use case is triggered only for deposits over 10,000 or when the age is over 55.



USE CASE DIAGRAM FOR ATM



Use Case Example

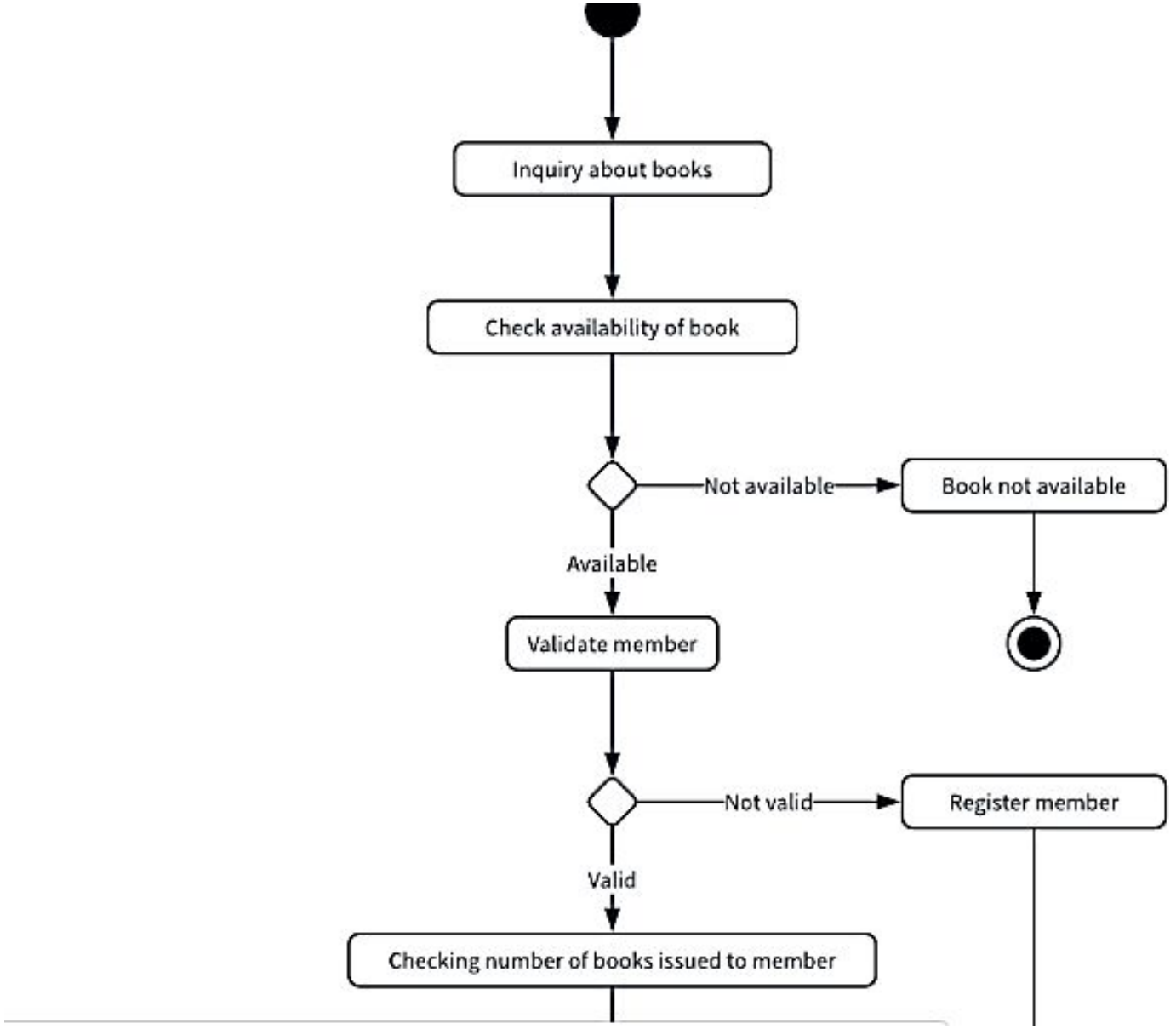
Name	Validate PIN
Summary	System validates customer PIN
Dependency	none
Actors	ATM Customer
Preconditions	ATM is idle, displaying a Welcome message.
Flow of Events	Activity Diagram
Alternatives	• If the system does not recognize the card, the card is ejected. • If the system determines that the card date has expired, the card is confiscated. • If the system determines that the card has been reported lost or stolen, the card is confiscated. • If the customer-entered PIN does not match the PIN number for this card, the system re-prompts for PIN. • If the customer enter the incorrect PIN three times, the system confiscates the card. • If the customer enters Cancel, the system cancels the transaction and ejects the card
Post condition	Customer PIN has been validated.

BEHAVIORAL MODELING

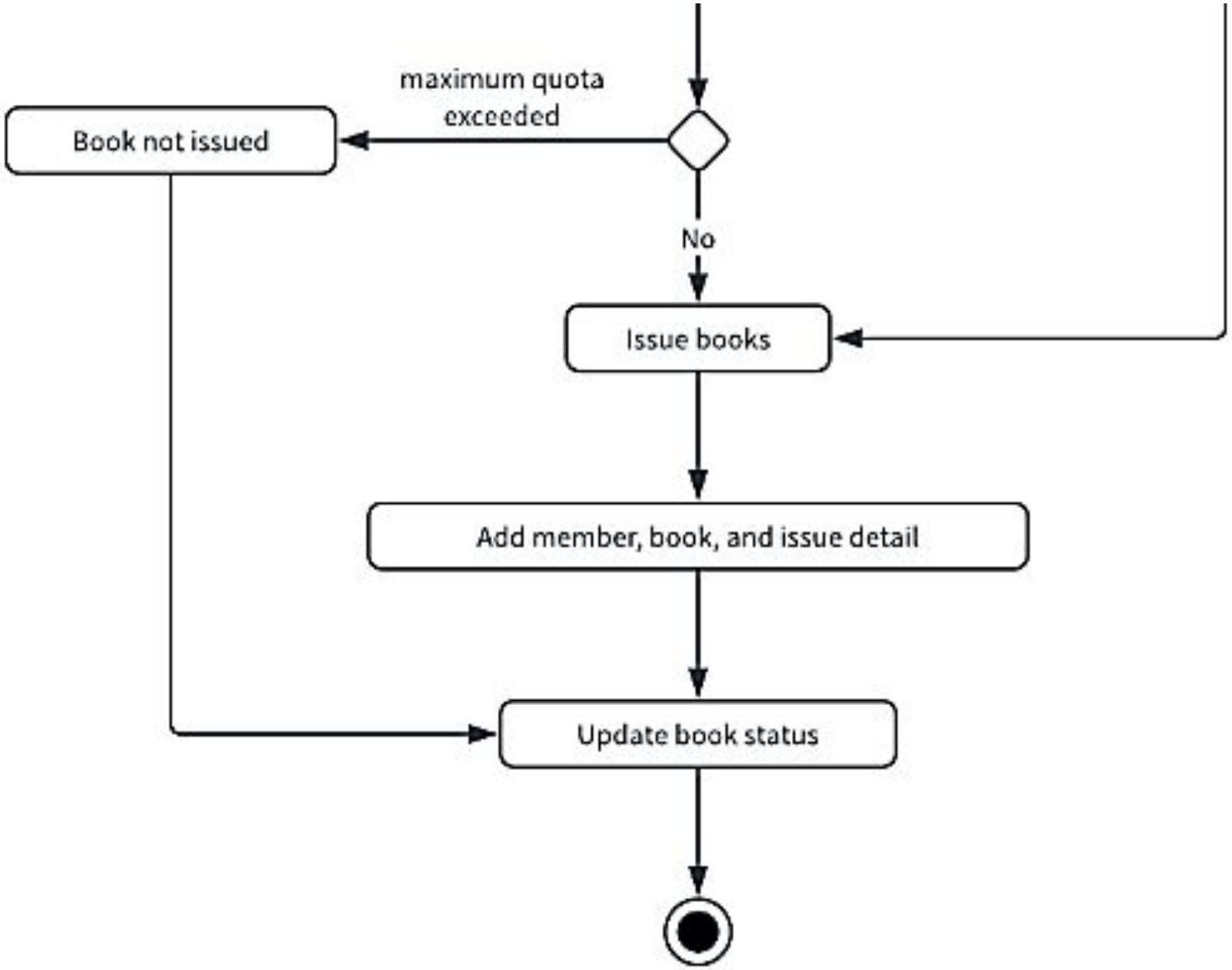
ACTIVITY DIAGRAM

- An activity diagram in the use-case model can be used to capture the activities and actions performed in a use case.
- It expresses the dynamic aspect of the system.

Activity diagram



Activity diagram



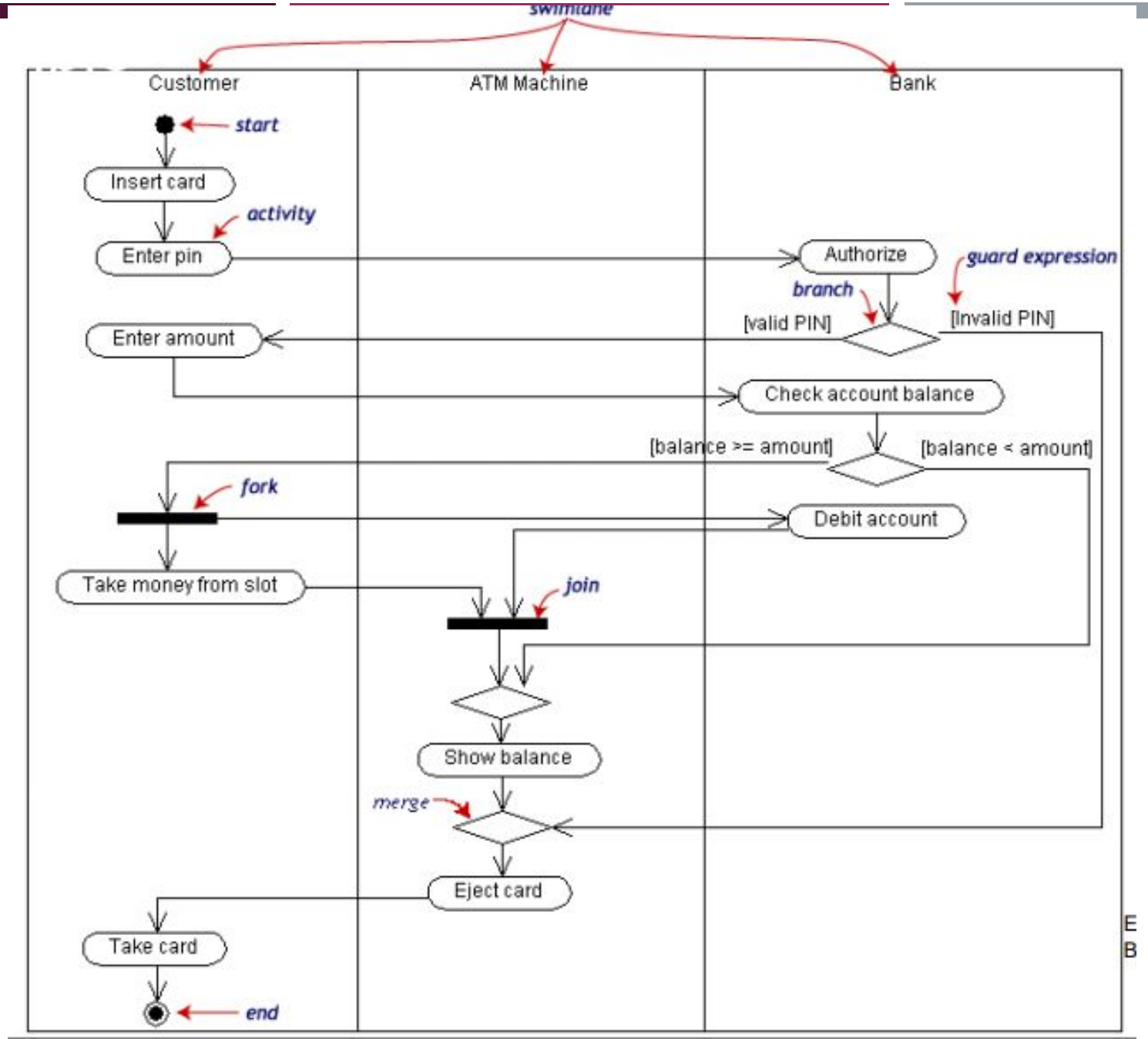
SWIMLANE DIAGRAM

Mapping Who Does What to Whom

You are assigning a responsibility to an actor.

Note, we did not say to an object - to an actor.

Swimlane Diagram

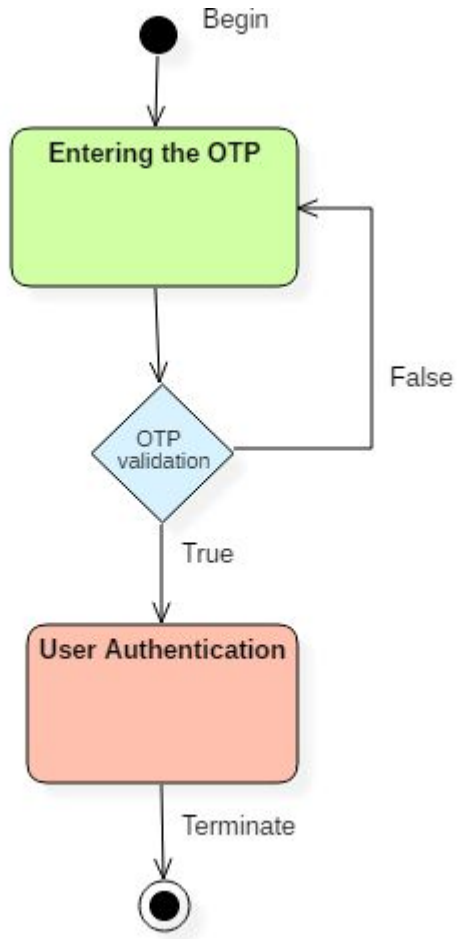


STATE MACHINE DIAGRAM

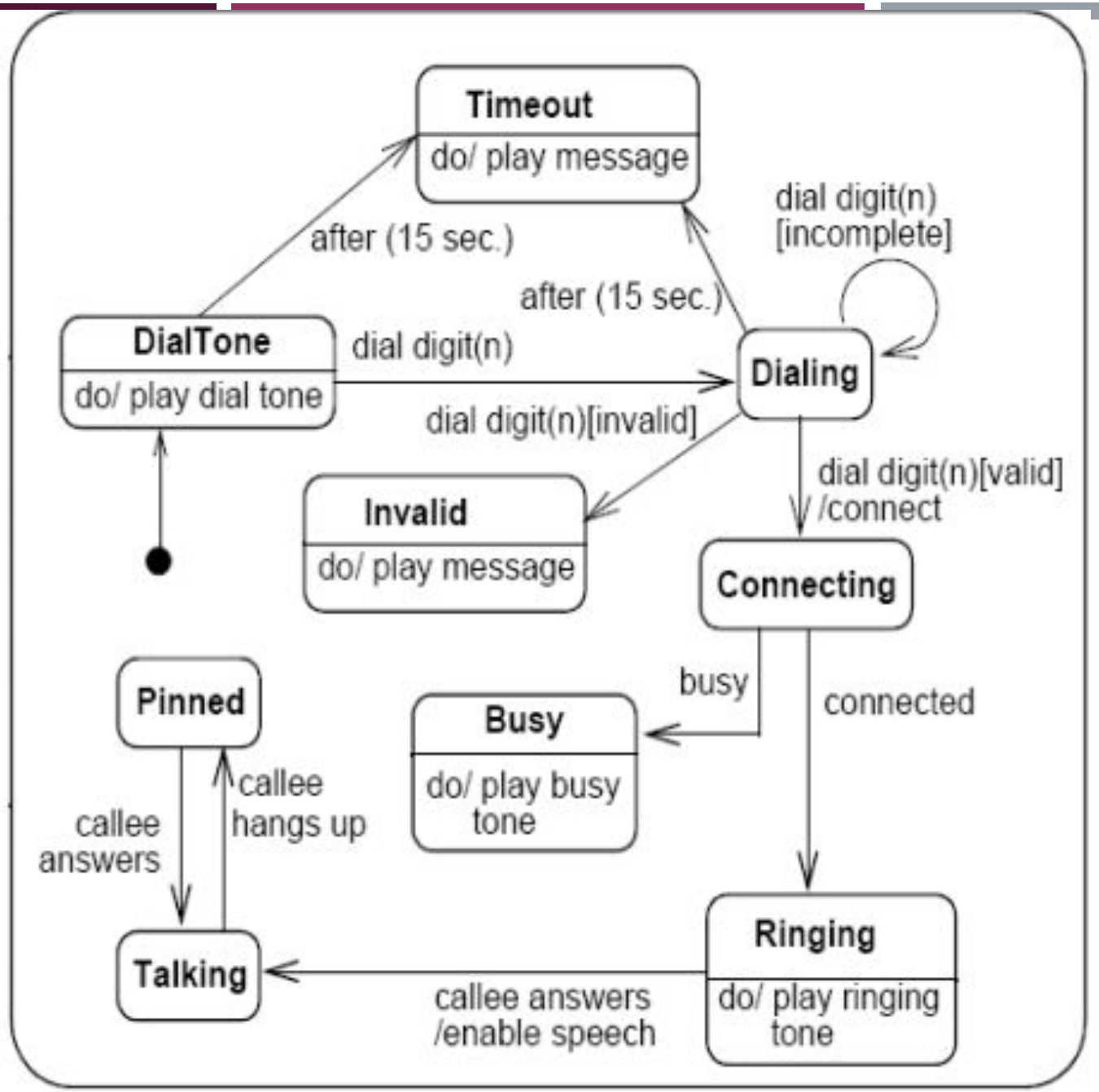
- A state machine diagram models dynamic behavior.
- It specifies the sequence of states in which an object can exist:
- The events and conditions that cause the object to reach those states.
- The actions that take place when those states are reached.

ELEMENT OF STATE MACHINE DIAGRAM

- States
- Events
- Transition

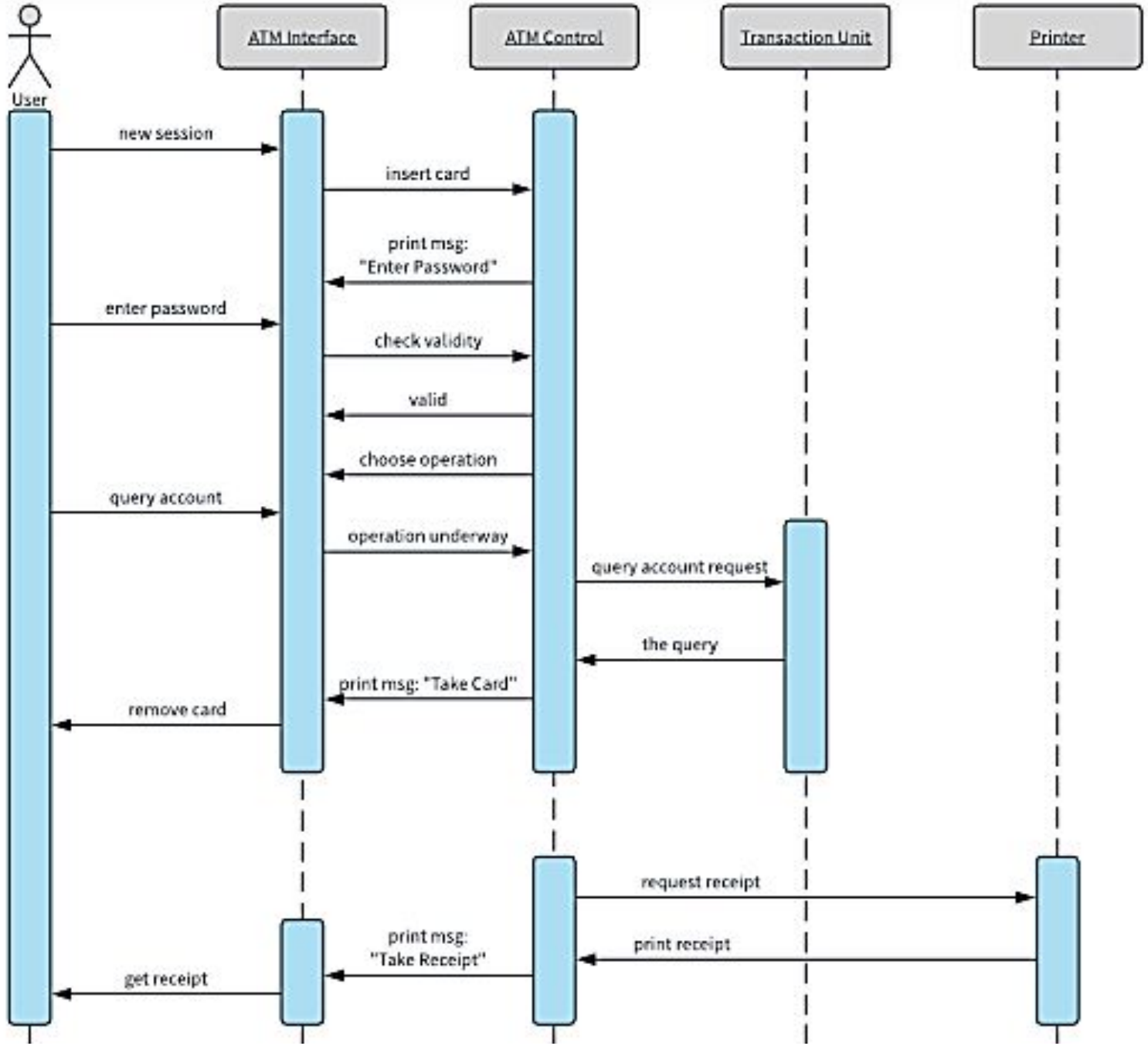


- An object has state, behavior, and a unique identity.
- An object goes through various states during its lifespan.

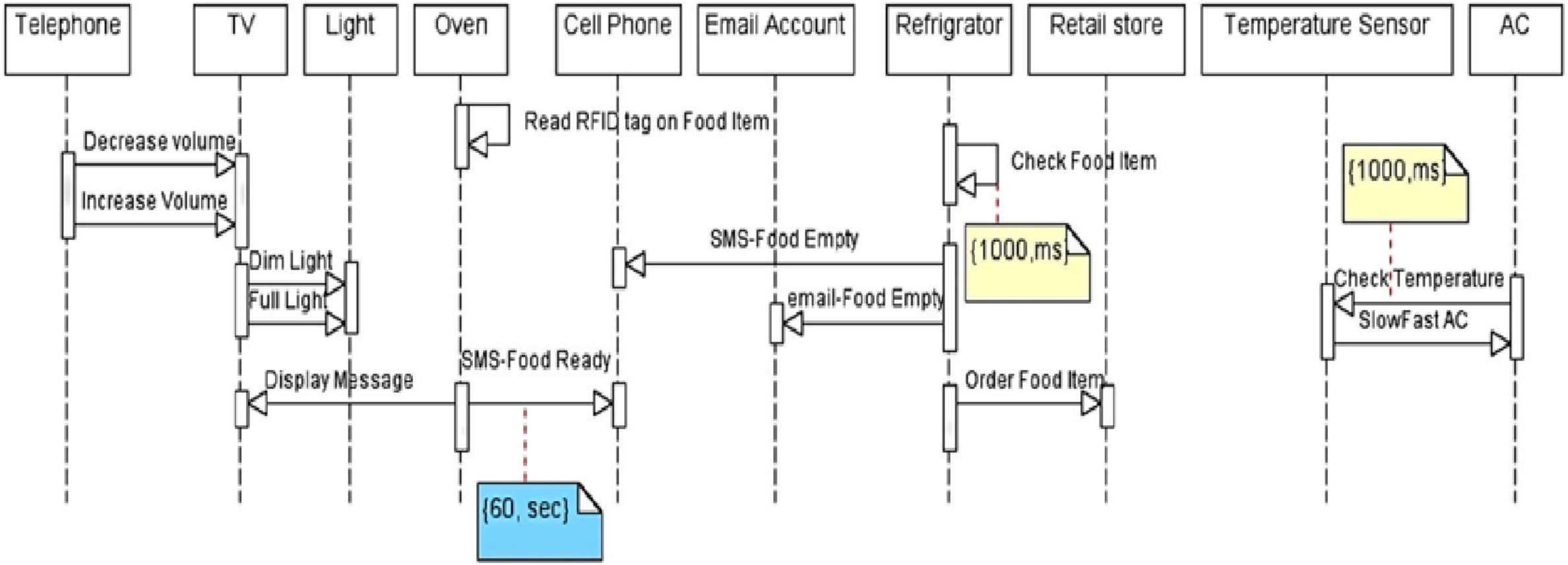

Timeout
do/ play message
after (15 sec.)
dial digit(n)
[incomplete]
after (15 sec.)
DialTone
do/ play dial tone
dial digit(n)
Dialing
dial digit(n)[invalid]
Invalid
do/ play message
dial digit(n)[valid]
/connect
Connecting
Pinned
busy
Busy
do/ play busy
tone
connected
callee
hangs up
callee
answers
Ringing
Talking
callee answers
/enable speech
do/ play ringing
tone

SEQUENCE DIAGRAM

- A **sequence diagram** shows object interactions.
- It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

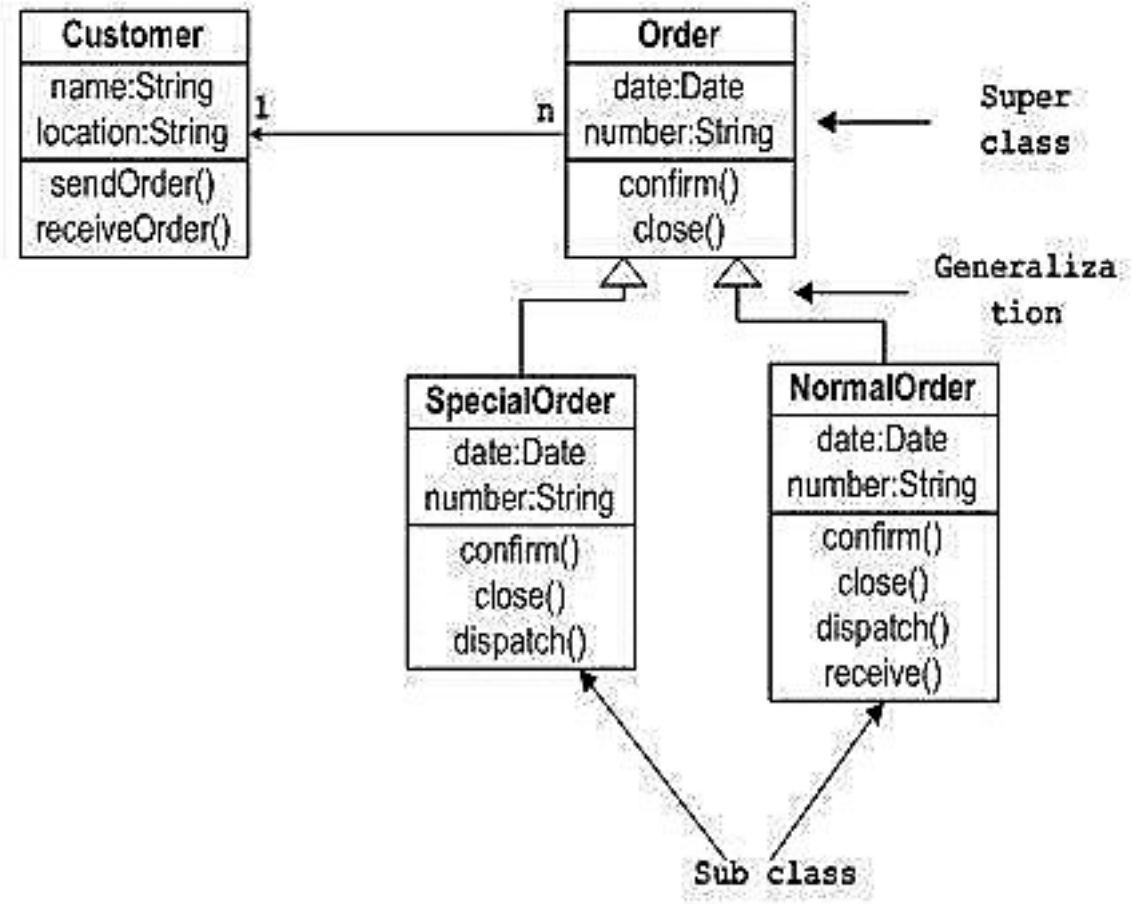

User
ATM Interface
ATM Control
Transaction Unit
Printer
new session
insert card
print msg:
"Enter Password"
enter password
check validity
valid
choose operation
query account
operation underway
query account request
the query
print msg: "Take Card"
remove card
request receipt
print msg:
"Take Receipt"
print receipt
get receipt

SYSTEM SEQUENCE DIAGRAM

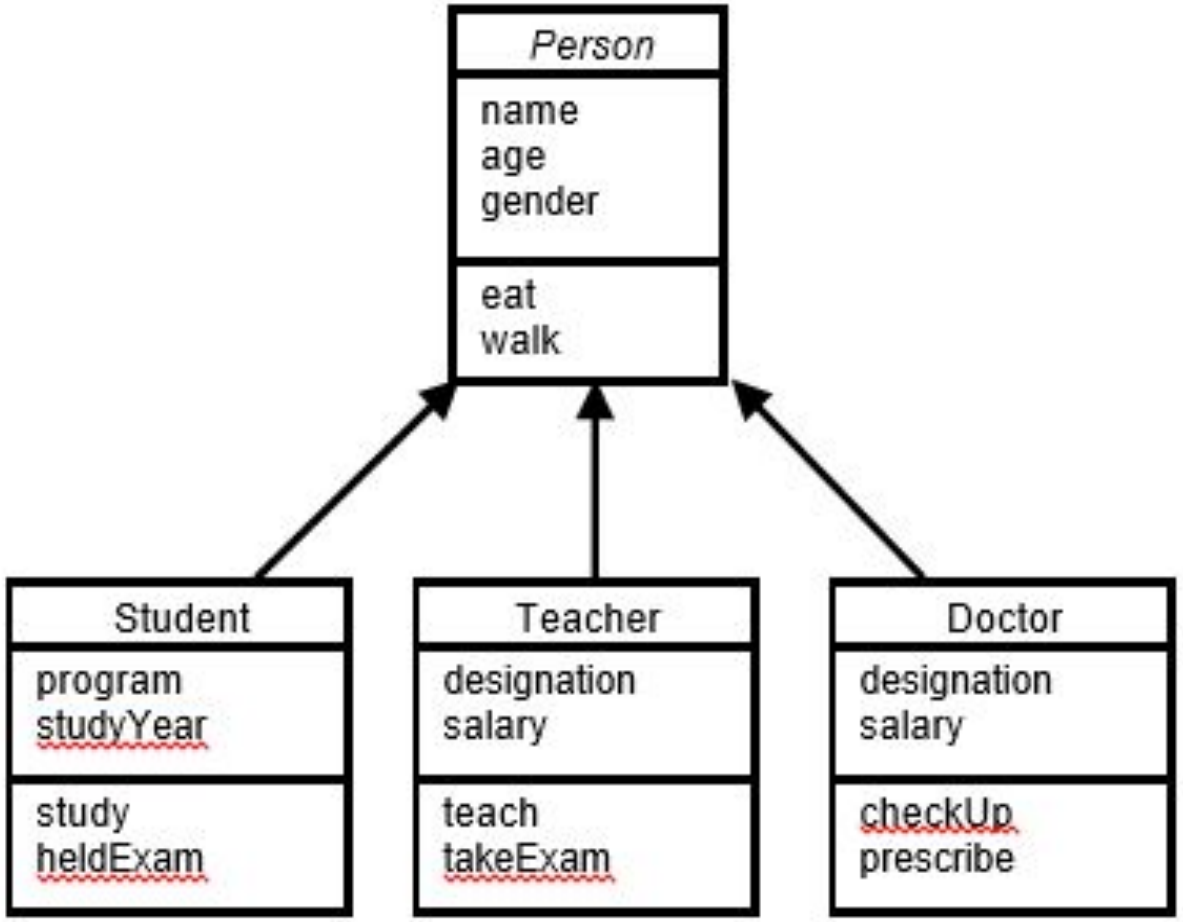


STRUCTURED ANALYSIS OF OBJECT ORIENTED PARADIGM

CLASS DIAGRAM



INHERITANCE



Advantages of Inheritance

1. *Reuse*
2. *Less redundancy*
3. *Increased maintainability*

KINDS OF ASSOCIATION

There are two main types of association which are then further subdivided i.e.

- Class Association
- Object Association

CLASS ASSOCIATION

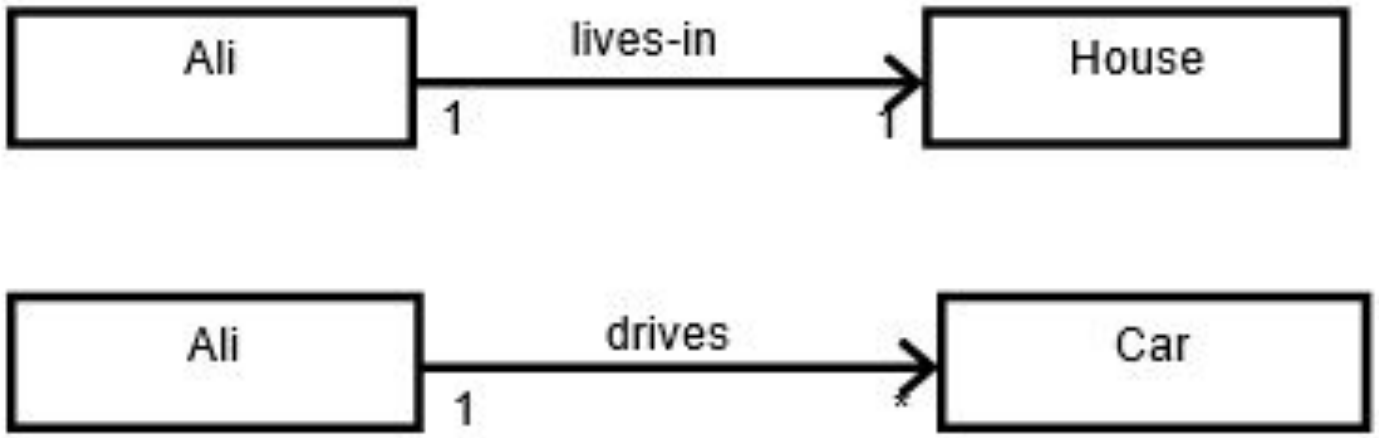
- *Class association is implemented in terms of Inheritance.*
- *Inheritance implements generalization/specialization relationship between objects.*
- *Inheritance is considered class association.*

OBJECT ASSOCIATION

It can be of one of the following types,

- *Simple Association*
- *Composition*
- *Aggregation*

SIMPLE ASSOCIATION



It is generally called as "**association**" instead of "**simple association**"

KINDS OF SIMPLE ASSOCIATION

Simple association can be categorized in two ways,

- With respect to direction (navigation)
- With respect to number of objects (cardinality)

KINDS OF SIMPLE ASSOCIATION W.R.T NAVIGATION

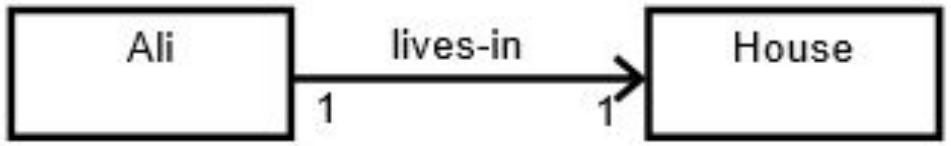
With respect to navigation association has the following types,

- One-way Association
- Two-way Association

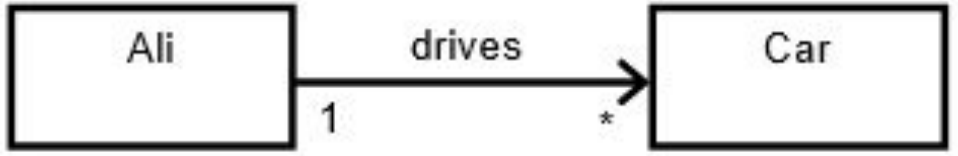
ONE-WAY ASSOCIATION

In one-way association we can navigate along a single direction only, it is denoted by an arrow towards the server object.

- **Examples:**



- Ali lives in a House

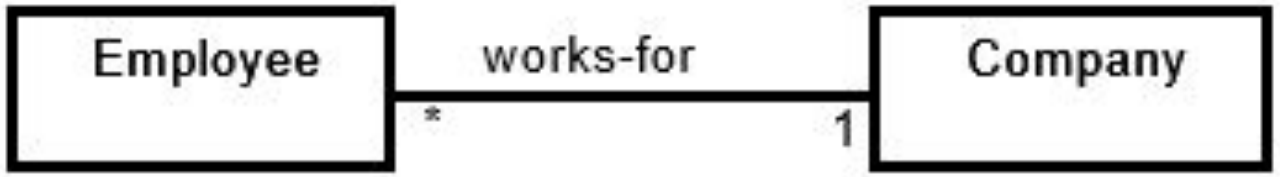


- Ali drives his Car

TWO-WAY ASSOCIATION

In two-way association we can navigate in both directions, it is denoted by a line between the associated objects.

Exar



Employee works for company
Company employs employees

KINDS OF SIMPLE ASSOCIATION W.R.T CARDINALITY

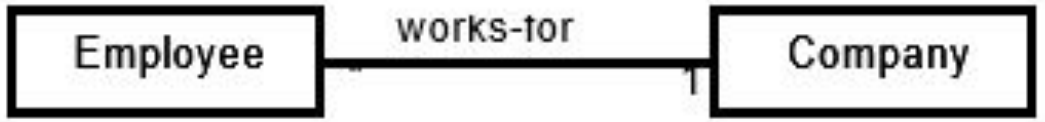
With respect to cardinality association has the following types,

- Binary Association
- Ternary Association
- N-ary Association

BINARY ASSOCIATION

It associates objects of exactly two classes; it is denoted by a line, or an arrow between the associated objects.

Example



Association "works-for" associates objects of exactly two classes

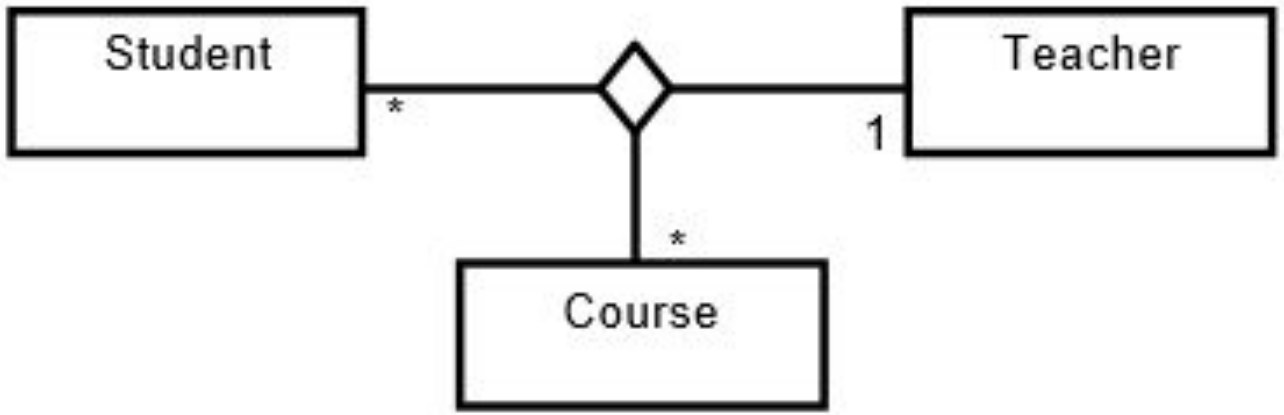
Association "drives" associates objects of exactly two classes

TERNARY ASSOCIATION

It associates objects of exactly three classes; it is denoted by a diamond with lines connected to associated objects.

Example:

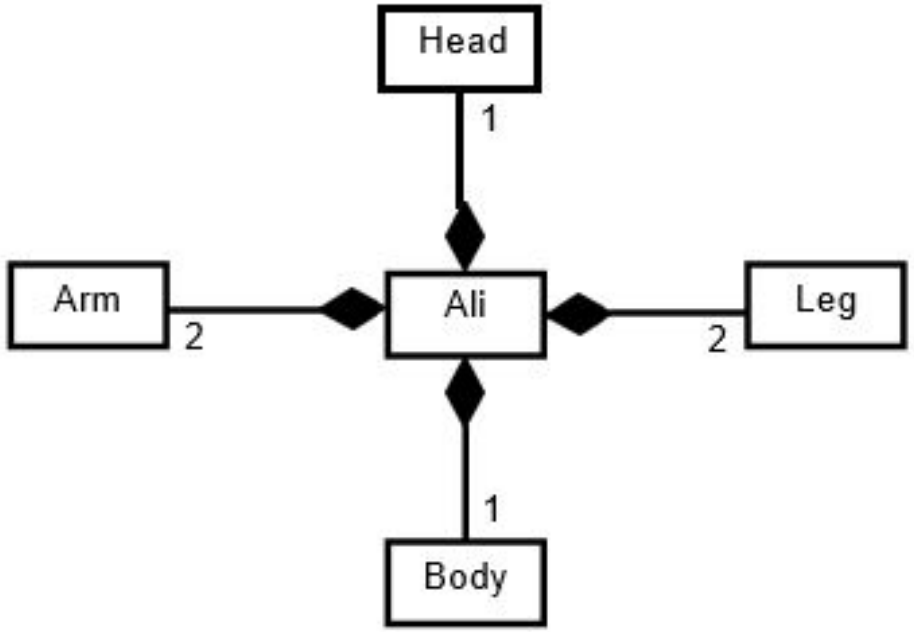
Objects of exactly three classes are associated



COMPOSITION

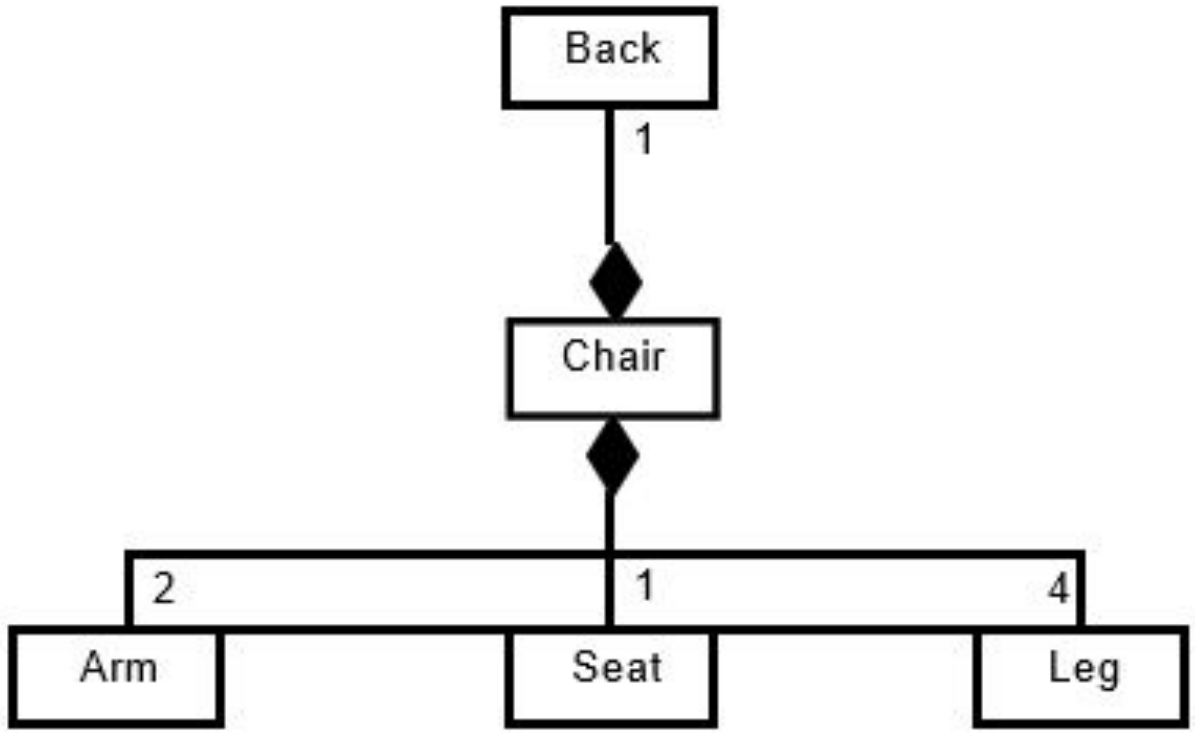
An object may be composed of other smaller objects, the relationship between the "part" objects and the "whole" object is known as Composition.

Composition is represented by a line with a filled-diamond head towards the composer object.



COMPOSITION

Composition of chair:



COMPOSITION

Composition is stronger relationship because

- *Composed object becomes a part of the composer.*
- *Composed object can't exist independently.*

Example I

- Ali is made up of different body parts They can't exist independent of Ali

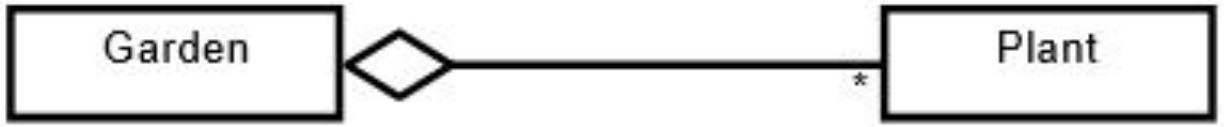
Example II

- Chair's body is made up of different parts They can't exist independently

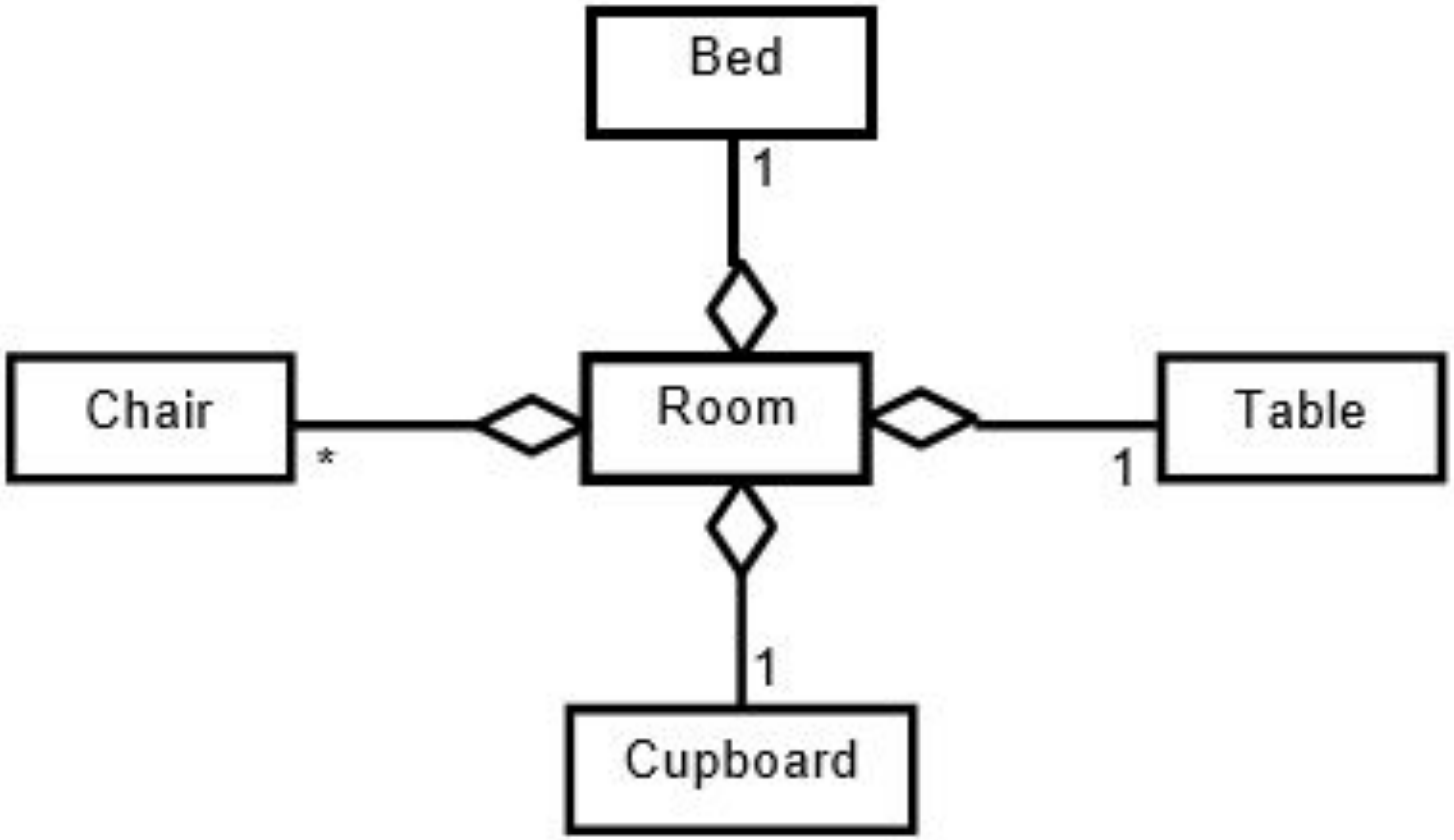
AGGREGATION

- An object may contain a collection (aggregate) of other objects, the relationship between the container and the contained object is called aggregation,
- Aggregation is represented by a line with unfilled-diamond head towards the container.

Example – Aggregation



AGGREGATION



AGGREGATION

Aggregation is weaker relationship because

- *Aggregate object is not a part of the container*
- *Aggregate object can exist independently*

Example I

- Furniture is not an intrinsic part of room
- Furniture can be shifted to another room, and so can exist independent of a particular room

Example II

- A plant is not an intrinsic part of a garden
- It can be planted in some other garden, and so can exist independent of a particular garden

POLYMORPHISM

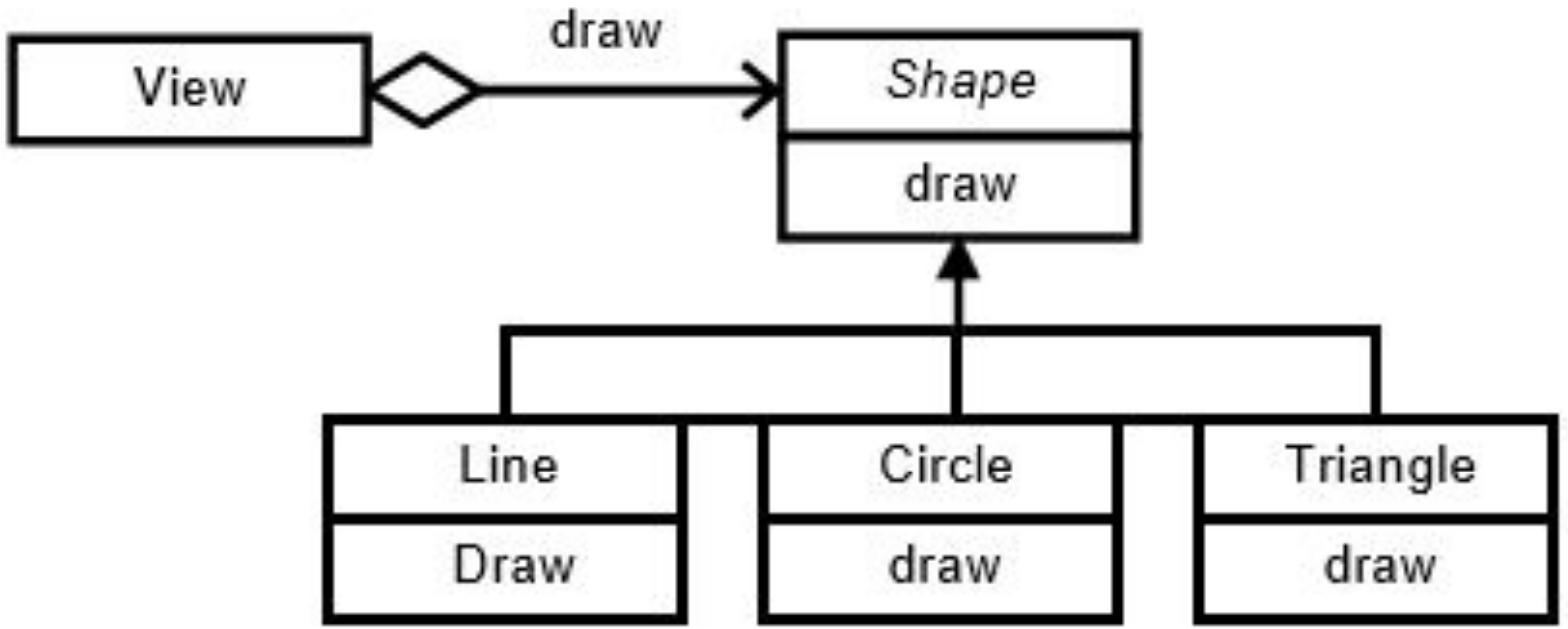
It is also essential component of object oriented modeling (paradigm).

In general, polymorphism refers to existence of ***different forms*** of a single entity.

For example, both Diamond and Coal are different forms of Carbon.

- *In OO model, polymorphism means that different objects can behave in different ways for the same message (stimulus).*

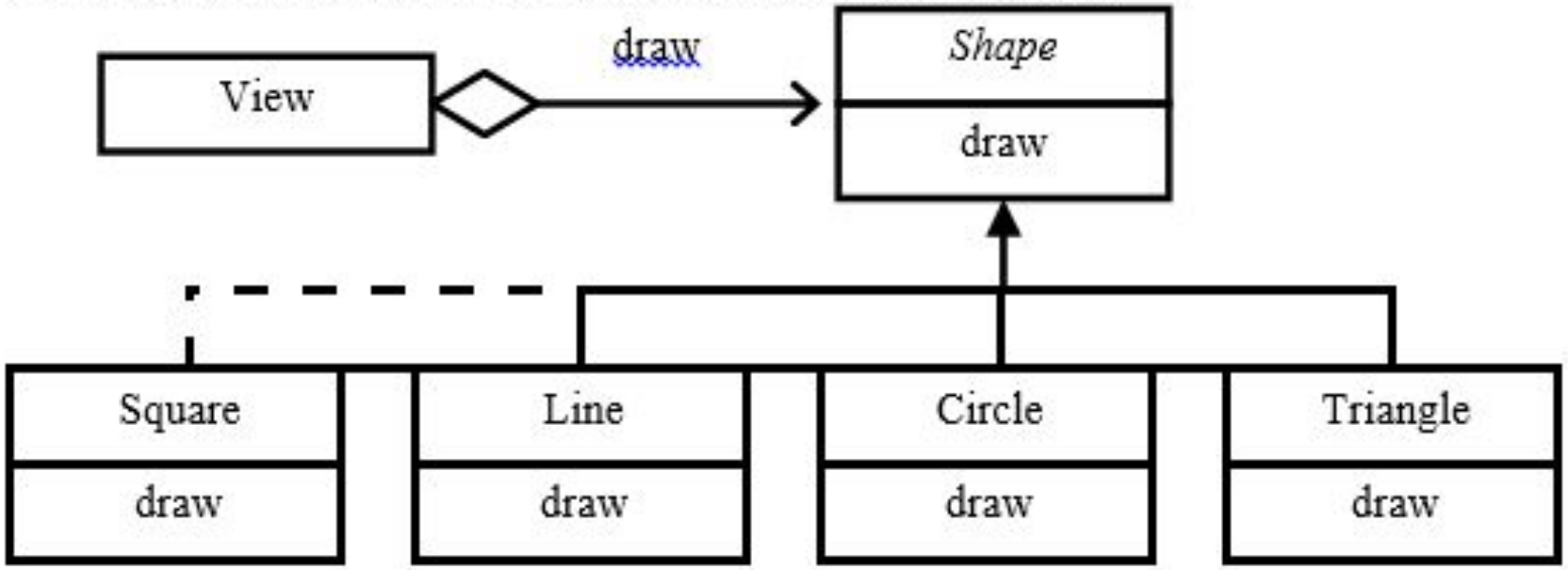
POLYMORPHISM



POLYMORPHISM ADVANTAGES

Messages can be interpreted in different ways depending upon the receiver class.

New classes can be added without changing the existing model.



In general, polymorphism is a powerful tool to develop flexible and reusable systems

ASSIGNMENT # 1

Create a working plan for banking system case study using agile process model (Extreme programming). You are required to

- Identify functional requirements (FR's) from the case study and write User Stories for each FR in order to have detail understanding.
- Create Iteration Plans.
- Perform Test First Development (Write test descriptions for user story cards).

ASSIGNMENT # 2

You are required to create a SRS (Software Requirement Specification) document for banking system case study. Steps for creating SRS,

After capturing functional requirements through user-story cards, state non-functional requirements as well for banking system.

Requirement Analysis

- User Stories card would help you to capture use-cases. Identify actors from story cards and draw use-cases for a banking system case study.
- Draw Activity Diagram to capyure detail functionalities by elaborating your use cases.
- Draw Swimlane Diagram to link activities with actors.

Specification

- After Analysis, write final set of requirements.

HAVE A GOO DAY!