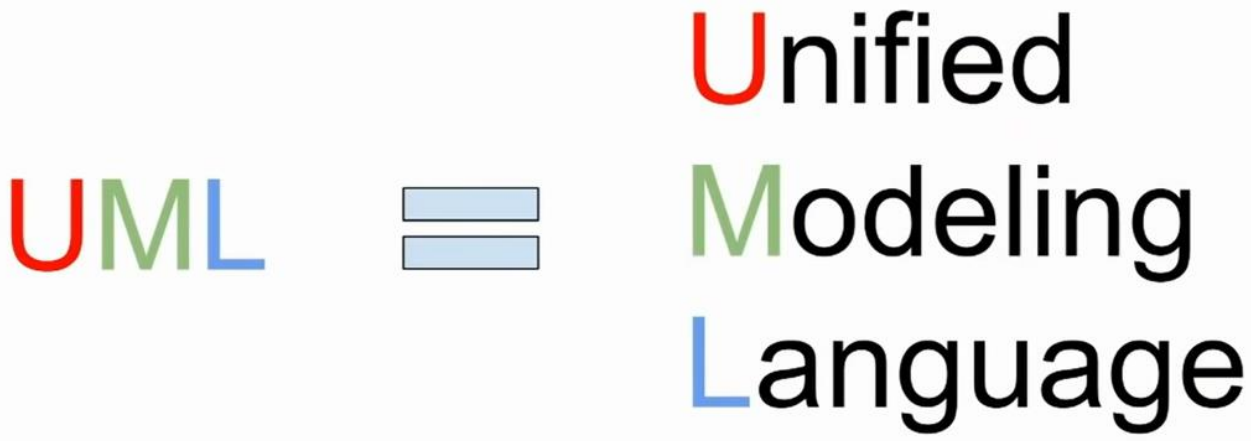
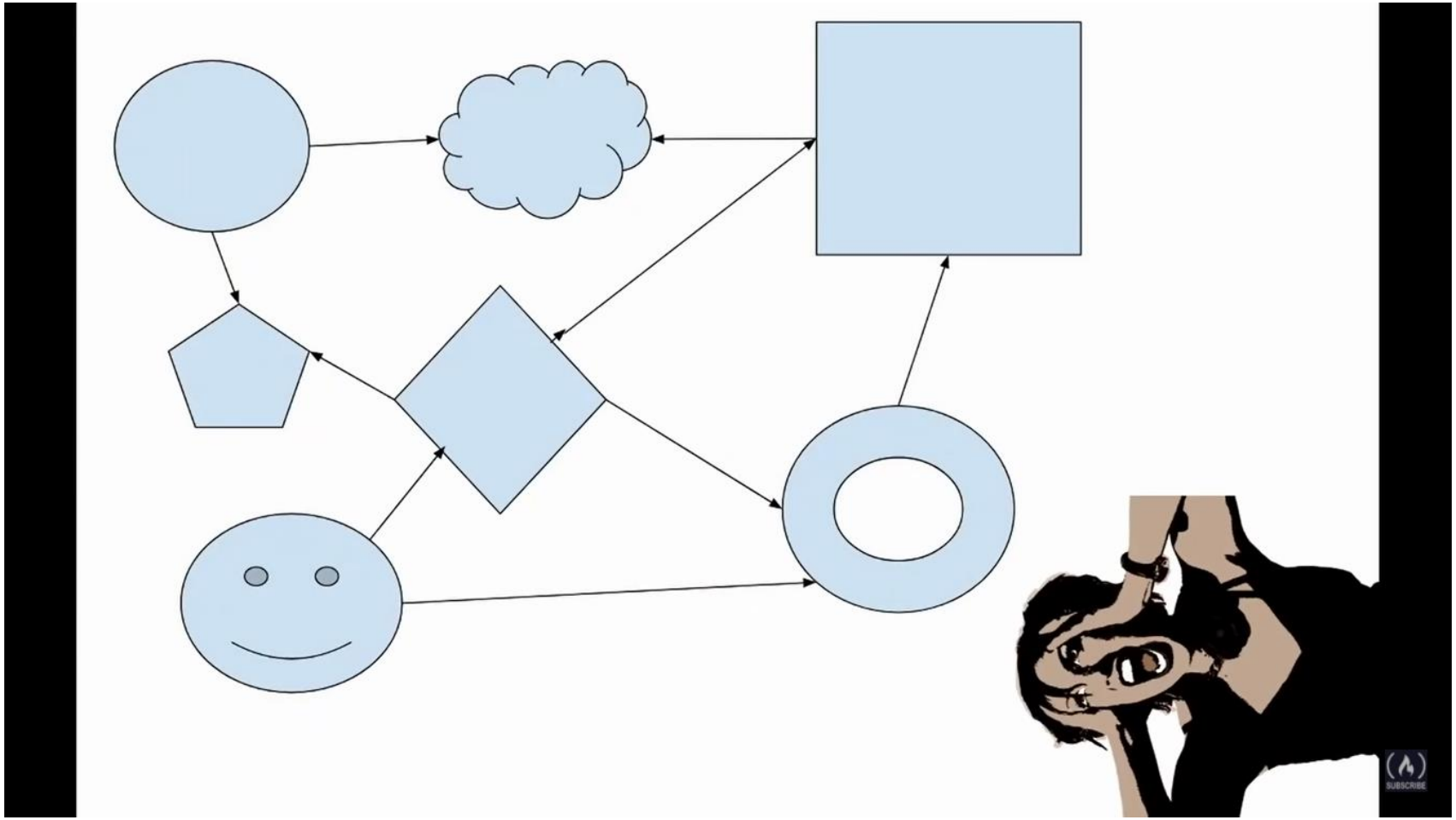
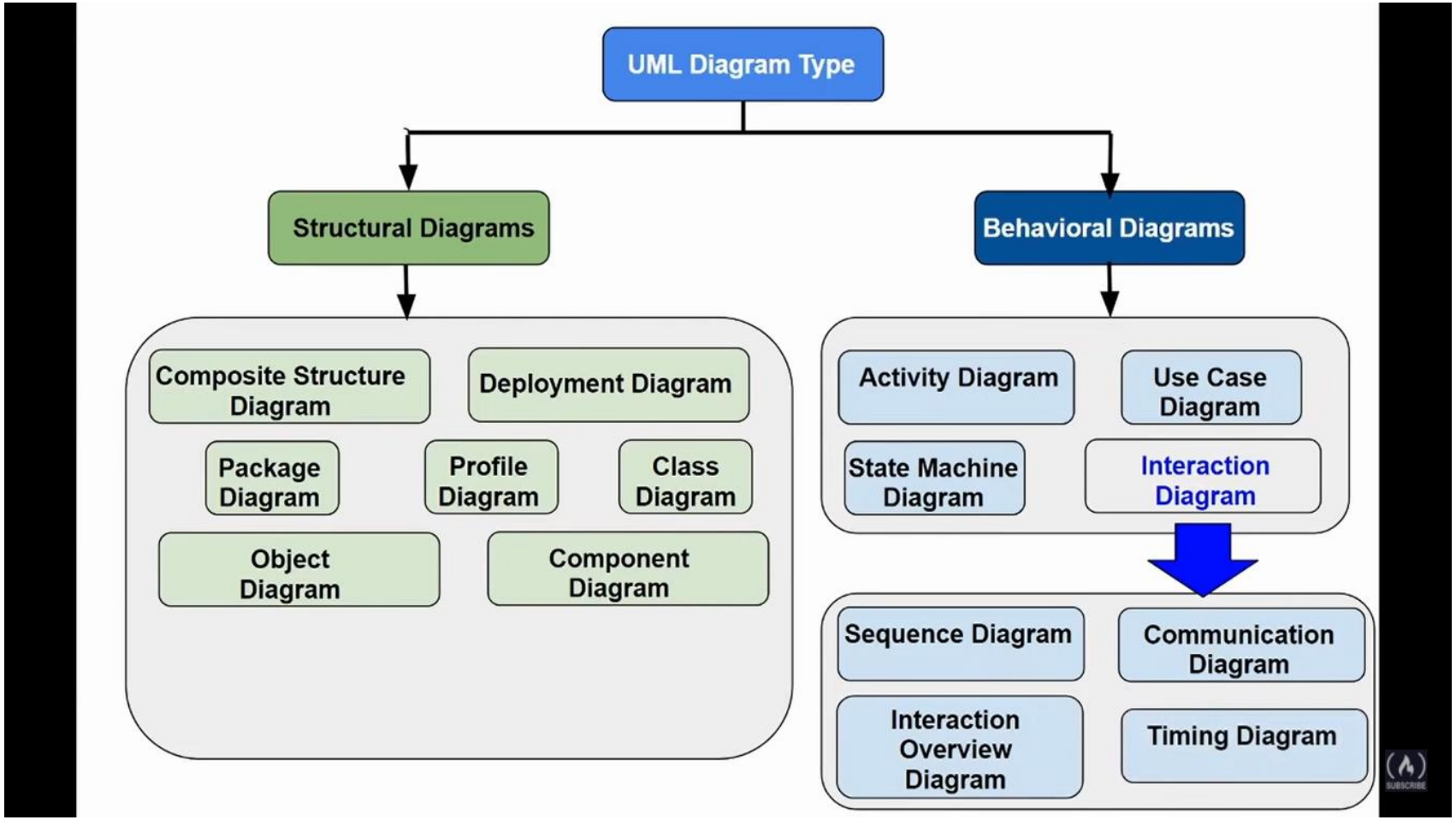
Day 3

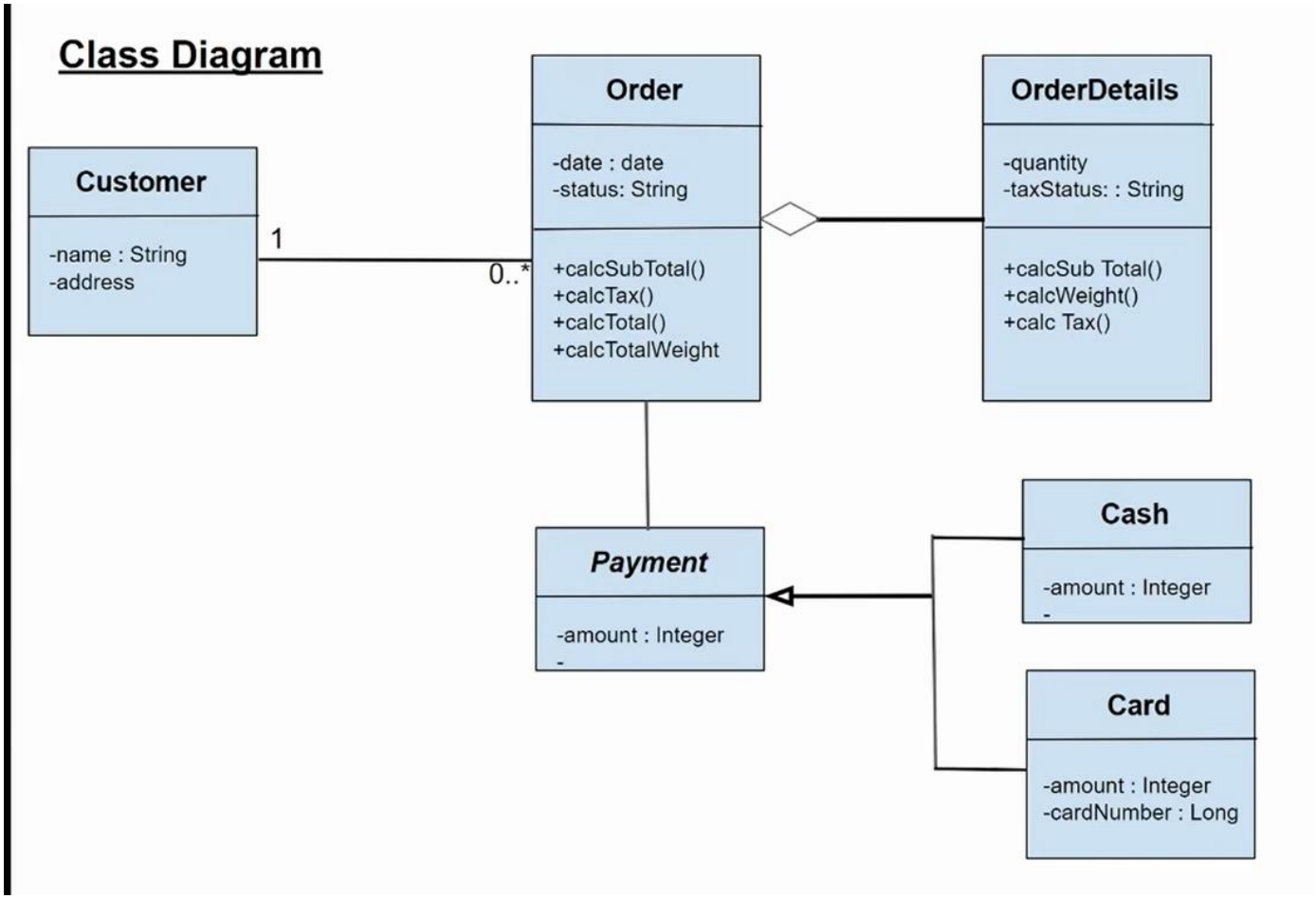
- UML with Example
- Design Engineering + Characterstics
- Architecture Design
- Coding Conventions & Programming Principles
- OOA & Design

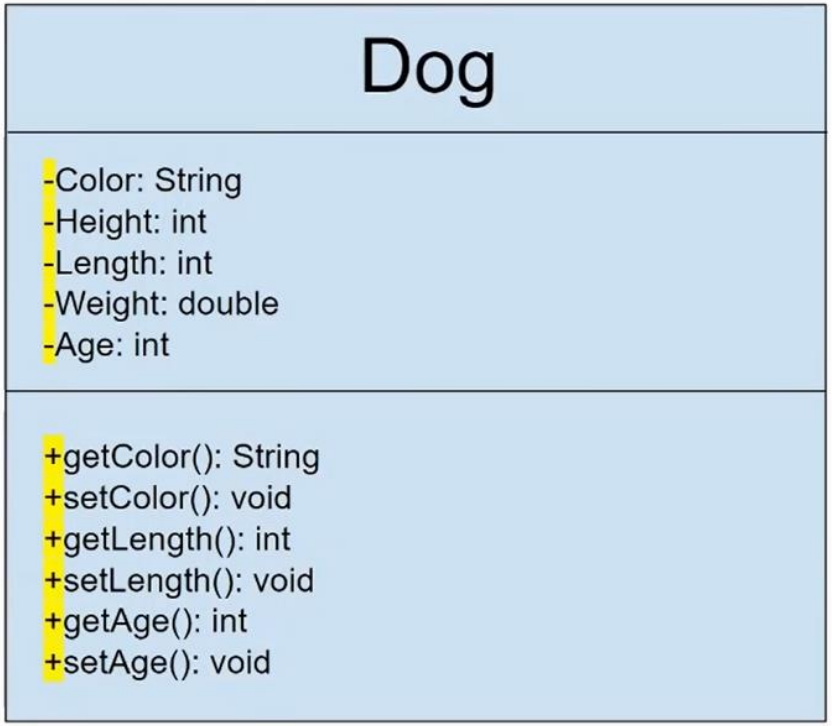
- RE (Communication –Plan)
- Modeling(AM+DM)
- =Phase1 - AM – SRS(High level Abstraction)
- =Phase2- DM (Low level Abstraction)


UML
Unified
Modeling
Language


SUBSCRIBE


UML Diagram Type
Structural Diagrams
Behavioral Diagrams
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Class Diagram
Object Diagram
Component Diagram
Activity Diagram
Use Case Diagram
State Machine Diagram
Interaction Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram


Class Diagram
Customer
-name : String
-address
1
0..*
Order
-date : date
-status: String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight
OrderDetails
-quantity
-taxStatus: : String
+calcSub Total()
+calcWeight()
+calc Tax()
Payment
-amount : Integer
-
Cash
-amount : Integer
-
Card
-amount : Integer
-cardNumber : Long

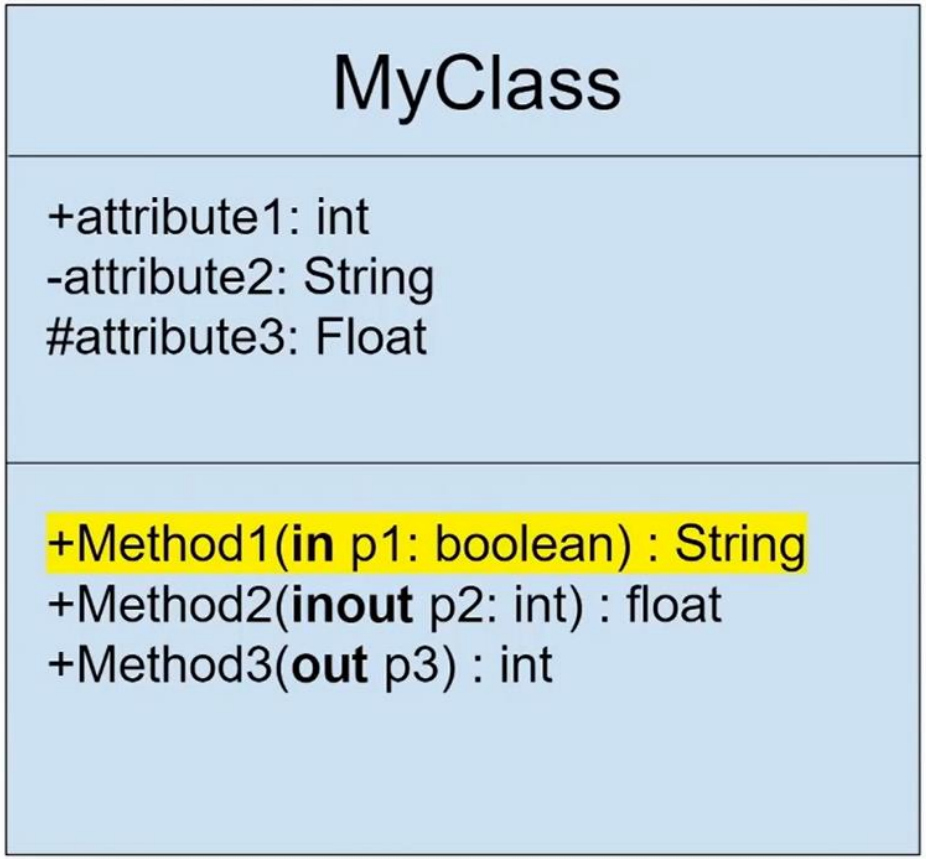

Dog
-Color: String
-Height: int
-Length: int
-Weight: double
-Age: int
+getColor(): String
+setColor(): void
+getLength(): int
+setLength(): void
+getAge(): int
+setAge(): void

Public

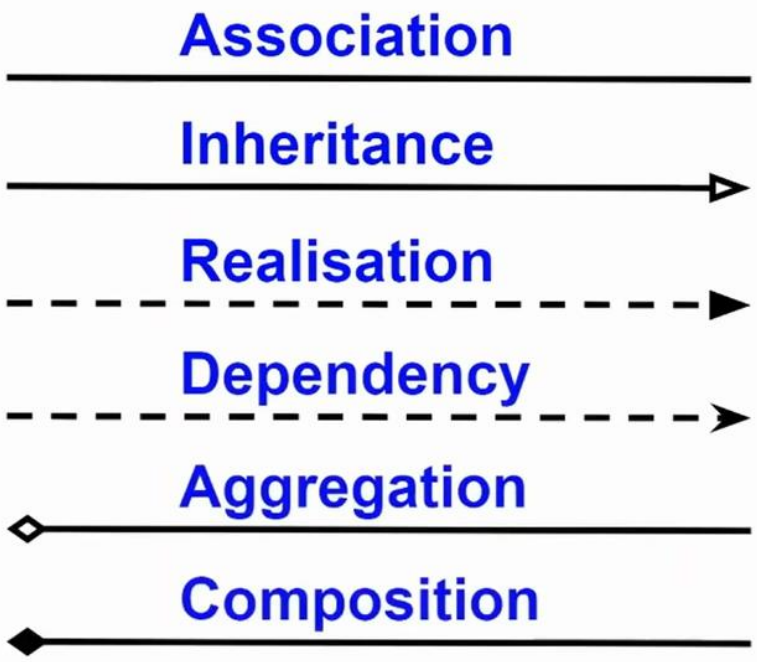
Private

Protected

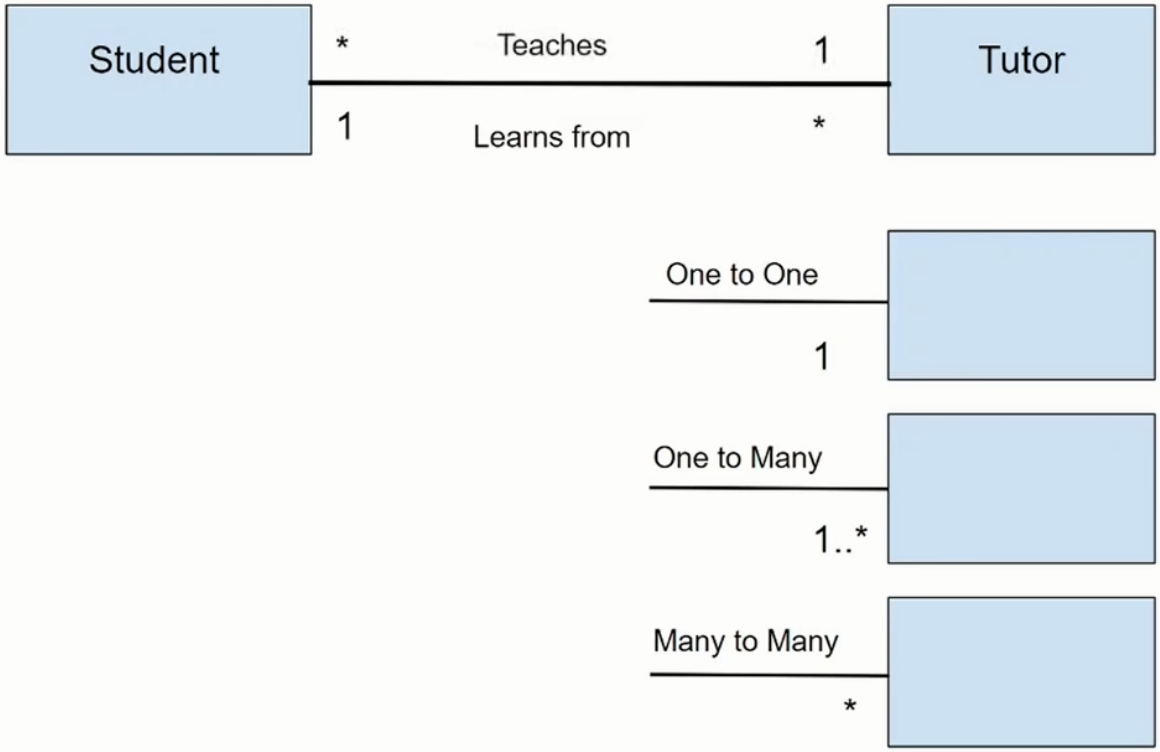
Package Local


MyClass
+attribute1: int
-attribute2: String
#attribute3: Float
+Method1(in p1: boolean) : String
+Method2(inout p2: int) : float
+Method3(out p3) : int

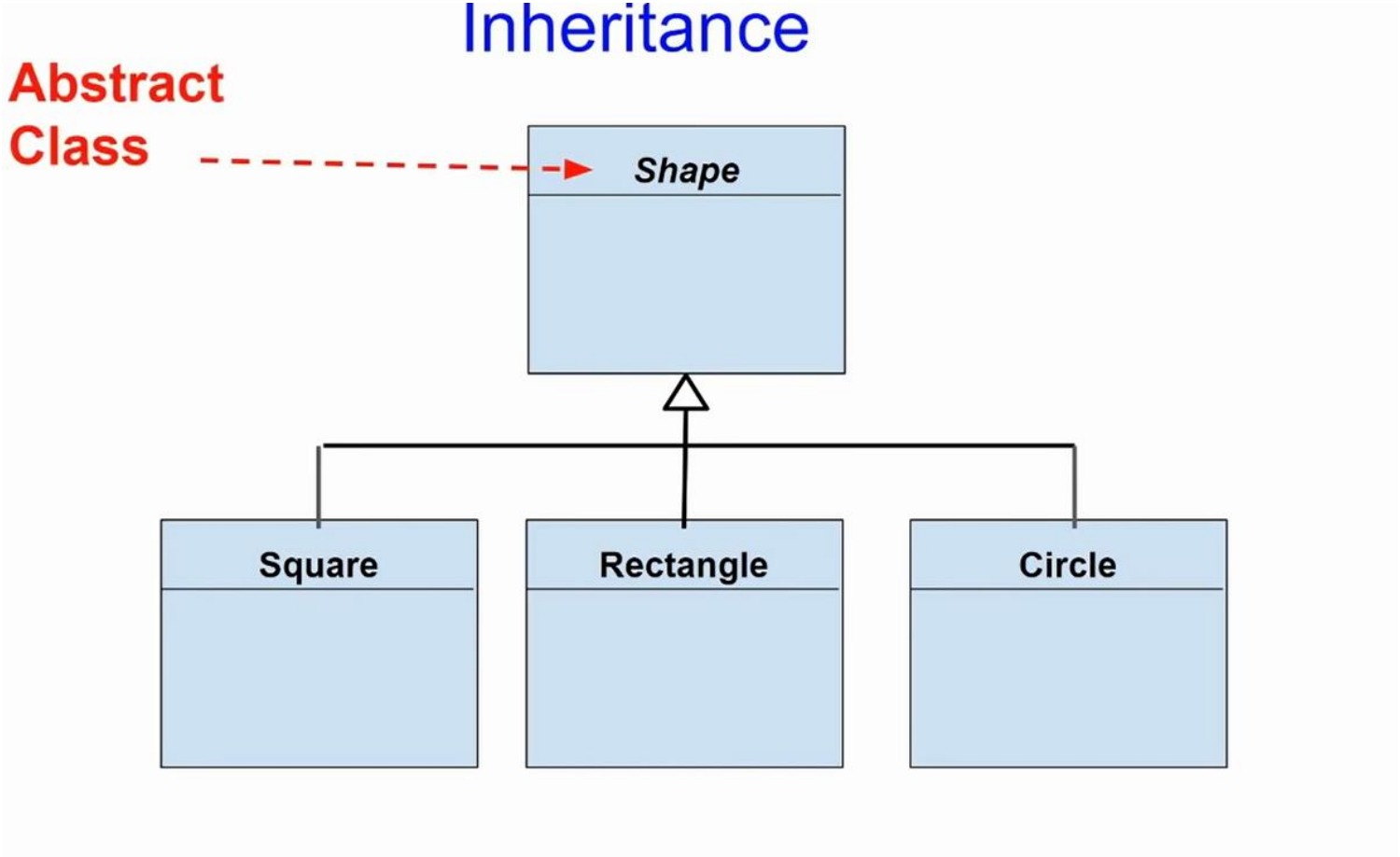
Relationships between classes in UML



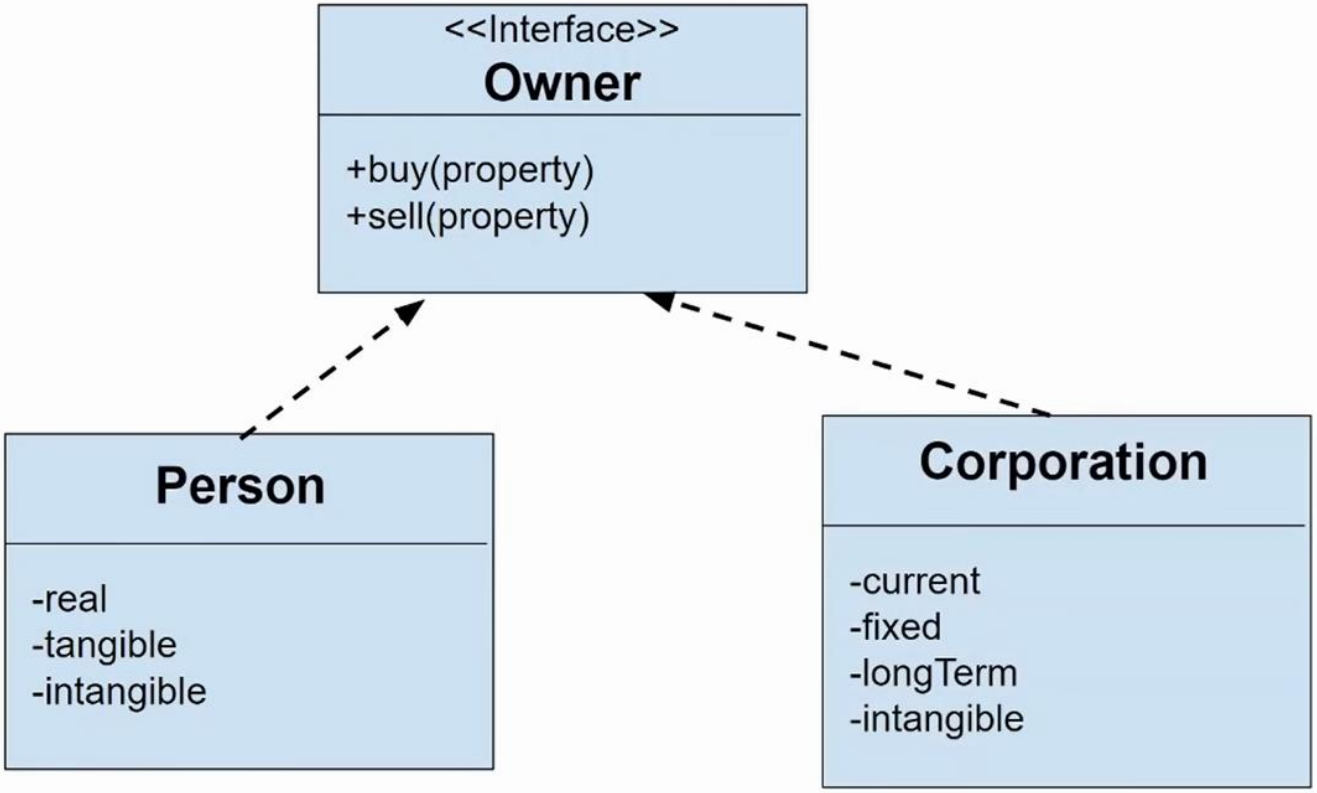
Association



Inheritance

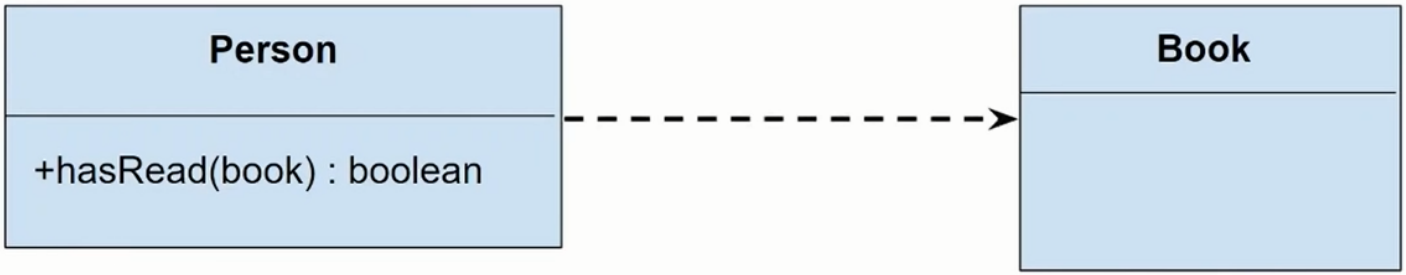


Realization

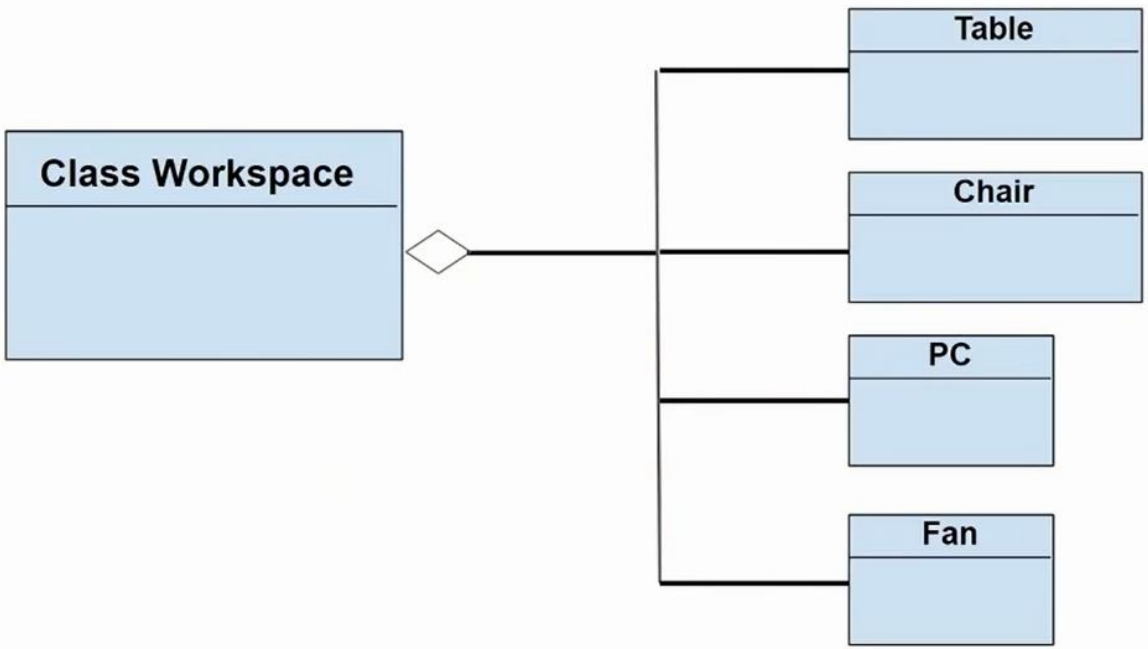


- A class diagram shows a set of classes, interfaces, and collaborations and their relationships.
- These diagrams are the most common diagram found in modeling object-oriented systems.
- Class diagrams address the static design view of a system.
- Class diagrams that include active classes address the static process view of a system.

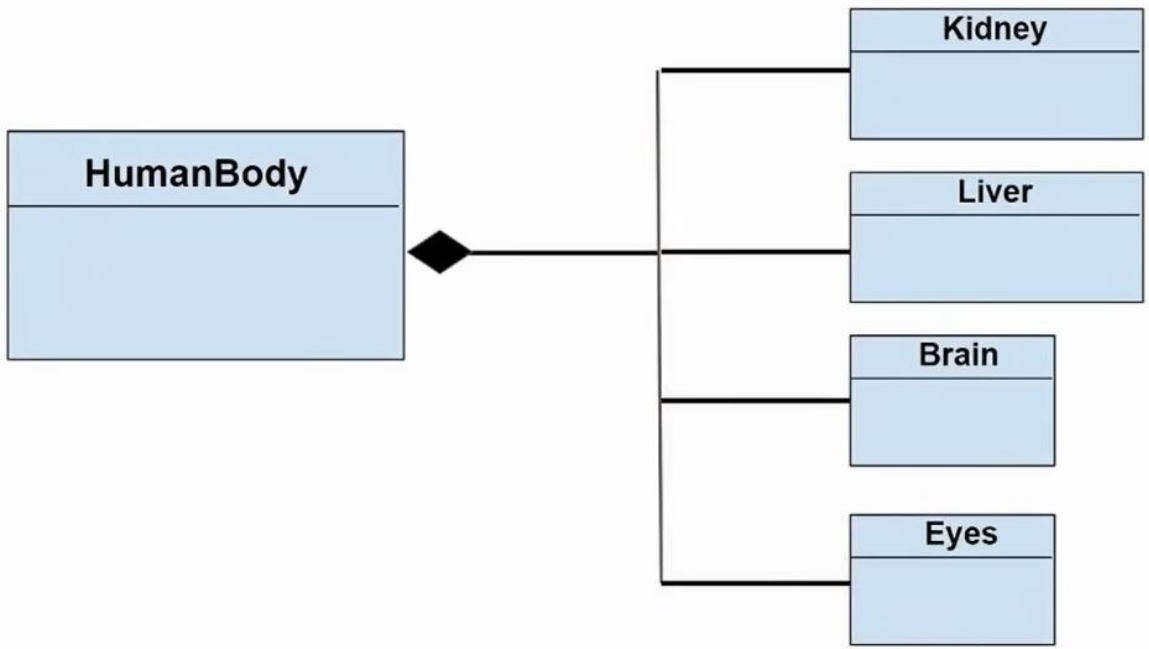
Dependency

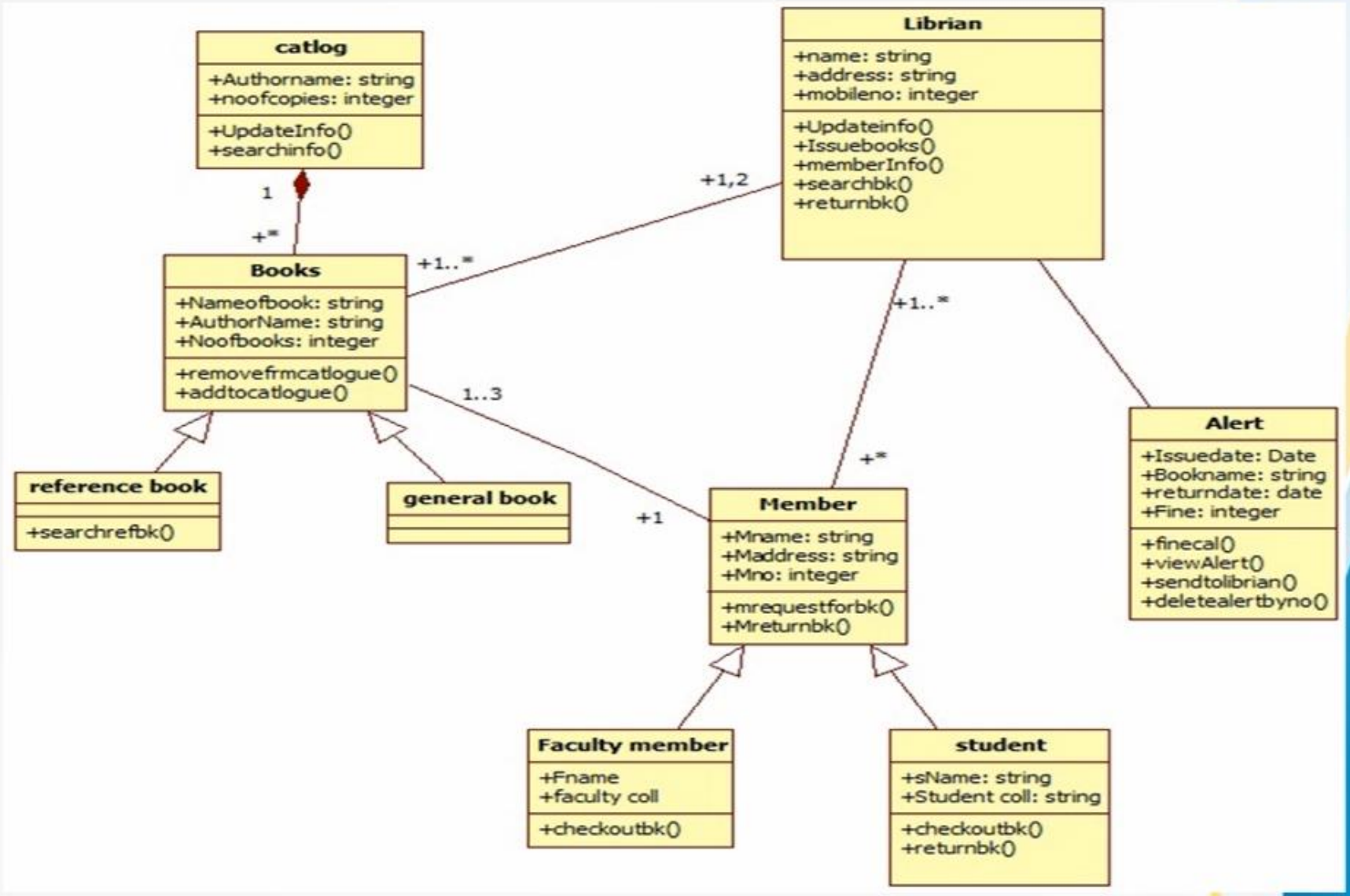


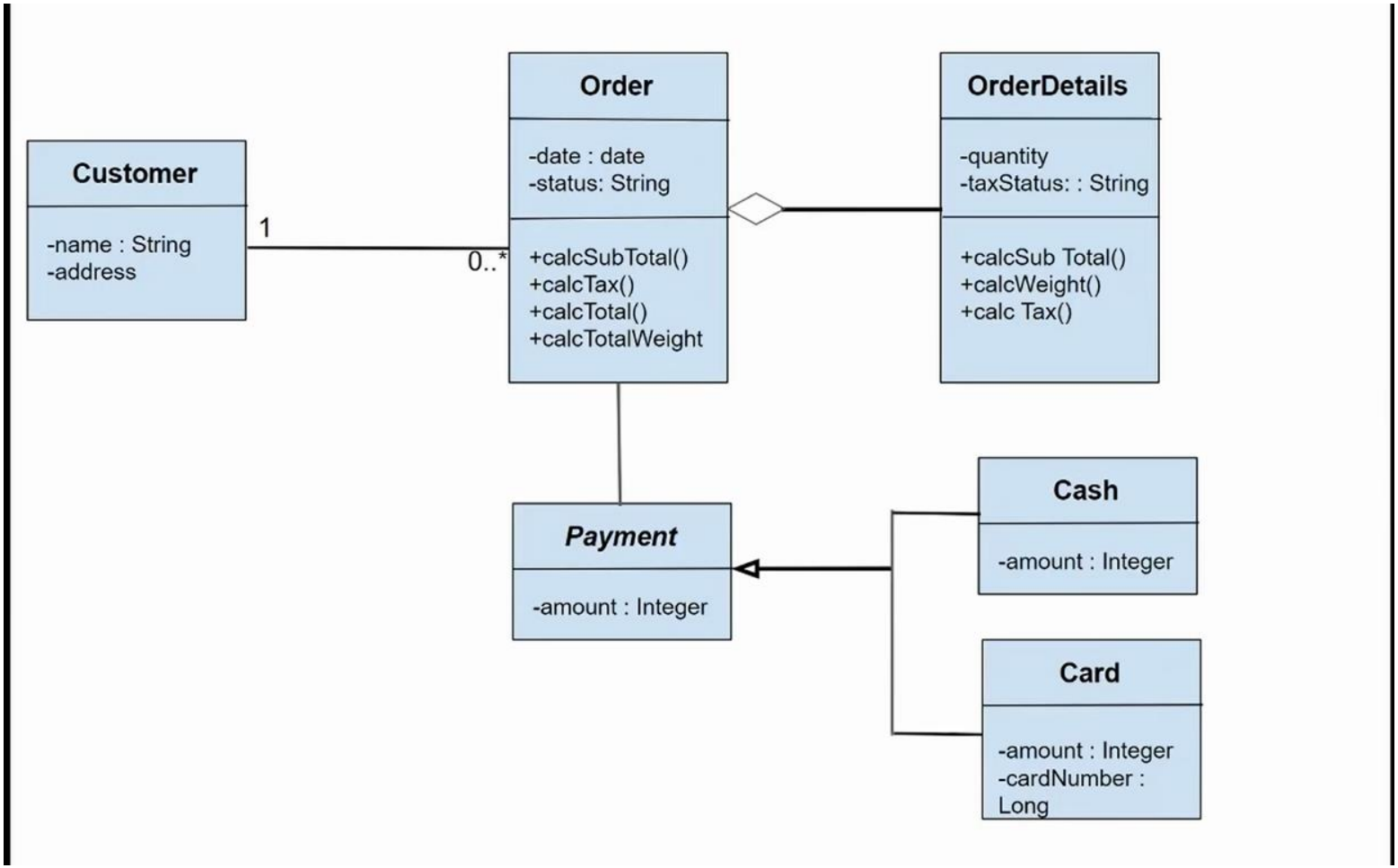
Aggregation



Composition

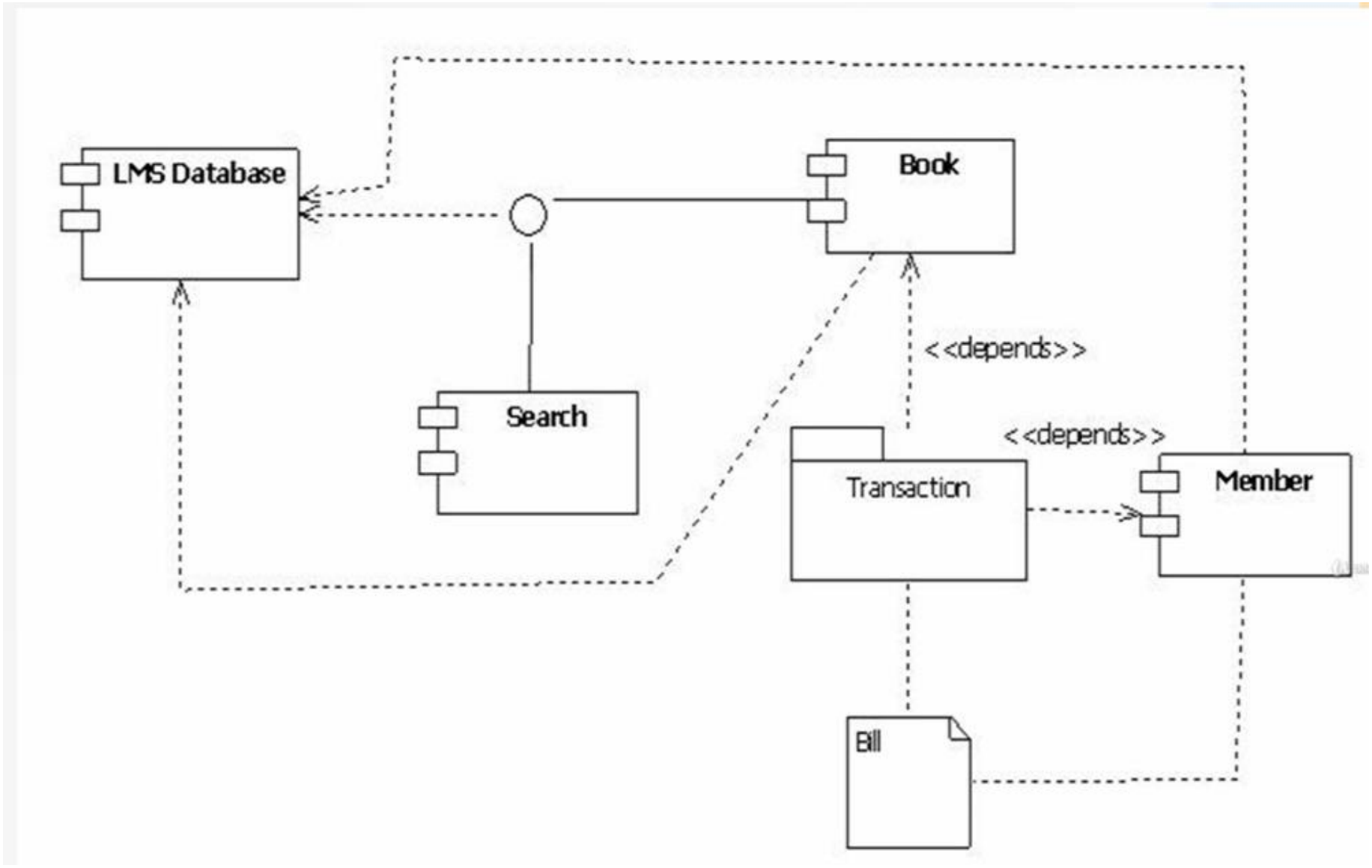



catlog
+Authorname: string
+noofcopies: integer
+UpdateInfo()
+searchinfo()
Librian
+name: string
+address: string
+mobileno: integer
+Updateinfo()
+Issuebooks()
+memberInfo()
+searchbk()
+returnbk()
1
+*
+1,2
Books
+Nameofbook: string
+AuthorName: string
+Noofbooks: integer
+removefrmcatlogue()
+addtocatlogue()
+1..*
1..3
+1..*
+*
+1
reference book
+searchrefbk()
general book
Member
+Mname: string
+Maddress: string
+Mno: integer
+mrequestforbk()
+Mreturnbk()
Alert
+Issuedate: Date
+Bookname: string
+returndate: date
+Fine: integer
+finecal()
+viewAlert()
+sendtolibrian()
+deletealertbyno()
Faculty member
+Fname
+faculty coll
+checkoutbk()
student
+sName: string
+Student coll: string
+checkoutbk()
+returnbk()


Customer
-name : String
-address
1
0..*
Order
-date : date
-status: String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight
OrderDetails
-quantity
-taxStatus: : String
+calcSub Total()
+calcWeight()
+calc Tax()
Payment
-amount : Integer
Cash
-amount : Integer
Card
-amount : Integer
-cardNumber :
Long

Component Diagram

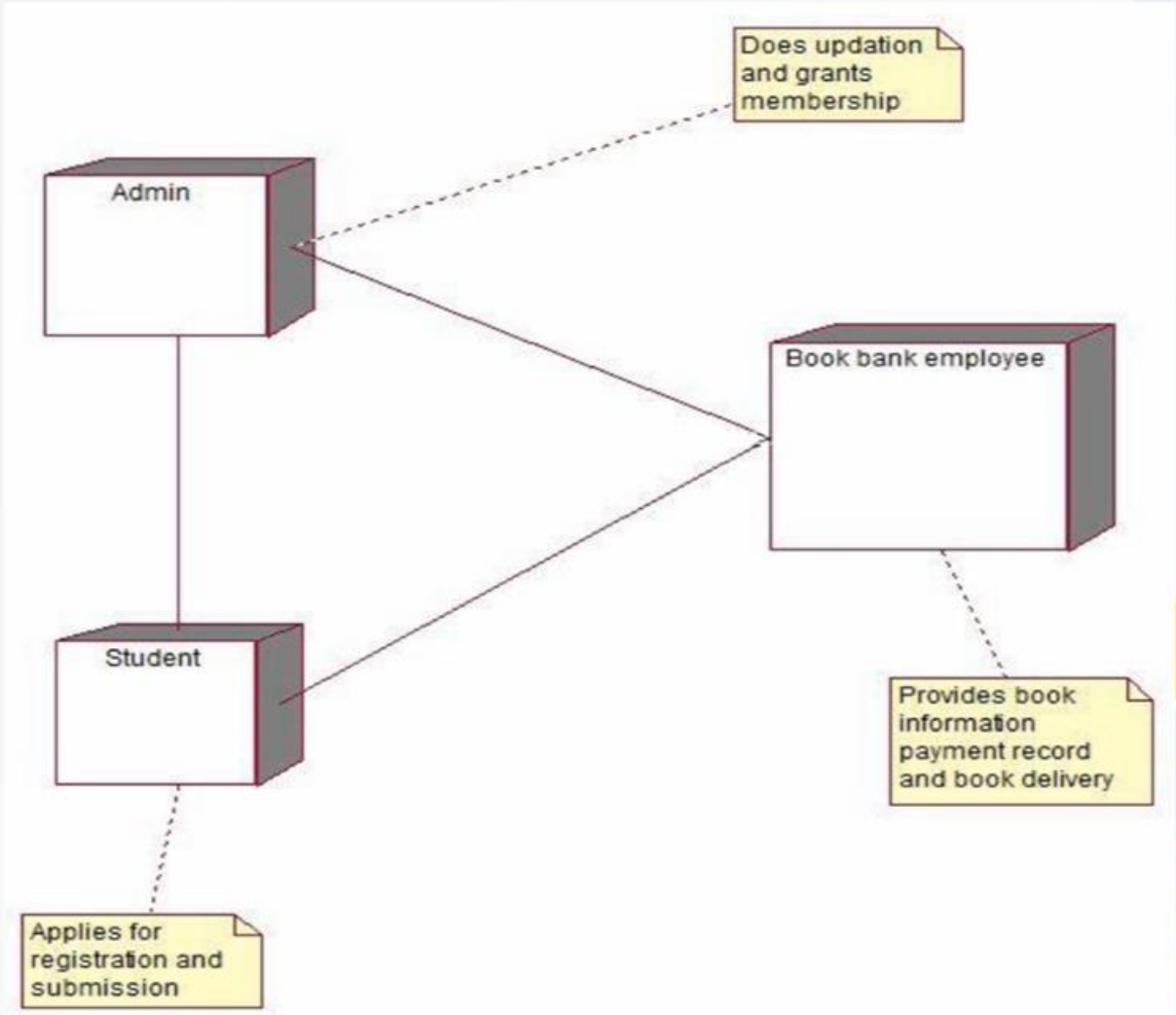
- Component diagrams address the static design implementation view of a system.
- They are important for building large systems from smaller parts.


LMS Database
Book
Search
<<depends>>
<<depends>>
Transaction
Member
Bill

Deployment Diagram

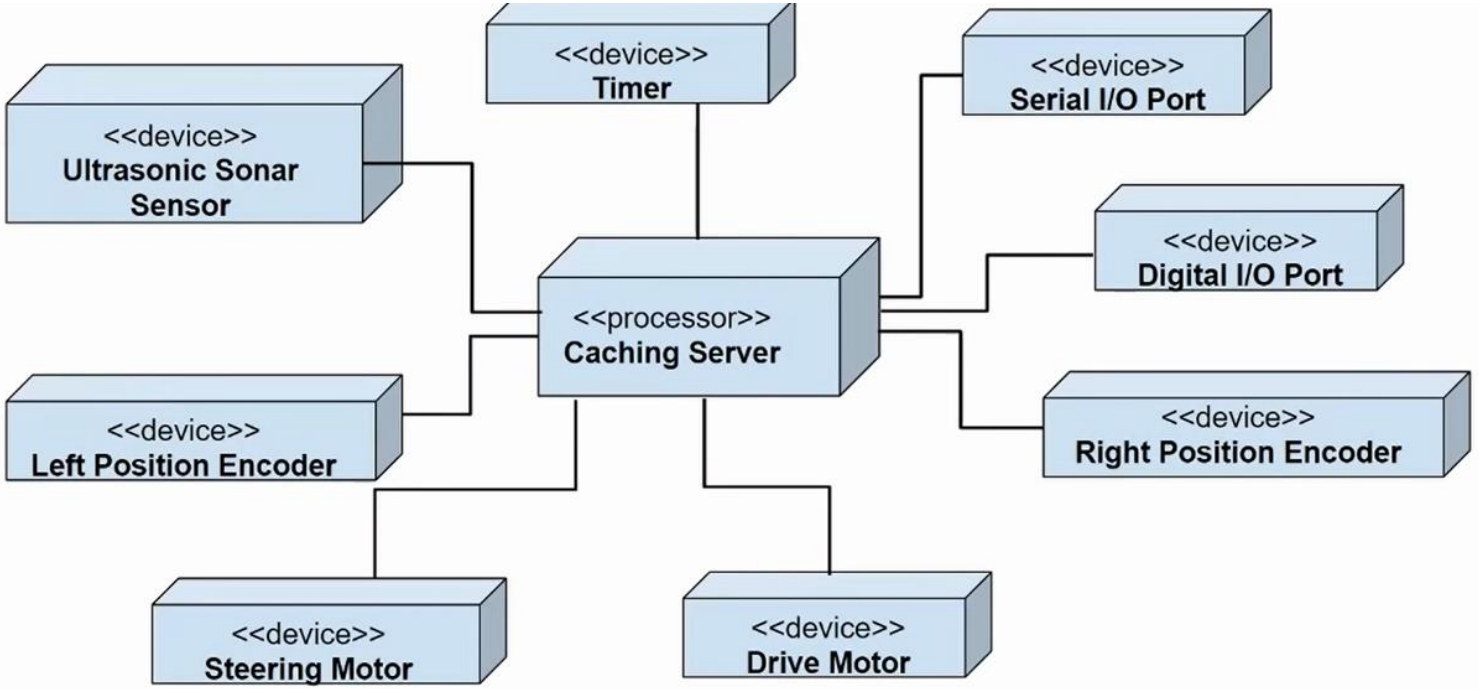
- A deployment diagram shows the configuration of run-time processing nodes and the components that live on them.
- Deployment diagrams address the static deployment view of an architecture.
- A node typically hosts one or more artifacts.

Deployment Diagram

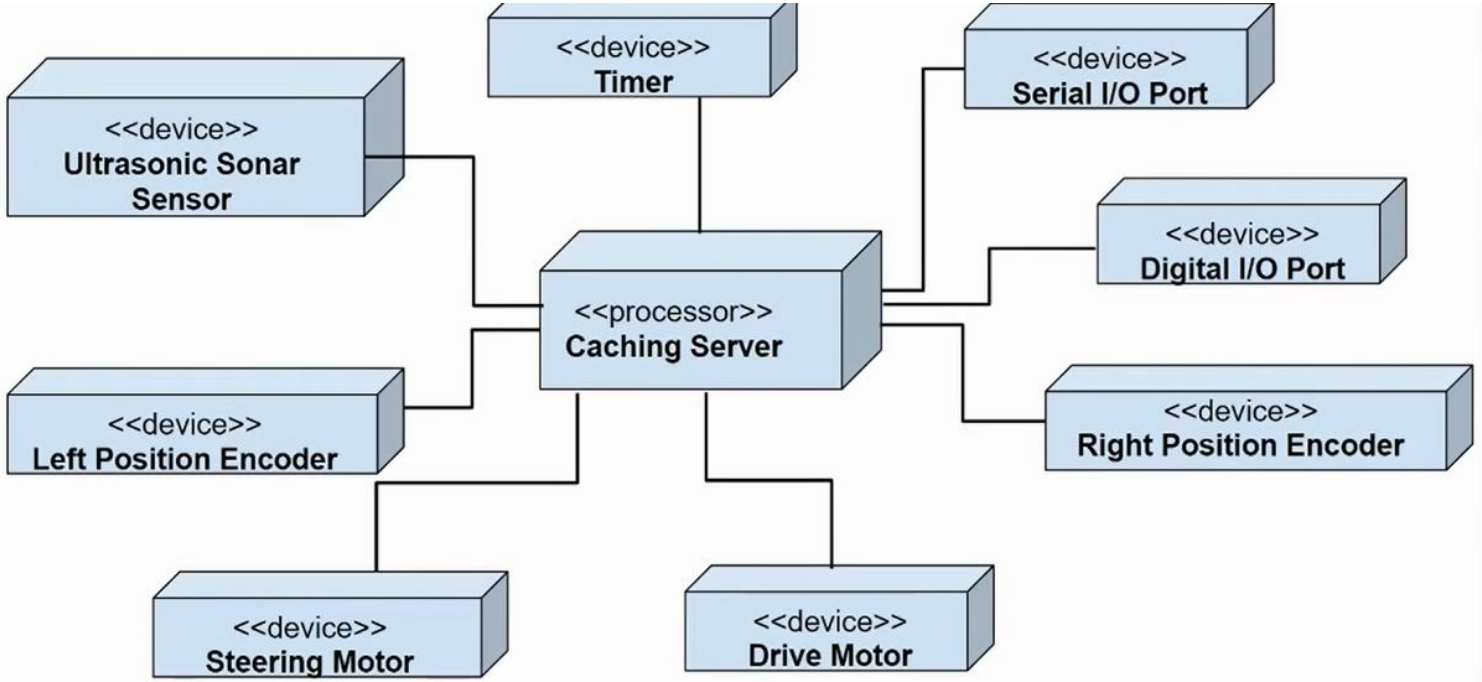


Difference between deployment & Componenet

- Component diagrams are used to describe the components and deployment diagrams shows how they are deployed in hardware.
- UML is mainly designed to focus on the software artifacts of a system.
- However, these two diagrams are special diagrams used to focus on software and hardware components.
- Most of the UML diagrams are used to handle logical components but deployment diagrams are made to focus on the hardware topology of a system.

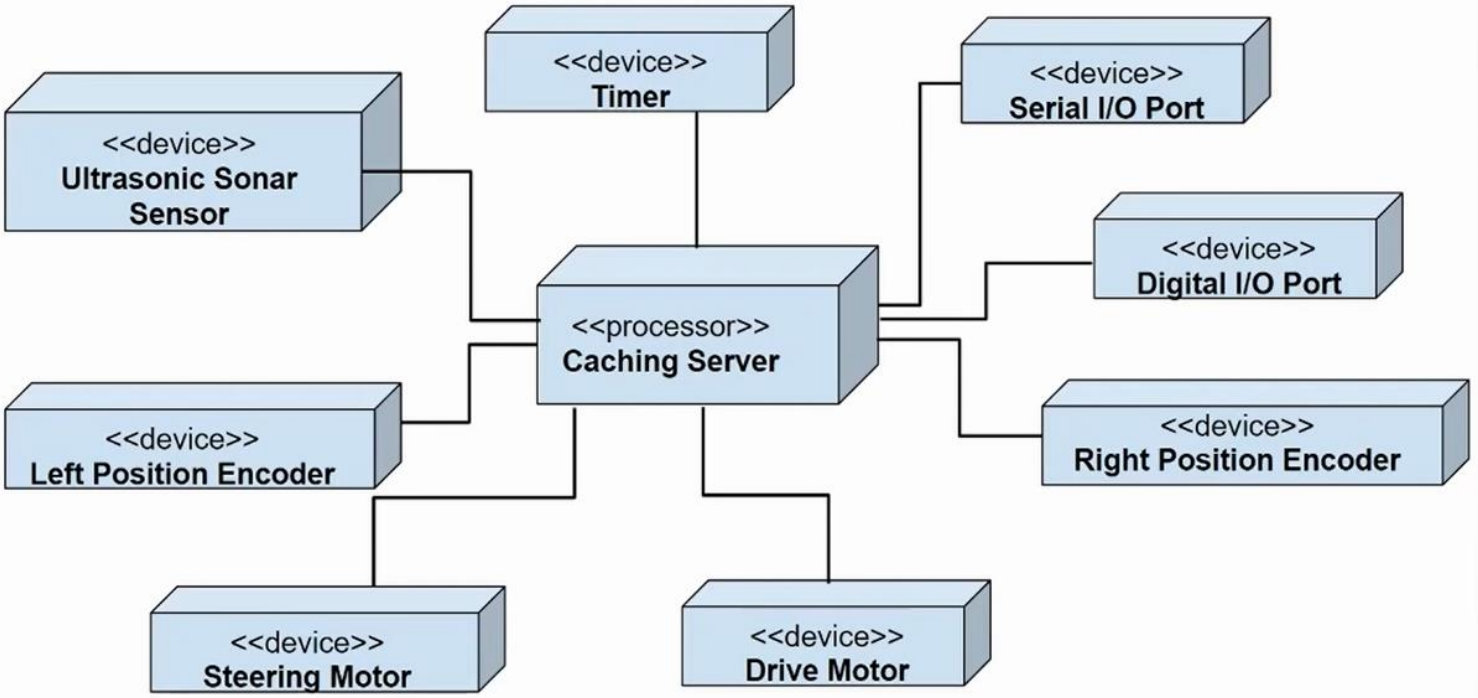


1 Identify devices and nodes that are unique to your system.

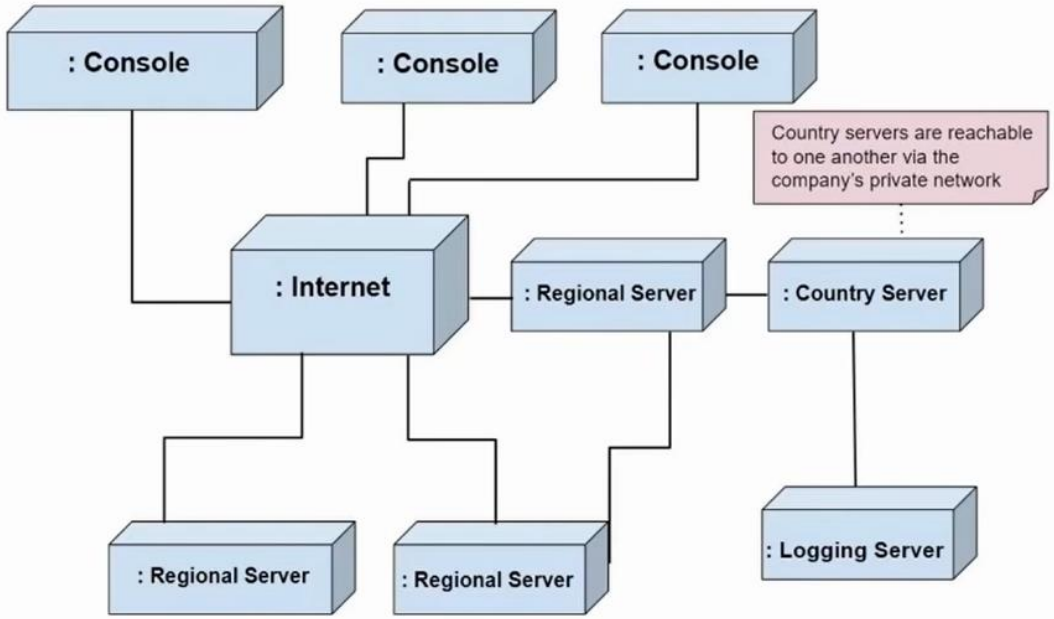


2 Provide visual cues, especially for unusual devices, using UML extensibility mechanisms to identify system stereotypes with matching icons. At the very least, you will want to distinguish between processors that contain software components and devices that at this level of abstraction do not directly contain software.

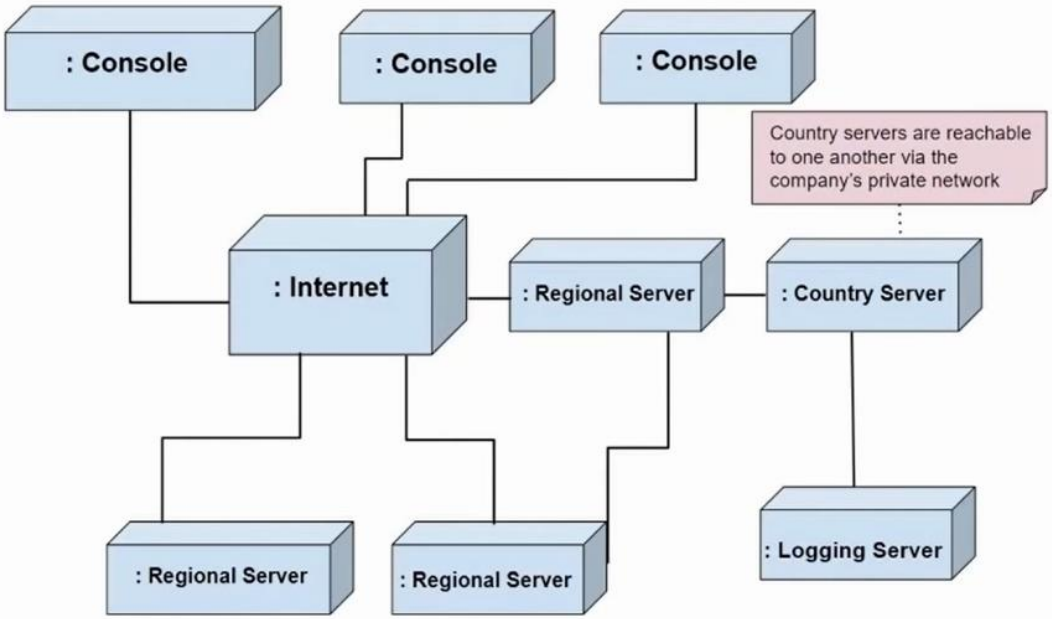
Deployment



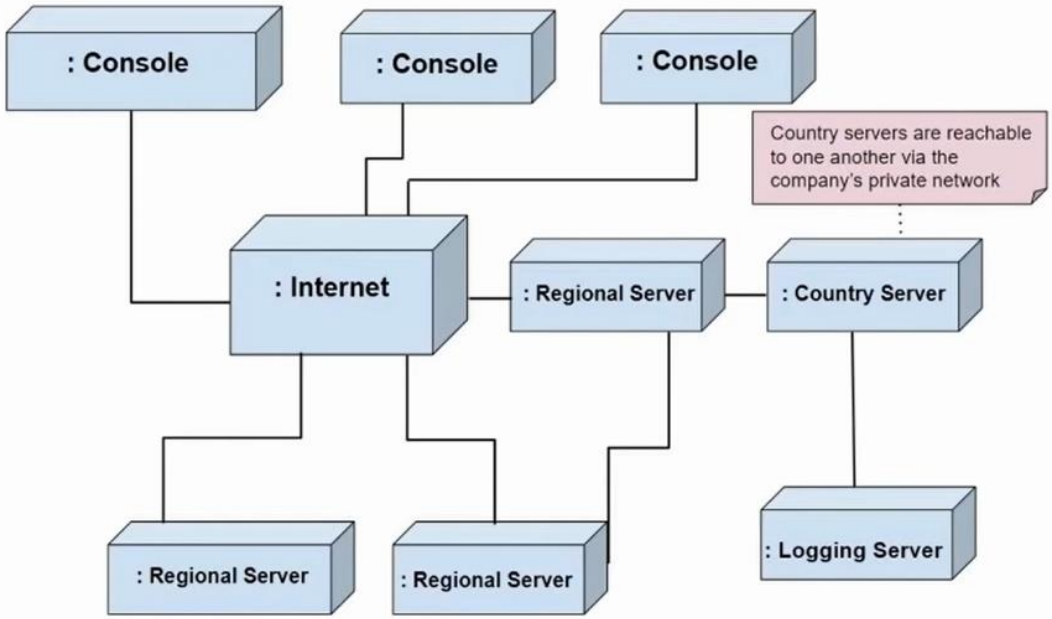
3 Model the relationships between these processors and devices in a deployment diagram. Similarly, specify the relationships between the components in the implementation view of your system and the nodes in the deployment view of your system.



3 Where possible, use tools that discover the topology of your system by walking your system's network.



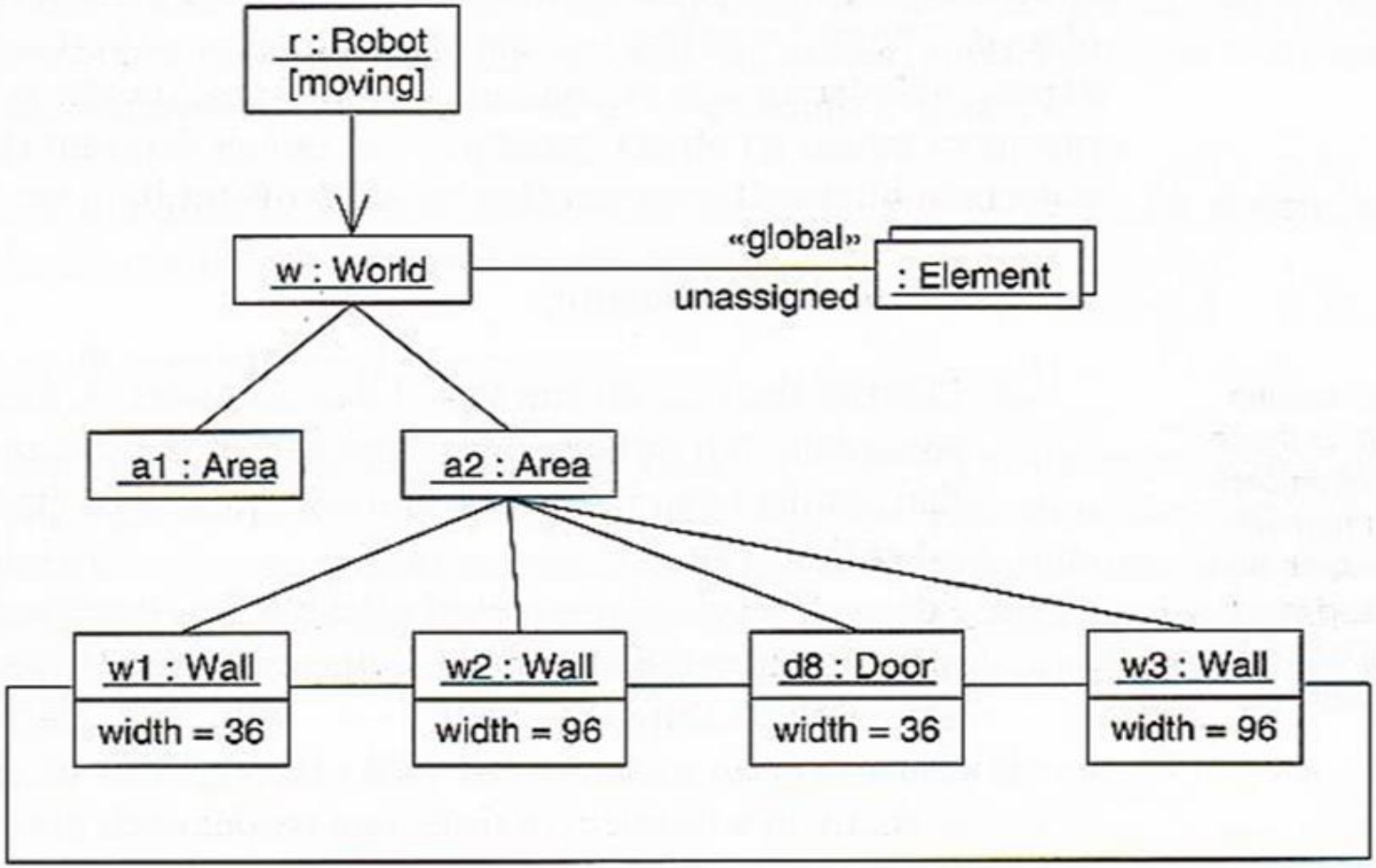
4 If you need to focus on the dynamics of your system, use the use case diagram, to indicate the types of behavior you are interested in, and expand these use cases with interaction diagrams.



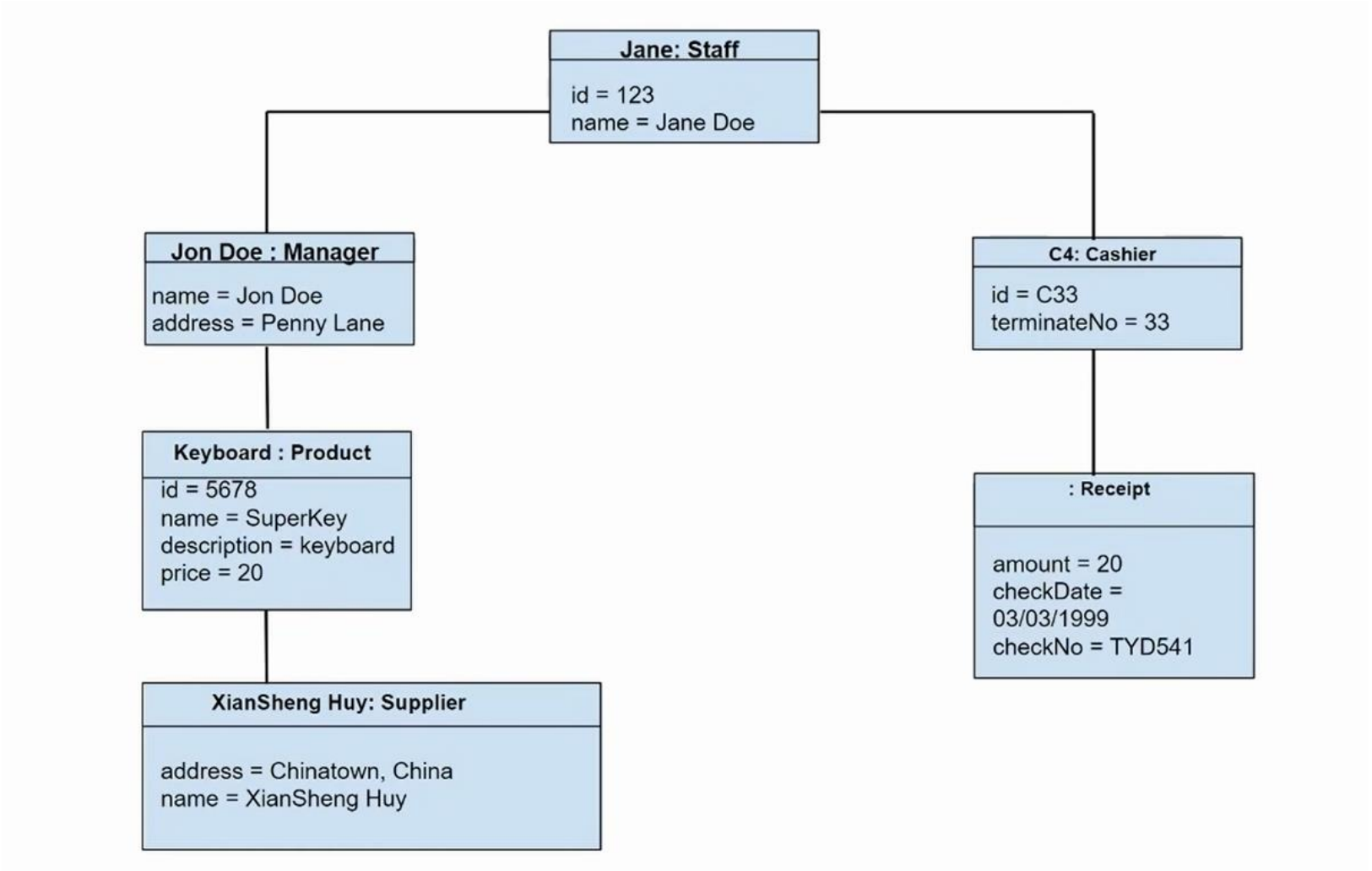
5

When modeling a fully distributed system, it is common to transform or represent the network itself as a node. That is, the Internet, LAN or WAN might be depicted as nodes in the form of a single box, rather than as a combination of elements.

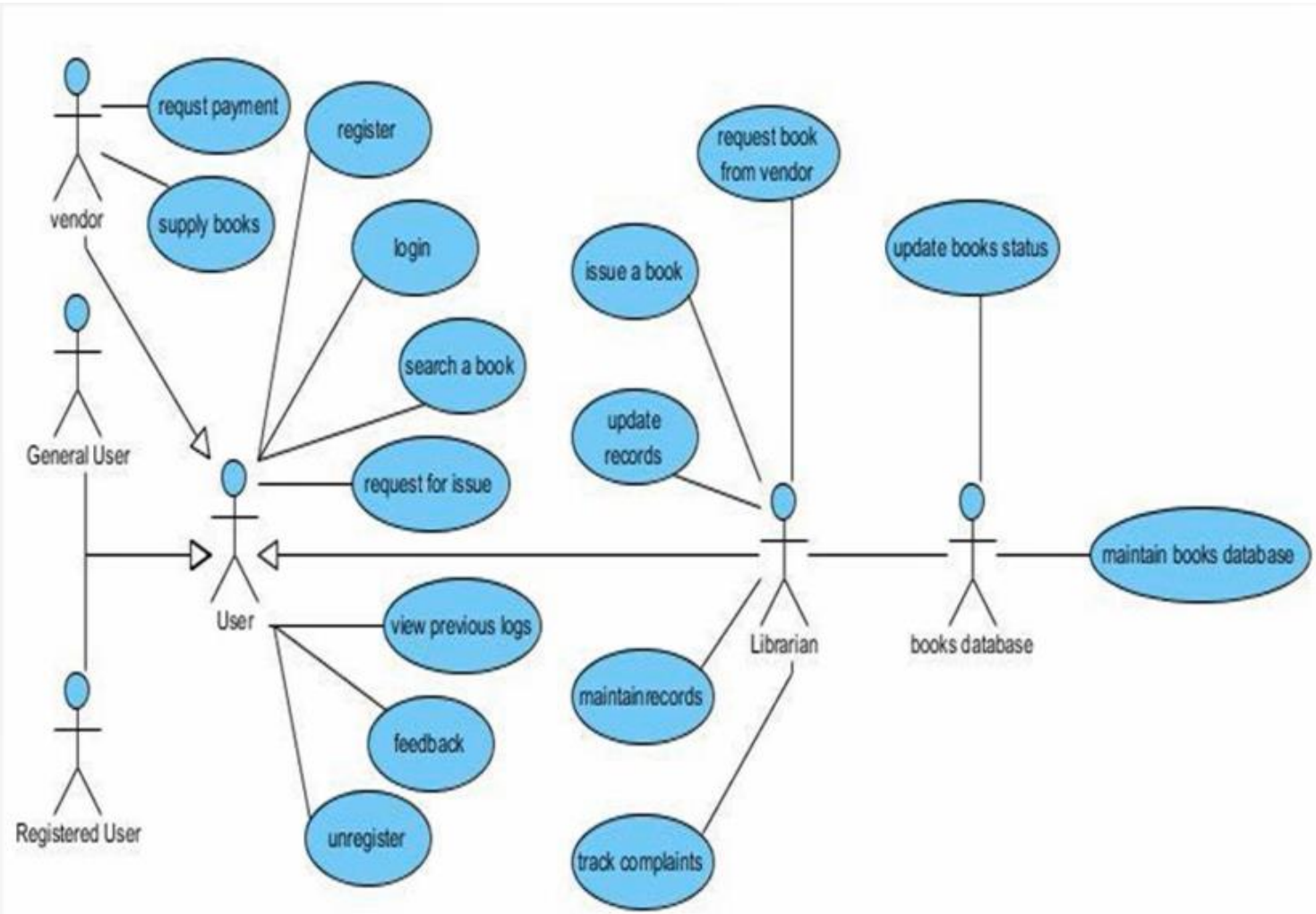
- An object diagram shows a set of objects and their relationships.
- Object diagrams represent static snapshots of instances of the things found in class diagrams.
- These diagrams address the static design view or static process view of a system as do class diagrams, but from the perspective of real or prototypical cases.


r : Robot
[moving]
w : World
«global»
unassigned
: Element
a1 : Area
a2 : Area
w1 : Wall
width = 36
w2 : Wall
width = 96
d8 : Door
width = 36
w3 : Wall
width = 96

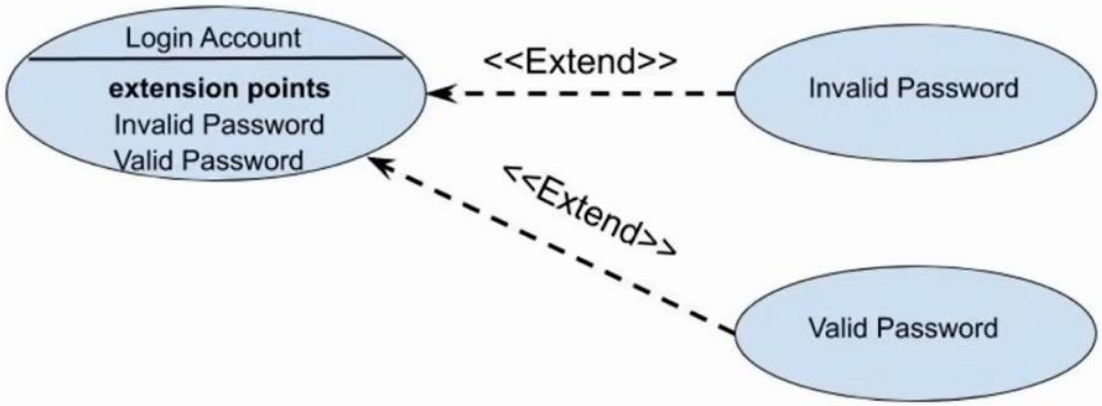
Object Diagram



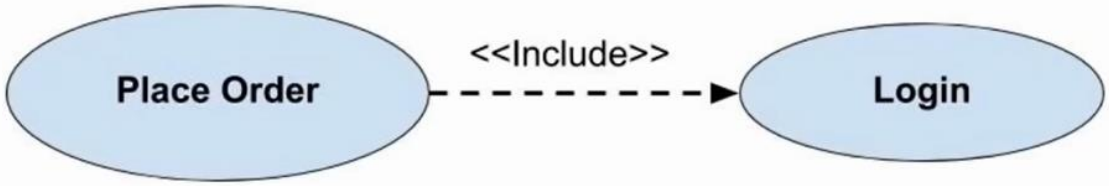
- A use case diagram shows a set of use cases and actors (a special kind of class) and their relationships.
- Use case diagrams address the static use case view of a system.
- These diagrams are especially important in organizing and modeling the behaviors of a system.


requst payment
register
request book from vendor
vendor
supply books
login
issue a book
update books status
search a book
General User
update records
request for issue
User
Librarian
books database
maintain books database
view previous logs
maintain records
feedback
Registered User
unregister
track complaints

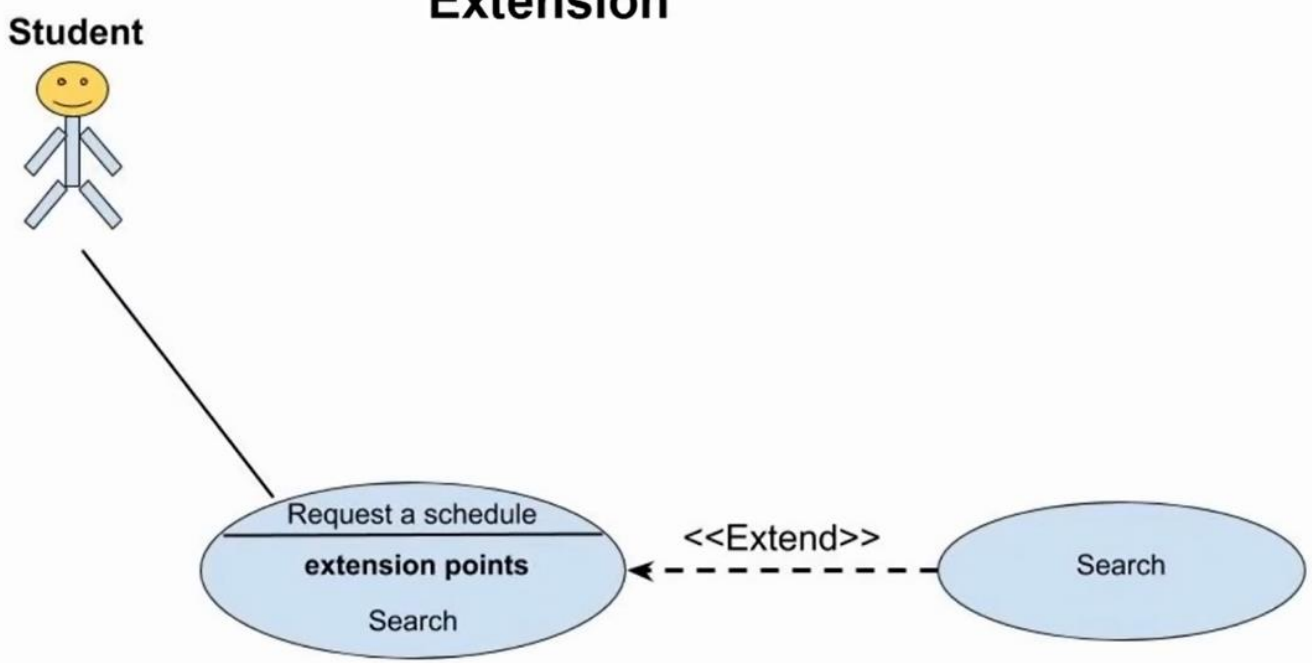
Extension


Login Account
extension points
Invalid Password
Valid Password
<<Extend>>
Invalid Password
<<Extend>>
Valid Password

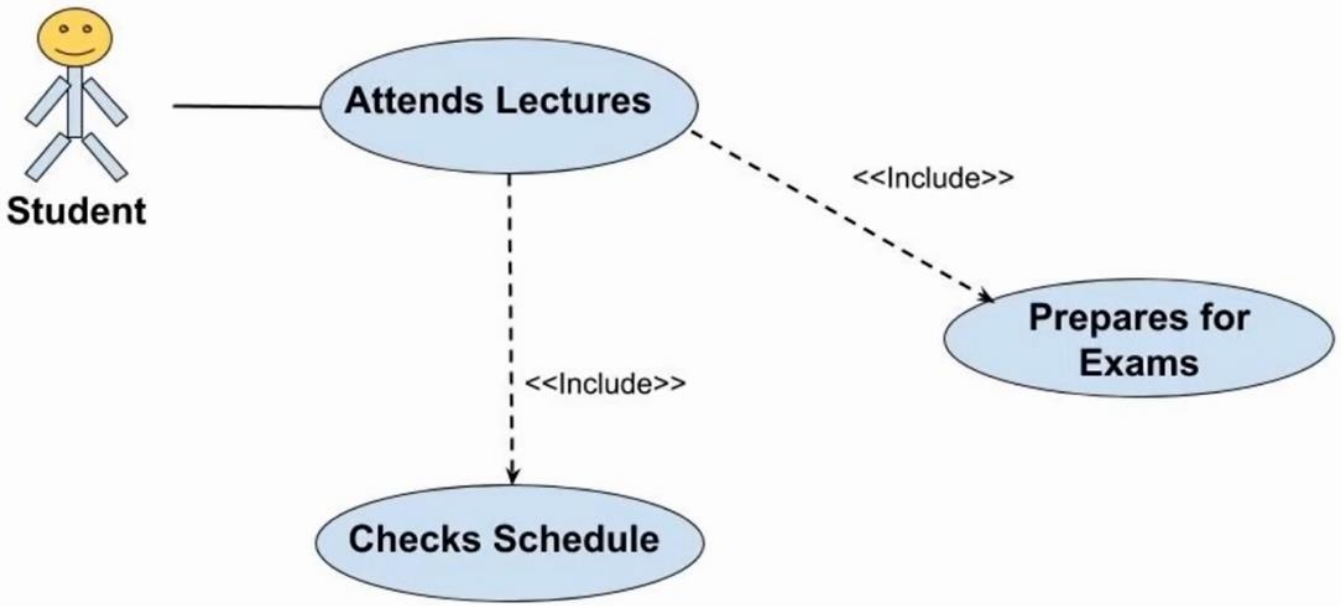
Inclusion



Extension


Student
Request a schedule
extension points
Search
<<Extend>>
Search

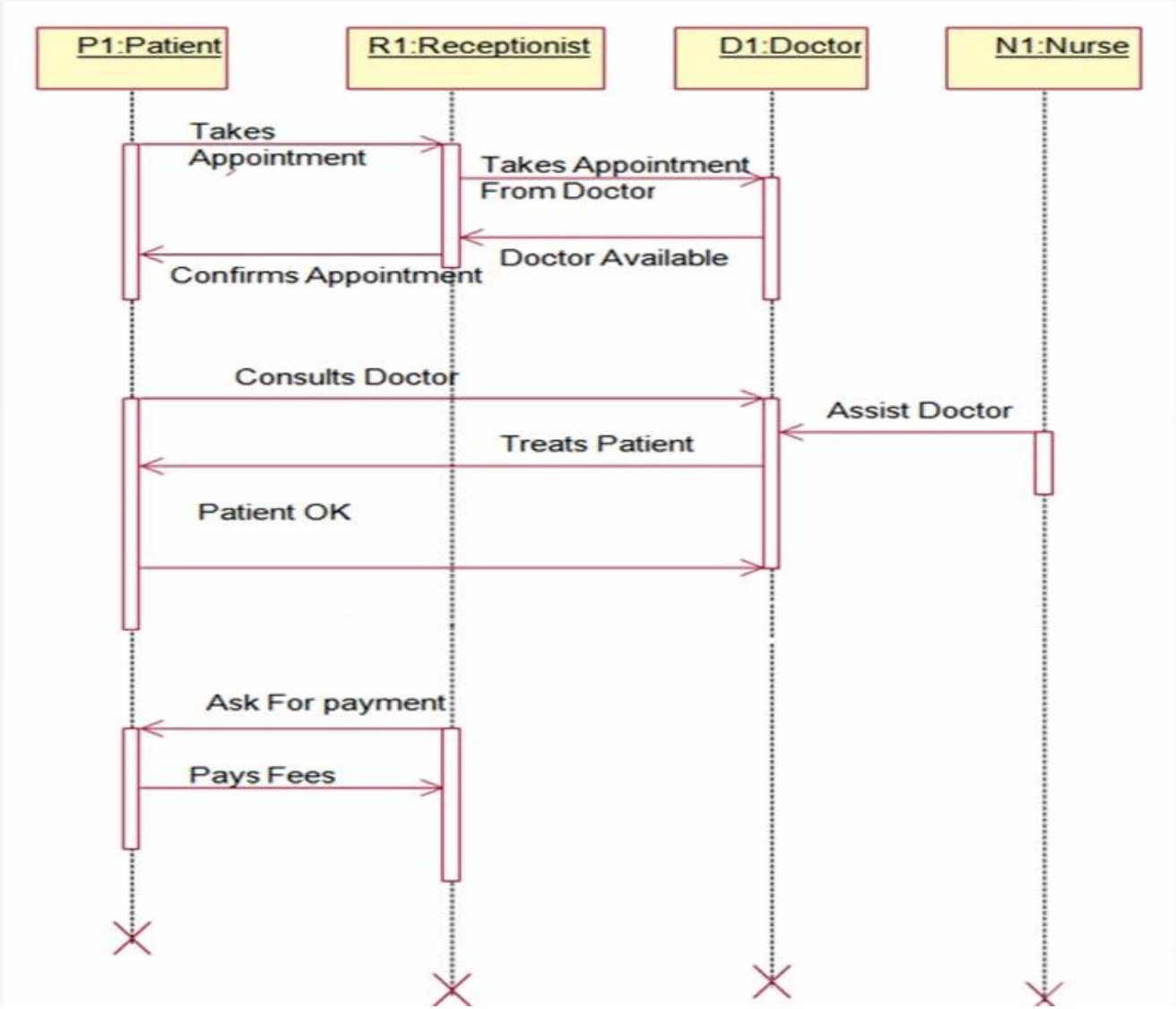
Inclusion


Student
Attends Lectures
<<Include>>
Prepares for Exams
<<Include>>
Checks Schedule

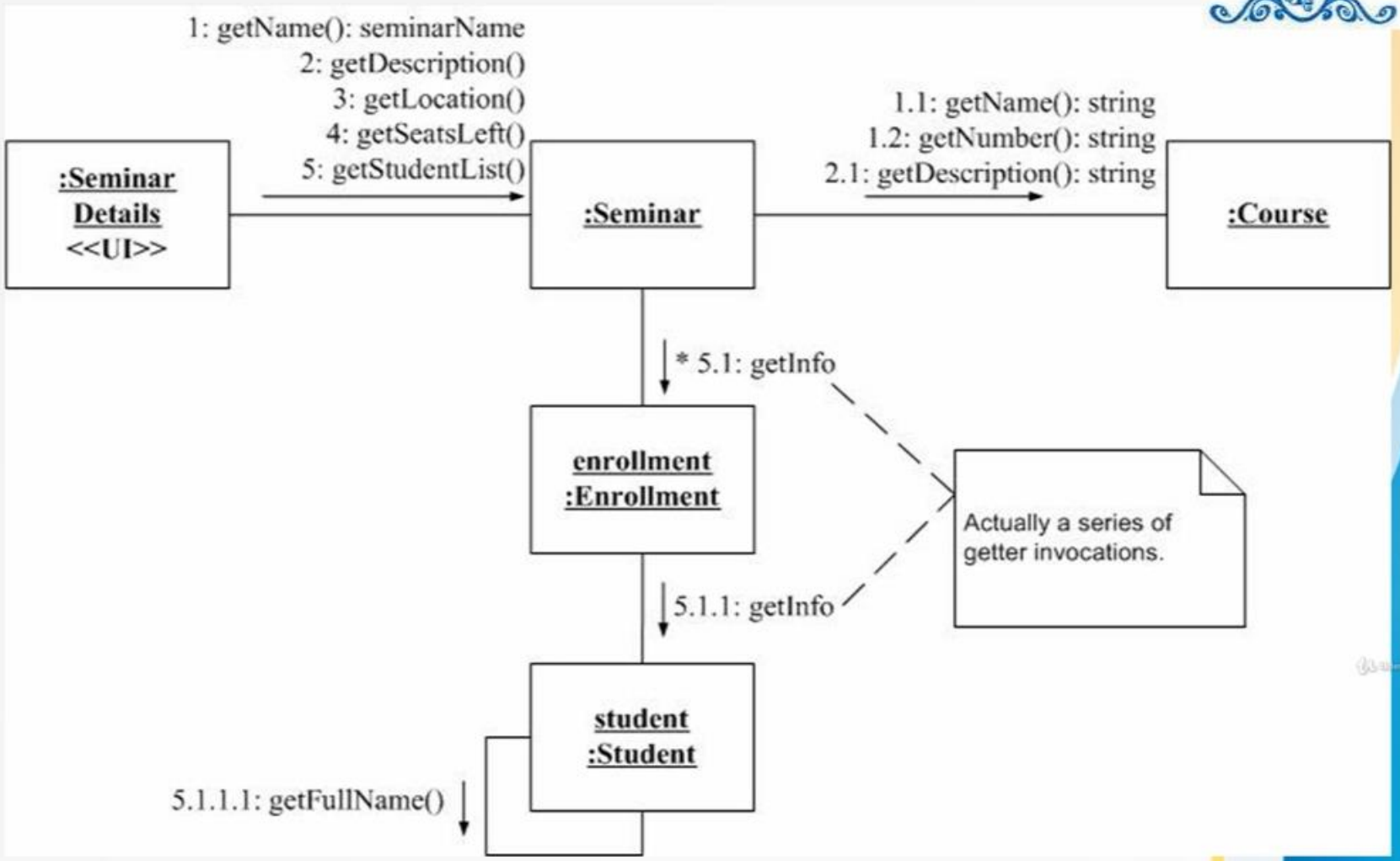
Sequence and Collaboration

- Both are interaction diagrams.
- Shows dynamic view of the system.
- Sequence diagram emphasizes on time ordering of messages.
- Collaboration diagram structural organization of objects.
- Both diagrams are isomorphic.

Sequence Diagram: Example



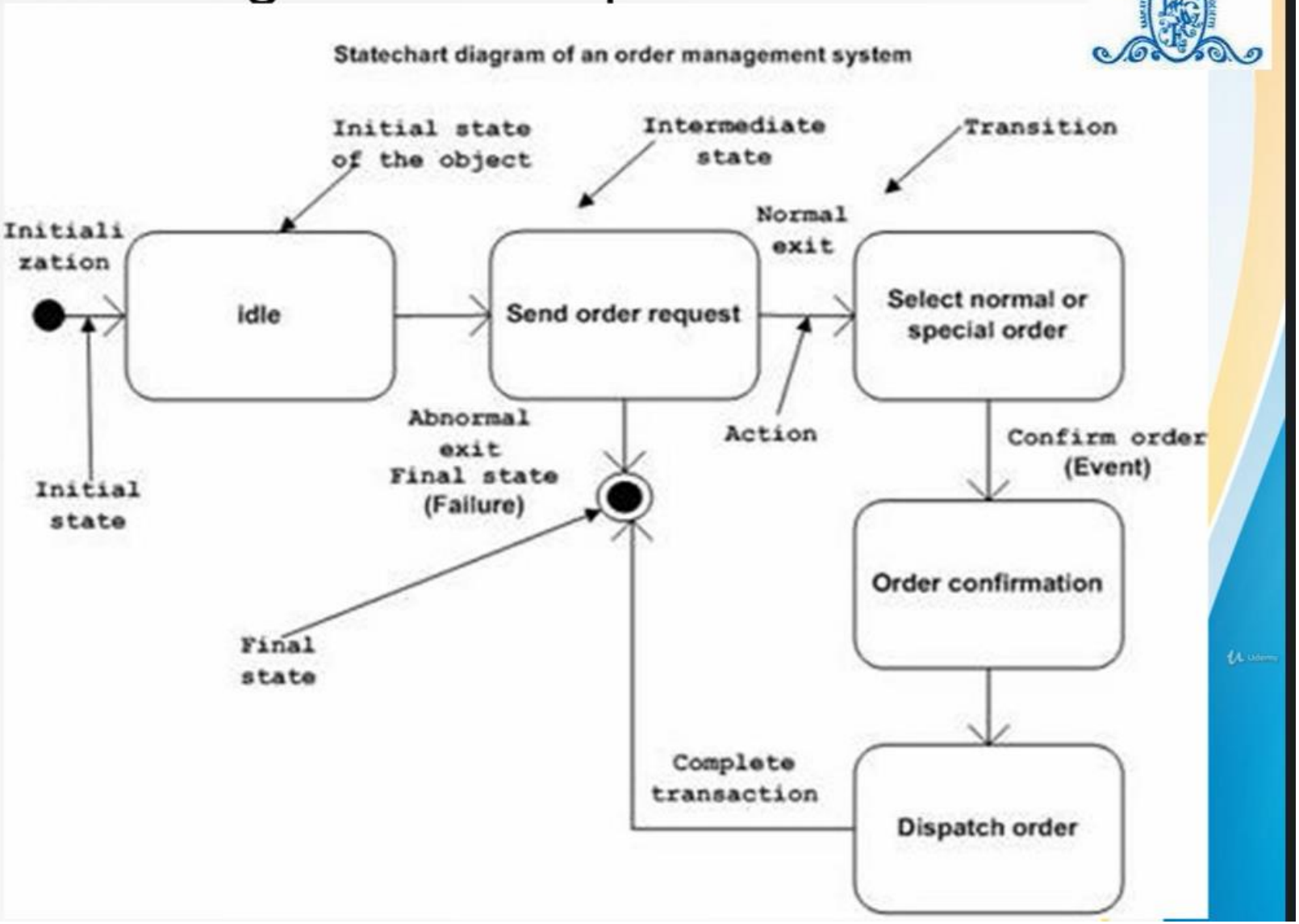
Collaboration Diagram: Example



State Diagram

- A state diagram shows a state machine, consisting of states, transitions, events, and activities.
- A state diagrams shows the dynamic view of an object.
- Useful in modeling reactive systems

State diagram: Example



Activity Diagram

- An activity diagram shows the structure of a process or other computation as the flow of control and data from step to step within the computation.
- Activity diagrams address the dynamic view of a system.
- They are especially important in modeling the function of a system and emphasize the flow of control among objects.

Activity Diagram: Example