Chapter 7

Object Oriented Structural Modelling

Content

Structural Diagram

- **Class Diagram**
- **Object Diagram**
- **Deployment Diagram**
- **Component Diagram**
- **Package Diagram (Reading Assignment)**

Static vs Dynamic View

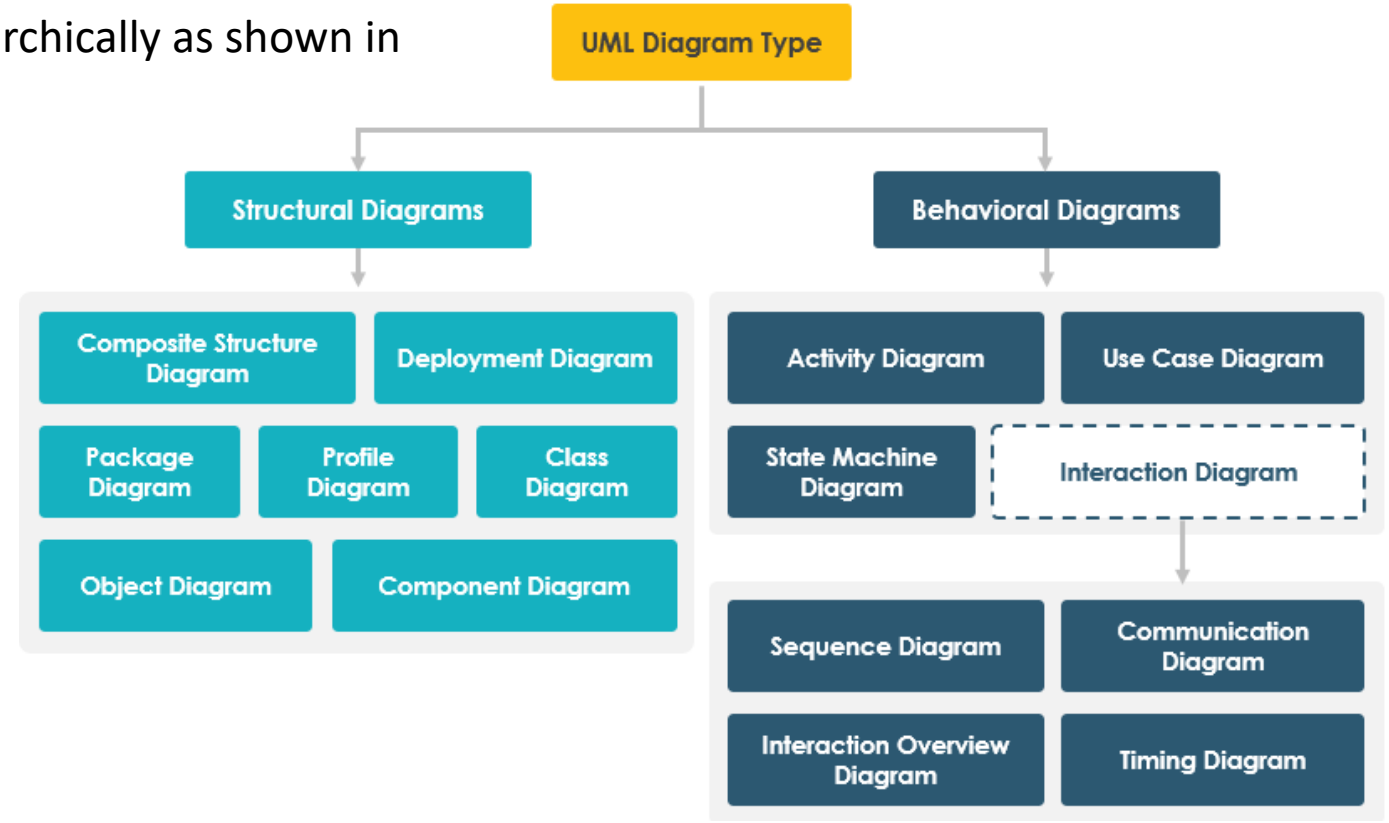
- Static modeling is used to specify the structure of the objects, classes or components that exist in the problem domain.
  - These are expressed using class, object or component.
- UML diagrams represent these two aspects of a system:
- **Structural (or Static) view**:
  - emphasizes the static structure of the system using objects, attributes, operations and relationships.
  - It includes class diagrams.
- **Behavioral (or Dynamic) view**:
  - emphasizes the dynamic behavior of the system by showing collaborations among objects and changes to the internal states of objects.
  - This view includes sequence diagrams, activity diagrams, and state machine diagrams.
- In UML 2.x there are a number of UML diagrams, which are divided into these two categories:

Structural Diagrams

- Structure diagrams depict the static structure of the elements in your system. i.e., how one object relates to another.
- It shows the things in the system – classes, objects, packages or modules, physical nodes, components, and interfaces.
- For example, just as the static aspects of a house encompass the existence and placement of such things as walls, doors, windows, pipes, wires, and vents.

UML Diagram Map

- These diagrams can be categorized hierarchically as shown in the following UML diagram map:



What is a Class

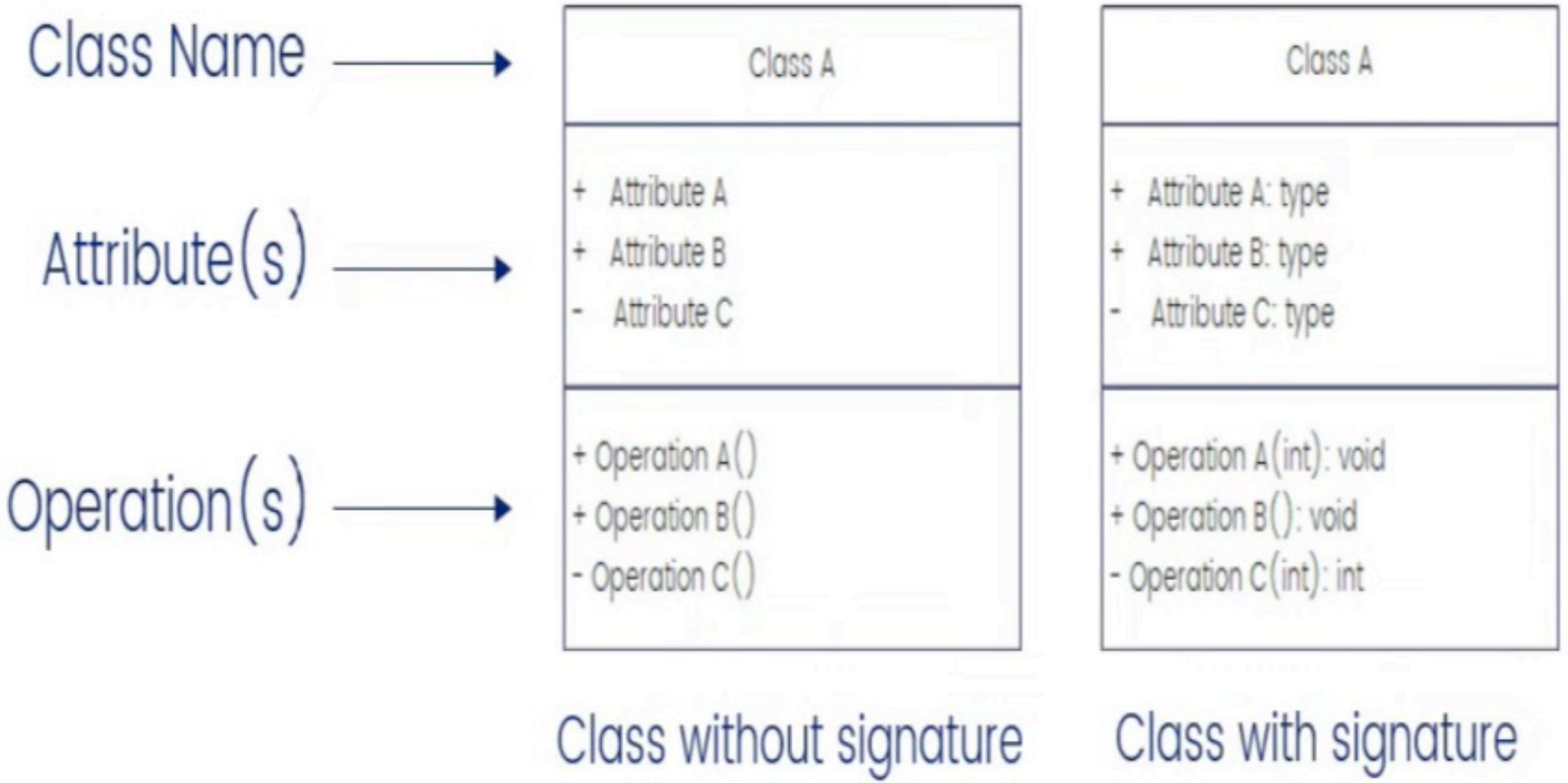
- A description of a group of objects all with similar roles in the system, which consists of:
- **Structural features (attributes)** define what objects of the class "know"
  - Represent the state of an object of the class
  - Are descriptions of the structural or static features of a class
- **Behavioral features** (**operations**) define what objects of the class "can do"
  - Define the way in which objects may interact
  - Operations are descriptions of behavioral or dynamic features of a class

Class Notation

A class notation consists of three parts:

- **Class Name**
  - The name of the class appears in the first partition.
- **Class Attributes**
  - Attributes are shown in the second partition.
  - The attribute type is shown after the colon.
  - Attributes map onto member variables (data members) in code.
- **Class Operations (Methods)**
  - Operations are shown in the third partition. They are services the class provides.
  - The return type of a method is shown after the colon at the end of the method signature.
  - The return type of method parameters is shown after the colon following the parameter name.
  - Operations map onto class methods in code

…Class Notation

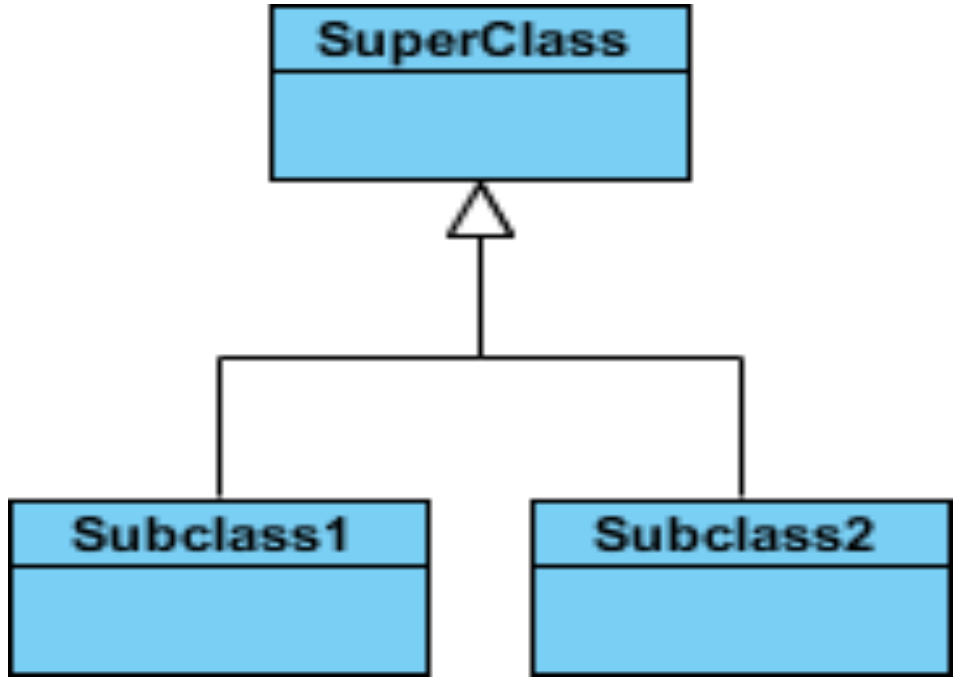


Class Relationships

- A class may be involved in one or more relationships with other classes. A relationship can be one of the following types: (Refer to the figure on the right for the graphical representation of relationships).

1. Inheritance (or Generalization):

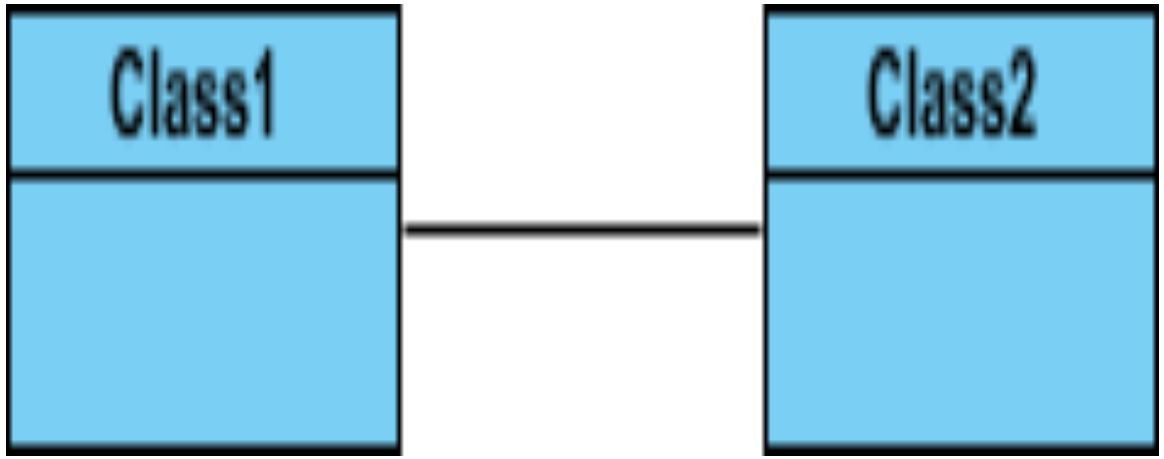
- Represents an **"is-a"** relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class



…Class Relationships

Simple Association:

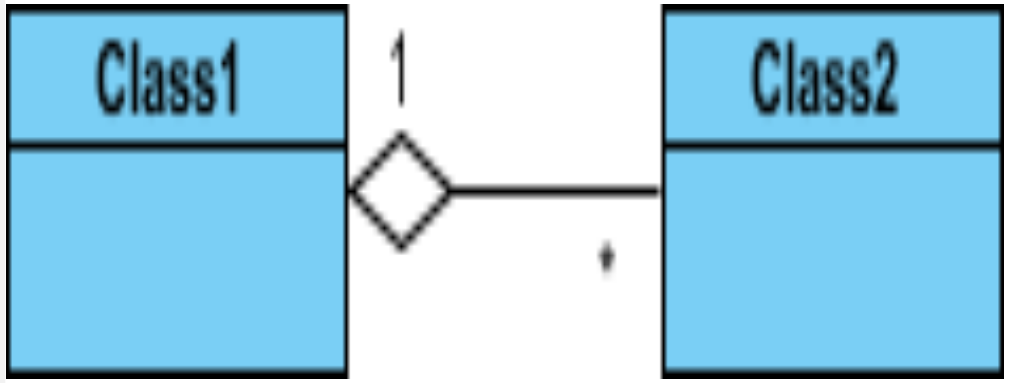
- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A **solid line** connecting two classes



…Class Relationships

Aggregation:

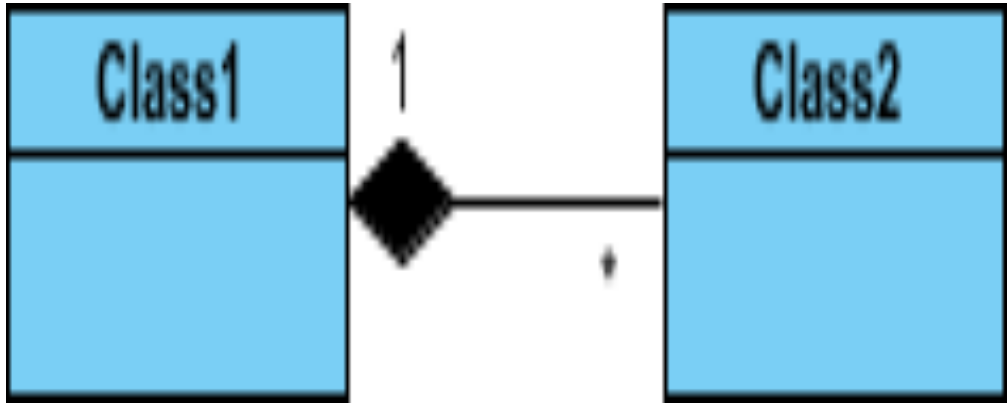
- A special type of association. It represents a "**part of**" relationship.
- **Class2 is part of Class1**.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- The aggregate is the parent class, the components are the children classes
- A solid line with an **unfilled diamond** at the association end connected to the class of composite



…Class Relationships

Composition:

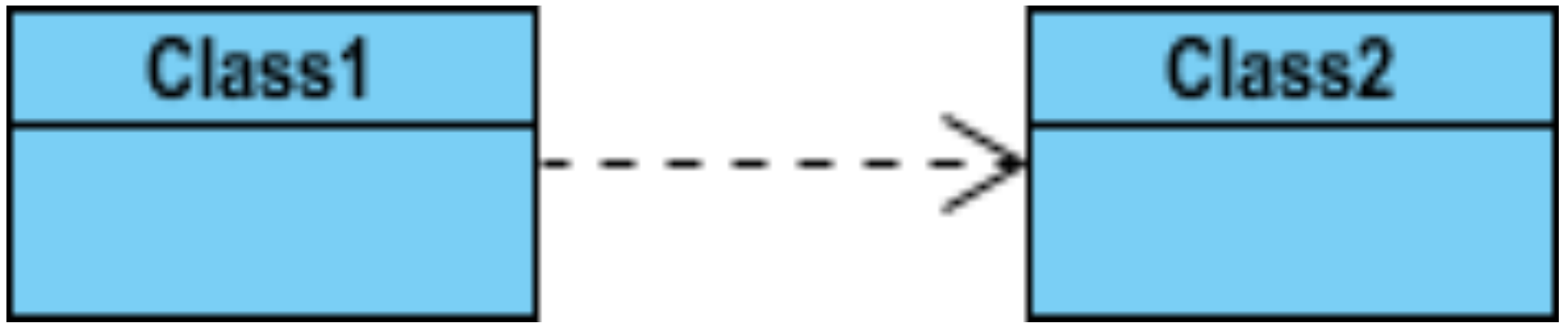
- A special type of aggregation where parts are destroyed when the whole is destroyed.
- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a **filled diamond** at the association connected to the class of composite



…Class Relationships

Dependency:

- Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).
- Class1 depends on Class2
- A dashed line with an open arrow



Visibility of Class attributes and Operations

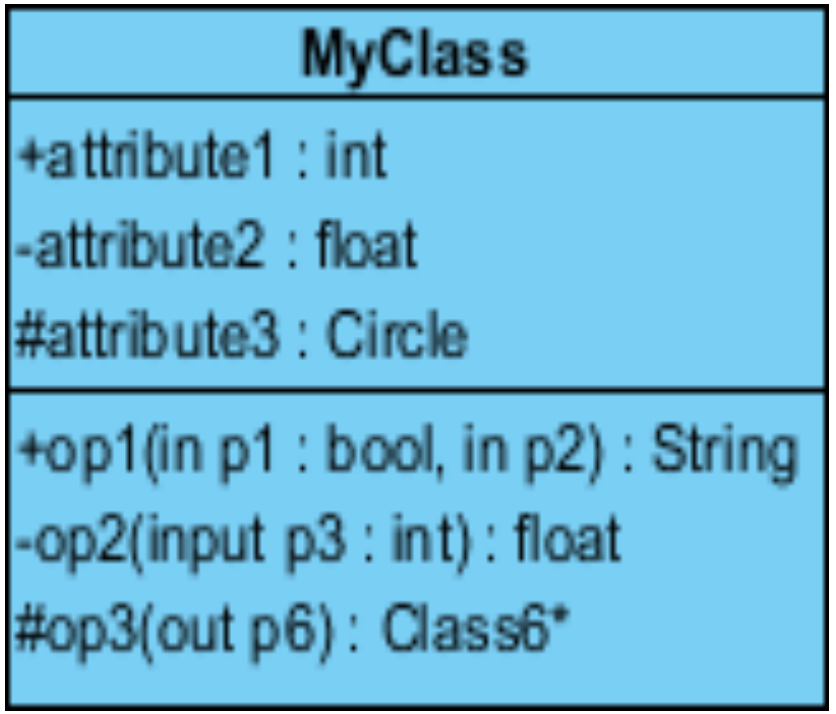
- In object-oriented design, there is a notation of visibility for attributes and operations.
- UML identifies four types of visibility: **public**, **protected**, **private**, and **package**.

+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

~ denotes package attributes or operations



Multiplicity

- How many objects of each class take part in the relationships and multiplicity can be expressed as:
  - Exactly one -> 1
  - Zero or one -> 0..1
  - Many -> 0..* or *
  - One or more -> 1..*
  - Exact Number -> e.g. 3..4 or 6
  - Or a complex relationship -> e.g. 0..1, 3..4, 6.*

Class Diagram

- A class diagram models the static view of a system.
- It comprises of the classes, interfaces, and collaborations of a system; and the relationships between them.
- It allow us to what classes we need , their functionality and their relationships with other system elements namely- other classes, operations, attributes and objects.
- Used to map out what the system looks like

Types of class stereotypes

- **Entities Class *(model)***

  -Objects representing system data.

  - Typical business entities like "person" and "bank account"

- **Boundaries Class *(view/service collaborator)***

  -The classes that we or other systems interact with

  - Common **boundary classes** include windows, communication protocols, printer interfaces, sensors, and terminals are examples of boundaries that interface with users.

- **Controls Class*(controller)***

  -Objects that mediate between boundaries and entities.

  - These serve as the glue between boundary elements and entity elements, implementing the logic required to manage the various elements and their interactions.

Types of class stereotypes

Four rules apply to their communication:

- Actors can only talk to boundary objects.
- Boundary objects can only talk to controllers and actors.
- Entity objects can only talk to controllers.
- Controllers can talk to boundary objects and entity objects, and to other controllers, but not to actors

Steps how to draw class diagram

Step-1 :- Identify the Class name

- Identify the primary objects of the system

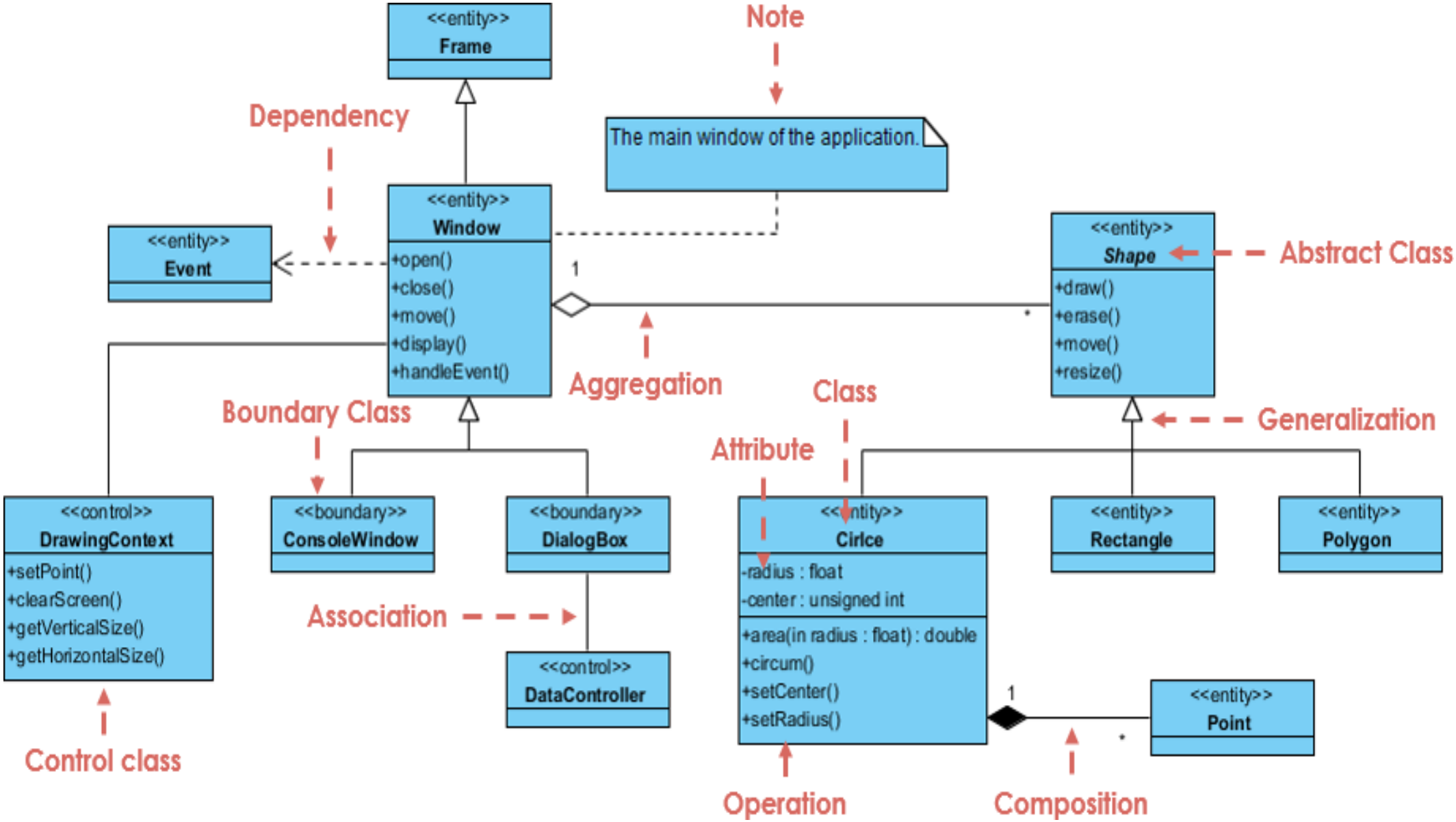
Step-2 :- Distinguishes relationships

- Determine each how each of the classes or objects are related to one another .Look out for commonalities and abstractions among them. This will help us when grouping them when drawing the class diagram

Step-3:- Create Structure

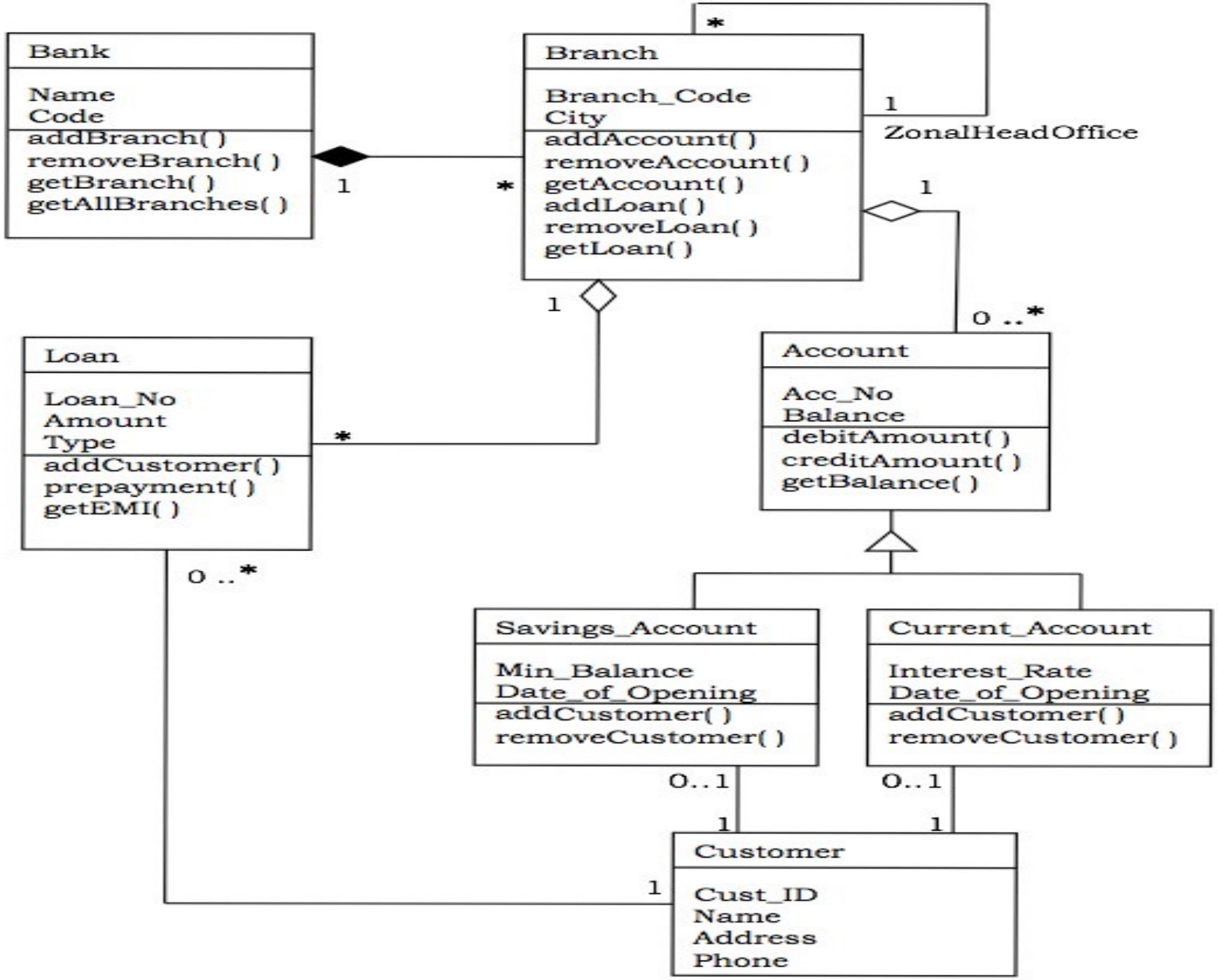
- First, add the class names and link them with appropriate connectors.
- We can add attributes and functions/methods/operations later.

Class Diagram:- Example



Class Diagram of a Banking System

.

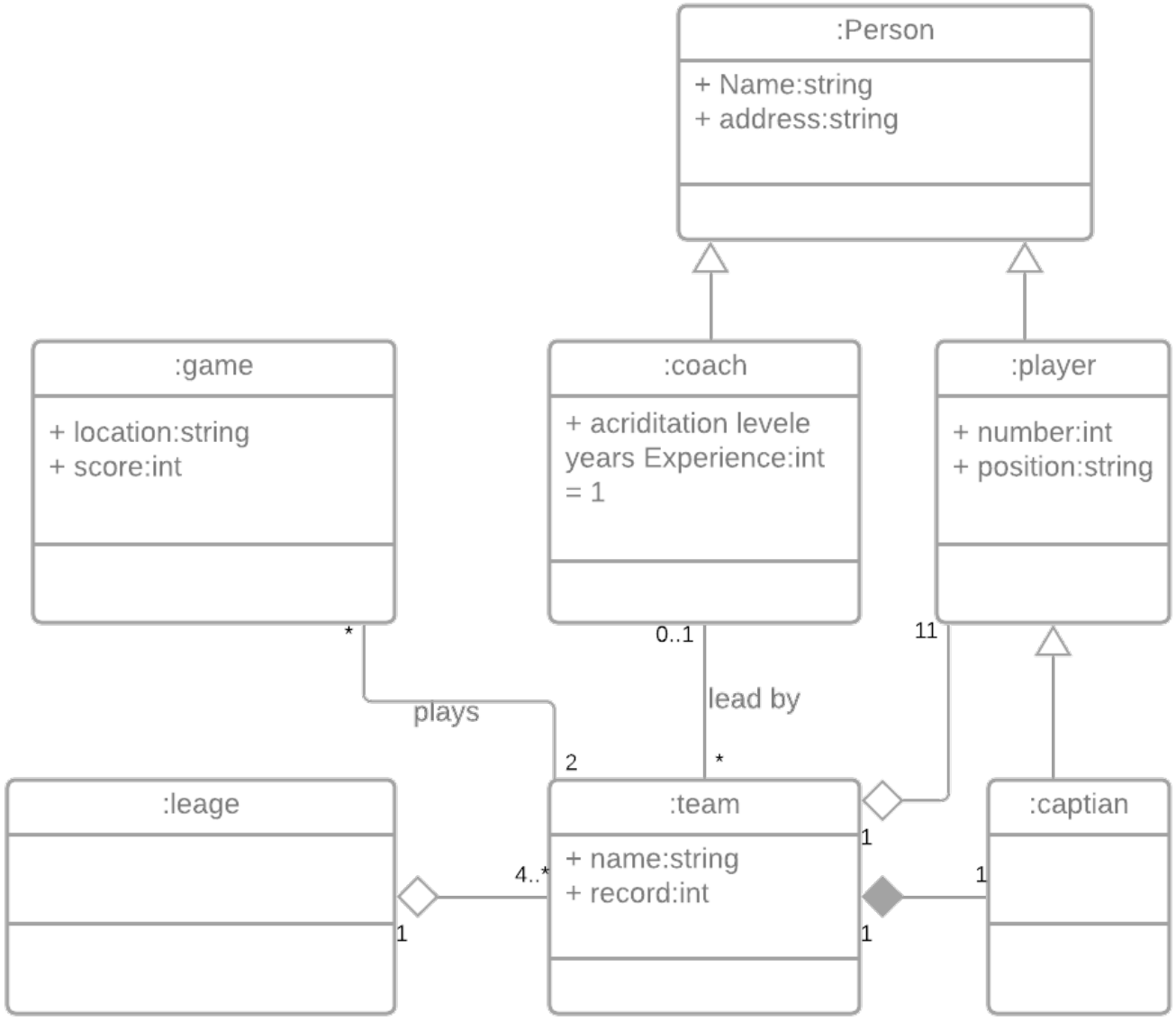


Exercise

- Draw a UML Class Diagram representing the following elements from the problem domain for a soccer league.

A soccer league is made up of at least four soccer teams. Each soccer team is composed of eleven players, and one player captains the team. A team has a name and a record. Players have a number and a position. Soccer teams play games against each other. Each game has a score and a location. Teams are sometimes lead by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses.

A Possible answer for the previous exercise



WHAT IS ENTITY?

- It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key.
- Every entity is made up of some ‘attributes’ which represent that entity.
- In business modeling entities represent objects that workers access, inspect, manipulate, produce, and so on.

….WHAT IS ENTITY?

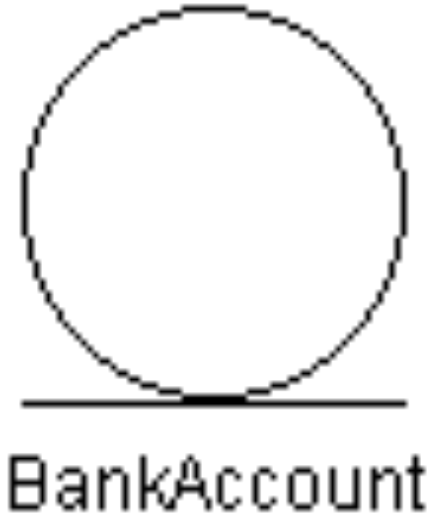
Examples of entities:

- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

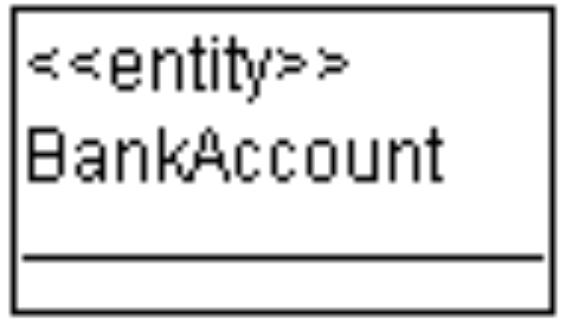
….WHAT IS ENTITY?

Entity Class

- An Entity Class is a stereotype of a class that is specified in UML Extensions for Business Modeling.
- It can be shown as a regular class rectangle with stereotype of "entity", or as this icon:



or

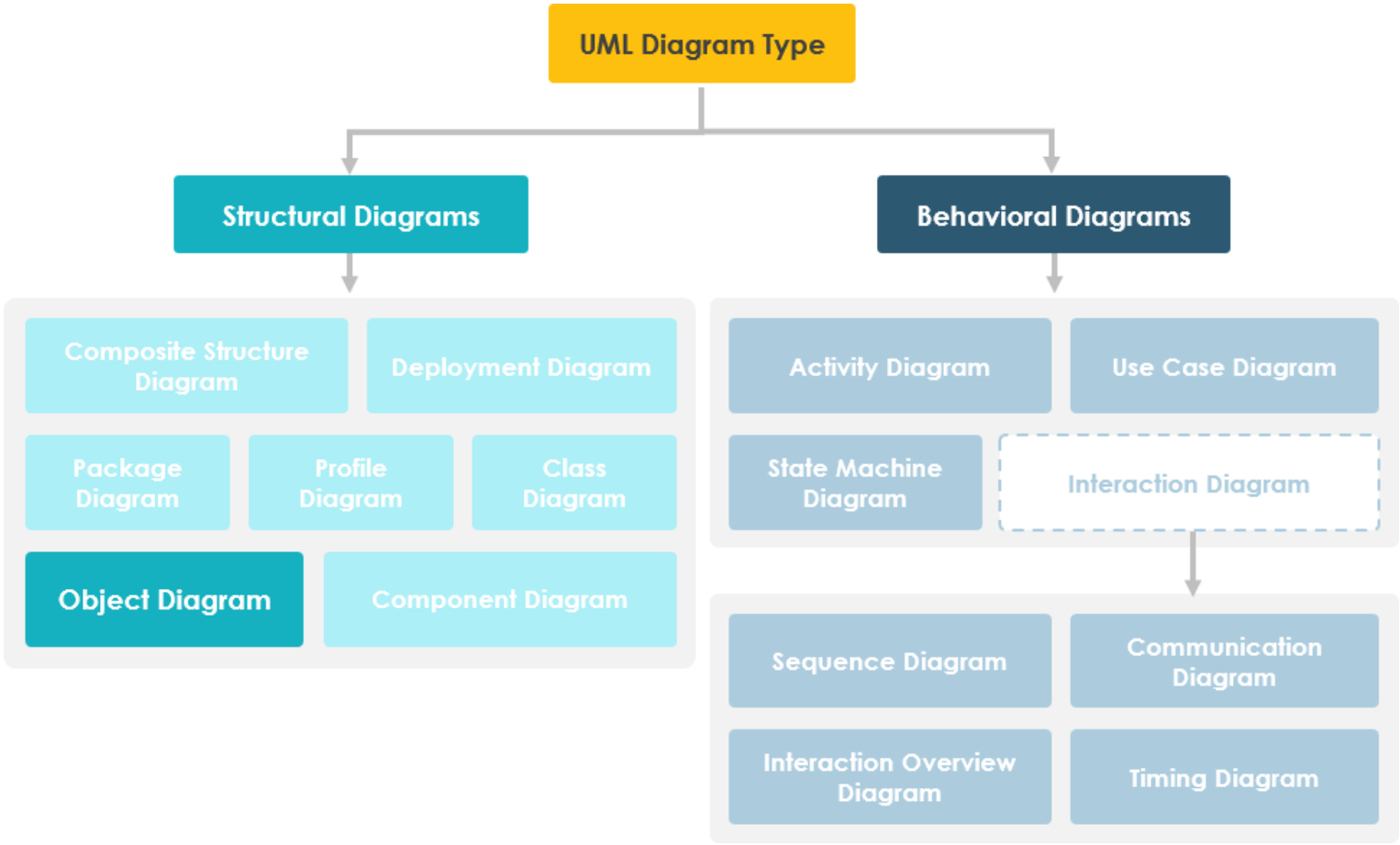


What is the difference between ER Diagram and Class Diagram?

- Although ER diagrams and Class diagrams are two of the design diagrams developers often come across during the design phases of software engineering projects, they have their key differences.
- ER diagrams represent the abstract representation of the data model, while class diagrams represent the static structure and behavior of the proposed system.
- Main building blocks of ER diagrams are entities, relationships and attributes but the main building blocks of class diagrams are classes, relationships and attributes.
- Class diagram are more likely to map in to real-world objects, while ER diagrams most often map in to the tables in the database.
- Usually, relationships found in ER diagrams are more difficult to understand for humans than relationships in class diagrams.

UML Object Diagram

Types of UML Diagrams



- **Structural diagrams** show the things in the modeled system to show different objects in a system.

- **Behavioral diagrams** show what should happen in a system and describe how the objects interact with each other to create a functioning system.

UML Object Diagram

- Object diagrams represent an instance of a class diagram.
- The basic concepts are similar for class diagrams and object diagrams.
- Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment.
- Object diagrams are used to render a set of objects and their relationships as an instance.
- It represents an instance at a particular moment which is concrete in nature.
- It means the object diagram is more close to the actual system behavior.
- The purpose is to capture the static view of a system at a particular moment.
- So the purpose of the object diagram can be summarized as:
    - Object relationships of a system
    - Static view of an interaction.
    - Understand object behavior and their relationship from practical perspective

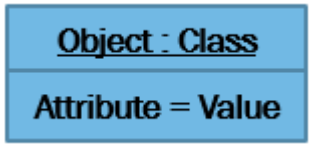
Basic Object Diagram Symbols and Notations

❑ Object Names:

- Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.

❑ Object Attributes:

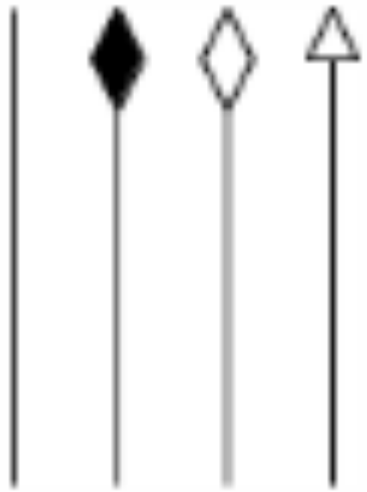
- Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.



Basic Object Diagram Symbols and Notations

Links:

- Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

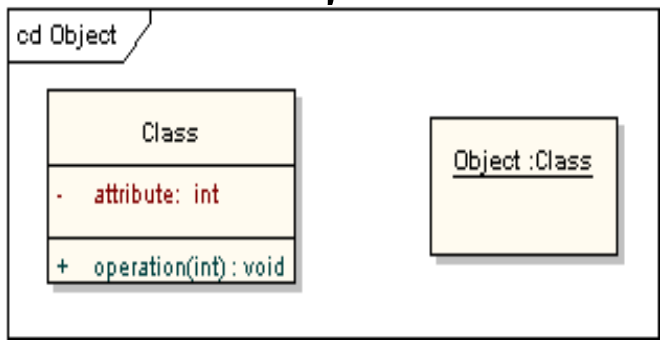


Class Diagram vs. Object Diagram

- In UML, object diagrams provide a snapshot of the instances in a system and the relationships between the instances.
- By instantiating the model elements in a class diagram, you can explore the behavior of a system at a point in time.
- An object diagram is a UML structural diagram that shows the instances of the classifiers in models.
- Object diagrams use notation that is similar to that used in class diagrams.
- Class diagrams show the actual classifiers and their relationships in a system
- Object diagrams show specific instances of those classifiers and the links between those instances at a point in time.
- You can create object diagrams by instantiating the classifiers in class, deployment, component, and use-case diagrams.

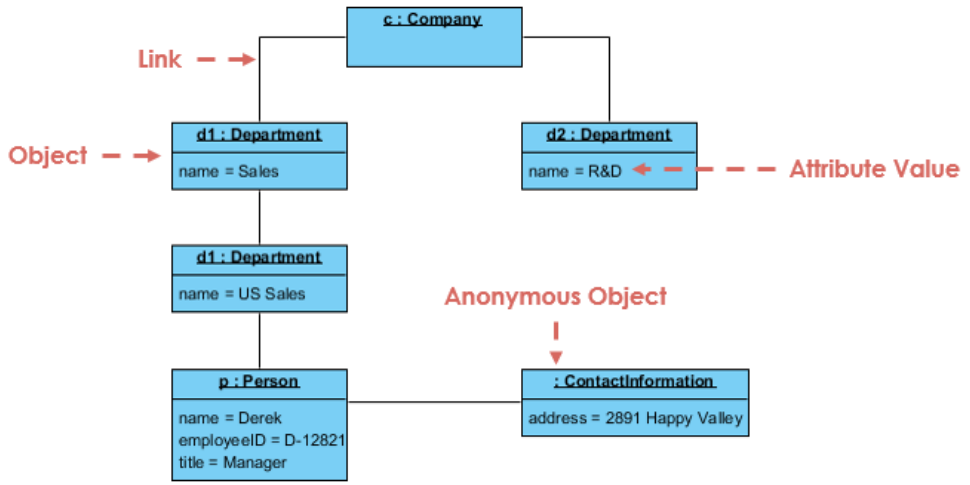
Class Diagram vs. Object Diagram

- **Appearance of Class and Object Elements**
- The following diagram shows the differences in appearance between a class element and an object element.
- Note that the class element consists of three parts, being divided into name, attribute and operation compartments; by default, object elements don't have sections.
- The display of names is also different: object names are **underlined** and may show the name of the classifier from which the object is instantiated.

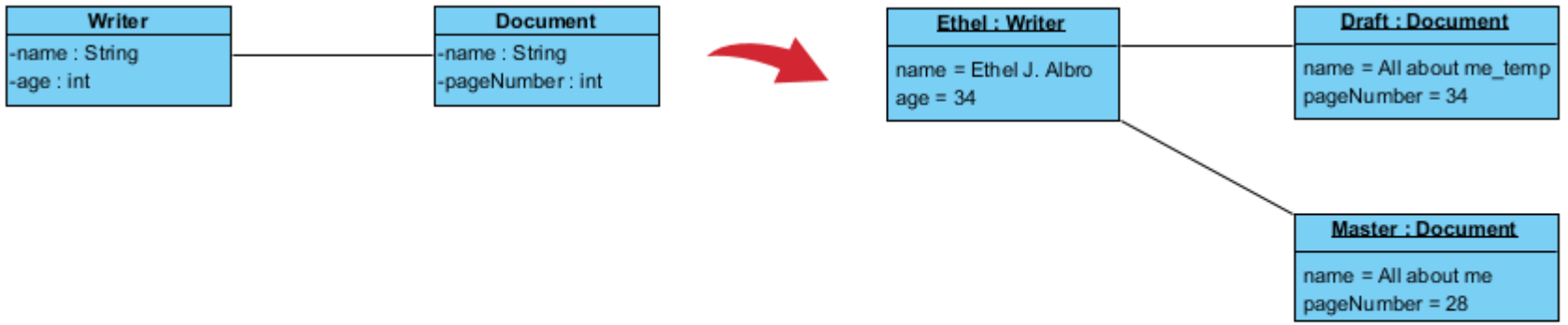


…Object diagram Examples

Object Diagram Example - Company Structure



Convert Class diagram to Object diagram example

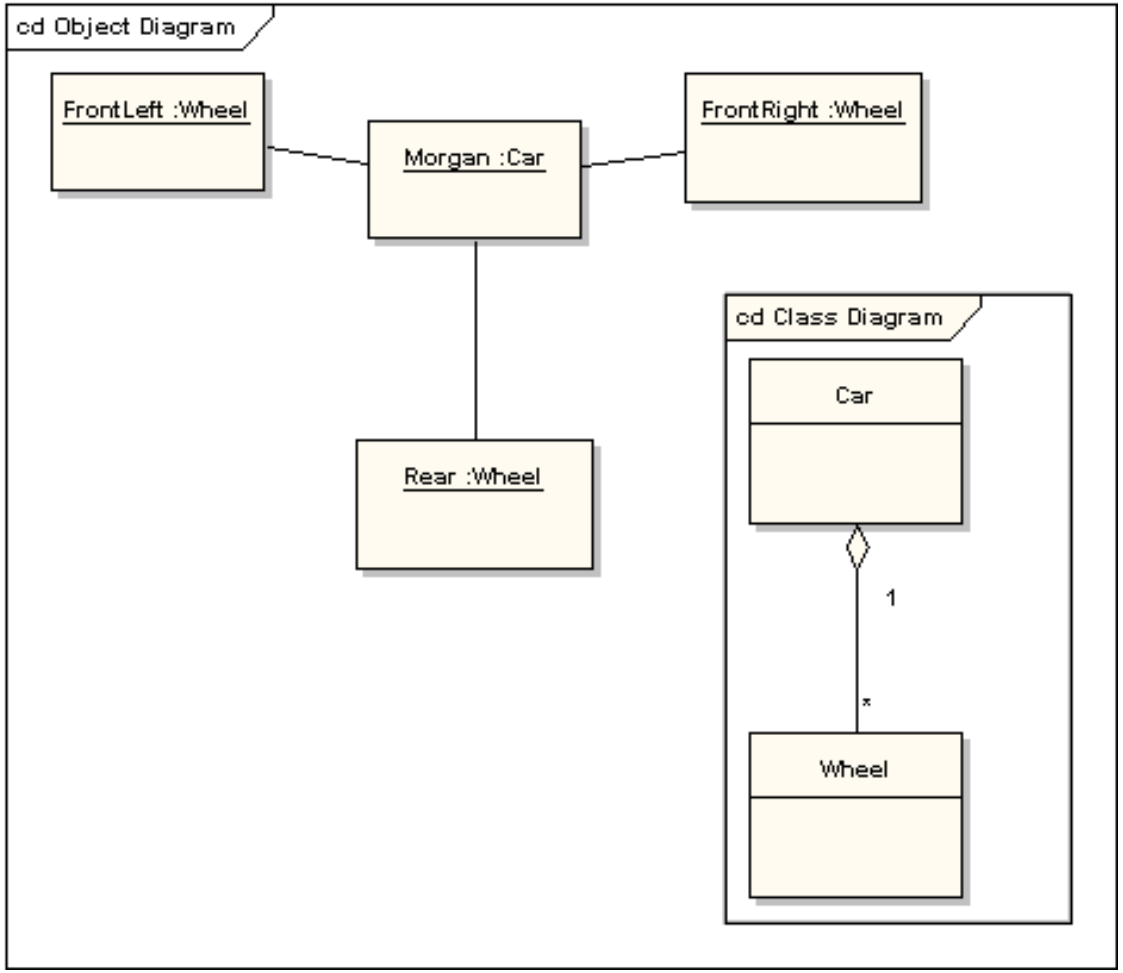


Class Diagram

Object Diagram

…Object diagram Example

The **car** class has a 1-to-many multiplicity to the **wheel** class



Architectural Design

Architectural Design

- It is about decomposing the system into interacting components.
- It is expressed as a block diagram defining an overview of the system structure, features of the components, and how these components communicate with each other to share data.
- It identifies the components that are necessary for developing a computer-based system and communication between them i.e. relationship between these components.
- It defines the structure and properties of the components that are involved in the system and also the interrelationships between these components.
- The architectural design process is about identifying the components i.e. subsystems that makeup the system and structure of the sub-system and they're interrelationship. It is an early stage of the system design phase. It acts as a link between specification requirements and the design process.

System properties of Architectural Design

- During the architectural design process, some system properties needs to be are as follows
  - **Security:** The system is secured against malicious users by encryption or any other security measures. The architectural design process uses a layered architecture with critical assets in the innermost layers.
  - **Performance**: It is nothing but a meantime taken between request and response of the page.
    - The performance of a system can be improved by avoiding critical operations and reducing communication between components. This is possible by using large components instead of small and fine-grained components.

…System properties of Architectural Design

- **Maintainability:** Architectural design process uses easily modifiable and replaceable components. Consider the components in a manner so that it is easy to change them over time according to the new requirements and build the software with the flexibility to change or maintain.
- **Safety:** Avoid critical functionalities in small components of the system.
- **Availability:** Architectural design process includes redundant components and corresponding functions for handling the occurrence of any type of errors.

Decisions for Architectural Design

- The architectural design process differs as the system differs depending upon the type of system being developed.
- But still, there are some common decisions that should be taken care of in any design process are as follows:
  - How can the system be distributed across the network?
  - Which approach can be used to structure the system?
  - Is there a generic application architecture procedure that can be used which can act as a template for the proposed system that is being designed and developed?
  - Which architectural styles are suitable for the proposed system?
  - How can software architecture be documented?
  - How can the system be decomposed into modules?
  - What control strategy must be used to control the operation of the components in the system?
  - How can architectural design be analyzed?

Software Architectural Models

- To document the architectural design process, architectural models are used
  - ***Static type of architectural structural model***:- represents the major system components.
  - ***Dynamic type of architectural process model*** :- represents the process structure of the system.
  - ***Distribution type of architectural model***:- represents how the component is distributed across various computers.
  - ***Interface type of architectural model***:- represents the interface of the components.
  - ***Relationships type of architectural model***:- represent models such as data flow diagram to represent the component interrelationship.

…Software Architectural Models

- Architectural design models are application domain-specific and the most common two types of domain-specific models are:
  - **Generic model:** These models are abstractions derived from a number of real systems and encapsulated the characteristics of these systems. This type of model usually follows a bottom-up approach.
  - **Reference models:** These models provide information regarding the class of the system. They are derived from the application domain rather than from existing systems. It usually follows the top-down approach. It provides a comparison between different Software architecture.

Advantages of Architectural

- Architectural design works as a tool for stakeholder communication. It is used as a support or roadmap in the discussion with system stakeholders
- It is used for system analysis. Architectural design is used to analyze whether the system will be able to meet its non-functional requirements or not.
- It facilitates large-scale re-use. The software architecture that is the output of the architectural design process can be reused across a range of the system.

Disadvantages of Architectural Design

- Architectural design re-use the components, the use of redundant components improves the availability but makes the security of the system difficult to handle.
- Use of large components may improve the performance as large components include all the related properties and function into one component but it reduces the maintainability as it becomes difficult to modify and replace the large component. It involves a very tiresome task.
- Avoiding critical features inside the small components leads to more communication among the components which in turn degrades the performance.

To be continued…

Chapter 7

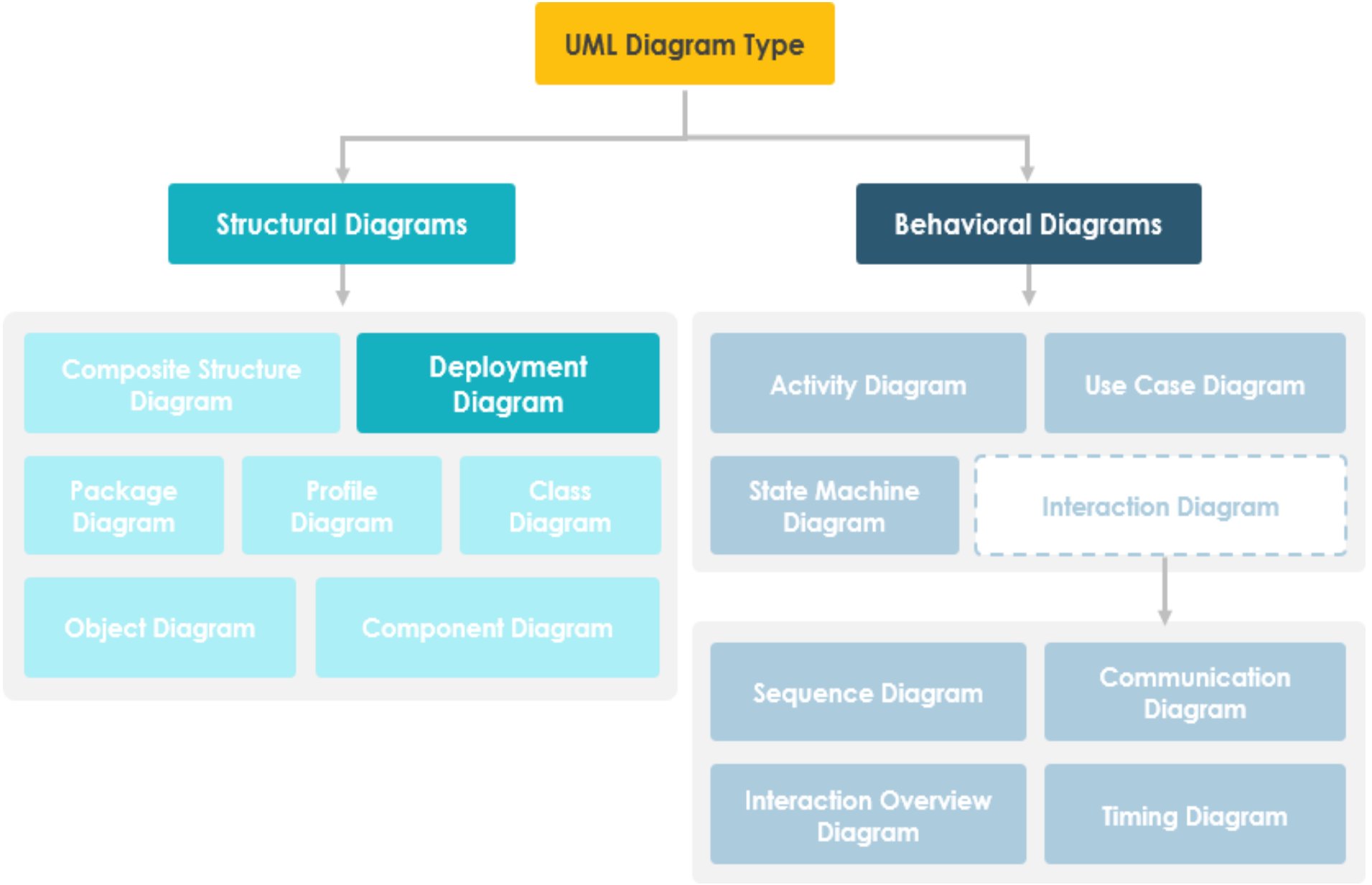
….Cont’d

Contents

Structural Diagram

- **Deployment Diagram**
- **Component Diagram**
- **Package Diagram (Reading Assignment)**

UML Deployment Diagram



What is Deployment Diagram?

- Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute.
- It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.
- The deployment diagram maps the software architecture created in design to the physical system architecture that executes it.
- In distributed systems, it models the distribution of the software across the physical nodes.
- In general, deployment diagram visualizes the topological view of an entire system.

….What is Deployment Diagram?

- A deployment diagram plays a critical role during the administrative process, and it must satisfy the following parameters,
  - High performance
  - Maintainability
  - Scalability
  - Portability
  - Easily understandable
- Nodes and artifacts are the essential elements of deployment.
- Before actually drawing the deployment diagram, all nodes and the relationship between every node of the system must be identified.

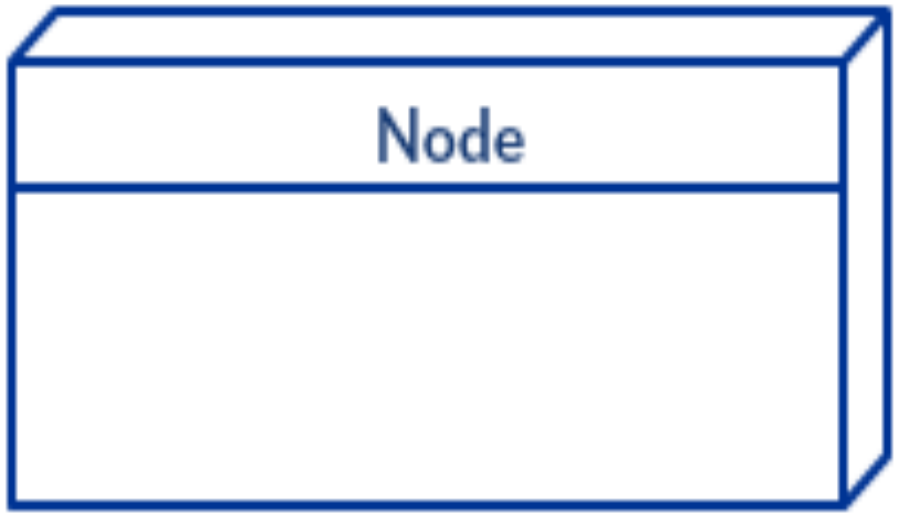
….cont’d

- We must know the architecture of a system:-
  - Is it a web application?
  - Cloud application?
  - Desktop application?
  - Or a mobile application? All these things are critical and plays a vital role during the development of a deployment diagram.
- If all the nodes, relations, and artifacts are known, then it becomes easy to develop a deployment diagram.

Deployment Diagram Notations

Node

- In order to draw a deployment diagram, we need to first become familiar with deployment diagram notations and deployment diagram elements.
- A node, represented as a cube, is a physical entity that executes one or more components, subsystems or executables.



…….Cont’d

- Node is computational resource upon which artifacts are deployed for execution.
- A node may vary in its size depending upon the size of the project.
- Node is an essential UML element that describes the execution of code and the communication between various entities of a system.
- An association between nodes represents a communication path from which information is exchanged in any direction.

Deployment Diagram Notations

Artifact

- Represents the specification of a concrete real-world entity related to software development.
- We can use the artifact to describe a framework which is used during the software development process or an executable file.
- Artifacts are deployed on the nodes.

<<Artefacts>>

- **The most common artifacts are as follows,**
  - Source files , Executable files ,
  - Database tables ,Scripts, DLL files,
  - User manuals or documentation,
  - Output files

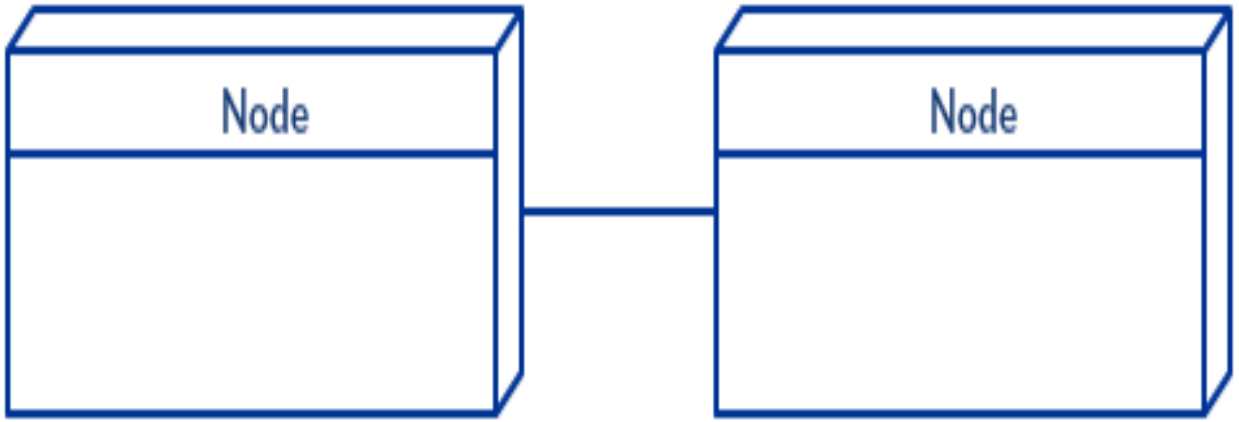
…….Cont'd

- Generally, There are two types of nodes in a deployment diagram: **device nodes** and **execution environment nodes**.
  - **Device nodes** are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones.
  - **An execution environment node, or EEN**, is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements. It could be an operating system, a JVM, or another servlet container.

…….Cont’d

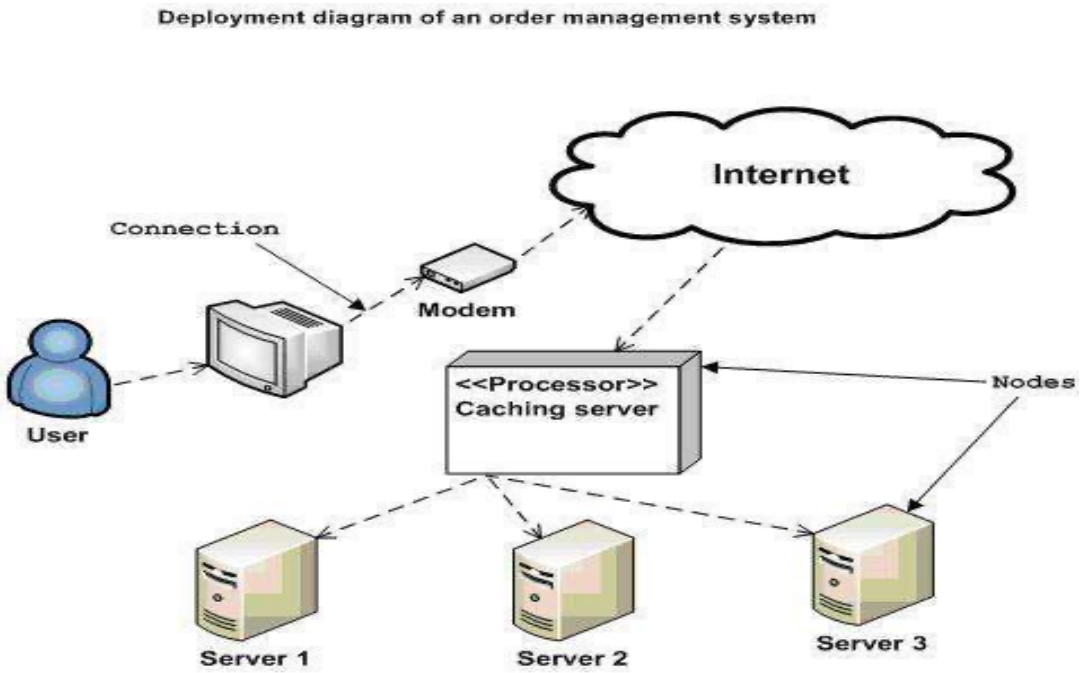
Communication Association

- This is represented by a solid line between two nodes. It shows the path of communication between nodes.



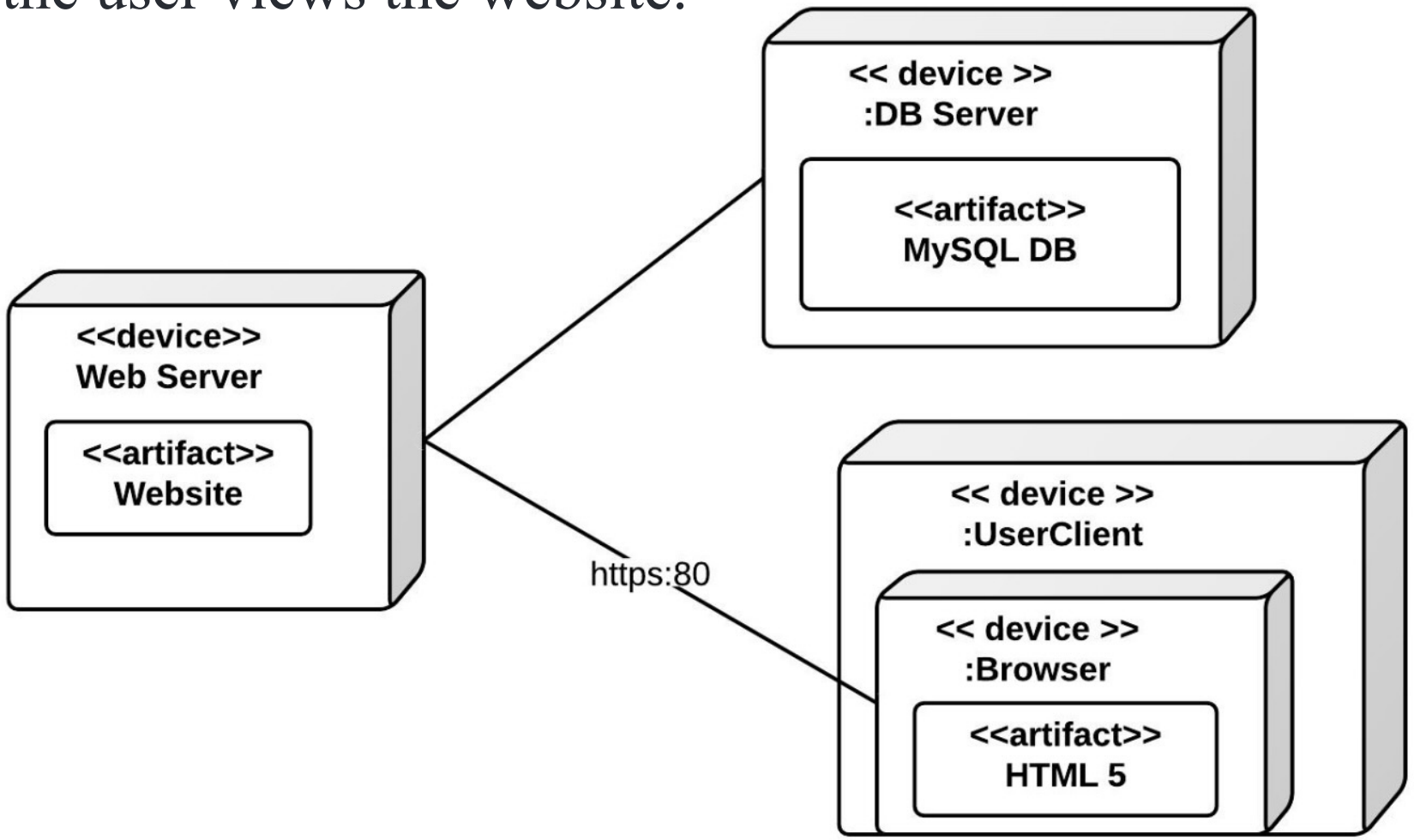
...Example of a Deployment diagram

- Following is a sample deployment diagram to provide an idea of the deployment view of order management system. Here, we have shown nodes as Monitor ,Modem, Caching server, and Servers.
- The application is assumed to be a web-based application, which is deployed in a clustered environment using server 1, server 2, and server 3. The user connects to the application using the Internet. The control flows from the caching server to the clustered environment.



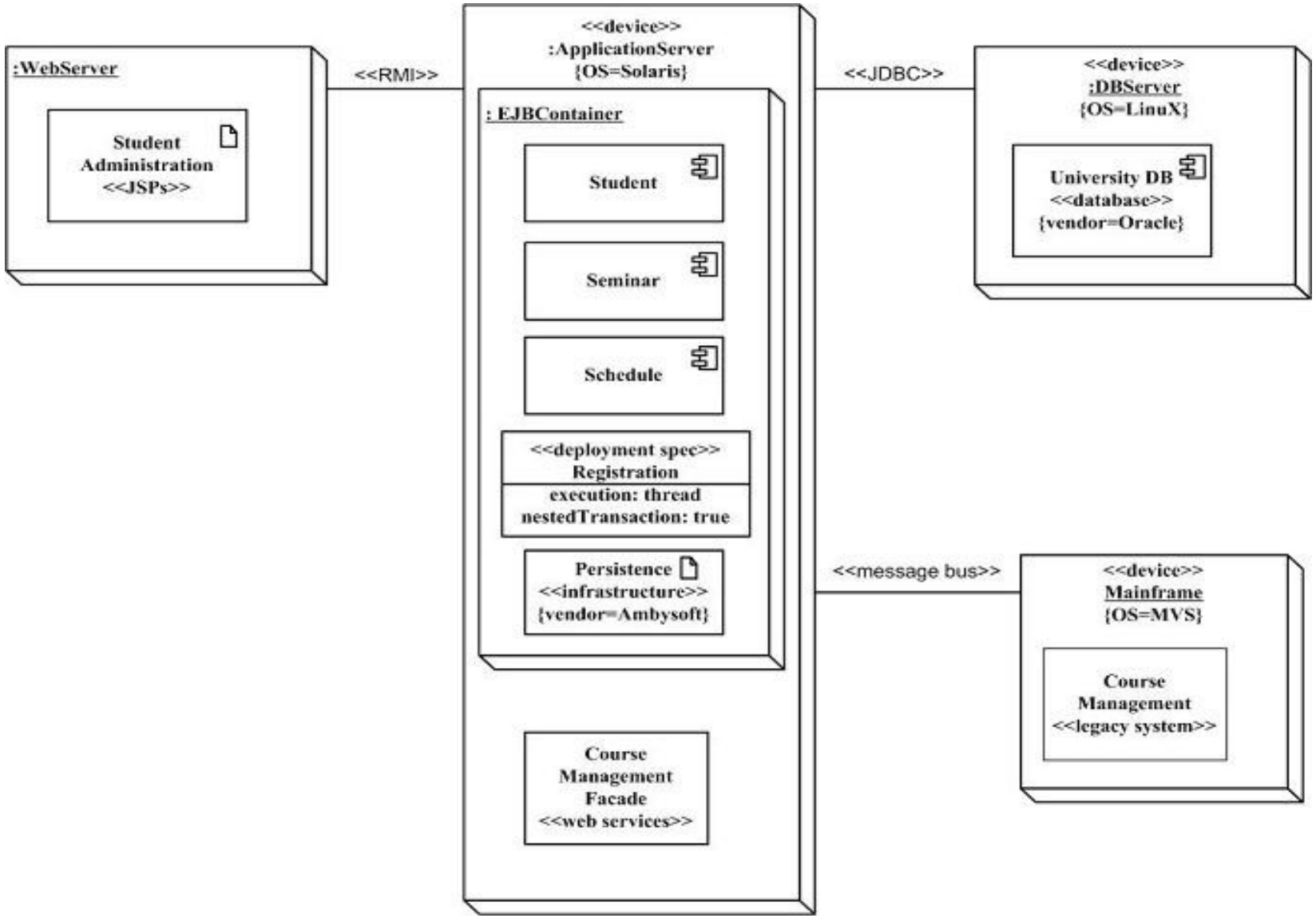
Example-2 Deployment diagram

- This example shows a basic deployment diagram which consists of a web server, a database server, and the machine where the user views the website.

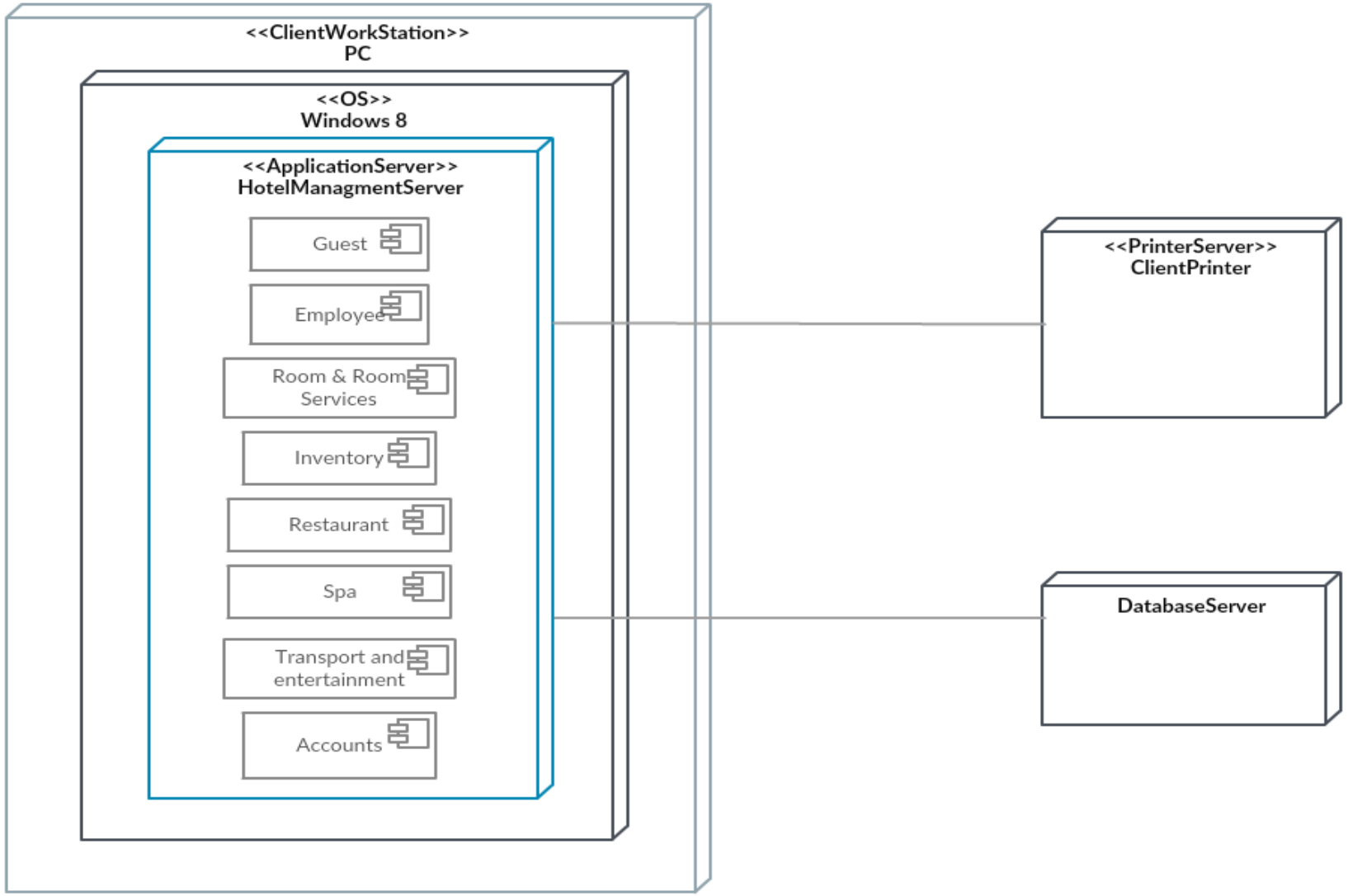


Example-3

- UML 2 deployment diagram for the university information system.



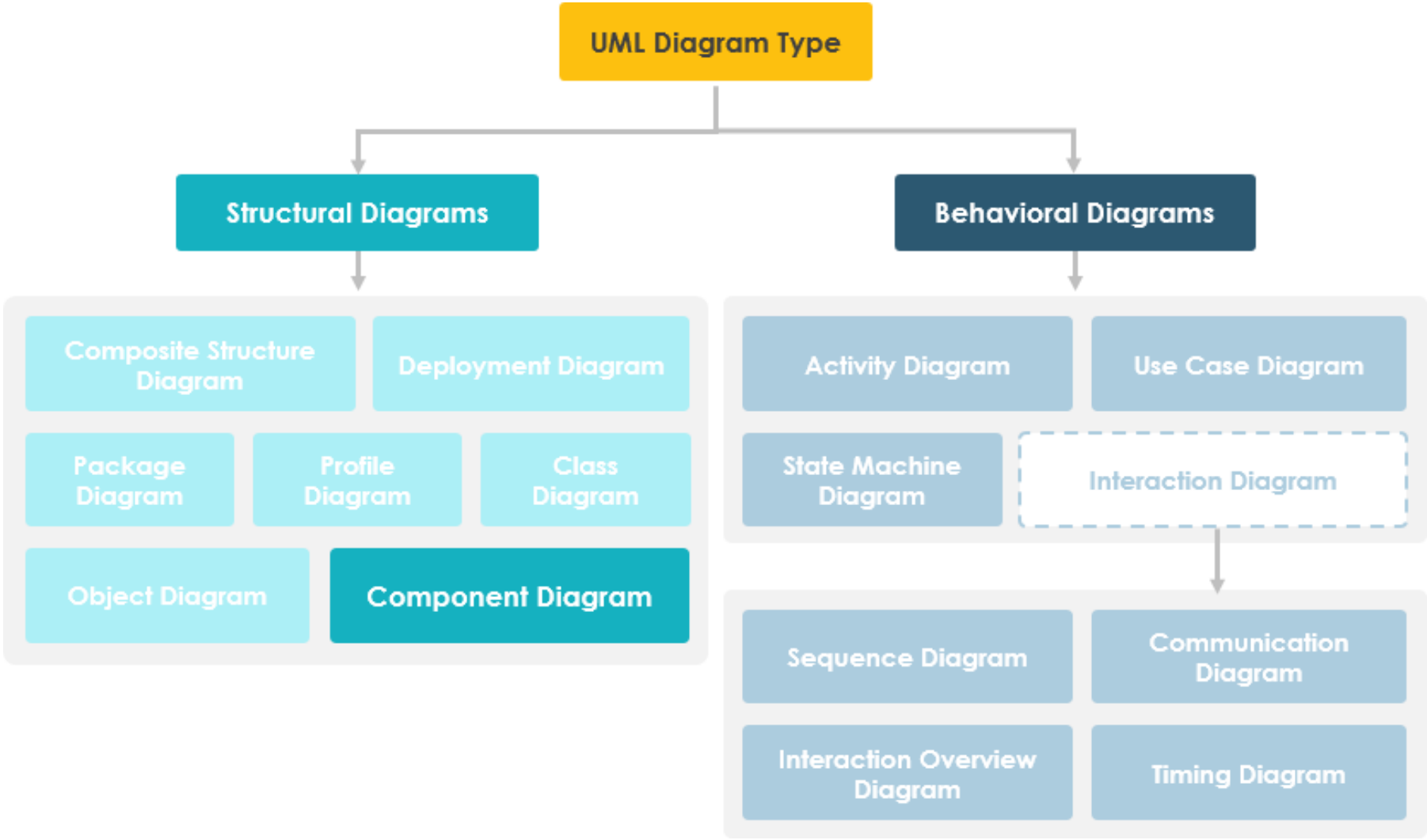
Deployment Diagram for Hotel Management System



Summary

- The deployment diagram maps the software architecture created in design to the physical system architecture that executes it.
- It maps software pieces of a system to the hardware that are going to execute it.
- Deployment diagram visualizes the topological view of an entire system.
- Nodes and artifacts are the essential elements of deployment.
- Node and artifacts of a system participate in the final execution of a system.

UML Component Diagram



Component diagrams

- The term "component" refers to a module of classes that represent independent systems or subsystems with the ability to interface with the rest of the system.
- The components can be a software component such as a database or user interface; or a hardware component such as a circuit, microchip or device; or a business unit such as supplier, payroll or shipping.
- Help communicate and explain the functions of the system being built to stakeholders
- Helps in identifying the components that make up the system and show the relationship between components.
- Component diagram is replaceable and executable peace of a system whose implementation detail is hidden.

…..Component diagrams

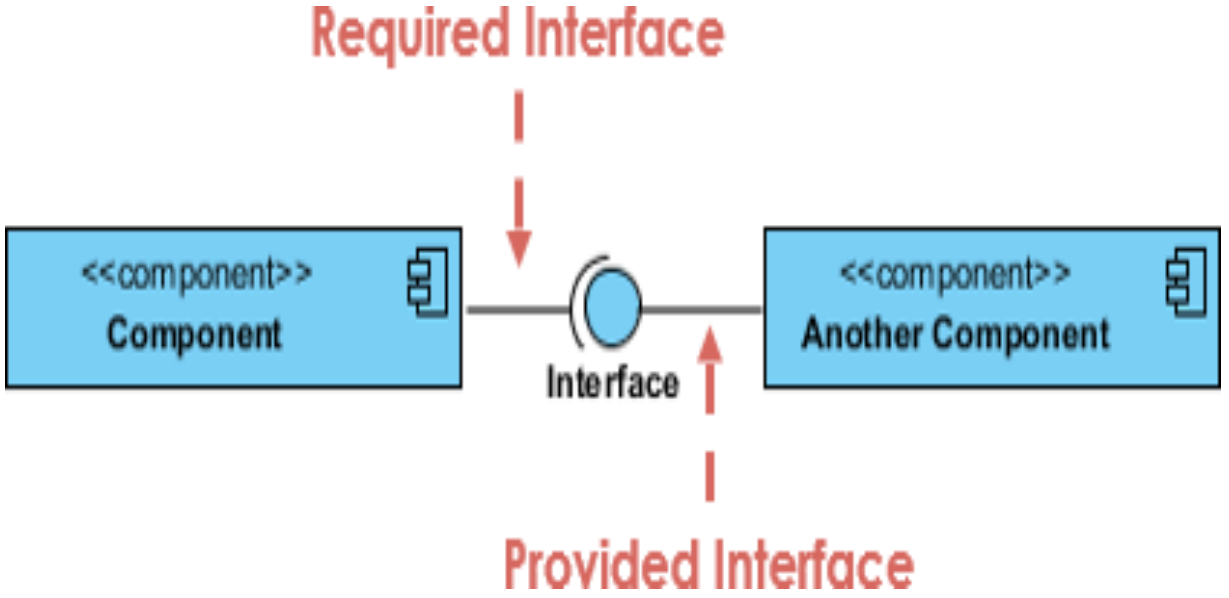
- Component diagram shows the structural relationship between the components of a system.
- Components are considered autonomous, encapsulated units within a system or sub-system that provides one or mode interfaces.
- Components can be interchanged and reused and they interacts via interface
- Helps to divides systems into components and show their relationships.

Component Diagram Symbols

- Important to see the common component diagram notations that are used to draw a component diagram.
- A high-level, abstracted view of a component in UML 2 can be modeled as:
  - A rectangle with the component's name
  - A rectangle with the component icon
  - A rectangle with the stereotype text and/or icon

….Component Diagram Symbols

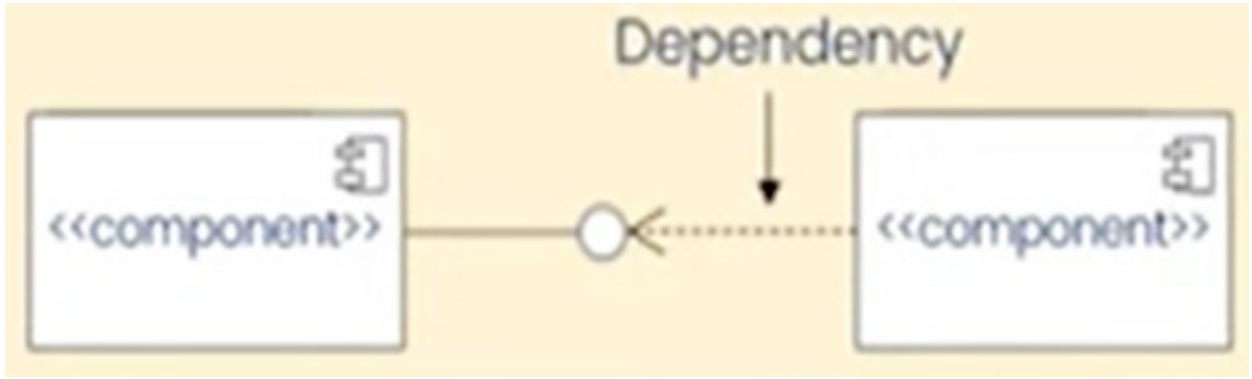
Provided Interface and the Required Interface

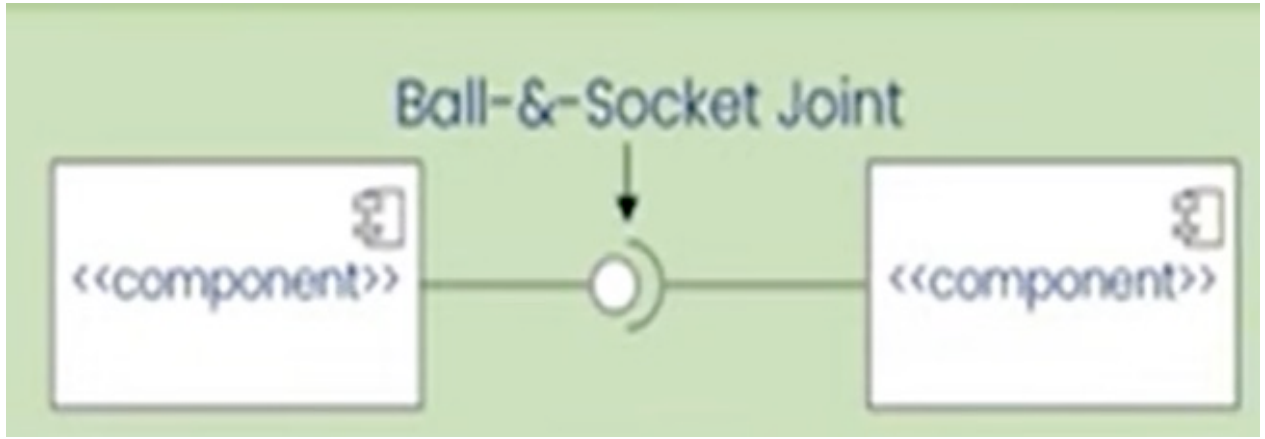


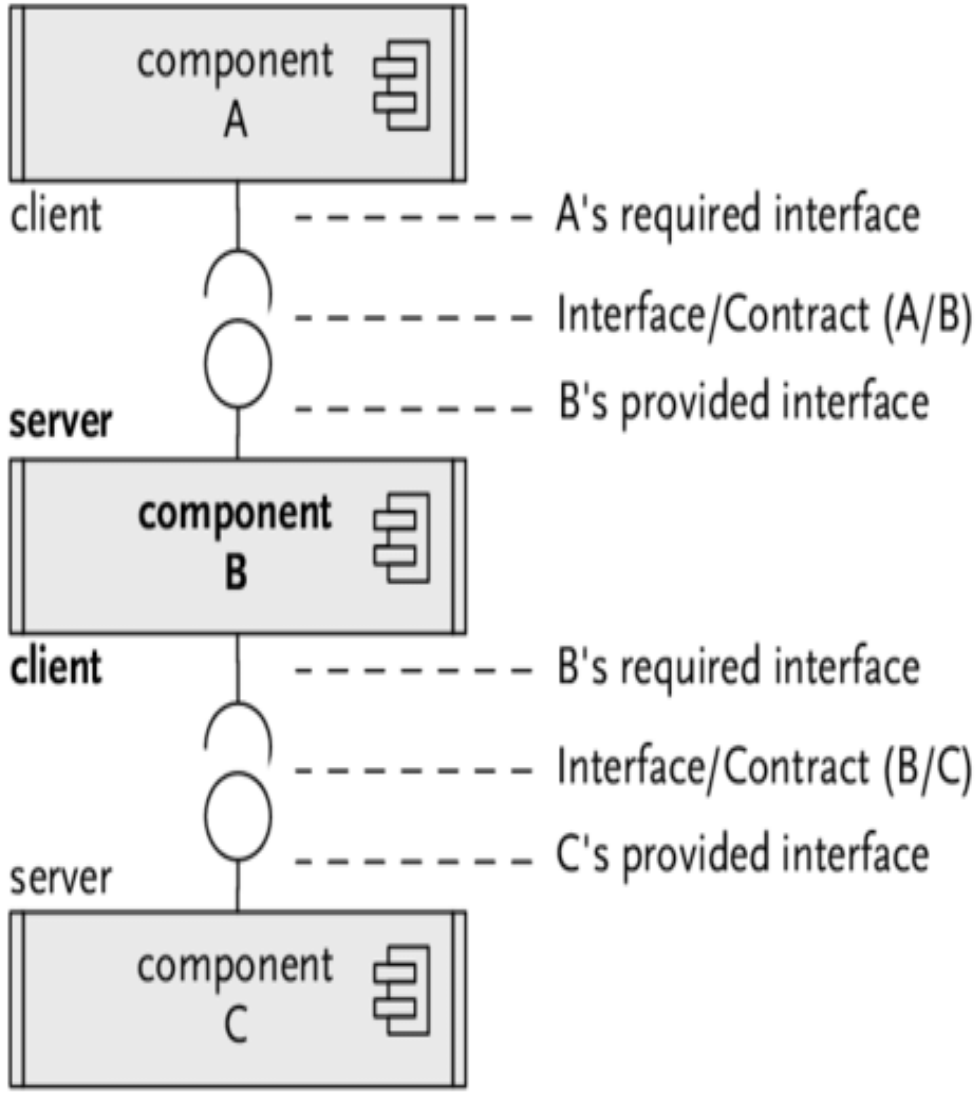
- Interfaces in component diagrams show how components are wired together and interact with each other.
- Required interface (represented with a semi-circle and a solid line)
- Provided interface (represented with a circle and solid line). This shows that one component is providing the service that the other is requiring.

….Component Diagram Symbols

Example: Interaction between Provided Interface and the Required Interface



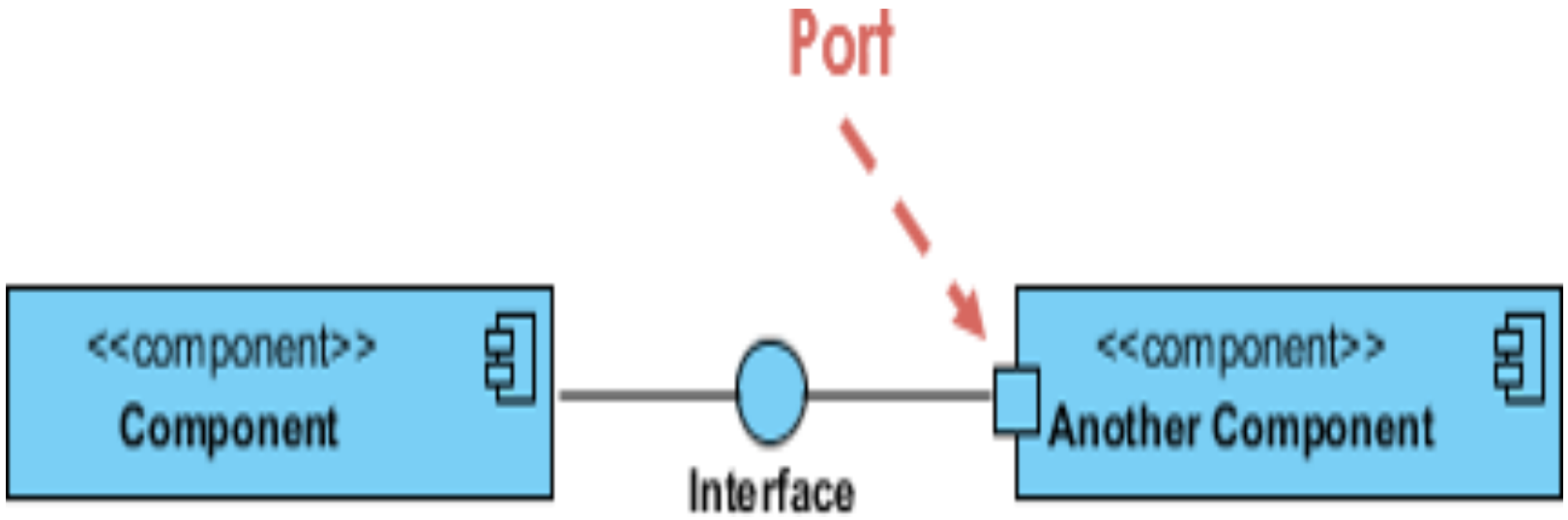




….Component Diagram Symbols

Port

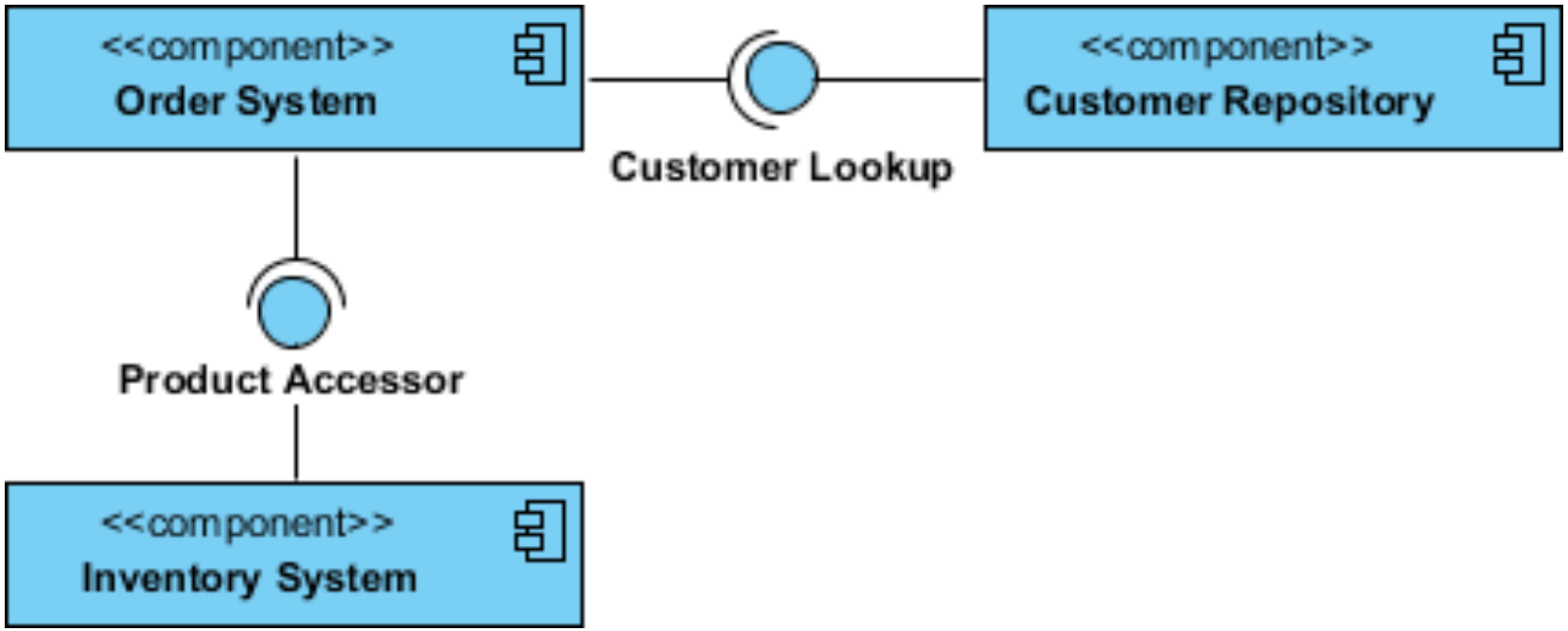
- Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.



- Although we can show more detail about the relationship between two components using the **ball-and-socket notation** (provided interface and required interface), we can just as well use a **dependency arrow** to show the relationship between two components.

….Component Diagram

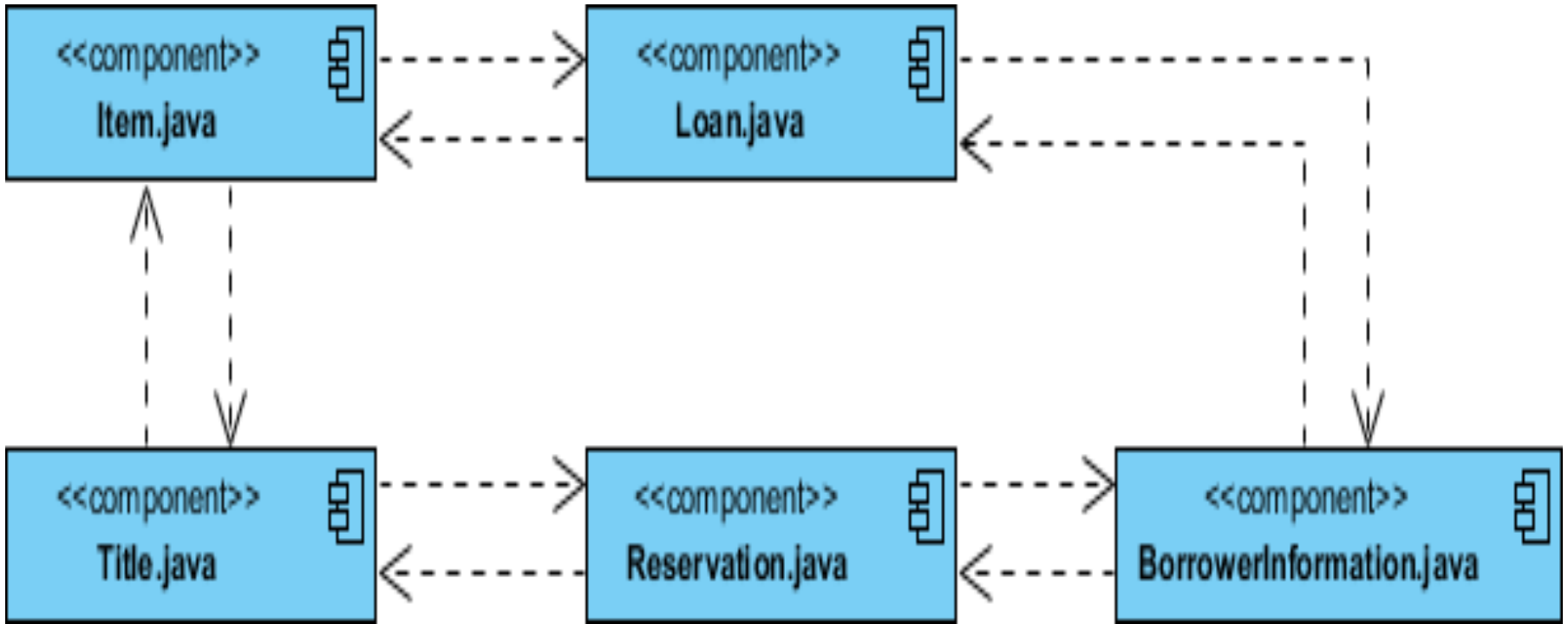
- Component Diagram Example - Using Interface (Order System)



Cont'd

Modeling Source Code

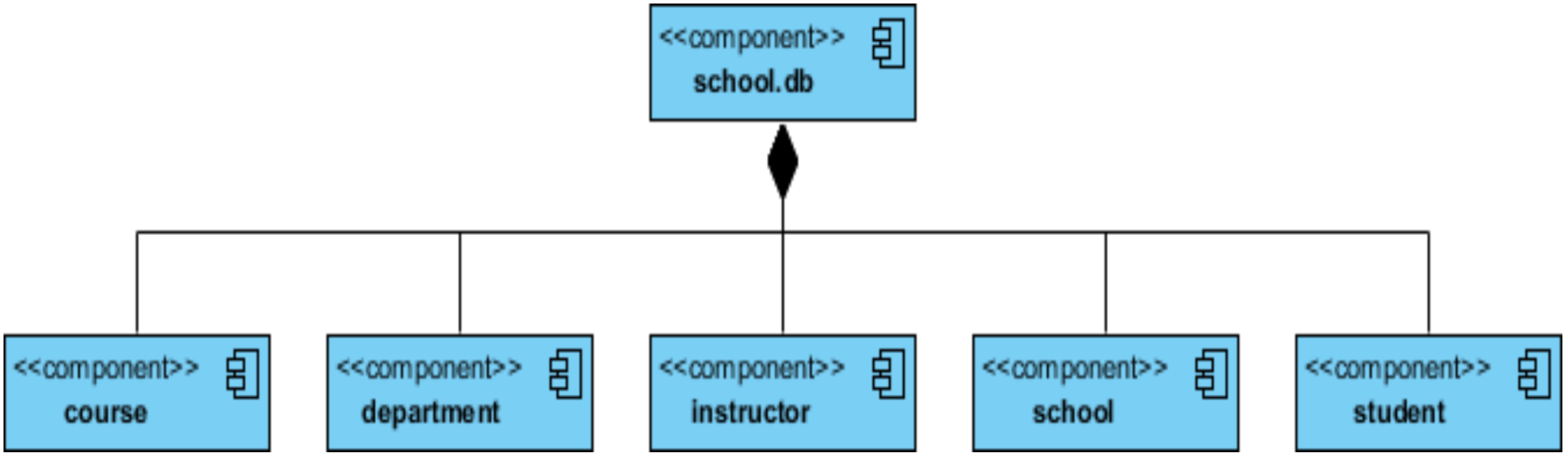
- Either by forward or reverse engineering, **identify the set of source code files** of interest and model them as components stereotyped as files.
- Component Example - Java Source Code



….Cont’d

Modeling a Physical Database

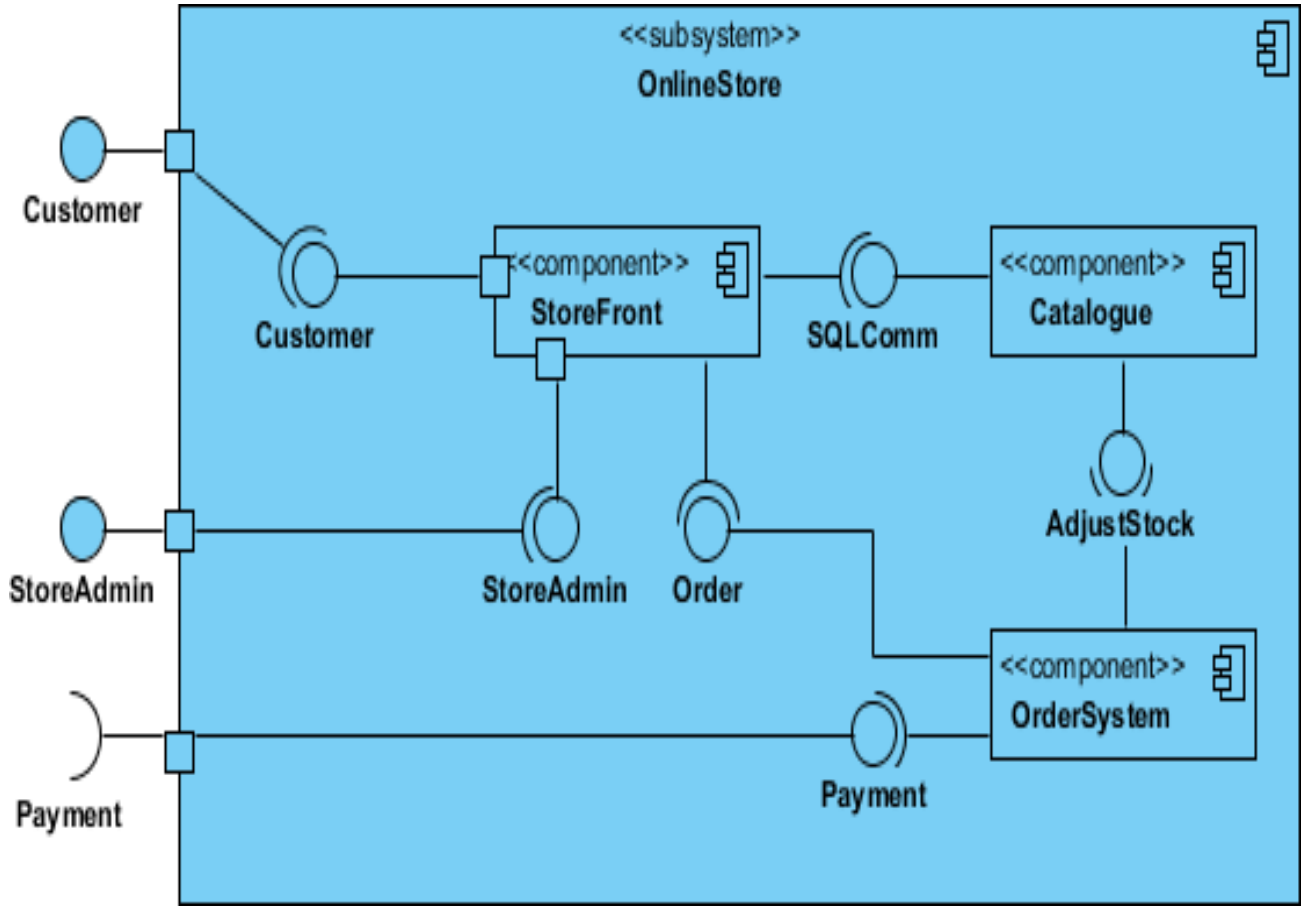
- Identify the classes in your model that represent logical database schema.
- To visualize, specify, construct, and document our mapping, create a component diagram that contains components stereotyped as tables.



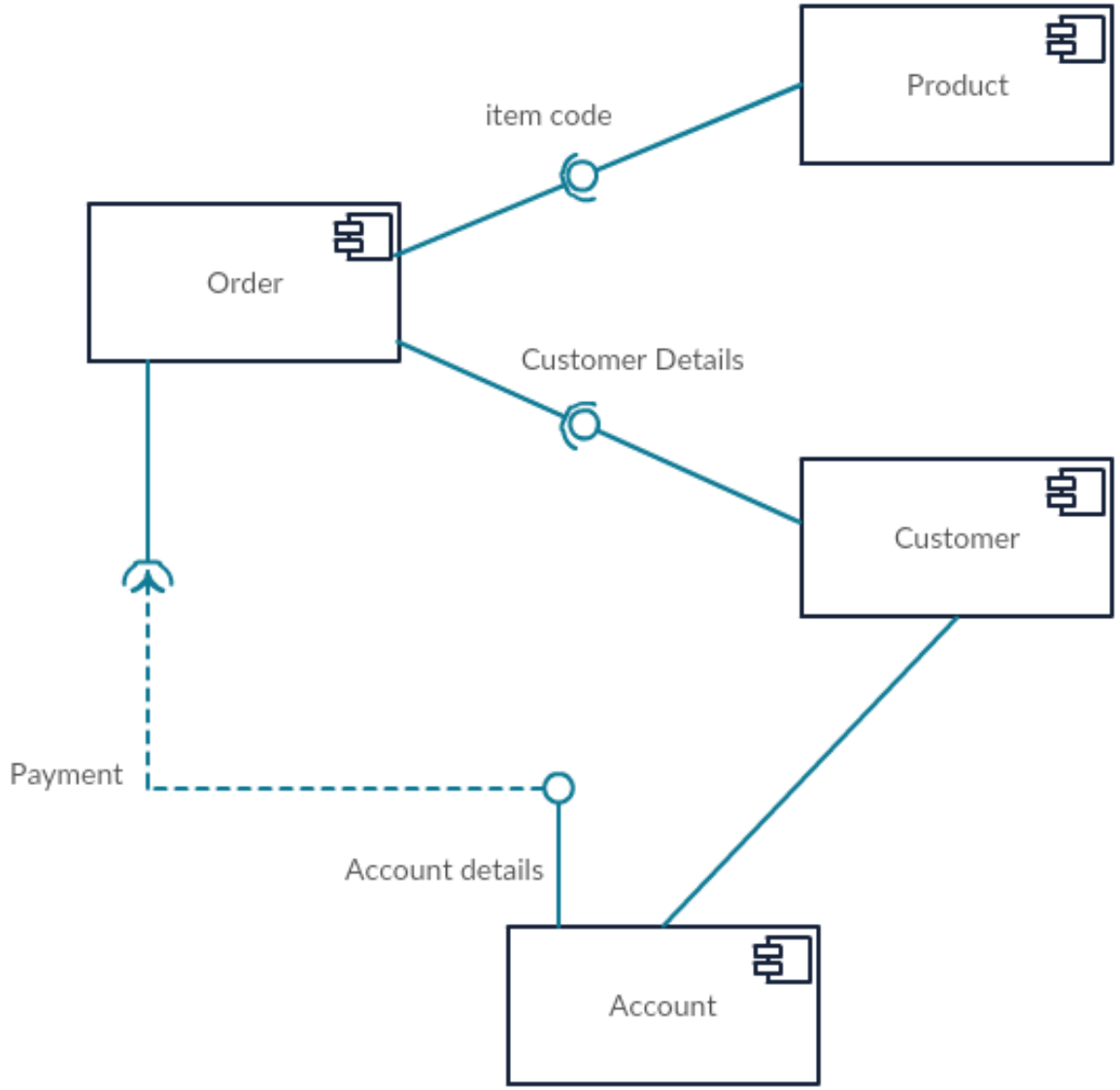
….Cont’d

Subsystems

- The subsystem notation element inherits all the same rules as the component notation element. The only difference is that a subsystem notation element has the keyword of subsystem instead of component.



Example :- Component Diagram for Online Shopping System



Exercise -1

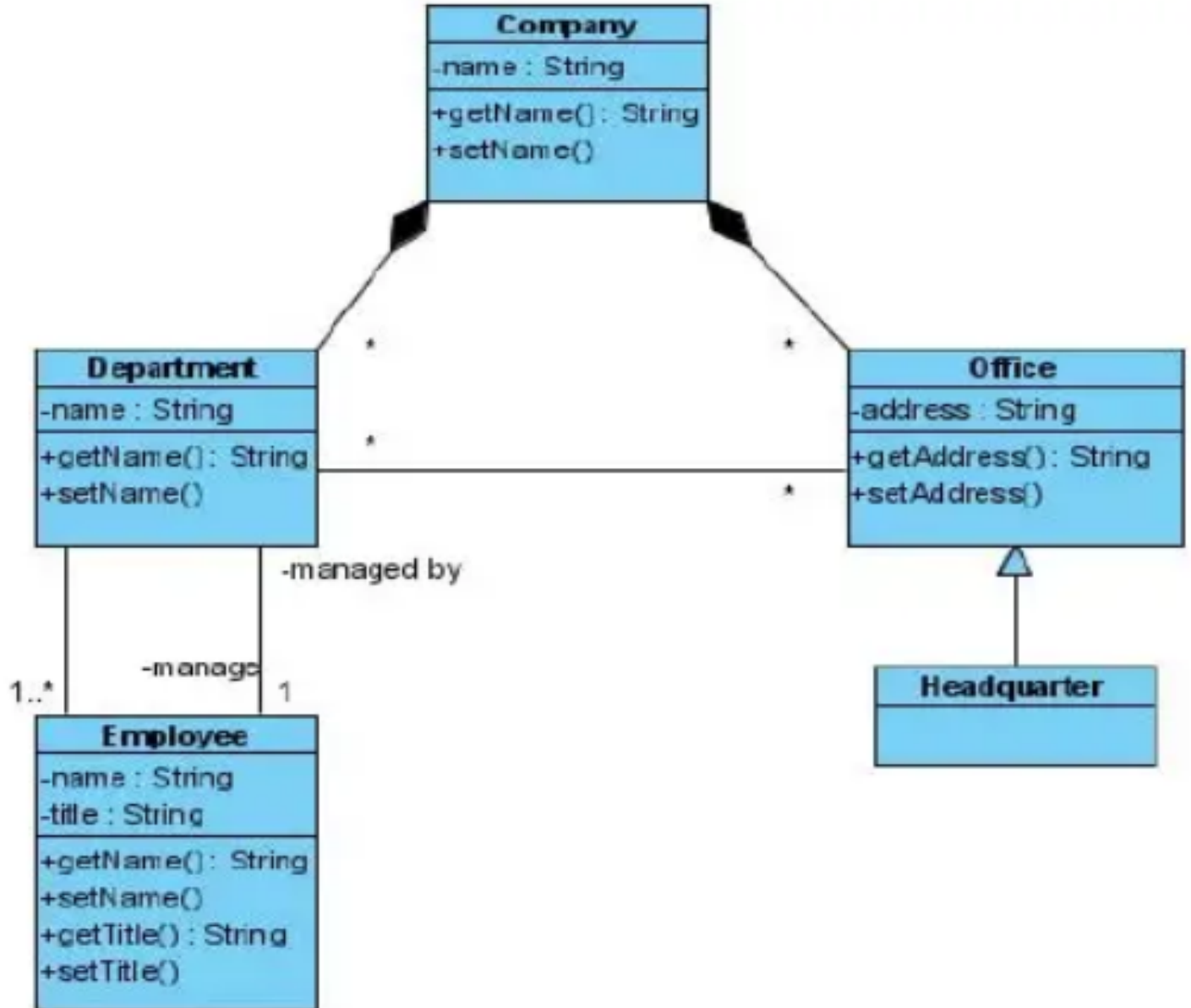
Question 1:

- A company consists of departments. Departments are located in one or more offices. One office acts as a headquarter. Each department has a manager who is recruited from the set of employees.

Task:

- Draw a class diagram which consists of all the classes in your system their attributes and operations, relationships between the classes, multiplicity specifications, and other model elements that you find appropriate

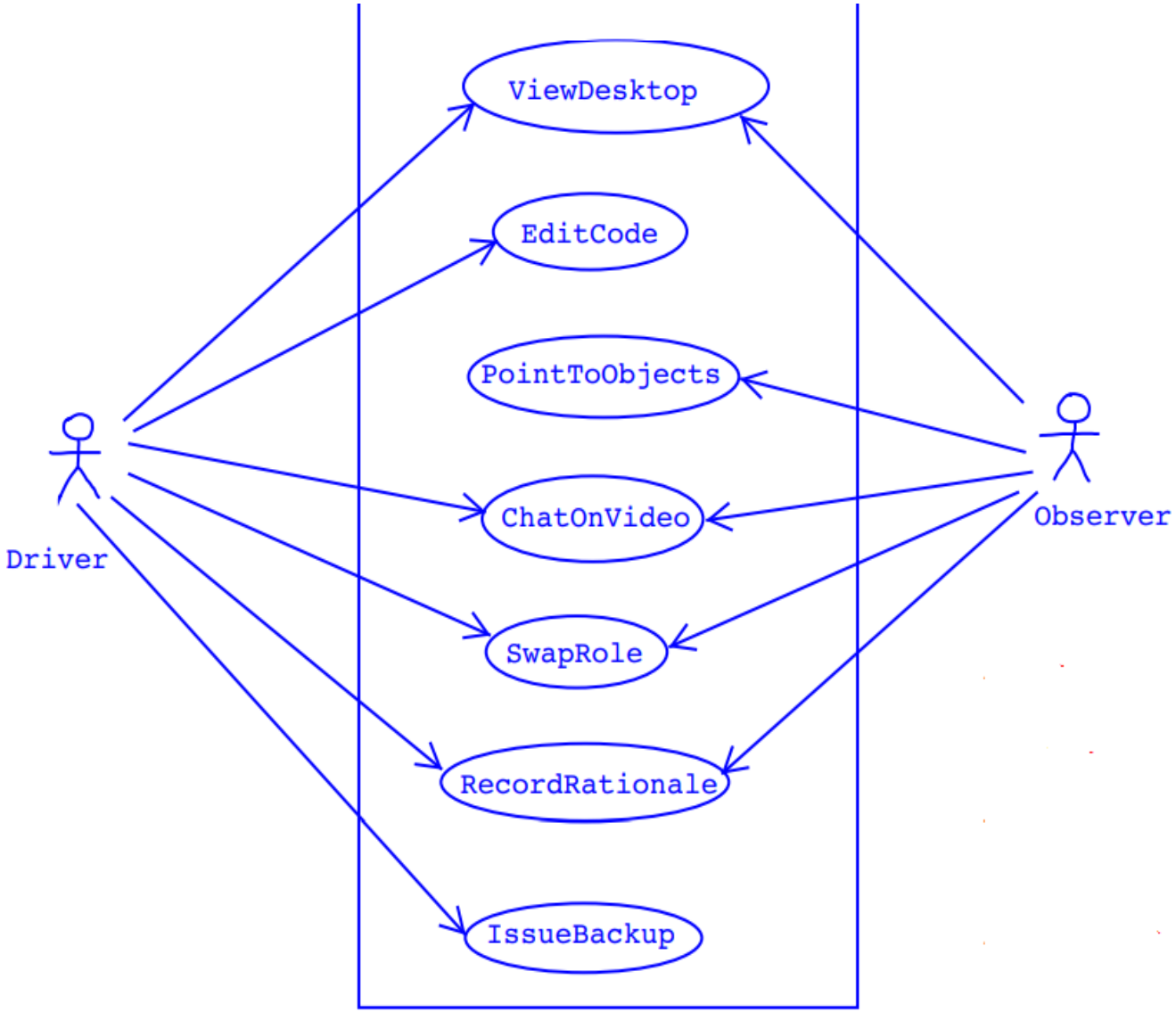
Possible Solution



Exercise -2

- Pair programming is an agile software development technique in which two programmers work together at one work station. One types in code while the other reviews each line of code as it is typed in. The person typing is called the **driver.** The person reviewing the code is called the **observer.** The two programmers switch roles frequently (possibly every 30 minutes or less).
- Suppose that you are asked to build a system that allows Remote Pair Programming. That is, the system should allow the driver and the observer to be in remote locations, but both can view a single desktop in real-time. The driver should be able to edit code and the observer should be able to "point" to objects on the driver's desktop. In addition, there should be a video chat facility to allow the programmers to communicate. The system should allow the programmers to easily swap roles and record rationale in the form of video chats. In addition, the driver should be able to issue the system to backup old work. Draw a use case diagram for the given system.

Possible solution



End