50.003
Elements of Software Construction
Lecture 3

Domain Modelling and UML Class Diagrams

Scope

- What is a Problem Domain and a Domain Model?
- Representing a Domain Model using UML Class Diagram
- Deriving a Domain Model
- UML Class diagrams
  - Relationships
  - Associations
  - Multiplicity
  - Association class

Learning Outcomes

❖ **To be able to do the following:**

- Explain what a Problem Domain is
- Explain what a Domain Model is
- Explain the following various elements of a domain model
  - – Relationships
  - – Associations
  - – Multiplicity
  - – Association class
- To derive a Domain Model

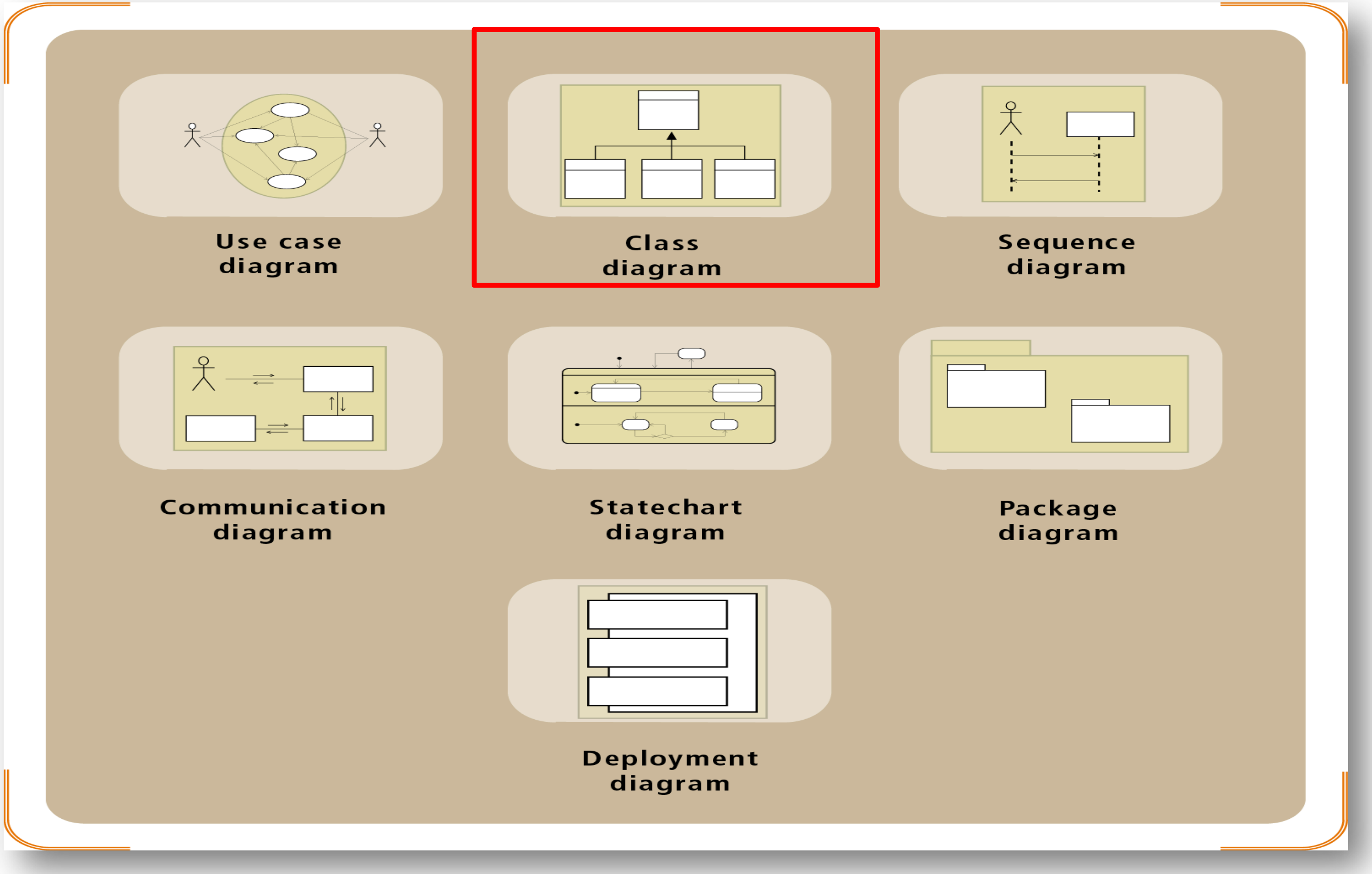
Problem Domain

- Specific area
  - an area of expertise or application that needs to be examined to solve a problem.
  - help focus on the relevant topics and exclude everything else irrelevant to the problem.
  - Contains things users deal with when they do their work and need to be part of the system
  - Eg: products, orders, invoices and customers
    - These things make up the data about which the system stores information
- Context
  - it defines the environment in which the problems exist.
- Scope
  - The problem domain outlines the boundaries of the issues to be solved.

Problem Domain Model

- Represent the real-world business concepts (things) and their relationships in a clear and unambiguous way
- Provides a structured representation of the key elements relevant to solving a particular problem
- steps in defining requirements
- Importance
  - Requirements Clarity
  - Design Decisions
  - Abstraction and Modeling
  - Trade-offs and Prioritization
- Identifying and understanding things in problem domain is a key initial

UML Diagrams used for Modelling



UML Class Diagrams

- Object Oriented approach to representing:
  - Things in a problem domain (**Domain** class diagram)
  - Objects that interact in the system (**Solution/Design** class diagram)
- Domain Class Diagrams model set of important data representation within a problem domain.
  - Examples: products, orders, invoices, customers
- Solution Class Diagrams model objects within the system that has the capability to interact to either keep or track information or both. (Java, C# or Python classes)

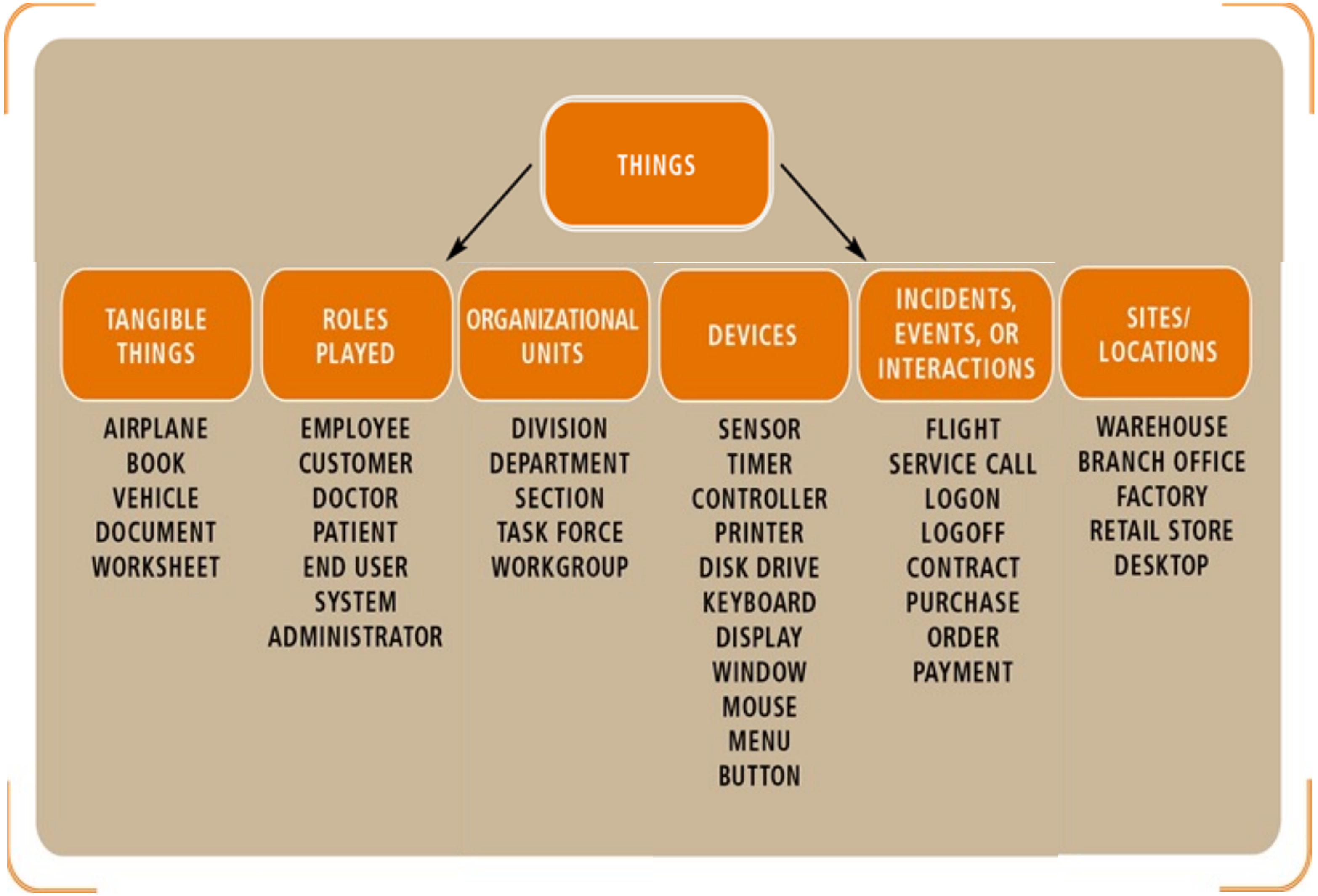
Types of Domain Concepts

- **Identified** through users sharing “things” regarding their work routine.
  - Include information from **all types of users**
  - supplies, materials, products and records of transactions
- Separate the **tangible** from the **intangible**
  - **Tangible:** Catalog, an item in the catalog
    - **Nouns** users mention when discussing system
  - **Intangible:** an order
    - Ask questions about **nature of event**
    - “What interactions should be acknowledged and recorded by the system?”

E.g. Customer places an order.

Order is an intangible object arising from customer interaction with item in inventory

Types of Things



THINGS

TANGIBLE THINGS

AIRPLANE
BOOK
VEHICLE
DOCUMENT
WORKSHEET

ROLES PLAYED

EMPLOYEE
CUSTOMER
DOCTOR
PATIENT
END USER
SYSTEM ADMINISTRATOR

ORGANIZATIONAL UNITS

DIVISION
DEPARTMENT
SECTION
TASK FORCE
WORKGROUP

DEVICES

SENSOR
TIMER
CONTROLLER
PRINTER
DISK DRIVE
KEYBOARD
DISPLAY
WINDOW
MOUSE
MENU
BUTTON

INCIDENTS, EVENTS, OR INTERACTIONS

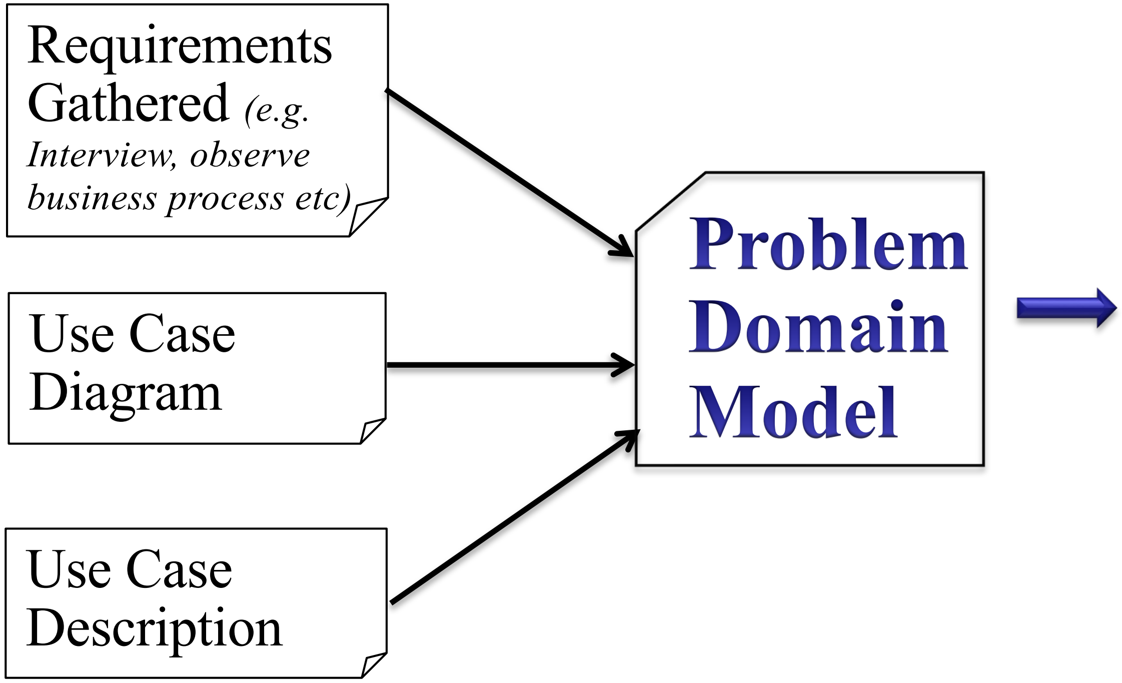
FLIGHT
SERVICE CALL
LOGON
LOGOFF
CONTRACT
PURCHASE ORDER
PAYMENT

SITES/ LOCATIONS

WAREHOUSE
BRANCH OFFICE
FACTORY
RETAIL STORE
DESKTOP

Source of potential things

- Use cases and their description, system events, external agents , system requirements, triggers and responses, interview notes etc.

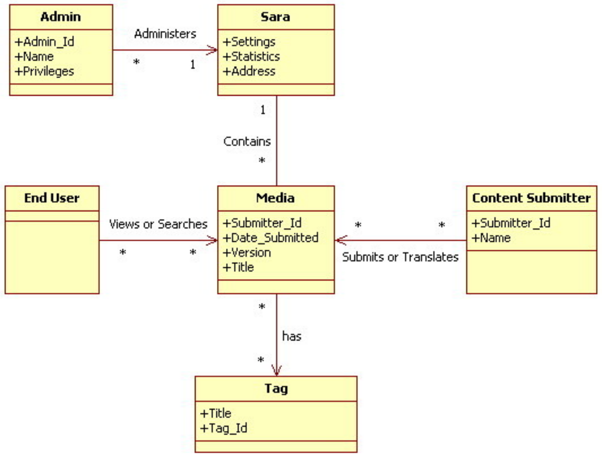


Requirements Gathered *(e.g. Interview, observe business process etc)*

Use Case Diagram

Use Case Description

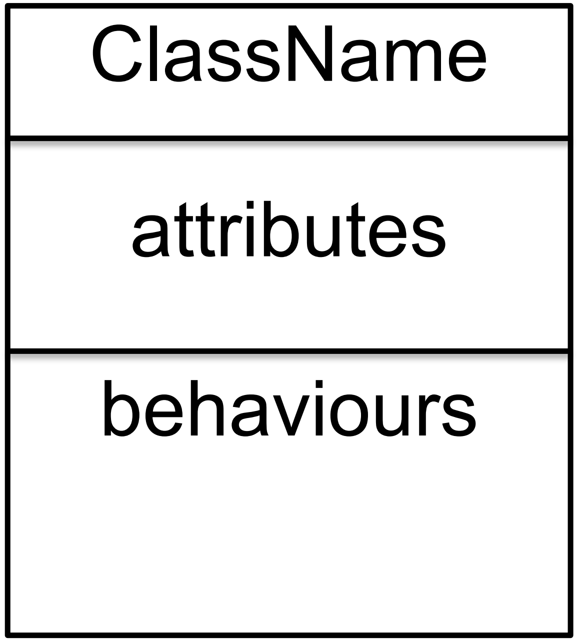
Problem Domain Model



Domain class Diagram

(UML) Class Diagram Notation

- Class diagram
  - General class symbol: rectangle with three sections
  - Sections convey class name (normally singular and upper case), attributes, and behaviors
  - Methods (behaviors) not shown in domain model class diagram



ClassName

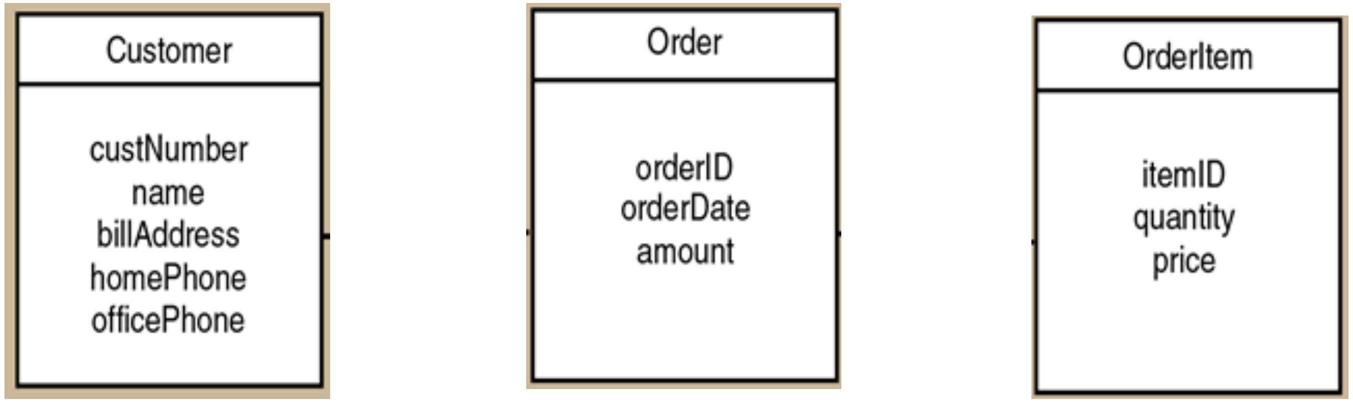
attributes

behaviours

UML Class Diagram

Attributes of Things

- Specific details of things are called attributes
- Attributes of things dictate the data that need to be stored with these things
- Identifier (key): attribute uniquely identifying thing
  - Examples: Social Security number, vehicle ID number, or product ID number



Customer

custNumber
name
billAddress
homePhone
officePhone

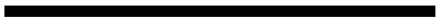
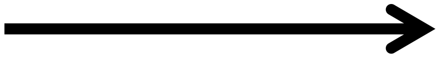
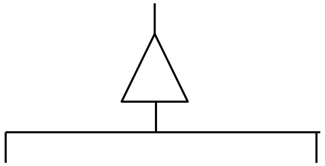
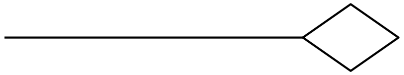
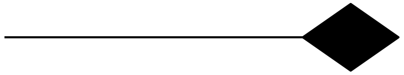
Order

orderID
orderDate
amount

OrderItem

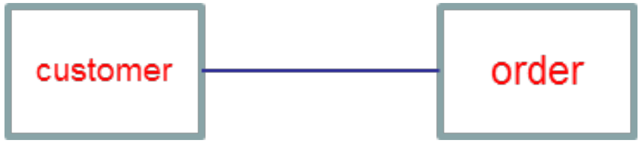
itemID
quantity
price

Types of Class Relationships

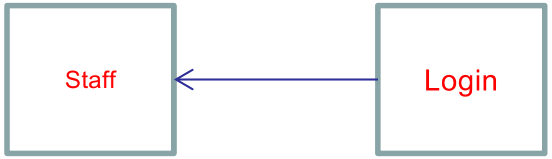
Relationship	**Notation**
1. Association	
i. Bi-directional	
ii. Uni-directional	
2. Generalisation/Specialisation	
i. a.k.a Inheritance ii. “Is a kind of”	
3. Aggregation i. “Consist of”	
4. Composition i. “made up of”	

Associations among Things

- An association is a naturally occurring relationship among specific things
  - Example: “Is placed by” and “works in”
- Associations apply in two directions
  - **Bi-directional** Associations
    - Customer places an order
    - An order is placed by a customer

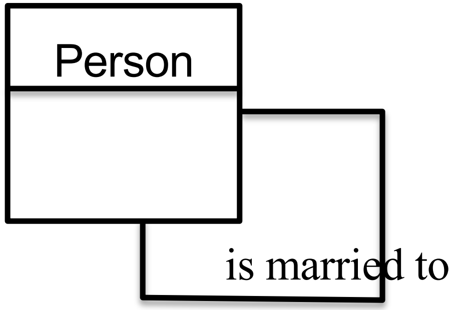


  - **Uni-directional** Association
    - Login Account has information of staff, BUT staff does not have information on Login Account.

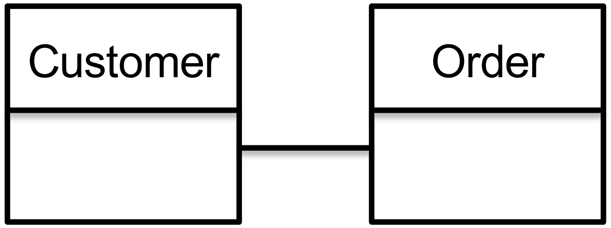


Associations between Classes

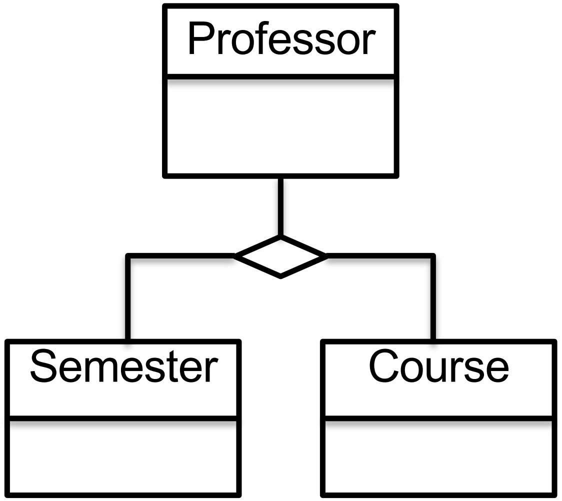
- Important to understand and know the nature of each association in term of the number of associations
  - Known as Multiplicity, eg one to one, one to many etc.
- The associations between types of things
  - Unary (recursive), binary, ternary, n-ary



Unary – relationship between two things of the same type

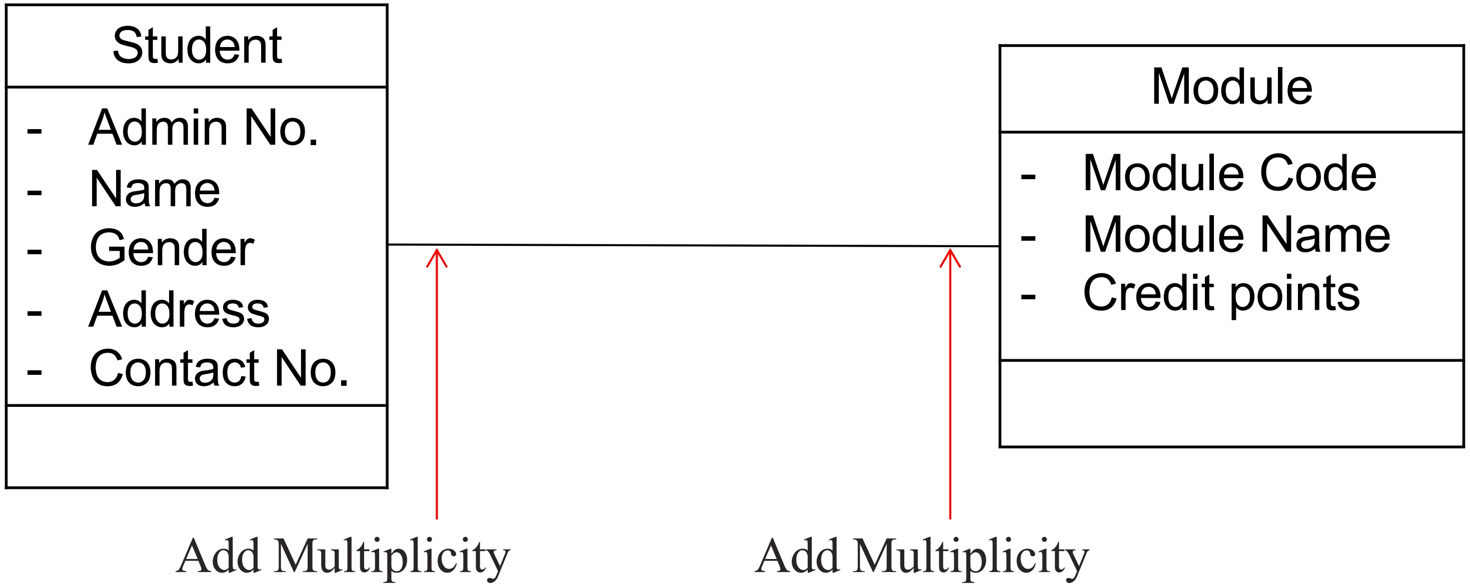


Binary– relationship between 2 different types of things



Ternary– relationship between 3 different types of things

Types of Multiplicity

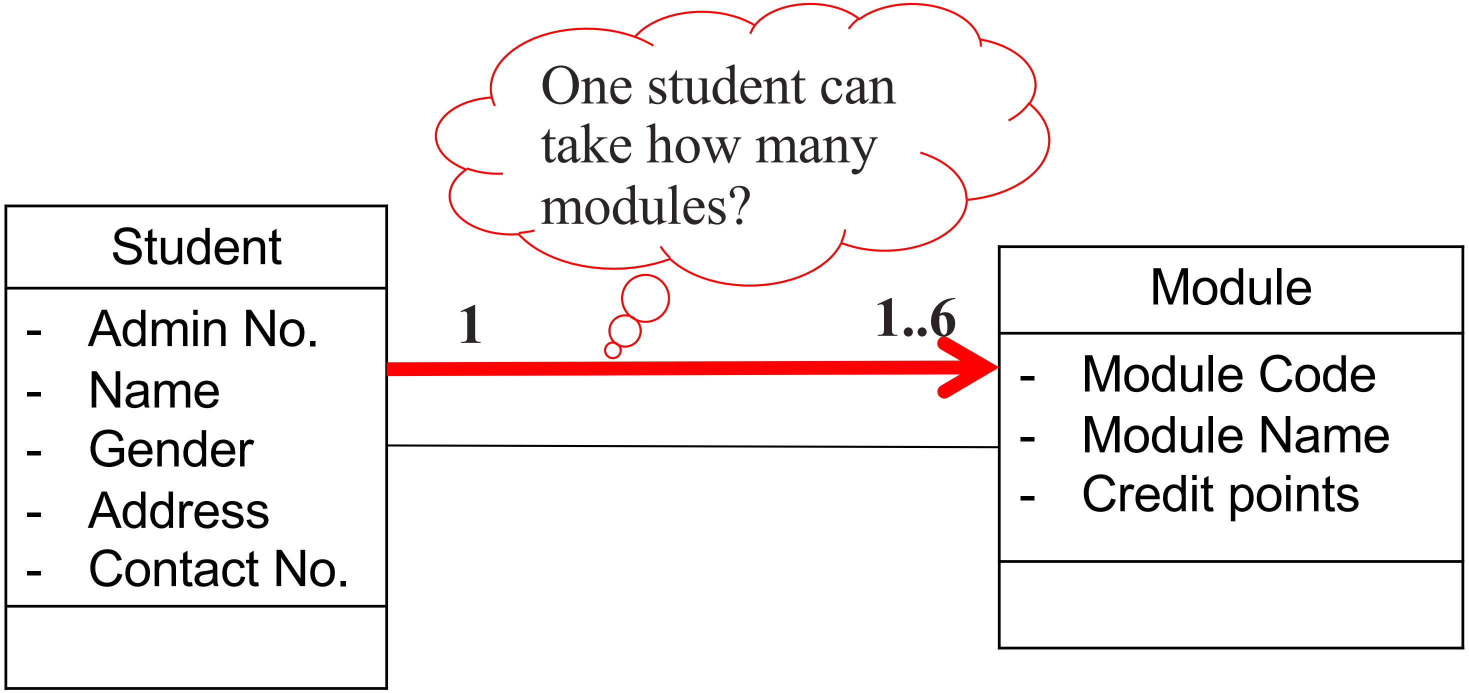


What are the Options?

1	One and only one
1..X	Min 1 and Max X
1..*	Min 1 and Max many
0..X	Min 0 and Max X
*	Min 0 and Max many

Types of Multiplicity

Can a student not take any modules? What is the min and max number of modules they must take?

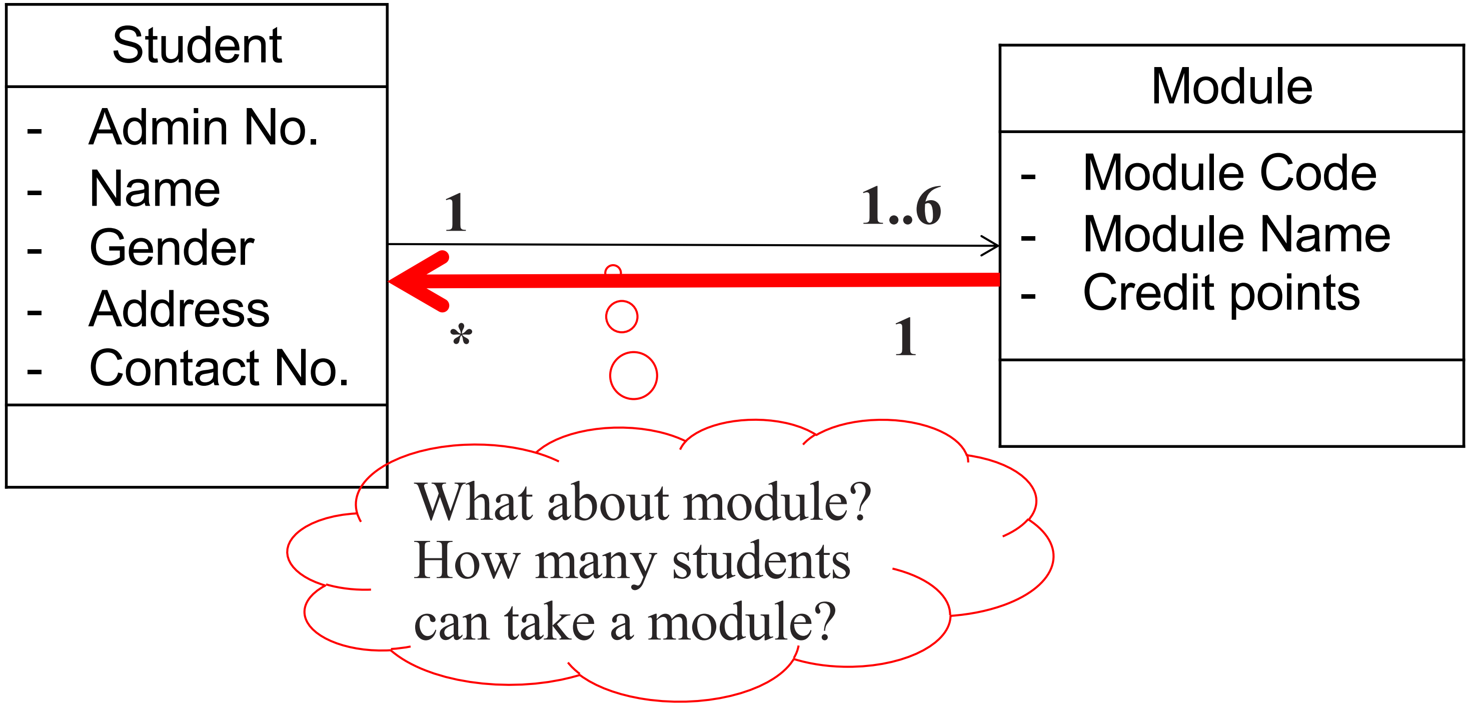


Step 1: Student to Module direction multiplicity

Conclusion: A student can take min 1 module and max 6 modules

Types of Multiplicity

Must a module be taken by a student? Can a module be taken by many students? What is the min and max number of students that can take the module?

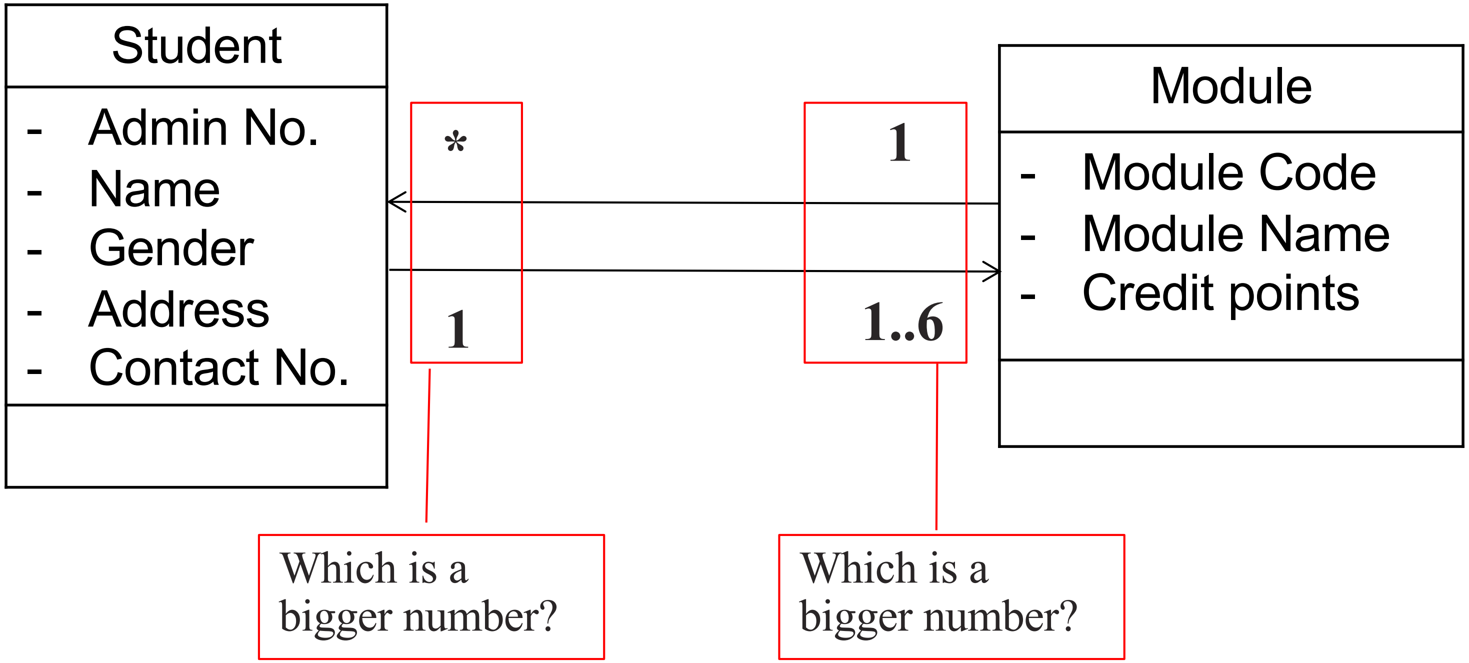


What about module? How many students can take a module?

Step 2: Module to Student direction multiplicity

Conclusion: A module can be taken by 0 or many students

Types of Multiplicity



Student

- Admin No.
- Name
- Gender
- Address
- Contact No.

Module

- Module Code
- Module Name
- Credit points

Which is a bigger number?

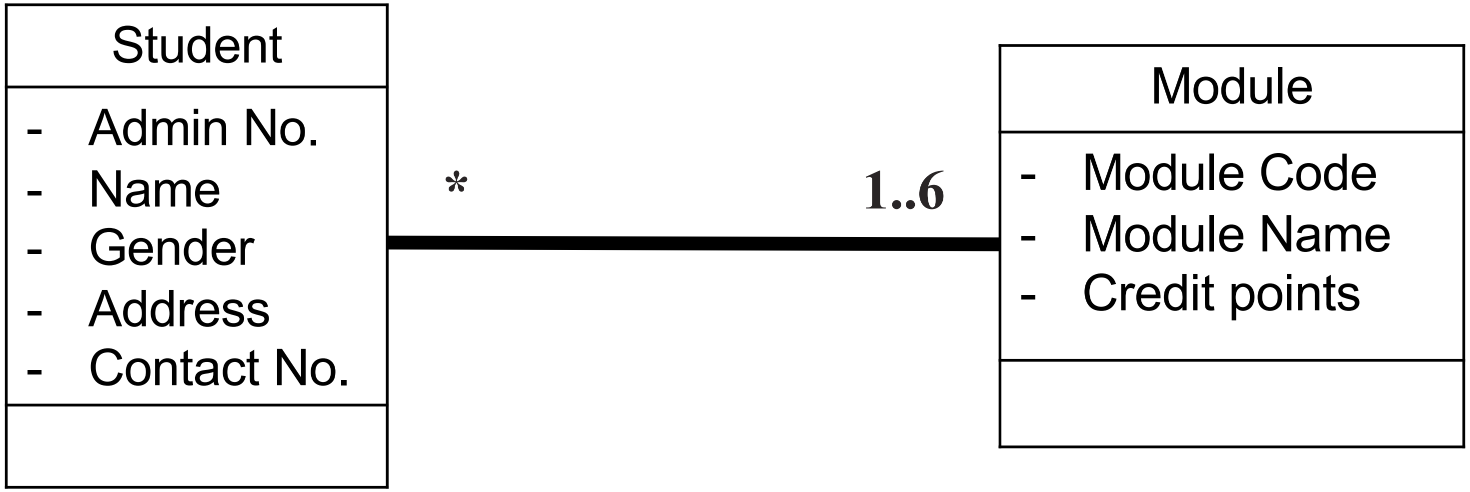
Which is a bigger number?

many is bigger than 1

1..6 is bigger than 1

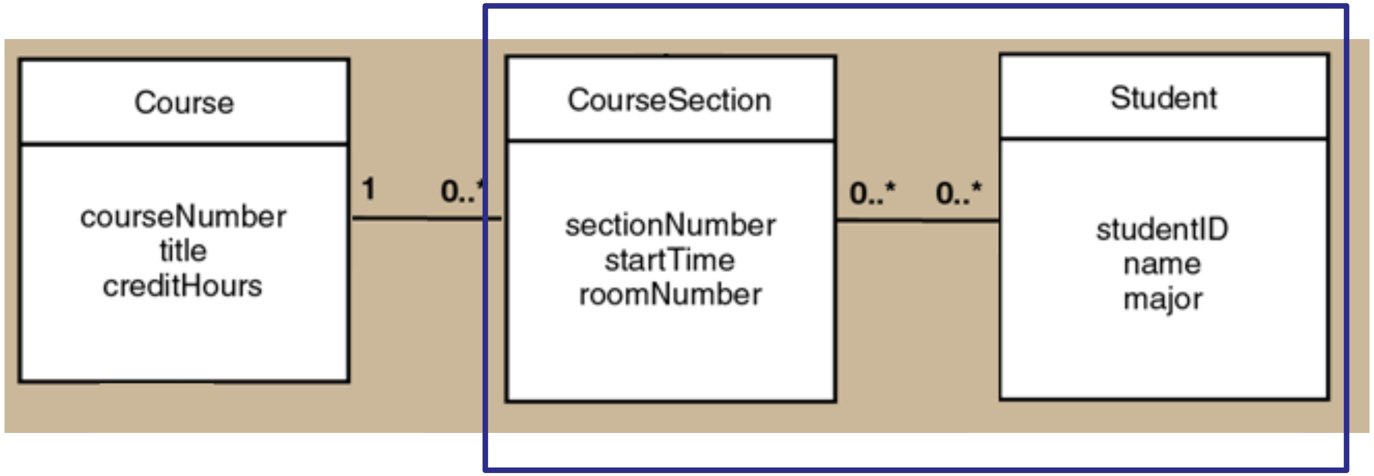
Step 3: For each class object take the bigger number

Types of Multiplicity



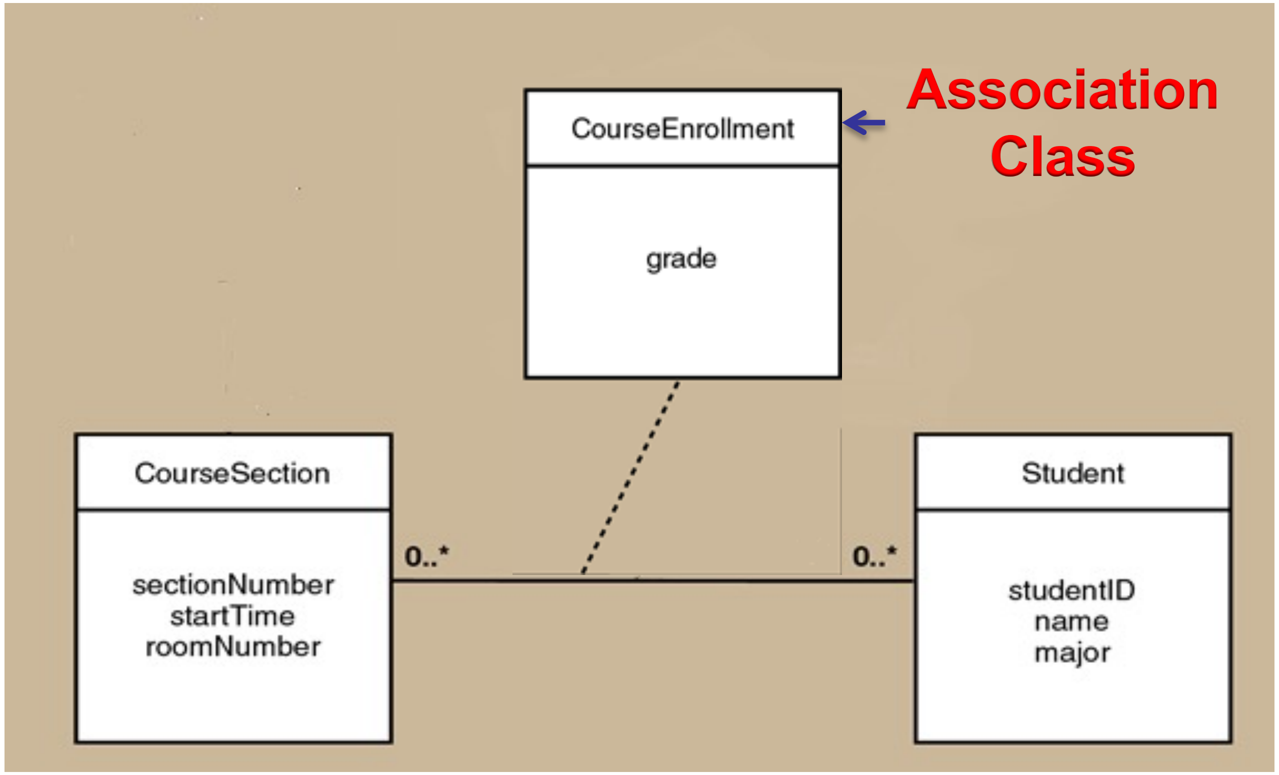
Since both direction has association, it forms a Bi-directional association.

Many-to-Many Association- Association Class



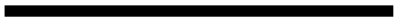
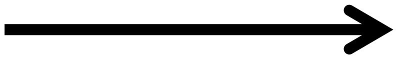
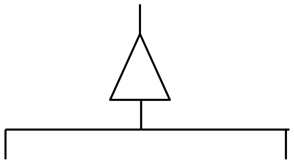
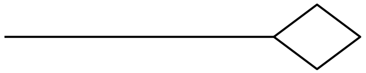
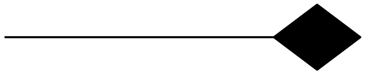
- What if we need to record the grade of Student and the CourseSection they take? Where can we put the attribute grade?
- Resolve the problem by adding a class to represent the association between Student and CourseSection

Association Class to store information between 2 classes with many-to-many associations

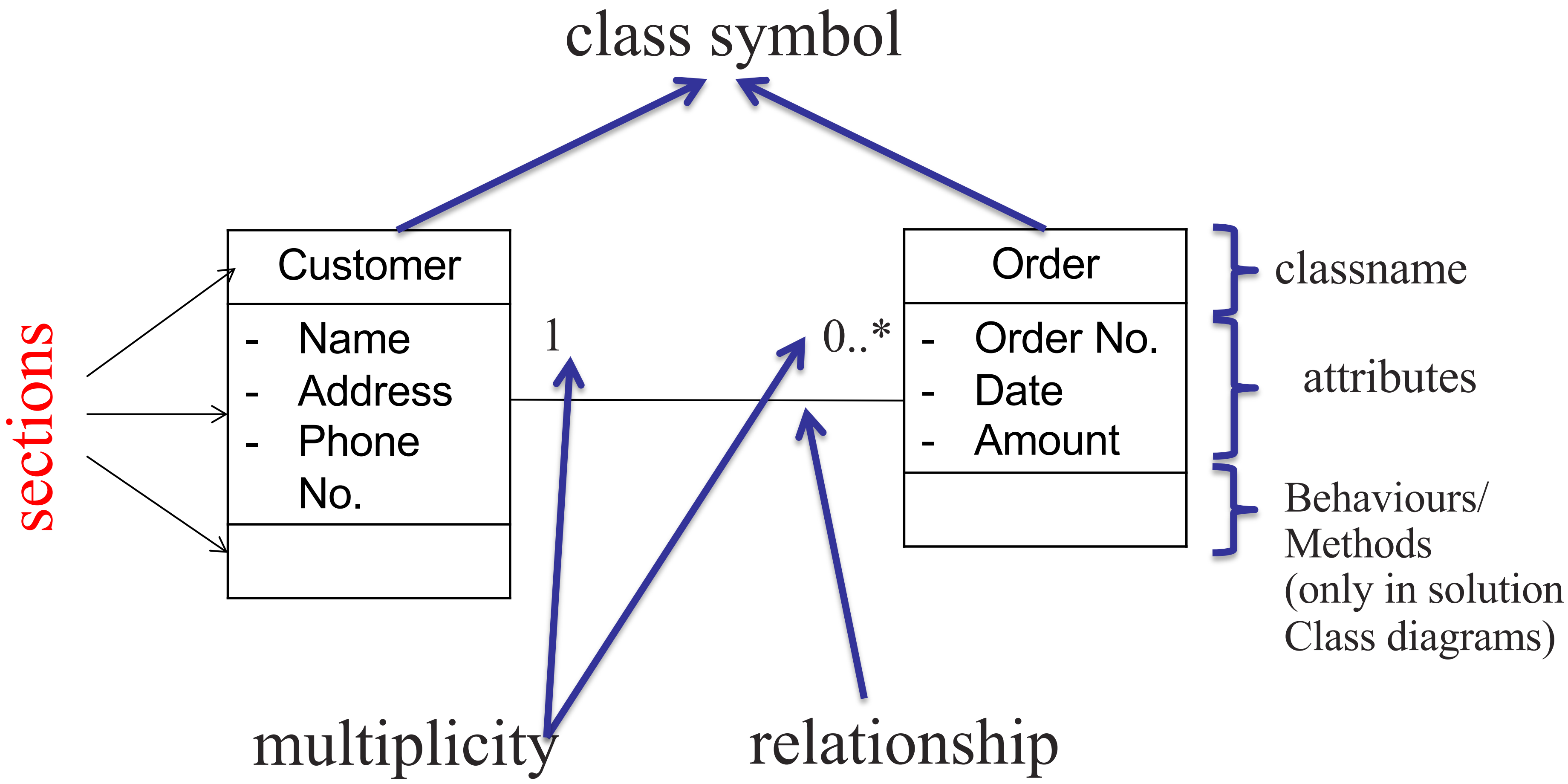


- Missing attribute is placed in the association class.
- An association class is connected to the many-to-may association with a dashed line.

Types of Relationship that need Multiplicity

Relationship	Notation	Multiplicity
1. Association		
i. Bi-directional		✓
ii. Uni-directional		✓
2. Generalisation/Specialisation		X
i. a.k.a Inheritance		
ii. “Is a kind of”		
3. Aggregation		✓
i. “Consist of”		
4. Composition		✓
i. “made up of”		

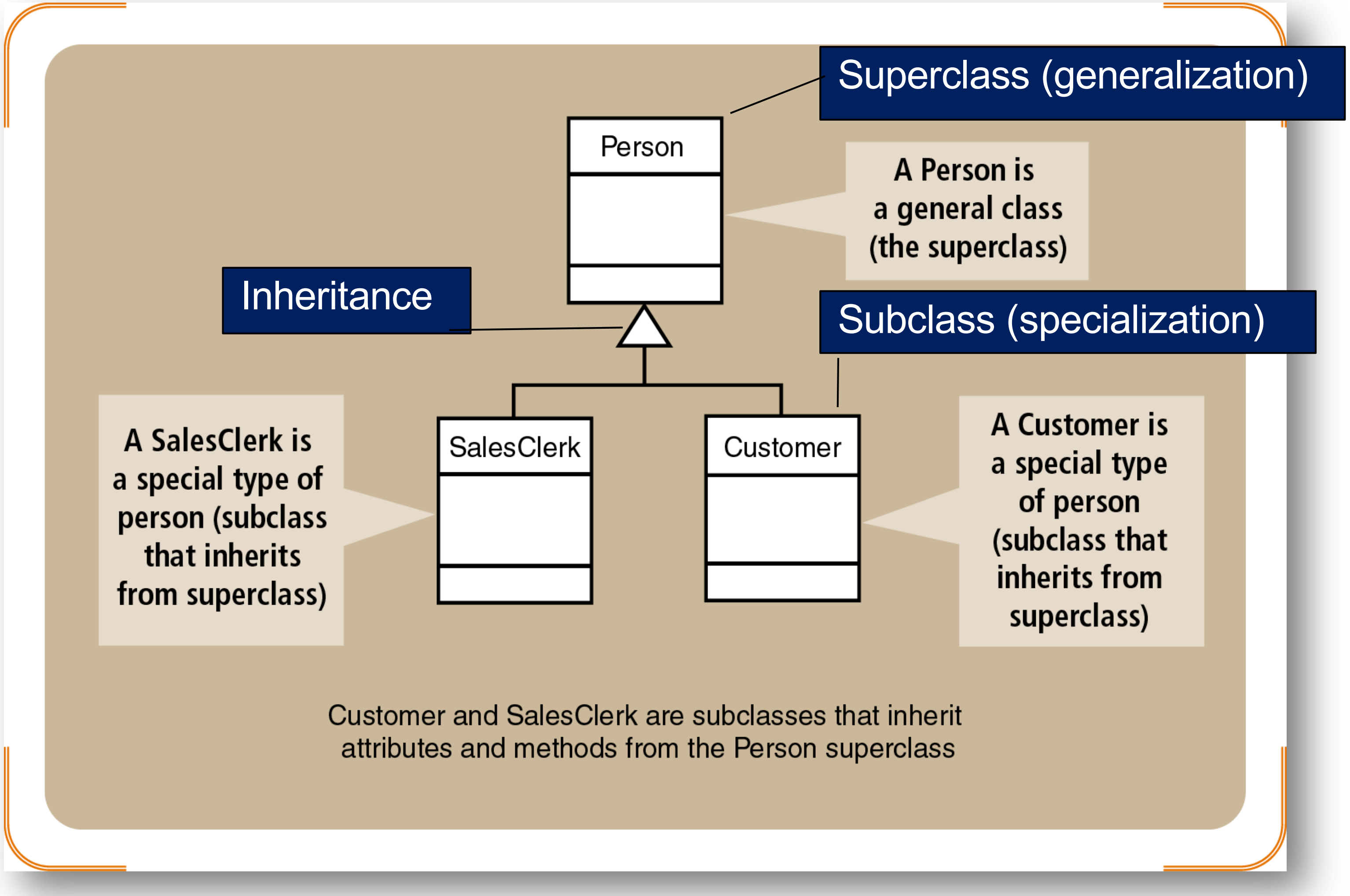
(UML) Class Diagram Notation



Generalization/specialization

- Also known as Inheritance in Object Oriented concepts
  - Rank things from more general to the more special
- **Classification**: means of defining classes of things
  - Superclass: generalization of a class
  - Subclass: specialization of a class

A Generalization/Specialization Hierarchy Notation

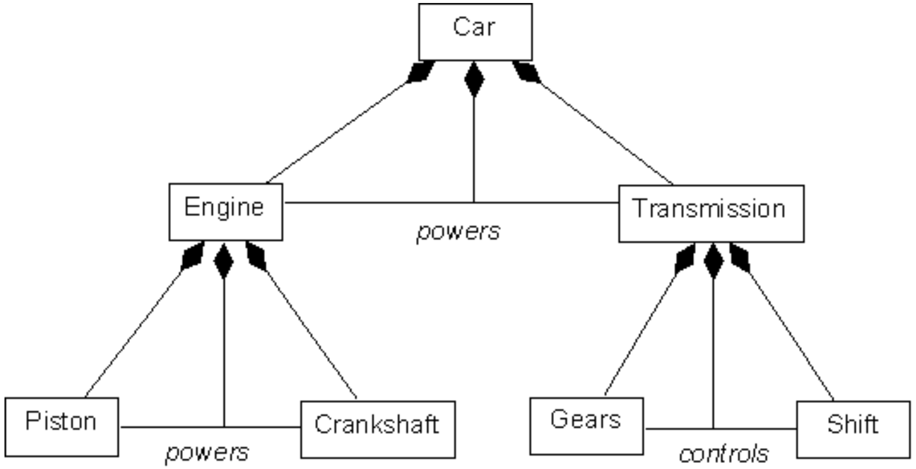

Superclass (generalization)
Person
A Person is a general class (the superclass)
Inheritance
Subclass (specialization)
SalesClerk
Customer
A SalesClerk is a special type of person (subclass that inherits from superclass)
A Customer is a special type of person (subclass that inherits from superclass)

Customer and SalesClerk are subclasses that inherit attributes and methods from the Person superclass

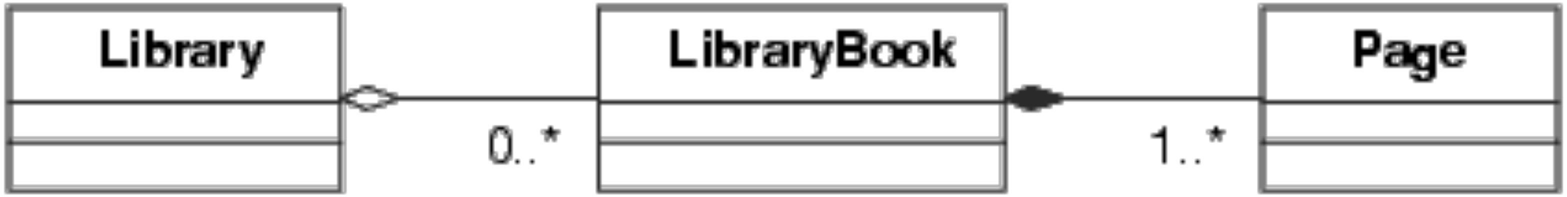
Whole-part Hierarchy Notation

- Capture relationship between objects and its components
  - “The whole is equal to the sum of the parts”
- **Two types** of whole-part hierarchies
  - Aggregation: association with independent parts
    - Example: keyboard is part of computer system
  - Composition: association with dependent part
    - Example: CRT and monitor
- Multiplicity also applies to whole-part relationships

Whole-part (Aggregation) Associations Between a Computer and Its Parts



Whole-part (Composition) Associations Between a Car and Its Parts



Whole-part (Composition) Associations Library Book and Its Parts

Steps to Constructing a Domain Class Diagram

Step 1: Identify nouns from resources. Mainly Use Case Diagram and Use Case Description.

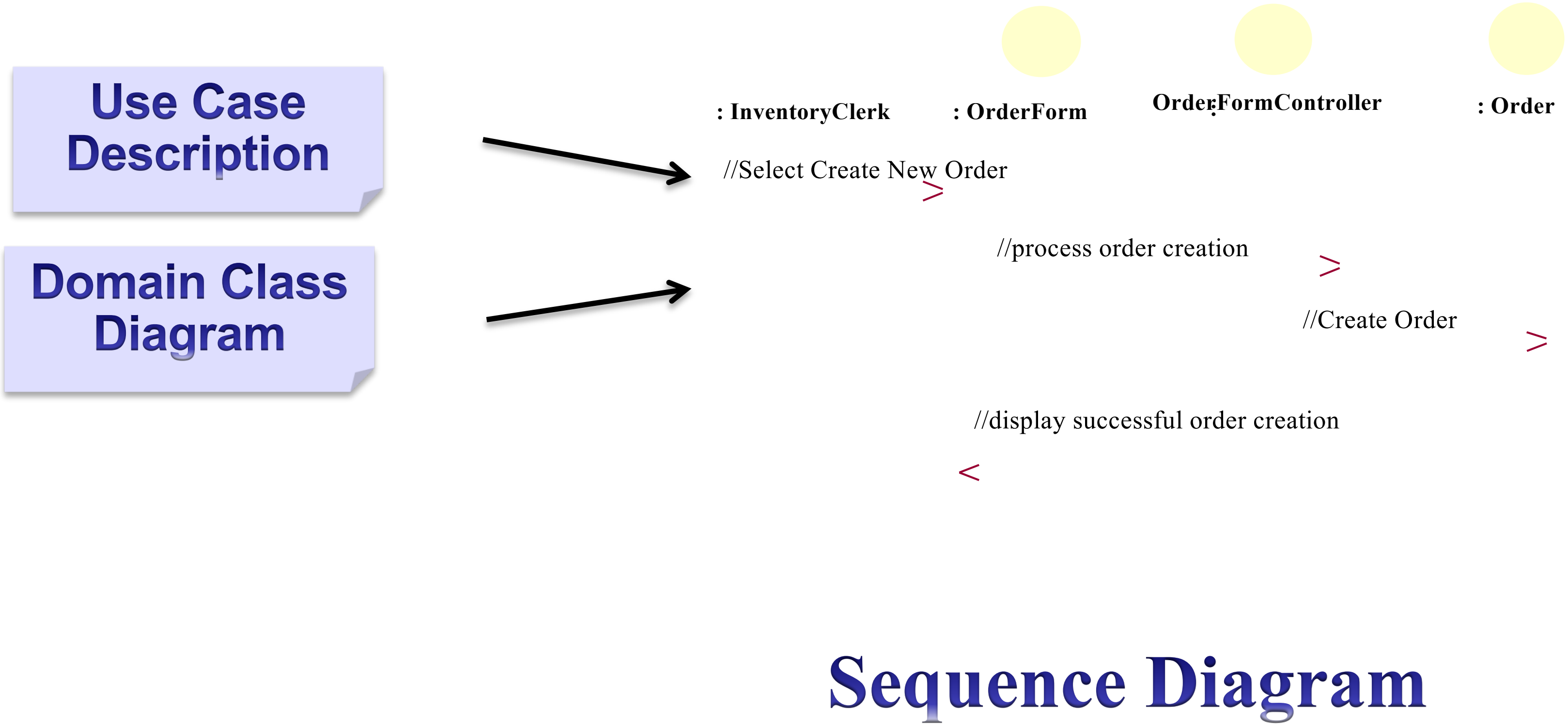
Step 2: Filter out un-important nouns and for important nouns categorize them as class objects or attributes

Step 3: Further identify required attributes for each class object

Step 4: Identify relationships between class objects

Purpose of Domain Class Diagram

Use case description and domain class diagram are input resources for the Sequence Diagram



Use Case Description

Domain Class Diagram

: InventoryClerk

: OrderForm

:OrderFormController

: Order

//Select Create New Order

//process order creation

//Create Order

//display successful order creation

Sequence Diagram

Summary

- Problem domain model and Domain class diagram
- Purpose of Domain Class Diagram/Model
- Domain Class Diagram/Model reflects attributes and associations between class objects.
- Associations among classes includes relationships such as Multiplicity, Generalization/Specialization, Whole-part hierarchies
- Association Class.
- Steps to constructing a domain class diagram/model

Cohort Exercise

What are the advantages and disadvantages of software modelling? What other types of modelling are you aware of and do the advantages and disadvantages apply to this type of modelling that you had identified?

Case Study Task:

As a team, derive a domain class diagram for the case study scenario.