

IT2755
Software Engineering

Domain Modelling and UML Class
Diagrams

Learning Outcomes

To be able to:

1. explain what a problem domain is
2. identity domain classes in a problem domain
3. identify relationships between domain classes
4. specify multiplicities between domain classes
5. derive and represent a domain model using UML class diagram
6. use association class in an association relationship between 2 classes.



Problem Domain and Domain Model

- What is a Problem Domain?
 - an area of expertise or application that needs to be examined to solve a problem.
- Problem Domain Models
 - represent the real world business concepts (things) and their relationships in a clear and unambiguous way
 - Contains relevant things that are essential part of the system that users deal with when they do their work and Eg: products, orders, invoices and customers
 - These things make up the data/information about which the system needs to track
 - Identifying and understanding these things in problem domain is a key initial step in defining requirements
- Eg Library systems – books, borrowers, librarians, loan records.

(Problem) Domain Model and Class Diagrams

- **Class diagrams** are 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 (software) objects within the system that interact to either process or track information. (Java or C# **classes in your codings**)

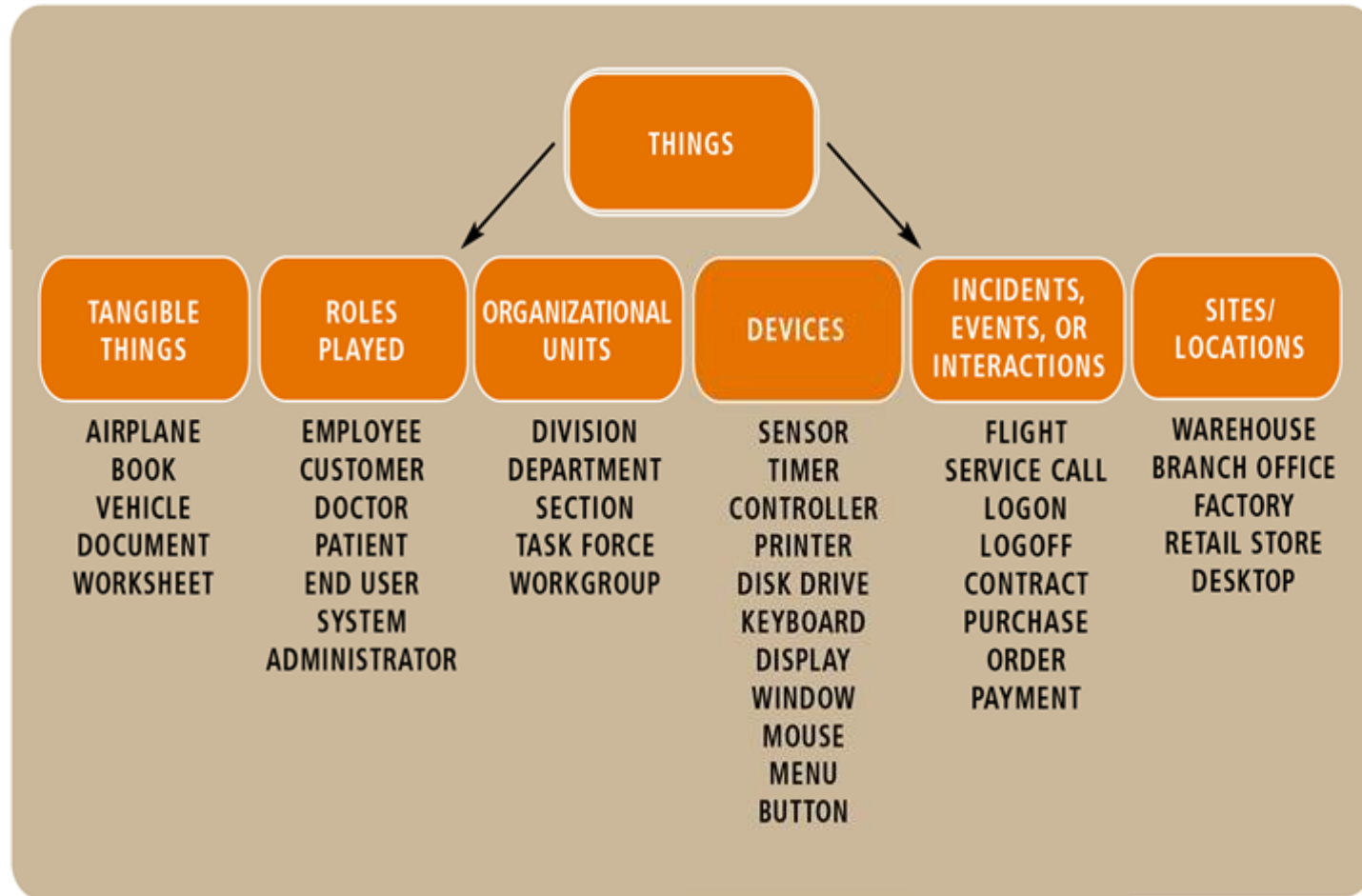
How to identify Domain Class Objects?

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

E.g. Customer places an order.

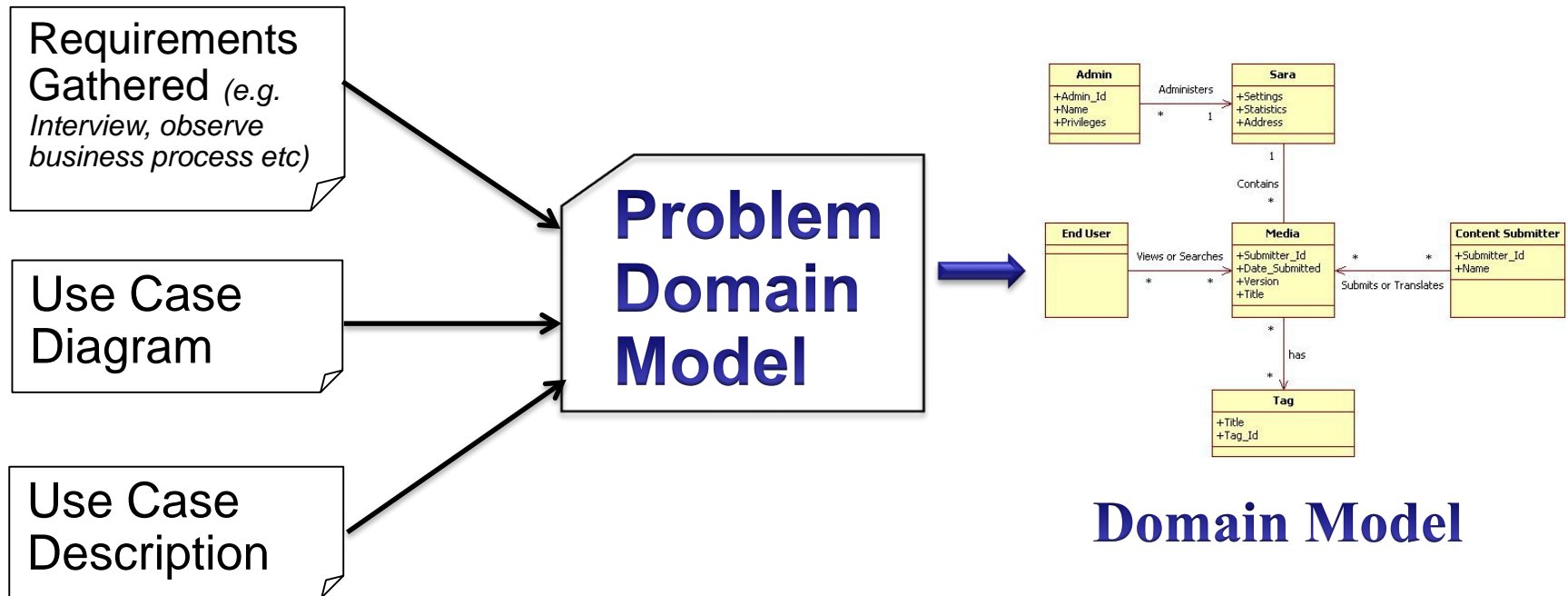
Order is an intangible object arising from customer interaction with item in inventory

Types of Things



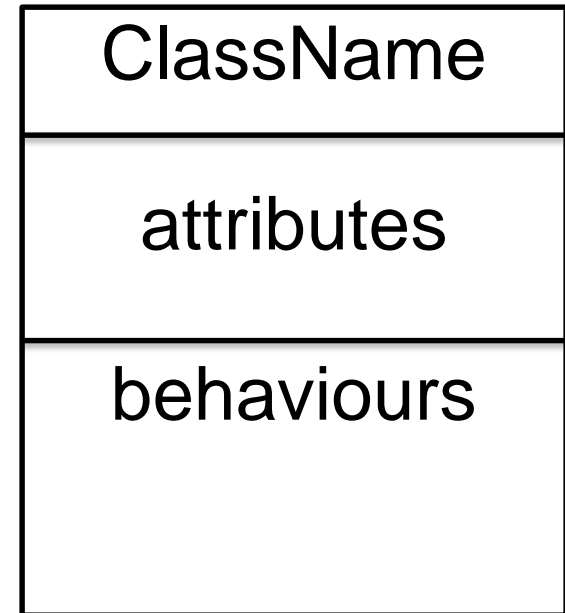
Source of potential things

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



(UML) Class Diagram Notation

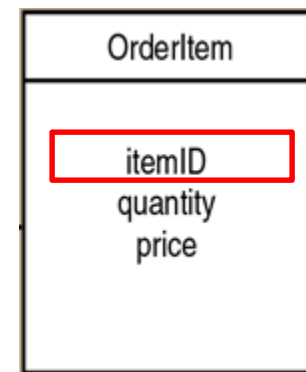
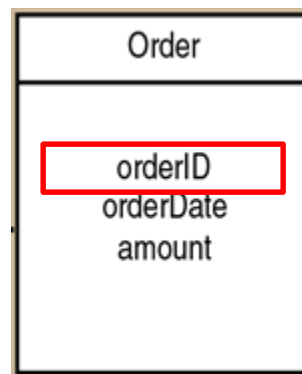
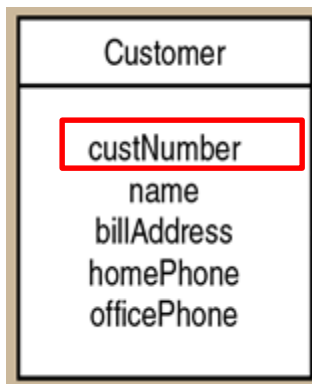
- Class diagram:
 - General class symbol: rectangle with three sections
 - Sections convey class name (normally singular and upper case), attributes, and behaviors
 - Methods (behaviors) **not shown in domain model class diagram**





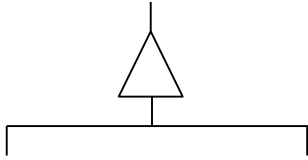
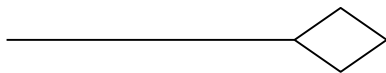
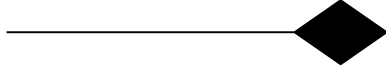
UML Class Diagram

Attributes of classes

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



Types of Class Relationships

<u>Relationship</u>	<u>Notation</u>
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"	

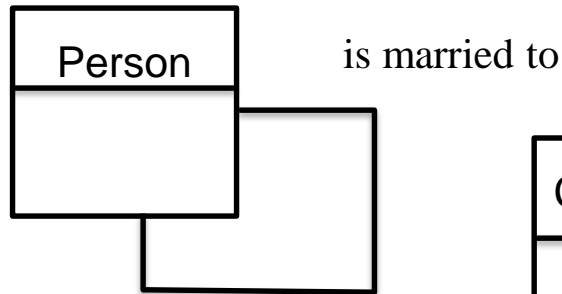
Associations among Classes

- An **association** is a naturally occurring **relationship** among specific classes
 - 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
 - Staff has information about Login information, BUT Login does not have information on Staff.

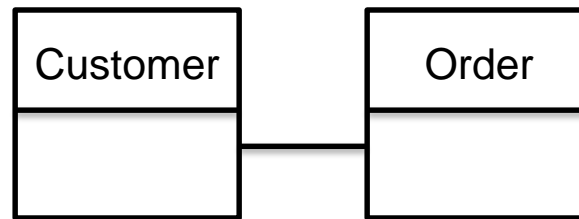


Associations among Classes

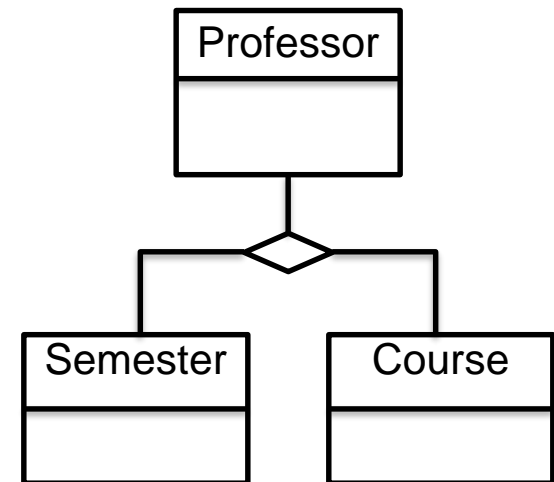
- The associations between classes
 - Unary (recursive), binary, ternary, n-ary



Unary – relationship between two classes of the same type



Binary – relationship between 2 different types of classes



Ternary – relationship between 3 different types of classes

Multiplicity 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.
- Types of Multiplicity:

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

Multiplicity

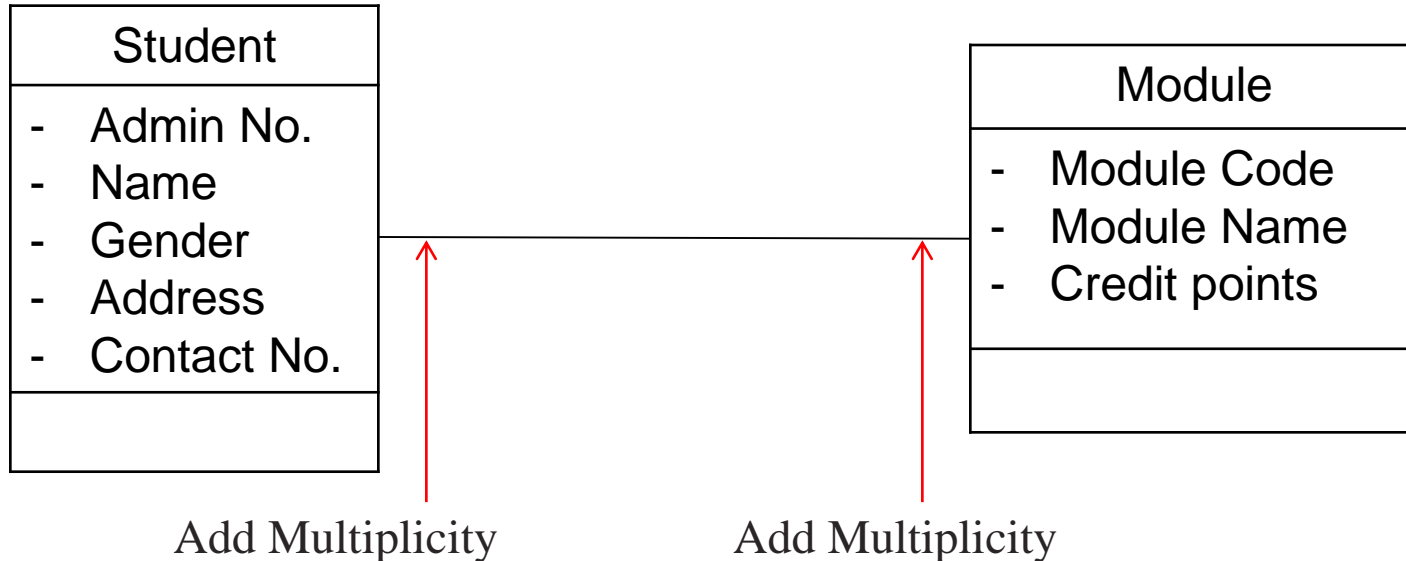
Student
<ul style="list-style-type: none">- Admin No.- Name- Gender- Address- Contact No.

Module
<ul style="list-style-type: none">- Module Code- Module Name- Credit points

How to handle (specify) the following details on a domain class diagram:

- 1 A student is allowed to enrol into min 1 module.
2. A student is allowed to enrol no more than 6 modules.
3. A module can have no student registered
4. A module can have at most 24 students registered

Types of Multiplicity

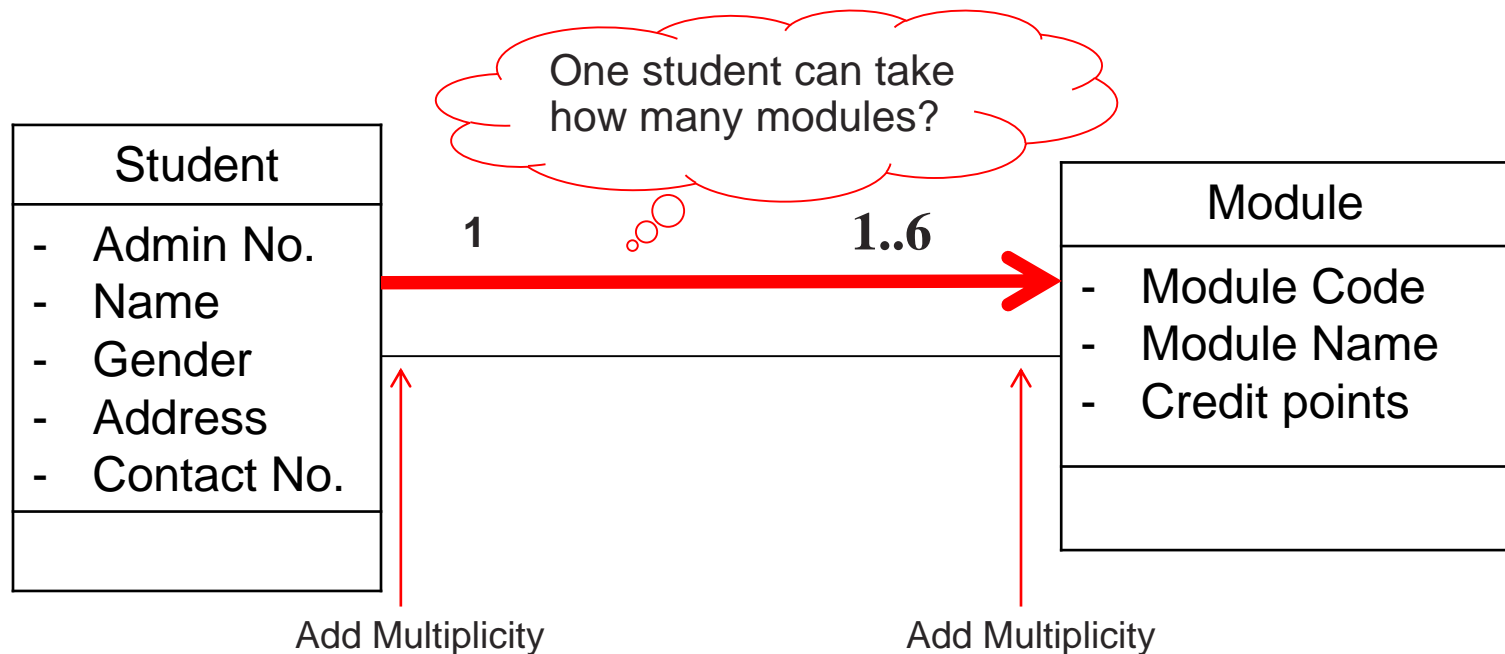


What are the Options?

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?

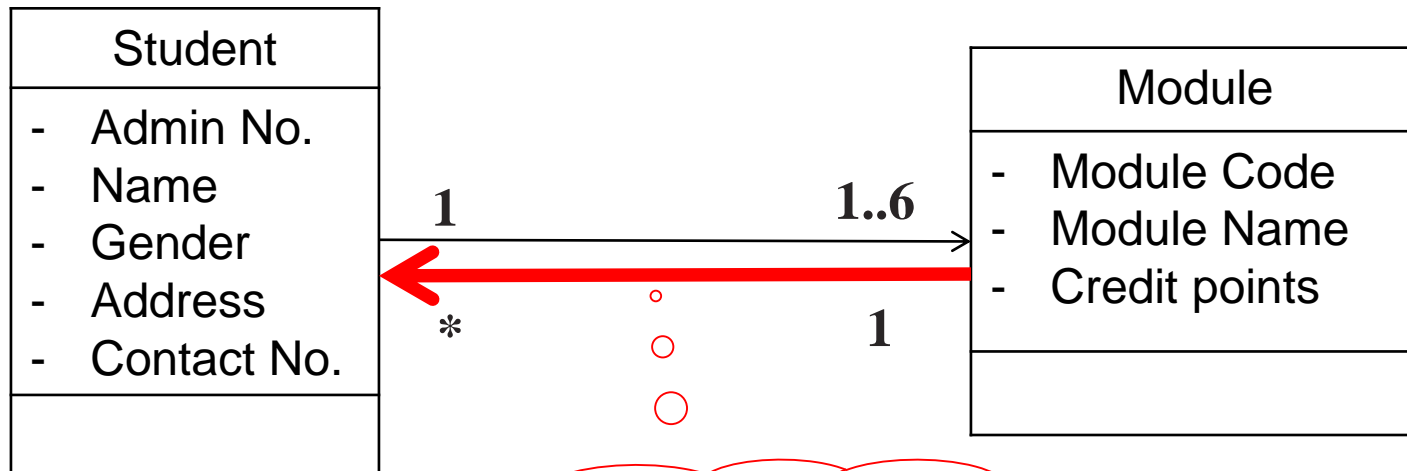


Step 1: Student to Module direction multiplicity

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?

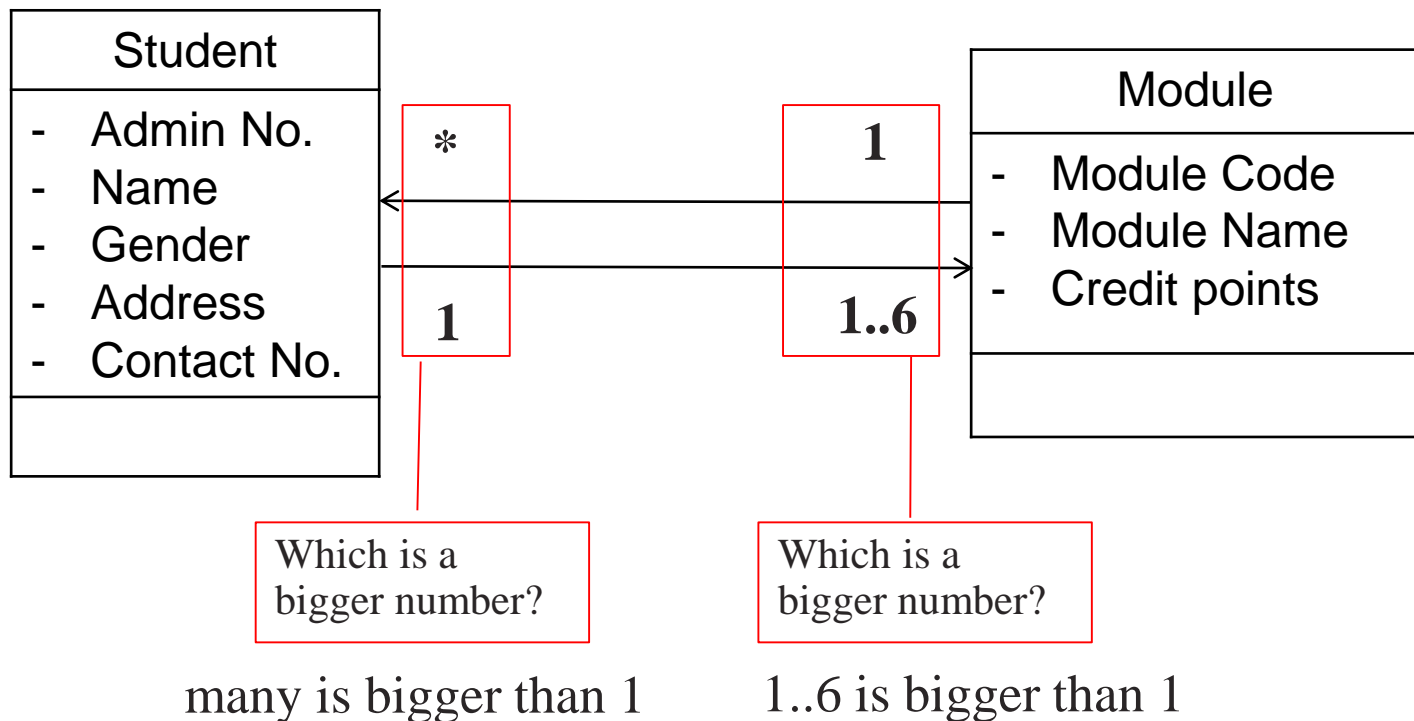


What about module? How many students can take a module?

Step 2: Module to Student direction multiplicity

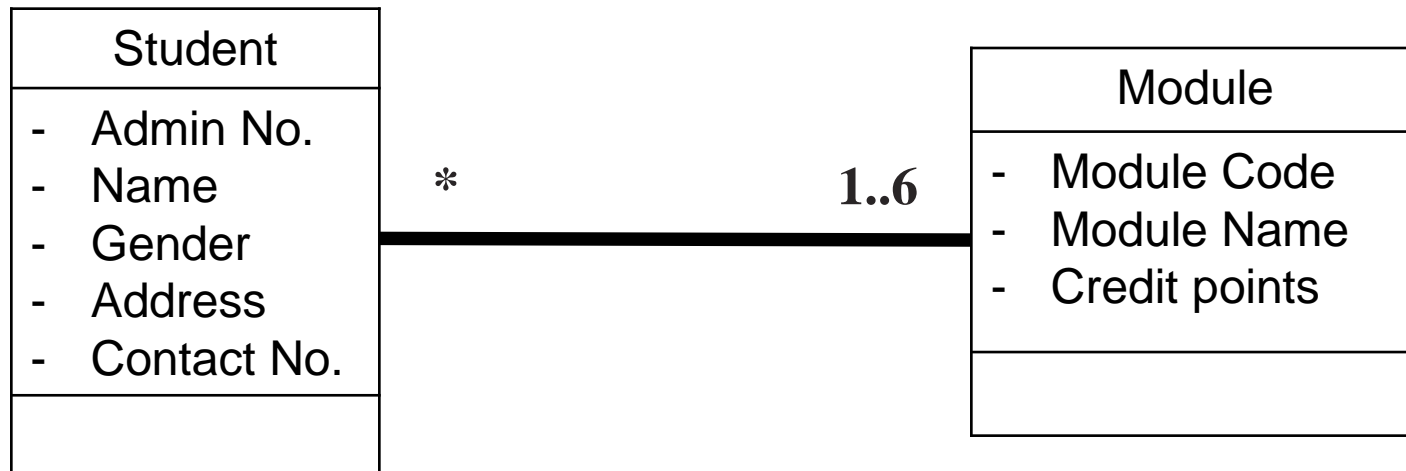
Conclusion: A module can be taken by 0 or many students

Types of Multiplicity



Step 3: For each class object take the bigger number

Types of Multiplicity



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

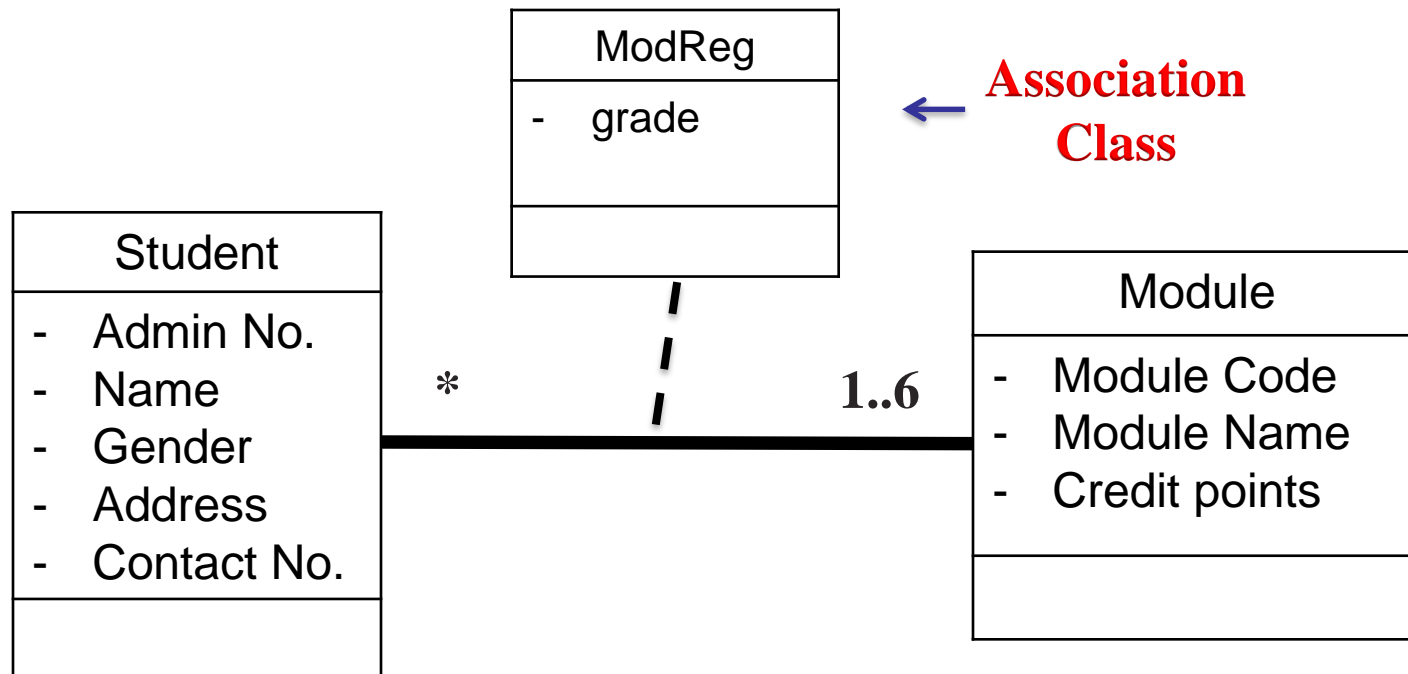
The multiplicity for this association is Many – to – Many

Many-to-Many Association- Association Class





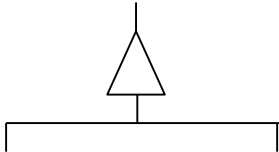
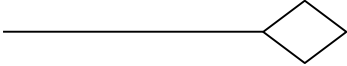
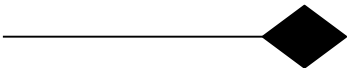
- What if we need to record the **grade** of Student and the Module they take? Where can we put the attribute grade?
- Resolve the problem by adding a class to represent the association between Student and Module

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

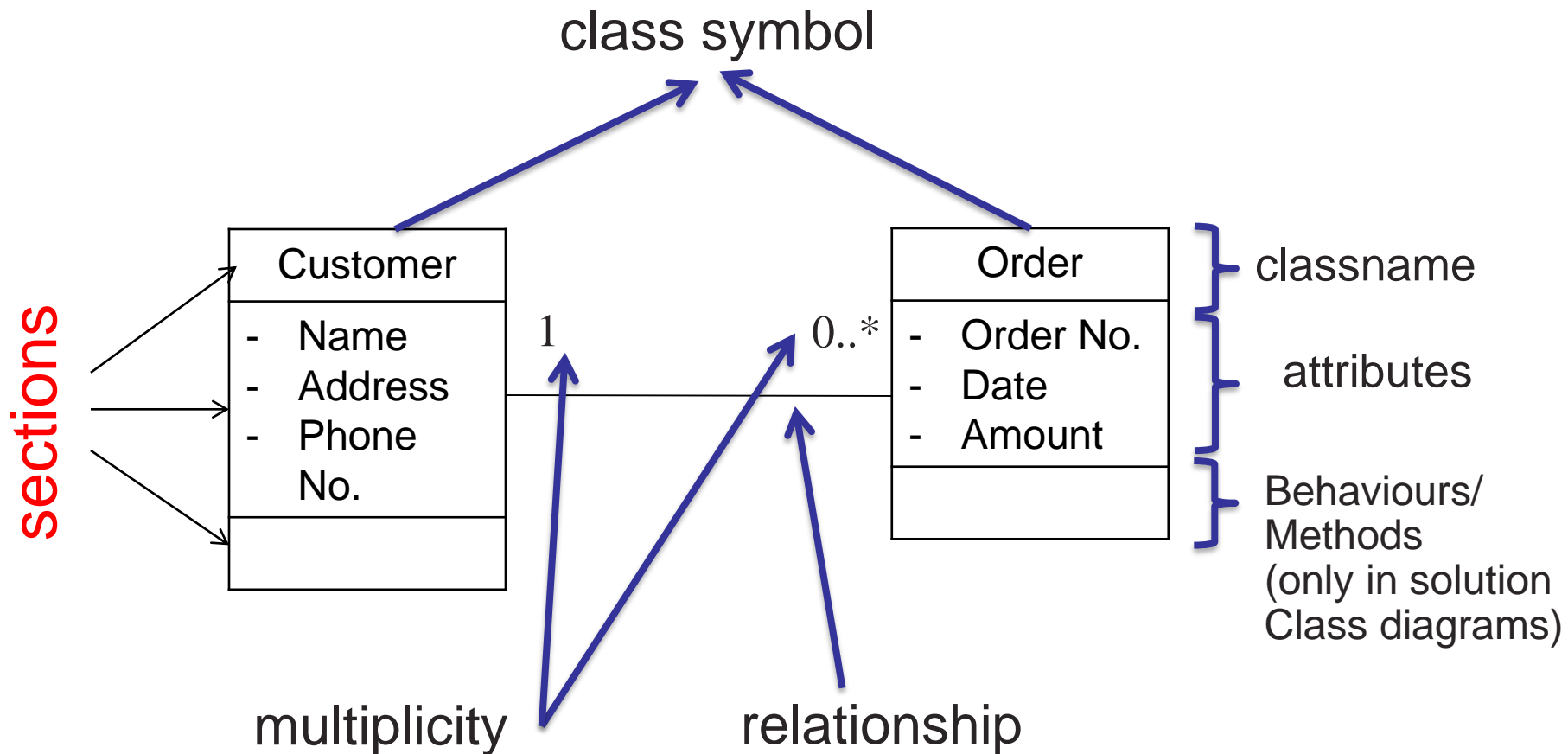


- Missing attribute is placed in the association class.
- An association class is connected to the many-to-many association with a dashed line.

Types of Relationship that need Multiplicity

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

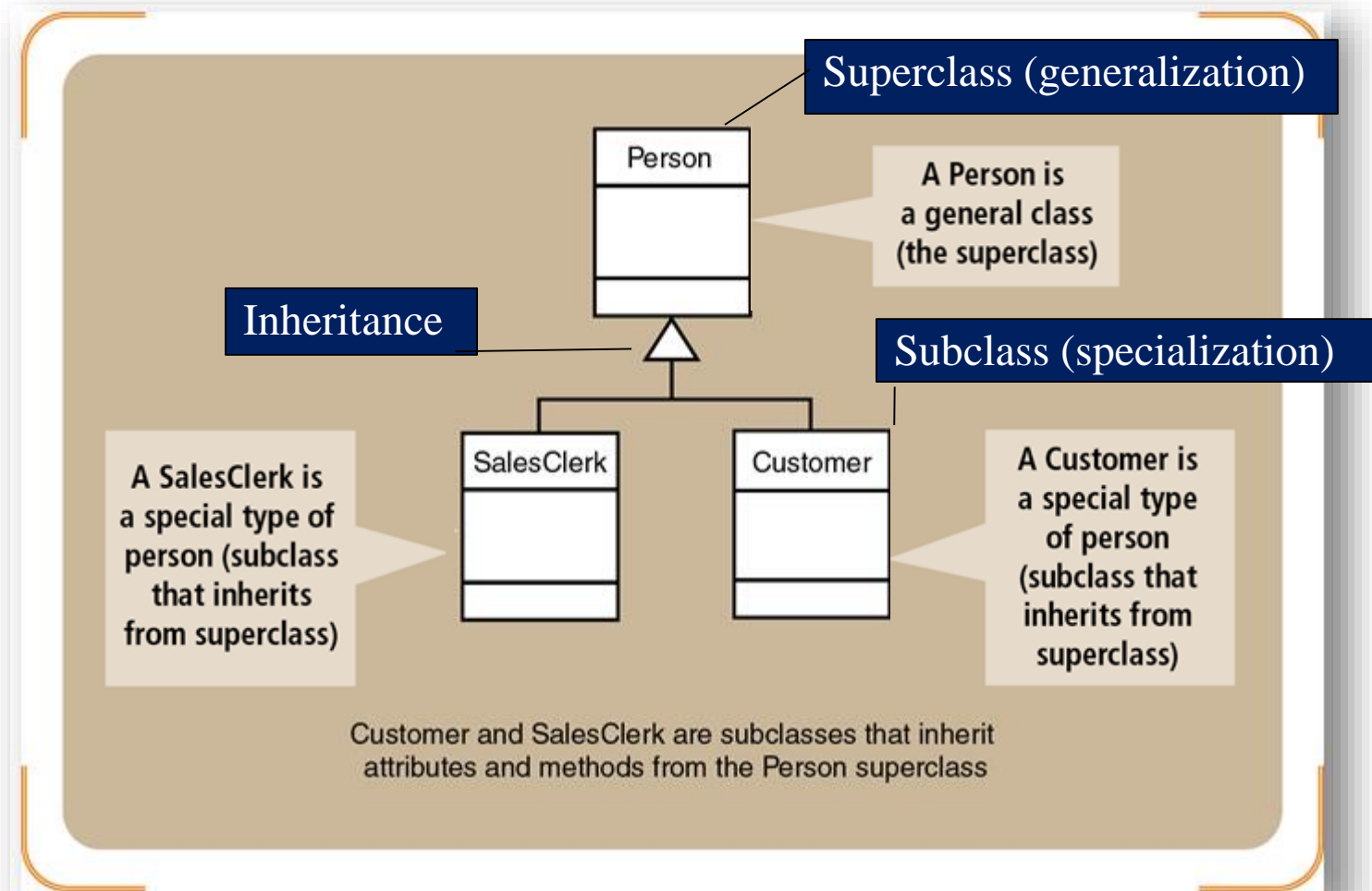
(UML) Class Diagram Notation



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

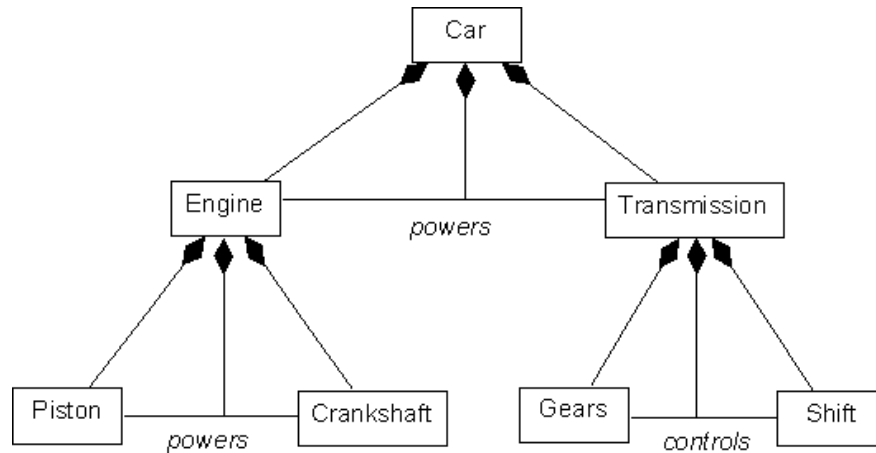
A Generalization/Specialization Hierarchy Notation



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 relationships
 - **Aggregation**: association with **independent** parts
 - Example: keyboard is part of computer system
 - Both keyboard and computer can exist independently
 - **Composition**: association with **dependent** part
 - Example: CRT and monitor
- CRT & monitor must exist together. One delete the other must also be deleted.

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

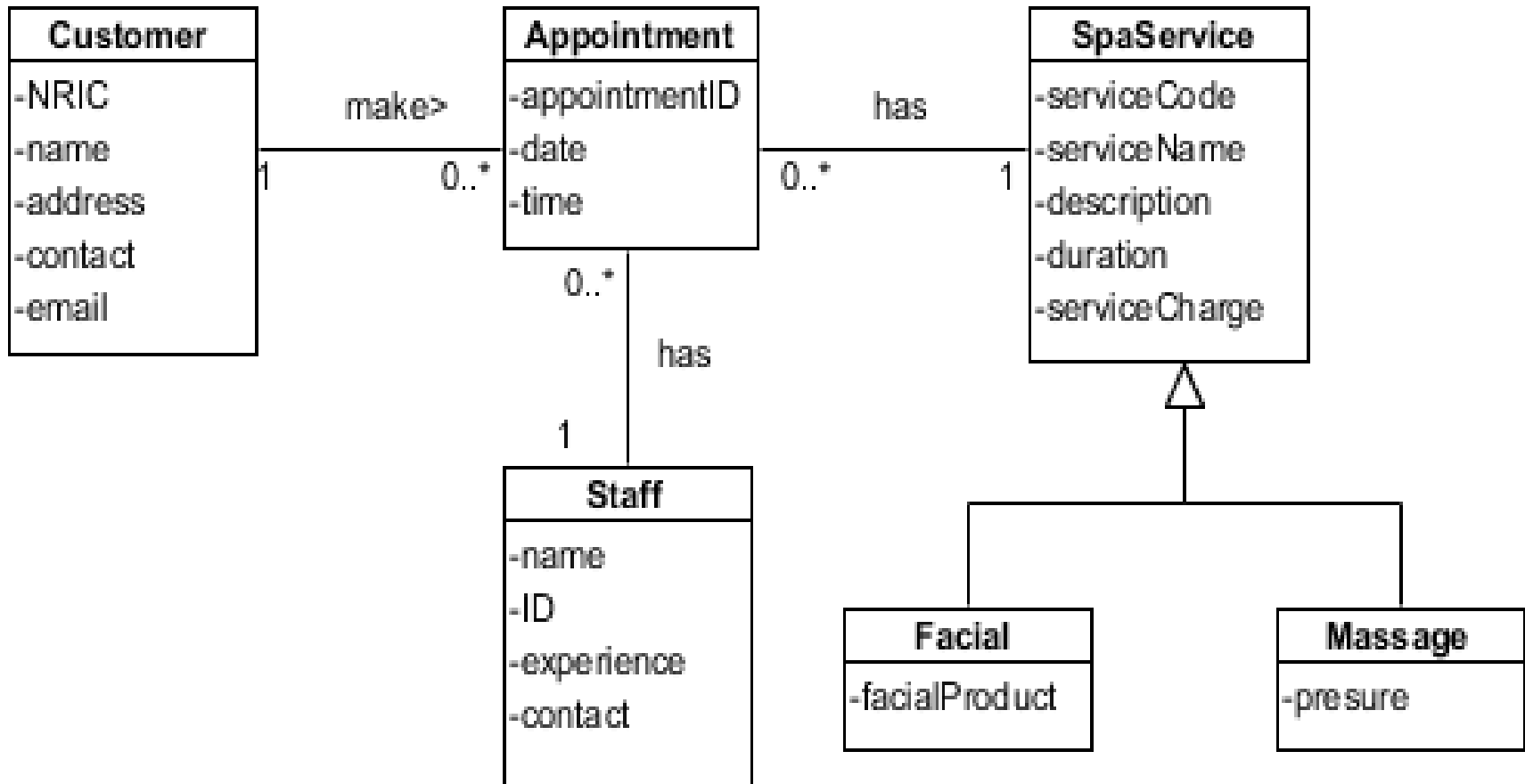


Whole-part (Composition) Associations Between a Car and Its Parts



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

Domain Model Example

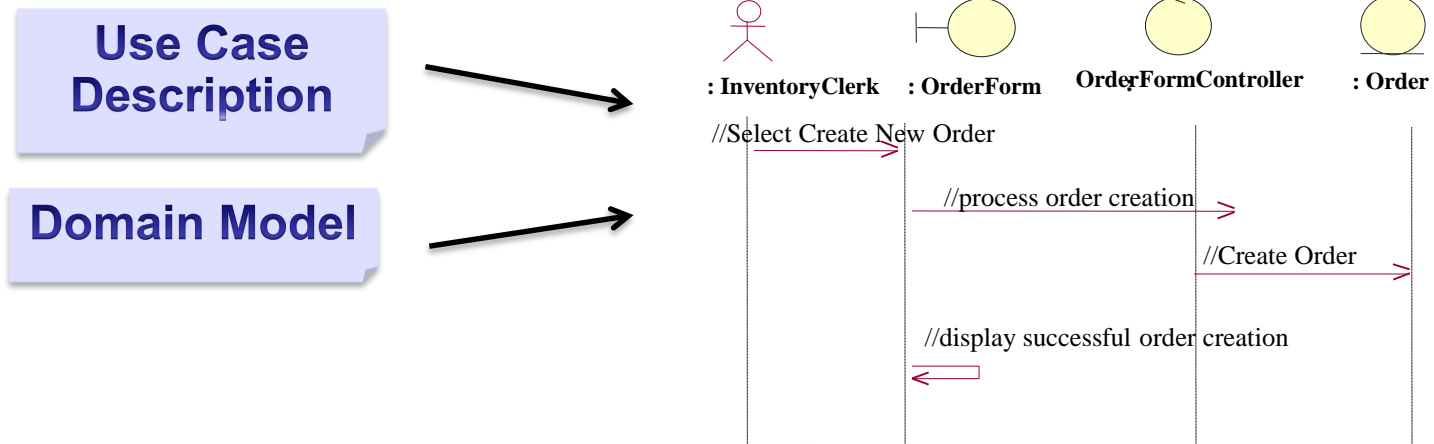


Steps to Constructing a Domain Model

1. Identify nouns from resources. Mainly Use Case Diagram and Use Case Description.
2. Filter out un-important nouns and for important nouns, categorize these as either classes or attributes of these classes
3. Further identify required attributes for each class
4. Identify relationships between these classes
5. Identify the multiplicities between these classes

Purpose of Domain Model

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



Sequence Diagram

- Domain model will be used in the design discipline and further developed into:
 - an E-R model and
 - Design model

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
- Steps to constructing a domain class diagram/model
- Association Class.