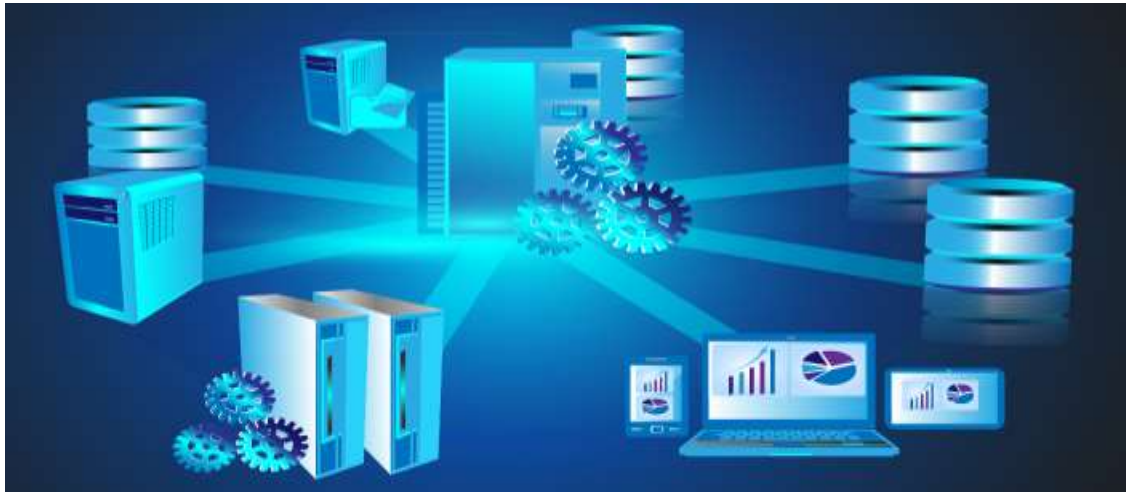
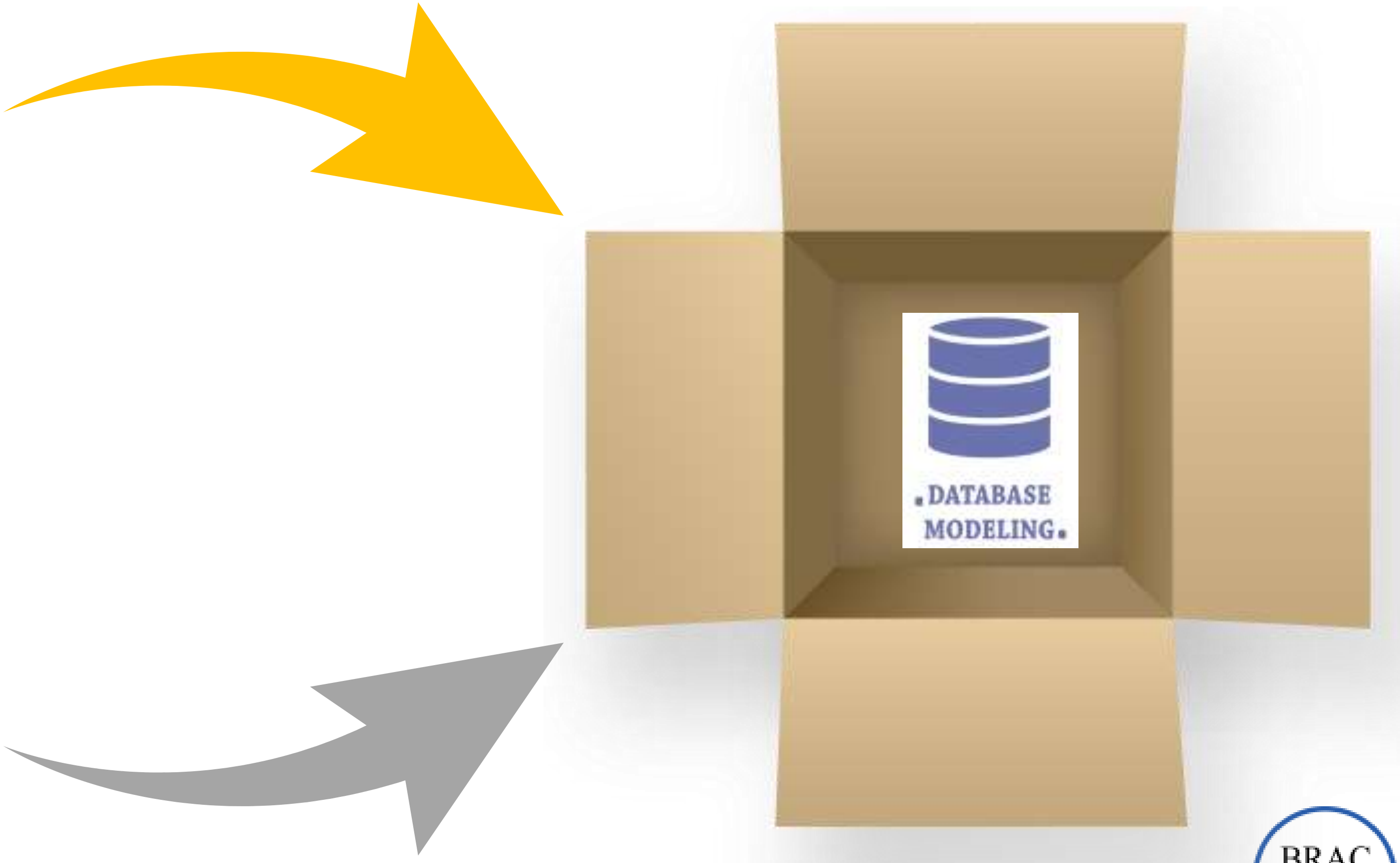
# CSE716:Advanced Database

Lecture by A.M.Esfar-E-Alam



# TOPICS

- Data Modeling
- Object Oriented Data Modeling



# Data Modeling

- *Data modelling* is the first step in the process of database design. This step is sometimes considered to be a high-level and abstract design phase, also referred to as conceptual design. The aim of this phase is to describe:
- The data contained in the database (e.g., entities: students, lecturers, courses, subjects)
- The relationships between data items (e.g., students are supervised by lecturers; lecturers teach courses)
- The constraints on data (e.g., student number has exactly eight digits; a subject has four or six units of credit only)

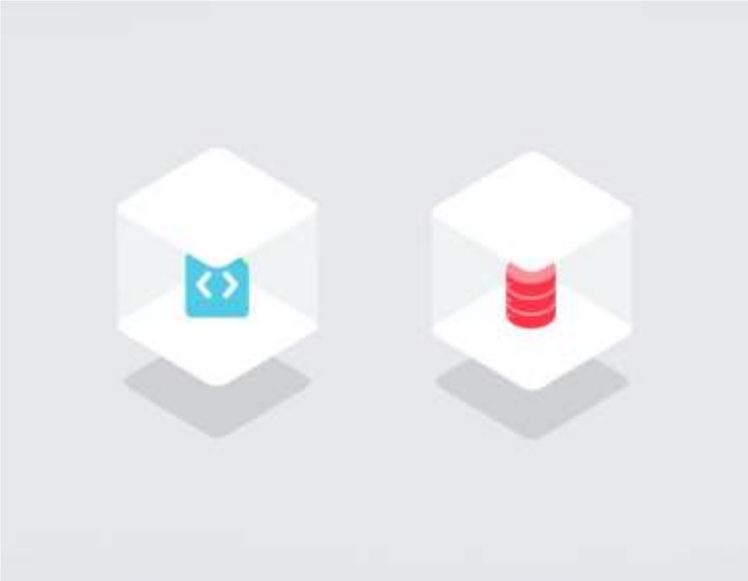
- In the database design phases, data are represented using a certain data model.
- The *data model* is a collection of concepts or notations for describing data, data relationships, data semantics and data constraints.
- Most data models also include a set of basic operations for manipulating data in the database.

# Types of Data Models

- **There are mainly three different types of data models:**
- **Conceptual:** This Data Model defines **WHAT** the system contains. This model is typically created by Business stakeholders and Data Architects. The purpose is to organize, scope and define business concepts and rules.
- **Logical:** Defines **HOW** the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures.
- **Physical**: This Data Model describes **HOW** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

# Object-Oriented data Modeling

- Here we will get introduced to object-oriented data modeling.
- We will describe the main concepts and techniques involved in object-oriented modeling, including objects and classes; encapsulation of attributes and operations; association, generalization, and aggregation relationships; cardinalities and other types of constraints; polymorphism; and inheritance.
- We will see show how we can develop class diagrams, using the UML notation, to provide a conceptual view of the system being modeled



# Representing objects and classes

- In the object-oriented approach, we model the world in objects.
- Before applying the approach to a real-world problem, therefore, we need to understand what an object and some related concepts really are.

➢ **Class**

- o An entity type that has a well-defined role in the application domain about which the organization wishes to maintain state, behavior, and identity

➢ **Object**

- o An instance of a class that encapsulates data and behavior.

➢ **state**

- o An object’s properties (attributes and relationships) and the values those properties have.

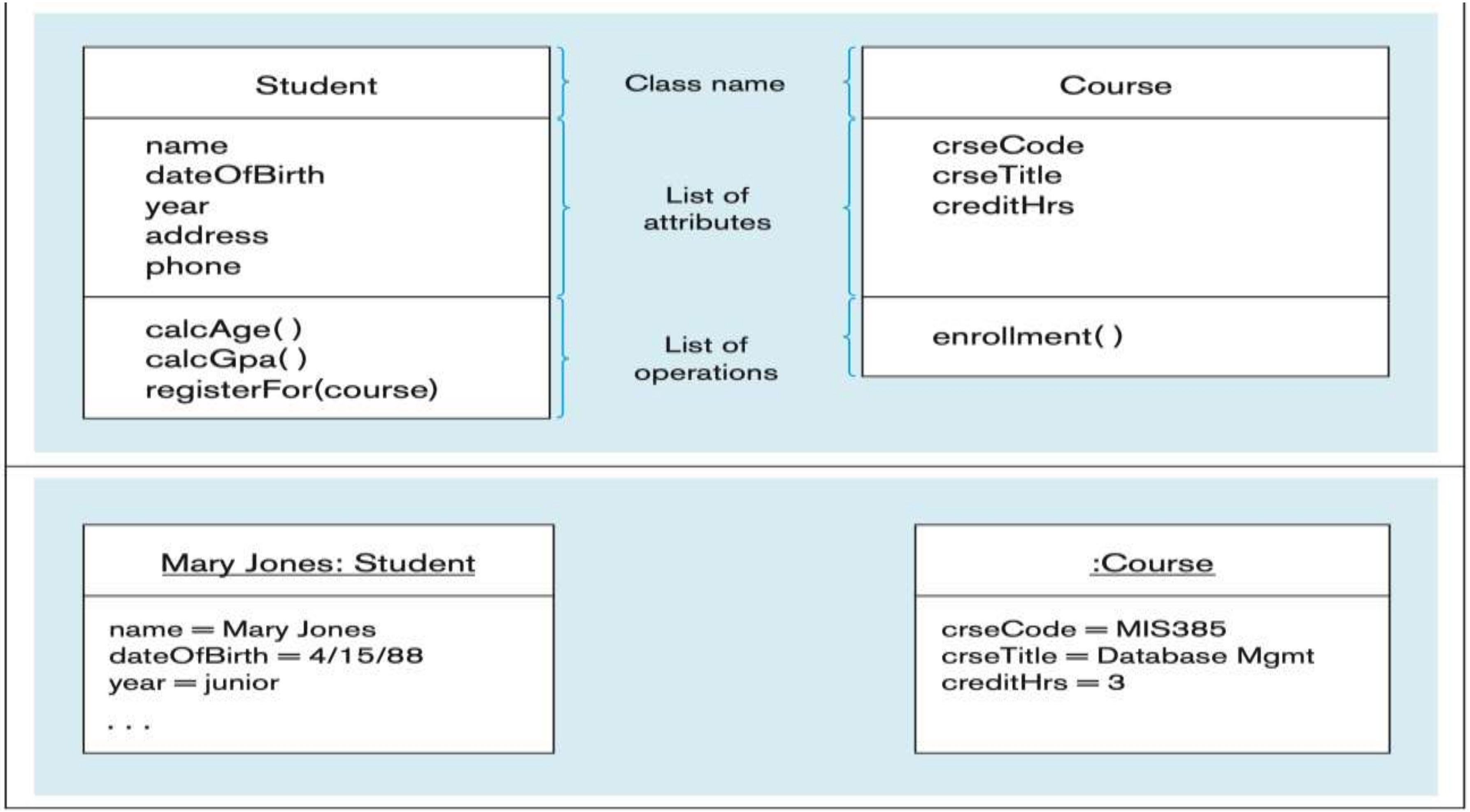
➢ **behavior**

- o The way in which an object acts and reacts.

# Class/Object Diagram

- We can depict the classes graphically in a class diagram
- A **class diagram** shows the static structure of an object-oriented model: the classes, their internal structure, and the relationships in which they participate.
- In UML, a class is represented by a rectangle with three compartments separated by horizontal lines.
- The class name appears in the top compartment, the list of attributes in the middle compartment, and
- the list of operations in the bottom compartment of a box.
- The figure shows two classes, Student and Course, along with their attributes and operations.
- There is an object diagram drawn below to class diagram
- An **object diagram**, also known as an *instance diagram*, is a graph of instances that are compatible with a given class diagram.

# Class Diagram-Object Diagram



Student

name
dateOfBirth
year
address
phone

calcAge( )
calcGpa( )
registerFor(course)

Class name

List of attributes

List of operations

Course

crseCode
crseTitle
creditHrs

enrollment( )

Mary Jones: Student

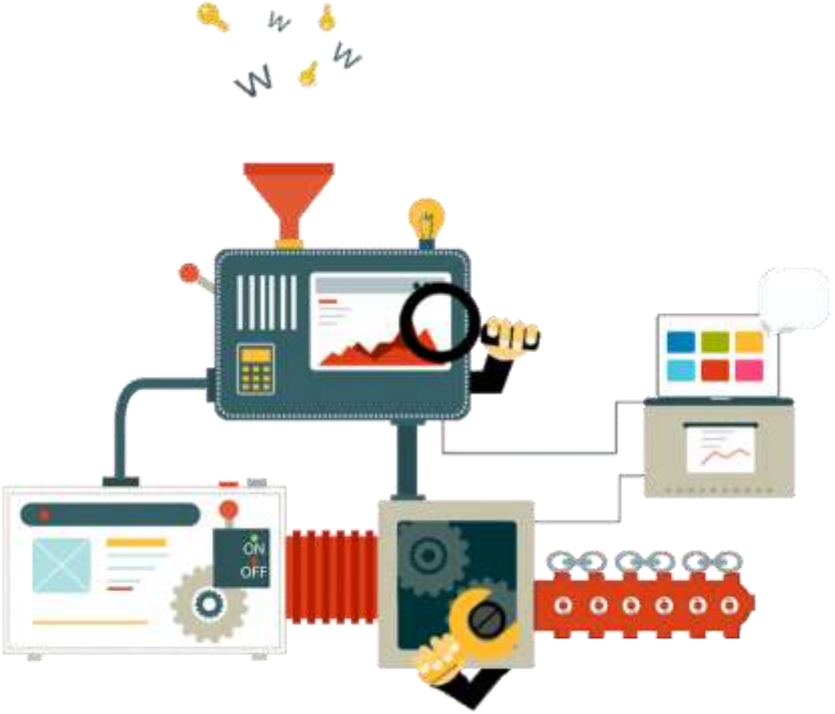
name = Mary Jones
dateOfBirth = 4/15/88
year = junior
. . .

:Course

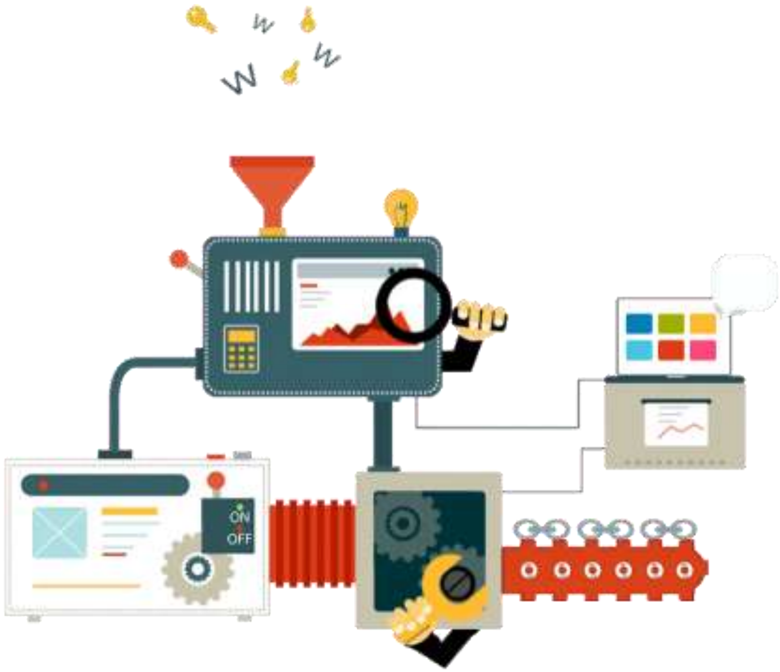
crseCode = MIS385
crseTitle = Database Mgmt
creditHrs = 3

# Operation and Types of Operation

- An **operation**, such as calcGpa in Student, is a function or a service that is provided by all the instances of a class.
- Typically, other objects can access or manipulate the information stored in an object only through such operations
- Operations can be classified into four types, depending on the kind of service requested by clients: (1) constructor, (2) query, (3) update, and (4) class-scope



- **constructor operation** :
    - An operation that creates a new instance of a class.
- **Query operation**
    - An operation that accesses the state of an object but does not alter the state.
- **Update operation**
    - An operation that alters the state of an object.
- **class-scope operation**
    - An operation that applies to a class rather than to an object instance.



BRAC UNIVERSITY
Inspiring Excellence

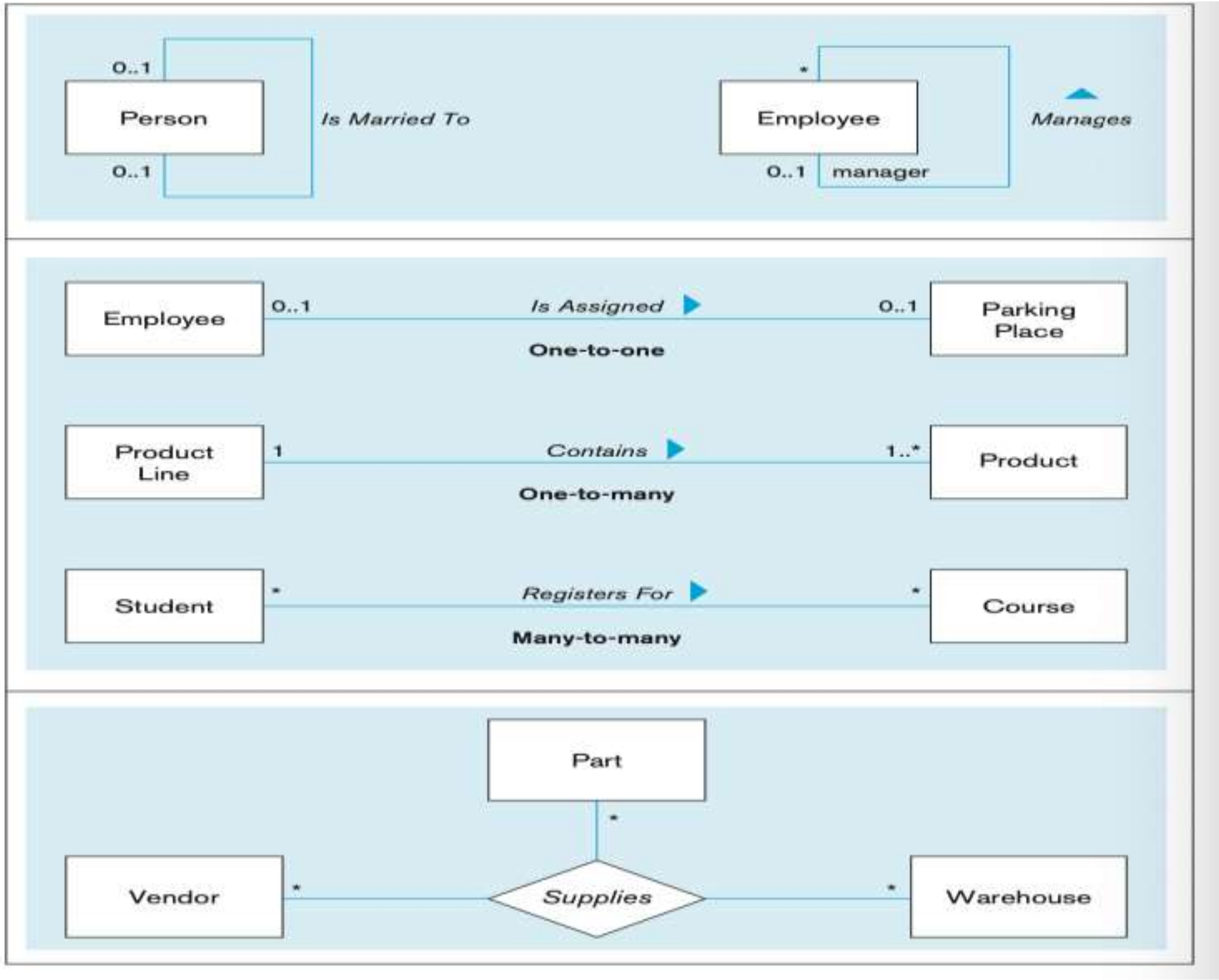
# Association

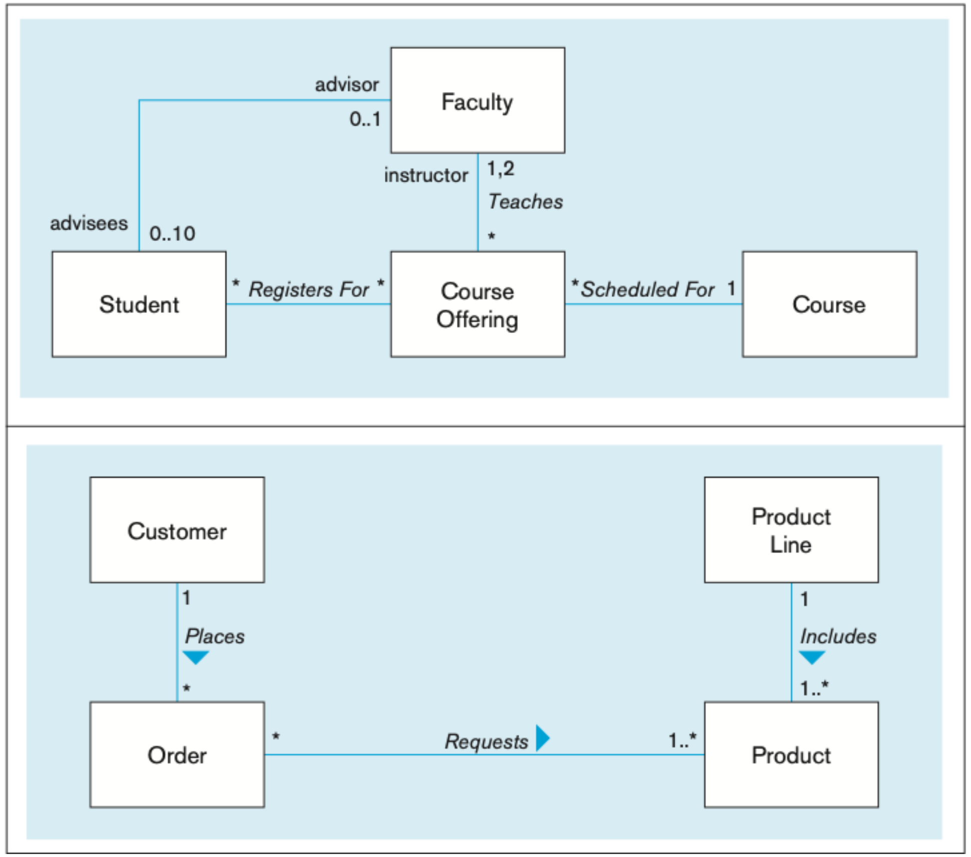
- Parallel to the definition of a relationship for the E-R model, an **association** is a named relationship between or among instances of object classes.
- As in the E-R model, the degree of an association relationship may be one (unary), two (binary), three (ternary), or higher (*n*-ary).
- An association is shown as a solid line between the participating classes. The end of an association where it connects to a class is called an **association role**
- **Multiplicity**
  - A specification that indicates how many objects participate in a given relationship.
  - Multiplicity is shown using following format lowerBound..upperBound

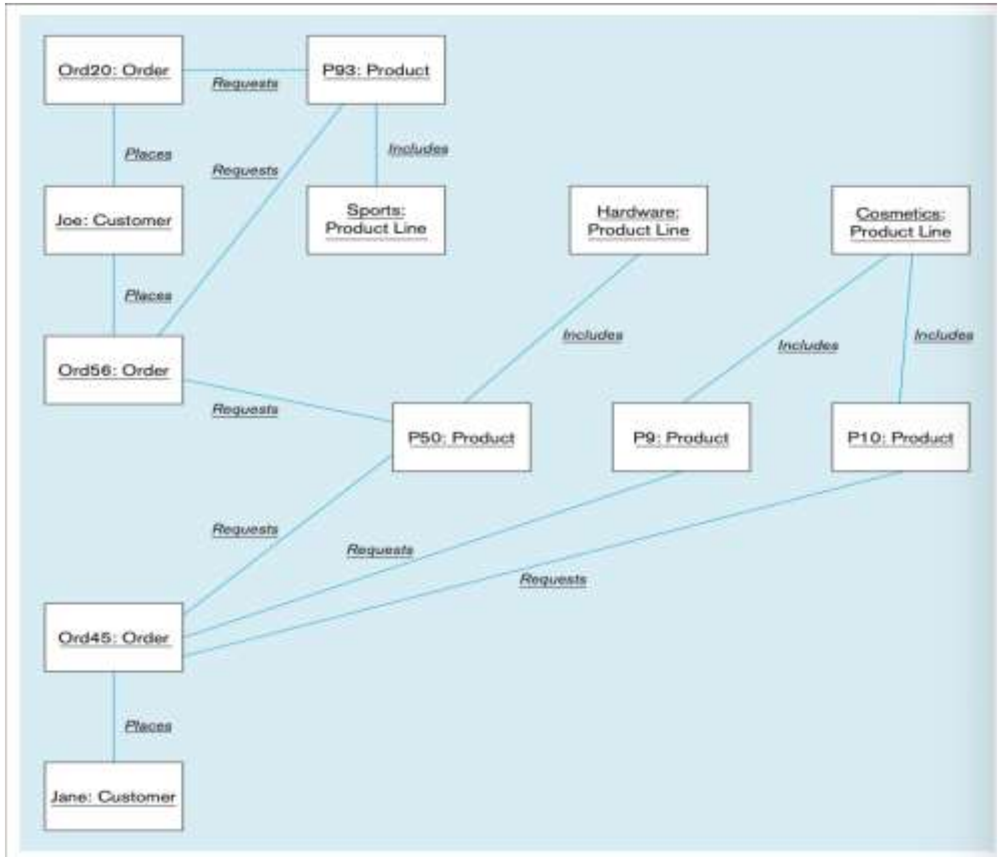
**FIGURE 13-3** Examples of association relationships of different degrees
(a) Unary relationships

(b) Binary relationships

(c) Ternary relationship

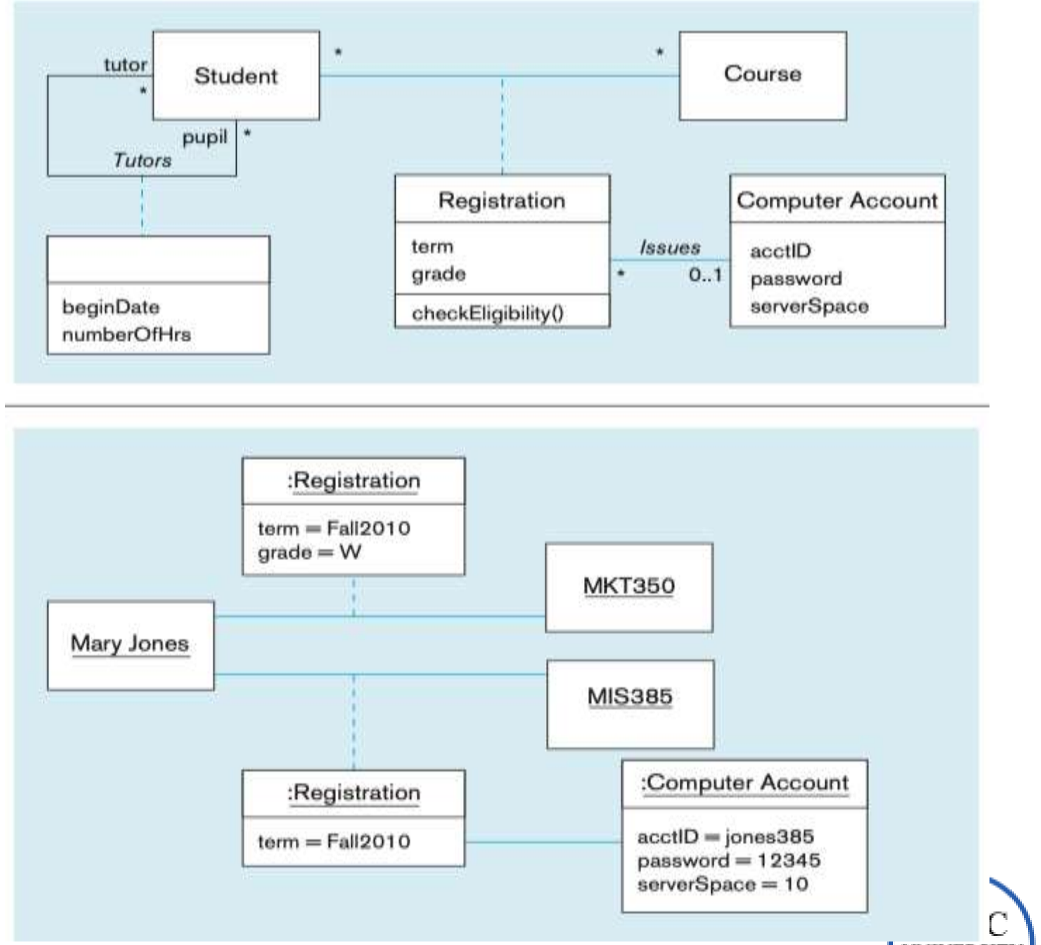



advisor
0..1
Faculty
instructor
1,2
Teaches
*
advisees
0..10
Student
* Registers For *
Course Offering
*Scheduled For 1
Course
Customer
1
Places
*
Order
*
Requests
1..*
Product
Product Line
1
Includes
1..*


Ord20: Order
Requests
P93: Product
Places
Requests
Includes
Joe: Customer
Sports: Product Line
Hardware: Product Line
Cosmetics: Product Line
Places
Includes
Includes
Includes
Ord56: Order
Requests
P50: Product
P9: Product
P10: Product
Requests
Requests
Requests
Ord45: Order
Places
Jane: Customer

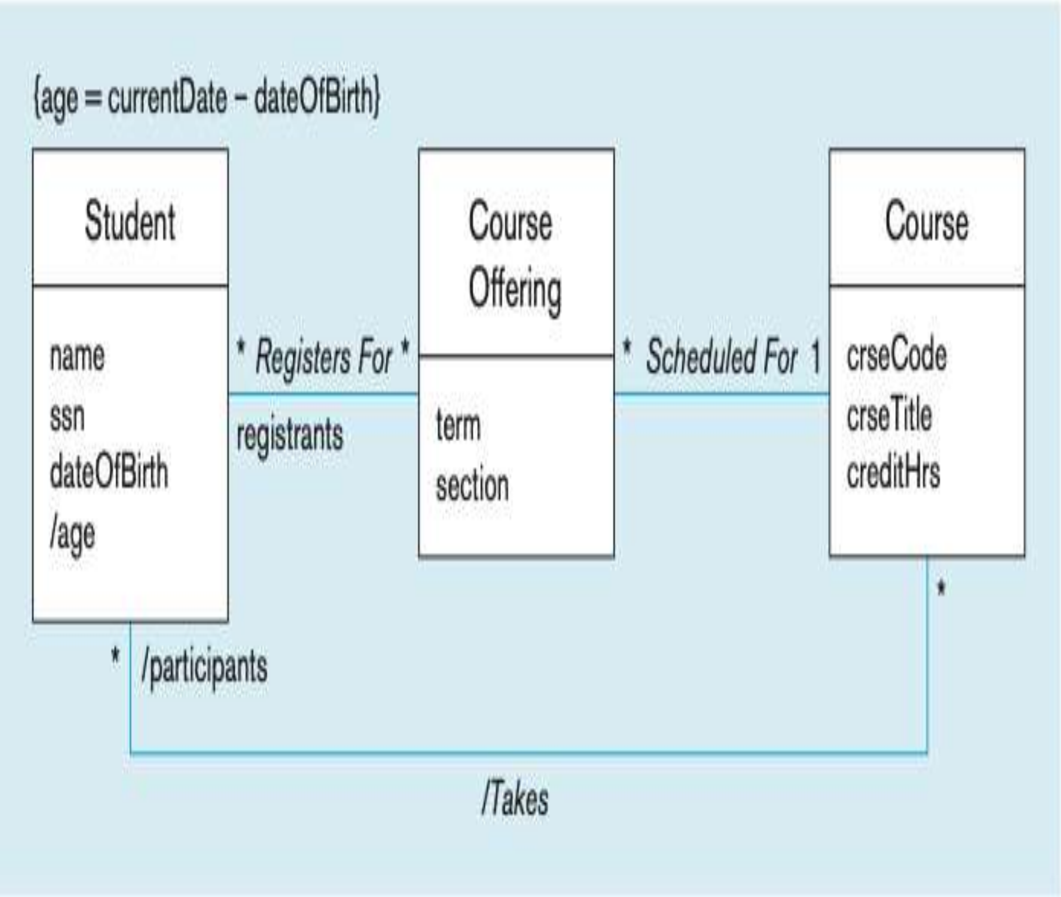
# Association Class Representation

- When an association itself has attributes or operations of its own, or when it participates in relationships with other classes, it is useful to model the association as an **association class**
- For example, in Figure, the attributes term and grade really belong to the many-to-many association between Student and Course.
- The grade of a student for a course cannot be determined unless both the student and the course are known.
- Similarly, to find the term(s) in which the student took the course, both student and course must be known.
- The checkEligibility operation, which determines whether a student is eligible to reg-ister for a given course
- We have also captured the fact that for some course registrations, a computer account is issued to a student



# Derived Atrribute

- A derived attribute, association, or role is one that can be computed or derived from other attributes, associations, and roles, respectively.
- A derived element (attribute, association, or role) is typi- cally shown by placing either a slash (/) or a stereotype of <<Derived>> before the name of the element.
- For instance, in Figure , age is a derived attribute of Student, because it can be calculated from the date of birth and the current date.
- Relation between Student and Course is also derived



# Generalization and Specialization

- It is same as ER modeling
- As this topics are widely covered in undergrad I assume you are familiar with the idea.
- If not please google and refresh your memory as you will be using this concept while drawing data models
- You will also have complex question related to generalization and specialization so don't skip. Ask me if you need help.

# Related Terms

- When drawing a data model you will come across following : overlapping, disjoint, complete, and incomplete. These terms have the following meanings:
- ***Overlapping*** A descendant may be descended from more than one of the sub-classes. (This is the same as the overlapping rule in EER diagramming.)
- ***Disjoint*** A descendant may not be descended from more than one of the sub-classes. (This is the same as the disjoint rule in EER diagramming.)
- ***Complete*** All subclasses have been specified (whether or not shown). No additional subclasses are expected. (This is the same as the total specialization rule in EER diagramming.)
- ***Incomplete*** Some subclasses have been specified, but the list is known to be in-complete. There are additional subclasses that are not yet in the model. (This is the same as the partial specialization rule in EER diagramming.)
- Overlapping and disjoint are mutually exclusive, as are complete and incomplete. Thus, the following combinations are possible: {complete, disjoint}, {incomplete, disjoint}, {complete, overlapping}, {incomplete, overlapping}

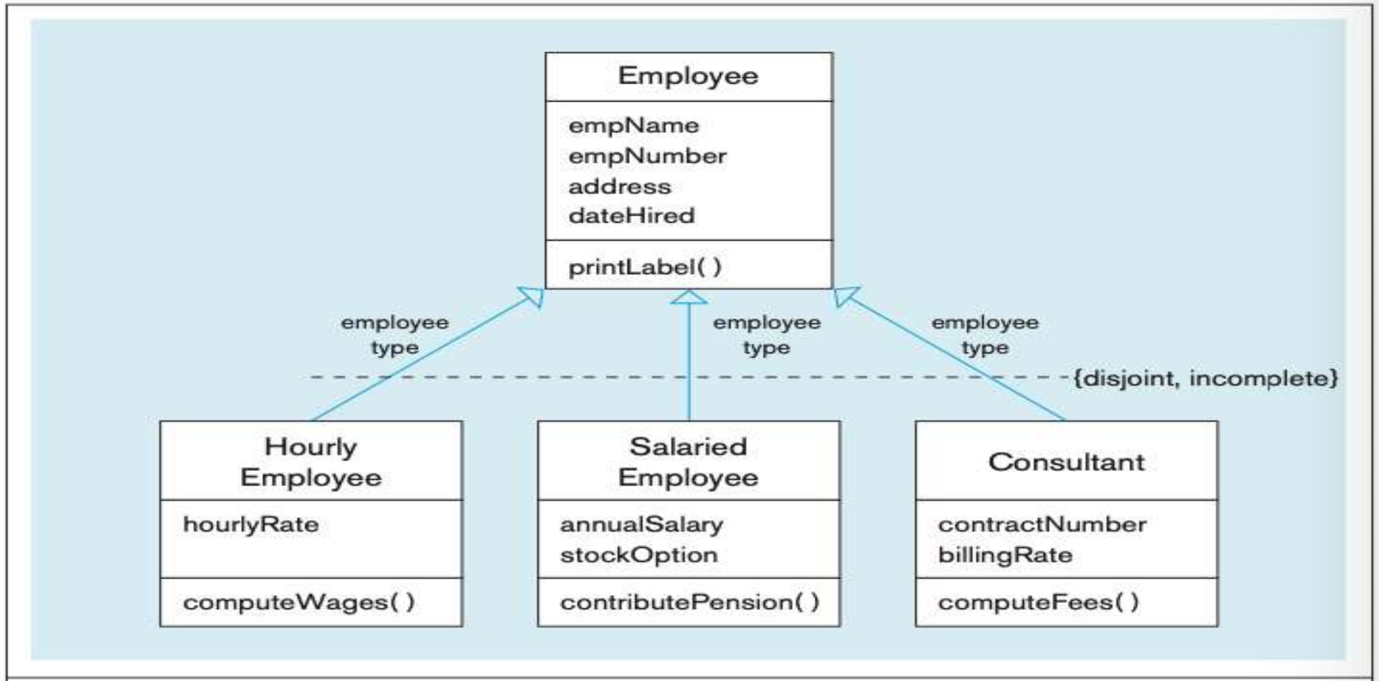
BRAC UNIVERSITY
Inspiring Excellence

- **abstract class** :A class that has no direct instances but whose descendants may have direct instances.
- **concrete class** :A class that can have direct instances.
- **class-scope attribute:** An attribute of a class that specifies a value common to an entire class rather than a specific value for an instance.
- **abstract operation:** An operation whose form or protocol is defined but whose implementation is not defined.

BRAC UNIVERSITY

Inspiring Excellence

**FIGURE 13-9** **Examples of generalization, inheritance, and constraints**
**(a) Employee superclass with three subclasses**



**(b) Abstract Patient class with two concrete subclasses**

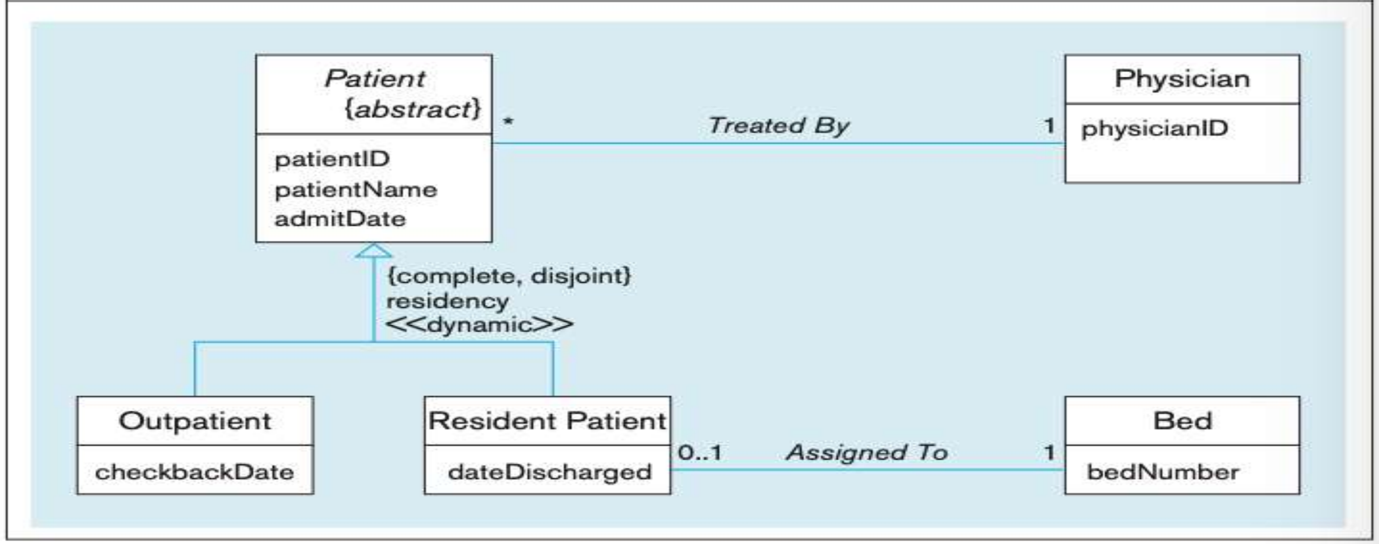


**FIGURE 13-10** Example of an overlapping constraint

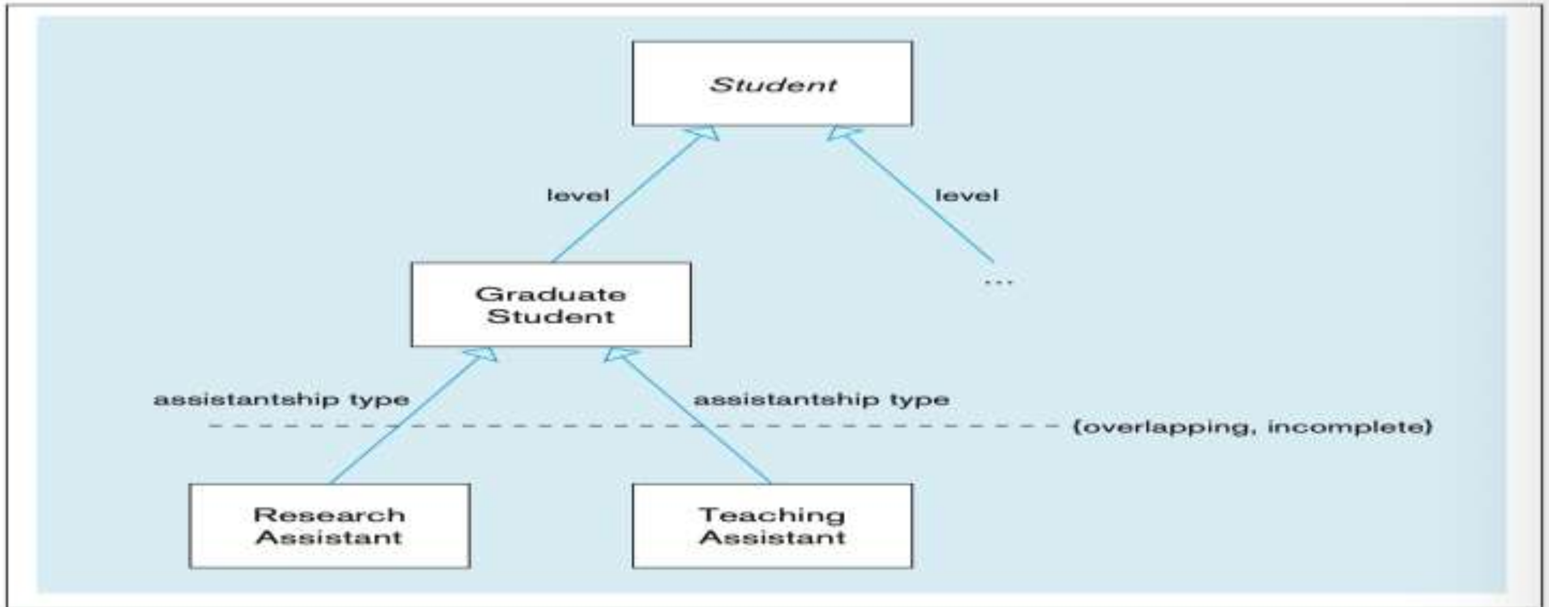
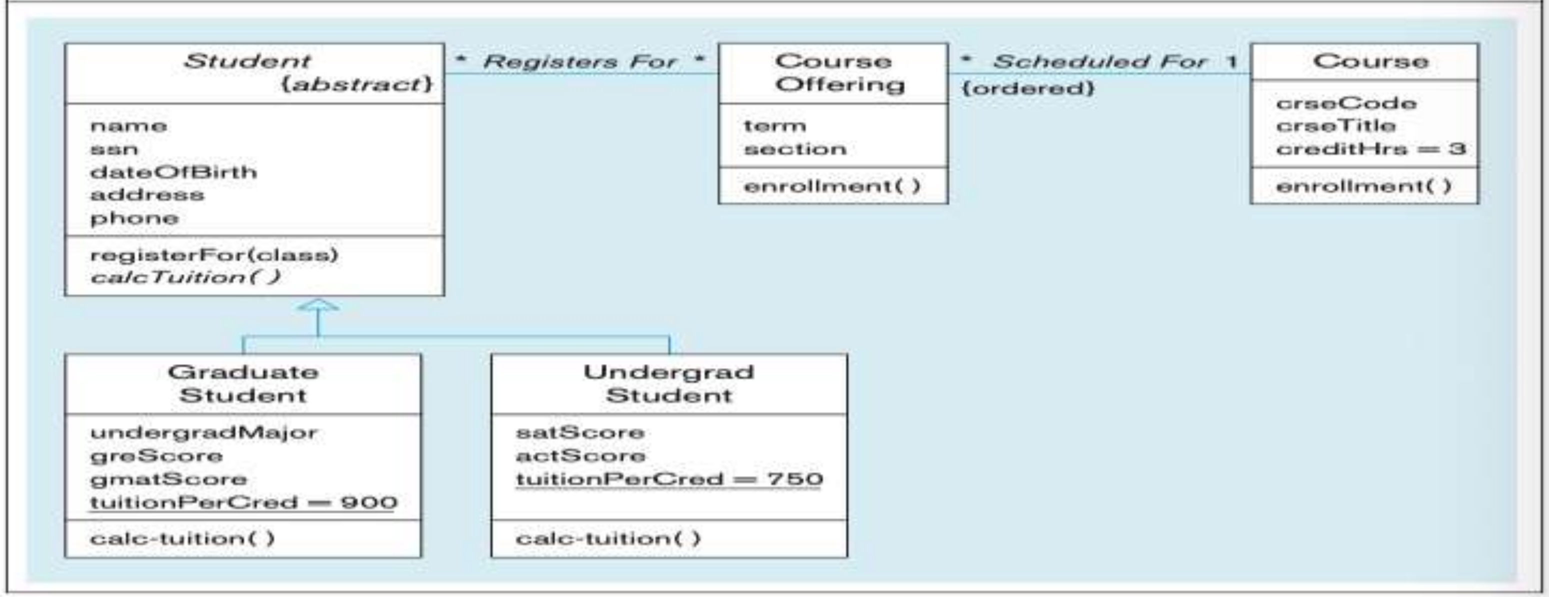


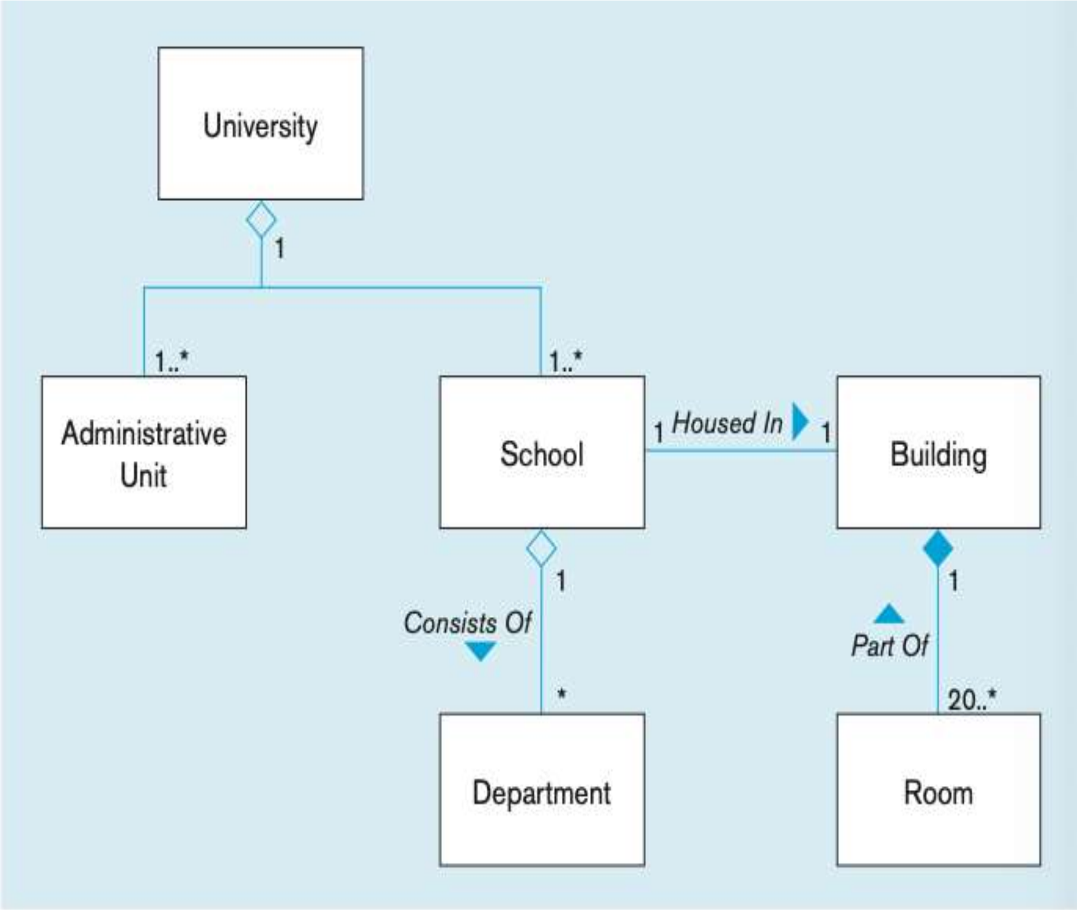
**FIGURE 13-11** Polymorphism, abstract operation, class-scope attribute, and ordering



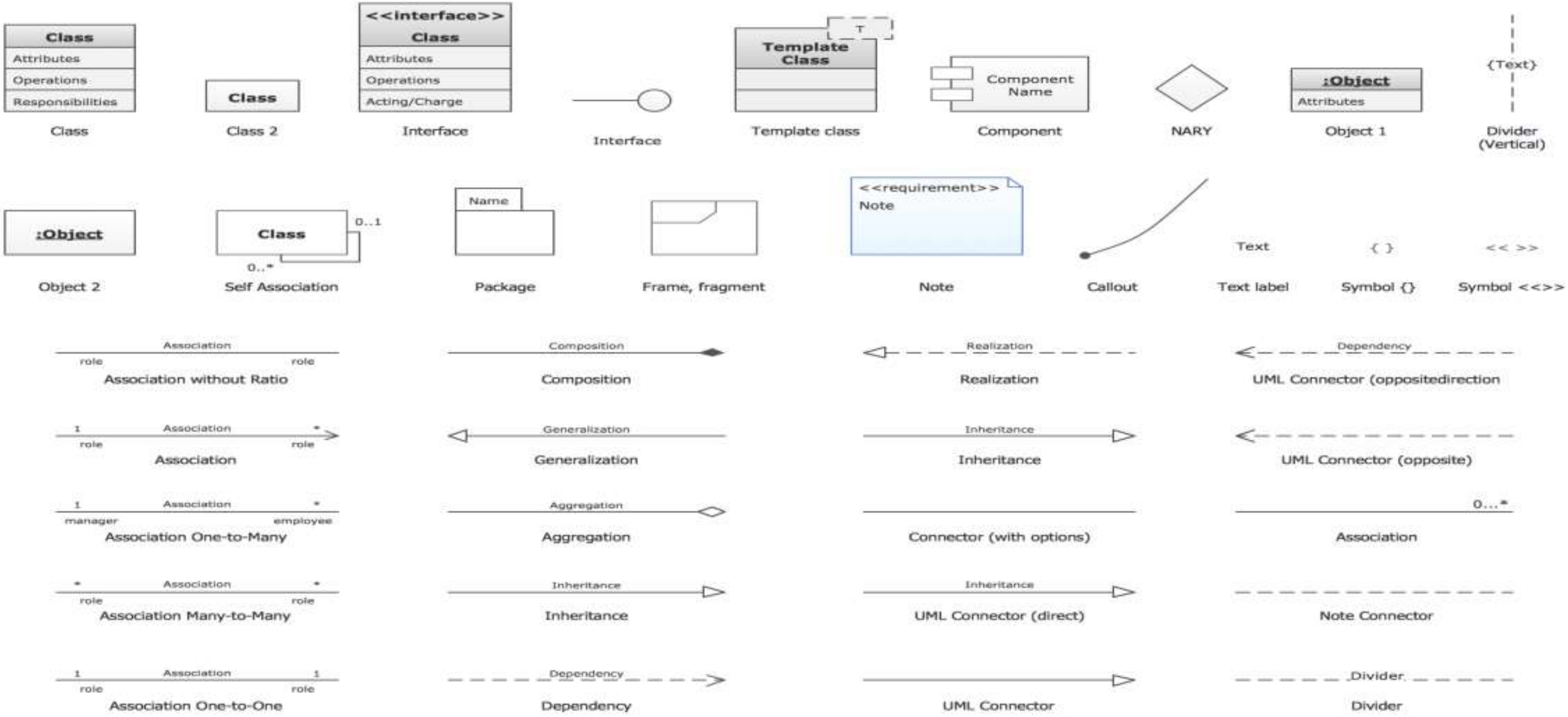
# Multiple Inheritance, Aggregation and Composition

- **Multiple classification**
  - A situation in which an object is an instance of more than one class.
- **aggregation**
  - A part-of relationship between a component object and an aggregate object. Shown with hollow diamond.
- **composition**
  - A part-of relationship in which parts belong to only one whole object and live and die with the whole object.
  - Shown with solid diamond

- Without building there is no existence of room. Its composition.
- School is made of departments. Dept of CSE may still exsists outside dept of engineering
- Demolishing a school(i.e:school of engineering) will not demolish dept of CSE.It will go under school of data science or exsist indepedently.



# All symbols for Data Modeling



# Data Modeling Tools

- **Lucid Chart:** Used by 90% of fortune 500 companies including google and amazon. It is *cloud-based* and can be accessed via the website only.
- **ER Studio:** ER Studio is an instinctive data modelling tool that supports single and multi-platform environments, along with local integration for big data platforms such as — Hive and Mongo DB.
- **IBM InfoSphere Data Architect:**InfoSphere is a data modelling tool that operates on an open-source platform — which is Eclipse. As eclipse is a very popular IDE it is often used to provide an end to end solution.

# Assignment

- We want add a data modeling tool to our list of familiar tools in our CV. So open an account on lucid chart
- Then you need to make a data model for stars.
- Related document is provided check thaqt out to understand the task and data model.

- Reading Reference: https://wps.prenhall.com/wps/media/objects/14735/15089538/M13_HOFF2253_11_SE_C13WEB.pdf