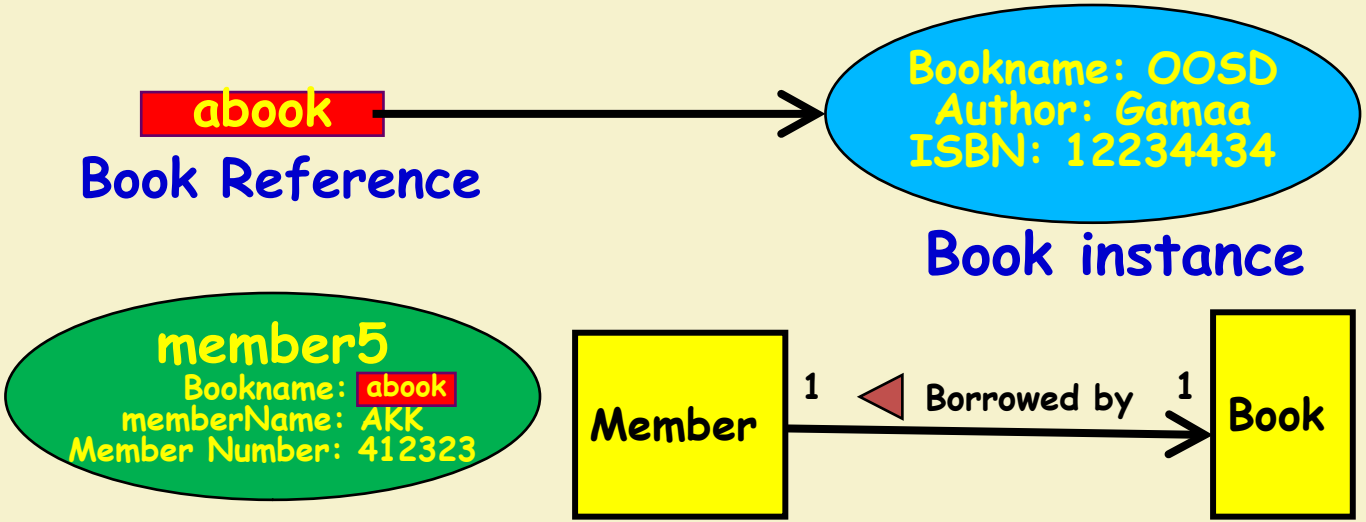
Association: Java Implementation

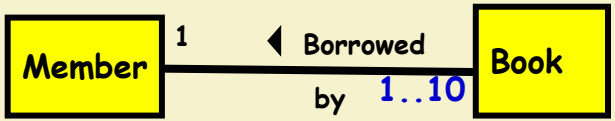
- To implement in Java:
  - Use a reference variable of one class as an attribute of another class



Implementing Association Multiplicities

- Fixed multiplicity is straightforward:

The reference field would be



Book [] book=new Book[10];.

- When the upper end is *:
  - Java collection classes can be used
  - Collection classes were added to Java as a part of JDK 1.2

Java Collections

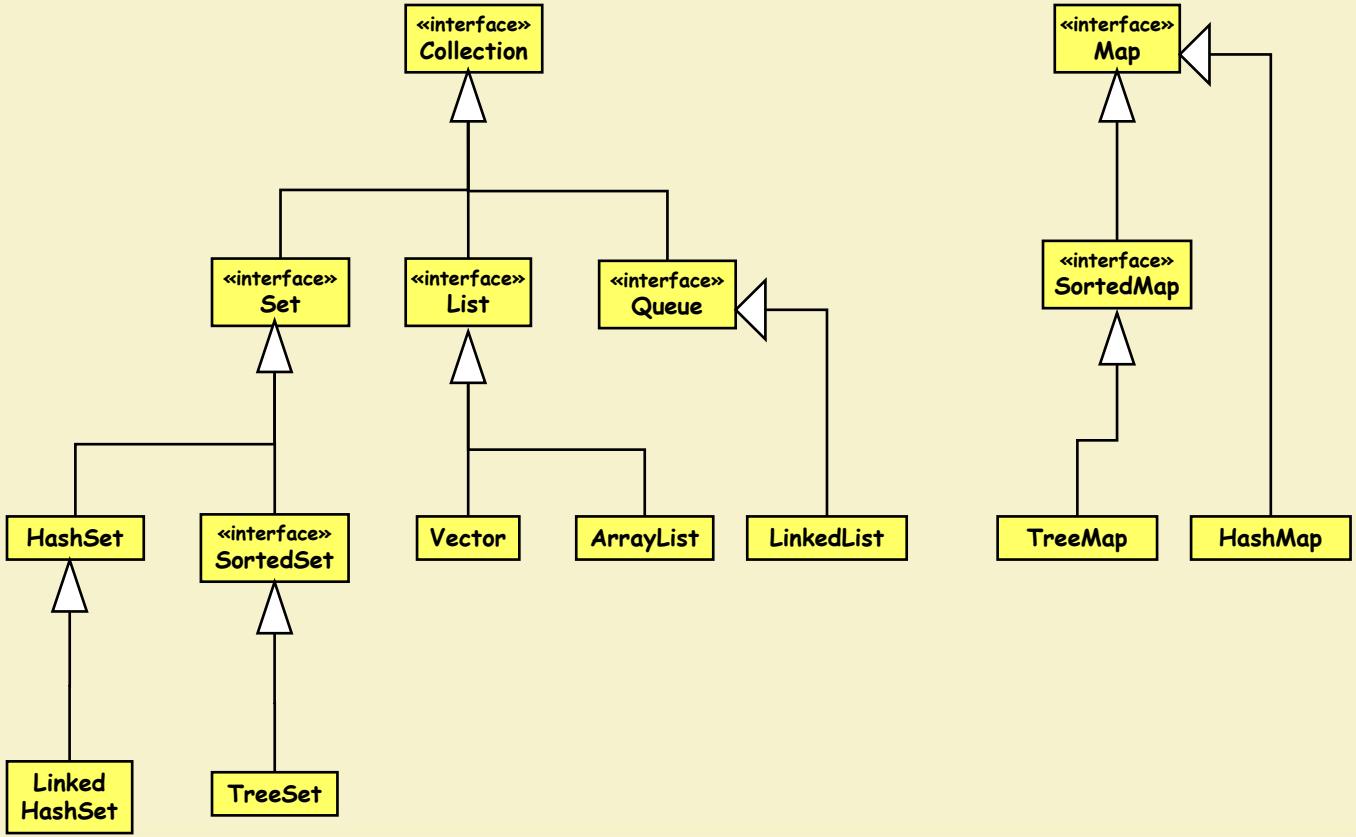
- When an association-end multiplicity is *, need to use Java Collections.
- Collection classes were added to Java as part of JDK 1.2
- Operations supported:
  - **Add**
  - **Remove**
  - **Access individual objects (Iterators)**

Java Collections

- Java collection classes are of three basic types:
  - **List (Ordered)**
  - **Set (Unordered)**
  - **Map (Key-value pairs)**

Bit of History…

- Before Java SDK1.2, Java provided a handful of data structures:
  - **Hashtable**
  - **Vector, etc.**
- These were useful and easy to use:
  - But were not organized into a framework.
- **In SDK1.2 legacy data structures retrofitted to the new model.**


«interface»
Collection
«interface»
Map
«interface»
Set
«interface»
List
«interface»
Queue
«interface»
SortedMap
HashSet
«interface»
SortedSet
Vector
ArrayList
LinkedList
TreeMap
HashMap
Linked
HashSet
TreeSet

Java Collections: What is it?

"The Collections Framework provides a well-designed set of interfaces and classes for storing and manipulating groups of data as a single unit, a collection."

-java.sun.com

- **List : Sequences**
  - Ordering is implicit
  - it is legitimate to ask questions like “what is the first object in the sequence?”
  - Add book as first book etc.
- **Sets: Unordered collection**
- For all Collection classes:
  - Can create an Iterator object to access each item in the collection once.
- **Maps: Qualified associations**
  - Each entry involves a pair of objects.
  - **A map is also called as a dictionary.**
  - it is legitimate to ask questions like “what object- if any - is associated with the following key?” or
  - “does this map contain the following key?”.

Which List to Use?

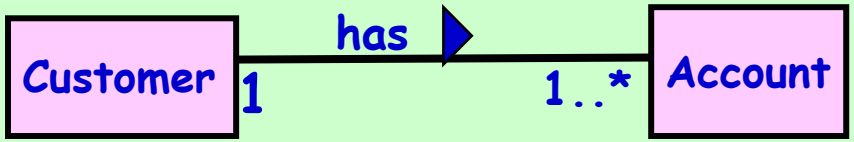
- **LinkedList:**
  - Good if the list changes size (grows or shrinks) frequently
  - Good for accessing either end of the list, but slower when accessing items in the middle of the list
- **ArrayList:**
  - Good if accessing elements by specific position, but slow for adds and removes.

Vector class

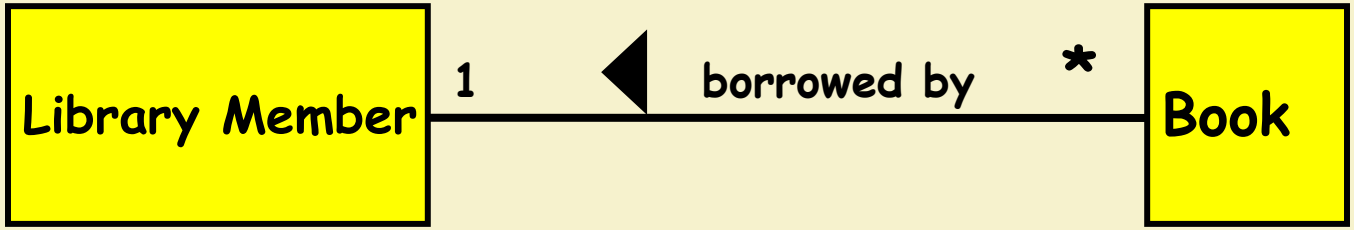
- **In what ways different from ArrayList?**
- Similar to an ArrayList, but synchronized for multithreaded programming.
- Mainly for backward-compatibility with old java.
- Used also as base class for *Stack* implementation.

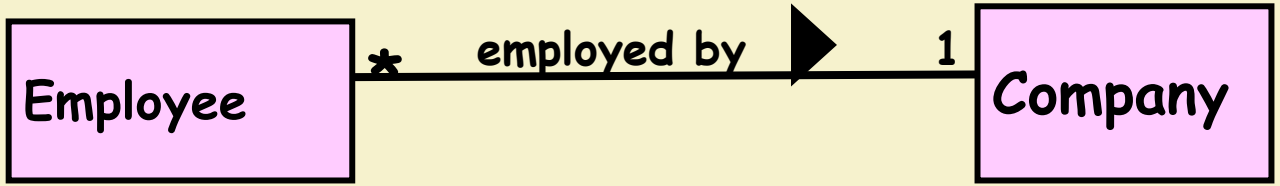
Code for Association Multiplicity

```
class Customer{
        private ArrayList <Account> accounts =
                        new ArrayList<Account>();
   public Customer() {
      Account defaultAccount = new Account();
       accounts.add(defaultAccount);
    }
}
```



Home Work: Write Code for Example Association Relationships





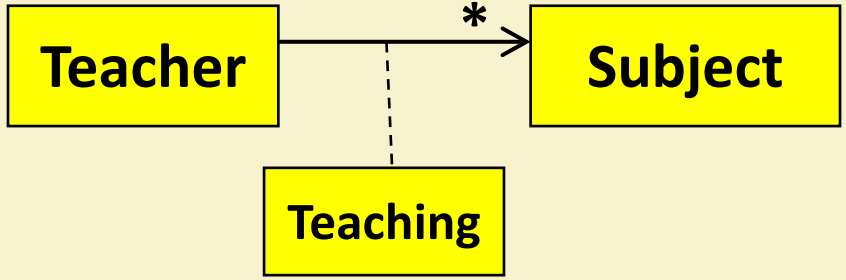
Association Class

- An association relation between classes often has its own attributes and methods:
  - **Example 1: A teacher teaches a subject:**
    - A teacher may teach the same subject multiple times over many semesters
    - Each time the students may be different
  - **Example 2: A person works for a company**
    - The date he joined a company
    - Pay scale and grade at which joined, promotion, so on
    - He may even join the company multiple times

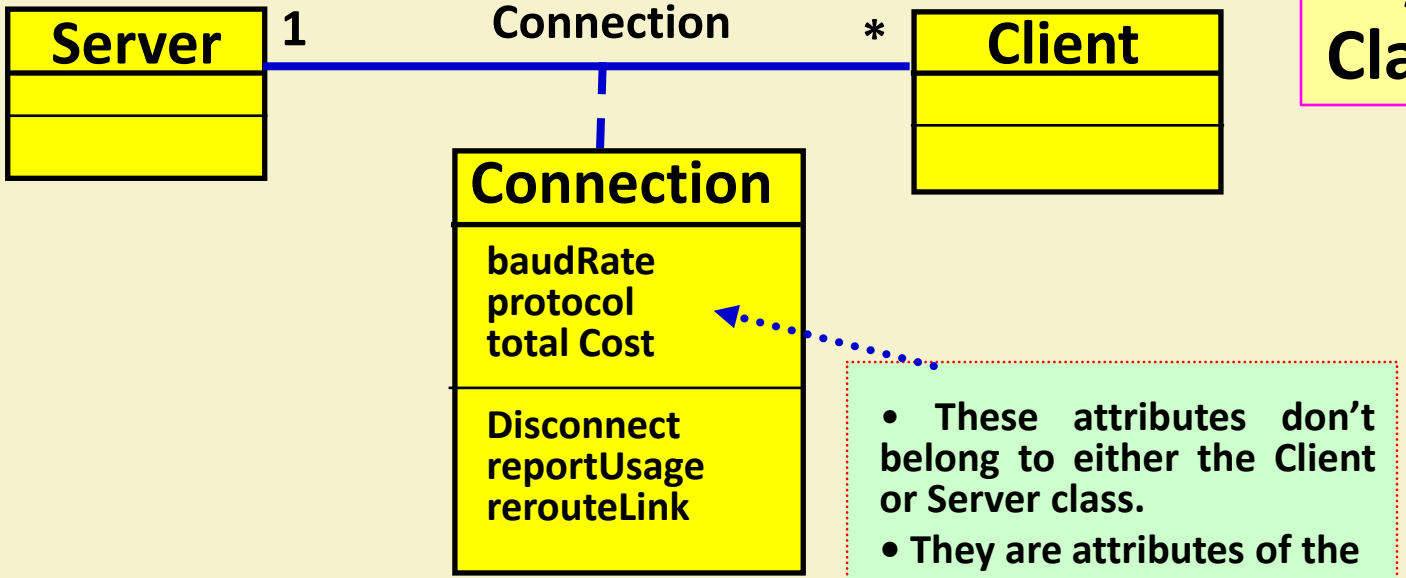
Association Class

Association Class

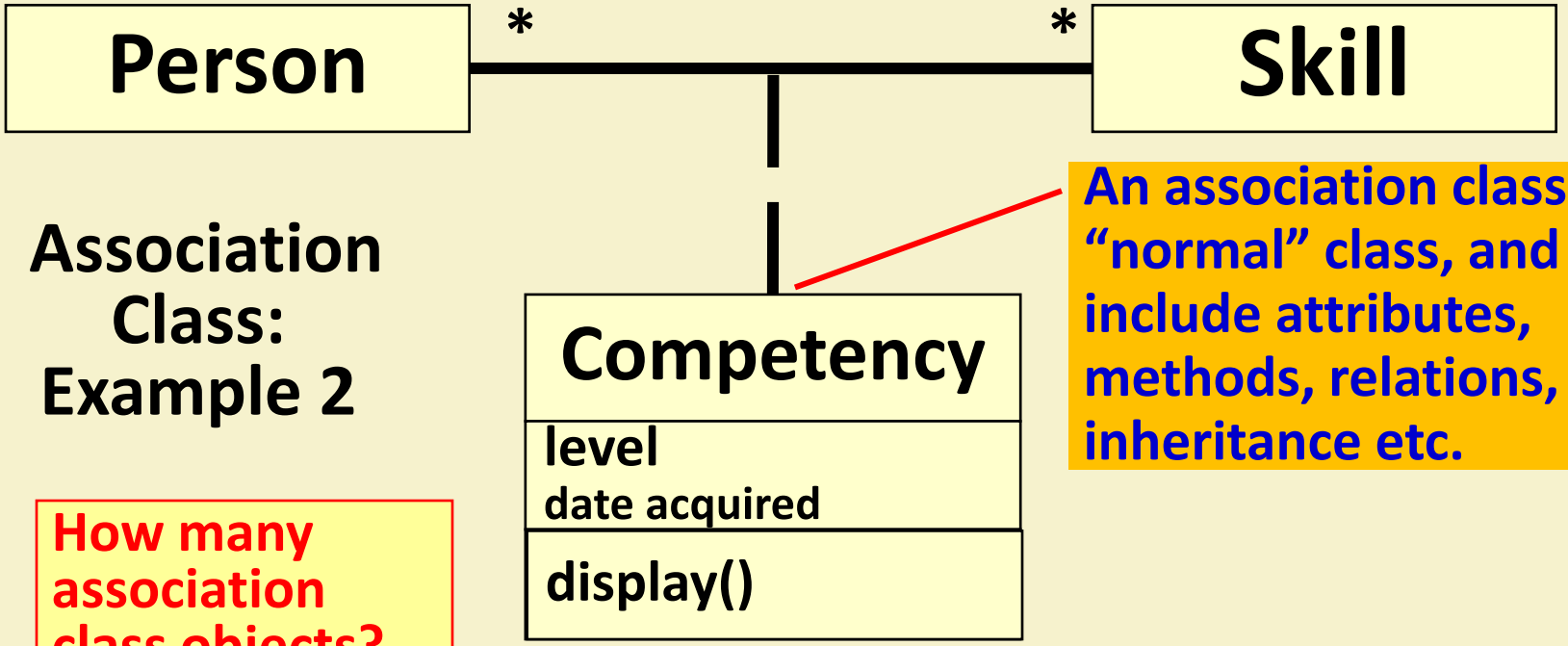
- An association class is shown as a class symbol:
  - Attached to the association symbol by a dashed line.
- An association class:
  - Same name as the association relation because...
  - It represents the association!



Association Class: Example 1



- An association class can have methods as well as attributes.

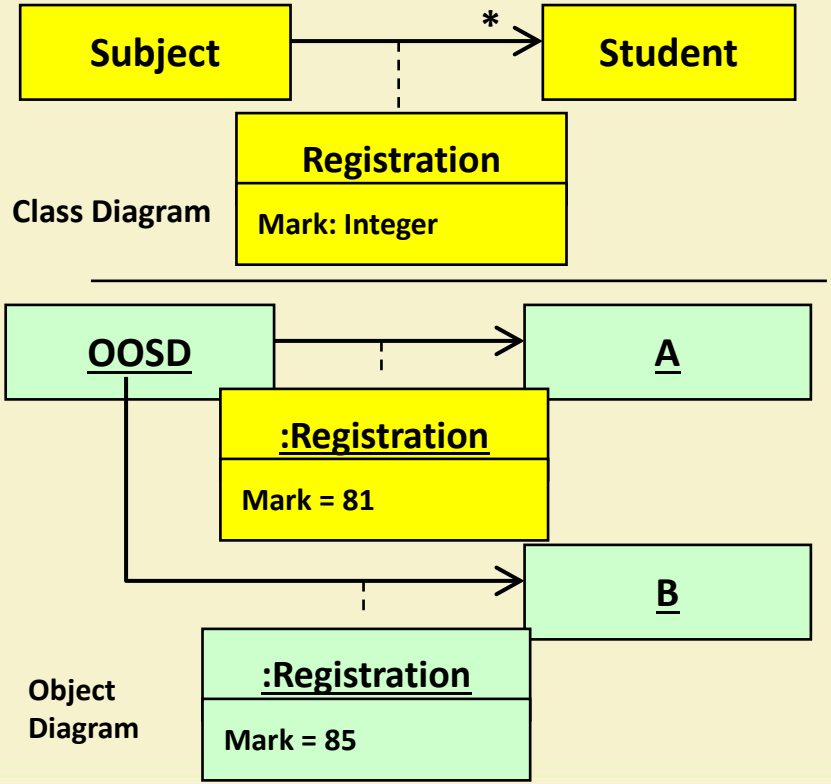


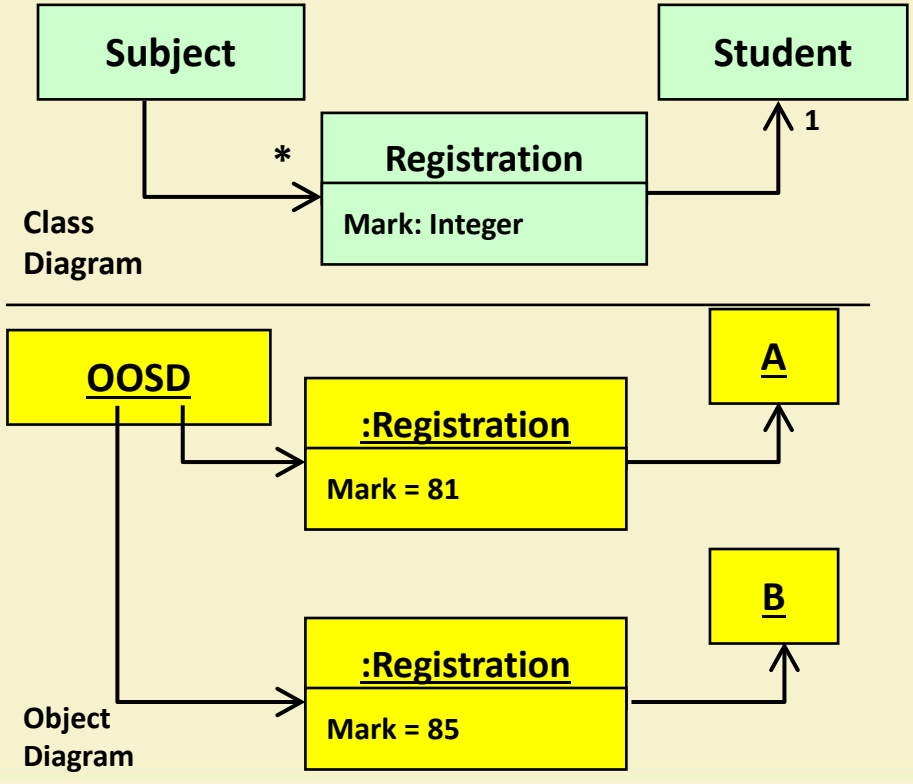
Association Class: Example 2

An association class is a "normal" class, and may include attributes, methods, relations, inheritance etc.

How many association class objects?

Implementing Association Class



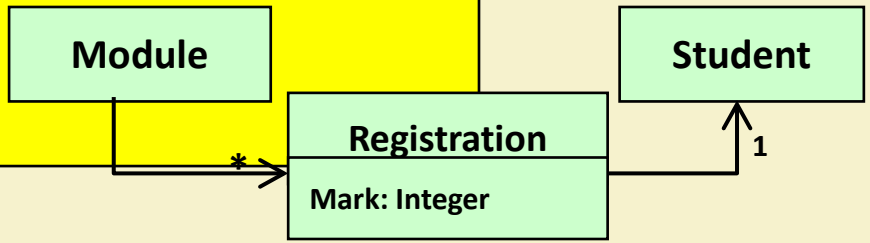


Java Code

```
public class Subject {
  private ArrayList <Registration> reg=new ArrayList<Registration>();
  public void enrol(Student st) {
    reg.add( new Registration(st) );
  }
  ...
}
```

Pass the Student to Registration.

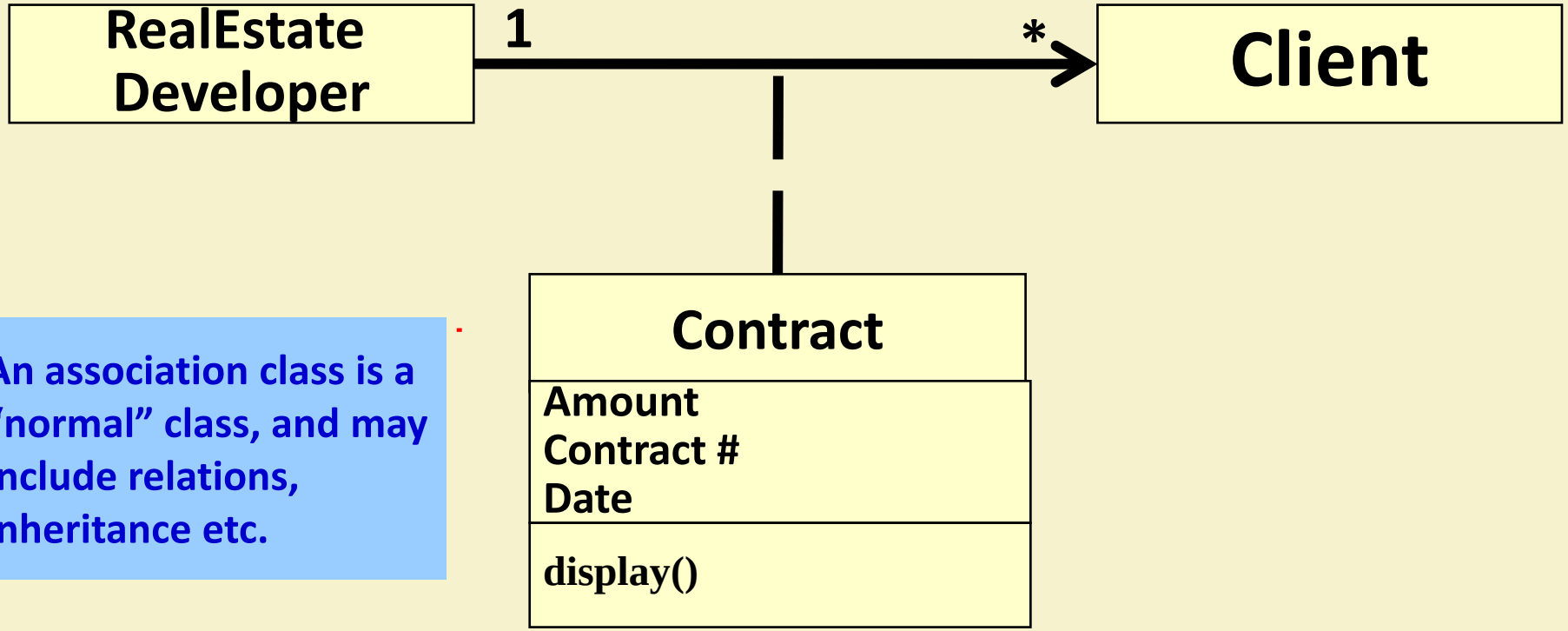
Maintain the link to Registration




```
class Registration {
  private Student student;
  private int mark;
  Registration(Student st) {
    student = st; mark = 0;
  }
  ...
}
```

Keep track of the Student reference.

Association Class: Example 4



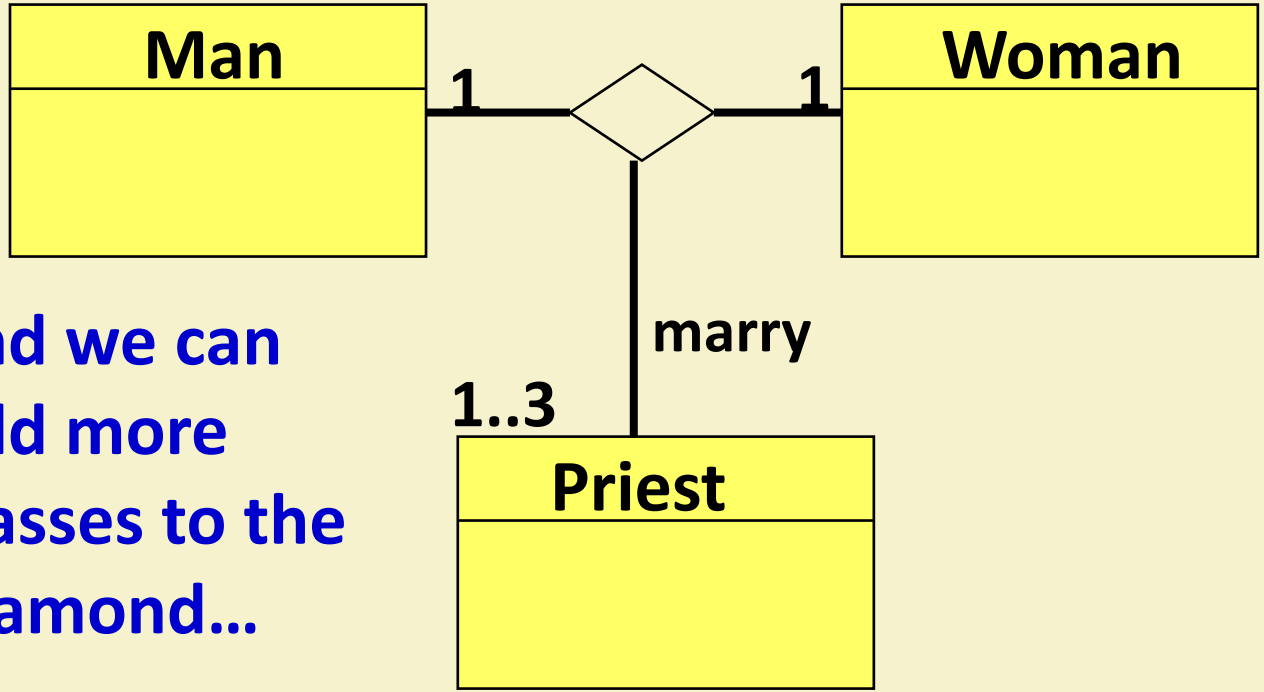
An association class is a "normal" class, and may include relations, inheritance etc.

```
public class RealEstateDeveloper{
    private Vector <Contract> contracts= new Vector <Contract>();
public void buy(Client c){contracts.add(new Contract(c))};
}
public class Client{
    private Address address;
    public Address getCurrentAddress(){}
}
public class Contract{
    private Client client;
    private int contractNo;
    public Contract(Client c){ client=c;}
}
```

Ternary Association

- An object of an association class:
  - Stores the details of one link for two or more associated class.
- Some times three (or more) classes may be associated:
  - In this case an association end represents the potential number of values at that end when the values at the other end is kept fixed.

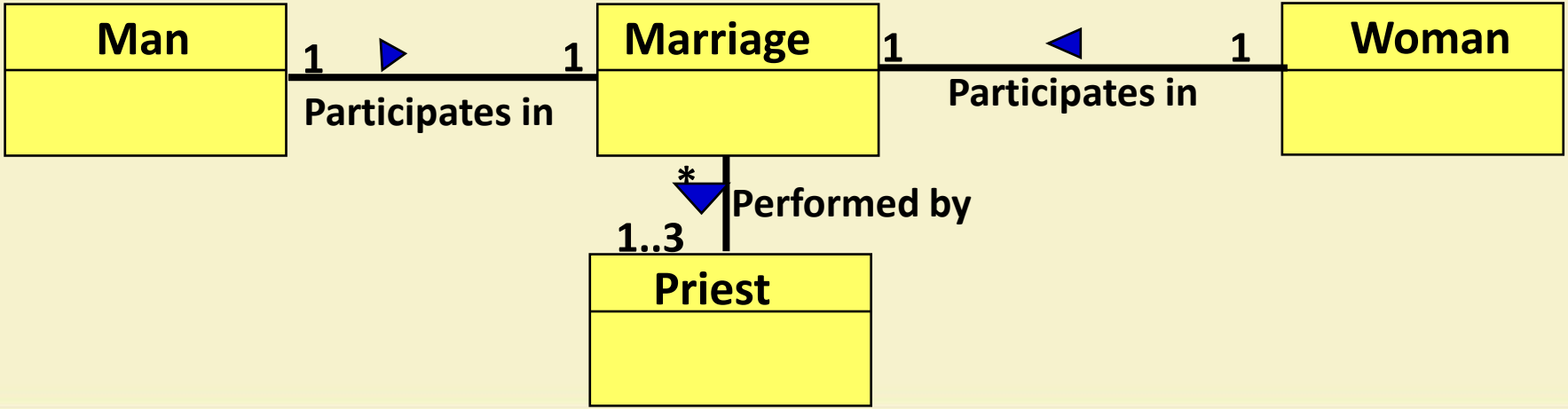
Ternary Association

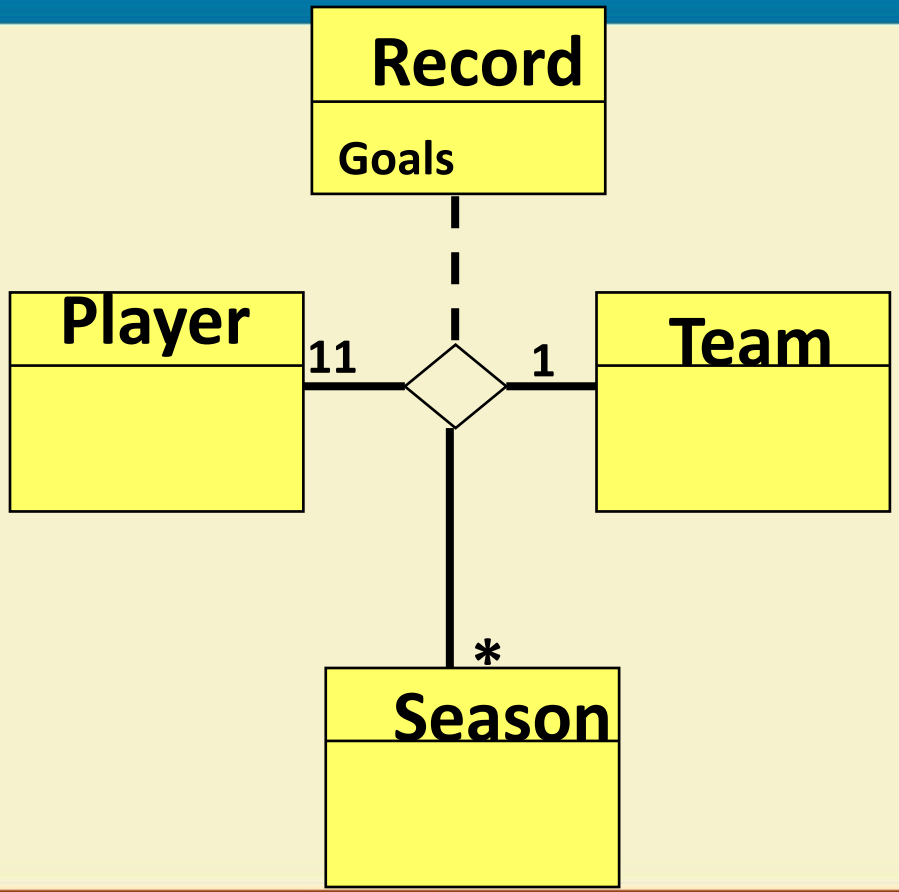


and we can add more classes to the diamond…

Implementation of Ternary Association

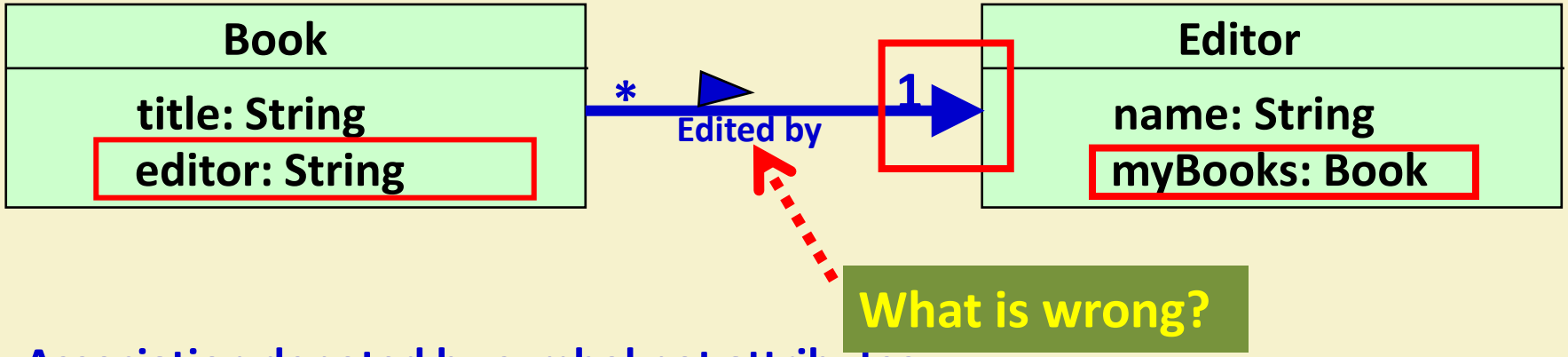
- Implementation of ternary association.
  - Decompose it into a set of binary associations.





As association arity increases, implementation complexity increases… We shall exclude from our discussions 3ary and higher-ary associations…

Association: Quiz



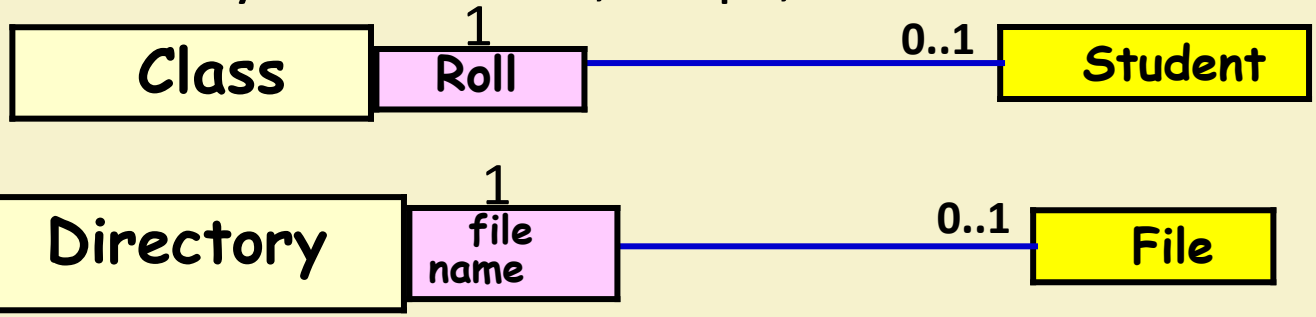
1. **Association denoted by symbol not attributes.**
   - Implementation (LinkedList, ArrayList, arrays, vectors, ids etc) is left to the detailed design phase.
2. **Wrong arrow type**

Qualified Association

Qualified Association

- A qualified association allows us to express uniqueness

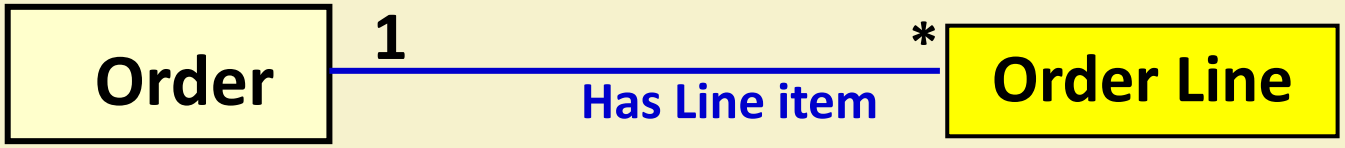
- Implemented by hash tables, maps, dictionaries.



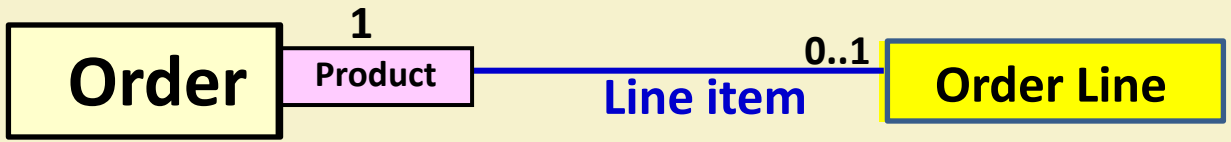
- How to read?
- There exists upto one file for each instance of filename in the directory .

An Example

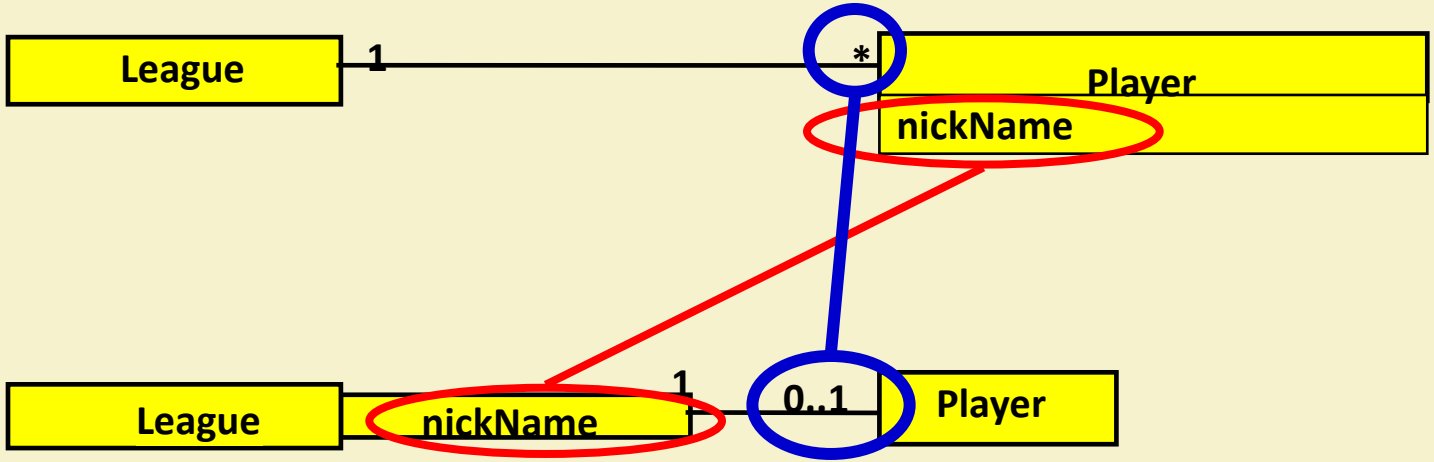
- Consider an order has many Order lines.



- Let us express: There is at most one Order Line in the Order for each instance of Product.

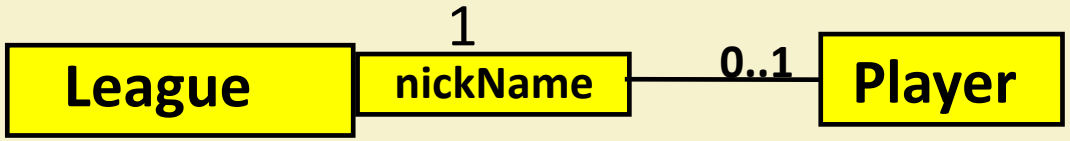


Qualified Association…



- The second conveys more information…
- Hint on implementation…

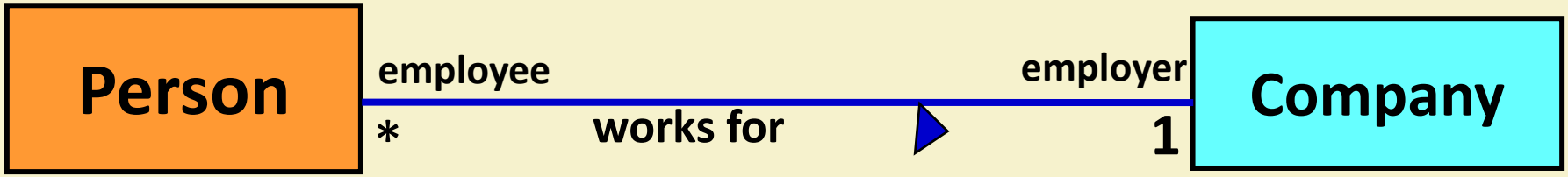
Qualified Association: Implementation




```
public class League {
  private Map players=new HashMap();
  public void addPlayer
    (String nickName, Player p) {
     if(!players.containsKey(nickName))
           players.put(nickName, p);
      }
}
```

```
public class Player {
     private League
                 league;
   }
```

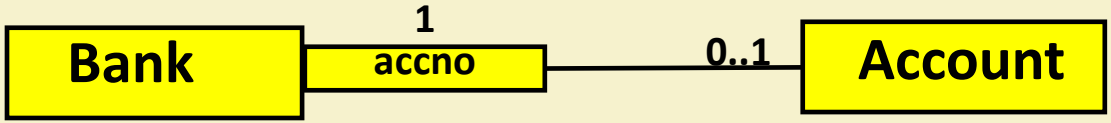
Converting to Qualified Association



Assume company assigns each employee a unique empId.

Give qualified association representation…

Qualified Association



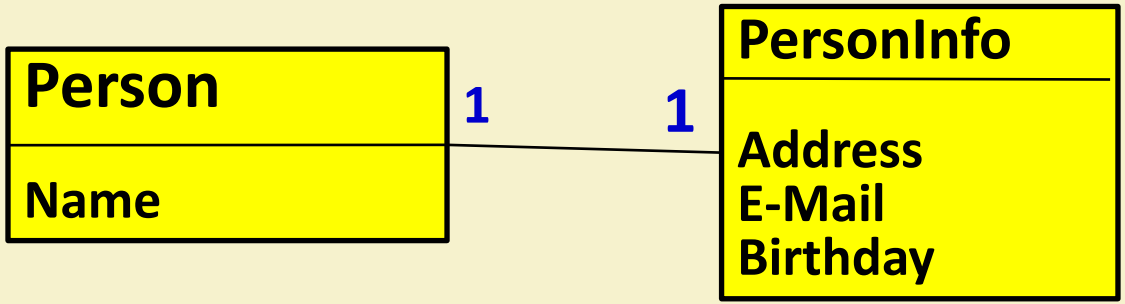
- Qualifier hints at setting up efficient access to linked objects:
  - For example, access accounts based only on the account number;
  - Also to avoid a linear search through all accounts.

Summary of Implementation of Association

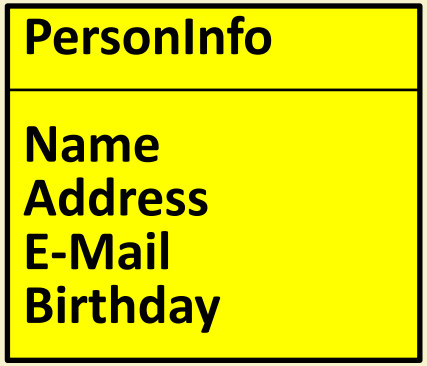
- 1-to-1 association:
  - Role names become attributes
- 1-to-many association:
  - Translate into a Vector or ArrayList
- Qualified association:
  - Translate into a Map or Hash table

Overdoing Associations

- Avoid unnecessary Associations

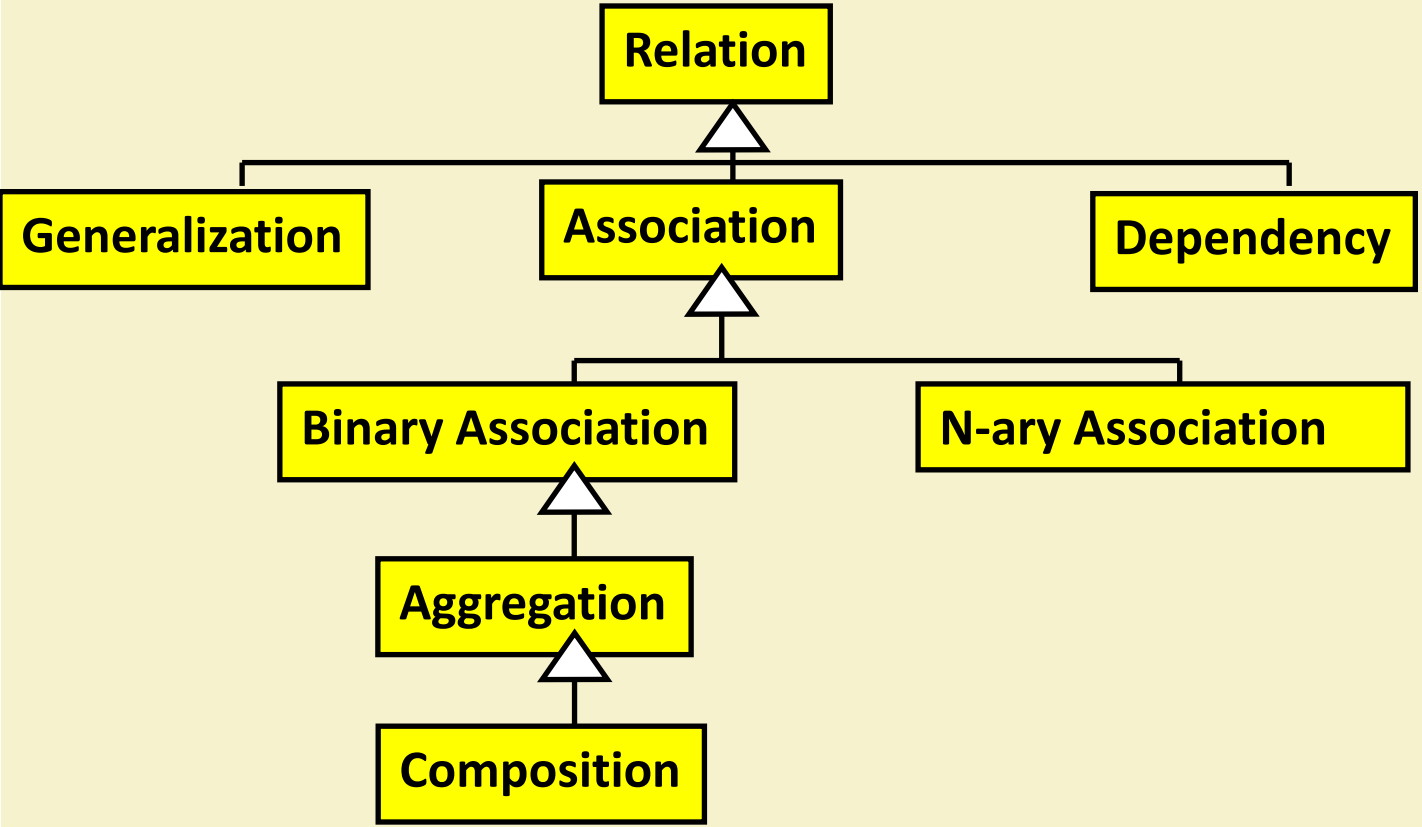


Avoid This…



Do This

Class Relationships

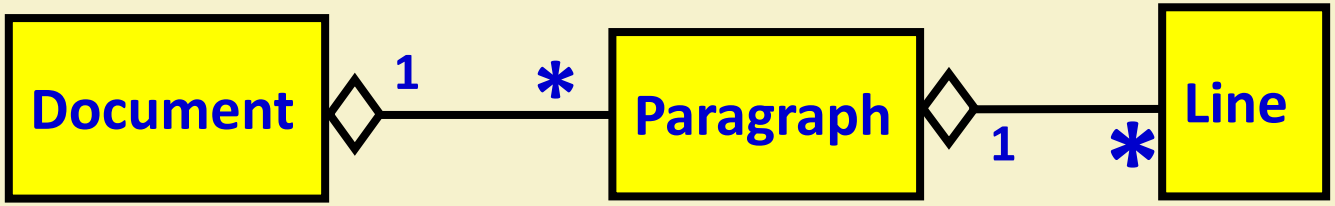


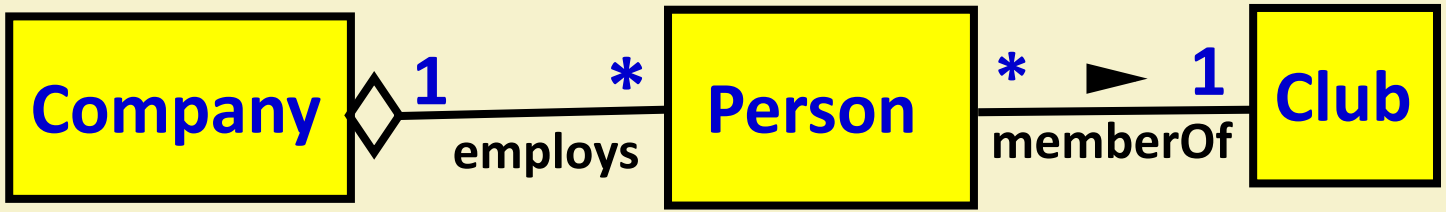
Aggregation Relationship

- Represents whole-part relationship
- Represented by a diamond symbol at the composite end.

- Usually creates the components:
  - But often indistinguishable from plain association.
  - **Except, aggregate usually invokes the same operations of all its components.**
- Usually owner of the components:
  - But can share with other classes

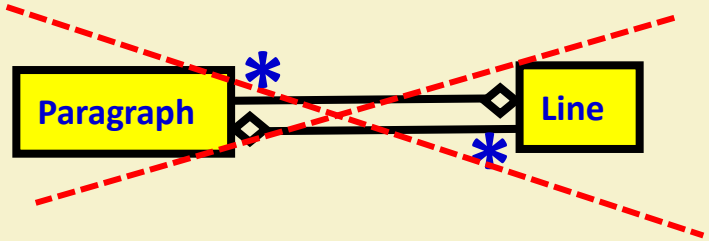
Aggregation: Examples





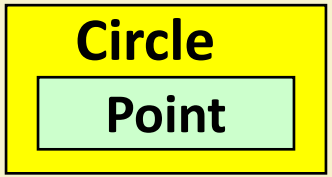
Aggregation cont...

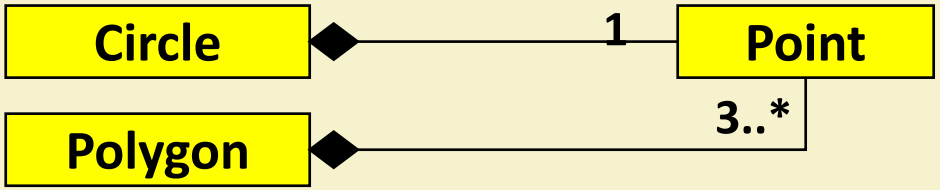
- An aggregate object contains other objects.
- Aggregation limited to tree hierarchy:
  - No circular inclusion relation.



Composition

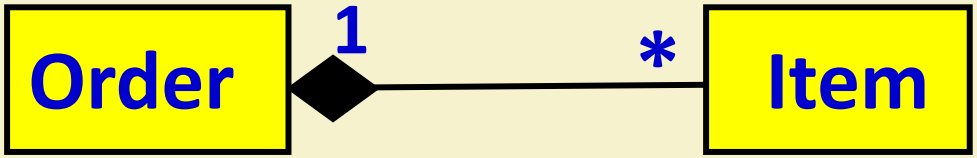
- A stronger form of aggregation
  - The whole is sole owner of its part.
    - A component can belong to only one whole
  - The life time of the part is dependent upon the whole.

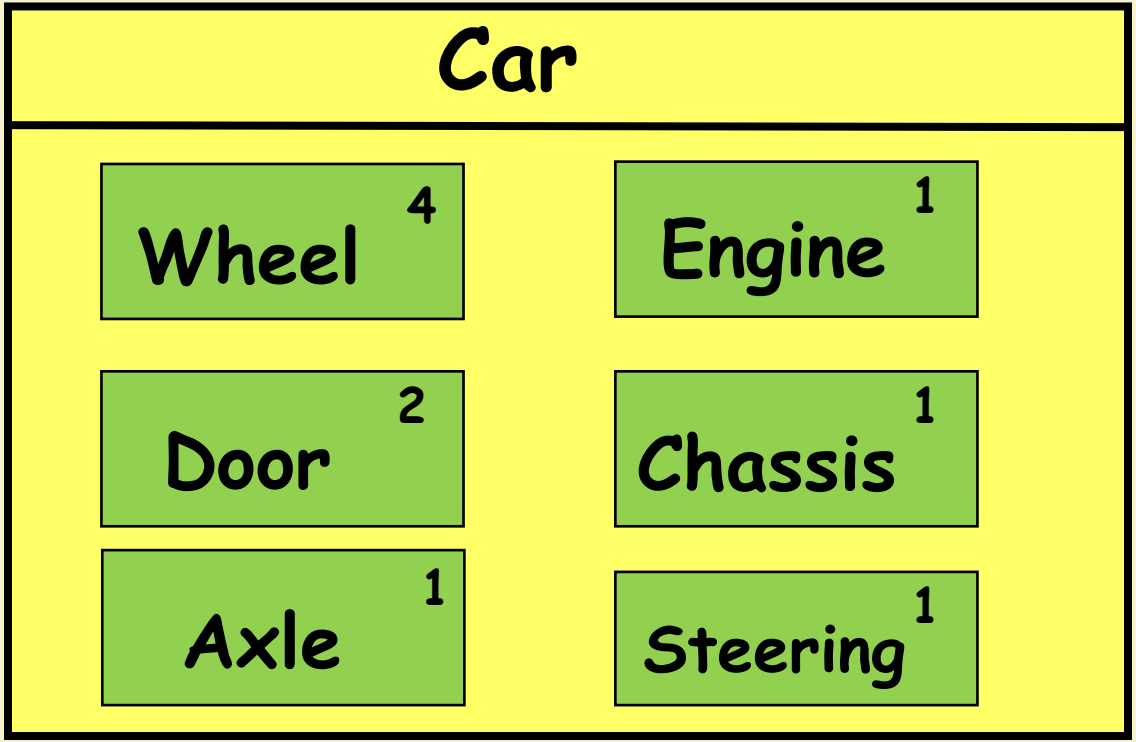




Composition Relationship

- Life of item is the same as that of order…



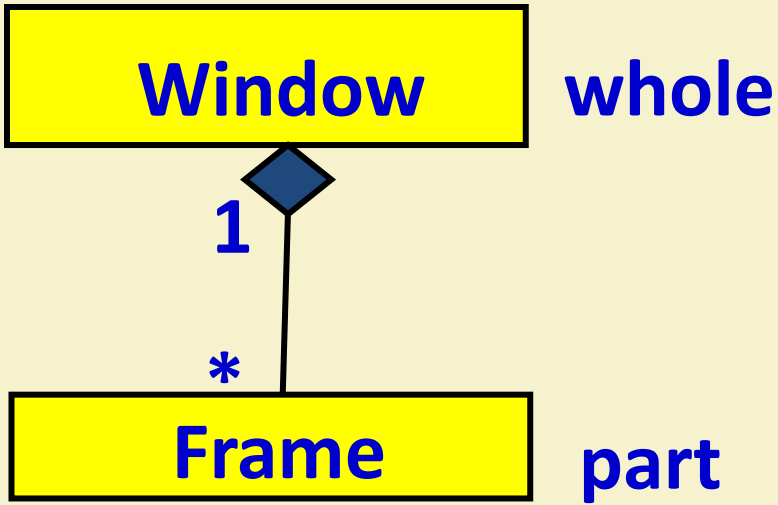


Composition: Alternate Notation

- An object (component) may be a part of ONLY one composite at a time.

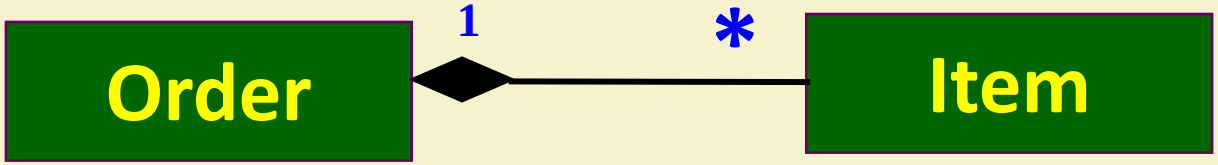
Composition

  - Whole is responsible for the creation and disposition of its parts.



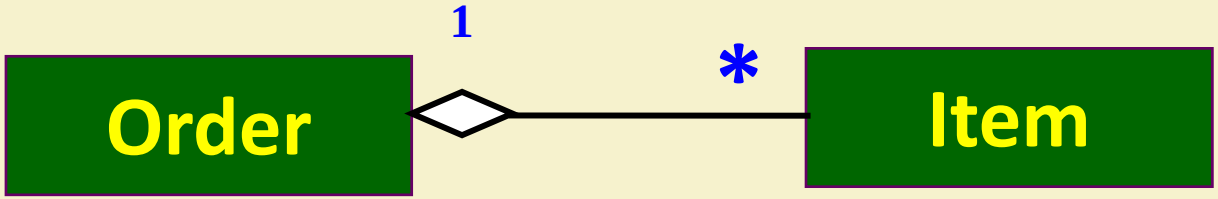
Aggregation vs. Composition

- **Composition:**
  - Composite and components have the same life line.
- **Aggregation:**
  - Lifelines are different.
- Consider an **order** object:
  - **Aggregation:** If order items can be changed or deleted after placing order.
  - **Composition:** Otherwise.



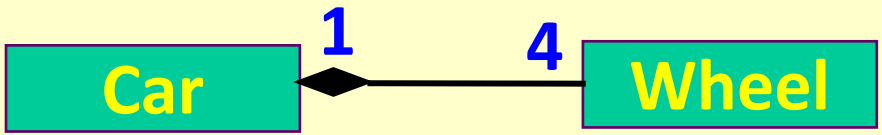
Composition

Composition versus Aggregation



Aggregation

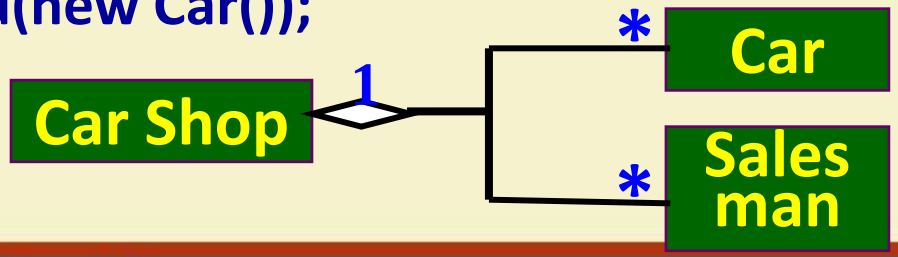
```
public class Car{
    private Wheel wheels[4];
    public Car (){
            wheels[0] = new Wheel();
            wheels[1] = new Wheel();
            wheels[2] = new Wheel();
            wheels[3] = new Wheel();
        }
    }
```



Implementing Composition

Implementing Aggregation

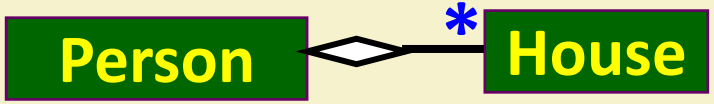
```
import java.util.ArrayList;

public class CarShop{

private ArrayList<SalesPerson> people = new
    ArrayList<SalesPerson>();

    private ArrayList<Car> cars = new ArrayList<Car>();

    public addCar()  {

        cars.add(new Car());

}
```



Inner Classes Translation

- **If house is used only in the Person class**:
  - It is usually declared as an inner class in Person.

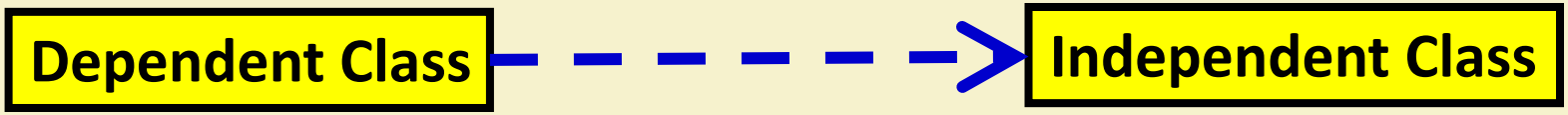
```
public class Person {
    private Name name;
    private House house;
      ...
    class House {
      ...
    }
 }
```



How to identify composition relationship?

- Lifetime of part is within lifetime of composite
  - There is a create-delete dependency
- There is an obvious physical or logical assembly
- Some properties of composite propagate to parts (e.g., location)
- Operations applied to composite propagate to parts (e.g., destruction, movement, recording)

Class Dependency



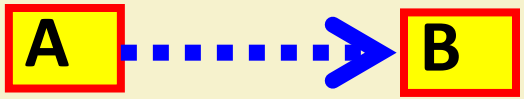
- Dependency relationship can arise due to a variety of reasons:
  - **Stereotypes are used to show the precise nature of the dependency.**

Type of dependency	Stereotype	Description
Abstraction	«abstraction»	Relates two model elements, that represent the same concept at different levels of abstraction
Binding	«bind»	Connects template arguments to template parameters to create model elements from templates.
Realization	«realize»	Indicates that the client model element is an implementation of the supplier model element
Substitution	«substitute»	Indicates that the client model element takes place of the supplier.

Dependency

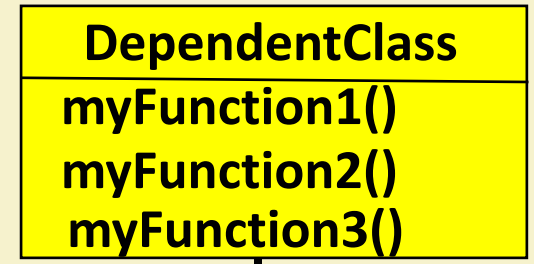
- Dependence are commonly caused by:
  - Local variable
  - Parameter
  - Return value

Dependency




```
Class A {
    B Foo(B x) {
        B y = new B();
        return y;}
}
```

Dependence – Possible Implementations



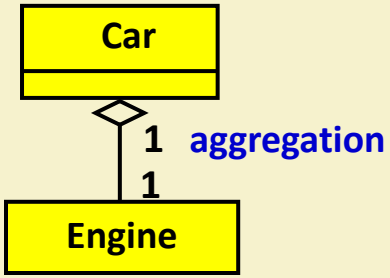

```
class DependentClass{
    void myFunction1(
    ReferencedClass r ) {..}
    referencedClass myFunction2(
    .. ) { .. }
    void myFunction3( .. ){
    ReferencedClass r; ..}
}
```

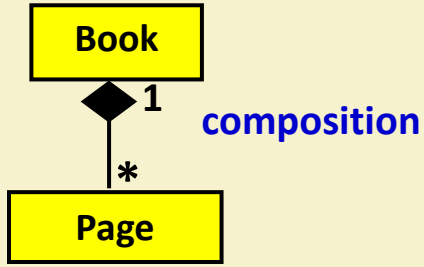

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

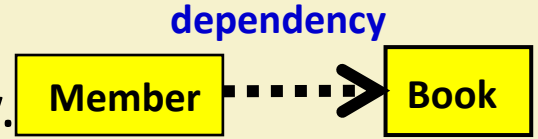
Association Vs. Aggregation

- Is aggregation an association?
- Is composition an aggregation?

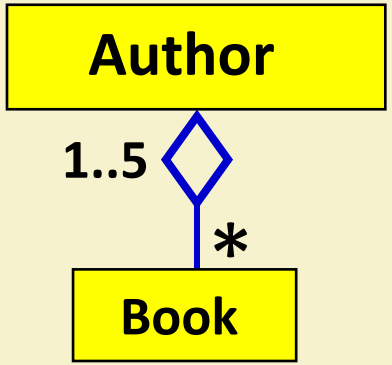
- **aggregation: "is part of"**
  - Symbolized by empty diamond
- **composition: "is made of"**
  - Stronger version of aggregation
  - The parts live and die with the whole
  - Symbolized by a filled diamond
- **dependency: Depends on…**
  - Represented by dotted arrow.



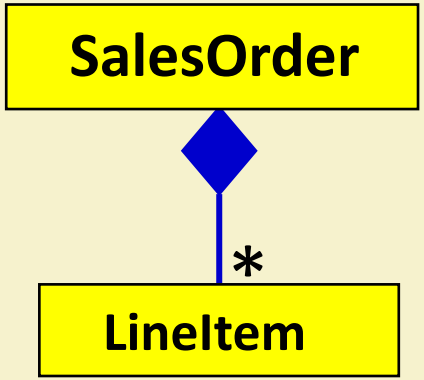




Aggregation

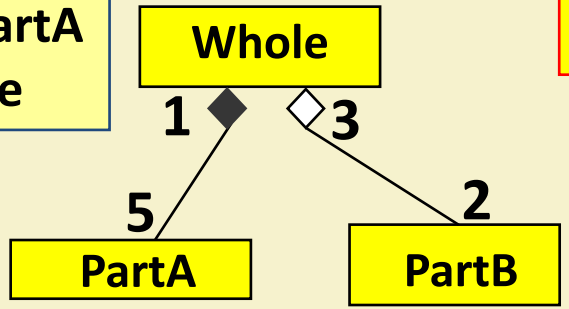


Composition



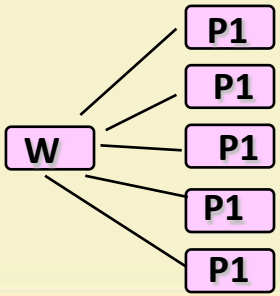
Multiplicity Quiz #1

- One Whole aggregates 5 PartA
- One Part1 is part of 1 Whole

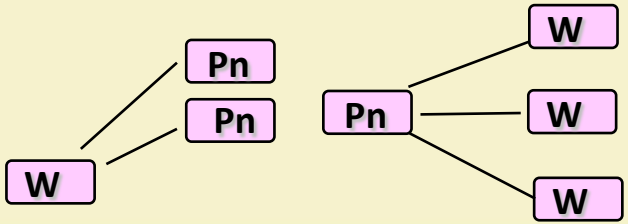


Draw Class diagram

- One Whole aggregates 2 PartB
- One PartB is part of 3 Whole



Object diagram



Class Relation Hints

- **Composition**
  - B is a permanent part of A
  - A contains B
  - A is a permanent collection of Bs
- **Subclass / Superclass**
  - A is a kind of B
  - A is a specialization of B
  - A behaves like B
- **Association (Collaboration)**
  - A delegates to B
  - A needs help from B
  - A and B are peers.

- A common or improper noun implies a class e.g. Book
- A proper noun implies an object (instance of a class): CSE Dept, OOSD, etc.
- An adjective implies an attribute e.g. price of book
- A “doing” verb implies an operation (method)
  - Can also imply a relationship e.g. student issues Book
- A “having” verb implies an aggregation relationship
- A predicate or descriptive verb phrase elaborates an operation e.g. ISBN numbers are integers
- An adverb implies an attribute of an operation e.g. fast loading of image…

Class Diagram Construction Based on Text Analysis

Identify Class Relations

- Faculty & student
- Hospital & doctor
- Door & Car
- Member & Organization
- People & student
- Department & Faculty
- Employee & Faculty
- Computer Peripheral & Printer
- Account & Savings account