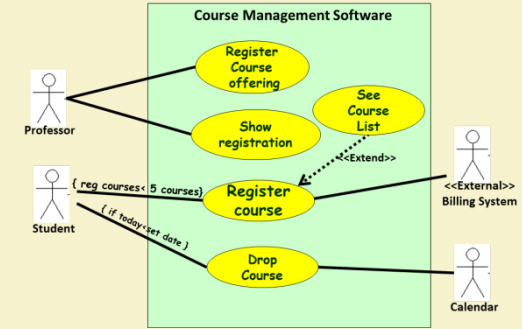


Use Case Modelling

- So far, we have looked into the basic syntax...
- Factoring use cases ...
- Text description ...
- Design of Use Case model from a given text description ...

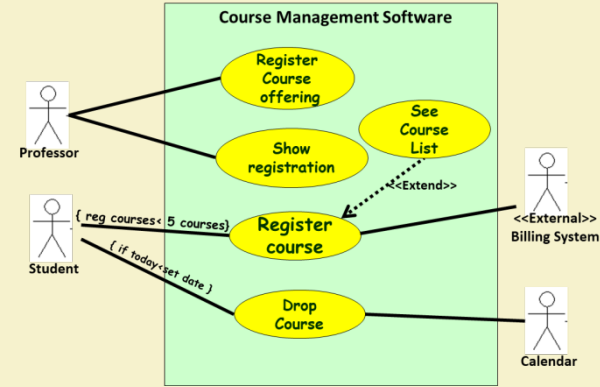
Style Notes (Ambler, 2005)

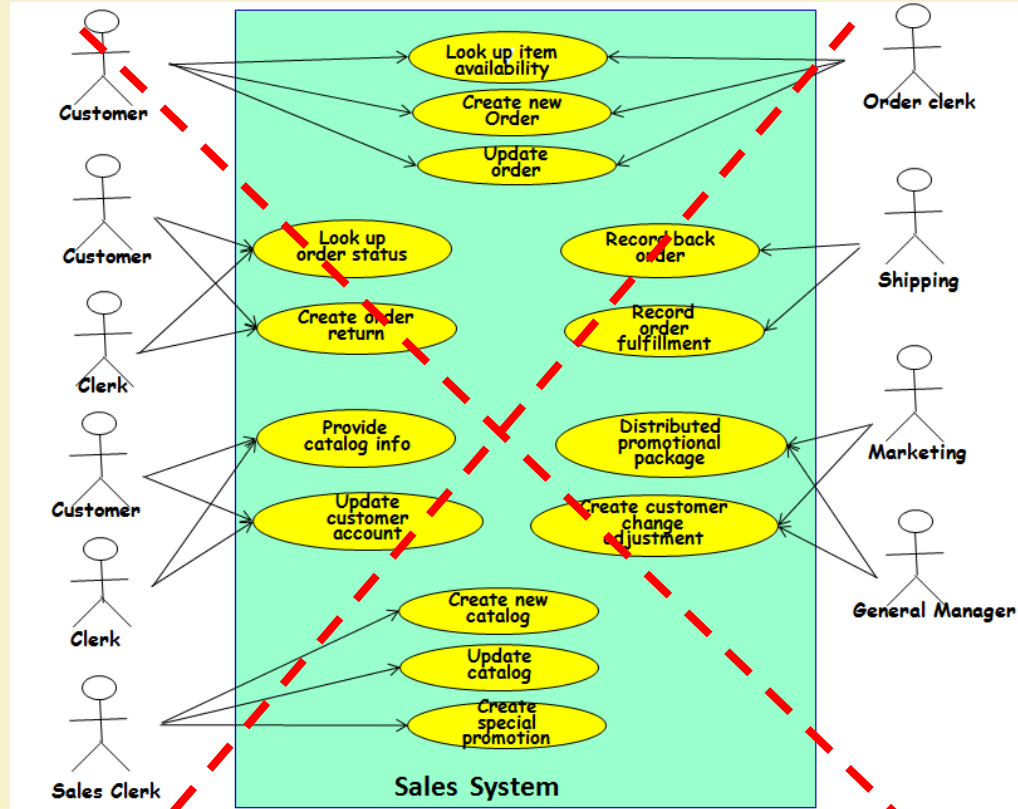
- Use case name should begin with a verb.
- While use cases do not explicitly imply timing:
 - Order use cases from top to bottom to imply timing -- it improves readability.
- **The primary actors should appear in the left.**
- Actors are associated with one or more use cases.
- Do not use arrows on the actor-use case relationship.
- **To initiate scheduled events include an actor called “calendar”**
- **Do not show actors interacting with each other.**
- <<include>> and <<extend>> should rarely nest more than 2 levels deep.



- Use cases should be named and organized from the perspective of the users.
- Use cases should start off simple and at as much higher view as possible.
 - Can be refined and detailed further.
- Use case diagrams represent functionality:
 - Should focus on the "what" and not the "how".

Effective Use Case Modelling





Is it OK?

Too many use cases
at any level should
be avoided!

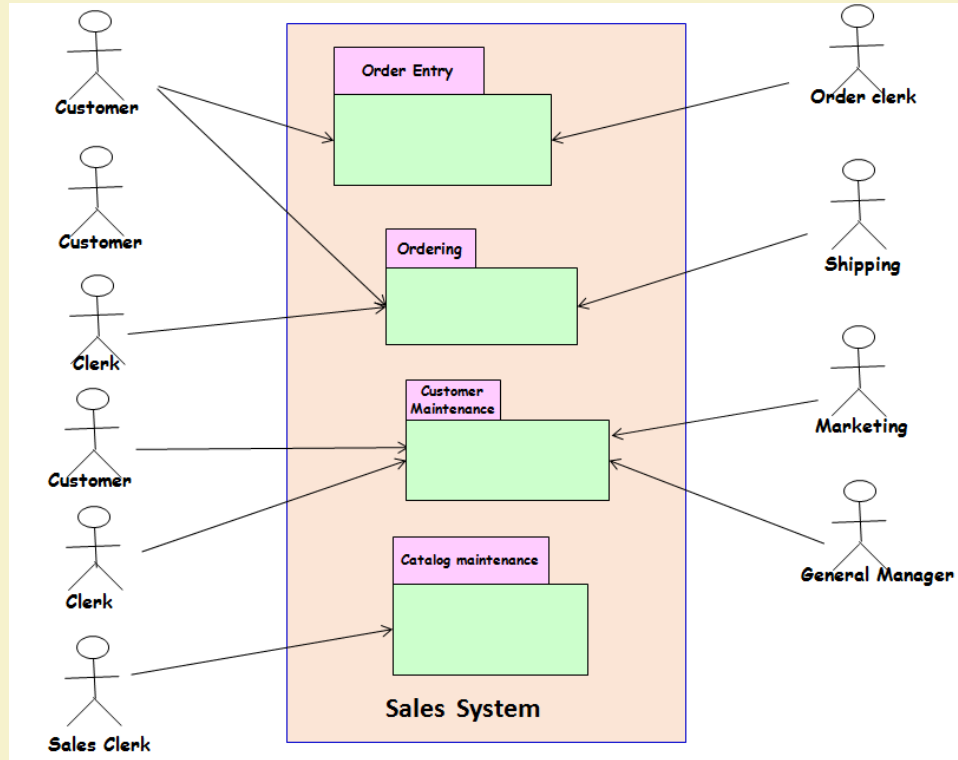
Use Case Packaging

Accounts

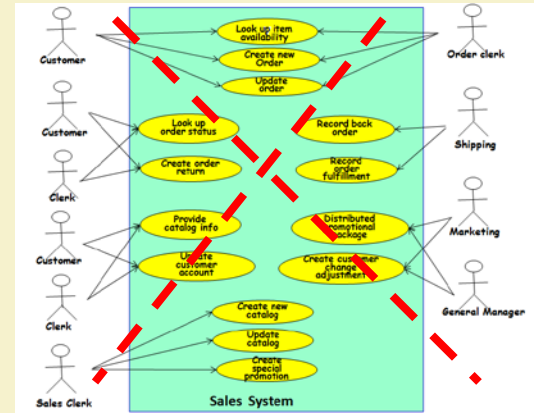
**Print
Balance sheet**

**Receive
grant**

**Make
payments**



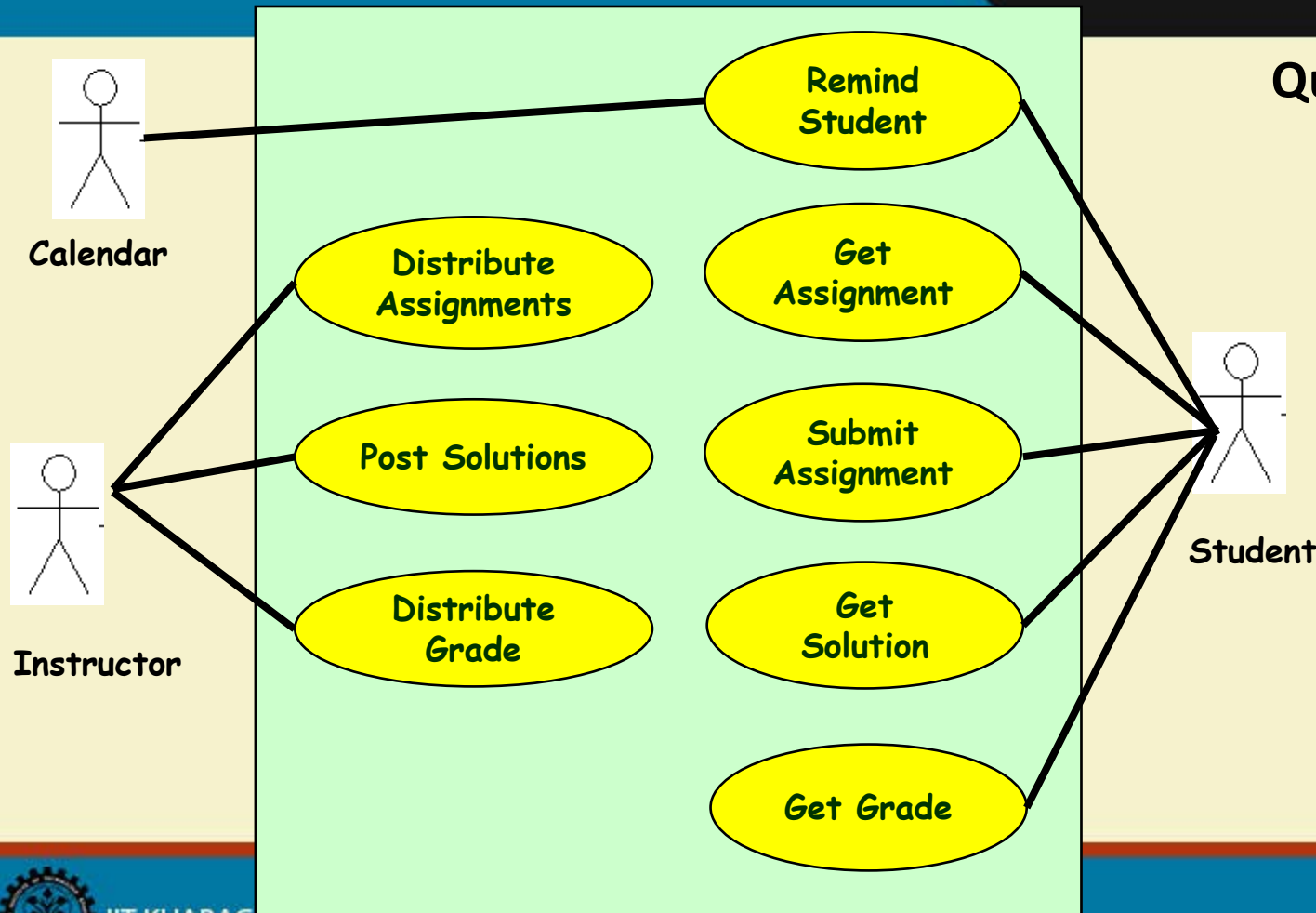
Which is more acceptable?



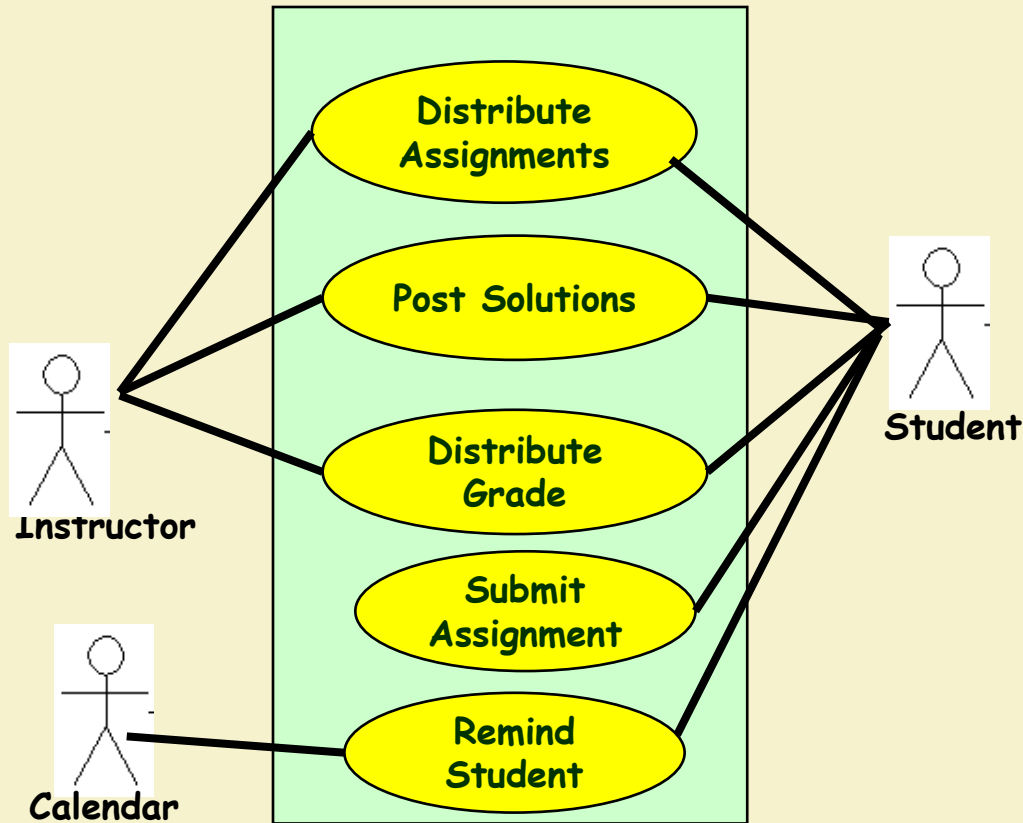
Quiz: Home Assignment System - Use Case Model

- HAS will be used by an instructor to:
 - Distribute homework assignments,
 - Review students' solutions,
 - Distribute suggested solution,
 - Assign a grade to each assignment.
- Students can:
 - Download assignments
 - Submit assignment solutions
- System:
 - Automatically reminds the students a day before an assignment is due.

Quiz: Solution 1 (Inferior)



Quiz: Alternate (Better) Solution



Class Diagram

- Template for object creation:

- Instantiated into objects



**Class: A
Fundamental
Object-Oriented
Concept**

- Examples: Employees, Books, etc.

- Sometimes not intended to produce instances:

- **Abstract classes**

- Entities with common features are made into a class.

Class Diagram

- Represented as solid outline rectangle with compartments.

- Compartments for **name, attributes, and operations.**

Window

- Attribute and operation compartments are optional ... used depending on the purpose of a diagram.

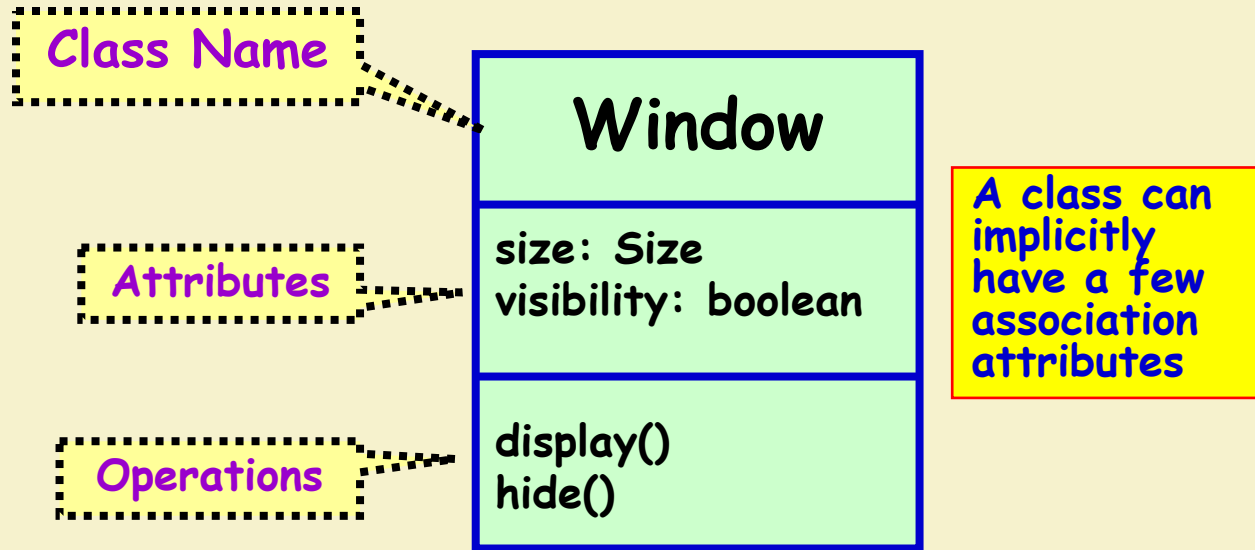
Window

size: Size
visibility: boolean

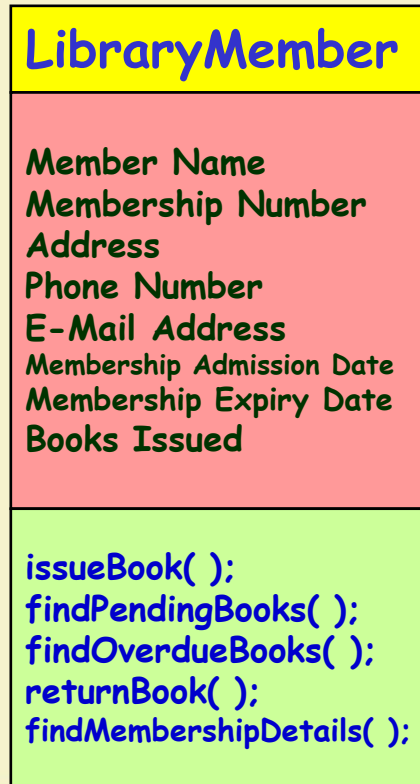
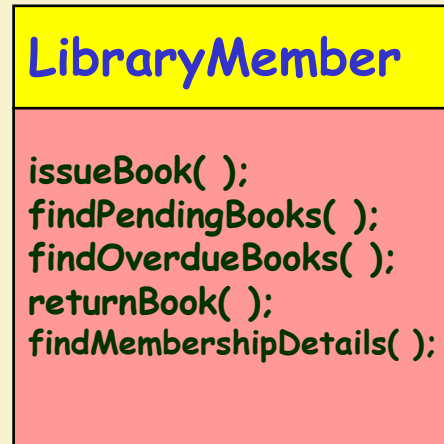
display()
hide()

UML Class Representation

- A class represents a set of objects having similar attributes, operations, relationships and behavior.



Different representations of the LibraryMember class



**Example UML
Classes**

Class Attribute Examples

Java Syntax	UML Syntax
Date birthday	Birthday:Date
Public int duration = 100	+duration:int = 100
Private Student students[0..MAX_Size]	-Students[0..MAX_Size]:Student

Visibilty	Java Syntax	UML Syntax
public	public	+
protected	protected	#
package		~
private	private	-

**Visibility
Syntax in
UML**

Methods vs. Messages

- Methods are the operations supported by an object:
 - Means for manipulating the data of an object.
 - Invoked by sending a message (method call).
 - **Examples:** calculate_salary(), issue-book(), getMemberDetails(), etc.

Method Examples

Java Syntax	UML Syntax
<code>void move(int dx, int dy)</code>	<code>~move(int dx,int dy)</code>
<code>public int getSize()</code>	<code>+int getSize()</code>

Are Methods and Messages Synonyms?

- No
- Message was the original concept in object-orientation...
- Methods are the later simplifications...
- Sometimes used as synonyms

Are Methods and Operations Synonyms?

- No
- An operation can be implemented by multiple methods.
 - Known as polymorphism
 - In the absence of polymorphism--the two terms are used as synonyms.

What are the Different Types of Relationships Among Classes?

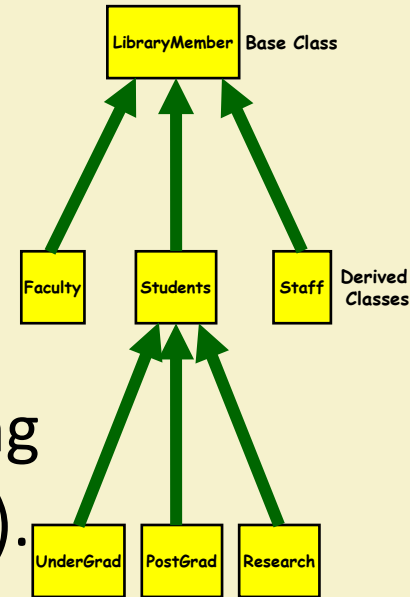
- Four types:
 - Inheritance
 - Association
 - Aggregation/Composition
 - Dependency

Inheritance

- Allows to define a new class (derived class) by extending an existing class (base class).

—Represents generalization-specialization relation.

—Allows redefinition of the existing methods (method overriding).

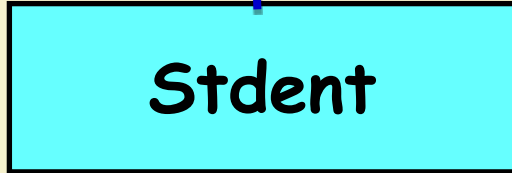


Inheritance One More Example



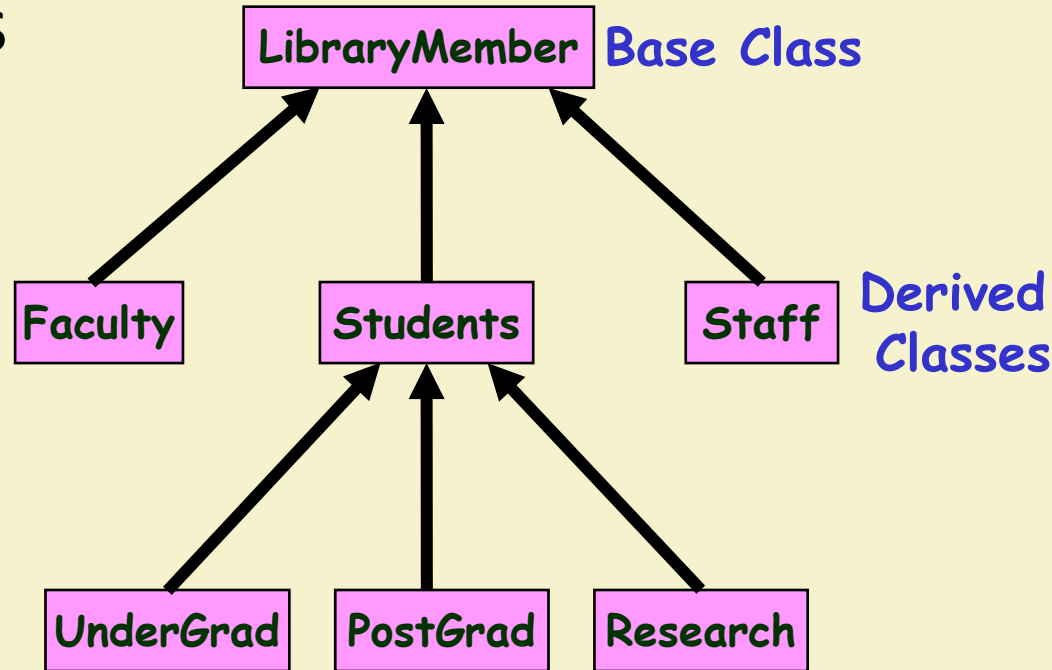
"A Student ISA Library Member"

"A Faculty ISA Library Member"

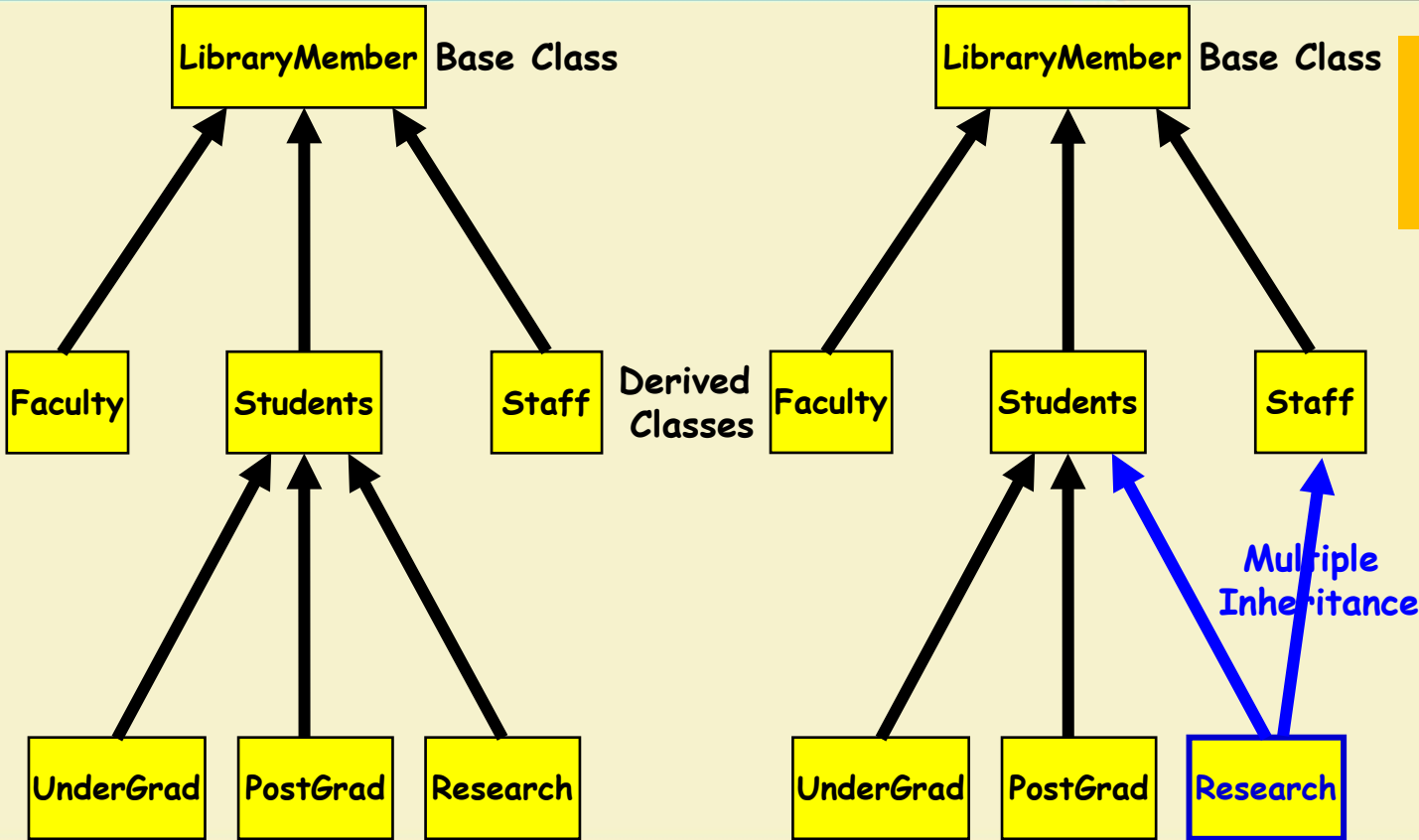


Inheritance: Semantics

- Lets a subclass inherit attributes and methods from a base class.



Multiple Inheritance



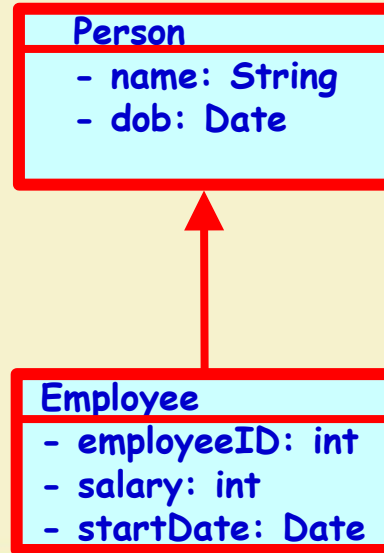
Inheritance Implementation in Java

- Inheritance is declared using the "extends" keyword
 - Even when no inheritance defined, the class implicitly extends a class called Object.

```
class Person{  
    private String name;  
    private Date dob;  
    ...  
}
```

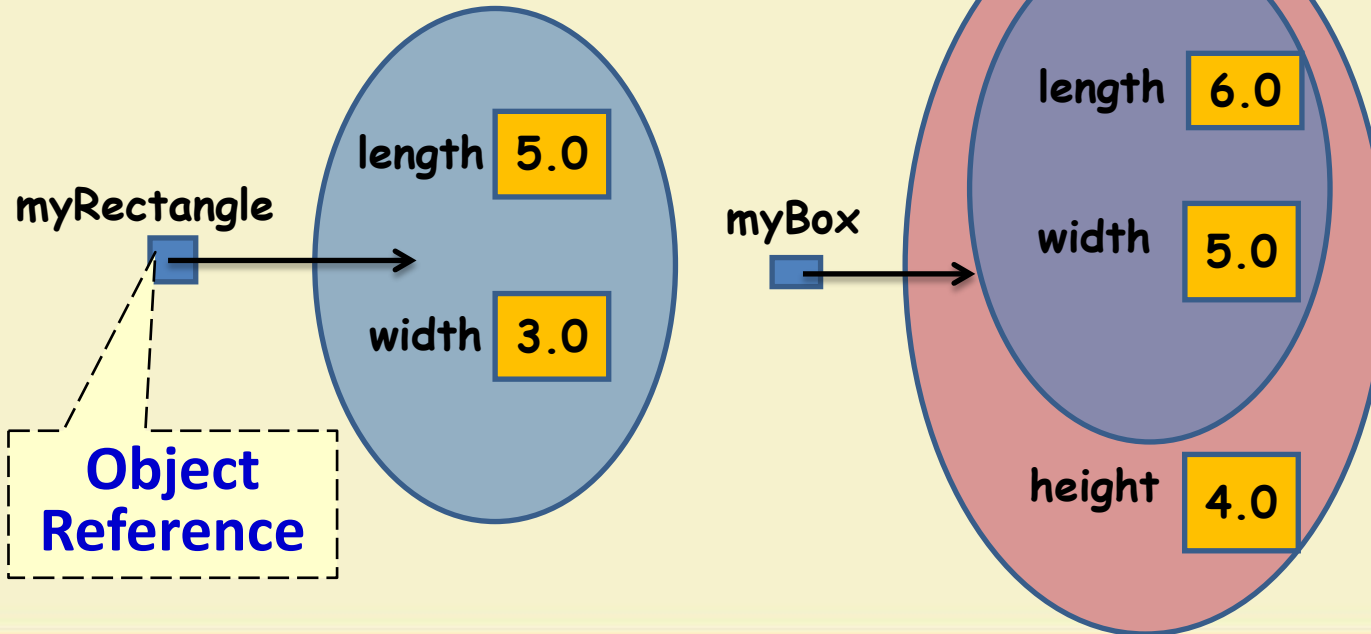
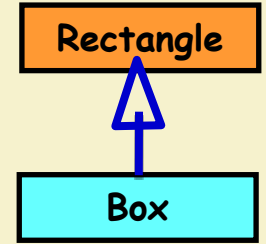
```
class Employee extends Person{  
    private int employeeID;  
    private int salary;  
    private Date startDate;  
    ...  
}
```

```
Employee anEmployee = new Employee();
```

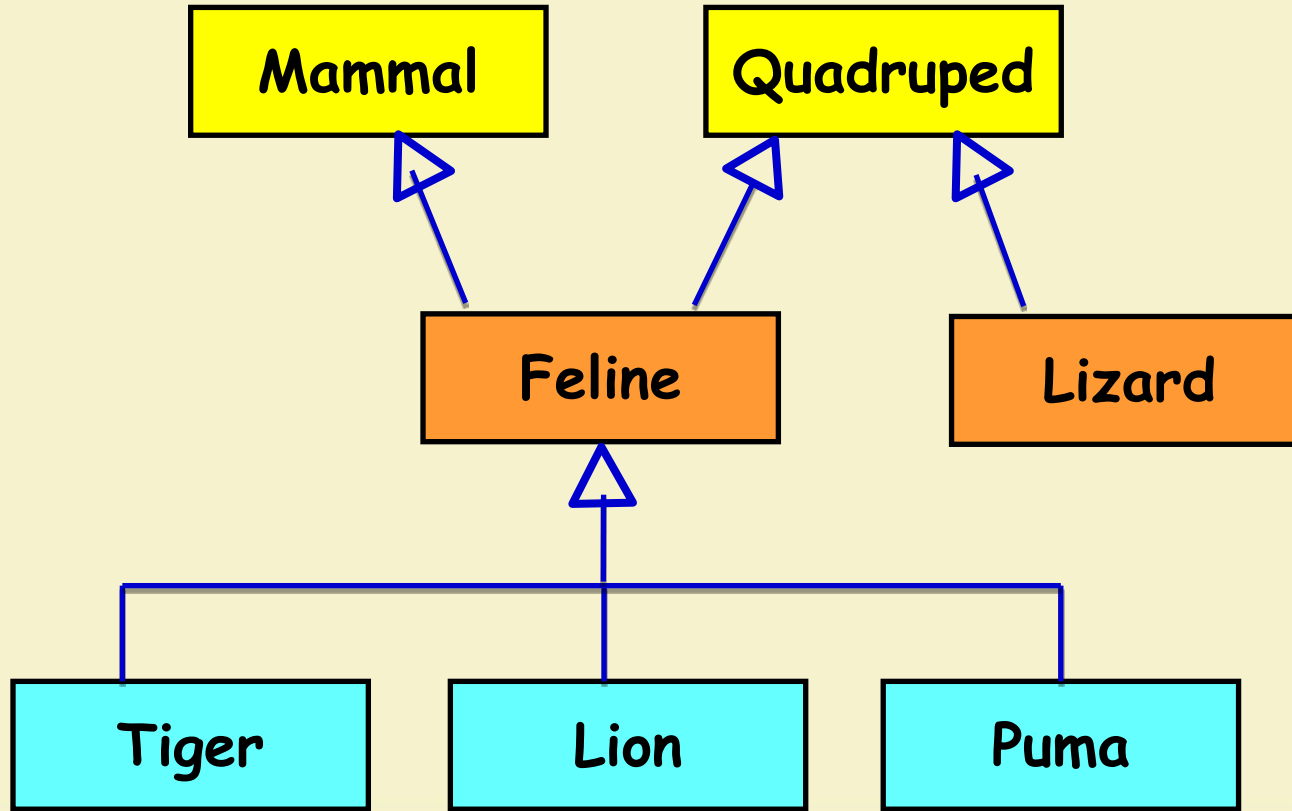


```
Rectangle myRectangle = new Rectangle(5, 3);  
Box myBox = new Box(6, 5, 4);
```

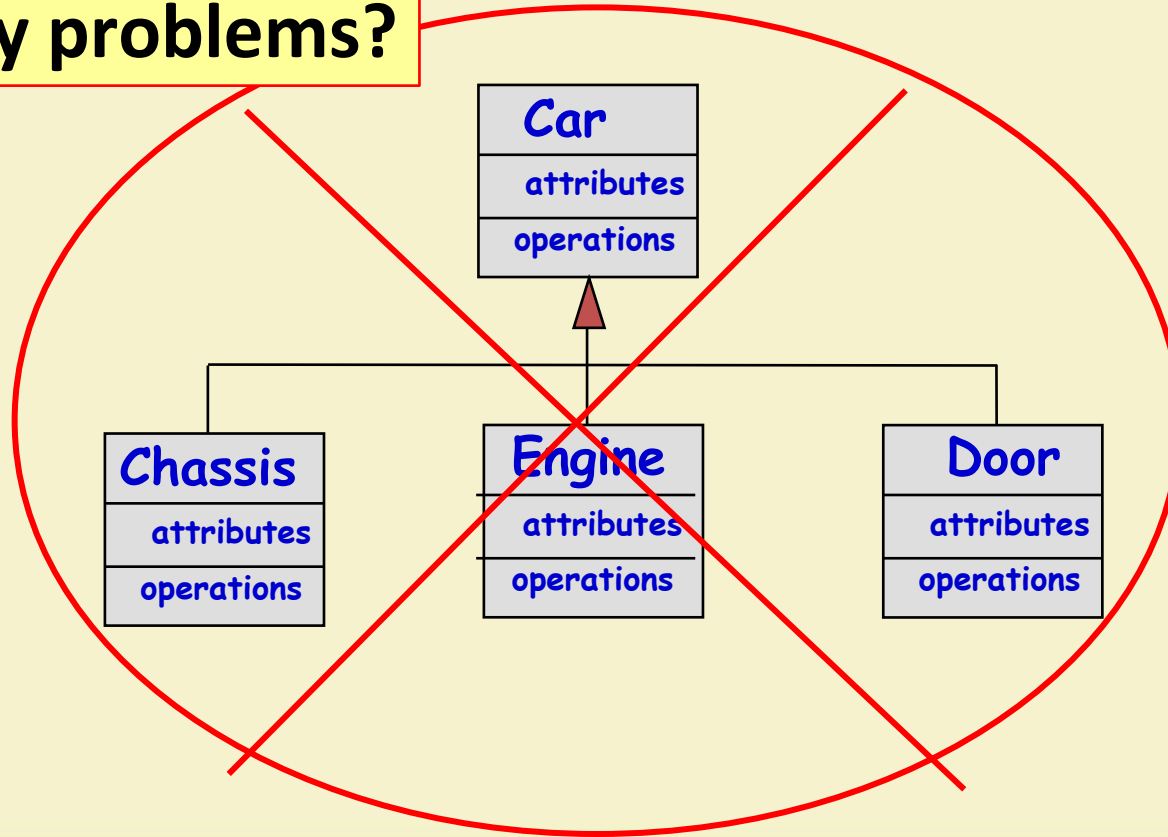
Objects myRectangle and myBox



More Generalization Examples...

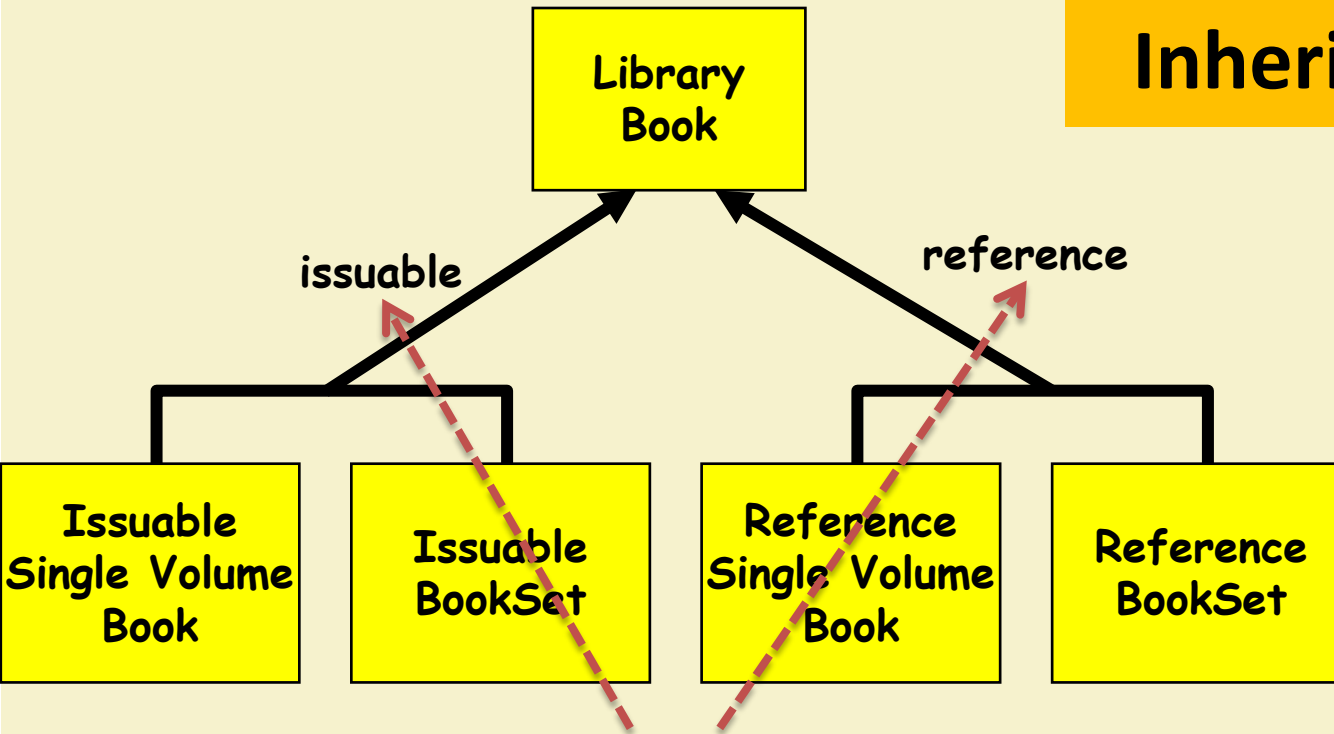


Any problems?



Wrong
Generalization -
--
violates "is a" or
"is a kind of"
heuristic

Inheritance Example



Discriminator: allows one to group subclasses into clusters that correspond to a semantic category.

Inheritance Pitfalls

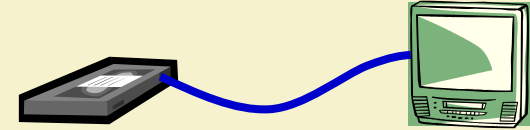
- Inheritance certainly promotes reuse.
- **Indiscriminate use can result in poor quality programs.**
- Base class attributes and methods visible in derived class...
 - Leads to tight coupling

Association Relationship

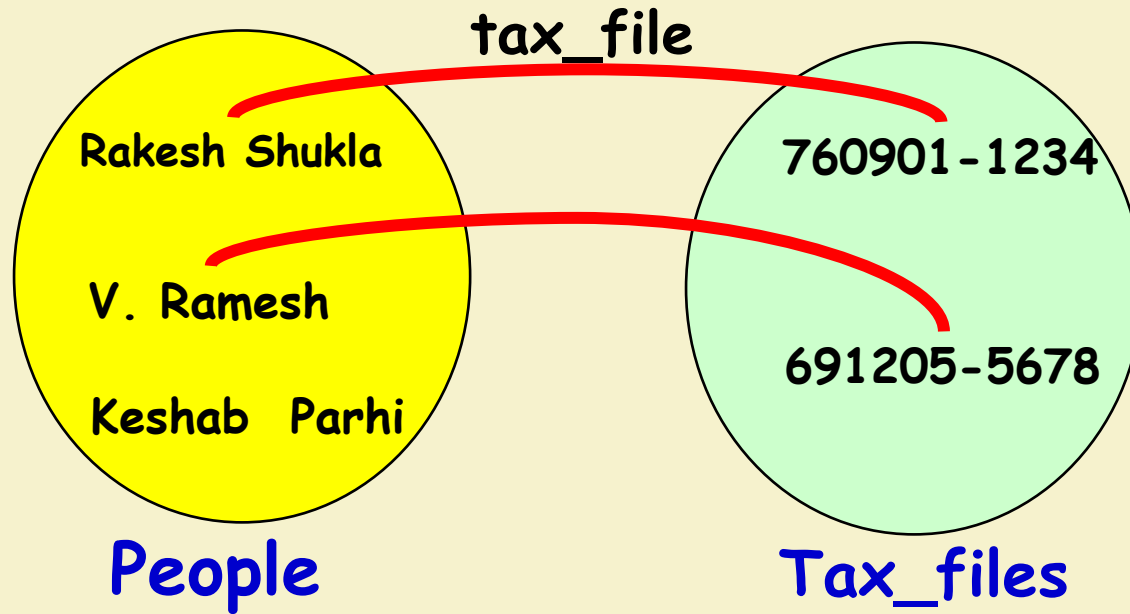
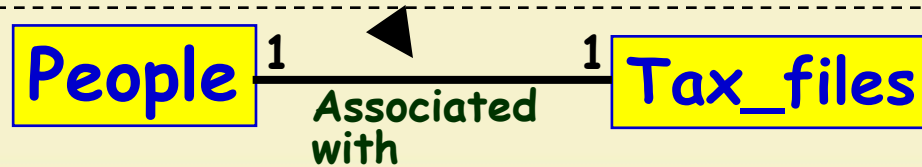
- How implemented in program?
- Enables objects to communicate with each other:
 - One object must “know” the ID of the corresponding object in the association.
- Usually binary:
 - But in general can be n-ary.

Association – An Example

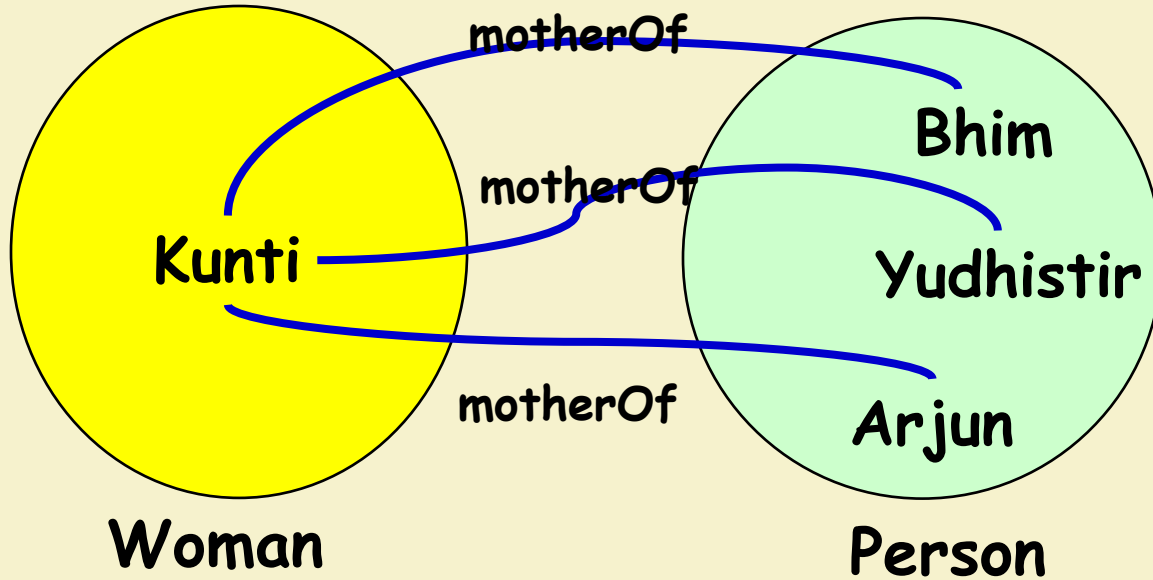
- In a home theatre system,
 - A TV object is associated with a VCR object
 - It may receive a signal from the VCR
 - VCR may be associated with remote
 - It may receive a command to record



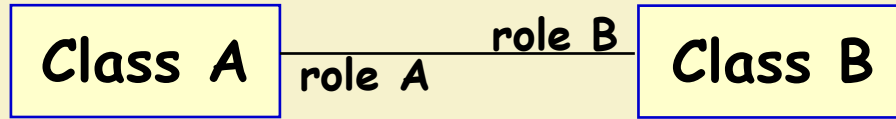
1-1 Association - example



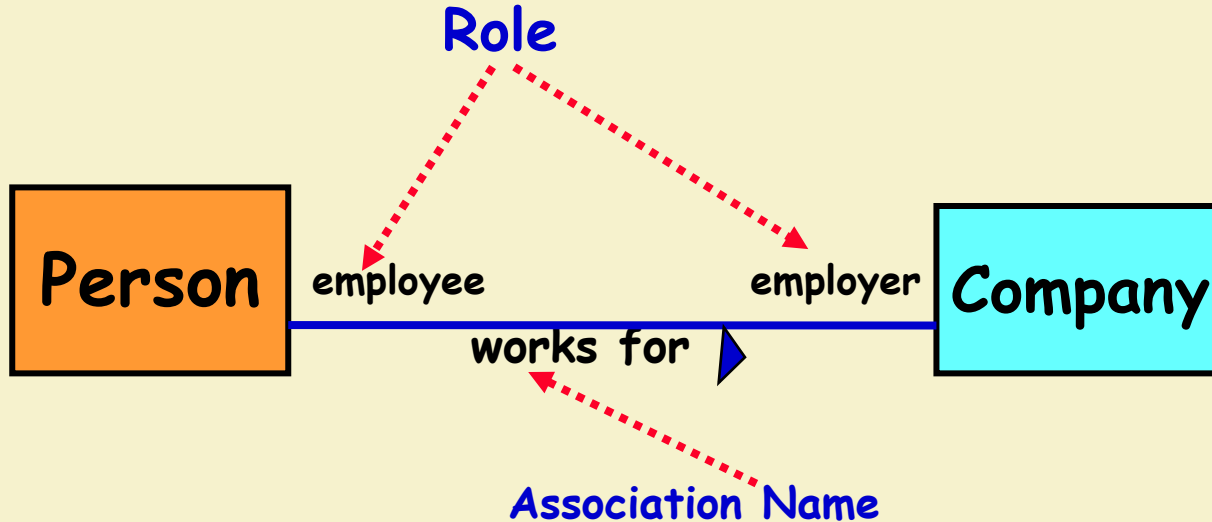
Multiple Association - example



Association UML Syntax

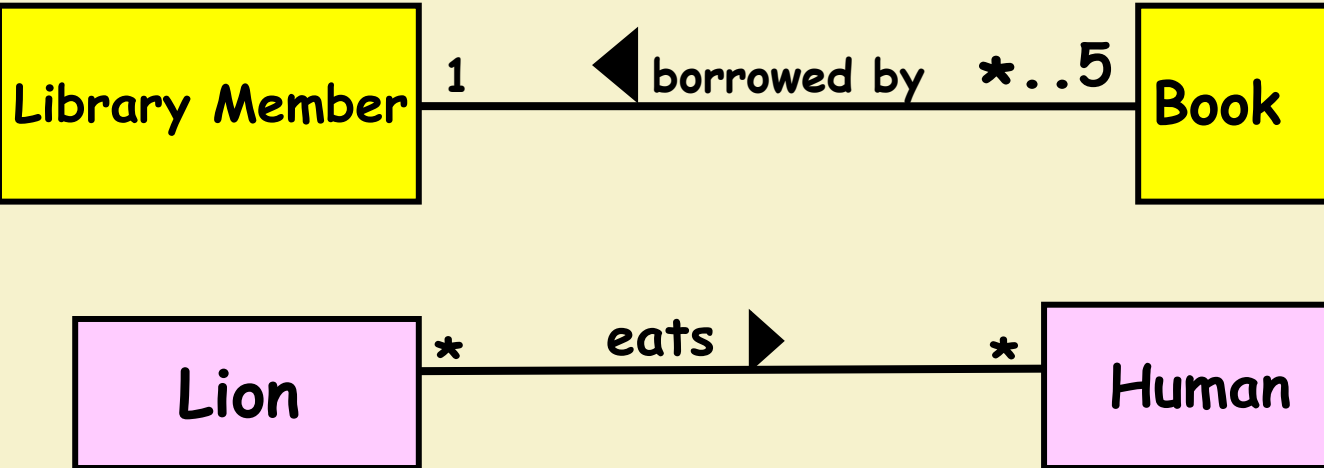


- A Person works for a Company.

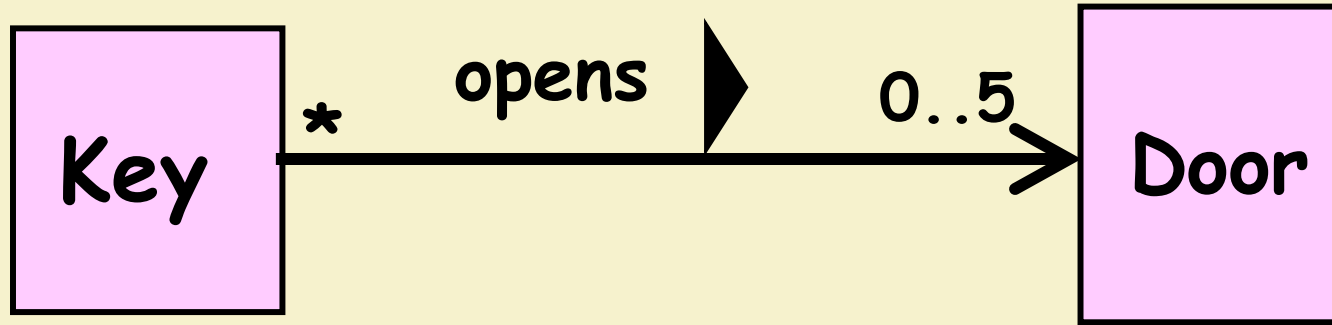


Multiplicity: The number of objects from one class that relate with a single object in an associated class.

Association - More Examples

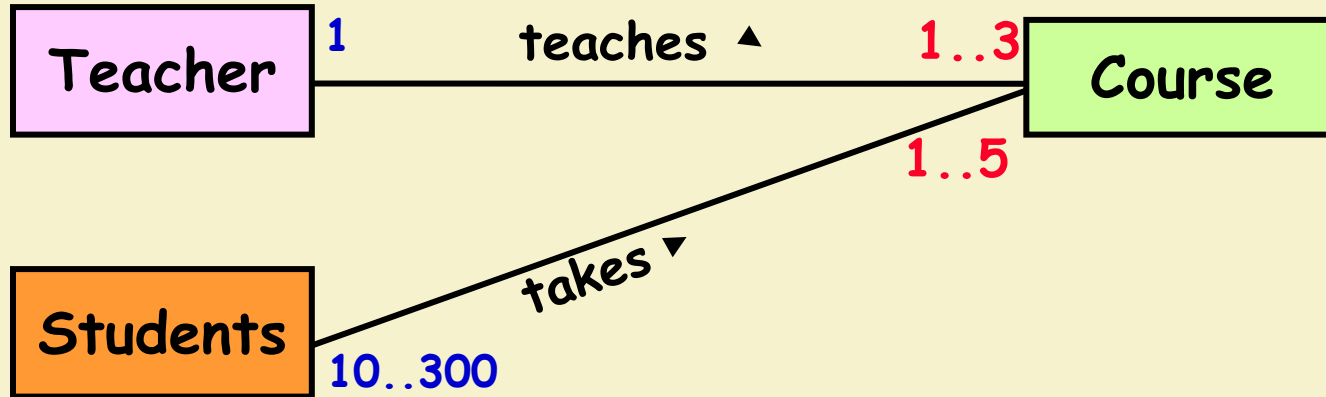


Navigability



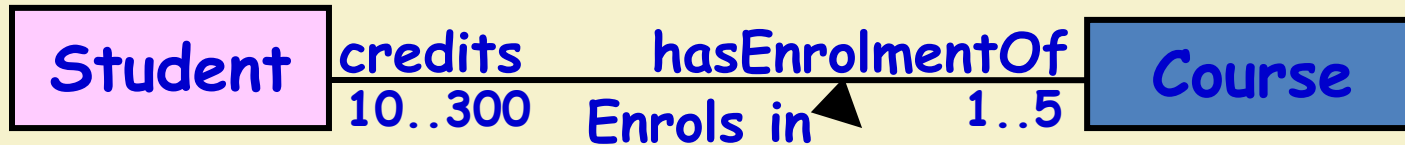
Association - Quiz 1

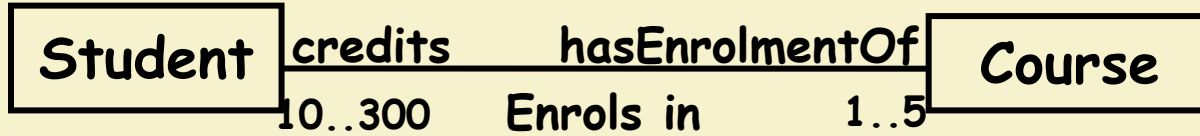
- A teacher teaches 1 to 3 courses (subjects)
- Each course is taught by only one teacher.
- A student can take between 1 to 5 courses.
- A course can have 10 to 300 students. Draw the class diagram.



Quiz 2: Draw Class Diagram

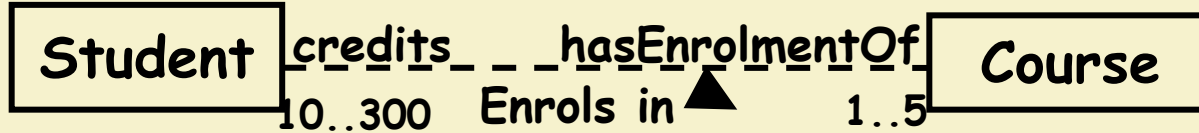
- A Student can take up to five Courses.
- A student needs to enroll in at least one course.
- Up to 300 students can enroll in a course.
- An offered subject in a semester should have at least 10 registered students.



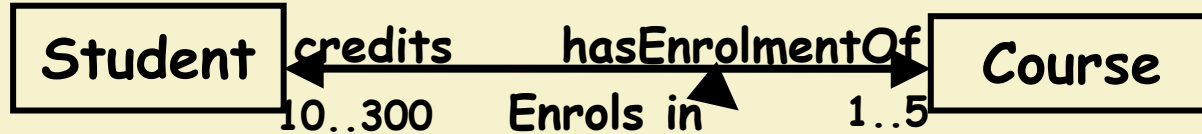


A

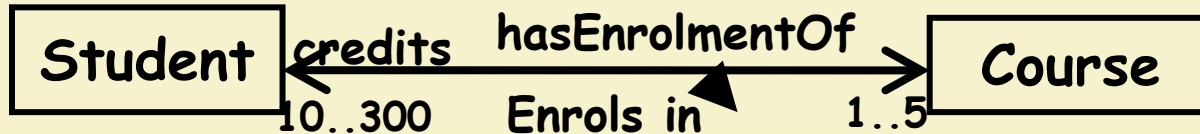
Identify as
Correct or
Wrong



B

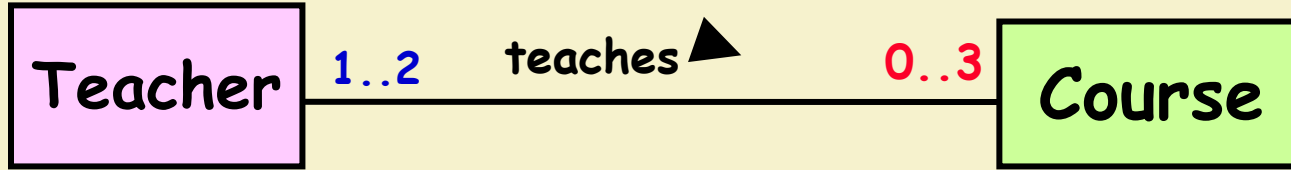
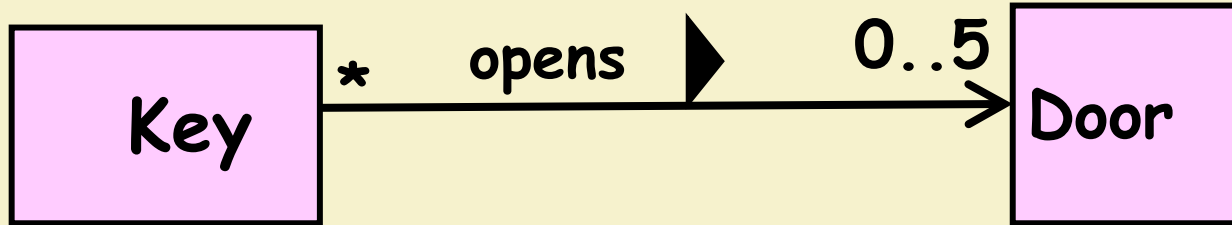


C



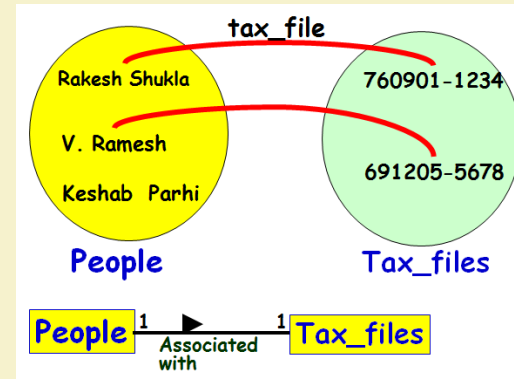
D

Quiz: Read the Diagram



- **A link:**
 - An instance of an association
 - Exists between two or more objects
 - **Dynamically created and destroyed as the run of a system proceeds**
- For example:
 - An employee joins an organization.
 - Leaves that organization and joins a new organization.

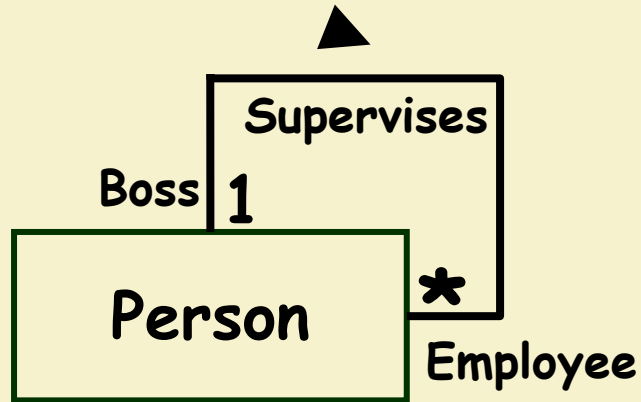
Association and Link



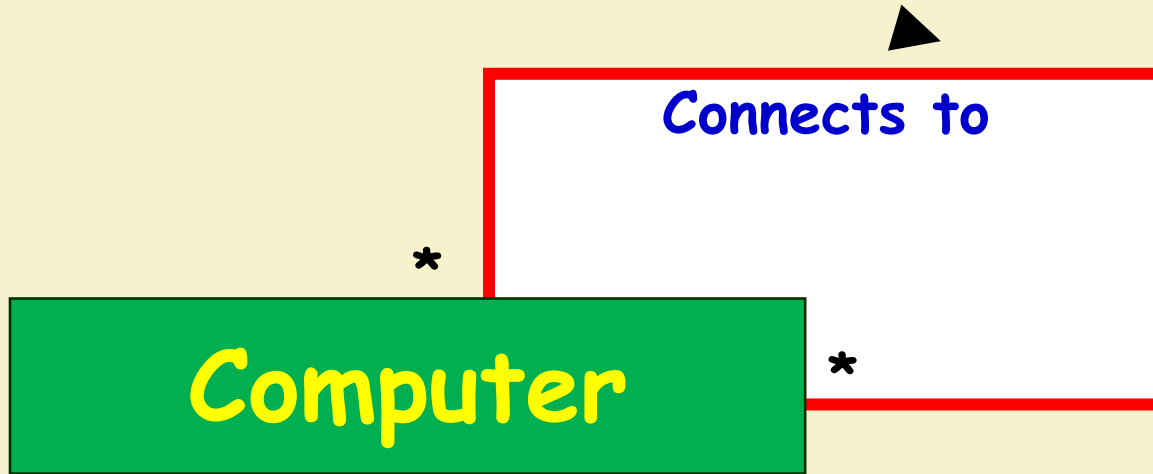
Unary Association

- A class can be associated with itself (**unary** association).
 - **Give an example?**
- An arrowhead used along with name:
 - Indicates direction of association.
- Multiplicity (association cardinality) indicates # of instances taking part in the association.

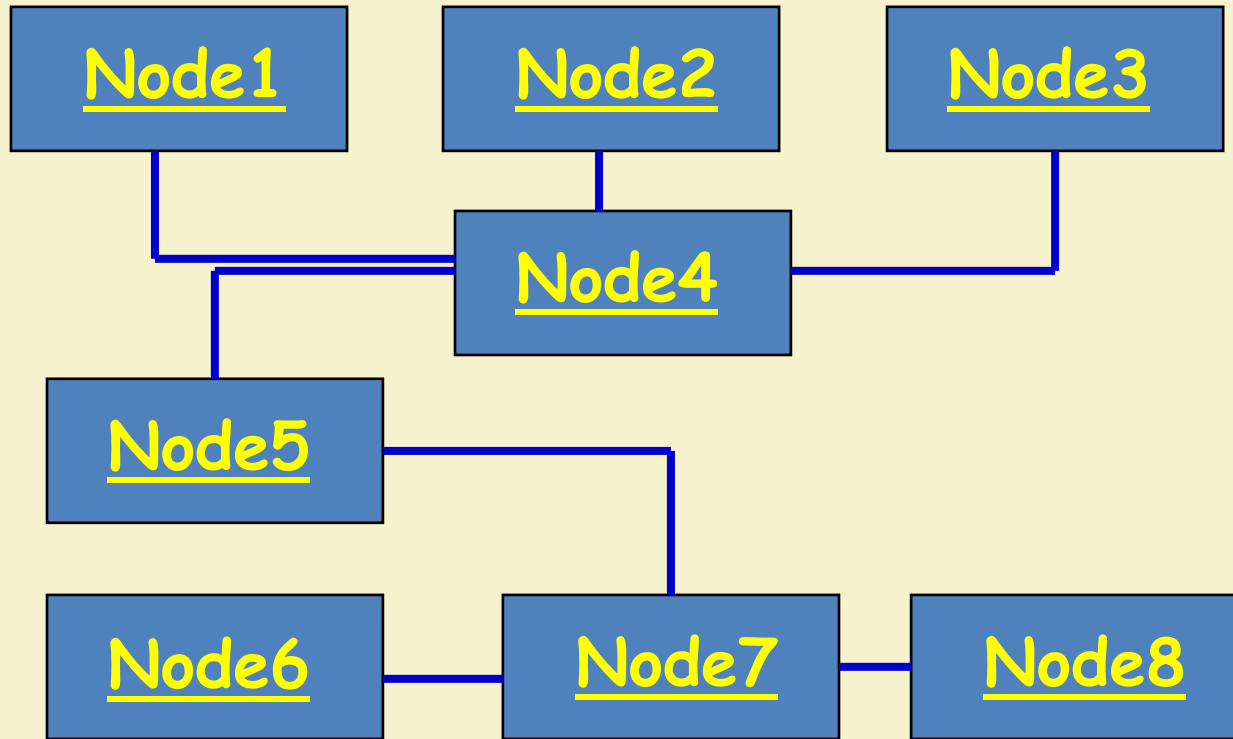
Unary Association: Example 1



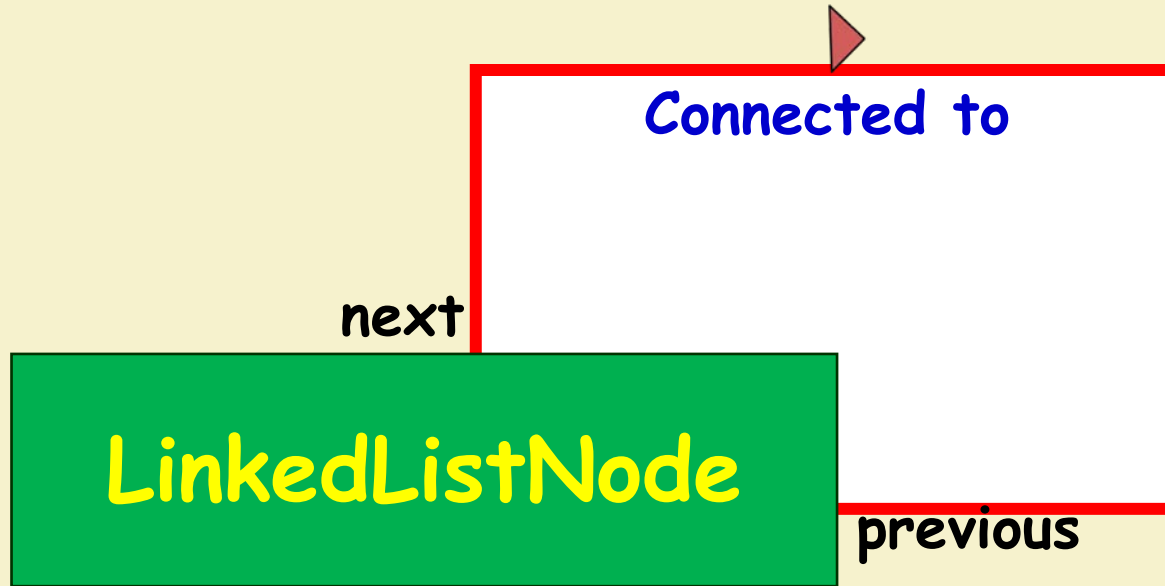
Self Association: Example 2 Computer Network



Computer Network: Object Diagram



Self Association: Example 3



Reflexive Association: Example 4

