

CONCEPTUAL DESIGN: UML CLASS DIAGRAM RELATIONSHIPS



A Simplified Object-Oriented Systems Analysis & Conceptual Design Methodology

Activities

1. Identify the information system's purpose
2. Identify the information system's actors and features
3. Identify Use Cases and create a Use Case Diagram
- ➡ 4. Identify Objects and their Classes and create a Class Diagram
5. Create Interaction/Scenario Diagrams
6. Create Detail Logic for Operations
7. Repeat activities 1-6 as required to refine the "blueprints"

Objects

- ***Objects have three responsibilities:***



What they know about themselves – (e.g., Attributes)



What they do – (e.g., Operations)



What they know about other objects – (e.g., Relationships)

Defining Class

A CLASS is a template (specification, blueprint) for a collection of objects that share a common set of attributes and operations.

Class



<i>HealthClubMember</i>
<i>attributes</i>
<i>operations</i>

Objects



• ***Relationships***

A RELATIONSHIP is what a class or an object knows about another class or object.

Four Types



Generalization (Class-to-Class) (Superclass/Subclass)

- ***Inheritance***
- ***Ex: Person - FacultyPerson, StudentPerson, Staff...***
- ***Ex: ModesOfTravel - Airplane, Train, Auto, Cycle, Boat...***



[Object] Associations

- ***FacultyInformation - CourseInformation***
- ***StudentInformation - CourseInformation***



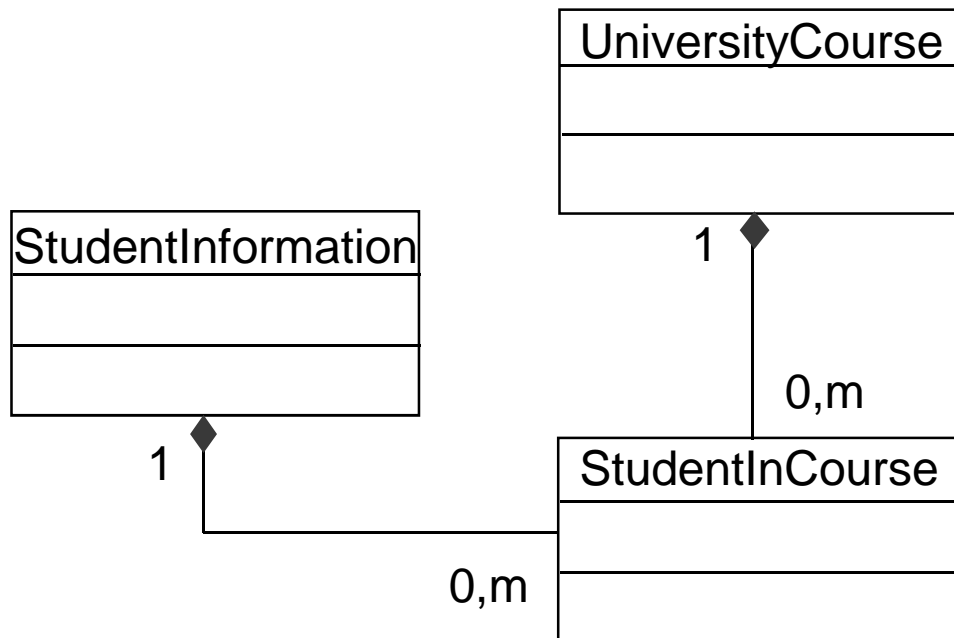
[Object] Aggregations & Composition (Whole-Part)

- ***Assembly - Parts***
- ***Group - Members***
- ***Container - Contents***

• *Relationships*

Exist to:

1) show relationships 2) enforce integrity 3) help produce results



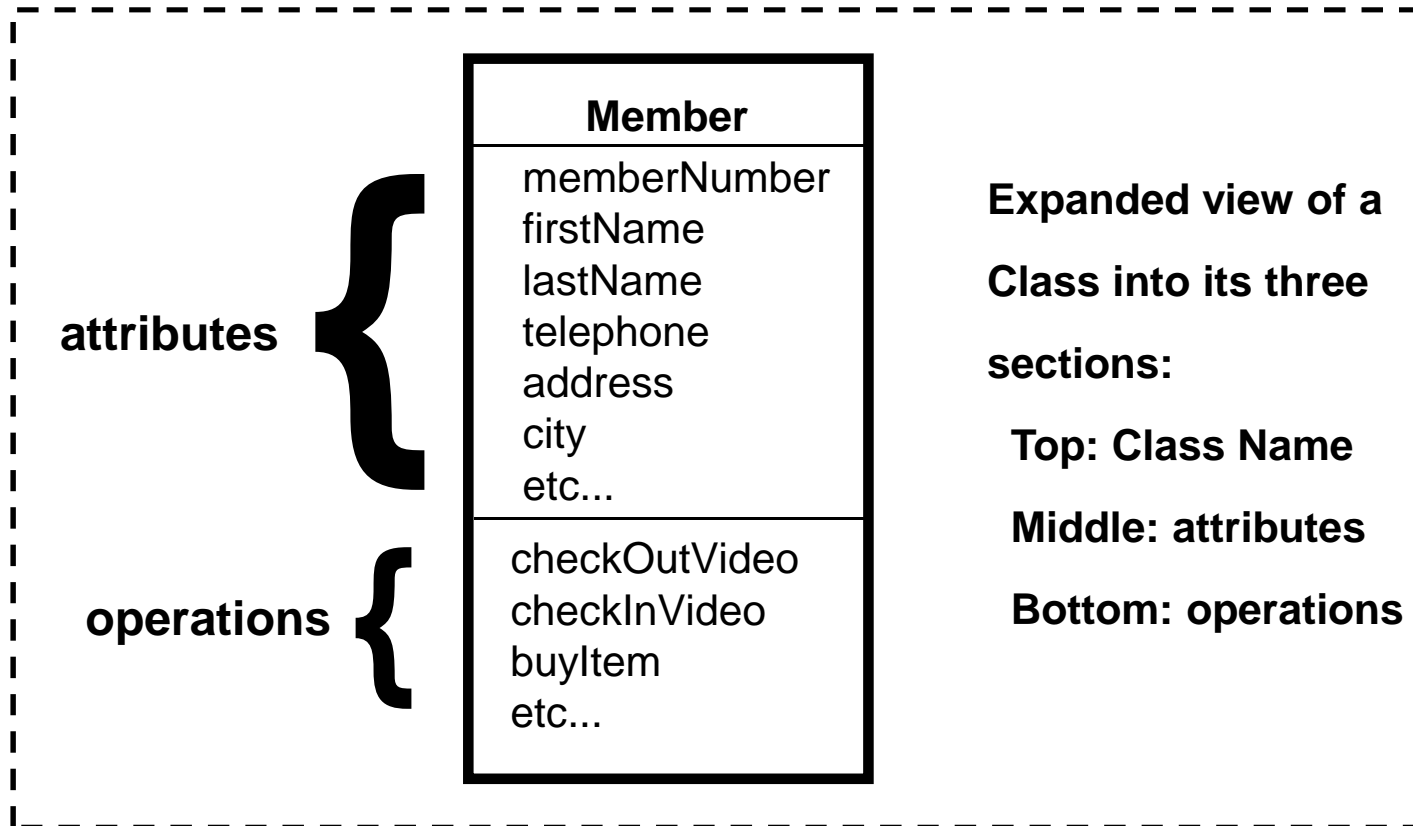
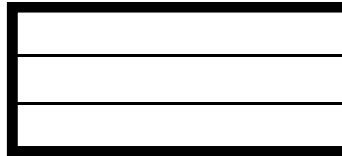
In this example:

- ***Removal of a University Course should also remove Students that are in the Course but not Student Information.***
- ***Removal of a Student should also remove the Courses that the Student is in but not the University Course.***
- ***Removal of a Student in a Course should not affect either University Course or Student Information.***

UML Class Diagram Notation

1 of 2

Class



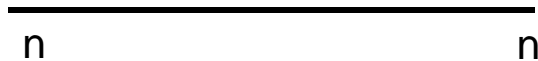
UML Class Diagram Notation

2 of 2

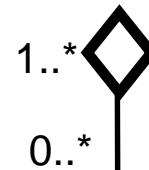
Class
Generalization
Relationship



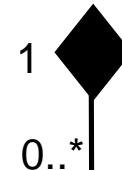
Object Association



Object
Aggregation
Association



Object
Composition
Association



Will always be "1"



Class Diagram Relationships

- ***Class***
 - ***Generalization***
- ***Object***
 - ***Association***
 - ***Aggregation***
 - ***Composition***



Generalization (Class-to-Class) (superclass – subclass; supertype – subtype)

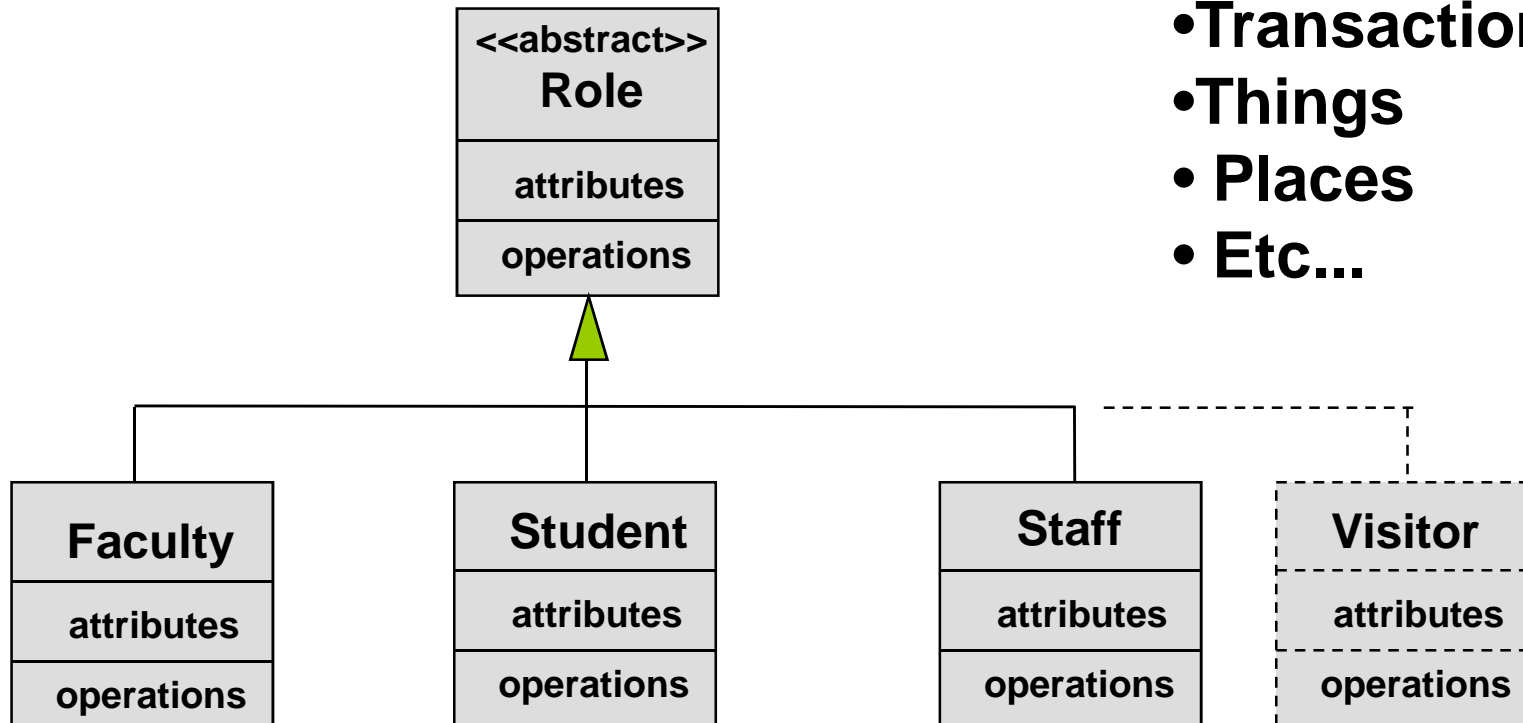
- **A Generalization follows a “is a” or “is a kind of” heuristic from a specialization class to the generalization class. (e.g., student “is a” person, video “is a kind of” inventory).**
- **Common attributes, operations and relationships are located in the generalization class and are inherited by the specialization classes**
- **Unique attributes, operations and relationships are located in the specialization classes.**
- **Inherited attributes and operations may be overridden or enhanced in the specialization class depending on programming language support.**
- **Inherited operations in the specialization classes may be polymorphic.**
- **Only use when objects do NOT “transmute” (add, copy, delete)**
- **Multiple inheritance is allowed in the UML but can complicate the class model’s understanding and implementation (e.g., C++ supports but Java and Smalltalk do not).**



Generalization Example

Others:

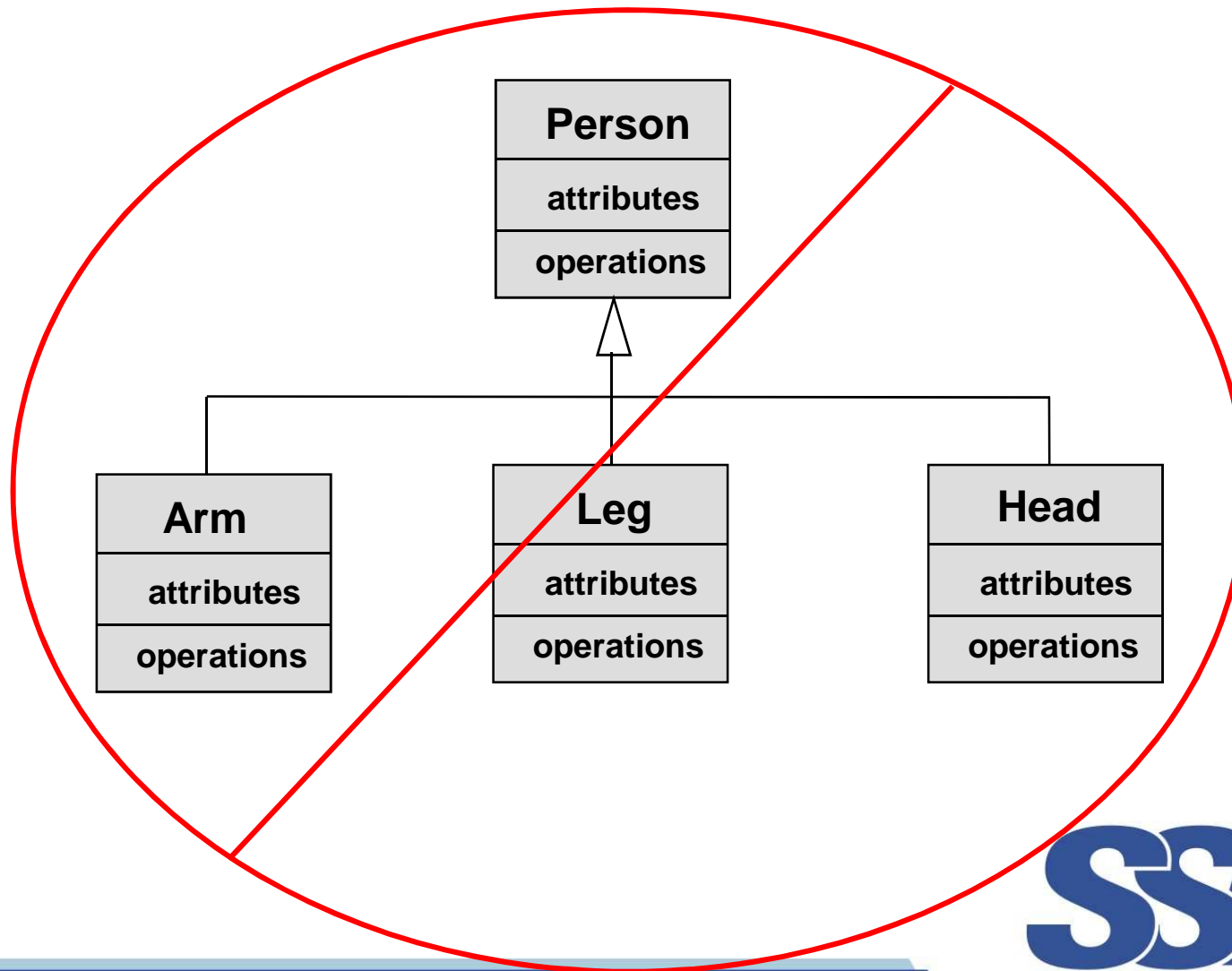
- Transactions
- Things
- Places
- Etc...



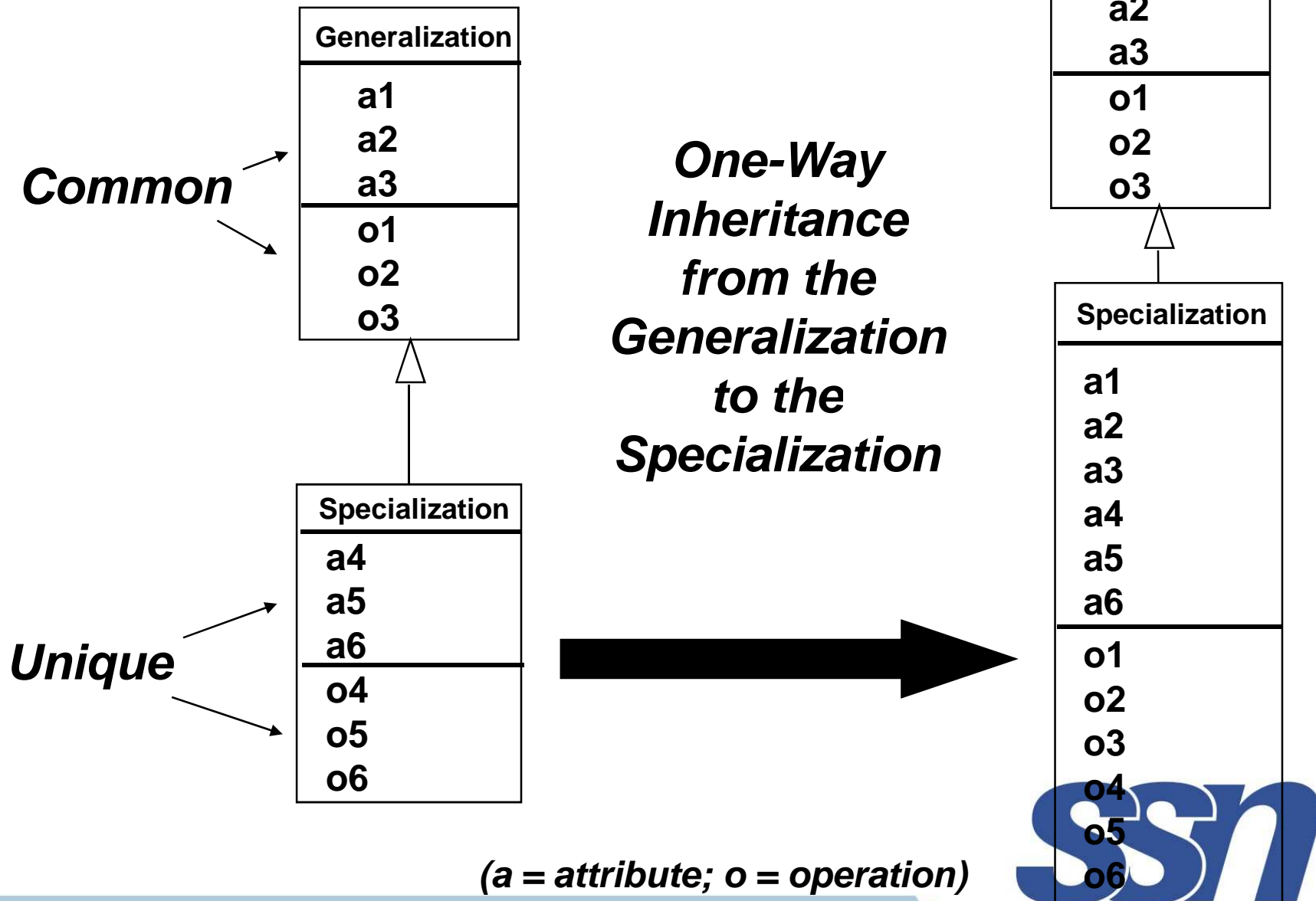
Note: <<abstract>> = no objects



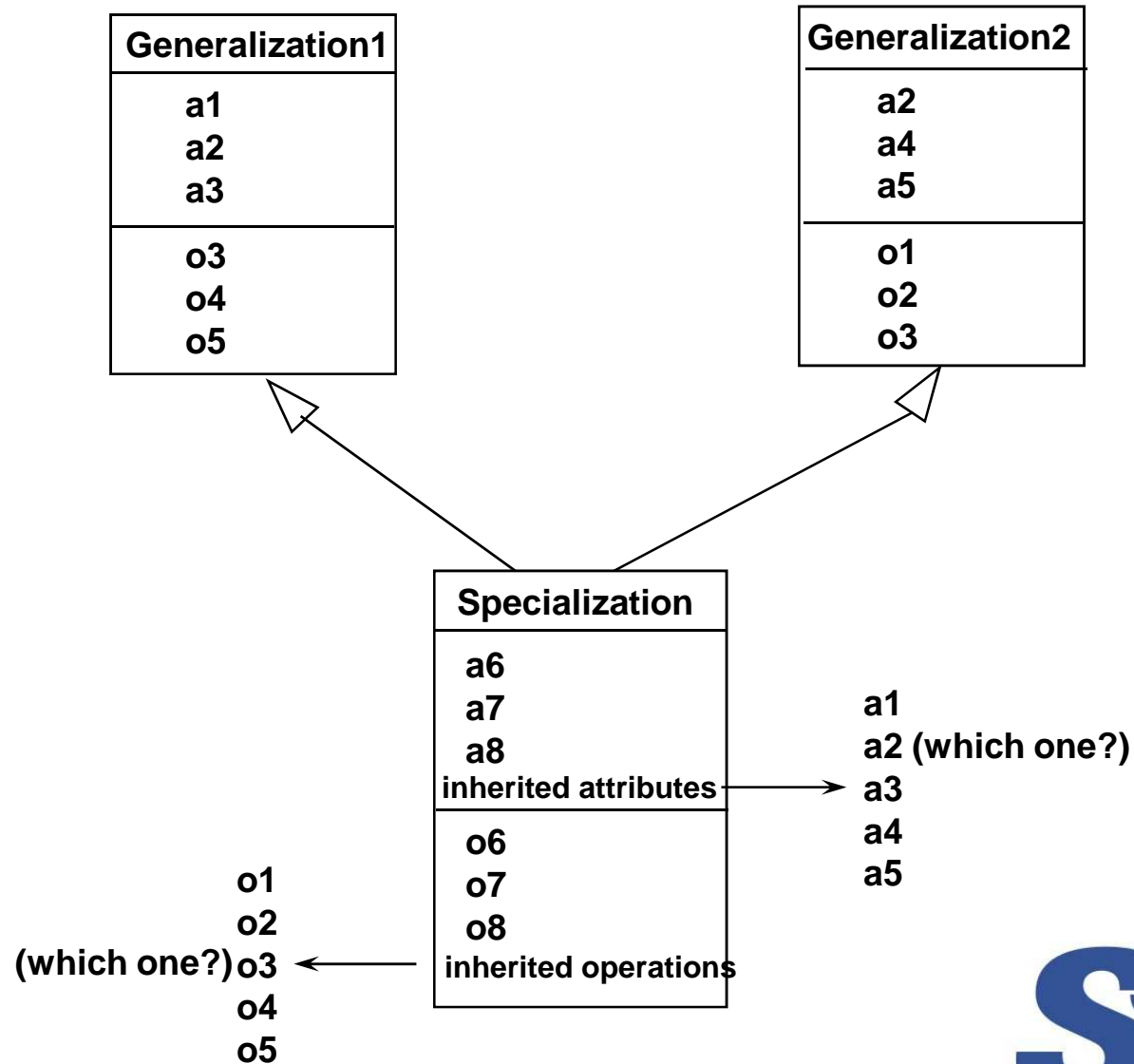
Poor Generalization Example ***(violates the “is a” or “is a kind of” heuristic)***



Generalization Inheritance

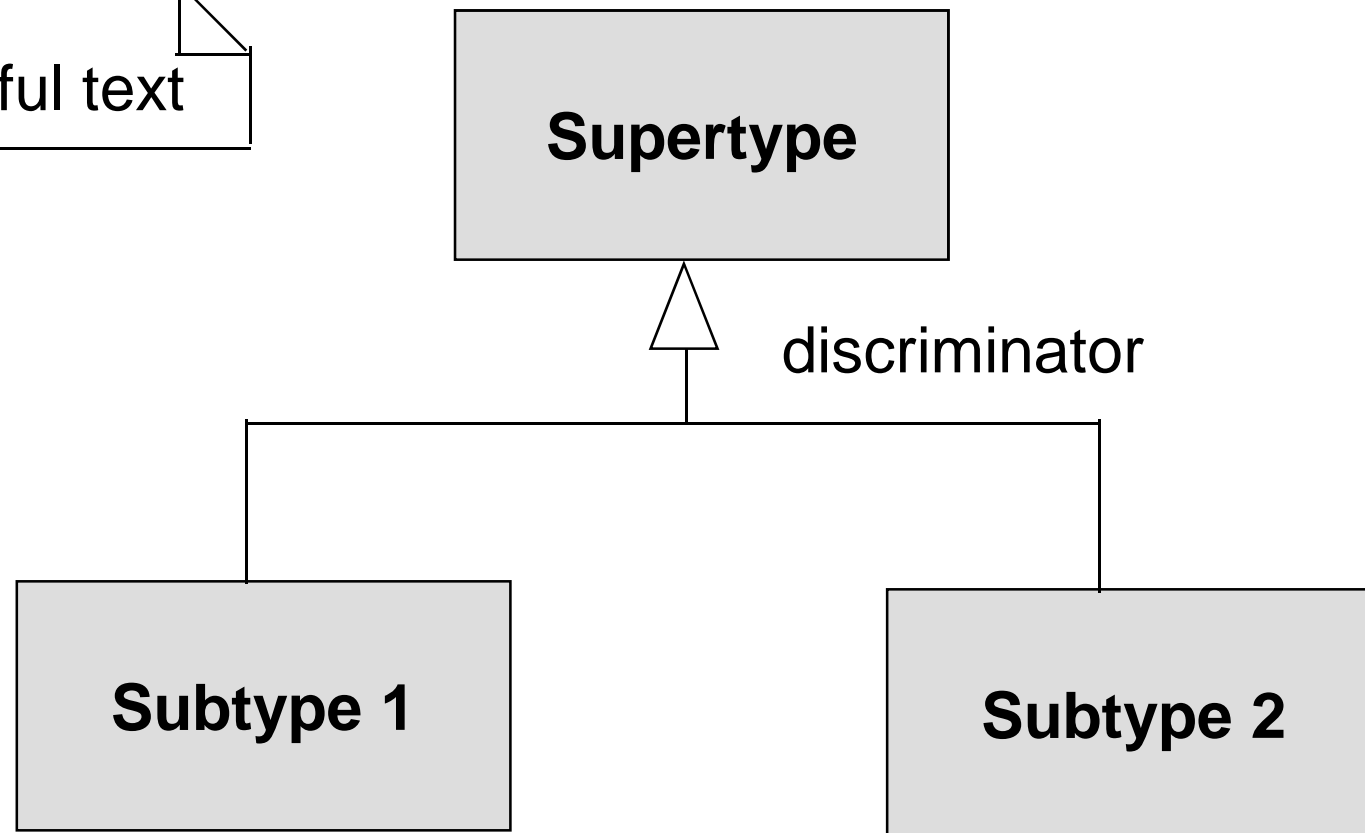


Generalization - Multiple Inheritance



UML Generalization Notation

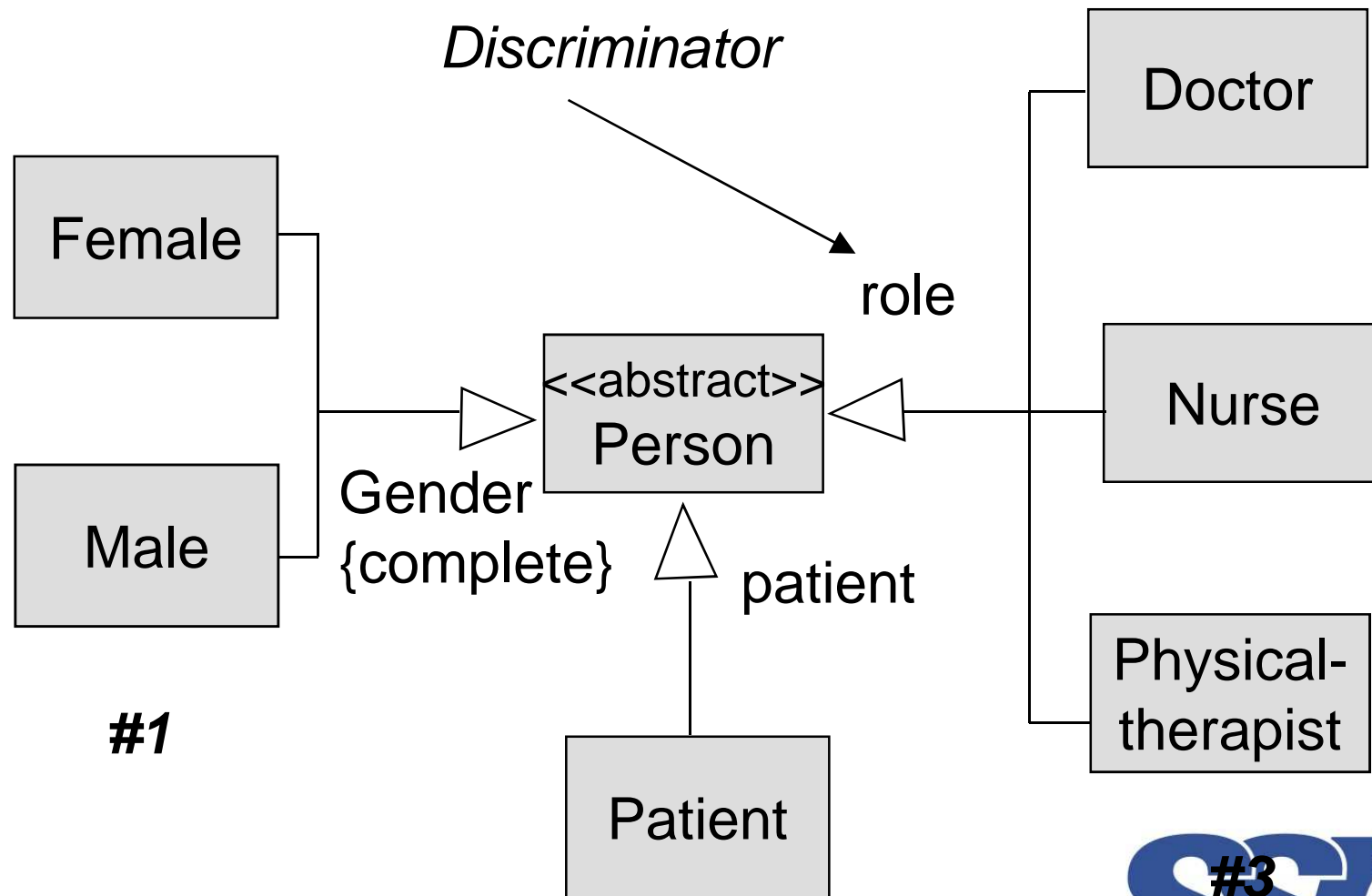
Note

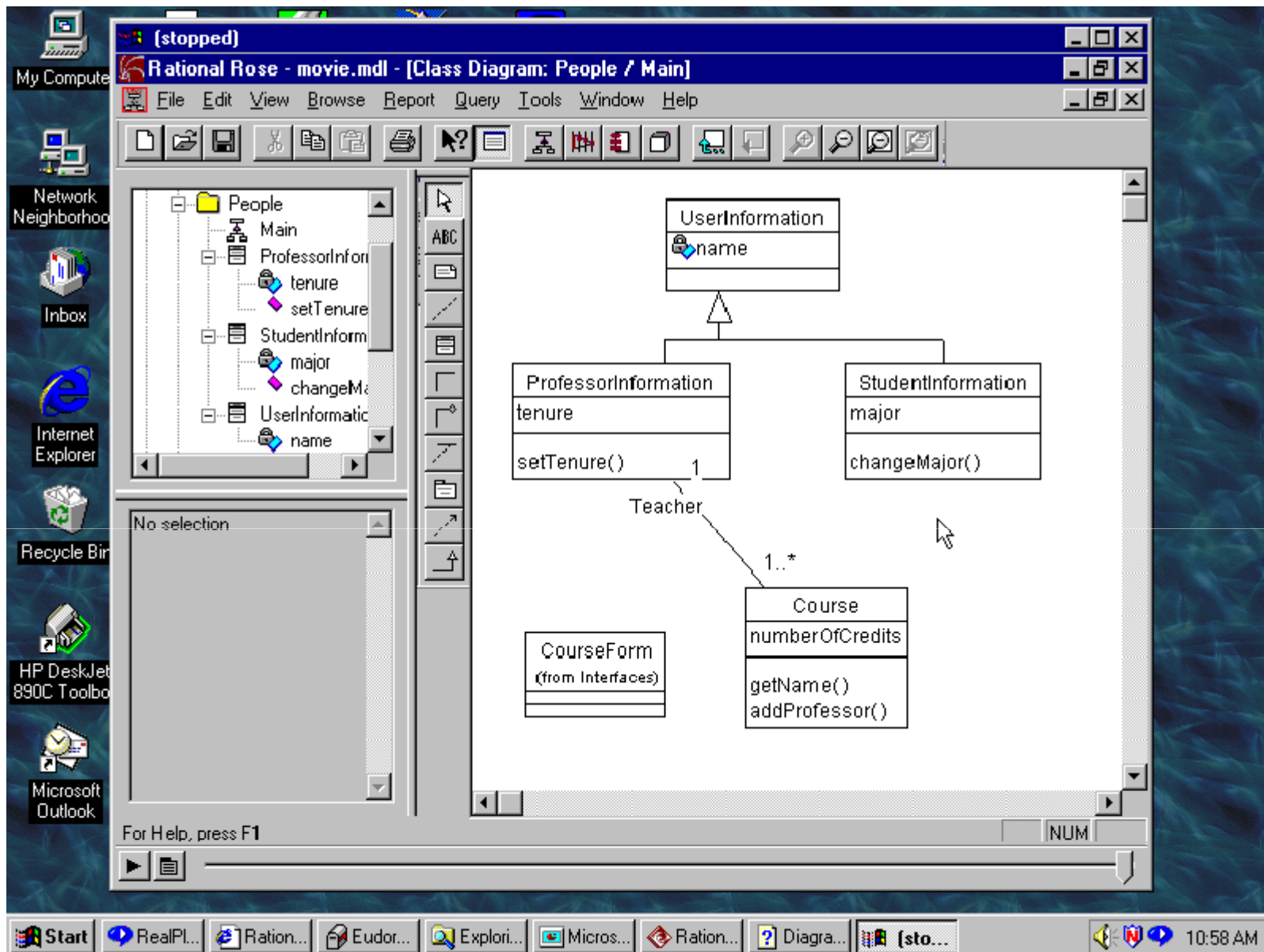


Note: Supertype = Superclass; Subtype = Subclass



Generalization - Multiple Classification





Rational Rose Class Diagram Example

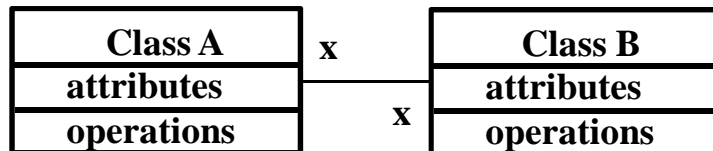
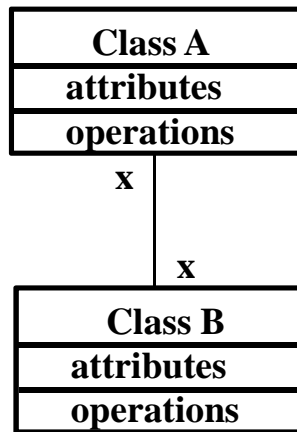


Associations

- Relationships between instances (objects) of classes
- Conceptual:
 - associations can have two roles (bi-directional):
 - source --> target
 - target --> source
 - roles have multiplicity (e.g., cardinality, constraints)
 - To restrict navigation to one direction only, an arrowhead is used to indicate the navigation direction
- No inheritance as in generalizations

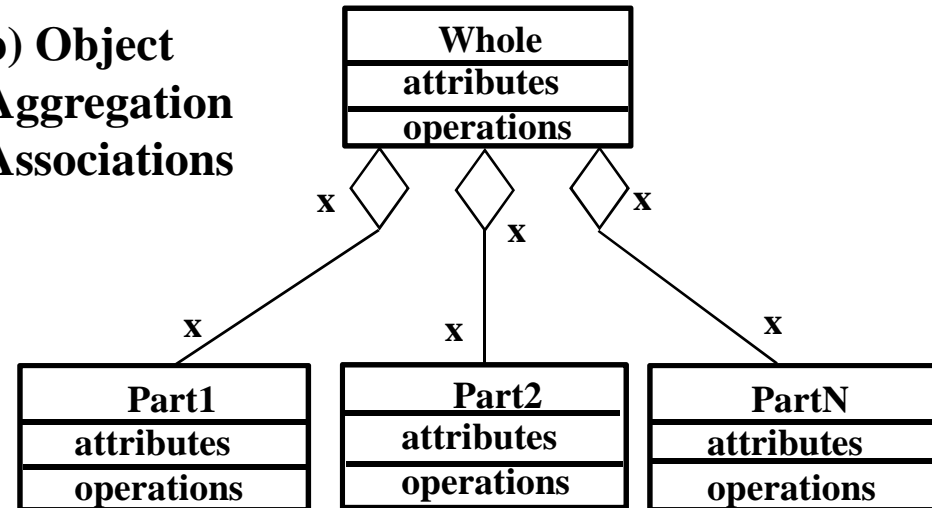


Object Association Relationship Patterns

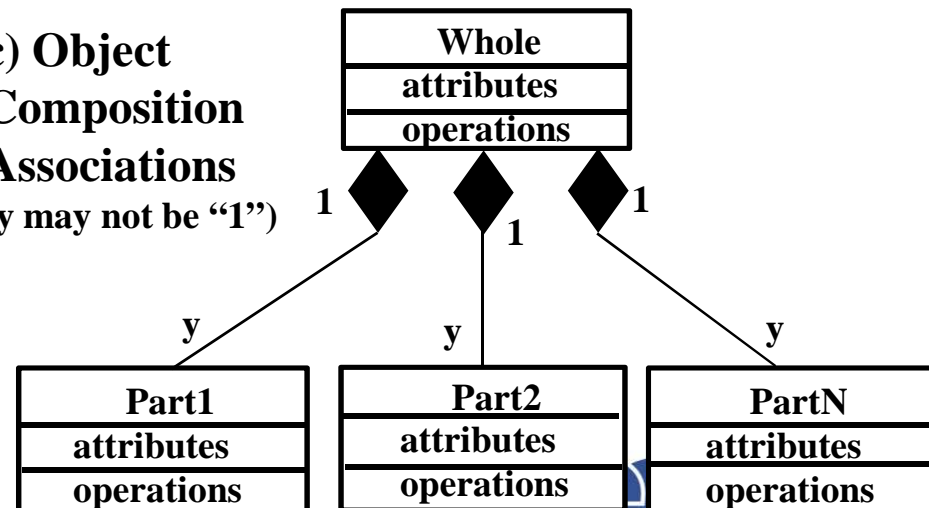


a) Object Associations

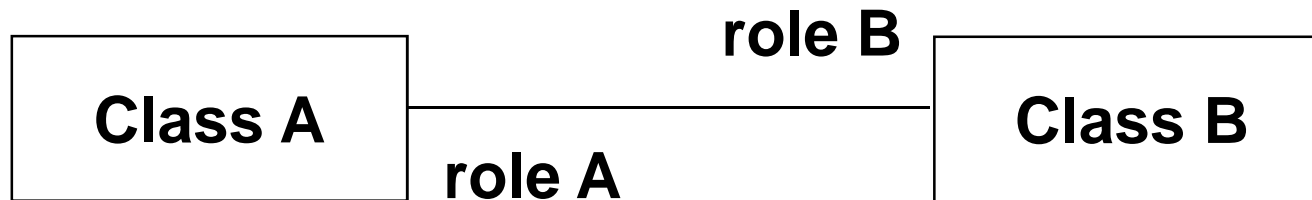
b) Object Aggregation Associations



c) Object Composition Associations
(y may not be "1")



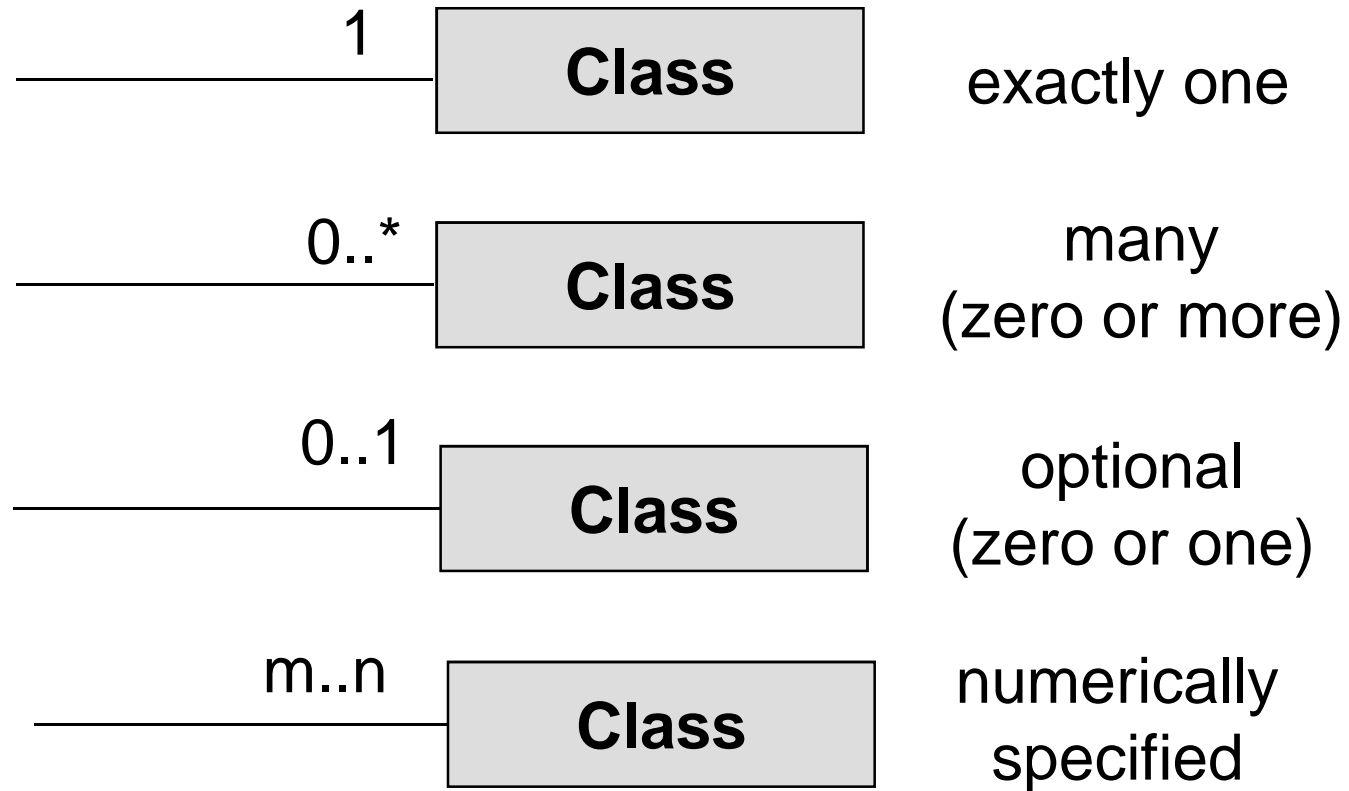
Associations



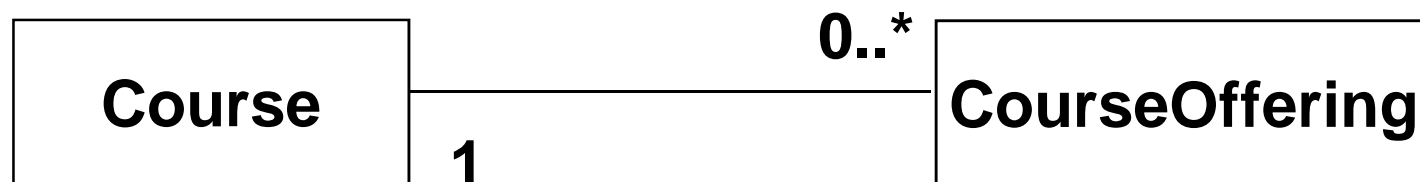
Example:



Multiplicities



Example:

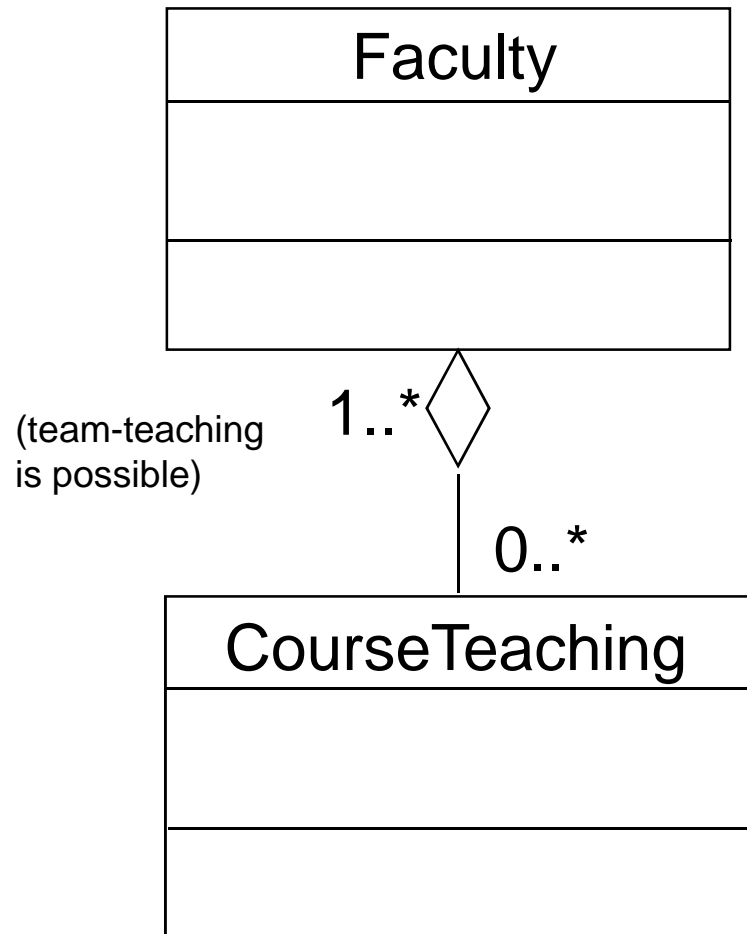


Aggregation & Composition

- **Aggregation (shared aggregation):**
 - *is a specialized form of ASSOCIATION in which a whole is related to its part(s).*
 - *is known as a “part of” or containment relationship and follows the “has a” heuristic*
 - *three ways to think about aggregations:*
 - *whole-parts*
 - *container-contents*
 - *group-members*
- **Composition (composite aggregation):**
 - *is a stronger version of AGGREGATION*
 - *the “part(s)” may belong to only ONE whole*
 - *the part(s) are usually expected to “live” and “die” with the whole (“cascading delete”)*
- **Aggregation vs. Composition vs. Association???**

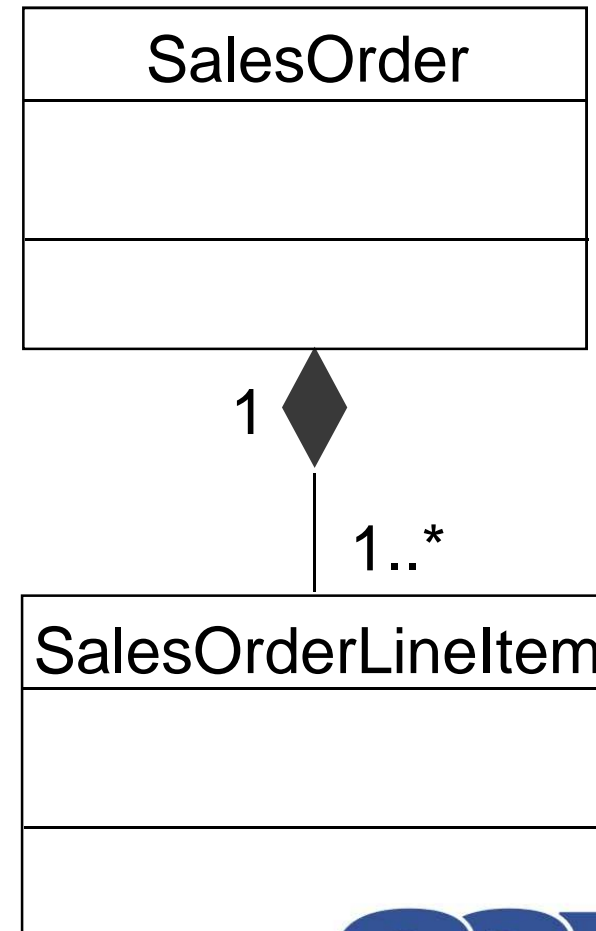


Aggregation



(another: assembly --> part)


Composition

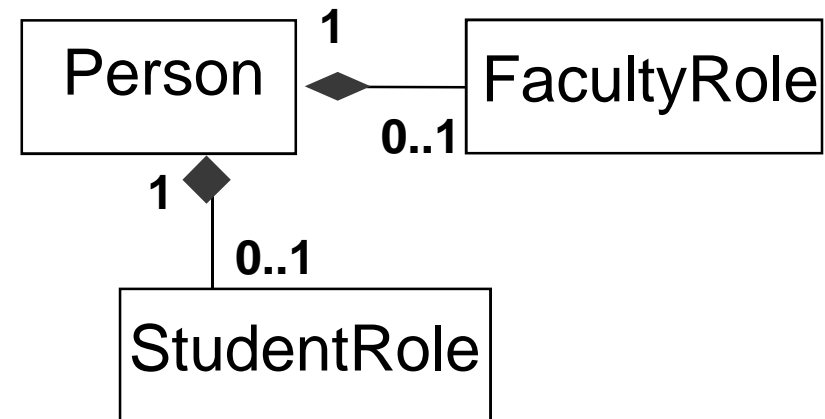
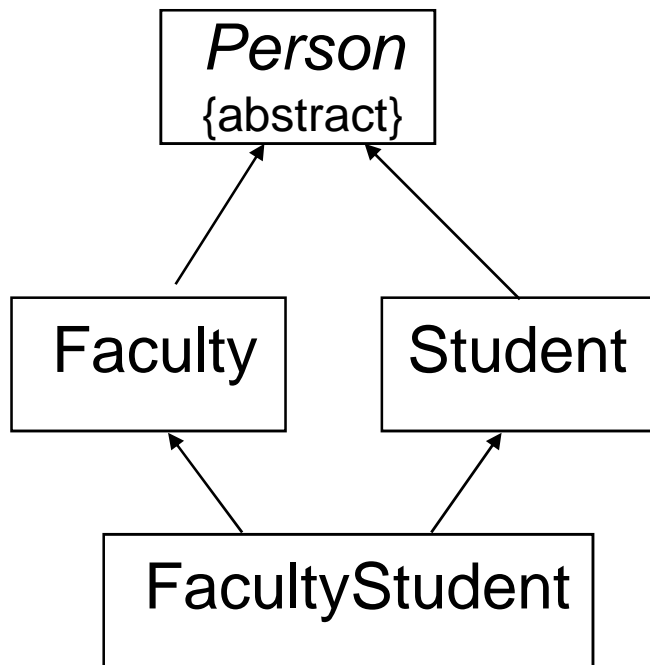


(another: hand --> finger)



Composition

 **Composition is often used in place of Generalization (inheritance) to avoid “transmuting” (adding, copying, and deleting of objects)**

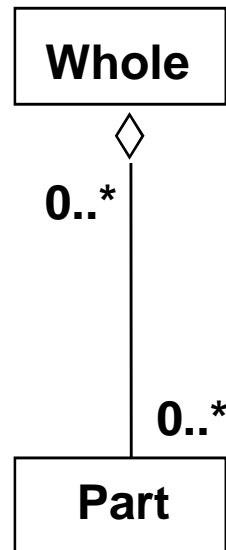


Note: Attributes may need to be considered to more-fully understand

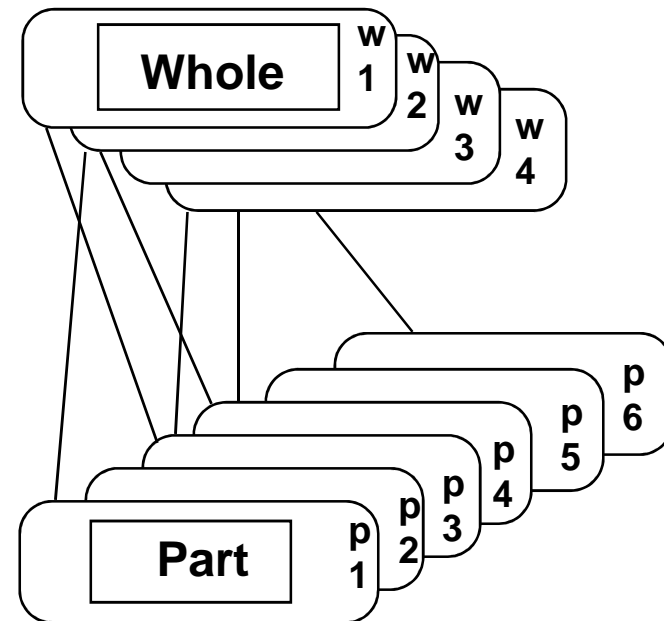


Association, Aggregation and Composition

Template/Pattern

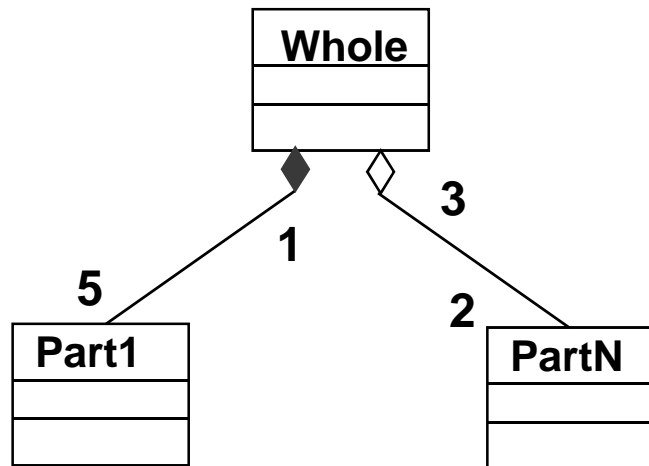


Example



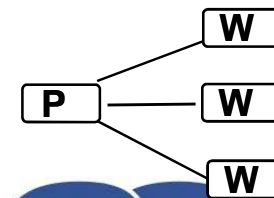
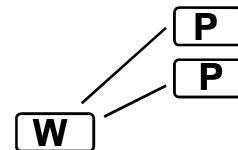
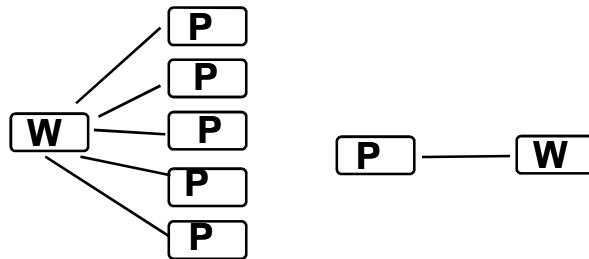
(association, aggregation & composition look the same)

Multiplicity Example #1

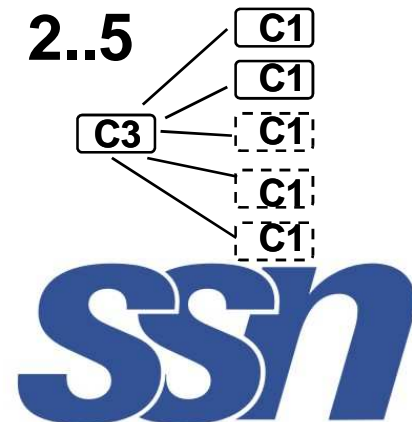
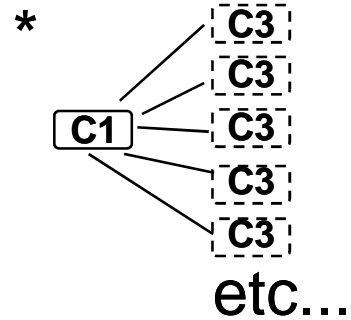
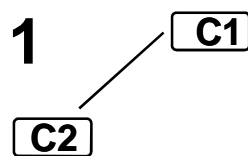
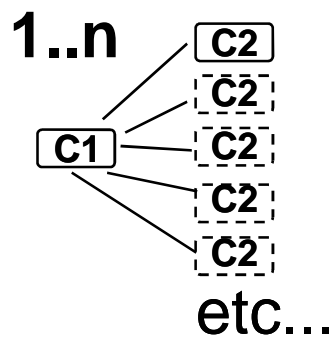
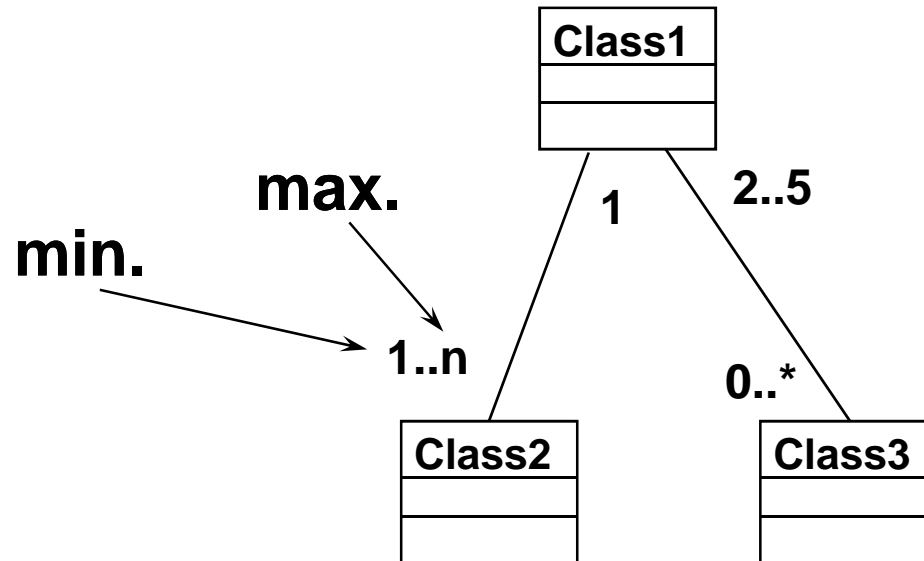


- One Whole is associated with 5 Part1
- One Part1 is associated with 1 Whole

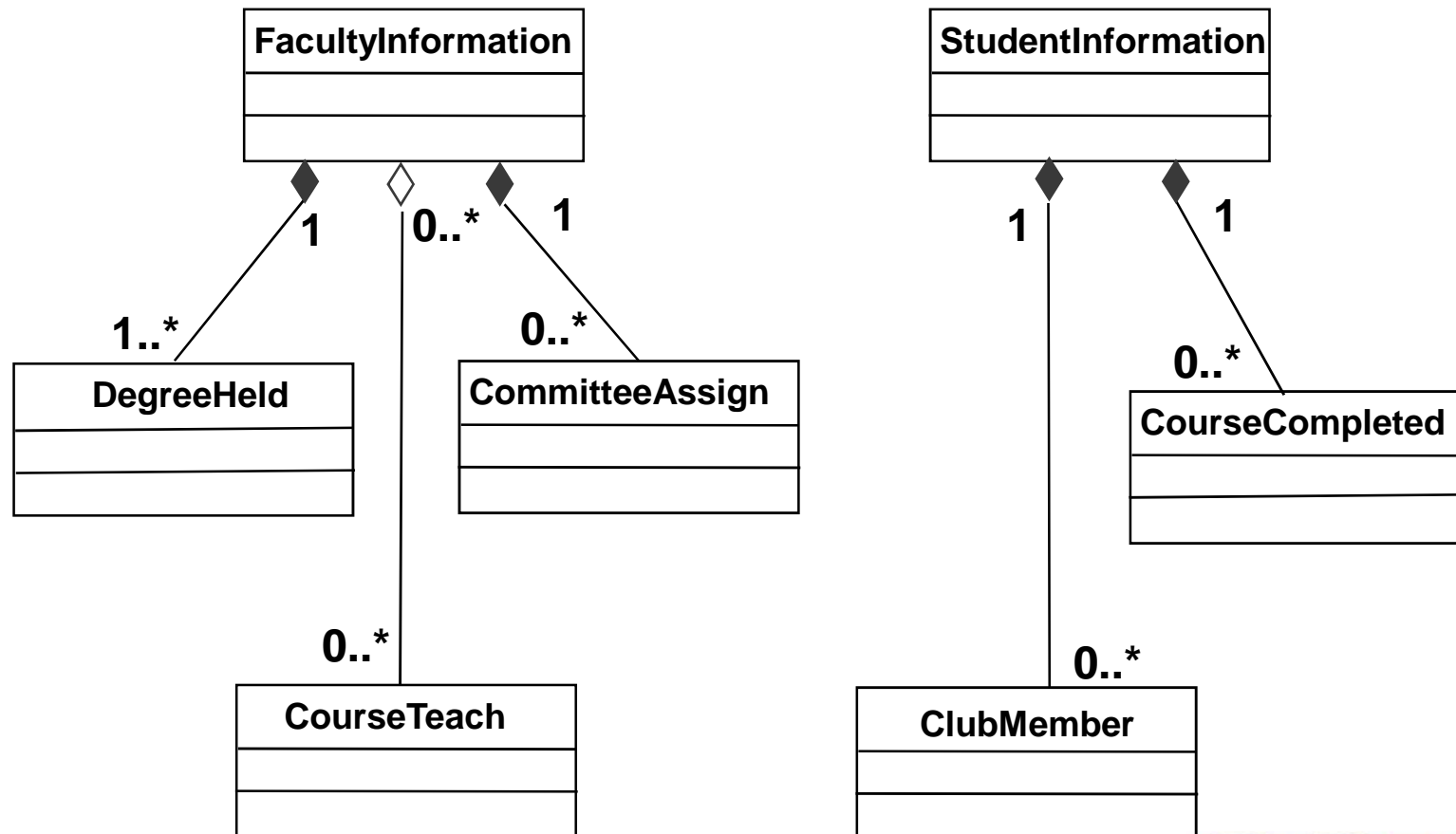
- One Whole is associated with 2 PartN
- One PartN is associated with 3 Whole



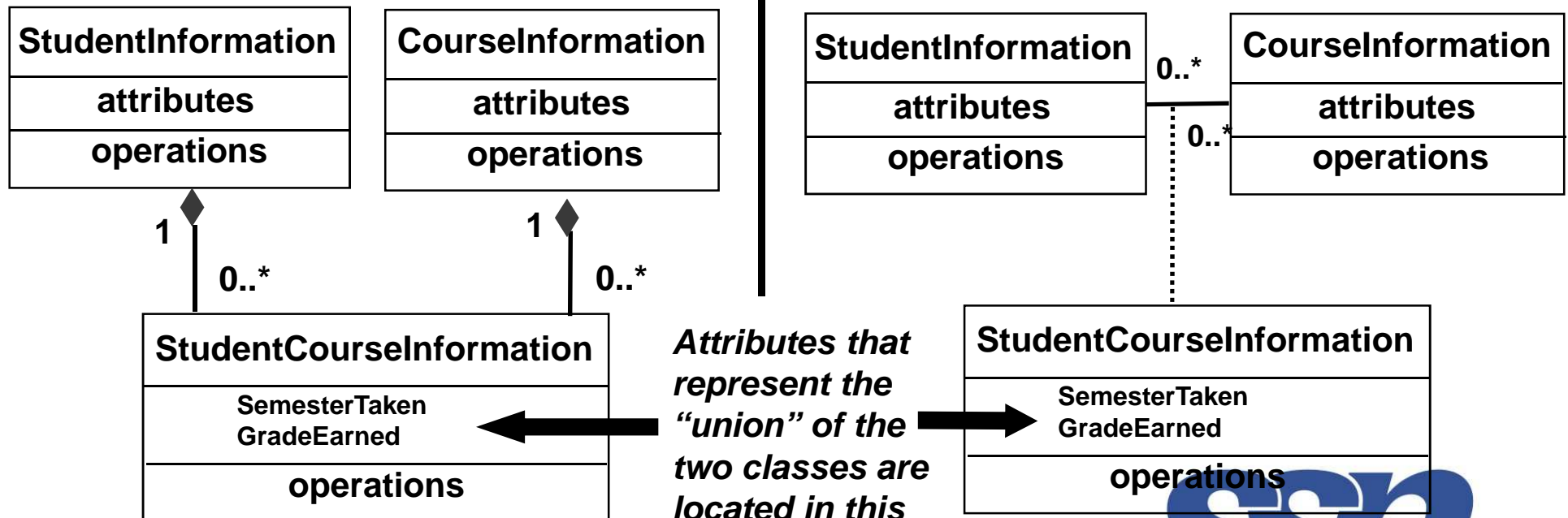
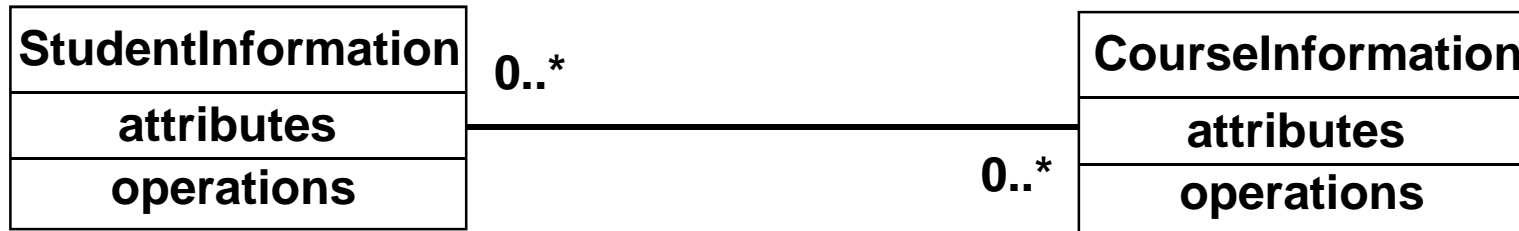
Multiplicity Example #2



Multiplicity Example #3



“many-to-many” multiplicity

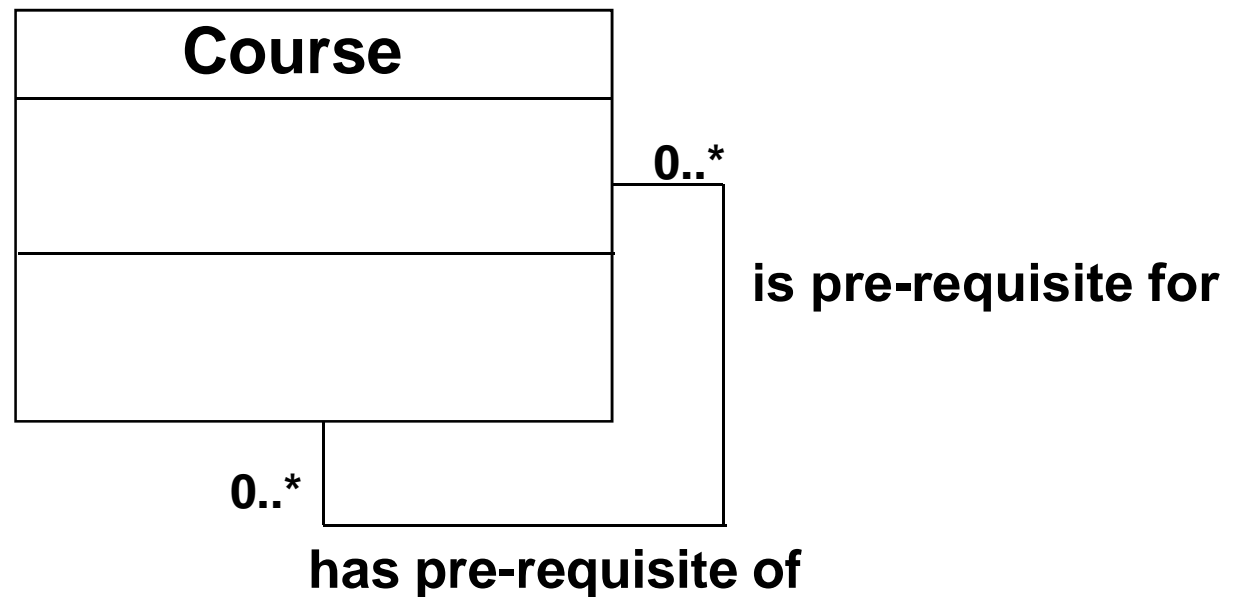


Attributes that represent the “union” of the two classes are located in this “association” class.

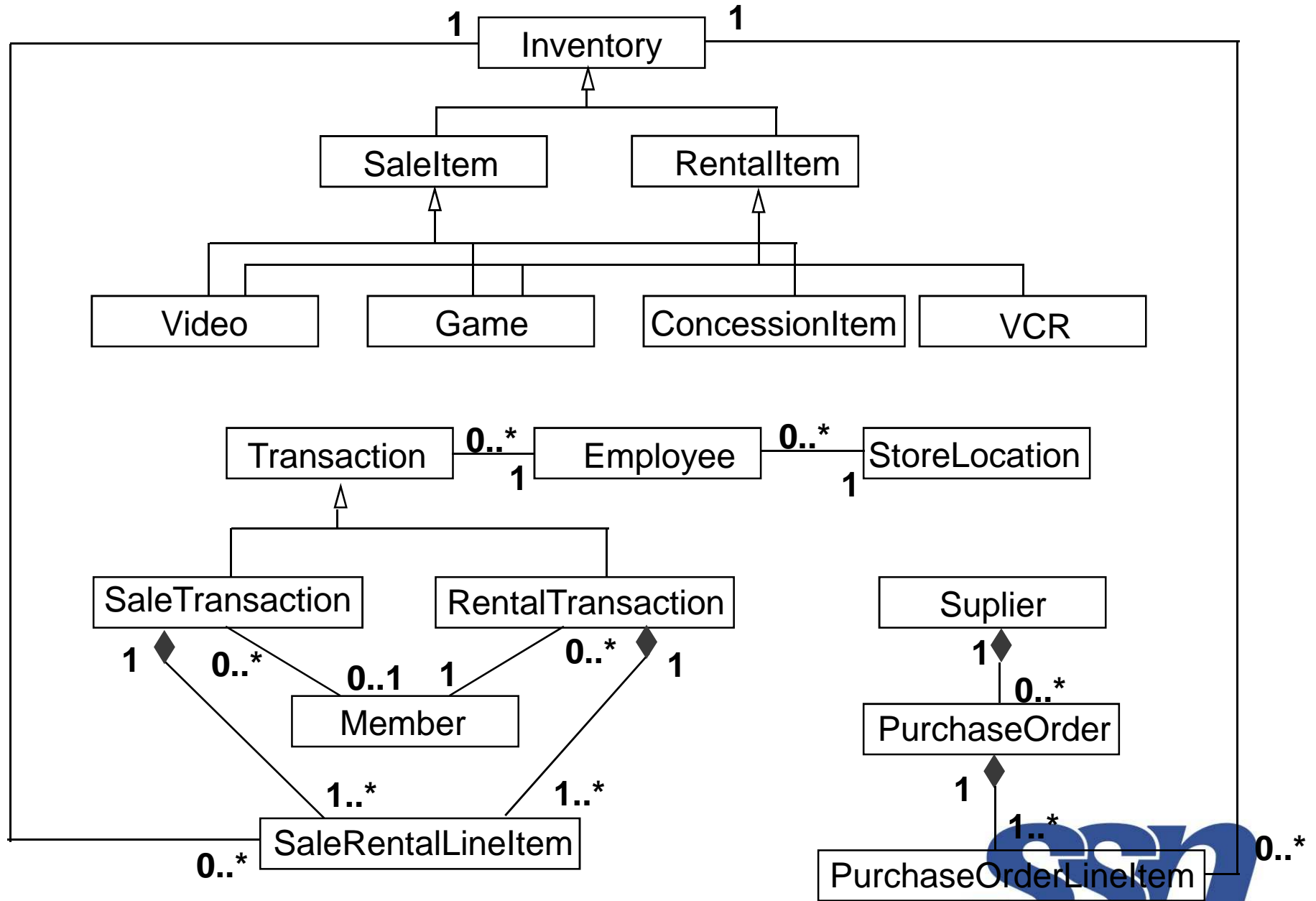


Reflexive Association Relationships

Objects within the same class have a relationship with each other.



Video Store – UML Class Diagram



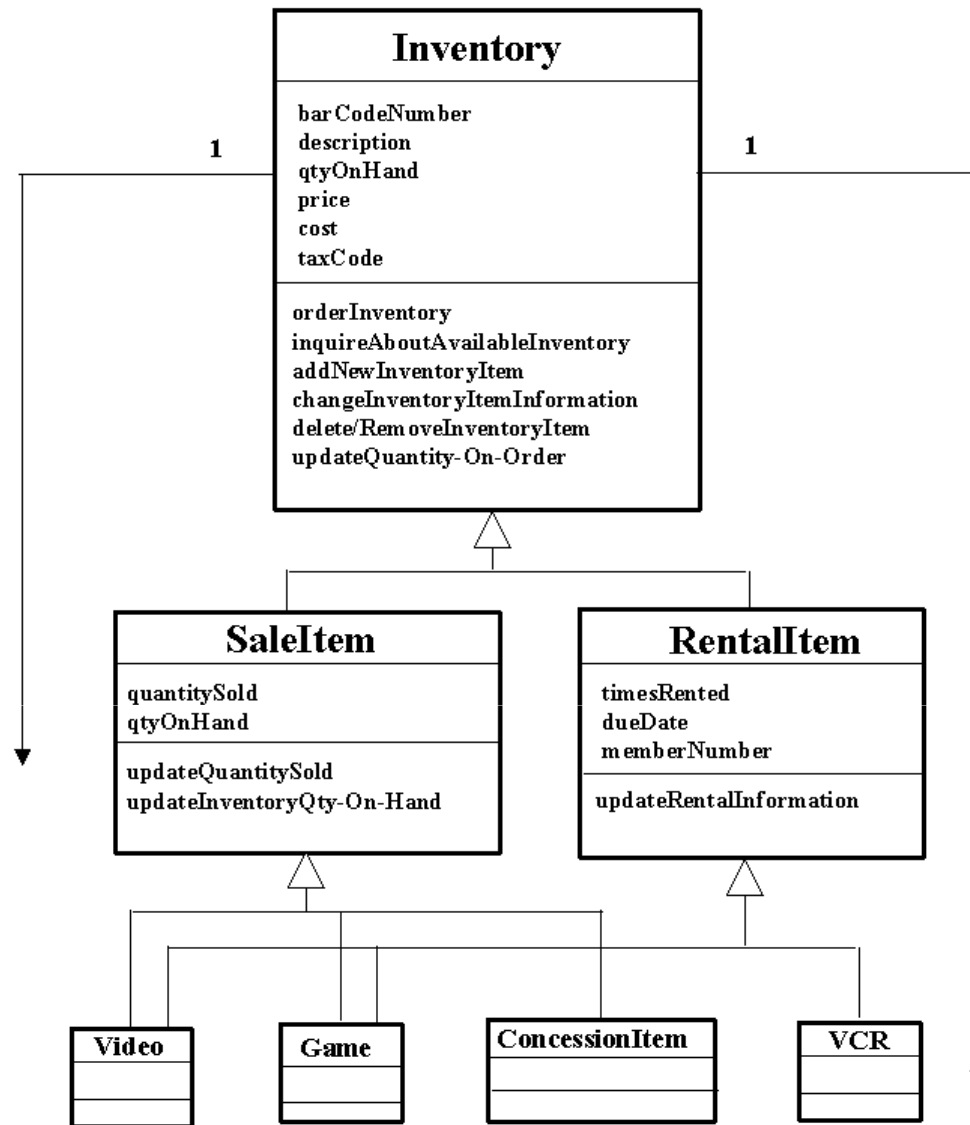


Figure 3.10a Video Store UML Class Diagram with Attributes & Operations



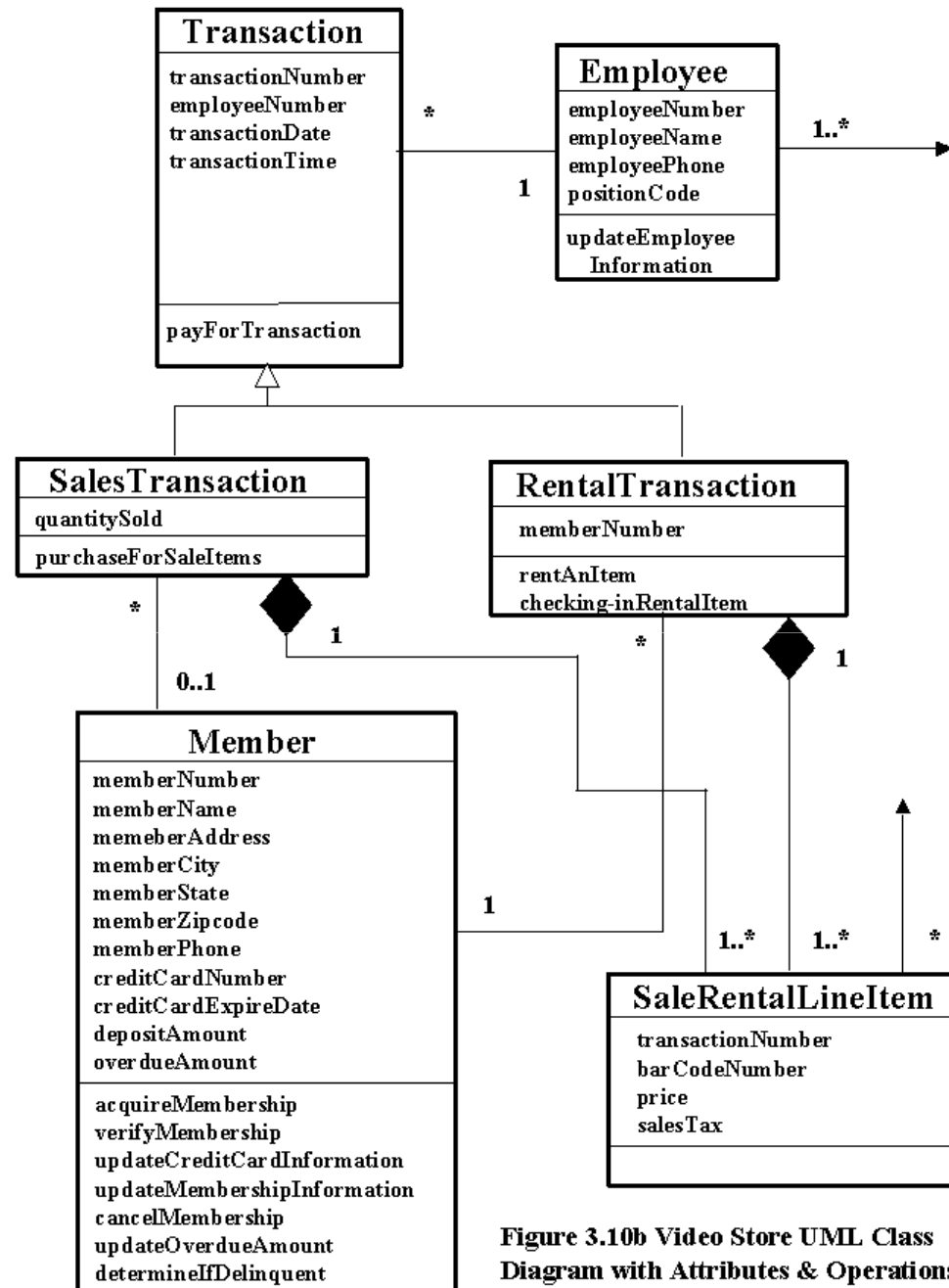


Figure 3.10b Video Store UML Class Diagram with Attributes & Operations



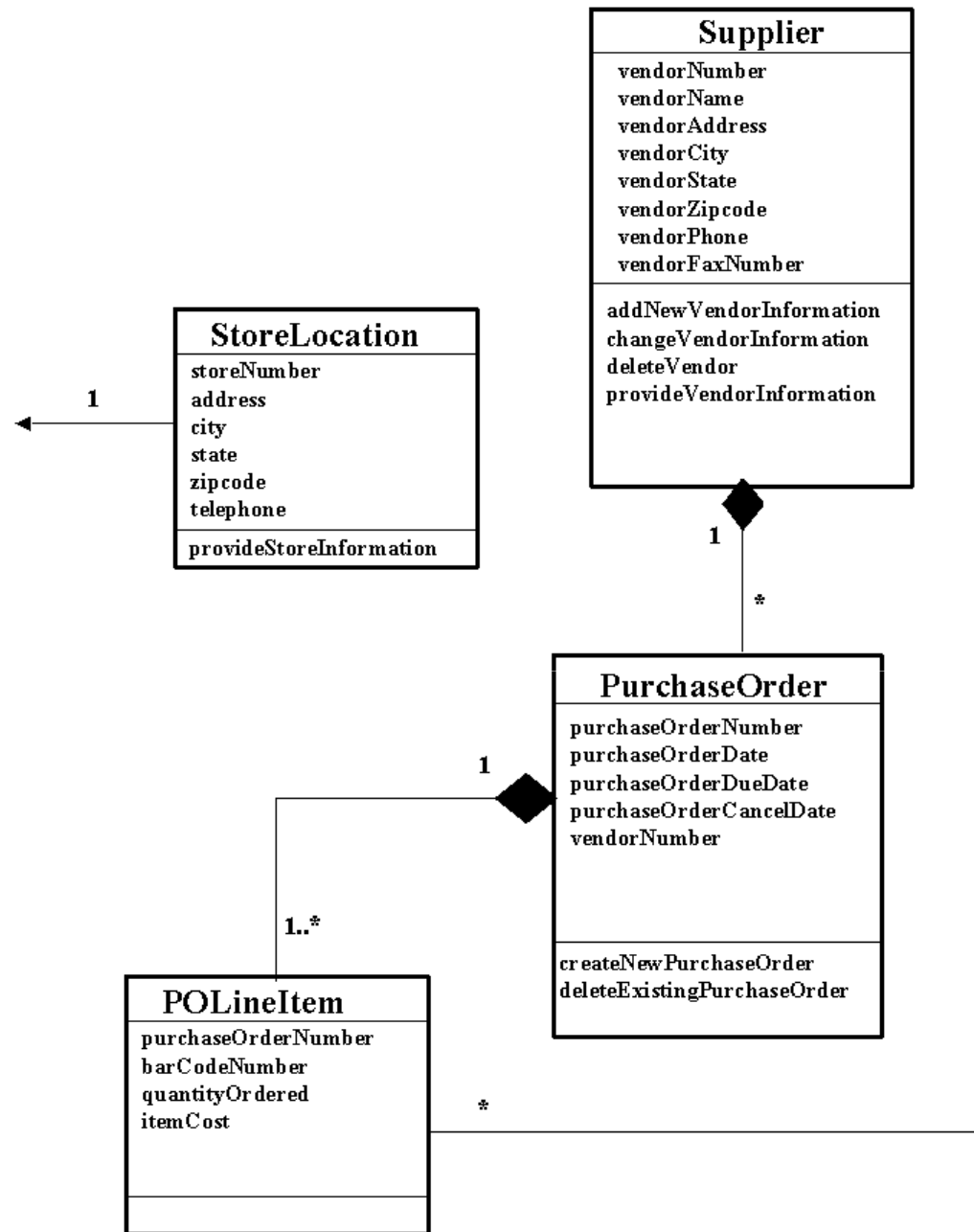


Figure 3.10c Video Store UML Class Diagram with Attributes & Operations

