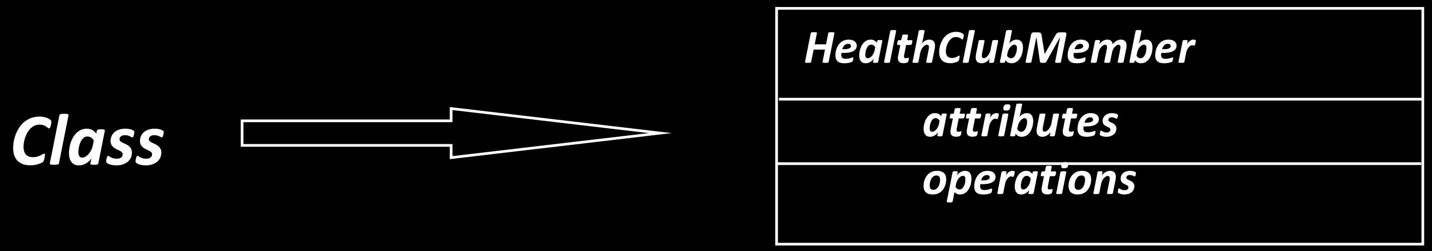
# LECTURE-15

# 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*

*HealthClubMember*

*attributes*

*operations*

*Objects*

# • *Relationships*

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

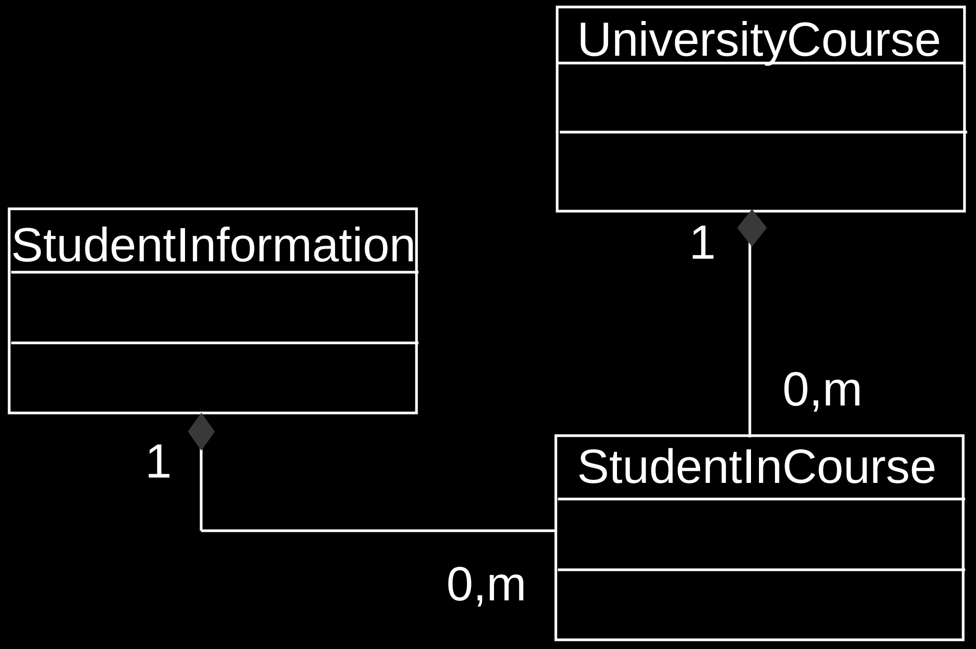
*Four Types*

- ***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.***

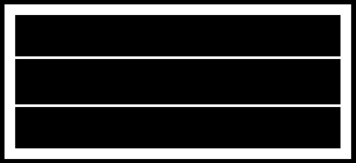
- ***Removal of a Student should also remove the Courses that the Student is in but not the University Course.***

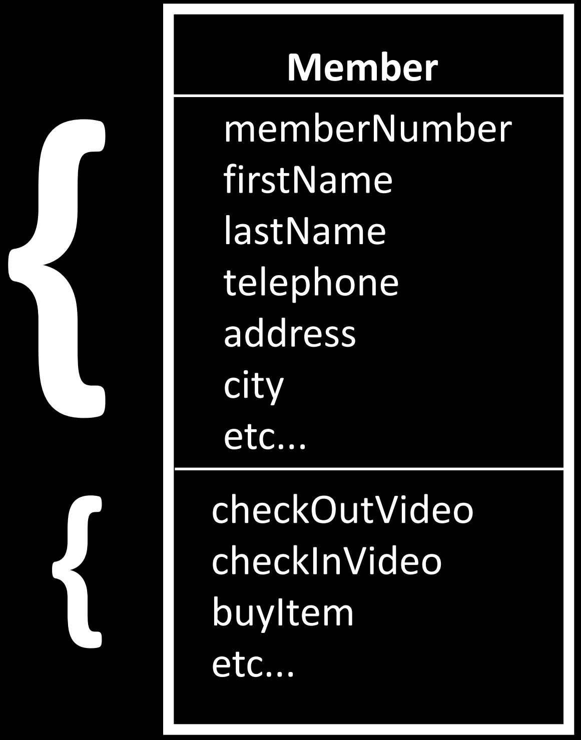
- ***Removal of a Student in a Course should not affect either University Course or Student Information.***

# *UML Class Diagram Notation*

***1 of 2***

Class





attributes

operations

**Member**

memberNumber
firstName
lastName
telephone
address
city
etc...

checkOutVideo
checkInVideo
buyItem
etc...

**Expanded view of a Class into its three sections:**

**Top: Class Name**

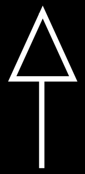
**Middle: attributes**

**Bottom: operations**

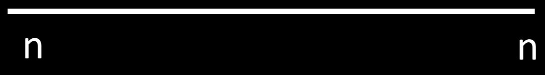
# *UML Class Diagram Notation*

***2 of 2***

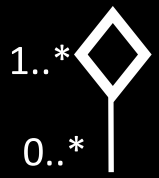
**Class Generalization Relationship**



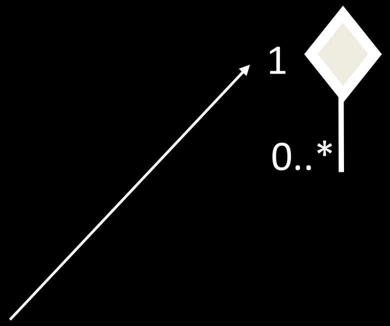
**Object Association**



**Object Aggregation Association**



**Object Composition Association**



**Will always be "1"**

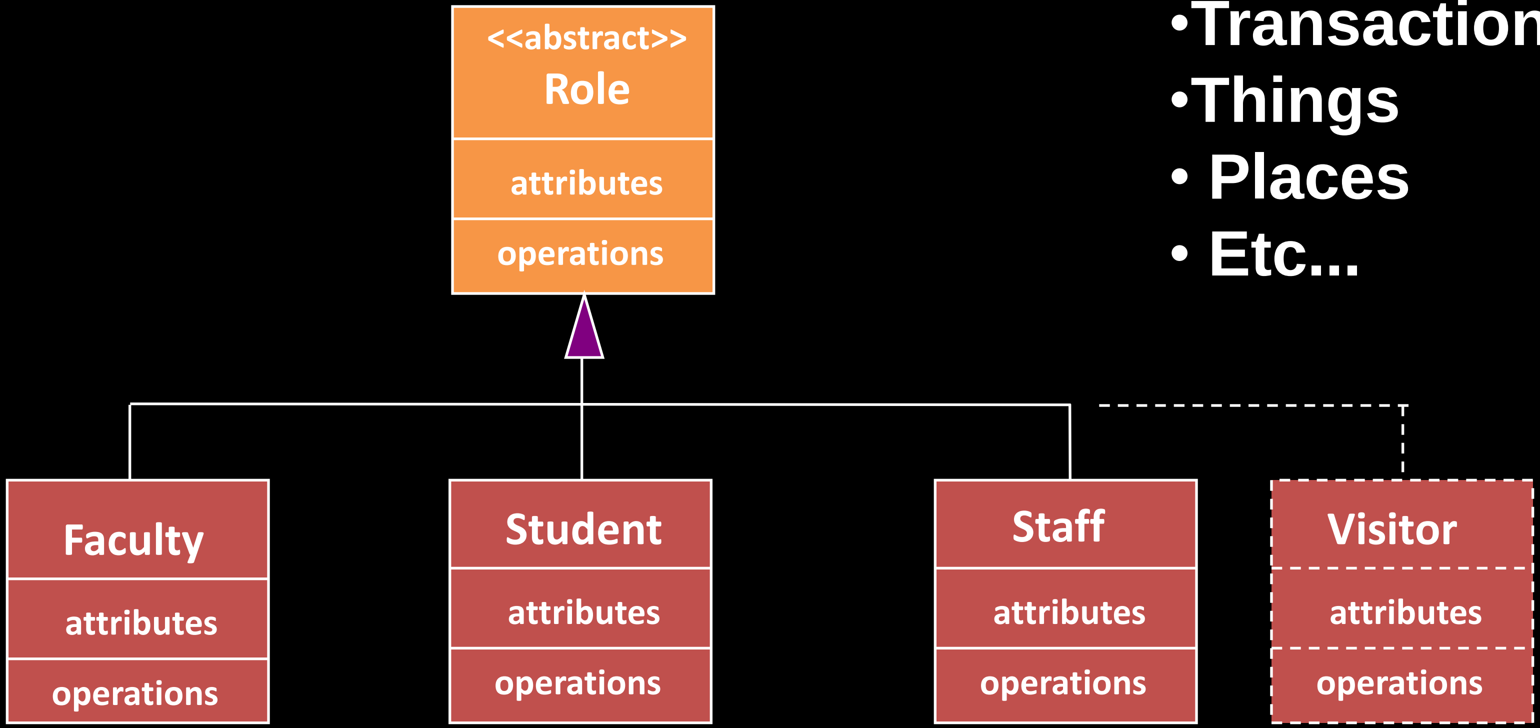
# *Class Diagram Relationships*

- ***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*



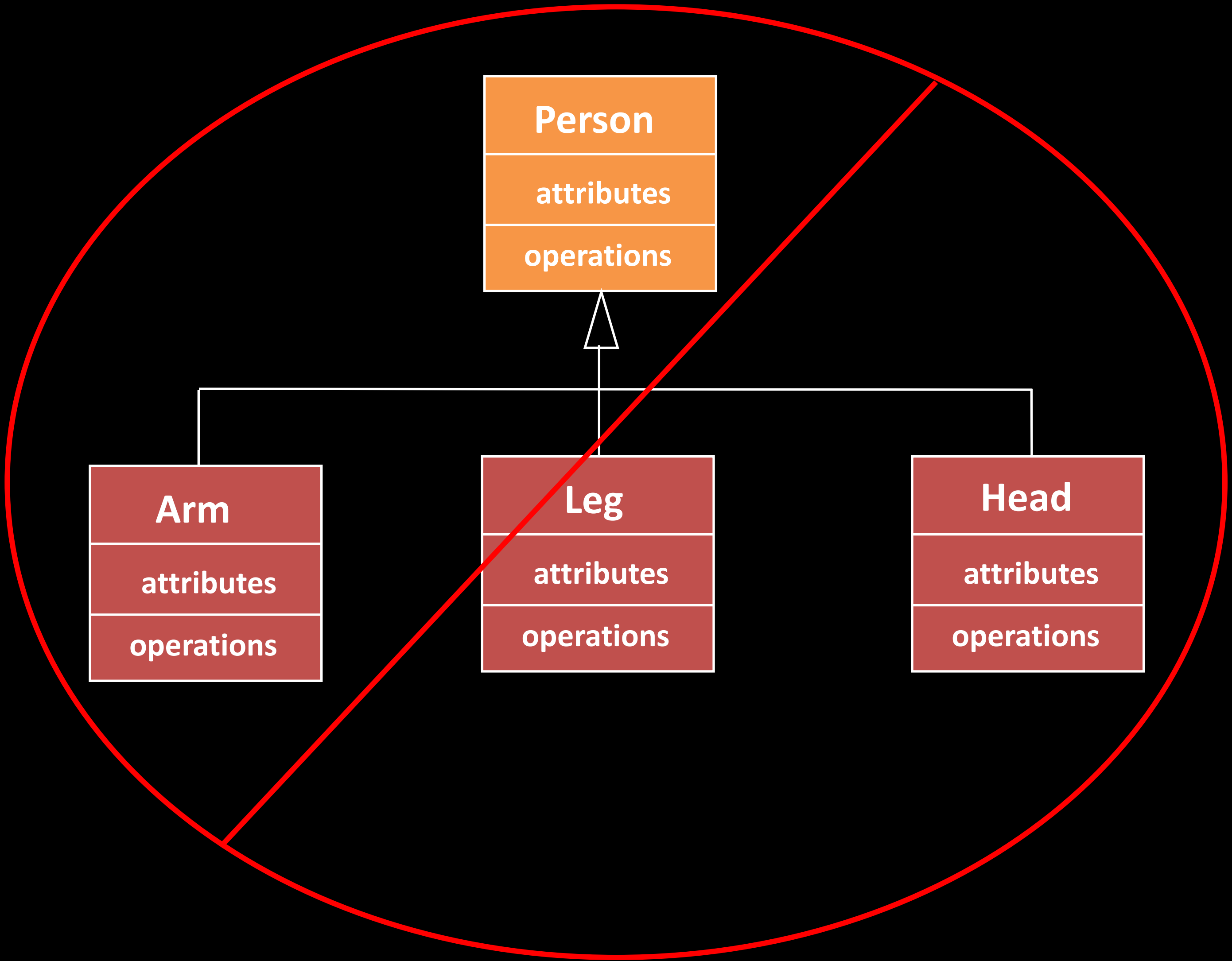
Others:

- Transactions
- Things
- Places
- Etc...

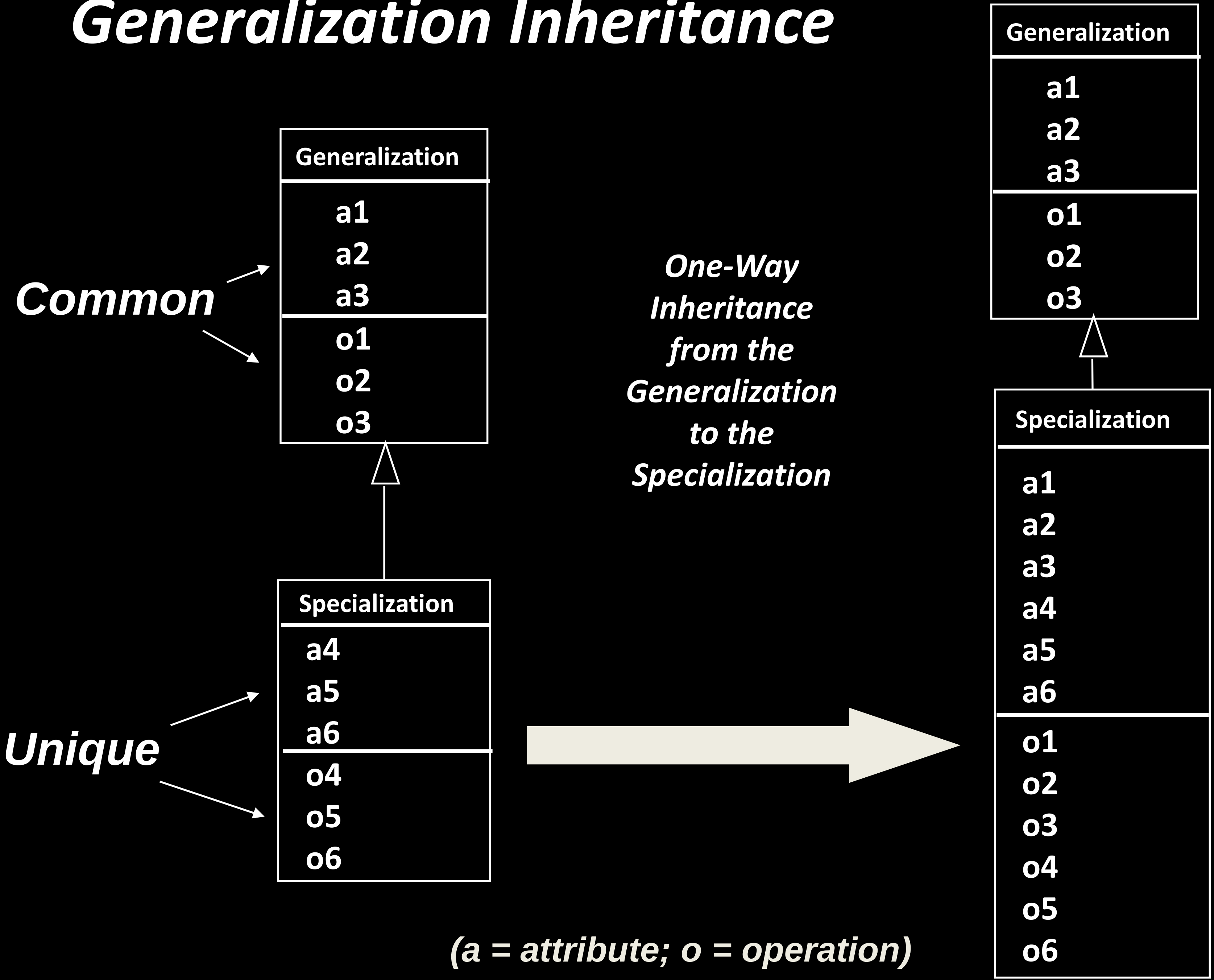
***Note: <<abstract>> = no objects***

# *Poor Generalization Example*

*(violates the "is a" or "is a kind of" heuristic)*



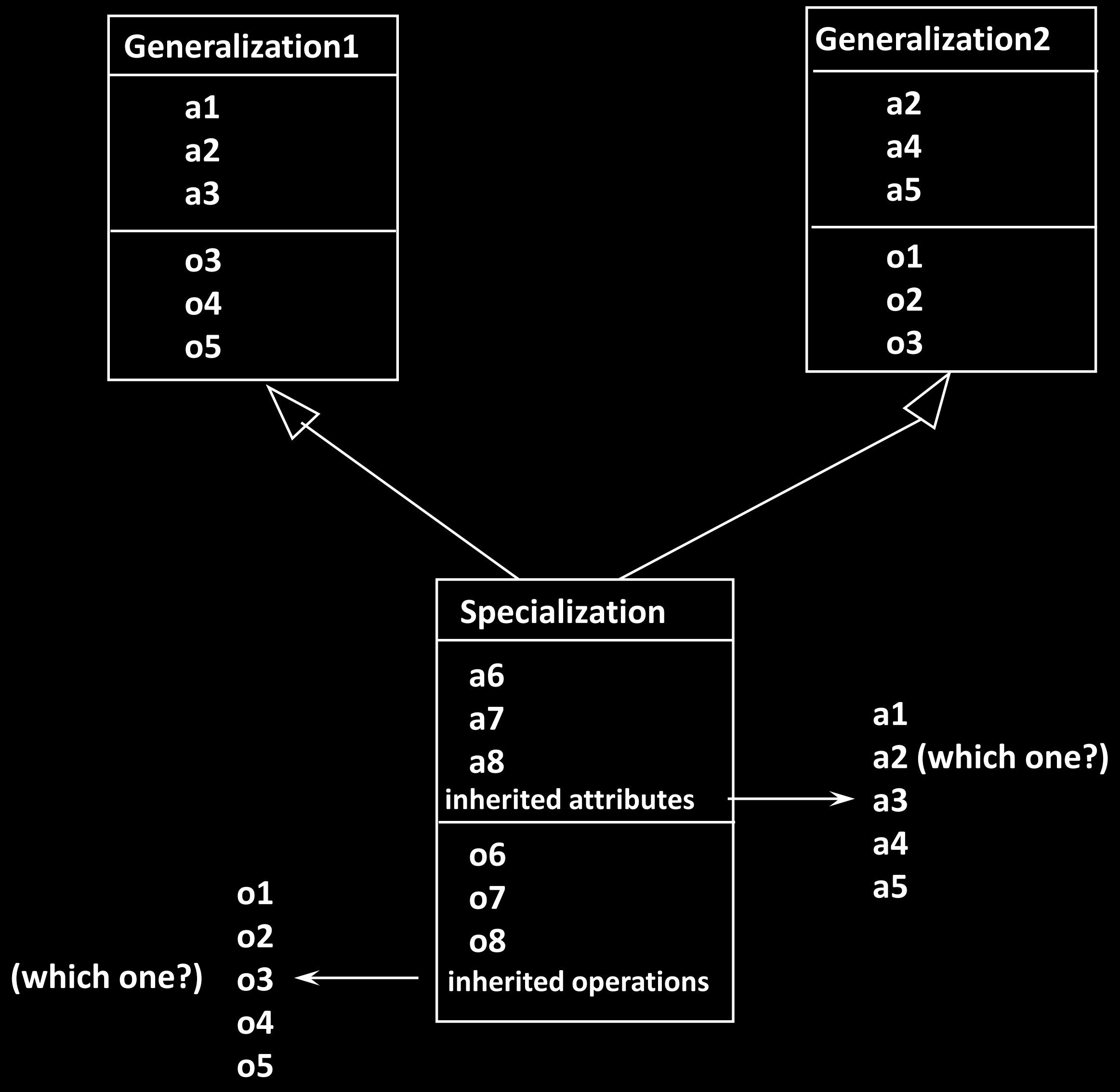
# Generalization Inheritance



*One-Way Inheritance from the Generalization to the Specialization*

*(a = attribute; o = operation)*

# *Generalization - Multiple Inheritance*



Generalization1

a1
a2
a3

o3
o4
o5

Generalization2

a2
a4
a5

o1
o2
o3

Specialization

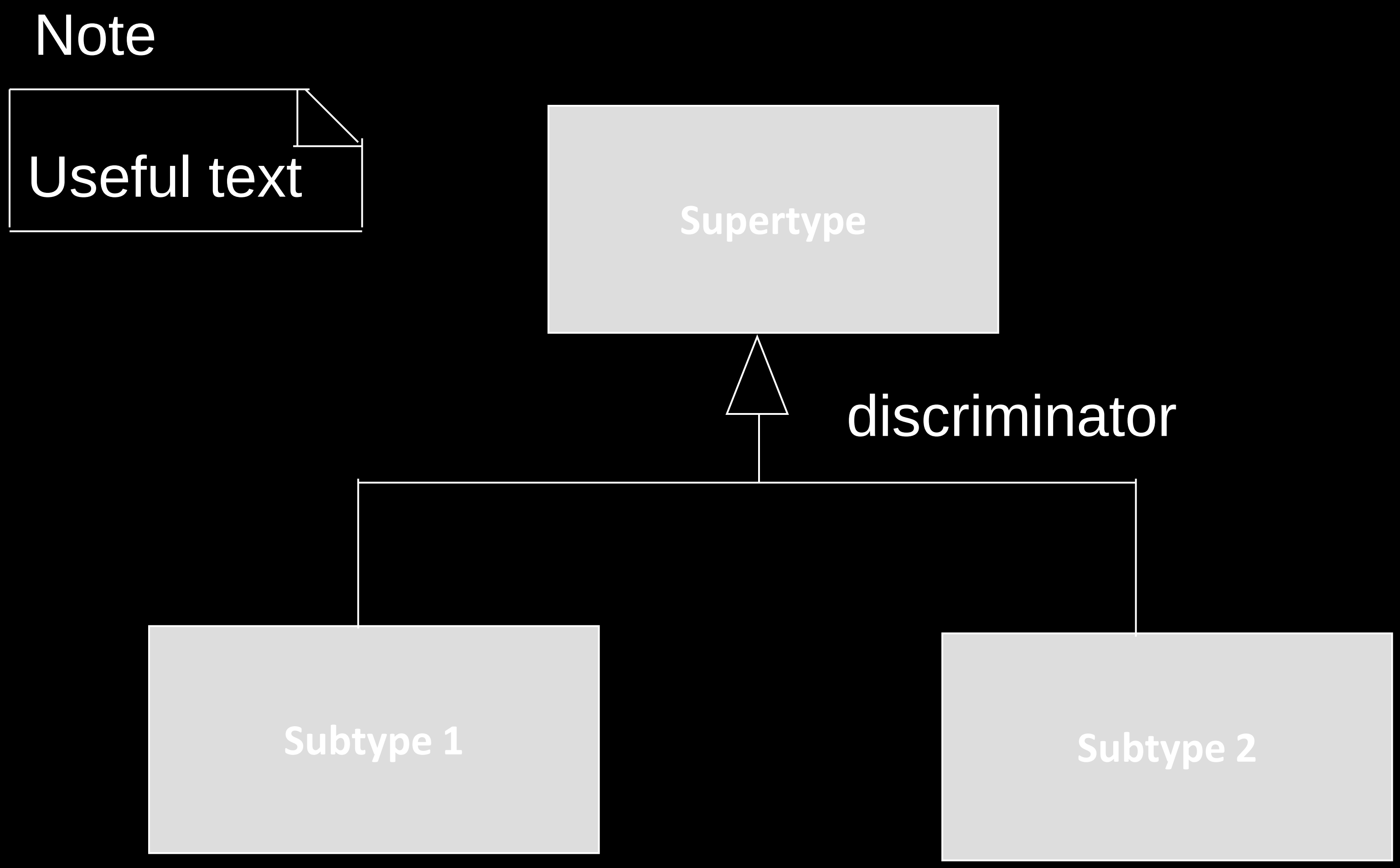
a6
a7
a8
inherited attributes

o6
o7
o8
inherited operations

a1
a2 (which one?)
a3
a4
a5

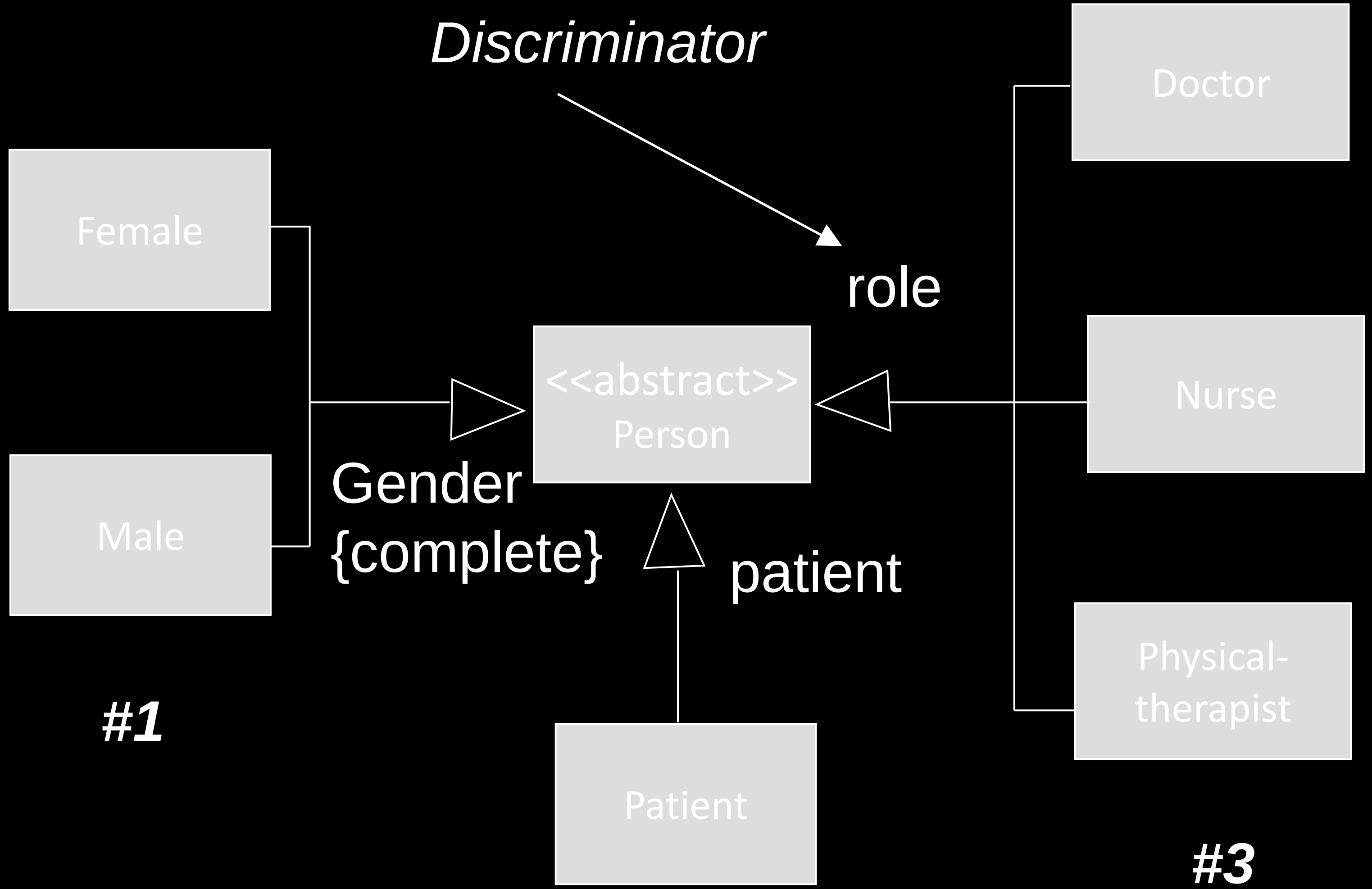
o1
o2
(which one?) o3
o4
o5

# ***UML Generalization Notation***



Note: Supertype = Superclass; Subtype = Subclass

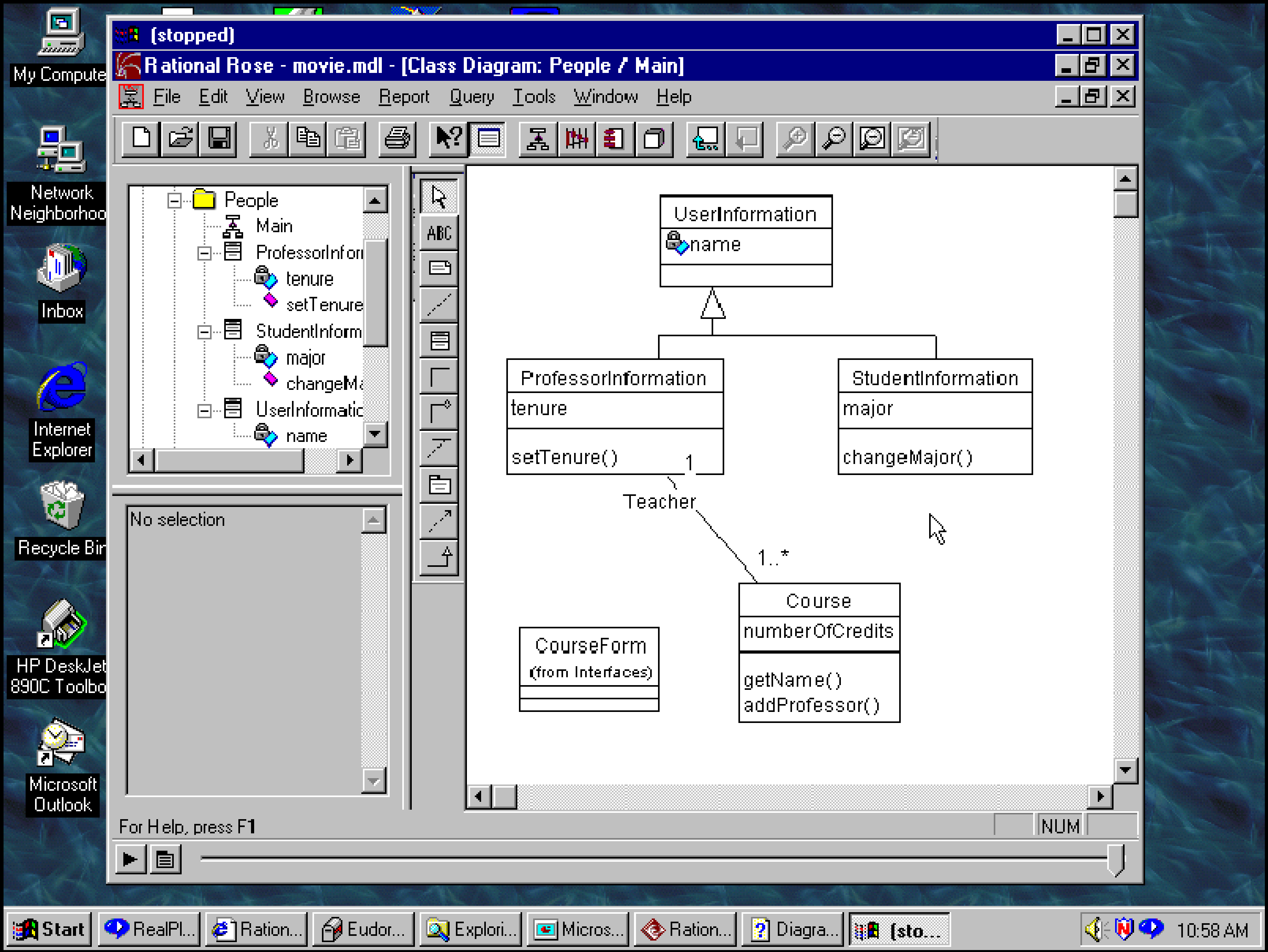
# *Generalization - Multiple Classification*



***#1***

***#3***

***#2***



No selection

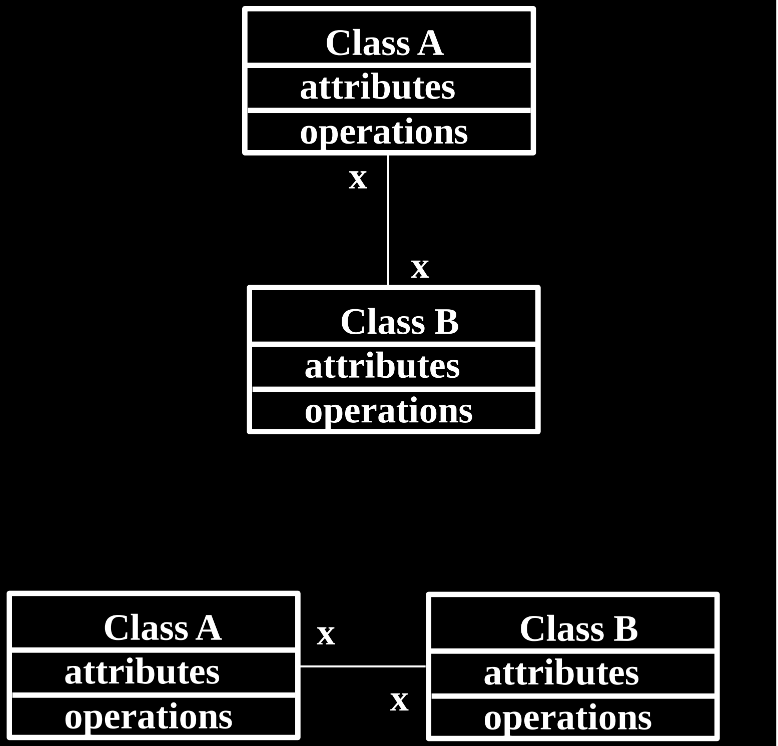
For Help, press F1

# *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



Class A
attributes
operations

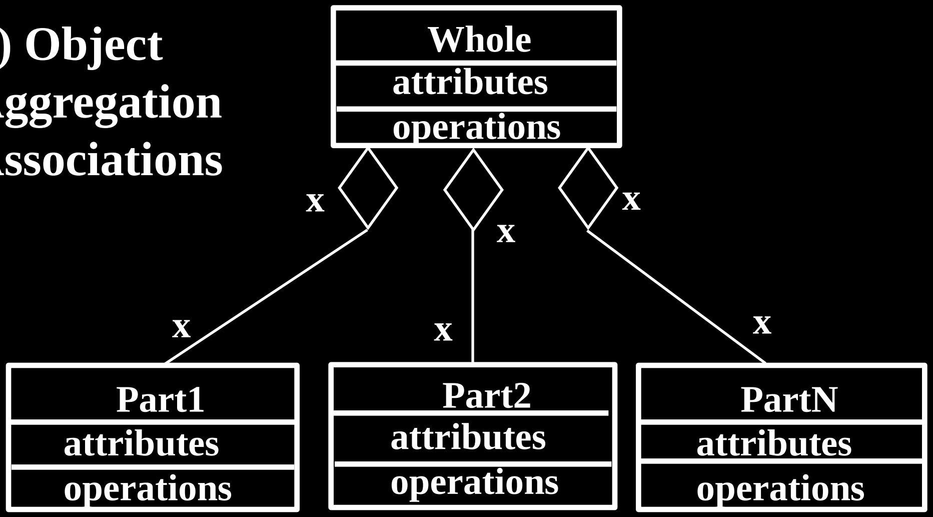
Class B
attributes
operations

Class A
attributes
operations

Class B
attributes
operations

a) Object Associations

b) Object Aggregation Associations



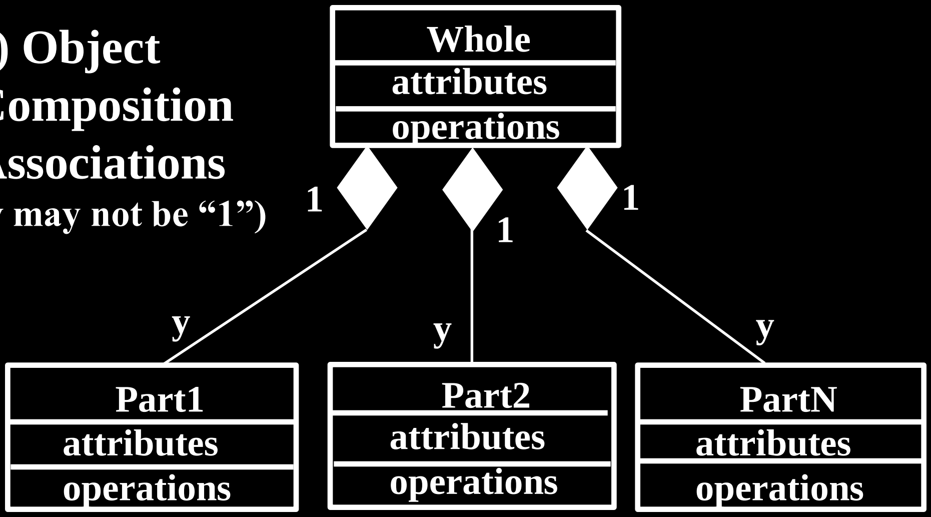
Whole
attributes
operations

Part1
attributes
operations

Part2
attributes
operations

PartN
attributes
operations

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



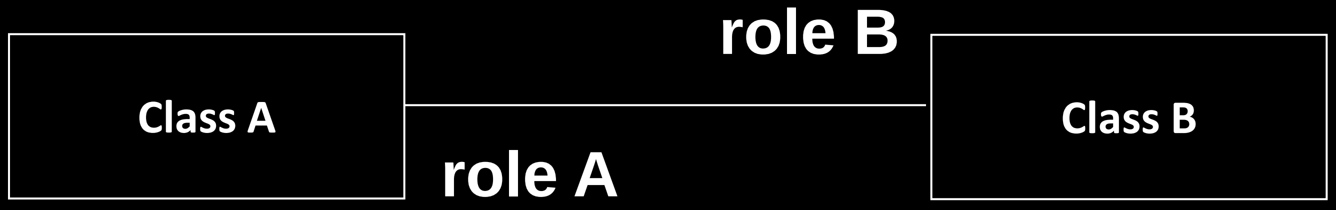
Whole
attributes
operations

Part1
attributes
operations

Part2
attributes
operations

PartN
attributes
operations

# *Associations*



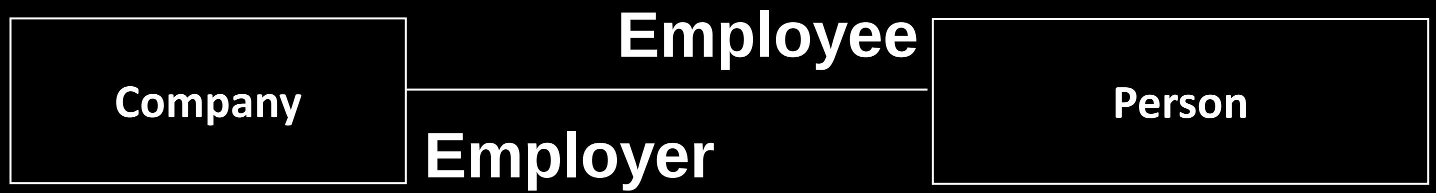
Class A

role B

Class B

role A

***Example:***



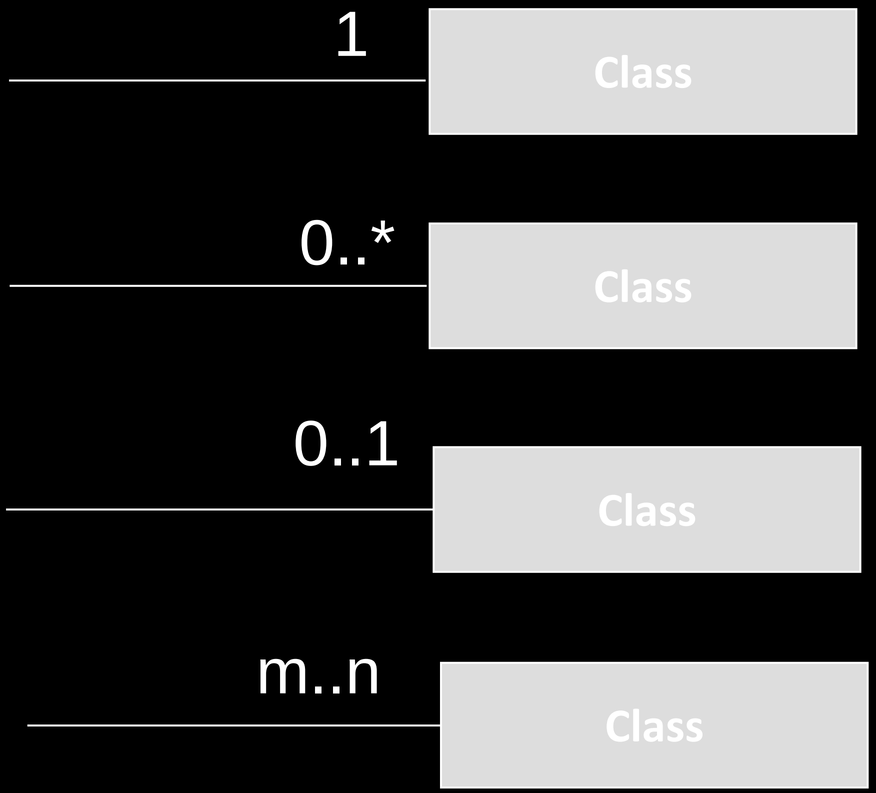
Company

Employee

Person

Employer

# Multiplicities



1

Class

exactly one

0..*

Class

many
(zero or more)

0..1

Class

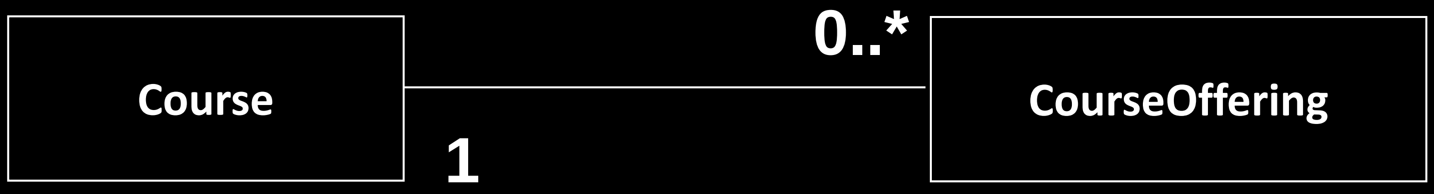
optional
(zero or one)

m..n

Class

numerically
specified

***Example:***



**Course**

**0..***

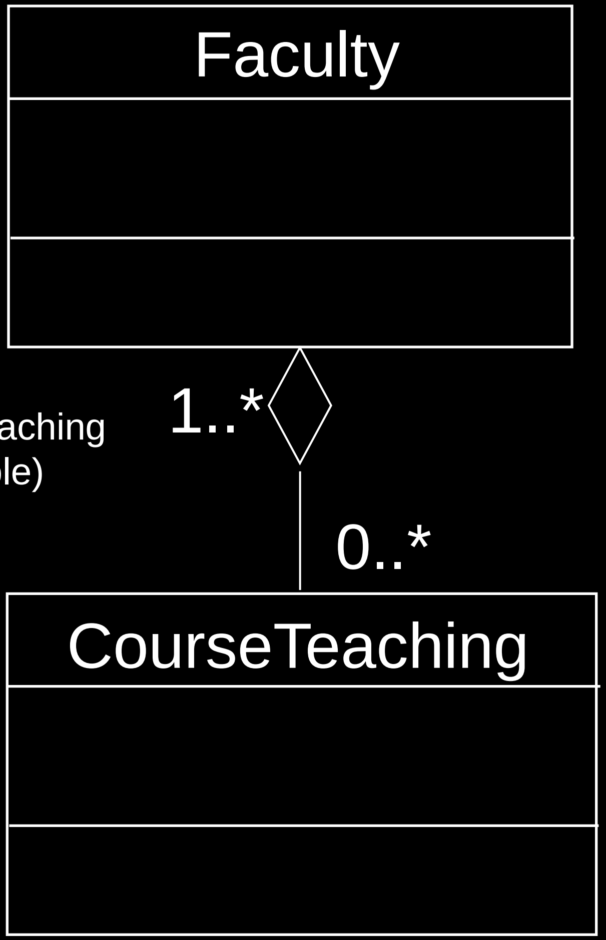
**1**

**CourseOffering**

# 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*



Faculty

1..*

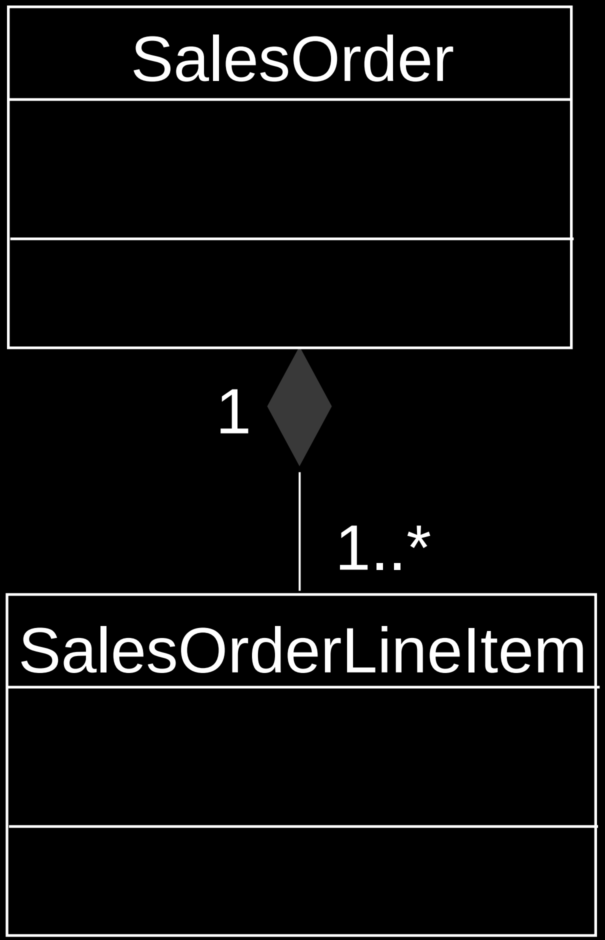
(team-teaching is possible)

0..*

CourseTeaching

*(another: assembly --> part)*

## *Composition*



SalesOrder

1

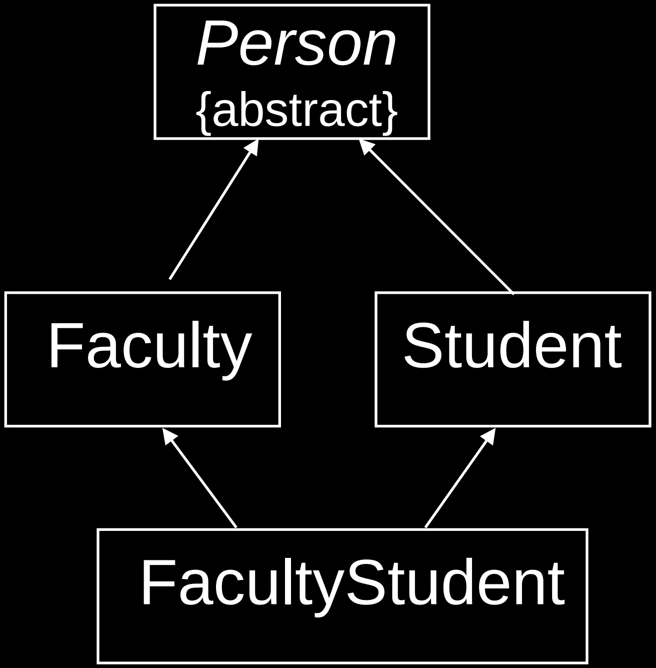
1..*

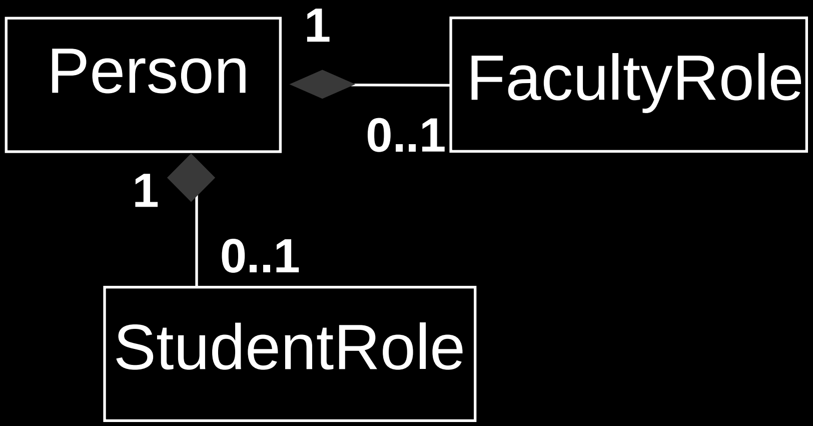
SalesOrderLineItem

*(another: hand --> finger)*

# Composition

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

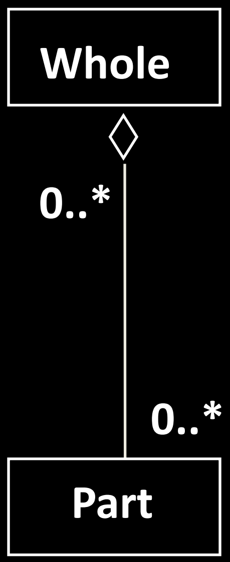




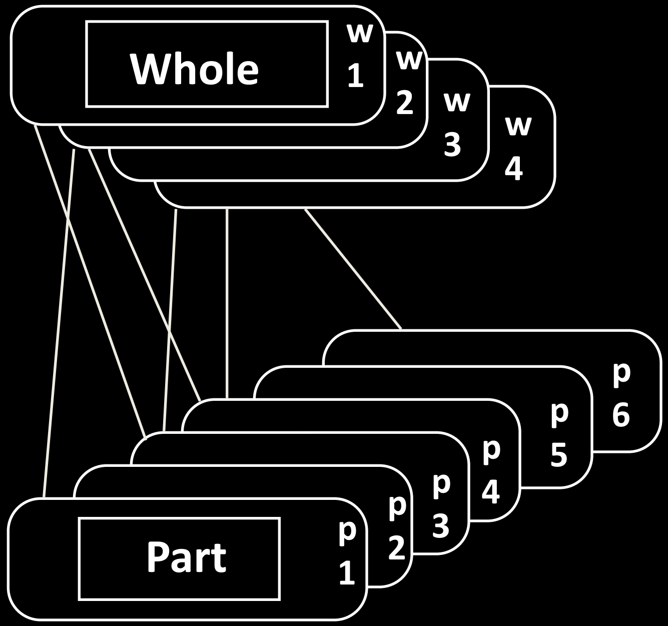
*Note: Attributes may need to be considered to more-fully understand*

# *Association, Aggregation and Composition*

***Template/Pattern***

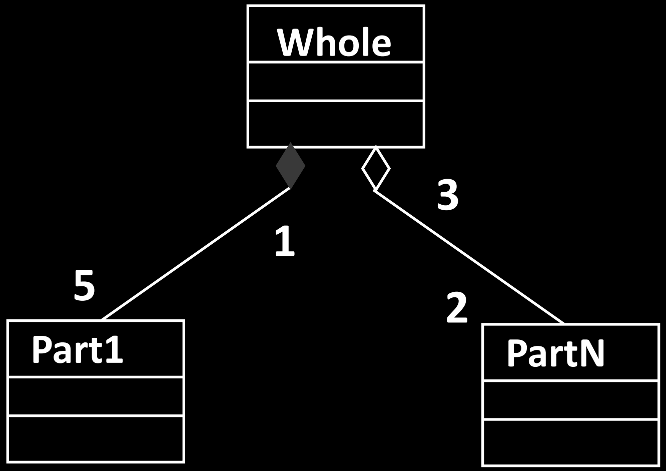


***Example***



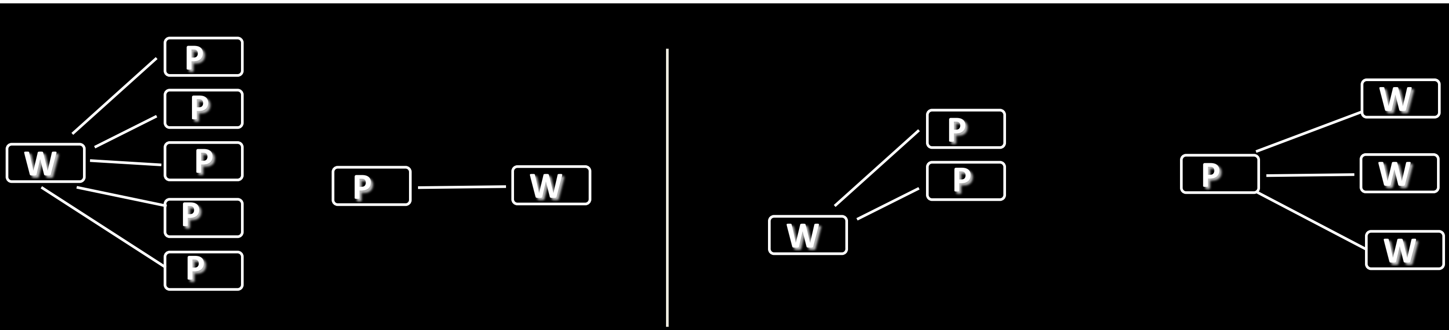
***(association, aggregation & composition look the same)***

# ***Multiplicity Example #1***

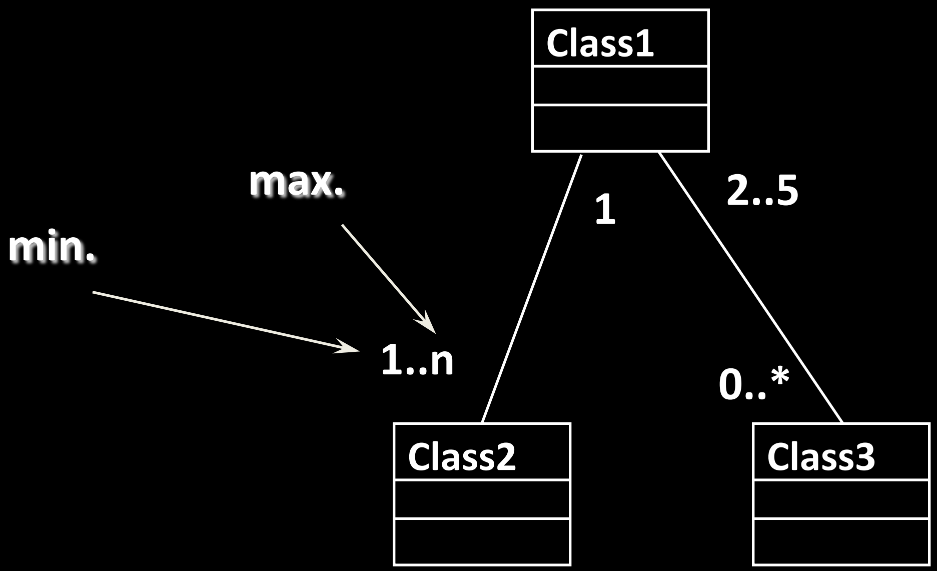


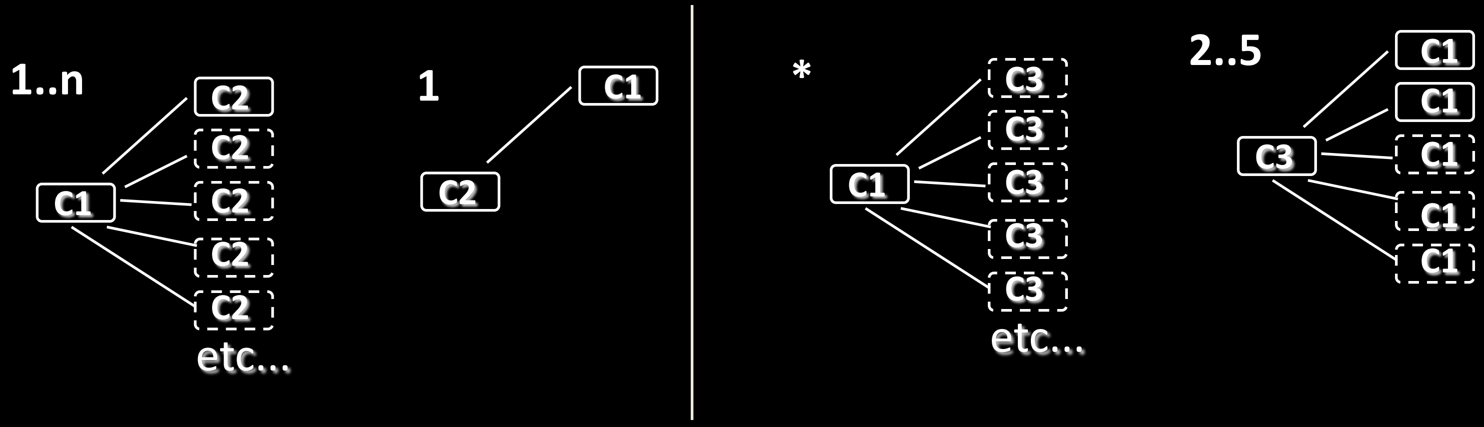
- One Whole is associated with 5 Part1
- One Part1 is associated with 1 Whole

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

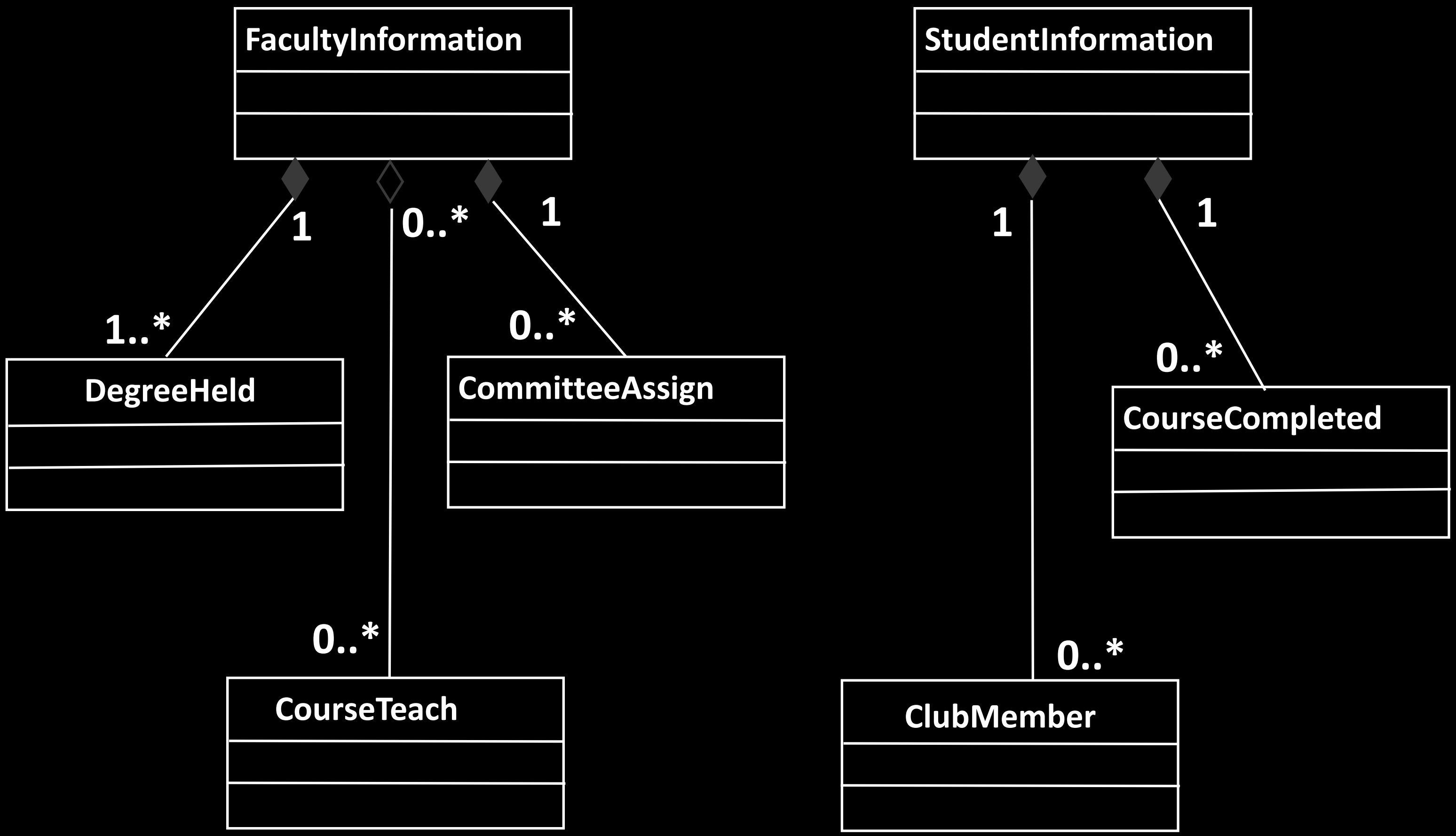


# *Multiplicity Example #2*

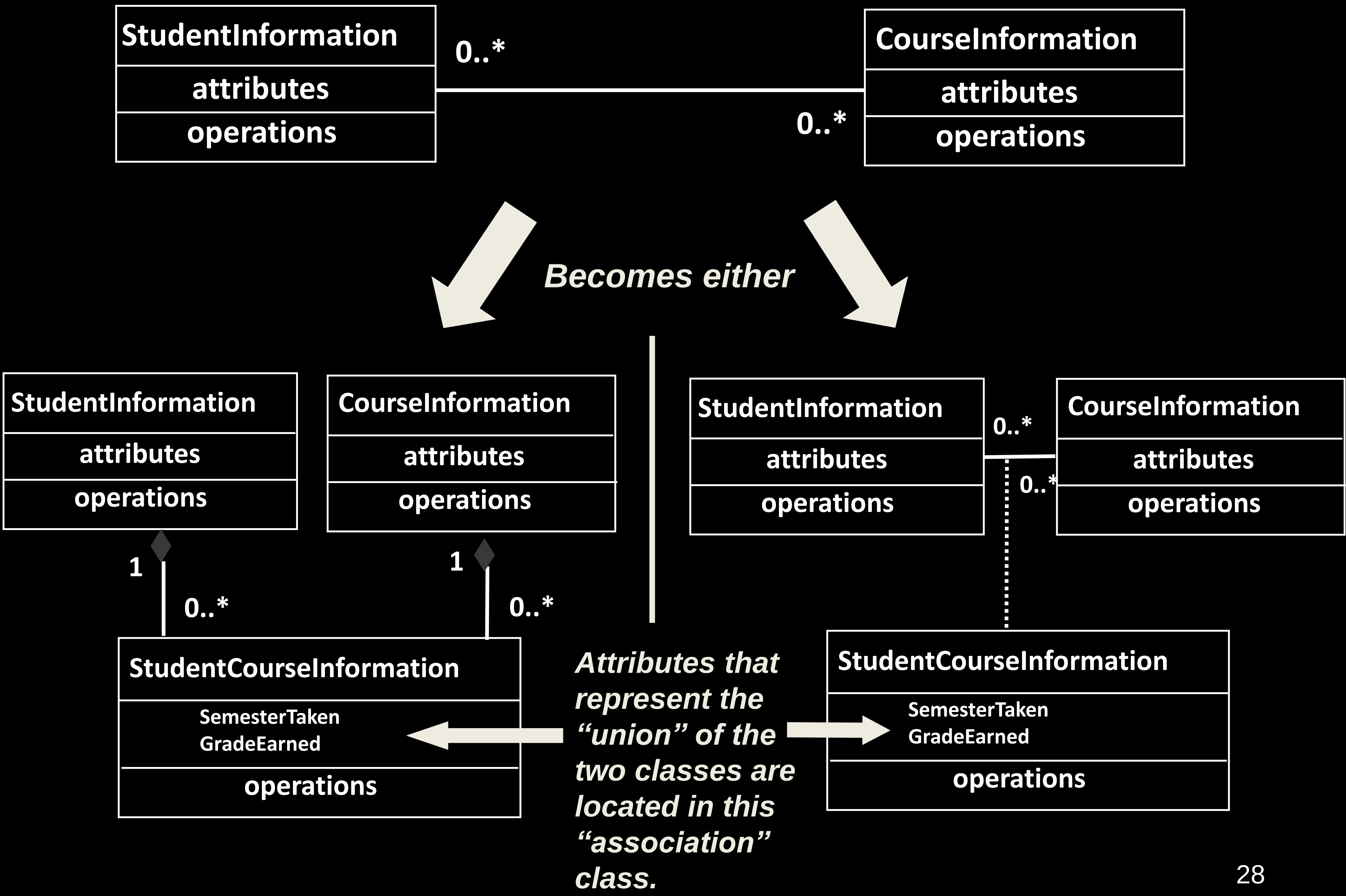




# *Multiplicity Example #3*



# *“many-to-many” multiplicity*

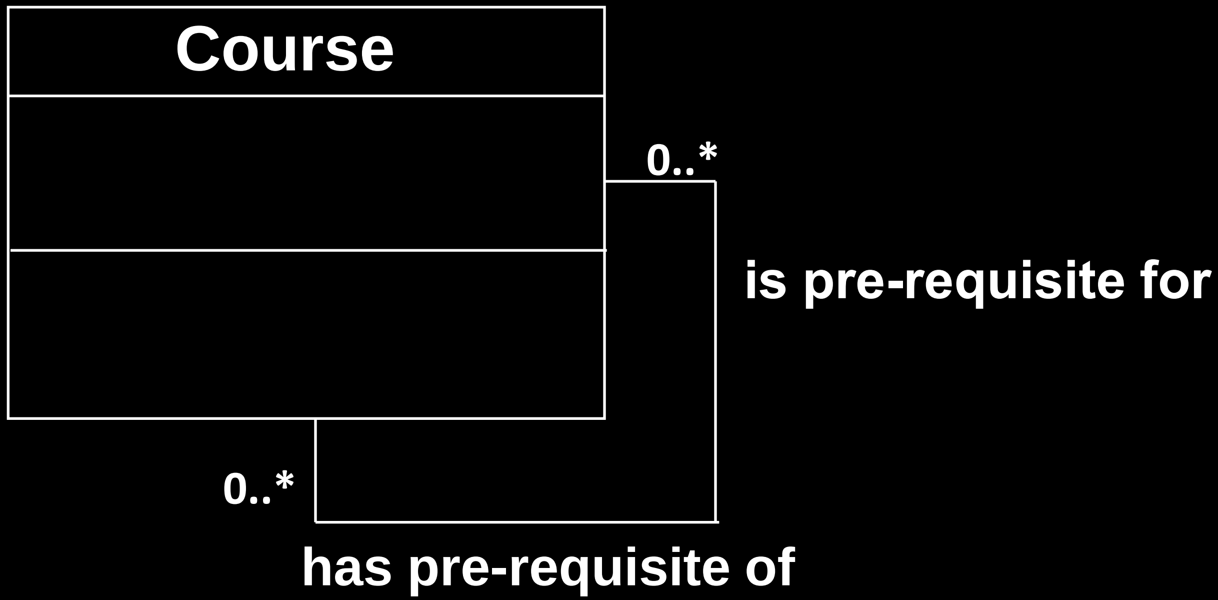


***Becomes either***

***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.***



Course

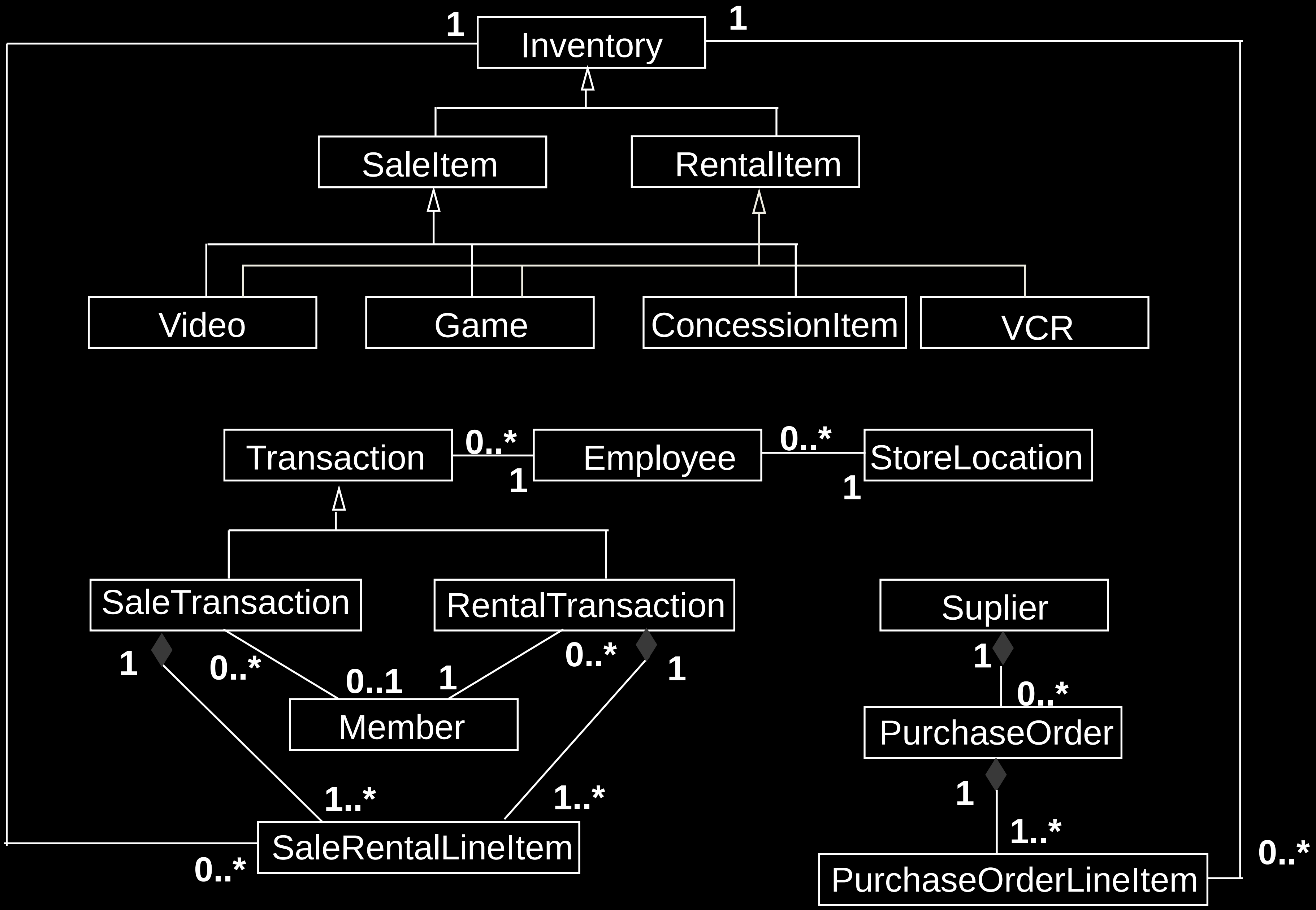
0..*

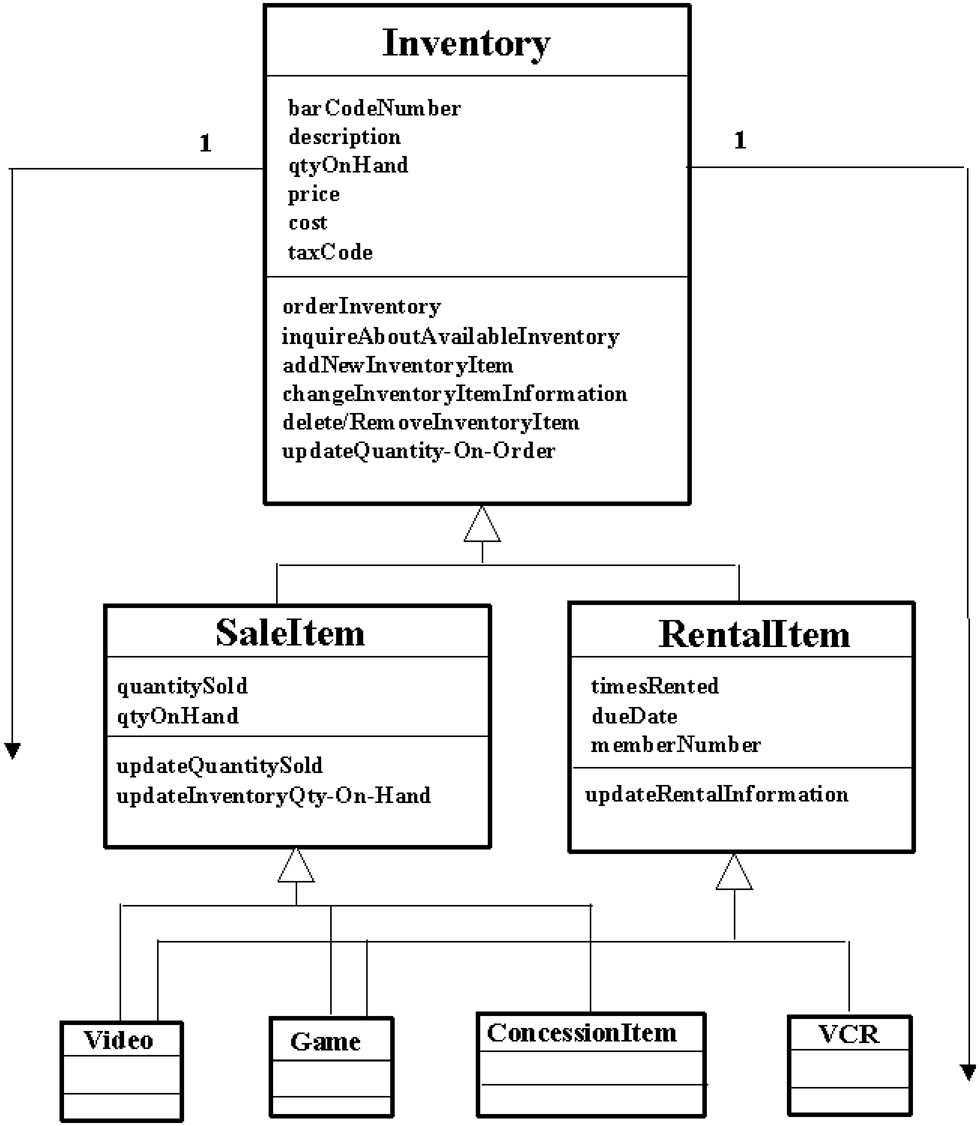
is pre-requisite for

0..*

has pre-requisite of

# Video Store – UML Class Diagram





Inventory

barCodeNumber
description
qtyOnHand
price
cost
taxCode

orderInventory
inquireAboutAvailableInventory
addNewInventoryItem
changeInventoryItemInformation
delete/RemoveInventoryItem
updateQuantity-On-Order

1

1

SaleItem

quantitySold
qtyOnHand

updateQuantitySold
updateInventoryQty-On-Hand

RentalItem

timesRented
dueDate
memberNumber

updateRentalInformation

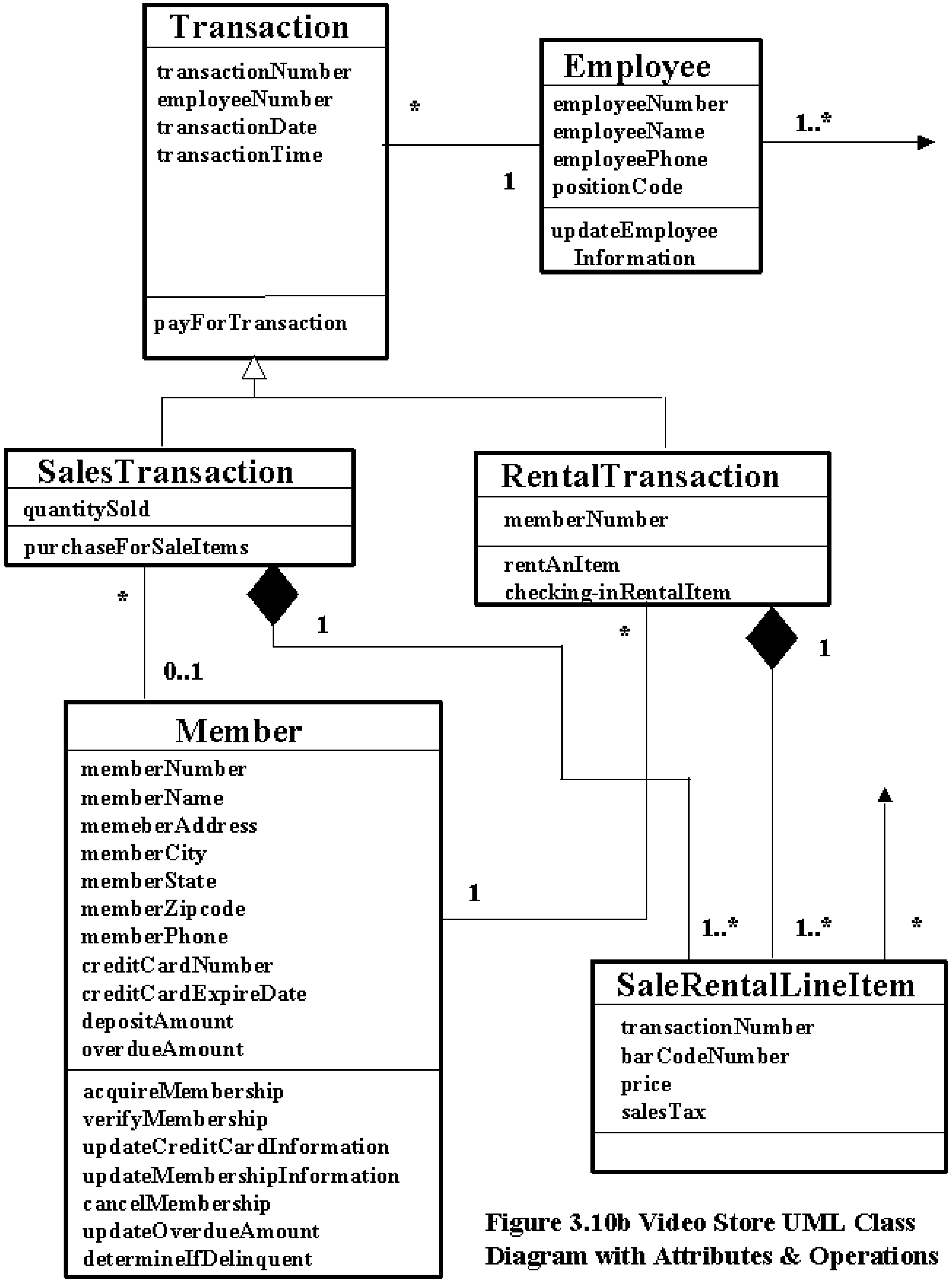
Video

Game

ConcessionItem

VCR

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



Transaction

transactionNumber
employeeNumber
transactionDate
transactionTime

payForTransaction

Employee

employeeNumber
employeeName
employeePhone
positionCode

updateEmployee Information

SalesTransaction

quantitySold

purchaseForSaleItems

RentalTransaction

memberNumber

rentAnItem
checking-inRentalItem

Member

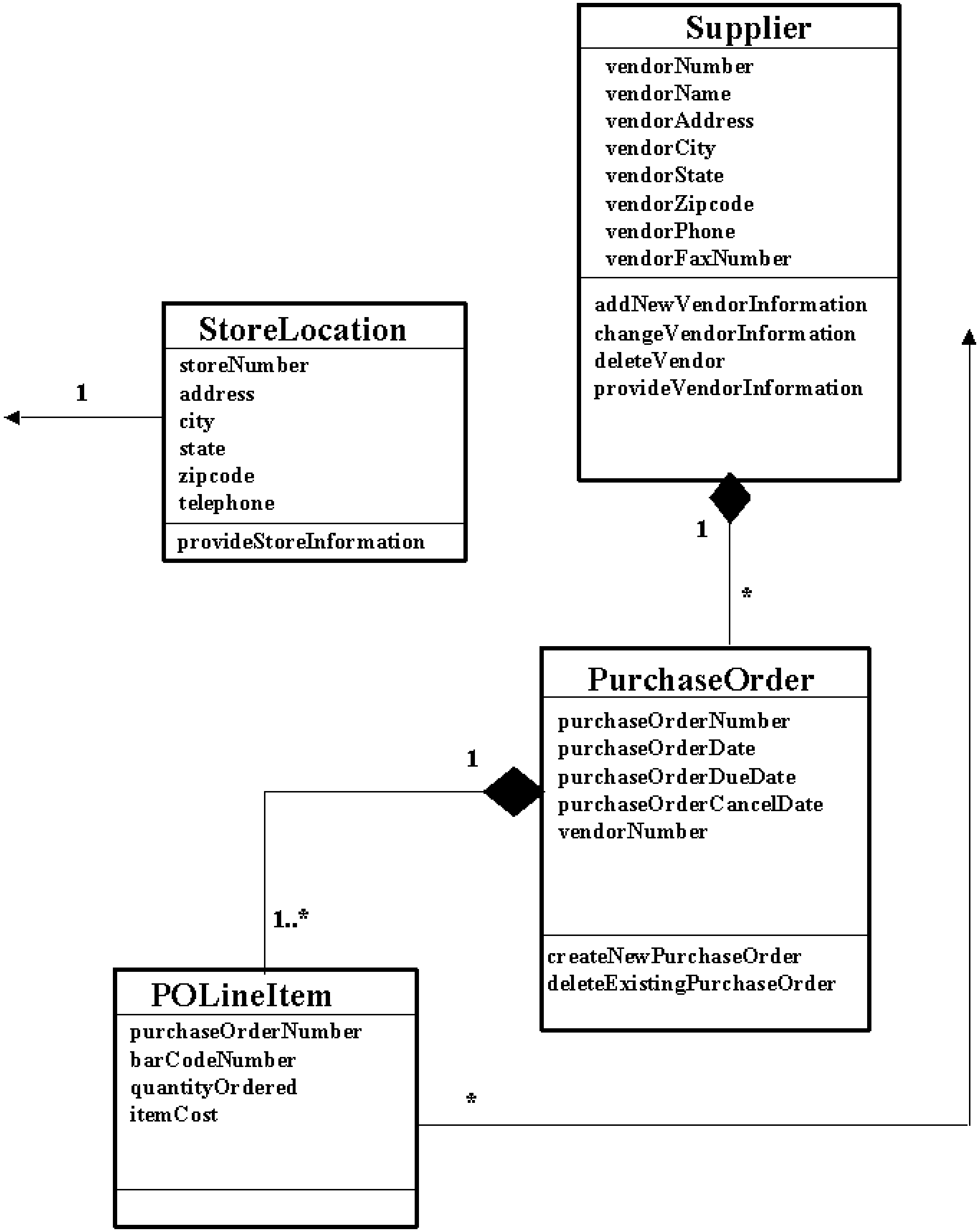
memberNumber
memberName
memeberAddress
memberCity
memberState
memberZipcode
memberPhone
creditCardNumber
creditCardExpireDate
depositAmount
overdueAmount

acquireMembership
verifyMembership
updateCreditCardInformation
updateMembershipInformation
cancelMembership
updateOverdueAmount
determineIfDelinquent

SaleRentalLineItem

transactionNumber
barCodeNumber
price
salesTax

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



**Supplier**

- vendorNumber
- vendorName
- vendorAddress
- vendorCity
- vendorState
- vendorZipcode
- vendorPhone
- vendorFaxNumber

- addNewVendorInformation
- changeVendorInformation
- deleteVendor
- provideVendorInformation

**StoreLocation**

- storeNumber
- address
- city
- state
- zipcode
- telephone

- provideStoreInformation

**PurchaseOrder**

- purchaseOrderNumber
- purchaseOrderDate
- purchaseOrderDueDate
- purchaseOrderCancelDate
- vendorNumber

- createNewPurchaseOrder
- deleteExistingPurchaseOrder

**POLineItem**

- purchaseOrderNumber
- barCodeNumber
- quantityOrdered
- itemCost

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