Class Modeling

COMP 3700
Software Modeling and Design

Shehenaz Shaik

Discussed so far …

A brief overview of

- Software Modeling and Design
  - Software
  - Design
  - Model
    - Object Oriented Approach
    - UML Representation

Object Oriented Methodology

- Process for OO development.

- Stages:
  - System conception
  - Analysis
  - System design
  - Class design
  - Implementation

OO Models

- Class Model
  - Class diagram
- State Model
  - State diagram
- Interaction Model
  - Use case diagram
  - Sequence diagram
  - Activity diagram

Class Model

- Captures the static structure of a system
  - Objects
  - Relationships
  - Attributes
  - Operations
- Close to Real world
- Resilient to change
- Graphical representation of system

Class Model: Concepts

- Object
- Class
- Link
- Association
- Association class
- Generalization
- Inheritance

- And more…

Class Model: Object

- An object is a concept, abstraction, or thing
  - Has identity
  - Is distinguishable
  - has meaning for an application
- How to identify
  - Proper nouns or specific references
  - Choice of objects depends on nature of problem

Class Model: Class

- Group of objects with similar
  - Properties
  - Behavior
  - Relationships
  - Semantics
- How to identify?
  - Common nouns or noun phrases
  - Choice of classes depends on nature and scope of problem

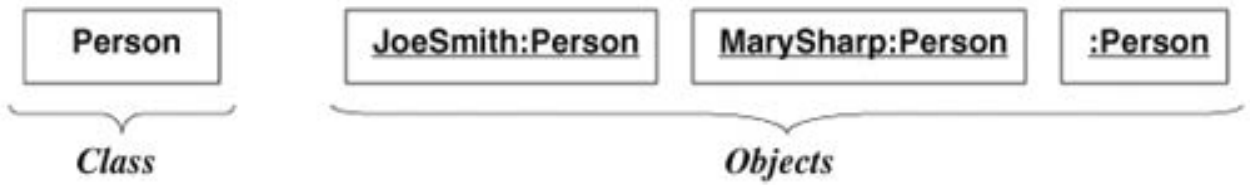
What is the need for Classes?

- By grouping objects into class,
  - Abstract a problem
  - Attribute names are stored once per class
  - Define operations per class
    - Code reuse

Class Model: Representations

- Class Diagram
  - Models classes and their relationships
  - Describes possible objects
  - Represent structure of applications
  - Useful for
    - Abstract modeling
    - Designing actual programs
- Object Diagram
  - Individual objects and their relationships
  - Useful for
    - Documenting test cases
    - Discussing examples
- Class diagram corresponds to infinite set of object diagrams

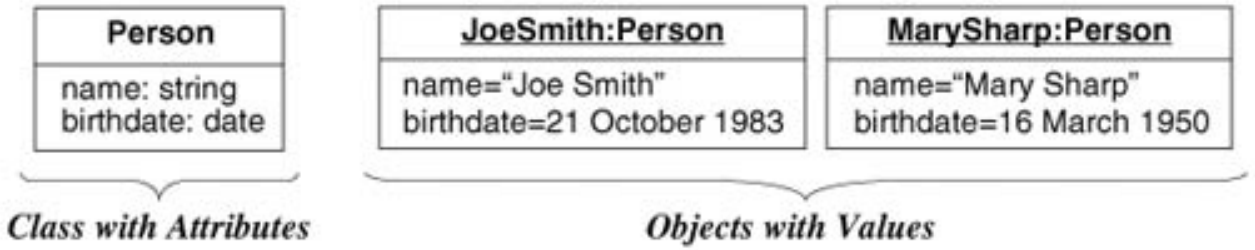
UML Notation: Class, Object

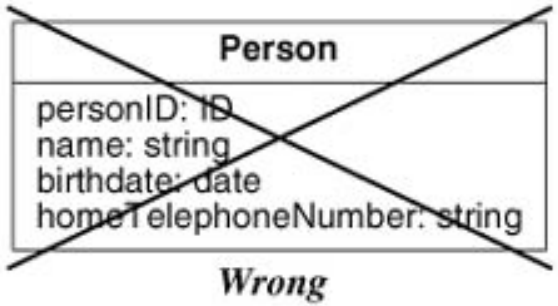


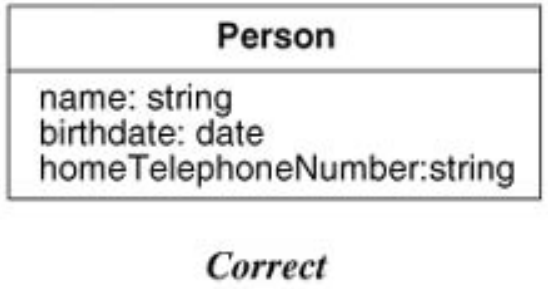
Class Model: Value, Attribute

- Value
  - Piece of data
  - How to identify?
    - Examples
- Attribute
  - Named property of a class
  - How to identify?
    - Adjectives, or by abstracting typical values
  - Unique attribute name within class
- Values lack identity

UML Notation: Value, Attribute



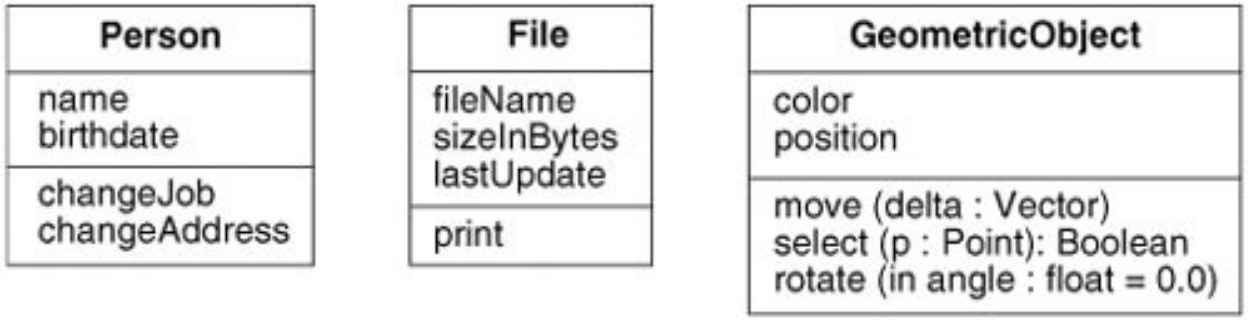




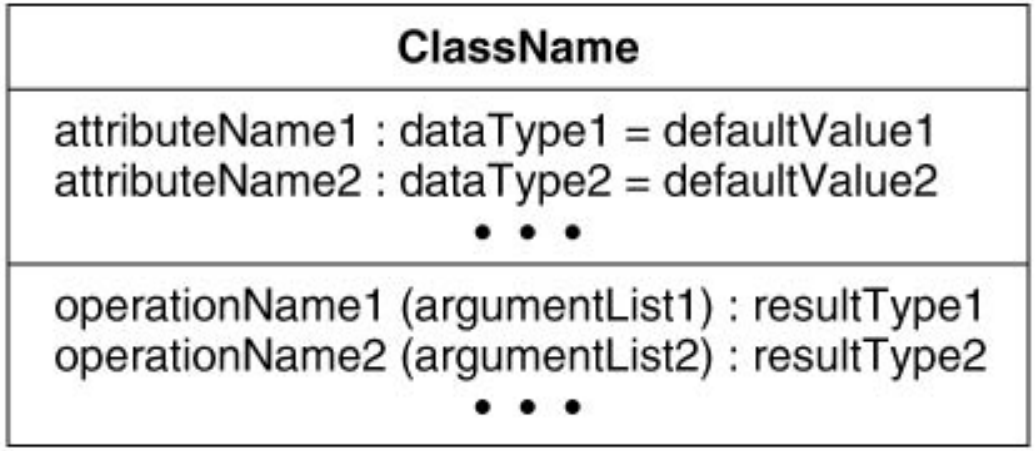
Class Model: Operation, Method

- Operation
  - Behavior of objects in a class
  - Objects share operations
- Method
  - Implementation of operation
- Polymorphic operation
  - Same operation takes different forms in different classes
  - Ensure same signature

UML Notation: Operations



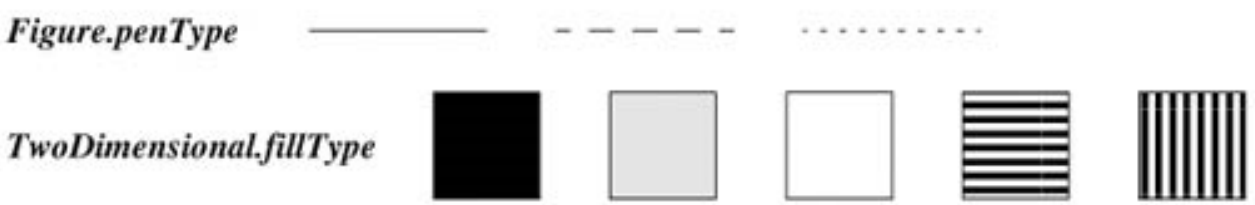
UML Notation: Class diagram



direction argumentName : type = defaultValue

Enumeration

- Data type with finite set of values



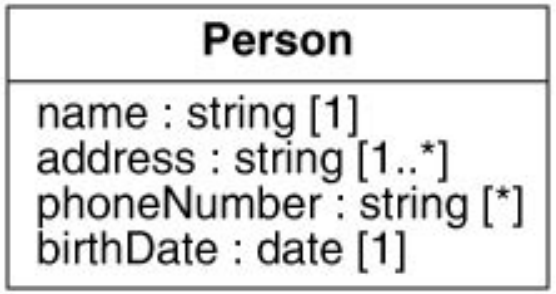
UML Notation: Enumeration

- UML Notation: << enumeration >>

Multiplicity: Attribute

- Number of possible values
  - [1] – Mandatory single value
  - [0..1] – Optional single value
  - [*] – any number of values
  - …
- Default: [1]

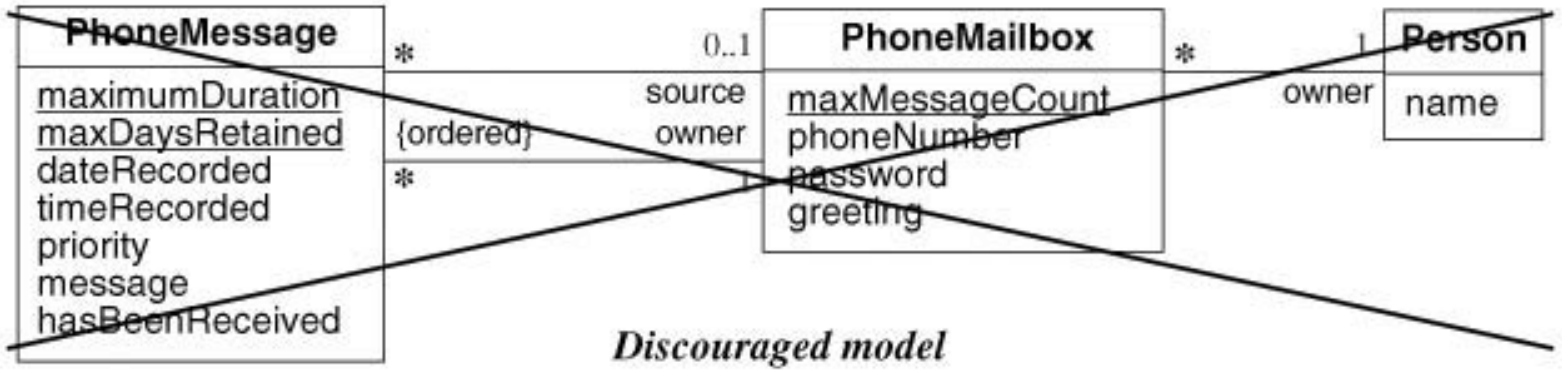
UML Notation



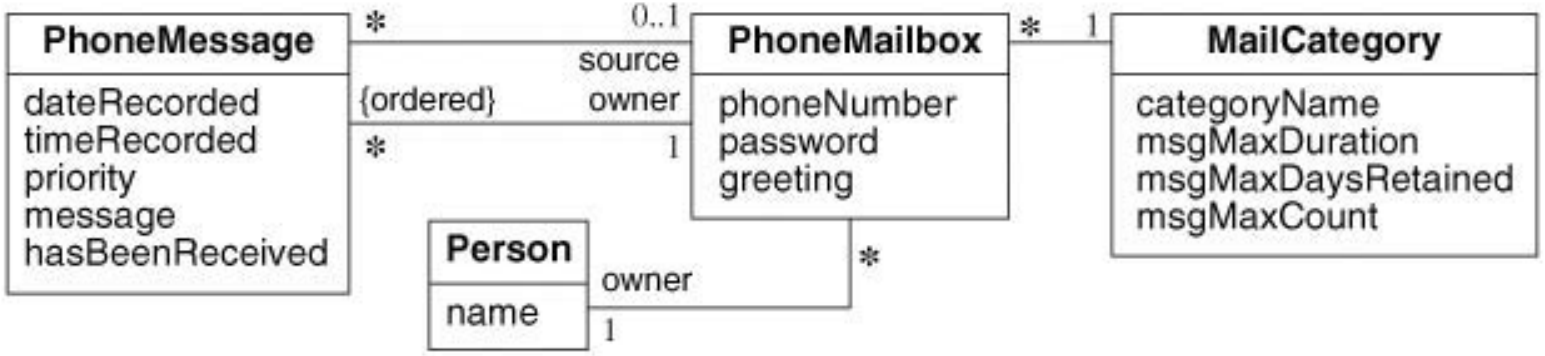
Scope

- Applicability of feature
  - Object scope
  - Class scope (Static feature)
    - UML Notation: Underline & Top of box
    - Attributes
    - Operations

Scope (Contd.)



Discouraged model



Preferred model

Visibility

- Ability of a method to reference a feature from another class
  - Public (+)
    - Any method can access
  - Private (-)
    - Methods of containing class
  - Protected (#)
    - Methods of containing class and its descendants via inheritance
  - Package (~)
    - Methods of classes defined in same package as target class
- UML Notation (shown in brackets)

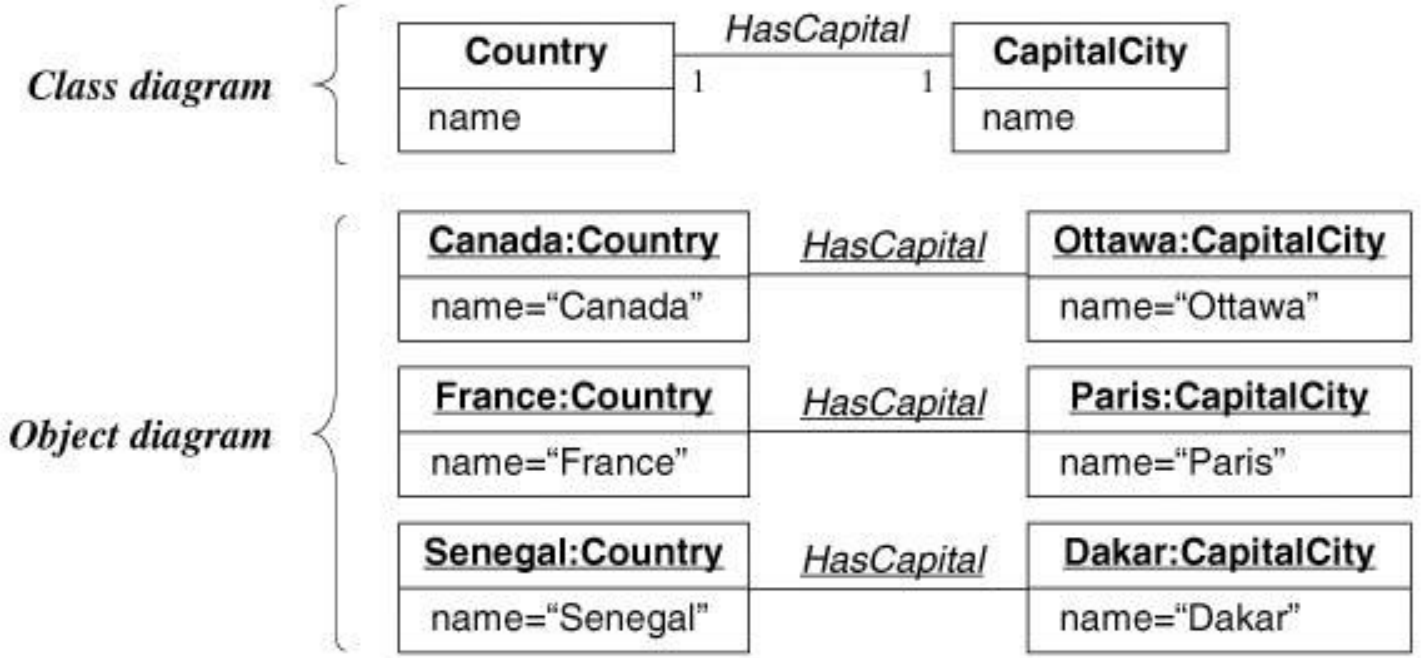
Class Model: Link, Association

- Link
  - Physical or conceptual connection among objects
  - An instance of an association
- Association
  - Description of a group of links with common structure and semantics
- How to identify?
  - Verbs

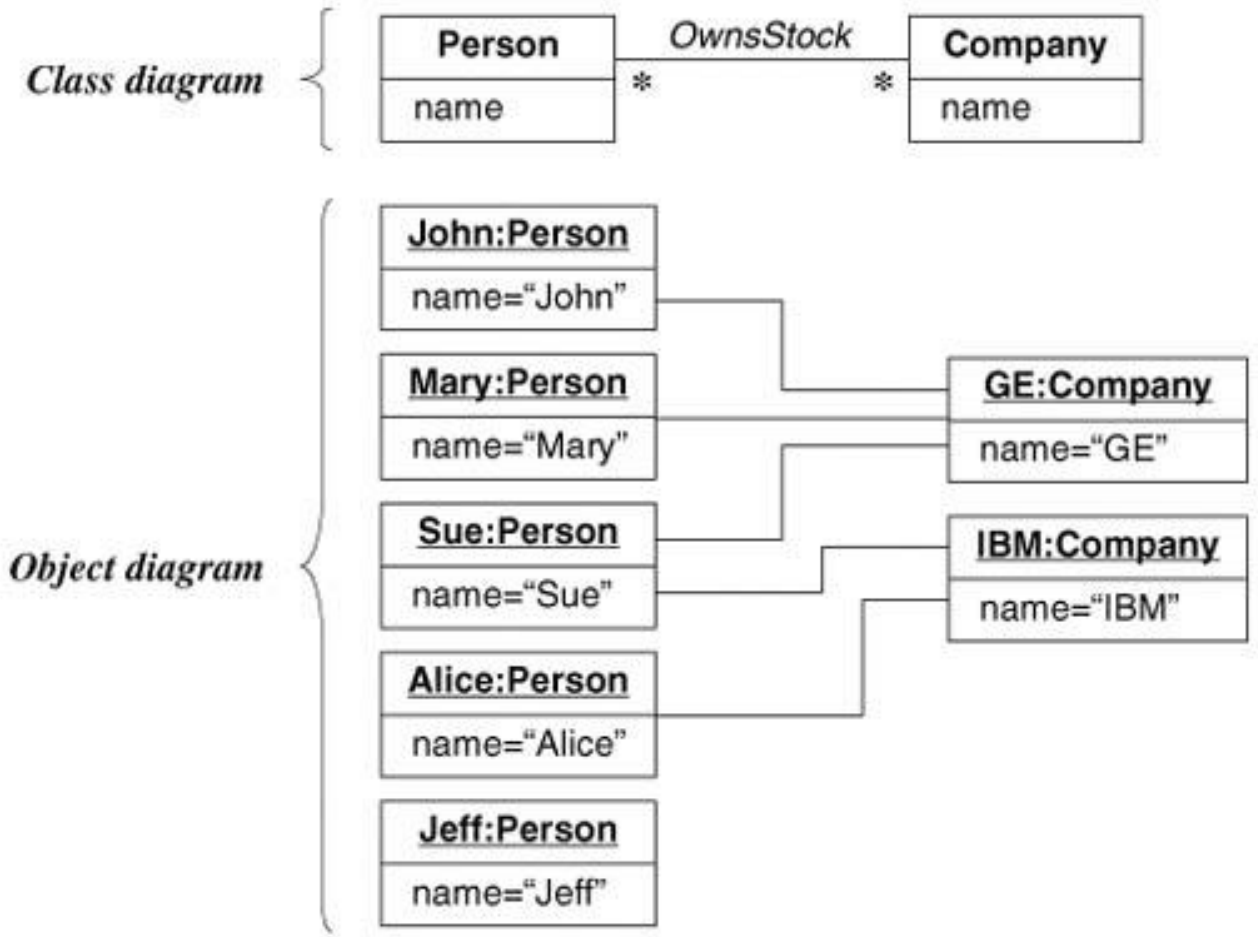
Association: Types

- Many-to-many Association
- One-to-many Association
- One-to-one Association

One-to-one Association



Many-to-many Association

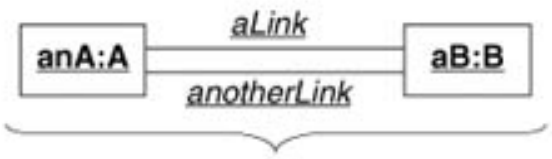


UML Notation: Association

Class diagram

Object diagram

Class diagram



Object diagram

Link Vs. Reference

- Link cannot be modeled as a reference
- Link is not part of either object by itself, but depends on both of them together
- Associations cannot be private to a class

Multiplicity

- Number of instances of one class that may relate to a single instance of an associated class
  - Shown at the ends of association lines; E.g.: 1, *, 2..*, 3..5
- Depends on assumptions and problem boundaries
- Exposes hidden assumptions built into the model
- Underestimation risk
- Overestimation risk

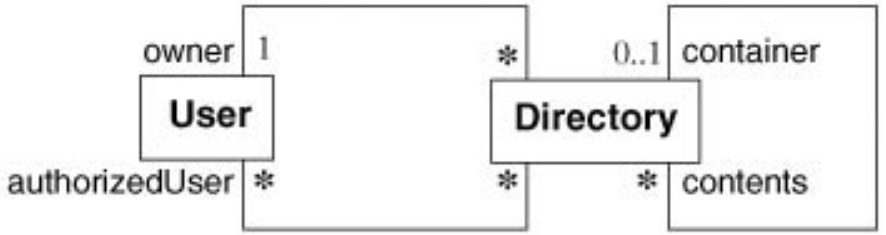
Association End Name

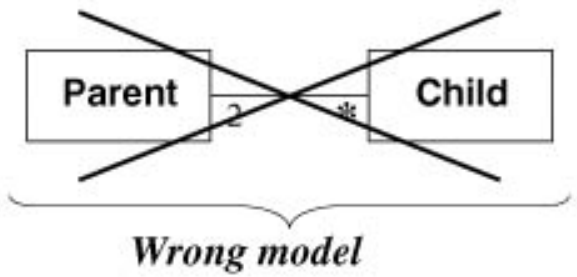
- Association End
  - Name, Multiplicity
  - Refer to an object or set of objects
- How to identify?
  - Appear as 'Nouns'
- Helpful in traversing
- Distinguish multiple associations between same pair of classes
- End name must be unique within the set of attributes of destination class

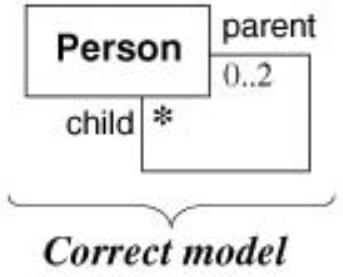
UML Notation: Association End Name

employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Association End Name (Contd.)





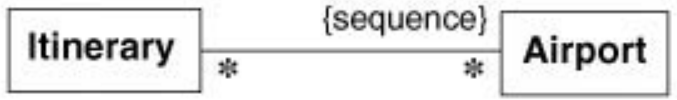


Association: Ordering

- Default
  - No specific order among objects at 'Many' end
  - Regarded as 'Set' of objects
- Format: {ordered}

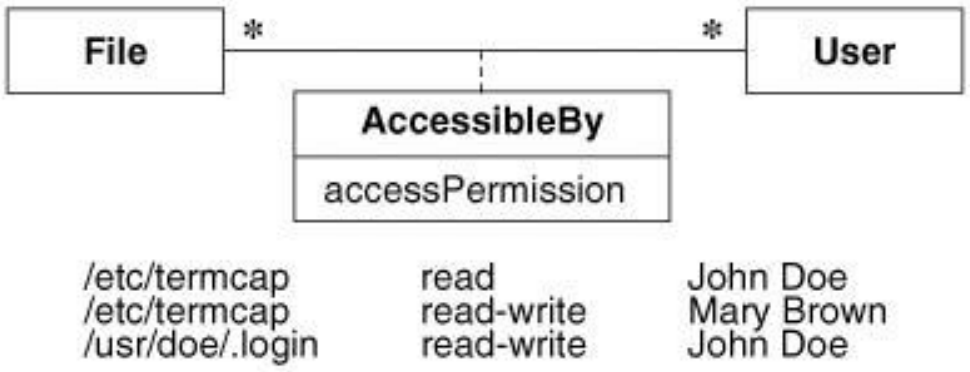
Association: Bag, Sequence

- Multiple links for a pair of objects in an association
- Bag
  - Collection of elements with duplicates allowed
  - Format: {bag}
- Sequence
  - Ordered collection of elements with duplicates allowed
  - Format: {sequence}

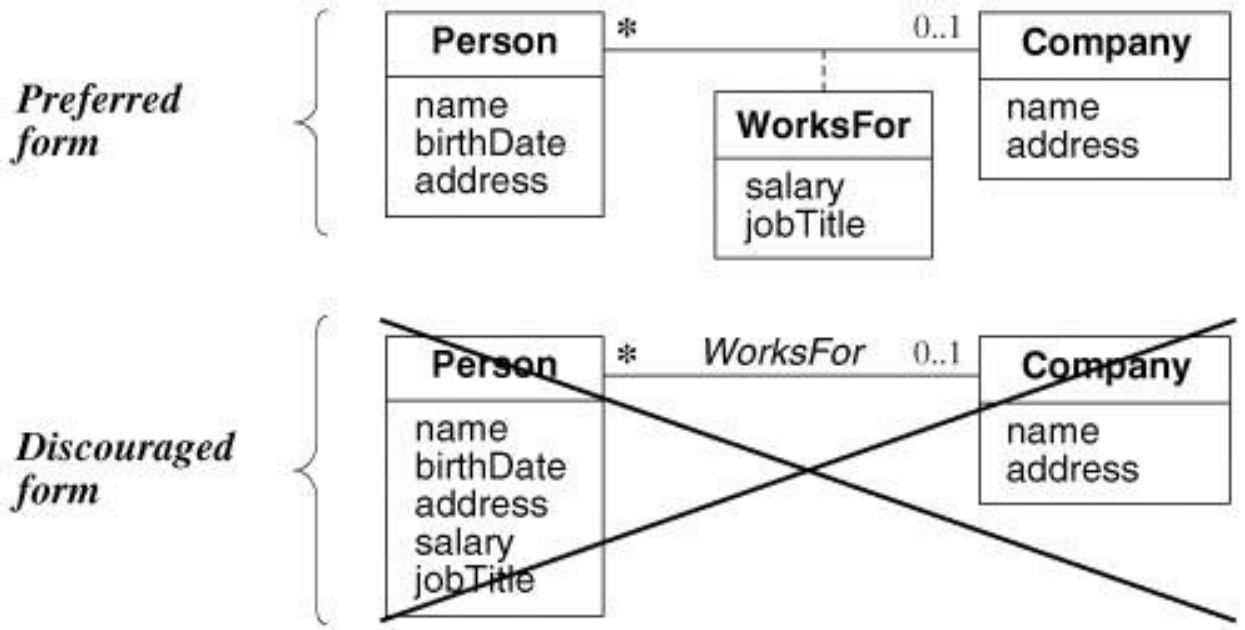


Association class

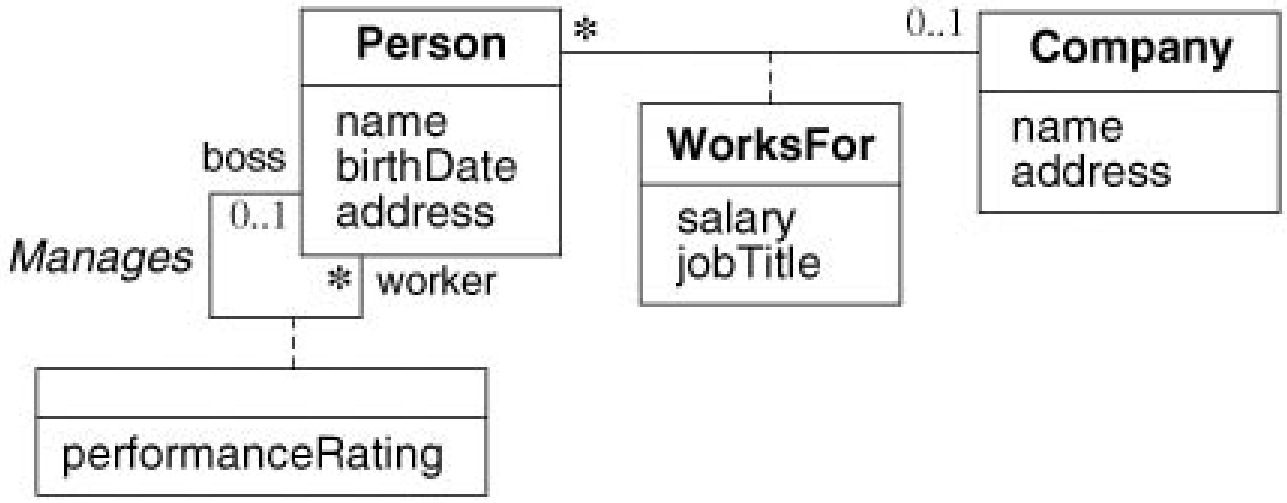
- Association that is also a class
  - Attributes, Operations, Associations
- Instances derive identity from constituent class instances
- How to identify?
  - Adverbs, or by abstracting typical values
- Help modeling many-to-many associations
- UML Notation
  - Attached to link by dashed line



Association Class (Contd.)

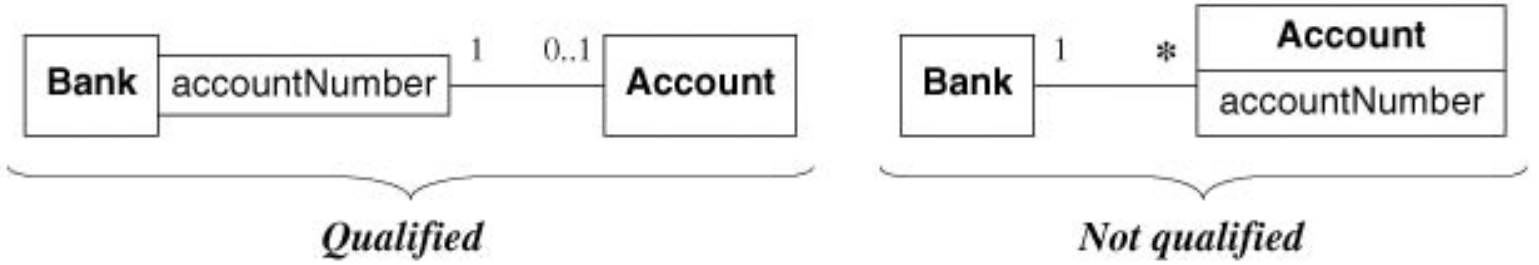


Association Class (Contd.)



Qualified Association

- 'Qualifier' attribute disambiguates the objects for a 'many' association end
- Qualifier
  - Selects among target objects
  - Specifies precise path for finding target object from source object
  - Adds multiplicity constraint
  - Reduces effective multiplicity from 'many' to 'one'
- Applicability:
  - One-to-many
  - Many-to-many



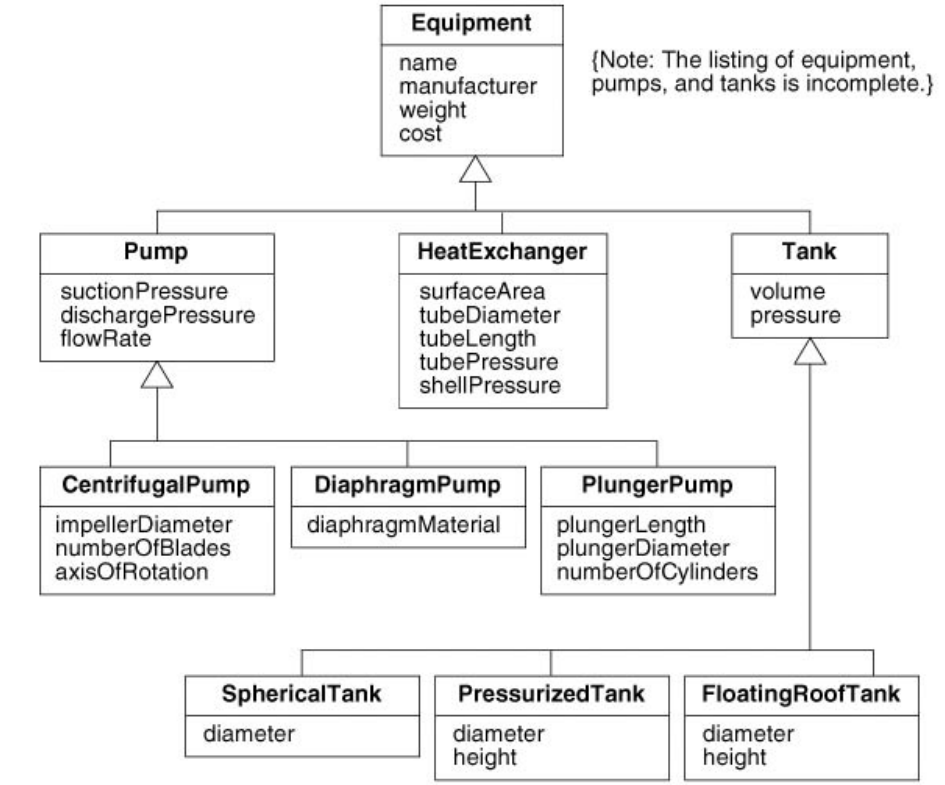
- UML Notation
  - Small box attached to source class near end of association line

Generalization

- Relationship between a class (superclass) and one or more variations of the class (subclasses)
- Organizes classes by similarities and differences
- Subclass inherits features of superclass
  - Superclass: common features
  - Subclass: specific features
  - Attributes, operations, associations
- IS-A relationship
  - Subclass instance IS-A Superclass instance

- UML Notation: Large hollow arrowhead

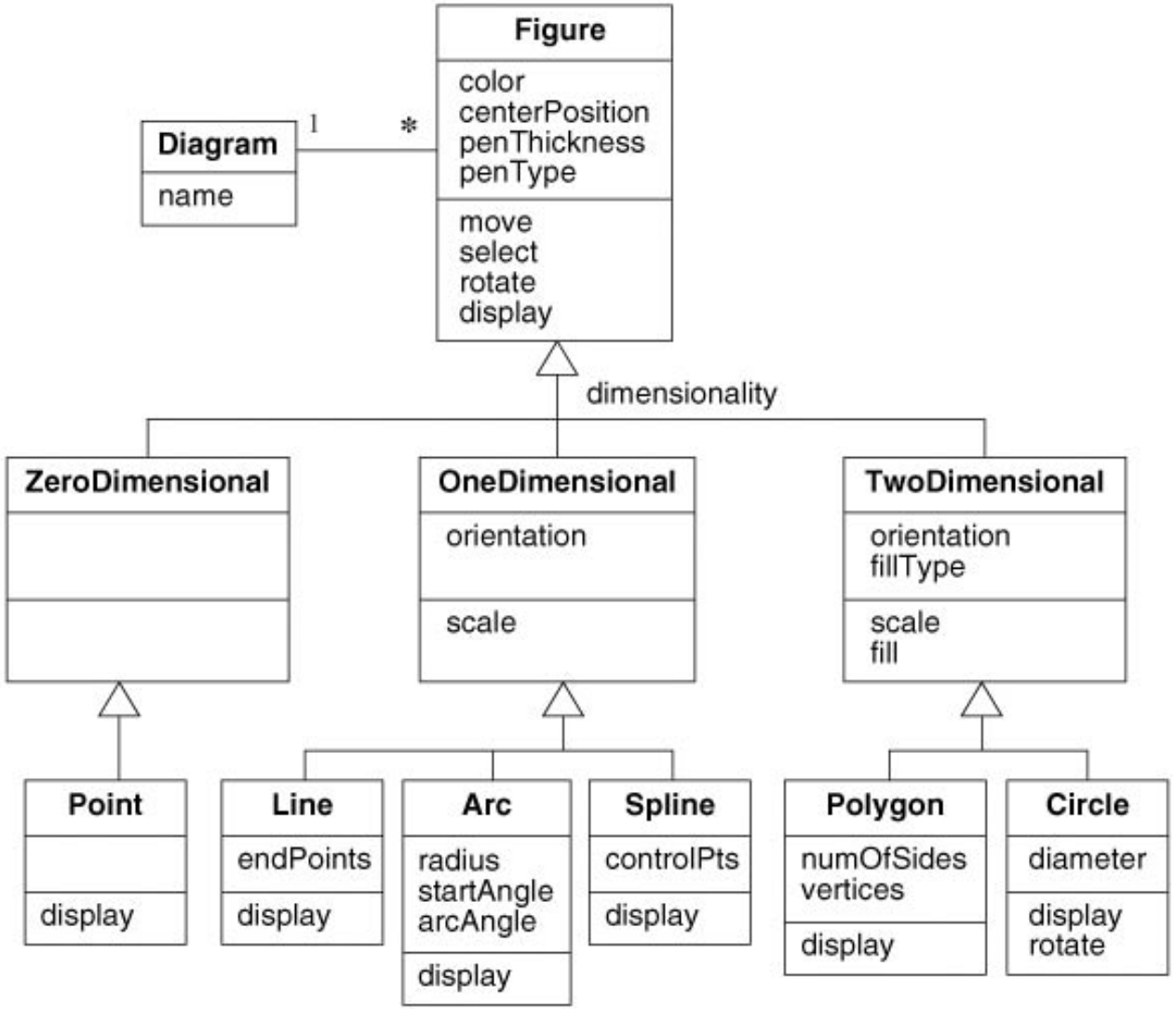
E.g. Multilevel inheritance hierarchy



Generalization set name

- Enumerated attribute that indicates which aspect of an object is being abstracted
  - Generalize only one aspect at a time
  - Set values have 1-to-1 correspondence with subclasses
  - Optional
- Terms
  - Generalization
  - Specialization
  - Inheritance

E.g. Generalization set name

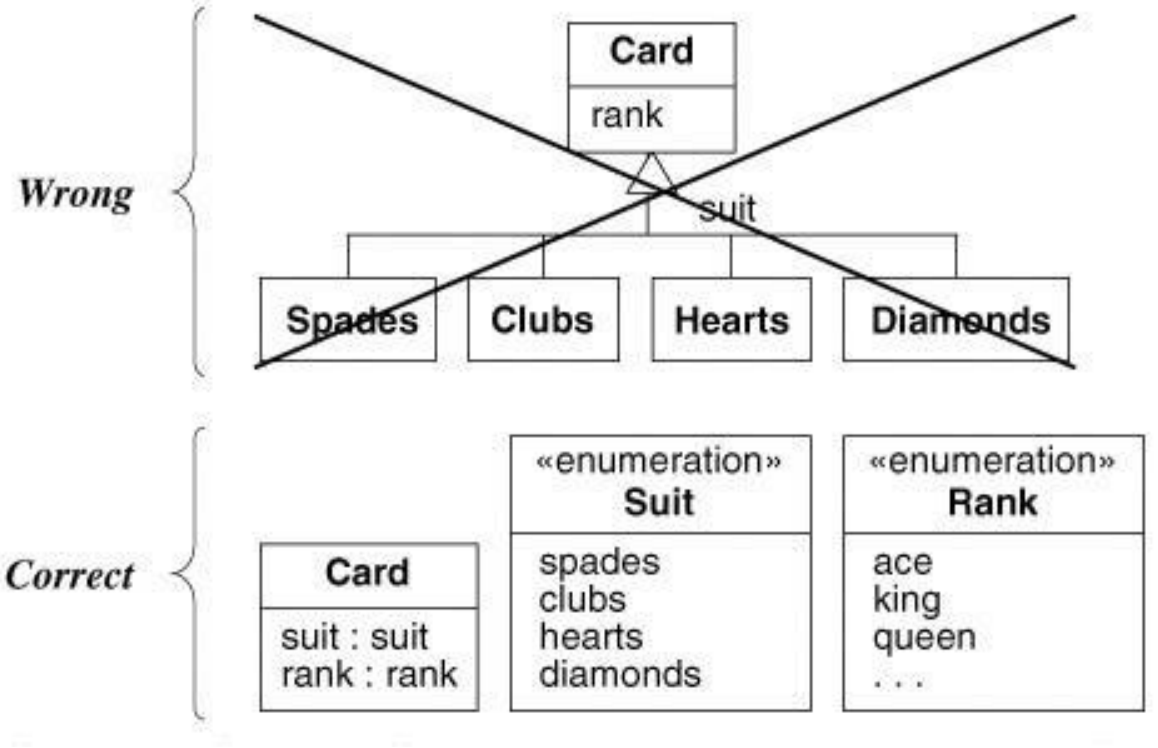


Purpose

- Generalization
  - Support for polymorphism
    - To increase software flexibility
  - To structure the description of objects
  - To enable reuse of code
- Specialization
  - *Overriding*: Refine and replace feature with same signature
  - To specify behavior that depends on subclass
  - To tighten the specification of a feature
  - To improve performance

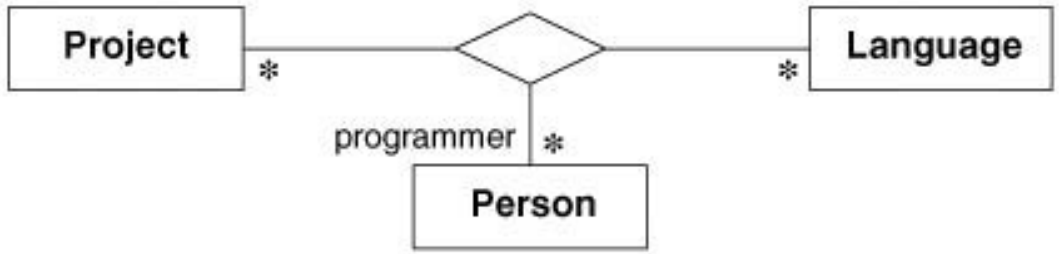
Enumeration: Not a Generalization

- Enumeration
  - List of values
- Generalization
  - Subclass has significant features that do not apply to superclass



N-ary Association

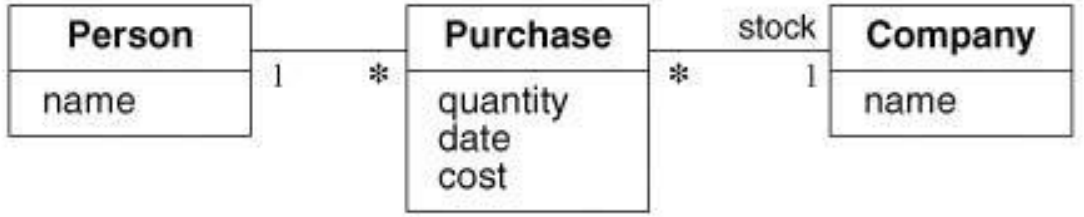
- Binary associations
  - Among 2 classes
- N-ary associations
  - among >=3 classes
  - Try to decompose with qualifiers / attributes
  - UML Notation
    - Diamond with lines connecting relating classes
    - Optional name



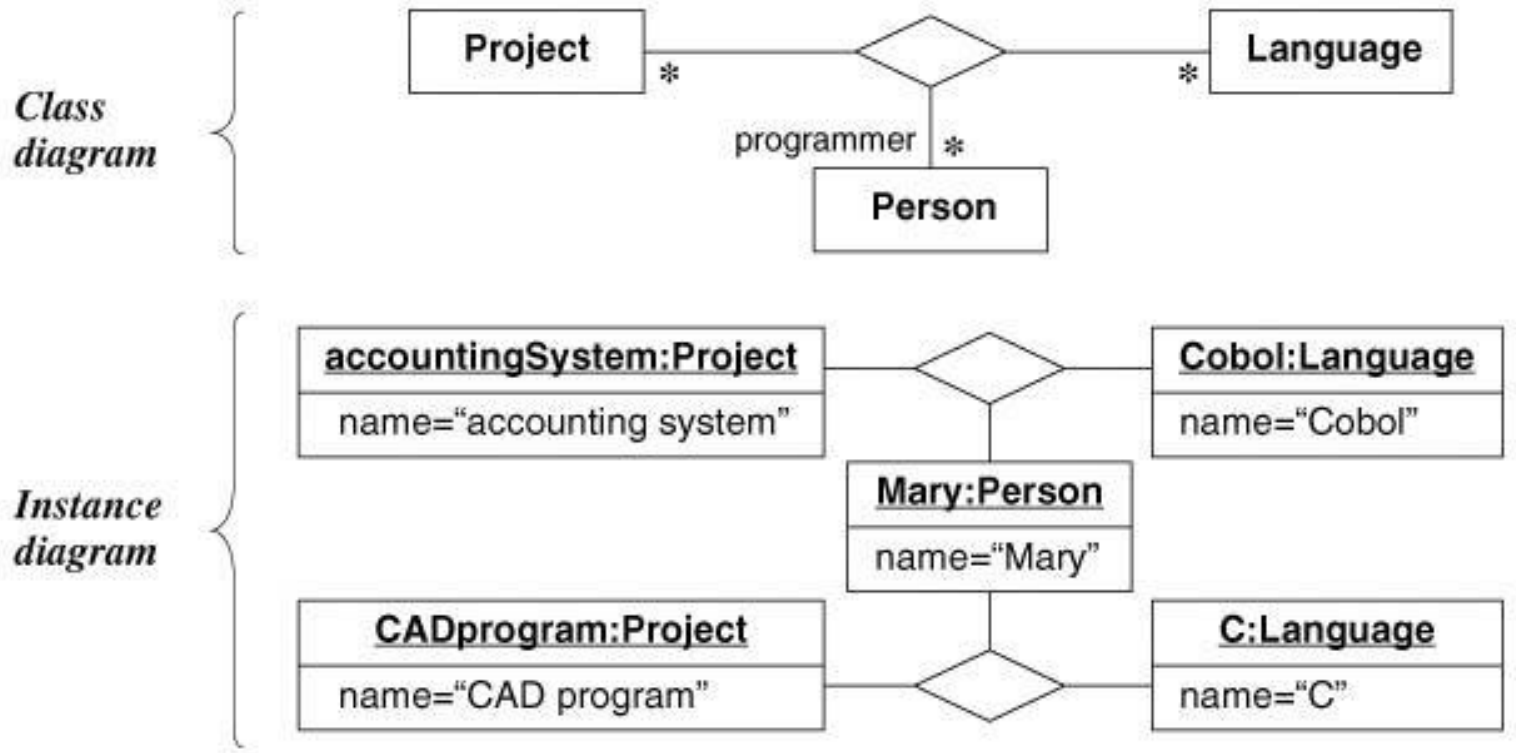
Non-Atomic N-ary Association

A nonatomic n-ary association—a person makes the purchase of stock in a company...

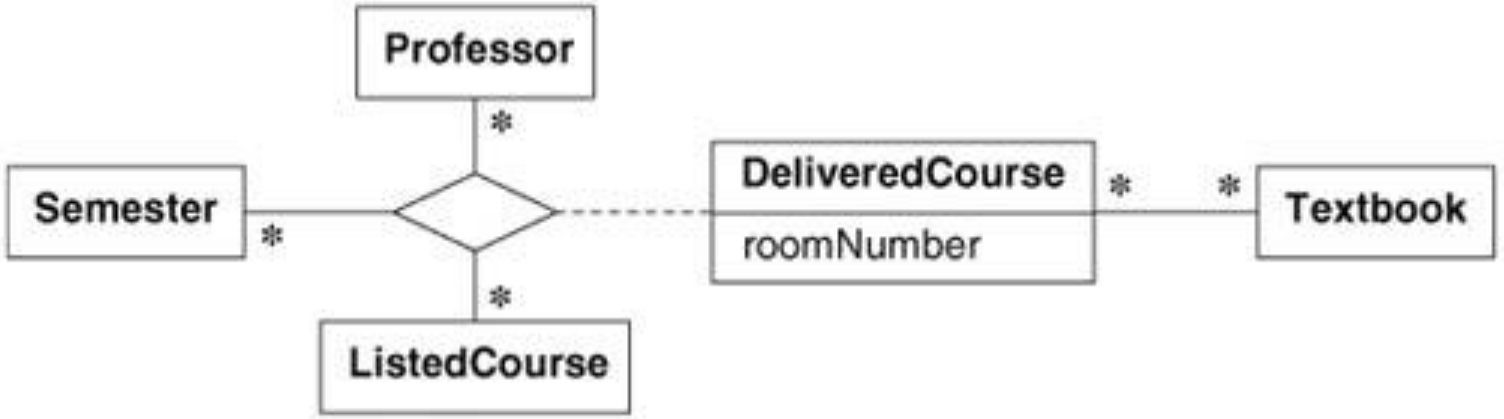
Can be restated as...



Genuine N-ary Association

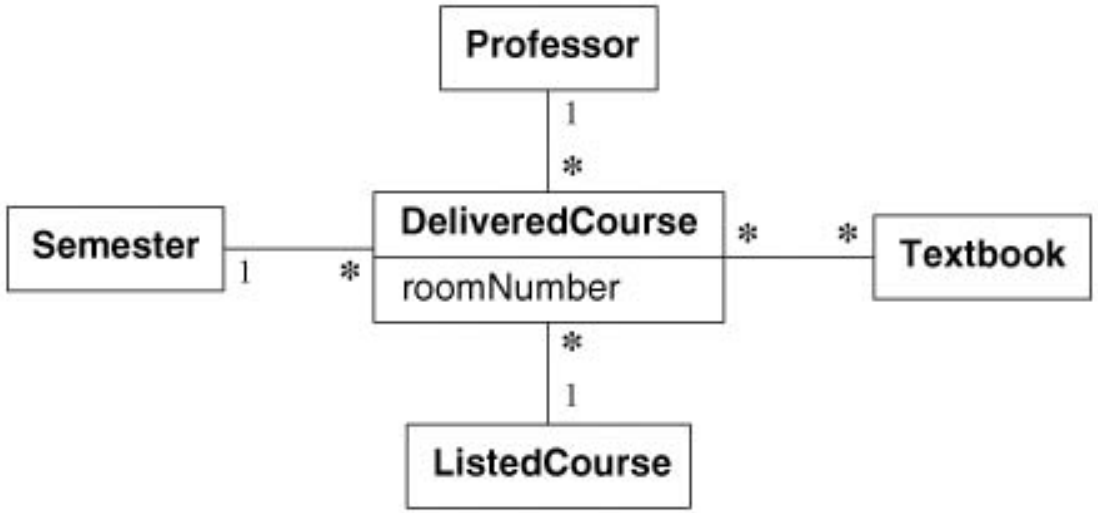


Genuine N-ary Association (2)



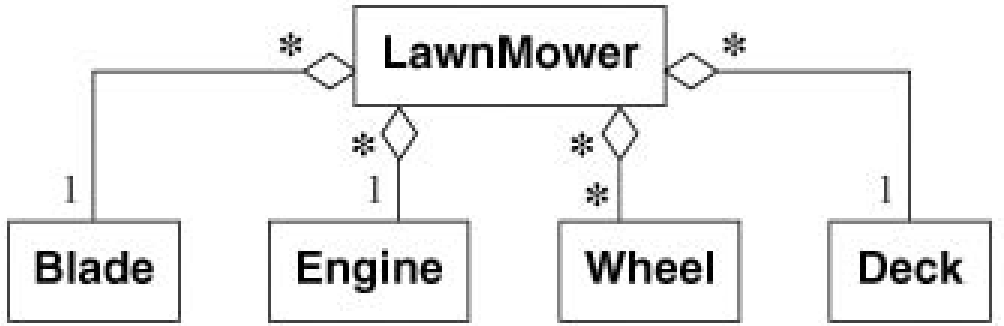
Promoting N-ary Association to classes

- No support by programming languages



Aggregation

- Relating an assembly class to one constituent part class
  - Whole/Part relationship
  - Each individual class pairing is an aggregation
  - Multiplicity specified individually
  - Special form of binary association
- Properties
  - Transitivity
  - Antisymmetric

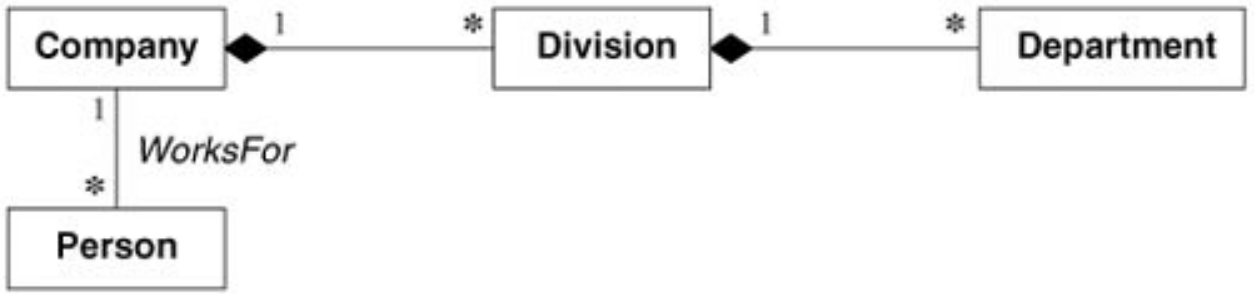


- UML Notation
  - Small hollow diamond at assembly end of association

Composition

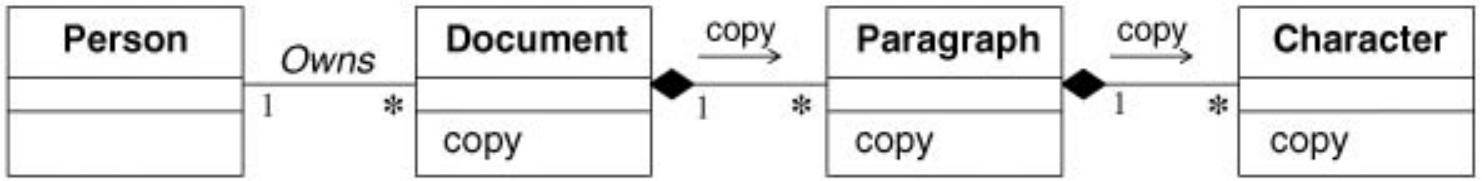
- Restrictive Whole/part relationship
- Aggregation with two constraints:
  - Constituent part can belong to only one assembly
  - Constituent part has coincident lifetime with assembly

- UML Notation
  - Small solid diamond at assembly end of association



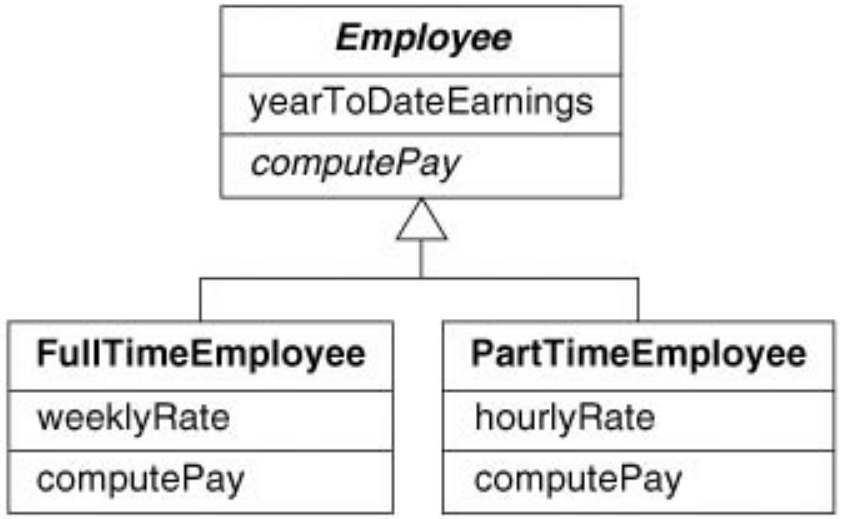
Propagation (Triggering) of operations

- Automatic application of an operation to a network of objects when the operation is applied to some starting object
  - Good indicator of aggregation
  - Propagates only in specified direction
- UML Notation
  - Small arrow indicating direction and operation next to affected association



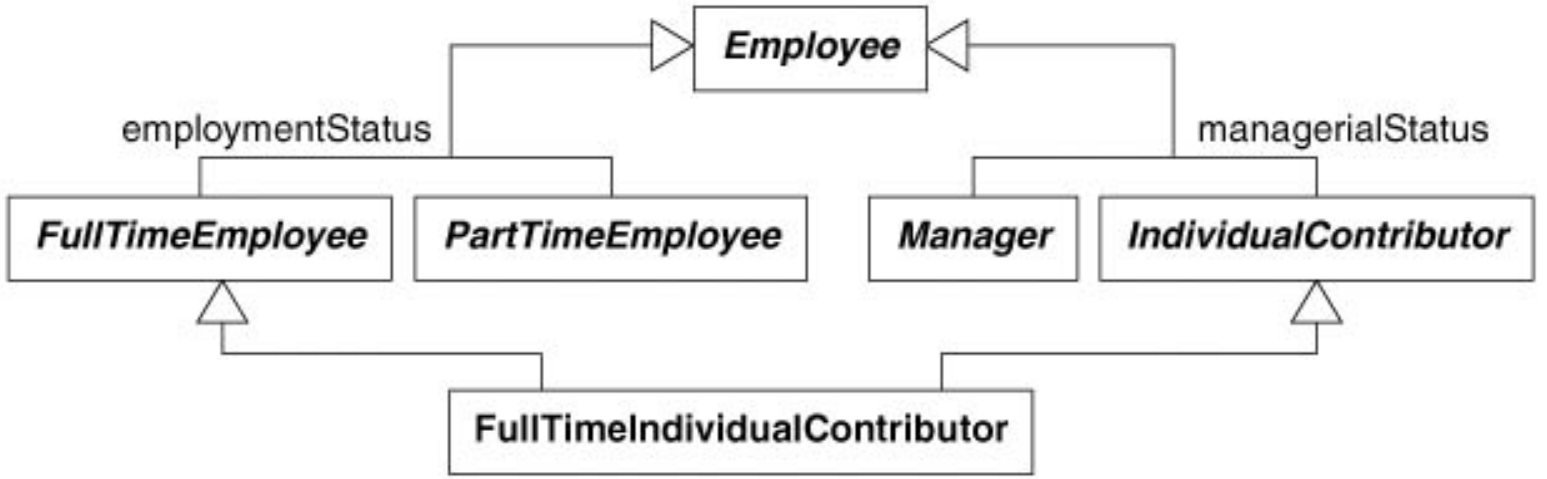
Abstract class

- Class with no direct instances
  - Descendant classes may be concrete classes, i.e. instantiable
- Abstract operation
  - Abstract class defines signature
  - Concrete class provides method
- UML Notation: Italics



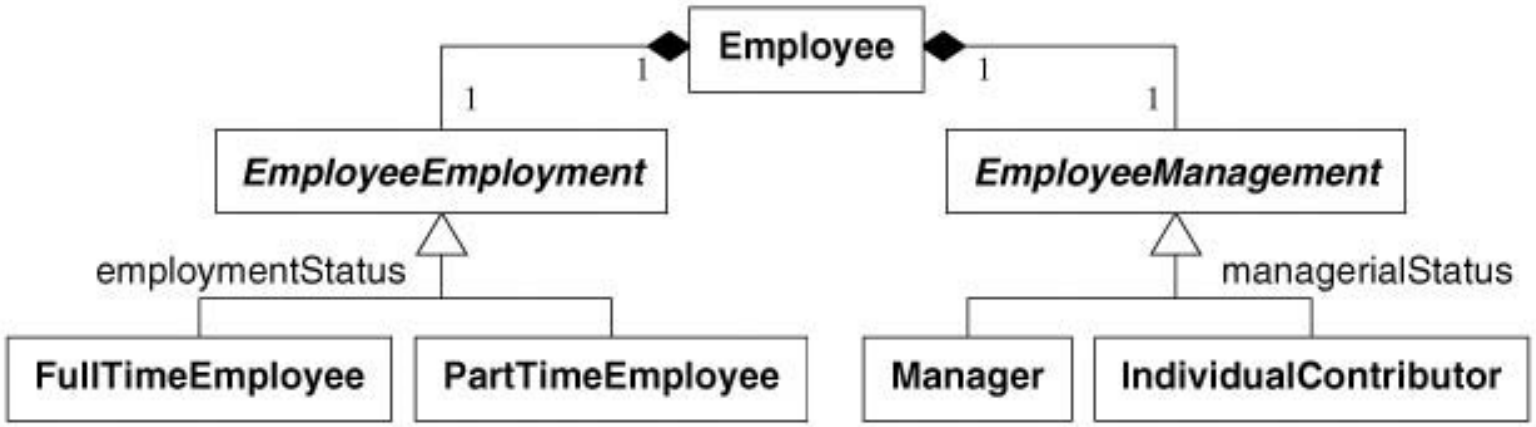
Multiple Inheritance

- Subclass has > one superclass (Inherits features from all parents
  - Advantage
    - Greater power in specifying classes
    - Increased opportunity for reuse
  - Disadvantage
    - Loss of conceptual and implementation simplicity



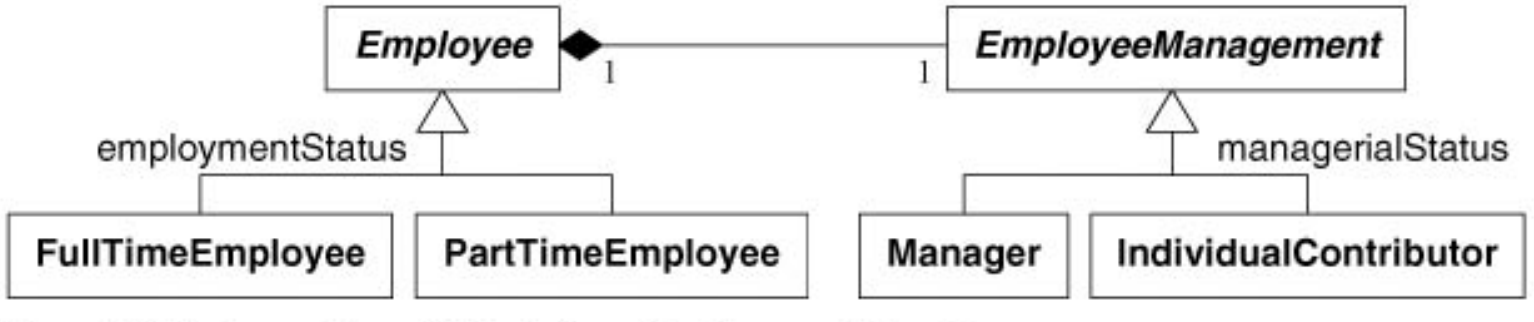
Multiple Inheritance: Workaround-1

- Delegation using composition of parts



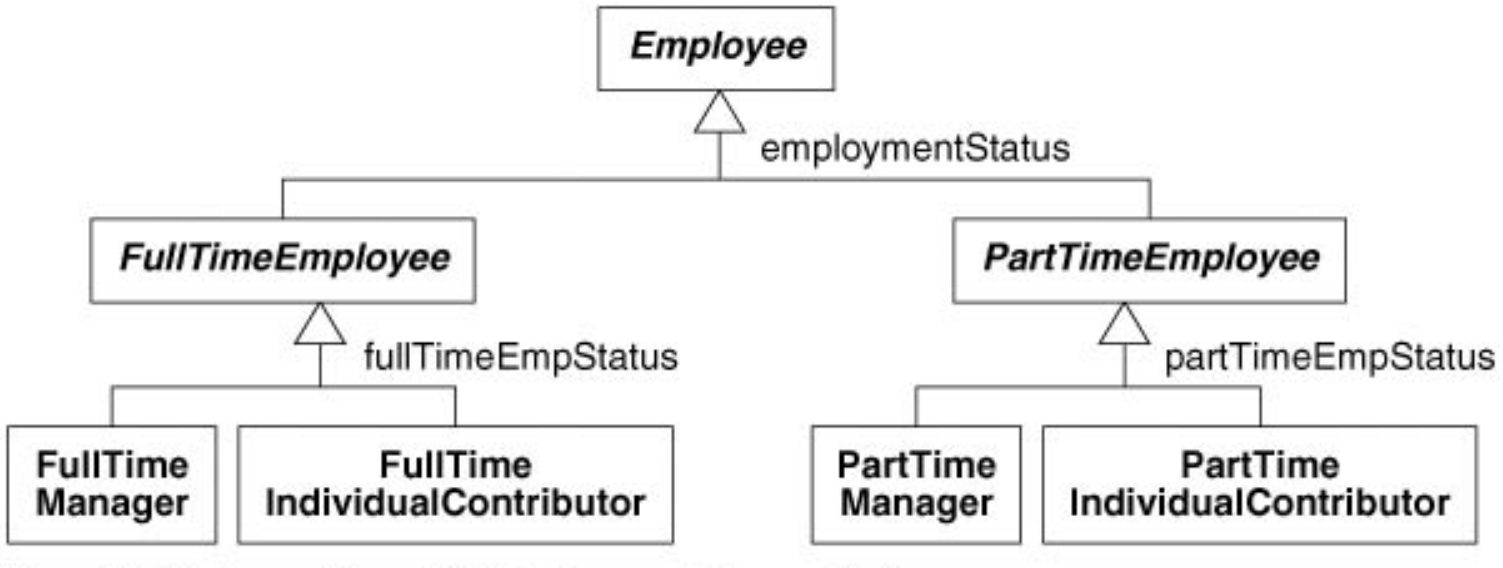
Multiple Inheritance: Workaround-2

- Inherit the most important class and delegate the rest



Multiple Inheritance: Workaround-3

- Nested generalization



Multiple Inheritance: Workarounds Summary

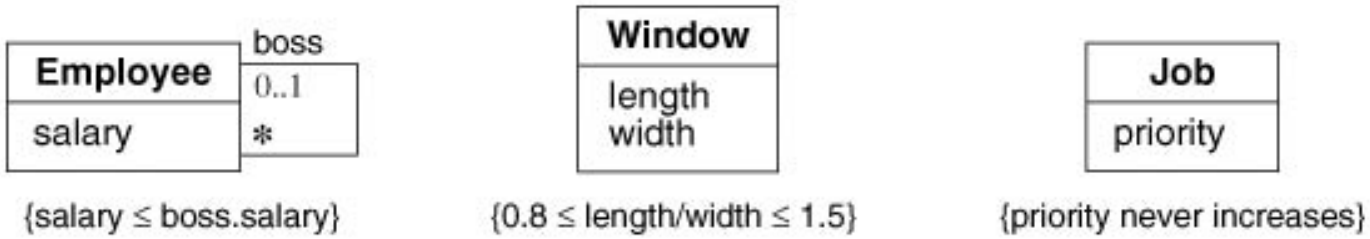
- All workarounds compromise logical structure and maintainability
- Super classes of equal importance
  - Use delegation
- Dominant superclass
  - Preserve inheritance through this path
- Few subclasses
  - If small number of combinations, use nested generalization
- Large quantities of code
  - Avoid nested generalization
- Maintaining identity
  - Use nested generalization
  - Factor on most important criteria first

Constraints

- Restricts values of model elements
  - Objects
  - Classes
  - Links
  - Attributes
  - Associations
  - Generalization sets

Constraints on Objects

- Between two objects at the same time
- Between attributes of a single object
- On same object over time

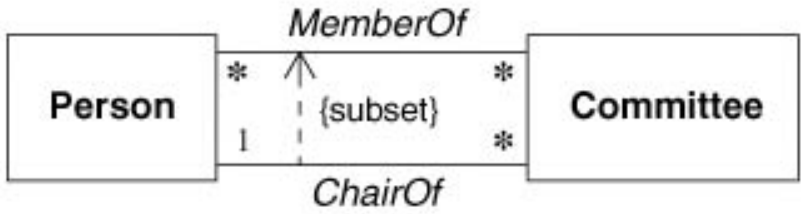


Constraints on Generalization Sets

- UML Keywords
  - Disjoint
  - Overlapping
  - Complete
  - Incomplete

Constraints on Links

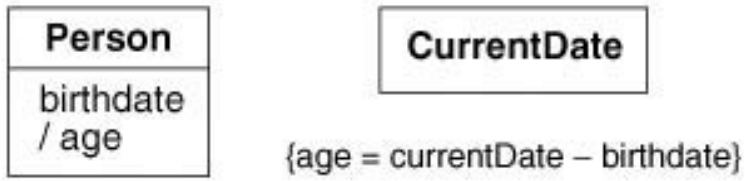
- Multiplicity restricts number of related objects
- Qualifier attribute
- Ordered constraint



Derived data

- Function of >=1 elements
  - Redundant information
  - Classes, attributes, associations

- UML Notation: /



Packages

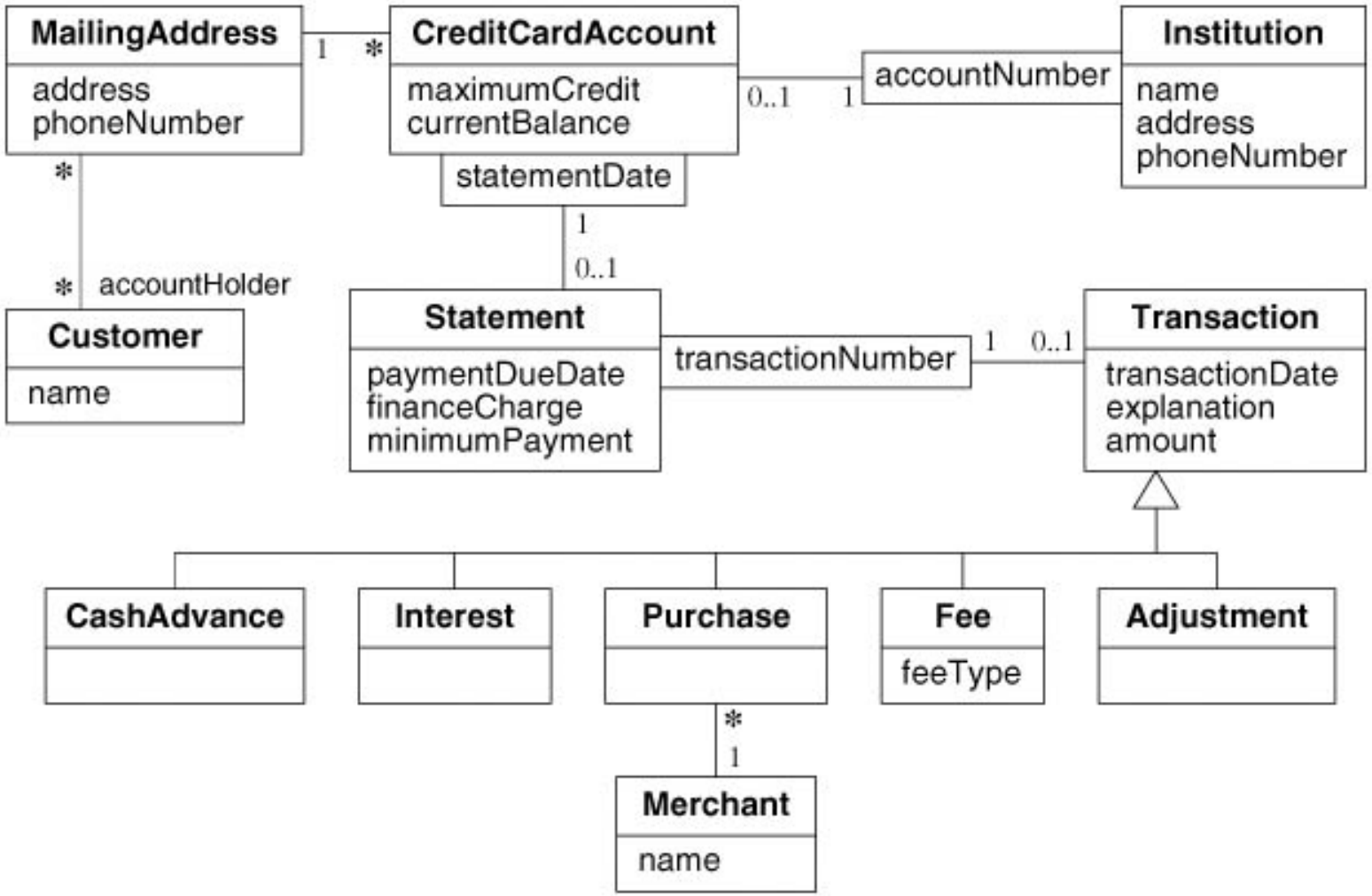
- Group of elements
  - Classes, associations, generalizations, and lesser packages
  - Multiple tiers
    - Top-level: Application

- UML Notation: Box with Tab

Navigation of class models

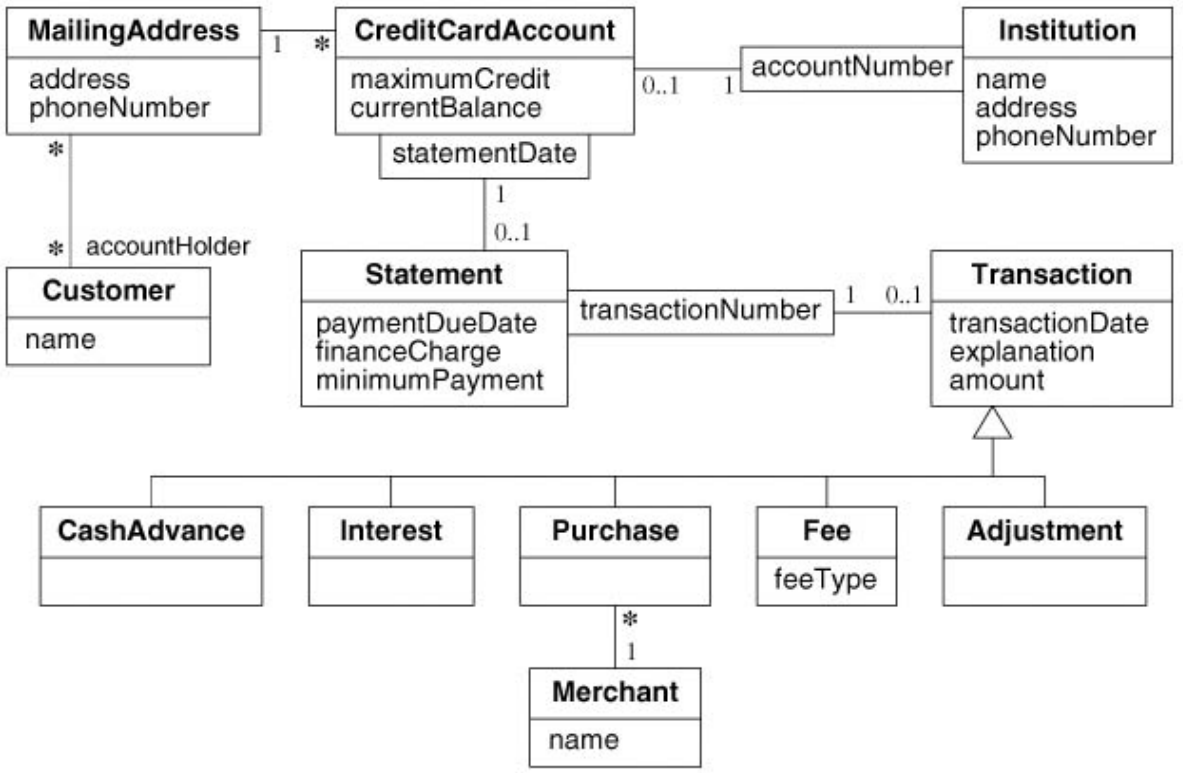
- Uncovers hidden flaws and omissions
- Navigation
  - Manually
  - Using navigation expressions
    - Object Constraint Language (OCL)

E.g. Class model for managing credit card accounts



OCL Constructs

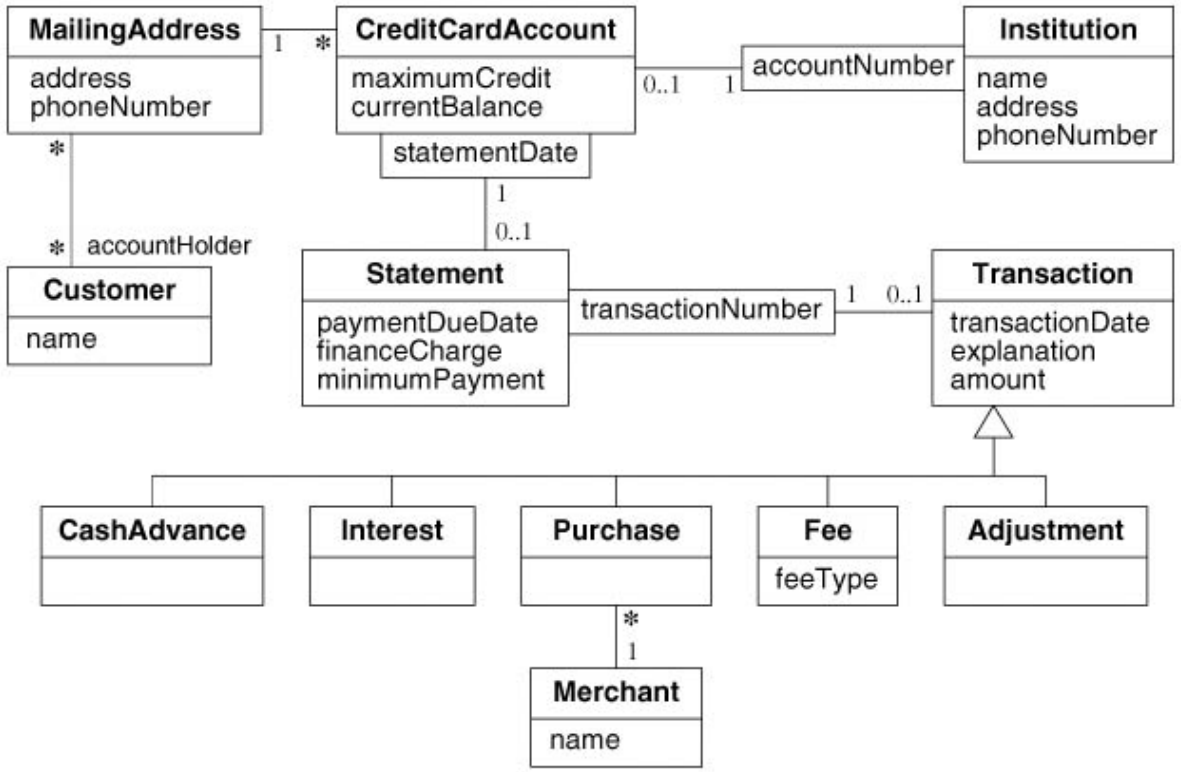
- Attribute
  - aCreditCardAccount.maximumCredit
- Operation
  - aCreditCardAccount.availableCredit()
  - Collection: use -> operator instead of dot
- Simple association
  - aCustomer.MailingAddress
- Qualified association
  - aCreditCardAccount.Statement[2020.09.08]
- Association class
- Generalization
  - Implicit traversal of hierarchy
- Filter
  - Filter objects in a set
  - aStatement. Transaction -> select(amount > $100)



OCL Constructs (Example: 1)

- What transactions occurred for a credit card account within a time interval?

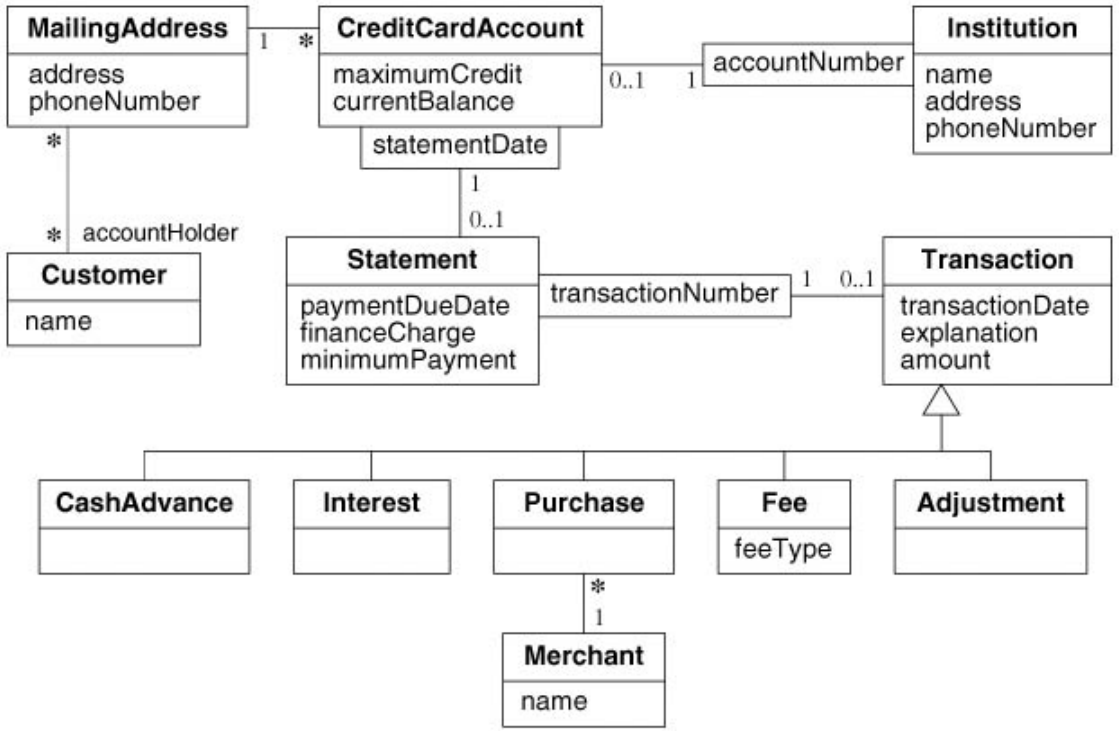
```
aCreditCardAccount.Statement.Transaction
        -> select (aStartDate <= transactionDate
                    and
                    transactionDate <= anEndDate)
```



OCL Constructs (Example: 2)

- What volume of transactions were handled by an institution in last year?

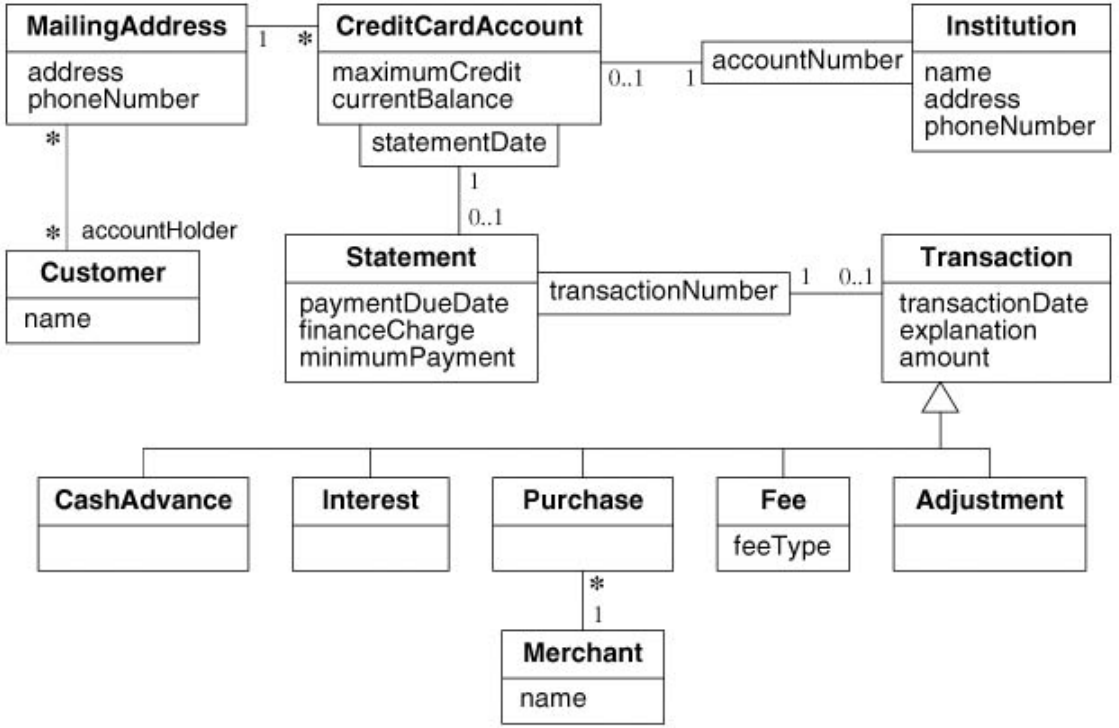
```
anInstitution.CreditCardAccount.Statement.Transaction
-> select (aStartDate <= transactionDate
                    and
                    transactionDate <= anEndDate
          ).amount -> sum()
```



OCL Constructs (Example: 3)

- What customers patronized a merchant in the last year by any kind of credit card?

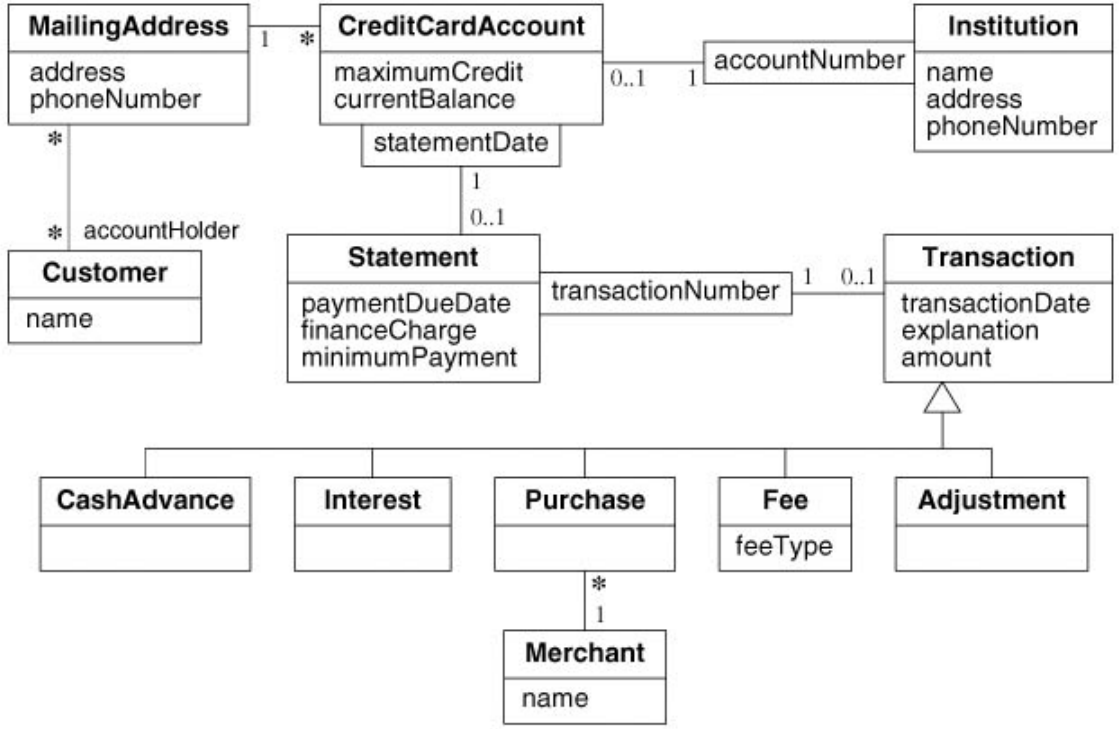
aMerchant. Purchase
-> select (aStartDate <= transactionDate
and
transactionDate <= anEndDate)
.Statement.CreditCardAccount
.MailingAddress.Customer
-> asSet()



OCL Constructs (Example: 4)

- How many credit card accounts does a customer currently have?

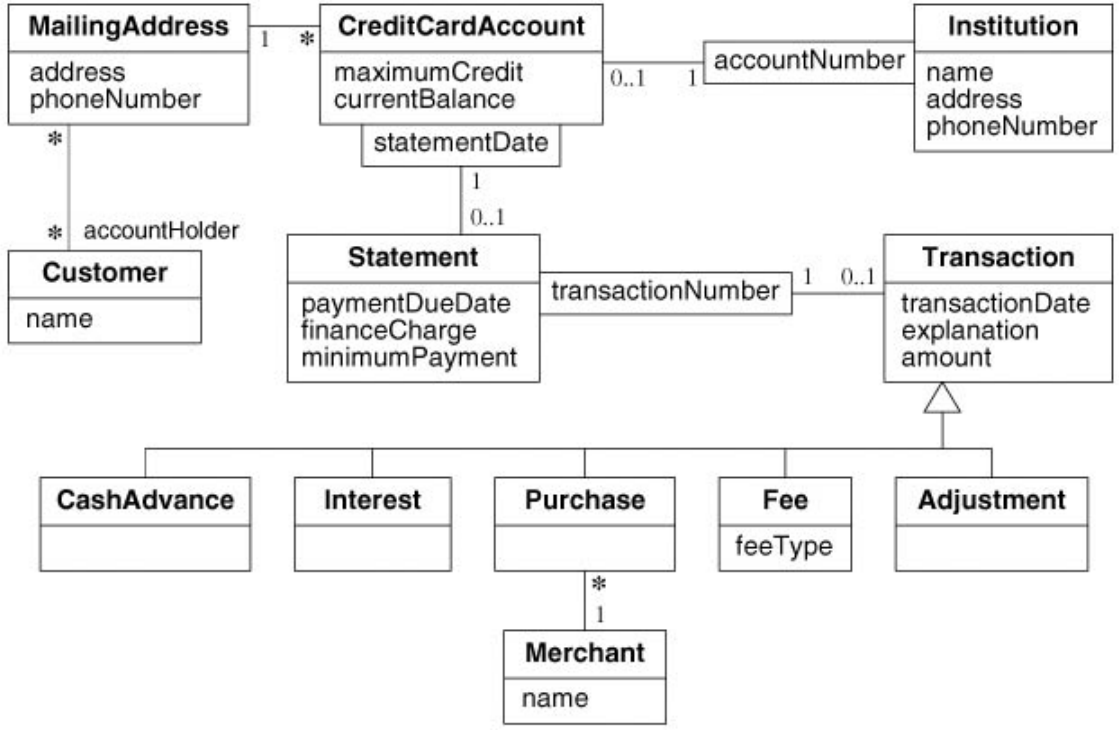
```
aCustomer
.MailingAddress
.CreditCardAccount
-> size()
```



OCL Constructs (Example: 5)

- What is the total maximum credit for a customer, for all accounts?

```
aCustomer
.MailingAddress
.CreditCardAccount
.maximumCredit
-> sum()
```



Next sessions…

- State Modeling
- Interaction Modeling