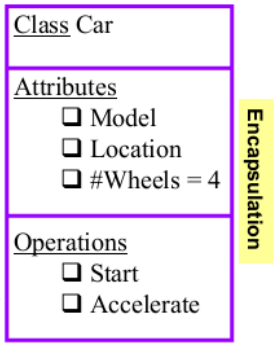
DA-IICT

IT 314: Software Engineering

Classes and their relationships

Saurabh Tiwari

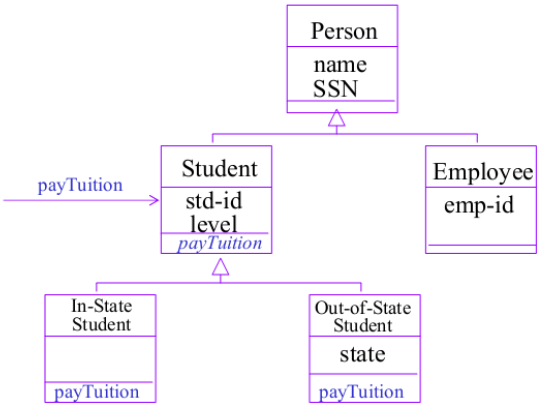
OO Concepts



- Objects
- Classes
- Interfaces
- Inheritance
- Attributes
- Methods
- Encapsulation
- Information Hiding, or Visibility
- Instances
- Delegation
- Polymorphism
- Dynamic Binding
- Aggregation
- Association
- Message Passing, or interactions
- Genericity

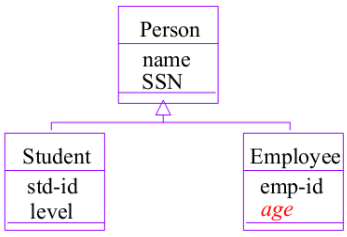
Polymorphism

Objects of different classes respond to the same message differently.



Super class vs. Sub class

What is Generalization??



Specialization: The act of defining one class as a refinement of another.

Subclass: A class defined in terms of a specialization of a superclass using inheritance.

Superclass: A class serving as a base for inheritance in a class hierarchy

Inheritance: Automatic duplication of superclass attribute and behavior definitions in subclass.

Class Modeling

5

Objects

The most fundamental entity in an OO Program

"An Object is a concept, abstraction or thing with identity that has meaning for an application"

- Defined by class data type

Consists of

- Identifier
- Values
- Behavior

JoeSmith: Person
name = "Joe Smith" birthdate = 21 October 1988

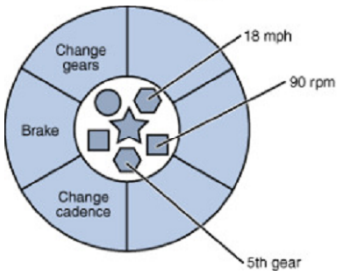
MarySharp: Person
name = "Mary Sharp" birthdate = 16 March 1990

Object with values

6

Software Object: Cycle

```
1.  class Bicycle {
2.      int cadence = 0;
3.      int speed = 0;
4.      int gear = 1;
5.      void changeCadence(int newValue) {
6.          cadence = newValue;
7.      }
8.      void changeGear(int newValue) {
9.          gear = newValue;
10.     }
11.     void speedUp(int increment) {
12.         speed = speed + increment;
13.     }
14.     void applyBrakes(int decrement) {
15.         speed = speed - decrement;
16.     }
17.     void printStates() {
18.         System.out.println("cadence:"+cadence+" speed:"+speed+" gear:"+gear);
19.     }
20. }
```



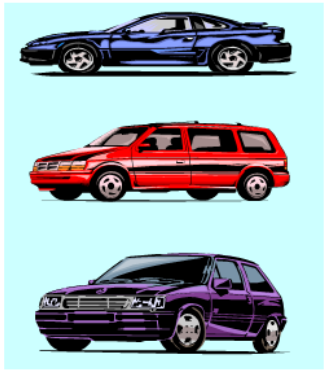
7

Classes

- A class is a pattern, a blue print, or a template for a category of structurally identical entities
- Created entities are called objects or instances
  - Class is like instance factory
  - Static entity
- ▫ A class has three components
  - Name
  - Attributes (also termed as variables, fields, or properties)
  - Methods (also termed as operations, features, behavior, functions)

8

Classes



Objects

<<instanceOf>>

<<instanceOf>>

<<instanceOf>>

Class Car

Attributes
- ❑ Model
- ❑ Location
- ❑ #Wheels = 4

Operations
- ❑ Start
- ❑ Accelerate

Encapsulation

- ***CLASS***
  - a collection of objects that share common properties, attributes, behavior and semantics, in general.
  - A collection of objects with the same data structure (attributes, state variables) and behavior (function/code/operations) in the solution space.
- **Classification**
  - Grouping of common objects into a class
- Instantiation.
  - The act of creating an instance.

9

Class

ClassName
Attributes
Operations

Customer
-name:String -userID:String -password:String
+getName():String +setName(newName:String) +getUserID():String +setUserID(newUserID:String) +getPassword():String +setPassword(newPassword:String)

Classes are represented by rectangles, with the name of the class, and can also show the attributes and operations of the class in two other “compartments” inside the rectangle.

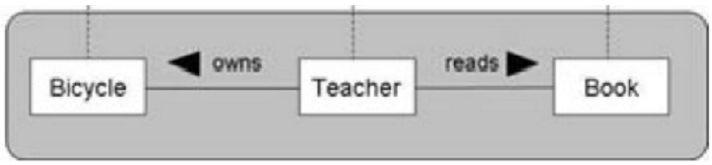
10

Class Diagrams

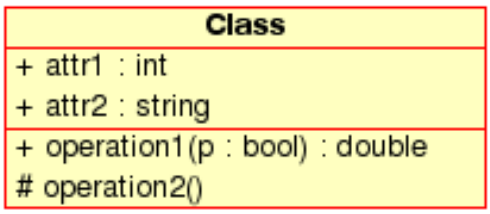
A Class defines the attributes and the methods of a set of objects.

All objects of this class (instances of this class) share the same behaviour, and have the same set of attributes (each object has its own set).

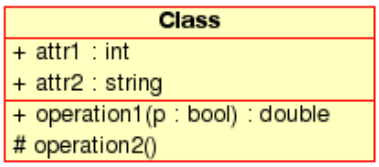
Class diagrams provide a graphic notation for modeling classes and their relationships thereby describing possible objects.



Class Diagrams



Class Diagrams

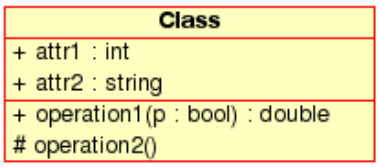


Attributes

In UML, Attributes are shown with at least their name, and can also show their type, initial value and other properties. Attributes can also be displayed with their visibility:

- \+ public attributes
- \# protected attributes
- \- private attributes

Class Diagrams

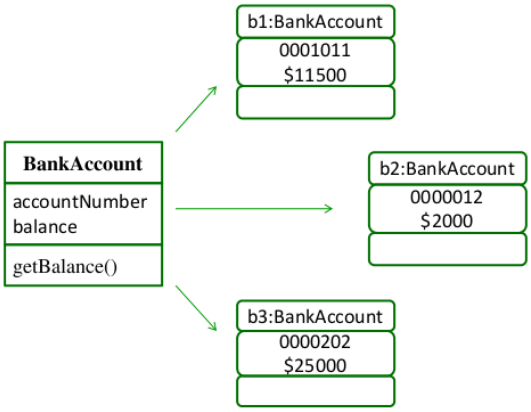


Operations

Operations (methods) are also displayed with at least their name, and can also show their parameters and return types. Operations can, just as Attributes, display their visibility:

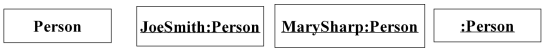
- \+ *public* operations
- \# *protected* operations
- \- *private* operations

Class and Objects



15

Class and Objects: Notations



Class Objects

Class Name
attribute:Type = initialValue
+ operation(arg list):return type

Person
name:string birthdate:date

Class with attributes

JoeSmith: Person
name = "Joe Smith" birthdate = 21 October 1988

MarySharp: Person
name = "Mary Sharp" birthdate = 16 March 1990

Object with values

Person
personID: ID name:string birthdate:date homeTelephoneNumber: string

Wrong

Person
name:string birthdate:date homeTelephoneNumber: string

Right

Do not list object identifier; they are implicit in models

16

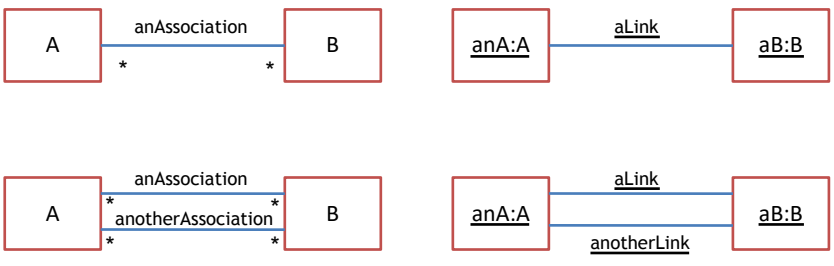
Object Relationships

- Association
  - Aggregation
  - Composition
- Inheritance
- Dependency

17

Link & Association

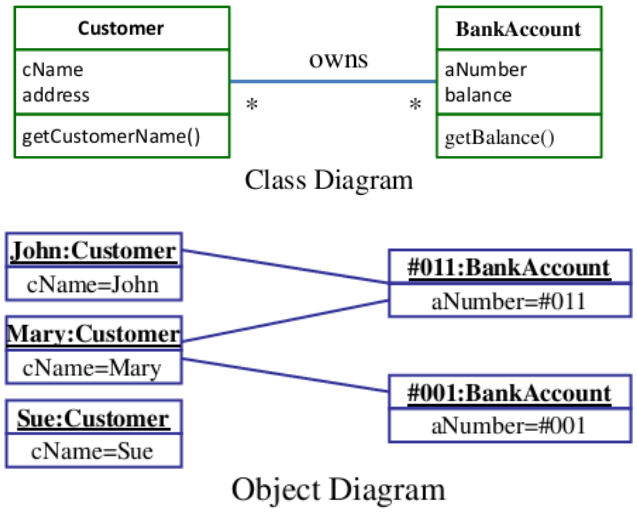
- Link is a physical or conceptual connection among objects
- Association is a description of a group of links with common structure and common semantics



Class Diagram *Object Diagram*

18

Association



19

Multiplicity

Multiplicity specifies "the number of instances of one class that may relate to a single instance of an associated class".

- *"One" or "Many"*
- *Infinite (subset of the nonnegative integers)*

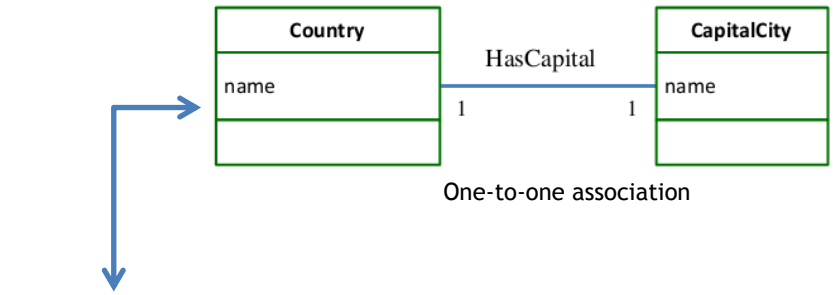
UML Specifies multiplicity with an interval

- *"1" (exactly one)*
- *"1…*" (one or more)*
- *"3..5" (three to five, inclusive)*
- *"*" (zero or more)*

20

Association

Visual representation



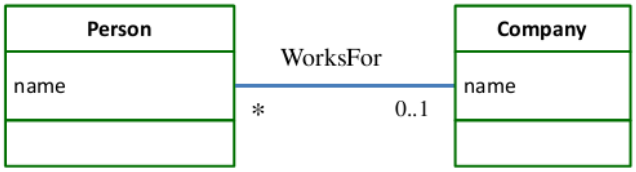
One-to-one association

Each country has one capital city

Textual representation

21

Association

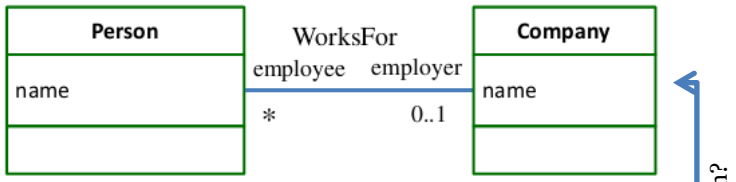


Many-to-one association

Many-to-many association

22

Association End Names



Person and Company participate in association "WorkFor"

Interpretation?

A Person is an employee with respect to a company
A Company is an employer with respect to a Person

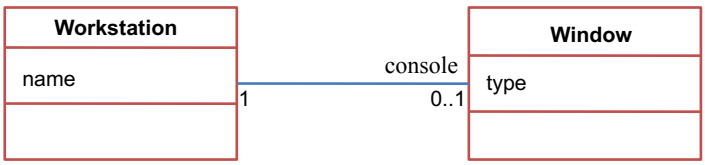
23

Association End Names

A Workstation may have one of its window designated as the console to receive general error messages.

Interpretation??

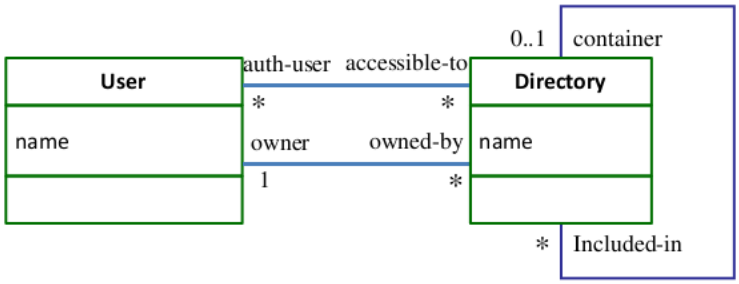
Is it possible that no console window exists?



24

Association End Names

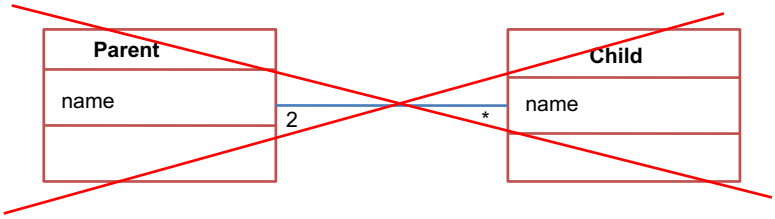
Association End Names are necessary for associations between two objects of a same class

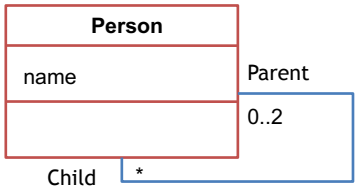


25

Question?

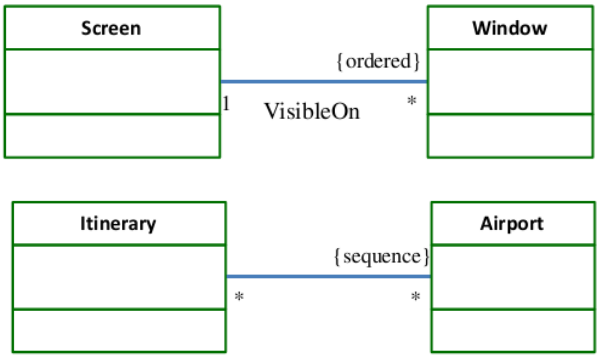
How to model parent-child relationship?





26

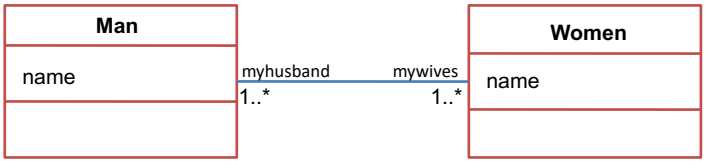
Association: Ordered, Sequence, Bag



- **Ordered** means that the objects are ordered in the collection of associated elements.
- **Sequence** means that the association implements an ordered bag, that is an ordered collection of elements where the same elements may occur several times

27

Question?



{……}

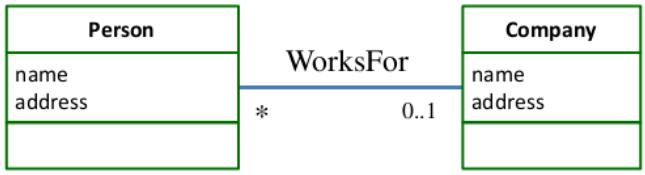
Ordered
Sequence

28

Association Class

UML offers the ability to describe links of association with attributes like any class.

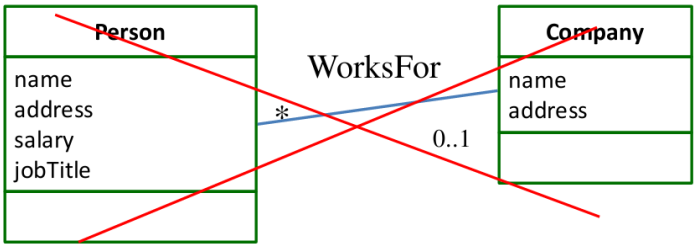
An association class is an association that is also a class.



Salary ?
Job Title ?

29

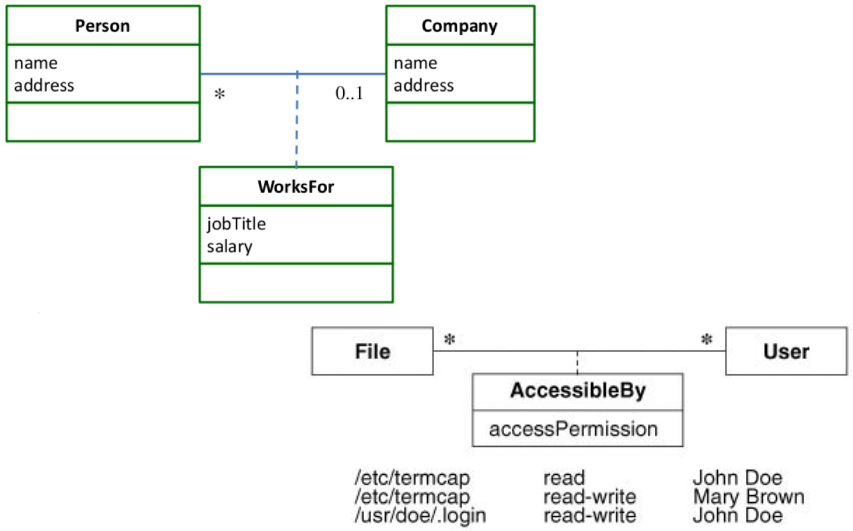
Association Class



Salary ?
Job Title ?

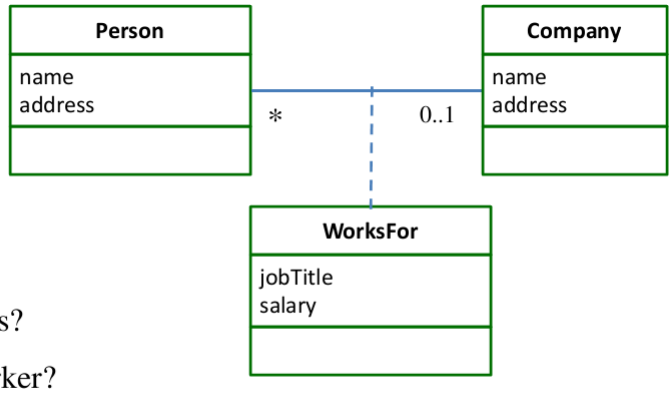
30

Association Class



31

Association Class



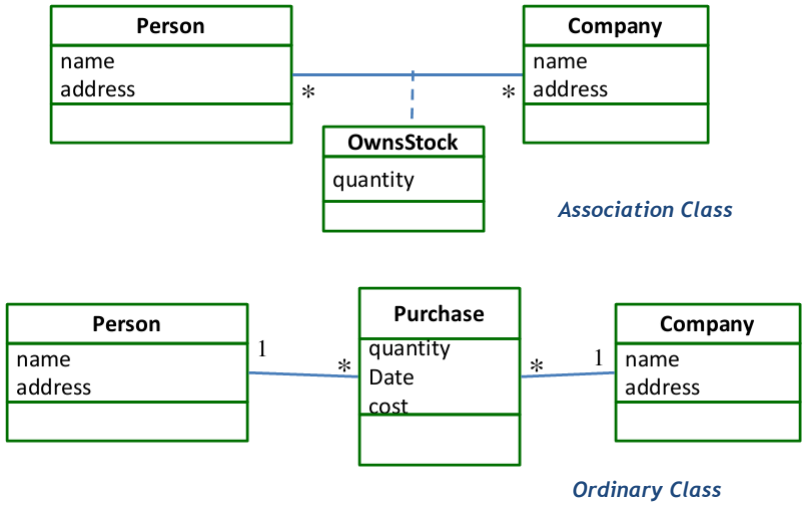
Boss?

Worker?

performanceRating?

32

Association Class vs Ordinary Class



33

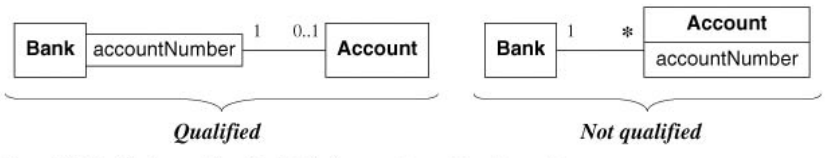
Question?

Users may be authorized on many workstations. Each authorization carries a *priority and access privileges*. A user has **a** *home directory* for each authorized workstation, but several workstations and users can share the same home directory.

34

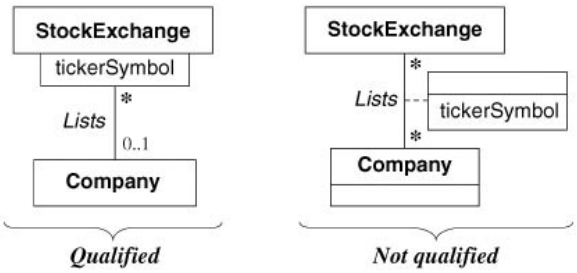
Qualified Association

- A qualified association is an association in which an attribute called Qualifier disambiguates the objects for a 'many' association end.
- A qualifier selects among the target objects, reducing the effective multiplicity for 'many' to 'one'.
- Both below models are acceptable but the qualified model adds information.



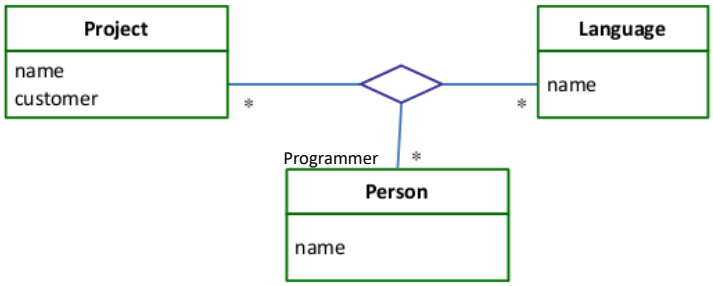
35

Qualified Association



36

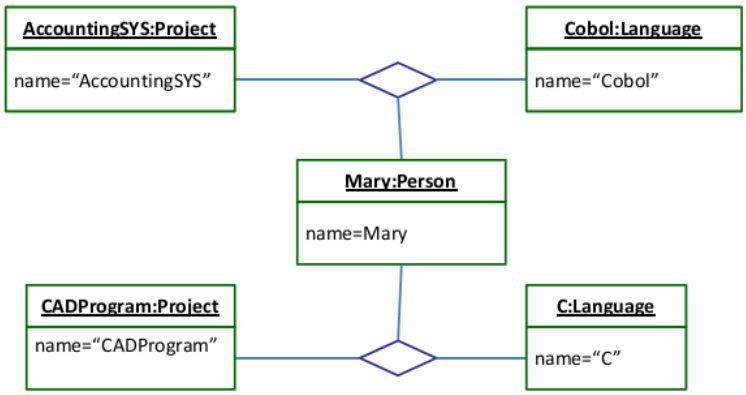
N-ary Association



Class Diagram

37

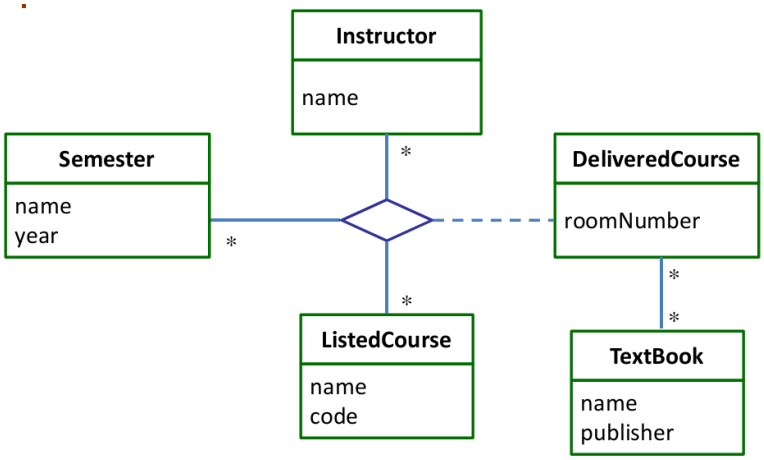
N-ary Association



Instance Diagram

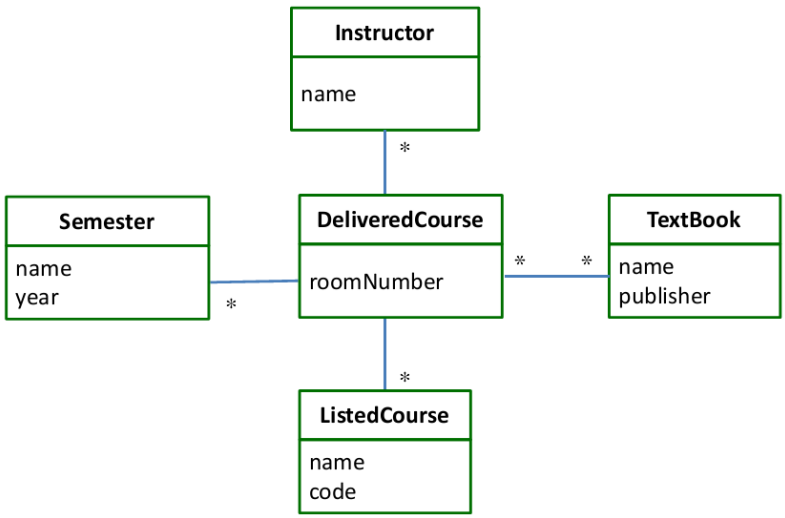
38

N-ary Association Class



39

N-ary Association Class



40

Summary

- Objects may have relationships (among the same type or different types)
- Loosely coupled (relatively independent) objects are related with Association
- Association is shown by (optional) association name and association end names
- Multiplicity represents the cardinality of the relationship stating one object of a class is associated with how many number of objects of the other class

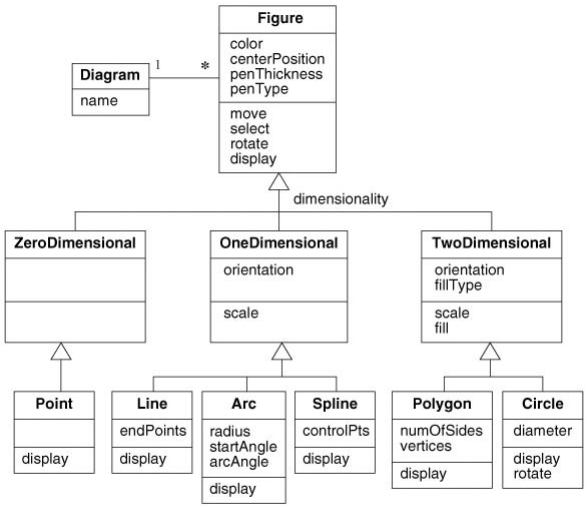
41

Next Lecture…

Object Relationships: Association (Aggregation, Composition) & Inheritance, Dependency

Generalization/Inheritance

- Generalization is the relationship between a class (**superclass**) and one or more **variations** of the class (**subclasses**).
- Generalization organizes classes by their **similarities** and their **differences, structuring** the descriptions of objects.
- A superclass holds **common** attributes, attributes and associations.
- The subclasses **adds specific** attributes, operations, and associations. They i**nherit** the features of their superclass.
- Often **Generalization** is called a **"IS A"** relationship
- **Simple generalization** organizes classes into a **hierarchy**.
- A subclass may **override** a superclass **feature (**attribute default values, operation) by **redefining a feature with the same name**.
- Never override the signature of methods.

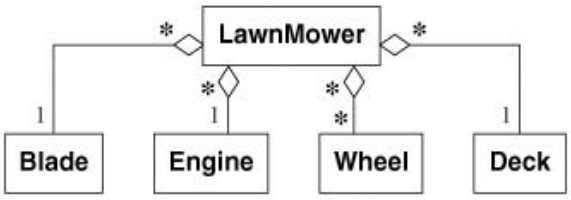


Use of generalization

- **Used for three purposes:**
  - Support of polymorphism:
    - polymorphism increases the flexibility of software.
    - Adding a new subclass and automatically inheriting superclass behavior.
  - Structuring the description of objects:
    - Forming a taxonomy (classification), organizing objects according to their similarities. It is much more profound than modeling each class individually and in isolation of other similar classes.
  - Enabling code reuse:
    - Reuse is more productive than repeatedly writing code from scratch.

Aggregation

- **Aggregation** is a **strong** form of **association** in which an **aggregate object** is made of **constituent parts.**
- The **aggregate** is semantically an extended object that is treated as a **UNIT in many operations, although physically it is made of several lesser objects.**
- **Aggregation is a transitive relation:**
  - if A is a part od B and B is a part of C then A is also a part of C
- **Aggregation is an antisymmetric relation:**
  - If A is a part of B then B is not a part of A.

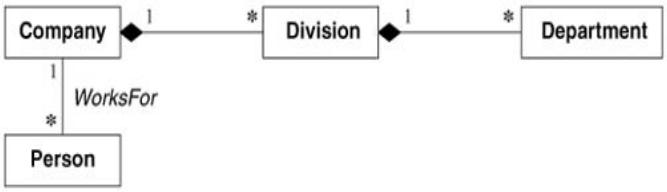


Aggregation versus Association

- Aggregation is a special form of association, not an independent concept.
- Aggregation adds semantic connotations:
  - If two objects are tightly bound by a part-whole relation it is an aggregation.
  - If the two objects are usually considered as independent, even though they may often be linked, it is an association.
- Discovering aggregation
  - Would you use the phrase **part of** ?
  - Do some operations on the whole automatically apply to its parts?
  - Do some attributes values propagates from the whole to all or some parts?
  - Is there an asymmetry to the association, where one class is subordinate to the other?

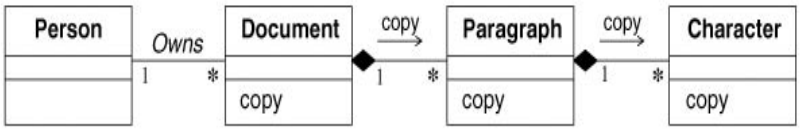
Aggregation versus Composition

- **Composition** is a form of aggregation with additional constraints:
  - A constituent part can belong to **at most one** assembly (whole).
    - it has a coincident lifetime with the assembly.
    - Deletion of an assembly object triggers automatically a deletion of all constituent objects via composition.
  - Composition implies ownership of the parts by the whole.
    - Parts cannot be shared by different wholes.



Propagation of operations

- Propagation is the automatic application of an operation to a network of objects when the operation is applied to some starting object.
- Propagation of operations to parts is often a good indicator of propagation.

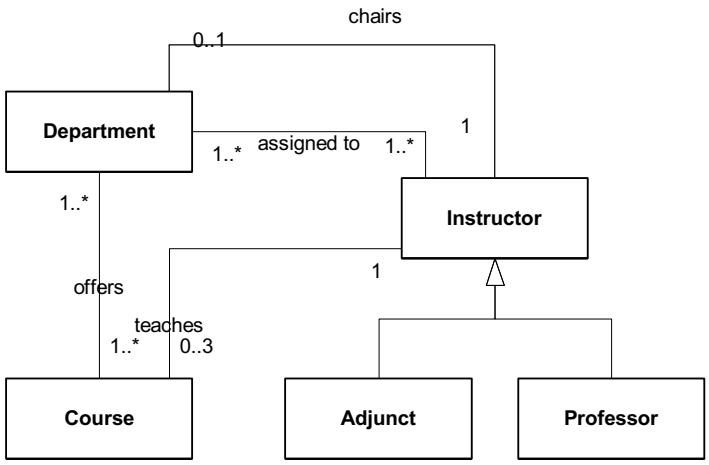


Exercises

Example 1: University Courses

- Some instructors are professors, while others have job title adjunct
- Departments offer many courses, but a course may be offered by >1 department
- Courses are taught by instructors, who may teach up to three courses
- Instructors are assigned to one (or more) departments
- One instructor also serves a department chair

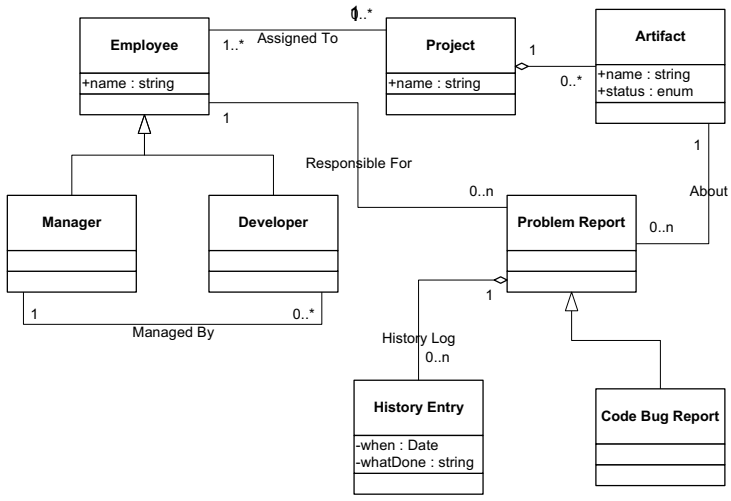
Class Diagram for Univ. Courses



Example 2: Problem Report Tool

- A CASE tool for storing and tracking problem reports
  - Each report contains a problem description and a status
  - Each problem can be assigned to someone
  - Problem reports are made on one of the "artifacts" of a project
  - Employees are assigned to one or more project
  - A manager may add new artifacts and assign problem reports to team members

Class Diagram for Prob. Rep. Tool



Questions??