

**Software Design**  
**CSE 335, Spring 2006**

**Static Data Modelling**

Stephen Wagner

Michigan State University

# Different Types of Models

- Sequence Diagrams model behavior
- Static Modeling is also important
  - How do different classes relate to each other
  - Extremely important in data base applications

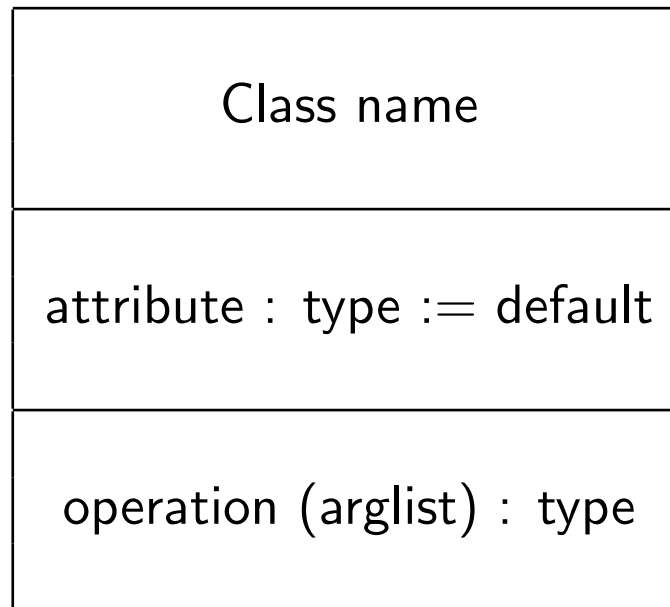
# Class Models

- In large systems it is important to understand the relationships among classes
  - OO libraries use lots of generalization
  - Other important relationships, e.g. associations and aggregations
- *Class model* is a graphical depiction of the classes and their relationships
- We specify class models in the *Unified Modeling Language (UML)*

# UML Terminology

- Objects
  - Meaningful concept or abstraction in an application
  - Objects have *identity*
- Class: group of objects with similar properties

## Class Notation (UML)



Attribute and operation boxes optional

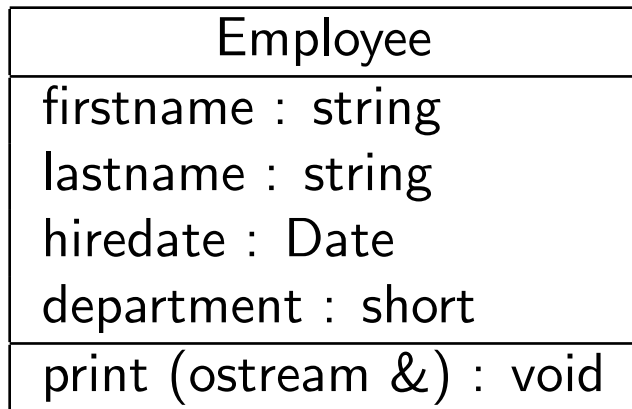
Class name is in *italics* if class is abstract

## Example

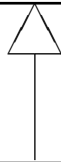
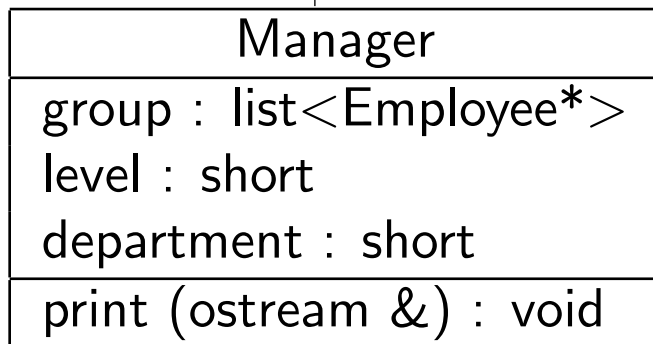
Employee
firstname : string lastname : string hiredate : Date department : short
print (ostream &) : void

Note: possible to depict almost everything about a C++ class. Usually do not list constructors/destructors

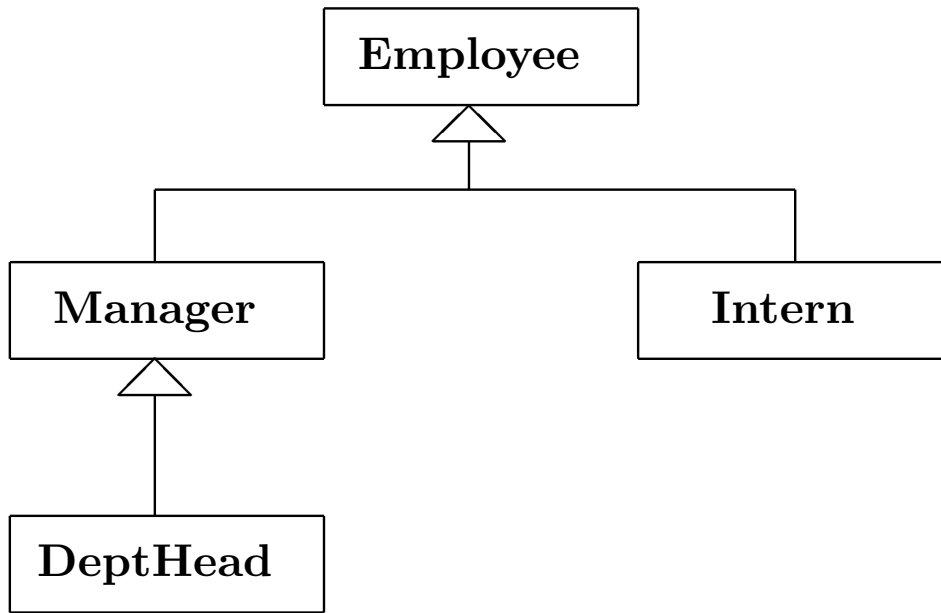
# Generalization in UML



Arrow points to base class



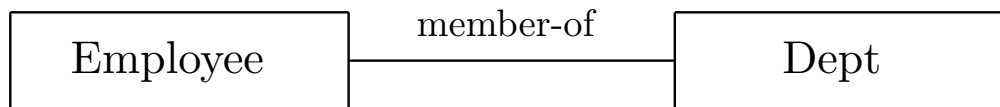
# Example



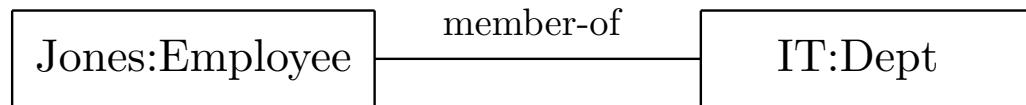
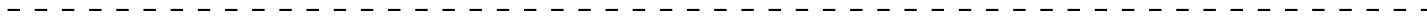
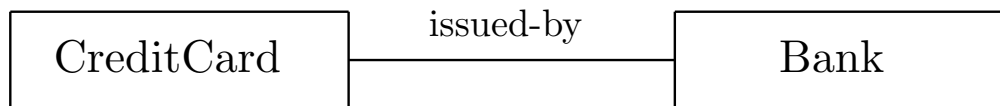
Generalization is transitive and anti-symmetric  
Leave out attributes and operations in deep hierarchies

# Relationships Between Classes

- *Association*: conceptual connection between classes
  - An employee is a member of a department
  - A credit card is issued by a bank



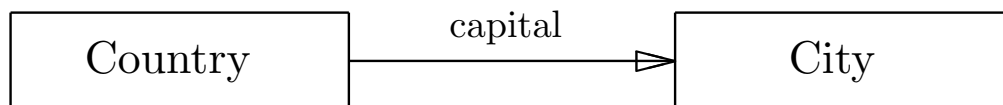
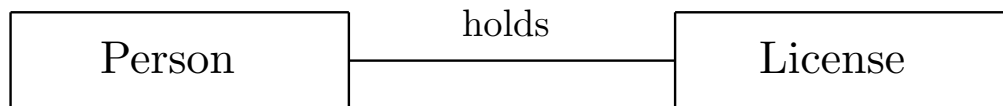
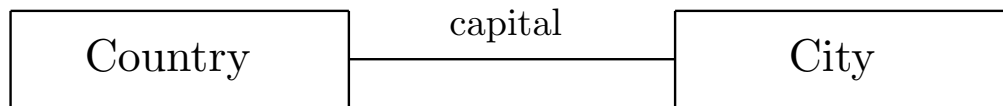
Class Diagrams



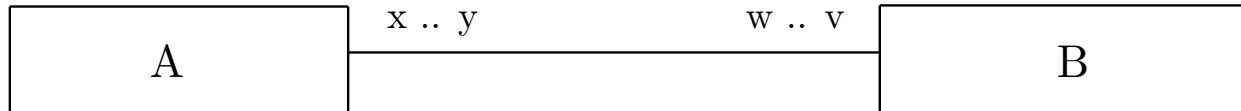
Instance Diagrams

# Associations are Bidirectional

- Associations goes in both directions
  - Not necessarily an implementation linkage
  - There are also *directed associations*



# Multiplicity Notation



There are  $x$  to  $y$  A's for each B  
There are  $w$  to  $v$  B's for each A

\* represents any number  
lists of numbers are also used

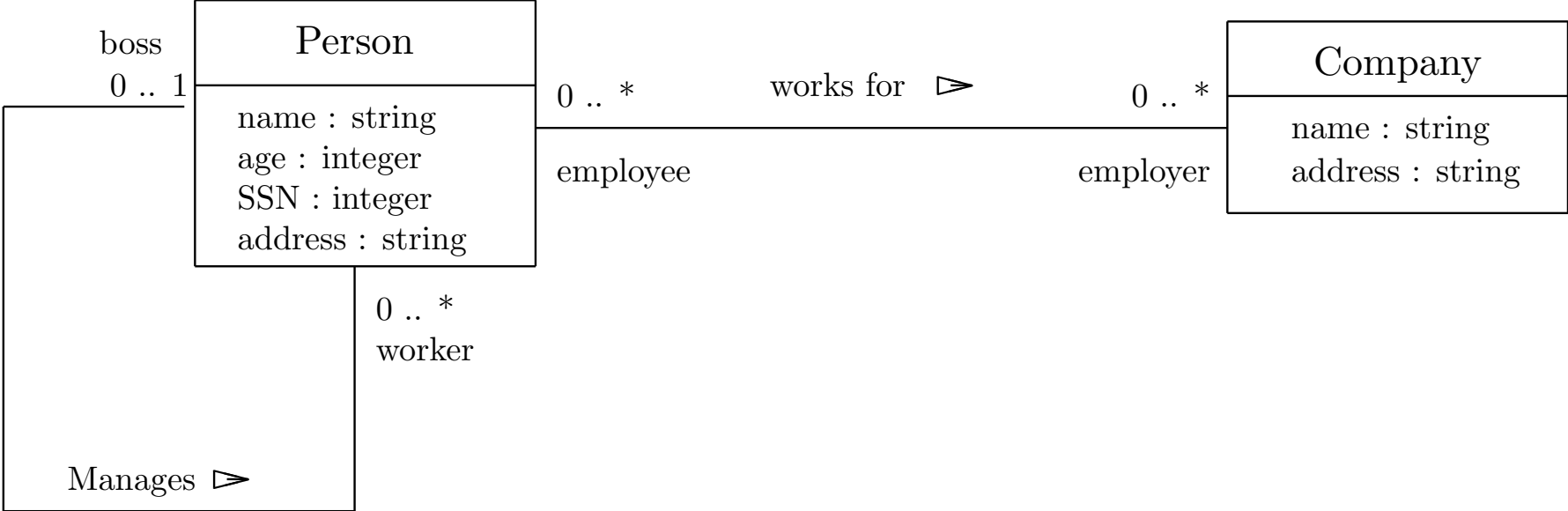
# Common Multiplicity Notations

0 .. 1 Optional

1 .. \* At least one

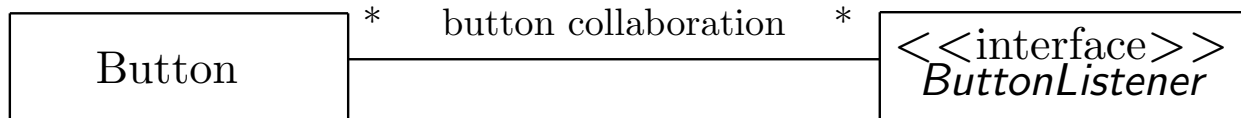
0 .. \* Any number

# Role Names



# Associations and Collaborations

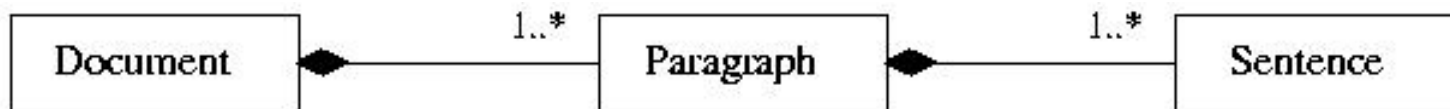
Often used to specify collaborations among objects of different classes



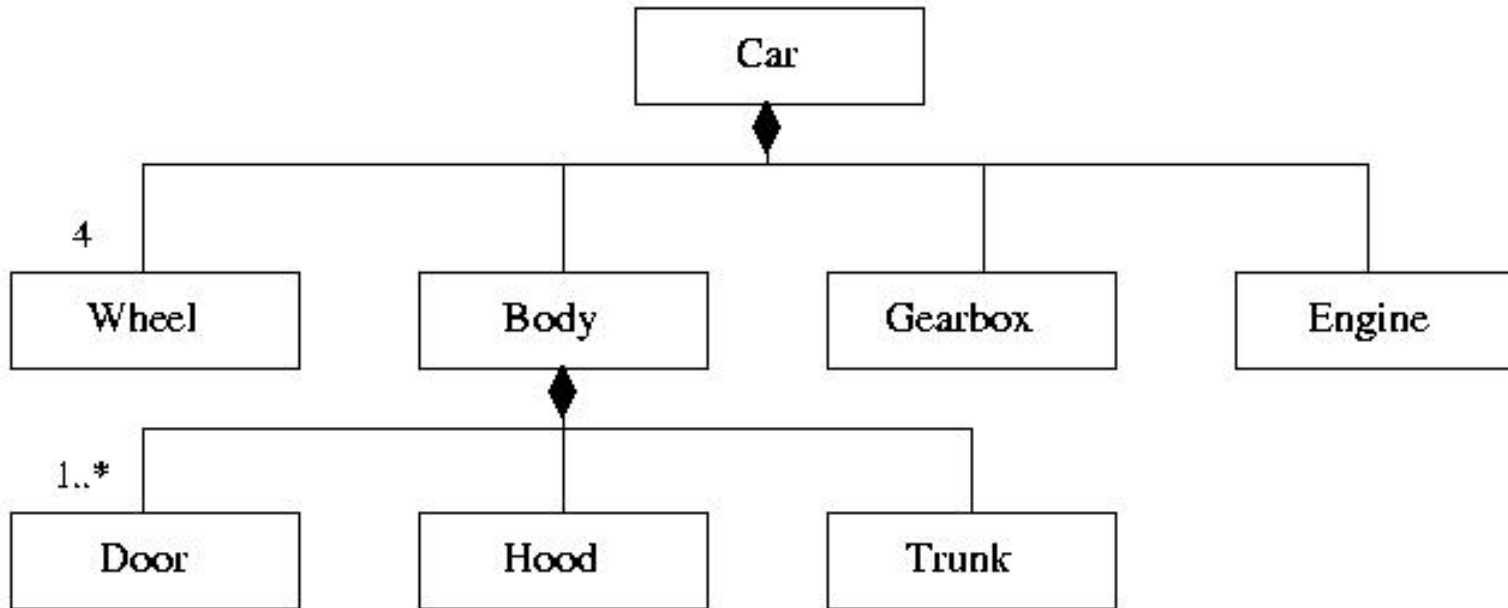
```
class Button
{
    protected:
    ...
    vector<ButtonListener *> listeners;
}
```

# Aggregation

- A special association, the *is-a-part-of* relationship
  - A sentence is part of a paragraph (a paragraph consists of sentences)
  - A paragraph is part of a document (a document consists of paragraphs)



# Parts explosion diagram



# Aggregation

- There are two types of symbols used for aggregation in UML
- Open diamond
  - The part may belong to more than one aggregation
  - Example: a song is part of a playlist, but it can be part of more than one playlist.
- Closed diamond
  - The part belongs to at most one aggregation
  - Example: a wheel is part of one and only one car
  - Multiplicity may be 0 or 1.
- There are some subtle semantics associated with this

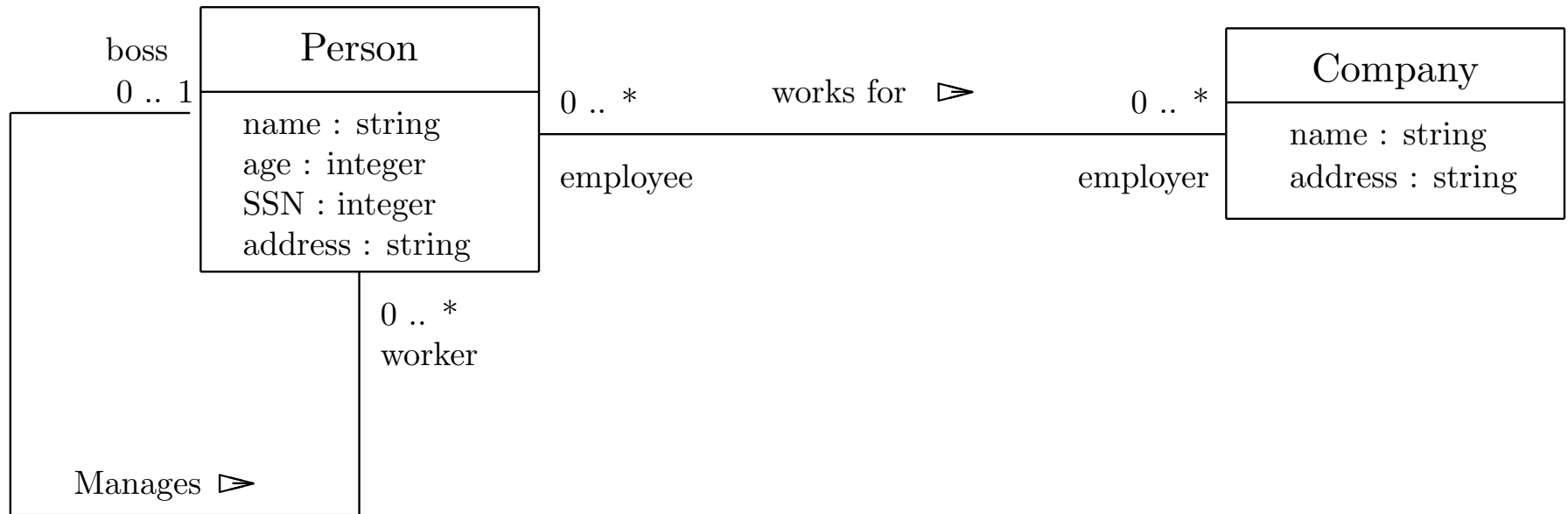
## Exercise

Draw a class diagram for the expression trees, including the appropriate associations

# Instance Diagrams

- An instance diagram portrays a set of objects
  - Each object will be the member of some class
  - Several objects may belong to the same class
- Associations will connect specific objects
- The following things do not appear in an instance diagram:
  - abstract classes
  - inheritance
  - multiplicities

# Instance Diagrams



Bob, Jim, Sue and Jill work for Ultracorp. Sue is Jim and Bob's boss. Bob is Jill's boss.

What does the instance diagram look like?

## Exercise

Draw an instance diagram for the following expression:

$$5 * (4 + 3) * (2 + 3 * 8)$$