

# Conceptual Modeling

---

Version 6/10/2016



**SoftEng**  
<http://softeng.polito.it>

© Maurizio Morisio, Marco Torchiano, 2016



# BP Aspects

---

- Information
  - ◆ Conceptual modeling
    - UML Class diagrams
    - (Entity-Relationships)
- Process flow
  - ◆ Process modeling
    - UML Activity Diagrams
    - BPMN
- Interaction
  - ◆ Interaction modeling
    - Use cases

# UML

---

- Unified Modeling Language
- Standardized by OMG
- Several diagrams
  - ◆ **Class diagrams**
  - ◆ **Activity diagrams**
  - ◆ **Use Case diagrams**
  - ◆ (Sequence diagrams)
  - ◆ (Statecharts)

Conceptual modeling

Process modeling

Functional modeling

---

Conceptual Modeling

# CLASS DIAGRAM

# Conceptual Modeling

---

- Construction of model
  - ◆ Providing an optimal description
  - ◆ From the stakeholders perspective
  
- Is the formalization phase after
  - ◆ Requirements elicitation and collection
  - ◆ Requirements analysis

# Goal

---

- Capture
  - ◆ Main (abstract) concepts
  - ◆ Characteristics of the concepts
    - Data associated to the concepts
  - ◆ Relationships between concepts

# Abstraction levels

---

**Abstract**

Concept  
Entity  
Class  
Category  
Type

**Concrete**

Instance  
Item  
Object  
Example  
Occurrence

---

# Class

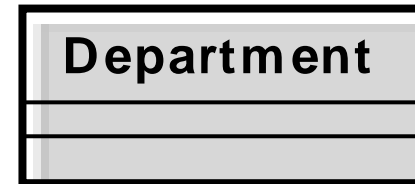
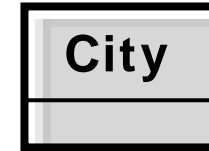
---

- A class represents a set of objects
  - ◆ Common properties
  - ◆ Autonomous existence
  - ◆ E.g. facts, things, people
- In an application for a commercial organization CITY, DEPARTMENT, EMPLOYEE, PURCHASE and SALE are typical classes.
  - ◆ Use a singular common noun



# Class - Examples

---



# Object

---

- Model of an item (physical or intangible within the software system)
  - ◆ ex.: a student, an exam, a window
- Characterized by
  - ◆ identity
  - ◆ attributes (or data or properties)
  - ◆ operations it can perform (behavior)
  - ◆ messages it can receive

# Object - Examples

---

john smith : Employee

turin : City

Computer and Control Engineering : Department

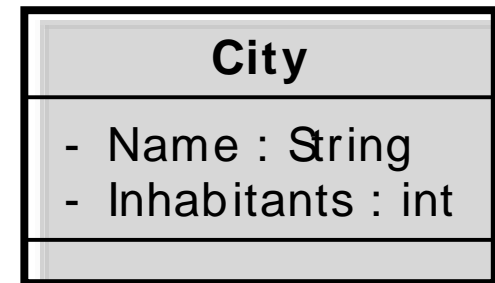
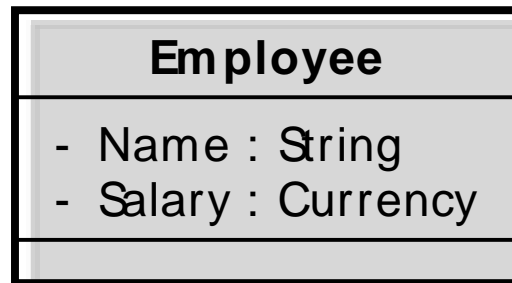
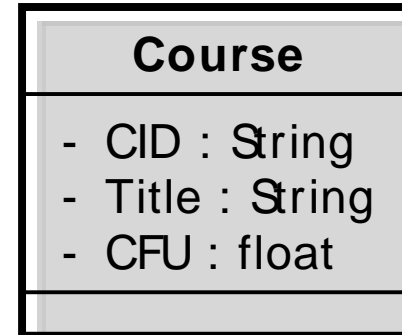
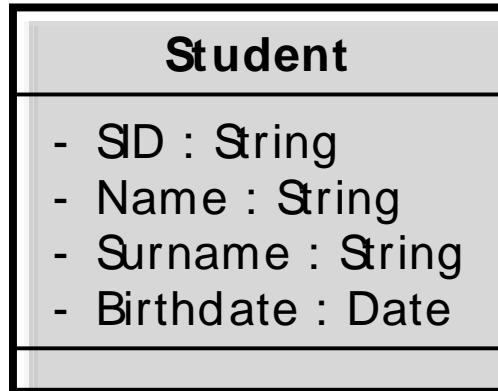
# Attribute

---

- Elementary property of a class
  - ◆ Name
  - ◆ Type
- An attribute associates to each object (occurrence of a class) a value of the corresponding type
  - ◆ Surname: String
  - ◆ ID: Numeric
  - ◆ Salary: Currency

# Attribute - Example

---



# Attribute Types

---

Tipo	Descrizione
<code>int</code>	Numero intero
<code>double</code>	Numero reale (singola prec.)
<code>float</code>	Numero reale (doppia prec.)
<code>boolean</code>	Valore logico (V/F, Si/No)
<code>String</code>	Stringa di caratteri / Testo
<code>Date</code>	Data (giorno-mese-anno)
<code>Time</code>	Ora (ore:minuti:secondi)

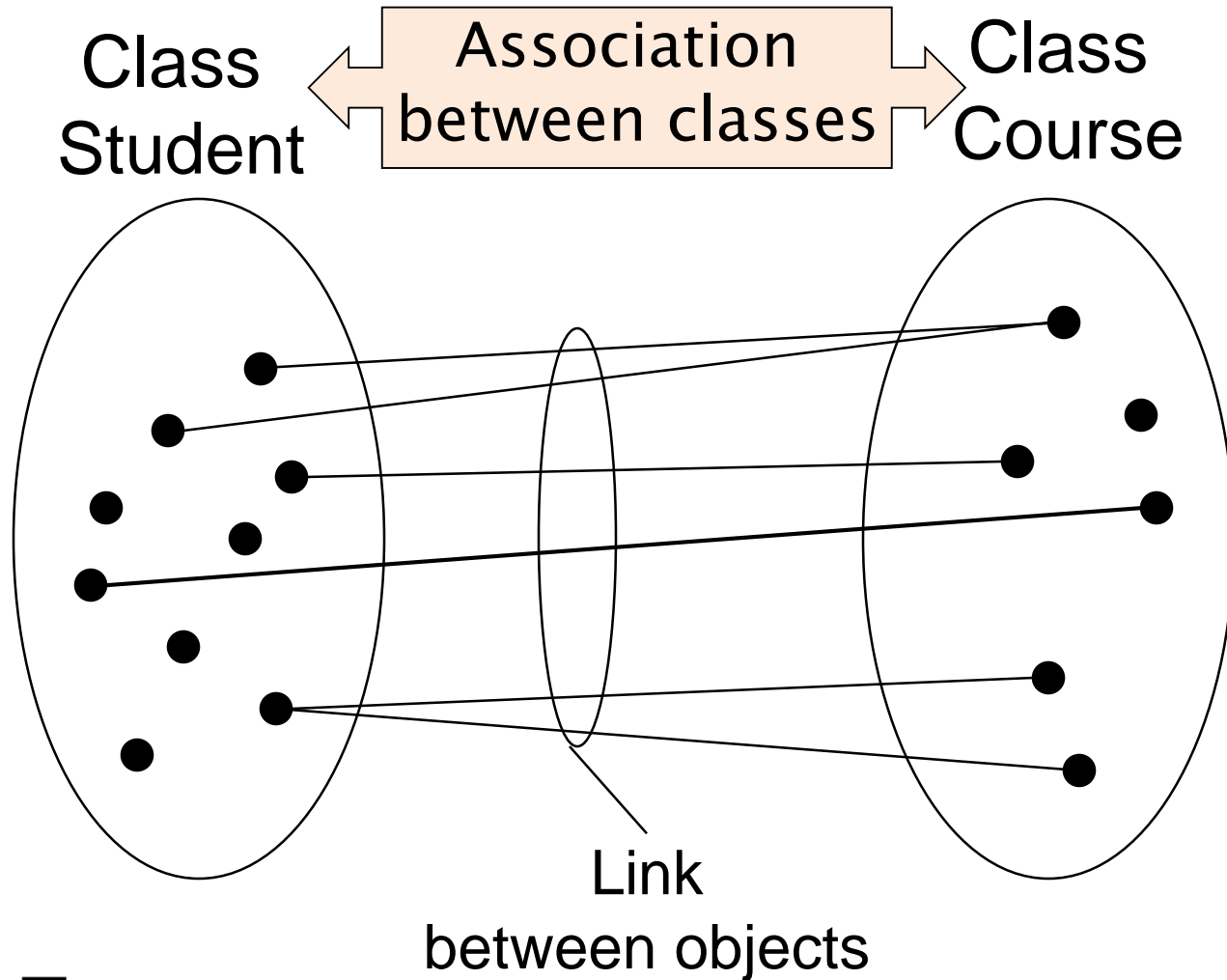
# Association

---

- Represent logical links between two classes.
- An occurrence of an association is an couple made up of occurrences of entities, one for each involved class
  - ◆ Residence can be an association between the classes City and Employee;
  - ◆ Exam can be an association between the classes Student and Course.

# Associations

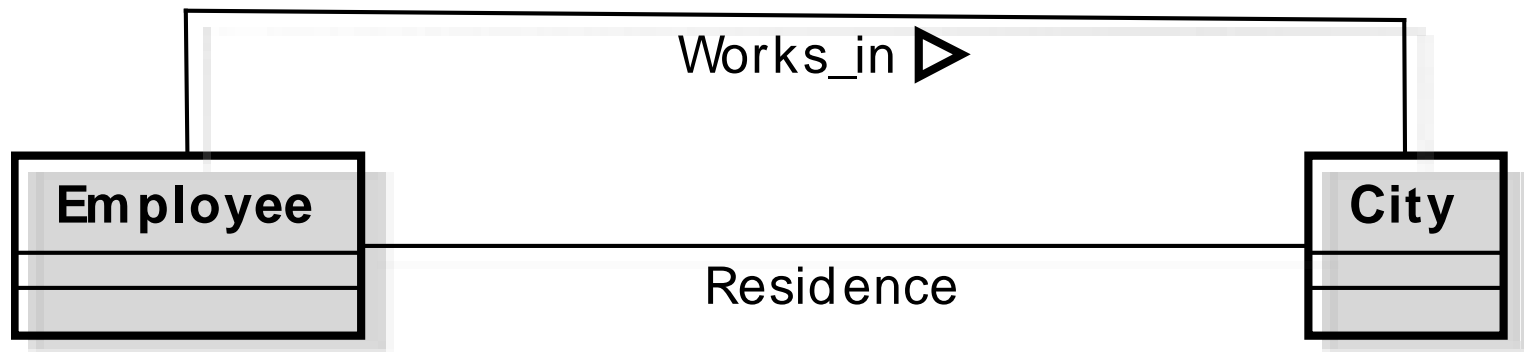
---





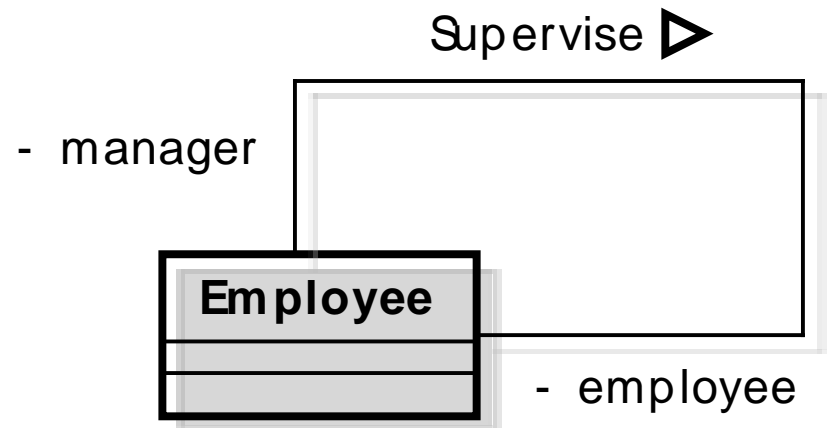
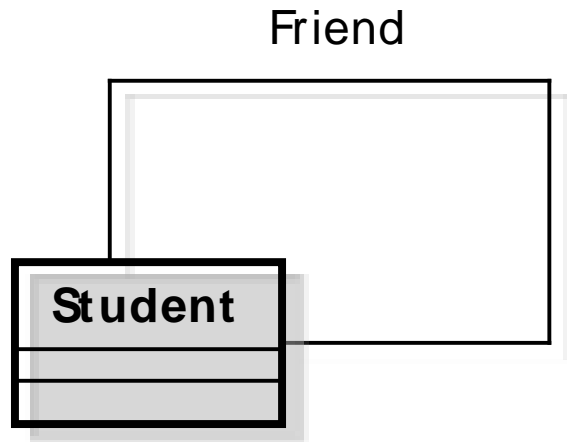
# Association – Examples

---



# Recursive association-Samples

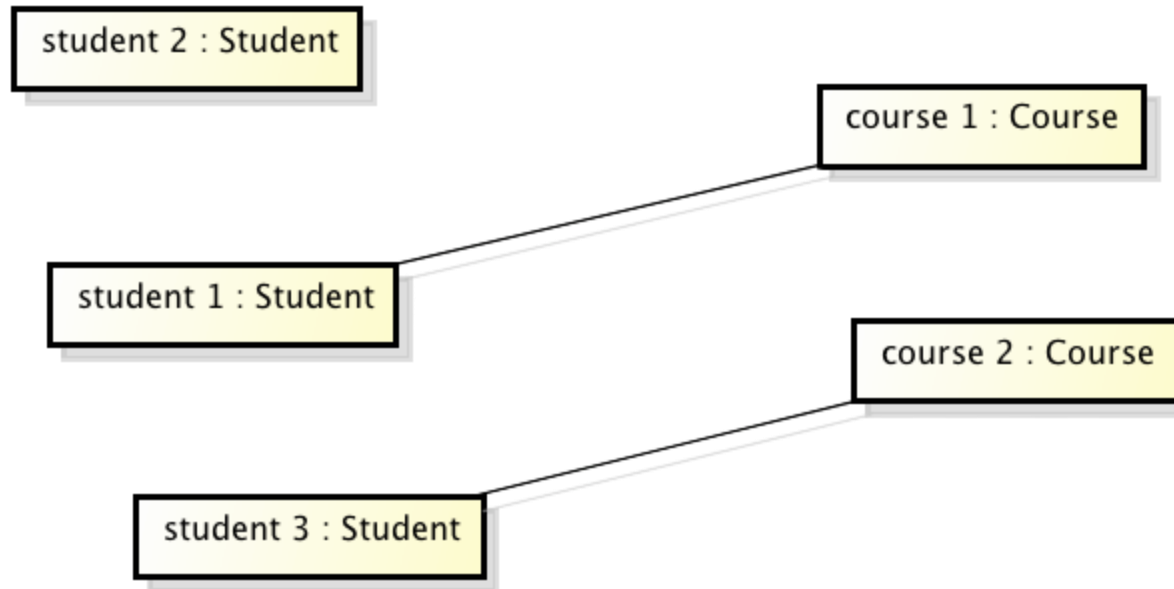
---



# Link

---

- Model of association between objects

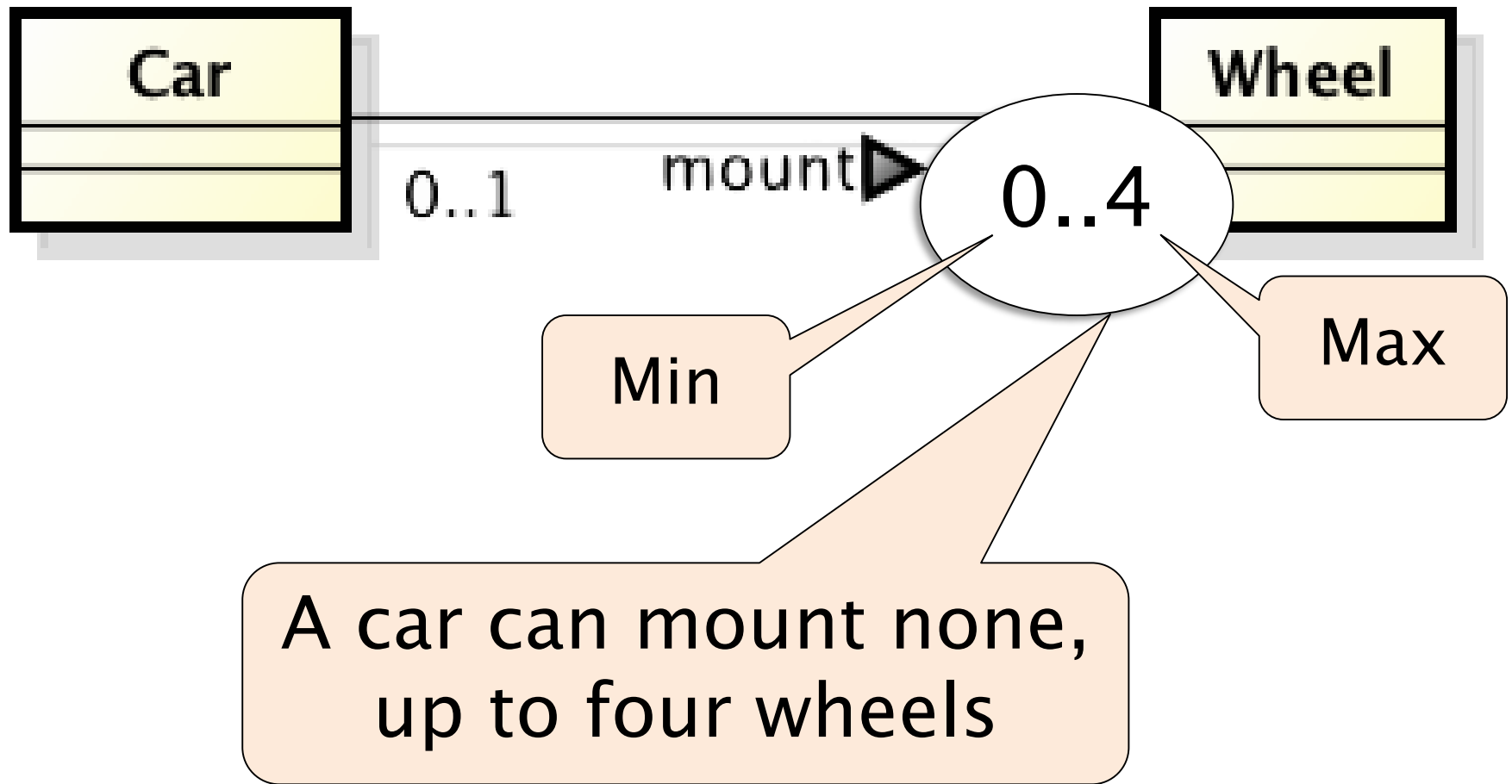


# Multiplicity

---

- Describe the maximum and minimum number of links in which a class occurrence can participate
  - ◆ Undefined maximum expressed as \*
- Should be specified for each class participating in an association

# Multiplicity – Example



# Multiplicity – Example

---



A wheel can be mounted on none or at most one car

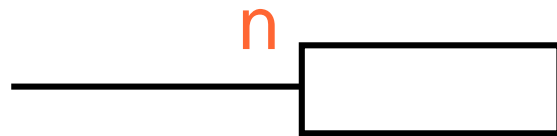
# Multiplicity

---

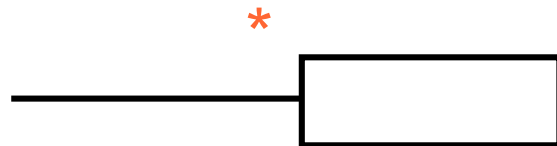
- Typically, only three values are used: **0**, **1** and the symbol **\*** (many)
- Minimum: 0 or 1
  - ◆ 0 means the participation is *optional*,
  - ◆ 1 means the participation is *mandatory*;
- Maximum: 1 or \*
  - ◆ 1: each object is involved in at most one link
  - ◆ \*: each object is involved in many links

# Multiplicity

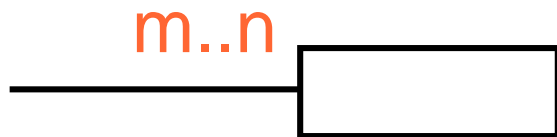
---



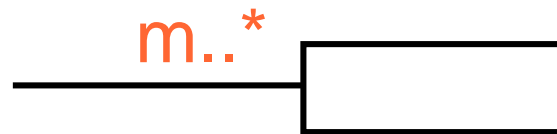
Exactly  $n$



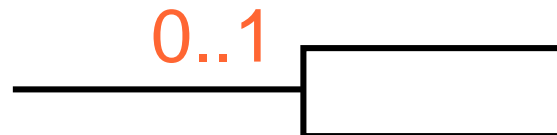
Zero or more



Between  $m$  and  $n$  ( $m, n$  included)



From  $m$  up

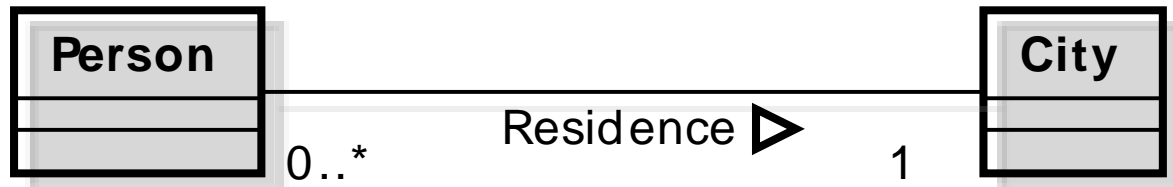
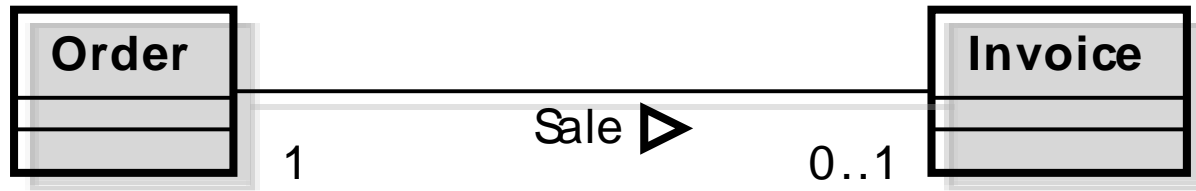


Zero or one (optional)

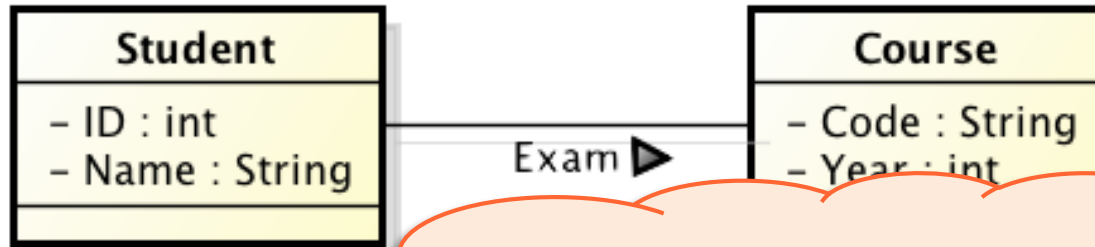


# Multiplicity

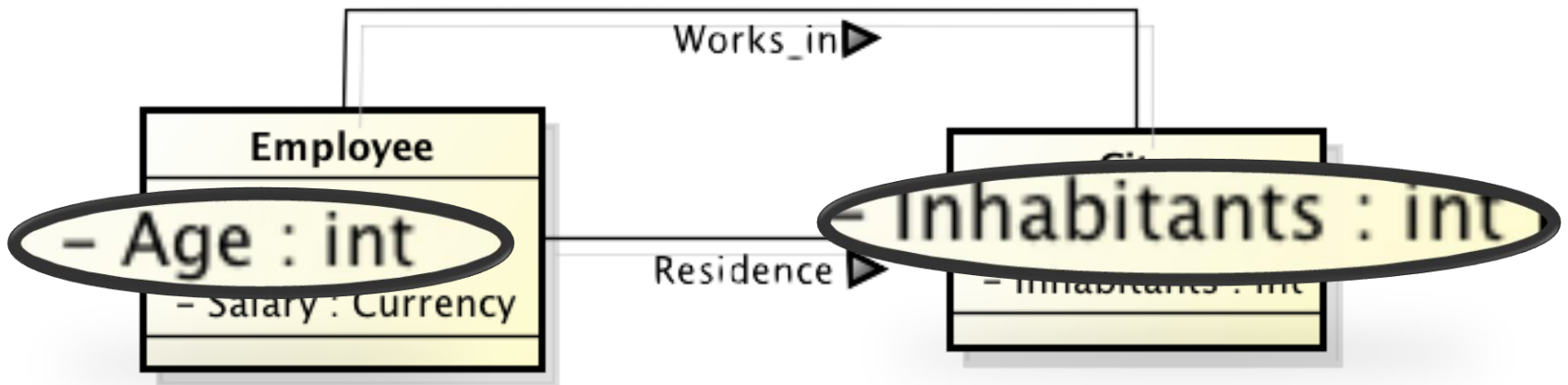
---



# Derived attributes



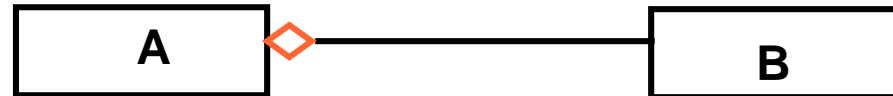
Is everything ok?



# Aggregation

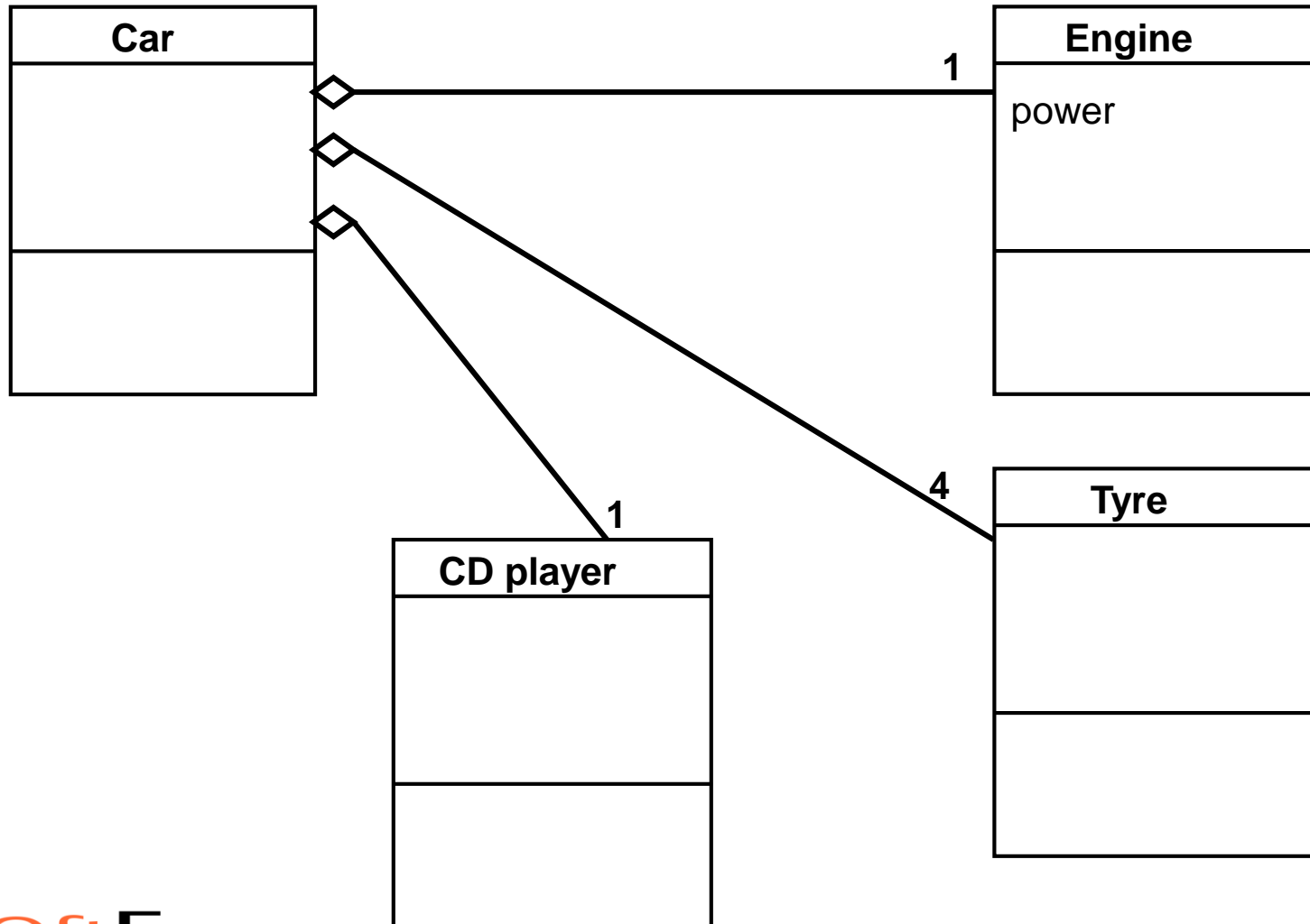
---

- *B is-part-of A* means that objects described by class B can be attributes of objects described by A



# Example

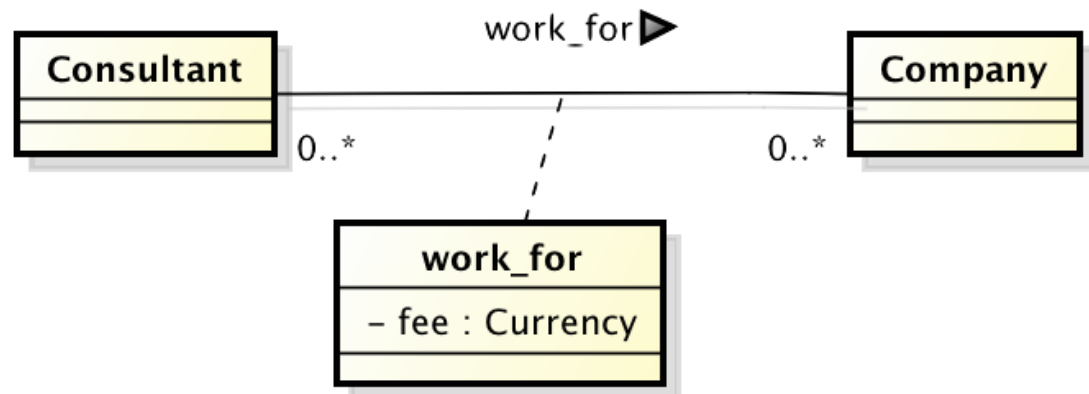
---



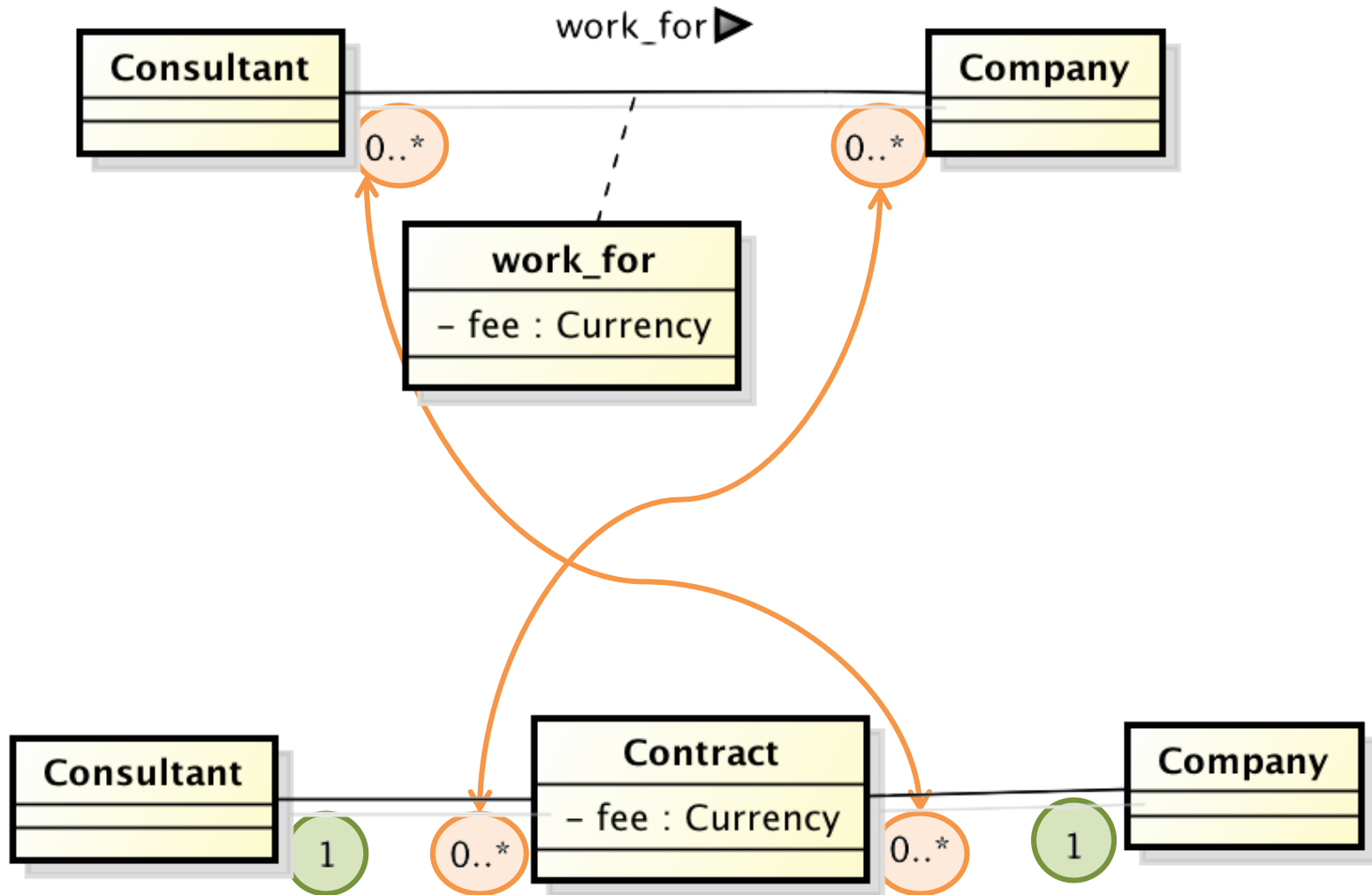
# Association Class

---

- The association class define the attributes related to the association
- A link between two object includes
  - ◆ The two linked objects
  - ◆ The attributes defined by the association class



# Association class – Equivalence



# Association Class Limitations

---

- Association class
  - ◆ Fee is a function of consultant and company
  - ◆  $fee ( Consultant , Company )$
  
- Intermediate class
  - ◆ Fee is a function of the contract
  - ◆  $fee ( Contract )$

# Association class limitation

---

- Case
  - ◆ Consultant working several times for the same Company
- Cannot be represented by association class
- Only representable through intermediate class



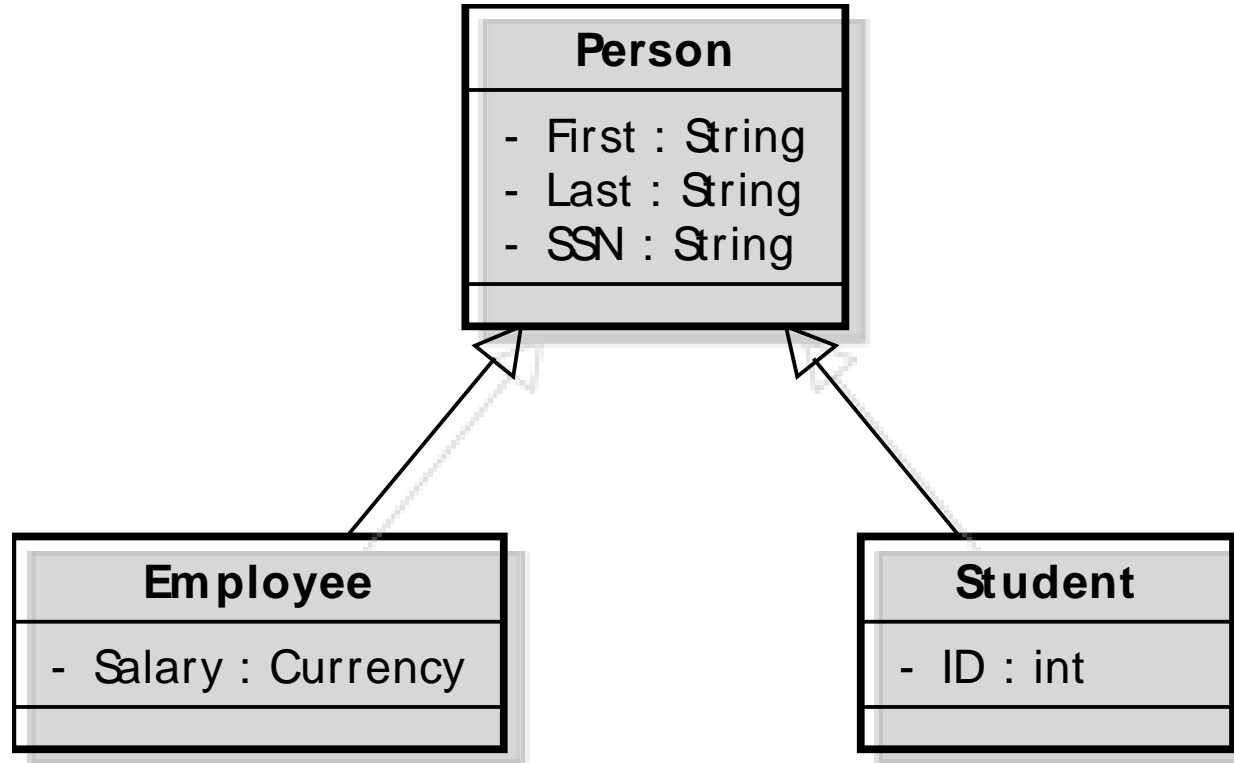
# Specialization / Generalization

---

- *B specializes A* means that objects described by B have the same properties of objects described by A
- Objects described by A may have additional properties
- B is a special case of A
- A is a generalization of B (and possible other classes)

# Generalization

---

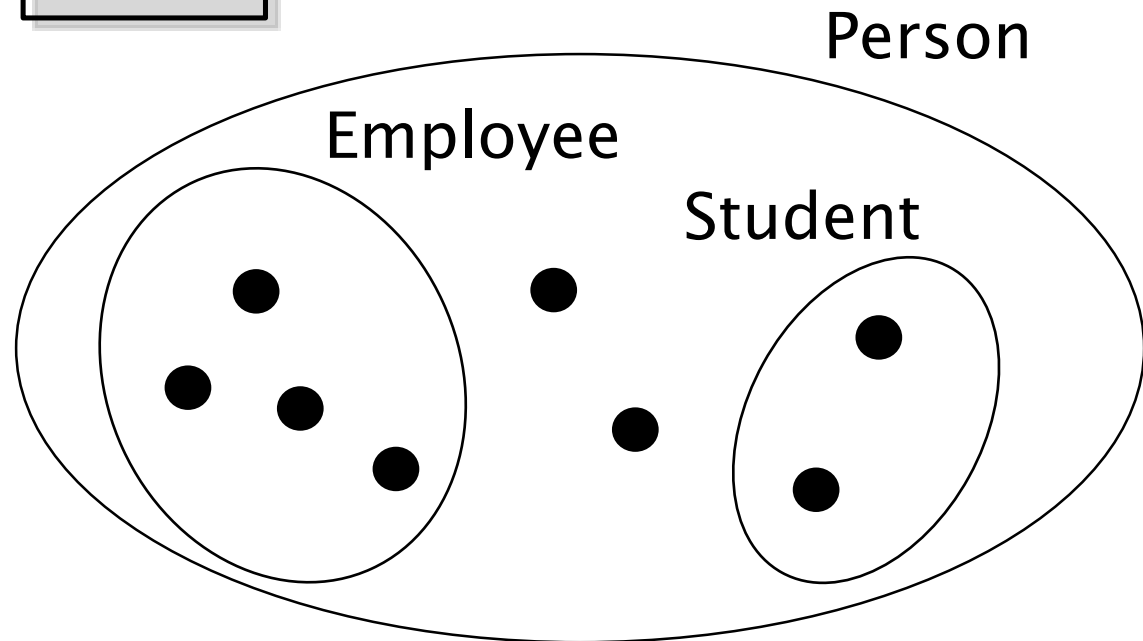
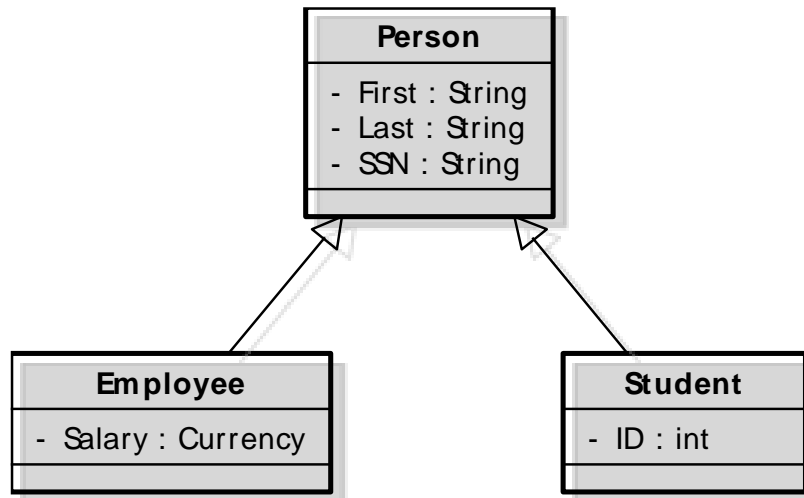


# Inheritance terminology

---

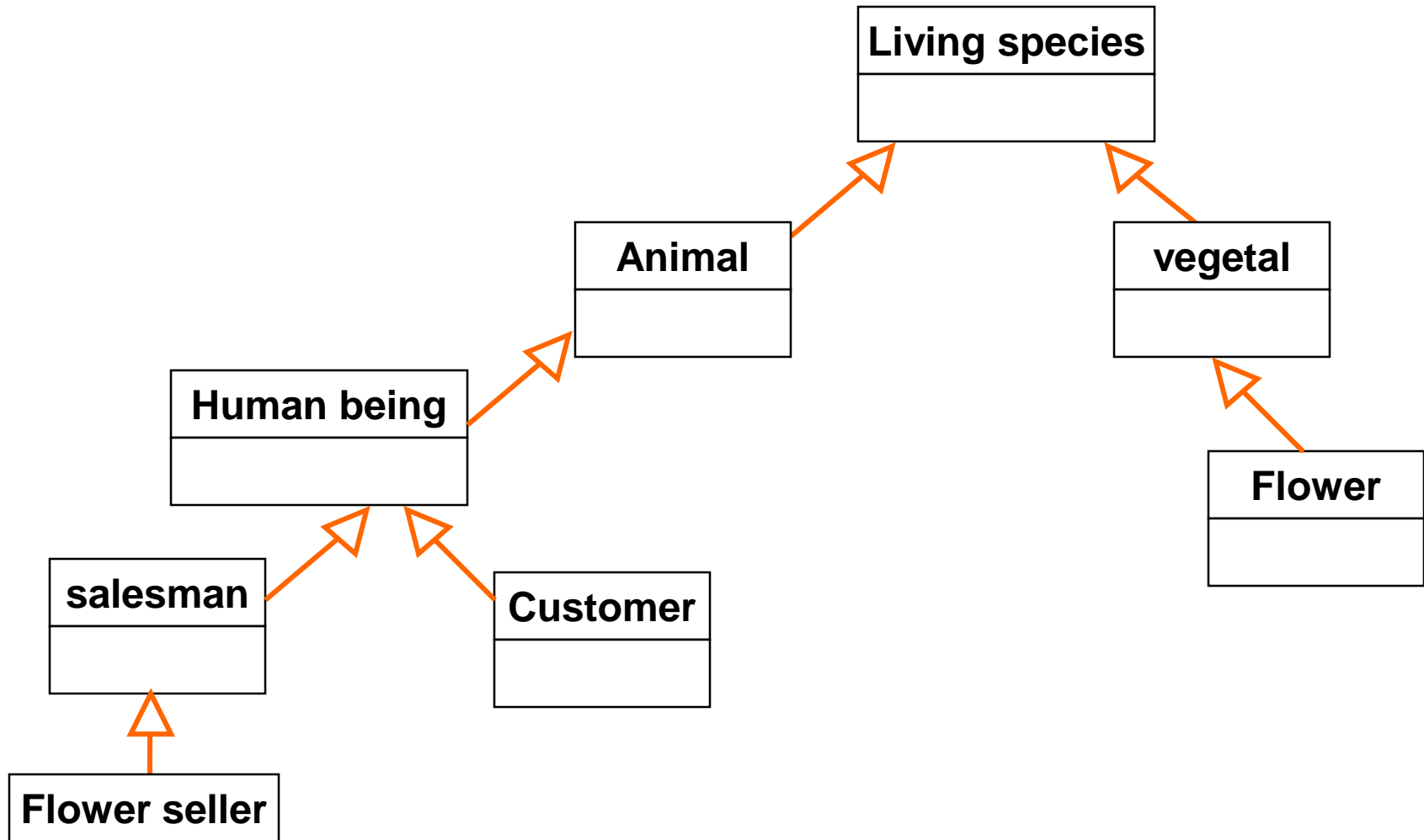
- Class one above
  - ◆ Parent class
- Class one below
  - ◆ Child class
- Class one or more above
  - ◆ Superclass, Ancestor class, Base class
- Class one or more below
  - ◆ Subclass, Descendent class, Derived class

# Set-Specialization



# Example of inheritance tree

---



# Specialization types

---

- Totality
  - ◆ Total: derived classes form a partition
  - ◆ Partial
- Exclusion
  - ◆ Mutual exclusive: object cannot belong to two or more derived classes
  - ◆ Inclusive

# NL Requirements Specification

---

- Requirements specifications are often written in natural language (NL)
  - ◆ At least in the first draft.
- NL is, by nature, subject to ambiguity and misinterpretation.
- Inaccuracies and ambiguous terms must be removed
  - ◆ Necessary an in-depth analysis of the specification document

# Essential guidelines

---

- If a concept has significant properties and/or describes types of objects with an autonomous existence, it can be represented by a **class**.
- If a concept has a simple structure, and has no relevant properties associated with it, it is likely an **attribute** of a class.
- If a concept provides a logical link between two (or more) entities, it is convenient to represent it by means of an **association**.
- If one or more concepts are particular cases of another concept, it is convenient to represent them by means of a **generalization**.



# Modeling strategies

---

- Top-down
  - ◆ Start with abstract concepts and perform successive refinements
- Bottom-up
  - ◆ Start with detailed concepts and proceed with integrating different pieces together
- Inside-out
  - ◆ Like bottom-up but beginning with most important concepts first
- Hybrid

# Conceptual model quality

---

- Correctness
  - ◆ No requirement is misrepresented
- Completeness
  - ◆ All requirements are represented
- Readability
  - ◆ It is easy to read and understand
- Minimality
  - ◆ There are no avoidable elements

# Example

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
  - For each course participant (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, employer's name, address and telephone number, previous employers (and period employed), the courses attended (there are about 200 courses) and the final assessment of each course.
  - We need also to represent the seminars that each participant is attending at present and, for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants.
  - If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
  - For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.
-

# Requirement analysis

---

- Choose the appropriate level of abstraction
  - ◆ Identify the main concepts
- Construct a glossary of terms
- Identify synonyms and homonyms, and standardize terms
- Make cross-references explicit
- Standardize sentence structure
- Avoid complex phrases

# Main concepts

---

- We wish to create a IS for a company that runs training **course**. For this, we must store data about the **trainee** and the **instructor**.
- For each **course** **participant** (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, **employer**'s name, address and telephone number, previous employers (and period employed), the courses attended (there are about 200 courses) and the final assessment of each course.
- We need also to represent the **seminar** that each participant is attending at present and, for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants.
- If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
- For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the **tutor** is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

# Glossary

---

Term	Description	Synonym	Links
Course	Course offered. Can have various editions.	Seminar	Instructor, Trainee
Trainee	Participant in a course. Can be an employee or self-employed.	Participant	Course, Employer
Instructor	Course tutor. Can be freelance.	Tutor	Course
Employer	Company by which a trainee is employed or has been employed.		Trainee

# Standardize and simplify

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
  - For each ~~course participant~~ <sup>trainee</sup> (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, ~~employer's name, address and telephone number,~~ <sup>current employer</sup> previous employers (and <sup>editions</sup> ~~period employed~~), the courses attended (there are about 200 courses) and the final assessment of each course. <sup>start date and the end date of</sup> For each employer we store the name address and phone number.
  - We need also to ~~represent the seminars that each participant is attending at present and,~~ <sup>record</sup> for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants.
  - If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
  - For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.
-

# Example

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
- For each trainee (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, current employer's, previous employers (and start date and end date of the period employed), the courses editions attended (there are about 200 courses) and the final assessment of each course edition.
- For each employer we store the name, address, and phone number
- Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. We need also to record for each day, the places and times the classes are held. For each edition, we represent the start date, the end date, and the number of participants.
- If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
- For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.



# Example

## Generic statements

We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.

- For each employer we store the name, address, and phone number
- We need also to represent course editions and, for each day, the places and times the classes are held. Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. For each edition, we represent the start date, the end date, and the number of participants.
- If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
- For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

# Example

## Statements about Trainees

For each trainee (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, current employer's, previous employers (and start date and end date of the period employed), the courses editions attended (there are about 200 courses) and the final assessment of each course edition.

- For each instructor (about 500), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

# Example

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
- For each trainee (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, current employer's, (and start date and end date if self-employed), the

## Statements about employers

For each employer we store the name, address, and phone number

- and the number of pa
- If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.
  - For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.
-

# Example

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors

## Statements about Courses

Each course has a code and a title and any course can be given any number of times. Each time a particular course is given, we will call it an 'edition' of the course. We need also to record for each day, the places and times the classes are held. For each edition, we represent the start date, the end date, and the number of participants.

An instructor can be part-time or full-time employed by the training company, or be a freelance.

# Example

---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
- For each trainee (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, current employer's, previous employers (and start date and end date of the period employed), the

Statements about types of employers  
If a trainee is a self-employed professional, we need to know his or her area of expertise, and, if appropriate, his or her title. For somebody who works for a company, we store the level and position held.

tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

# Example

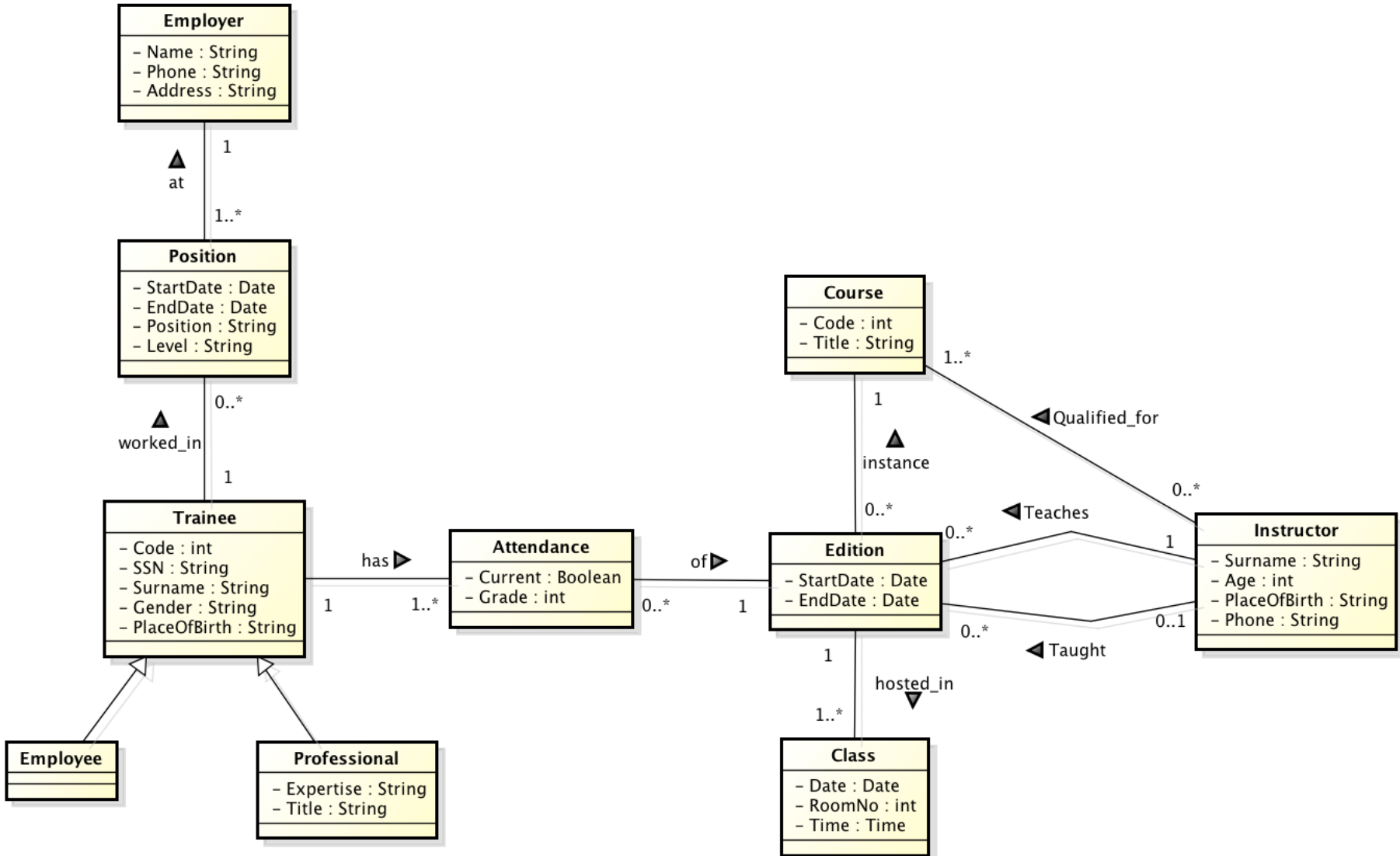
---

- We wish to create a IS for a company that runs training courses. For this, we must store data about the trainees and the instructors.
- For each trainee (about 5000), identified by a code, we want to store the social security number, surname, age, gender, place of birth, current employer's, previous employers (and start date and end date of the period employed), the

Statements about types of instructors

For each instructor (about 300), we will show the surname, age, place of birth, the edition of the course taught, those taught in the past and the courses that the tutor is qualified to teach. All the instructors' telephone numbers are also stored. An instructor can be permanently employed by the training company or can be freelance.

pkg TrainingCompany



# References

---

- Fowler, M. “UML Distilled: A Brief Guide to the Standard Object Modeling Language – 3<sup>rd</sup>ed.”, Addison–Wesley Professional (2003)
- Lindland, O.I., Sindre, G. and Solvberg, A.: Understanding quality in conceptual modeling. IEEE Software, 11(2):42–49, (1994).
- Bolloju, N. and Leung, F.: Assisting novice analysts in developing quality conceptual models with UML. Communications of the ACM, 49(7), (2006).