Software Systems

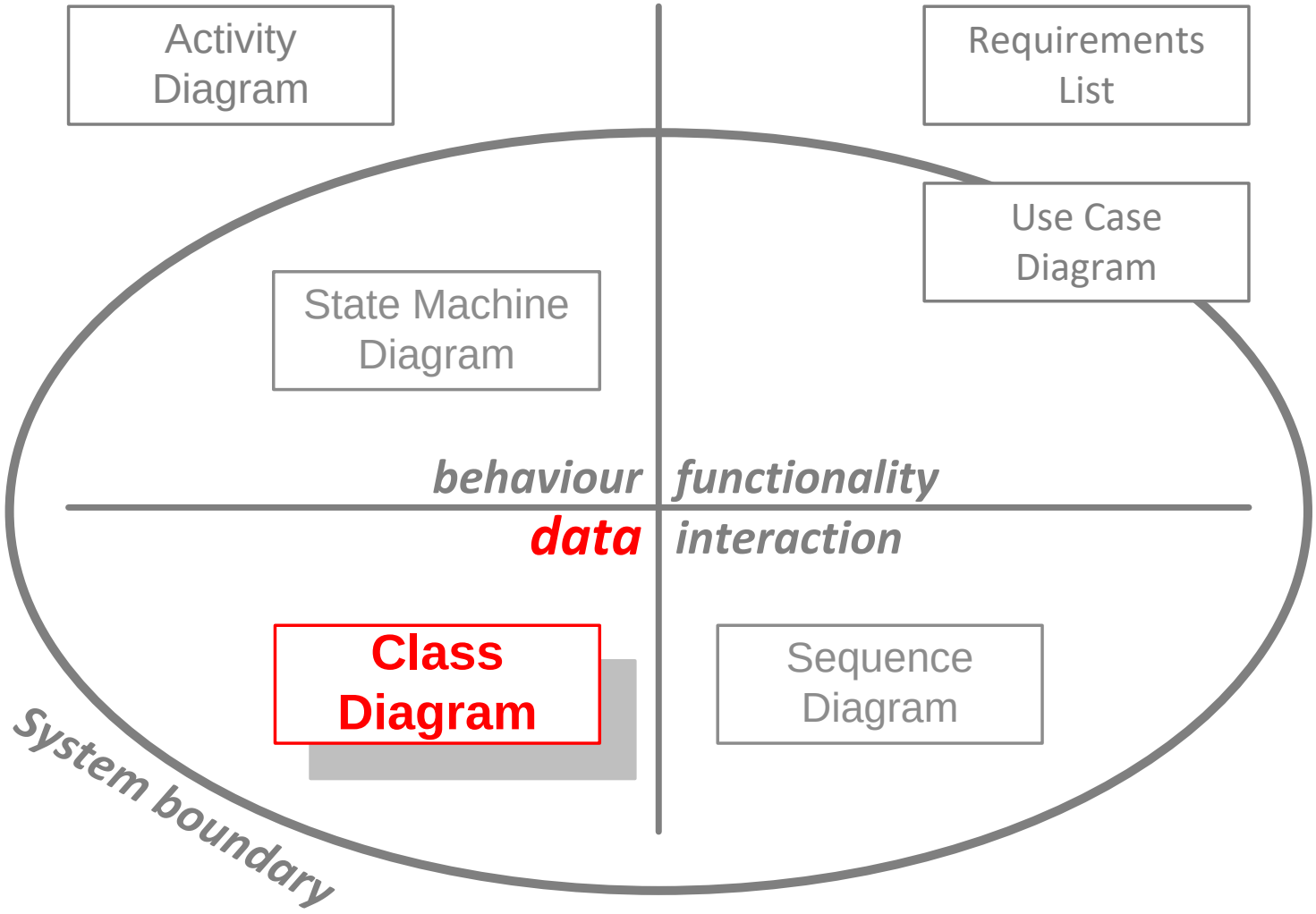
# Design – 2A

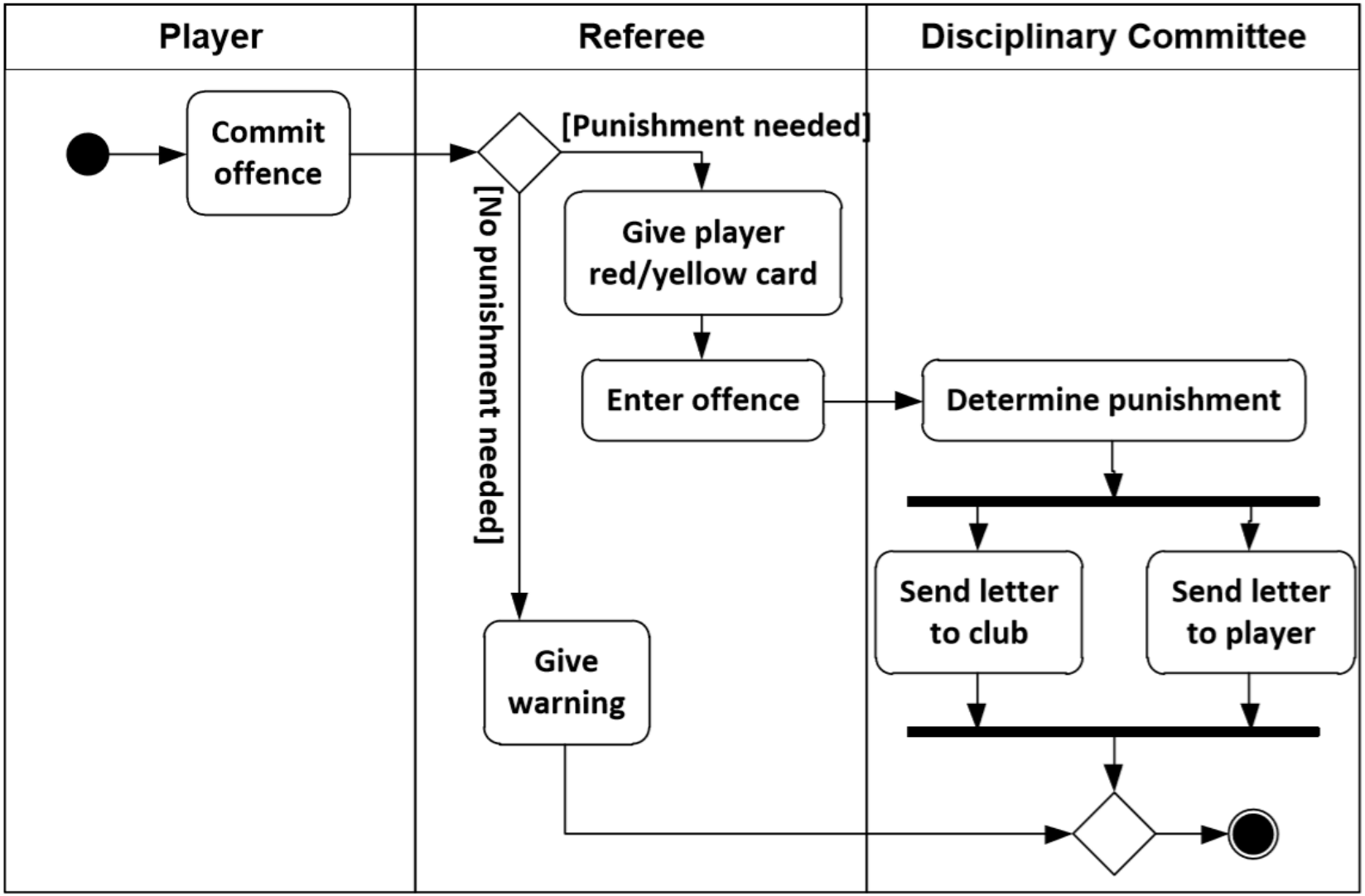
# Class Diagrams

Joke van Staalduinen

Credits to Klaas Sikkel

# Today's topic




Player
Referee
Disciplinary Committee
Commit offence
[Punishment needed]
[No punishment needed]
Give player red/yellow card
Enter offence
Determine punishment
Send letter to club
Send letter to player
Give warning

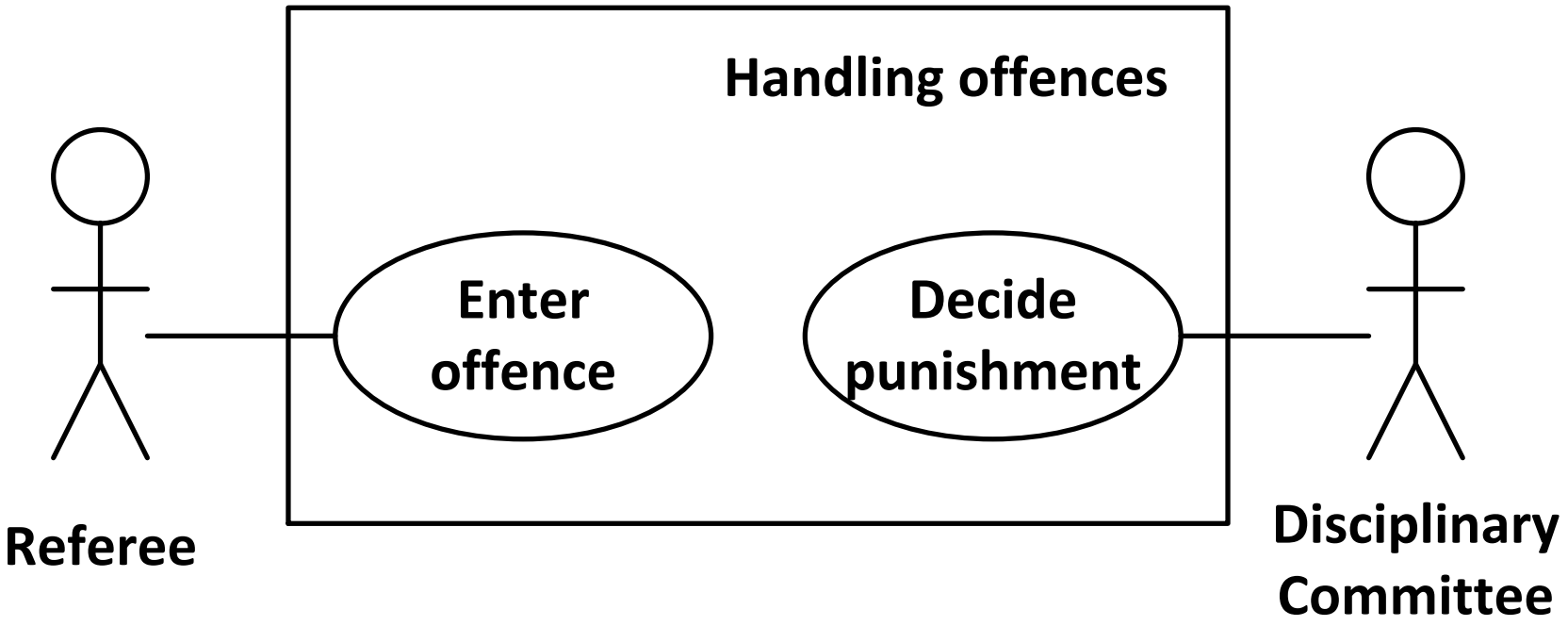
# 2.2. Requirements list

| Nr | *Requirement* | *Use case(s)* |
|---|---|---|
| 1 | ***As a referee I want ...***<br>**To enter details about red / yellow card that I gave** | **Enter offence** |
| 2 | ***As the Disciplinary Committee I want...***<br>**To decide which punishments are given for which offences** | **Decide punishment** |

# 2.3. Actor list

| *Actor* | *Description* |
|---|---|
| **Referee** | **Referees matches, gives warnings and yellow or red cards** |
| **Disciplinary Committee** | **Determines the punishment which a player will get after receiving a card, taking into account the player's previous behaviour** |

# 1. Use case diagram (basics)



# 2.1 Glossary

| *Term* | *Description* |
|---|---|
| **Offence** | **Act of a player in violation of the rules** |
| **Warning** | **For a minor offence, a player can be reprimanded (not punished). Warnings are not recorded** |
| **Red card** | **Signals a major offence. The player must leave the match immediately and will receive punishment** |
| **Yellow card** | **Signals a minor offence. With one yellow card a player can continue the match, but the player will receive punishment** |

# 2.5. Use case descriptions (short)

| *Use case* | *Description* |
|---|---|
| **Enter offence** | **After the match the referee enters a description of the offence and the type of card** |
| **Decide punishment** | **The Disciplinary Committee, taking the circumstances into account, decides on a suitable punishment for the offence** |

# 2.6. Use case descriptions (extended)

Use case description: **Enter offence**

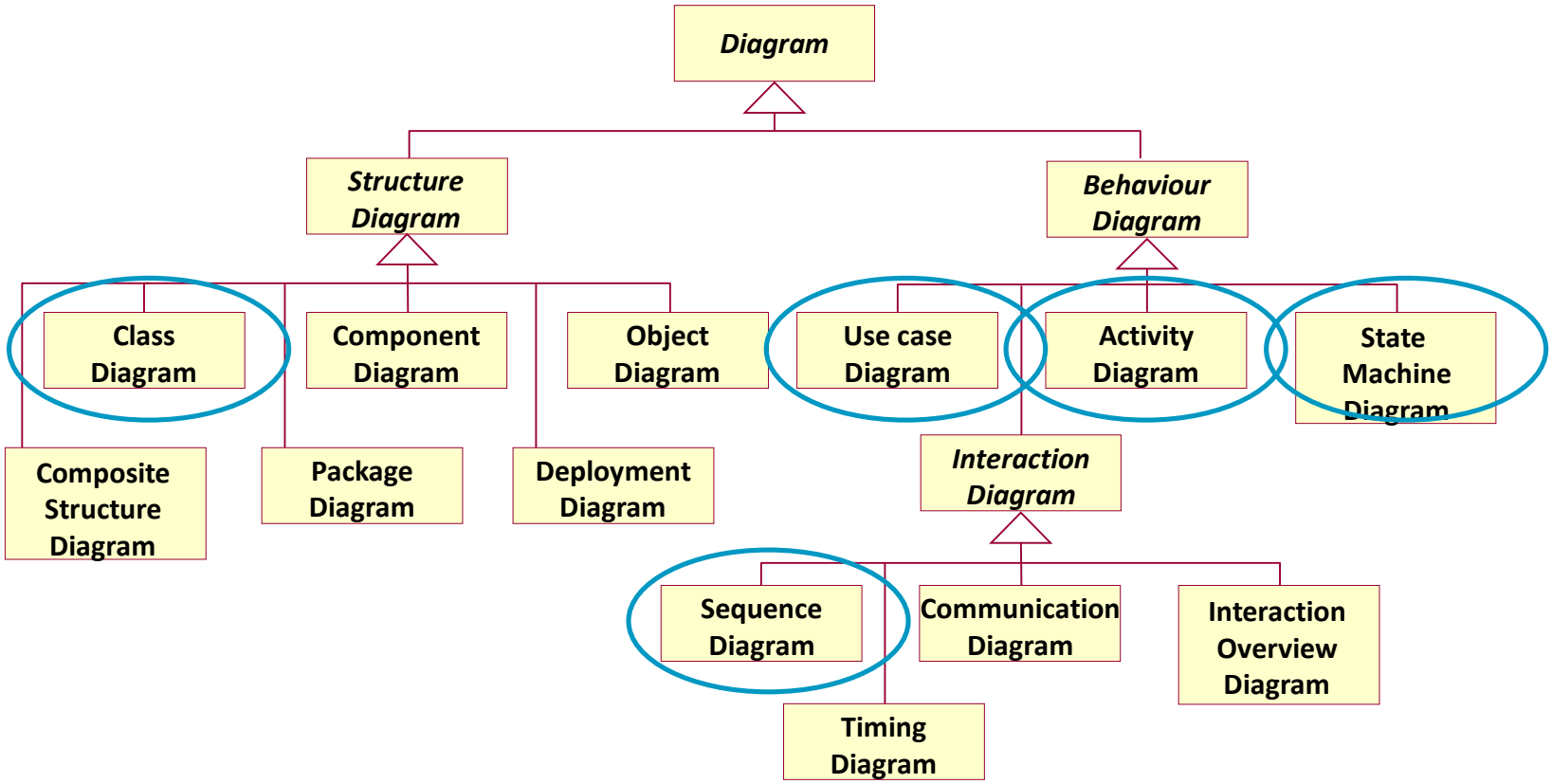
| *Actor Action* | | *System Response* | |
|---|---|---|---|
| **1** | **Start application** | **2** | **Show data entry fields** |
| **3** | **Enter name of player and club** | **4** | **Show all details of player** |
| **5** | **Enter match details, Description of offence, and type of card** | **6** | **Save and send notification to the Disciplinary Committee** |

**Alternatives:**

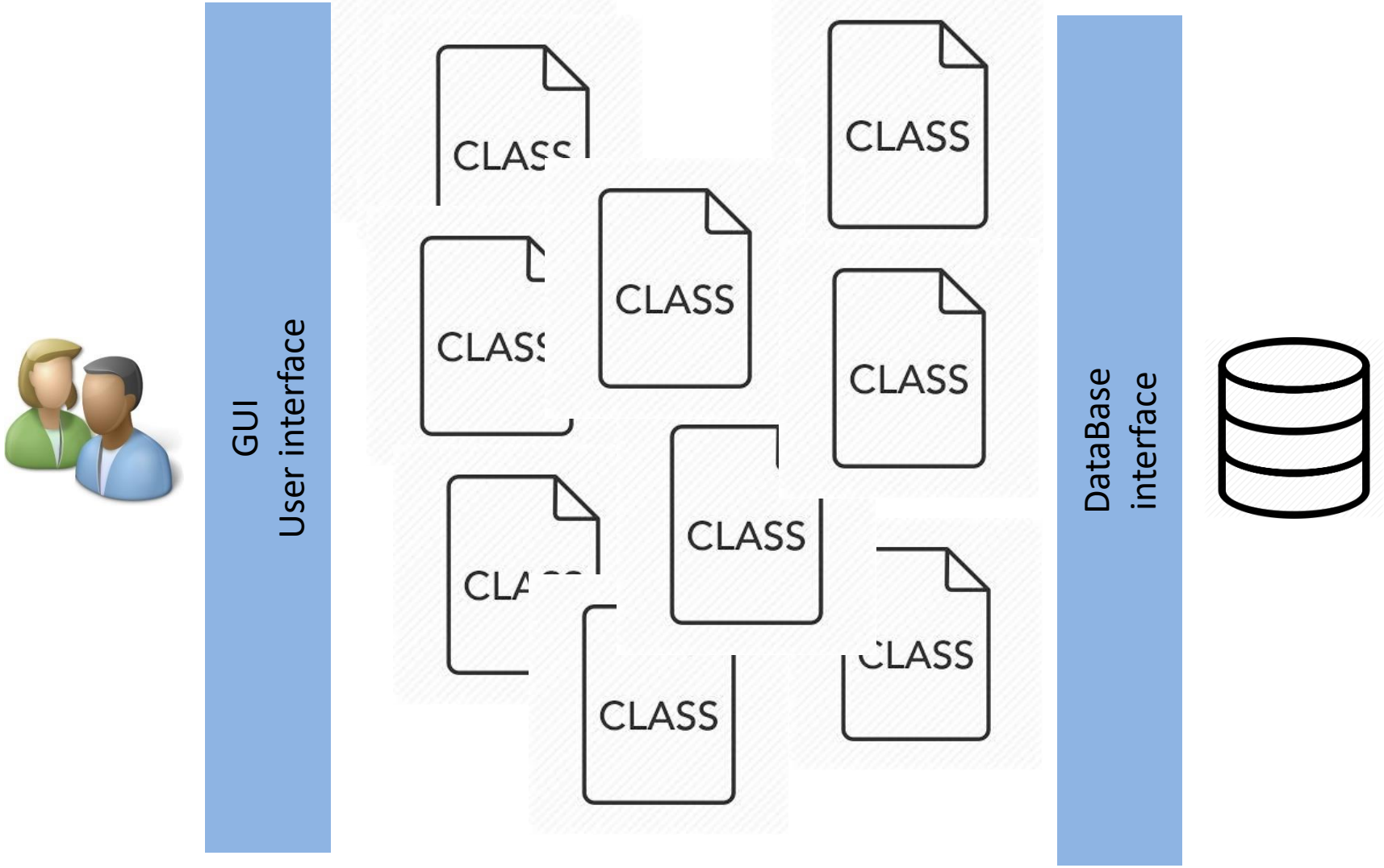
**Step 4: Show list if it matches more than one person**

**Stap 4A: Select person from the list**

# Unified Modelling Language



# Developing an Information System



# Contents

# 1. What is object-orientation?

# What is object-orientation?

Programming / Software Engineering perspective:

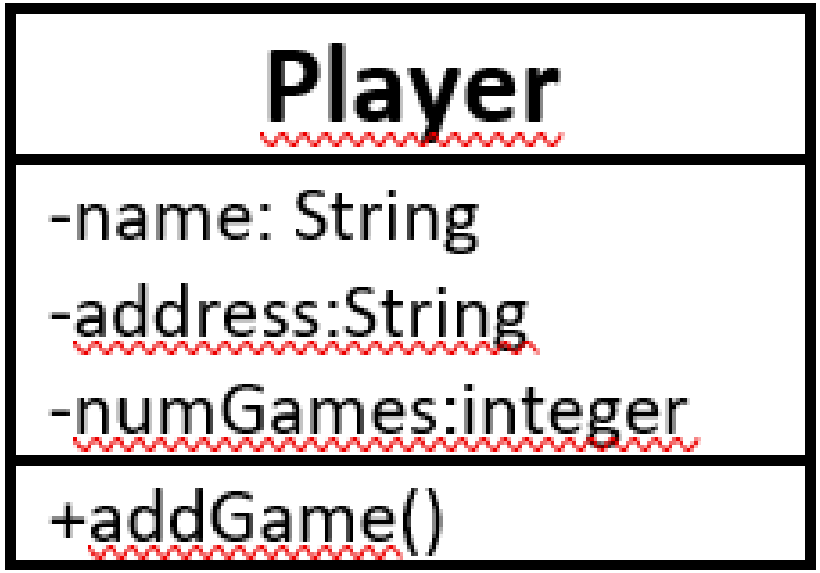
- A programming paradigm in which functionalities are assigned to particular kinds of data

# Pre-object-oriented programming

Oversimplification:

- In the early days of programming, the focus was mostly on programming logic. What happened to the data depended on the logic.
- In object-oriented programming the data are more central and the programming logic depends on the (type of) data.

# Simplified example of a class



# Attributes

| | | |
|---|---|---|
| **public** | + | anywhere in the program and may be called by any object within the system |
| **private** | - | the class that defines it |
| **protected** | # | (a) the class that defines it or (b) a subclass of that class |
| **package** | ~ | instances of other classes within the same package |

# 2. Class diagrams (static structure)

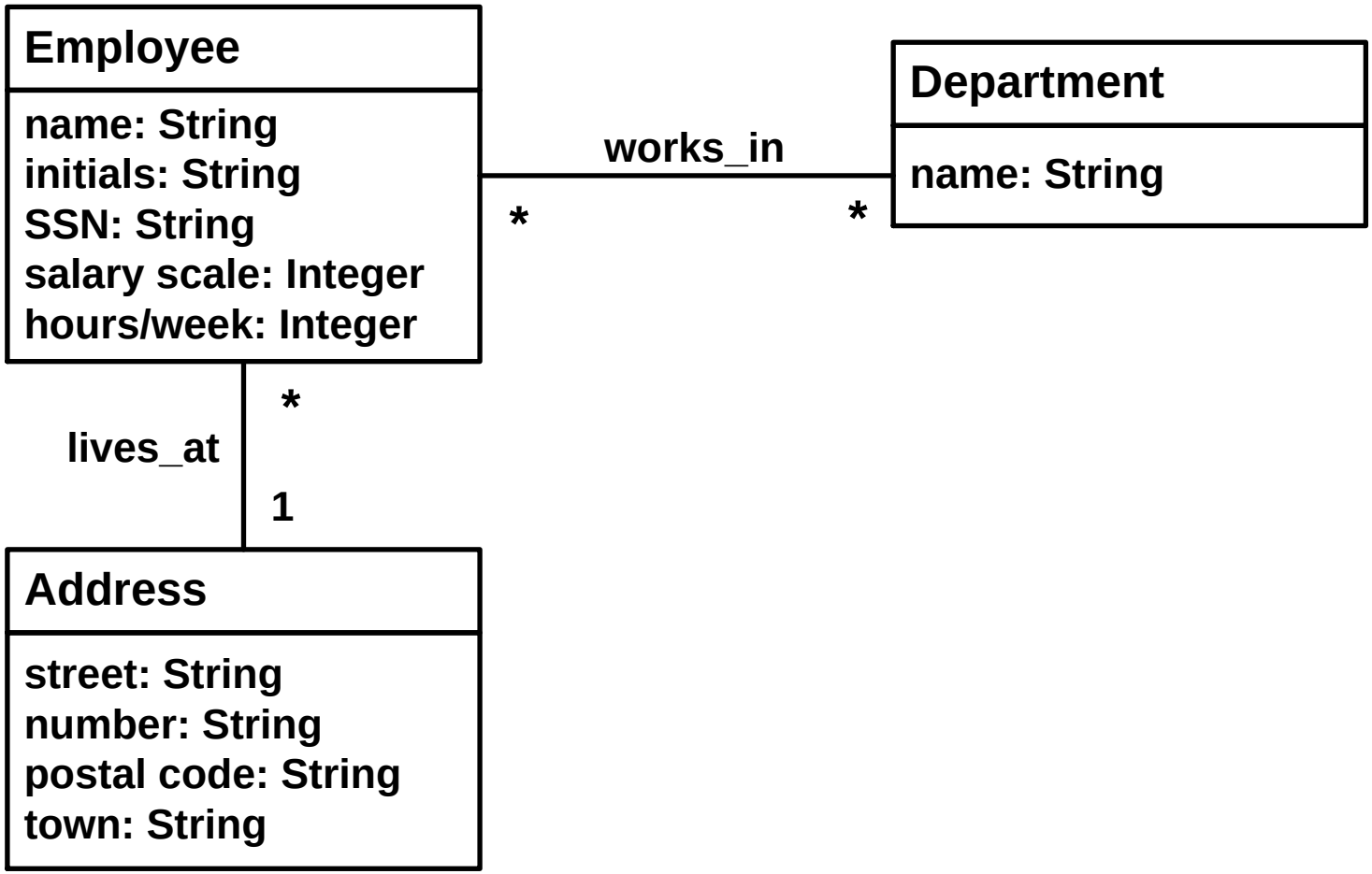
# 2. Class diagrams

- A class diagram is used to describe
  - What the objects are that should be stored in the system
  - How these objects are structured
  - How these objects relate to each other

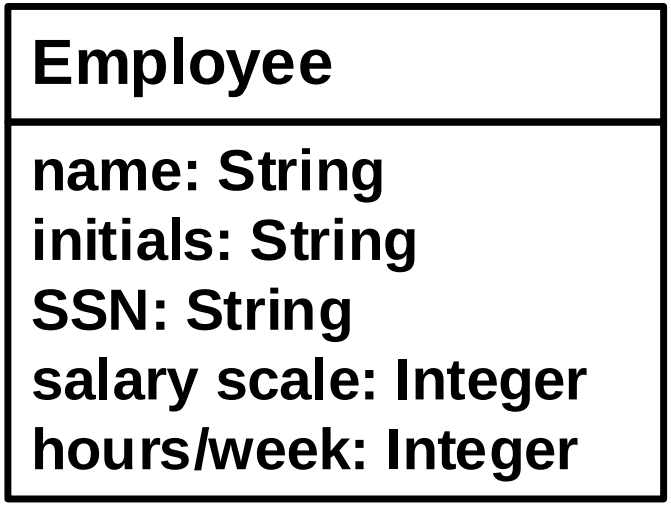
  - *(For now: Ignore the functionality these objects offer, will be added when we do sequence diagrams)*

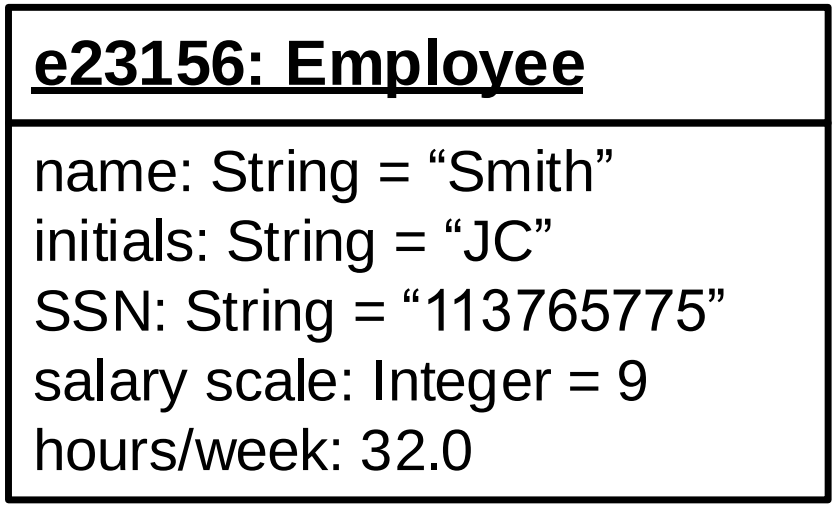
# Simple example

*CD for (part of) a company's administration*



# Class vs. Object




Class

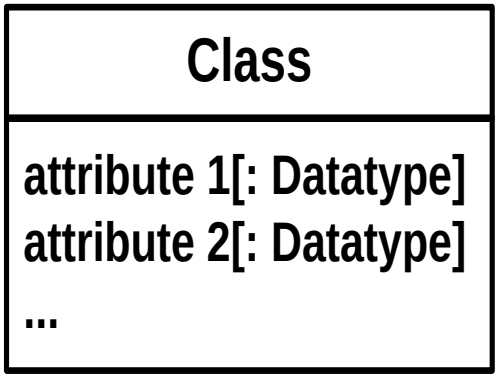
- Describes what we want to know about an employee

Object

- Stores data of one particular employee

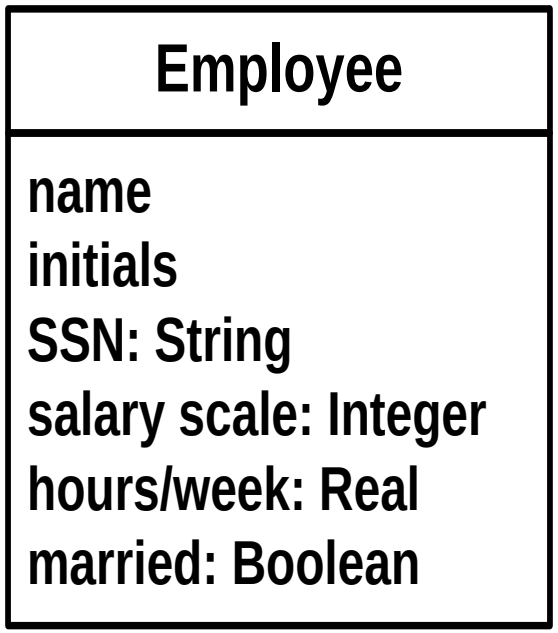
# Elements of a CD (1): Class

Notation:





Example:




*N.B.: class name is a singular noun*

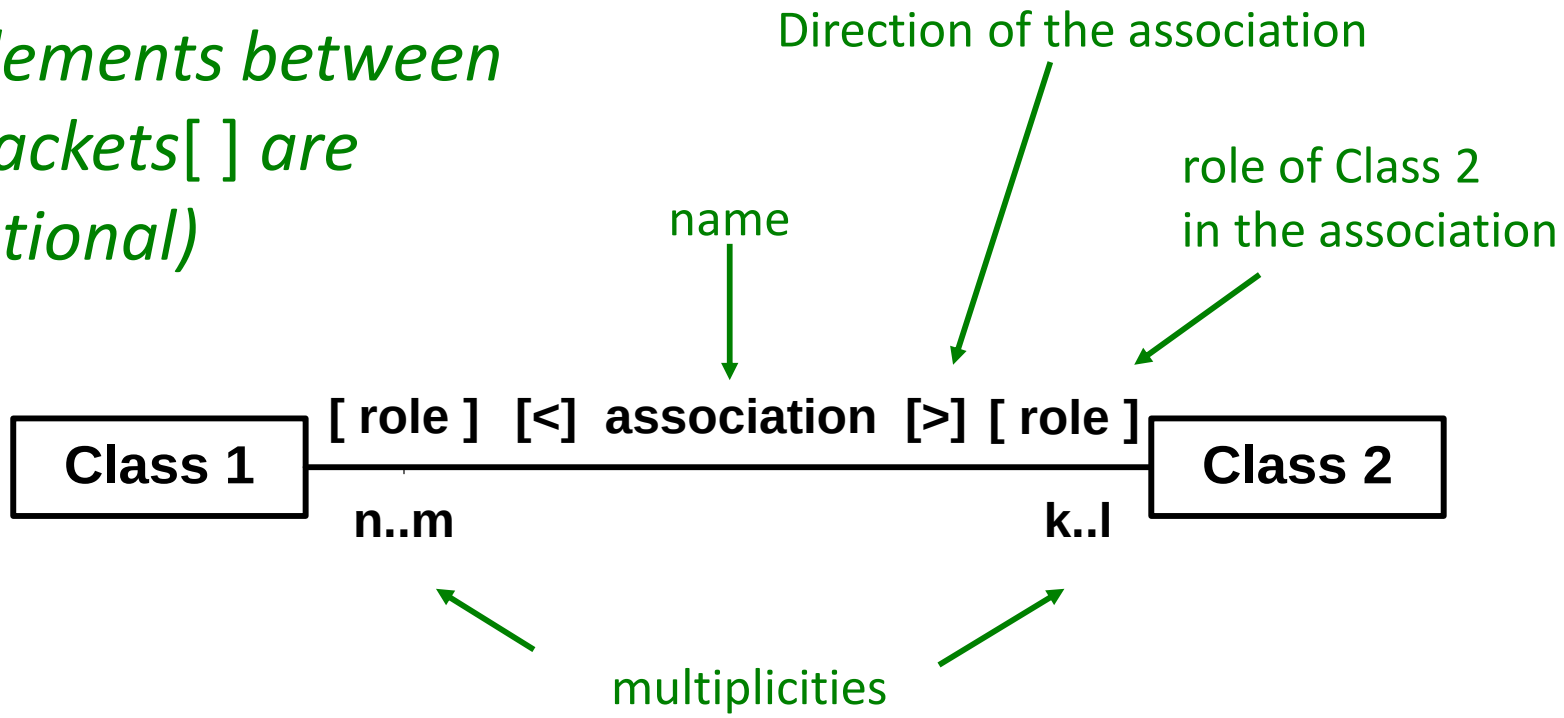
# Elements of a CD (2): Attributes

- A class has zero or more attributes
- For every attribute a standard data type can be specified
  - Can be omitted if the data type is obvious and/or not interesting in a given context
- Standard data types include:
  - Integer
  - Real
  - Boolean
  - String
  - Date
  - Time
  - Money

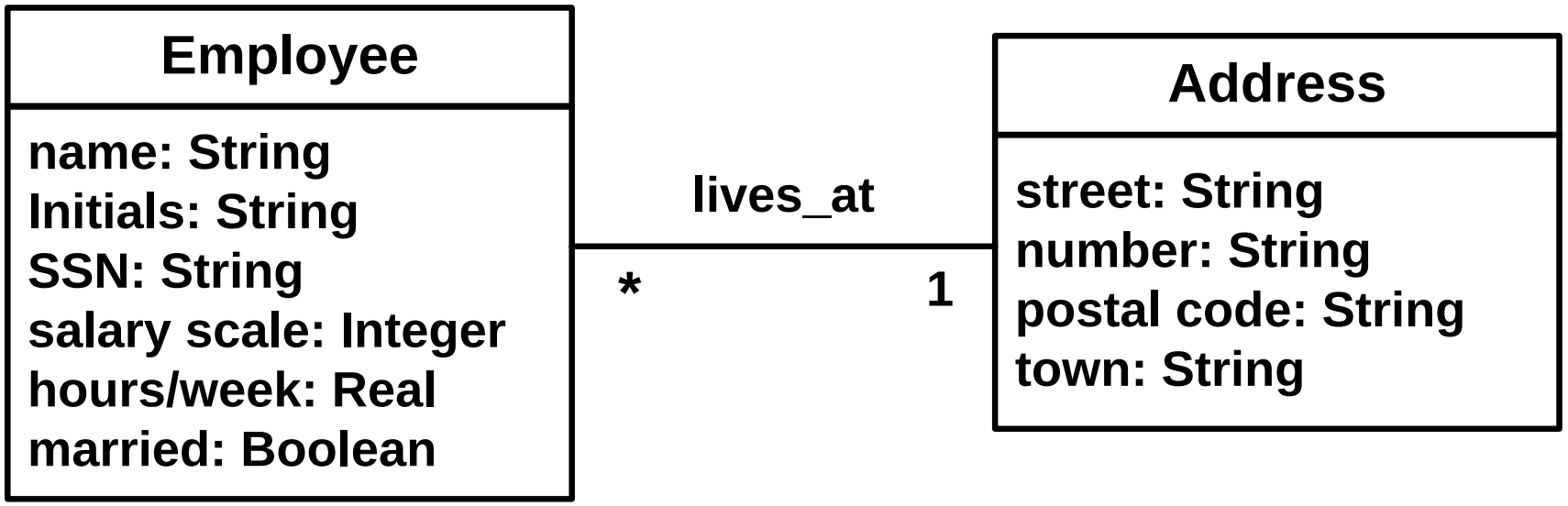
# Elements of a CD (3): Association

(Notation will be elaborated on next slides)

*(Elements between brackets[ ] are optional)*

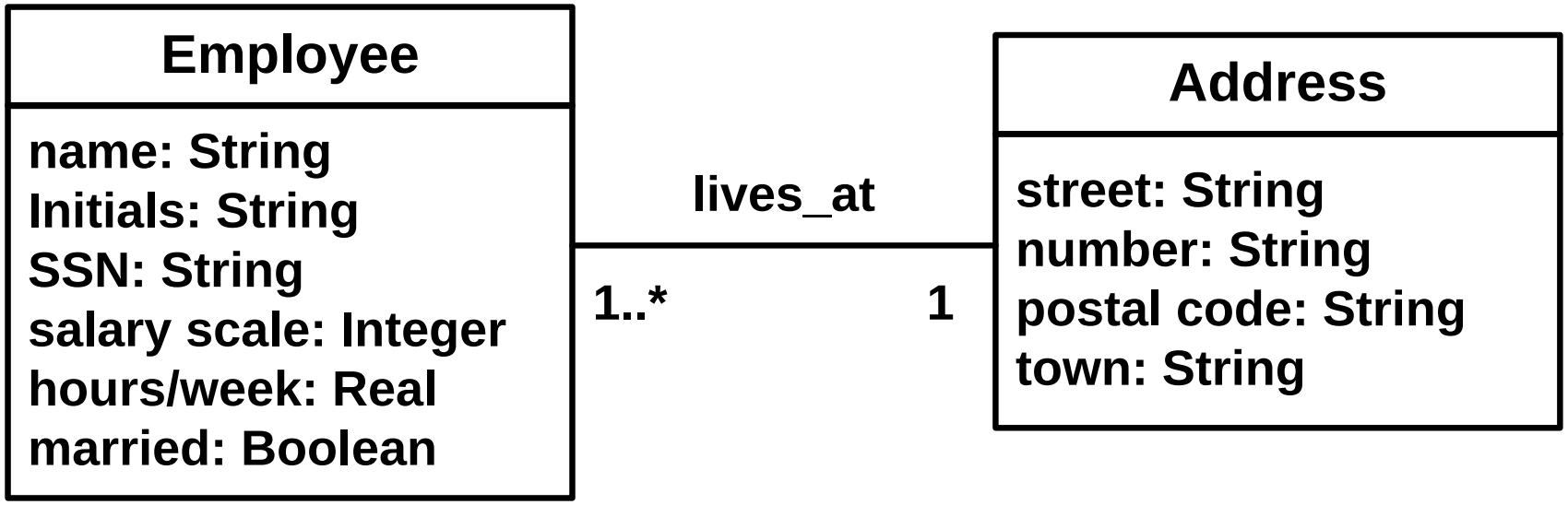


# Examples



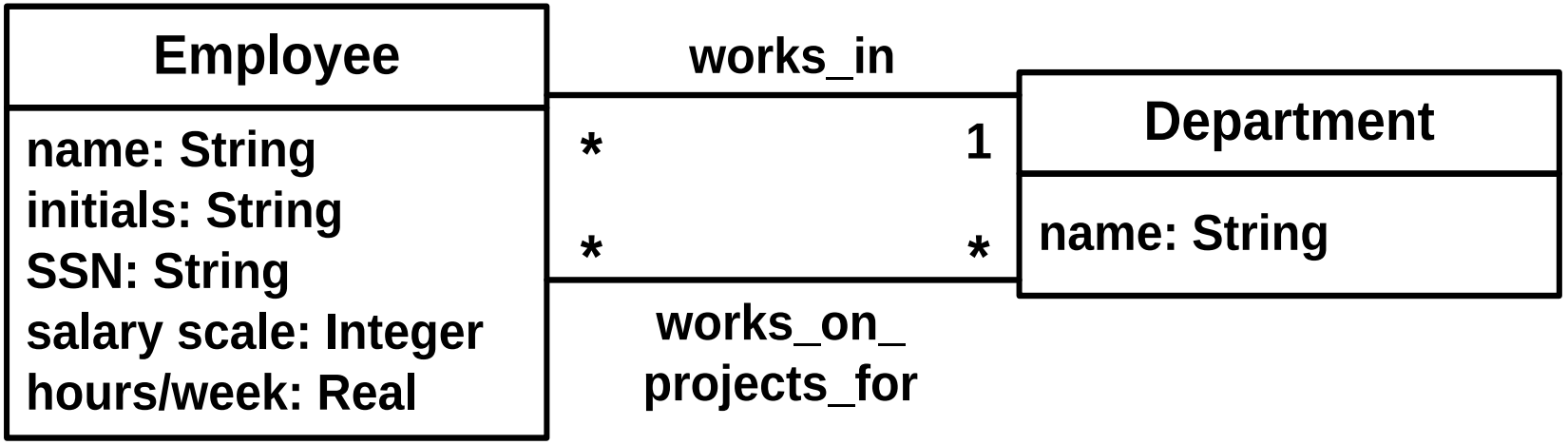

- An employee lives at an address
- At any address zero or more employees are living

# Examples




- An employee lives at an address
- At any address one or more employees are living

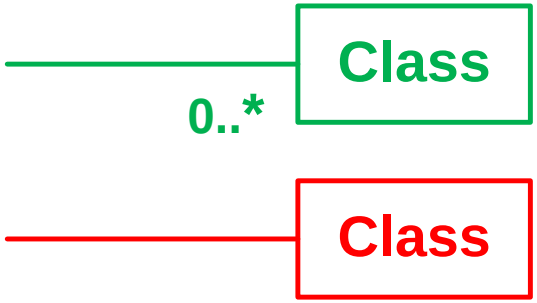
# Examples

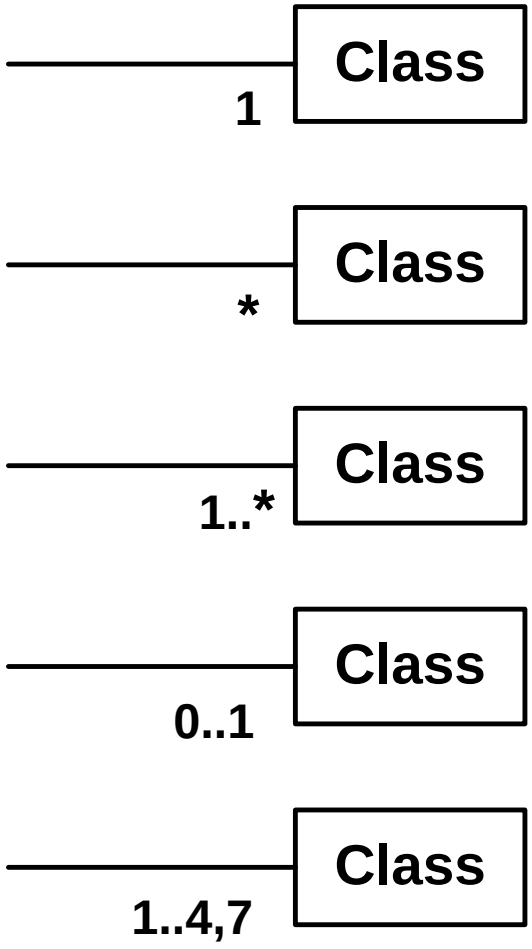



- Multiple associations between two classes

# Multiplicities

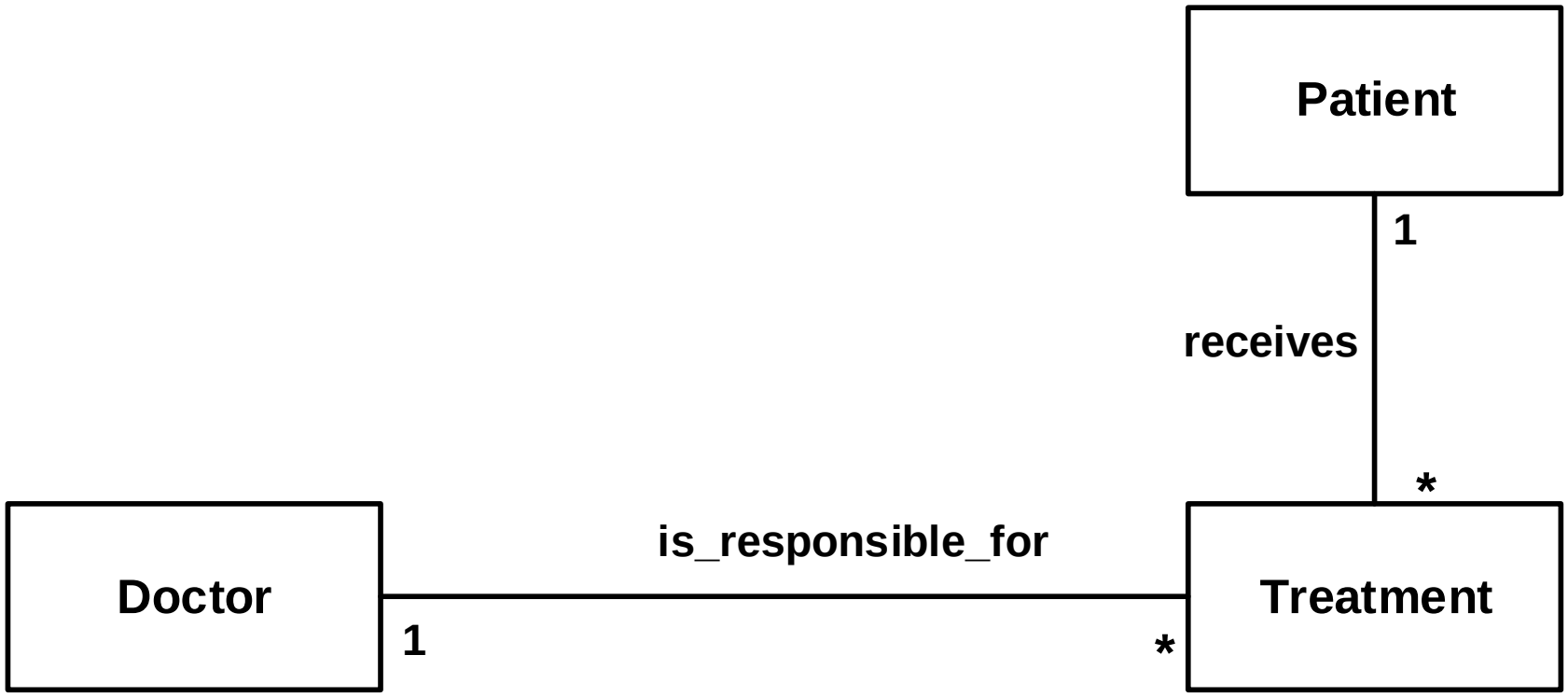
***alternative notations for zero or more***



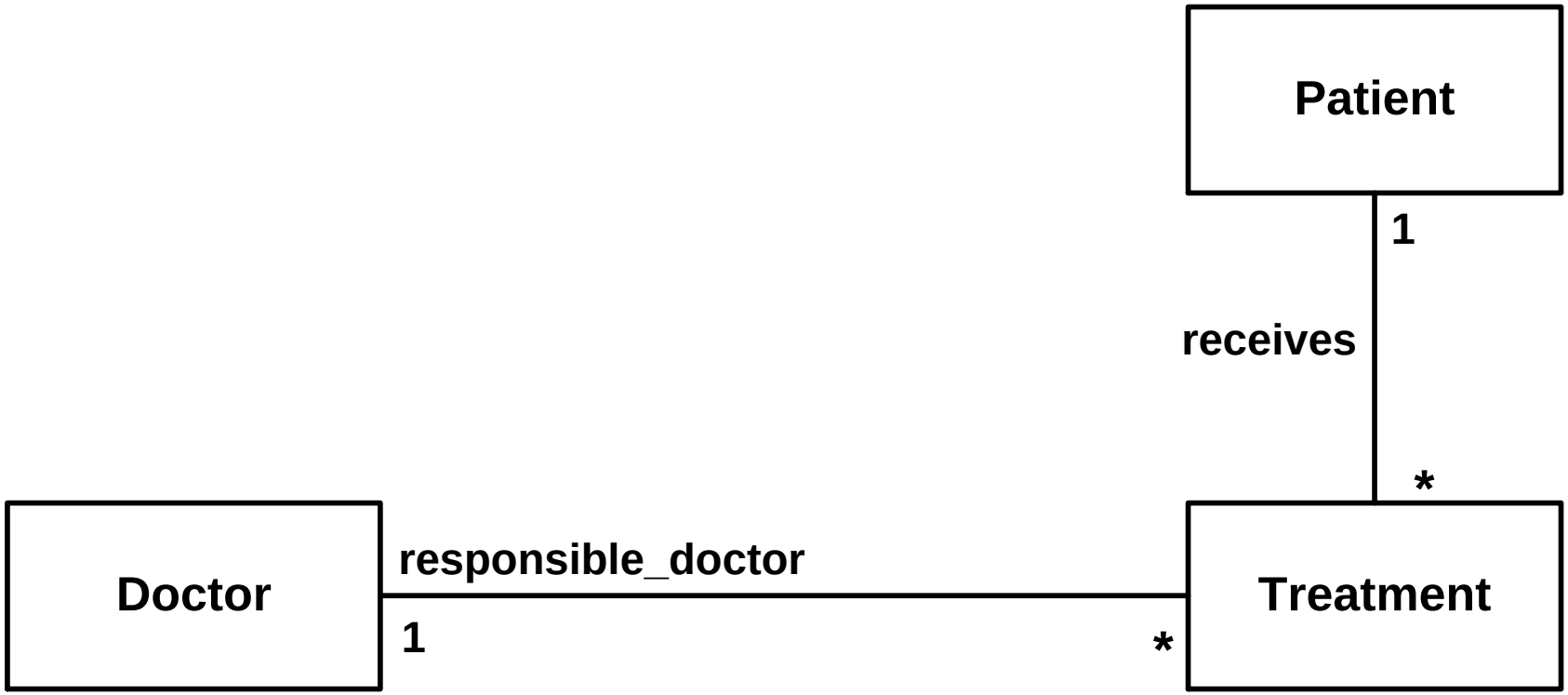
# Roles in an association

- Consider the following CD



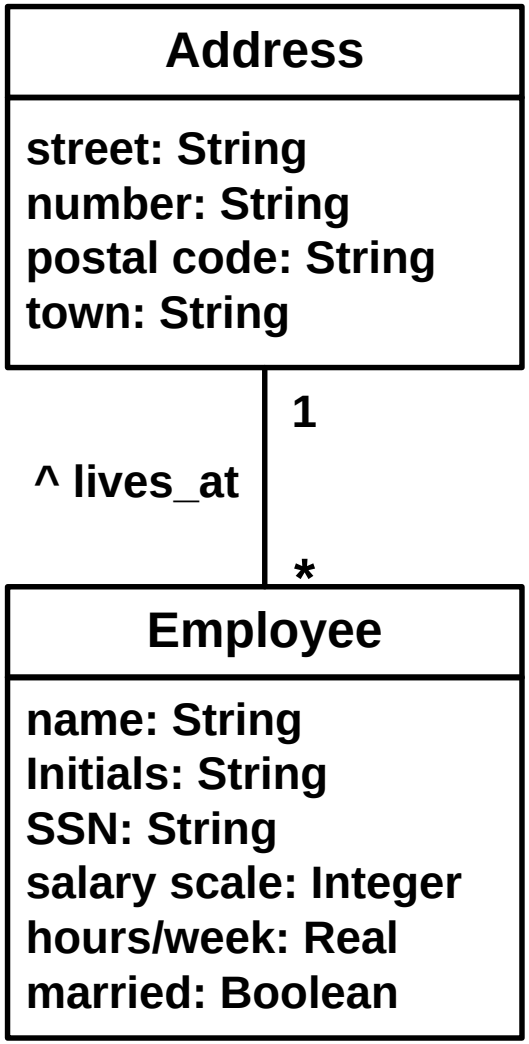
# Roles in an association

- We can use a role name, rather than association name, if a class has a specific role in the association

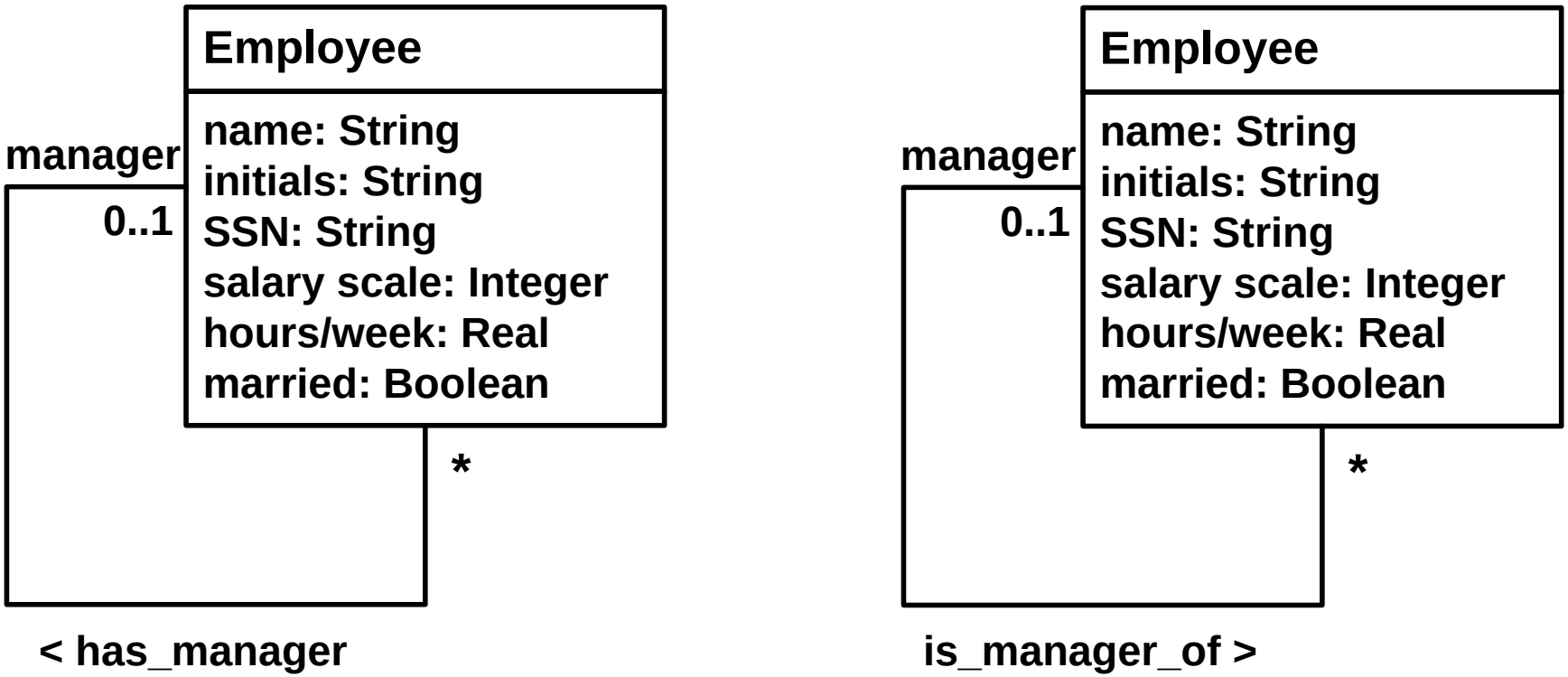


# Direction of an association

- An association has a direction
  (by default:
  - from left to right
  - from top to bottom
- We can explicitly indicate direction with
  < , > , ^ , v



# Direction of an association



# Conventions for associations

- In principle, every association has a name.
  If a role is indicated, the association name can be discarded
- By default an association is read from left to right or top to bottom
  A direction can be stated explicitly by ‘<‘, ‘>’, ‘^’, ‘v’
- All multiplicities are explicitly given

# 3. What does a class diagram represent?

# CD as a model of the physical world

[Lecture 1a]: A model is a simplified representation of part of the world, from a particular view

For Class Diagrams specifically:

- View: Focus on data (not functions, etc.)
- Part of the world: The subject we're modelling
- Simplified: Only those data that should be represented in the system

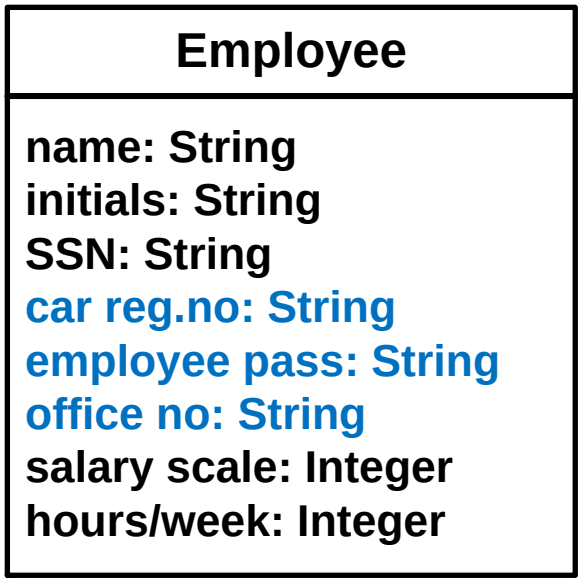
# CD as a model of the physical world

*John Smith*

| **e23156: Employee** |
| --- |
| **name: String = “Smith”**<br>**initials: String = “JC”**<br>**SSN: String = “113765775”**<br>**salary scale: Integer = 9**<br>**hours/week: 32.0** |

*Representation of John Smith in the system*

# Class or attribute?



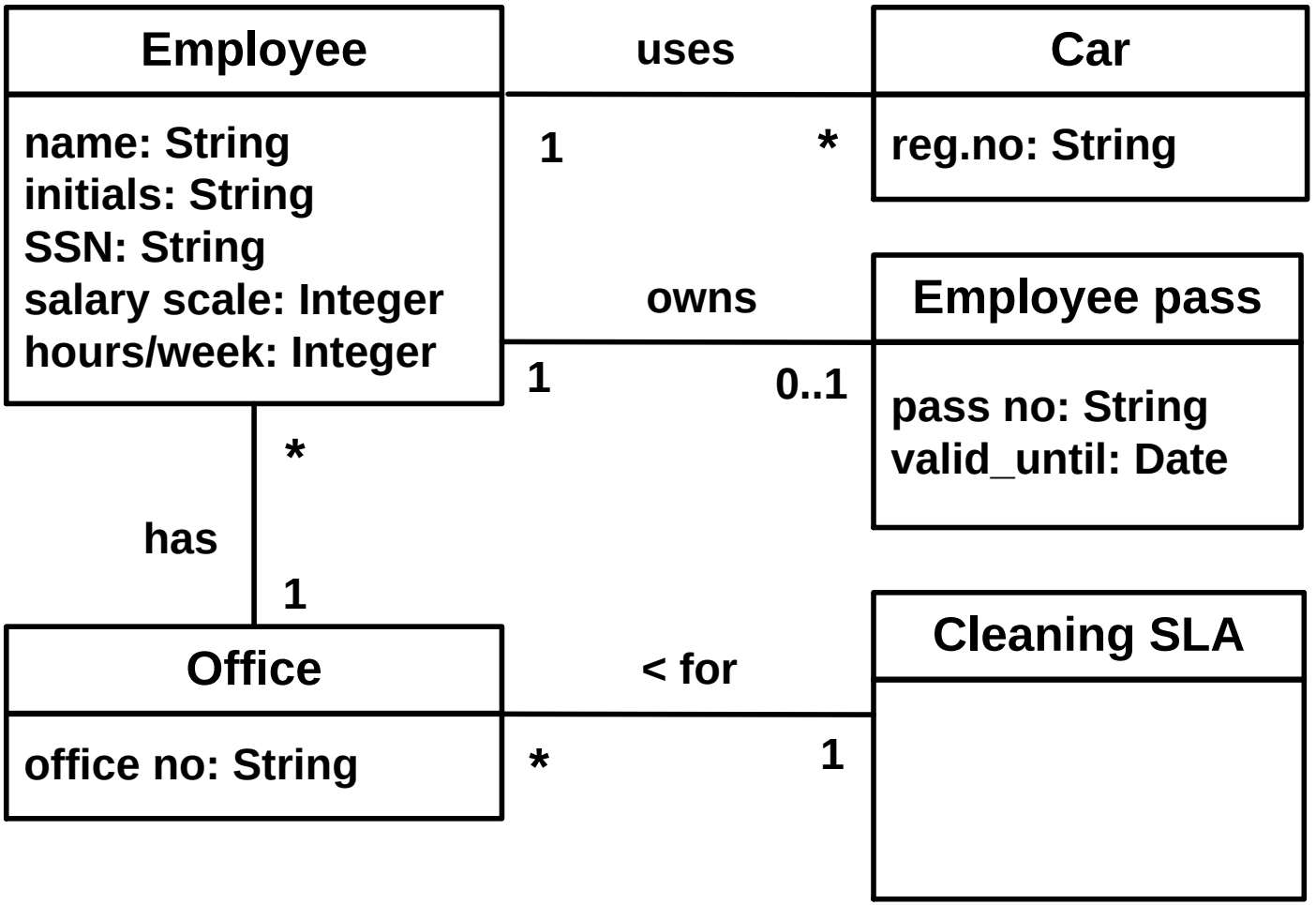

- The blue attributes would normally be OK as attributes
- When more information is added it could be necessary to expand them into classes

# Class or attribute?

Rule of the thumb:

- An entity is an attribute if it is only used as a particular data item of a class
- An entity is a class
  - If it can have multiple values
  - If it has attributes by itself
  - If it has associations with other classes

# Class or attribute?



# Pitfalls

- Synonyms
  (different words for the same concept)

- Homonyms
  (one word for different concepts)

# Special case of homonym

**Animal Farm**

by George Orwell

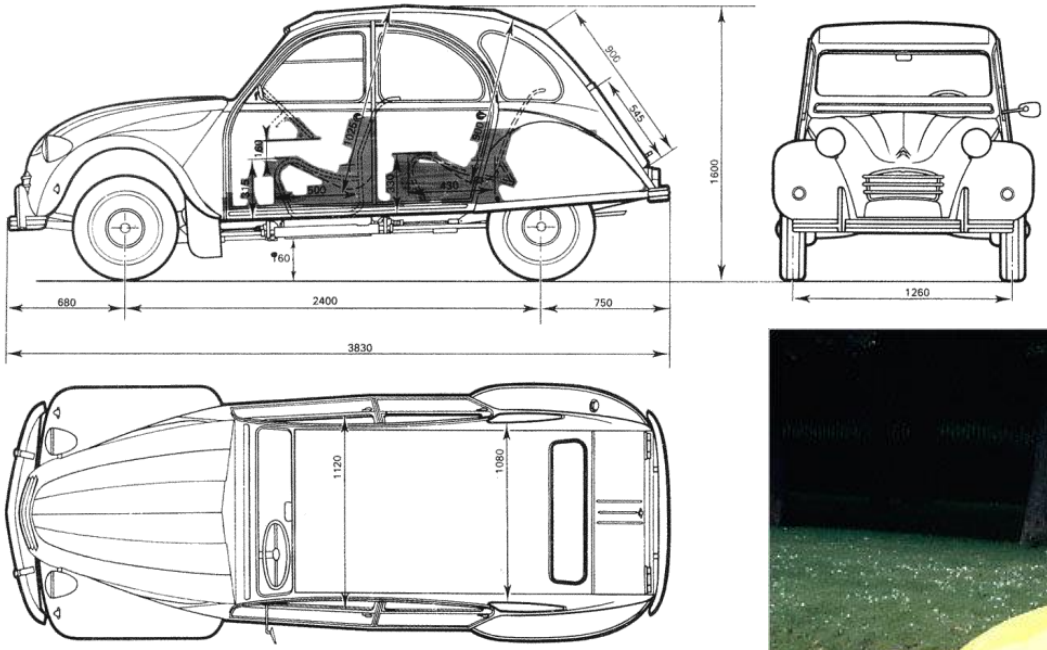
★★★★☆ 3.85 · Rating Details · 1,831,927 Ratings · 30,419 Revie

"All animals are equal but some animals are more equal th
others."

Revolution is in the air at Manor Farm after old Major, a pr
boar, tells the other animals about his dream of freedom a
teaches them to sing "Beasts of England." Mr Jones, the
drunken farmer, is deposed and a committee of pigs takes
the running of the farm. The animals are taught to read an
...more


The book “Animal Farm”

# Special case of homonym





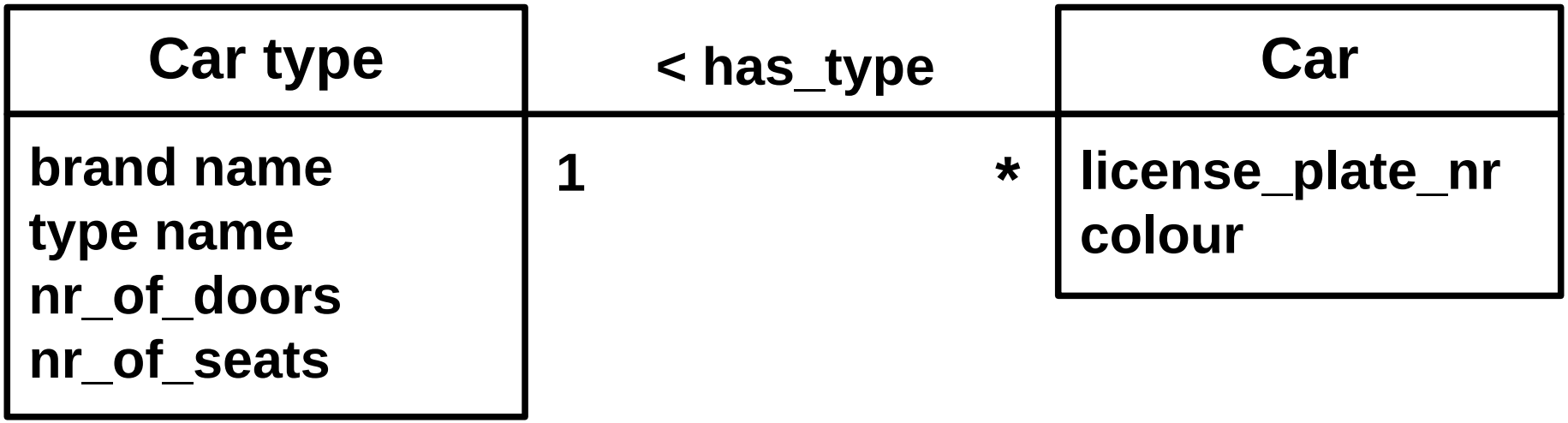
Citroën 2cv

# Different levels of abstraction

Special case of homonym

- **We use one word for a concept on different levels of abstraction**
  - The book ‘Animal Farm’ by George Orwell
  - One copy of the book ‘Animal Farm’
  - The Citroën 2cv, a famous car type with certain properties
  - The Citroën 2cv  which is parked in front of this building

# Different levels of abstraction

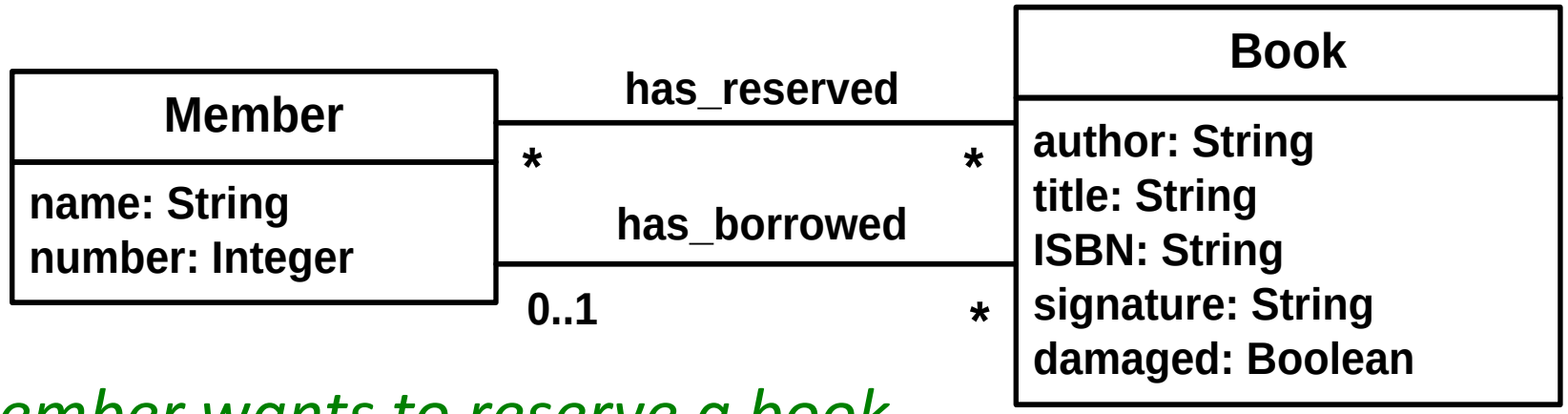



**This distinction is important!**

**And it remains to be difficult!**

# Yet another example

Reservations and loans in a library

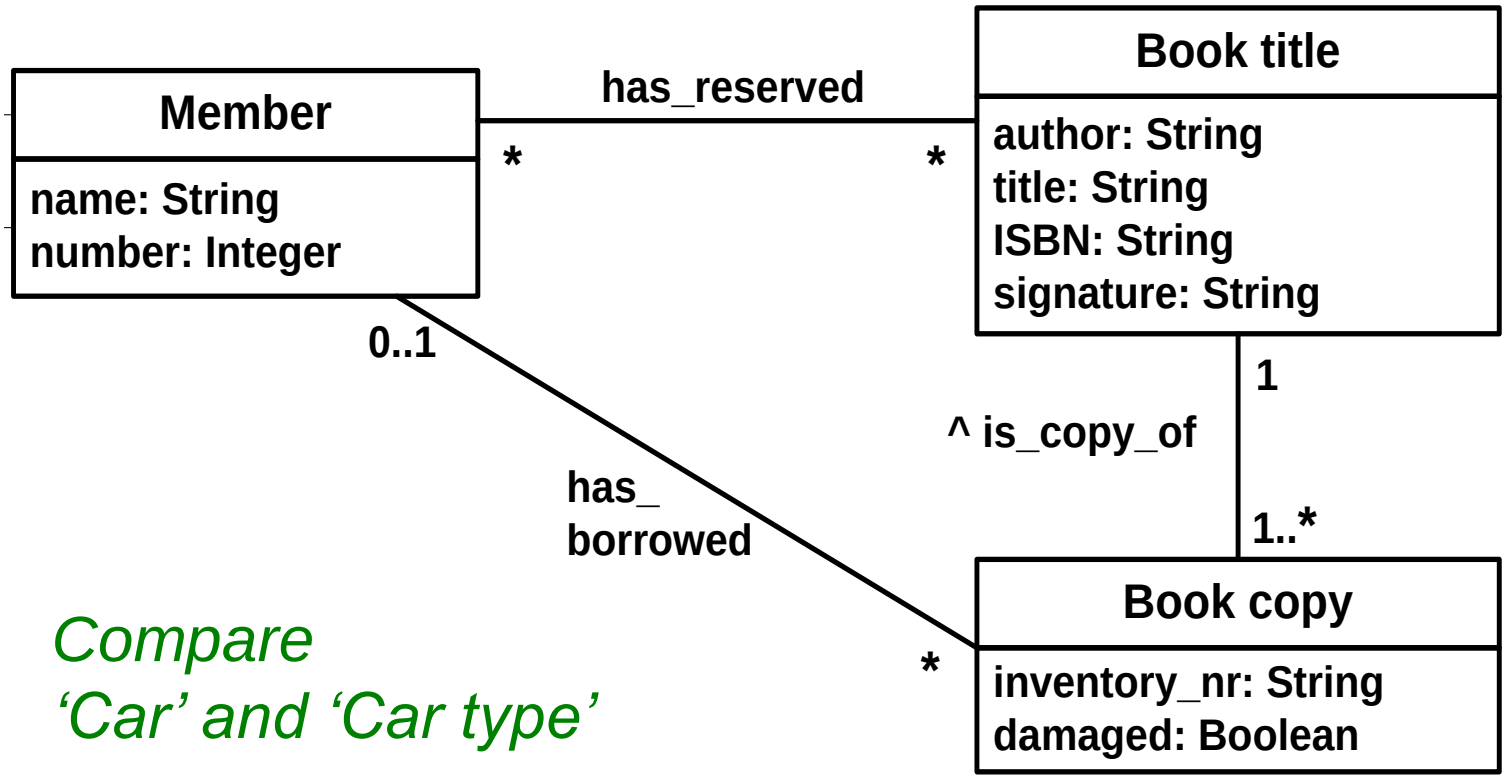



*Member wants to reserve a book. The system says that it has been borrowed by someone and will Be returned in two weeks. However, there are two copies still in the library.*

*Inappropriate model: Impossible to indicate multiple copies of the Same book*

# Yet another example

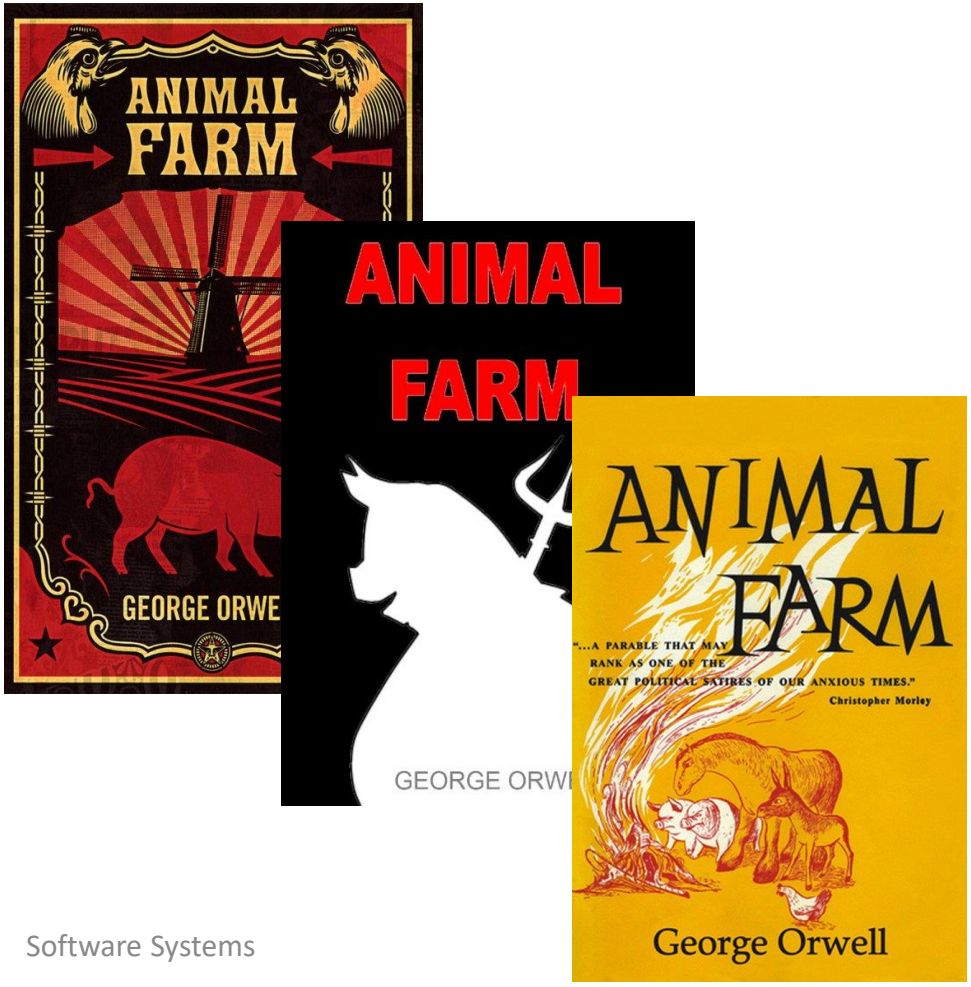
Reservations and loans in a library

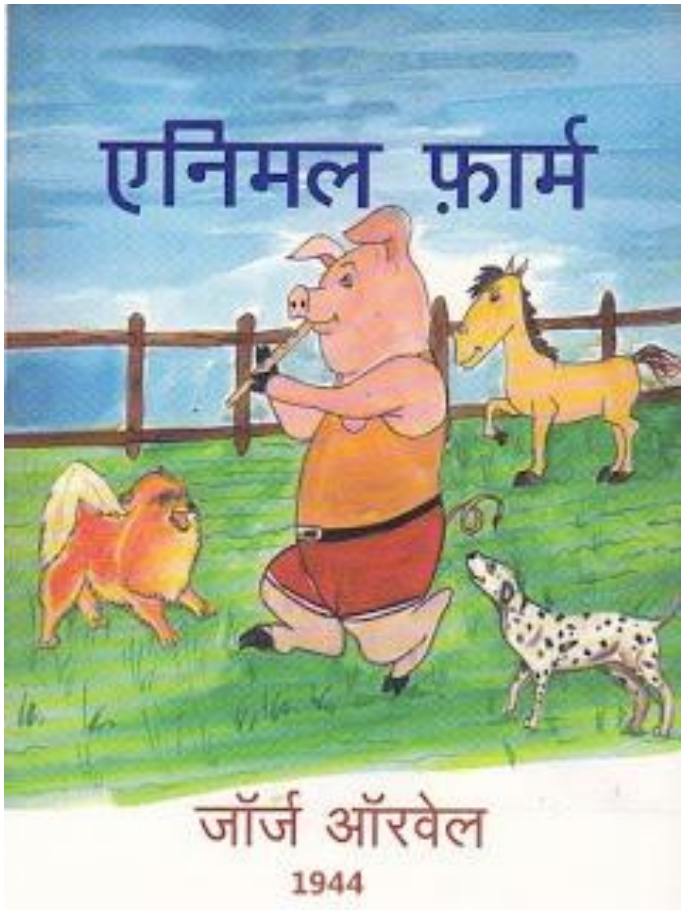



*Compare*
*'Car' and 'Car type'*

# Reality can be more complicated…

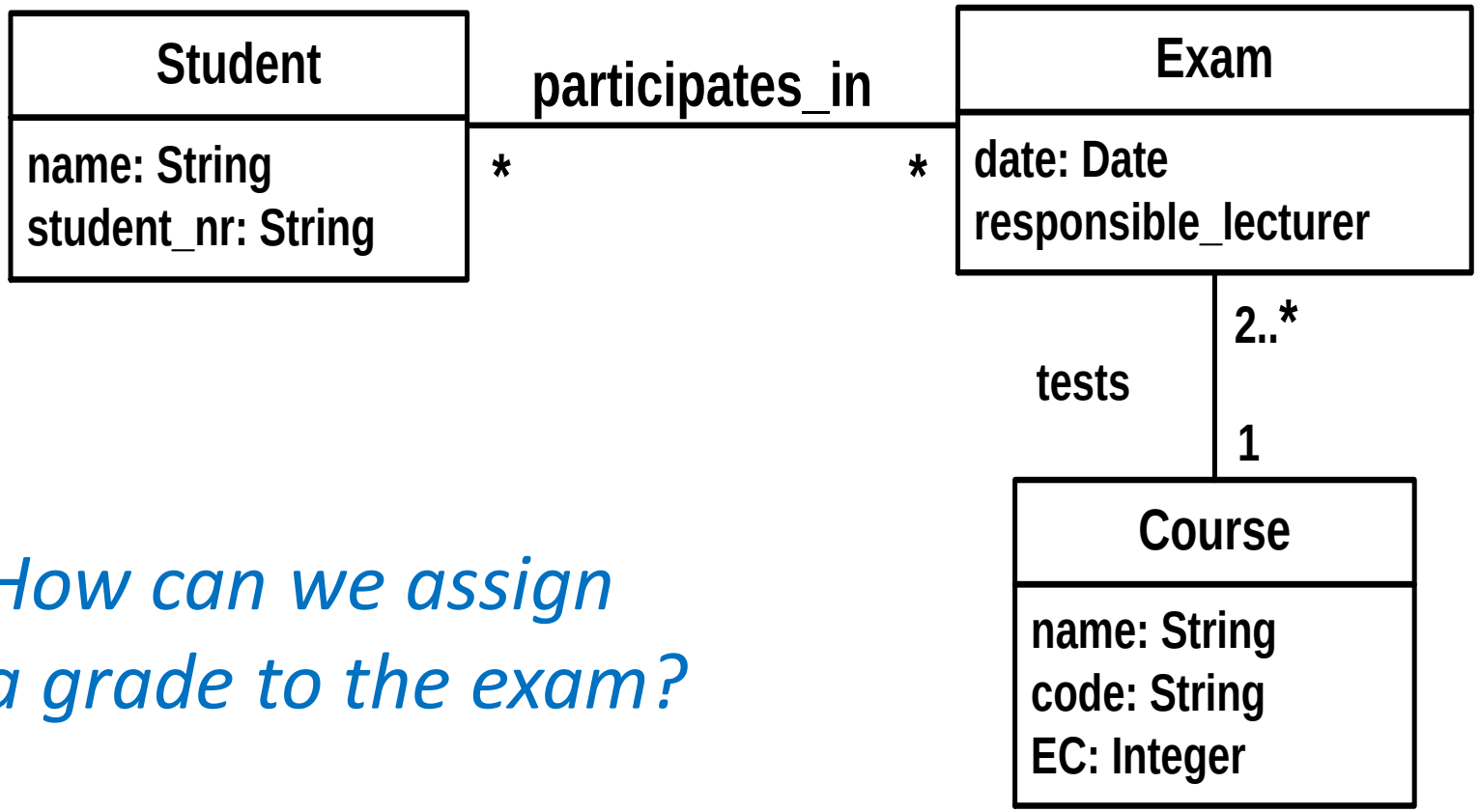
## *Are these the same book (title)?*



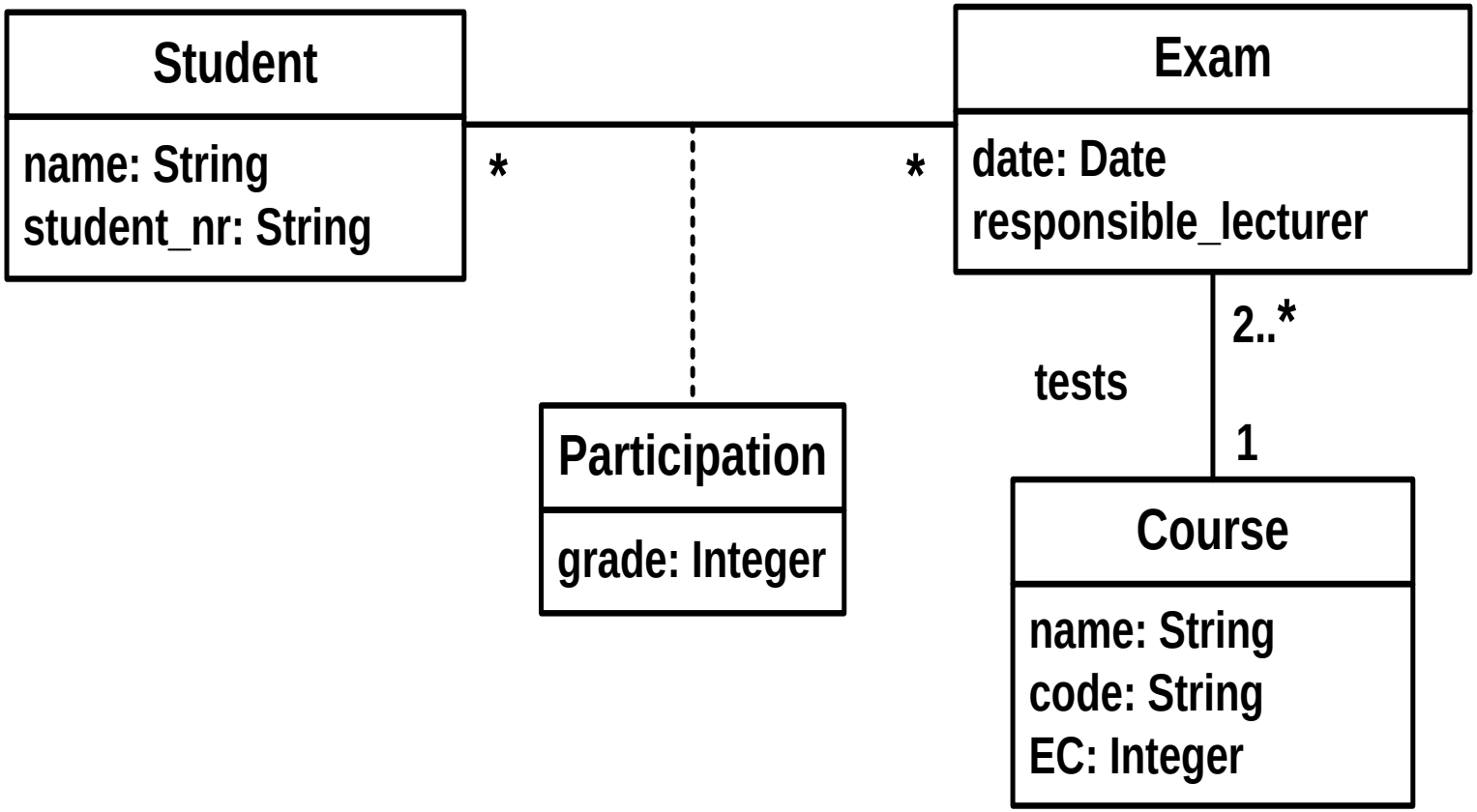
# 4.1 Association classes

# Problem




*How can we assign a grade to the exam?*

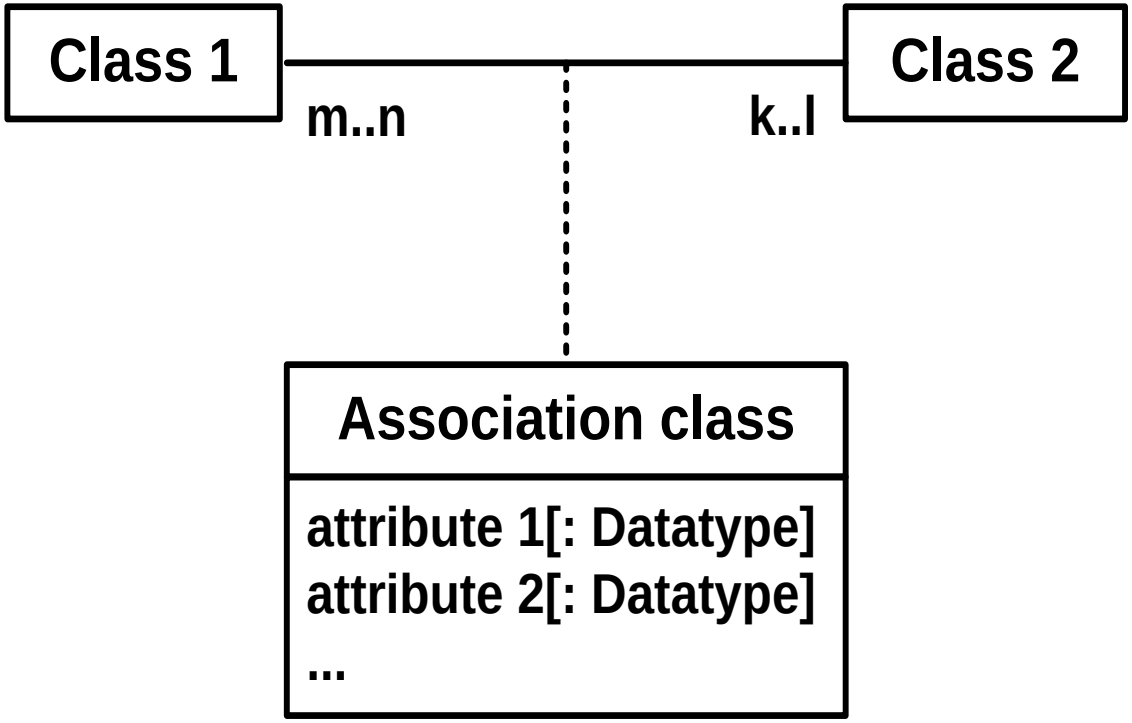
# Association class: Example



# Identity of an association class

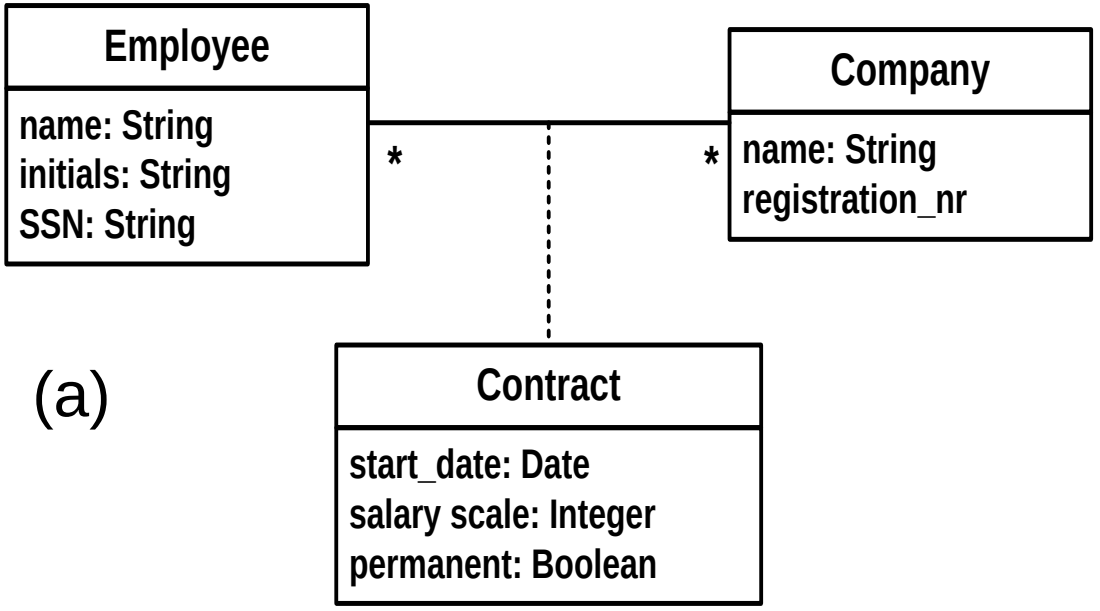
- The identity of an object of an association class is defined as combination of the identity of the associated objects.
- That means:
  - One ‘participation’ is identified by the combination of one student and one exam

# Elements of a CD (4): Association class



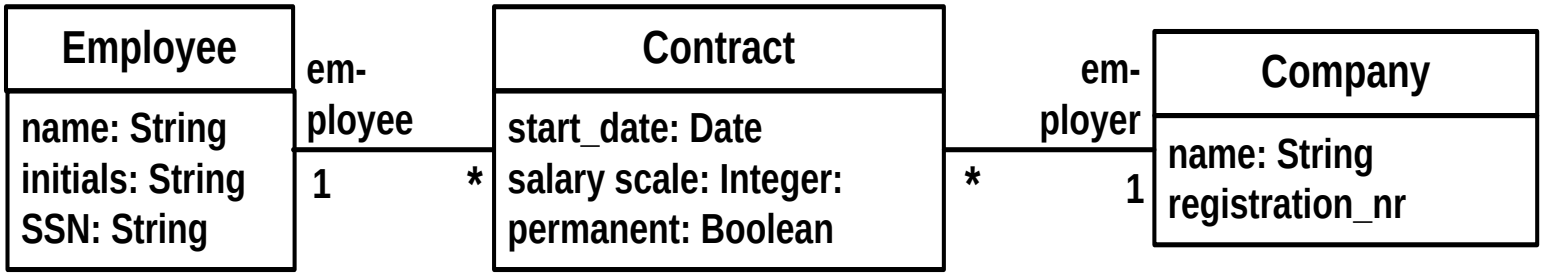
# Association class vs. class

What is the difference between these models?




(a)

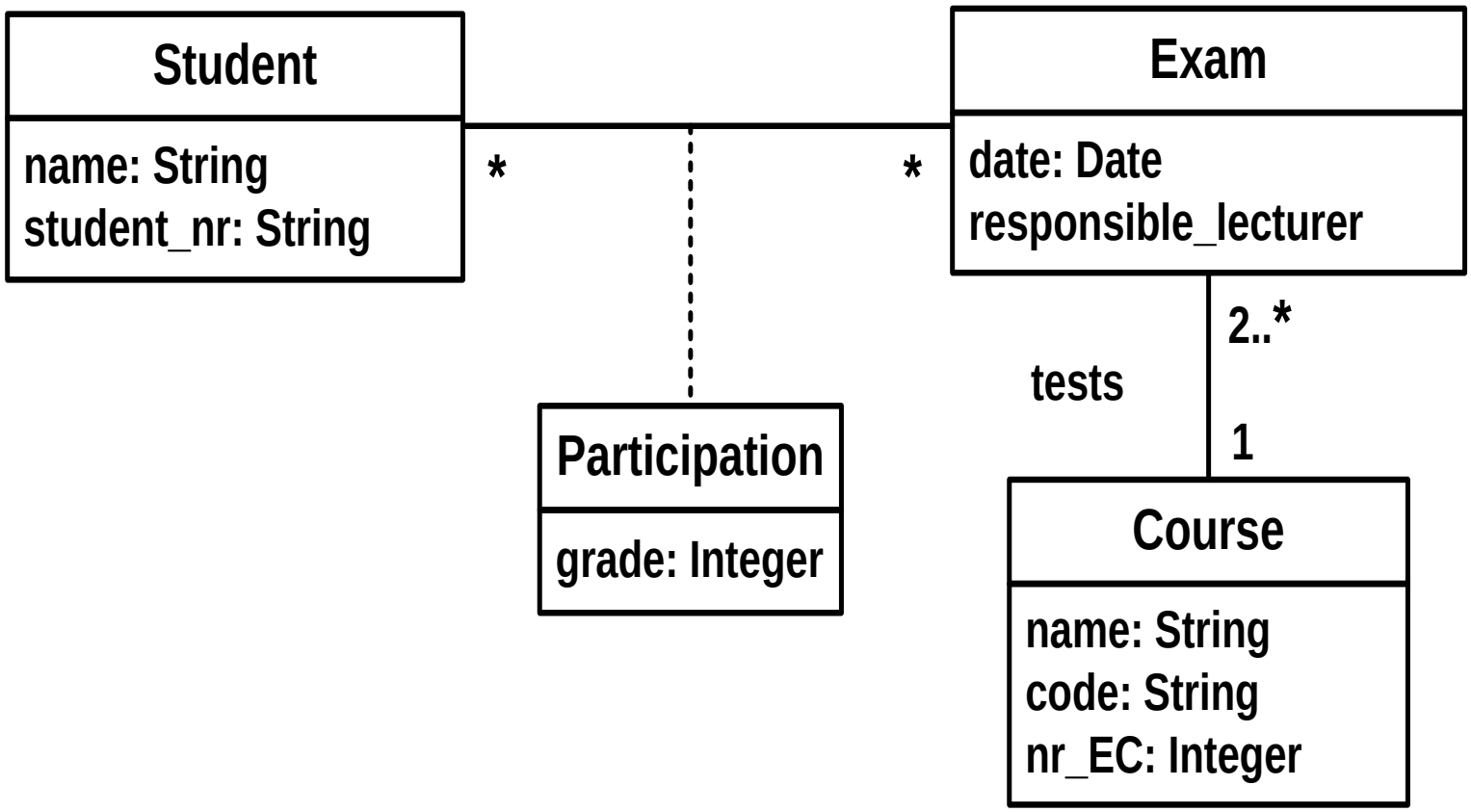
(b)



# Identity of an association class

*Identity is combination of the identities of the associated classes*

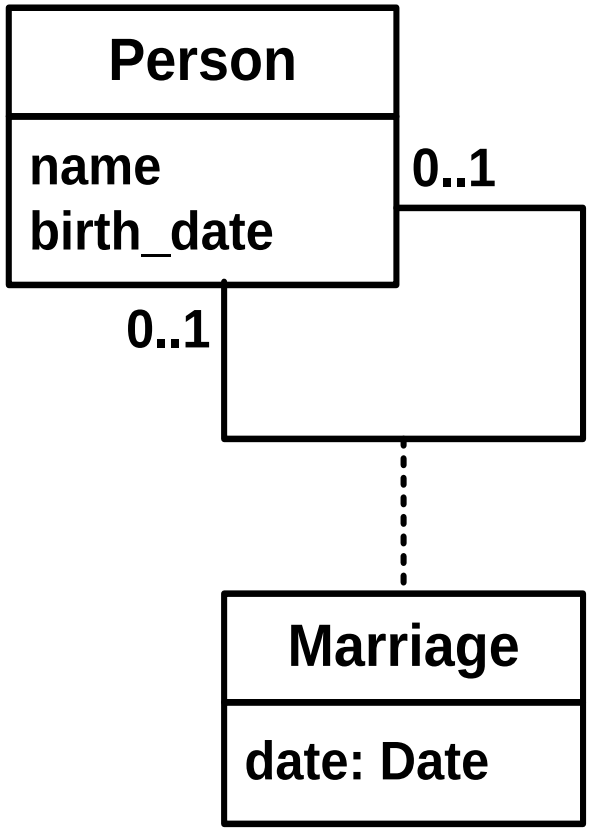
- That implies, for (a):
  - A contract is defined by a unique combination of employee and company
  - Differently stated: For a combination of one person and one company there can only be a single contract
- But in case (b):
  - A person and a company can have multiple contracts



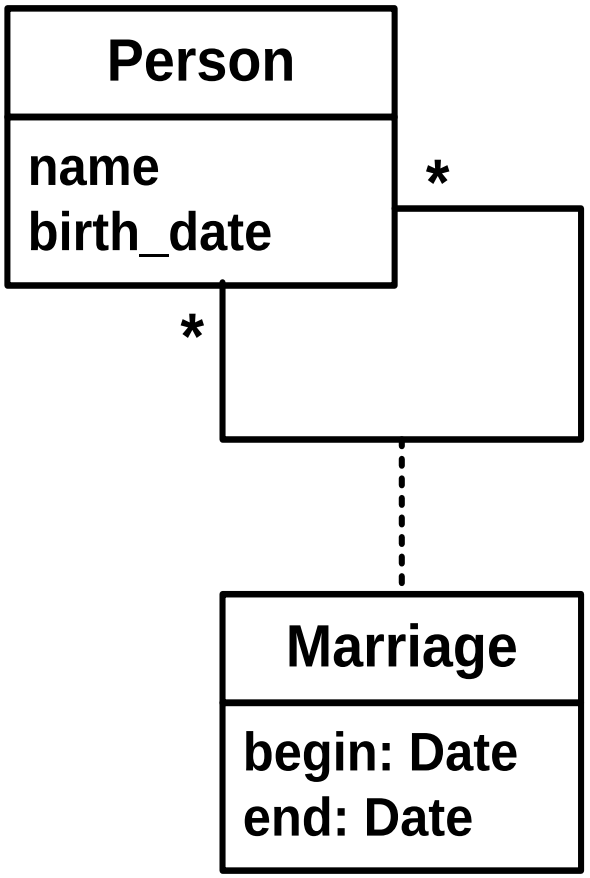

For a student who participates in an exam there is exactly one grade

# A last example

*Marriage: snapshot view*




*Marriage: historical view*

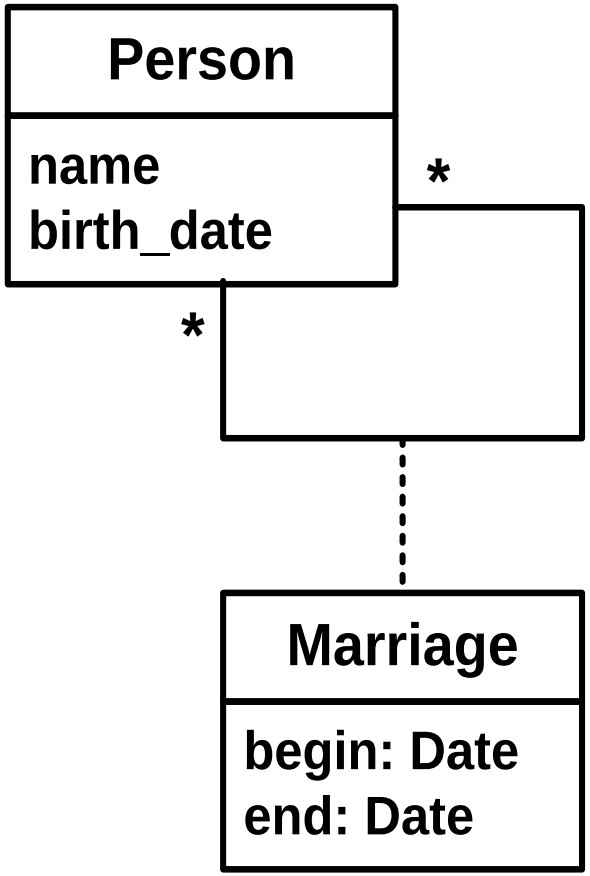


# Special case: Liz Taylor & Richard Burton

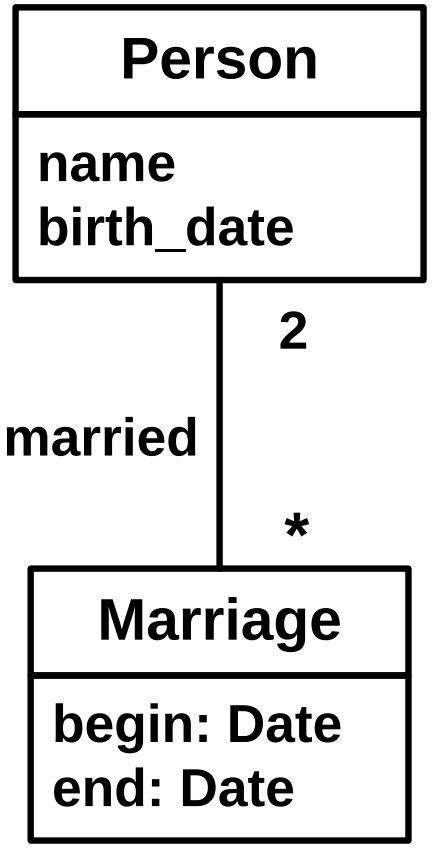
Married:  15 March 1964 — Divorced 26 June 1974

Married: 10 October 1975 — Divorced:  29 July 1976

# Liz Taylor & Richard Burton



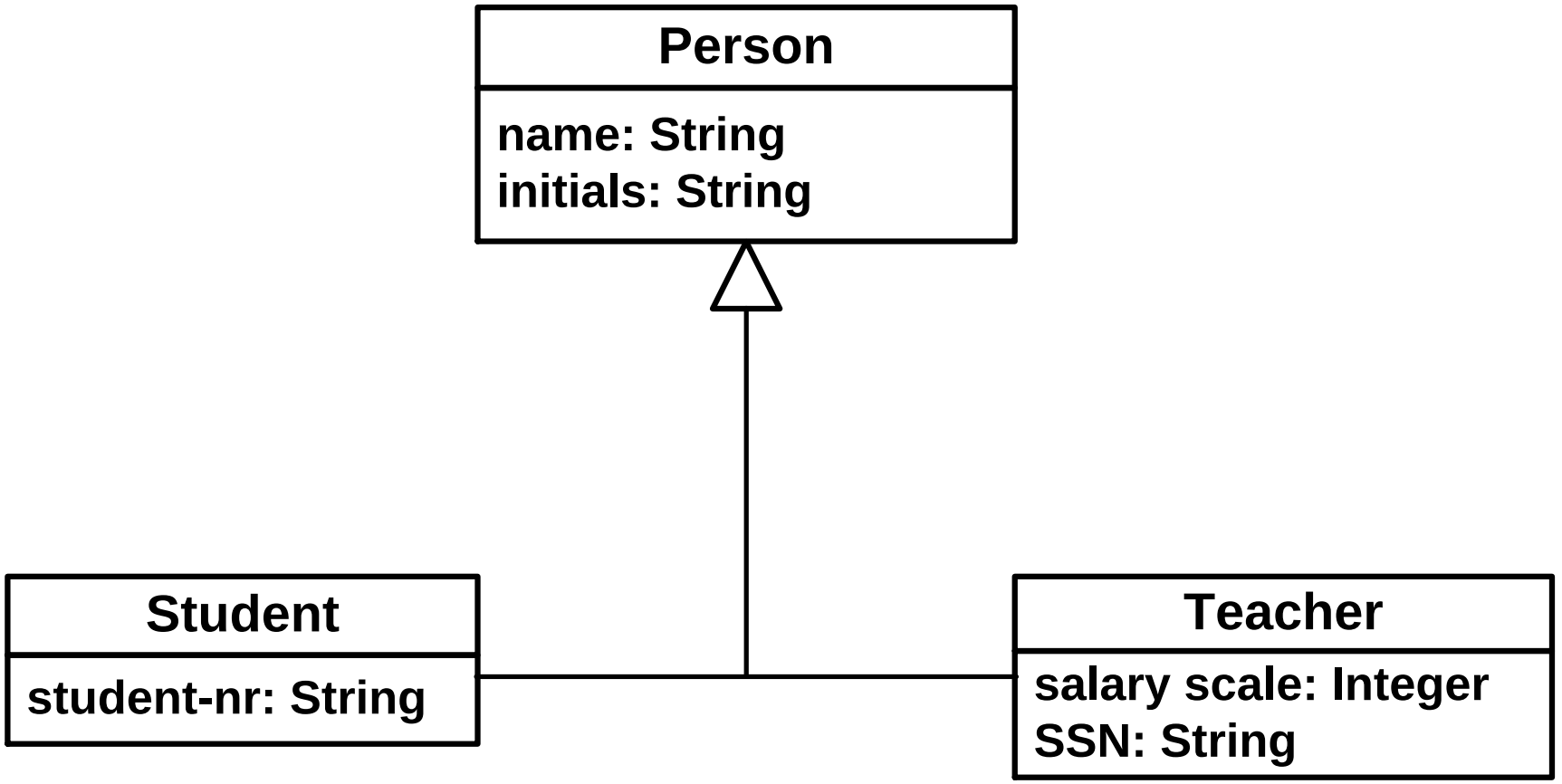

*Inadequate model*




*Adequate model*

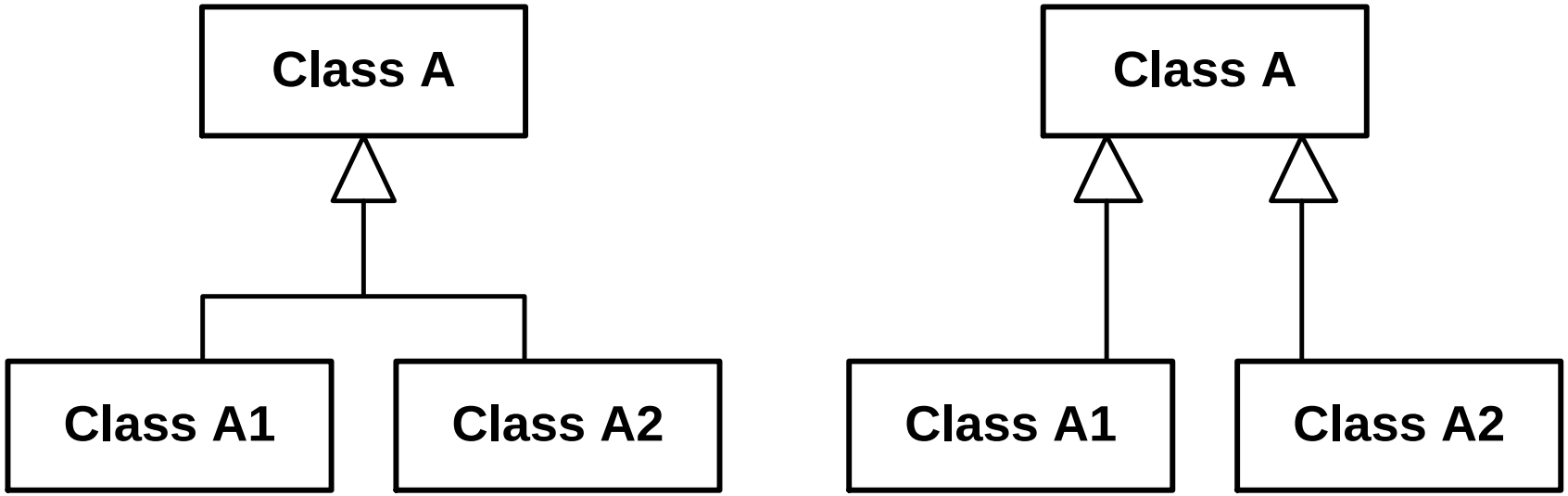
# 4.2 Generalization
## (also called inheritance(Java))

# Generalization example



# Elements of a CD (5): generalization

*Two equivalent notations*




*Class A is the superclass of Classes A1 and A2*

*A1 and A2 are subclasses of A*

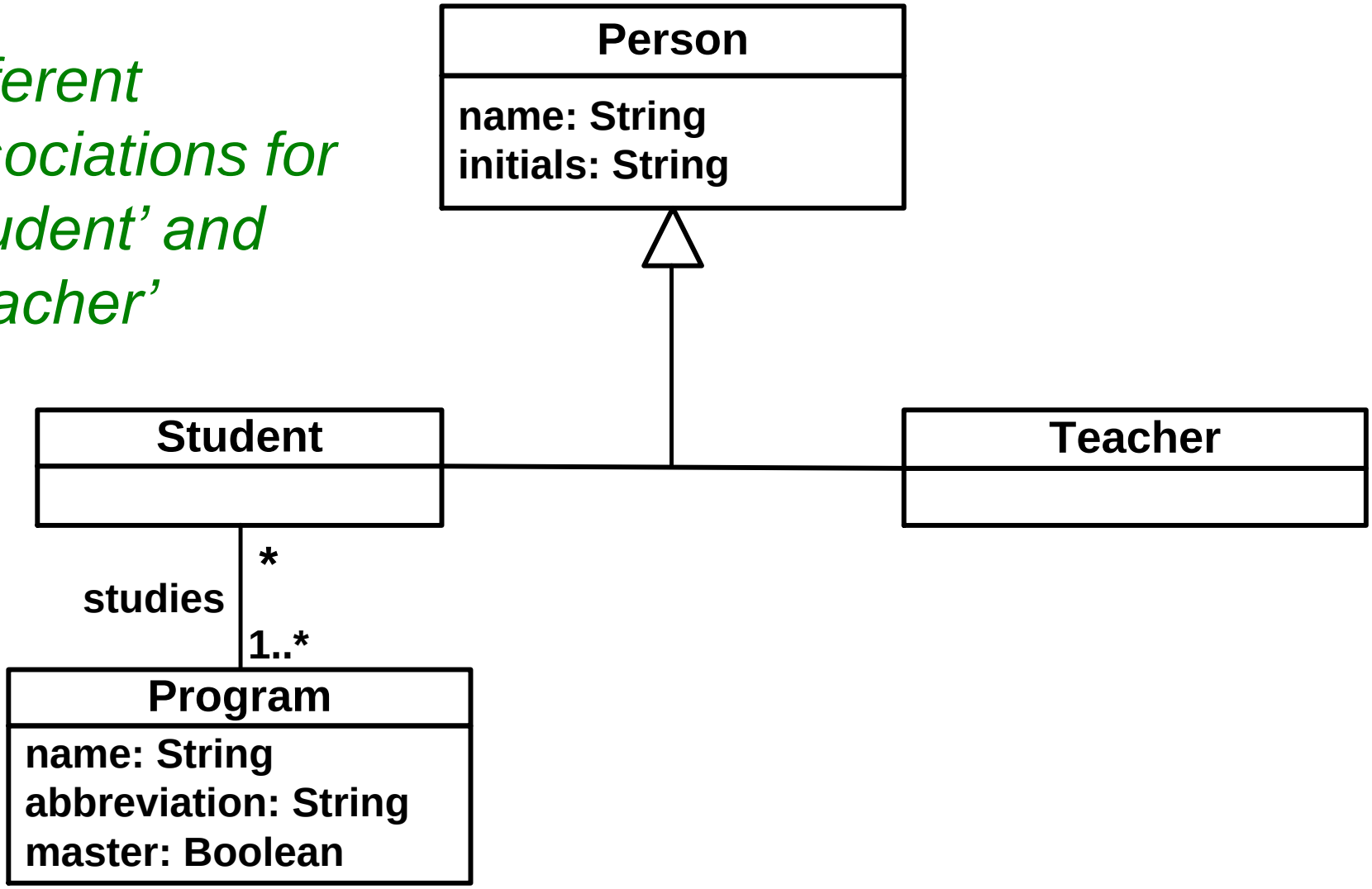
# Use of generalization

We use a generalization if …

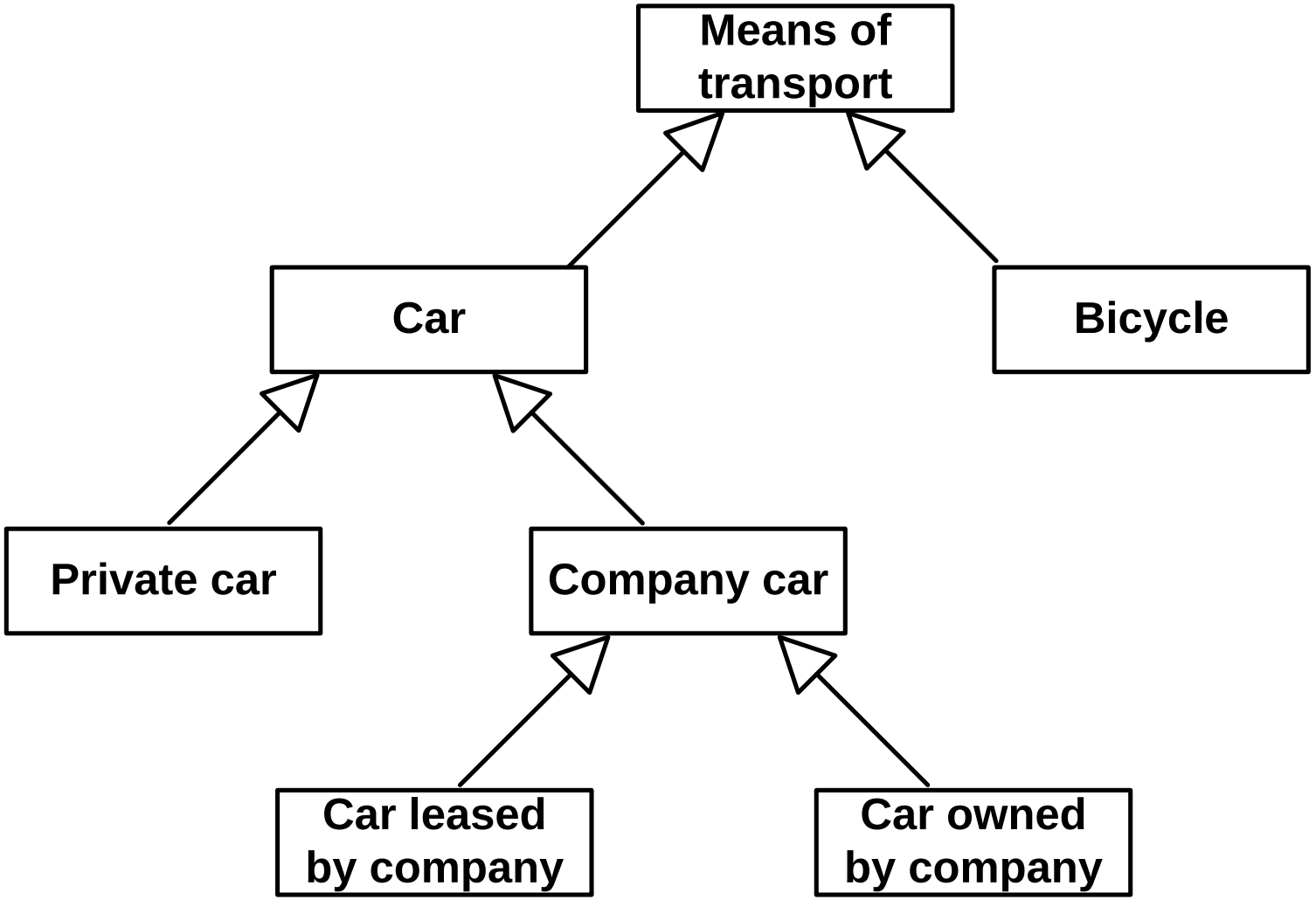
- …subclasses can be distinguished that have different attributes
- … subclasses can be distinguished that have different associations

# Use of Generalization

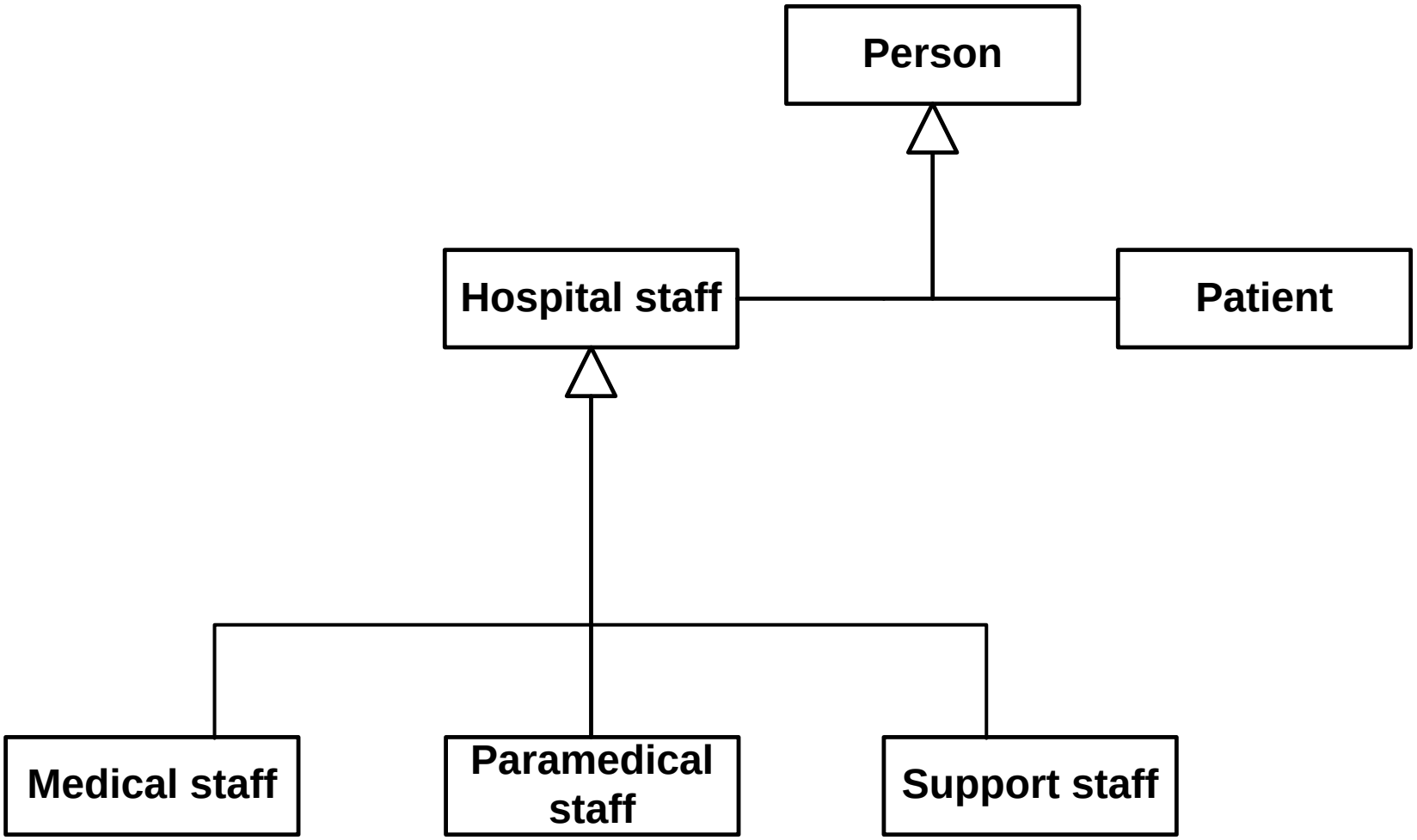
*Different associations for 'Student' and 'Teacher'*



# Generalization: more examples



# Generalization: more examples



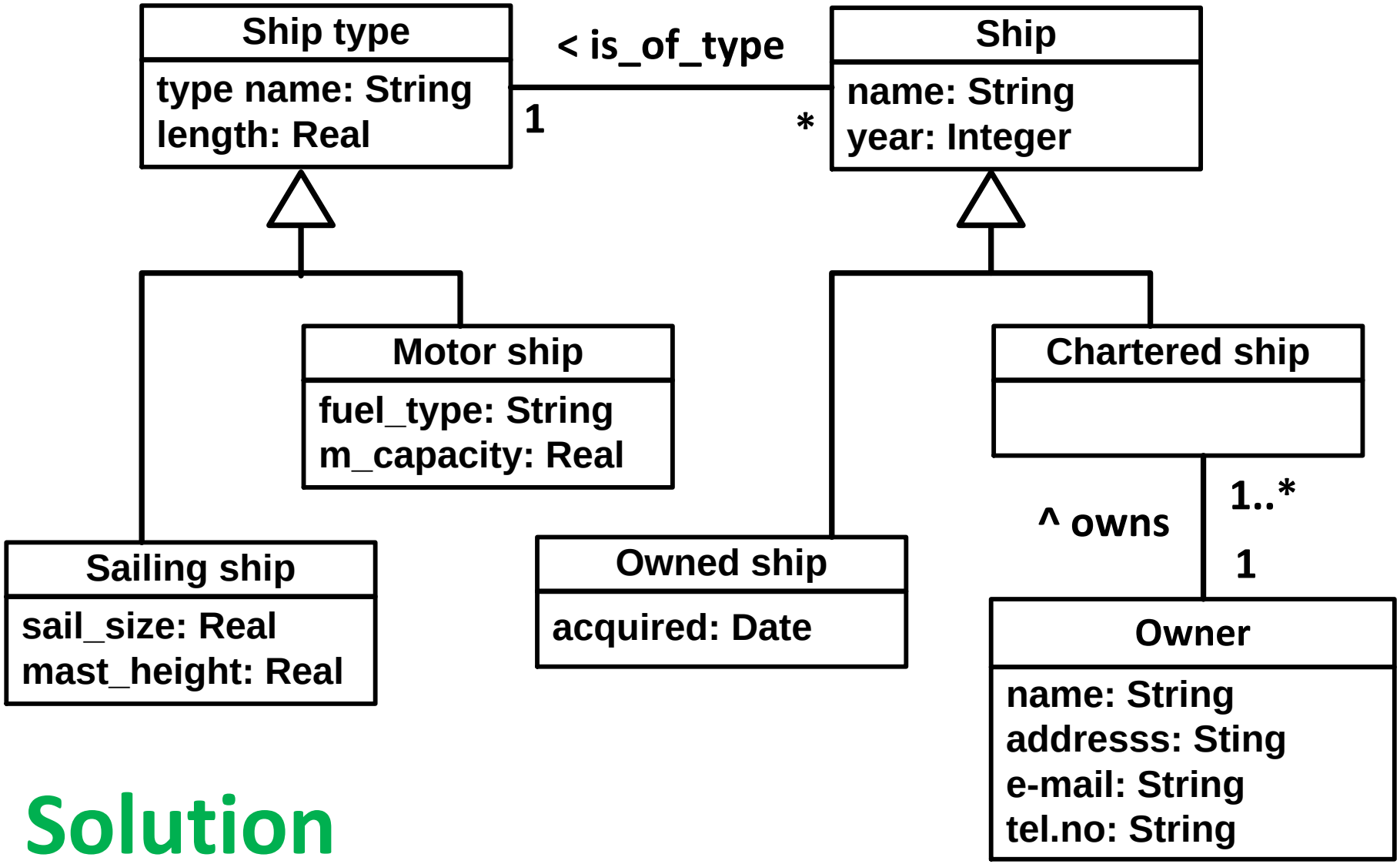
# A class diagram for '**Yacht Rental**'

- YR started by renting ships of private owners, who wanted to make some money when they didn't use their ship themselves
  (i.e. the ships were chartered by YR).
- Some owners later sold their ships, which are now owned by YR.
- For rent are a variety of sailing ships and motor ships (no human-powered ships).

# A class diagram for '**Yacht Rental**'

*Please include the following information:*

- **Each ship has a name, type of ship, length, and year of construction. Ships of the same type have the same length.**
- **For each type of sailing ship the sail size and mast height are given; for each type of motor ship the fuel type and motor capacity are given.**
- **Some ships are owned by YR. For these the date of acquisition is known. Other ships are chartered. For these the owner (with name, address, tel.no, e-mail) is known.**


Ship type
type name: String
length: Real
< is_of_type
1
*
Ship
name: String
year: Integer
Motor ship
fuel_type: String
m_capacity: Real
Chartered ship
Sailing ship
sail_size: Real
mast_height: Real
Owned ship
acquired: Date
^ owns
1..*
1
Owner
name: String
addresss: Sting
e-mail: String
tel.no: String

# Solution

# Lab session Wed 1+2

- If you have pending exercises to sign off
  - Small repairs: do them immediately
  - Large repairs: do them at home
    ***(it is OK to sign off Friday)***
    - **General rule: signed off one week later**
      **= two lab sessions later**

- It is important that you get working on class diagrams now – you may need help with that

# Design project

- The description in the manual is for a group of 4 persons