Object Oriented System Development

Lecture 2

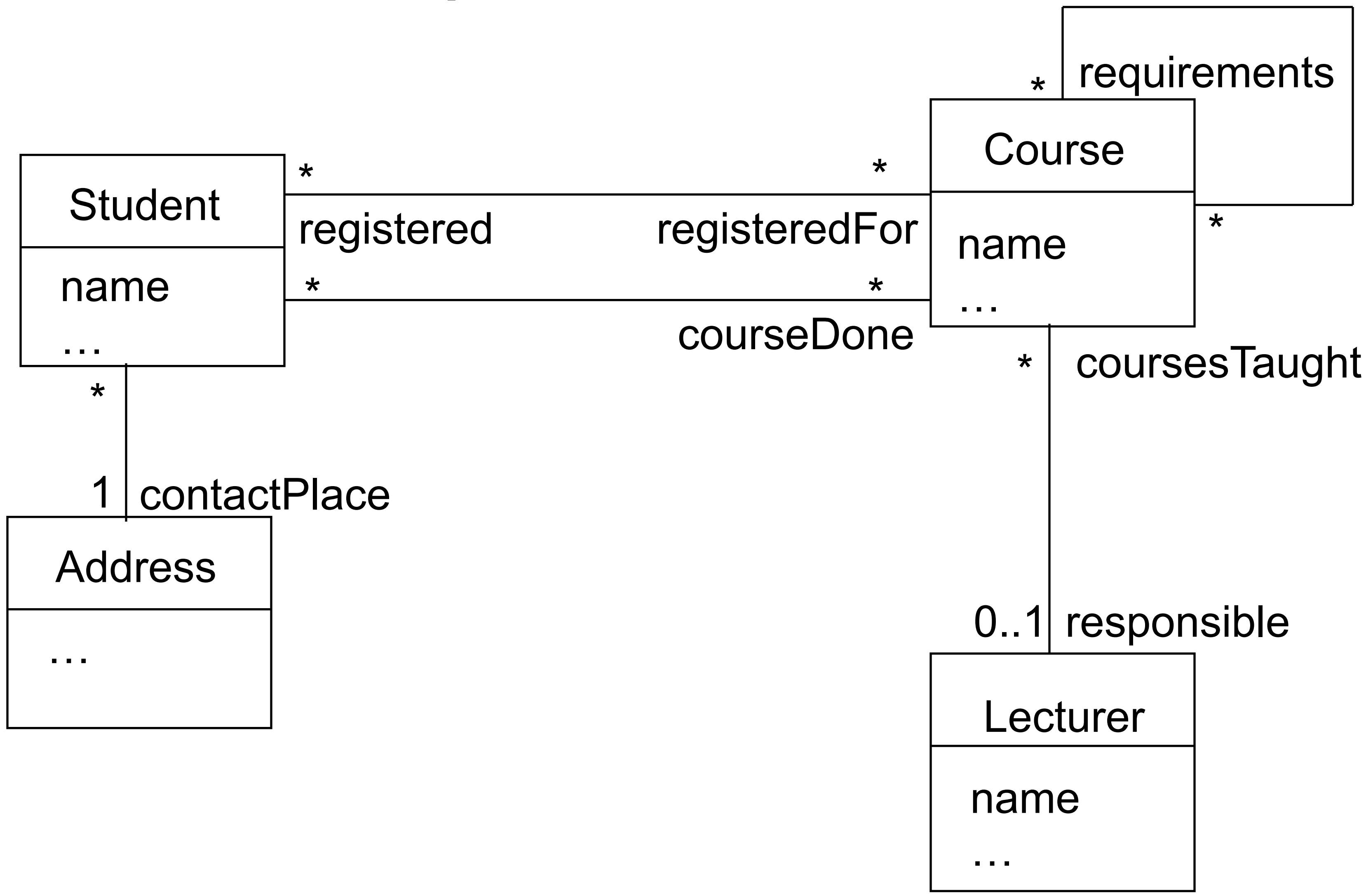
Domain model (Analysis)

Rogardt Heldal

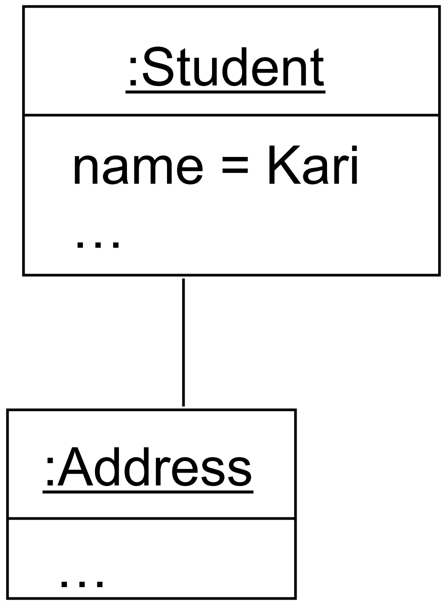
Objectives

- Write a domain model according to domain descriptions given by the customers.
- Be able to:
  - Identifying concepts
  - Finding associations
  - Finding attributes

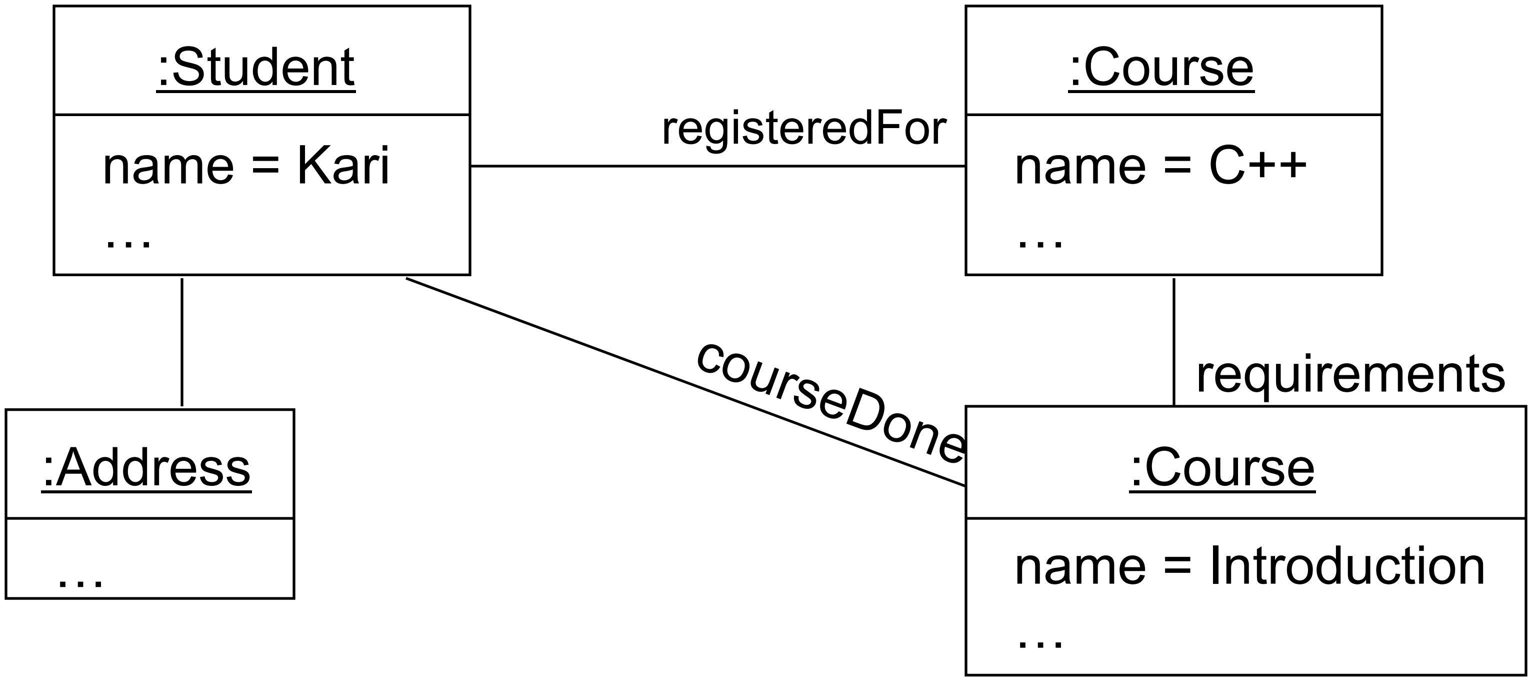
Example: Domain model



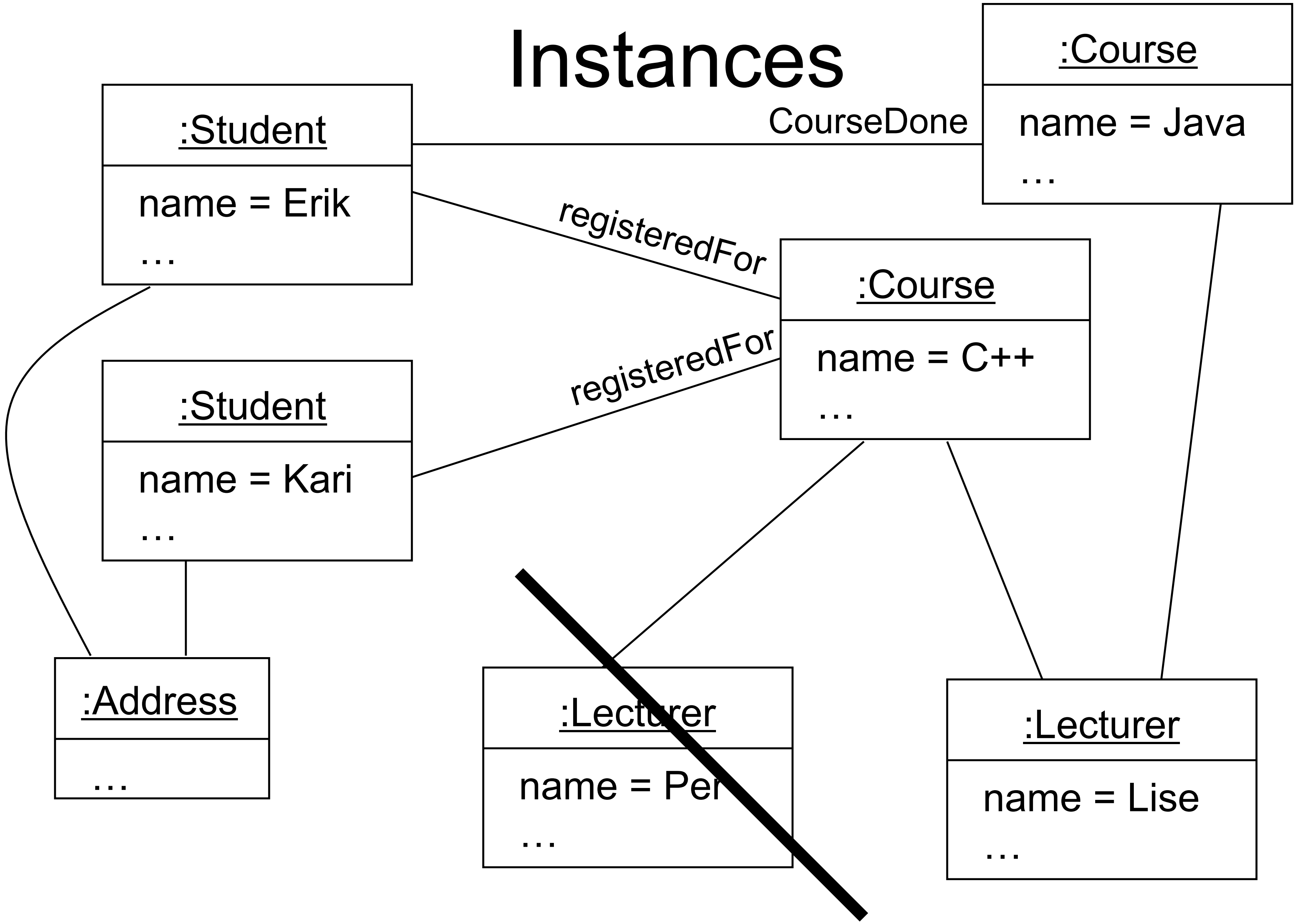
Instances



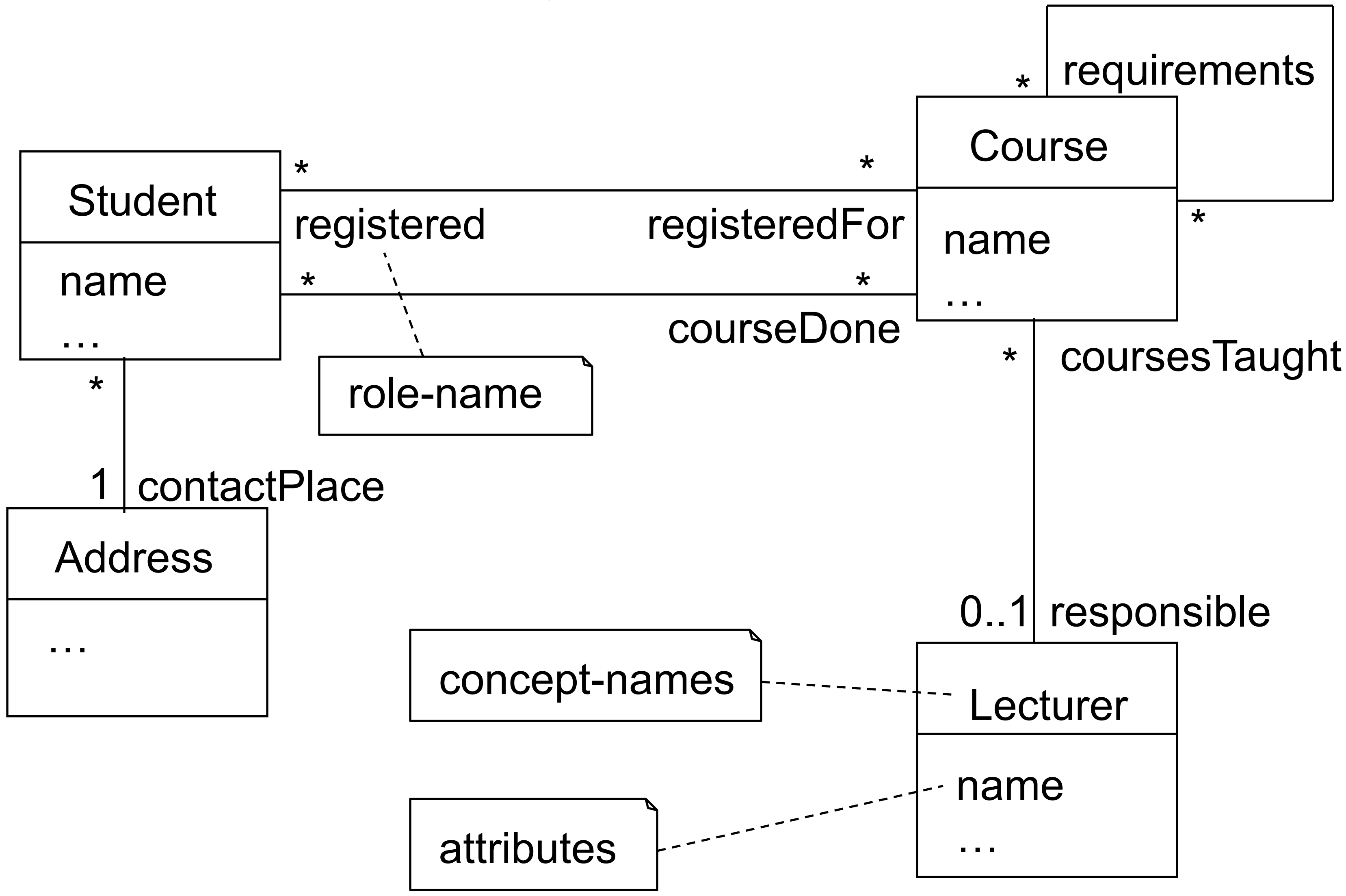
Instances


:Student
name = Kari
…
registeredFor
:Course
name = C++
…
courseDone
requirements
:Address
…
:Course
name = Introduction
…

Instances


:Student
name = Erik
…
:Student
name = Kari
…
:Address
…
:Course
name = Java
…
CourseDone
registeredFor
registeredFor
:Course
name = C++
…
:Lecturer
name = Per
…
:Lecturer
name = Lise
…

Vocabulary of the Customer



The Domain Model

Construct a model of the problem (*problem domain model*). I.e. not a model of the software.

– A collection of concepts and their interconnection.

Identifying concepts

Identifying concepts (1)

- Read through descriptions of the problem area, choose all nouns and determine which ones are suitable concepts. (Simple and popular method.)

- You can read e.g. :
  - set of requirements
  - use cases (appear later in this course)
  - nouns that emerge at interviews

Identifying concepts (2)

- You can also find concepts by looking at different concept categories:
  - physical objects, specifications, descriptions, places (geographical), transactions, objects of transaction, roles, containers, contents of containers, surrounding system, abstract concepts, organizations, events, processes, rules, directories, registrations, financial instruments, manuals, books etc

Example (1) – Cash Dispenser

- Example of concept categories:
- physical objects
  - cash dispenser
  - card reader
  - display
  - key pad
  - …

Example (2) – Cash Dispenser

- transactions
  - withdrawal
  - balance
  - transference
  - deposit

    …

- roles
  - customer
  - bank employee
  - service person

Group

- Preferably be 5-7 persons.
- Contain a mixture of persons, e. g. analysts, designers, programmers, domain expert.
- One person is the leader.

A process for identifying concepts (1)

The process:

1. each person is assigned to search for concepts in different places.
2. the whole group meets and finds concept candidates using brainstorming. All possible concepts should be includes! All concepts are equally important!

A process for finding concepts (2)

3. When no more concepts are found, you have a good starting point for identifying the suitable concepts for the problem area. Divide the concepts into three groups:
   1. Concepts which everybody thinks are important
   2. Concepts that fall outside the problem area, are more suitable as attributes, concepts with the same meaning etc.
   3. Concepts which fall in between these groups.
4. Discuss the concepts in group 3 which not everyone agrees on. These concepts should be treated consistently.

Principles for brainstorming

- All ideas are equally important
- Think quickly, not too much consideration; think more later
- Let everyone speak (Round-Robin Technique)

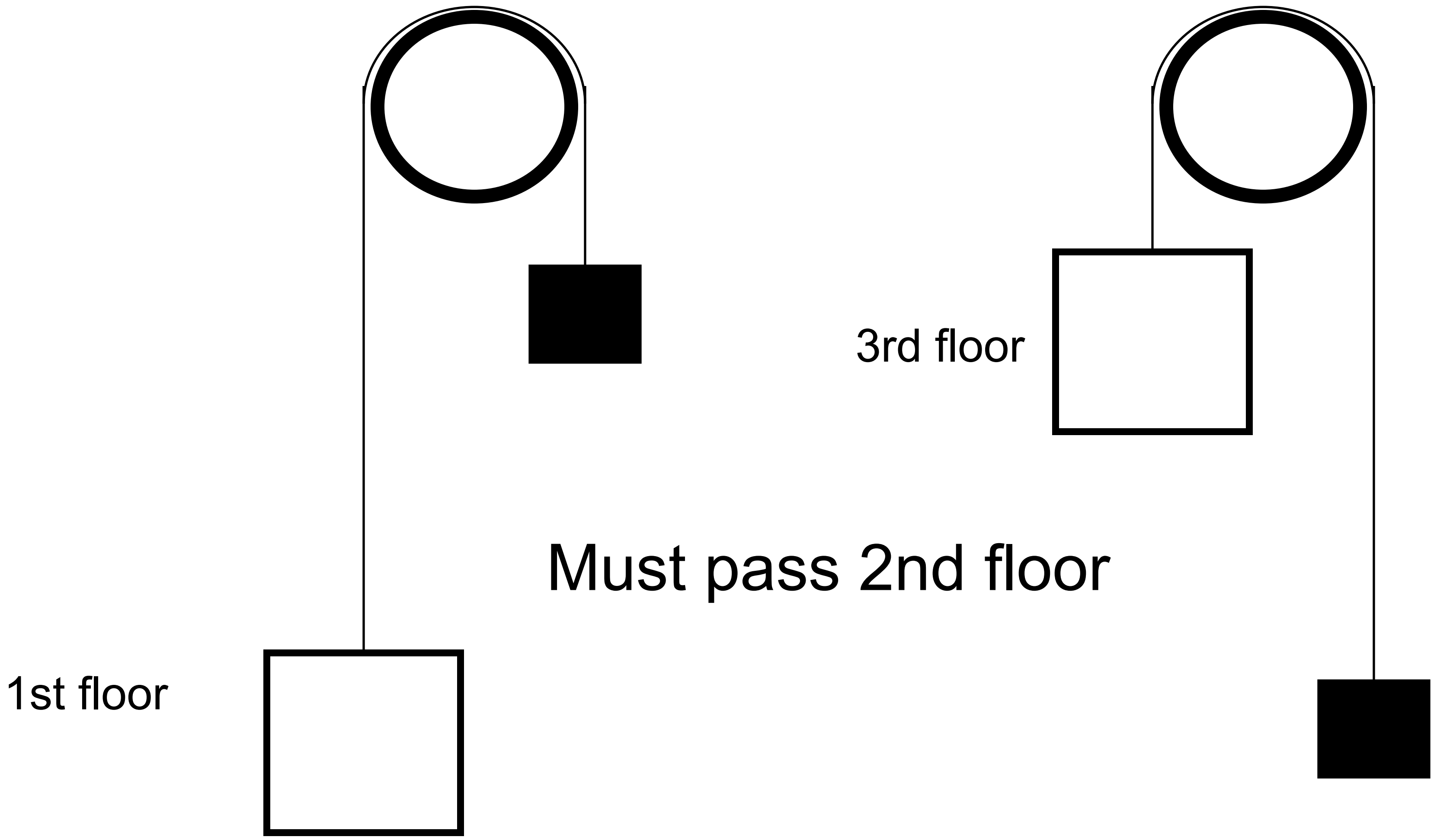
Good concepts

- It is important to point out that you probably haven’t found all appropriate concepts after looking for them the first time, if the problem area isn’t very trivial!
- The more you work with the domain model, the better it gets. It can take several iterations to reach a stable model.

Limitations

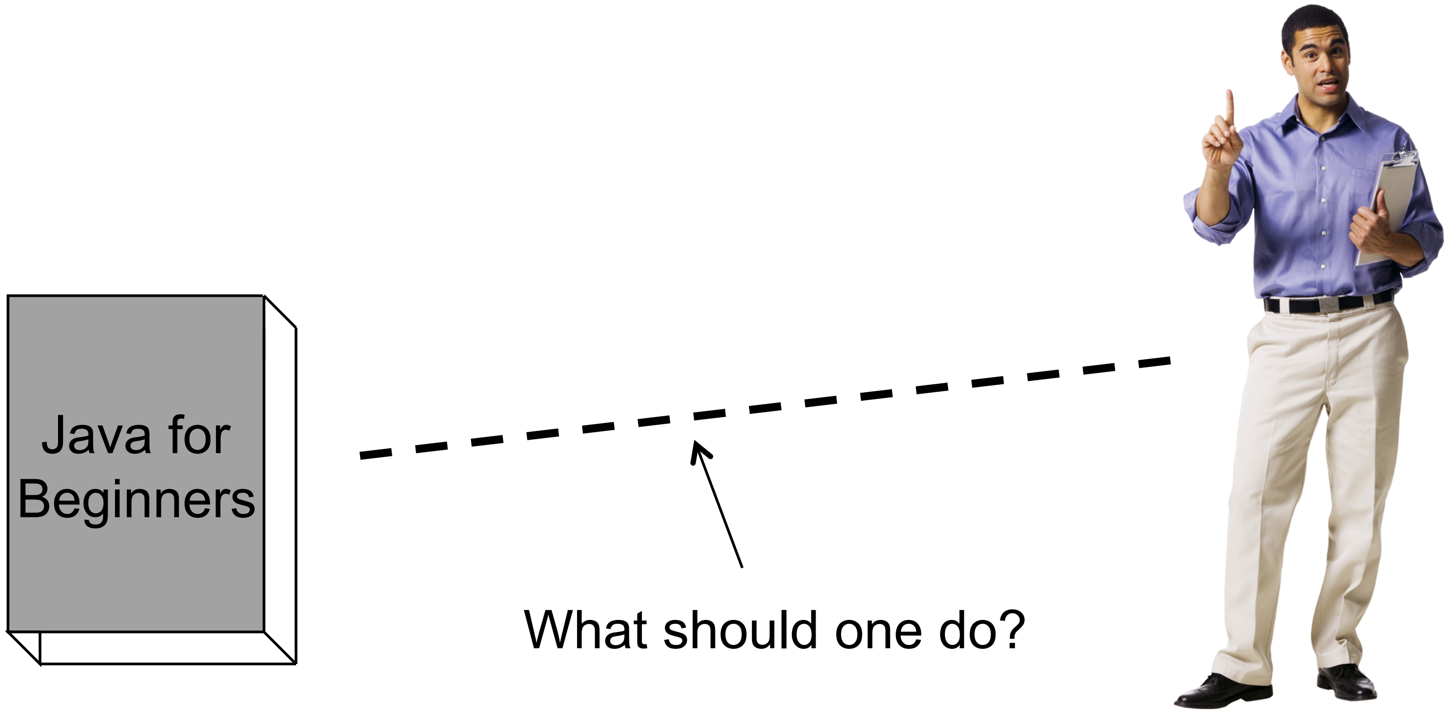
- It’s very important to describe the limitations that the concepts have in the real world. We will show two examples:
  - Controlling system for elevator
  - Library system

Elevator



Must pass 2nd floor

Library



A person can move, die etc

Responsibility

- **Responsibility**: Make a list of responsibilities that the concept should have.
- If a concept has no responsibilities there is something wrong, for example:
  - Car is a perfect concept for some domains, but not for a course registration system.
- Too many responsibilities
  - There might be further decomposition of the concept.

Problem

- Some companies want a common system for lending vehicles to employees. This is one of many attempts to make it more attractive to work for the companies. They want you to make a domain model which will be discussed in the next meeting. They already have a short description of what they want from the system. Here are some possible uses of the system: a person does not have a car, but needs a car to shop, a minibus or bikes for a weekend trip with colleagues/ friends, or is flying to another part of Sweden and needs a car (and a company which is part of the deal is located there).

Domain description:

- Every company which is part of the deal will have a number of vehicles: cars, buses, and bikes. These can be borrowed up to one week if one has worked in the company for more than one year (carefully consider how to represent employee start dates in the domain model). People who mistreat the car can be prevented from borrowing vehicles in the future. Furthermore, one needs a driver’s licence to borrow a car or a bus. Partners of employees can also borrow vehicles if the they are registered as partners. This booking system is meant to be flexible so that workers from other companies which are part of the deal can also borrow vehicles.

Domain description:

- Every company which is part of the deal will have a number of vehicles: cars, buses, and bikes. These can be borrowed up to one week if one has worked in the company for more than one year (carefully consider how to represent employee start dates in the domain model). People who mistreat the car can be prevented from borrowing vehicles in the future. Furthermore, one needs a driver's licence to borrow a car or a bus. Partners of employees can also borrow vehicles if the they are registered as partners. This booking system is meant to be flexible so that workers from other companies which are part of the deal can also borrow vehicles.

Possible Concepts

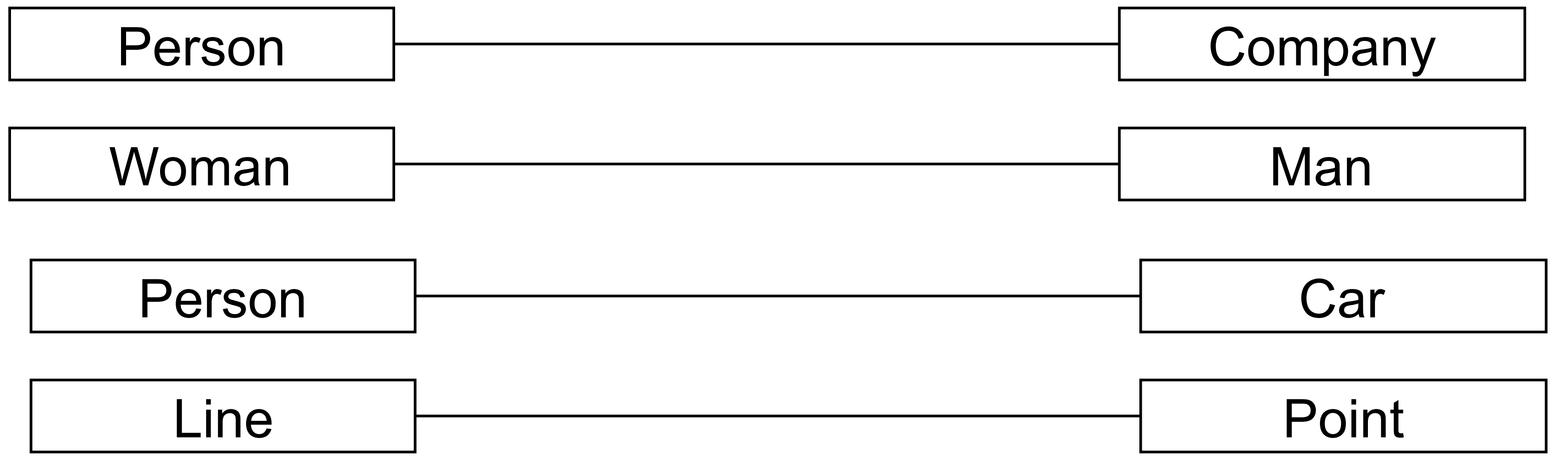
- Company
- Vehicle
- Car
- Bus
- Bike
- Employee
- Address
- Licence
- Person
- Complaint

Finding associations

Associations (know of-relation)

Concepts must know of each other in order to be able to communicate.

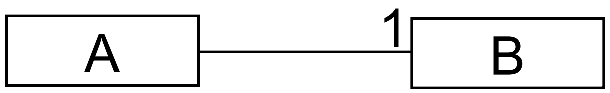
UML-notation:



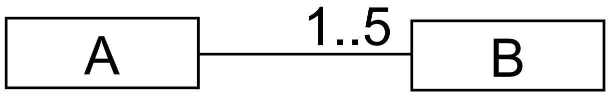
Multiplicity

We use multiplicity to show how many objects can be associated in a given relation.

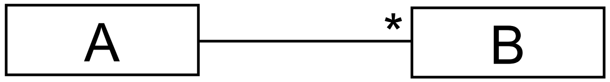
UML notation



Exactly one object of type B should be associated to each object of A



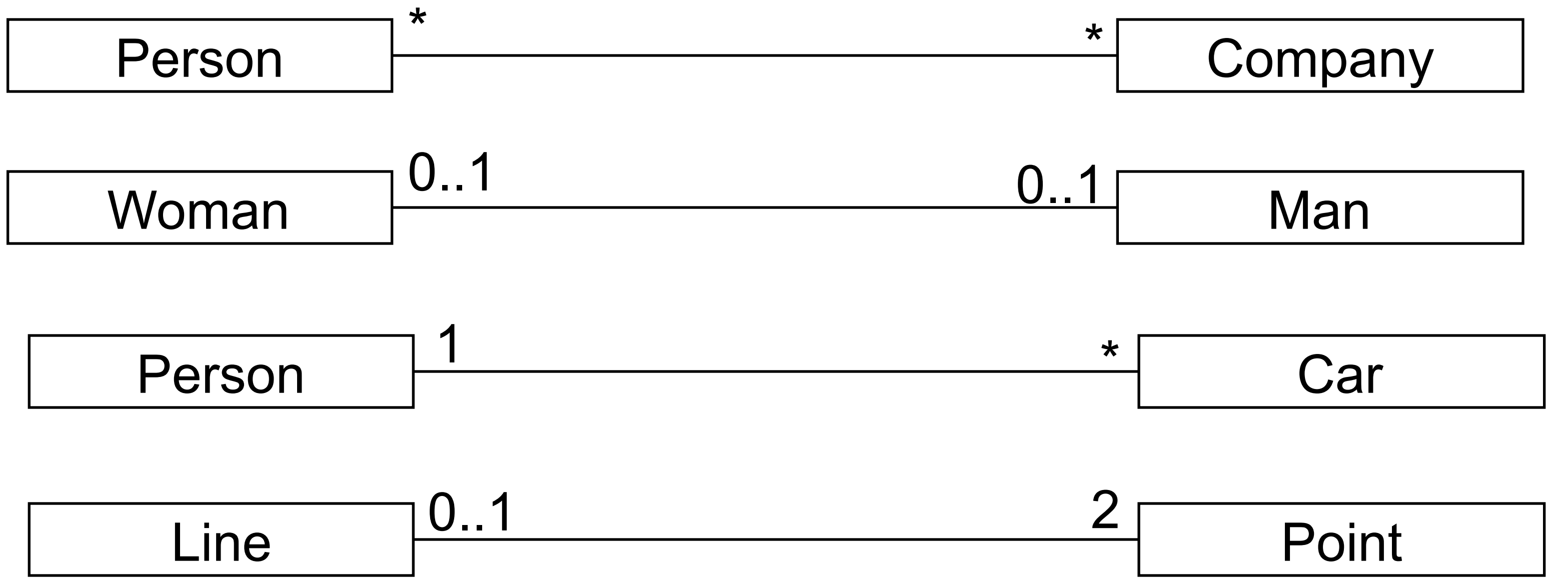
1,2,3,4 or 5 objects of type B should be associated with each object of A



0 or any other number of objects of type B can be associated to an object of A

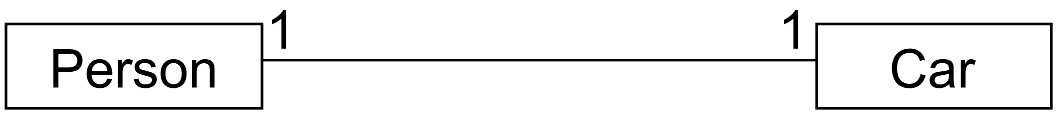
Example: Multiplicity

Give multiplicities for the associations below.



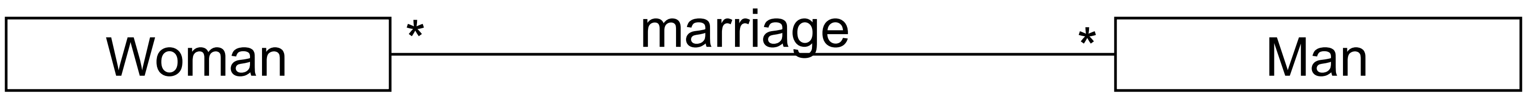
Too restricted

- Make sure unnecessary restrictions are not imposed, e.g. below we only allow a person to own one car and a car can only be owned by one person:



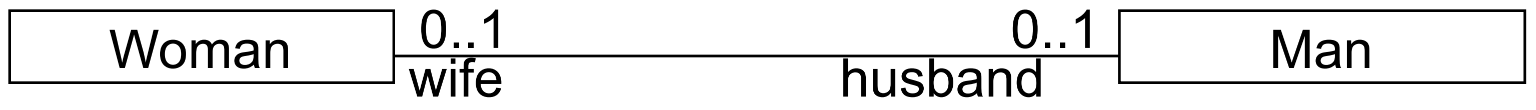
Too general

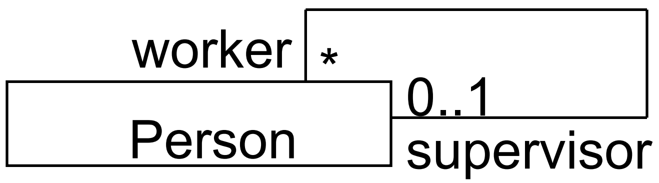
On the other hand you shouldn’t be too general either, e.g.

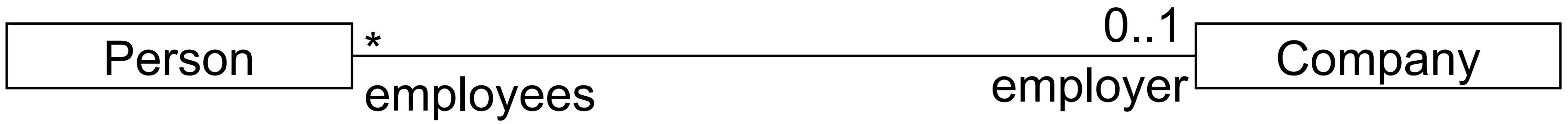


in Sweden you are only allowed to be married to at most one person at a time.

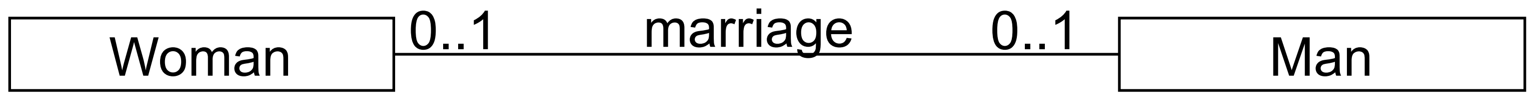
Role

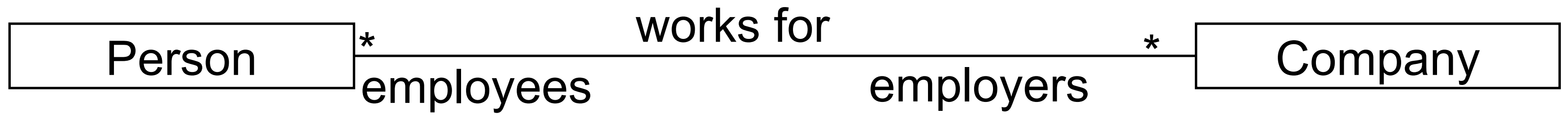

Woman
0..1
wife
0..1
husband
Man


worker
*
Person
0..1
supervisor

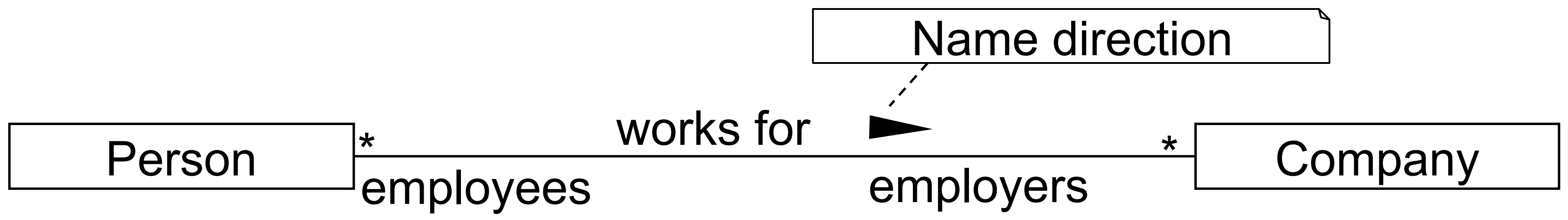

Person
*
employees
0..1
employer
Company

Association names

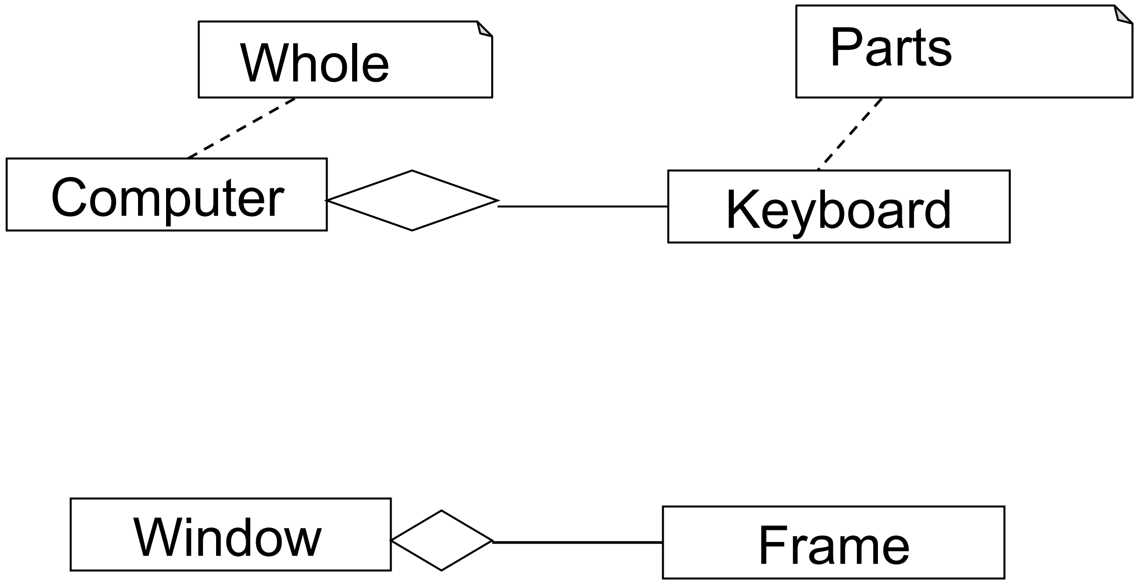




Direction of association names:



Aggregation and composition



Problem with aggregates

Aggregates creates certain problems. Most of us agree on that

- ”Frame” is a part of a ”Window”.

But is

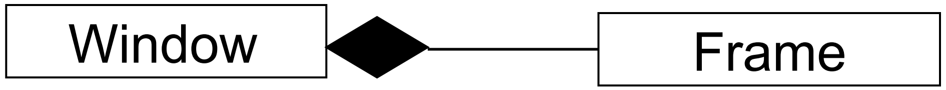
- a secretary a part of a company?

or is

- a parrot a part of a cage?

Here there is no common opinion. You should avoid the misuse of aggregates, but they can be powerful if you really want to show that something is part of a bigger entity.

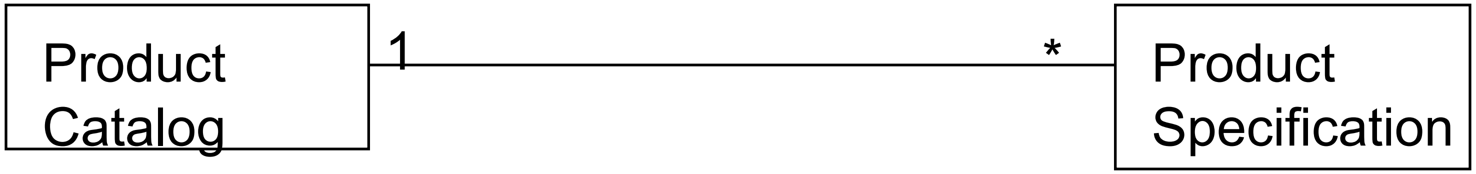
Composition



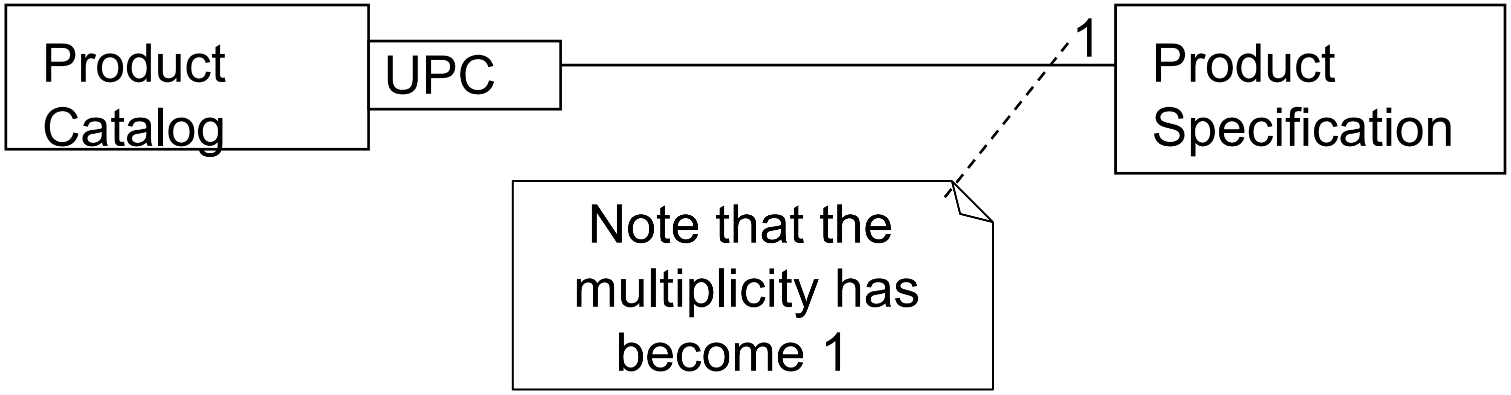
- Composition is a very strong relation. The part can only exist when the object it is part of exists. A part object can only be part of one composition.

Qualifiers

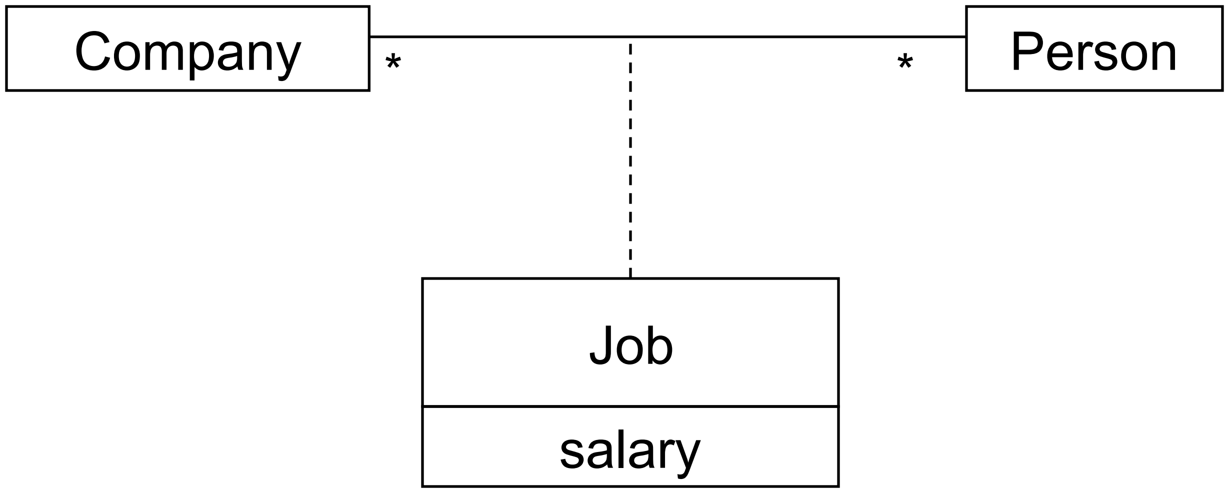
Given:

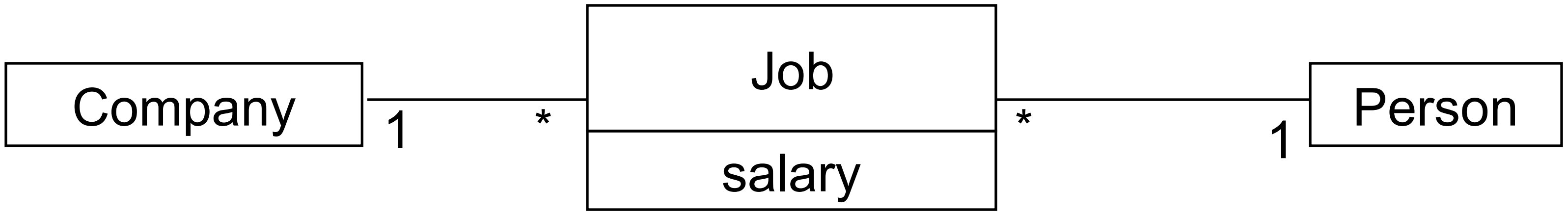


We would like to state that given a ”Universal product code” (UPC) and a ”Product Catalog” you can get a given ”Product Specification”. This can be shown in UML using qualifiers:



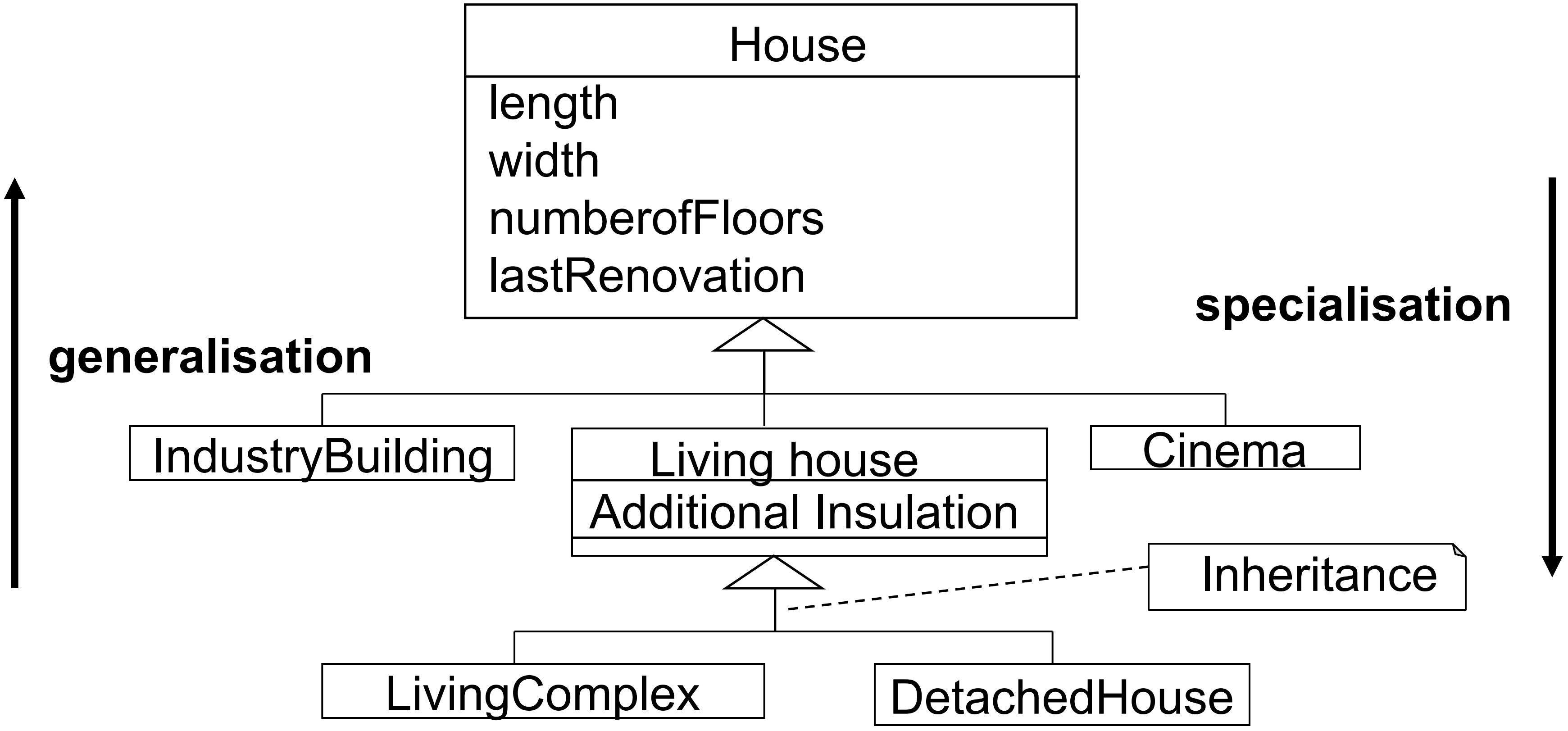
Association Class





Class-hierarchy

One can often find class hierarchies in the analysis.



Treatment of associations and inheritance

- To begin with you should only focus on finding
  - Associations between concepts
  - Inheritances
  - Multiplicities
  - Role names
- Again, brainstorming is good method.

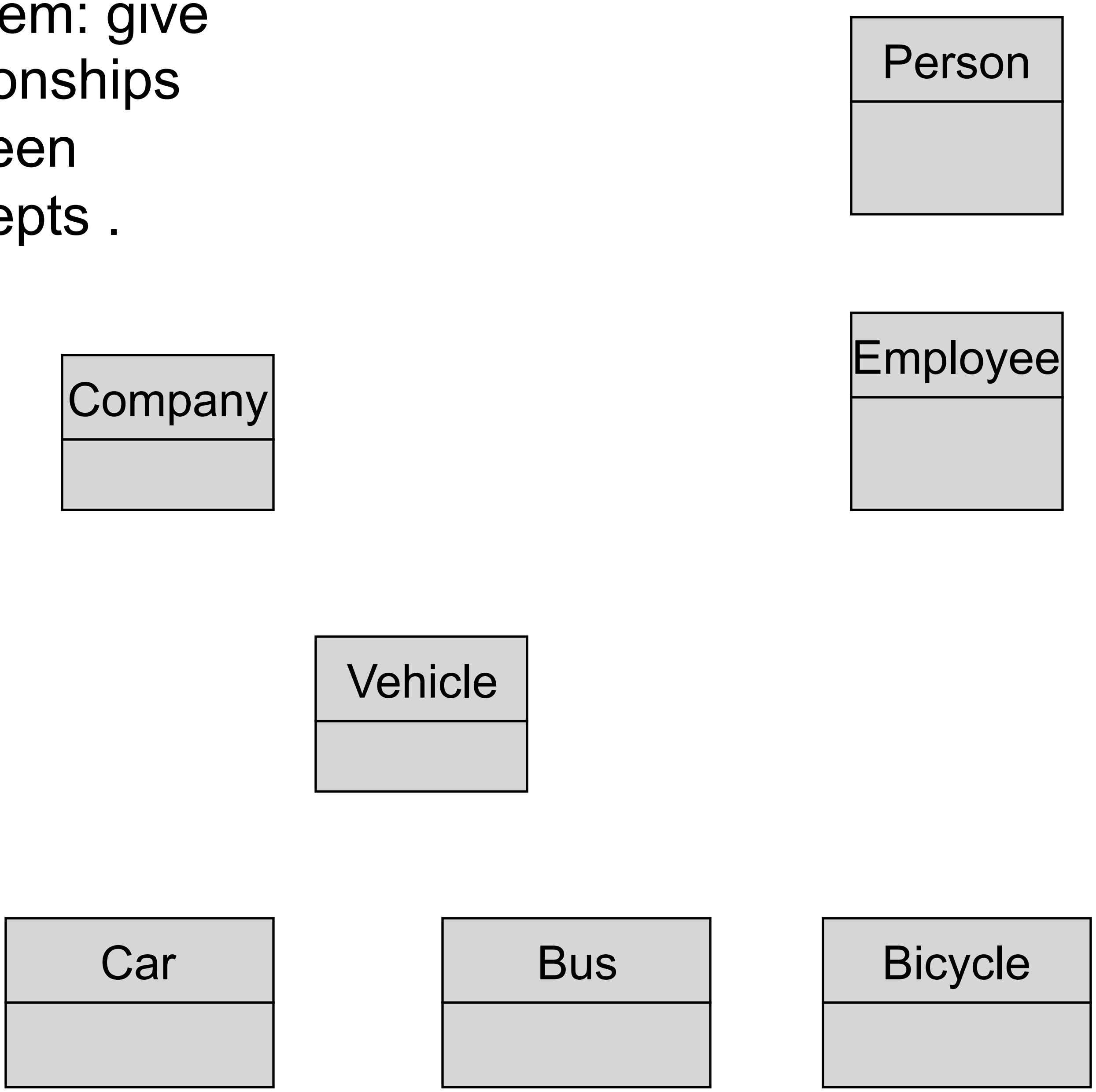
Refining associations

- Aggregates, composition, association names, qualifiers etc. should be used with the purpose to make the domain model more clear and readable. It’s important to choose good association and role names.

New concepts

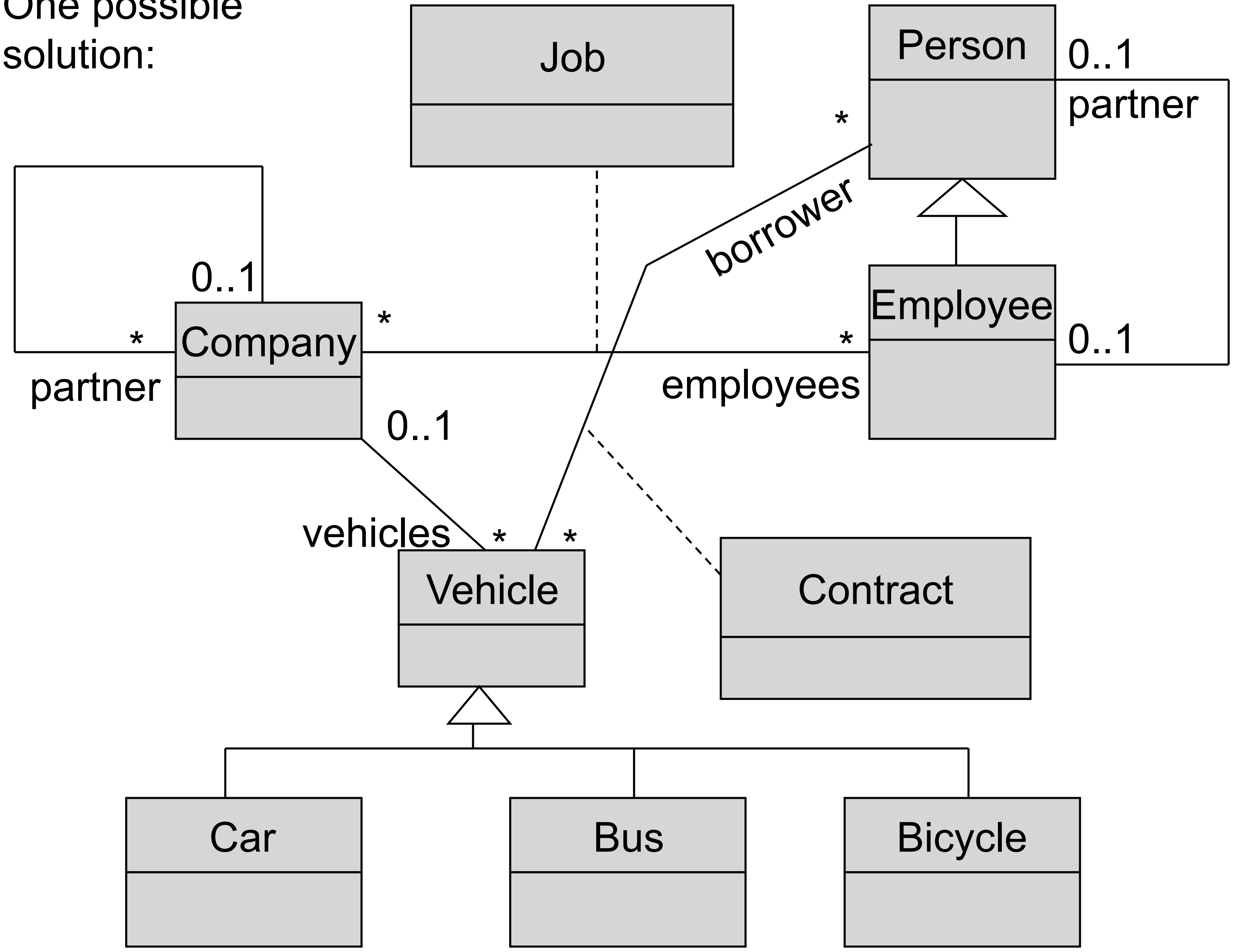
- You cannot expect to work with associations separately. By looking for associations you often find new concepts too.
- There is also a strong relation between association and attribute, which we will return to when we look at attributes.

Problem: give relationships between concepts .



Haven't added:
Complaint
License
Address
…

One possible solution:



Finding attributes

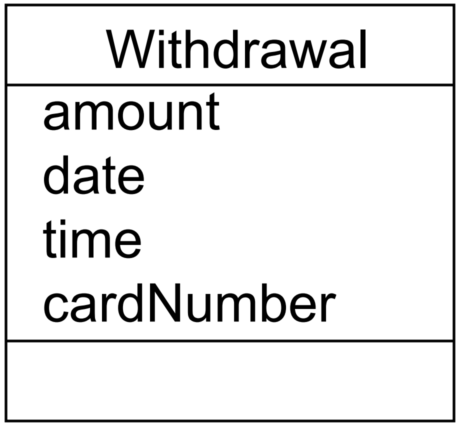
Attributes

- Attributes describe the properties of concepts.
- Attributes are needed to store information that an object must remember.

Attributes

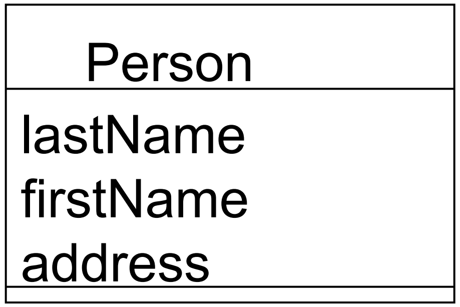
- Question you can ask:
  - What information do the concepts need to store?
  - What states can the concepts be in?

E.g. here is the concept withdrawal with attributes:

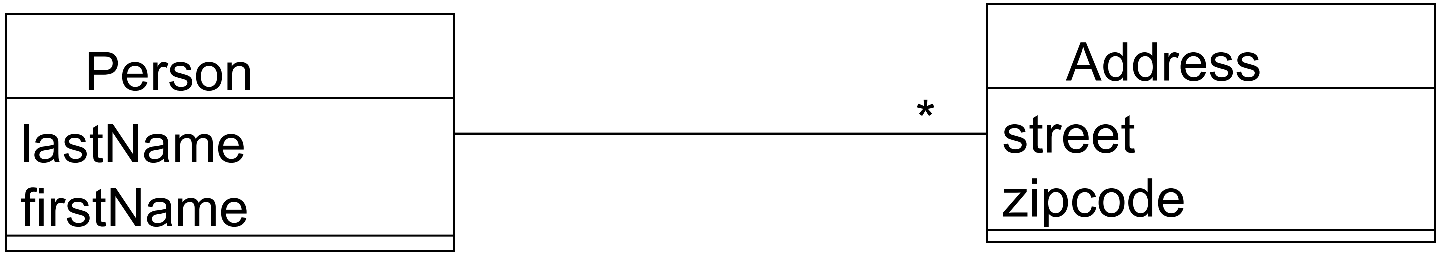


Attribute or concept (1)?

- This is not always easy to decide? You often see the concept Person with these attributes:



- A rule of thumb is that an attribute should have values of a primitive type, e.g. int, bool. lastName and firstName are both good attributes, but what about address? You could represent address with a string. But an address is not primitive, it contains itself different parts, e.g.:

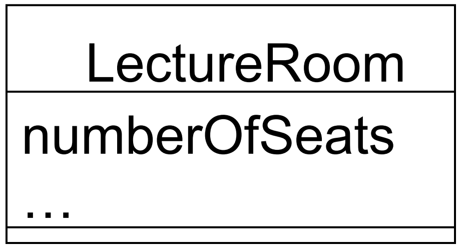


Attribute or concept (2)?

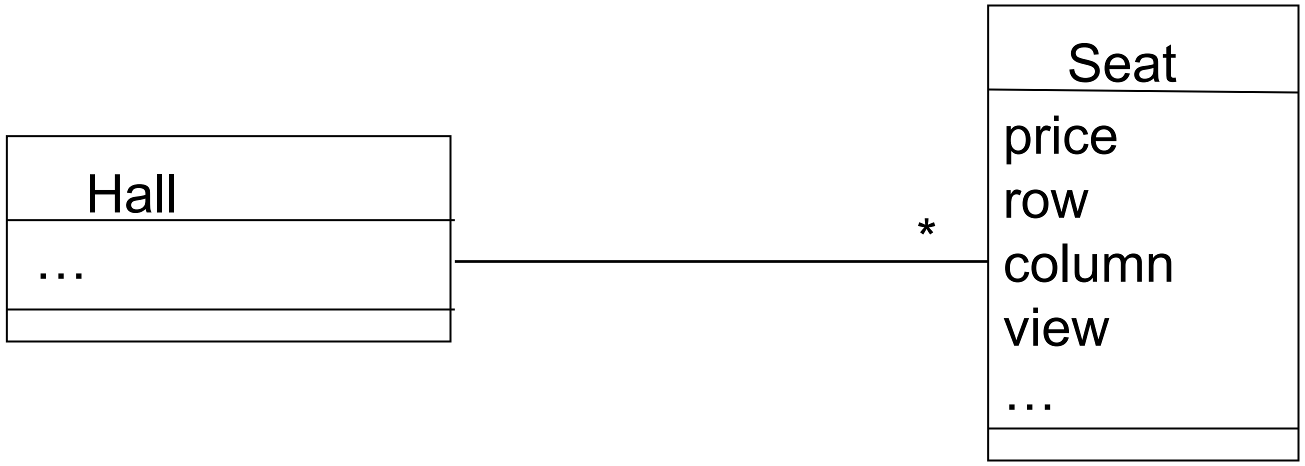
- The problem area and the responsibility can often indicate whether to make something a concept or an attribute. We will look at two problem area:
  - Booking system for courses.
  - Booking system for a theatre seats

Problem area

Booking system for courses

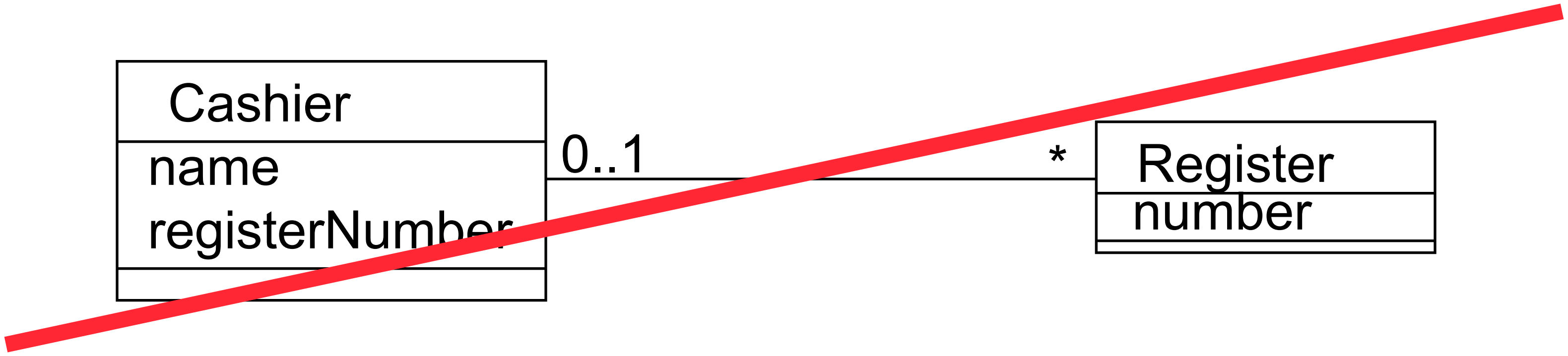


Booking system for a theatre seats

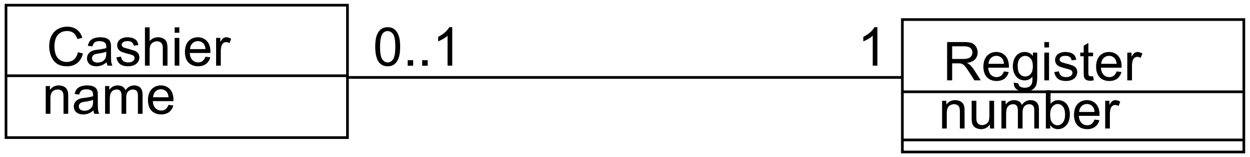


Foreign keys

- Foreign keys are used in database-tables to refer to other tables. For example:
  - Register(number)
  - Cashier(name,registerNumber)
    - registerNumber -> Register.number



- This is correct:

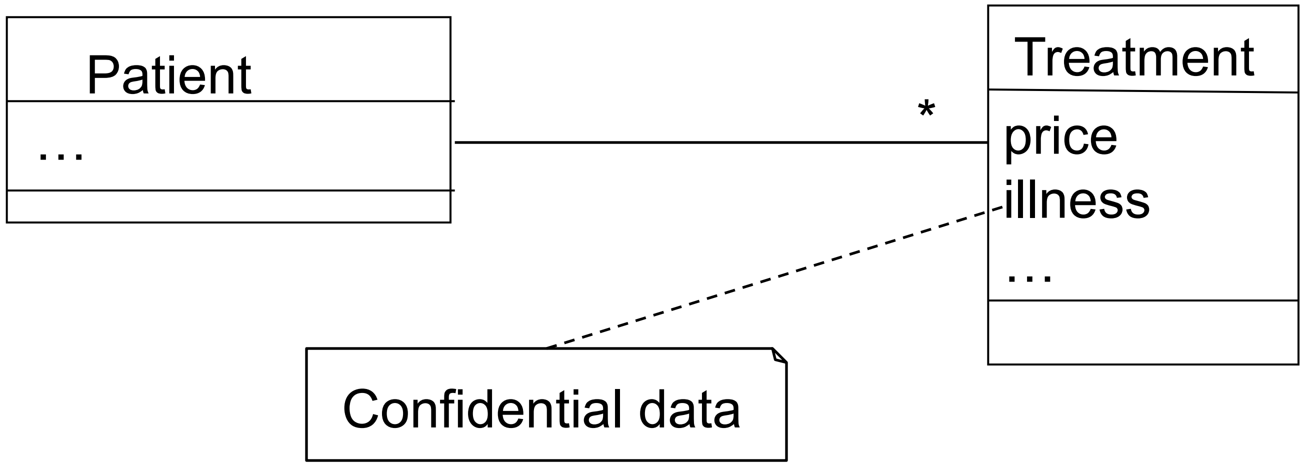


Security

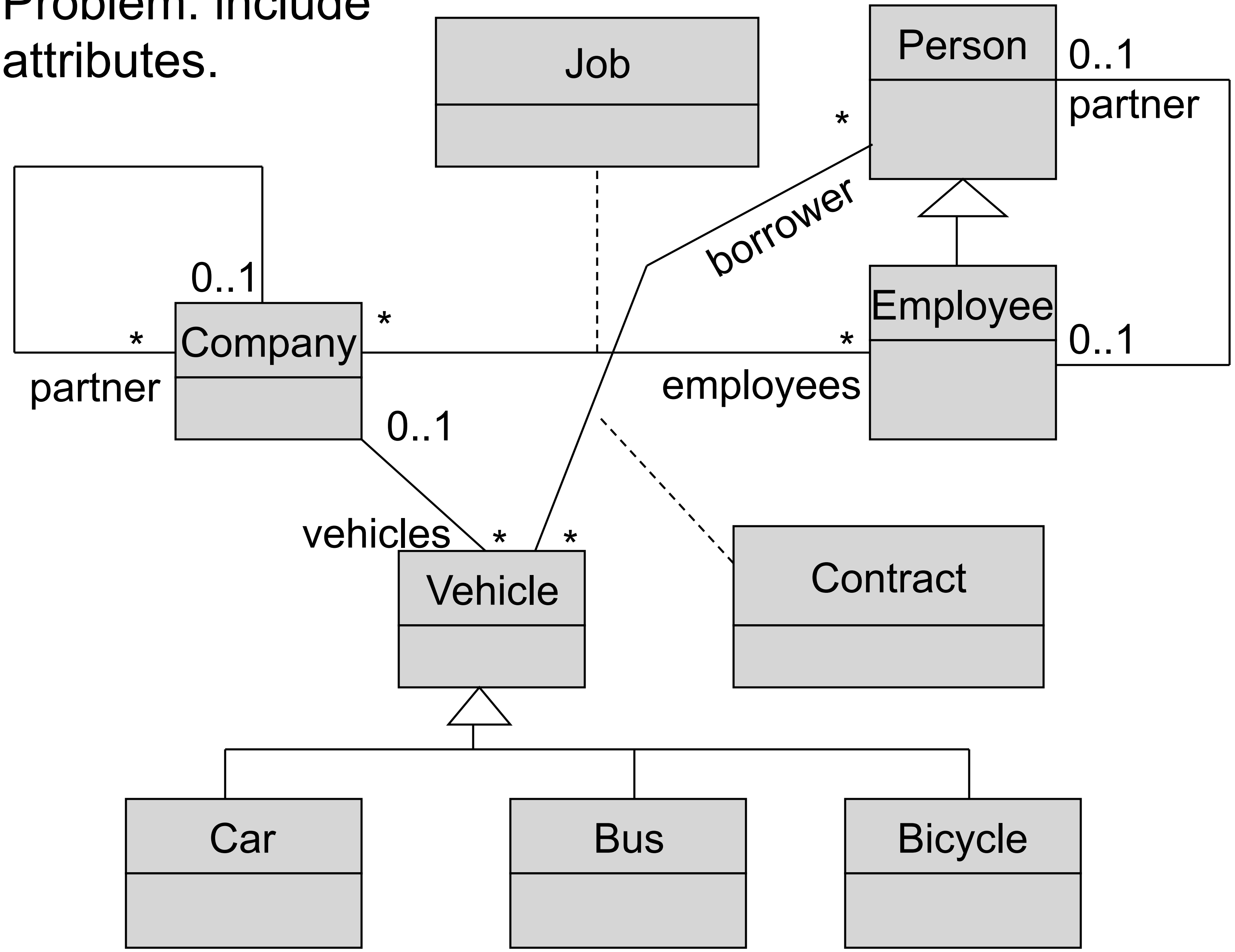
You should discuss the concepts in terms of security. Some concepts may contain information that shouldn’t be read or written.

Example: Confidential data

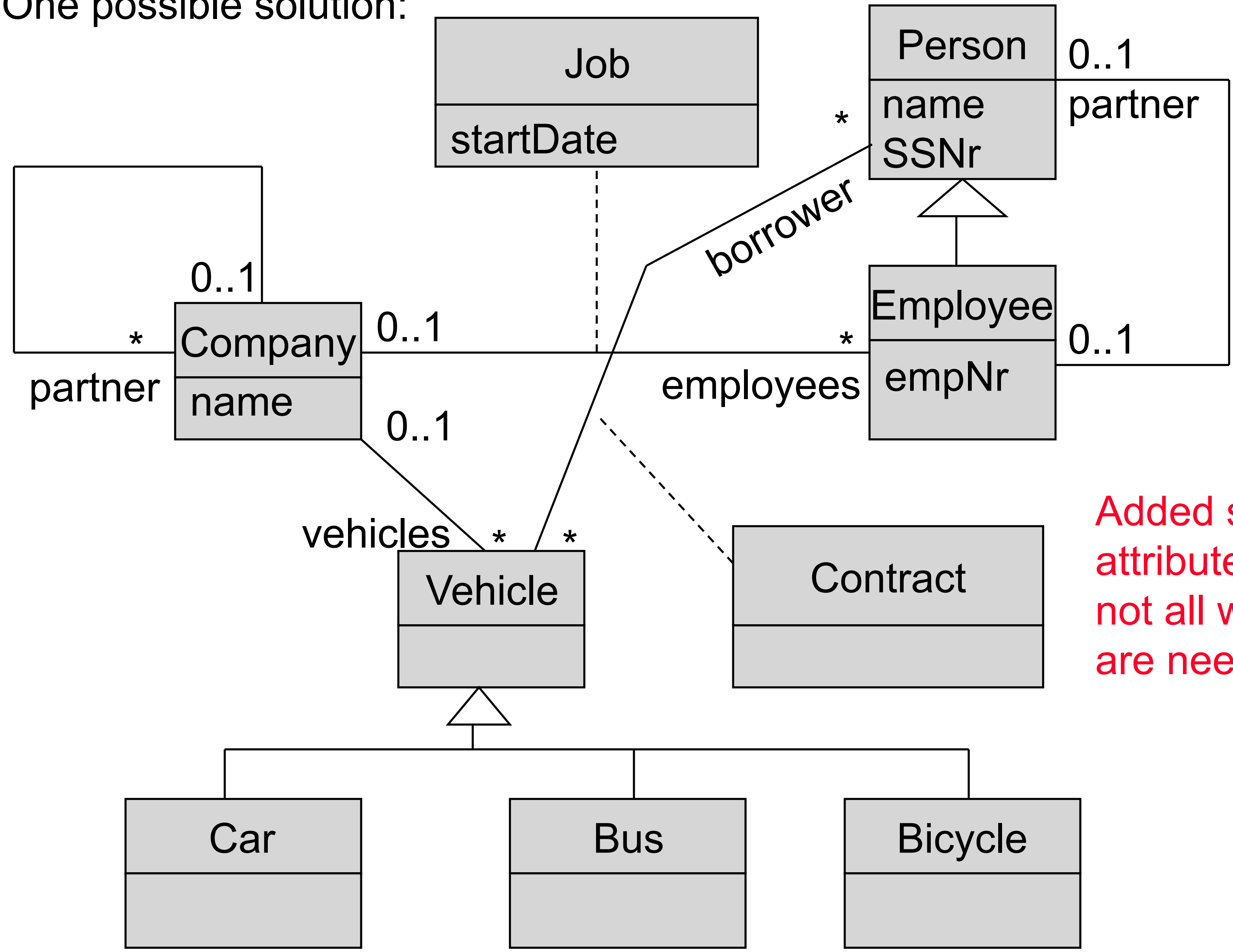
Hospital system:



Problem: include attributes.

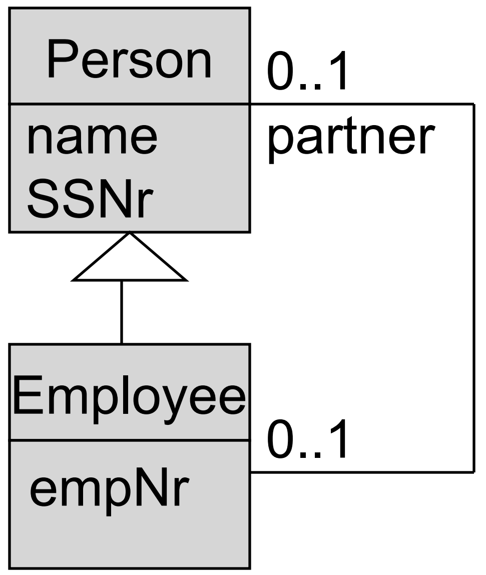


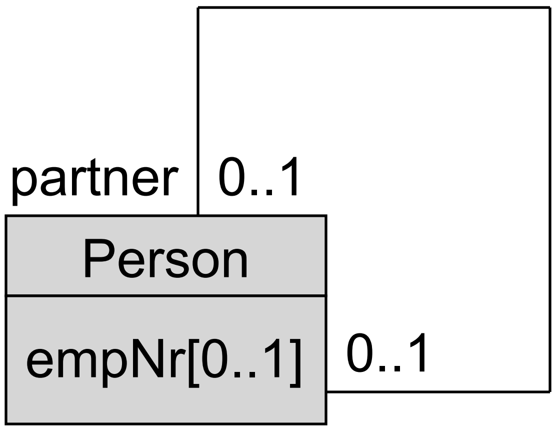
One possible solution:

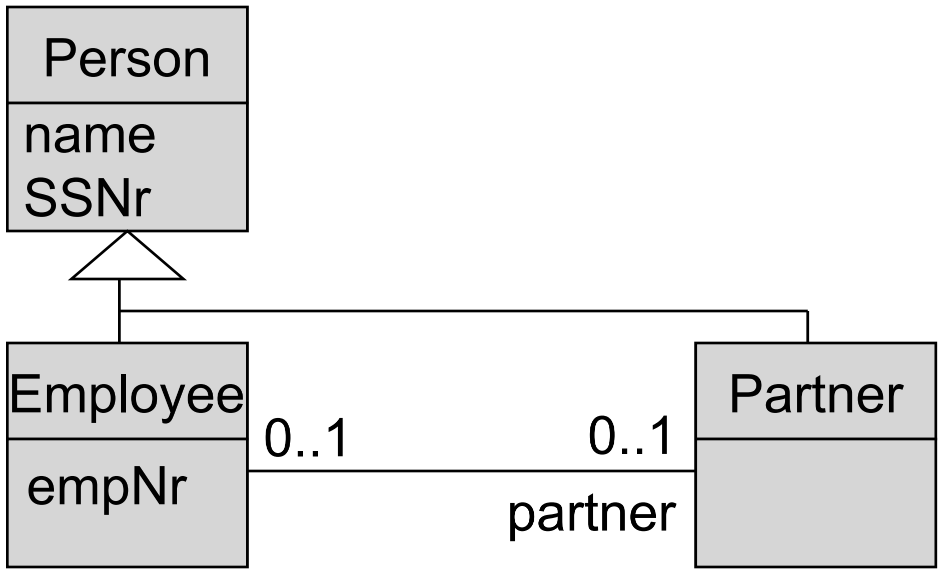


Added some attributes, but not all which are needed!

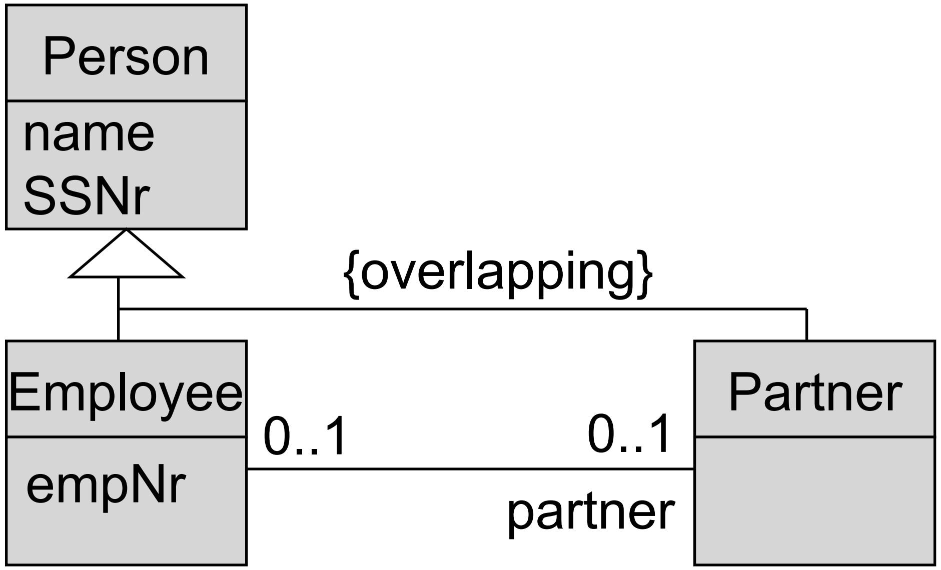
Partner (1)


Person
name
SSNr
0..1
partner
Employee
empNr
0..1


partner
0..1
Person
empNr[0..1]
0..1


Person
name
SSNr
Employee
empNr
0..1
0..1
partner
Partner

Partner (2)


Person
name
SSNr
{overlapping}
Employee
empNr
0..1
0..1
partner
Partner

Explanation document

- The domain model in itself is not enough
- Need a document do describe decisions made

Services

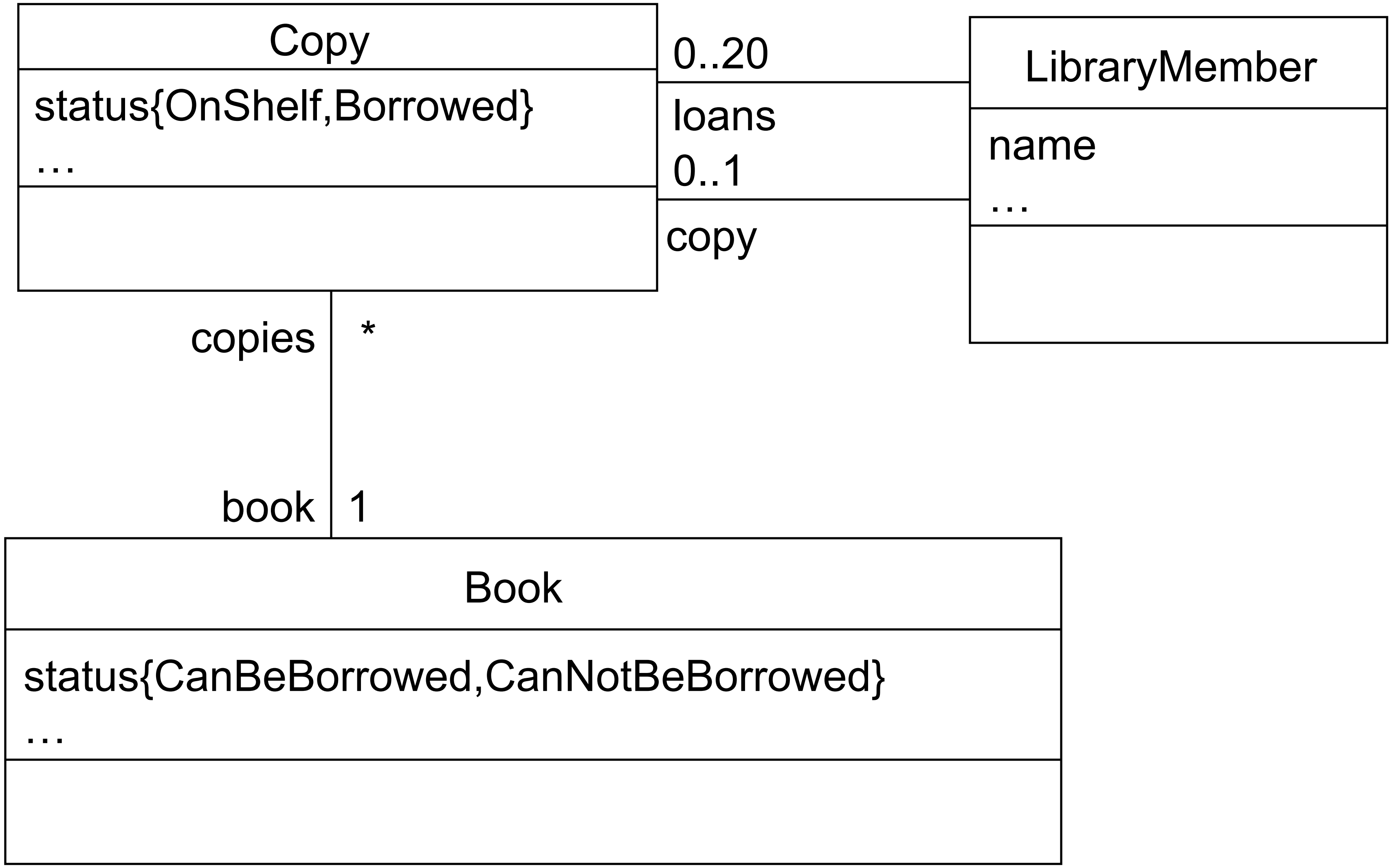
- Specifying at a high level the key services that the concept much offer.
  - The key responsibility of the concept

- Comments: if a concept have no responsibility one might question whether the concept is good or not.

Finding services

- Verb and verb phrases in a text indicate responsibility, services the concept provides. Here one can use
  - Description of the system to be built
  - Use cases
  - Requirements
  - Any thing else (architecture, vision document etc.)

Concept Diagram



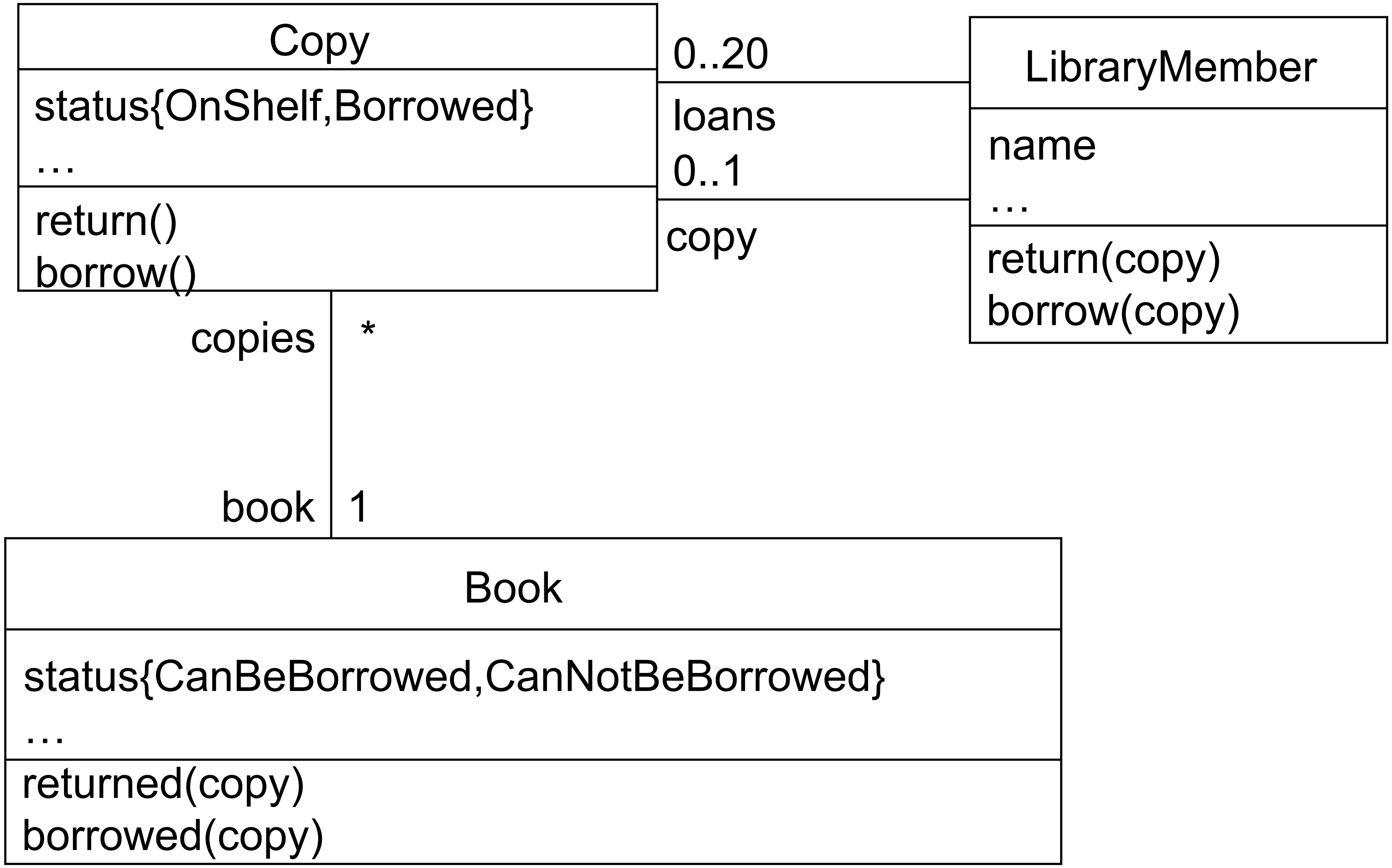
CRC

Library Member	
Responsibility	Collaboration
Maintain data about borrowed book copies. Handle requests to borrow and return book copies.	Copy

Copy	
Responsibility	Collaboration
Maintain data for a certain copy of a book. Inform corresponding book when the copy is borrowed and returned.	Book

Book	
Responsibility	Collaboration
Maintain data for a book. Know if there is a copy to borrow.	

Class Diagram



Summary

After the analysis you should have:

Concept-models in UML
- concepts
- associations
- attributes
- multiplicities
- etc.

You should also have responsibilities for each concept, limitations and a concept dictionary.