TA Office Hour

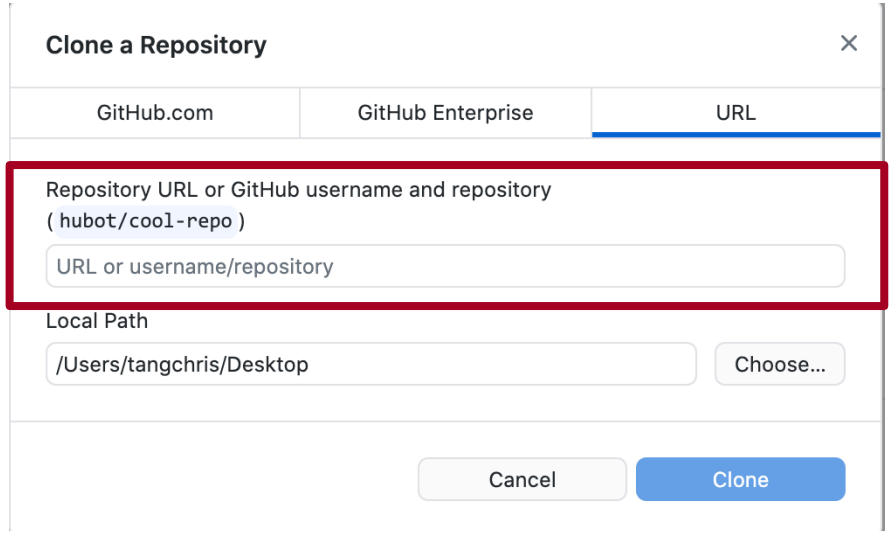
- TA Office Hour:
  - Loc: 1B103, SIST;
  - Time slots: 18:55-20:35, Wednesday;
  - Start from this week;
  - All 3 projects;

Project

- 2-member team:
  - Elevator (1R,2D,1V)
  - Railway Control Center (2R,1D,2V)
- Documentation (i.e., specification, report), Weekly report
  - Write with Markdown;
- Confirm the team composition and job allocation at the following link:
  - https://shimo.im/sheets/ckjdGXtjYQQQKXhq/MODOC

Project (2)

- Gitlab
  - Command Line;
  - Github Desktop;



With your git repository

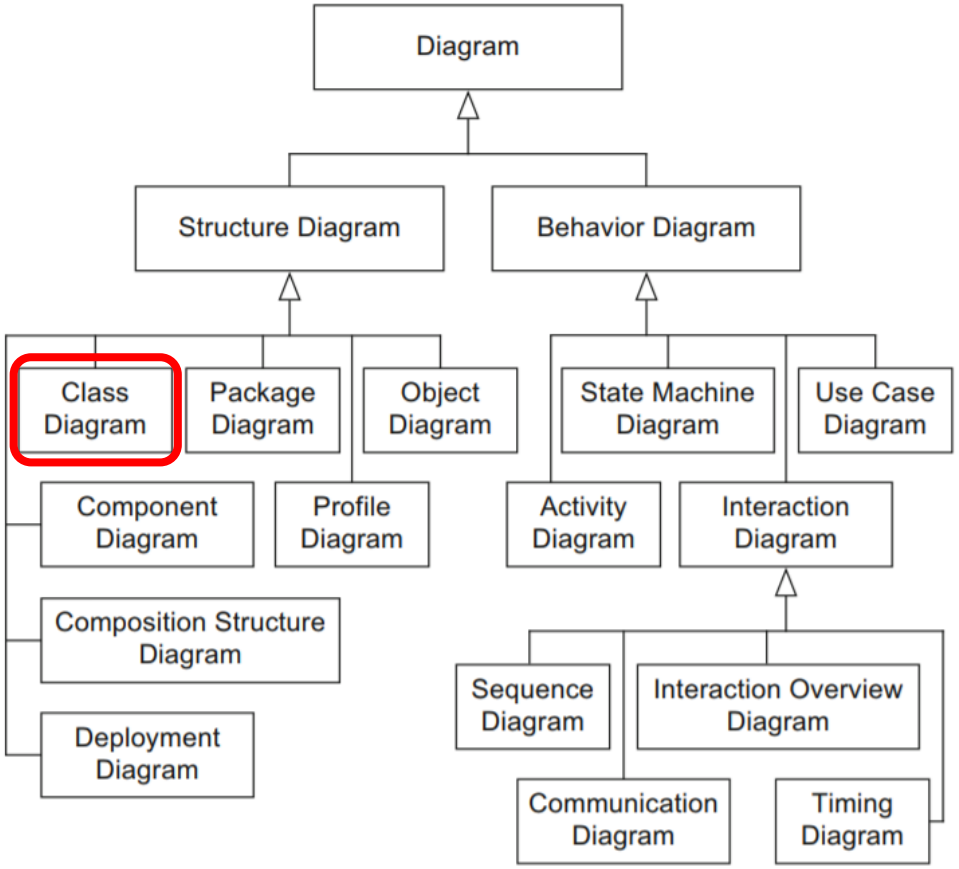
- Contact us if there are any problems with your Gitlab account and repos.

Markdown (doc. + weekly report)

- Files with the .md or .markdown extension
- Editor:
  - https://dillinger.io/
  - Atom, Sublime Text, …
- Guide (Syntax):
  - https://guides.github.com/features/mastering-markdown/
  - The Markdown Guide (uploaded to BB)

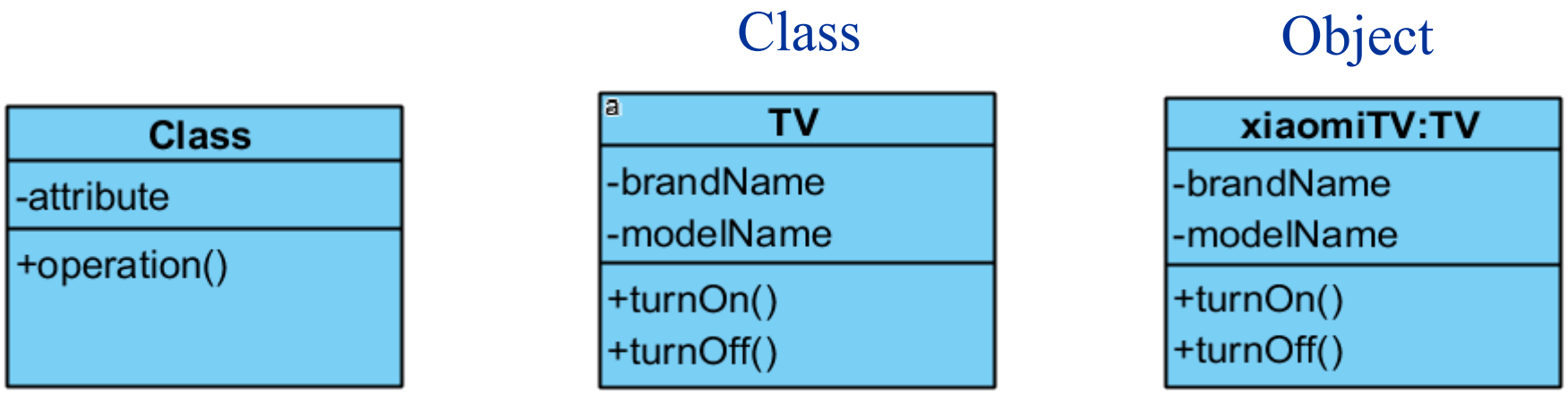
Lecture 9: Class Diagram

UML Diagrams



Class and object

- Class: The basic component of object-oriented approaches
- A class is a construction plan for a set of similar objects of a system
- Objects are *instances* of classes.
- Each class has a list of attributes and operations

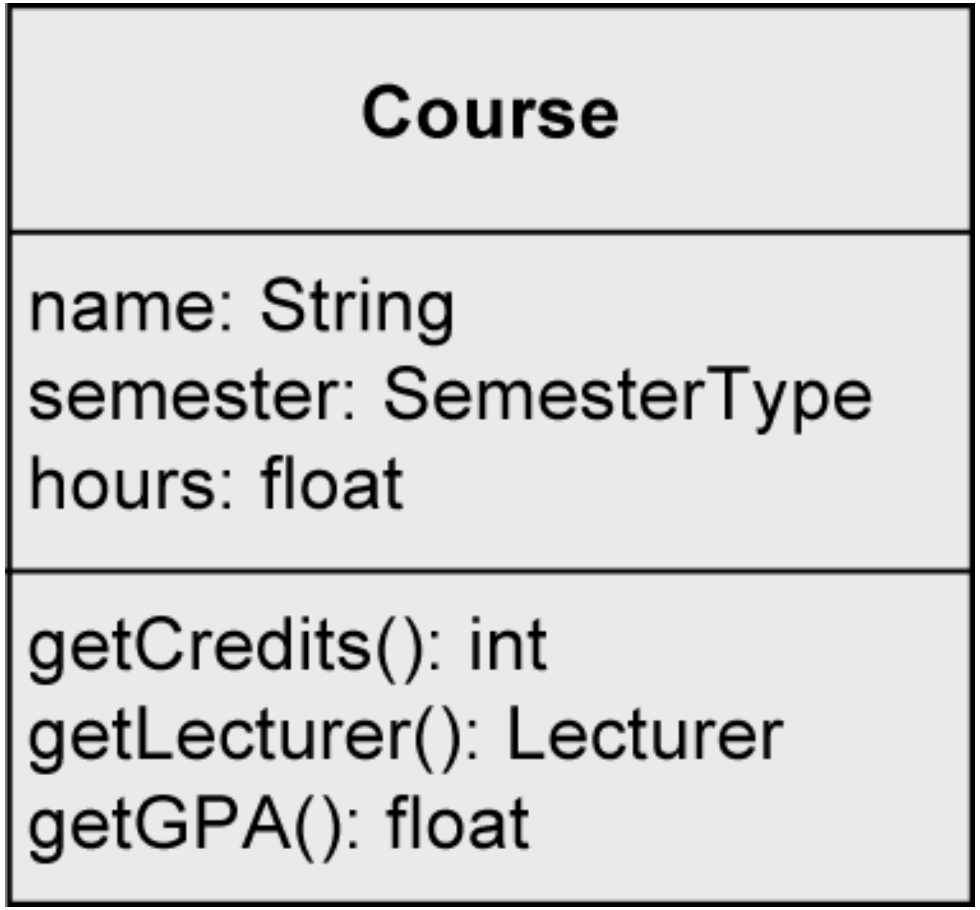


Class (2)

Class name

Attributes

Operations



Class (3): Visibility

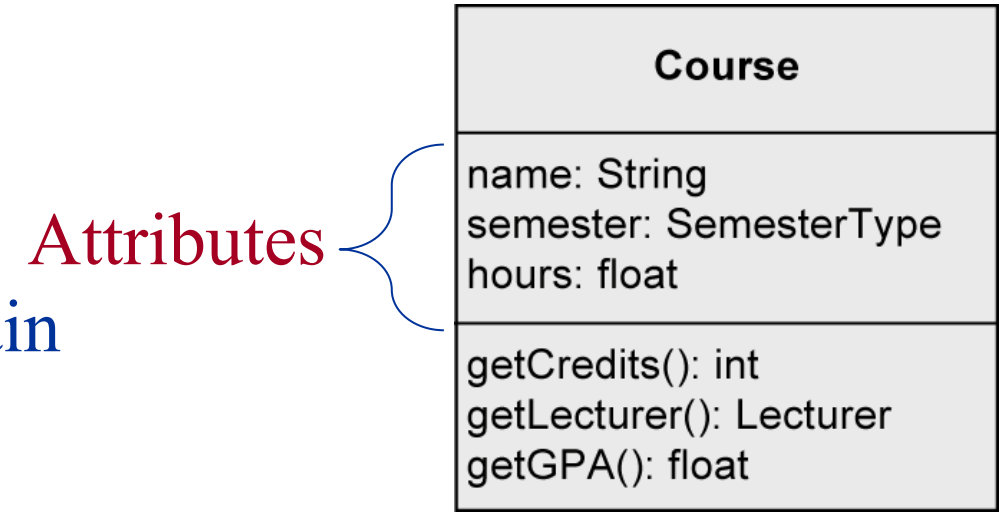
- Who is permitted to access:
  - + global: accessible to all
  - - private: accessible within the object
  - # protected: accessible by class itself and its sub-classes

Person

+ firstName: String
+ lastName: String
– dob: Date
address: String[1..*] {unique, ordered}
– ssNo: String {readOnly}
– /age: int
– password: String = "pw123"
– <u>personsNumber: int</u>

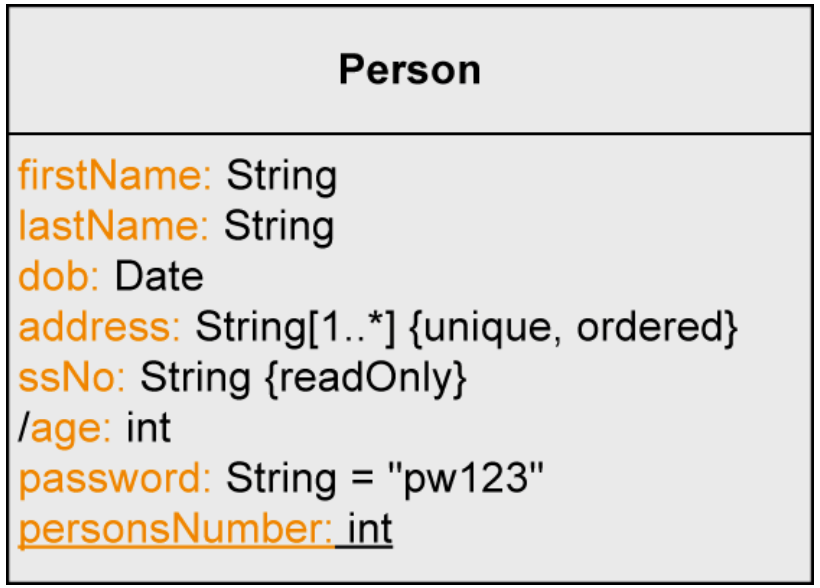
Attributes

- Name
  - Noun clause, lowercase first letter, then uppercase for latter words
    - i.e. gradStudent, firstName
- Type
  - i.e. String
- Multiplicity: how many value it can contain
  - [min .. max]: i.e. [0 .. 1]
- Properties
- Which attributes to include depends on the stage of development
  - The closer to implementation, the more detailed the models are



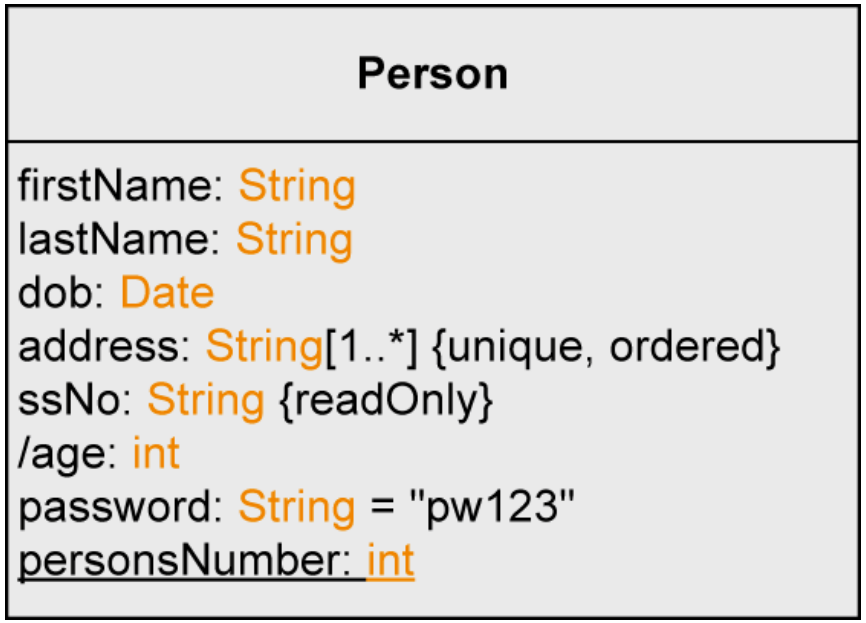
Attributes -- Name

- Name:
  - Noun clause, lowercase first letter, then uppercase for latter words
    - i.e. gradStudent, firstName



Attributes -- Type

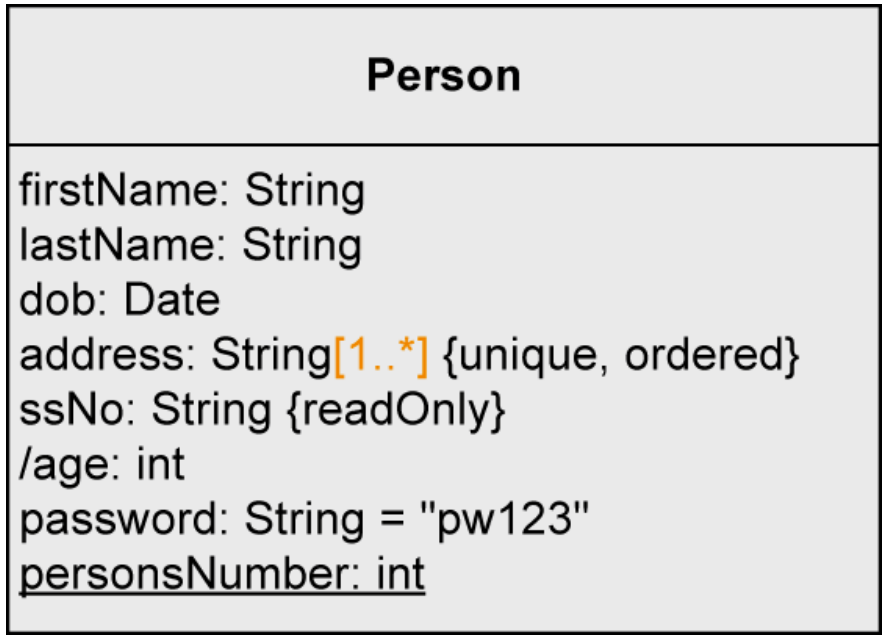
- Type
  - User-defined classes
  - Data Type



Attributes -- Multiplicity

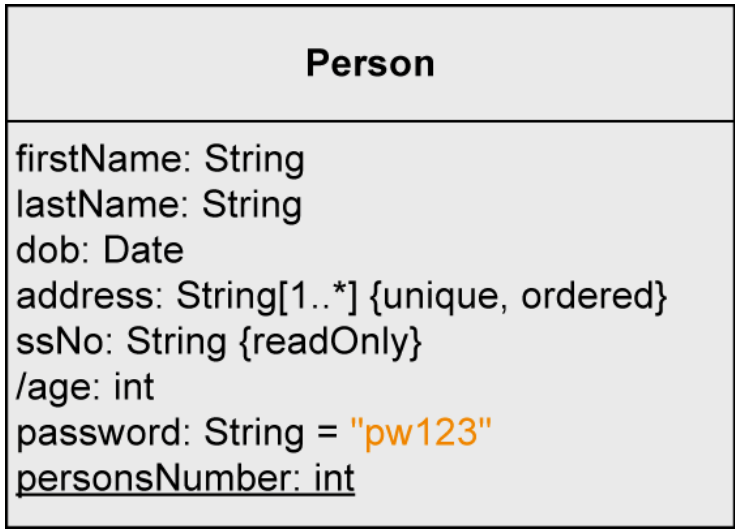
- Number of values an attribute may contain
- Default value: 1
- Notation: **[min..max]**
  - no upper limit: `[*]` or `[0..*]`
- E.g. address: String[1…*]

A person can have one address or multiple addresses



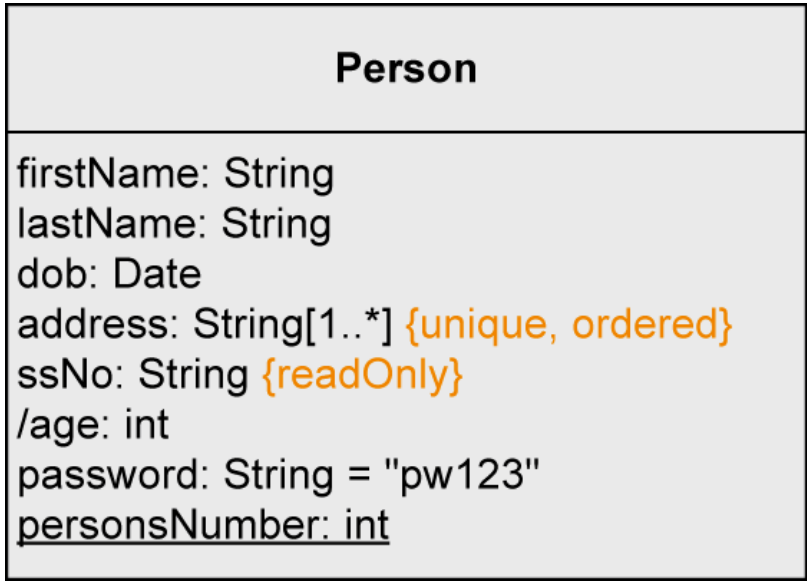
Attributes -- Default Value

- Default value
  - Used if the attribute value is not set explicitly by the user



Attributes -- Properties

- Pre-defined properties
  - `{readOnly}` … value cannot be changed
  - `{unique}` … no duplicates permitted
  - `{non-unique}` … duplicates permitted
  - `{ordered}` … fixed order of the values
  - `{unordered}` … no fixed order of the values
- Attribute specification
  - Set: `{unordered, unique}`
  - Ordered set: `{ordered, unique}`
  - List: `{ordered, non-unique}`



Attributes -- Operations

- Name
  - Verb clause: i.e. getGrade()
- Parameters
  - Direction: in, out, inout
  - Name
  - Data type
- Return value
  - Only need a data type
- Example
  - getName(out fn: String, out in: String): void
  - updateLastName(in newName: String): boolean

Attributes -- Parameters

- Notation similar to attributes

Person
…
+ getName(out fn: String, out ln: String): void + updateLastName(newName: String): boolean + getPersonsNumber(): int

Attributes -- Return

- Type of the return value

Person
...
getName(out fn: String, out ln: String): void updateLastName(newName: String): boolean getPersonsNumber(): int

Class Variable and Class Operation

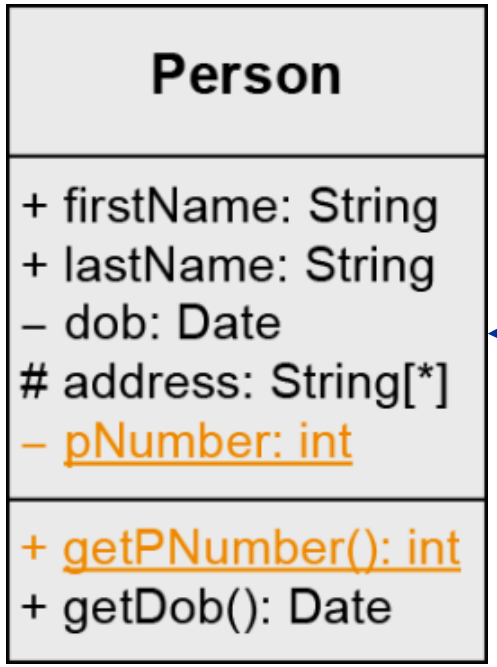
- Instance variable (= instance attribute): attributes defined on instance level
- Class variable
  - DefineE.g. counters for the number of instances of a class, constants, etc.

- Class operation
  - Can be used if no instance of the corresponding class was created
  - E.g. constructors, counting operations, math. functions (sin(x)), etc.

Class Variable and Class Operation (2)

Class variable

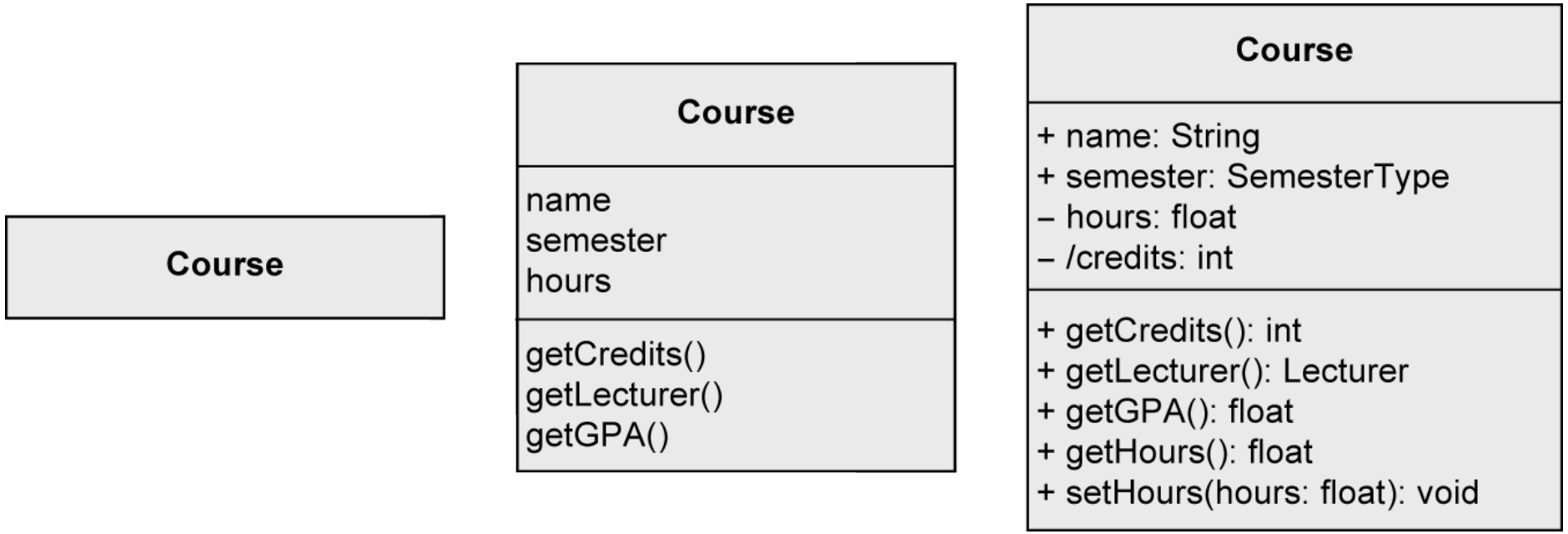
Class operation




```
class Person {

   public String firstName;
   public String lastName;
   private Date dob;
   protected String[] address;
   private static int pNumber;
   public static int getPNumber()
{…}
   public Date getDob() {…}
}
```

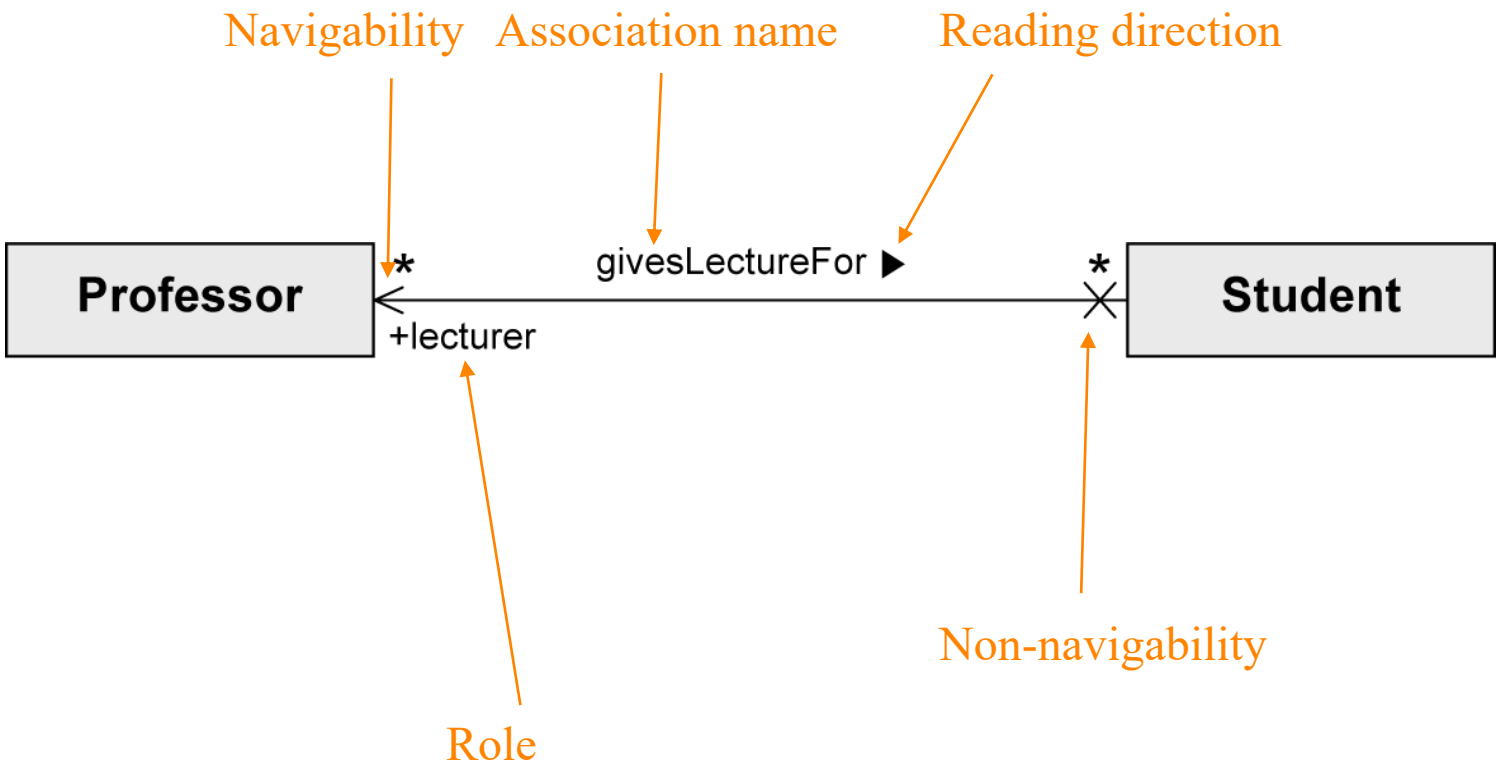
Specification of Classes: Different Levels of Detail

coarse-grained ⟷ fine-grained

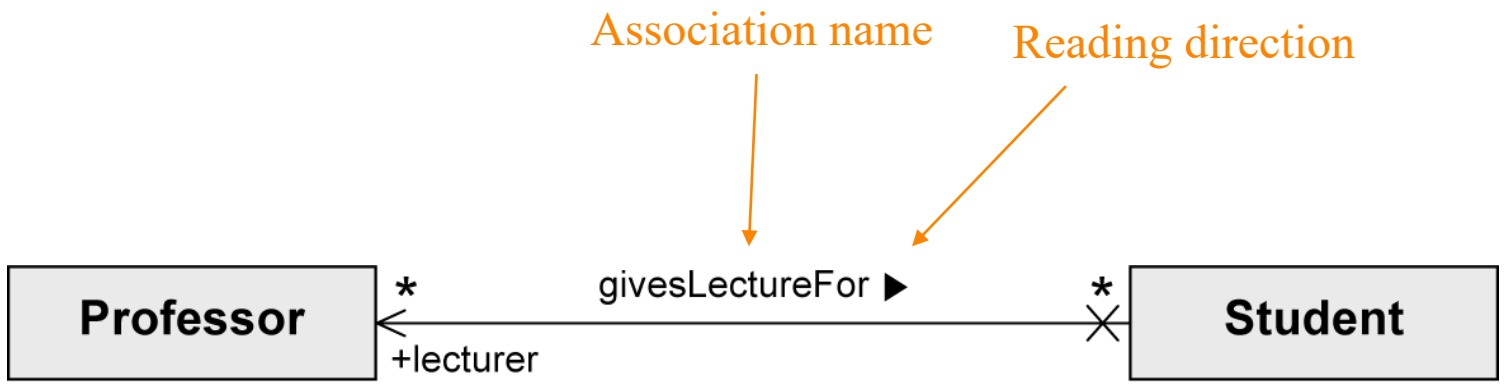


Association

- Model relationships between the classes. They describe which classes are potential communication partners.



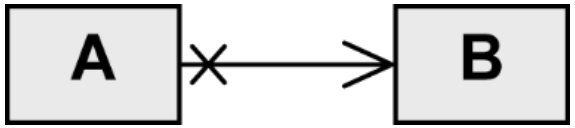
Association – Association Name



- Association name: givesLectureFor
- Reading direction: ▶
  - Professor givesLectureFor Student

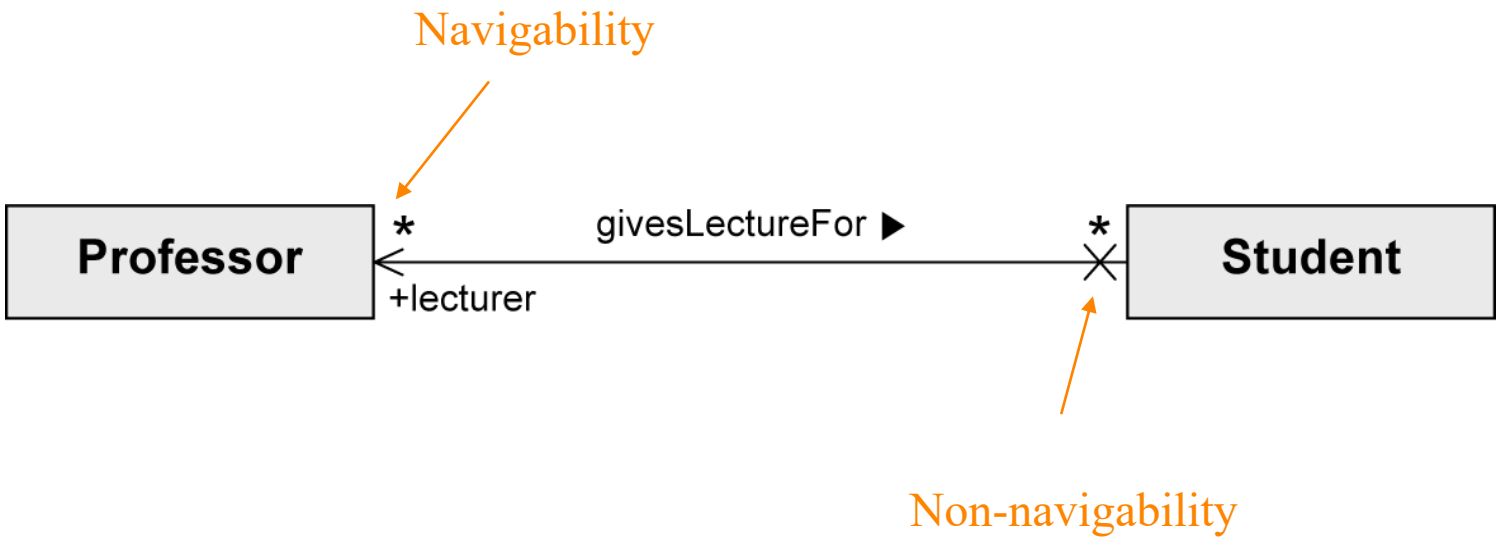
Binary Association - Navigability

- Navigability: an object knows its partner objects and can therefore access their visible attributes and operations
  - Indicated by open arrow head
- Non-navigability
  - Indicated by cross
- Example:
  - **A** can access the visible attributes and operations of **B**
  - **B** cannot access any attributes and operations of **A**



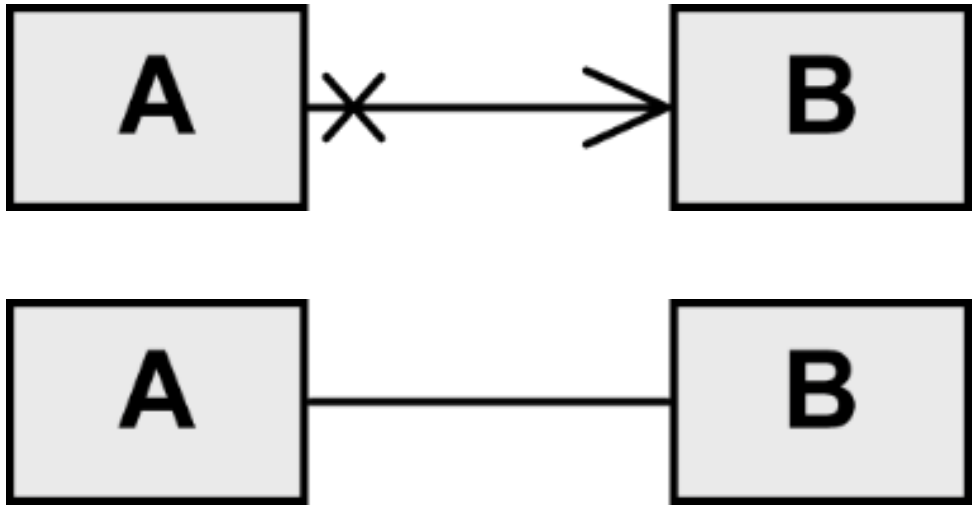
Binary Association – Navigability (2)

- Navigability
  - Student can access the attributes and operations of Professors;
  - Processors cannot access the attributes and operations of Professor



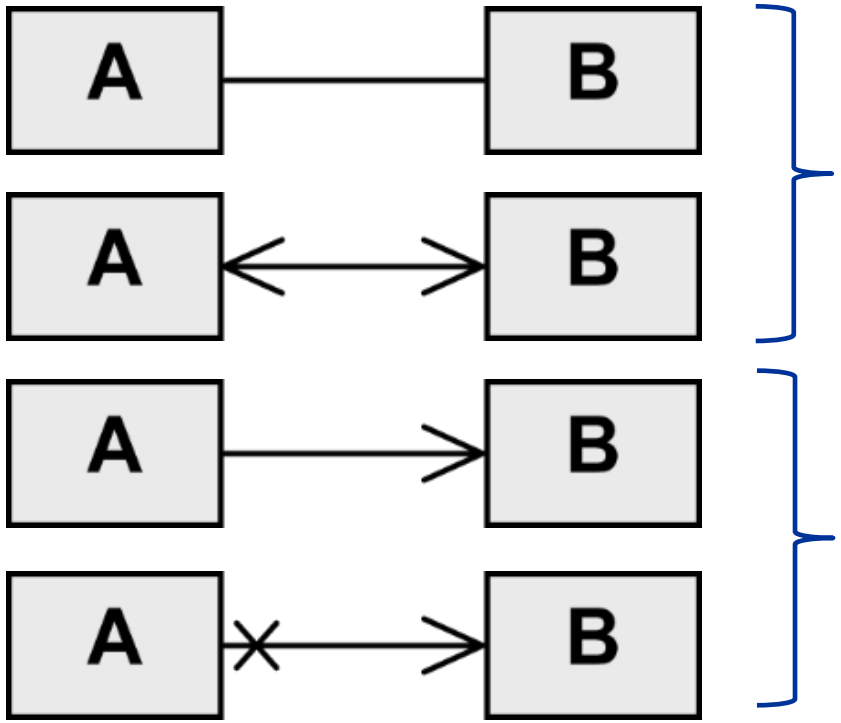
Binary Association – Navigability (3)

- Navigability undefined
  - Bidirectional navigability is assumed

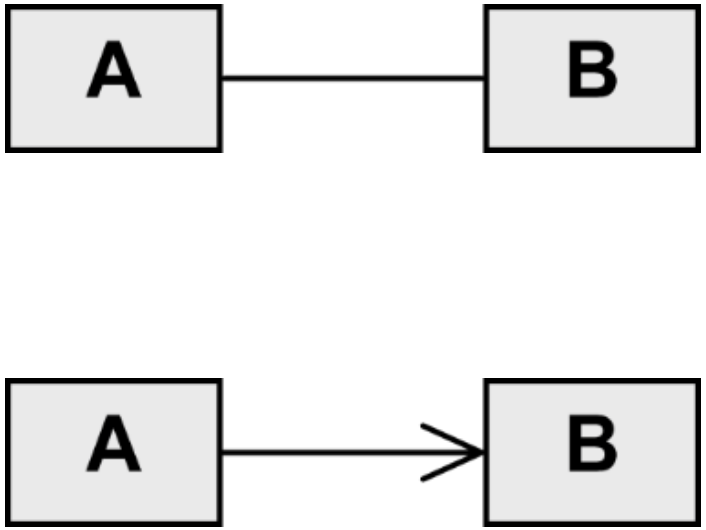


Navigability – UML Standard vs. Best Practice

UML Standard

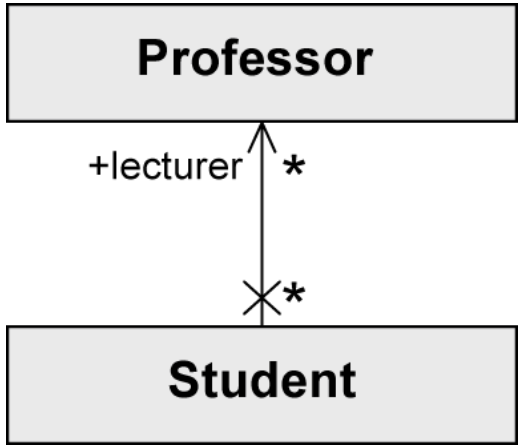


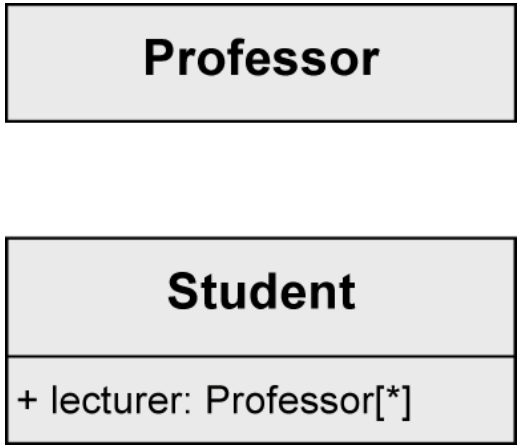
Best Practice



Binary Association as Attribute

Preferable



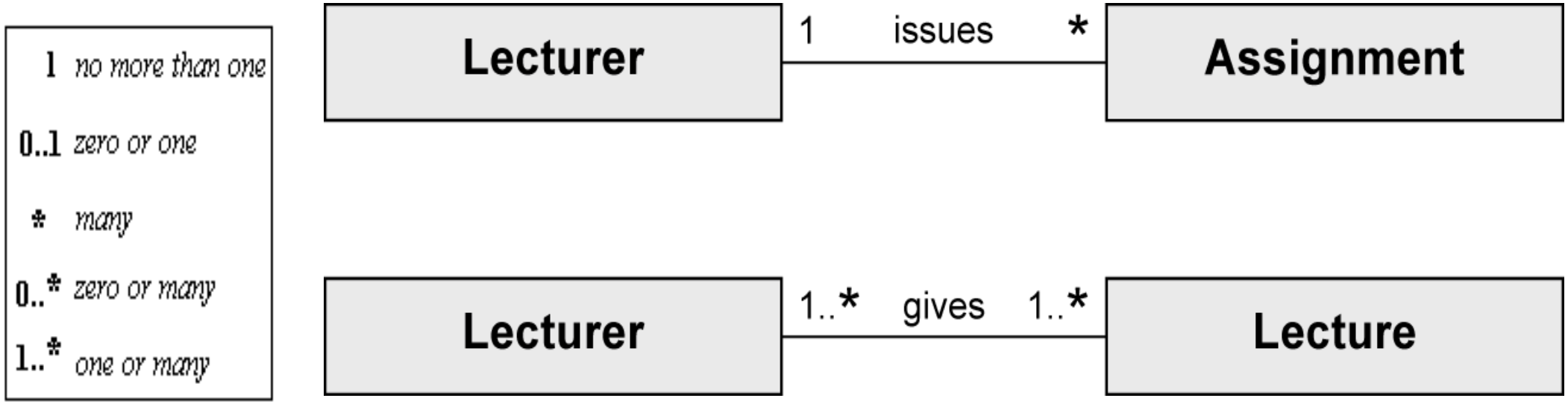


- Java-like notation:

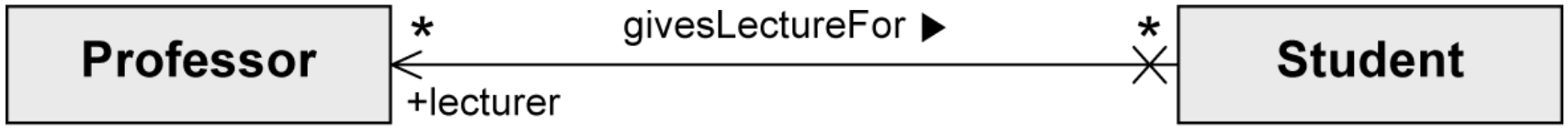
```
class Professor {…}

class Student{
   public Professor[] lecturer;
   …
}
```

Binary Association – Multiplicity

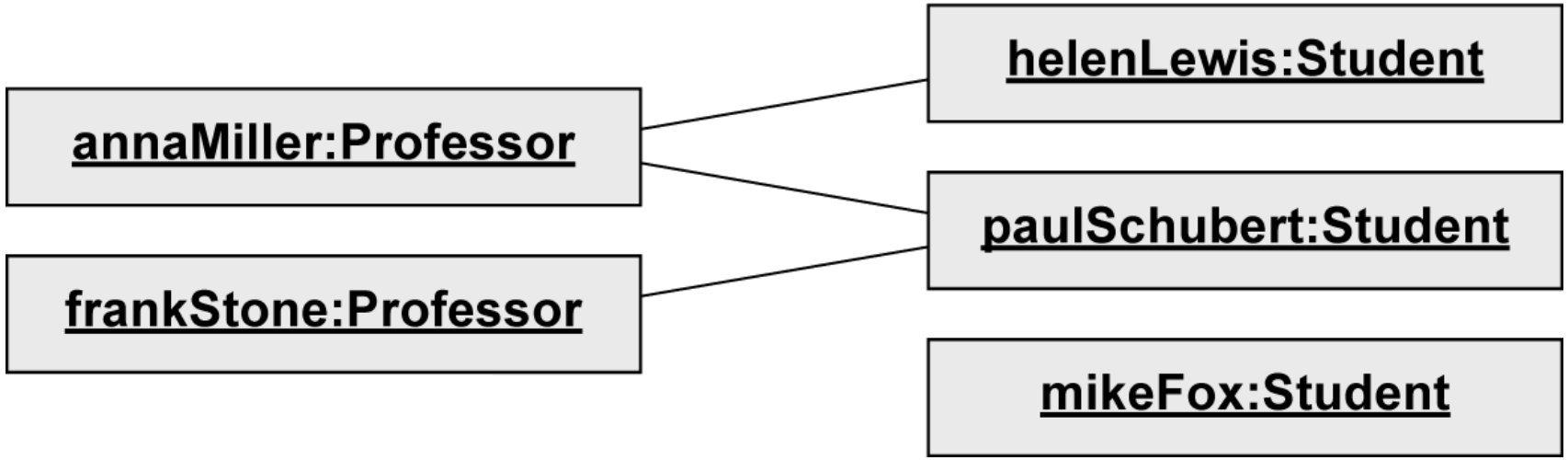
- Multiplicity: Number of objects that may be associated with exactly one object of the opposite side



Binary Association – Multiplicity (2)

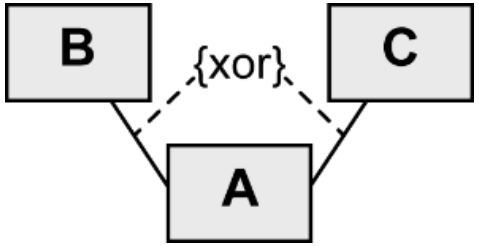


Class Diagram

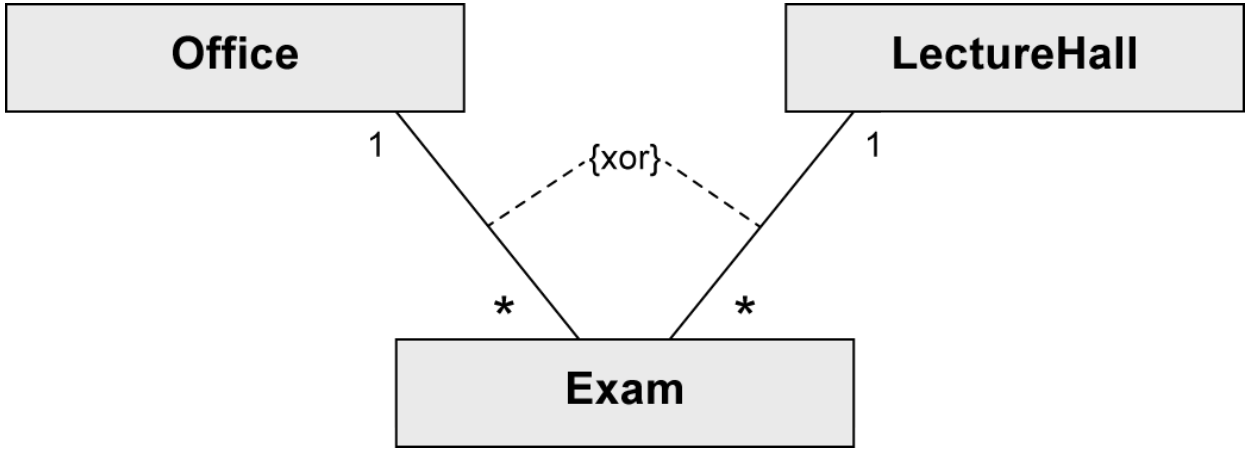


Object Diagram

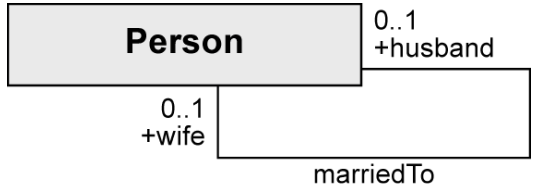
Binary Association – `xor` constraint



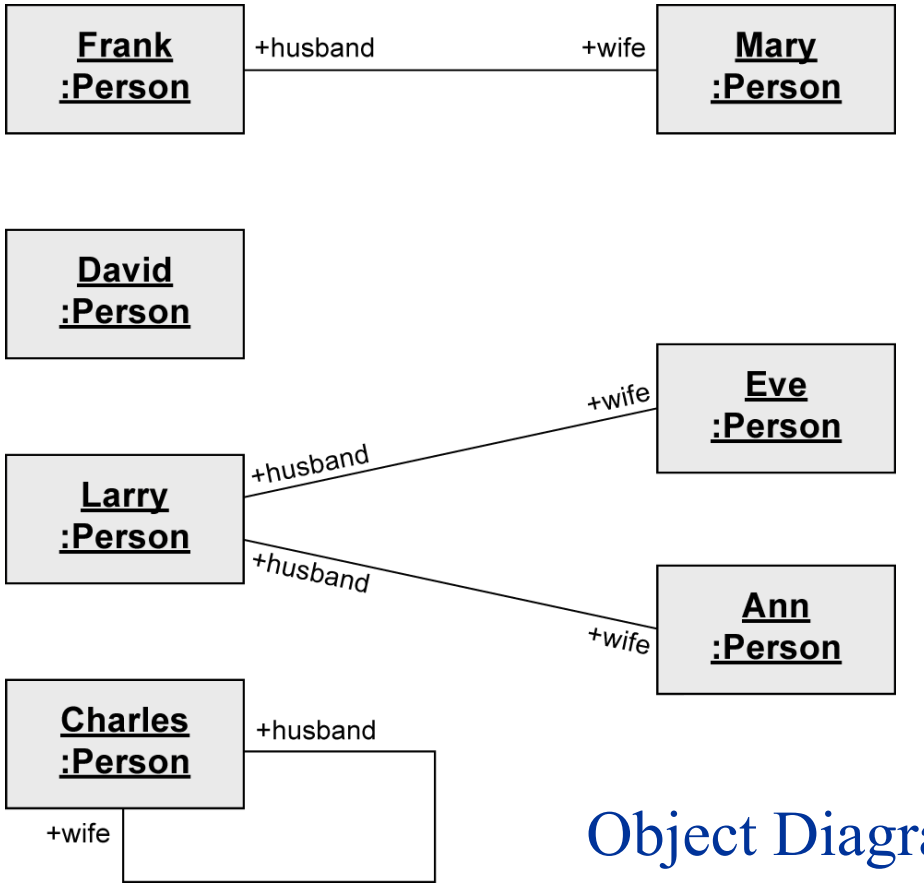
- "exclusive or" constraint
- An object of class **`A`** is to be associated with an object of class **`B`** or an object of class **`C`** but not with both.



Unary Association - Example



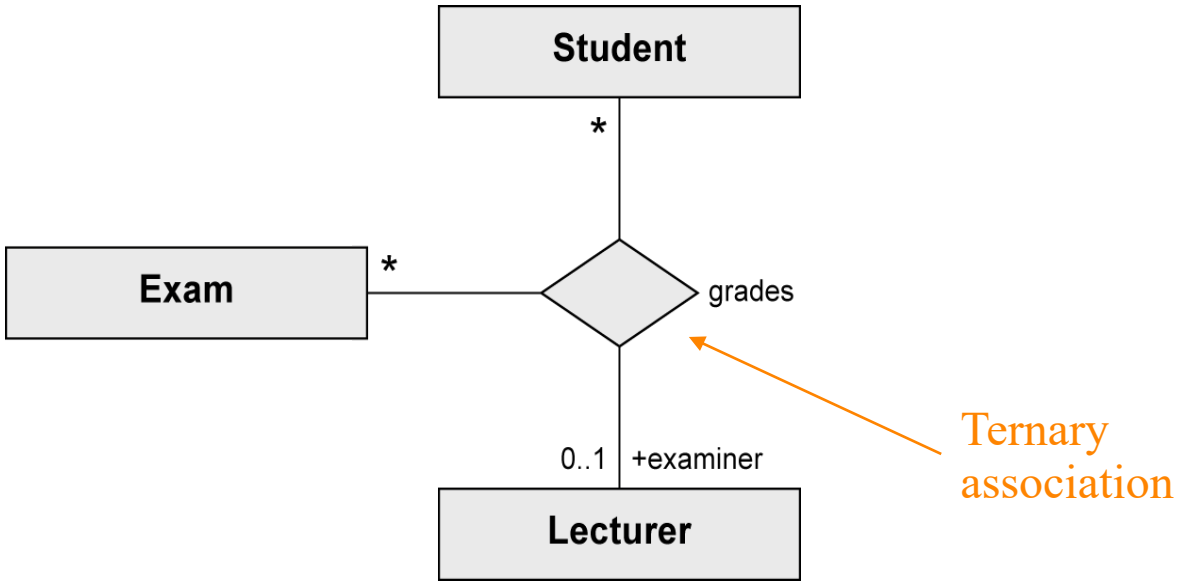
Class Diagram



Object Diagram

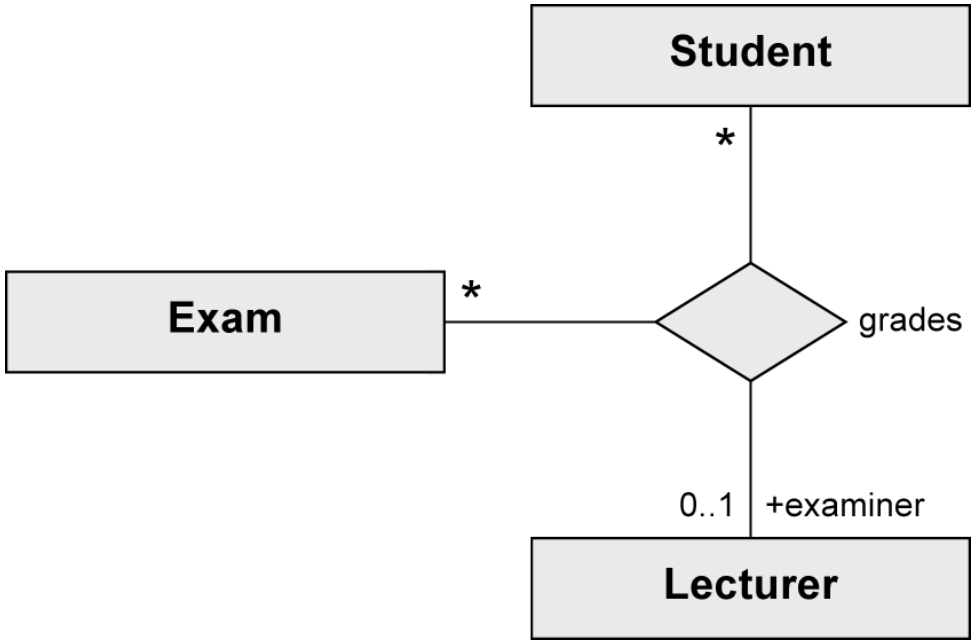
N-ary Association

- More than two partner objects are involved in the relationship.
- No navigation directions



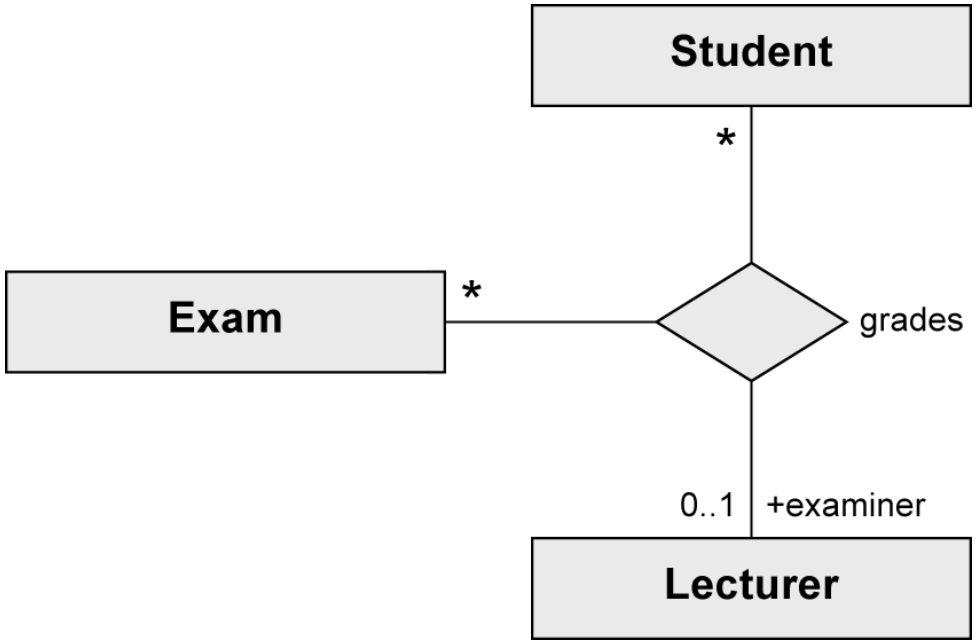
N-ary Association (2)

- Example
  - (**`Student`**, **`Exam`**) → (**`Lecturer`**)
    - One student takes one exam with one or no lecturer



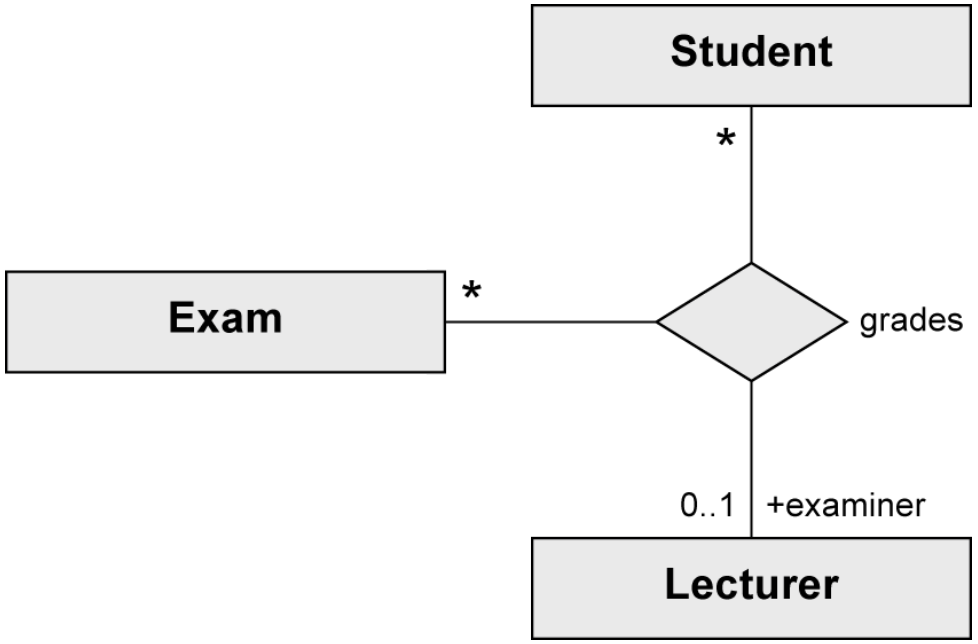
N-ary Association (3)

- Example
  - (**`Exam`**, **`Lecturer`**) → (**`Student`**)
    - One exam with one lecturer can be taken by any number of students



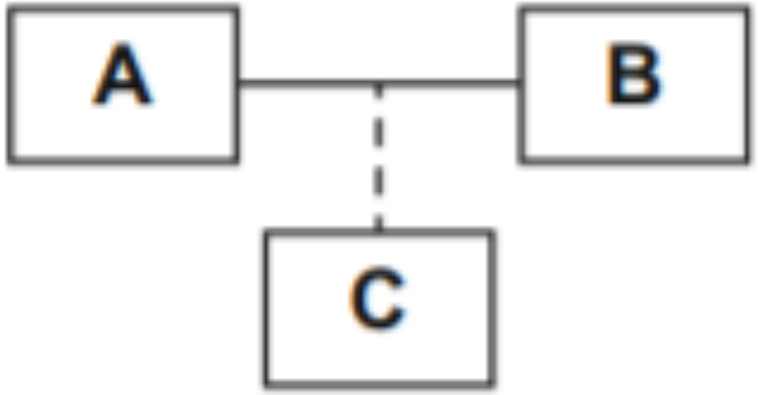
N-ary Association (4)

- Example
  - (**`Student`**, **`Lecturer`**) → (**`Exam`**)
    - One student can be graded by one **`Lecturer`** for any number of exams



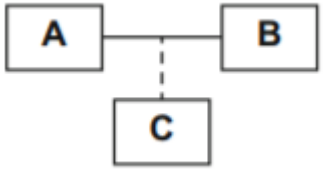
Association Class

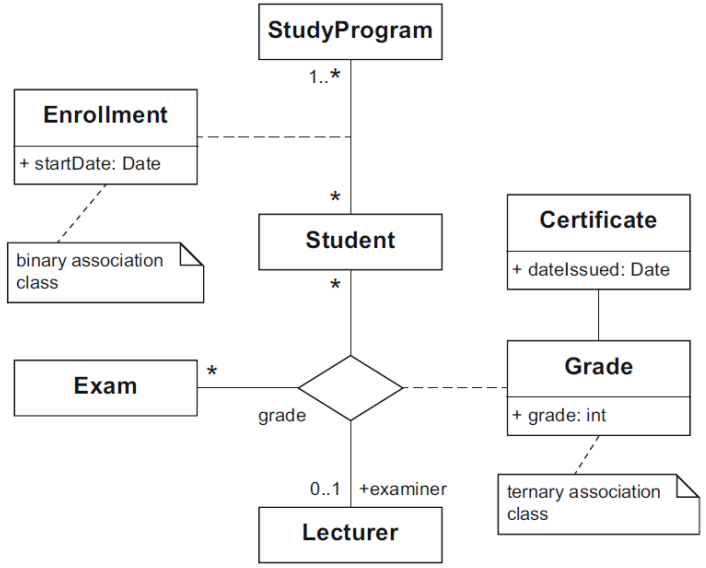
- If you want to assign attributes or operations to the *relationship* between one or more classes rather than to a class itself, you can do this using an **association class**.

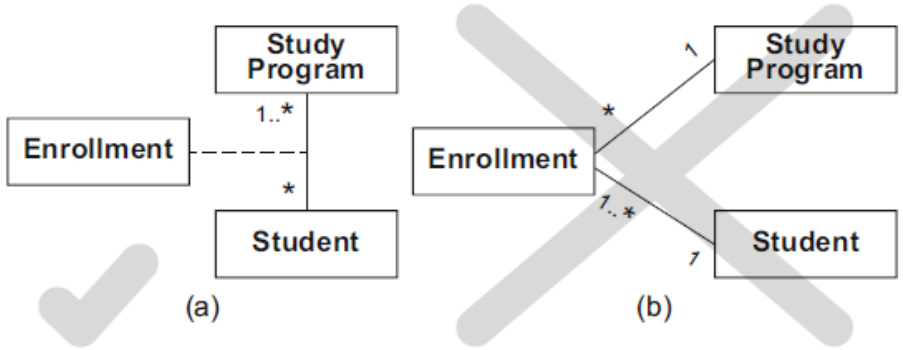


Association Class (2)

- Has the property of both a class and an association
- Cannot be simply replaced by a “normal” class
- In (b), a student can enroll a study program multiple times







Aggregation

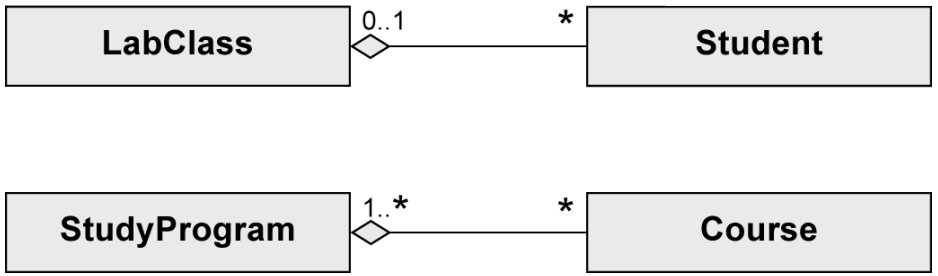
- A special form association: A is part of B

- Shared aggregations
  - A can also be part of something else
  - When B is gone, A can still exist
- Compositions
  - A specific part can only be contained in at most one composite object at one specific point in time.
  - A much stronger bond (normally physical)

Shared aggregations

- Expresses a weak belonging of the parts to a whole
  = Parts also exist independently of the whole
- Multiplicity at the aggregating end may be >1
  = One element can be part of multiple other elements simultaneously

- Syntax: Hollow diamond at the aggregating end
- Example:
  – **`Student`** is part of **`LabClass`**
  – **`Course`** is part of **`StudyProgram`**



Compositions

- Existence dependency between the composite object and its parts
- One part can only be contained in at most one composite object at one specific point in time
  - Multiplicity at the aggregating end max. 1
- If the composite object is deleted, its parts are also deleted.
- Syntax: Solid diamond at the aggregating end
- Example: **`Beamer`** is part of **`LectureHall`** is part of **`Building`**

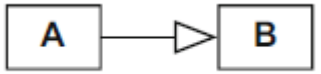
Compositions (2)

If the `Building` is deleted, the `LectureHall` is also deleted

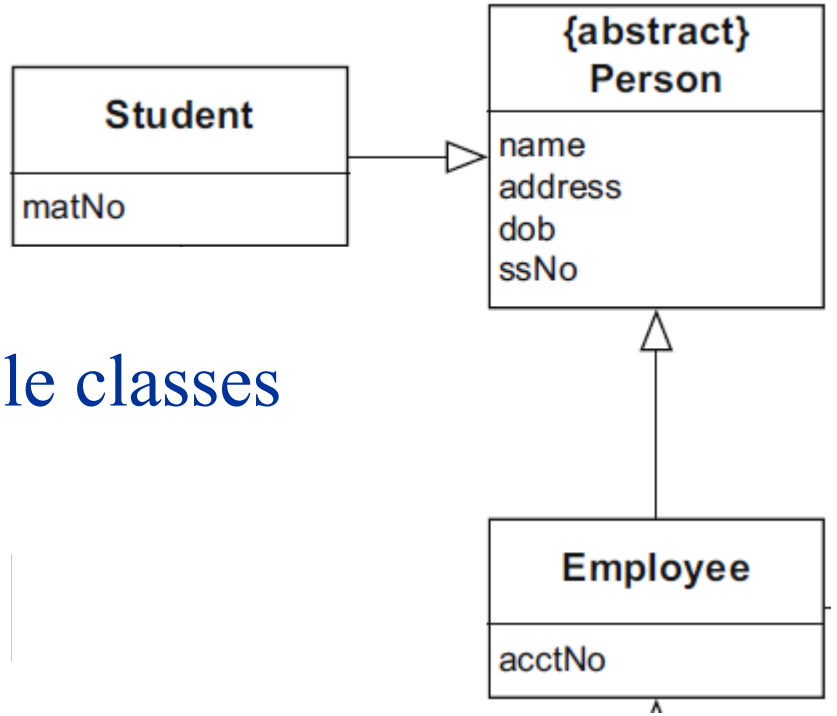
If it is contained in the `LectureHall` while it is deleted, the `Beamer` is also deleted

Generalization/Inheritance

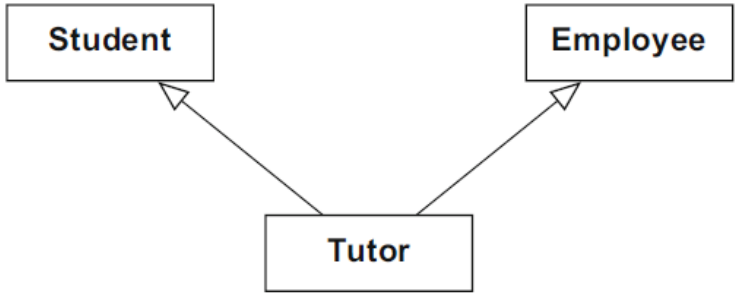
- Highlight common attributes and methods of objects and classes



- Abstract class
  - No instances



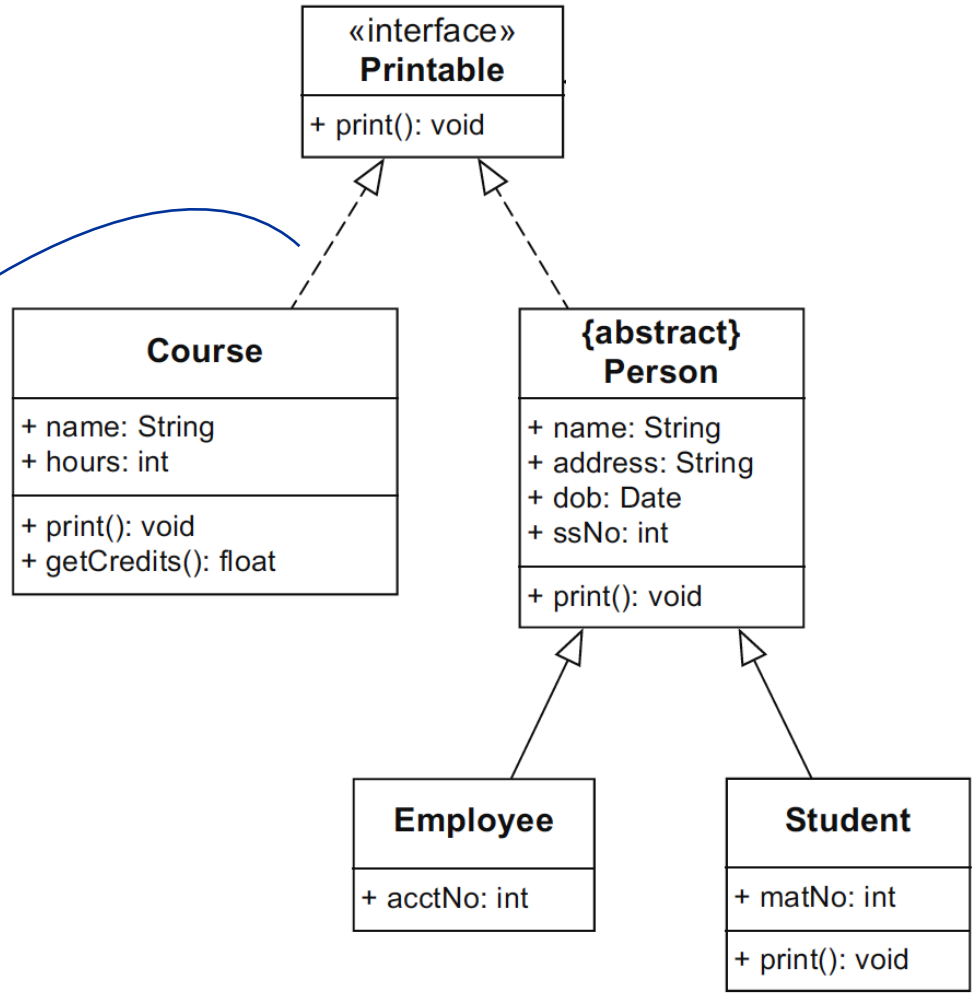
- A class can inherit from multiple classes



Interface

- An *interface* is denoted like a class but with the additional keyword «interface» before the name.

dashed line for interface



An Example

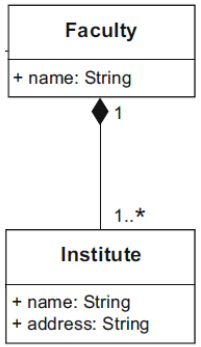
- A university consists of multiple faculties which are composed of various institutes. Each faculty and each institute has a name. An address is known for each institute.
- Each faculty is led by a dean, who is an employee of the university.
- The total number of employees is known. Employees have a social security number, a name, and an e-mail address. There is a distinction between research and administrative personnel.
- Research associates are assigned to at least one institute. The field of study of each research associate is known. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. Some research associates teach courses. They are called lecturers.
- Courses have a unique number (ID), a name, and a weekly duration in hours.

An Example

- A university consists of multiple faculties which are composed of various institutes. Each faculty and each institute has a name. An address is known for each institute.
- Each faculty is led by a dean, who is an employee of the university.
- The total number of employees is known. Employees have a social security number, a name, and an e-mail address. There is a distinction between research and administrative personnel.
- Research associates are assigned to at least one institute. The field of study of each research associate is known. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. Some research associates teach courses. They are called lecturers.
- Courses have a unique number (ID), a name, and a weekly duration in hours.

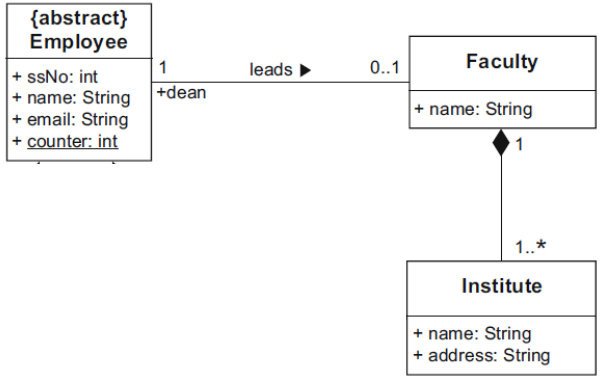
An Example

- A university consists of multiple faculties which are composed of various institutes. Each faculty and each institute has a name. An address is known for each institute.



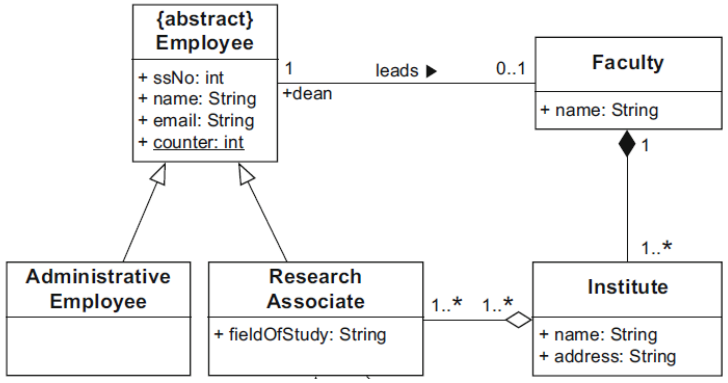
An Example

- Each faculty is led by a dean, who is an employee of the university.



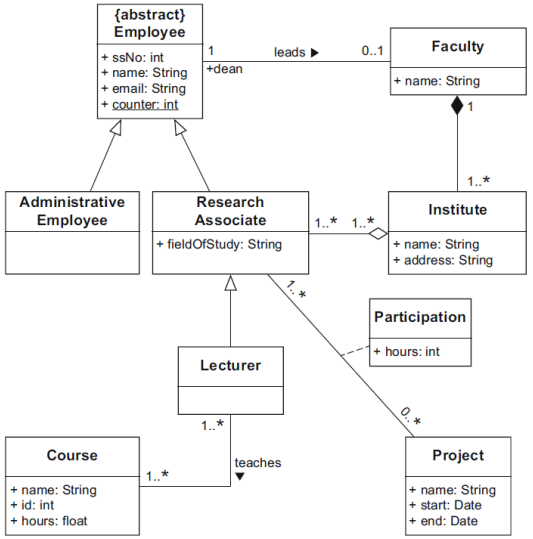
An Example

- The total number of employees is known. Employees have a social security number, a name, and an e-mail address. There is a distinction between research and administrative personnel.
- Research associates are assigned to at least one institute.



An Example

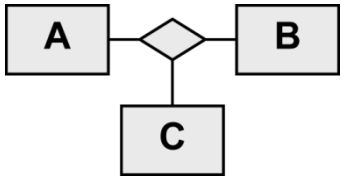
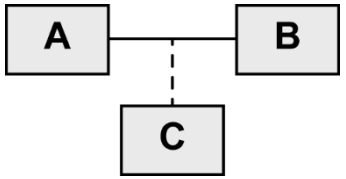
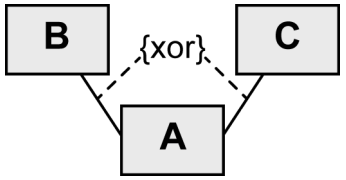
- The field of study of each research associate is known. Furthermore, research associates can be involved in projects for a certain number of hours, and the name, starting date, and end date of the projects are known. Some research associates teach courses. They are called lecturers.
- Courses have a unique number (ID), a name, and a weekly duration in hours.



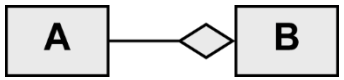
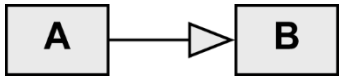
Notation Element

Name	Notation	Description
Class	A – a1: T1 – a2: T2 + o1(): void + o2(): void	Description of the structure and behavior of a set of objects
Abstract class	*A* {abstract} A	Class that cannot be instantiated
Association	A — B A ↔ B A ×→ B	Relationship between classes: navigability unspecified, navigable in both directions, not navigable in one direction

Notation Element (2)

Name	Notation	Description
n-ary Association		Relationship between n (here 3) classes
Association class		More detailed description of an association
`xor` relationship		An object of **`C`** is in a relationship with an object of **`A`** or with an object of **`B`** but not with both

Notation Element (3)

Name	Notation	Description
Shared aggregation		Parts-whole relationship (**A** is part of **B**)
Strong aggregation = composition		Existence-dependent parts-whole relationship (**A** is part of **B**)
Generalization		Inheritance relationship (**A** inherits from **B**)