Programmering og Problemløsning

16 December 2016

Christina Lioma
c.lioma@di.ku.dk

Today’s lecture

UML diagrams

- What they are
- UML diagrams of classes
- UML diagrams of class relations

Today's lecture

UML diagrams
- What they are
- UML diagrams of classes
- UML diagrams of class relations

Beginning Java Objects, Jacquie Barker pp. 355 – 407

UML: Unified Modeling Language

UML: Unified Modeling Language

Graphical language for communicating system design

UML: Unified Modeling Language

Graphical language for communicating system design

Industry standard for OO design notation

UML: Unified Modeling Language

Graphical language for communicating system design

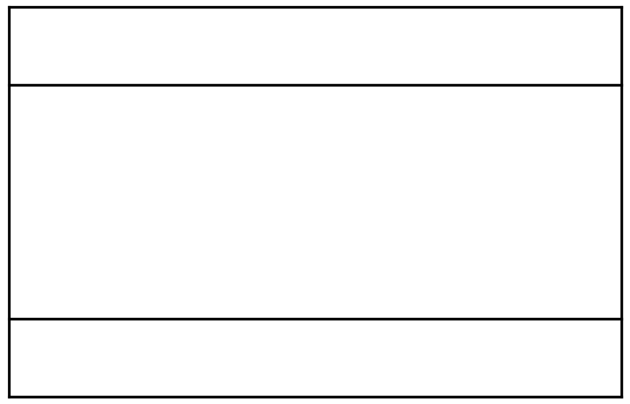
Industry standard for OO design notation

http://www.umlet.com

UML in latex (labs)

UML diagram of a class

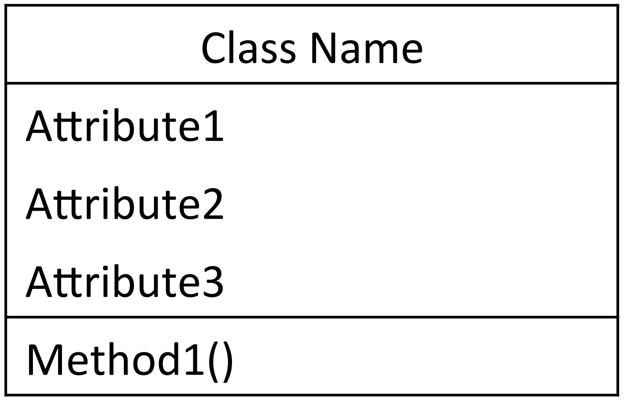
Class: rectangle split into 3 horizontal parts



UML diagram of a class

Class: rectangle split into 3 horizontal parts

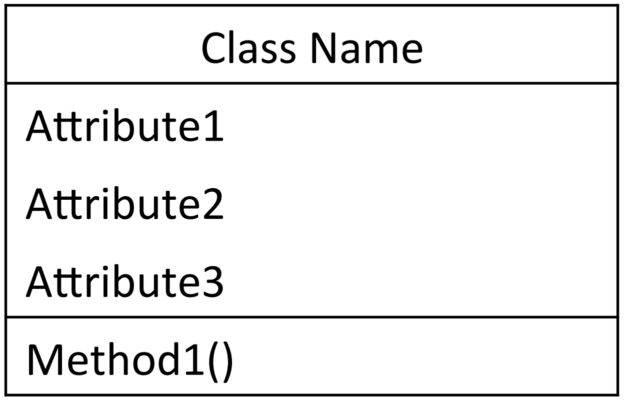
- Class name
- Attributes (one per row)
- Methods (one per row)



UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
- Methods (one per row)

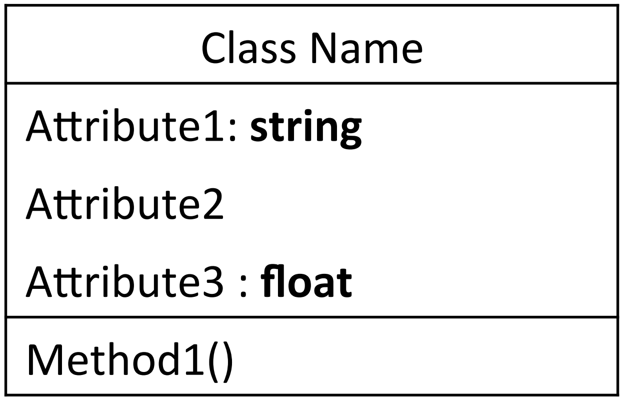


(Methods are also referred to as *operations* in UML)

UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
  - Optional: data type
- Methods (one per row)

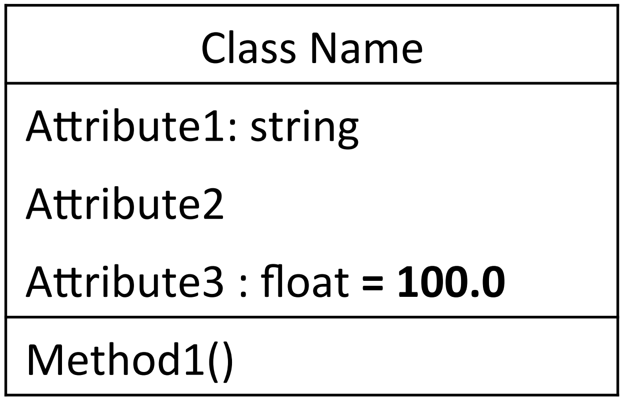


(Methods are also referred to as *operations* in UML)

UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
  - Optional: data type
  - Optional: initial value
- Methods (one per row)

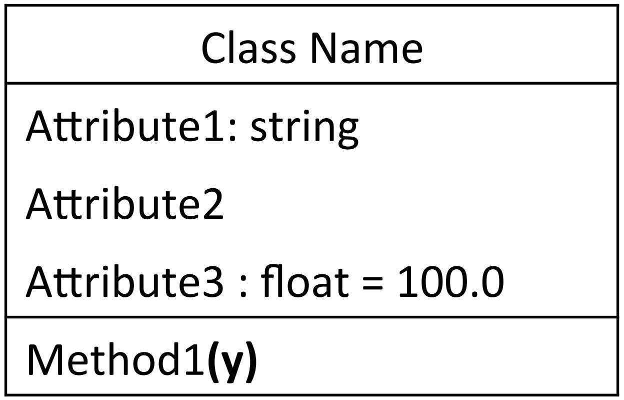


(Methods are also referred to as *operations* in UML)

UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
  - Optional: data type
  - Optional: initial value
- Methods (one per row)
  - Optional: parameter list

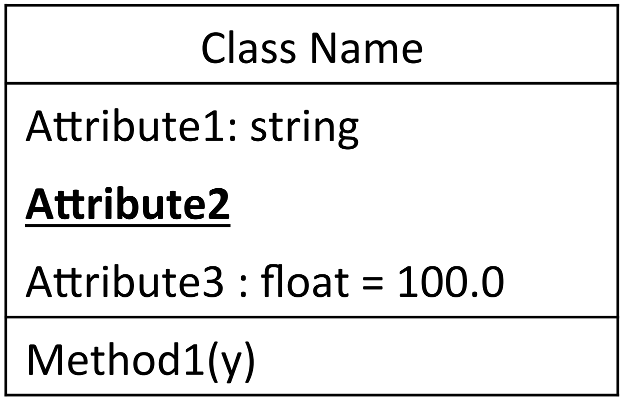


(Methods are also referred to as *operations* in UML)

UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
  - Optional: data type
  - Optional: initial value
- Methods (one per row)
  - Optional: parameter list



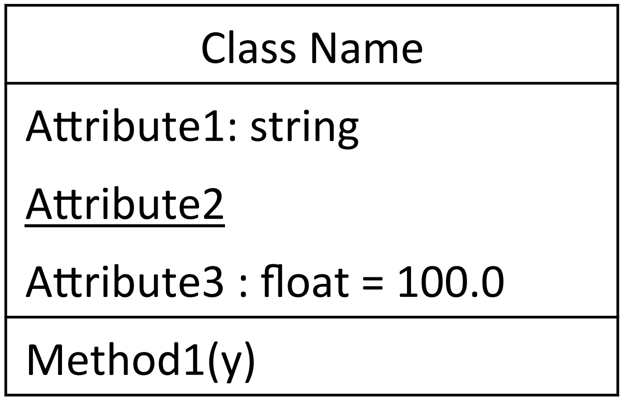
Static members are underlined

(Methods are also referred to as *operations* in UML)

UML diagram of a class

Class: rectangle split into 3 horizontal parts

- Class name
- Attributes (one per row)
  - Optional: data type
  - Optional: initial value
- Methods (one per row)
  - Optional: parameter list



Static members are underlined

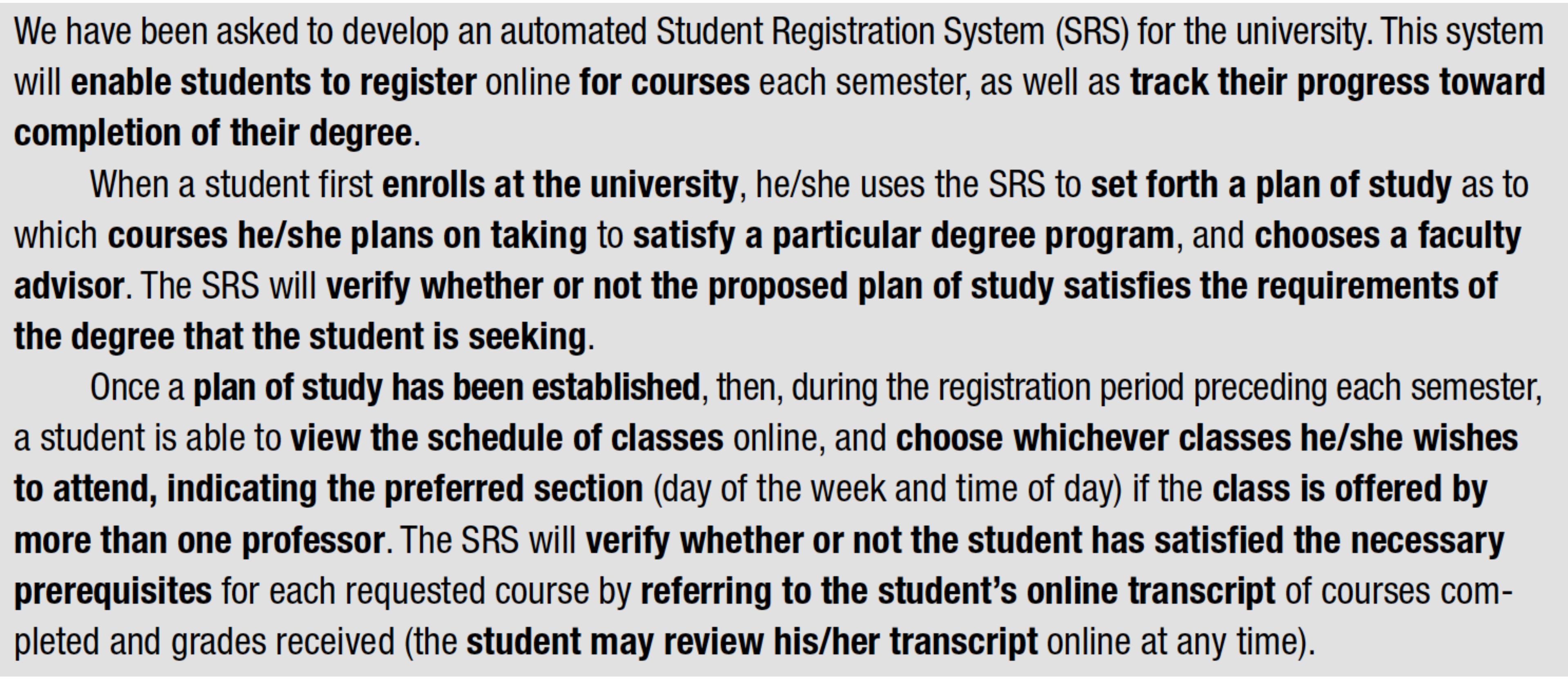
Include *only important* attributes and methods, e.g. not get() set()

(Methods are also referred to as *operations* in UML)

UML diagrams of a class relations

- What are class relations
- How to draw their UML diagrams

Example (p. 369)



We have been asked to develop an automated Student Registration System (SRS) for the university. This system will **enable students to register** online **for courses** each semester, as well as **track their progress toward completion of their degree**.

When a student first **enrolls at the university**, he/she uses the SRS to **set forth a plan of study** as to which **courses he/she plans on taking** to **satisfy a particular degree program**, and **chooses a faculty advisor**. The SRS will **verify whether or not the proposed plan of study satisfies the requirements of the degree that the student is seeking**.

Once a **plan of study has been established**, then, during the registration period preceding each semester, a student is able to **view the schedule of classes** online, and **choose whichever classes he/she wishes to attend, indicating the preferred section** (day of the week and time of day) if the **class is offered by more than one professor**. The SRS will **verify whether or not the student has satisfied the necessary prerequisites** for each requested course by **referring to the student's online transcript** of courses completed and grades received (the **student may review his/her transcript** online at any time).

Program to process university records

Abstract object classes

University

School

Course

Plan of study

Student

Professor

Transcript

Program to process university records

Abstract object classes

University: consists of Schools (of Science, Humanities, …)

School: belongs to the university; employs people…

Course: taught by professors; belongs to plan of study…

Plan of study: consists of courses; followed by students…

Student: studies at the university; follows courses…

Professor: teaches students; works at the university…

Transcript: record of all courses and grades per student

Program to process university records

Abstract object classes **are often related**

University: consists of **Schools** (of Science, Humanities, …)

School: belongs to the **university**; employs people…

Course: taught by **professors**; belongs to **plan of study**…

Plan of study: consists of **courses**; followed by **students**…

Student: studies at the **university**; follows **courses**…

Professor: teaches **students**; works at the **university**…

Transcript: record of all **courses** and grades per **student**

Program to process university records

Abstract object classes **are often related**

University: consists of **Schools** (of Science, Humanities, …)

School: belongs to the **university**; employs **people**…

Course: taught by **professors**; belongs to **plan of study**…

Plan of study: consists of **courses**; followed by **students**…

Student: studies at the **university**; follows **courses**…

Professor: teaches **students**; works at the **university**…

Transcript: record of all **courses** and grades per **student**

People: students; professors…

Inheritance

Relations between classes (e.g. inheritance) are called *Associations*

Relations between classes (e.g. inheritance) are called *Associations*

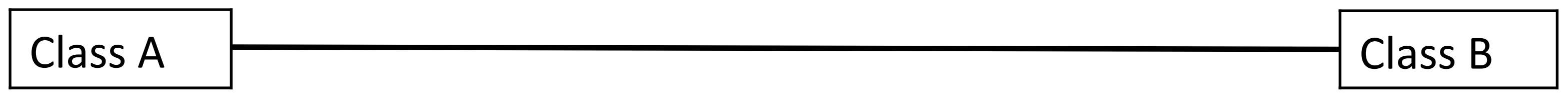
We use UML diagrams to represent the associations between our classes

Class A

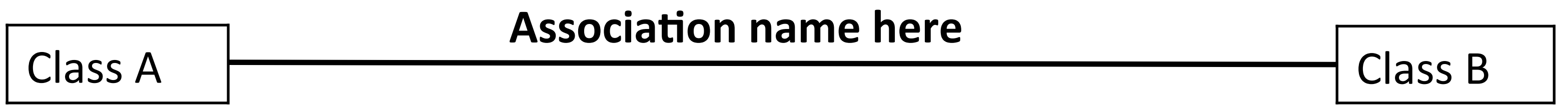
Class B

Solid line: relationship between classes

Association between classes

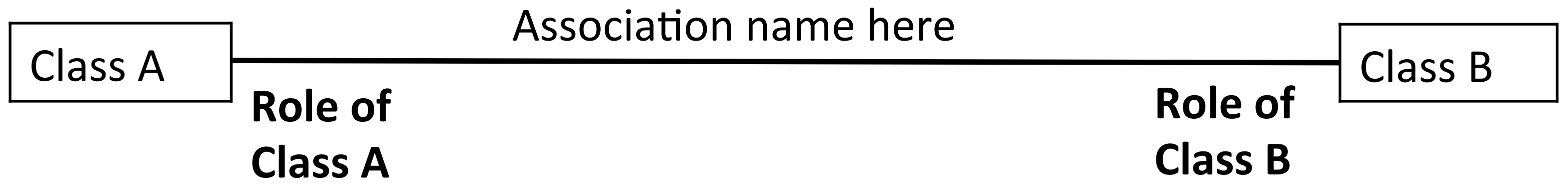


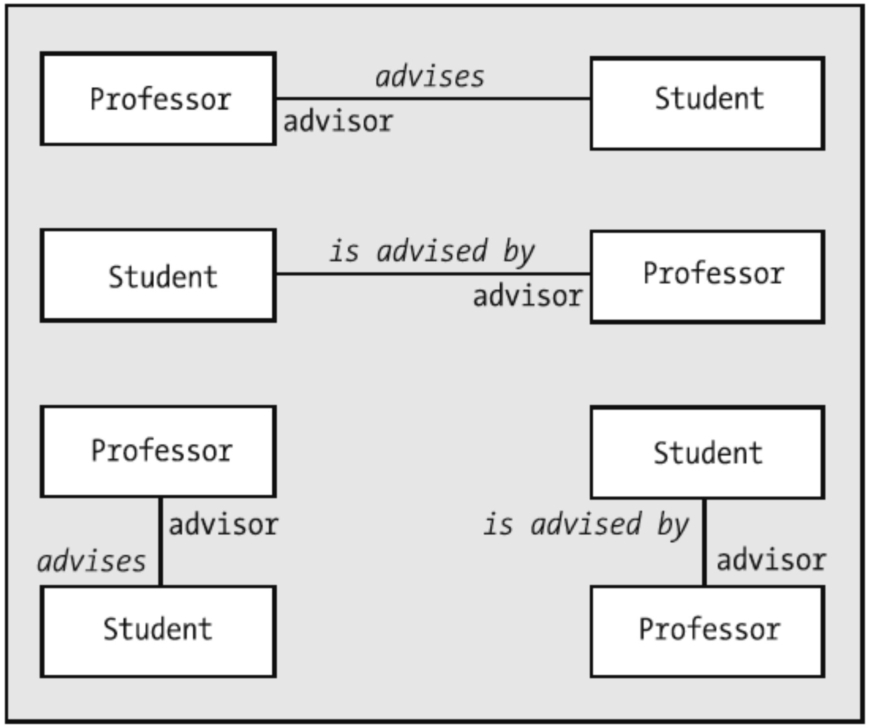
Solid line: relationship between classes (can be named)

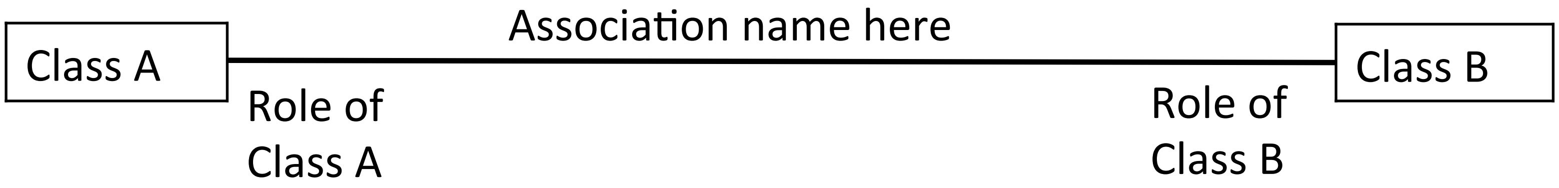


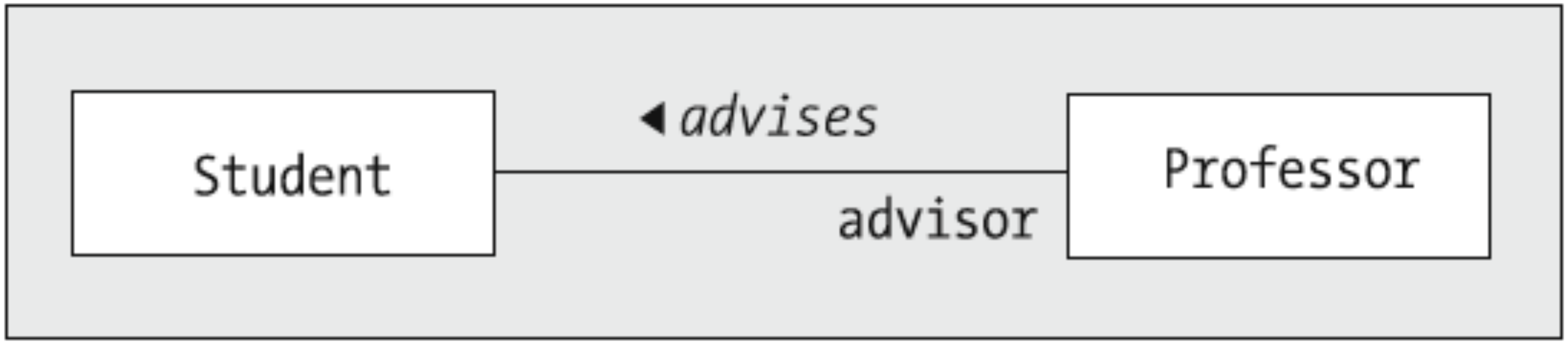
Solid line: relationship between classes (can be named)

Within the association classes may have roles

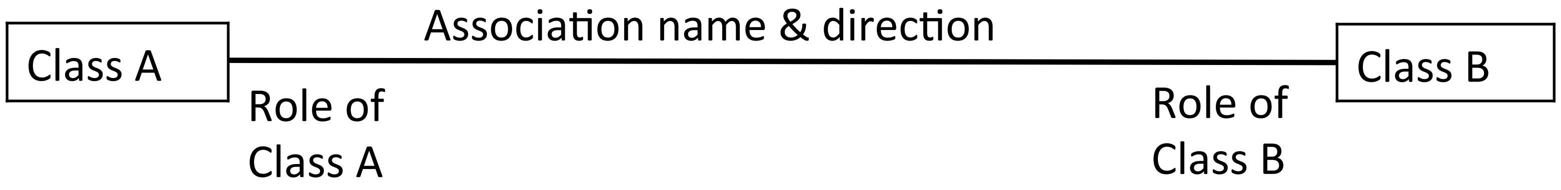



Professor
advises
advisor
Student
Student
is advised by
advisor
Professor
Professor
advisor
advises
Student
Student
is advised by
advisor
Professor


Class A
Association name here
Class B
Role of Class A
Role of Class B



Small arrowhead reflects the direction of the association

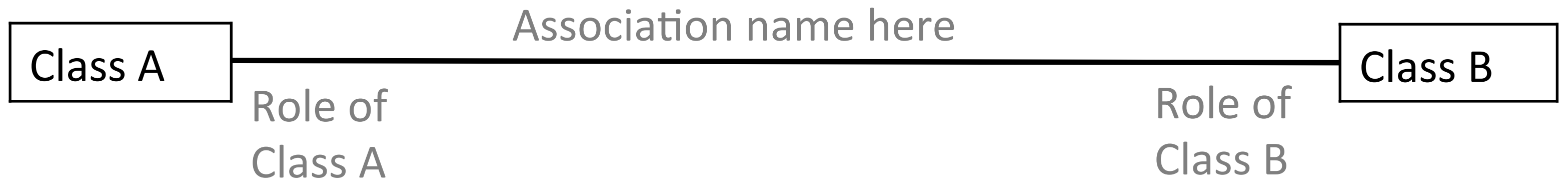


Solid line: relationship between classes

Within the association classes may have roles

Association name & roles are optional in UML

(use only when needed to clarify abstraction)



Three main types of associations (relations between classes):

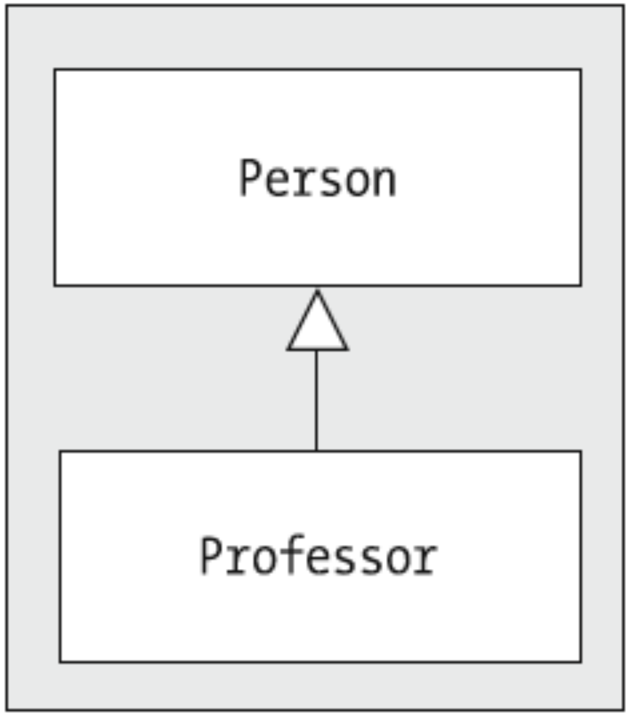
Inheritance

Aggregation

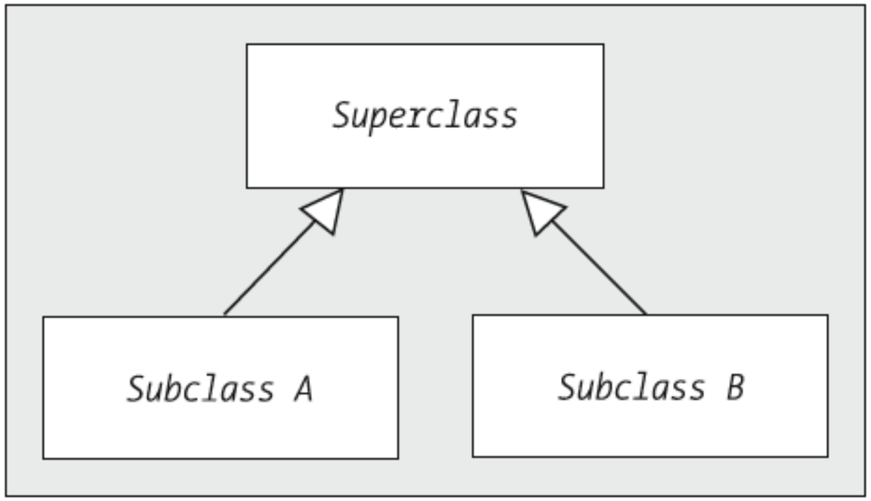
Composition

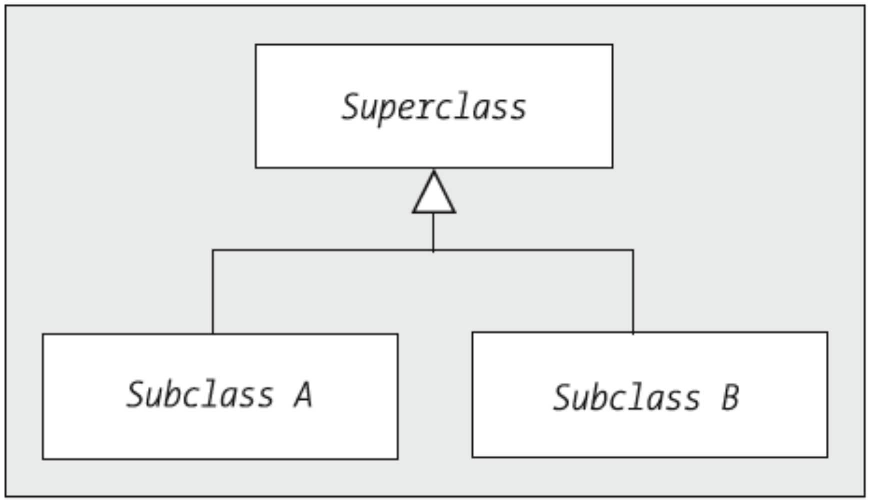
1. Inheritance (class B is a type of class A)

UML: *white arrow* points to the Base class



Equivalent UML representations of inheritance





2. Aggregation (class A contains class B):

A university contains faculties, schools, departments

2. Aggregation (class A contains class B):

A university contains faculties, schools, departments

Difference between inheritance and aggregation:

2. Aggregation (class A contains class B):

A university contains faculties, schools, departments

Difference between inheritance and aggregation:

A university contains a department	*AGGREG*
A department is not a type of university	~~*INHERIT*~~

2. Aggregation (class A contains class B):

A university contains faculties, schools, departments

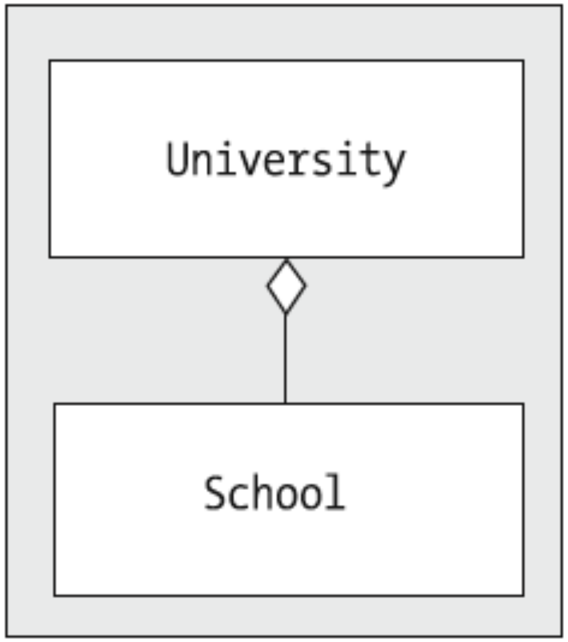
Difference between inheritance and aggregation:

A university contains a department	*AGGREG*
A department is not a type of university	*~~INHERIT~~*

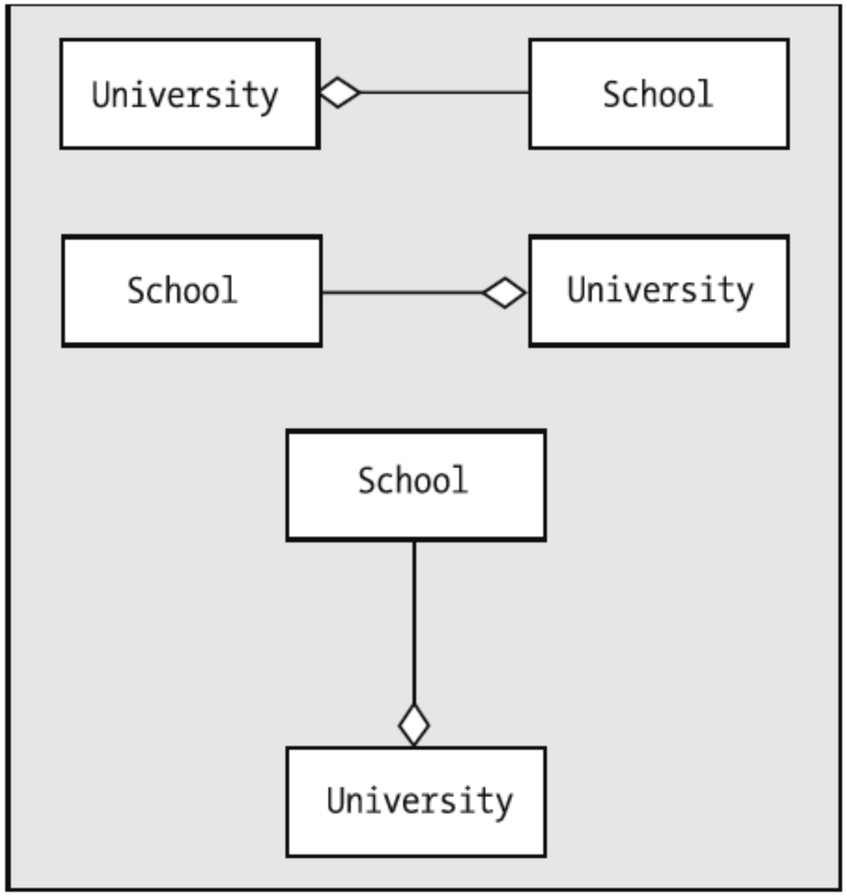
A student is a type of person	*INHERIT*
A person does not contain students	*~~AGGREG~~*

2. Aggregation (class A contains class B)

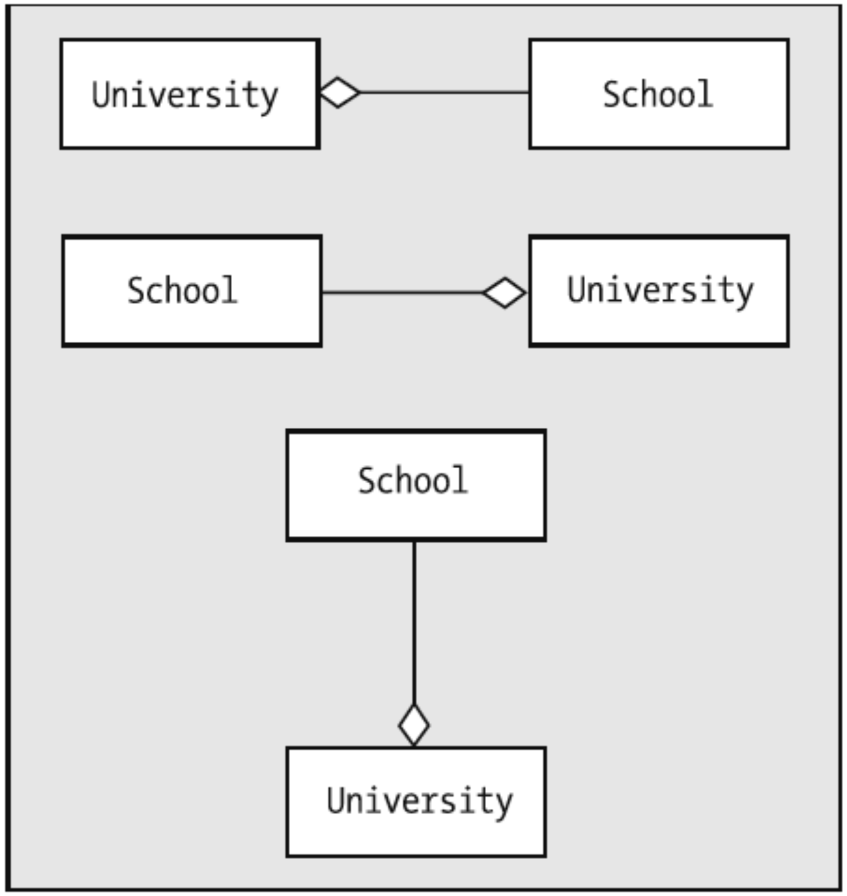
UML: *diamond* points to the containing class

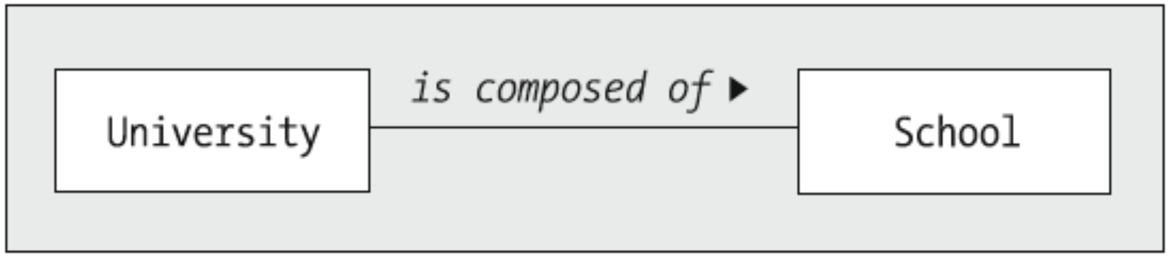


Any orientation

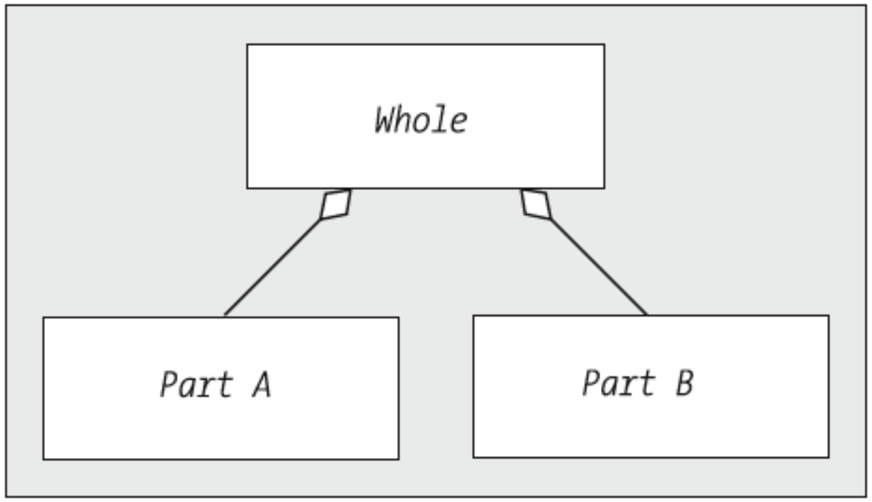


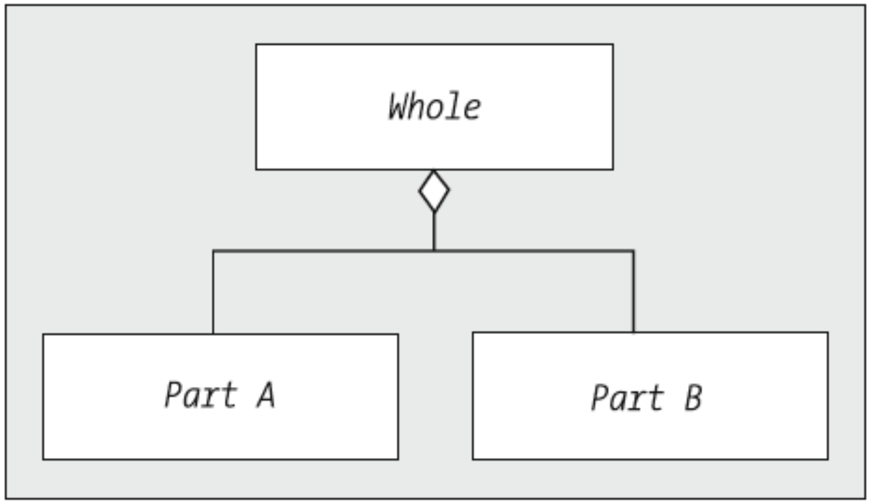
Equivalent


University
School
School
University
School
University


University
is composed of ▸
School

Equivalent UML representations for aggregation





3. Composition (strong case of aggregation)

3. Composition (strong case of aggregation):

class A contains class B and

class B cannot exist without class A

3. Composition (strong case of aggregation):

class A contains class B and

class B cannot exist without class A

A book contains chapters

(chapters cannot exist without books)

3. Composition (strong case of aggregation):

class A contains class B and

class B cannot exist without class A

A book contains chapters

(chapters cannot exist without books)

A car contains wheels

(wheels exist without cars)

3. Composition (strong case of aggregation):

class A contains class B and

class B cannot exist without class A

A book contains chapters	*COMPOS*
(chapters cannot exist without books)	*~~AGGREG~~*

A car contains wheels	*AGGREG*
(wheels exist without cars)	*~~COMPOS~~*

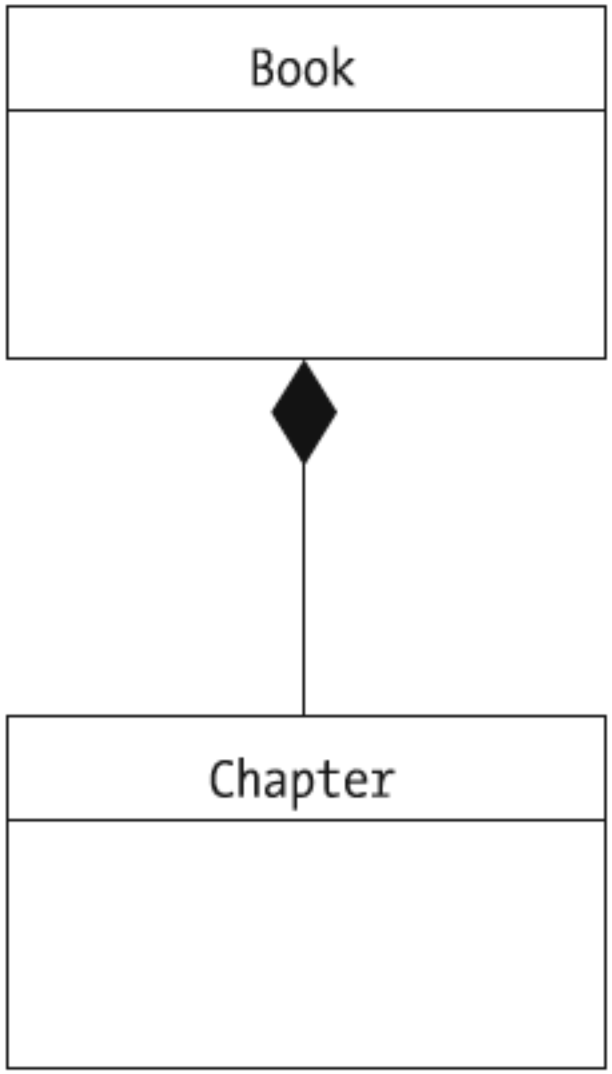
3. Composition (strong case of aggregation):

class A contains class B and

class B cannot exist without class A

A book contains chapters	*COMPOS*
(chapters cannot exist without books)	*~~AGGREG~~*
(chapter is not a type of book)	*~~INHERIT~~*
A car contains wheels	*AGGREG*
(wheels exist without cars)	*~~COMPOS~~*
(wheels are not a type of car)	*~~INHERIT~~*

3. Composition in UML: black diamond points to the containing class



Class associations in UML

Inheritance: *white arrow*

class B is a type of class A

Aggregation: *diamond*

class A contains class B

Composition: *black diamond*

class A contains class B

and

class B cannot exist without class A

Class associations in UML

<u>Inheritance</u>: *white arrow*

class B is a type of class A *is-a*

<u>Aggregation</u>: *diamond*

class A contains class B *has-a*

<u>Composition</u>: *black diamond*

class A contains class B *has-a*

and

class B cannot exist without class A

Class associations in UML

Inheritance

Aggregation → ***how*** classes are related

Composition

Class associations in UML

Inheritance

Aggregation → ***how*** classes are related

Composition

Binary

Unary → ***how many*** classes/instances are related

Multiplicity

Binary versus unary class associations

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)
- Unary: between instances of the *same class*

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)
- Unary: between instances of the *same class*

Course "Math" is a prerequisite of course "CompSci"
(Math and CompSci are *different instances of the class* Course)

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)
- Unary: between instances of the *same class*

Course "Math" is a prerequisite of course "CompSci"
(Math and CompSci are *different instances of the class* Course)

Student X represents student Y in the council
(X and Y are *different instances of the class* Student)

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)
- Unary: between instances of the *same class*

Course “Math” is a prerequisite of course “CompSci”
(Math and CompSci are *different instances of the class* Course)

Student X represents student Y in the council
(X and Y are *different instances of the class* Student)
But student X represents also himself in the council
(X-representative and X-student are the *same instance of the class* Student)

Binary versus unary class associations:

- Binary: between two classes (what we have seen so far)
- Unary: between **(different or the same)** instances of the *same class*

Course “Math” is a prerequisite of course “CompSci”

(Math and CompSci are *different instances of the class* Course)

Student X represents student Y in the council

(X and Y are *different instances of the class* Student)

But student X represents also himself in the council

(X-representative and X-student are the *same instance of the class* Student)

Binary versus unary class associations:

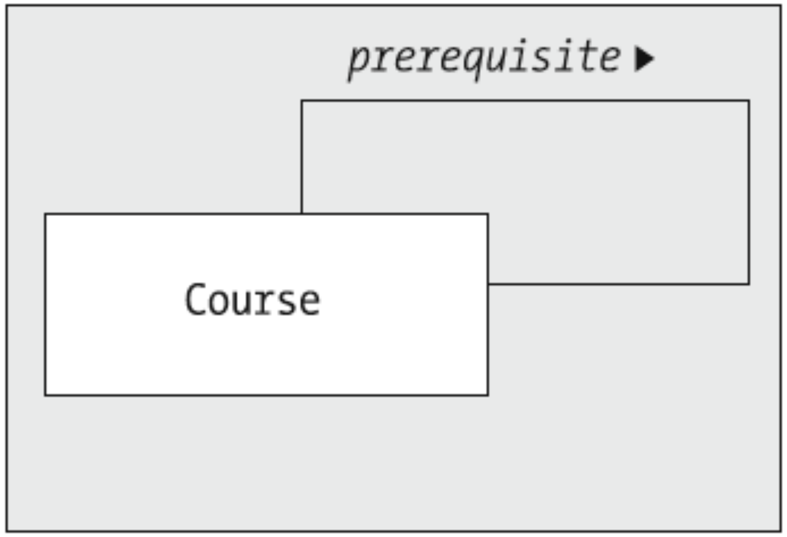
- Binary: between two classes (what we have seen so far)
- Unary: between **(different or the same)** instances of the *same class*

Course “Math” is a prerequisite of course “CompSci”
(Math and CompSci are *different instances of the class* Course)

Student X represents student Y in the council
(X and Y are *different instances of the class* Student)
But student X represents also himself in the council
(X-representative and X-student are the *same instance of the class* Student)

How to represent unary associations in UML?

Unary associations in UML
(between instances of the *same* class)

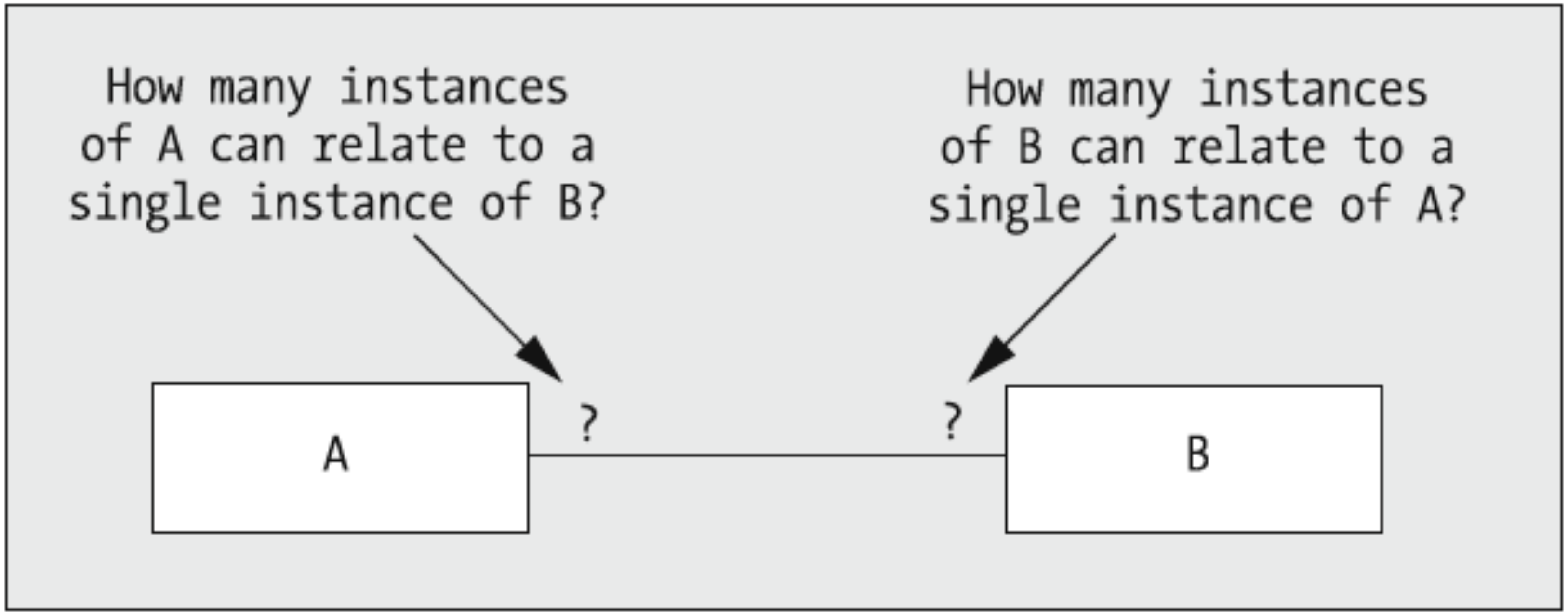


Association multiplicity

How many instances of class A can be associated with an instance of class B

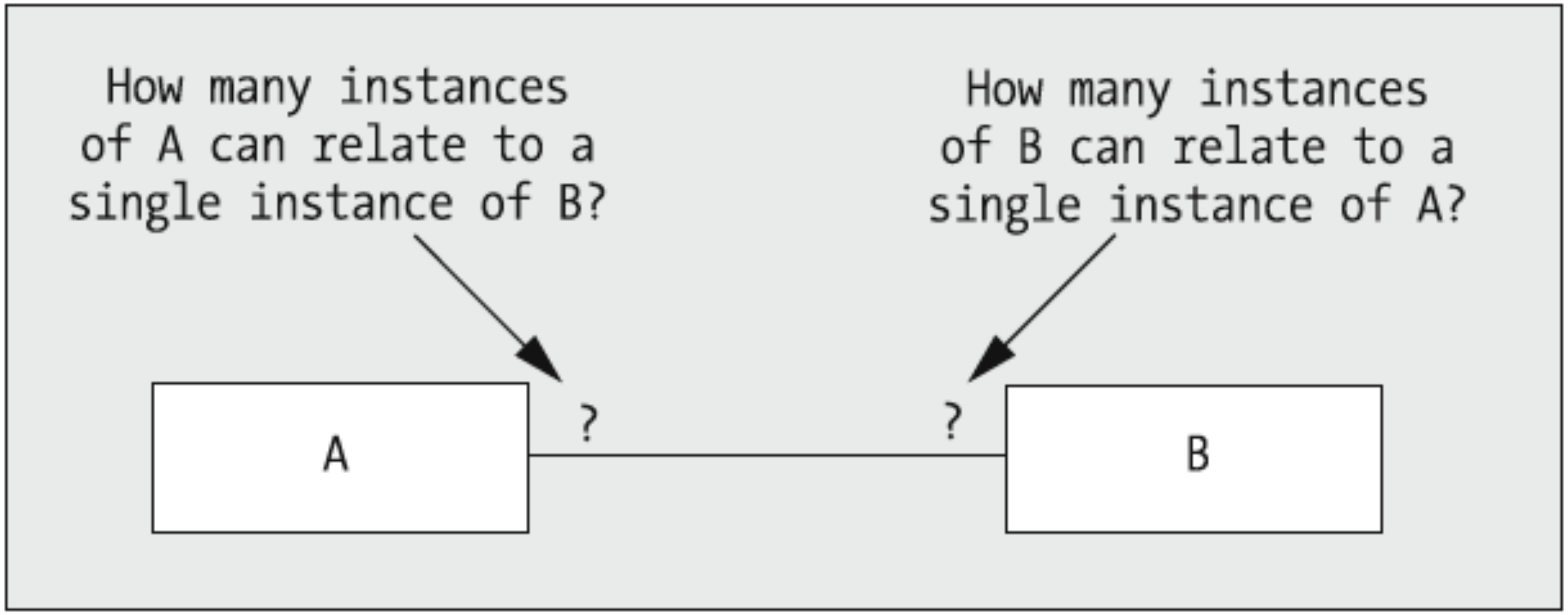
Association multiplicity

How many instances of class A can be associated with an instance of class B


How many instances of A can relate to a single instance of B?
How many instances of B can relate to a single instance of A?
?
?
A
B

Association multiplicity

How many instances of class A can be associated with an instance of class B

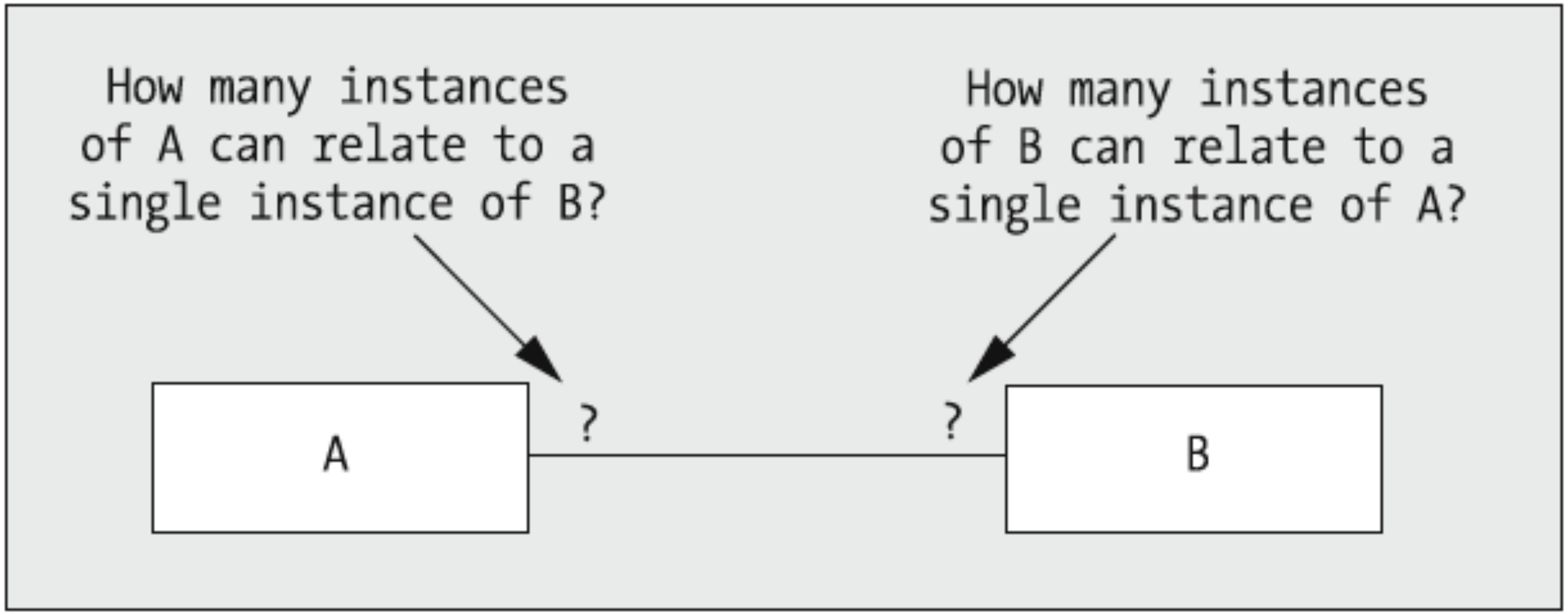

How many instances of A can relate to a single instance of B?
How many instances of B can relate to a single instance of A?
?
?
A
B

Exactly one: 1

Several: *

Association multiplicity

How many instances of class A can be associated with an instance of class B

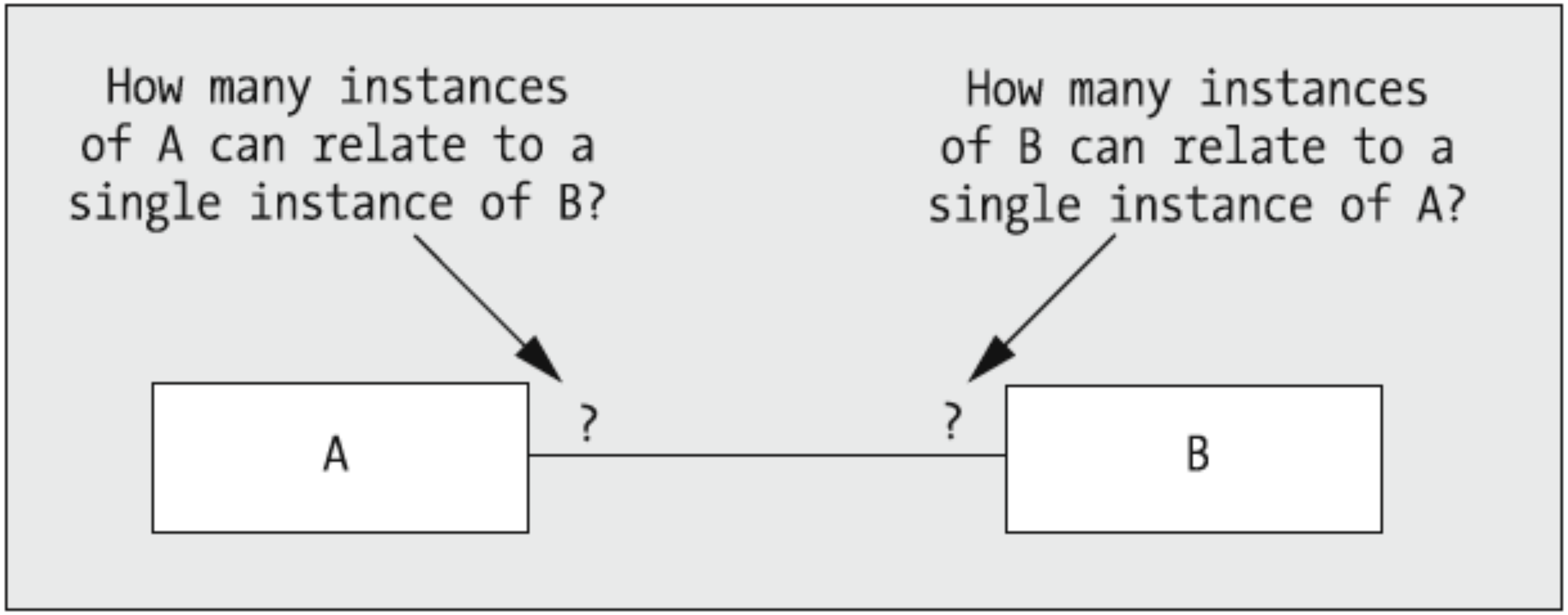

How many instances of A can relate to a single instance of B?
How many instances of B can relate to a single instance of A?
?
?
A
B

Exactly one: 1
From e.g. 3 to 7: 3..7

Several: *

Association multiplicity

How many instances of class A can be associated with an instance of class B

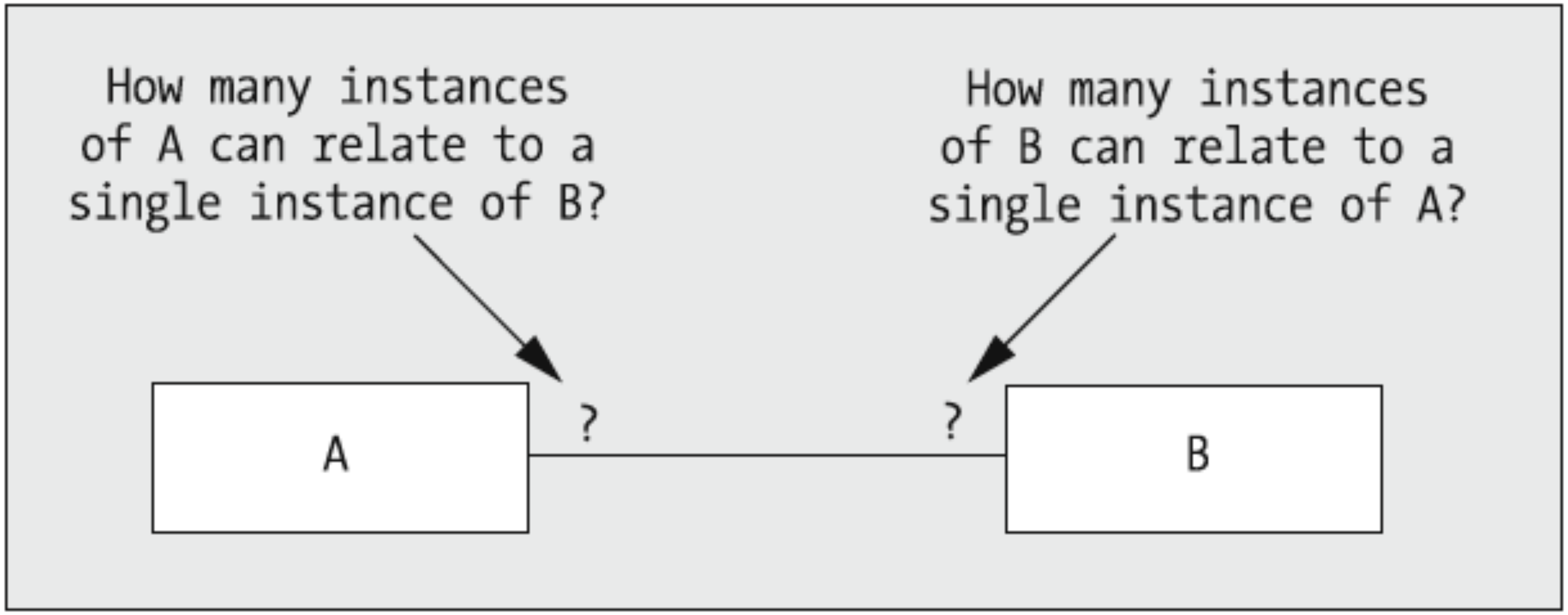

How many instances of A can relate to a single instance of B?
How many instances of B can relate to a single instance of A?
?
?
A
B

Exactly one: 1
From e.g. 3 to 7: 3..7

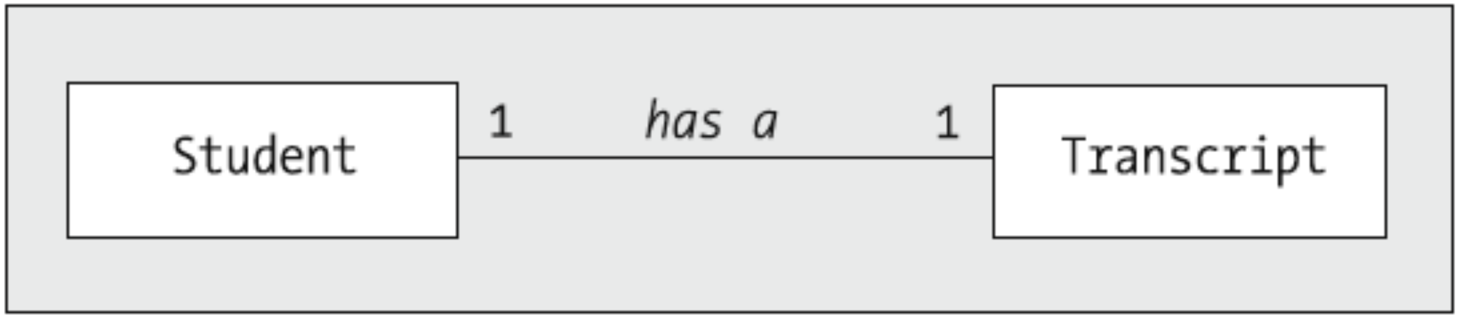
Several: *
Zero or more: 0..*

Association multiplicity

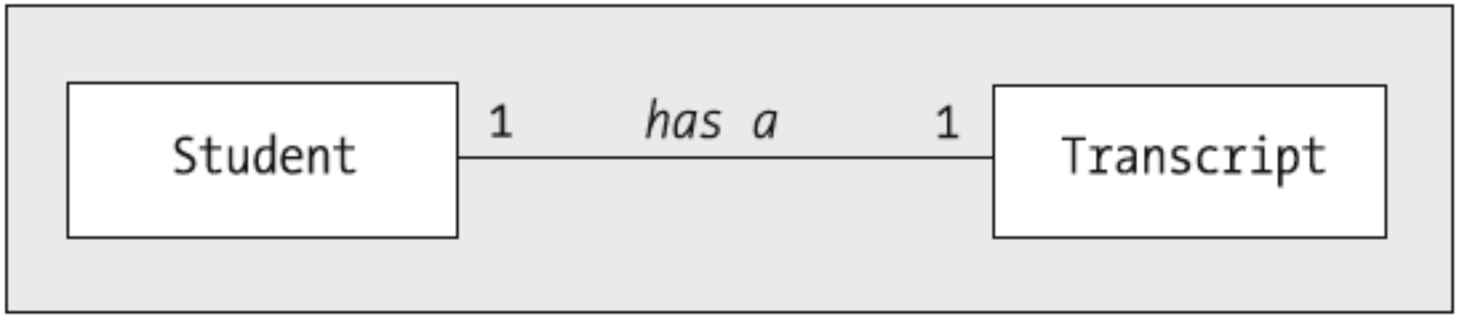
How many instances of class A can be associated with an instance of class B


How many instances of A can relate to a single instance of B?
How many instances of B can relate to a single instance of A?
?
?
A
B

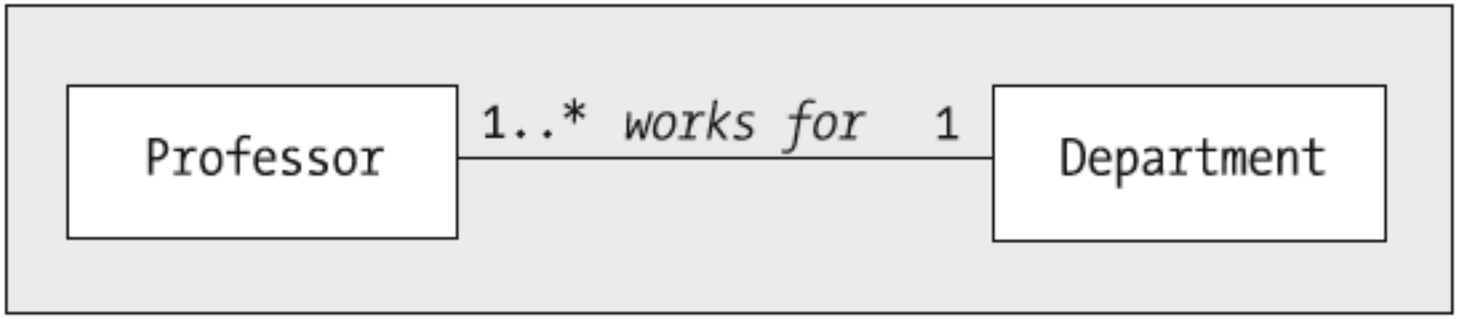
Exactly one: 1	Several: *
From e.g. 3 to 7: 3..7	Zero or more: 0..*
One or more: 1..*	At most one: 0..1



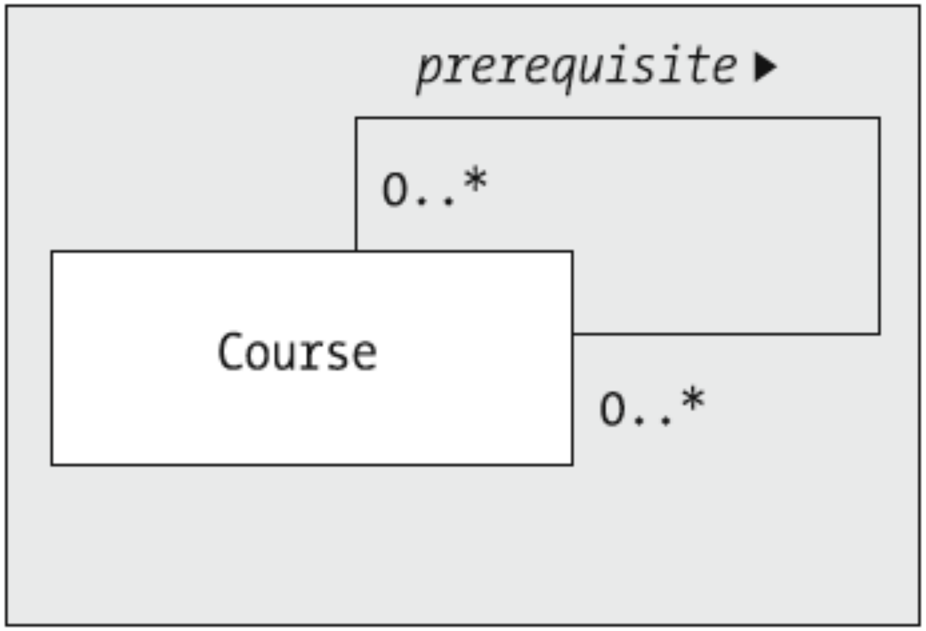
A student has **one** transcript and a transcript belongs to **one** student



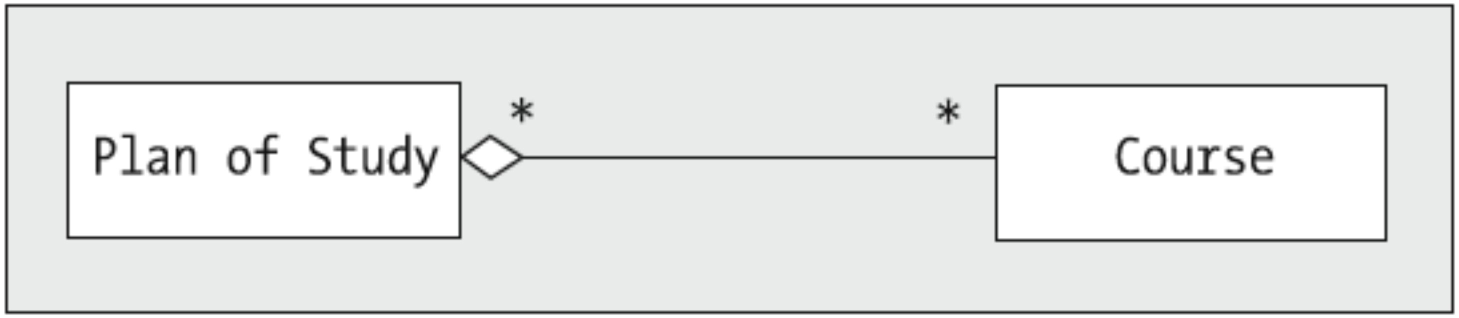
A student has **one** transcript and a transcript belongs to **one** student



A professor works for **one** department but a department has **many** professors



A course can be a prerequisite for **zero or more** courses and a course can have **zero or more** prerequisites

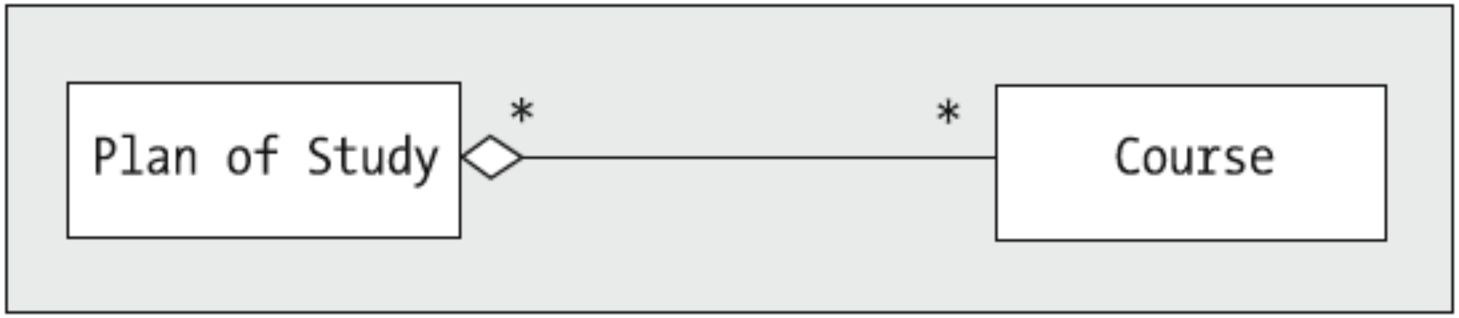


A study plan is composed of **several** courses

and

a course can be included in **several** study plans

(Aggregation & multiplicity)



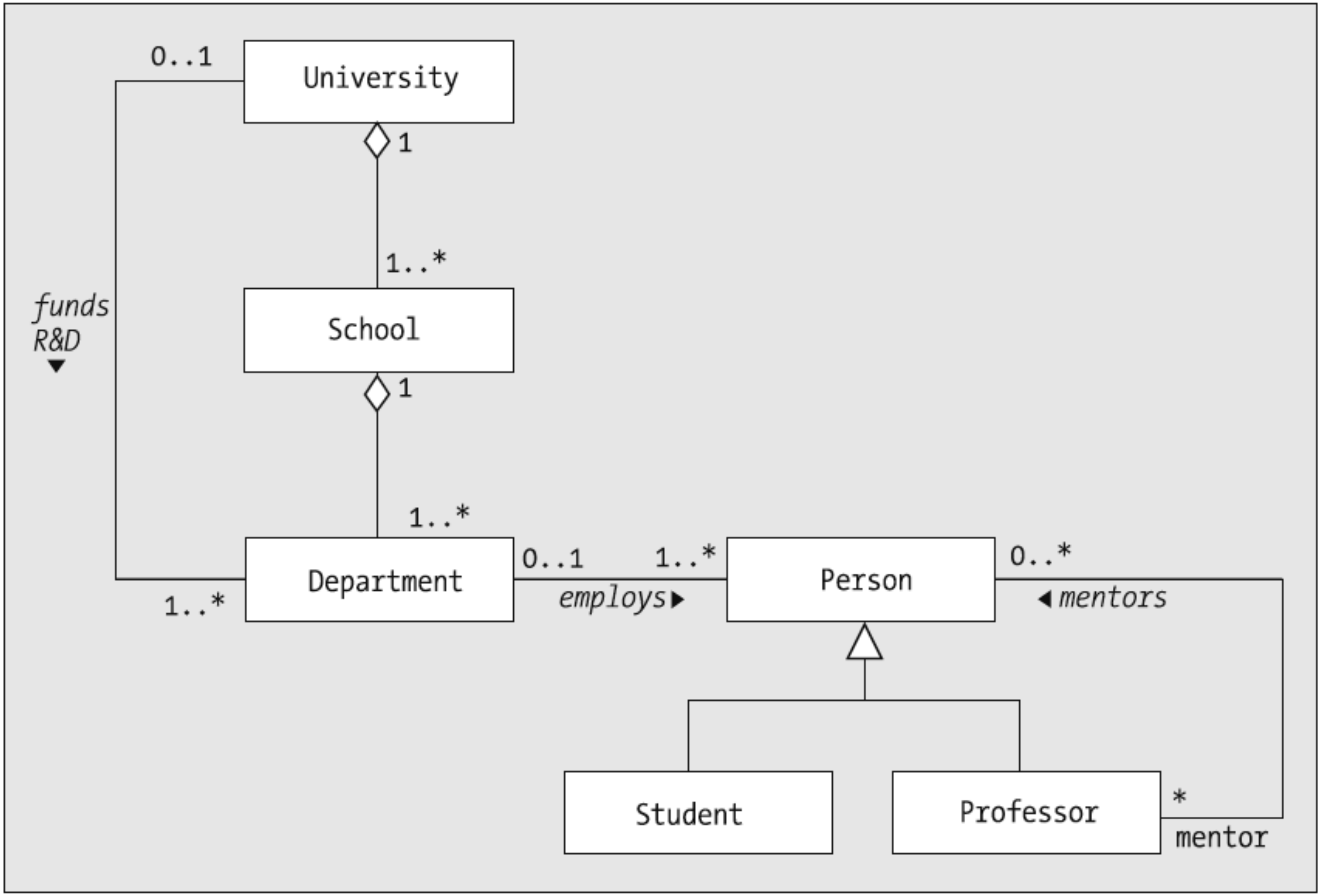
A study plan is composed of **several** courses

and

a course can be included in **several** study plans

(Aggregation & multiplicity)

Why not composition?


0..1
University
1
1..*
funds R&D ▼
School
1
1..*
Department
1..*
0..1
1..*
employs ▶
Person
0..*
◀ mentors
Student
Professor
*
mentor

UML best practice

- Less is more (legible when printed on A4)

UML best practice

- Less is more (legible when printed on A4)
- Orthogonality (right angles)

UML best practice

- Less is more (legible when printed on A4)
- Orthogonality (right angles)
- Parents up

UML best practice

- Less is more (legible when printed on A4)
- Orthogonality (right angles)
- Parents up
- Align elements when possible
- Make elements the same size when possible

Recap today’s lecture

- Unified Modeling Language (UML)
  - Classes
  - Class relations
  - Instance relations