

Inf2-SEPP: Lecture 9 Part 2: UML class diagrams

Adriana Sejfia

School of Informatics
University of Edinburgh

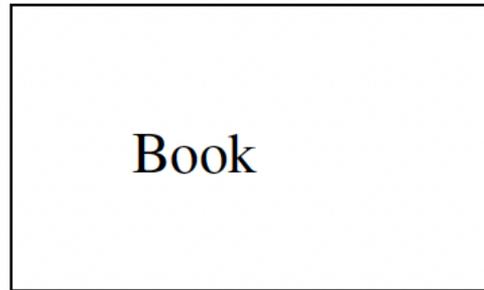
This lecture

- Class diagrams
 - Notation:
 - Classes: representation, attributes and operations
 - Associations: representation, rolenames, multiplicity, navigability
 - Generalisation
 - Interfaces
 - Abstract classes and operations
 - Static attributes and operations
 - An approach for identifying objects and classes

Class diagrams: introduction

- Part of UML
- Used to represent the static design of the system, i.e. classes and their relationships.
- **Important! As for use case diagrams, we must respect the notation imposed by UML, which has a (fairly) precise meaning.**

Representing classes



Identifying classes from a system description

Simplest and best: look for noun phrases!

Identifying classes from a system description

Simplest and best: look for noun phrases!

Then abandon things which are:

- redundant
- outside scope
- vague
- attributes
- operations and events
- implementation classes.

(May need to add some back later, especially implementation classes: point is to avoid incorporating premature design decisions into your conceptual level model.)

Identifying classes from a system description: example

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

Identifying classes from a system description: example

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

- **Eliminate:** library, short term loan, member of the library, week, time, item.
- **Left with:** book, journal, copy (of book), library member, member of staff.

Identifying classes from a system description: example

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

- **Eliminate:** library, short term loan, member of the library, week, time, item.
- **Left with:** book, journal, copy (of book), library member, member of staff.

After talking to stakeholders, in addition to the system description, they tell you that they want the book object to have a title and when prompted, to show the number of copies available. What do we need to depict in the UML class diagram?

Showing attributes and operations in classes (optional)

Compartments for attributes and operations can be added

Book
title : String
copiesOnShelf() : Integer borrow(c:Copy)

Showing attributes and operations in classes (optional)

Compartments for attributes and operations can be added

Book
title : String
copiesOnShelf() : Integer borrow(c:Copy)

- Syntax for types can be adapted for different programming languages.
- Types and operation argument names are optional.
- Choosing attributes and operations involves conceptual approach (what makes sense in the real world?) and pragmatic approach (what can help meet requirements?)

Showing visibility in classes (optional)

Book
- title : String
+ copiesOnShelf() : Integer # borrow(c:Copy)

Can show whether an attribute or operation is

- public (visible from everywhere) with +
- private (visible only from inside objects of this class) with -
- (Or protected (#), package (~) or other language dependent visibility.)

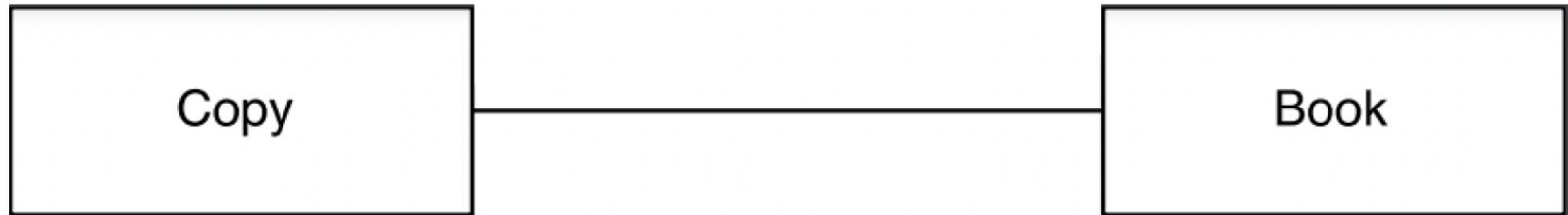
Identifying classes from a system description: example

Books and Journals: The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals.

- **Eliminate:** library, short term loan, member of the library, week, time, item.
- **Left with:** book, journal, copy (of book), library member, member of staff.

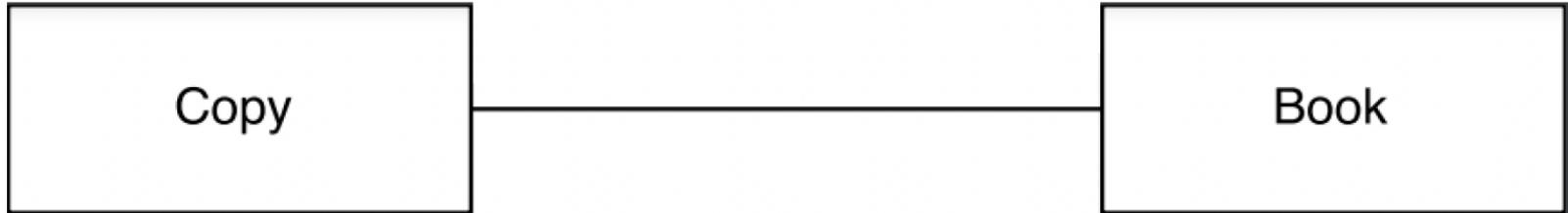
After talking to stakeholders, in addition to the system description, they tell you that they want the book object to have a title **and when prompted, to show the number of copies available**. What do we need to depict in the UML class diagram?

Representing association between classes



Two classes are **associated** if some object of the first has to know about some object of the second.

Representing association between classes



Two classes are **associated** if some object of the first has to know about some object of the second.

- An **object** is an instance of a class.
- A **link** is an instance of an association.
- Each link consists of a pair of objects, each an instance of the class at each end of the association.

E.g. ⟨Copy 3 of War and Peace, War and Peace⟩

Detailed meaning of an association

Classes A and B are associated if:

- An object of class A sends a message to (i.e. calls an operation in) an object of class B

Detailed meaning of an association

Classes A and B are associated if:

- An object of class A sends a message to (i.e. calls an operation in) an object of class B
- An object of class A creates (i.e. calls a constructor of) an object of class B

Detailed meaning of an association

Classes A and B are associated if:

- An object of class A sends a message to (i.e. calls an operation in) an object of class B
- An object of class A creates (i.e. calls a constructor of) an object of class B
- An object of class A has an attribute whose values are objects/ collections of objects of class B

Detailed meaning of an association

Classes A and B are associated if:

- An object of class A sends a message to (i.e. calls an operation in) an object of class B
- An object of class A creates (i.e. calls a constructor of) an object of class B
- An object of class A has an attribute whose values are objects/ collections of objects of class B
- An object of class A receives a message (i.e. performs an operation) with an object of class B as an argument.

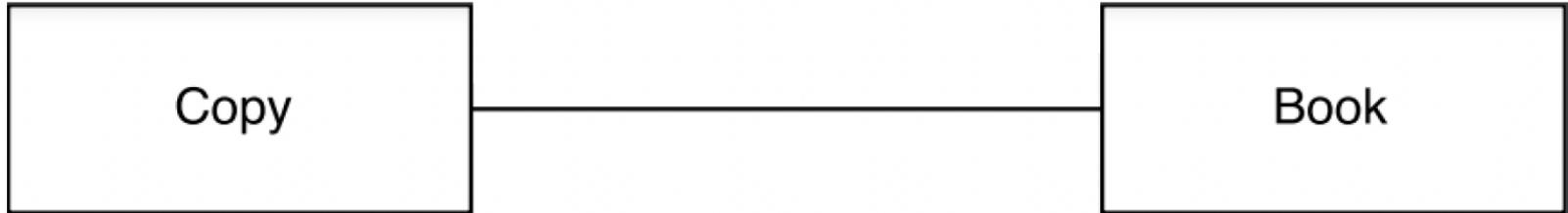
Detailed meaning of an association

Classes A and B are associated if:

- An object of class A sends a message to (i.e. calls an operation in) an object of class B
- An object of class A creates (i.e. calls a constructor of) an object of class B
- An object of class A has an attribute whose values are objects/ collections of objects of class B
- An object of class A receives a message (i.e. performs an operation) with an object of class B as an argument.

Choosing associations also involves a conceptual and a pragmatic approach.

Representing association between classes

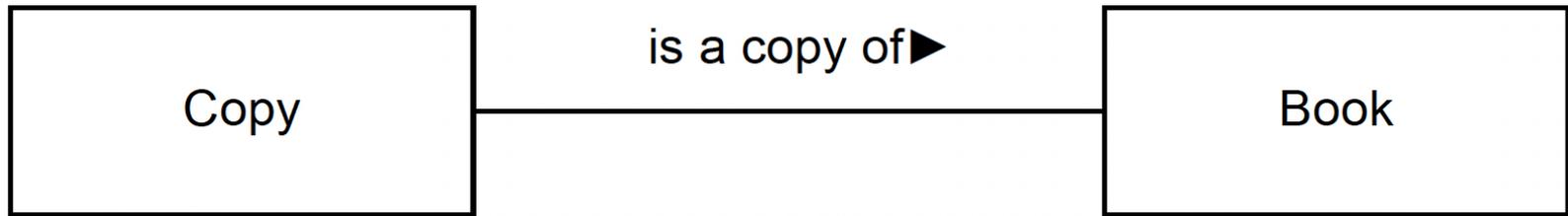


Two classes are **associated** if some object of the first has to know about some object of the second.

- An **object** is an instance of a class.
- A **link** is an instance of an association.
- Each link consists of a pair of objects, each an instance of the class at each end of the association.

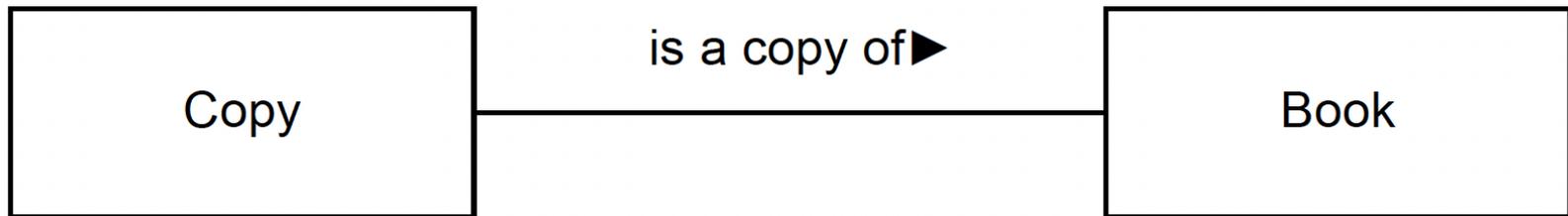
E.g. ⟨Copy 3 of War and Peace, War and Peace⟩

Showing association name and direction (both optional)



The name is recommended, even if optional. It is useful to clarify the relationship between the objects of the classes.

Showing association name and direction (both optional)



The name is recommended, even if optional. It is useful to clarify the relationship between the objects of the classes.

The direction arrow can improve readability, especially in complex diagrams where reading an association may not be left to right.

UML class diagram example

University system: You are building a system for a university. [...] Personal tutors are members of staff. The university students (tutees) each get a personal tutor to help them navigate university life. Each personal tutor must have at least one and up to 30 tutees. Each student must sign up to at least one course. A course can have multiple students and it “must know” about the students enrolled for it so that the students can be notified for any changes.

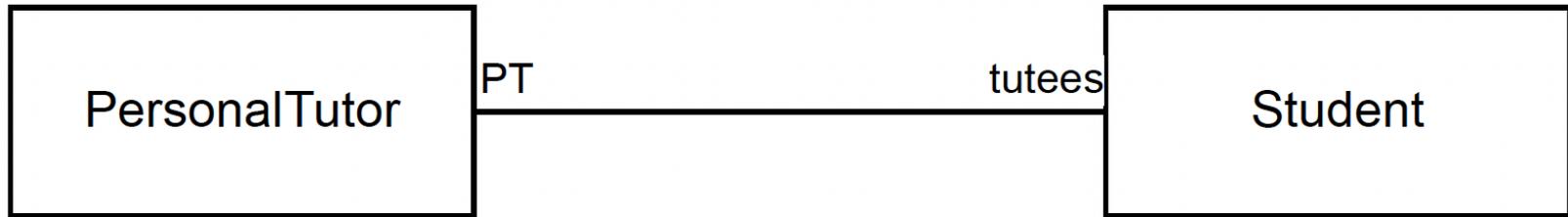


UML class diagram example

University system: You are building a system for a university. [...] **Personal tutors are members of staff. The university students (tutees) each get a personal tutor to help them navigate university life.** Each personal tutor must have at least one and up to 30 tutees. Each student must sign up to at least one course. A course can have multiple students and it “must know” about the students enrolled for it so that the students can be notified for any changes.

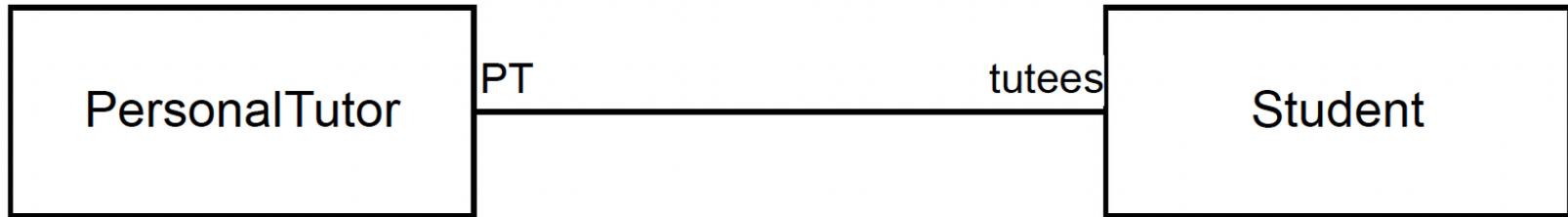
Showing rolenames on associations (optional)

Rolenames show the roles that objects play in an association.



Showing rolenames on associations (optional)

Rolenames show the roles that objects play in an association.



The above association shows that

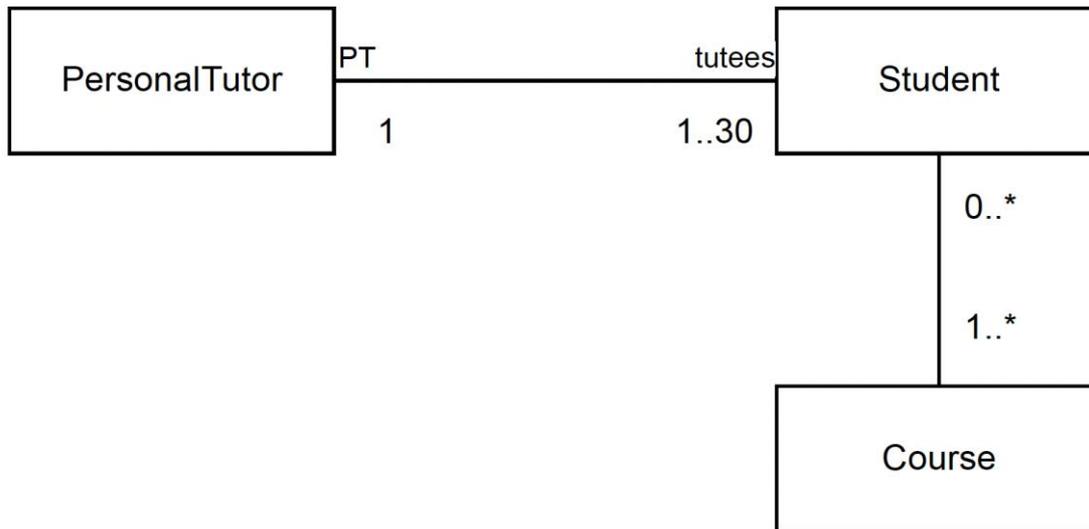
- students are tutees of a personal tutor
- a personal tutor is a PT for some students

Can use visibility notation + – etc on role names too.

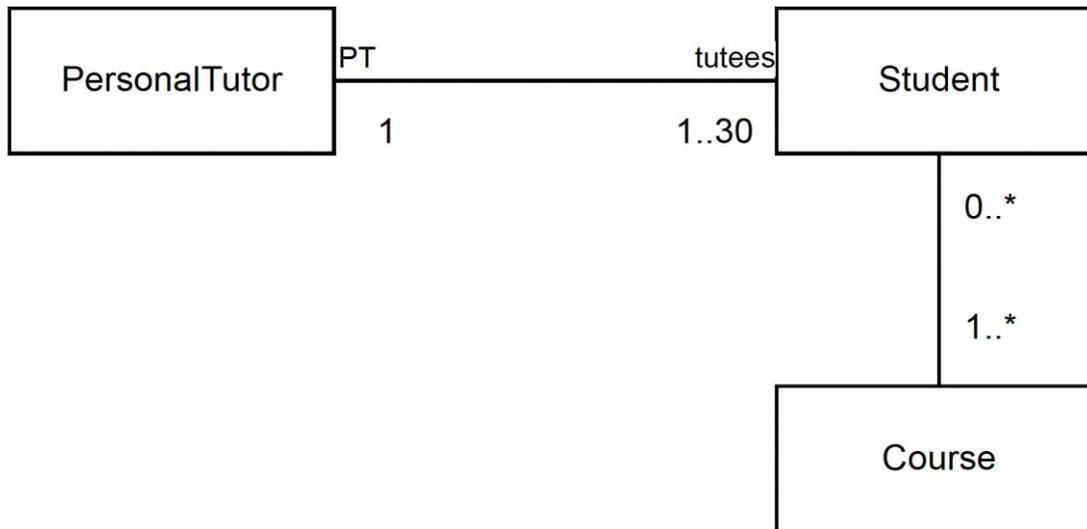
UML class diagram example

University system: You are building a system for a university. [...] Personal tutors are members of staff. The university students (tutees) each get a personal tutor to help them navigate university life. **Each personal tutor must have at least one and up to 30 tutees. Each student must sign up to at least one course. A course can have multiple students** and it “must know” about the students enrolled for it so that the students can be notified for any changes.

Showing association multiplicities (optional)

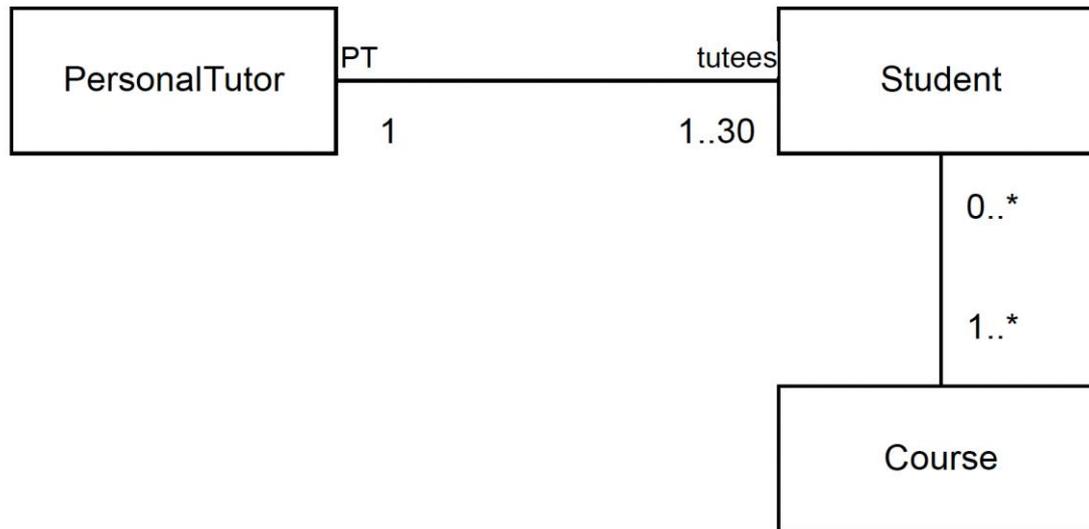


Showing association multiplicities (optional)



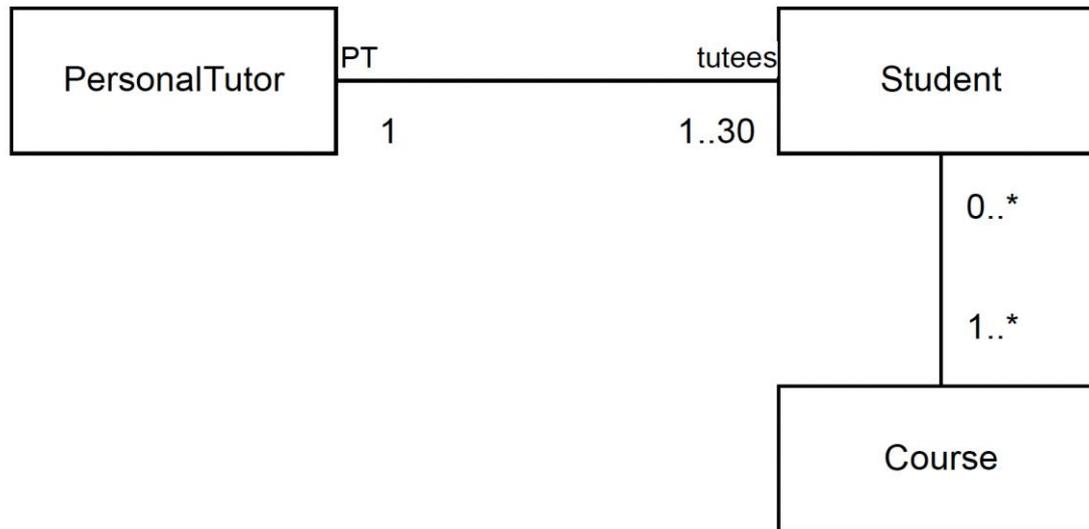
- For each personal tutor there are between 1 and 30 students

Showing association multiplicities (optional)



- For each personal tutor there are between 1 and 30 students
- For each student there is exactly one personal tutor

Showing association multiplicities (optional)



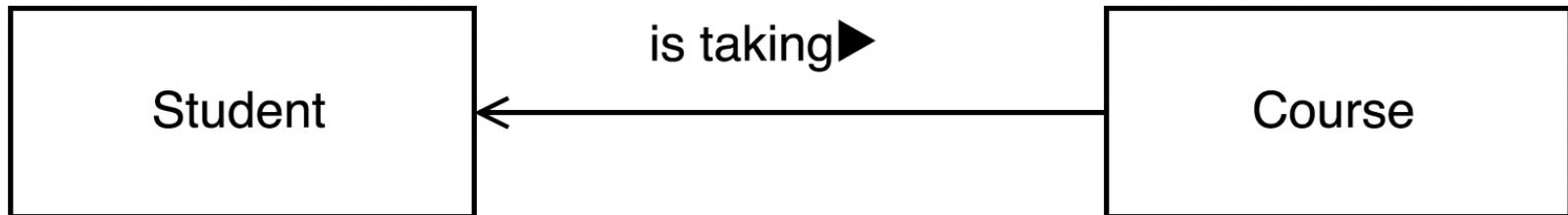
- For each personal tutor there are between 1 and 30 students
- For each student there is exactly one personal tutor
- * for unknown number: each student takes one or more courses. (0..* often abbreviated as *)

UML class diagram example

University system: You are building a system for a university. [...] Personal tutors are members of staff. The university students (tutees) each get a personal tutor to help them navigate university life. Each personal tutor must have at least one and up to 30 tutees. Each student must sign up to at least one course. **A course** can have multiple students and it “**must know**” **about the students enrolled for it so that the students can be notified for any changes.**

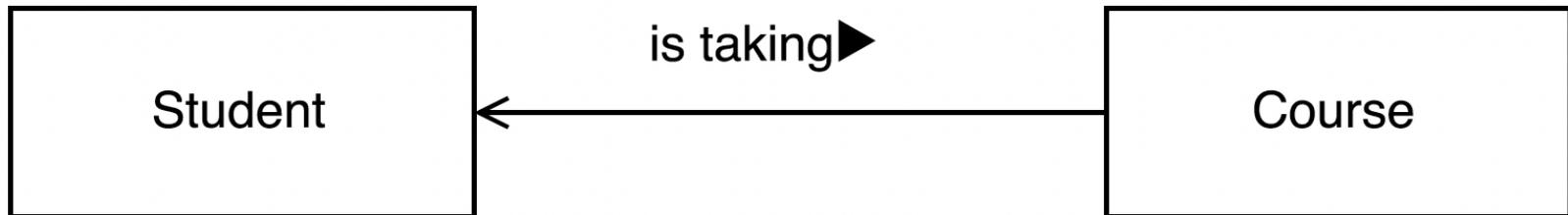
Showing association navigability (optional)

Adding an arrow at the end of an association shows that some object of the class at one end can access some object of the class at the other end, e.g. to send a message.



Showing association navigability (optional)

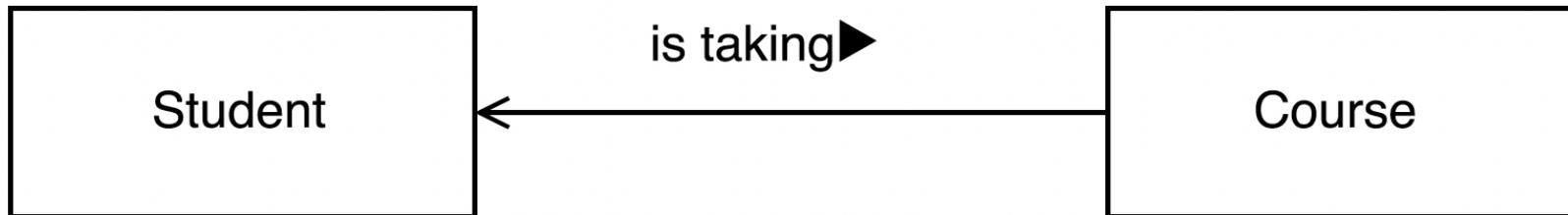
Adding an arrow at the end of an association shows that some object of the class at one end can access some object of the class at the other end, e.g. to send a message.



- Use × near an association end to show non-navigability

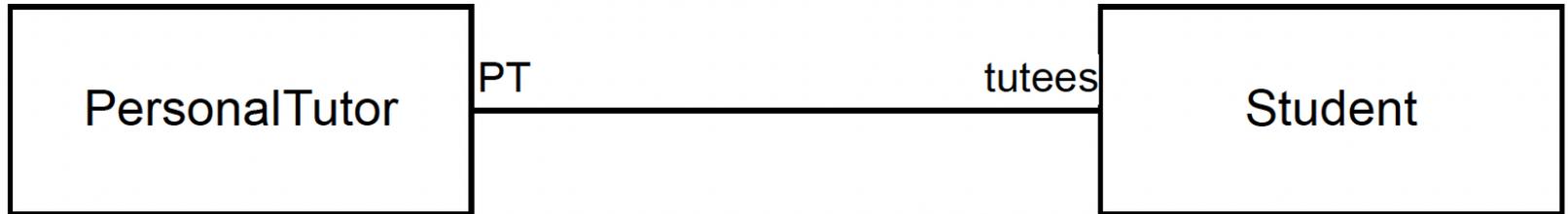
Showing association navigability (optional)

Adding an arrow at the end of an association shows that some object of the class at one end can access some object of the class at the other end, e.g. to send a message.

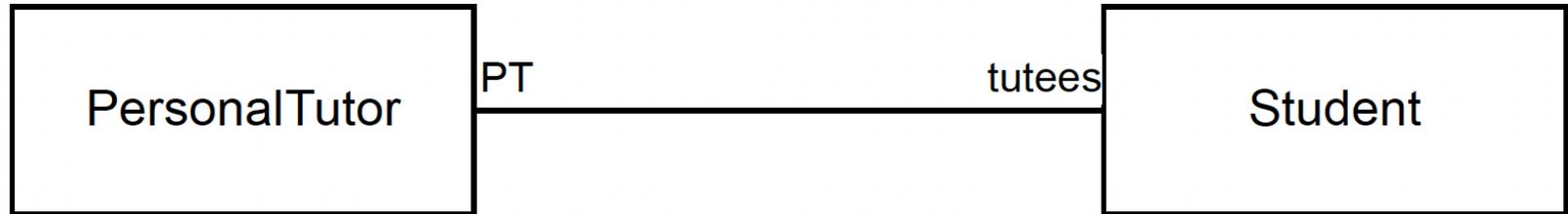


- Use × near an association end to show non-navigability
- **Direction of navigability has nothing to do with direction in which you read the association name**

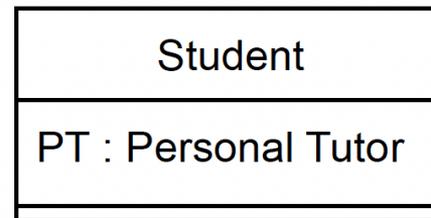
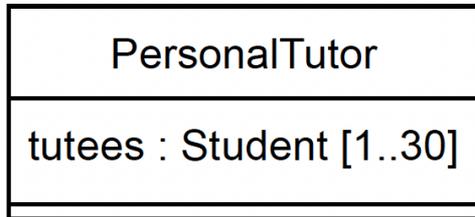
Attributes vs associations



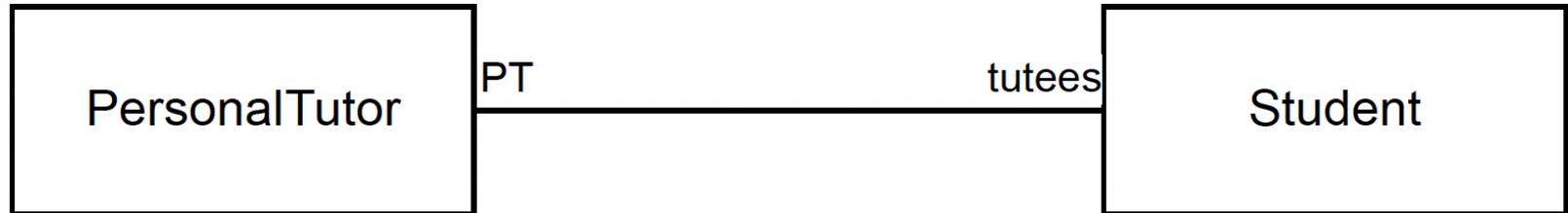
Attributes vs associations



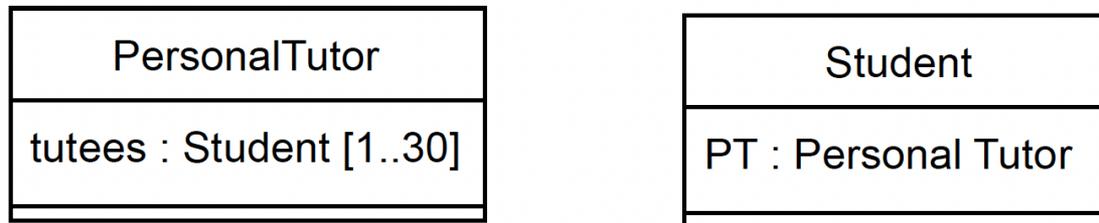
One way (not the only one!) of realising the above association is:



Attributes vs associations

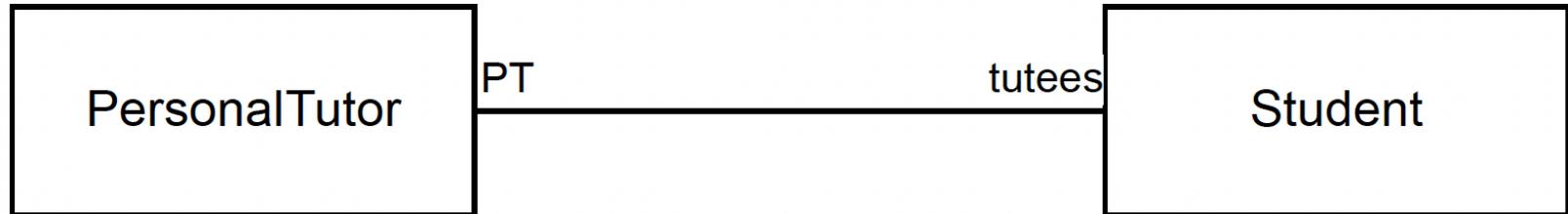


One way (not the only one!) of realising the above association is:

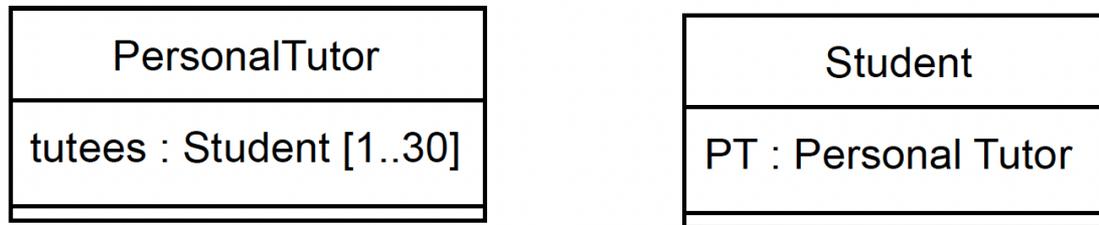


Don't show both association and attributes, as it is redundant!

Attributes vs associations



One way (not the only one!) of realising the above association is:



Don't show both association and attributes, as it is redundant!

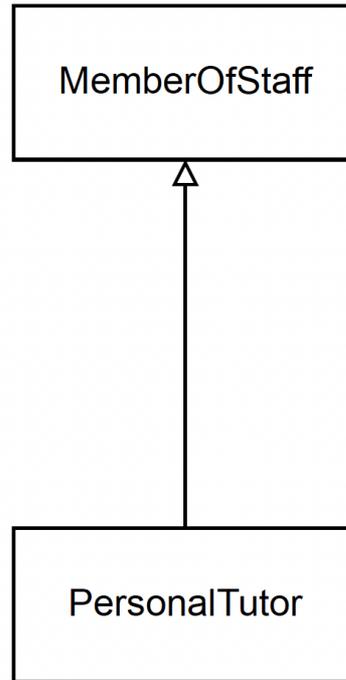
Attribute preferred if attribute type is primitive or predefined Java class, or library class.
Otherwise, association is better!

UML class diagram example

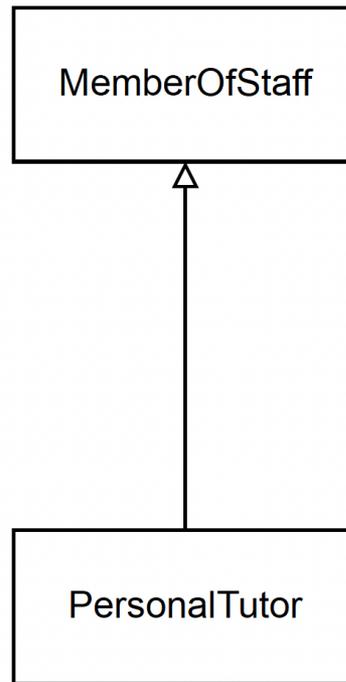
University system: You are building a system for a university. [...] Personal tutors are members of staff. The university students (tutees) each get a personal tutor to help them navigate university life. Each personal tutor must have at least one and up to 30 tutees. Each student must sign up to at least one course. A course can have multiple students and it “must know” about the students enrolled for it so that the students can be notified for any changes.

[...] Personal tutors are members of staff. Each member of staff has an ID that is incremented with each new additional staff member.

Representing class generalisation

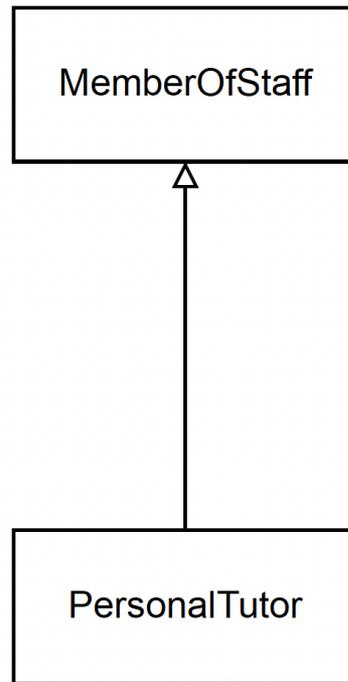


Representing class generalisation



Usually, corresponds to implementation with inheritance.

Representing class generalisation

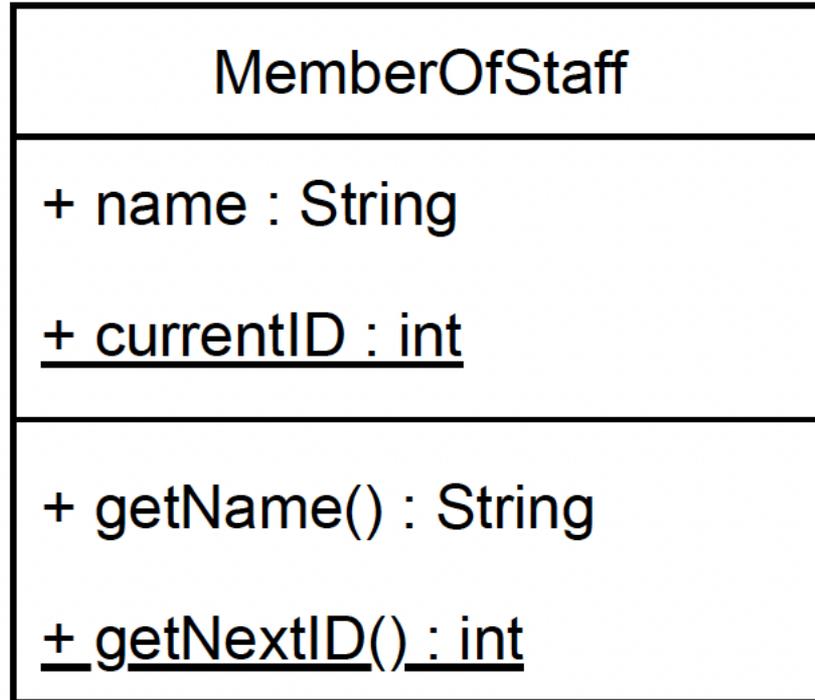


Usually, corresponds to implementation with inheritance.

Usually, can be read as is a: e.g., a personal tutor is a member of staff

Representing static attributes and operations

In UML, a static attribute or operation is an attribute or operation belonging to a class rather than the instances of the class.



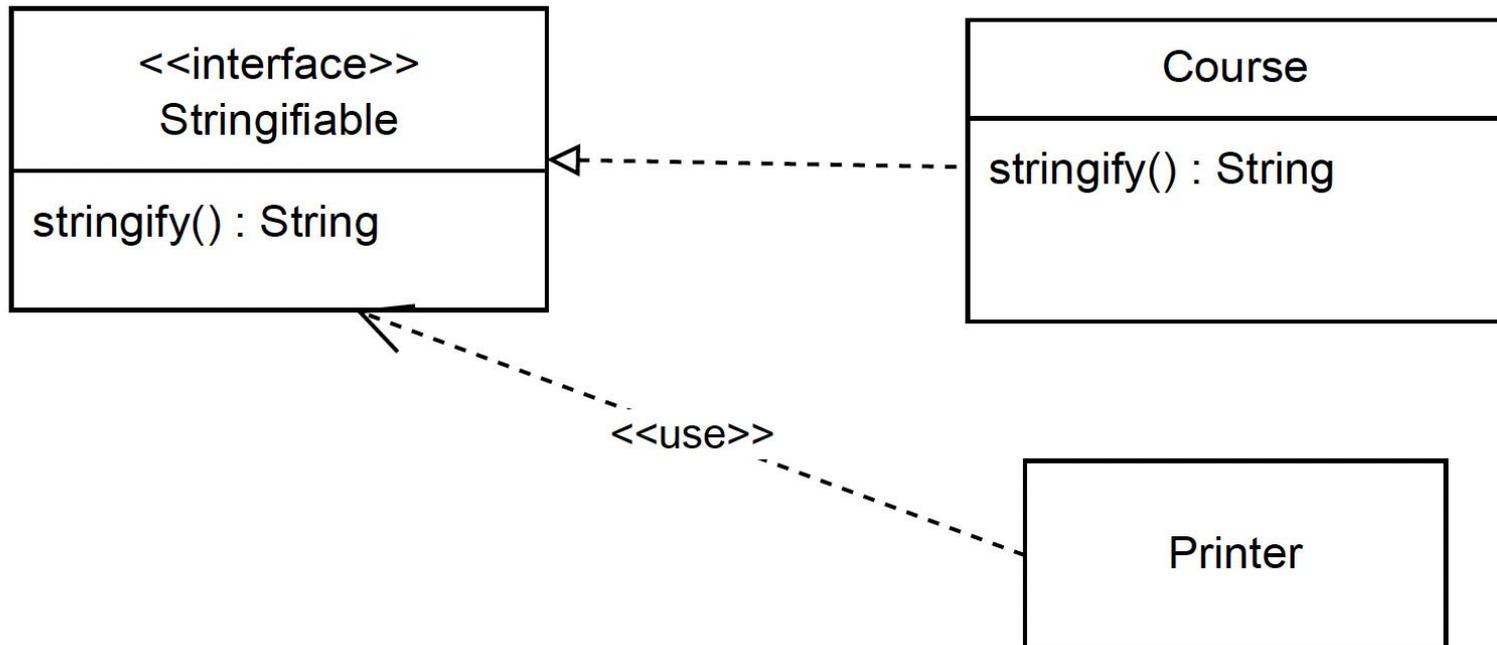
Representing interfaces and realisation

In UML an interface is just a collection of operations, that can be realised by a class.

Representing interfaces and realisation

In UML an interface is just a collection of operations, that can be realised by a class.

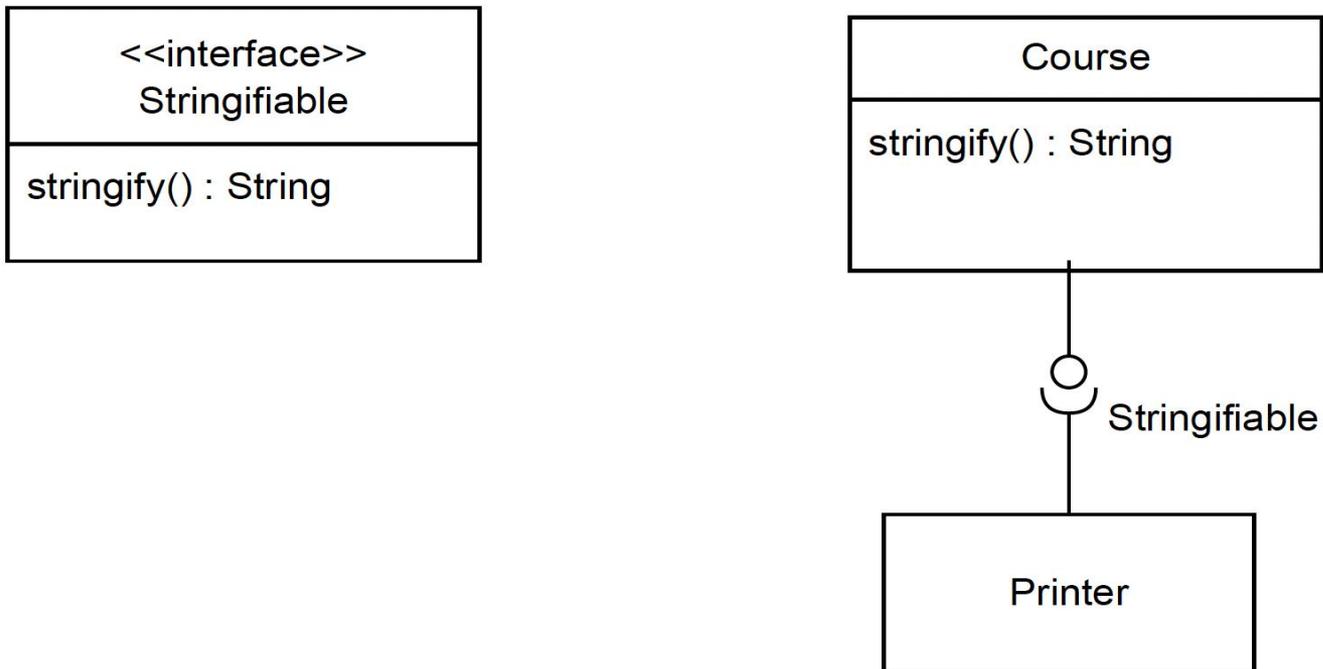
- E.g. The Course class realises the Stringifiable interface
- The Printer class interacts with the Stringifiable interface (i.e. some object of it interacts with whatever object will be implementing the interface).



Representing interfaces and realisation

In UML an interface is just a collection of operations, that can be realised by a class.

- E.g. The Course class realises the Stringifiable interface
- The Printer class interacts with the Stringifiable interface (i.e. some object of it interacts with whatever object will be implementing the interface).



Representing abstract classes and operations

- In UML, an abstract class is one that is intended to be used by subclasses, and does not have instances.

Representing abstract classes and operations

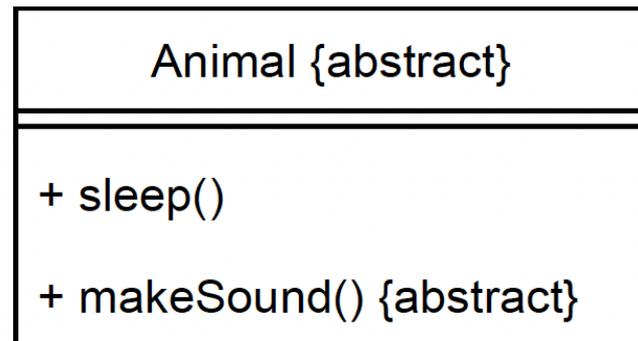
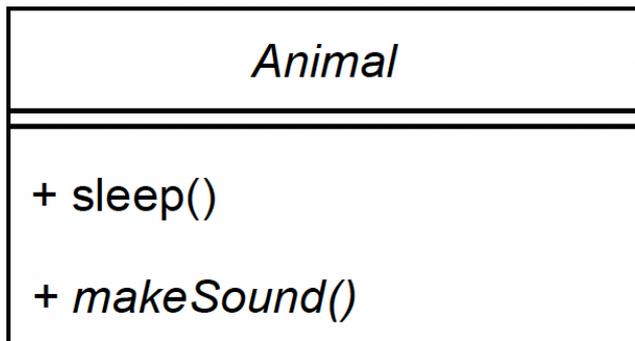
- In UML, an abstract class is one that is intended to be used by subclasses, and does not have instances.
- Abstract operations can only be present in abstract classes. They must be defined by subclasses.

Representing abstract classes and operations

- In UML, an abstract class is one that is intended to be used by subclasses, and does not have instances.
- Abstract operations can only be present in abstract classes. They must be defined by subclasses.
- An abstract class may have both abstract and regular operations.

Representing abstract classes and operations

- In UML, an abstract class is one that is intended to be used by subclasses, and does not have instances.
- Abstract operations can only be present in abstract classes. They must be defined by subclasses.
- An abstract class may have both abstract and regular operations.
- Two notation alternatives provided:



Reading

Stevens

- Recommended: Ch 2: Object concepts
- Essential: Ch 3: Introductory case study 3, until section 3.5
- Essential: Ch 5: Essentials of class models, until section 5.6
- Essential: Ch 6: More on class models sections 6.1.2, 6.1.3, 6.2, 6.6