UML crash course

UML crash course

The main questions

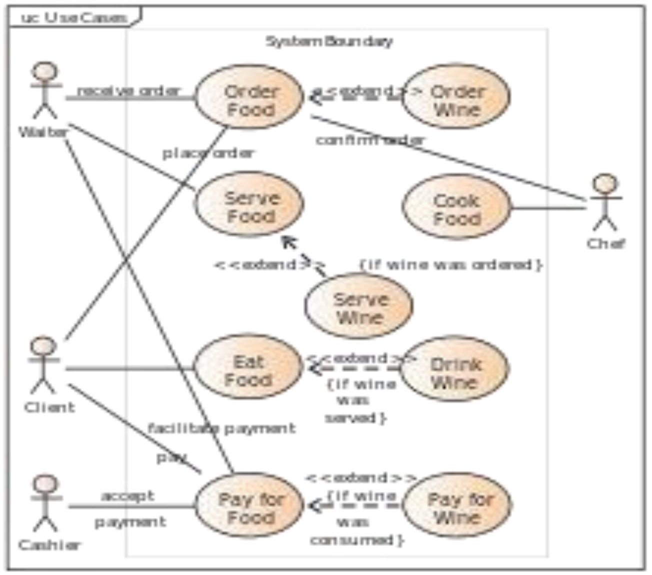
- What is UML?
- Is it useful, why bother?
- When to (not) use UML?

What is UML?

- Unified Modeling Language.
- Developed in the mid 90’s, improved since.
- Standardized notation for modeling OO systems.
- A collection of diagrams for different viewpoints:
  - Use case diagrams
  - Component diagrams
  - Class and Object diagrams
  - Sequence diagrams
  - Statechart diagrams
  - ...

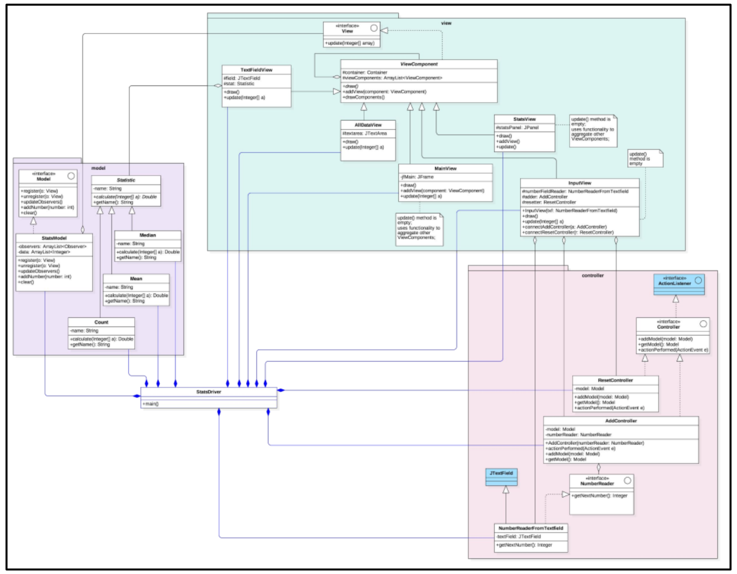
What is UML?

- Unified Modeling Language.
- Developed in the mid 90’s, improved since.
- Standardized notation for modeling OO systems.
- A collection of diagrams for different viewpoints:
  - **Use case diagrams**
  - Component diagrams
  - Class and Object diagrams
  - Sequence diagrams
  - Statechart diagrams
  - ...

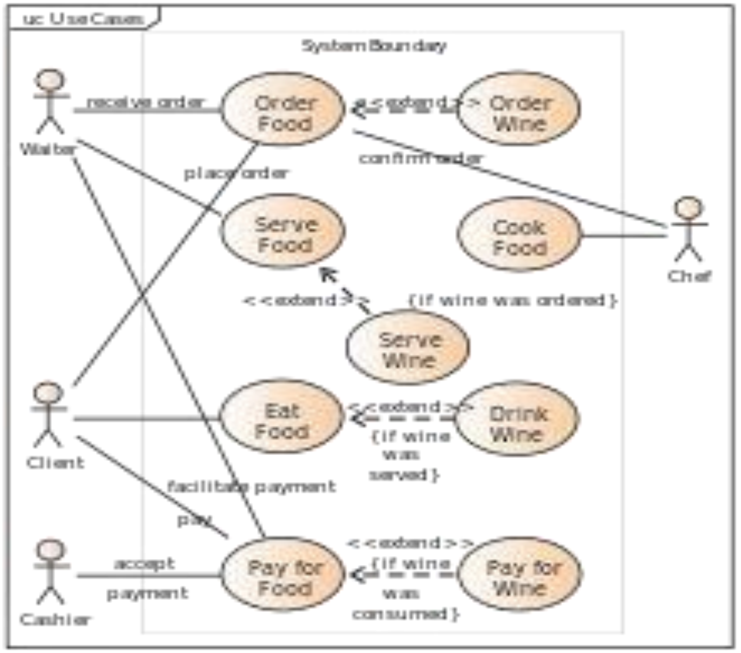


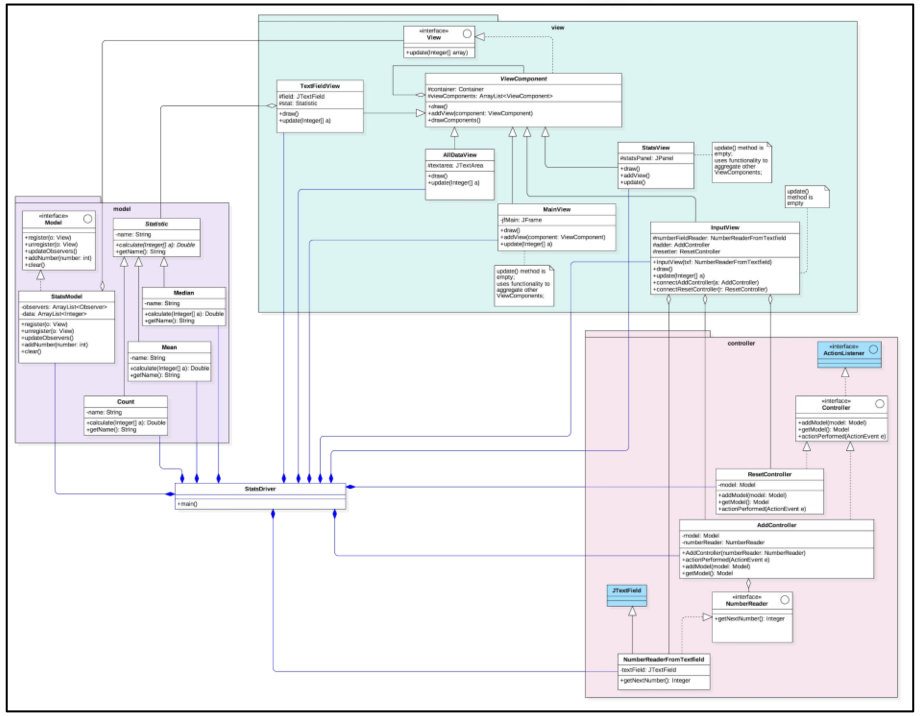
What is UML?

- Unified Modeling Language.
- Developed in the mid 90’s, improved since.
- Standardized notation for modeling OO systems.
- A collection of diagrams for different viewpoints:
    - Use case diagrams
    - Component diagrams
    - **Class and Object diagrams**
    - Sequence diagrams
    - Statechart diagrams
    - ...



Are UML diagrams useful?





Are UML diagrams useful?

Communication

- Forward design (before coding)
    - Brainstorm ideas (on whiteboard or paper).
    - Draft and iterate over software design.

Documentation

- Backward design (after coding)
    - Obtain diagram from source code.

Classes vs. objects

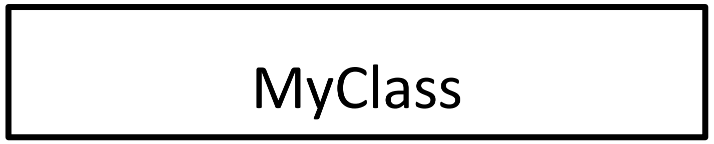
Class

- Grouping of similar objects.
  - Student
  - Car
- Abstraction of common properties and behavior.
  - Student: Name and Student ID
  - Car: Make and Model

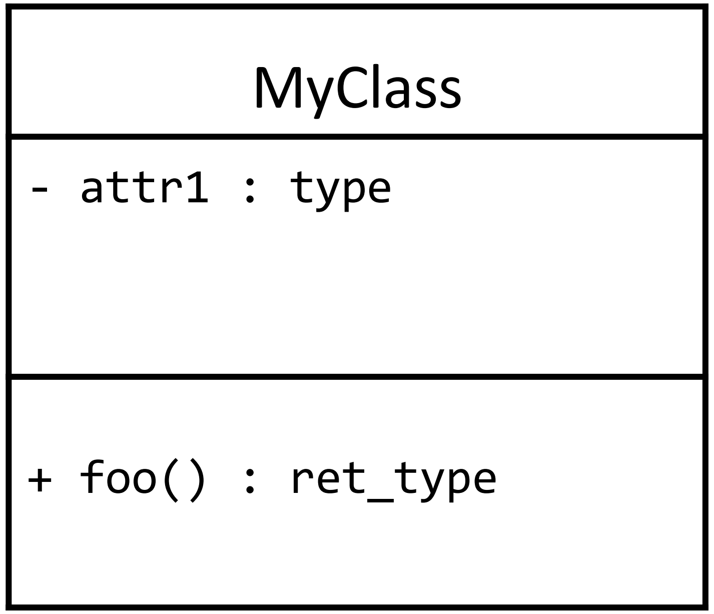
Object

- Entity from the real world.
- Instance of a class
  - Student: Joe (4711), Jane (4712), …
  - Car: Audi A6, Honda Civic, ...

UML class diagram: basic notation



UML class diagram: basic notation



Name

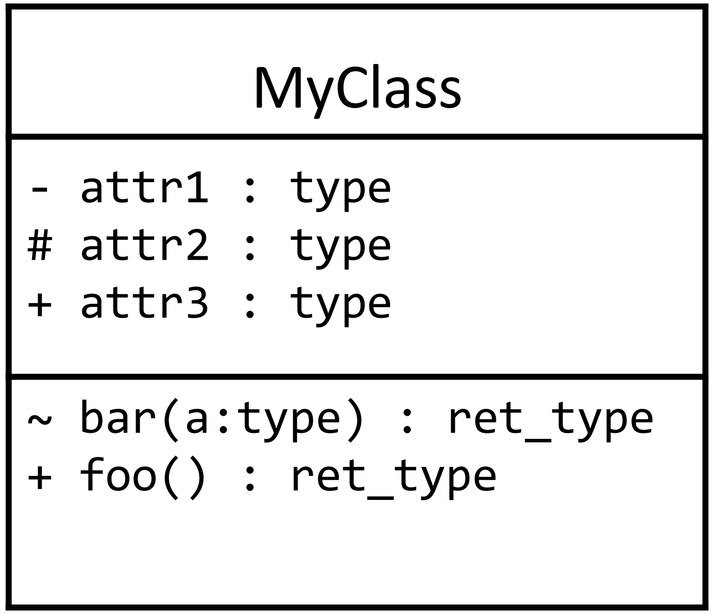
Attributes

<visibility> <name> : <type>

Methods

<visibility> <name>(<param>) :*
<return type>
<param> := <name> : <type>

UML class diagram: basic notation



Name

Attributes

<visibility> <name> : <type>

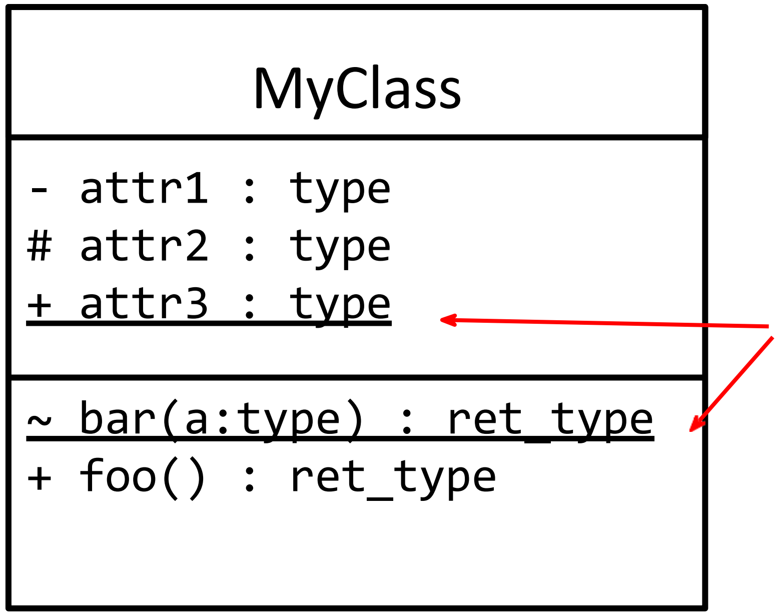
Methods

<visibility> <name>(<param>) :*
<return type>
<param> := <name> : <type>

Visibility

- private
~ package-private
# protected
+ public

UML class diagram: basic notation



Name

Attributes

<visibility> <name> : <type>

Static attributes or methods are underlined

Methods

<visibility> <name>(<param>) :*
<return type>
<param> := <name> : <type>

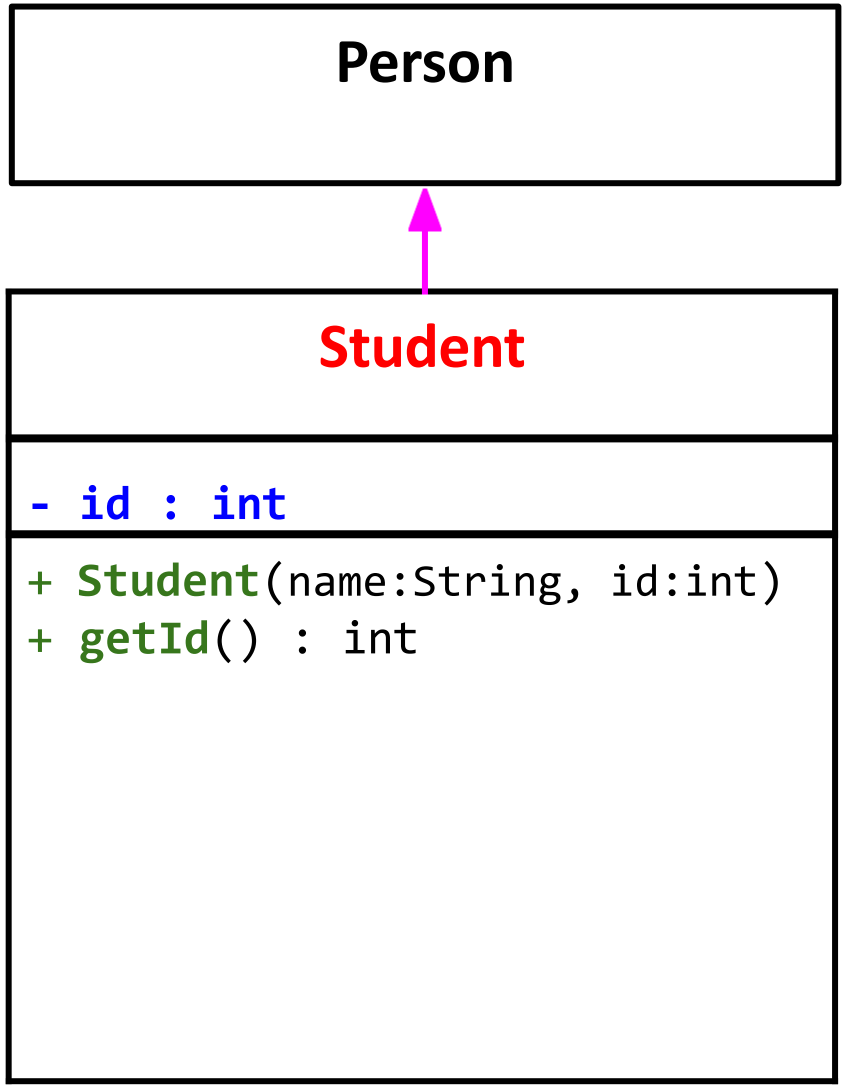
Visibility

- private
~ package-private
# protected
+ public

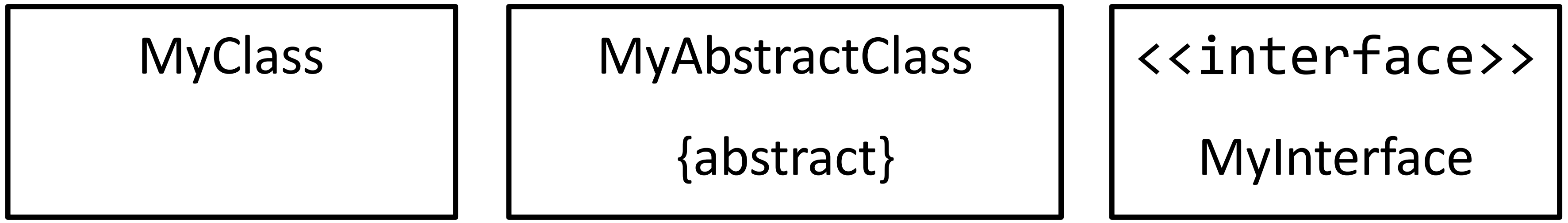
UML class diagram: concrete example

```
public class Person {
  …
}
```

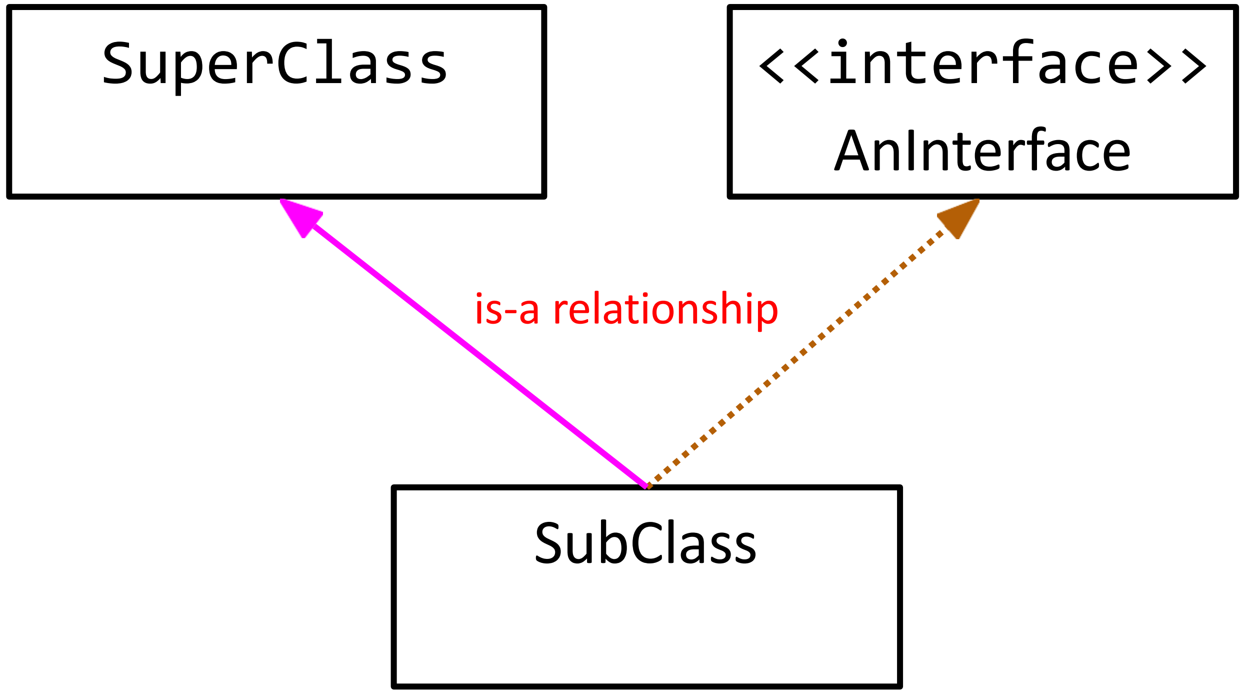
```
public class Student
    extends Person {

  private int id;

  public Student(String name,
                 int id) {
     ...
  }

  public int getId() {
    return this.id;
  }
}
```



Classes, abstract classes, and interfaces



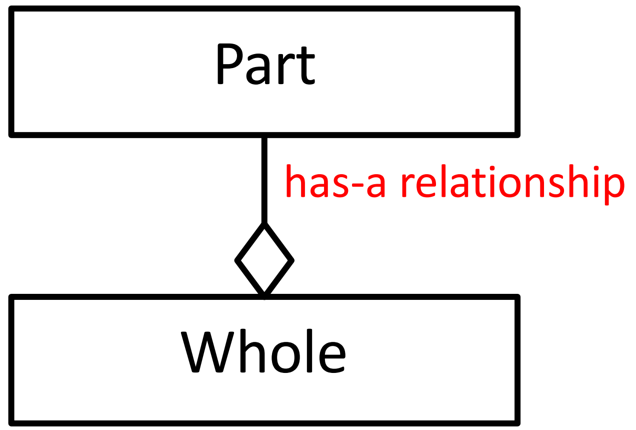
UML class diagram: Inheritance



```
public class SubClass extends SuperClass implements AnInterface
```

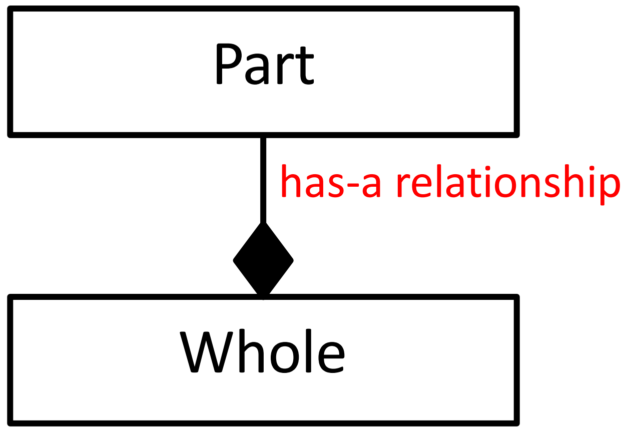
UML class diagram: Aggregation & Composition

Aggregation



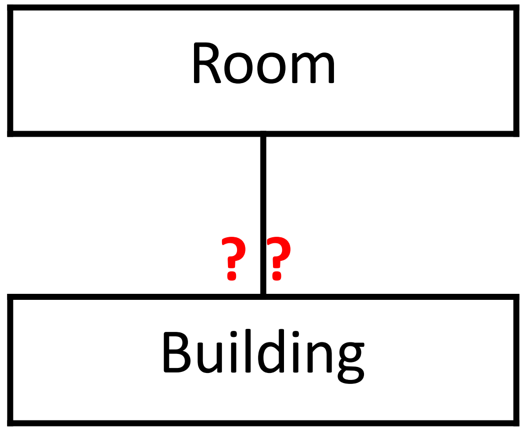
- Existence of Part does not depend on the existence of Whole.
- Lifetime of Part does not depend on Whole.
- No single instance of whole is the unique owner of Part (might be shared with other instances of Whole).

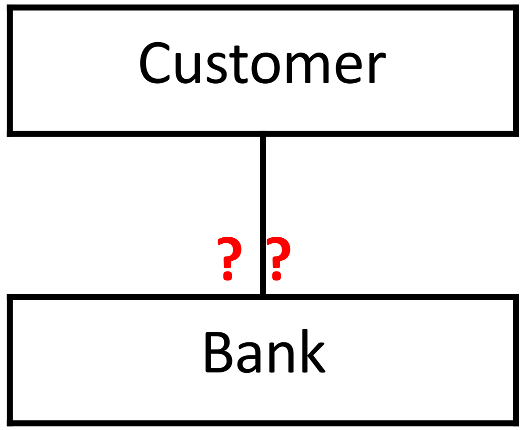
Composition



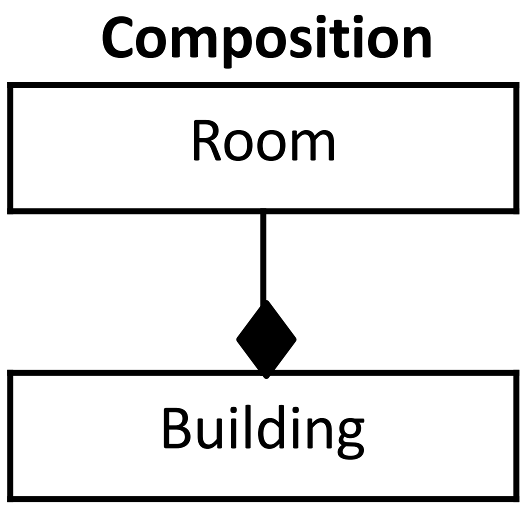
- Part cannot exist without Whole.
- Lifetime of Part depends on Whole.
- One instance of Whole is the single owner of Part.

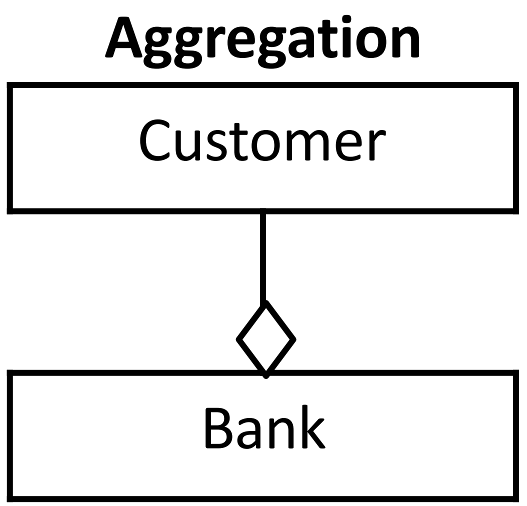
Aggregation or Composition?



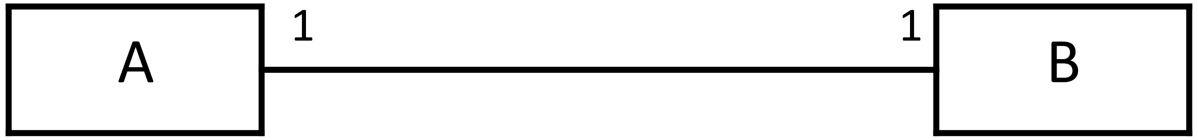


Aggregation or Composition?

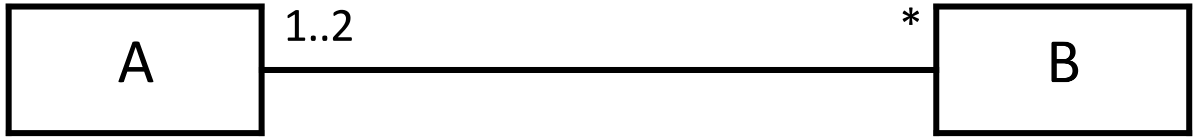




UML class diagram: multiplicity

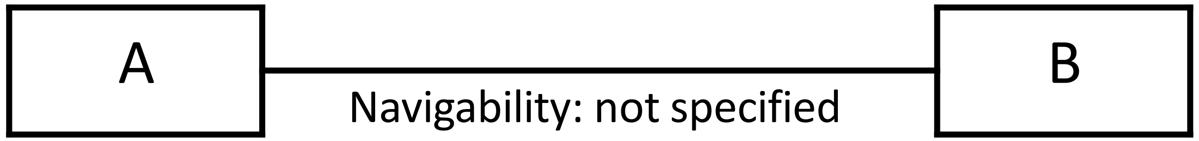


Each A is associated with exactly one B
Each B is associated with exactly one A

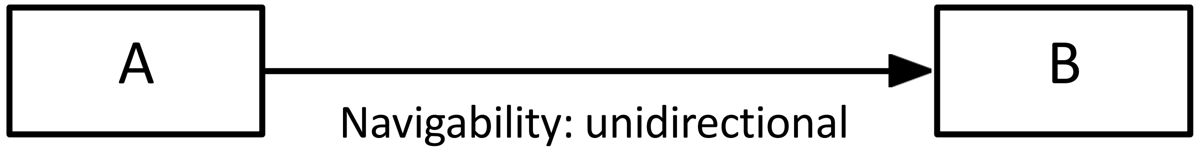


Each A is associated with any number of Bs
Each B is associated with exactly one or two As

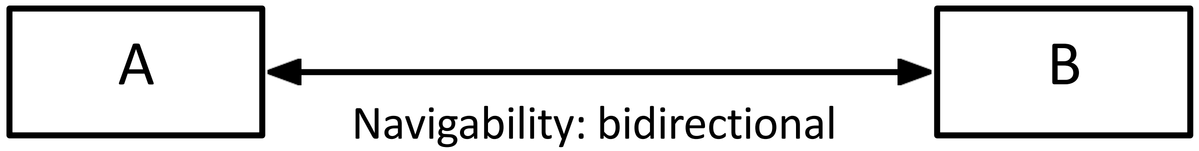
UML class diagram: navigability



Navigability: not specified

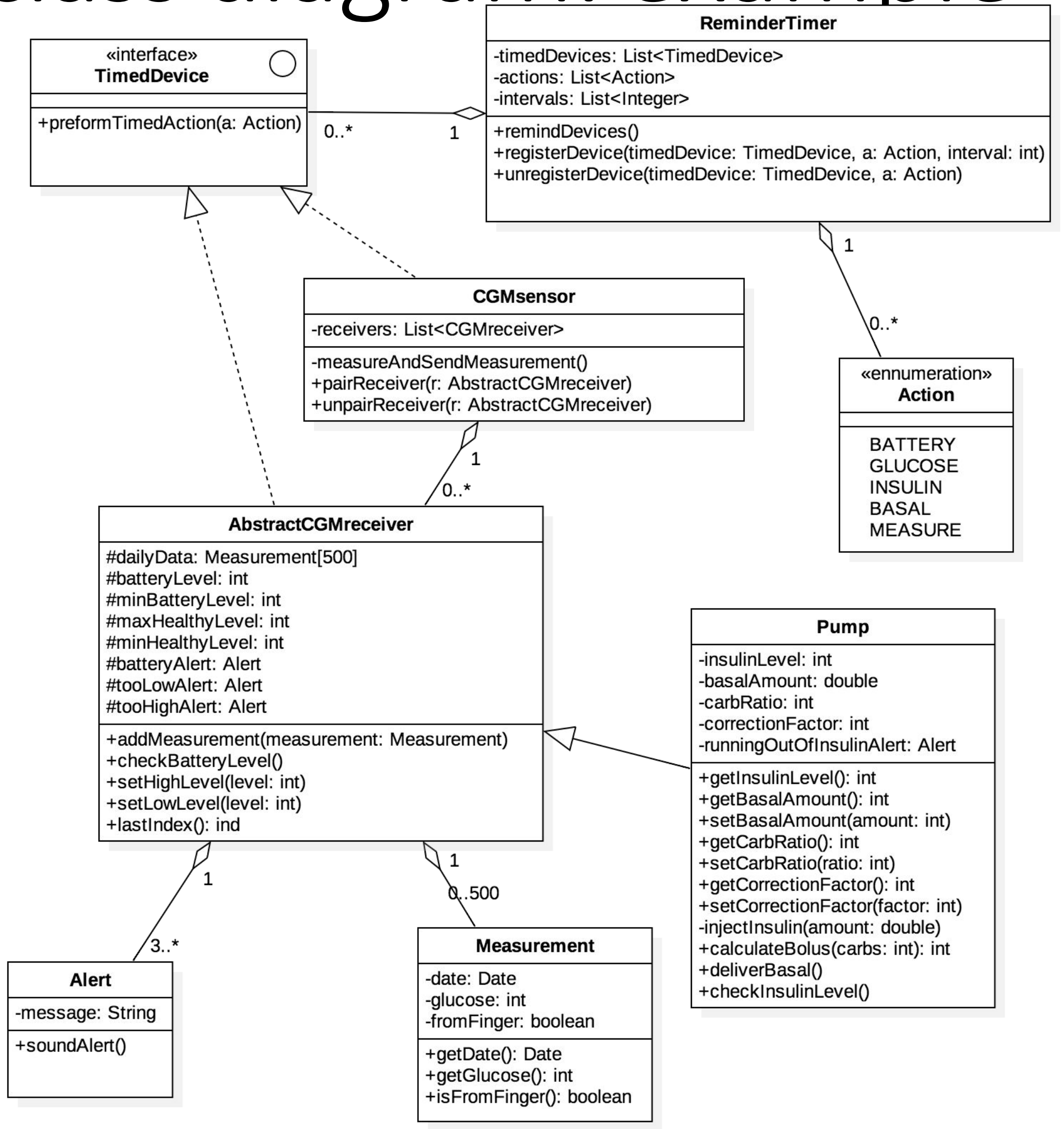


Navigability: unidirectional
“can reach B from A”



Navigability: bidirectional

UML class diagram: example



Summary: UML

- Unified notation for modeling OO systems.
- Allows different levels of abstraction.
- Suitable for design discussions and documentation.