Universität Zürich[UZH]

A Short Introduction to UML Sequence Diagrams

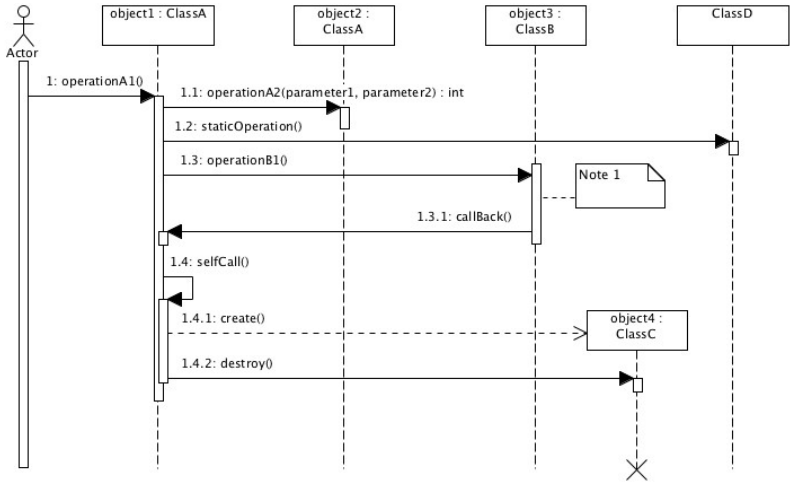
with some remarks on class diagrams

Software Engineering, HS 2016
Exercise session of 27th of September

Universität Zürich[UZH]

Overview

- Part I: Sequence Diagrams



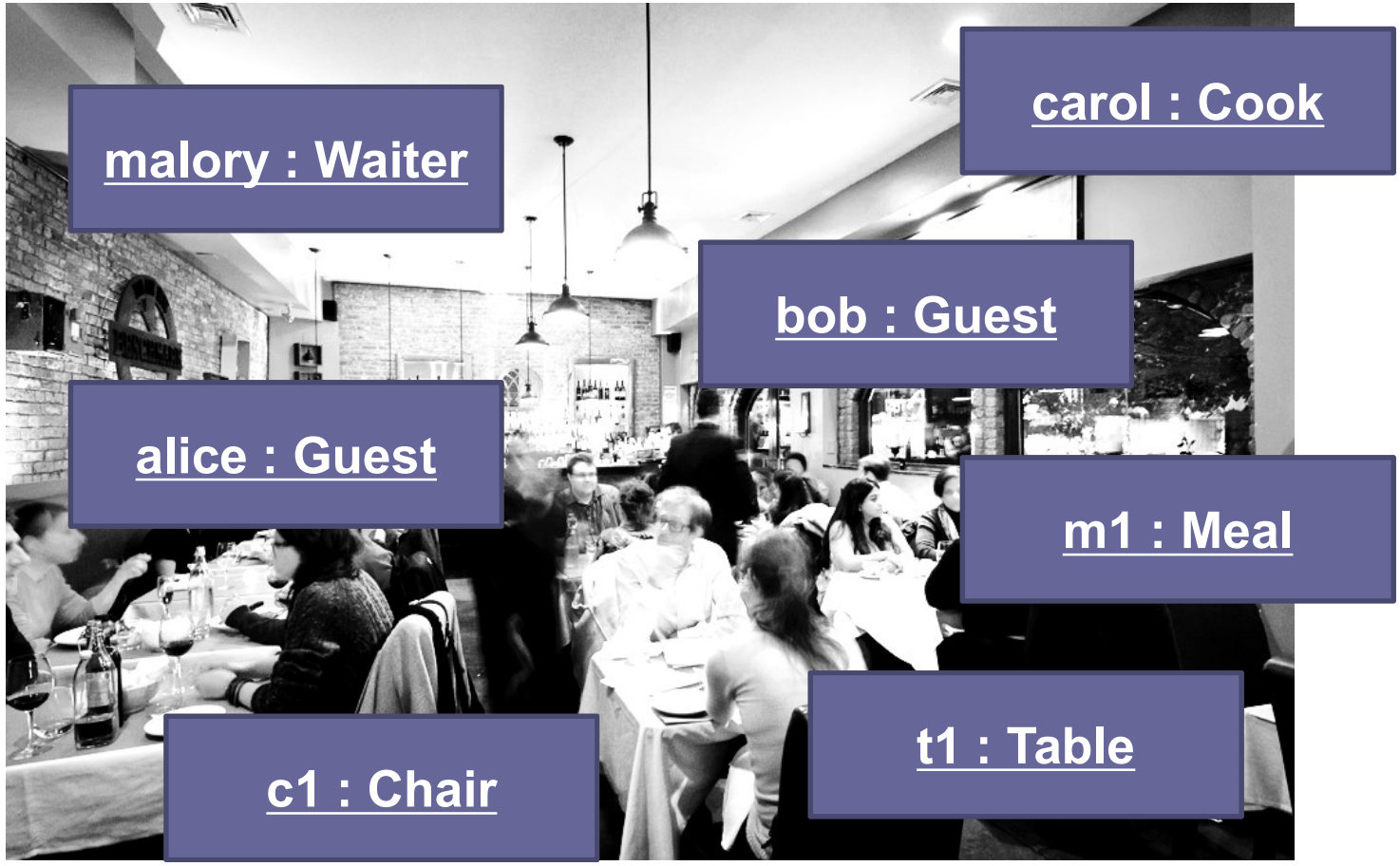
- Part II: Class Diagrams

An Example

What does a programmer see here?

[1]

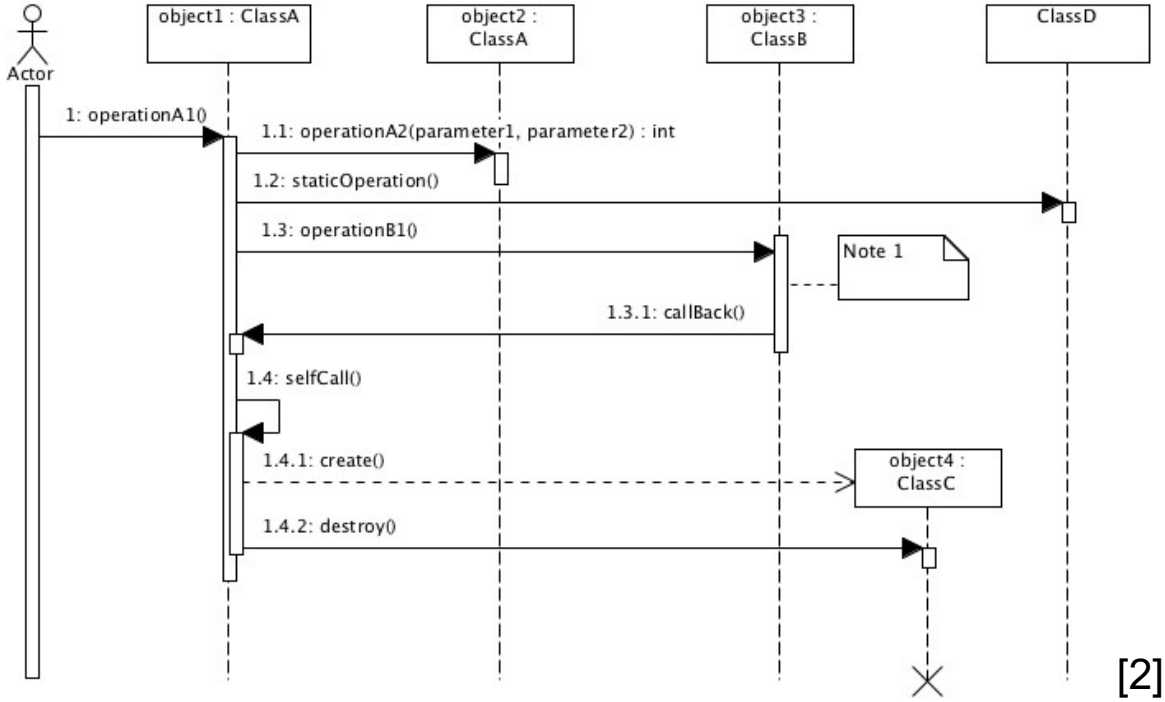
Set of Objects Communicating



Universität Zürich[UZH]

Basics

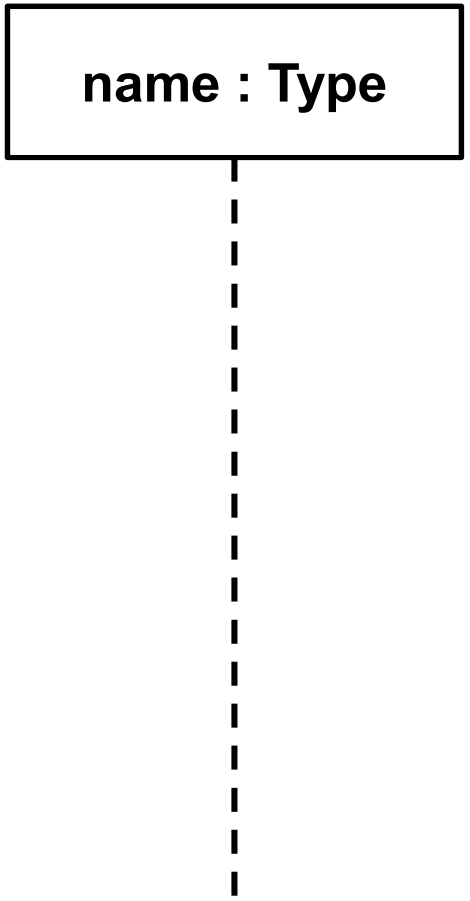
- Time is increasing from top to bottom
- Communication participants and messages are aligned horizontally



[2]

Communication Participants: Lifelines

- Represent exactly one single participant in a communication
- Represents roles
  - Name is often omitted (anonymous lifeline)
- Notation:
  - Box und dashed line
  - Other symbols instead of box are allowed



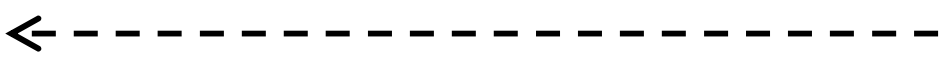
Messages

- Horizonally from one lifeline to another
- Different types:

Asynchronous message

Synchronous message

Reply message



Message parameters

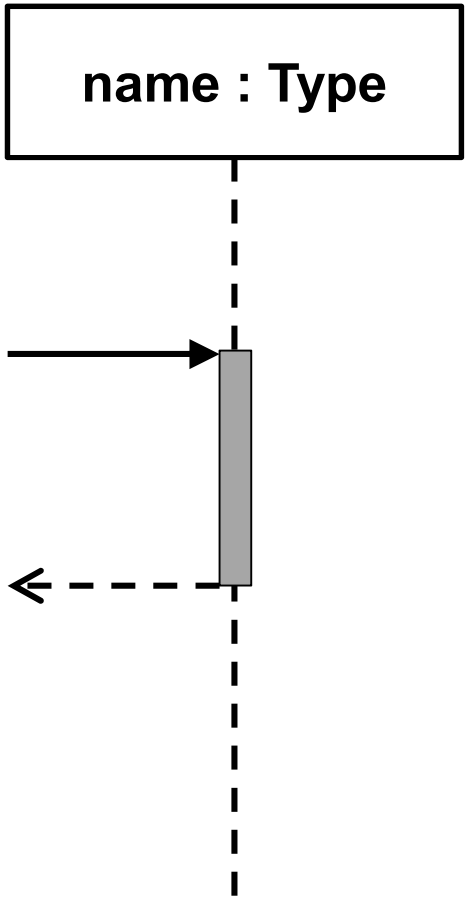
- Notation:
  - Only the name is mandatory

- Example:

Universität Zürich[UZH]

Occurrence Specifications

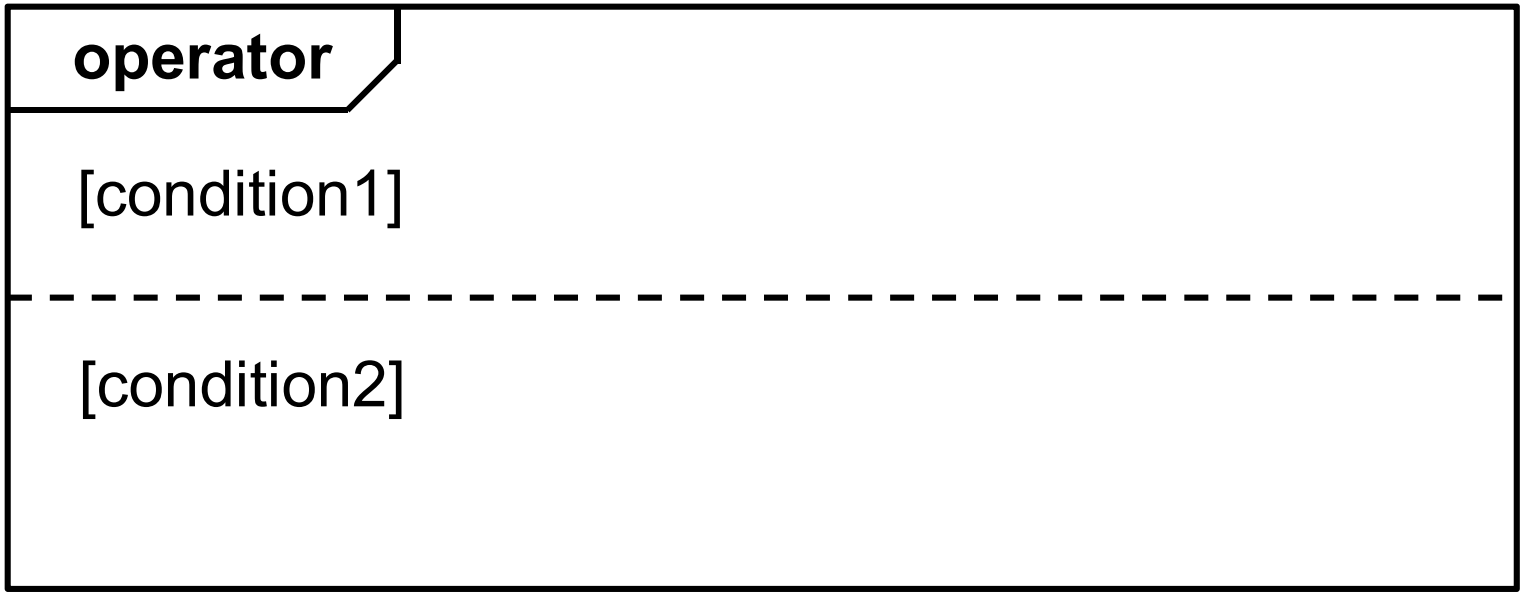
- Illustrates the timespan during which a lifeline is acitve (has the execution focus)
- Starts and ends with an execution occurrence (normally a message being sent or received)
- Shown as grey or white boxes
- Drawing is optional
- Sometimes called «activation»



Combined Fragments

- Used to model diverging control flows
- Interaction operators

- Notation:



Restaurant example

```
01 public class Waiter {
02
03         //...
04
05         public void serve(Guest aGuest, Cook theCook) {
06                 String order = aGuest.getOrder();
07                 Meal orderedMeal = theCook.getMeal(order);
08                 aGuest.serveMeal(orderedMeal);
09         }
10 }
```

Restaurant example

```
01  public class Guest {
02
03          //...
04
05          public String getOrder() {
06                  return "Spaghetti";
07          }
08
09          public void serveMeal(Meal orderedMeal) {
10                  while (!orderedMeal.equals("finished")) {
11                          orderedMeal.eat();
12                  }
13          }
14  }
```

Restaurant example

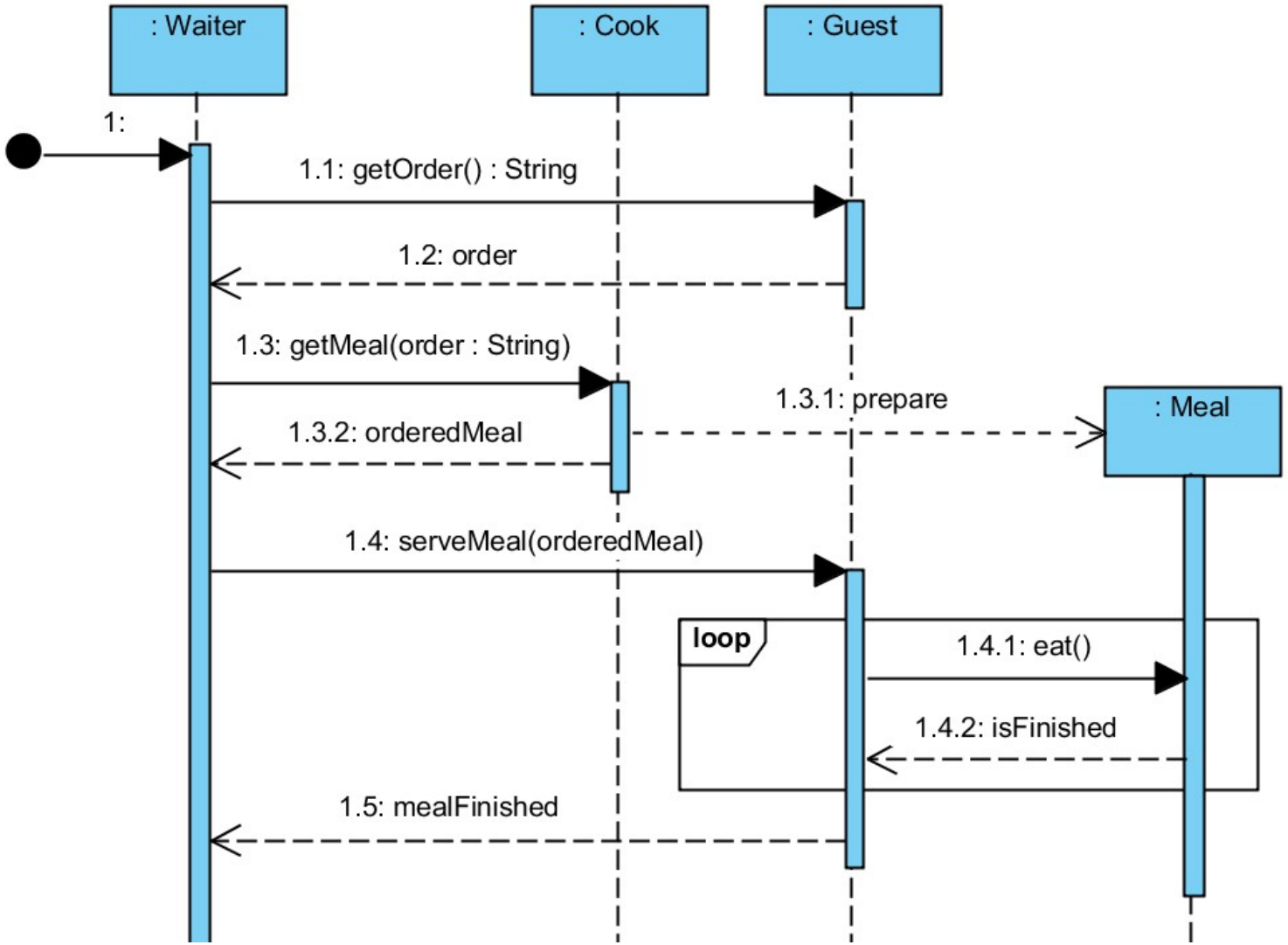
```
01 public class Cook {
02
03         public Meal getMeal(String order) {
04                 Meal newMeal = new Meal(order);
05                 return newMeal;
06         }
07 }
```

Restaurant example

```
01  public class Meal {
02
03          //...
04
05          public boolean eat() {
06                  if (remainingParts > 1) {
07                          this.remainingParts--;
08                          this.state = "eating";
09                          return true;
10                  } else {
11                          this.remainingParts--;
12                          this.state = "finished";
13                          return false;
14                  }
15          }
16  }
```

Universität Zürich[UZH]

Complete restaurant example

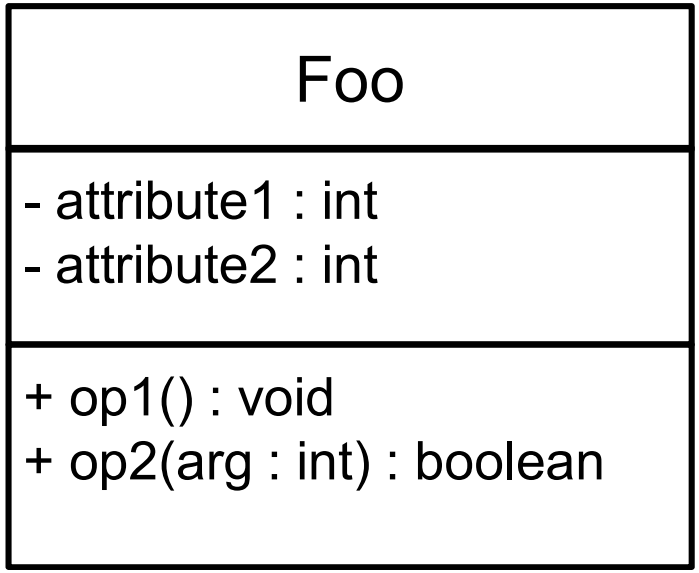


Class Diagrams

- What should be included?

Is this class diagram complete?

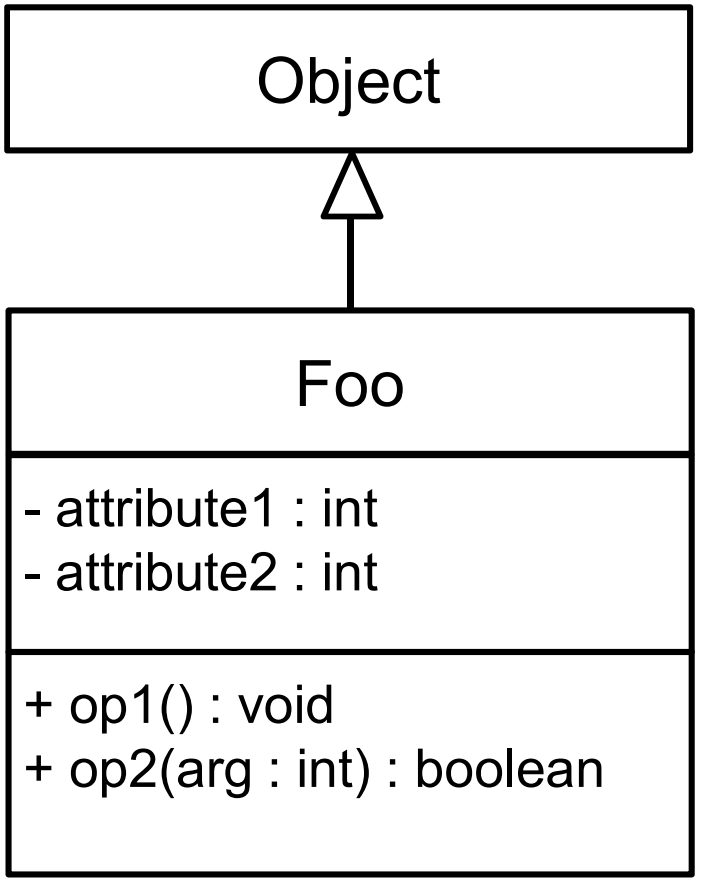
```
01 public class Foo {
02    private int attribute1;
03    private int attribute2;
04
05    public void op1() {
06       this.attribute1++;
07    }
08
09    public boolean op2(int arg) {
10       return this.attribute2 > arg;
11    }
12 }
```



«There are exactly three methods that can be called on an instance of class Foo.»

Is this class diagram complete?

```
01 public class Foo {
02    private int attribute1;
03    private int attribute2;
04
05    public void op1() {
06       this.attribute1++;
07    }
08
09    public boolean op2(int arg) {
10       return this.attribute2 > arg;
11    }
12 }
```



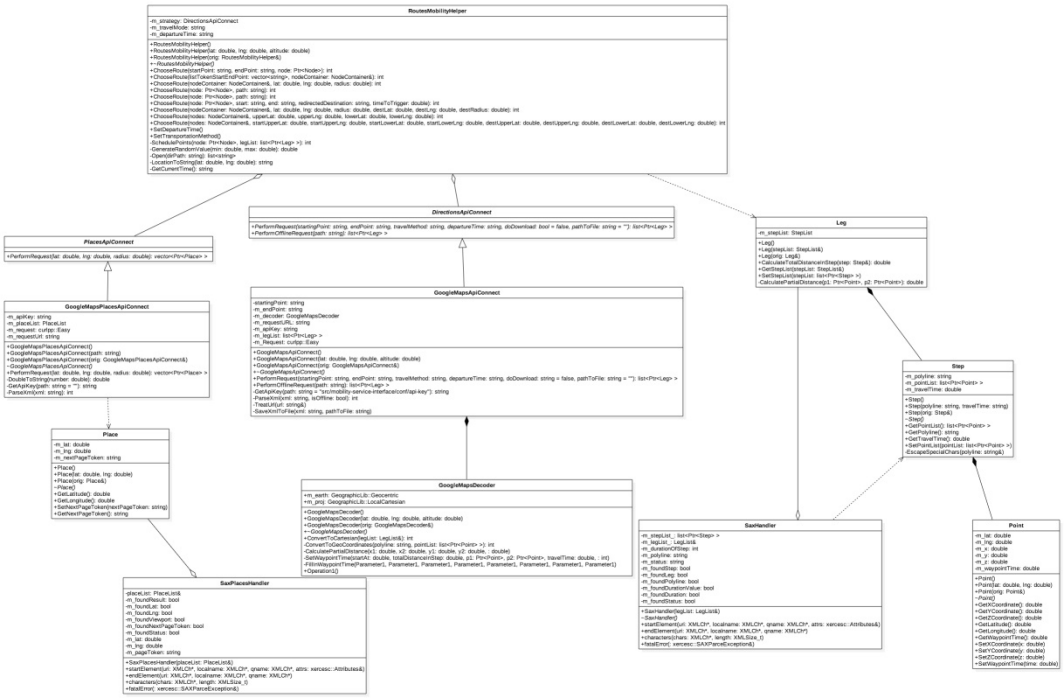

```
myFoo.toString();
```

Pragmatism in Class Diagrams

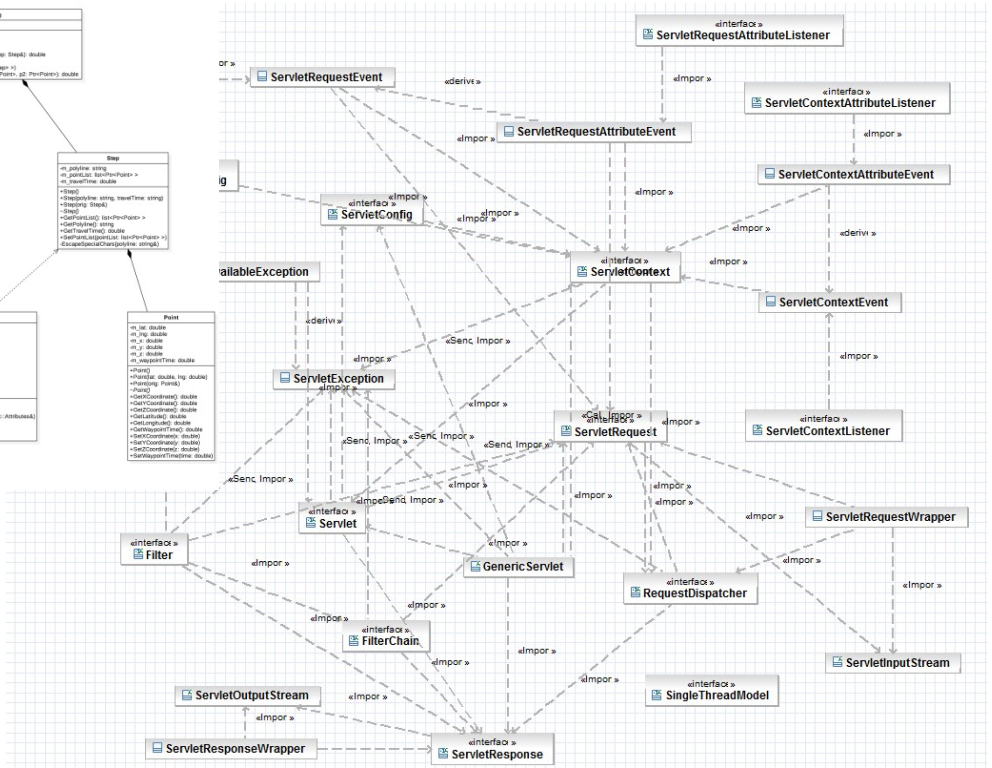
- You cannot infer anything from absence in a UML class diagram
- If something is not present in a class diagram, this can mean that it actually does not exist or the modeler deemed it unimportant.
- It's the obligation of the modeler to decide what should be shown in a class diagram and what shouldn't.
    - This depends on the use of the model (pragmatism).
- It's important to make sure that all involved parties have a common understanding of the model (iterations with validation)

Universität Zürich[UZH]

Probably not so good examples



[3]



[4]

Universität Zürich[UZH]

Questions…

Universität Zürich[UZH]

Thank you for your attention.