



# Design Patterns & Software Architecture Introduction

---

dr. Joost Schalken-Pinkster

Windesheim University of Applied Science

The Netherlands

The contents of these course slides is (in great part) based on:

Chris Loftus, *Course on Design Patterns & Software Architecture for NEU*. Aberystwyth University, 2013.

Jeroen Weber & Christian Köppe, *Course on Patterns and Frameworks*. Hogeschool Utrecht, 2013.

Leo Puijt, *Course on Software Architecture*. Hogeschool Utrecht, 2010-2013.



# Session overview

---

- Short introduction about me...
- Design Patterns & Software Architecture
- UML Refresher (if required)
- The contents of the course



# Short introduction about me...

---

# About Joost Schalken-Pinkster





× [in Joost Schalken-Pinkster | LinkedIn](#) × +

https://www.linkedin.com/in/jschalken/ 🔍 ☆




**in 领英** 🔍 Search

Home My Network Jobs Messaging Notifications Me Work Upgrade to Premium

**4 Machine Learning types** - Interested in knowing the impact in their business? Find out now! Ad ...



**Joost Schalken-Pinkster**  
Senior Lecturer ICT at Windesheim  
Zwolle Area, Netherlands



 Windesheim  
 Vrije Universiteit Amsterdam  
 See contact info

Edit public profile & URL ?

Add profile in another language ?

Ad ...

Get the latest jobs and industry news



Joost, explore relevant opportunities with  
ABIN accountants

[Follow](#)



# About me/the Netherlands





# What are your experiences?

- Where are *you* from?
- Who has programmed in....
  - Language: Java, C#, C, C++, Objective C, Haskell...
  - Size of programs: in LOC >1.000, 1.000–10.000, > 10.000
  - OS: Browser based, Windows, MacOS, iOS...
- Who has experience in industry?
  - Internship, summer job, part-time job
- I love to know more about you!
  - I'd love to share a meal or do other forms of exchange!



# **Introduction to the course**

## **Design Patterns & Software Architecture**

---



# Style of teaching

- Will talk around slides, some examples on the blackboard
  - 32 hours talks
- Lab sessions
  - 8 hours, with worksheets
- Interactive:
  - Where you see “...” I will say more and perhaps ask questions ...





# Best way to learn:

The best way for me to teach you is if you ask questions...

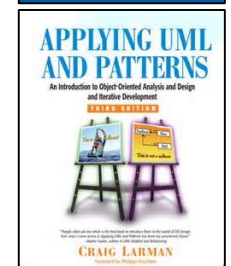
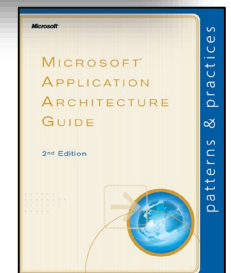
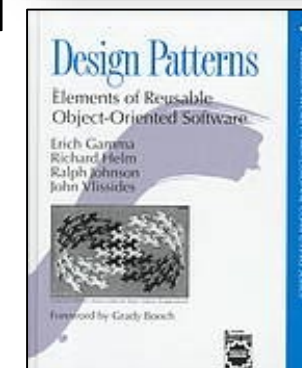
- Ask short questions during the presentation
- Ask longer questions at the end of the class
- Ask questions by e-mail: [joost@schalken.me](mailto:joost@schalken.me)  
(or [jjp.schalken@windesheim.nl](mailto:jjp.schalken@windesheim.nl) if that doesn't work)
- Ask questions by WeChat: Joost SP  
(until the end of the course, not available afterwards)

Remember there are no dumb questions...

# Literature



- Course textbook:
  - Eric Freeman et al, *Head First Design Patterns*, O'Reilly 2004. (book is also available in pdf in English and Chinese).
  - E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley, 1996
- Useful reference (not mandatory):
  - E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design patterns: Elements of reusable object-oriented software*, Addison-Wesley, 1996.
  - David Hill et. al (2009) Microsoft Application Architecture Guide, 2nd Edition. Microsoft Press, 2009.  
<http://msdn.microsoft.com/en-us/library/ff650706.aspx>
  - C. Larman (2004), *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd ed. Addison Wesley, 2004.



# Topic of this course:



Software Design Patterns

~ 85%

Software Architecture

~ 15%



# Tests/grading:

## Grading:

- Attendance\* 25.0%
- Short in-course choice quizzes (beginning of week 2 & 3; 17<sup>th</sup> and 24<sup>th</sup>)\* 2x 12.5%
- Two lab assignments (in groups, handed out at end of week 1)  
(deadline: Friday December 28<sup>th</sup> by the latest) 2x 12.5%
- Big open design problem quizzes 25.0%

## Some rules the tests:

- Theoretical tests are closed book tests, so only a (paper) dictionary is allowed.
- Cheating on any test or on the lab work always leads to a fail-grade.
- Lab work is done in groups of three students, no cooperation outside your group or copying code of the internet (that is considered cheating).

- \* For students who have a valid, official reason for not being able to be present (hospital, official internship, big family events like a marriage), there is the possibility to do an alternative assignment provided by the teacher for the attendance and the two short in-course quizzes... Contact me in time for this.

# Course site => Need your help...



A screenshot of the Canvas LMS interface for a course titled "NEU-DPSA-2018". The browser address bar shows the URL "https://canvas.instructure.com/courses/1491035". The left sidebar contains navigation icons for Home, Announcements, Assignments, Discussions, Grades, People, Pages, Files, Syllabus, Outcomes, Quizzes, Modules, Conferences, and Collaborations. The main content area shows the "Modules" section with a "01. Introduction" module containing a file named "01-Introduction.pdf". The right sidebar displays "Course status" with "Unpublished" and "Publish" buttons, and a list of actions including "Import from Commons", "Choose home page", "View Course Stream", "Course setup checklist", "New announcement", and "Student view". At the bottom right, it shows "Coming up" with a "View calendar" link and the text "Nothing for the next week".

<https://canvas.instructure.com/courses/1491035>





# Design Patterns & Software Architecture

---

# Software reuse is hard!



Up to now in your degree you have been taught the OO basics and some higher level principles...

# Definition Design Patterns

---



*A software design pattern is a **solution** to a commonly occurring **problem** in a **context**...*

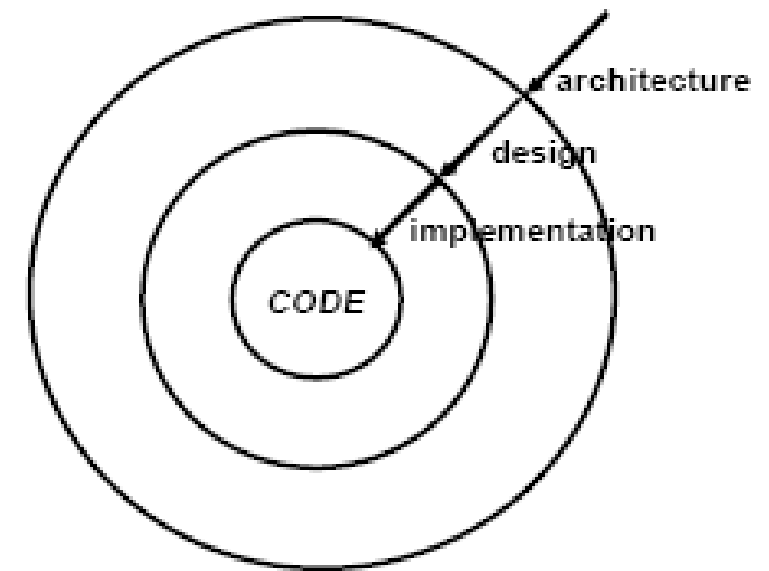


# Definition Software Architecture

Software architecture encompasses the set of significant decisions about the **organization** of a software system:

- Selection of the **structural elements** and their interfaces by which a system is composed
- **Behavior as specified in collaborations** among those elements
- **Composition** of these structural and behavioral elements into larger subsystems
- **Architectural style** that guides this organization

Philippe Kruchten, Grady Booch, Kurt Bittner, and Rich Reitman, s.d., Rational Unified Process (RUP) definition of Architecture



Architecture decisions are the most fundamental decisions. And changing them will have significant ripple effects!



# UML Refresher (if required)

---





# Just to know where we stand

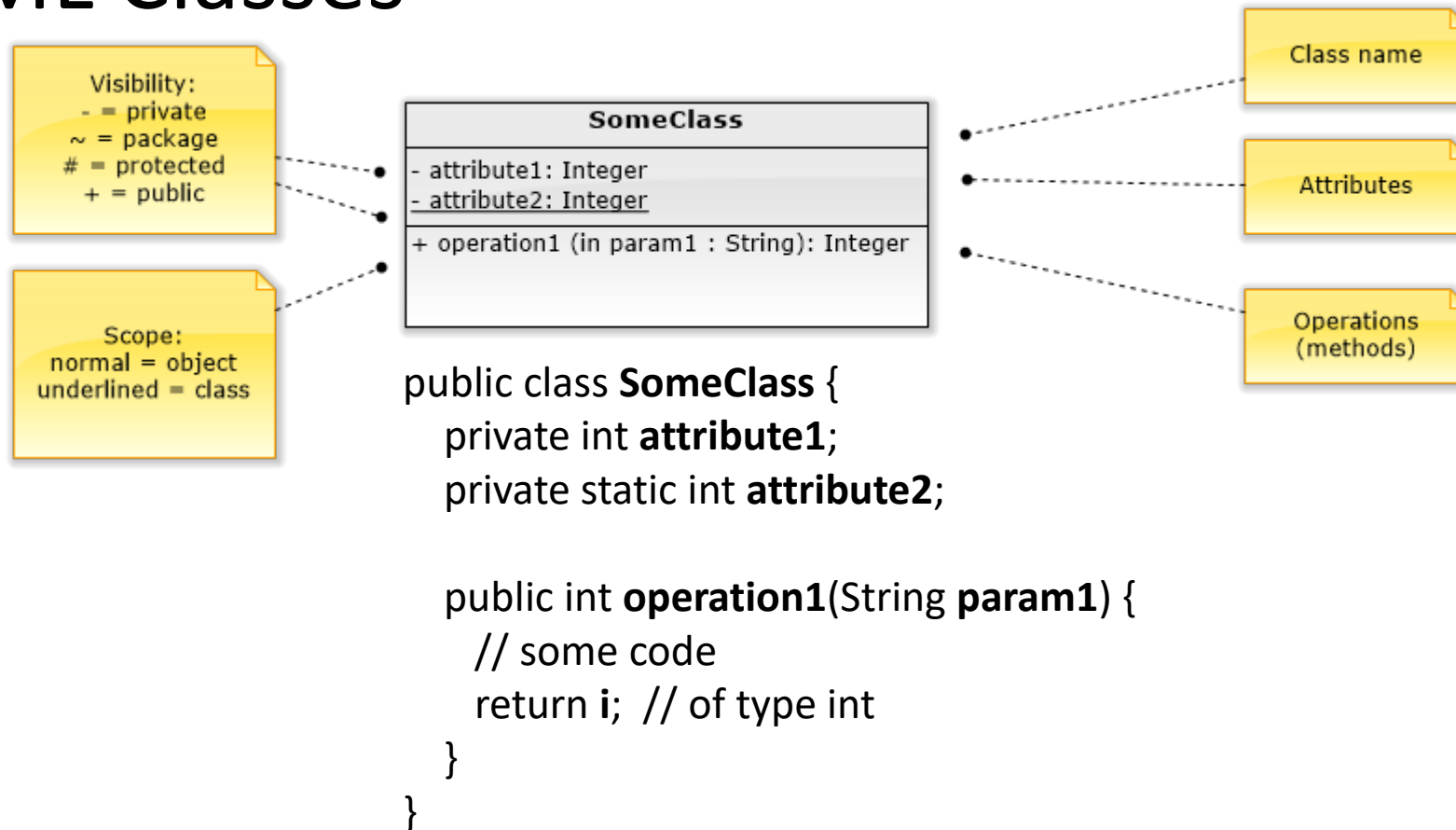
What do you know about...

- UML
  - Class diagrams
  - Class relations
  - Sequence diagrams
- Object oriented design
  - Abstraction
  - Encapsulation
  - Polymorphism
  - Inheritance
- UML tools

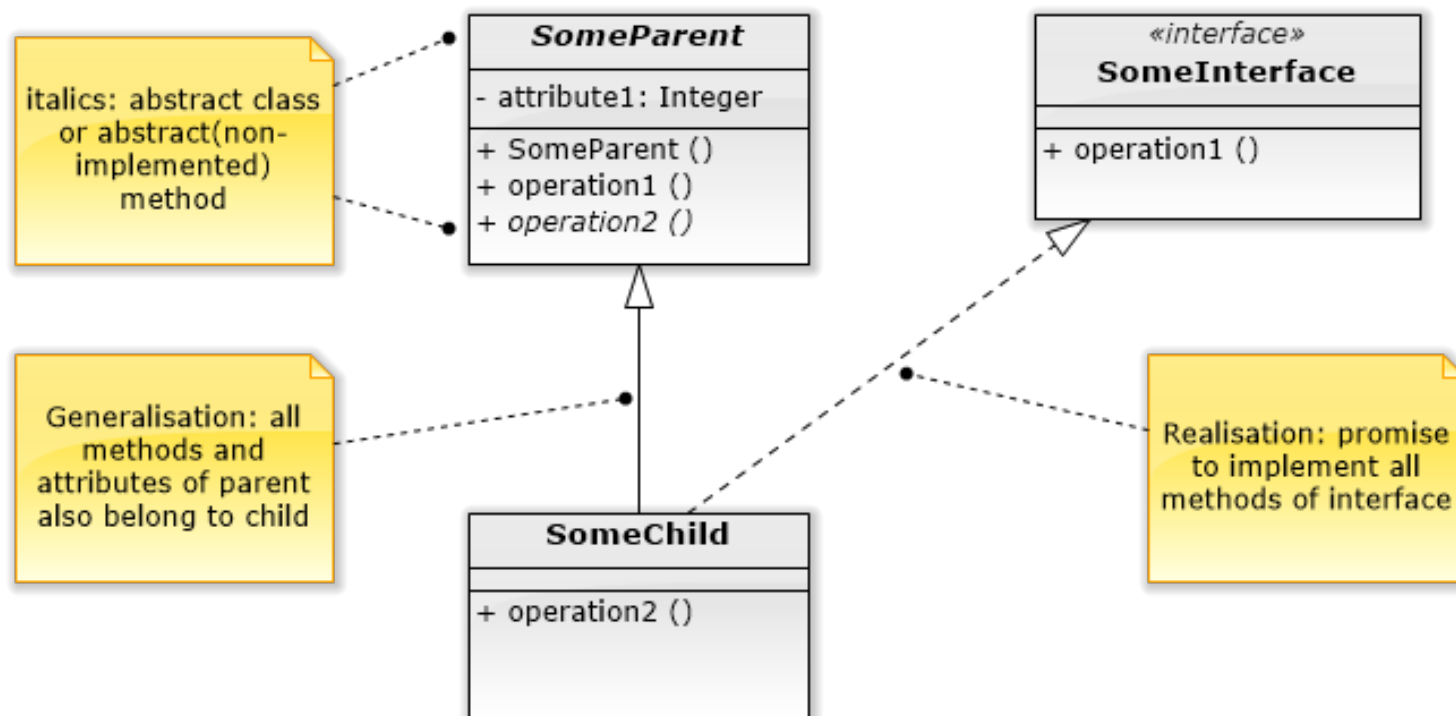
# UML Refresher



## UML Classes



# UML generalisation & realisation

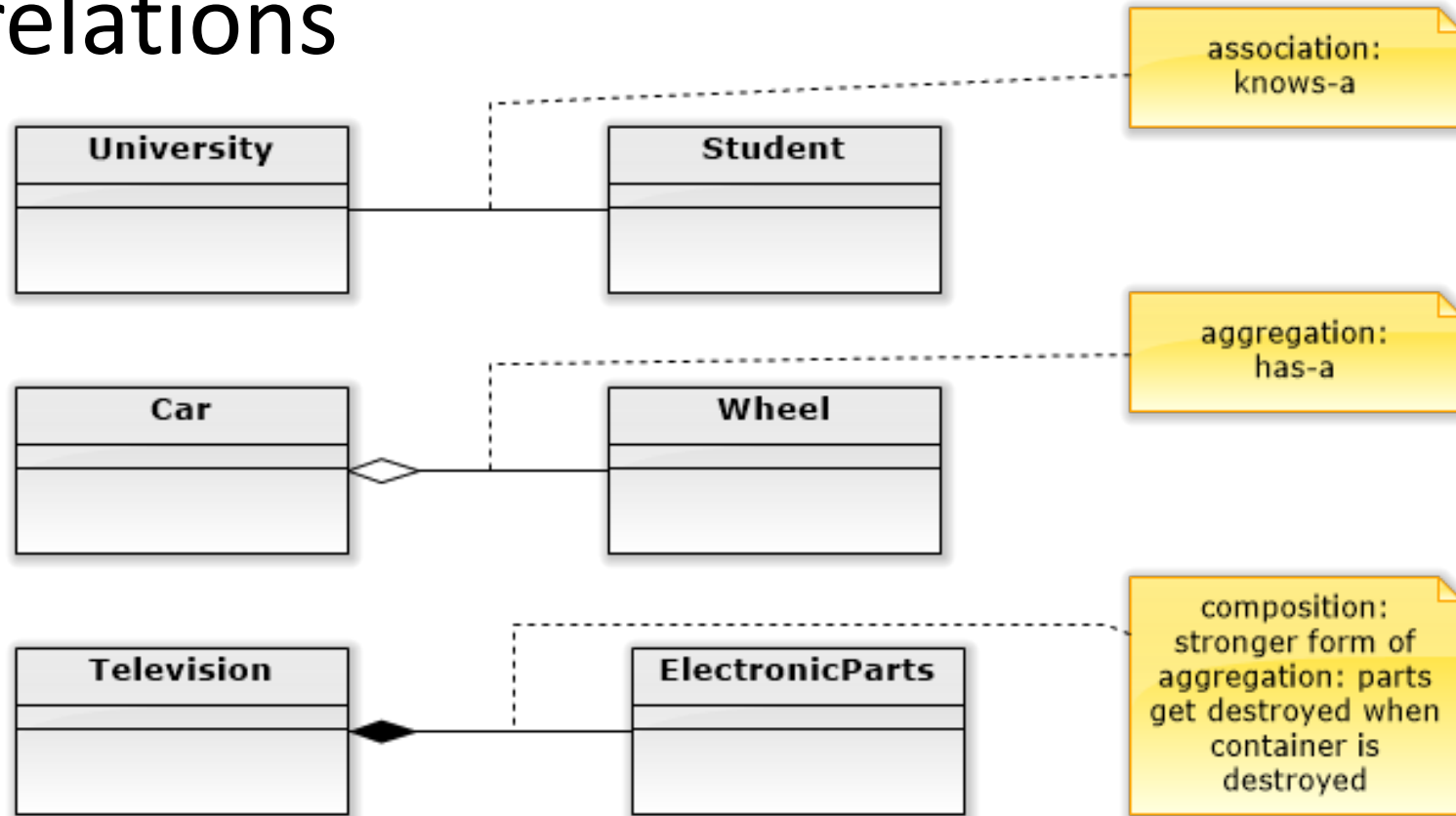


```
public abstract class SomeParent {
    //
}
```

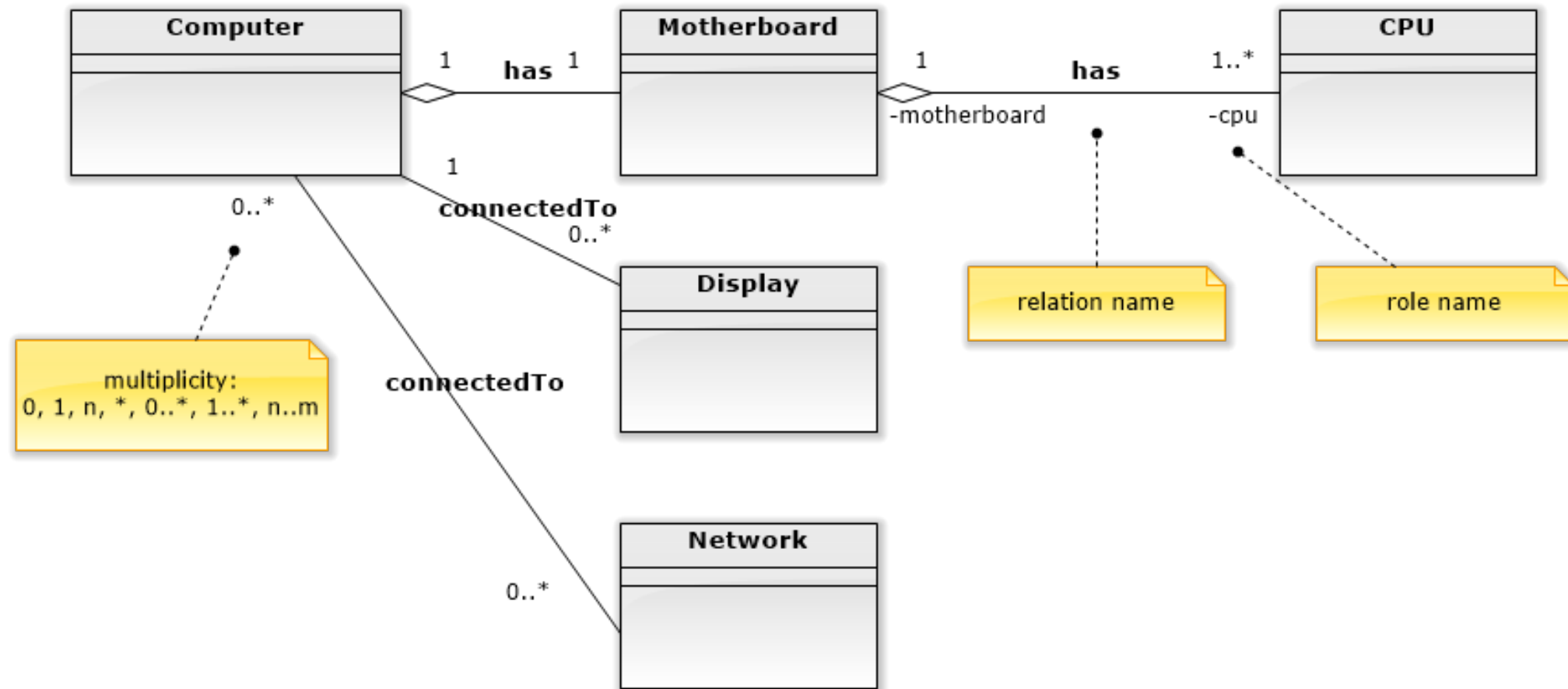
```
public interface SomeInterface {
    //
}
```

```
public class SomeChild
    extends SomeParent
    implements SomeInterface {
    //
}
```

# UML relations

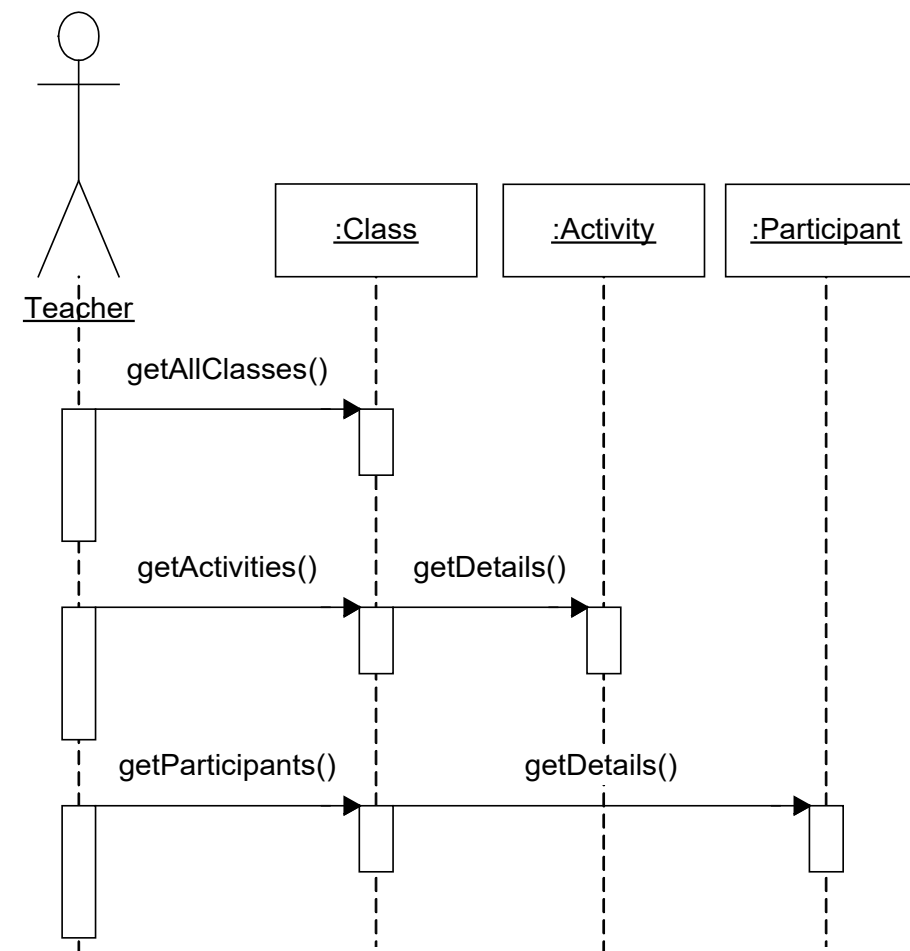


# UML relations continued





# UML sequence diagrams





# Course overview

---



# Overall content course

- Introduction (1 lecture)
  - Design Patterns (10 lectures)
  - Software Architecture (2 lectures)
  - Wrap-up by Joost Schalken-Pinkster (1 lecture)
  - Wrap-up by Song Jie (1 lecture)
  - Big examination (1 lecture)
- 
- Plus four lab sessions for the two practical assignments.
    - Early hand-in bonus: if you hand in the lab assignments on/before Dec 21th by the latest you can retake the lab assignments (with 10% penalty) if you do not like the score.



# Provisional lesson table 1/2

| Lesson (a 2 hours) | Content   |
|--------------------|---|
| 1                  | Joost, Netherlands, Course Introduction, Examination, UML recap   |
| 2 (Ch.1)           | Strategy, Encapsulate what varies, Favor Compos. over Inheritance, Program to interfaces                                |
| 3 (Ch.2)           | Observer, Strive for Loosely Coupled Designs,   |
| 4 (Ch.3)           | Decorator, Closed for modification, open for extension  |
| 5 (Ch.4)           | Factory method + Abstract Factory,<br>Depend on Abstractions not concrete classes, (possibly interning & copy on write) |
| 6 (Ch.5)           | Singleton, plus issues with singletons plus dependency injection frameworks   |
| 7 (Ch.6 & 7)       | Command, Memento & Adapter, example of text editor with undo  |
| 8 (Ch.10)          | State, plus other methods to implement STDs   |



# Provisional lesson table 2/2

| Lesson (a 2 hours) | Content  |
|--------------------|--|
| 9 (Ch.8)           | Template method, Only talk to your friends (Demeter's law), Don't call us we call you... |
| 10 (Ch.9)          | Iterator, One reason to change   |
| 11 (Ch.9)          | Composite  |
| 12 (Ch.11 + GOF)   | Chain of responsibility<br>Proxy (short, could be skipped)                               |
| 13 (Ch.13)         | Wrap up Design patterns (structure + gof), Intro Software Arch                           |
| 14 (Ch.12)         | MVC 1  |
| 15 (Ch.12)         | MVC 2  |
| 16                 | Wrap up/exam preparation   |





# Design Patterns covered (14-16 patterns)

## Creational Patterns

- Abstract Factory
- Factory Method
- Singleton
- ~~Builder~~
- ~~Prototype~~

## Structural Patterns

- Decorator
- Adapter
- Composite
- *Façade (time permitting also when discussing components)*
- *Proxy (time permitting)*
- ~~Bridge~~
- ~~Flyweight~~

## Behavioural Patterns

- Strategy
- Observer
- Command
- Memento
- Template method
- Iterator
- State
- Chain of responsibility
- ~~Visitor~~
- ~~Interpreter~~
- ~~Mediator~~

# Design Principles covered

