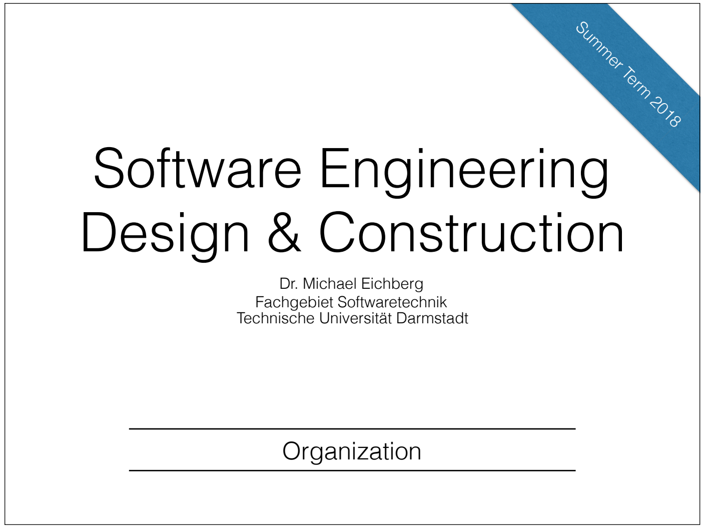


Summer Term 2018

Software Engineering Design & Construction

Dr. Michael Eichberg
Fachgebiet Softwaretechnik
Technische Universität Darmstadt

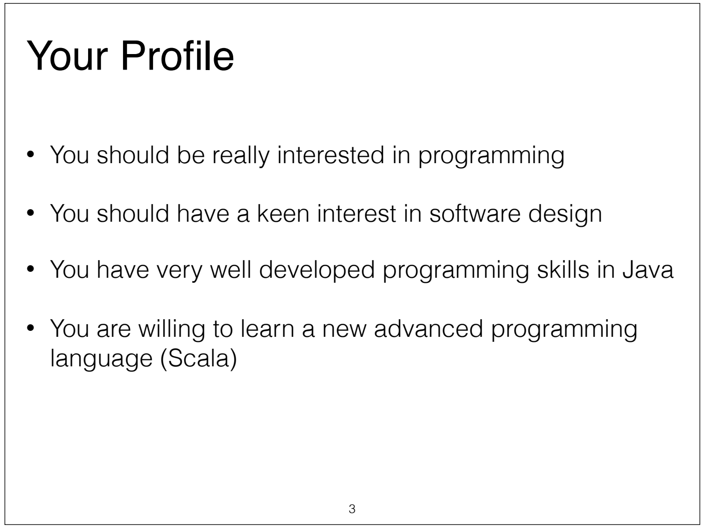
Organization

The goal of this lecture is to teach you fundamental software design principles that will foster your understanding of the intricacy when designing and developing software. It will help you to become a better software engineer.

Main Topics

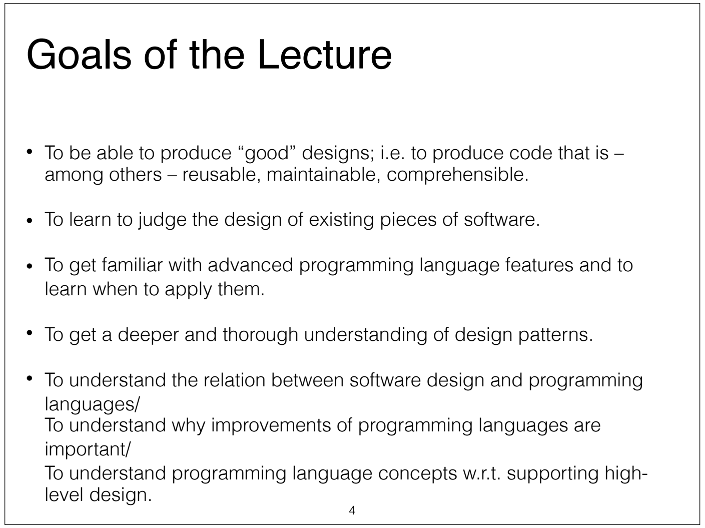
- Understanding Software Design and Development
- Software Design Principles
- Software Design Patterns
- Advanced Programming Language Features

… and the relation between programming language features and being able to solve “bigger and bigger” problems



Your Profile

- You should be really interested in programming
- You should have a keen interest in software design
- You have very well developed programming skills in Java
- You are willing to learn a new advanced programming language (Scala)



Goals of the Lecture

- To be able to produce “good” designs; i.e. to produce code that is – among others – reusable, maintainable, comprehensible.
- To learn to judge the design of existing pieces of software.
- To get familiar with advanced programming language features and to learn when to apply them.
- To get a deeper and thorough understanding of design patterns.
- To understand the relation between software design and programming languages/
  To understand why improvements of programming languages are important/
  To understand programming language concepts w.r.t. supporting high-level design.

After successfully completing this lecture you should be able to develop software that is well designed and which uses (advanced) programming language features to support the design.

Organization

Exercises

- We will have (in most cases) one exercise per week; most exercises will have ~10points. Exercise will start next week (Friday 20th.)
- You are required to submit your own, unique solution. (Discuss the tasks with others, but solve the tasks on your own.)
- All exercises will be automatically graded.
  (The grading will start after the deadline)
  - If you try to tamper with the system you're out.
  - We will compare submitted exercises *at some point in time* to filter to identify "shared" submissions.
  - In both previous cases, you won't get any bonus points at all.

Organization

Exercises

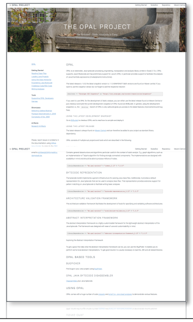
- The maximum bonus is equivalent to a full grade (1,0).
- You'll get the maximum bonus when you get 100% of all exercise points.
- The bonus for the exam is calculated by multiplying the percentage of gained exercise points with the number of points required to get a full grade better.

```
⌊exercise bonus (in %) * points required to get a full grade better⌋
```

Additional Bonus Points

Improving OPAL

- You can get additional (extra) bonus points by helping to improve/refactor/test the OPAL project.



OPAL PROJECT

Getting Started · ScalaDoc · Repository · Maven Central

THE OPAL PROJECT

Be Relaxed - Static Analysis is Easy

OPAL

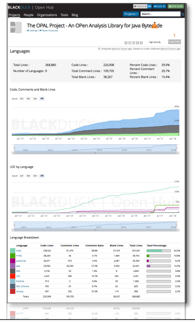
USING THE LATEST DEVELOPMENT SNAPSHOT

USING THE LATEST RELEASE

BYTECODE REPRESENTATION

ARCHITECTURE VALIDATION FRAMEWORK

ABSTRACT INTERPRETATION FRAMEWORK



BLACKDUCK | Open Hub

Projects · People · Organizations · Tools · Blog

The OPAL Project - An OPen Analysis Library for Java Bytecode

Languages

Total Lines :	368,880	Code Lines :	220,908	Percent Code Lines :	59.9%
Number of Languages :	9	Total Comment Lines :	109,705	Percent Comment Lines :	29.7%
		Total Blank Lines :	38,267	Percent Blank Lines :	10.4%

Code, Comments and Blank Lines

LOC by Language

Language Breakdown

Additional Bonus Points

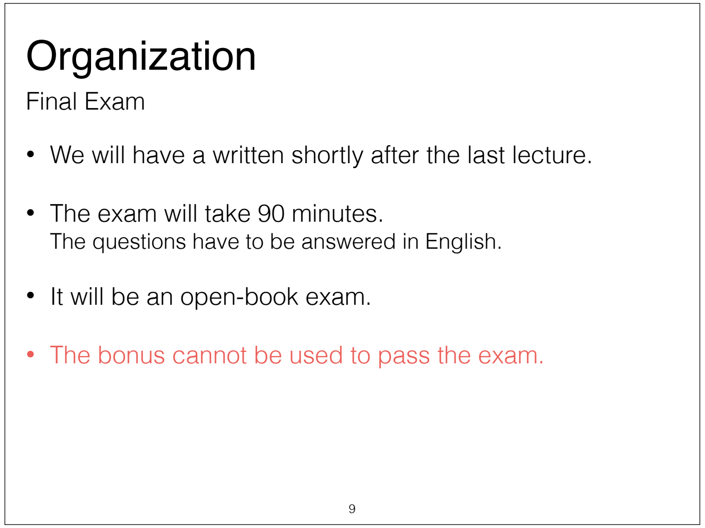
Improving OPAL

- For pull requests fixing (simple) typos you’ll get 0.1 bonus point per (not yet fixed) typo.

- For pull requests fixing significant documentation issues where the domain of the expected and/or returned value(s) is not correctly described, you’ll get 0.3 Points.

- For pull requests completely addressing an “IMPROVE [LX]” you’ll get X points.

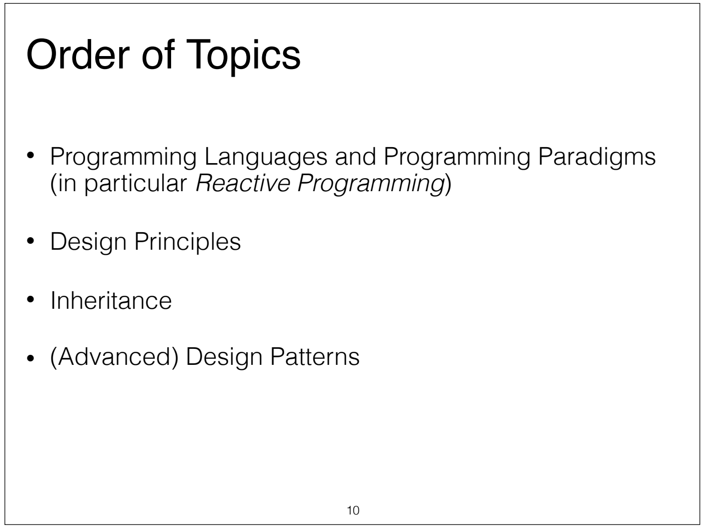
- Pull requests are processed on a first-come-first-serve basis



Organization

Final Exam

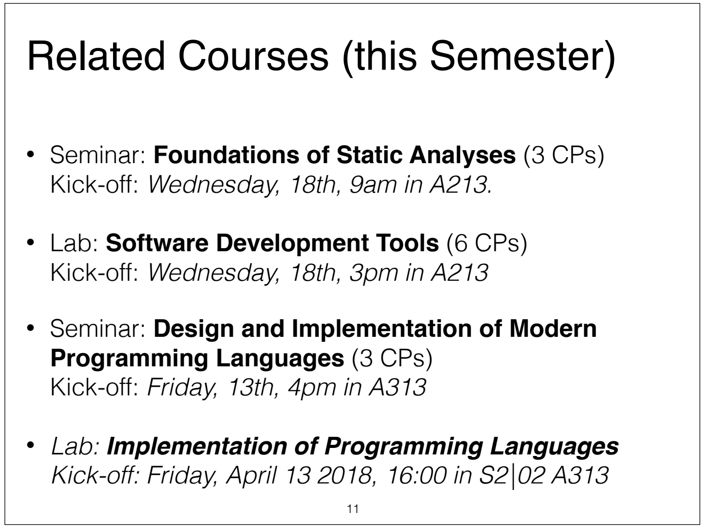
- We will have a written shortly after the last lecture.
- The exam will take 90 minutes.
  The questions have to be answered in English.
- It will be an open-book exam.
- The bonus cannot be used to pass the exam.



Order of Topics

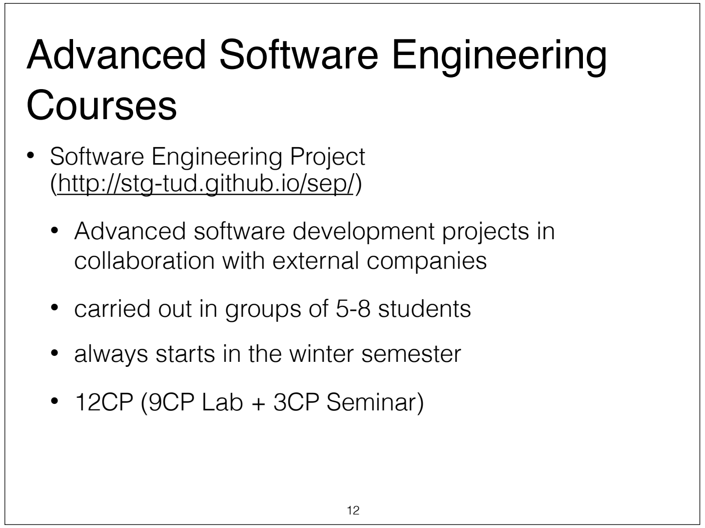
- Programming Languages and Programming Paradigms
  (in particular *Reactive Programming*)
- Design Principles
- Inheritance
- (Advanced) Design Patterns

The goal is to understand the effect of choosing a programming language on the (overall) design.



Related Courses (this Semester)

- Seminar: **Foundations of Static Analyses** (3 CPs)
  Kick-off: *Wednesday, 18th, 9am in A213.*
- Lab: **Software Development Tools** (6 CPs)
  Kick-off: *Wednesday, 18th, 3pm in A213*
- Seminar: **Design and Implementation of Modern Programming Languages** (3 CPs)
  Kick-off: *Friday, 13th, 4pm in A313*
- *Lab: **Implementation of Programming Languages***
  Kick-off: Friday, April 13 2018, 16:00 in S2|02 A313



Advanced Software Engineering Courses

- Software Engineering Project (http://stg-tud.github.io/sep/)
  - Advanced software development projects in collaboration with external companies
  - carried out in groups of 5-8 students
  - always starts in the winter semester
  - 12CP (9CP Lab + 3CP Seminar)