Welcome to
OO Project Course 2016

Joachim von Hacht

Participants

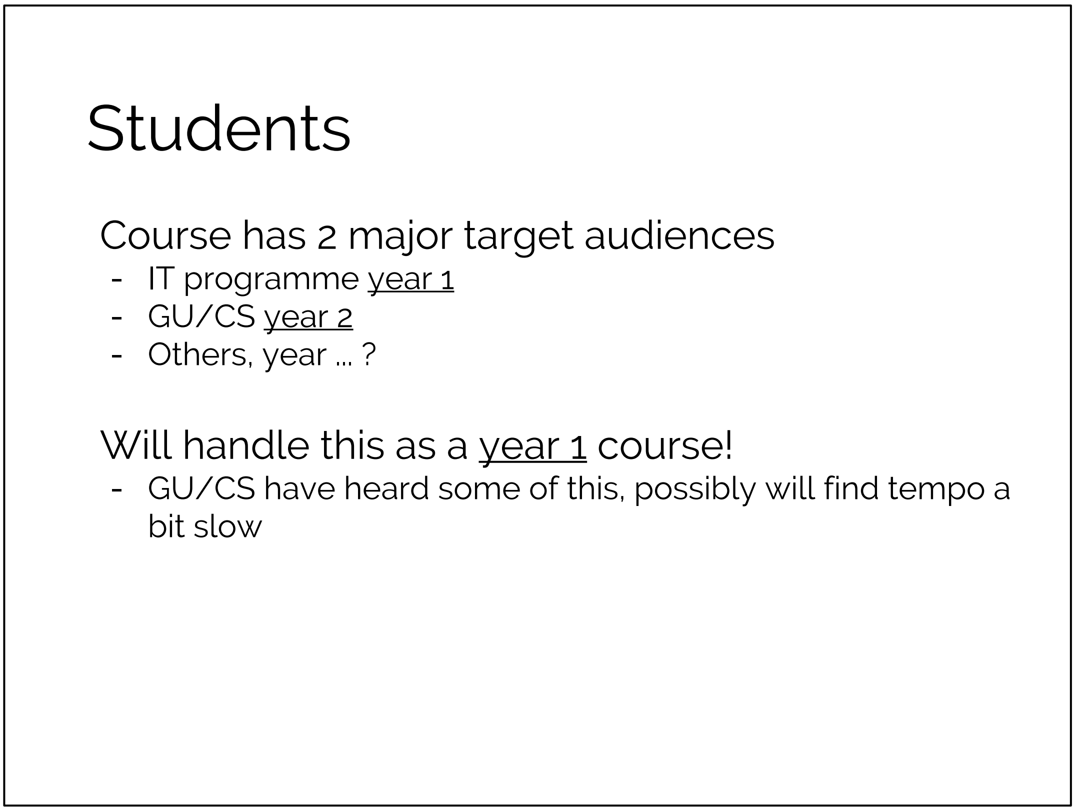
Lecturer, Course responsible, Examiner

- Joachim von Hacht, hajo@chalmers.se, 772 10 03

Assistants

- Adam Waldenberg, adam.waldenberg@gmail.com
- Julie Jarmar, juliajarmar@gmail.com
- Christer Carlsson, carlsson@chalmers.se
- Sebastian Blomberg, sebblom@student.chalmers.se

… and of course …



Students

Course has 2 major target audiences

- IT programme <u>year 1</u>
- GU/CS <u>year 2</u>
- Others, year … ?

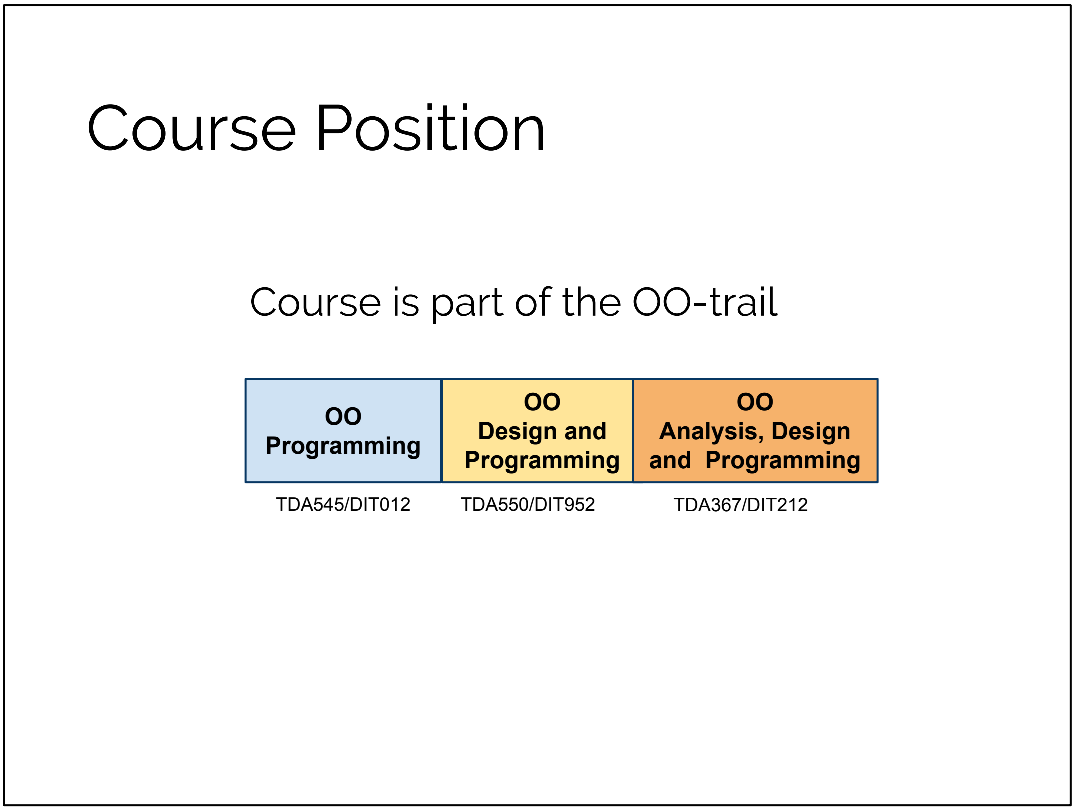
Will handle this as a <u>year 1</u> course!

- GU/CS have heard some of this, possibly will find tempo a bit slow

Must have passed TDA545/DIT011 or TDA550/DIT950.

If failed both preceding OO course, this is <u>not a good course to take</u>

- Take any programming (or other useful) course instead!
- IT-program has been informed



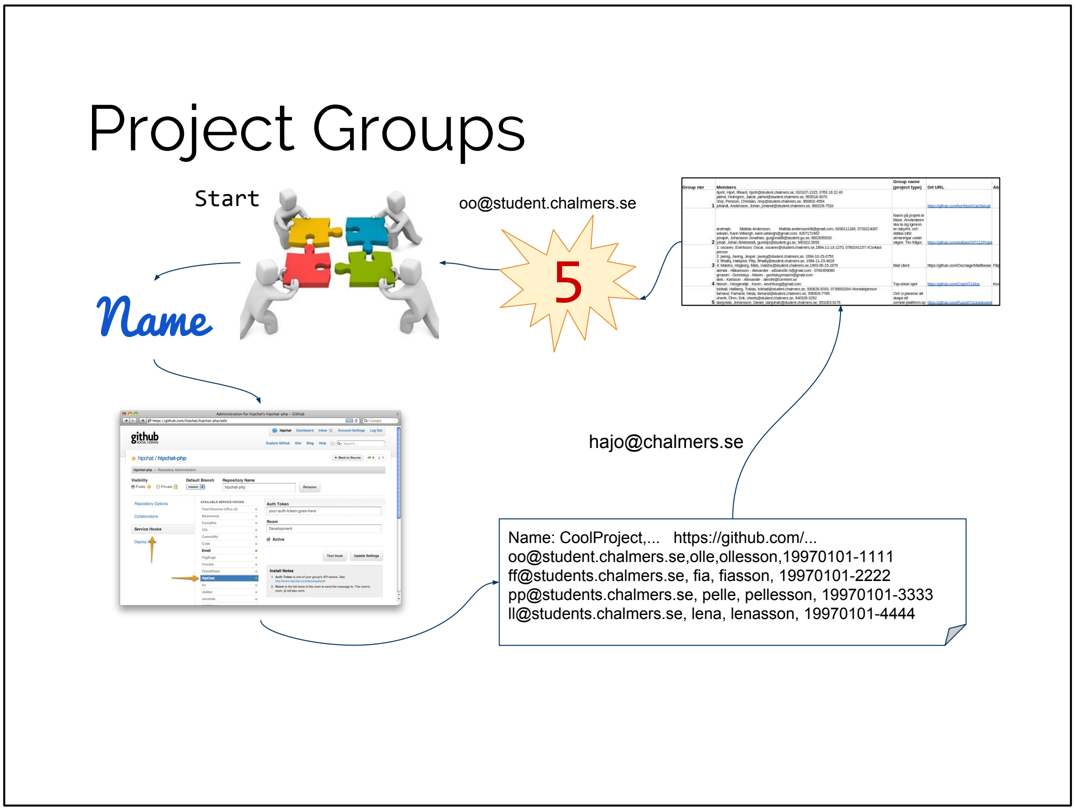
Course Position

Course is part of the OO-trail

OO Programming	**OO Design and Programming**	**OO Analysis, Design and Programming**
TDA545/DIT012	TDA550/DIT952	TDA367/DIT212

In real life reversed

- First OOA ...
- then OOD...
- and finally OOP



Project Groups

Name: CoolProject,...   https://github.com/...
oo@student.chalmers.se,olle,ollesson,19970101-1111
ff@students.chalmers.se, fia, fiasson, 19970101-2222
pp@students.chalmers.se, pelle, pellesson, 19970101-3333
ll@students.chalmers.se, lena, lenasson, 19970101-4444

See Course and Project PM for details.

- Must be done NOW (to be able to schedule groups for Thursday)

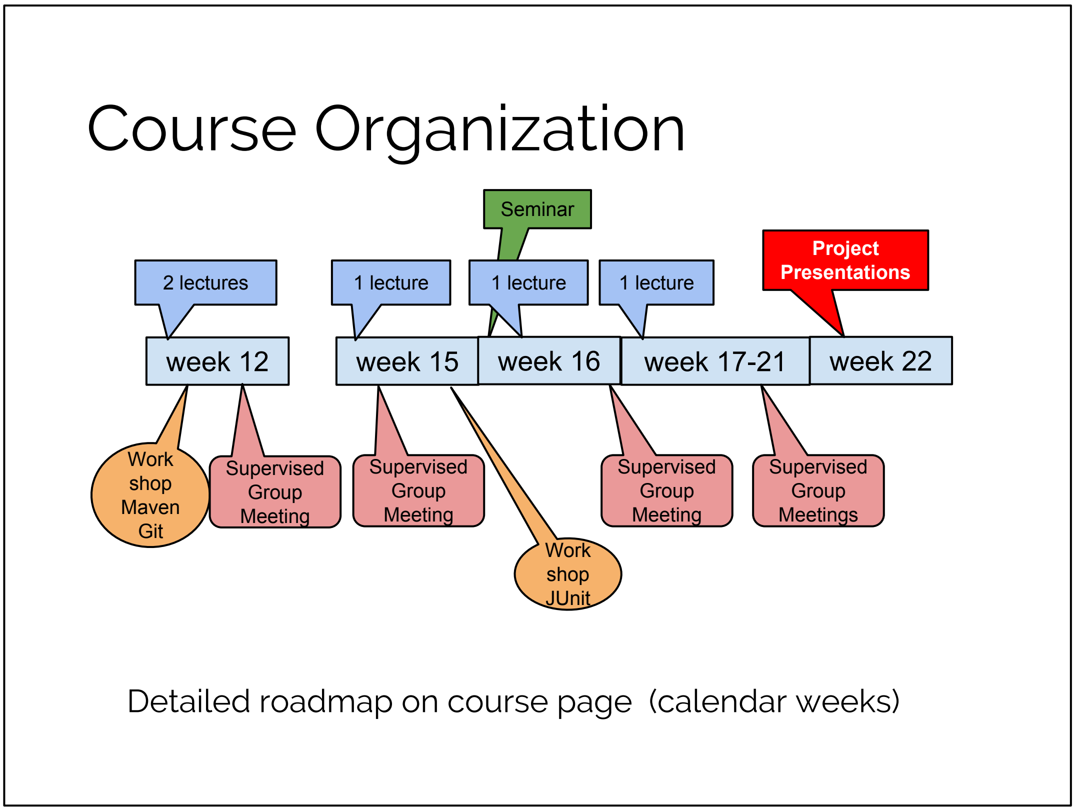
The Project

You will not be able to finish (you don’t need to)
- We expect a prototype (but of course much functionality will impress)
- When is an application finished?

Selected project normally not crucial for the grade
- Almost any project can be "complexified"
- Discuss with assistant
- If in trouble simplify, emulate, … (should be possible to back to previous version)

Expected application type is a standalone or mobile with a GUI
- Highest grade can be achieved by this

Of course you may create more technically advanced projects
- But it's not a prerequisite for highest grade



Course Organization

Seminar

2 lectures

1 lecture

1 lecture

1 lecture

Project Presentations

week 12

week 15

week 16

week 17-21

week 22

Work shop Maven Git

Supervised Group Meeting

Supervised Group Meeting

Work shop JUnit

Supervised Group Meeting

Supervised Group Meetings

Detailed roadmap on course page (calendar weeks)

Lectures
- The lectures will mostly be a real time demonstration of the software process you are supposed to use
- Some readings to do at home before lecture
- Slides after lecture

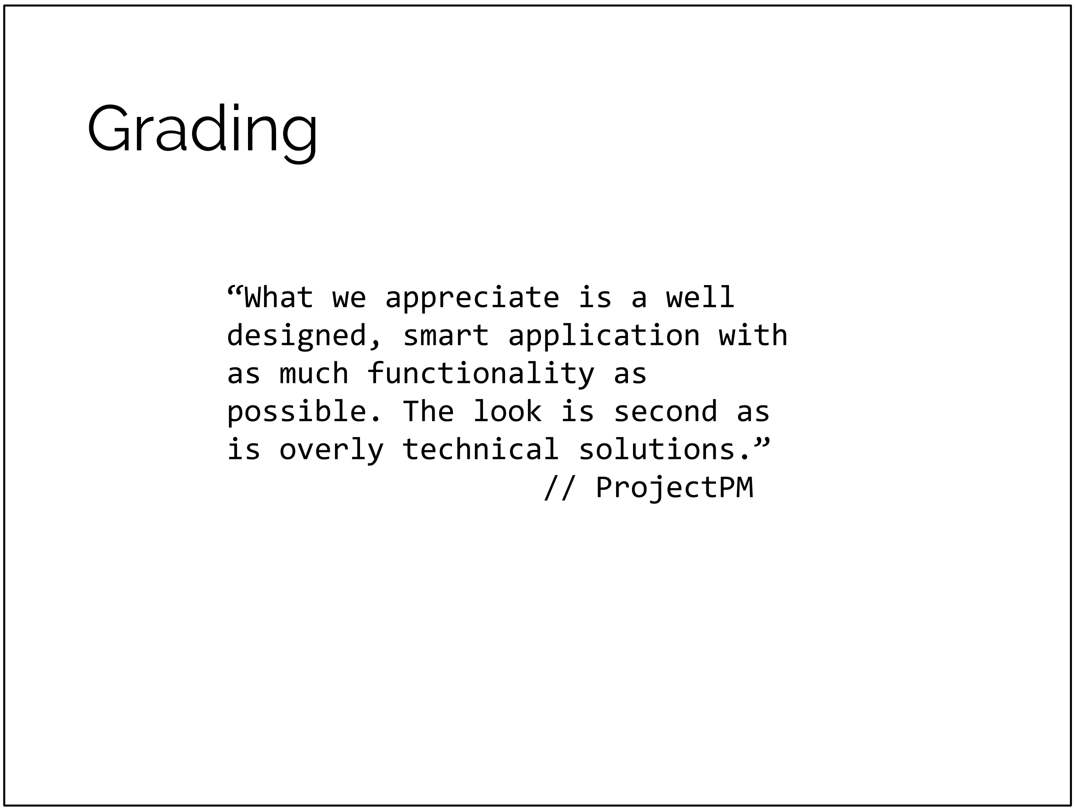
Supervised meetings 1h/week. Mandatory!
- Group + Assistant.
- Role of assistant is to help on an overall level (process, design, extensions, simplifications, ...)
- Not a bug fixer (assistants don't know all graphics libraries/physics engines...)
- You are supposed to push ... collect questions
- Any problems: Contact me

Workshops
- To get you going with some tools (Maven, JUnit, Git)
- Some scheduled lab session but should be self instructing
- Maven Git mandatory, must be presented.

Begin of study week 3 you will present a preliminary domain model for your application, more to come...
- Short ca 10 min.
- What are we going to do...
- Overview of the analysis model (more to come)

Project presentation (during exam week)

- Presentation is a part of the learning (and the grading)
    - Will give you perspective on your project.
- About 20 min
- See ProjectPM



Grading

> “What we appreciate is a well
> designed, smart application with
> as much functionality as
> possible. The look is second as
> is overly technical solutions.”
> // ProjectPM

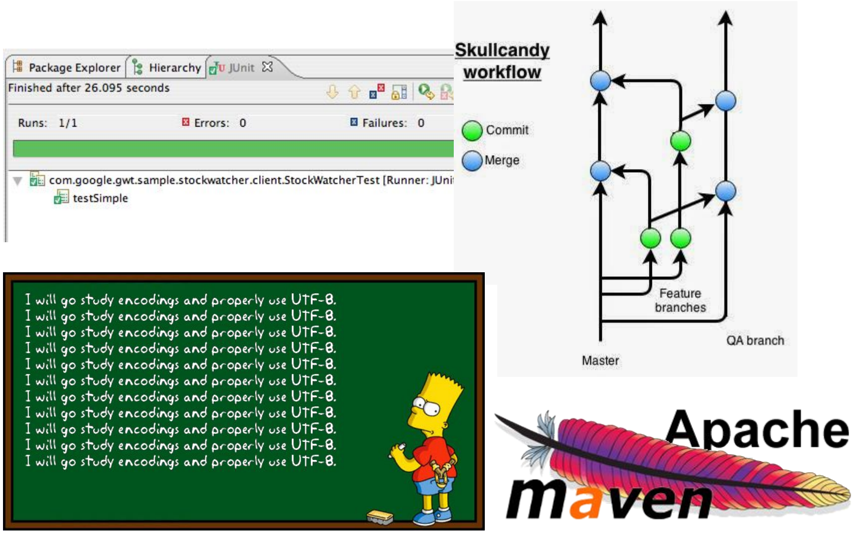
Step 1: The project
- The project will get an overall grade
- For criteria see Project PM (course page)

Step 2: Individual
- Each individual will get a grade
- Mostly project and individual grade will be the same but if we see big differences they can vary
- If varying grades the project mean should hold (or be close to)

What we expect from each individual
- Actively contribute to the process, attending meetings etc.
  - Ok, with different ambitions, group decide, speak out!
- About 800 SLOC
  - If less add some documentation to explain (hard problems, advanced techniques … etc.)
- You (and we) must be able to confirm your contributions!
  - Use gitinspector to check?

Some advices
- Take turns when committing to code version handling system
- Annotate classes with **@author** and use "revised by..."
- Document in agendas: Who is responsible for …
- You must make it possible for us to trace your participation and activity.

Basic Tools



Finished after 26.095 seconds

Runs: 1/1 Errors: 0 Failures: 0

com.google.gwt.sample.stockwatcher.client.StockWatcherTest [Runner: JUni

testSimple

Package Explorer
Hierarchy
JUnit
Skullcandy workflow
Commit
Merge
Feature branches
QA branch
Master
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
I will go study encodings and properly use UTF-8.
Apache
maven

Basic tools involved

- NOTE: All should set default file encodings to UTF-8
- Any IDE supporting Maven projects. Our standard IDE is: NetBeans 8.x.x. Many tutorials on the Web.
  - To start Netbeans in STUDAT (in terminal, Linux): /chalmers/groups/ws-devel/netbeans-8.0.2/bin/netbeans
- Maven, a project management and comprehension tool. Mandatory (bundled with NetBeans)
- Gradle, also a project management tool (optional)
- Git, a distributed version control systems (VCS). Mandatory
  - Being able to handle Git may be critical to the project (disasters may happen)
- JUnit, unit test framework. Mandatory (bundled with NetBeans)
- More optional tools later …

Questions