OO Project Intro

TDA367/DIT212

Joachim von Hacht

Course Intention

Exercise what you learned so far, i.e. creating a non-trivial modest sized application from scratch

- So far you have got (a lot of) starting help ...
- Now, you are on you own!

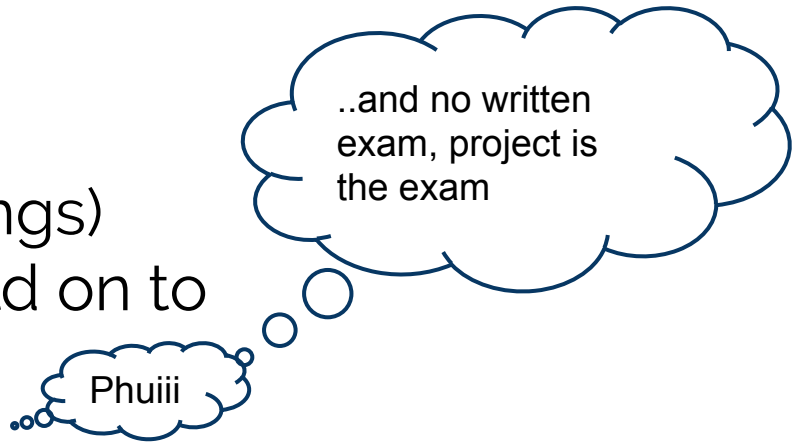
As you notice this course is spoken in Swedish and mostly written in English

Sorry..it's 212

Correction ...

Not completely on your own

- Have a group (4 stud/group)
- Have an assistant (weekly meetings)
- Will have a process model to hold on to
- ...so this will be great fun!



Last year was a fair

- Overall student impression about 4
- Improvements: Reorganization of lectures and rework of slides, reworking of doc. templates, reworked (and new) samples, improved info to assistants, a Git workflow, using Maven, ...

Target audiences

Course has 2 major target audiences

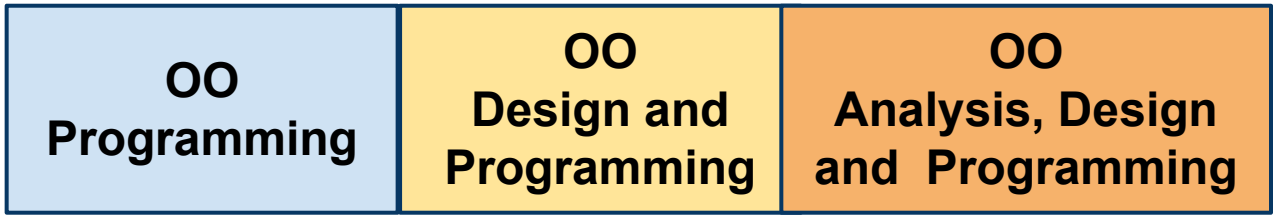
- IT programme year 1
- GU/CS year 2
- Others, year ... ?

Will handle this as a year 1 course!

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

Course Position

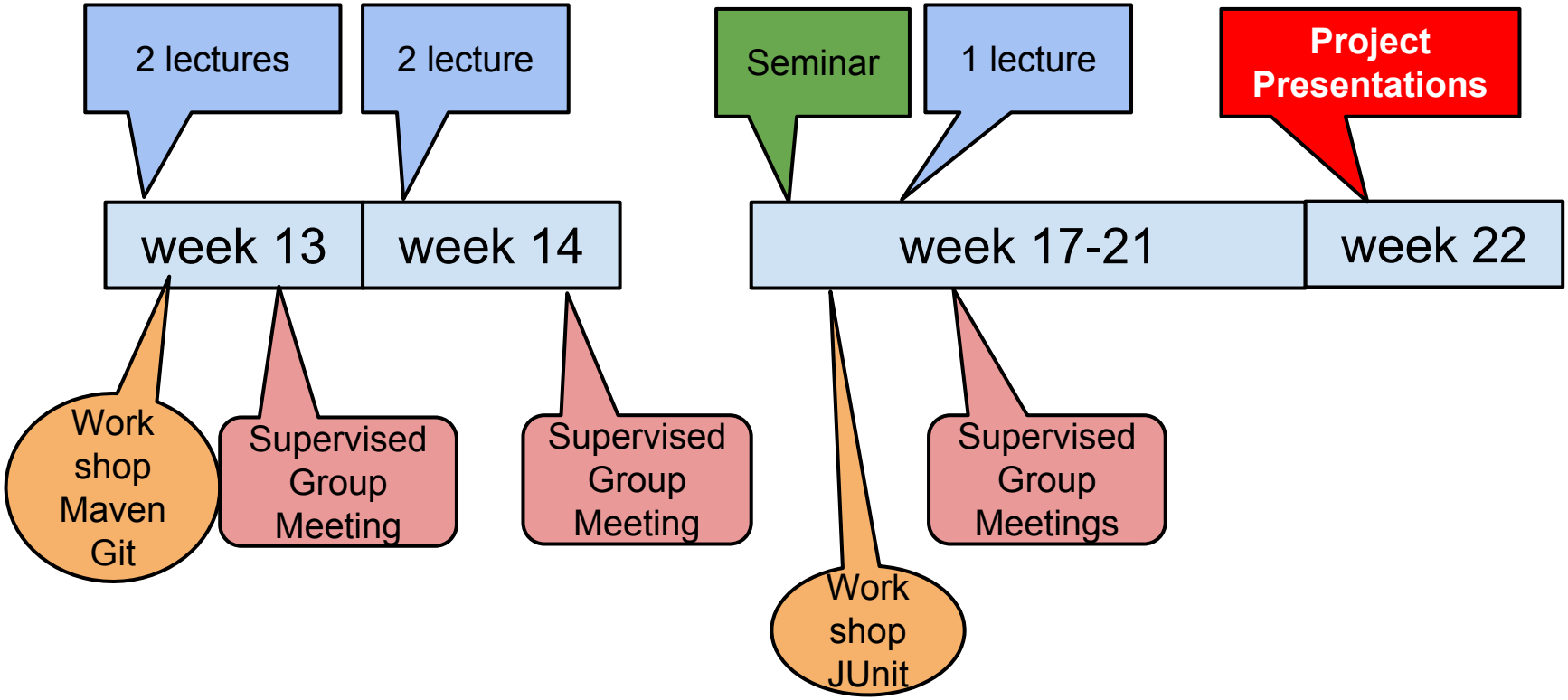
Course is part of the OO-trail



In real life reversed

- First OOA ...
- then OOD...
- and finally OOP (we do)

Roadmap and Organization



Detailed roadmap on course page (calendar weeks)

Lectures

Mostly as usual

- Only 4-5 lectures
- Will try to be somewhat interactive (because I'll try to explain a process)
- There are slides, I'll try to talk freely from the slides, possibly will not show each and every one, you look at home (slides on course page)
- Slides mostly will come after lecture (because of interactivity)

Tools

Quite a few tools involved

- 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)
- Git, a distributed version control systems (VCS). Mandatory
- JUnit, unit test framework. Mandatory (bundled with NetBeans)
- Others (quality tools): FindBugs, STAN, JDepend, Jacoco Gitinspector (Optional). More later.

Workshops

Designed to quickly get you going with the tools

- Content: Basic Git, Maven and JUnit.
- Workshops are self instructing, but assistant present just in case…
- Workshops optional, but if you can't handle a Git workflow you should attend (or at least do the workshop at home)

Could potentially save you hours and hours of frustration!!

Group Meetings

You are supposed to:

- Organize 2 weekly meetings on your own
  - The meetings should be documented (i.e. written agendas)
  - Template for agenda on course page
  - Use English or Swedish

- Attend one supervised meeting/week (1 hour)

Supervised Meetings

Group + Assistant.

- Role of assistant is to help on an overall level (process, design, extensions, simplifications, ...)
- Not a bug fixer
- If adding extra libraries you are on your own (assistants don't know all graphics libraries/physics engines...)
  - You are encourage to try (will give edge to project)

You are supposed to push ... collect questions (use an issue tracker)

Any problems: Contact me

Seminar

Begin of study week 3 you will present a preliminary analysis model for your application, more to come...

- Short 8-10 min.
- What are we going to do...
- Overview of the model (a class diagram, pick up your UML-skills...)

Presentation of Project

Presentation is a part of the learning (and the grading)

- About 15 min
- Running demo of project
- Technical walkthrough
- You will also act as an opponent for some other groups project, i.e. questions to the group regarding the project (technical solutions/alternatives etc)
- Must be present for one day, you're encourage to question (after opponents). Will give you perspective on your project.
- For IT-program presentation part of cooperation with "LSP310 - Kommunikation och ingenjörskompetens"

The project

You will not be able to finish

- When is an application finished?
- We expect a prototype (but of course much functionality will impress)

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)

Project Type

Expected application type is a standalone, end user application with a GUI

- Highest grade can be achieved by this

Of course you may create more advanced projects

- But it's not a prerequisite for highest grade
- Avoid overly technically complex project (involving databases, web servers, … there will be other courses)

Course Grading

Step 1: The project

- The project will get an overall grade

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)

See course and project PM on course page for more details

Project Grading

Criteria

- See Course PM and Project PM for details (course page)

Individual Grading

Actively contribute to the process, attending meetings etc

- Ok, with different ambitions, group decide, speak out!

Contribute to the project (i.e. code)

- **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 be able to confirm your contributions!
- Use gitinspector to check?

Who should Participate?

If failed both preceding OO course, this is not a good course to take

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

Questions

???