Project PM: OO programming project, TDA367/DIT212, LP4 2017

General (ASAP)

• Form a project group of 4 students and give the group a name.
• Setup the project site using the group name. The project must use the Git versioning system (use GitHub or other).
• Send a mail to course responsible (hajo@chalmers.se) with the following content;

- Group name
- URL to Git repository
- A preferred weekly time slot for meetings with teaching assistant, see course page.
- Info about all members in the following format

```
Lastname Firstname email pnumber, phone // Phone only for crisis
Lastname Firstname email pnumber
Lastname Firstname email pnumber
Lastname Firstname email pnumber
```

• You will get a group number as confirmation. Keep it (used for schedules, etc.)!
• Check course page for group meeting schedule (after confirmation).
• Start looking for some nice project. Project isn't normally critically related to the grade, almost any application can be ”complexified” by adding more requirements. Have a talk with assistant and/or Joachim if in doubt.

Requirements

All required documents and code should be downloadable from the Git repository. It's assumed that the branch ”master” is the final delivery.

The application

We expect a standalone, desktop or mobile Java end user application (i.e. a graphical user interface). Application should use “some kind of” MVC architecture. There must be an OO-model as the core of the application. The project must be possible to run on Win/Mac/Linux. If additional information is required to run the project there should be a README-file explaining how to do.

You may extend the application (client/server, database, etc.) but note;

- It's much harder to get a clean design when combining different technologies.
- It's not necessary to extend to get the highest grade.

You will not be able to finish (you don't need to)

- We expect a prototype (but of course much functionality will impress)
- If in trouble simplify, emulate, … (should be possible to back to previous version)

The documentation

All documentations should be in pdf format.

The RAD and SDD

Use templates from course site.

- A section in RAD or SDD should be short and concise probably require at most 1-3 paragraphs. If something isn't applicable just add a NA (not applicable) under the actual section (the sections are there to guide you, they are not mandatory).
- UML-diagrams should be on class/package/module-level. We are normally not interested in variables and methods except for methods in interfaces.
- For the dynamic model (RAD) we use sequence-diagram.
- No auto generated UML (other UML welcome).
- Javadoc not mandatory but helpful comments are appreciated.
- Use cases texts should be in RAD.

The group meeting agendas

Use templates from course site.

Project grading

It's very hard to formulate exact criterion for the project. There are a some examples of projects thats got the highest grade on course page.

We will consider the following (high to low priority):

- Domain and design models.
- Functionality: Number of working use cases, average complexity of use cases.
- Design: Separation of concerns, programming to interface, inversion of control, MVC, etc.
- Implementation: General technical level for implementation, Motivated use of interfaces abstract classes, generics, ... Motivated use of DP
- Tests: Unit tests and test coverage
- General quality: Comments from tools: Findbugs, STAN, Exception handling
- GUI
- Documentation: RAD and SDD. Class comments for all classes
- Agendas. Possible to follow process.

Your grade = Project grade +/- your contribution (mean grade for group should be stable)

- 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.