# Project PM: OO programming project, TDA367/DIT212, LP4 2014

## General

- The project group's first task is to decide on one of the project proposals on course page (or better if you have an idea of your own). This isn't critically related to the grade, almost any application can be "complexified" by adding more requirements.
- Second task is to name the project. The group number (assigned) and the name must be used in all communication with assistants, course responsible, etc.!
- The third task is to setup the project site. The project must use the Git versioning system use Google Code, GitHub or other.
- Final task is to send a mail to course responsible (hajo@chalmers.se) with the following content; Group name, selected project (very short description), URL to Git repository and lastly info about all members. The info format should be (phone optional);

```
cid, Lastname, Firstname, email, pnumber, phone
cid, Lastname, Firstname, email, pnumber, (phone)
cid, Lastname, Firstname, email, pnumber, (phone)
cid, Lastname, Firstname, email, pnumber, (phone)
```

## Requirements

This is what you are supposed to handle in. 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

Your group is supposed to develop a standalone desktop Java application with a graphical user interface. The application must use some kind of MVC design, either an "in-house" solution (you code it yourself, full control, but takes some time) or possible use some framework (lack of control but possible faster). Using suitable libraries or frameworks will put an edge on the project.

The project must be possible to run on Win/Mac/Linux. There should be s script to launch the application. If additional information is required to run the project there should be a README-file explaining what to do.

You may extend the application (client/server, database, et.c.) 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.

### The documentation

All documentations should be in pdf format. Standard templates should be used (see course site).

- The RAD and SDD
  - 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.

    INITIAL UML SKETCHING USING PAPER AND PEN IS OPTIMALLY EFFICIENT (use tool later to get a nice look)
  - For the dynamic model (RAD) it suffices with two sequence-diagram.
  - NO auto generated UML (other UML welcome).
  - NO Javadoc.
- The Use cases
  - There should be at least 4 documented use cases (texts).
- The group meeting agendas.

## Project grading

It's very hard to formulate exact criterion's for the projects but if a project consists of less than 2804 SLOC[1] (source lines of code) including comments the project should have other qualities that compensate for the "smallness" of the project. That is, at least 701 SLOC/person. The official SLOC calculator is gitinspector (link on course page).

### Examples

Below are some typical examples (non exhaustive).

### U (U)

Not possible to run. Project to small, to few use cases, design to strange/bad (hard to understand). Documentation missing or poor. Not possible to trace any process from agendas.

[1] This is a very crude measurement, but it gives you some idea of what we expect

### Grade 3 (G)

A project fulfilling the base requirements for all grades. Which means: Functionality; 5-10 use cases implemented and working. Simple but functional GUI. In-house MVC. Clean implementation of at least one subsystems, application uses interfaces to reduce dependencies. There are some usable tests. Documentation is short but correct (and in sync. with the application). It's possible to trace requirements and follow the process. The presentation is ok.

### Grade 4 (G)

(NOTE: This is not the union of ..., it's a list of possibilities) A somewhat larger application with a more sophisticated design, possible use of (needed) design patterns. Solid code and packaging, everything is easy to locate. Clean subsystems. More functionality (use cases) implemented, possible attention to non-functional requirements. Good control over dependencies, clean interfaces. Possible use of external libraries. A more advanced GUI. External configuration and data. Test suites cover a lot of application code. Documentation is short and correct and obviously useful for others. The presentation gives a good view of the strength and weakness of the application.

### Grade 5 (VG)

Like grade 4 but even more and with higher technical level. Possible some kind of modular design with plug-ins, advanced use of frameworks.