Assignment 3
Refactoring the design of an existing project

Software Design & Modeling

Submit by: Wednesday, 26 November 2025 at 23:00

1 The assignment

Your assignment in a nutshell:

1. Pick a software **project** that you have access to. This can be written in any language, and it can be:
   - an open-source project available on GitHub or other public repositories,
   - a project you developed in the past (for example an assignment or course project), or
   - any other project whose source code is available and that you can share.
2. Identify a **part of the project** whose design can be improved. For example, the implementation of a certain *feature* is scattered over many classes; or a group of *classes* are poorly organized and hard to change. The project part that you target should have significant size (no toy example) but also be *feasible* given the amount of time you have for the assignment (no huge code bases).
3. **Refactor** the project part to improve its design without changing its behavior.
4. Document your refactoring work, and its motivation, in a **report**.

   Maximum length of the report: 7 pages (A4 with readable formatting) including any pictures and code snippets.

The assignment must be done in *individually*.
This assignment contributes to **28%** of your overall grade in the course.

2 Resources and references

You can browse opens-source projects that are not too large and look for **issues** that mention *refactoring* or *anti-patterns*. A few examples:

- https://github.com/opendatakit/collect/issues/507
- https://github.com/Dissem/Jabit/issues/26
- https://github.com/Ben-Wolf/IBEIS-web/issues/24

The following *exercises* on refactoring can provide inspiration for the kinds of refactoring tasks you may want to undertake:

Legacy code retreat trivia game: https://github.com/jbrains/trivia

Refactoring katas: https://github.com/marcoemrich/Refactoring-Katas

Note these exercises are only suitable as preparatory work, not as projects for the actual assignment – which should target more realistic code.

2.1 Amount of work

There are no strict guidelines on how large the project you target for refactoring should be. You can follow one of three strategies, or any combination of the first two strategies:

- Pick a large project, select a manageable subset (e.g., a package or just a few classes), and refactor that in depth.
- Pick a large project, select a few features (e.g., naming, the definition of a fundamental data type, . . .), and refactor those features across the whole project.
- Pick a smaller project, and refactor it in depth.

Tentatively, you can aim for refactoring that targets at least 1000 lines of code. Ideally, you would implement a combination of "shallow" refactorings (e.g., renaming of identifiers) together with some "deeper" refactorings (e.g., revisiting parts of the inheritance hierarchy).

3 What to discuss in the report

The structure of the report is free; it should include all the significant findings emerged during your work.

Things to include in the report's discussion:

- the project selection process, including possibly mentioning projects you discarded; notice that choosing a suitable project, with a variety of features that lend themselves to refactoring, is an integral part of the assignment's work

- the goals of your refactoring: what aspects of design you want to improve, and why they need improving
- a few concrete examples of refactoring (showing *before* and *after*)
- a presentation of what you did to ensure that your refactoring did not change program behavior (such as running tests, code inspection, . . .)
- a discussion of what was harder and what was simpler to refactor
- an assessment of what else could be improved in the project's design (if anything) but didn't address in your work

4 How and what to turn in

Turn in:

- Your **report** as a single PDF file in *iCorsi* under *Assignment 3*
- The **source code** of your project *before* and *after* refactoring it in the Git repository at `https://gitlab.com/usi-si-teaching/msde/2025-2026/software-design-and-modeling/assignment3-your-last-name/`.

  Use *branches* to store the *before* and *after* versions:

  1. add the project code *before* refactoring to the `main` branch
  2. create a new branch called `refactored`
  3. apply your refactorings to the content of the `refactored` branch

  We will grade the most recent commit in the `refactored` branch before the project deadline.

4.1 Plagiarism policy

Students are allowed to generally discuss assignments and solutions among them, but each student must work on and write down their assignment independent of others. In particular, both sharing solutions of assignments among students and reusing solutions of assignments done by students of previous years or by anyone else are not allowed and constitute cheating.

In a similar vein, you are allowed, in fact encouraged, to look up any examples and material that is available online you find useful; however, you are required to write down the solution completely on your own, and, if there is publicly available material (such as a website, blog, paper, or book chapter) that you based your work on, credit it in the report (explaining what is similar and how your work differs).

ChatGPT & Co.

The plagiarism policy also applies to generative AI tools such as ChatGPT or CoPilot:

1. You are allowed to get the help of such tools; however, you remain entirely responsible for the solution that you submit.

2. If you use any such tools, you must add a section to the report that summarizes which tool(s) you used and for what tasks, what kinds of prompts you used with the tools, how you checked the correctness and completeness of their suggestions, and what modifications (if any) you introduced on top of the tool's output.

Failure to abide by these rules, including failing to disclose using AI tools, will be considered plagiarism.