# Effective Software Testing Lab (Assignment 1)

- Deliverable: One single zip file containing all the folders that you find attached to this assignment, augmented with the tests you will write for each problem, as described below. Additionally, a `Documentation.md` file where you document your decisions and report test coverage requested nelow.
- Deadline: March 25, 2024 at 18:00 (Zurich, CH, time).

## Forming Groups

The assignment should be submitted by groups of *four* students. If you do not have a group yet, join one in OLAT. Carry out the assignment **with your group only**. You are allowed to discuss solutions with other groups, but each group should come up with its own personal solution. A strict plagiarism policy will be applied to all the artifacts submitted for evaluation.

## Exercises

This assignment consists of eight (8) exercises. Each exercise has its own folder and contains a problem definition and the corresponding solution in Java. **Some solutions intentionally contain bugs**. Your task is to test the solutions, according to the instructions given in the Tesing section below. If your tests reveal any bugs, you must fix them until all the tests pass.

The IntelliJ IDEA is strongly recommended, and you can downloaded for free here. Optionally, as a student at UZH, you have the opportunity to obtain a free one-year IntelliJ IDEA license through JetBrains' student licensing program. Sign up here, or log in if you already have one. Apply for the free student license using your @uzh.ch email address provided by the University of Zurich.

Importing an Exercise

To import an exercise in IntelliJ IDEA, click "File"->"Open" and select the exercise folder.

Structure of an Exercise

Every exercise contains a `src` folder, which contains at least two files, `ExerciseName.java` and `ExerciseNameTest.java`. The `ExerciseName.java` file contains the solution which you have to test, and the `ExerciseNameTest.java` file is the file where you have to write your tests. If your test reveal any bugs, fix them by modifying `ExerciseName.java`.

In addition to the `src` folder, each project contains a `README.md` file that describes the problem, and a `pom.xml` file that contains the Maven configuration. Maven is a tool for simplifying the process of building Java code (see here for more).

Finally, you have to create a `Documentation.md` file in the root folder where

you document your testing decisions and report the requested coverage (a single `Documentation.md` should contain the documentation for all the exercises).

Requirements

Java 11 is used for all the exercises. There is no guarantee things will work with other versions. Use JUnit 5 for implementing your test suite.

## Testing

You are asked to perform effective and systematic software testing for each exercise's solution. Specifically: 1. Perform **specification-based testing**, following the principles taught in the book and the lectures. Document each principle in the `Documentation.md`. If you find a bug, report the bug, the test that revealed the bug, as well as the bug fix.

2. Enhance the previous test suite using **structural testing**. Specifically, aim for maximizing *condition+branch* coverage, which can be measured using the JaCoCo plugin. Document the process in the `Documentation.md` file, by reporting which conditions, branches, or lines did you miss with specification-based testing (if any), and what tests did you add to cover them.

3. Now that you have a good testing suite, augment it further using **mutation testing** (you will need PItest and PItest plugin for JUnit 5). Report the mutation coverage in the `Documentation.md`. Explain whether the mutants that survive (if any) are worth writing tests for or not. If the surviving mutants are more than three (3), choose one from each mutation category (*mutator* in PITest terminology).

## Grading Criteria

Each assignment will be graded with either 0 (fail) or 1 (pass). In order for an assignment to score 1, it must: 1. Provide clear evidence of effort. **and** 2. Provide a clear `Documentation.md` file **and** 3. Provide code that fulfills the requirements of the assignment.

Please note that the lab assignments are propaedeutic to a proper preparation for the final exam, i.e., the lab work should be considered as important not only for the bonus points, but also for preparing the final exam.

## Note:

Using Generative AI tools in the assignments is allowed, given that you deliver an appendix in the `Documentation.md` file with all the prompts you used.