Object Oriented Architectures & Secure Development

Week 2

First things first

Make sure you have a working `jBamaflex`-application. This means your have at least a `Course`-class and a `Student`-class, which allow you to enroll students in courses and grade them. The grades should also be retrievable.

A *working* application means you can prove the features above are behaving as expected in both the happy cases as the unhappy cases (something goes wrong). The only way to prove this is by having JUnit-tests...

Roadmap

We do not provide you with a roadmap on how to solve the problem described below. We only give you the following hint: **do not try to make this excerices in one go!** Rather, try to reach the goal incrementally. The assignment will ask you to read and write files in different directories, for example, but we suggest to try reading from and writing to **a single** file first (or do the directory logic first, but not at the same time).

Do not wait with writing tests until you think the assignment is ready. Rather, write (some) test before you start coding *the real stuff*.

Finally, make sure to put all code that has, even remotely, something to do with files in a separate class. Courses, nor students have any relation to files, it is only we (the developers) who *want* to store the information in files.

Actual assignmnent.

Write all information of a course to a single file and make sure it is possible to retrieve that information as well (i.e., read the data from file into a `Course`-object:

- General course information (e.g., name, ECTS-points, ...); and
- Grades per students.

Before coding, think about the file format you want to use: it should be human readable and writable, and it should be easy to enroll new students. (Do not worry about grading students later on).

It is common for courses to be tought for multiple academic years. Therefore, group the course data in directories per year, so you cna use the title of the course as name fof the file.

```
2020-2021
|- concrete I
|- concrete II
|- concrete III
|- metal and wood
|- Intro to building
2021-2022
|- concrete I
|- concrete II
|- concrete III
|- wood and metal
|- Introduction to building
2022-2023
|- concrete I
|- concrete II
|- concrete III
|- wood
|- metal
|- Introduction to building
```

When writing the data of course to file, only write the minimally needed information of a student to that file (e.g., do not include telephone number etc). When reading this data back from file you won't have sufficient information to construct a full student (or lecturer). Do not worry about that (for now), but just create a student with a name only for instance.

Students

Now, make the details of a student (birthdate, telephone nr, name, ...) also writable and readable from file. Here, there is no need for humans to read or write theses files; and you may assume student names are unique (or find another strategy to name your files).

Finally, forsee the functionally to read **all students at once** from the `students` directory.

Students and courses

Now, refine the course-from-file-implementation and use the code from the previous part to `find` the student for storing in the `Course` -object instead of makeing a student-with-a-name-only.