Department Mathematik/Informatik, Abteilung Informatik
Software & System Engineering Group
Software Technology Group
Lecture Object-Oriented Software Engineering, SS 2024

Alexander Korn, Adrian Bajraktari,
Prof. Dr. Michael Felderer


# Project Sheet 5. Fifth Sprint

**Hand out: 05.07.2024, 18:00**
**Hand in: 12.07.2024, 14:00**

**Remark**: This is the last sprint and thus the last sheet. The above hand-in date is fixed. By then, you should have all sprints completed.

## Exercise 1. This Sprint's Work

This week, we want to add more functionality concerning students.

a) Begin by merging the updated template from `https://github.com/lxndio/oose-24-project` into your project. Review sheet 3 for information on how to do that.

b) As students can now be added, we also want to the ability to edit existing students. As mentioned on the last sheet, the same view as for adding students can be used to accomplish that. Similarly to adding students, create a GET and a POST endpoint allowing to edit students. Use the existing validator to check if the modified data is correct.

c) Next, we want to add functionality for enrolling students to courses and deleting such enrollments. To do this, you'll need to complete the following steps:

- Refactor the `Enrollment` model. An enrollment uses a composite key consisting of a student ID, and a course ID. We need to combine both of those attributes to a single key to be able to create a repository for enrollments. This is necessary for adding and removing enrollments.
  - Create an `EnrollmentId` model with the attributes `course_id`, and `student_id`. Both should have the `@JoinColumn` annotation. Add a sensible constructor.
  - Replace the IDs in `Enrollment` with the `EnrollmentId`.

**Hint:** In Spring, you can create composite keys such as `EnrollmentId` by annotating the class with `@Embeddable`. When using such a class as an actual ID, the attribute must be annotated with `@EmbeddedId`. As you'd still want to access the entities references by the key, in case of a foreign key, you can annotate them as shown in the following example for `Enrollment`:

```
@ManyToOne
@MapsId("course_id")
private Course course;
```

- Create an `EnrollmentRepository`. The repository should contain a method that allows removing enrollments by student and course.
- In the student controller, add three endpoints with the following functionality:

| Endpoint | Type | Request Parameters | Description |
|---|---|---|---|
| `/student/enroll` | GET | `id`, `semester` | This endpoint should open the view `enrollment` with a view model containing the following attributes:<br>• `student`: The student object with the given `id`<br>• `enrollments`: All enrollments of the student in the given `semester`<br>• `semester`: The given `semester`<br>• `courses`: All courses that the student can enroll in (given their study subject) |
| `/enrollment/enroll` | GET | `student_id`, `course_id`, `semester` | This endpoint should create a new enrollment for the given student, course, and semester. After that, the endpoint should open the `enrollment` view. This can be done by using the previous endpoint's function instead of filling the view model again. |
| `/enrollment/delete` | GET | `student_id`, `course_id`, `semester` | This endpoint should delete the enrollment of the student in the given course during the given semester. |

**Hint:** When creating a new enrollment, make sure set the student and course ID in the composite key, as well a to set the actual student and course references for the corresponding attributes in `Enrollment`.

**[6 points]**

$\sum$ **6.0 points**