Java Technologies - Lab 4

[valid 2025-2026]

Project: PrefSchedule
The goal of this project is to design a system that automatically assigns students to optional courses based on their preferences, course capacity, and instructors' preferences.
The system should ensure a fair distribution of students using a stable-matching algorithm.
We will continue working on this project during the next labs.

Java Persistence API

Compulsory (1p)

- Create a Spring Boot Project that has support for Spring Data JPA, and PostgreSQL.
- Using an SQL script, create tables for:
  - students (id, code, name, email, year)
  - instructors (id, name, email)
  - packs (id, year, semester, name)
  - courses (id, type, code, abbr, name, instructor_id, pack_id, group_count, description)

  A course can be compulsory or optional. Optional courses are grouped in packs. A student will be assigned exactly one course from each pack in their year.
- Create the entity class and a repository for students.
- Implement a simple test using a CommandLineRunner.

Homework (2p)

- Create all the entity classes. Use `@OneToMany` and `@ManyToOne` associations. (You may use Lombok, but it is not necessarily required.)
- Think carefully if `Student` and `Instructor` should inherit a base abstract class.
- Create Spring JPA repositories for all entities.
- Think what queries may be necessary and define them in the repos: use at least one query based on a specific JPQL string, one derived query, and one transactional, modifying query.
- Create a `@Service` class for each entity, that uses the corresponding repo.
- Use Java Faker to populate the database and test CRUD operations for courses.

Advanced (2p)

- Use Criteria API to implement a dynamic search for courses, with multiple optional filters (e.g., "which optional courses are in year 3, semester 2", or "which courses are taught by a specific instructor")
- Implement first-level caching using Spring Cache and measure read performance improvements.
  Use a micro-testing or benchmarking library, such as JMH, for measuring the execution time of methods. (You need to handle JIT compilation, warm-up, and measurement accuracy correctly.)