TDD Assignment: Managing Courses

Guillermo Román Díez

Programming Project

Escuela Técnica Superior de Ingenieros en Informática
Universidad Politécnica de Madrid

October 11, 2018

1 Norms

- The deadline for finishing the assignment is on **Sunday 4th November 2018**. Modifications done after this date will not be considered.
- For correcting the project, it will be cloned from the `GitLab` URL provided in the assignment registration form, thus, check that you have the correct version for the deadline date.
- A system for detecting plagiarisms will be used and the groups involved in plagiarism will be penalized with a failing grade. According to the UPM norms, other disciplinary measures can be adopted in case of plagiarism.

2 Tools

The use of the following tools and libraries is mandatory:

- `Java` 1.8 (or greater)
- `GIT` as version control system
- `JUnit` 5 for test automation
- `Maven` as build tool

2.1 `GitLab`

Remember that the use of `GitLab` as remote GIT server is mandatory. The `GitLab` server of our course is available at the URL:

`http://costa.ls.fi.upm.es/gitlab`

The `GitLab` project used to develop the assignment must fulfill the following requirements:

- The *Project visibility* must be **private**. If your code is accessible by other groups is your responsibility.
- All members of the group must be registered in `GitLab` . The `GitLab` username must be the *left part* of your `@alumnos.upm.es` email. For instance, given the email `email.example@alumnos.upm.es`, the `GitLab` username must be `@email.example`

- Both members must have, at least, *developer* role for accessing the project.
- Email notifications are mandatory and must be sent to both users from the beginning of the project.
- The username `@pproject` must have access to the project with *developer* role.
- In order to monitor the activity of the project, the email `programming.project@fi.upm.es` must be included in the notification emails from the beginning of the assignment.

The repository must include a README.md file to include relevant information to use the system, mainly a description of the interface and how to use the public methods of the program. Remember that the format of the README.md file is *Markdown*[1].

2.2 `Maven`

The use of `Maven` as build tool is also mandatory and the file `pom.xml` file must be in the root directory of the project. The project must follow the standard directory layout of Maven and details of this layout can be found at:

`https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html`

The `pom.xml` must include the dependencies needed to work with `JUnit` 5.20 and it must work without any manual intervention, that is, after cloning the GIT project, the command `mvn test` must properly work passing all tests of the system.

2.3 `JUnit`

It is mandatory to develop your own test suite using `JUnit 5.20` to check the correct behaviour of the program developed. Take a look to the *Testing* slides to read the good practices in testing. Remember that all tests must be automatically run by the `Maven` commands. Those tests that do not executed automatically by means of the `Maven` command will not be considered.

3 Managing Courses and Students

The goal of the practical assignment is the development of a Java program (without graphical user interface) to manage the students that are enrolled in the different courses offered by the university. The program does not require any kind of graphical user interface, it is only needed to develop the classes required to implement the capabilities described below. The use of a Java *interface* defining all methods needed to interact with the program is strongly recommended and the description of its methods should be included in the `README.md` file.

The following list summarizes the requirements of the system that must be implemented:

1. A **new course** can be registered. The information associated to a course is its *code* (a numeric value), its *name* (a `String` value) and its *coordinator* (a `String` value). The following checks must be done before registering the course and a proper exception must be thrown when they are not satisfied:
   - The course code is not duplicated in the system.
   - The code, the name and the coordinator cannot be blank.
2. A **new student** can be registered. The information associated to a student is its *identification number* (a numeric value), its *name* (a `String` value) and its *email address* (a `String` value). The following checks must be done before registering the student an a proper exception must be thrown when they are not satisfied:

[1] `https://www.markdownguide.org/`

- The identification number is not duplicated in the system.
- The identification number, the name and the email cannot be blank.
- The format of the email address must be correct, that is, it must contain the character `'@'` and it cannot end in the character `'.'`.

3. The system can **enroll a student in a course**, by using the identification number of the student and the course code. The following restrictions must be checked and an a proper exception must be thrown when they are not satisfied:
   - The student must be registered in the system.
   - The course must be registered in the system.
   - A course could have, at most, 50 students matriculated.
   - A student cannot be enrolled in the same course twice.
4. Given a course code, the system must return the **list of matriculated students**, returning its identification number, name and surname. The list must be sorted by the identification number. If the course code does not exist the system must throw an exception.
5. The student can **cancel** its enrollment in a course. The system must check if the student is registered in the system and matriculated in the course and throws an exception when it does not happen.
6. A course can be **restarted** and this operation must remove all students matriculated in the course. If the course code does not exist the system must throw an exception.
7. The **list of all users registered** in the system, including its name, email and identification number, can be obtained. The list must be ordered by its name.
8. The **list of all courses**, sorted by their code, can be obtained.

3.1 Recommendations

- Follow the practices of *Test Driven Development* described in the lectures. Start the development thinking in the interface of the system and preparing a list of test cases for this interface.
- The system must produce *logging* information. The use of *log4j*[2] is recommended, avoid using `System.out.println` for printing debugging information.
- Do not push to `GitLab` binary files or files than can be generated automatically (.jar, .log, ...)
- Push your files frequently to avoid conflicts and integration problems.
- Avoid commiting your changes with only one `GitLab` user, each member of the group should push his own work.

[2] `https://logging.apache.org/log4j/2.x/`