Laboratory/Project 2 – Project Assignment

(May 26, 2018/2019)

1. Goals

In this project, students should be able to apply concepts of analysis, modelling and object-oriented programming to develop a Java application to manage services requests from clients and the activity of service providers. In compliance with good practices learned and applied during the first part of the semester, on the course units Software Engineering (ESOFT), Programming Paradigms (PPROG), Discrete Mathematics (MDISC), Computational Mathematics (MATCP) and Lab/Project 2 (LAPR2), an iterative and incremental development process is used. The main core of the software should be implemented in Java. To increase the maintainability of the software analysis best practices, Object-Oriented (OO) software design must be adopted and implementation should follow a Test-Driven Development (TDD) approach. Technical documentation must be produced during the project.

2. Project Description

The project that will be developed in this course is the continuation of the project started in the Software Engineering course (ESOFT). In ESOFT the students developed an application to manage services requests from clients and the activity of companies dedicated to providing these services.

In LAPR2 course the students will start by implementing the application developed in ESOFT (called AGPSD). The students should update all artifacts generated during the software development process and code all the application in Java programming language.

Additionally, in LAPR2 course the students will implement a set of scheduling algorithms, add some functionalities to the AGPSD application and will implement a new application that will be used by the Service Provider.
Methods that are used to send e-mails will not be implemented by the students.

2.1 Extensions to the AGPSD application

After each service has been executed, the Service Provider (SP) should use the AGPSD application to report the end of the work. The SP should see the details of the service and should evaluate how the service ran. If the service was executed as stipulated, the SP only acknowledges the end of the service. In cases where something unexpected happened the SP should introduce a description reporting the issue and troubleshooting strategy. At any time, the client should use the AGPSD application to rate the service (scale is 0 - 5) and see the invoice.

The Human Resources Officer (HRO) should use the system to evaluate the SP. The HRO should see charts (a histogram showing SP rating distribution) and statistics showing the performance of each SP. Among these statistics, the application should compute: the mean and standard deviation

of the ratings given to each SP and to all SPs; the absolute differences (deviation) between the average ratings of each service provider and the mean rating of all service providers; using the population mean rating, for each provider the application should label as “Worst Providers”, “Regular Providers” and “Outstanding Providers”, all providers whose mean is, respectively, less than $\mu - \sigma$, between $\mu - \sigma$ and $\mu + \sigma$, and above $\mu + \sigma$. Here μ represents the population mean and σ is the standard deviation. After seeing these statistics the HRO should accept or change the labels given to each SP.
The mean rating and the classification (label) of each SP should be seen by the client every time the scheduling algorithm assigns a SP to a service requested by the client.

The AGPSD application should include two algorithms for job scheduling: First-Come First-Served and Random Scheduling. The SP that will be scheduled to do the service will be selected using, firstly, the SP rating, secondly, the distance from the client.

The external service used to Specify the Geographical Area (UC5) should be implemented using geographical coordinates (latitude and longitude) of each zip code. The latitude and longitude of each zip code will be provided as a Comma Separated Values (CSV) file (resource available in Moodle). To reduce the computational complexity (memory and run-time) and easy the implementation and testing, only a subset of zip codes corresponding to Porto region will be used in this work.

2.2 Service Provider Application

The Service Provider will have an application to see the service execution orders. Therefore, this application should include a functionality to import execution orders. Moreover, this application should allow the SP to see and sort records by any field describing the service execution orders (name of the client, distance from SP facilities to client’s home, service category, service start date and time, type of service and client’s address). Moreover, the application should allow the SP to see the historical (timeline) of services for each client.

2.3 Non-functional requirements

The software development process should be iterative and incremental while adopting good design practices and coding standards. The students should use the software development process introduced in ESOFT.

All the algorithms implemented in this work should be highly efficient.

Both applications should be implemented in Java and the user interface should be implemented using JavaFX, a Java library used to build Rich Internet Applications.

The implementation process must follow a TDD (Test Driven Development) approach. Unit tests should be developed to validate all domain classes. Code changes must follow the same criteria, i.e. when changing existing components, unit tests must be developed or updated. When developing Input/Output (IO) methods for files, unit tests are recommended but not mandatory under the LAPR2 project. The final evaluation of the project will include an analysis of the quality

of testing and the use of a test-driven development approach. Code coverage and mutation coverage is based on unit testing and is performed using quality assurance tools.
In this project all members of the team should use Jenkins and Sonar Cube for continuous integration and code quality assessment, respectively. For task and issue management the team should use Trello.

3. Deliverables

This section describes all the deliverables necessary for the project. Failure to comply with these may invalidate proper assessment of the project.

Your project should always be up to date on Bitbucket. The version that will be assessed is the one with the commit time closest to the deadline. Nevertheless, all team members should submit the following files on Moodle. At the end of the project (June 16, 11.55 p.m.), students must submit on LAPR2's Moodle a final delivery that should contain the following zip files:
1. Project's Wiki, in a single ZIP file, named LAPR2-YYYY-GXXX-WIKI .zip containing technical documentation;
2. The Project Report, named LAPR2-YYYY-GXXX-Report.pdf;
3. The project repository (all folders in the repository), in a single ZIP file named LAPR2-YYYY-GXXX-Application.zip.

4. Assessment

Assessment is performed everyday while students are in class and receive immediate feedback and through a final project assessment. Therefore, class attendance is mandatory.

4.1 Class Assessment

Each class is assessed by the teacher in the classroom. Such assessment is based on the students' attendance to class and their performance. Class attendance of students with worker-student status will not be considered in the assessment.

4.2 Commit Messages

Git commit messages should include a description, a keyword and the task/issue number.
Examples:
- [Unit Testing] - commit introduces changes to Unit Testing
- [Implementation] - commit introduces changes to Implementation
- [UC-XX] - commit introduces changes to a Use Case XX
- [fixes issue #YY] - commit is associated to Trello Issue YY

Example commit message: "Add class Exhibition [UC-01] [Implementation] [fixes issue #10]."

4.3 Plagiarism

Any attempt of copy or plagiarism (using third-party code) not explicitly mentioned in the report, will be heavily penalized and may lead to project annulment. Failure to comply with these policies and procedures will result in disciplinary action.

5. Relevant Hyperlinks

- LAPR2
  - https://moodle.isep.ipp.pt/course/view.php?id=6435

- Bitbucket
  - https://bitbucket.org/

- Jenkins
  - https://jenkins.dei.isep.ipp.pt/
    - Note: login with student number (e.g. 1010101)

- SonarQube
  - https://sonarqube.dei.isep.ipp.pt
    - Note: login with student number (e.g. 1010101)

6. Revision History

Date	Description	Type
27/5	"Moreover, the application should allow the client to see the historical (timeline) of services for each client." → "Moreover, the application should allow the SP to see the historical (timeline) of services for each client."	**typo**