Laboratory/Project 2 – Project Assignment

Payments Management

(May 23, 2019/2020)

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 make and manage organization payments to freelancers. 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 the implementation should follow a Test-Driven Development (TDD) approach. Technical documentation must be produced during the project.

2. Project Description

In the ESOFT course the students developed an application for the T4J startup company that allows companies to publish tasks and manage the process of assigning those tasks to freelancers. In the LAPR2 course the students will develop an application, also for the T4J company, but that is completely independent of the application developed in ESOFT (it is a completely new application!) and that will be used to manage payments made by organizations to freelancers.

The main goal of this project is to develop an application that allows organizations to make and manage payments to freelancers. Moreover, the application should enable the T4J administrator to monitor both the payments made by the organizations and the performance of the freelancers.

The payments application should allow the collaborator of the organization to use the graphical interface of the application to create payment transactions. To create a payment transaction, for each finished/executed task the collaborator should introduce a brief description of the task (id, brief description, time duration (in hours), cost per hour (in euros) and task category), details about the execution of the task (end date, delay, brief description of the quality of the work) and information about the freelancer that worked on the task (id, name, level of expertise, e-mail, NIF, bank account (IBAN), address and country). Moreover, after introducing each transaction the collaborator should check the amount to pay to each freelancer. The amount to pay to the freelancer should be automatically computed. The collaborator of the organization can only add transactions that are associated to tasks and freelancers that already exist in the system. When the information describing the task and the freelancer is not available in the system, the collaborator should first create a task and a freelancer in the system. The algorithm that will be used to compute each payment includes the time duration of the task, the cost per hour and the degree of expertise of the freelancer (junior and senior). The execution delay is not considered to compute the payment value. A senior freelancer receives twice more money per hour than a junior freelancer (the description of each task includes the cost per hour of a junior freelancer).

The payments application should allow the collaborator of each organization to load a file (txt or csv) that registers a set of (historical) transactions. Each line of this file records one transaction and includes information that describes the execution of one task (a brief description of the task, details about the execution of the task and information about the freelancer that worked on the task). An example of such file is available in LAPR2 moodle repository. All the information needed to describe a transaction is included in that file and there will be no missing values. The collaborator can load an unlimited number of such files into the payments application.

The manager of the organization can use the application to define when (the day of the month and the time of day) all the tasks/transactions, that exist in the system and that have not been paid yet, will be paid. The payments will be made automatically by the system whenever we arrive to the time scheduled by the manager of the organization. The payment will be made by bank transfer to the freelancer account and the freelancer will also receive one e-mail with a receipt describing the amount to pay for each task and the overall payment value. The payment values must be presented both in euros and using the currency of the freelancer country. To convert between currencies the application should use an API developed by third party companies. The application should be prepared to use any API to convert between currencies and the API that will be used should be defined at the time of installing the application. Moreover, the application should register all payments, made by all organizations, in a single txt file.

At any time, both the manager and the collaborator of the organization can use the application to see overall statistics about task execution times (the mean and the standard deviation of the delays, of each one and of all freelancers, and a histogram to analyze the delays of each one and all the freelancers working to the organization) and freelancer payments (the mean and the standard deviation of the payments made to each one of the freelancers). The application should allow to sort the freelancers by name or by payment value.

The T4J administrator can add new organizations to the system. When, adding a new organization the administrator should also specify one manager of the organization and one collaborator of the organization.
At any time, the T4J administrator should be able to see statistics describing the performance of the freelancers (the mean and the standard deviation of the delays and payments, of each one and of all freelancers, and histograms to analyze the delays and payments of each one and all the freelancers that exist in the system). Regarding the execution delay time of the freelancers, and considering that the task delay is normally distributed with a mean of 2 hours and a standard deviation of 1.5 hours, the application should compute and show, to the T4J administrator, the

probability that the sample mean is higher than 3 hours. Moreover, in the last day of every year the system should automatically send an e-mail to all the freelancers who have a mean task delay time (during the current year) that is higher than 3 hours and have a percentage of delays (during the current year) that is higher than the overall percentage of delays.

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.

The histograms presented in this application should have three levels/intervals: $]-\infty, \mu-\sigma]$, $]\mu-\sigma, \mu+\sigma[$ and $[\mu+\sigma, +\infty[$. Here μ represents the sample mean and σ is the standard deviation.

The currency values of each transaction are specified in euros.

The application should use object serialization to ensure persistence of the data between two runs of the application.

At this point the client does not know which email service (and API) it will use to send e-mails. Therefore, the system should be prepared to use any e-mail service/API and, for now, instead of sending e-mails, all the e-mail messages should be written to a single file (named *e-mail.txt*).

The application 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.
In this project all members of the team should use Bitbucket and 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 13, 11.55 p.m.), students must submit on LAPR2's moodle a final delivery that should contain the following zip files:

- The project repository (all folders in the repository), in a single ZIP file named LAPR2-YYYY-GXXX-Application.zip;
- The Project Report, named LAPR2-YYYY-GXXX-Report.pdf;
- PowerPoint presentation, named LAPR2-YYYY-GXXX-Presentation.pptx.

4. Assessment

Assessment is performed everyday while students are in class and receive immediate feedback and through a final project assessment. Therefore, class attendance (in Microsoft Teams) 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 of 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
    - o https://moodle.isep.ipp.pt/course/view.php?id=8317

- Bitbucket
    - o https://bitbucket.org/isep-dei-2020-lapr2/lapr2-2020-g0XX, where XX represents the number of each team.

6. Revision History

Date	Description	Type
25/5/2020	**Invoice -> Receipt; "…**should allow the **Manager…" -> "…**should allow the **Collaborator…"**	