THE INTERNATIONAL INSTITUTE OF
SUPINFO
INFORMATION TECHNOLOGY

Java Standard Edition

Academic Year – 2013 - 2014

SupPlanner

Mini-Proiect

Context

You have been hired to create new planner software for a company called Acme. Thanks to your love for Java, you naturally choose to develop this with it!

Our application is simple: it will allow managers to create projects for employees. After log in, employees can create tasks about what they're doing. Everybody will have a dashboard displaying the entire application status.

Specifications

To represent SupPlanner, you need a client application with these functionalities:

- Register, Login, Log out
- Dashboard with all projects
- Project dashboard
- For managers:
    - Create a project and modify it
    - Invite others (employees and managers) into the project
- For employees:
    - Create a task and modify it
    - Mark a task as completed

And also a server application developed in Java with sockets and a MySQL database.

1. Data Structure

Before starting the project, draw an UML class diagram representing the JPA Entities you will need with their relationships.

This diagram will be useful for you and for the team that will develop the next version of the platform.

The class diagram must be returned in ***jpeg***, ***png*** *or* ***pdf*** format (otherwise your STA will hate you!).

2. SupPlanner application

a. Register, Log in and out

Your application must display a simple form at launch, to register or to log in.
For registration, an email address "@supinfo.com" must be entered. A password (entered twice) must be at least eight characters long with an uppercase character and a number. A role must also be set: "Manager" or "Employee". As Acme is a small company, no need for an administrator to check the role truthfulness.

You are free to create other fields if you want to.

b. Dashboard with all projects

After logging in, SupPlanner must display a dashboard with all projects involving the specified user. Clicking on one of them must display the relative project's dashboard. As every dashboard, this one must be easy to use and group all useful information, such as (but not limited to):

- Projects' names
- Projects' completion, as a percentage
- Number of employees working on it
- Scheduled dates (Begin and End)
- Edit link for a manager

c. Project dashboard

As the previous one, this dashboard must be easy to use. Here, all tasks will be displayed for each employee, as a Gantt chart. A task is (at least) represented by:

- A name
- A start date/hour and an end date/hour
- An author (the current user)
- A description
- A status (completed or not)

All current tasks must be displayed here. As tasks are meant to be day-based, you need to display views week by week. For this version, a week is seven-days based and neither takes weekends nor holidays into account.

Also, information about the project must be displayed, as a completion percentage of the project.

d. Create a project and modify it

Managers need a way to create projects and modify them. Projects have (at least) these properties:

- Name
- Start date and end date
- Employees working on it (thanks to an “Invite” feature explained later)

Find a way to include this feature so that it can be easy to use.

e. Invite others into project

As described before, a form must allow managers to invite employees and managers into a specific project. No need for confirmation at each end, the selected user is directly included into the project.

f. Create a task and modify it

Employees will work on tasks, they need to create and modify them. Tasks have (at least) these properties:

- Name
- Start date and end date
- Employee working on it (current user)

Creating a task must refresh others users’ dashboard in a fine way.

g. Mark a task as completed

Mark a task as completed will in one click put the end date at the current day.

h. Design

Use Swing to represent these functionalities over a minimalist GUI. Use your functions in both CLI and GUI. Follow **DRY** principle: **D**on't **R**epeat **Y**ourself!

3. SupPlanner server

The server must handle clients' connections and store data received in database when necessary. You are completely free about how it should work, as long as you follow these rules:

- The source code for server is put in a separate project
- For each request, the server must notice clients about the request's state: success or error. In case of error, some details must be provided to display errors on client side

Notation

Functionalities	Points
PROJECT	**25**
Data Structure	**2**
Register, Log in and out	**2**
Dashboard with all projects	**3**
Project dashboard	**3**
Create/Edit project	**2**
Invite others into project	**2**
Create/Edit task	**2**
Mark task as completed	**1**
Design	**4**
Server side	**2**
Code Quality & Conventions	**2**
ORAL DEFENSE	**15**
Project presentation	5
Technical questions	10
TOTAL	**40**

Appendix

JavaDoc: http://download.oracle.com/javase/6/docs/api/

For code quality, just remember this quote:

> *"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live"* - ***Martin Golding***

Return

You will return your graded exercise inside a ZIP archive named: **SupPlanner_Campus_IdBooster.zip**.
For example: SupPlanner_Lille_10000.zip
Not respecting this convention will be sanctioned by penalties points.
You will send the archive **to your STA SUPINFO email address only** and **before the May 14th at 11:59 PM**. After that delay, your project **will not be corrected and the mark 0 will be assigned to you**.