Data Structure Design

PR2: Practice 2

Statement

Below we present the statement of the second part of the practice (PR2) of the course. The practice consists of an evolutionary programming exercise based on the first practical exercise (PR1). It is **MANDATORY** that the coding of this exercise is based on the data structures defined in the official solution of CAA1 and CAA2; and use the PR1 coding (official solution) as a starting point. **DO NOT use your solution**. If any additional structure is necessary for any of the functionalities, it can be added justifying this decision.

We remind you that to pass the subject it is necessary to obtain a minimum grade of 5 for practices (PR0, PR1 and PR2). On the other hand, **this practice must be resolved individually** and any indication of copying will be notified to those responsible for the studies so that they can take the appropriate measures.

Description of the PR2 to be carried out

In PR2 we will basically work with the following entities and concepts:

- **Worker**: The workers that exist in the system
- **Company:** Existing companies
- **JobOffer**: Existing job offers
- **Request:** Job application
- **Enrollment:** Registration for a job offer
- **Rating:** Evaluation of a job offer
- **Equipment:** Material that the platform manages and that can be assigned to the rooms
- **Room:** Rooms managed by the platform
- **Employee:** Existing employees

Features

The functionalities required for PR2 are:

1. addRole(roleId, description)
2. addEmployee(dni, name, surname, birthday, roleId)
3. addRoom(roomId, name, description, type)
4. assignEmployee(id, roomId)
5. getEmployeesByRoom(roomId): Iterator
6. getEmployeesByRole(roleId) : Iterator
7. addEquipment(equipmentId, name, description)
8. assign Equipment(equipmentId, roomId)
9. getLevel(workerI d): Level
10. getWorkersByJobOffer(jobOfferId): Iterator
11. getSubstitutesByJobOffer(jobOfferId): Iterator
12. getRoomsWithoutEmployees(): Iterator
13. best5EquippedRooms(): Iterator
14. addFollower(employeeDni, employeeFollowerDni)
15. getFollowers(employeeDni): Iterator
16. getFollowings(employeeDni): Iterator
17. recommendations(employeeDni): Iterator
18. getUnfollowedColleagues(employeeDni): Iterator

Additionally, new operations have been defined that allow inspecting the data structure and validating test sets:

- numRoles(): int
- numEmployees(): int
- numEmployeesByRole(String roleId): int
- numRooms(): int
- numEquipments(): int
- numEquipmentsByRoom(String roomId): int
- whereIs(String equipmentId): Room
- getRole(String role): Role
- getEquipment(String equipmentId): Equipment
- …

All operations required in PR2 are defined in the **CTTCompaniesJobsPR2 interface** , which is provided along with this document.

Project launch

As indicated in the Resources section, a base project is provided to carry out the exercise. Specific:

- src/main/java/uoc.ds.pr. **CTTCompaniesJobs / CTTCompaniesJobsPR2.java** : Interface that specifies the TAD operations to implement
- src/test/java/uoc.ds.pr. **FactoryCTTCompaniesJobs.java** : Class that implements the factory design pattern and initializes data structures with initial values.
- **CTTCompaniesJobsPR1Test.java, CTTCompaniesJobsPR2Test.java** : TAD test classes **CTTCompaniesJobs**
- **lib/DSLib-2.1.3.jar** : Version 2.1.3 of the DSLib.
- Classes pending implementation:
  - **CTTCompaniesJobsImpl, CTTCCompaniesJobsPR2Impl** , and the defined model entities
  - Additionally, all exceptions defined in the interface that must inherit from the **DSDException class** provided in the statement must be implemented.

Facility

The main steps to launch the project of this practice are the following:

- Download, unzip the .zip with the PR2 statement and import the project into the corresponding IDE.
- incorporate the solution of PR1 on this statement.
- implementation of the parts pending implementation.
- run the JUnit tests src/test/java

NOTE: Contact the teaching staff of the subject's Laboratory classroom for any questions on this point and others related to the coding of this practice.

Planning

The planning that we propose for **PR2** is the following:

1. (12/21-12/22). Analysis of the solution of CAA1, CAA2, and PR1, and identification of the TADs of the library.

2. (12/22-12/23). Analysis of the test set and validation that the data structures proposed in CAA1 and CAA2 are sufficient to meet the requirements of the test set. If there is any modification, it must be indicated in the delivery text document. With this action, we are following a test-*driven development methodology* (TDD).

3. (12/23-1/19). Implementation of the manager with the operations defined by the provided interface.

   NOTE: For the implementation of tests related to the social network (**CTTCompaniesJobsPR2TestPlus), it is recommended to previously analyze the SocialNetworkWithDirectedGraphTest** unit test from the DSLib library that can be consulted through the GIT repository.

4. (19/1-24/1). Testing, validation of test set, and updating if necessary.

Delivery format

The delivery must be made in a compressed file (ZIP), through the "Delivery and EC registration" space, organized as follows:

- A text document (readme.txt) indicating the scope of delivery, modifications, and updates made to the initially proposed design (official CAA2 solution) with its justification in case any additional data structure has been necessary, problems, and additional comments.
- A presentation of the tests executed. Remember that you do not have to stay only with the test set that we provide you: if you consider it appropriate, you can expand said test set. They must be included directly in the ZIP.
- The project with its sources: code (*.java) maintains the structure of the project (src/main folder, src/test, and package structure. DO NOT ATTACH *.class files or the TADs library of the subject in the ZIP. For practical purposes, you can zip the project without the binaries.
- IT IS PROHIBITED TO MODIFY the code of the test files (src/test) or the contracts defined in the interfaces.