Data Structure Design

PR3: Practice 3

Statement

Below, we present the statement for the second part of the practice (PR3) of the course. The practice consists of an evolutionary programming exercise based on the first practical exercise (PR2). 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 uses the PR2 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 to justify this decision.

We remind you that to pass the subject, it is necessary to obtain a minimum grade of 5 for practices (P = 10% PR1 + 35% PR2 + 55% PR3). 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.

AI not allowed

In this activity, the use of artificial intelligence tools is not allowed. In the teaching plan and on the UOC's website on academic integrity and plagiarism, you will find information about what is considered irregular behavior in assessment and the consequences it may entail.

Description of the PR3 to be carried out

In PR3 we will basically work with the following entities and concepts:

- **Reader:** Reader
- **Loan:** Loan of a book
- **Book:** Books stored in the library
- **CatalogedBook:** Books cataloged in the library
- **Workers:** Library workers
- **Copies:** Copies of books
- **Rating:** Ratings of a book
- **Theme:** Theme of a book
- **Level:** Reader level
- **Author:** Author of a book

Features

The functionalities required for PR3 are:

1. addTheme(themeId, name)
2. addAuthor(uniqueCode, name, surname)
3. getCopies(bookId): Iterator
4. createRequest(readerId, copyId, date)
5. getBooksByTheme(themeId): Iterator
6. getBooksByAuthor(uniqueCode): Iterator
7. getReaderLevel(readerId): Level
8. addReview(bookId, readerId, rate, comment)
9. getDestroyedCopies(): Iterator
10. getReviewsByBook(bookId): Iterator
11. best5Books(): Iterator
12. best5Readers(): Iterator
13. getRecommendationsByBook(bookId): Iterator
14. getRecommendationsByReader(readerId): Iterator
15. getRecommendedAuthors(uniqueCode): Iterator

Additionally, some operations have been defined that allow inspecting the data structure and validating test sets: **numXXXX, getXXXX**

All operations required in PR3 are defined in the **LibraryPR3** interface, 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. **Library / LibraryPR3.java**: Interface that specifies the ADT operations to implement.
- src/test/java/uoc.ds.pr. **FactoryLibrary.java**: Class that implements the factory design pattern and initializes data structures with initial values.
- **LibraryPR2Test.java, LibraryPR3Test.java** : ADT test classes **Library**
- **lib/DSLib-2.1.6.jar**: Version 2.1.6 of the DSLib.
- Classes pending implementation:
    - **LibraryImpl, LibraryPR3Impl**, 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.

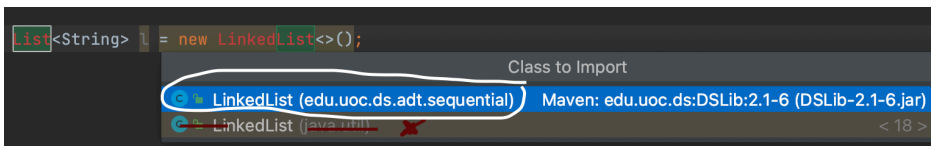
Facility

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

- Download, unzip the .zip with the PR3 statement and import the project into the corresponding IDE.
- Incorporate the solution of PR2 on this statement.
- Implement the parts that are pending.
- Run the JUnit tests src/test/java

NOTE:

- Be careful when importing certain classes whose names also exist in Java collections, such as `List`, `LinkedList`, `Queue`, `Stack`, etc. As shown in the image, always check that these classes are being imported from the course library and not from the `java.util` package.



Planning

The planning that we propose for **PR3** is the following:

1. (12/19 - 12/21). Analysis of the solution of CAA1, CAA2, and PR2, and identification of the ADTs of the library.

2. (12/21-12/24). 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. (12/25-01/19). Implementation of the manager with the operations defined by the provided interface.

   NOTE: For the implementation of tests related to the recommendations system (**LibraryPR3TestPlus), it is recommended to previously analyze the SocialNetworkWithDirectedGraphTest** unit test from the DSLib library that can be consulted through the GIT repository.

4. (01/19-01/22). 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 ADTs 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.