Juan Fernando Martinez Hidalgo – A00358232

**Laboratory 2 Report – Turns control**

GitHub Repository link: <https://github.com/JuanF2019/turns>

**Functional requirements:**

1. Manage different turn types with a name and a predefined duration, both characteristics should be given by the user, at a beginning there are no types.

2. Create and assign turns, whenever a turn is going to be created, it must be initiated with a type, if there are no types or the type selected is invalid the program will show an error message. Created turns must be associated with a user, the program will ask for user identification number.

3. Manage and show the current time of the turns system. When it is updated all turns that can be attended in that time are attended. If there are no turns the system will notify the user and update the date with no changes.

4. Automatic advance turns. The user will give how many minutes the program will advance its date and attend all possible turns in that interval considering the type duration and a 15seconds waiting time between each turn.

5. Generate a report with all the turns that a person has had since the beginning of time. User would select if the report will be saved to a file or just printed in the console.

6. Generate a report with all the users that have been assigned a given turn. User would select if the report will be saved to a file or just printed in the console.

7. Add a user given its name, surname, document type, document number, phone number and address, the user will give each field. The phone number and address are not mandatory all other data will be checked and if it is not present it would not add the user.

8. Add random users, generates random users using information from an online name generator. The user gives the number of users to be generated.

9. Generate random turns. Randomly assign new turns to users. The user will give the number of turns to be generated. Random turns are only assigned to existing users.

10. Show operation execution time. Prints the time that any main operation took and prints it in milliseconds.

**Nonfunctional requirements**

1. Persistency is achieved using serializable.

2. Class Date is defined to manage time in the software, is internally using java.time.LocalDateTime class to define LocalDateTime attribute and to create methods to modify the date.

3. Two searching algorithms are applied in the following methods

a. Sequential search, applied in the method findUserByTurn and using comparable interface to compare each turn with user’s assigned turn, if they are equal it returns the user position in the users lists, else returns -1.

b. Binary search used in findUserByDN after bubble sorting the users list to find the location of a given user in that list.

4. The three classic sorting algorithms are applied in the following methods

a. Bubble sort used in findUserByDN to sort array and then use binary search to find the user location using predefined compareTo method from String class.

b. Selection sort in selectionSortBySurname method (Unused).

c. Insertion sort in insertionSortByName method (Unused).

5. Five different comparation techniques are used to sort or search in the following methods.

a. Comparator using anonym class in method generateRandomUsers of class TurnManager to keep track of the user with the highest document number, so if the user generates later more users the document number doesn’t repeat. Users list is sorted, and the maximum is in the last position

b. Comparator using external class in method addUser when giving a comparator to the collections sort and reverseOrder methods.

c. Collection reverse sort used after adding a user.

d. Used a predefined compareTo method in findUserByDN to find the location of a user given its document number.

e. Collections sort used after adding a user.

f. Comparable in method findUserByTurn, it doesn’t sort the turns list, but it is used to find the user with a given turn.

**Class Diagram**

Imagen que contiene texto

Descripción generada automáticamente