Final Project Requirements

This is a team project. You will be working on the project in a team of two people.

Note: if there are students who are willingly refused to work on the project, let the instructors know by **19.12.2021 @18:00**.

DEADLINE to submit FINAL PROJECT (to Blackboard): January 13th @23:59 (SHARP! No extensions)

The project is about Passenger Plane Crashes dataset. You are going to develop a UI (console based) which will help the user to work on the dataset. The main functionalities the program needs to provide are the following:

1. **List** all the entities **(20 points)**
    a. List **all the fields** of each entity
    b. List only the **selected fields** of each entity
    c. List entities based on the range of rows (e.g., range is given, 5 100)
    d. **Note**: all lists should be followed by the number of entities listed.
2. **Sort** the entities **(30 points)**
    a. Based on **any field**
    b. In **any order** (i.e., ASC, DESC)
3. **Search** entity(ies) based on any given field and value **(15 points)**
    a. **string fields** and values must be checked based on **contains** not exact equality.
    b. **non-string fields** and values must be checked based on **exact equality**.
4. **List column names (5 points)**
5. **Filter** entities **(30 points)**
    a. Based on any given field or set of fields and according to some rules:
    b. string fields
        i. starts with
        ii. ends with
        iii. contains
        iv. null
    c. date, time, and numeric fields
        i. equal (eq)
        ii. greater than (gt)
        iii. less than (lt)
        iv. greater and equal to (ge)
        v. less and equal to (le)
        vi. between (bt)
        vii. null
    d. date and time
        i. in a specific year (y)

   ii. in a specific month (m)
   iii. in a specific day (d)
e. [IF you take clustId as a Boolean value (i.e., is high fatality]
   i. Equal (eq) [true or false]
f. Examples:
   i. select all the crashes happened in 1962 which has death rate (opposite of survivalRate) greater than 50%.
   ii. select all the crashes happened any time in 20:00 - 00:00 by operator Aeroflot
   iii. select all the crashes happened in Cameroon
   iv. select all the crashes which caused more than 20 dead on the ground

Several important notes:

1. To make is simple to use, make sure you take inputs of both fields and values in the same format as you list them.
   a. E.g., if you print date, say, in the format **1994-11-22** expect the user to enter the same format in search, filter, etc.
2. Menu should be interactive, as the user might want to perform some sequence of requests.
   a. E.g., user might want to sort the entities based on the date (**request 1**) and pick top 10 entities to list (**request 2**).
3. To increase the reusability, after all the requests or sub-requests, return a collection of results so that you can print them, or use it in some subsequent requests.
4. You are free to use any notes or internet resources while developing the project, but **not misuse** them.
5. If ANY kind of cheating is detected, then **both sides (Cheater and Provider) will end up getting 0 pts**! The case will then be reported to the honor code committee.

Technical aspects:

1. You may use any topics we covered during the semester to maximum extend.
2. What might come handy are: collections, functional interface and stream api, exception handling, file reading, OOP principles (at least Encapsulation and method delegation), LocalDate and LocalTime classes from java.time package, enums, etc.
3. Make sure you are not printing any stack trace of any exception, instead show the user a nice error message formatted in a nice way.
4. Make sure you use methods wisely; each method should be responsible as specific as possible task.

Prosses of development:

1. You are expected to team up and discuss the project.
2. Share responsibilities.
3. Create a public github repository and put a nice description of your project in the README.md file.
4. Both team members will be committing to the same remote repository every time they have some feature ready and TESTED.
    a. Before committing, make sure you test the currently developed functionality and whether it suits to the entire application.
5. Once finished, write a report about the process, and submit it to the Blackboard Assignment Grade as well as the compressed (.zip) file of the project.
    a. The name of files:
        i. {TeamNo}_PROJECT
        ii. {TeamNo}_REPORT
    b. **Note**: Only source files and data files (if there are any) are to be included in the project file to submit. DO NOT SUBMIT the entire project folder if you are using any IDE.

Final Report:

1. A brief report about the responsibilities of each team member and their percentage (e.g., 50-50 or 40-60)
2. User manual, i.e., how the system must be used by the end user.
    a. Guidelines and snapshots would be enough per each request type.
3. Link to the public github repository
    a. Also add the link to the profiles of the team members.

BONUS:

1. There will be times where you need to get values of fields on a given entity. It is possible to do it in old fashioned way (the way we have learned). You may have if…else or switch…case to decide the field and call the getter method.
    a. There is better and more professional approach, using **Reflection API**.
    b. Note: *If it is possible to perform an operation without using reflection, then it is preferable to avoid using it.*
    c. Using the Reflection API (package java.lang.reflect) will not only result in shorter and better codes but also + **20 points**.
2. After search and filter requests we always print the results on the console.
    a. Giving an extra option to the user to EXPORT the result in the form of a report to a .csv file AFTER LISTING IT + **15 points**.
    b. **Note**: make sure you have proper headers (the names of the columns) and proper names for the files (hint: you may use the request info as the name of the file, since multiple reports should not overwrite)