Solving complex problems: search, sort and other algorithms

Objectives

Development of classes in Python

- Develop an application based on the layered architecture
- Implement exceptions, document and test the code
- Learn how to develop algorithms for search and sort

Deadline

During **lab 11**: present class from domain layer

During **lab 12**: present general search and sort functions

Beginning of **lab 13**: upload the whole solution

Requirements

The application should be developed along 3 consecutive iterations as follows:

1st. Iteration
   a. Implementation
      i. Domain and repository classes with CRUD operations
      ii. test
   b. Use modular programming
   c. Give at least 3 data examples in the application (to facilitate testing)
   d. Each function should be documented and tested (at least 3 assertions/function)

2nd. Iteration
   a. Implementation
      i. sort the content of the repository
      ii. filter the content of the repository
   b. A **single** sort/search should be used for all requirements.
   c. Give at least 3 data examples in the application (to facilitate testing)
   d. Each function should be documented and tested (at least 3 assertions/function)

3rd. Iteration
   a. Implementation: group generation
   b. A single backtracking algorithm should be used
   c. Give at least 3 data examples in the application (to facilitate testing)

d. Each function should be documented and tested (at least 3 assertions/function)

Use your registration number ($\boldsymbol{n_{reg}}$) to define the number of the exercise you have to solve: $\boldsymbol{n_{reg} \bmod 2 + 1}$

e.g. my registration number is 1491

$1491\ mod\ 2 + 1 = 1 + 1 = 2$

⇒ I have to solve exercises: P**2**

Problem specification

P1. Planes and Passengers

Considering an airport, there are several planes (identified by name/number, airline company, number of seats, destination, list of passengers) and each plane has certain passengers (identified by first name, last name and passport number).

1. Develop an application to allow CRUD operations on $\boldsymbol{Passenger}$ and $\boldsymbol{Plane}$.
2. The application should be layered, tested and validated.

---

3. Sort the passengers in a plane by last name
4. Sort planes according to the number of passengers
5. Sort planes according to the number of passengers with the first name starting with a given substring
6. Sort planes according to the string obtained by concatenation of the number of passengers in the plane and the destination
7. Identify planes that have passengers with passport numbers starting with the same 3 letters
8. Identify passengers from a given plane for which the first name or last name contain a string given as parameter
9. Identify plane/planes where there is a passenger with given name

---

10. Form groups of $\boldsymbol{k}$ passengers from the same plane but with different last names ($\boldsymbol{k}$ is a value given by the user)
11. Form groups of $\boldsymbol{k}$ planes with the same destination but belonging to different airline companies ($\boldsymbol{k}$ is a value given by the user)

P2. Hospital Departments and Patients

In a hospital, there are several departments (identified by id, name, number of beds and list of patients) and each department has certain patients (identified by first name, last name, personal numerical code and disease).

1. Develop an application to allow CRUD operations on departments and patients.

2. The application should be layered, tested and validated.

---

3. Sort the patients in a department by personal numerical code
4. Sort departments by the number of patients
5. Sort departments by the number of patients having the age above a given limit (for example, above 50 years old)
6. Sort departments by the number of patients and the patients in a department alphabetically
7. Identify departments where there are patients under a given age (for example, below 30)
8. Identify patients from a given department for which the first name or last name contain a given string
9. Identify department/departments where there are patients with a given first name

---

10. Form groups of $\boldsymbol{k}$ patients from the same department and the same disease ($\boldsymbol{k}$ is a value given by the user)
11. Form groups of $\boldsymbol{k}$ departments having at most $\boldsymbol{p}$ patients suffering from the same disease ($\boldsymbol{k}$ and $\boldsymbol{p}$ are values given by the user)

Submission

Total points: **10**

You need to submit an **archive** (e.g. .zip, .rar, etc) with the source code (**only** your own *.py* files created, without venv or other generated files) to the assignment on **Teams** before the deadline. Please use the following convention to name the archive file:

$sfmie1234\_A5.zip$, where s – first letter of your surname
f – first letter of your first name
mie – stand for mathematics informatics in English
1234 – is your registration number
$A5$ – number of the assignment

If something is not clear, please ask me.

Key

1p Default
1p Work during lab 11
1p Work during lab 12
1p Repository classes
1p Sort algorithms
1p Backtracking

1p Correct layered architecture
1p At least 10 data examples for each iteration
1p Unittests for each function (each containing at least 3 assertion statements)
1p Documentation