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

Deadlines

- ***Lab 11****:* Iteration 1 – should include all general CRUD operations and the specific operations specified during the lab *(work during the same lab)* without exceptions and user interface
- ***Lab 12:*** Exceptions, data validation and tests for Iteration 1 *(homework from Lab11)* and sort and search operations from Iteration 2 *(work during the same lab)*
- ***Lab 13:*** Iteration 3 *(homework from Lab12)* + any new functions specified during the lab *(work during the same lab)*

Problem 1: 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).

Iteration 1 (Lab 11):

- Develop an application to allow CRUD operations on *passengers* and *planes*.
- The application should be layered, tested and validated.

Iteration 2 (Lab 12):

Extend the application to include the following features:

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

A single sort algorithm should be used for all sort requirements.

A single search algorithm should be used for all search requirements.

Iteration 3 (Lab 13):

Extend the application to include the following features:

- Form groups of *k* passengers from the same plane but with different last names (*k* is a value given by the user)
- Form groups of *k* planes with the same destination but belonging to different airline companies (*k* is a value given by the user)

A single backtracking algorithm should be used.

Problem 2: 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).

Iteration 1 (Lab 11):

- Develop an application to allow CRUD operations on *departments* and *patients*.
- The application should be layered, tested and validated.

Iteration 2 (Lab 12):

Extend the application to include the following features:

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

A single sort algorithm should be used for all sort requirements.

A single search algorithm should be used for all search requirements.

Iteration 3 (Lab 13):

Extend the application to include the following features:

- Form groups of *k* patients from the same department and the same disease (*k* is a value given by the user)
- Form groups of *k* departments having at most *p* patients suffering from the same disease (*k* and *p* are values given by the user)

A single backtracking algorithm should be used.