LABORATORY 05-07

REQUIREMENTS

- Use simple feature-driven software development process
- Iterations will be scheduled for three successive labs.
- Use layered architecture (UI, Controller, Domain, Repository)
- Data validation - when the user gives invalid inputs values, they will be notified about the invalid command (using ***exceptions***)
- The documentation will contain the problem statement, feature list, iteration plan, usage scenario, work items/tasks. You can reuse the same documentation file, but you must update it accordingly.

NB!

- Use object-oriented programming!
- Solutions without tests or specifications will not be graded!
- The documentation must include module documentation generated via **pydoc** (or a similar tool) for the Controller, Repository and Domain modules.
- ***Deadline*** – you have one week to complete each iteration. The grade deduction is then ***1p/week*** for each remaining iteration. Deduction for delays will be of maximum 5p.

PROPOSED PROBLEM STATEMENTS

1. STUDENTS REGISTERS MANAGEMENT

A faculty stores information about:

- Students: *<studentID>, <name>.*
- Grades: *<discipline>, <studentID>, <teacher>, <grade>*.

Create an application which allows to:

- Manage a list of students and a list of disciplines.
- Add, remove, update, list students and disciplines.
- Search for a student based on his/her ID and search for a discipline based on the title.
- Create statistics: list of students and grades at a single discipline ordered: alphabetically, by their grades, the first 20% of students according to the average grades at all the disciplines.
- Unlimited undo/redo functionality. Each step will undo/redo the previous operation that modified the data structure.

NB! *Provide detailed tests as well as specification for each non-UI function. Generate additional documentation for non-UI modules using* ***pydoc****.*

2. STUDENT LAB ASSIGNMENTS

Write an application that manages lab assignments for students at a given discipline. The application will store:

- Students: *<studentID>, <name>, <group>.*
- Assignment: *<studentID>, <description>, <deadline>, <grade>.*

Create an application that allows to:

- Manage a list of students and a list of assignments.
- Add, remove, update, list students and assignments
- Search for a student based on his/her ID.
- Create statistics: list of students and grades at a single assignment ordered: alphabetically, by their grades as well as all students with the average grade lower than 5.
- Unlimited undo/redo functionality. Each step will undo/redo the previous operation that modified the data structure.

NB! *Provide detailed tests as well as specification for each non-UI function. Generate additional documentation for non-UI modules using* ***pydoc****.*

3. MOVIE RENTAL

Write an application for movie rental. The application will store:

- Movies: *<id>, <title>, <description>, <type>*.
- Clients: *<clientId>, <name>, <CNP>*.

Create an application which allows the user to:

- Manage the list of movies and clients.
- Add, remove, update and list movies and clients
- Search for a movie; search for a client.
- Rent/return movies.
- Reporting. The reporting part of the application will allow generating a list of clients or movies ordered based on the user preference. Examples: most rented movies, most active clients, clients with rented movies ordered alphabetically, by number of rents, by year of apparition.
- Unlimited undo/redo functionality. Each step will undo/redo the previous operation that modified the data structure.

NB! *Provide detailed tests as well as specification for each non-UI function. Generate additional documentation for non-UI modules using* ***pydoc****.*

4. LIBRARY

Write an application for a book library. The application will store:

- Books: *<id>, <title>, <description>, <author>*.
- Clients: *<clientId>, <name>, <CNP>*.

Create an application which allows the user to:

- Manage the list of books and clients.
- Add, remove, update and list books and clients.
- Search for a book; search for a client.
- Rent/return book.
- Reporting. The reporting part of the application will allow generating a list of clients or books (maybe of certain type) ordered based on the user preference. Examples: most rented book, most active clients, clients with rented books ordered alphabetically, by number of rents, by year of publishing, etc.
- Unlimited undo/redo functionality. Each step will undo/redo the previous operation that modified the data structure.

NB! *Provide detailed tests as well as specification for each non-UI function. Generate additional documentation for non-UI modules using* ***pydoc****.*

5. NAME AND ADDRESS BOOK (NAB) MANAGEMENT

In a NAB the following information may be stored:

- Persons: *<personID>, <name>, <phone number>, <address>*
- Activities: *<personID>, <date>, <time>, <description>*

Create an application which allows the user to:

- Manage the list of persons and activities.
- Add, remove, update and list persons and activities
- Search for a person; search for an activity.
- Searching a person based on his/her name, and search for an activity on a given date.
- Create statistics: list of activities for a person ordered: alphabetically, by their date; list of persons having activities in a certain interval [date1, date2], ordered according to different criteria (date1, alphabetic order of the description).
- Unlimited undo/redo functionality. Each step will undo/redo the previous operation that modified the data structure.

NB! *Provide detailed tests as well as specification for each non-UI function. Generate additional documentation for non-UI modules using* ***pydoc****.*