UNIVERSITATEA TEHNICĂ
A MOLDOVEI

FAF.OOP22.1

Laboratory work: #1

Building OOP solutions

2023

HOMEWORK N° 1

Assignment

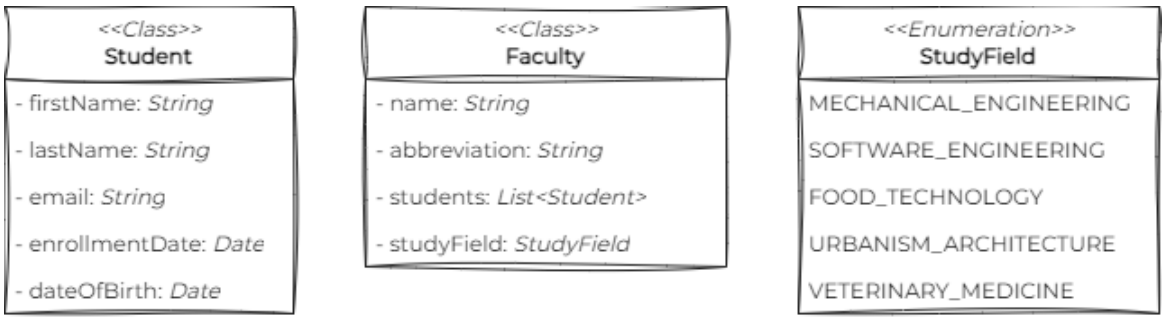
In this laboratory you have to think like an Object-Oriented system designer, by building an OOP solution modelling real world data using Classes.

Task:

Design a Student Management application for Technical University of Moldova.

The OOP class teacher and the *High Council of TUM* requested a simple *Student Management* system from you!

The one they were using is experiencing a major blackout! You have to quickly hack a temporary solution. OOP teacher knew his students are up for the task, and he quickly scribbled some classes directed by the heads of the Software Engineering faculty:



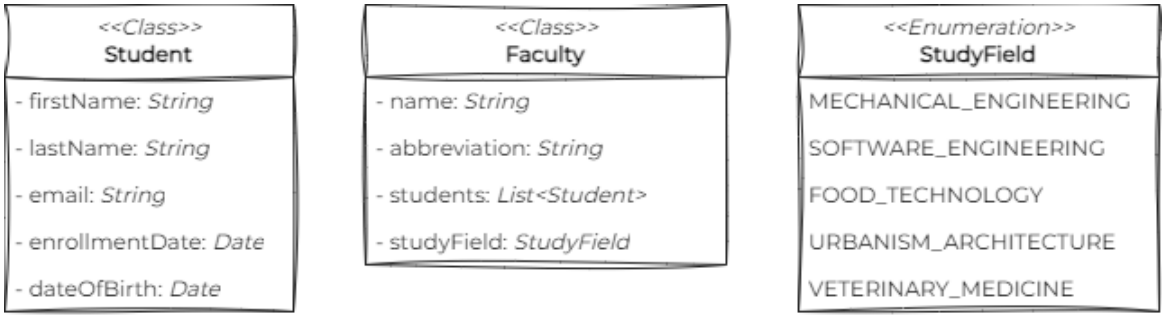
Using just these scribbles you have to design your version of the system. Conveniently enough it will be a good laboratory work for OOP classes to grade.

Fortunately, you have the inside info which tells you that the OOP teacher will grade the solutions according to the below criteria.

Task: LABORATORY RESTRICTIONS!

- **No third party libraries are allowed, you can use only the libraries that are available by default.**
- **You are free to create as many classes as you need to achieve a well structured working system.**
- **No limitation of concepts: You can create interface classes, use concepts like Polymorphism, Inheritance, Abstraction if you deem necessary. It is not a must!**

Base Laboratory (8 Grade)



Task:

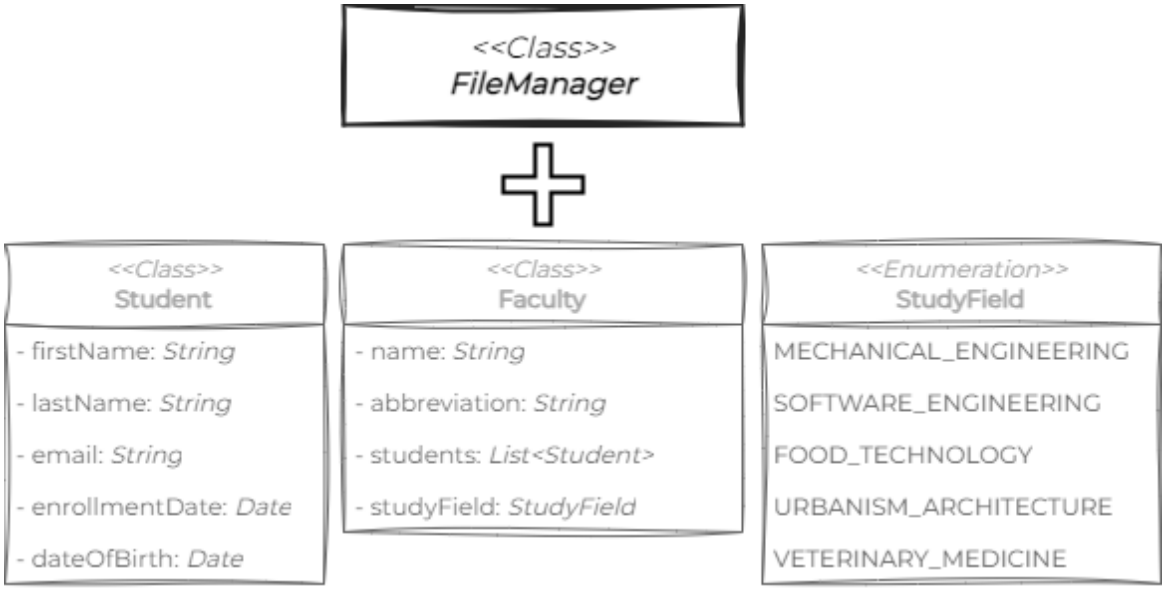
Build a program loop, an interactive command line for the TUM Board to be able to do the next operations:

- **Faculty operations:**
  1. Create and assign a student to a faculty.
  2. Graduate a student from a faculty.
  3. Display current enrolled students (Graduates would be ignored).
  4. Display graduates (Currently enrolled students would be ignored).
  5. Tell or not if a student belongs to this faculty.
- **General operations:**
  1. Create a new faculty.
  2. Search what faculty a student belongs to by a unique identifier (for example by email or a unique ID).
  3. Display University faculties.
  4. Display all faculties belonging to a field. (Ex. FOOD_TECHNOLOGY)

Example of a program loop (you can totally create a better one).[1]

Improved Laboratory (9 Grade)

The program should be able to save it's own state, so when the TUM Board returns to it, all the previous operations are still valid. Faculties and their students added in the previous session should still be there when the program boots up. So maybe you need to create an additional **SaveManager** (or **FileManager**) class to achieve this (you can create as many classes as you deem necessary).



Task:

Design a saving and loading system for the Student Management system.

Working System (10 Grade)

The Student Management system should be reliable, and keep track of the changes that happen in the program. For it to be a working system, you have to log every operation so the TUM Board can monitor every meaningful operation (student creation/graduation, faculty creation) that introduced changes in the system (+ Some extra operations).

Task:

- Design an operation Logging System.
- Add extra operations:
    1. Batch enrollment operation for students via a text file.
    2. Batch graduation operation for students via text file.
    3. Validate inputs with meaningful error statements. (**Can't graduate student: ivan@isa.utm.md (student not present)**, **Operation <operation> is not a valid operation**, **Operation requires extra data** etc.)

Earning Extra Point(s)/Penalties:

*You may earn an extra point or lose if you don't:

- Keep a clean project structure;
- Follow language Coding Conventions;
- Proving SOLID, DRY, KISS principles.

Examples

Examples of the program loop:

```
Welcome to TUM's student management system!
What do you want to do?
g - General operations
f - Faculty operations
s - Student operations

q - Quit Program

your input> g
```

```
your input> g

General operations
What do you want to do?

nf/<faculty name>/<faculty abbreviation>/<field> - create faculty
ss/<student email> - search student and show faculty
df - display faculties
df/<field> - display all faculties of a field

b- Back
q - Quit Program

your input> nf/Media și Telecomunicații/MTC/SOFTWARE_ENGINEERING
```

Figure 1.1: General operations example

```
your input> f

Faculty operations
What do you want to do?

ns/<faculty abbreviation>/<first name>/<last name>/<email>/<day>/<month>/<year> - create student
gs/<email> - (g)raduate (s)tudent
ds/<faculty abbreviation> - (d)isplay enrolled (s)tudents
dg/<faculty abbreviation> - (d)isplay (g)raduated students
bf/<faculty abbreviation>/<email> - check if student (b)elongs to (f)aculty

b - Back
q - Quit Program

your input> /ns/fcim/Ionel/Gavrev/i.gavrev@isa.utm.md/1/4/2002
```

Figure 1.2: Faculty operations example