# Team project (20 %)

In teams of 2.

# Due date : 26 may 2024 Project Requirement

• Completed UML Class Diagrams for all classes and interfaces with their relationship  
• The implementation of all required features (the source code)

• A snapshot of output demonstrating the functionality of the application whether correct or incorrect, complete, or incomplete, included in the README on GitHub.  
• Using console to display and input data  
• The output should be clear and well-presented containing messages to reflect the code testing.

**Project Objectives**

This project incorporates programming techniques with other concepts that you have learned in this class. You will:

1. Apply OOP concepts, write subclasses, and use aggregation.
2. Create a console interface
3. Read and write your class data into a .txt file
4. Test your classes
5. Create documentation for end users

**Project Description**

A tiny college has asked you to be a part of their team because they need a programmer, analyst, and designer to help them in implementing a model of a human resources management system.

In your model, you will have department objects (representing departments). A department contains lists of teachers (either part-time or full-time teachers) and lists of staff; each of which belongs exclusively to one department. A department has a dean, who should be a teacher of that department.

**Implementation Details**

1. Implement an abstract class Person that includes at least five shared fields and contains at least one abstract method of your choice (for example to define a person's category (teacher or staff)).
2. Extend class Person with concrete classes, and override the toString() and equals() methods of each class.
3. The class Teacher has two instance variables: specialty and degree and some other fields for its subclasses (part-time or full-time teachers)
4. The class Staff has also two instance variables: duty and workload
5. Implement an interface PayRoll that contains ComputePayRoll() method to be implemented as follows: a. For each full-time teacher, the salary is computed as (32 \* degreeRate \* 2) \* 0.85, where degreeRate is 112, 82, 42 for PhD, Master, and Bachelor, respectively.  
   b. For each part-time teacher, the salary is computed as (hoursWorked \* degreeRate \* 2) \* 0.76, where degreeRate is 112, 82, 42 for PhD, Master, and Bachelor, respectively.  
   c. For each staff, the salary is computed as (workload \* 32 \* 2) \* 0.75, where the workload is the weekly working hours. The working hours cannot exceed 40.  
   d. Implement the method ComputePayRoll() inside Teacher and Staff classes.
6. Teachers and staff are added/assigned to a department, first by loading/reading from a text file during the first execution of the application. Moreover, new teachers and staff must be added/assigned through the GUI application.
7. Trying to add a teacher or a staff member to an inexistent department (based on department id) should rise/throw an exception.
8. Trying to add a teacher or a staff member that already exists/is added (based on the id) to the department should throw an exception.
9. A department class has a list of teachers and staff. A department class also has a dean, who should be a teacher of that department, otherwise, an exception must be thrown.
10. Each new teacher and staff added to a department should be stored in the text file before exiting the application, generating a new version of the text file. So, if the text file does exist (which is the case), new teachers or staff should be appended. There are files of teachers and staff.
11. You should be able to add Teachers/Staffs/Departments through the console and the info should be save in the corresponding text file. Next time the application is run, it should load the previously entered info.

**Design Requirements**

1. A clear and precise console interface should be designed for input and output
2. Create all the classes
3. Include all the supported class Libraries
4. Document your code using comments

**A Few Notes**

1. The pseudocode that you are asked to do could be in a Word document format, only for the methods that are doing some calculations. You don't need to write it for getters/setters, toString(), equals(), etc.
2. You have to write all the methods inside your UML diagrams excluding getters/setters, toString(), etc.
3. A teacher's specialty is something like "Computer Science" and a teacher's degree is a "Master's".
4. Part-time or full-time is a field in the Teacher's class
5. A Dean is a special type of teacher.