REPORT

- ABSTRACT -

The project we have created is intended to deal with the maintenance of university, student, staff information within the university. This project of Intranet involved the automation of student information that can be implemented in different university managements. Our project deals with the retrieval of information through an Intranet based campus-wide portal. It collects related information about student's academic performance, attendance and other important aspects.

- MAIN INFORMATION -

Our code can be splitted into four main parts:

[1] - Driver

Driver – it is our working environment, the place where we will describe the whole process, more formally there is held authorization, login etc. (sort of entry point for our program, also known as main).

[2] - Storage

Storage – roughly speaking, it is the place where the information about all students, employees, courses, news and other different stuff is stored. Via serialization and deserialization corresponding files can be either updated or saved. (kinda database/warehouse).

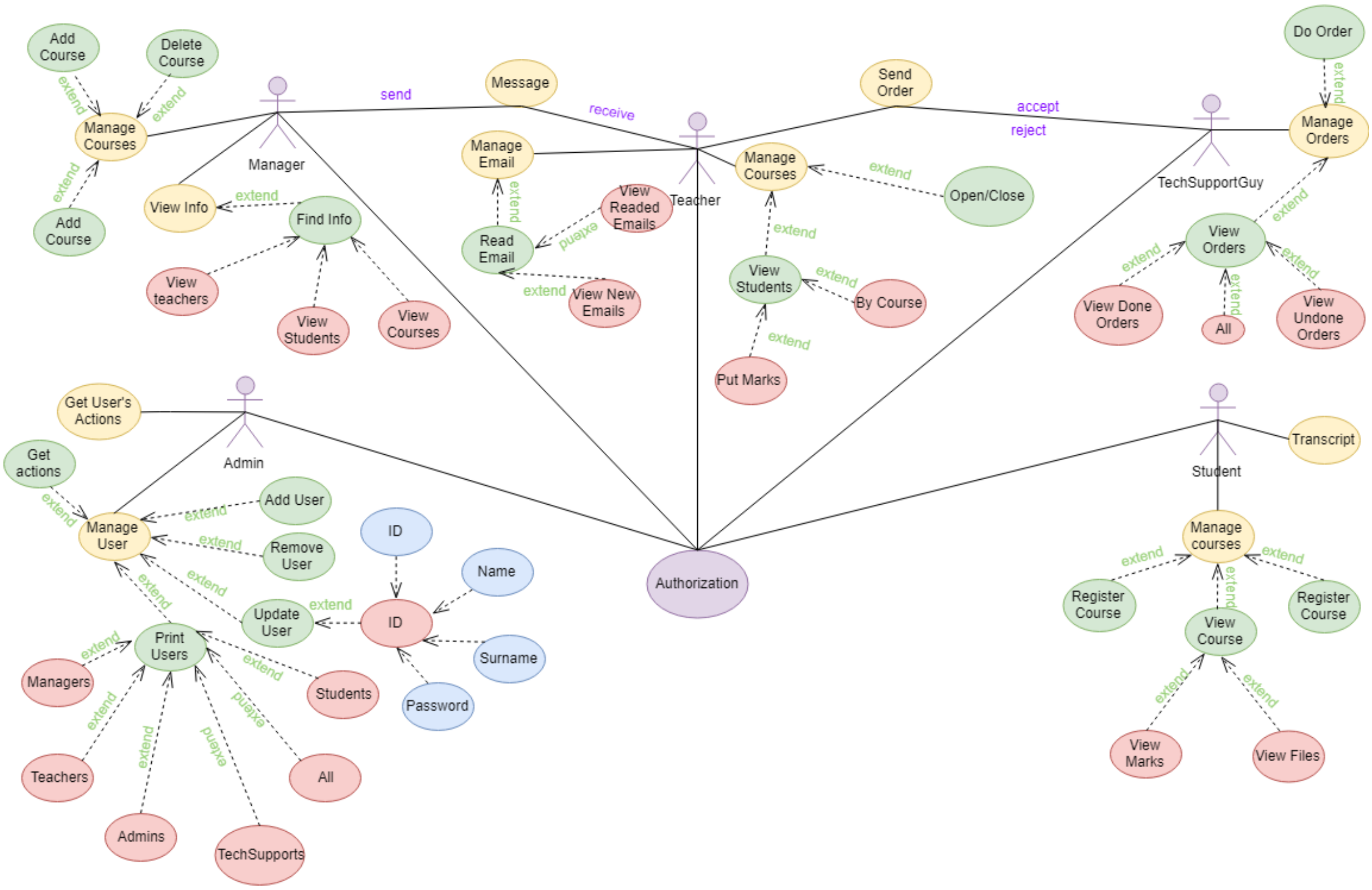
[3] - Users

Users - this is where the interaction among all the users takes place, each class has it's own fields and functions, which affect the state of data that is stored in warehouse. (if you want to see more detailed information, then the use-case diagram is definitely for you).

[4] - Course

Course – describes the state of all existing courses, each subject is considered as separate object with corresponding fields.

CASE DIAGRAM



PICTURE – 1, CASE DIAGRAM

Classes:

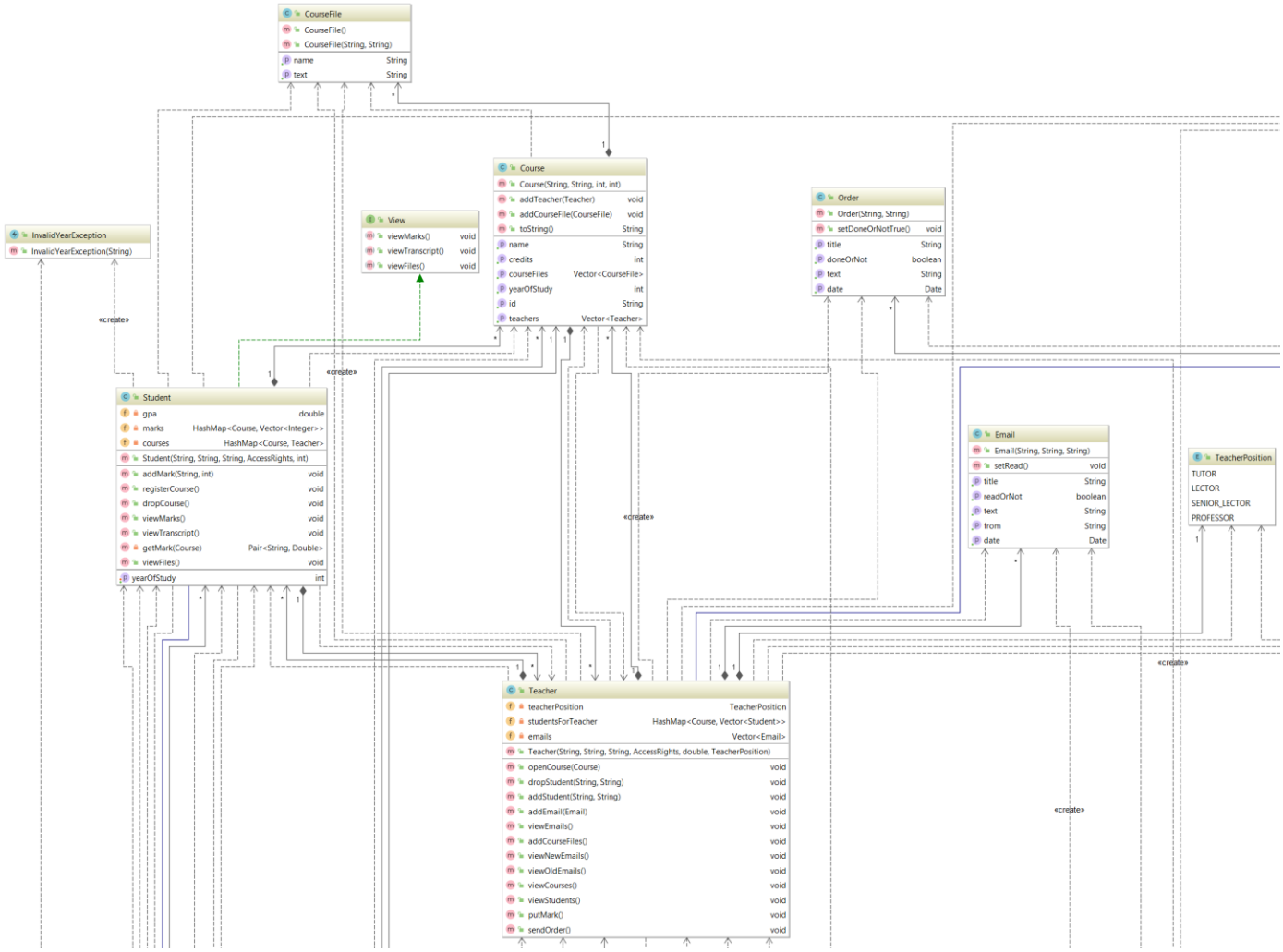
- WorkSpace: entry point of our program
- User: this class is to have variables(fields) for id, name, surname, password, and enum for AccessRIghts, e.g. enum that describes what type of user it is. And, also it has a constructor, accessor and mutator methods for the fields, as well as toString() and equals() methods. Briefly, this class generalizes all the users of our Intranet.
- Student: extends from user, variable to include are year of study, gpa, and collections consisted of chosen courses.
- Employee: abstract class, that extends from user and generalizes the university's staff. And, the only variable to include is salary.
- Teacher, Executor, ORManager: perform corresponding tasks, Teacher has enum for TeacherPosition to represent teacher titles'(tutor, lecturer etc). Regarding ORManager, mainly this guy open/close course for register. And the Executor, he is responsible for processing different orders from teachers.
- Admin: has full access to the intranet system, can create, delete, and update users. (as well as latter 3 classes extends from Employee);
- Order: class that is needed for teacher to send order to a tech support guy, in case of a problem. Variable are: order title, body, date and boolean which tells whether the order is done or not.
- Email: pretty similar to Order class.
- Storage: database of our system. Has static collections for all users.
- Course: simply subject that student can choose to study, fields are: name, id, credits, year of study and some additional collections.
- CourseFile: file that Teacher can upload to a particular course.
- ChangeFunctions: a helper class needed for the optimization of code that enhances readability. Here are some functions necessary for the correct operation of the Admin.
- Driver: plays the role of the application layer, making the connection between our program and the user

For more information, please contact our UML-diagram.

Enums: We have got 2 enums in our project, they are: AccessRights and TeacherPosition. The reason we made them enum is because we've experienced need in a predefined list of values which do not represent some kind of numeric or textual data.

Interfaces: We have only one interface that is called view, we expected that classes such as Student would implement this interface. Furthermore, via this interface different classes are able to provide it's own implementation for view marks, files, transcript methods.

UML DIAGRAM



PICTURE – 2, UML DIAGRAM

CONCLUSION

While we was performing this project we have encountered with some problems regarding class implementation. For example, in Teacher class, there is putMark() method, it is tightly related with Student and Course classes, that's why changes should have been relational. Besides, we also experienced some problems in real life in terms of time-management, communication etc.