My mini-project is a company employee management system.

I set out to make a piece of software that could help in the organisation of employees split into departments.

The CEO of the company will create the Departments and hire, and manage, managers.

Each department will be assigned a manager, once that occurs, the manager of the department can add employees to his department and remove them.

Each Department will have a Name, Manager and a group/array of Workers associated with itself.

Each Employee in the company will have a unique Employee ID associated with them, generated by the program.

Each Employee has a name, address Eircode and date of employment.

The program will generate their salary based on an initial value and the amount of years that the employee has worked with the company.

The manager and worker will be dynamically assigned a position e.g. “Manager of Software Department” and “Worker of Software Department” respectively.

The manager will have a relation with the department itself, meanwhile the worker will only have the name of the department associated with itself.

The program itself is 100% GUI driven. The user has to log in to the system in order to interact with it. Based on the username and password combination the user would have different privileges e.g. CEO vs Manager. In order to increase security, the password of the user is not stored in memory, instead it is encrypted and sent back for comparison. If the user fails to input a valid combination of username and password, an error message is displayed, along with an attempt counter. If the counter reaches 0, the user is locked out of the system with an appropriate message

The CEO can add, remove and list the departments and managers that are in the company.

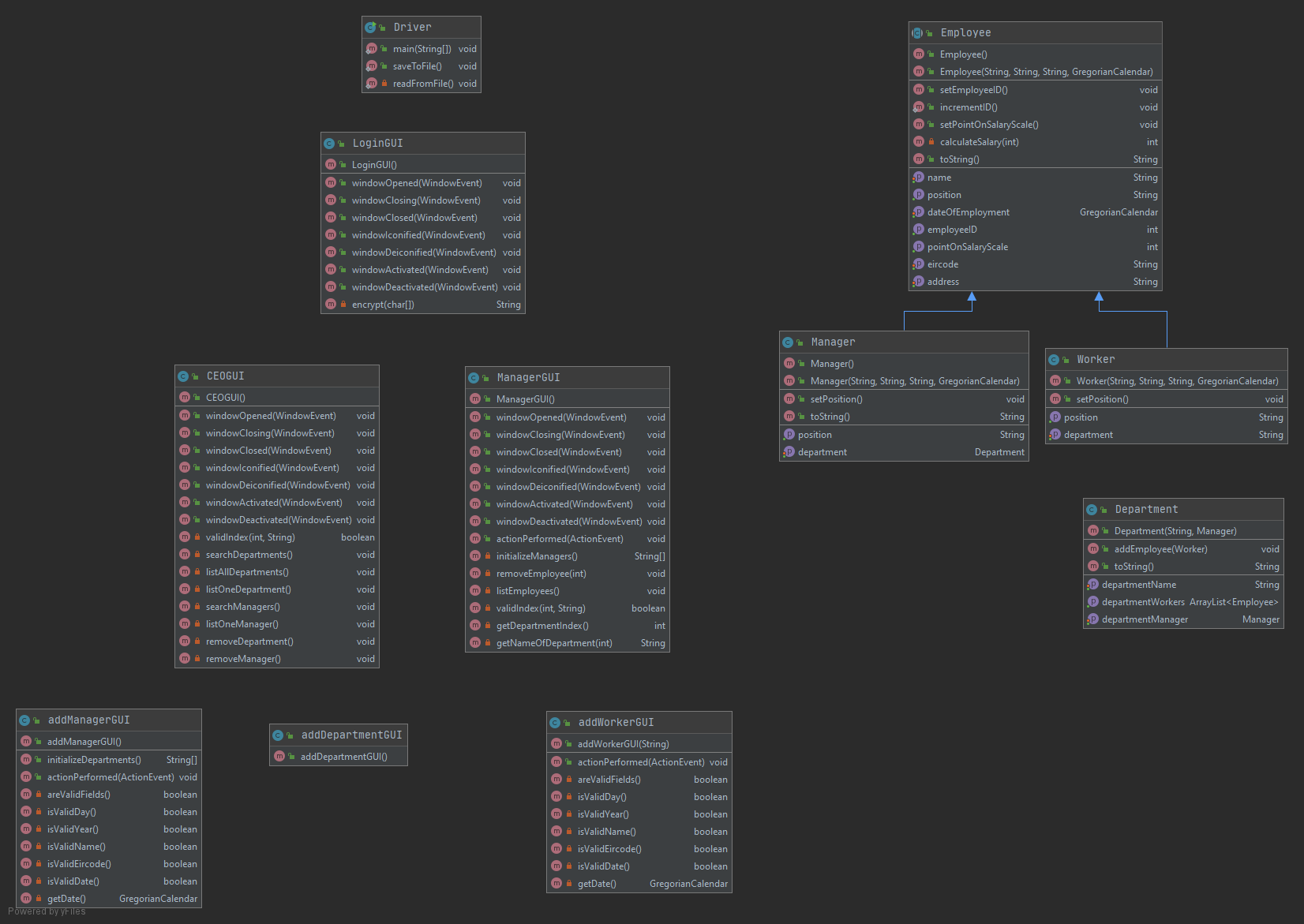
The Manager can add, remove and list the Employees/Workers in his department.

The program has been programmed to incorporate all of the above in a fairly robust way that ensures little to no crashing.

The program has many validation processes that have to be satisfied before the user can perform actions. For example, the program checks to see if the name of the employee is alpha, only letters, the program validates that the address field is not empty, the program validates the Eircode inputted by the user to ensure it is an Eircode that could exist and the program validates the date entered by the user.

The program is also able to persist data in the hard-drive, allowing the data to be stored persistently.

Generated VOPC Diagram



Javadoc comments were used in the department instantiable class.

GitHub Link:

<https://github.com/LewandowskiK/OOPMiniProject>