EOOP 21L PRELIMINARY PROJECT
“HOSPITAL”

by Artur Sebastian Miller, 310698

The project will include a variety of objects, serving as data containers. They will have different functionalities. First class to consider is a ‘hospital’. Every hospital needs to have a basic name identifier, and for that purpose we will have a member string to differentiate between objects. This basic piece of information would be assigned during object construction. Class declaration should also include three object lists: patients, staff and rooms. Data retrieval has to be divided into more methods: to display staff, patients, and rooms list. Next, we need some dedicated methods for checking patients in and out, hiring and dismissing medical staff, and managing workforce allocation across hospital rooms. We also need to be able to search for patients, their doctors, rooms - using dedicated methods. This class acts as a data unifier for the project.

Next object type we will need to achieve the goal will be a ‘room’. A room will serve as a care area for a certain (undefined) number of patients with related conditions. This room will have a name identifier string, assigned at construction. The class could be implemented as a container for dynamic list of pointers to patients from the object list kept in the ‘hospital’ object. Such a room would be cared for by an assigned medical doctor of suitable specialty. To satisfy the requirements for this, ‘room’ needs a ‘staff’ object pointer member, and a method to assign and remove its value. Room class also needs methods for adding into, and removing patients. It also needs a print method, and a method to assign to and free a staff member from room management.

Of course, to start even considering the possibility of having people involved in the design, a base class for derivation of medical workers and patients is required. For that need, I have come up with class ‘person’. This class would only be used as inheritance base for patients and staff - not suitable for standalone usage. It would not include a constructor or a destructor, as these are not inherited. ‘person’ would have a name and a surname strings, set using a member method. For medical purposes, age information is also needed. For this the class would have a member integer, set using appropriately secured method. Additionally we can implement a summary printing method (name, surname, age). Since derived classes will only use name and surname for identification, this combination needs to be unique. Room’s methods should be kept private for access throgh class hospital only.

Next we get two final classes inheriting methods and data from class ‘person’. First of these would be a ‘patient’ class. Apart from having a name, surname and age, this class also has a member string describing the patients illness set and cleared, and also retrieved into standard output using specialised methods. The other class derived from ‘person’ would be a ‘staff member’. Similarly to ‘patient’, the staff member also has basic members and methods inherited. Apart from these, this object would include a staff type describing string, set and retrieved using a pair of specialised methods. This type would indicate the profession of the given individual which would then help in managing hospital employees. Both of these classes would be constructed using their respective construtors that preinitialise basic member strings like name, surname but also age integer with known values. Apart from these, both will have pointers to their respective rooms, and hospitals they are assigned to, and a method to easily list all information about them. ‘Patients’, ‘staff members’, and ‘rooms’ would then be added to one common object ‘hospital’ for unification. Patients, staff members, and rooms would be listed in a hospital on their respective lists for easy information management. Rooms would house a pointer list to patients being healed in a given place under a specific name. To these rooms we would assign a managing medical professional, also with a pointer referencing a staff member from the list. This way we can track patients being healed in the hospital, their conditions, their place in the building, and which doctor takes care of them in their respective room. One caveat is that user of these classes would have to manually track capacity of these seemingly endless rooms, and patient and staff lists, possibly with help of status (object count) printing methods in class ‘hospital’. All of these objects need to be assigned to each other and deassigned according to a defined procedure to avoid segmentation faults.