**Emergency Room Simulator**

UML Diagram:

A close up of a sign

Description automatically generated

General goal of the simulator:

* The simulator target is to simulate the operation of a hospital, where there will be a certain arrival rate of patients, a certain number of doctors and nurses to take care of the patients
* By the end of the simulator, the program will output the average time patients spent in the hospital (the time when they got in to the time when they finish)

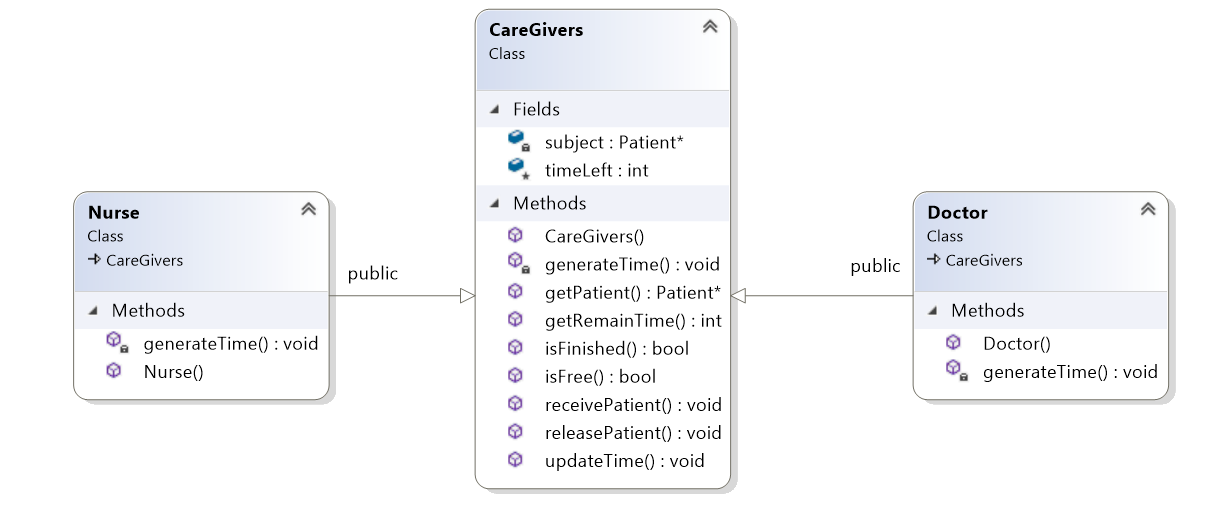
**Specification**

**Class**: There will be 8 classes in total:

* Record, Patient, Resident, Village, Hospital, Caregivers, Doctor, Nurse.

**Class Doctor and Nurse** will inheritance from the **class Caregivers**.

* Since Caregivers itself does not make sense, it will be an abstract class, using virtual method to enforce the design of Nurse and Doctor. This will also help the program to call these method using Caregivers pointer, instead of dynamic cast to the original class



* generateTime() will be virtual method that is different for Nurse and Doctor.
* Nurse and Doctor will hold a pointer to their patient where they are treating, after they finish treating, the patient should not be patient anymore, hence the object will be delete, only the information such as the treatment time, severity is recorded.

**Class Resident** represent the people live in 273Village. Since the population is 2000, there will be 2000 Residents object, all will be stored as pointer in a vector in Village class. The pointer to resident object in the village will be add/remove as the resident go back to the village/get sick

Std::vector<Resident \*> residentVector;

Pseudo-code: Resident \* getSick(){

If (resident vector not empty)

Return residentVector[random index]

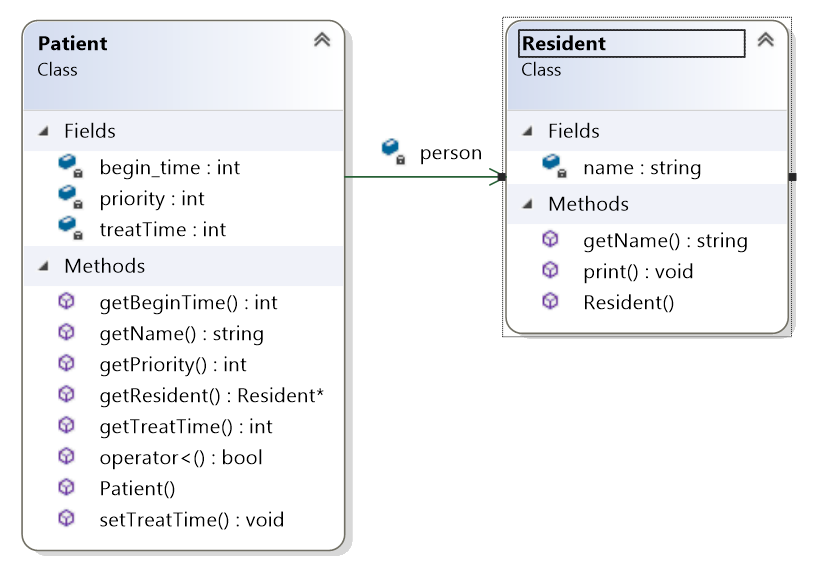
Pseudo-code: void goBack(Resident \*)

residentVector.push\_back(resident)\

Class Resident also contains the name and the method to retrieve the name

**Class Village** will have method to randomly choose a resident as a sick person, it also has method for receiving the resident back to the village as

**Class Patient** is created as a resident enter the hospital and become a patient, every patient class will have a pointer point to a resident, this will be the owner of that patient class



* Patient constructor will construct the object with a pointer to a resident object, a pointer to a record and time when the patient enter the hospital. These information should/only available in the hospital object, therefore this constructor is only called within the Hospital. Patient class also has method for getting the owner name, the record pointer, the resident pointer, a less than operator overload to be used in priority queue. This should compare the priority point, then the name
  + Pseudo-code: bool operator < (Patient& another)

If (this->priority < another.priority)

Return true;

Else if (this->getName() < another.getName())

Return true;

Else return false;

**Class Record** represent a patient record in the hospital. This record will only store the number of visit and how severe each visit is. There are methods to retrieve number of times and severity for each time. This record is stored inside hospital as a map, each record is mapped with a name – assumed that the name of the residents are unique

Std::map<std::string, Record \*> hospitalFile;

**Class Hospital** represent the hospital in the village, this class will contain two vectors storing pointer to caregivers and patients, it will also have a map to match a person name to their record. There will be method in this class to add patient, start the treatment by pair a caregiver to a patient, check to see if the treatment is completed and return back to the village

**Destructor/delete object:**

* since the program is only designed to run the simulator, it is assumed that the program will be closed after the simulator finish and user finish with all their task, which means that all objects should not be deleted until the program ends. Therefore no deconstructor is created as they will be automatically delete when the program exit.
* If the program is extended to do more than just simulating the hospital, then there should be deconstructor for some class (Hospital, Village,…) to prevent memory leak as all the object ( is stored as a pointer, not an object
* One exception is in the caregivers class, where the patient pointer is manually deleted as after the treatment, the patient is gone and new patient class will be created even if the same resident enter the hospital.

Use case:

* Run the simulation:

|  |  |  |
| --- | --- | --- |
| Step | User’s action | System response |
| 1. |  | System ask the arrival rate |
| 2. | User input arrival rate | System ask number of Doctor |
| 3. | User input number of doctor | System ask for number of nurse |
| 4. | User input number of nurse | System run the simulation, display the average time per person |

1. List all the name of people treated by the hospital

|  |  |  |
| --- | --- | --- |
| Step | User’s action | System response |
| 1. | User issues the command to list all name | System list all the name of people treated in the hospital |

1. View the record of the hospital

|  |  |  |
| --- | --- | --- |
| Step | User’s action | System response |
| 1. | User issues the command to retrieve a record | System ask for the name of the record to be retrieved |
| 2. | User input the name for the record | System output the list of time that person is in hospital with the severity of each time |