Xavier Betancourt, Apoorv Joshi, Caleb Grode

Professor Matthew Bell

CS-273-1

12 December 2019

FINAL PROJECT

**Abstract:**

An emergency room simulator is based on the concept of treating patients and recording their diagnosis information. This program takes in doctors, nurses, and patients according to the user and then outputs the record of all the patients that have been treated. The patients are first separated according to the severity of illness which determines the probability of medical attention they require. The record shows patients in an alphabetical order and includes the severity of the patient and the time taken to treat the patient. It also shows the number of times each patient visited the emergency room and by each visit, the severity of the patient becomes low. The patient’s severity determines if they’ll get treated by the doctor or the nurse. Patients with high severity are automatically forwarded to get treatment from a doctor and those with low severity gets treated by the nurse. The number of patients that can be accustomed in an hour is 60. Once the information of the patients has been outputted on the terminal, the user gets a menu with the help of which they can search for the specific patient whose data they want to view and the average time taken by the treatment. The average time is the time a person takes with a specified doctor or a nurse. This simulation runs for a period of one week and all the data that it shows is of this amount of time.

**Use Cases for the Program**

The user’s interactions with the ER set up

|  |  |  |
| --- | --- | --- |
| Step | User’s Action | System’s Response |
| 1. |  | Prompts user for number of doctors to staff the ER |
| 2. | Inputs number |  |
| 3. |  | Prompts user for number of nurses to staff the ER |
| 4. | Inputs number |  |
| 5. |  | Prompts user for number of patients to arrive hourly |
| 6. | Inputs number |  |
| 7. |  | Fills the ER with desired number of nurses and doctors. Sets the hourly arrival rate to that which the user entered |

The user’s interactions with the menu loop

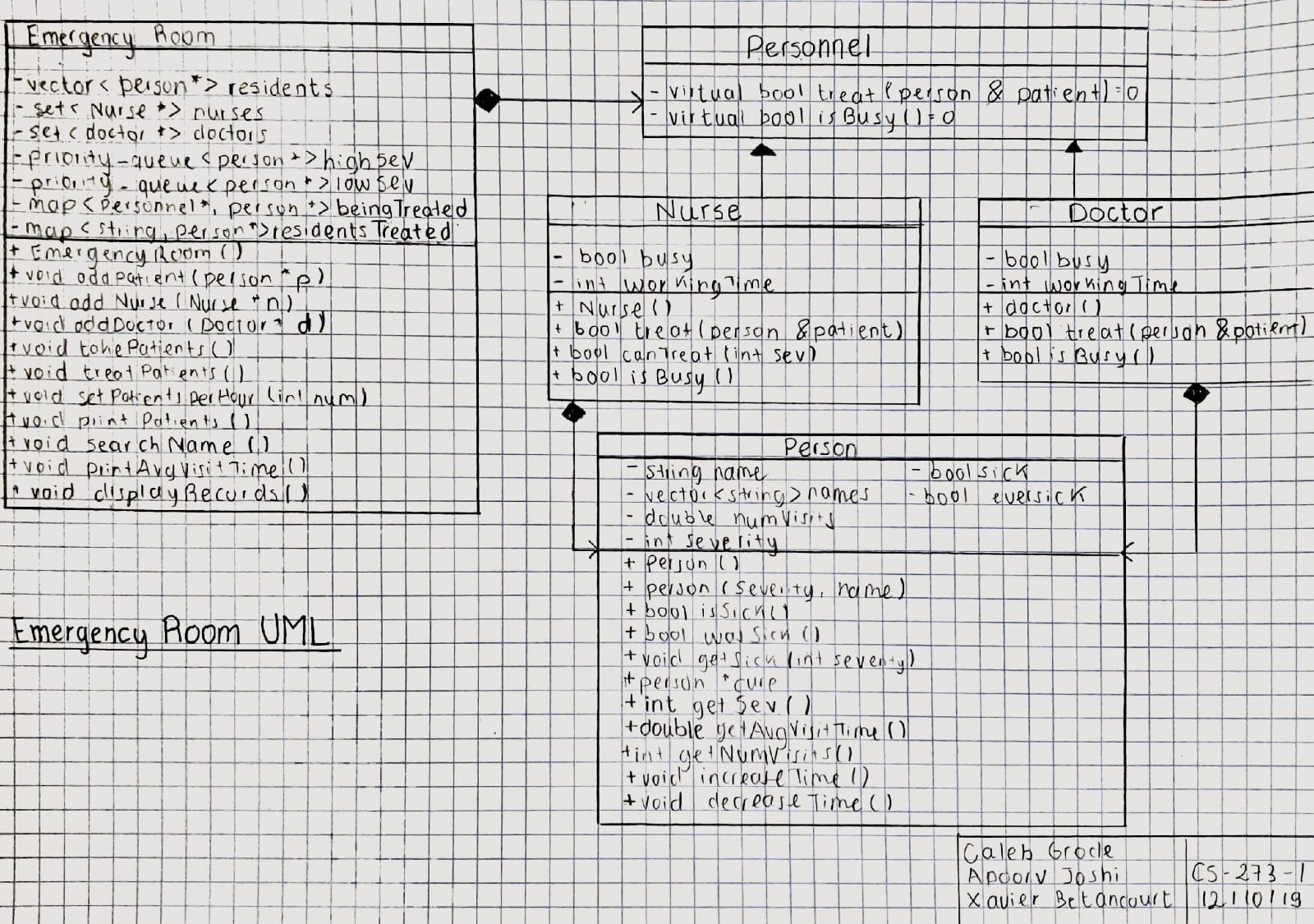
|  |  |  |
| --- | --- | --- |
| Step | User’s Action | System’s Response |
| 1. | Inputs option 1 |  |
| 2. |  | System prints the name, number of visits and average visit time of each patient |
| 3. |  | Restarts menu |

|  |  |  |
| --- | --- | --- |
| Step | User’s Action | System’s Response |
| 1. | Inputs option 2 |  |
| 2. |  | System prompts user to input a name |
| 3. | Inputs name |  |
| 4. |  | Returns the information about the patient whose name was input |
| 5. |  | Restarts menu |

|  |  |  |
| --- | --- | --- |
| Step | User’s Action | System’s Response |
| 1. | Inputs option 3 |  |
| 2. |  | System calculates and displays the average visit time/average severity |
| 3. |  | Restarts menu |

|  |  |  |
| --- | --- | --- |
| Step | User’s Action | System’s Response |
| 1. | Inputs option 4 |  |
| 2. |  | Program exits |

**UML**



**Pseudo Code for some important functions**

A. TreatPatients();

1. Define a set of doctors and an iterator to the beginning of the set
2. Define a set of nurses and an iterator to the beginning of the set
3. While the iterator of doctor set hasn’t reached the end of the set

If the iterator finds a busy doctor

Call treat() to keep treating a patient from the map of patients being treated

Once patient is cured erase it from the map of patients

1. Increment the doctor iterator by one each clock time
2. While the iterator of nurse set hasn’t reached the end of the set

If the iterator finds a busy nurse

Call treat() to keep treating a patient from the map of patients being treated

Once patient is cured erase it from the map of patients

1. Increment the nurse iterator by one each clock time

B. TakePatients();

1. find not busy doctor in set

//assign the not busy doctor and make them begin treating the highest priority patient

//check if the high priority queue is empty

if (not) take and

treat from there

if (true) take

from low priority queue (if it is not empty)

1. iterate through the set of doctors and repeat this process for every non busy doctor

//now that the doctors are treating the highest priority patients:

1. find not busy nurse in the set

if the low priority queue has person(s) in it

take and treat them

C. updateRoomQueue(Emergency room \*r, int seed, vector<string>\*names);

int clockTick = seed;

Seed++; //Incrementing the seed

srand(seed); //generating random numbers

int prob = rand() % 10; //generating probability for the patient

If (check if the person is coming)

if (probability < 7) //70% chance; low severity

rand() % 10;

//create and add a new person with a random name to the ER

if (probability > 6 and probability < 9) //20% chance; med severity

11 + rand() % (15-11+1); //number between 11 and 15

//create and add a new person with a random name to the ER

if (probability > 8) //10% chance; high severity

16 + rand() % (20-16+1); //number between 16 and 20