Jonathan Laughlin

CS273

Pete Tucker

7/15/19

Final Project Design Documents

**Requirements specifications**

The software for the Emergency Room Simulator should be able to simulate an emergency room in a small town as accurately as possible. This means it should be able to simulate different injuries and illnesses and give the more severe of them a higher priority on the waiting list. There must be an equal chance of everyone within the small town needing to go to the hospital as currently they are all healthy. Due to the townspeople all being healthy, there must be differentiating chances of each of them receiving a severe illness. Because the simulation is based on a hospital, all the medical records from patients must be stored for future treatments (i.e. list of surgeries they have had or illnesses) therefor the hospital must keep records of each patient who was treated there. The simulation will need to triage a patient when they arrive in the waiting room, and they will need to be placed by priority in the waiting room. The more severe their illness, the higher the priority. There must be both nurses and doctors within this simulation; each being able to treat differentiating injuries and at differing speeds. The simulations must be as accurate as possible.

**Use Case:**

Running program

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **System’s Response** |
| 1. | User issues command to start the program to simulate an emergency room for the town of 273ville. |  |
| 2. |  | The Emergency room simulator is started, and the town of 273ville’s emergency room is initialized. In addition, the system prompts the user for the total run time of the simulation in hours. |
| 3. | User inputs the total runtime of the simulation. | If user enters a zero or negative arrival rate, system re-prompts. |
| 4. |  | System changes the hours user entered into minutes. System prompts for arrival rate of patients. |
| 5. | User inputs arrival rate of patients. | If user enters below 1 or more than 60 system re-prompts. |
| 6. |  | System prompts user for number of doctors working. |
| 7. | User inputs number of doctors. | If user inputs below 1 doctor system re-prompts. |
| 8. |  | System prompts user for number of nurses working. |
| 9. | User inputs number of nurses. | If user inputs below 1 nurse system re-prompts. |
| 10. |  | Simulation runs calculating the average visit time of all patients. It also displays a menu with options to list all names of patients treated, as well as retrieve the medical record of the patient by name. |

List all names of residents that were treated

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **System’s Response** |
| 1. | User issues command to list all names of patients who were treated in the emergency room. |  |
| 2. |  | System searches through records of patients treated and displays each patients name. Program exits. |

Retrieve record of resident by name

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **System’s Response** |
| 1. | User issues command to retrieve record of resident of 273ville by their name. |  |
| 2. |  | System prompts for name of resident. |
| 3. | User enters name of resident. | If resident does not exist, system re-prompts for name. |
| 4. |  | System searches through residents and displays their medical record. Prompts user for name of resident. If exit command entered, program exits. |

**UML Diagram:**

|  |
| --- |
| **Medical\_Records** |
| - int number\_visits  - vector<Current\_Visit> visit\_record |
| + Medical\_Records()  + void print\_medical\_record()  + void add\_visit(Current\_Visit \* visit) |

|  |
| --- |
| **Person** |
| **+** string name  + int age  + bool can\_admit  + Medical\_Records \*medical\_history |
| + Person(int age, string name)  + bool get\_can\_admit()  + void set\_can\_admit()  + int get\_age()  + string get\_name()  + void print\_medical\_record() |

|  |
| --- |
| **Current\_Visit** |
| **-** int illness\_severity  - int visit\_time  - int arrival\_time  - int start\_service\_time  - int discharge\_time |
| + Current\_Visit(int arrival\_time)  - void set\_illness\_severity()  +int get\_illness\_severity()  +void set\_discharge\_time(int clock)  + void set\_start\_service\_time(int clock)  + void get\_start\_service\_time()  + void print\_visit() |

|  |
| --- |
| **Patient** |
| - int arrival\_time  - int discharge\_time  - Current\_Visit \* visit  + Person \* person |
| + Patient(int clock, Person\* person)  + Current\_Visit \* get\_visit()  + bool operator< (const Patient& other) const |

|  |
| --- |
| **Hospital** |
| - vector<Patient \*> patient\_records  - vector<Person \*> town  - priority\_queue<Person \*> current\_patients  - ServiceRoom \* Emergency\_Room;  - int number\_doctors  - int number\_nurses  - int arrival\_rate |
| + Hospital(vector<string> Town, int arrival\_rate, int number\_doctors, int number\_nurses)  + void update(int clock)  + bool check\_insert(Patient \* patient)  + void menu() |

|  |
| --- |
| **Simulation** |
| **-** int clock  - string name  - int total\_time  - int number\_nurses  - int number\_doctors  - int arrival\_rate  - Hospital \* hospital\_simulation  - vector<Person\*> town |
| + Simulation()  + void get\_data()  + void run\_sim()  + void create\_273ville()  + void menu() |

|  |
| --- |
| **ServiceRoom** |
| - map<Doctor\*,Patient\*> doctors  - map<Nurse\*,Patient\*> nurses  - int num\_nurses  - int num\_doctors |
| + ServiceRoom(int num\_doctors, int num\_nurses)  + void update\_doctor(int clock)  + void update\_nurse(int clock)  + int get\_doctors\_size()  +int get\_nurses\_size()  + void service\_patient\_doctor(Patient \*patient, int clock)  + void service\_patient\_nurse(Patient \*patient, int clock)  + bool nurse\_is\_full()  + bool doctor\_is\_full() |

|  |
| --- |
| **Caretaker** |
| # int service\_time |
| + Caretaker()  + virtual void random\_service() = 0  + int get\_service\_time() |

|  |
| --- |
| **Nurse** |
| - int service\_time |
| + Nurse()  + void random\_service() |

|  |
| --- |
| **Doctor** |
| - int service\_time |
| + Doctor()  + void random\_service() |

|  |
| --- |
| **Random** |
| - int probability |
| + Random()  + void random\_probability()  + int random\_servity()  + int random\_time(int max)  + double random\_dbl()  + int random\_age()  + int random\_person() |

**Pseudo -code:**

Hospital.h update(int clock)

Calculate if new patient has arrived

If new patient arrived insert into waiting list based on their priority and add record to hospitals records

Set patient no longer able to admit to hospital

If there are patients waiting

If first in list has below 11 priority

If not all nurses occupied

Send patient to room for nurse to treat

Remove patient from waiting list

Else if not all doctors are occupied

Send patient to room for doctor to treat

Remove patient from waiting list

Else update doctors and nurses

Else if first in list has below a 21 priority

If not all doctors are occupied

Send patient to room for doctor to treat

Remove patient from waiting list

Else update doctors

ServiceRoom.h update\_doctor(int clock)

Go through rooms doctors are treating patients

If service time for current room is up

Discharge patient from hospital

Add this visit to their records

Set patient to be able to admit into hospital

Erase room holding patient and doctor, as it no longer exists

Else go to next room

ServiceRoom.h update\_nurse(int clock)

Go through rooms nurses are treating patients

If service time for current room is up

Discharge patient from hospital

Add this visit to their records

Set patient to be able to admit into hospital

Erase room holding patient and nurse, as it no longer exists

Else go to next room