Nate Idica

CS-273: Data Structures

Dr. Pete Tucker

19 July 2019

Emergency Room Simulation: Project Specifications

**REQUIREMENT SPECIFICATIONS:**

All based on the user’s input, the emergency room simulator is supposed to mirror what it would be like in a hospital’s emergency room during one week. The user can choose the amount of patients that show up per hour, as well as, the amount of doctors and nurses that are currently at the emergency room of the week that is being modeled. First, the text files that contain the 2000 people’s names that are simulated in CS273Ville are opened and put into the first and last name vectors, where it will be used later. The function of enteringData() in EmergencyRoom class will ask the user a set of questions, mainly to input the amount of patients show up within the hour, and the amount of doctors and nurses present in the emergency room or hospital at the time. After all of this information is inputted and passes the exceptions, runShrinersSimulation() in EmergencyRoom is then called and starts the simulation where the names of each patient, from the text files, are moved into the different queues so that they can be treated through the emergency room simulation.

Via the simulation function, if the amount of patients\_per\_hour is larer than the random number that is generated through the RandomsGenerator class, then it will go ahead and choose a random number between 1 and 2000 to determine a first and last name from the vectors containing the names from the text files. From there, it will go ahead and create a new patient into the doctor or nurse queue, depending on the illness level. The illness level is dependent on the clock and their name and the random Illnes Level function from RandomsGenerator class, where the chances of having an illness level of 1 to 10 is 70%, illness level of 11 to 15 is 10%, and illness level of 16 to 20 is 10%. This function will continue to run until it reaches the amount of minutes in a week, which is 10080 minutes. This will take place in the waiting room queue.

From the waiting room queue, the patient and simulation is then moved to the treatment queue. When moving queues, the clock will update as well as the clocks for the doctors and nurses in the treatment queue. If there is a patient awaiting treatment, and the queue of doctors is not empty, the patient is then brought the queue. If the amount of treatment time is less than how they have been waiting in the treatment queue, pop the patient from treatment queue since they are fully treated and can be sent to the discharge queue. The amount of doctors are incremented to those that have been treating patients. If there are more than 0 doctors in the waiting room queue, then a new patient can be brought into the treatment queue and be treated, thus decrementing the number of doctors in the waiting room queue and pushing a patient into the queue of doctors of the treatment room queue. This will run the same way for the nurses, but they will treat patients that have an illness level of 1 to 10.

From the treatment queue, the simulation then moves into the discharge queue and updates the clock using the update function. When the queue of doctors of the discharge queue class is not empty, the patient is then in the front of the queue of doctors and popped from the queue and the patients discharge time is set to the time on the clock. The arrival time of patient is subtracted with the discharge time of the patient to set the amount of time the visit took the patient. Once the patient is discharged, the set patients treated function increments every time a person is treated and discharged.

When done in the Discharge class, the program will return to source. It will then call the showStats function of the Emergency Room during this frantic week. The program will ask the user to select from the 3 options of the switch case menu (1. Listing all the treated patients; 2. Finding a patient by name and showing the records associated with the specific patient; and 3. Will basically end the program). If the user chooses option 1 or 2, after the information is displayed, it will go back to the switch case menu and ask the user to choose another option. If they would like to end the program, successfully, they would go ahead and choose option 3.

**USE CASES:**

Number 1:

* **PROGRAM**: Prompts user to enter the number of patients that show up in a hour, from 1 to 60.
  + **USER**: Enters the number of patients that show up in within an hour.
* **PROGRAM**: Prompts user to enter the number of doctors currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters the number of doctors in range.
* **PROGRAM**: Prompts user to enter the number of nurses currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters a number outside of the 1-100 range.
* **PROGRAM**: Throws the exception, out of range, error to the user.
  + **USER**: Re-enters a number inside of the range.
* **PROGRAM**: Displays the user’s desired information, via input. Displays stats based on the information inputted and gives the user the option to look at any additional information.
  + **USER**: Enters 3 to end the program.

Number 2:

* **PROGRAM**: Prompts user to enter the number of patients that show up in a hour, from 1 to 60.
  + **USER**: Enters a string instead of a number.
* **PROGRAM**: Throws the exception, incorrect variable type, error to the user.
  + **USER**: Re-enters a number inside of the range.
* **PROGRAM**: Prompts user to enter the number of doctors currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters the number of doctors in range.
* **PROGRAM**: Prompts user to enter the number of nurses currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters a number inside of the 1-100 range.
* **PROGRAM**: Displays the user’s desired information, via input. Displays stats based on the information inputted and gives the user the option to look at any additional information.
  + **USER**: Enters 3 to end the program.

Number 3:

* **PROGRAM**: Prompts user to enter the number of patients that show up in a hour, from 1 to 60.
  + **USER**: Enters the number of patients that show up in within an hour.
* **PROGRAM**: Prompts user to enter the number of doctors currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters the number of doctors in range.
* **PROGRAM**: Prompts user to enter the number of nurses currently at the emergency room/hospital, from 1 to 100.
  + **USER**: Enters a number inside of the 1-100 range.
* **PROGRAM**: Displays the user’s desired information, via input. Displays stats based on the information inputted and gives the user the option to look at any additional information.
  + **USER**: Enters 1 to list all of the patients that have been treated in the emergency room.
* **PROGRAM**: Gives entire list of patients treated along with their corresponding illness levels. Returns to the menu after the list is presented to the user.
  + **USER**: Enters 3 to end the program.

**PSEUDO-CODE:**

shriners.enteringData();

1. Call addPeopleToVector()
   1. Opens text files of first and sur names. Puts contents into the first and sur name vectors, then sends these vectors to allPatients object.
2. Display a welcome message and asks user to enter amount of patients that show up within an hour.
3. User enters amount of patients and readInt function checks exceptions with what was inputted by user.
4. Prompts user to enter the number of doctors present.
5. User enters amount of doctors and readInt function checks exceptions with what was inputted by user.
6. Prompts user to enter the number of nurses present.
7. User enters amount of nurses and readInt function checks exceptions with what was inputted by user.
8. Sets the number of patients that show up within the hour in object in waitingRoomQueue.
9. Sets first name vector in waitingRoomQueue from all\_patients get first name vector.
10. Sets sur/last name vector in waitingRoomQueue from all\_patients get last name vector.

shriners.runShrinersSimulation();

1. Function loops as long as the count on the clock is less than the amount of minutes in a week.
2. Goes to waiting room queue and call update passing through with int clock.
   1. Set the number, illness level, time of doctor treatment, and time of nurse treatment if the random number that is generated between 0 and 1 is less than the rate of patients per minute.
   2. If the number of nurses or doctors are greater than 0
      1. Push a new patient to the queue of doctors in the waiting room queue if the number of doctors is greater than 0 and the illness level is greater than 10.
      2. This is the same for the queue of nurses, only difference is that the illness level is that less than or equal to 10.
3. Goes to treatment queue and calls update passing through with int clock.
4. Update doctor
   1. If queue of doctors in treatment queue is not empty
      1. Set the patient to the front of the queue of doctors.
         1. If the patient has been in the queue for past the set treatment time, pop from the queue of doctors and increment the number of doctors.
         2. Set patient to discharge time.
         3. Send patient to queue of doctors in discharge queue.
   2. If waiting room queue number of doctors are more than zero.
      1. Point a patient to the front of the queue of doctors in the waiting room queue if the queue of doctors in the waiting room queue is not empty.
      2. Pop from the waiting room queue, decrement the number of doctors in the waiting room queue and set the patient start treatment time to the clock.
      3. Push the patient into the queue of doctors in the treatment class.
5. Update nurse
   1. If queue of nurses in treatment queue is not empty
      1. Set the patient to the front of the queue of nurses.
         1. If the patient has been in the queue for past the set treatment time, pop from the queue of nurses and increment the number of nurses.
         2. Set patient to discharge time.
         3. Send patient to queue of nurses in discharge queue.
   2. If waiting room queue number of nurses are more than zero.
      1. Point a patient to the front of the queue of nurses in the waiting room queue if the queue of nurses in the waiting room queue is not empty.
      2. Pop from the waiting room queue, decrement the number of nurses in the waiting room queue and set the patient start treatment time to the clock.
      3. Push the patient into the queue of nurses in the treatment class.
6. Go to the discharge queue and update passing through with int clock.
   1. Point the patient to the front of the queue of doctors if the queue of doctors in the discharge queue is not empty.
   2. Set patient to the time on the clock.
   3. Pop the patient from queue of doctors.
   4. Set the visit time of the patient.
   5. Set the amount of patients treated and increment the number of patients treated.
7. Increment clock.

shriners.showingStatistics();

1. getNumOfPatientsTreated function in DischargeQueue class displays the number of patiens that have bee ntreated.
2. getVisitTime function in DischargeQueue class displays the average visit time.
3. Displays a message for the user to select from the menu.
4. Calling the menu function.
   1. User enters the menu option input and readInt function checks exceptions with what was inputted by user.
   2. If user chooses Case 1:
      1. listAllPatients() function is displayed to show the user the patients that have been treated.
      2. Calling menu function.
   3. If user chooses Case 2:
      1. Prompts user to enter a patient’s name.
      2. Based on the user’s input, it searches for the first and last name vectors for the name inputted.
      3. If the name of the patient is found, it displays the stats of the patient.
      4. If the name of the patient is not found, the message of “patient not found” is displayed.
   4. If user chooses Case 3:
      1. Ends the program.

**UML DIAGRAM**

Also scanned via .jpeg file on Github repository.

A close up of text on a white background

Description automatically generated