Elisha Parslow

CS273 Data Structures

Final Project: Hospital Simulator

Project Specifications

**Requirement Specification:**

The hospital simulator is meant to simulate one week’s time in an emergency room based on the user’s desired input. The user has the option to select how many patients arrive per hour, up to sixty patients, as well as how many doctors and how many nurses are working during the week. Before the user enters this data, the text files of all 2000 people’s names being simulated in CS273ville will be opened and put into first name and last name vectors for use later. After the program asks the user for this information through the function EnterData() inside the EmergencyRoom class, the program will then set the user entered numbers for use in the waitingRoomQueue. After this function is completed, the program will access the runGreysAnatomySimulation() function inside of the EmergencyRoom class to move patients through the queues to be treated in the emergency room.

Inside the simulation function, based on clock time of every second, the function will try to addToQueue(int clock) inside waitingRoomQueue. Inside this function, odds will be determined of a patient entering that minute based on what the user entered for patientsPerHour in comparison to a random number generated from the RandomsGenerator class. If patientsPerHour is larger than the random number, the function will then generate another random number between 1 and 2000 to grab a first and last name from the vectors of names. It then will push a new Patient into the either the doctorQueue or nurseQueue depending on the illness level taking the clock, first name, sir name, and will set the patient’s illness level. Illness level is based on the randomIllnessLevel function from the RandomsGenerator class following the probability of a patient having a 70% chance of an illness level from 1 to 10, 20% chance of an illness level from 11 to 15 and a 10% chance of an illness level from 16 to 20. The simulator will run through this function until it reaches 10,080 seconds, the time in a week.

The simulator will then go to treatmentQueue where it will update(int clock) and then updateDoctor(int clock) and updateNurse(int clock). If the doctorQueue is not empty, there is a patient waiting for treatment and then they will then be brought to the front of the queue. If their total treatment time becomes less then how long they have been in the treatment queue, pop the patient from the treatment queue because they have been fully treated and send them to the discharge queue. Also increment the amount of doctors that have been treating patients. If the number of doctors in the waitingRoomQueue is greater then zero, a new patient can be brought into the treatment queue to be treated. Then decrement the number of doctors in the waitingRoomQueue and push a patient into doctorQueue of TreatmentRoomQueue. The updateNurse(int clock) function will run the exact same except it will take care of patients with an illness level between one and ten.

The simulator next goes into the DischargeQueue class calling the update(int clock) function. If the doctorQueue of the DichargeQueue class is not empty, point the patient to the front of the doctorQueue and pop that patient from the queue. Set the patients discharge time to the time on the clock. The n call the setVisitTime of patient using the function in DischargeQueue and subtract arrivaltime from discharge time of that patient. Call the setPatientsTreated function to increment that another patient was treated.   
 Once done with the DischargeQueue class, the program will leave the runGreysAnatomySimulation function inside of EmergencyRoom and return to main. It will then call the showStats function of EmergencyRoom to display the events of the hospital for the week. The stats will display the number of patients treated and the average visit time of each patient. The program then asks the user to select from a switch case menu with three options: 1. List all the residents treated. 2. Find a patient by name. 3. End the program. If the user selects 1, all the patients will be treated, if the user selects 2, they will be asked to enter a name and the name will be searched in the first names vector. If the user selects 3, the program will end. At the end of 1 and 2, the program will take the user back to the menu until the user selects 3 to successfully end the program.

**Use Cases:**

**#1:**

Program: Prompts user to enter the number of patients per hour from 1 to 60.

User: Enters the patients per hour.

Program: Prompts user to enter number of doctors from 1 to 100.

User: Enters number of doctors out of range.

Program: Throws exception that user is out of range.

User: Re-enters a new range.

Program: Prompts user to enter number of nurses from 1 to 100.

User: Enters number of nurses.

Program: Takes in user’s desired information and uses it to run a solution. Divides user’s patient per hour by 60 to give a rate. Program then runs simulation and sends users information through 10080 minutes in an emergency room. Program completes simulation then displays stats on simulation. The program gives user menu options for additional statistics.

User: enters 3 to end the program.

**#2:**

Program: Prompts user to enter the number of patients per hour from 1 to 60.

User: Enters the patients per hour as a symbol.

Program: Throws exception that user inputs incorrect variable type.

User: Re-enters patients per hour.

Program: Prompts user to enter number of doctors from 1 to 100.

User: Enters number of doctors.

Program: Prompts user to enter number of nurses from 1 to 100.

User: Enters number of nurses.

Program: Takes in user’s desired information and uses it to run a solution. Divides user’s patient per hour by 60 to give a rate. Program then runs simulation and sends users information through 10080 minutes in an emergency room. Program completes simulation then displays stats on simulation. The program gives user menu options for additional statistics.

User: Enters 1 to display all of the patients treated.

Program: Displays list of all of the patients treated in the hospital simulation. Returns to menu and asks user for a new input.

User: Enters 3 to end program.

**#3:**

Program: Prompts user to enter the number of patients per hour from 1 to 60.

User: Enters the patients per hour.

Program: Prompts user to enter number of doctors from 1 to 100.

User: Enters number of doctors.

Program: Prompts user to enter number of nurses from 1 to 100.

User: Enters number of nurses out of range.

Program: Throws exception that input is out of range. Program prompts user to enter number of nurses.

User: Re-enters number of nurses.

Program: Takes in user’s desired information and uses it to run a solution. Divides user’s patient per hour by 60 to give a rate. Program then runs simulation and sends users information through 10080 minutes in an emergency room. Program completes simulation then displays stats on simulation. The program gives user menu options for additional statistics.

User: enters 2 to enter a specific patient’s name.

Program: prompts user to enter a name.

User: Enters a name.

Program: Searches vector of first names and sir names. Does not find name, displays message to user that name was not found. Displays menu and prompts user for menu option.

User: Enters 2 to enter specific patient’s name.

Program: Prompts user to enter a name.

User: Enters a name.

Program: Searches vector of first names and sir names. Finds name and displays patient stats. Displays menu and prompts user for menu option.

User: Enters 3 to end program.

**Pseudo-code:**

EnterData();

1. Call addPeopleToVector() function.
   1. Open text files of first name and sir names.
   2. Put contents into first name and sir name vectors.
   3. Send those vectors to allPatients object of class All Patients.
2. Display welcome message and prompt user for patients per hour.
3. User enters patients per hour. Program checks for exceptions through readInt function.
4. Prompts user for number of doctors.
5. User enters number of doctors. Program checks for exceptions through readInt function.
6. Prompts user for number of nurses.
7. User enters number of nurses. Program checks for exceptions through readInt function.
8. Sets patients per hour in waitingRoomQueue object.
9. Sets first name vector in waitingRoomQueue from allPatients getFNVector function.
10. Sets sir name vector in waitingRoomQueue from allPatients getSNVector function.

runSimulation();

1. Loop through a clock count while less than one weeks’ time; 10080 minutes.
2. Go to waitingRoomQueue and call addToQueue passing through clock.
   1. If a random number between 0 and 1 is less than the rate of patients per minute, set num, illnessLevel, totalDoctorTreatment, and totalNurseTreatment.
   2. If number of nurses or doctors is greater than zero
      1. If illness level ios greater than 10 and num of doctors is greater than 0, push a new patient to the doctorQueue inside WaitingRoomQueue class and add the patient to patient records.
      2. If illness level is less than or equal to 10 and nurses is greater than zero, push a new patient to the nurseQueue inside of WaitingRoomQueue class and add the patient to patient records.
3. Go to treatmentQueue and call update passing through clock.
4. Update doctor.
   1. If the doctorQueue inside treatmentQueue is not empty
      1. Set a patient pointer to the front of the doctorQueue.
         1. If the patient has been in the queue past their set treatment time, pop from the doctorQueue and increment number of doctors.
         2. Set patient discharge time.
         3. Send the patient to the doctor queue inside dischargeQueue.
   2. If the waiting room queue number of doctors is greater than zero
      1. While the doctorQueue in waitingRoomQueue is not empty, point a patient object to the front of the doctorQueue in waitingRoomQueue. Pop from the watingRoomQueue. Decrement number of doctors in waitingRoomQueue. Set the patients start treatment time to clock. Push the patient into the doctorQueue of TreatmentQueue class.
5. Update Nurse.
   1. If the nurseQueue inside treatmentQueue is not empty
      1. Set a patient pointer to the front of the nurseQueue.
         1. If the patient has been in the queue past their set treatment time, pop from the nurseQueue and increment number of doctors.
         2. Set patient discharge time.
         3. Send the patient to the doctor queue inside dischargeQueue.
   2. If the waiting room queue number of nurses is greater than zero
      1. While the nurseQueue in waitingRoomQueue is not empty, point a patient object to the front of the nurseQueue in waitingRoomQueue. Pop from the watingRoomQueue. Decrement number of nurses in waitingRoomQueue. Set the patients start treatment time to clock. Push the patient into the nurseQueue of TreatmentQueue class.
6. Go to dischargeQueue and call update passing in clock.
   1. If the doctorQueue inside of the dischargeQueue class is not empty
      1. Point a patient object to the front of the doctor queue.
      2. Set the patient discharge time to clock.
      3. Pop the patient from doctor queue.
      4. Se the visit time of the patient.
      5. Call the setPatientsTreated and increment the number of patients treated.
7. Increment clock.

showStats();

1. Display number of patients treated calling the function getPatientsTreated inside of the dischargeQueue class.
2. Display the average visit time calling the getVisitTime function inside the dischargeQueue class.
3. Display message to user to select from a menu.
4. Call menu function.
   1. Prompt user for a menu option using the readInt function to throw exceptions from invalid input.
   2. User enters input and program checks for exceptions.
   3. If case 1:
      1. Display the patients treated calling the listPatientsTreated fuction.
      2. Call menu function.
   4. If case 2:
      1. Prompt user to enter a patients name.
      2. Search first and sir name vectors for name.
      3. If found, display patient’s stats, if not display patient not found message.
   5. If case 3:
      1. End program.

**UML Diagram: (Also in the final project in Visual Studio for a more legible version)**

