Requirements Specification:

* Decide on a real world problem to simulate
* High Level UML Diagram
* Use case
  + Tick Tock use case
  + Write user input and outcome
* Prototype
* List of requirement specifications
* Pseudo code
* Always updated designs of all of those
* Inheritance and polymorphism
* Set, map, and/or hash table
* One or more queues
* One page project summary report
  + Includes answers to the questions given below in the project descriptions
  + Includes the required graph
  + Indicates what was changed from your initial design in your final implementation
  + If implementation did not work, describe your design, implementation challenges, and lessons learned
* Simulate a hospital emergency room for a town pop. 2000
* Every person is equally likely to go to the emergency room
* Run the emergency as efficiently as possible
* When the patient arrives, they are triaged and given a illness severity number from 1 to 20
  + Higher priority value means a more ill person and is treated first
  + Priority 1 - 10 occurs 70%
  + Priority 11 - 15 occurs 20%
  + Priority 16 - 20 occurs 10%
* Keep record of all visits
  + Number of visits
  + Severity of each visit
* Examine a week in emergency room (7x24x60) minutes
* User input data
  + Average hourly patient arrival
  + Number of doctors
  + Number of nurses
* Calculate average arrival to discharge time
* At the end of simulation display a menu with options to list the names of all residents that were treated, and retrieve the record of a resident by name
* For your final report and presentation
  + Compare and comment on the average patient visit time (for some fixed patient arrival rate) when the emergency room has
    - 1 doctor and 1 nurse
    - 1 doctor and 2 nurses
    - 2 doctors and 1 nurse
  + Display a plot of the visit time for increasing patient arrival rates, for a combination of doctors and nurses of your choice

High Level UML:

At the beginning, these were the classes I thought I needed:

* Hospital
  + Contains Healthworkers
  + Contains Citizens
  + Private:
    - Priority queue of admitted patients
    - Queue of doctors
    - Queue of patients
  + Public:
    - Citizens will be put in the priority queue based on the priority given to them
* Healthworker
  + Can be a Doctor
  + Can be a Nurse
  + Doctors and nurses need to be split because they can help different people and take different times to do it
* Doctor
  + Is a Heathworker
* Nurse
  + Is a healthworker
* Visit
* Priority Queue (I forgot I did not need to manually implement this)
* Citizen
* 273ville (This was going to be to read the names in files and put all the citizens in an array, but then I realized that would be stupid)

Eventually, these were the classes I finalized on:

* Citizen
  + Pirvate:
    - Name
    - Priority
  + Public:
    - Get and set for private variables
    - Overloaded < operator for priority queue
* Doctor
  + Private:
    - Healthworker variables
    - How much time to spend on the current patient
  + Public:
    - Update work cycle every minute
* Healthworker
  + Private:
    - The current patient they are helping
    - How long they have been with that patient
  + Public:
    - Getters and setters
    - Virtual update function
* Hospital
  + Private:
    - High priority citizen queue
    - Low priority citizen queue
    - Doctor queue
    - Nurse queue
  + Public:
    - Getters and setters
    - Update to bring in more patients and help the patients in need
* Nurse
  + Exactly the same as doctor
* Simulator
  + Private:
    - How long to run the simulation
    - How many nurses/doctors
    - Rate of new patients
  + Public:
    - Add initial data
    - Run the simulation at every minute
    - Show the end stats
* Visit
  + Private:
    - The patient
    - The healthworker
    - Time of arrival to hospital
    - Time of getting help
  + Public:
    - Getters and setters

Tick Tock use case:

|  |  |
| --- | --- |
| User Input | Expected Outcome |
| Number of doctors |  |
|  | Push that many doctors into the doctor vector |
| Number of nurses |  |
|  | Push that many nurses into the nurse vector |
| Rate of new patients per hour |  |
|  | Divide this number by 60 |
| Total time |  |
|  | Run the simulation for this long |

Pseudo code:

Create simulation

Input number of doctors, number of nurses, rate of patients, total time for the simulation to run

Create hospital with these number of doctors and nurses - this includes making enough doctors and nurses

Run simulation:

See if anyone became ill each minute

If they do, give them a random severity along this path (1-10:70%, 11-15:20%; 16-20:10%)

Add this information to a visit

Check if a doctor is free every minute

If a doctor is free and there is someone to treat above 11, give the doctor the patient, start them working on the patient

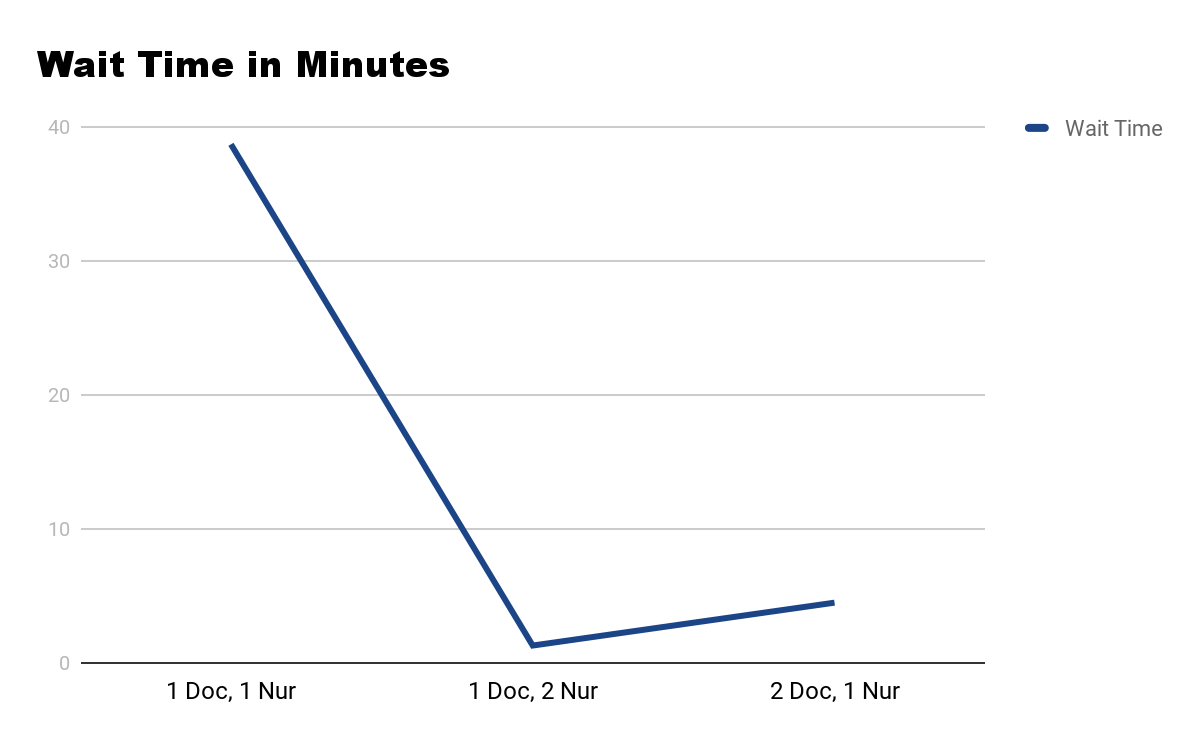
Then ditto for nurses and patients under 10

Then ditto for doctors and patients under 10

Add this information to a visit

End simulation

Display information



Changes from Initial to Final design:

I removed the Priority Queue and 273ville classes because they were useless

I did not realize how many getters, setters, and constructors I was going to use, so adding those was a bit of work

Initially I did not have the idea of how I was going to ask the doctors, then nurses, then doctors again if they were free to take a patient, but I did implement that.

The implementation of the menu was okay, but the console looks kind of messy. I wish I could have worked on that front-end some more

At the very beginning I forgot the priority queue, but that helped as I put it in my first prototype.