**Rudyjay Keopuhiwa**

**CS273**

**Pete Tucker**

**July 17,20189**

**Final Specification:**

1. Requirements specifications:
   1. The different software that will be needed is multiple classes. The list of classes that are required is a Patient class that will hold the information on the patient of the hospital. Some of that information will be a name of the patient and a priority number that will define the severity of the illness. Patients with a higher priority number will always be treated ahead of patients with lower numbers. A nurse and doctor class will also be used. These two classes will be used to treat the patients. Nurses will only be able to treat patients with priority 1-10 but doctors will be able to treat patients with priority with 1-20. Also, a simulation class will need to be used to help run the simulation of the emergency room. Within the simulation class it will hold a record of the number of people served and amount of time it took. Another class that will need to be created is a waiting room queue that will be a priority\_queue of the patients. Also, a Service set class will be needed to access different patients to check if they have been treated or not. Lastly a class that may be needed is a random class that will help assign the priority number of the patients randomly.
2. Use Cases:
   1. Use Case for treating a patient

|  |  |  |
| --- | --- | --- |
| Steps  1  2  3  4 | User Actions  User inputs average rate and number of doctors and nurses working | System response  System starts to run simulation based on the information the user inputted.  When a patient comes to the simulated emergency room they are put in a queue and is placed in the queue depending on their priority  Depending on how much nurses and doctors there are present at the hospital the program will react accordingly |

* 1. Use Case for inputting data values

|  |  |  |
| --- | --- | --- |
| Steps  1  2  3  4 | User Actions  User runs program  User enters data values | System response  System prompts user to enter different data values  System stores those data values and later are used within the simulation |

* 1. Use Case for running simulation

|  |  |  |
| --- | --- | --- |
| Steps  1  2  3 | User Actions  User issues the command to  Running the simulation | System response  System runs simulation using a for loop and update() methods in service and waiting class  Program reacts accordingly to what is called in the update methods until the for loop ends |

|  |
| --- |
| Simualator |
| -int clock  -read\_int  -vector<string> residents  -Waitingqueue\* waiting\_queue  -Serviceset\* service\_set  -int read\_int(const string prompt, int low, int high) |
| Simulator()  +void create\_data()  +void enter\_data()  +void run()  +void menu() |

|  |
| --- |
| ServiceSet |
| -int total\_time  -int number\_served  -Waitingqueue\* waitingqueue  -multiset<Patient\*>myset  -vector<Nurse\*> caregivers |
| +int doctors  +int nurses  +void update()  +void set\_waiting\_queue()  +void create\_caregivers()  +void get\_num\_served()  +void get\_total\_time() |

1. UML Diagrams:

|  |
| --- |
| WaitingroomQueue |
| -double arrival\_rate  -int total\_wait  -int num\_served  -queue<Patient\*>the\_queue  -vector<string> residents |
| +vector<Patient\*>resident\_record  +void update()  +void set\_arrival\_rate()  +int get\_visited()  +void set\_residents(vector<string> other) |

|  |
| --- |
| Doctors: |
| Doctor(int num)  +void set\_service\_time |

|  |
| --- |
| Random |
| Random()  +Random(int seed)  +int next\_int(int n)  +double next\_double() |

|  |
| --- |
| Patient |
| +int number  +Patient(int number, int clock)  +int service\_time  +int arrival\_time  +int start\_service\_time  +int priority\_num  +int num\_visits  +vector<int> priorityNumbers  +string name  +Nurse\* assignedCare  +bool operator<(const Patient& other) |

|  |
| --- |
| Nurses |
| -int nurseNumber  -bool isOpen |
| +bool getOpen()  +void changeOpen()  +void set\_service\_time |

1. Pseudocode:
   1. update() for waiting room queue
      1. If a patient arrives it creates a new patient dynamically and adds it to the queue
      2. A random priority number is selected for the patient
      3. Based on patient priority number they are put into the queue (1 being the lowest and 20 being highest)
      4. Program then checks to see if a doctor or nurse is open for service
      5. If a doctor or nurse is open then it adds patient to service queue and then returns
      6. If a doctor or nurse is not opens then it returns and continues the simulation
   2. update() for serviceset
      1. Checks to see if the service multiset is empty
      2. if the service multiset is empty it adds a patient to the multiset and assigns a doctor or nurse to them.
      3. Then the program calls set\_service\_time() and randomly generates a number that will represent the minutes it will take to treat patient
      4. If service multiset is not empty then it will check to see if the patients in the service multiset has been treated in the number of minutes that was chosen for them.
      5. Records patient information into a vector of residents
      6. If they were treated then it pops them out of the multiset
      7. Also adds 1 to number served and calculates total time for the patient
   3. getOpen() for nurses and doctors
      1. Boolean that will check if the nurse can treat the patient by seeing if they are open or not
   4. changeOpen() for nurses and doctors
      1. This method will just change the Boolean isOpen element to either true or false
   5. set\_service\_time(Patient\* patient, int clock) for nurses
      1. If patient priority number is less than 10 generate a number between 1 – 10 to represent service time
      2. If patient priority number is greater than 10 return because only doctors can treat them
   6. set\_service\_time(Patient\* patient,int clock) for Doctor
      1. If patient priority number is less than 20 generate a number between 1 – 20 because doctors take 1 – 20 minutes to treat patients
   7. menu()
      1. Displays a menu with different options after simulation finishes
      2. Takes user input and reacts accordingly to the input
   8. next\_int(int n)
      1. Genereates a random number between 1 and n
      2. Returns that number
2. Hospital Data Storage Description:
   1. For the data storage of the hospital I was thinking of using a multiset to hold the patients that are being treated already. A multi set will work perfectly because we can access elements within the multiset, and we don’t have to erase data out of the set-in order like a queue. Also, once a multiset element is erased it means that 1 patient has been treated so it increments n integer num\_served by 1 each time an element in the multiset is erased. Another data storage that is being used are vectors. Vectors are used so that we can store the resident names from a file into the program. Then we will use those names to create patient instances and store those instances in another vector so we can have access to the information within the instances after the simulation is over.