Jake Tantorski

Professor Arias

Software Dev I

5 April, 2108

Marist Hospital

**Abstract**

In my project, I will creating be database for a hospital check-in desk. I would like the project to have the ability to do simple tasks that would be done by a check in area. It will have the ability to take the name of the patient, record what disease they have at the time and what doctor they are assigned to in the hospital. It will be able to check people in and out of the hospital that way you can keep track of who is in what room. Overall the system should be able to keep track of what doctor the patients have had assigned to them. The user has the option to pick between five different options to check certain aspects of the hospital to make sure that no doctor has to many patients or doesn't have the wrong patient.

**Introduction**

The reason for making this system was to ease the ability for patients to come and go from the hospital. By adding patients to their the needed doctor it saves so much time. Most of my family works in the hospital and I would always here how difficult it is to know whose patient is whose. When it comes to hospitals the easier it is to identify and treat a patient the better. I have always wanted to make something that can help and place where people constantly are asking for help. The system should handle the basic needs and allow for easy access from the user interface. The system will cause the user to know what they want more quickly. With less confusion in the

hospital the reliability of the hospital greatly increases. By saving time the doctors will be able to focus their energy on the patients themselves rather than if they are the right patient or no. By walking into the room and knowing everything beforehand can help the doctor figure out the diagnosis for the patient at a much more rapid pace.

**DSD**

In the Marist Hospital System there are four main classes in which it runs. The way that the system begins is in the Project class. This class is where all of the other subclasses are called and execute depending if called on. In the Project class I set up the data hospital using the Hospital class that I had created.From there the choice variable is set to 0. The reasoning behind this is because the system uses a switch statement to decide what option you want to choose from. Following the choice variable the doctors of the hospital are added. By using the Doctor class I was able to set the name and the speciality of the doctor. The speciality of the doctor is important to matching the doctor with the patient. Each doctor has its own variable to make them unique and findable. The next two statements are output statements are for the introduction to the hospital which shows up first when the program begins to run. After that statement, the program outputs the statement that shows the different options for the user to use. I used the \n function before each number to make the screen cleaner and easier to read. Following that, I prompt an input statement for the variable input. With this input the next line reads what number the user inputs and executes the case that the number corresponds with. This changes the value in the choice variable from 0 to the next integer typed into the console. The switch statement follows that. The statement takes the input of the variable choice to determine what case to use. The first case is for the hospital to add the doctors to the hospital. Without the doctors there would be no

one to be assigned to the patients. The second case allows you have the hospital show you who the doctors are that are currently at the hospital. This case calls the showDoctors function which displays the them on the console. The third case is where the system adds all the patients to the hospital for the day. The system user has to enter the information about the patient to add them to the system. After they are all added the system uses the showPatients function to display them to the console. The fourth cases is where the doctors are assigned to their patients. Finally, the fifth case just displays the the doctor and who their patients are. All five cases provide information that the doctors and the hospital can use. In the hospital class, there are two array lists; one for Doctors and one for Patients. Also, a string variable hospitalName is a data field at the top. After that, there are many void methods that add patients and doctors. Also, the system has Hospital(String name){this.hospitalName=name;} which allows the name of the hospital to be accessed. Other methods include showing both lists by just returning the lists of each. The biggest method of this class is the assignDoctor() method. Here the system goes through both lists and uses the .equals() method to determine which doctor will take which patient dependent on their disease. Once the disease matches the doctor's specialty the patient is added to that doctor's patient list. In the Doctor class, there two string variables that hold the name and speciality of the doctor. The class is mostly for getting the name and list of the doctor. It also has the toString method that prints out both variables of the class. The last case is the Patient class. This class has any private variables such as: name, age, gender and disease. This will help determine who the right doctor for the patient is. A constructor is built for the Patient class where all the variables are the parameters of itself. The only functions of the class are the toString method again and the getter for the disease variable that way the system can access it and match

it with the doctor. The system has many classes to eventually distributed the information in an organized fashion.

**Requirements**

The problem that the system is addressing is the problem of doctors and patients not knowing who they are paired with. In a time of need people want to know as much information as they can about people they are going to help. This allows the doctor to already have the diagnosis on the patient and can prescribe a treatment before they even walk in the door. Knowing basic information can also calm the patient. If the doctor comes in with confidence due to the fact that he already knows a lot about how he is going to treat can provide the patient with a sort of comfort. Another problem that this system is solving is the ability to add patients to doctors list on the go. If the doctor can see his list while using a mobile device this will increase his productivity. By having the system sort out of the information the hospital will be able to understand who needs to do what and to whom they need to do it to. Due to the fact that hospitals can be so hectic. A hospital is a place where big decisions are made and it is vital that the doctors know who they are helping.

**Literary Survey**

There has been other work done in this area to address this problem yet the ability to patients while still in the hospital seems to be a distant reality. There has been systems done like this in hotels and it had worked well enough for them to continue to use it. Pen and paper are becoming more and more obsolete, more individuals rely on their computers and tablets to keep track of what they are doing during the day. The system would be an easier way to divide up the work of the giant system in the hospital. Other systems have been created in hospitals but have been

shown to be less effective due to the speed of the system. With a more simplified version it will be easier for the doctor to know the essentials and have the data in certain categories.

**User Manual**

The system should be used by the hospital and can be viewed by the patient. The whole goal is to make your own data easily accessible. The doctor should import the medicine that the patient is taking at the moment into the system. The system then will sum up all of the patients expenses into a bill which they can pay before they even leave. To use this system, the user must choose a number one through 5 to select what action they want the program to take. Depending on the input, different outputs will be shown to the user with the information they seek. By using option one, the user can add a new doctor to the doctor list. With option two, the user can see all of the doctors that have been added to the hospital. The third allows patients to be added to the hospital with a diagnosis. The fourth will allow a doctor to be assigned to a patient if they match up. Lastly, the fifth option will contain all of the patients for a particular doctor to show who they are treating. Each of these options will allow the user to collect the information they need about anyone who is in the hospital at that time.

**Conclusion**

The goals of the system are to make patients at the hospital more comfortable and ready for the doctor to take care of the specific need of that person. With the system in place many doctors and patients alike should find the time at the hospital more convenient to them. Having a system to tell you the only information you need for a certain person will maximize efficiency and minimize time to perform task. The earlier people know what they need to do they can do it. Knowing that a hospital can function more properly shows that the use of this system is needed.

**UMLs**

**Doctor**

---

- doctorName : String
- doctorSpeciality: String
- doctorStatus: String

---

Doctor(c : String, cc : String)
+ getDoctorName() : String
+ getDoctorPatientList(): ArrayList
+ addPatientToDoctor(o : ArrayList) : void
+getDoctorSpeciality(): String
+toString(): String

**Hospital**

---

hospitalName: String

---

Hospital(name : String)
+ addPatient(o : ArrayList) : void
+ addDoctor(o : ArrayList ): void
+ showDoctors() : ArrayList
+ showPatients() : ArrayList
+assignDoctor(): void

**Patient**

---

- patientName: String
- patientAge: int
- patientGender: String
- disease: String

---

Hospital( patientName: String, patientAge: int, patientGender: String, disease: String, )
+ getDisease() : String
+ toString() : String