COS221 Fundamental Data Structures

##### Project Specification

##### Student: Nikolay Zahariev

# HOSPITAL MANAGEMENT SOFTWARE

*Analysis:*

The program assists hospital staff in creating a database to store information on patients and the kinds of illnesses they are suffering from (or diseases, in this case there are three: cancer, bronchopneumonia, and malaria). The user will be able to both input info on new patience that suffer from one of those conditions or will be able to search for already sighed-in patience. Furthermore, users can input data (name, ID, specialty) on available doctors. All the data the user inputs will be saved onto a text file. The main function will be responsible for the input and search of information on patience or doctors as well as viewing the entire database and deleting needed entries from the database. Currently the database is limited on the amount of information one can enter with regards to a given patient or doctor, but it is sufficient for framework on which a more complex program can be constructed.

*Program design:*

The different classes in the program will mostly represent the three different conditions the patient can be suffering from.

**Level 1.**

***Patient info***: Contains data on patient (name, age, id, etc.), function responsible for searching for a specific patient, function that deletes a given patient record.

**Level 2.**

***Cancer*** (inherits " ***Patient*** "): Contains function which allows user to enter what treatment the patient requires

***Bronchopneumonia*** (inherits " ***Patient*** "): Contains function which allows user to enter what treatment the patient requires

***Malaria*** (inherits " ***Patient*** "): Contains function which allows user to enter what treatment the patient requires

**Level 3.**

***Treatable cancer*** (inherits "***Cancer*** "): "): Contains function which allows user to enter what treatment the patient requires

***Untreatable cancer*** (inherits "***Cancer*** "): Contains function which allows user to enter what treatment the patient requires

There will also be another class that will contain info on the hospital ward: doctors available and rooms reserved for malaria patients and for bronchopneumonia patients.

*Analysis and design for* int valid\_input\_int()*:*

This function is used to validate user input. This function is called when you want to check if the user entered only integers. If he did not then the function will continue to ask for new input until the user enters a correct value. It accepts user input and uses a loop to check if it matches the declared type of the variable, in this case an integer. If it does not then the user is prompted to reenter data. When the user enters the correct type of input we exit the loop and return the correct user input.

*Analysis and design for* string getLetters():

This function is used to validate user input. This function is called when you want to check if the user entered only integers. If he did not then the function will continue to ask for new input until the user enters a correct value. It accepts user input and uses a loop to check if it matches the declared type of the variable, in this case a string that must contain only characters in the range of [a-z] and [A-Z]. If it does not then the user is prompted to reenter data. When the user enters the correct type of input we exit the loop and return the correct user input.

*Analysis and design for* class patient*:*

It contains three functions:

void patient\_data();

void entry\_del (string) ;

void search(int);

The first method: **patient\_data()** is responsible for getting the user input on the name of the patient, his age, id, room number, and doctor responsible for overseeing the treatment. After this info is inputted, this functions copies it onto a text file named **Patient.txt.**

The second function: void entry\_del(string) is responsible for deleting the desired by the user patient or doctor record. It takes user input in the form of the ID of the doctor or patient we want to remove. We do this by first creating a new file called **temp.txt**. We then use a loop that traverses **Patient.txt** and copies all data that is not the one we want to delete to the new **temp.txt** file. After that is done we delete the original **Patient.txt** file and rename the temp file as **Patient**.

The last function void search(int) is responsible for traversing the text file and based on integer inputted by the user finding the desired patient, doctor, or room info. It does this with a loop and the **getline()** procedure. After the desired record is found we display it for the user to see.

The patient class contains two virtual functions that are used in its derived classes.

*Analysis and design for* class Ward\_info

This class contains two function:

void ward\_data\_doctors();

void ward\_data\_rooms();

The first one is responsible for accepting user input about doctors: their name, ID, and specialty. The user is given a choice of how many such records he wishes to make and all the inputted data is checked for its validity. The second method has the same function with the only difference is that here the user enters data on the different patient rooms available. All the information entered by the user and after checking if it is valid is entered in the **Patient.txt** file.

*Analysis and design for* class Cancer:public patient:

This class is basically empty, containing only two virtual functions that are later used in its derived classes.

*Analysis and design for* class Cancer\_Untreatable:public Cancer:

This class has one function: void input\_data\_unreatable(). It contains a switch function that allows the user to choose the type of treatment the patient suffering from this affliction needs. After it receives the user input, the function then copies it onto the text file.

*Analysis and design for* class Cancer\_Treatable:public Cancer:

This class has one function: void input\_data\_treatable().It contains a switch function that allows the user to choose the type of treatment the patient suffering from this affliction needs. After it receives the user input, the function then copies it onto the text file.

*Analysis and design for* class Broncho:public patient:

This class has one function: void input\_data\_broncho().It contains a switch function that allows the user to choose the type of treatment the patient suffering from this affliction needs. After it receives the user input, the function then copies it onto the text file.

*Analysis and design for* class Malaria:public patient:

This class has one function: void input\_data\_malaria().It contains a switch function that allows the user to choose the type of treatment the patient suffering from this affliction needs. After it receives the user input, the function then copies it onto the text file.

*Analysis and design for* void display\_all():

This function is responsible for displaying the entire contents of the text file Patients whenever the user calls it. It does this with the help of a loop that traverses the entire file and then with a **getline()** function we output the file content one line at a time with each loop repetition.

*Analysis and design for* void del\_whole():

This function is responsible for deleting the entire contents of the text file Patients whenever the user calls it. It does this by first closing the file, deleting it, and finally creating a new empty file with the same name. Before deleting the file, the program will first query the user whether he is sure of his decision to delete all his data. If the user inputs “Y” then the program will proceed with the deletion. If, however, the user enters any other character then the method will terminate and send the user back to the main menu without touching the database file.

*Analysis and design for* void main():

First thing this function does is to open the **file Patient.txt** where all data will be stored. After that we declare all the class variables and pointers to the addresses of these class variables (needed for the virtual functions) and also declare all other variables such as counters for loops, etc. We begin by displaying the main menu to the user. It contains seven options, however, the user is recommended to first use the first two: creating a doctor entry, and creating a room entry. We do this because fields from both of these records must later be added to the patient records and it will be illogical to, for example, say a patient resides in a room that does not yet exist according to the database. The entire menu is inside a loop that allows the user to return to the menu multiple times as long as he does not enter **‘E’** when prompted. The menu consists of a switch statement that calls the function that will accomplish the user’s choice. For the first case of the switch however there is another, nested, switch which makes the user choose what type of cancer the patient has. It then calls the appropriate functions either from class Cancer\_Treatable or class Cancer\_Untreatable.

Another interesting point is that for the fifth case of the switch (the one which deals with deleting records) we first need to close **Patient.txt** otherwise we would not be able to delete it because of permission issues. After we call the delete function we open the file again, but now it has one entry less due to us deleting the wanted patient data.

Finally at the end of the file we close the text file.