Student Number: 26003406
Unit Code: FIT3077
Name: Jeremiah Wu

Design Rationale Assignment 3

Assignment 3 was approached in a similar design as Assignment 2, with some extensions to fulfil the additional requirements. I focused on a few key patterns and principles for the implementation of the application. Namely, the MVC pattern (Model-View-Controller) as well as the Observer Pattern.

The MVC Pattern [1] is split into 3 separate components. Each of these components is responsible for specific functions of the software such as decoupling JSON data and storing, displaying, and updating of data [2]. The TKinter library was brought forward from assignment 2 as the base UI design as I have now been familiarised with its syntax and limitations.
The uncertainties regarding the appearance of the application had been cleared and the team were able build upon the UI from the previous iteration. The changes were easier to implement as the all the UI elements were in the View class and did not affect the functionality of the rest of the program. Implementing the MVC has its benefits e.g. Easier to modify and test certain functions. However, it brought some complexity. The adherence of the separation of these components were necessary. The con of having MVC as a design pattern was that when it came down to the debugging of problems, one often had to modify more than 1 class as they all communicate with each other.

The second design pattern that was utilised was the Observer Pattern. This pattern kept in this iteration as one-to-many dependencies were abundant. As our program was handling many patients that were seen by a practitioner, we needed this pattern to notify the observer and update the information in our program when the state of our data was changed. [3] The fact that the program does not have no reference each individual patient to invoke an update of the system allows this system to be extended for future purposes as well as increases its efficiency.

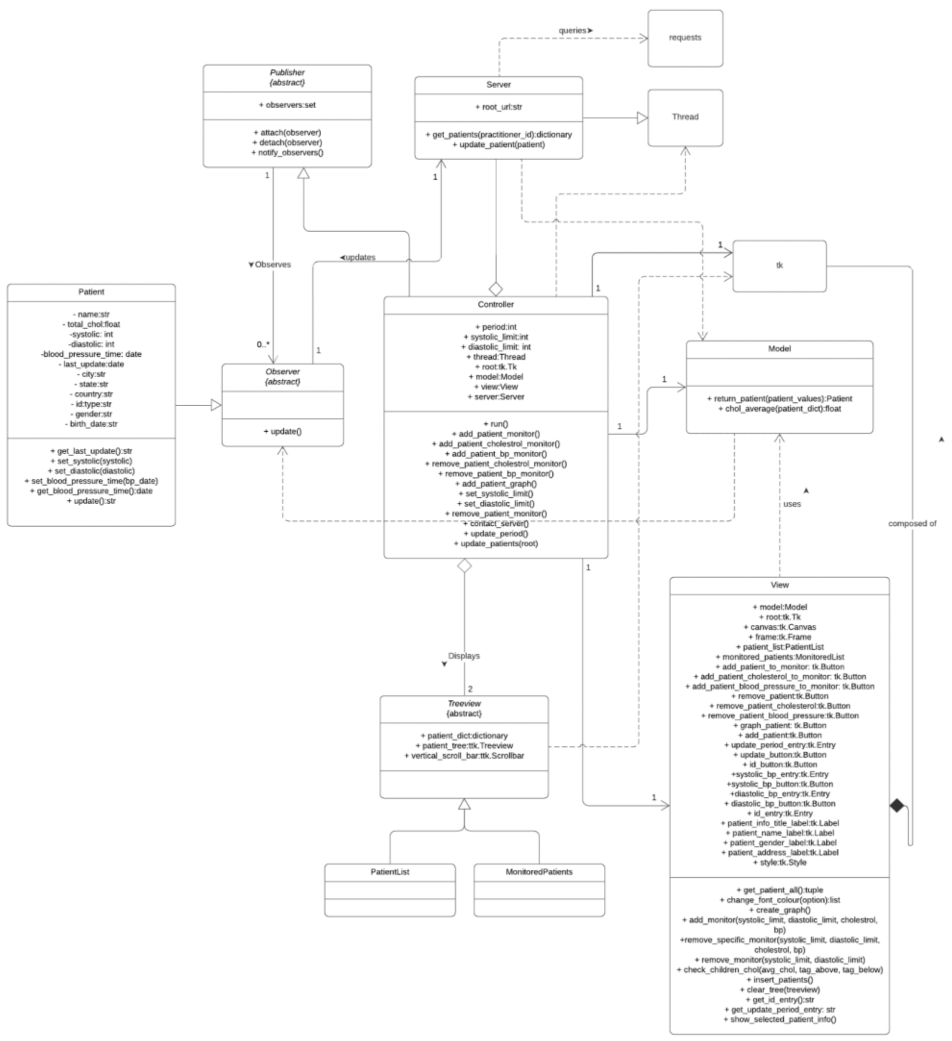
There were 2 design principles that implemented in the program was the LSP (Liskov Substitution Principle) and the Open/Closed Principle. LSP states that for a class S to be a true subtype of class T, then S must conform to T [4]. This is evident throughout our program where the Patient and Monitor List classes are subtypes of the TreeView Class. This was done to increase the reusability of the classes that implement TreeView.

Open/Closed Principle was also applied to the TreeView class and the Observer Pattern. The TreeView class now has implementation for placing inside the UI removed, this was done so that other objects can easily reuse the TreeView class without having to worry about UI. [4]

Some assumptions remained, and new ones were formed in this iteration of the program. One assumption was that it is assumed that when the server is contacted, the patients that will be necessary to update are the patients that currently have cholesterol report or a blood pressure reading. No new patients with cholesterol reports, or existing patients who didn't get a report conducted will be considered and fetched. However, when a new practitioner uses his ID to retrieve the patient list, the system will retrieve all patients, including the new and existing ones and only then will it be considered.

The program was not modified, but rather new methods were added to extend the functionalities of the system.

Student Number: 26003406
Unit Code: FIT3077
Name: Jeremiah Wu


queries
requests
Publisher {abstract}
+ observers:set
+ attach(observer)
+ detach(observer)
+ notify_observers()
Server
+ root_url:str
+ get_patients(practitioner_id):dictionary
+ update_patient(patient)
Thread
1
1
Observes
updates
1
1
tk
Patient
- name:str
- total_chol:float
-systolic: int
-diastolic: int
-blood_pressure_time: date
- last_update:date
- city:str
- state:str
- country:str
- id:type:str
- gender:str
- birth_date:str
+ get_last_update():str
+ set_systolic(systolic)
+ set_diastolic(diastolic)
+ set_blood_pressure_time(bp_date)
+ get_blood_pressure_time():date
+ update():str
0..*
1
Observer {abstract}
+ update()
Controller
+ period:int
+ systolic_limit:int
+ diastolic_limit: int
+ thread:Thread
+ root:tk.Tk
+ model:Model
+ view:View
+ server:Server
+ run()
+ add_patient_monitor()
+ add_patient_cholestrol_monitor()
+ add_patient_bp_monitor()
+ remove_patient_cholestrol_monitor()
+ remove_patient_bp_monitor()
+ add_patient_graph()
+ set_systolic_limit()
+ set_diastolic_limit()
+ remove_patient_monitor()
+ contact_server()
+ update_period()
+ update_patients(root)
1
1
Model
+ return_patient(patient_values):Patient
+ chol_average(patient_dict):float
uses
composed of
1
Displays
2
Treeview {abstract}
+ patient_dict:dictionary
+ patient_tree:ttk.Treeview
+ vertical_scroll_bar:ttk.Scrollbar
PatientList
MonitoredPatients
1
View
+ model:Model
+ root:tk.Tk
+ canvas:tk.Canvas
+ frame:tk.Frame
+ patient_list:PatientList
+ monitored_patients:MonitoredList
+ add_patient_to_monitor: tk.Button
+ add_patient_cholesterol_to_monitor: tk.Button
+ add_patient_blood_pressure_to_monitor: tk.Button
+ remove_patient:tk.Button
+ remove_patient_cholesterol:tk.Button
+ remove_patient_blood_pressure:tk.Button
+ graph_patient: tk.Button
+ add_patient:tk.Button
+ update_period_entry:tk.Entry
+ update_button:tk.Button
+ id_button:tk.Button
+systolic_bp_entry:tk.Entry
+systolic_bp_button:tk.Button
+diastolic_bp_entry:tk.Entry
+ diastolic_bp_button:tk.Button
+ id_entry:tk.Entry
+ patient_info_title_label:tk.Label
+ patient_name_label:tk.Label
+ patient_gender_label:tk.Label
+ patient_address_label:tk.Label
+ style:tk.Style
+ get_patient_all():tuple
+ change_font_colour(option):list
+ create_graph()
+ add_monitor(systolic_limit, diastolic_limit, cholestrol, bp)
+remove_specific_monitor(systolic_limit, diastolic_limit, cholestrol, bp)
+ remove_monitor(systolic_limit, diastolic_limit)
+ check_children_chol(avg_chol, tag_above, tag_below)
+ insert_patients()
+ clear_tree(treeview)
+ get_id_entry():str
+ get_update_period_entry: str
+ show_selected_patient_info()

Student Number: 26003406
Unit Code: FIT3077
Name: Jeremiah Wu

References:

1. Rj46, "*Simple Example of MVC (Model View Controller) Design Pattern for Abstraction*", Code Project, April 2008. Available:https://www.codeproject.com/Articles/25057/Simple-Example-of-MVC-Model-View-Controller-Design
2. Monash University. (2020). The Model-View-Controller Architectural Pattern. [Online] Available:https://lms.monash.edu/pluginfile.php/10698523/mod_resource/content/1/MVC.pdf
3. Monash University. (2020). Design Patterns 1. [Online] Available:https://lms.monash.edu/pluginfile.php/10580885/mod_resource/content/1/Design_Patterns_1.pdf
4. Monash University. (2020). Principles of Object-Oriented Design 1. [Online] Available:https://lms.monash.edu/pluginfile.php/10536249/mod_resource/content/2/OOP1.pdf