OOP Coursework Report S/N:2316590

Section 1: Summary of My Program

My Java web application I have designed has fulfilled requirements 1 to 7, 9, and 10.

You can change the CSV file you want to read by changing the file path in the model.java file. In public Model() throws IOException function you can change your file path on the line dataloader.loadData(filePath) (line 21), where filePath is the directory of the CSV file in a String format. (Eg. filePath = `"data/patients100.csv"`

After you run Maven on the file, you need to run http://localhost:8080/ in your local browser to load the webpage application.

Once loaded, on the main page, the website provides 5 possible things you can do: view the patient List, search patients, view the custom patient List, view graphs, and create a JSON file. If you click to view the patient List, you will be given the list of hyperlinks of the patient's ID coupled with their surname. When you click one of these hyperlinks, it will direct you to a patient data page of the user with the ID you clicked on, where all the information about the patient (from ID to ZIP) will be displayed.

On the main page, if you click on search, it will redirect you to a search page, where you can enter any keywords. If any of the data contains the keyword you have typed, the search result will show the ID that owns the data that contains the keyword, followed by a list of data that contains its keyword. For example, if I inputted "u" as the keyword, the search result for each ID will look like this format, be861ed0-39b3-4cd7-8b75-c56a62894868 - [Granville Massachusetts US, 205 Rutherford Hollow, Massachusetts]. If the ID does not have any information that contains the keyword, the ID will not be shown in the search result.

If you click on Custom Patient List, it directs you to a page where you can view 5 types of modified lists. You can look at a list of dead patients, alive patients, male patients, and female patients. You can also look at the patient list, which displays the patients in their age order(oldest to youngest). If you click on graphs on the main page, there are three types of graphs you can view. The three graphs are the age distribution bar graph, gender proportion pie chart, and lastly marital status proportion pie chart. Lastly, there is the Create JSON file button on the main page. When you click this hyperlink, a JSON copy of all the patient data will be downloaded to your computer.

Section 2: Describe and evaluate your design and programming process

The final program that I have designed and the design process have both pros and cons.

I will talk about the cons first. Firstly, I believe that the program I have implemented lacked design. I have used a very simple design when displaying patient lists or patient data. I could have represented the data in a table format, instead of just displaying a list of IDs and other data in a line format. For example, I was not sure how to display the hashtable I output in my getPatientList() function into a table format, and didn't know what CSS style I had to implement. Thus I lacked a lot in the design process.

Secondly, the program I have developed has not used abstract classes and has poor cohesion. I was unclear when it came to the concept of abstract classes, and it failed multiple times whenever I tried using them, and hence I decided to avoid using abstract classes. Regarding cohesion, some of my classes like JsonWriter and DataLoader classes had high cohesion, as their sole purpose was to write out JSON files and load the CSV files. However, classes such as the Model class, showed a very low cohesion as the model class served so many purposes such as getPatientData(String id), getPatientList(), searchFor(String keyword), patientListbyGender(String gender), and a lot more. Thus, my classes such as the model showed low cohesion, which I could have better improved on.

There were also positives. Firstly, during the design process, I followed the Model View Controller(MVC) design pattern. For example, when I developed the web application for getting age distribution bar graphs, I created a function in my Model class called getAgeDistribution() (model component) which returns an ArrayList of integers that represents the number of people falling into each age category. Next, I developed the jsp file called ageDistribution.jsp, which is my view component, and used chart.js to display the ArrayList I output from my model component, in a bar graph format. The AgeDistributionServlet.java works as the controller component which passes on the age distribution ArrayList from the model component to the jsp file.

Secondly, the search function in my program is developed well. I made sure that for my searchFor() function, I returned a Hashtable<String, ArrayList<String>>, which allowed me to display all the relevant data that contains the keyword, instead of just displaying the ID. Such a view component allows the users to know why specific IDs are returned as a search result, by showing all the relevant data that are related to the keyword the user inputs.

Overall, it was a good experience to develop the Java webpage. Although much more could indeed have been done, such as finishing up requirement 8 or using more abstract and cohesive classes, I believe that the Java web application I have developed is a fine and usable webpage to read CSV files and use the data in the CSV files. It was a good opportunity for me to learn how to use JSP and HTML files and to try out general software development.