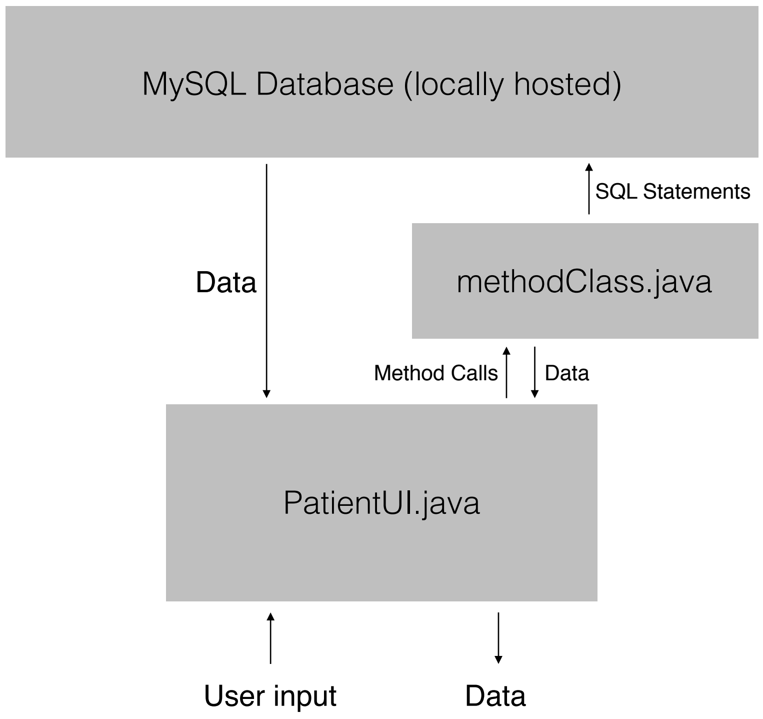
John Z Powell
Patient Management System.
Database and Method Class Documentation

Application Diagram:



Design Specification:

The application is designed to take user input in the form of patient data, store that information in a locally hosted database and return queries at the users command.

The database is a MySQL database hosted locally on the users (me) machine. It consisted of one database, titled "Patient_Queue" and one table, titled "Pat_List". Within Pat_List there are 11 columns:

- Patient_ID
- Pat_FName
- Pat_MName
- Pat_LName
- Pat_StreetAddrs
- Pat_City
- Pat_State
- Pat_Height
- Pat_Weight
- Pat_DOB
- Pat_Priority

All columns are of the type varchar(). This was done simply because taking information from the GUI is easier in the String format.

The Method Class houses all of the callable methods that eat GUI uses to communicate with the Database. Here you can find many different methods to manipulate the database. Not all methods were included in the final version of the GUI, but they can be easily implemented in future versions/updates of the application.

The GUI, documented elsewhere, communicates to the database by calling specific methods when buttons are pressed and taking user input in text fields and passing that information through the methodClass to the database.

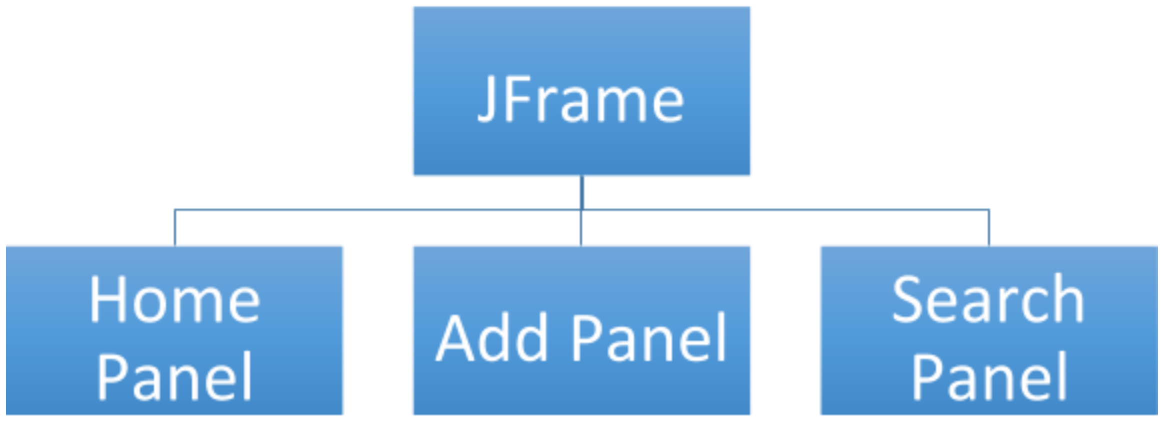
QA testing was done on both the GUI level and the command-line level. During the creation of the method class, command line database entries and SQL statements were used to verify the functionality of the method class. After the GUI was created, it was used to ensure that the GUI was properly calling the methods and the user input was going to the correct area the database.

Example non-GUI database entry method for methodClass testing:

```
methodClass.addEntry("Julia", "Dean", "Powell", "1714 Old Cartersville RD", "Dallas", "GA", "5-03", "120", "02141995", "5");
```

Currently, the only deployable process is for MacOS. Once a user has MySQL installed on their machine, I have written an AppleScript to for Terminal to create a database with the appropriate information, i.e. matching database, table, and column names/datatypes. After this, the user would run the patientUI.java file in the Priority_Queue package, which includes the methodClass.java file. Running the patientUI,java file will bring up the GUI and grant the user access to all functions of the app.

Graphical User Interface:



Home Panel:

- This panel is main window that opens when the program runs.
- It displays the current running patient queue using a Jtable.
- It contains two buttons, one to add new patients and one to search patients that are already in the system.

Add Patient Panel:

- This panel is displayed after the add patient button is clicked from the home panel.
- It contains the basic patient information textfields that need to be entered to add the patient.
- It also contains a submit button that adds the patient into the database and will reflect in the current patient queue in the home panel.
- The necessary information to add a patient is label above each respective textfield; for example, the labels are first name, last name, address, height, weight, and date of birth.
- There is also a back to home button that when clicked the window is changed from add panel to home panel.
- Submit button and home button are activated through the ActionListeners.

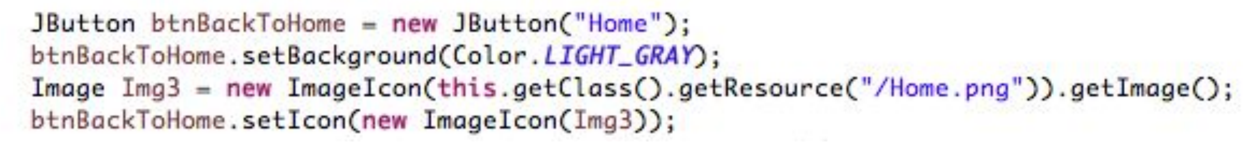
Search Patient Panel:

- In this panel the user can search for a specific patient by either entering the patient's personal information such as name and date of birth or the user can enter the desired patient's patient ID.

- The panel contains a search button that when clicked searches the database for the patient and displays the result of the search inside the result field Jtable.
- There is also a back to home button that when clicked the window is changed from add panel to home panel.
- If the patient entered into the search entry is not found in the database, then another dialog box will pop up to declare that the patient was not found.

Logo:

- We used various logos to make the appearance more professional.
- First we searched for the desired logo such as a home logo for the back to home button.
- Then we saved the logo in a file inside the package of the GUI program.
- Inside the hard code, we set the image of the logo to the desired location. For example, the back to home button logo was set as shown below.



```
JButton btnBackToHome = new JButton("Home");
btnBackToHome.setBackground(Color.LIGHT_GRAY);
Image Img3 = new ImageIcon(this.getClass().getResource("/Home.png")).getImage();
btnBackToHome.setIcon(new ImageIcon(Img3));
```

Layout:

- We used a card layout for the JFrame so that each individual panel would act as cards; thus, enforcing only one panel to be displayed at specific times.
- In addition, we applied an absolute layout to each individual panel.

First Process:

- Deciding what would be the best implementation of the priority queue, whether to use stack, linked list, queue, or array lists.
- The size of the database influenced our decision of implementing a stack using arrays to sort priority of the patient's injury.

Precondition:

- Use the information based off the database.

Post-condition:

- Orders the priority of the patient's injury despite their order "in line".

Patient Class:

- Lists information needed to input into the database.
- Includes standard patient information such as name, address, phone number, injury, etc.
- Assigns what would be considered top priority to least using numeric 1 being the top priority to numeric 3 being the least.
- Used a two-dimensional array to order the priority and the various injuries.
- As good practice, used abstraction for patients' personal information.

Patient Test:

- Assigns the patient based off his or her priority.
- Since queue is implemented, we must include a sort (), peek (), insert (), and delete ()
    - sort () – compares two patients by their priority number. If patient *i* is less than patient *x* then move to the right, otherwise it is shifted to the front of the list.
    - Insert () – adds patients to the list based on their order
    - Peek () – checks who is the top priority
    - Print () – prints order of the patient starting from the top priority.

Installation:

- Patient and Priority Queue class are public classes through Eclipse.

Testing:

- Originally had to hardcode the patient class.
- Adjusted to work coincide with the database.
- Checking breakpoints in periodically with one patient at a time.