Java-Based Hospital Management System

Khalid metwally

Documentation

The main classes used in this project are 3 classes. The doctor class, staff class and the patient class(the program is based on them). Then we have the following:

1. AddDoctor.java(controller+view): This class provides the GUI for adding a doctor to the hospital information system. It extends JFrame and implements ActionListener. It allows the user to enter the doctor's ID, name, specialization, and floor, and then adds the doctor to the system when the "Add Doctor" button is clicked.
2. AddPatient.java(controller + view): This class provides the GUI for adding a patient to the hospital information system. It extends JFrame and implements ActionListener. It allows the user to enter the patient's ID, name, illness, and floor, and then adds the patient to the system when the "Add Patient" button is clicked.
3. AddSpecialization.java(controller+ view): This class provides the GUI for adding a specialization to the hospital information system. It extends JFrame and implements ActionListener. It allows the user to enter the specialization name, floor number, and capacity, and then adds the specialization to the system when the "Add Specialization" button is clicked.
4. AddVitalSigns.java(controller + view): This class provides the GUI for adding vital signs to the hospital information system. It extends JFrame and implements ActionListener. It allows the user to select a patient, enter vital sign values such as low and high blood pressure, breathing rate, pulse rate, and temperature, and then adds the vital signs to the system when the "Add Vital Signs" button is clicked.
5. ChangeDoctorData.java(controller + view): This class provides the GUI for

changing doctor data in the hospital information system. It extends JFrame and implements ActionListener. It allows the user to enter a new specialization and floor number for a doctor and update the data when the "Update Data" button is clicked.

6. CheckPatientData.java(view+controller): This class creates the GUI for doctors to check patient data in the hospital information system. It extends JFrame and implements ActionListener. It allows the doctor to enter a patient ID and retrieve the corresponding patient data by clicking the "Check Data" button.
7. Doctor.java(model): This class represents a Doctor object in the hospital information system. It encapsulates the properties and behaviors of a doctor, such as ID, name, password, specialization, floor, status, and first login status.
8. DoctorScreen.java(view) : The DoctorScreen class represents the main GUI screen for the doctor in the Hospital Information System. It extends the JFrame class and implements the ActionListener interface to handle events.
9. Floordata.java (view + controller) : Creates a graphical user interface (GUI) for a doctor to check floor data in a hospital information system. 10. LoginScreen.java (view) : This is the main class for Hospital Information System It will provide a login screen to staff/doctors-.
11. Patient.java(model): The Patient class represents the patient data in the Hospital Information System. It encapsulates the attributes of a patient, such as their ID, name, illness, and assigned floor.
12. PerformStatistics.java(controller+view) : The PerformStatistics class creates a GUI window to display statistics related to vital signs of patients in the Hospital Information System.

13. Specialization.java(model): This class holds the specialization data, including the name of the specialization, the floor number associated with it, and the capacity (maximum number of patients) of that specialization. Staff.java(model): This class represents a staff member in the hospital information system. It contains properties such as ID, username, password, and status. The status indicates whether the staff account is active or inactive.
14. StaffScreen.java(controller + view): This class creates the GUI for the staff members to perform various operations in the hospital information system. It extends JFrame and implements ActionListener. The GUI allows staff members to add specializations, add patients, add vital signs, perform statistics, and add doctors. Each operation is triggered by clicking on the respective button or menu item.
15. Utility.java(model): This class provides methods for various read and write operations. It handles the loading and saving of data for staff, doctors, specializations, patients, and vital signs from/to corresponding text files. Also, it includes a method to generate a 10-digit random password.
16. VitalSign.java(model): This class represents the vital signs of a patient. It includes attributes such as patient ID, low and high blood pressure, breathing rate, pulse rate, and body temperature.

The GUI components used:

- JFrame
- JTextField
- JPasswordField:
- JTable

- JButton
- JComboBox
- JRadioButton
- ButtonGroup
- JOptionPane
- DefaultTableModel
- JMenuBar
- JMenu
- JMenuItem
- JCheckBox
- JFileChooser
- GroupLayout
- Etcetera

Main features of the program :

For Staff:

- Assign specializations and capacity to floors.
- Add patients with patient details (Name, ID, Illness, floor, designated doctor)
- Enter vital sign readings for a specific patient or for a list of patients from a file.
- Display alarms for vital signs outside the normal range.
- Perform statistics on vital sign readings (min, max, average) and display them in a separate frame.
- Add doctors to the system with doctor details (Name, ID, specialization, floor)
- Generate random passwords for doctors and store them securely.
- (Bonus) Multithreading to allow multiple users to use the system at the same time by using swingworker class.

For Doctors:

- Change their data (Name and specialization)
- Link the doctor to the floor corresponding to their specialization.
- Receive randomly generated password on first login and be able to change it.
- Check patient's data (patient details and vital sign readings etc)
- Check floor data (number of patients, average vital sign readings etc)
- (Bonus) Multithreading to allow multiple users to use the system at the same time by using swingworker class.