<u>User</u>

public boolean isPatient() - Tells whether a user is a patient or not based on the userType enum type.
public boolean isDoctor() - Tells whether a user is a doctor or not based on the userType enum type.
public boolean isNurse() - Tells whether a user is a nurse or not based on the userType enum type.
public boolean isReceptionist() - Tells whether a user is a receptionist or not based on the userType enum type.
public String getUserName() - Return the user's login name.
public String getPassHash() - Return the hashed password for the user.
public String getFullName() - Return the user's full name.
public Date getLastLogin() - Return the last login date.
public Appointment[] appointments - An array of Appointments associated with the current user.
public Report[] reports - An array of Reports associated with the current user.
private userType typeOfUser - The type of user – patient, doctor, nurse, or receptionist.
private String userName - The login name of the user.
private String passHash - The user's hashed password.
private String fullName - The user's full name.
private Date lastLogin - The last time the user logged in. Used to determine whether there are new reports or appointments made since the last login.

The User class is used to hold important information at the client level for the user profile of the logged in user. An enumerator is used and queried with isPatient() or isDoctor(), etc, in order to determine the user type for access to certain use cases. All other data is accessed with getter functions.

The constructor populates the user data from the database with a query to the users table. It populates the reports[] and appointments[] members with separate queries to the database, looking for entries whose patient or doctor (depending on the user type) match the User's login name.

No setters are used, as all data will be populated from the SQL database. No information stored in the User class will ever need to be updated in the same session, so changes in data are carried out at the database level.

<u>Report</u>

public boolean isViewed() - Returns the value of the boolean viewed.
public Date getDateTime() - Gets the time the report was submitted.
public String getDoctor() - Gets the name of the doctor to whom the report is addressed.
public String getPatient() - Gets the name of the patient who submitted the report.
public int getPain() - Returns the value the patient associated with pain, stored as symptoms[0].
public int getDrowsiness() - Returns the value the patient associated with drowsiness, stored as symptoms[1].
public int getNausea() - Returns the value the patient associated with nausea, stored as symptoms[2].
public int getAnxiety() - Returns the value the patient associated with anxiety, stored as

symptoms[3].
public int getDepression() - Returns the value the patient associated with depression, stored as symptoms[4].
public String getComments() - Return comments.
public String getSuggestion() - Getter for practitioner's suggestions.
public void submit() - Submit the report to the database.
public void view() - Set viewed to true in the client object and the database.
public Appointment appointment - An appointment associated with the report.
private boolean viewed - Whether or not the report has been viewed by a doctor.
private Date dateTime - The time the report was submitted.
private String doctor - The name of the doctor who the report is addressed to (the patient's doctor).
private String patient - The name of the patient who submitted the report.
private int[] symptoms - The magnitude of the 5 symptoms reported by the user.
private String comments - Any additional comments the patient needs to include.
private String doctorSuggestion - Response from a healthcare practitioner.

The Report class is used to hold data from the SQL database pertaining to patient-submitted reports. They are created for all reports in the database for the healthcare practitioner or patient who is logged in on user login. Variables and their associated getters are used to avoid making repetitive MySQL queries to populate the reports list.

Appointment

private Date timeMade - The time the appointment was made.
private Date appointmentTime - The scheduled time of the appointment.
private patient String - The name of the patient.
private doctor String - The name of the doctor.
private Report patientReport - The patient report associated with the appointment. The appointment is to address the concerns in the symptom report.
private Appointment followUp - If this appointment is a followup, followUp is the previous appointment.
private String notes - The notes left by the healthcare practitioner.
private boolean authorized - Whether a doctor has authorized the appointment.
private boolean active - Whether the appointment is active or not. It is not active if it has not been authorized or if it has been cancelled.
public void setAppointmentTime(Date) - Set the time and date of the appointment.
public void authorize() - Set authorized to true.
public void setActive(boolean) - Set active to the value in the parameter.
public void setFollowUp(Appointment) - Indicate the previous related appointment.
public void setNotes(String) - Set the notes variable and update notes in the database.
public Date getTimeMade() - Getter for the timeMade variable.
public Date getAppointmentTime() - Getter for the appointmentTime variable.
public String getPatient() - Getter for the name of the patient.
public String getDoctor() - Getter for the name of the doctor.
public Report getPatientReport() - Getter for any patient symptom report associated with the appointment.

public Appointment getFollowUp() - Getter for any previous appointment, if one exists.
public boolean getAuthorized() - Returns whether or not the appointment has been authorized.
public boolean getActive() - Return whether the appointment is active or not.
public String getNotes() - Return notes.

The appointment class holds information for quick access regarding appointments between a patient and a doctor. An appointment MAY but does not need to be a followup of a previous appointment, and MAY but does not need to have an associated symptom report.

The object is populated from the SQL database on creation of a User (login use case) to avoid repetitive SQL queries.

<u>DatabaseController</u>

public ResultSet queryDatabase(String) - query the database based on a String query passed as a parameter. Returns a ResultSet for use by other classes.
private Connection con - Used for holding a connection to the SQL database.
private Statement st - Used internally by Java's Connector/J mySQL interface.
private String url - URL of the SQL database.
private String un - Username for accessing the SQL database.
private String pw - Password for accessing the SQL database.

DatabaseController is used to control the connection and queries to the SQL database and act as an interface between the rest of the client and the database.

The constructor opens a connection to the database, which is used by the queryDatabase() function to populate functional classes. The destructor closes the connection before clearing the object from memory. This class has no visible function to the user, or any direct reference to other classes in the package. The class has no getters or setters, as all fields are constant and are only referenced from within the class.