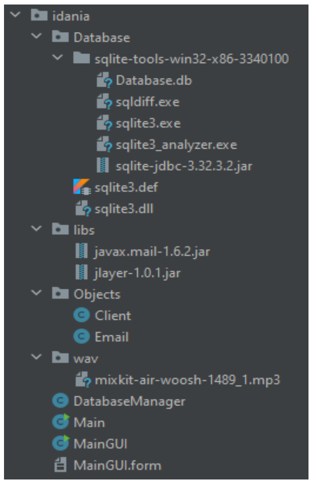
Introduction:
The program performs 3 main functions as requested by my client, keeping track of client information, updating payment and session statuses, and sending emails to multiple clients at a time. Through the use of the netbeans IDE to create my program, a simple and effective graphical user interface was developed making it easy and comfortable for my client to use.

Summary List of All Techniques:

- For loops and while loops
- Sorting and Searching
- SQLite API (application programming interface) for java
- Fetching/Storing data in a database
- ArrayLists, HashMap, Arrays
- External Libraries (JARs - Java Archives)
    - SQLite-JDBC
    - javax.mail
    - Swingx
    - Javazoom
    - Jcalendar
- Error Handling
- OOP (Object Oriented Programming) techniques
    - Encapsulation
    - Abstraction
    - Polymorphism
    - Inheritance
- Conditional Statements
    - Switch Case
    - If/else
- Option pane generation to communicate with the user
- Dynamic searching
- Use of global variables for example clientList (ArrayList)

Structure of The Program



```
idania
  Database
    sqlite-tools-win32-x86-3340100
      Database.db
      sqldiff.exe
      sqlite3.exe
      sqlite3_analyzer.exe
      sqlite-jdbc-3.32.3.2.jar
    sqlite3.def
    sqlite3.dll
  libs
    javax.mail-1.6.2.jar
    jlayer-1.0.1.jar
  Objects
    Client
    Email
  wav
    mixkit-air-woosh-1489_1.mp3
  DatabaseManager
  Main
  MainGUI
  MainGUI.form
```

Client

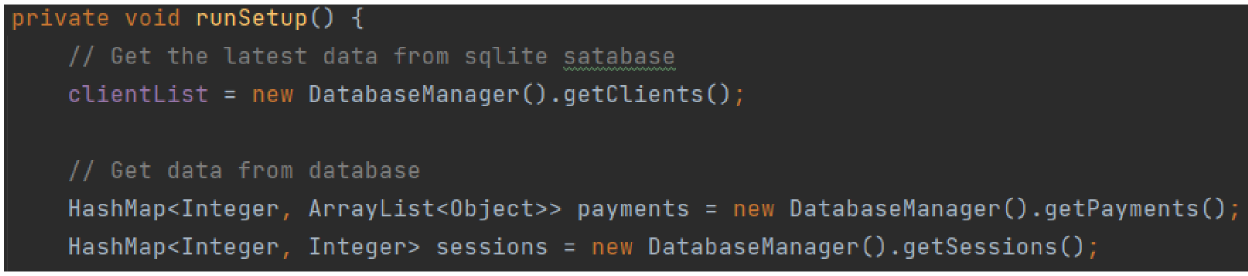
The client object class aids with the use of accessing the attributes of each client. For example, throughout the program, there are various jtables each varying in client information. Thereby, when accessing specific elements from the client class they are easily accessible to each table where applicable.

Data Structures Used:

- ArrayList (Mutable length)
    - Used to store the clients onto the ArrayList. This collection is dynamic and can shrink and grow in size because through the input, clients can be dynamically added and removed from the ArrayList.
- HashMap (Mutable length)
    - Consists of keys that map to certain values. For example, it's convenient to access certain elements using an identifier such as the client ID. The client ID can be used to obtain certain elements such as payment date and payment status.
- Array (Immutable length)
    - A way to contiguously store data of the same type. Because they are contiguous, the iteration of them is relatively fast when dealing with small amounts of data. For example, when removing clients from a jTable, an array of the selected indices is constructed using the getSelectedRows() method from the jTable model object.

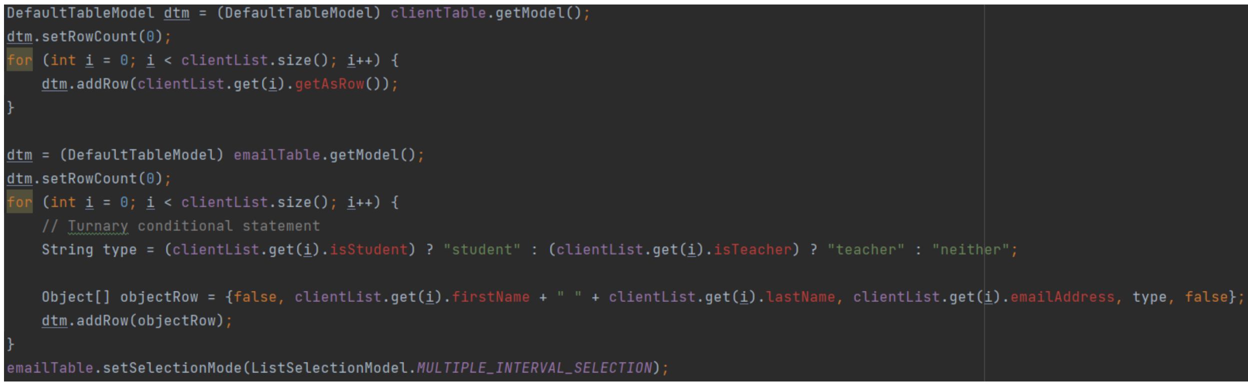
<u>Summary of Programming Techniques:</u>

- Filling all tables with updated values fetched from the DatabaseManager class
  - Fetching latest data



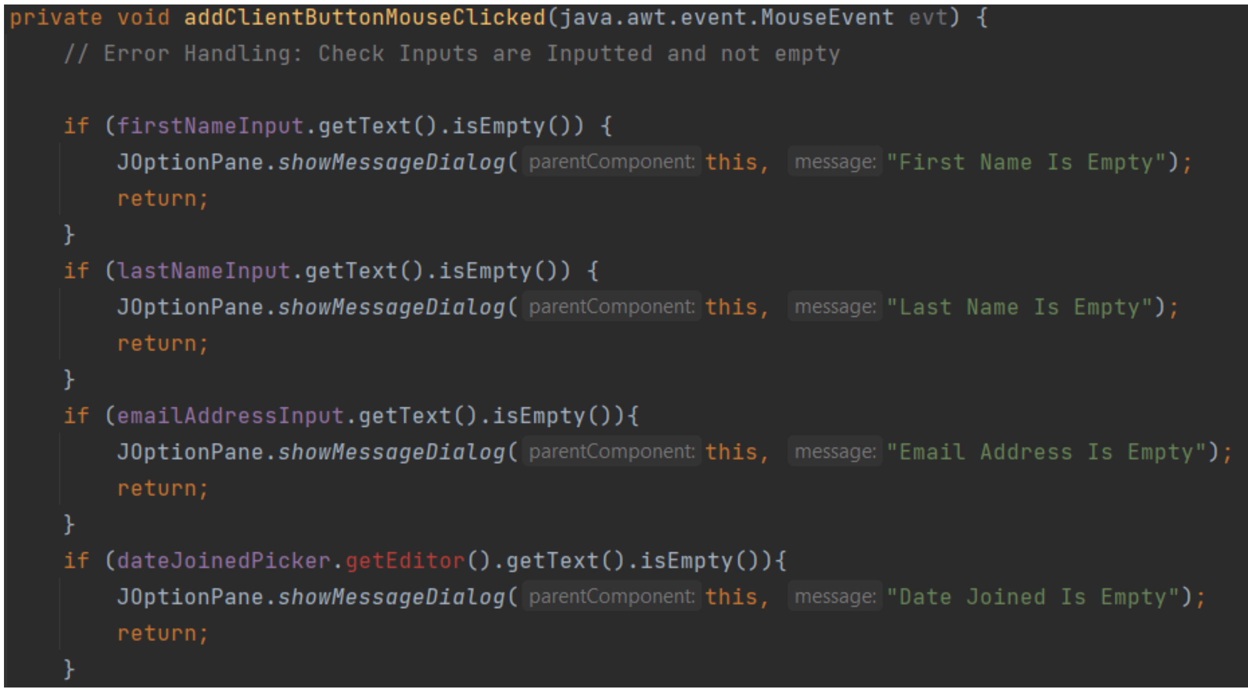
```java
private void runSetup() {
    // Get the latest data from sqlite satabase
    clientList = new DatabaseManager().getClients();

    // Get data from database
    HashMap<Integer, ArrayList<Object>> payments = new DatabaseManager().getPayments();
    HashMap<Integer, Integer> sessions = new DatabaseManager().getSessions();
```

  - Writing data to client and email tables *example*



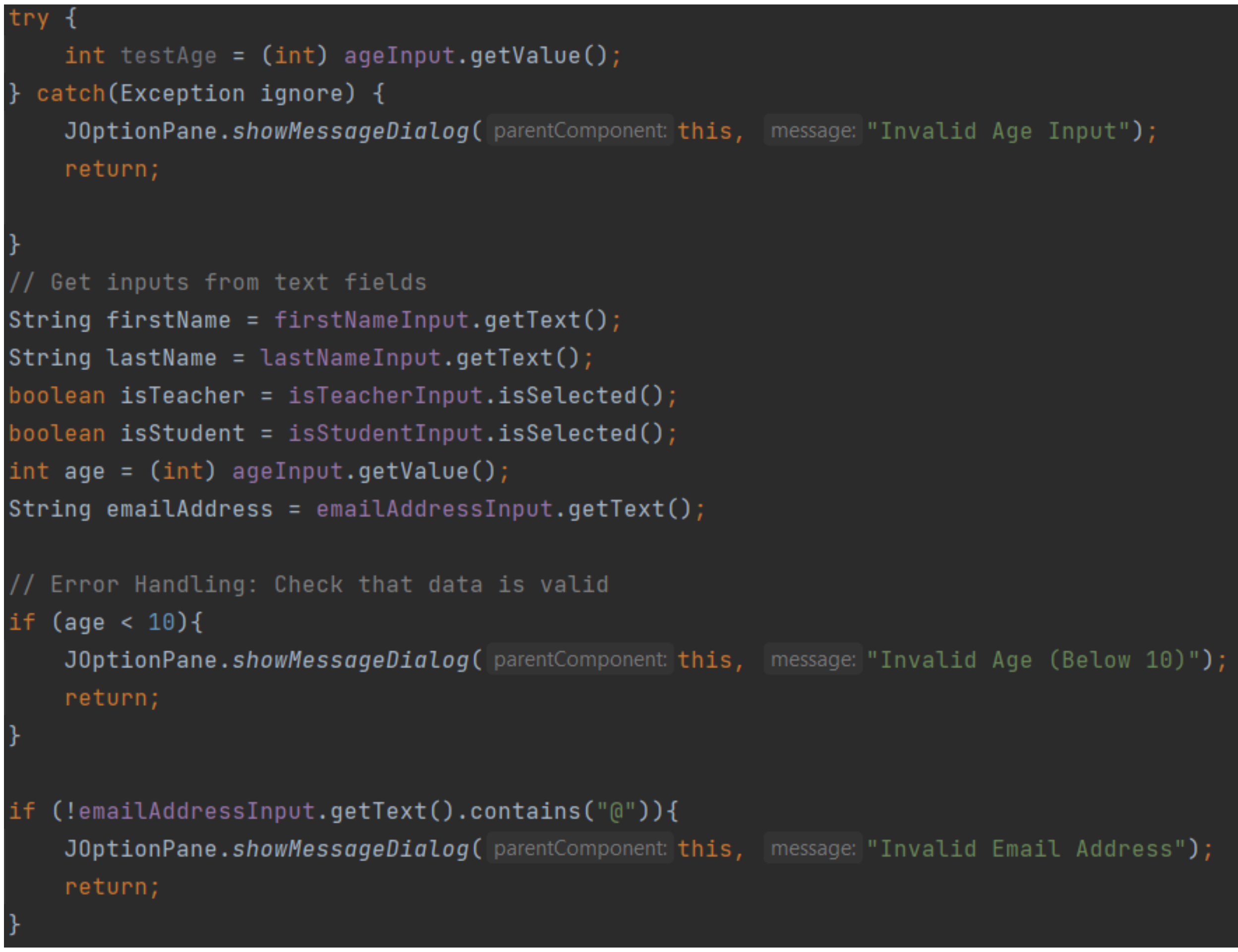
```java
DefaultTableModel dtm = (DefaultTableModel) clientTable.getModel();
dtm.setRowCount(0);
for (int i = 0; i < clientList.size(); i++) {
    dtm.addRow(clientList.get(i).getAsRow());
}

dtm = (DefaultTableModel) emailTable.getModel();
dtm.setRowCount(0);
for (int i = 0; i < clientList.size(); i++) {
    // Turnary conditional statement
    String type = (clientList.get(i).isStudent) ? "student" : (clientList.get(i).isTeacher) ? "teacher" : "neither";

    Object[] objectRow = {false, clientList.get(i).firstName + " " + clientList.get(i).lastName, clientList.get(i).emailAddress, type, false};
    dtm.addRow(objectRow);
}
emailTable.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
```

- Retrieve data from input fields and write to database after the data is validated
  - Check for empty fields



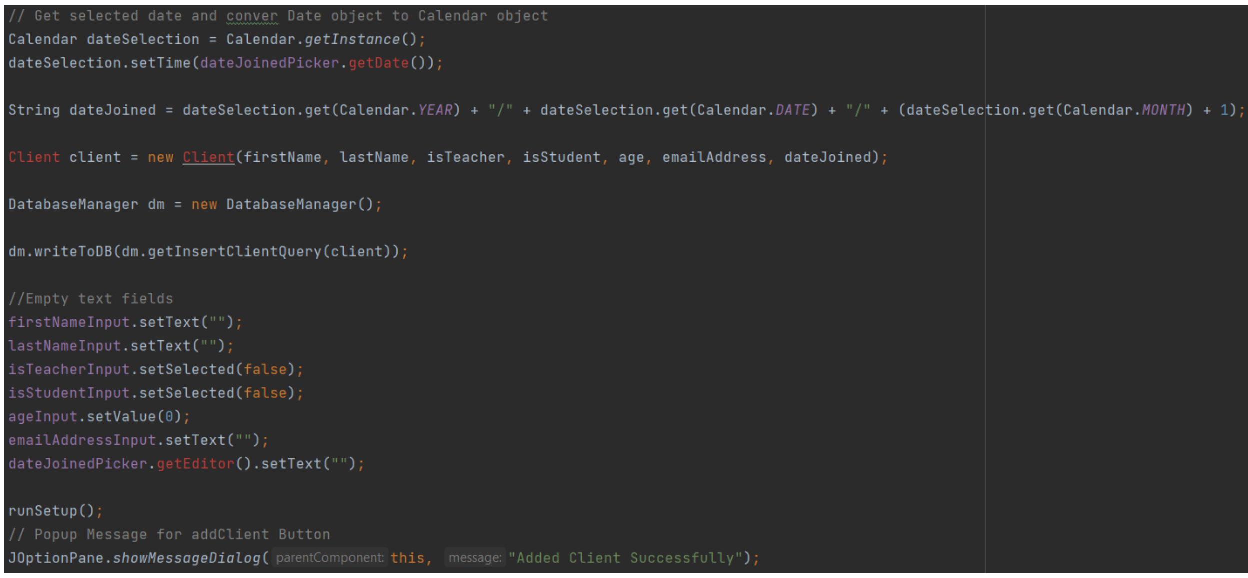
```java
private void addClientButtonMouseClicked(java.awt.event.MouseEvent evt) {
    // Error Handling: Check Inputs are Inputted and not empty

    if (firstNameInput.getText().isEmpty()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "First Name Is Empty");
        return;
    }
    if (lastNameInput.getText().isEmpty()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Last Name Is Empty");
        return;
    }
    if (emailAddressInput.getText().isEmpty()){
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Email Address Is Empty");
        return;
    }
    if (dateJoinedPicker.getEditor().getText().isEmpty()){
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Date Joined Is Empty");
        return;
    }
```

- Check that the data is valid (Valid age/ email etc…)



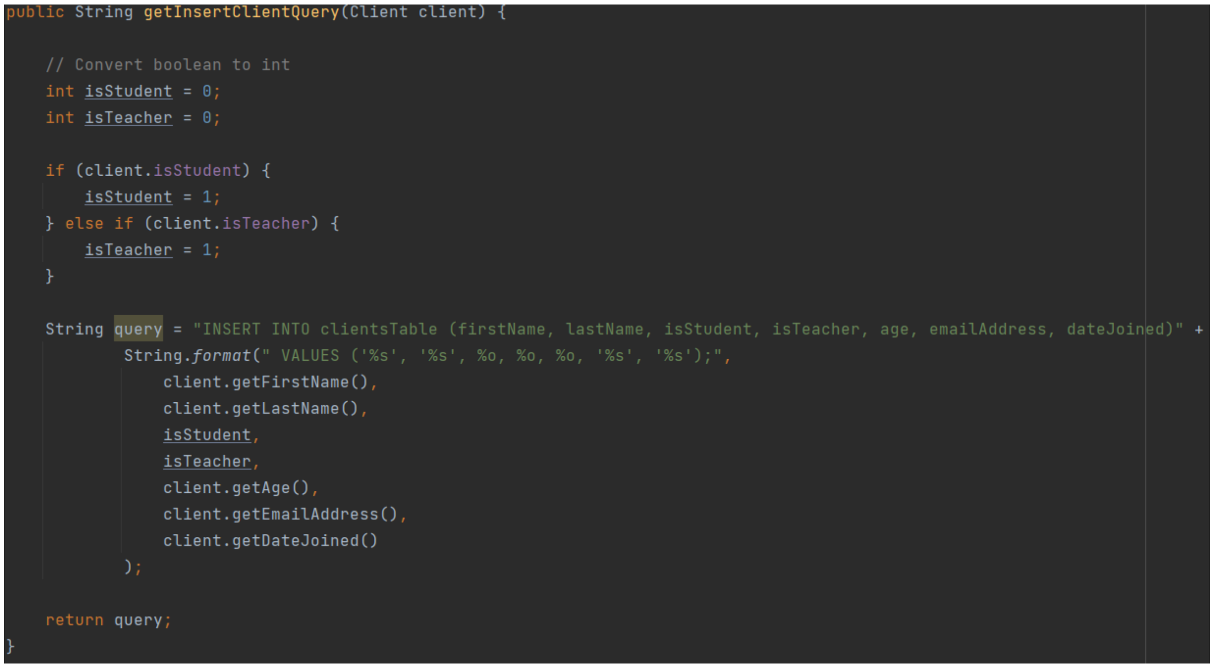
```java
try {
    int testAge = (int) ageInput.getValue();
} catch(Exception ignore) {
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Invalid Age Input");
    return;

}
// Get inputs from text fields
String firstName = firstNameInput.getText();
String lastName = lastNameInput.getText();
boolean isTeacher = isTeacherInput.isSelected();
boolean isStudent = isStudentInput.isSelected();
int age = (int) ageInput.getValue();
String emailAddress = emailAddressInput.getText();

// Error Handling: Check that data is valid
if (age < 10){
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Invalid Age (Below 10)");
    return;
}

if (!emailAddressInput.getText().contains("@")){
    JOptionPane.showMessageDialog( parentComponent: this,  message: "Invalid Email Address");
    return;
}
```

- Send data to query composer function and write to database



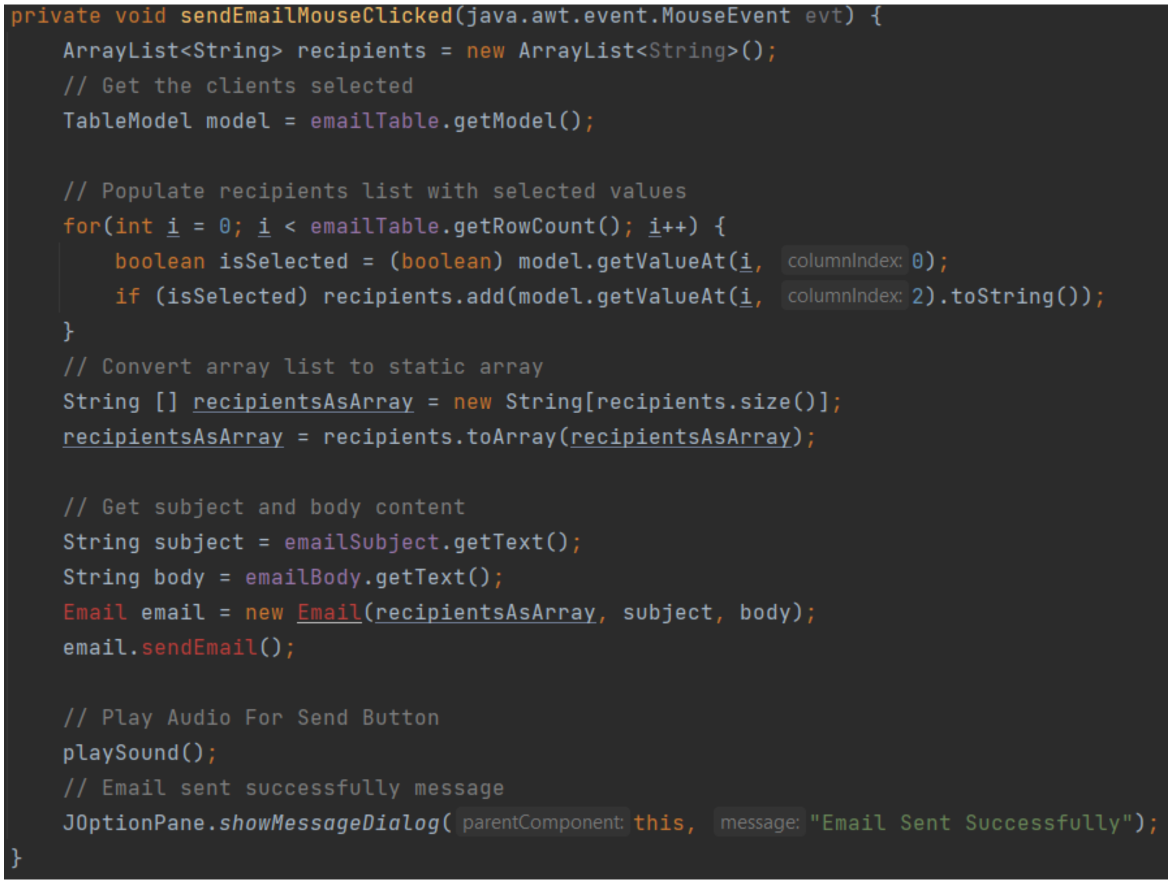
```java
// Get selected date and conver Date object to Calendar object
Calendar dateSelection = Calendar.getInstance();
dateSelection.setTime(dateJoinedPicker.getDate());

String dateJoined = dateSelection.get(Calendar.YEAR) + "/" + dateSelection.get(Calendar.DATE) + "/" + (dateSelection.get(Calendar.MONTH) + 1);

Client client = new Client(firstName, lastName, isTeacher, isStudent, age, emailAddress, dateJoined);

DatabaseManager dm = new DatabaseManager();

dm.writeToDB(dm.getInsertClientQuery(client));

//Empty text fields
firstNameInput.setText("");
lastNameInput.setText("");
isTeacherInput.setSelected(false);
isStudentInput.setSelected(false);
ageInput.setValue(0);
emailAddressInput.setText("");
dateJoinedPicker.getEditor().setText("");

runSetup();
// Popup Message for addClient Button
JOptionPane.showMessageDialog( parentComponent: this,  message: "Added Client Successfully");
```

- Composing SQLite insertion query when adding a new client



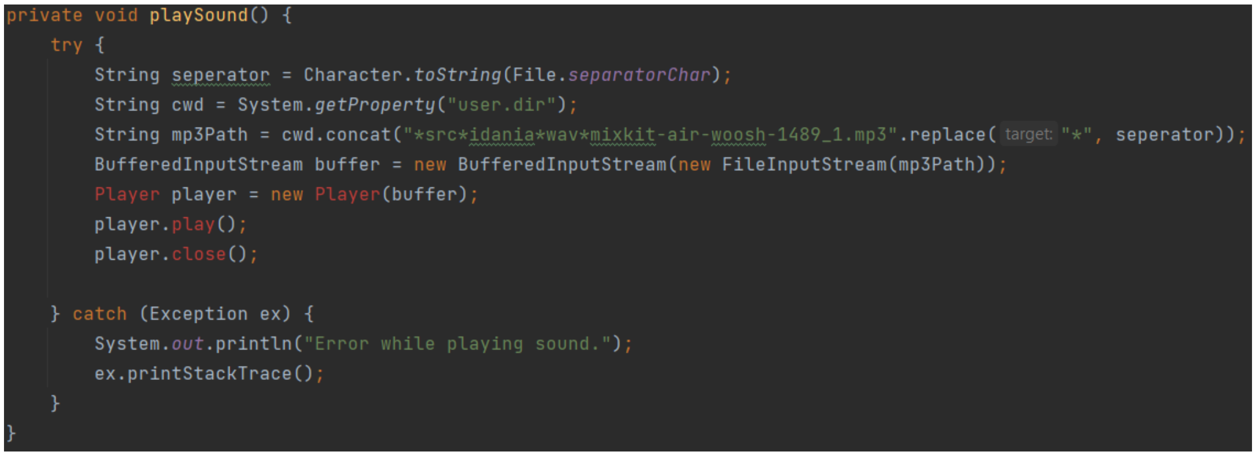
```java
public String getInsertClientQuery(Client client) {

    // Convert boolean to int
    int isStudent = 0;
    int isTeacher = 0;

    if (client.isStudent) {
        isStudent = 1;
    } else if (client.isTeacher) {
        isTeacher = 1;
    }

    String query = "INSERT INTO clientsTable (firstName, lastName, isStudent, isTeacher, age, emailAddress, dateJoined)" +
            String.format(" VALUES ('%s', '%s', %o, %o, %o, '%s', '%s');",
                client.getFirstName(),
                client.getLastName(),
                isStudent,
                isTeacher,
                client.getAge(),
                client.getEmailAddress(),
                client.getDateJoined()
            );

    return query;
}
```

- Send email to clients when the “sendButton” is clicked using the Email class. If the sendEmail method is executed successfully, the client will be prompted with a JOptionPane as well as a “swoosh” sound effect.



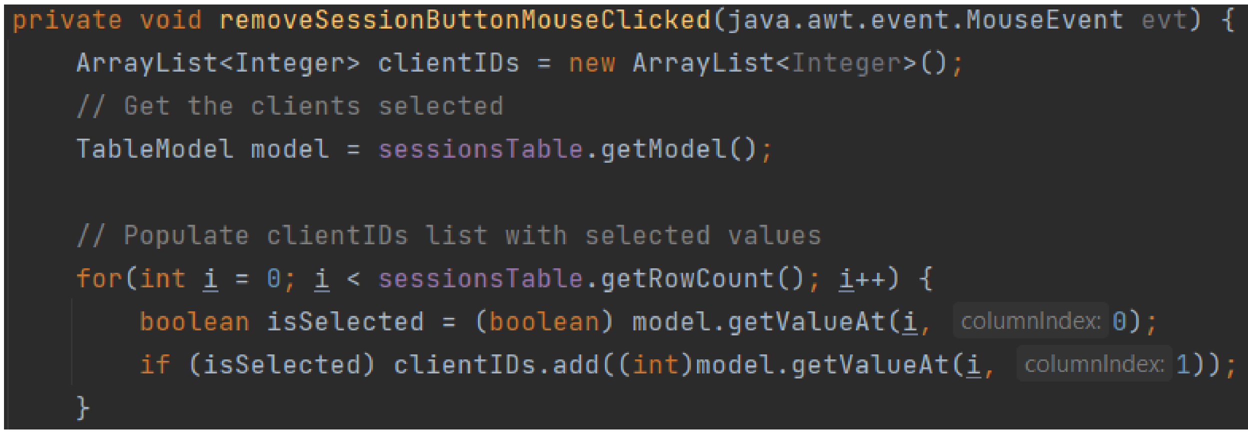
```java
private void sendEmailMouseClicked(java.awt.event.MouseEvent evt) {
    ArrayList<String> recipients = new ArrayList<String>();
    // Get the clients selected
    TableModel model = emailTable.getModel();

    // Populate recipients list with selected values
    for(int i = 0; i < emailTable.getRowCount(); i++) {
        boolean isSelected = (boolean) model.getValueAt(i, columnIndex: 0);
        if (isSelected) recipients.add(model.getValueAt(i, columnIndex: 2).toString());
    }
    // Convert array list to static array
    String [] recipientsAsArray = new String[recipients.size()];
    recipientsAsArray = recipients.toArray(recipientsAsArray);

    // Get subject and body content
    String subject = emailSubject.getText();
    String body = emailBody.getText();
    Email email = new Email(recipientsAsArray, subject, body);
    email.sendEmail();

    // Play Audio For Send Button
    playSound();
    // Email sent successfully message
    JOptionPane.showMessageDialog( parentComponent: this, message: "Email Sent Successfully");
}
```

- Playing sound after the “sendEmail” button is clicked.



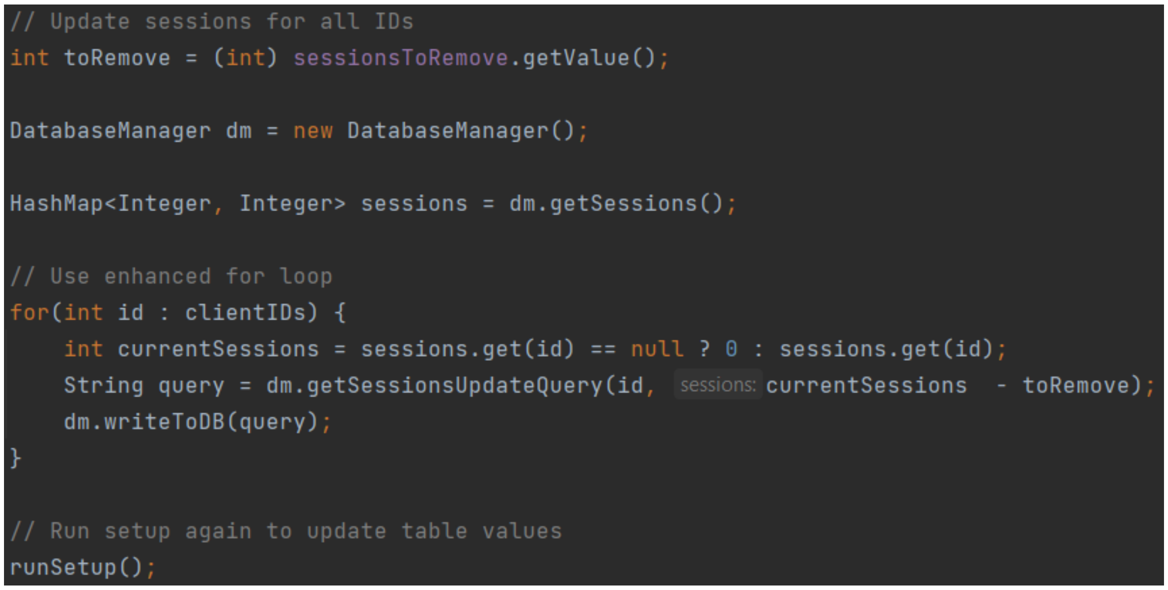
```
private void playSound() {
    try {
        String seperator = Character.toString(File.separatorChar);
        String cwd = System.getProperty("user.dir");
        String mp3Path = cwd.concat("*src*idania*wav*mixkit-air-woosh-1489_1.mp3".replace( target: "*", seperator));
        BufferedInputStream buffer = new BufferedInputStream(new FileInputStream(mp3Path));
        Player player = new Player(buffer);
        player.play();
        player.close();

    } catch (Exception ex) {
        System.out.println("Error while playing sound.");
        ex.printStackTrace();
    }
}
```

- Updating removed session status and sending query to database
    - Populate selected clients to recipients HashMap (quantity of removed sessions)



```
private void removeSessionButtonMouseClicked(java.awt.event.MouseEvent evt) {
    ArrayList<Integer> clientIDs = new ArrayList<Integer>();
    // Get the clients selected
    TableModel model = sessionsTable.getModel();

    // Populate clientIDs list with selected values
    for(int i = 0; i < sessionsTable.getRowCount(); i++) {
        boolean isSelected = (boolean) model.getValueAt(i,  columnIndex: 0);
        if (isSelected) clientIDs.add((int)model.getValueAt(i,  columnIndex: 1));
    }
```

    - Looping over selected clients in recipients HashMap with enhanced for loop and running the session status update query for each one within the loop. After all updates are made, the runSetup method is called to update the jTables.



```
// Update sessions for all IDs
int toRemove = (int) sessionsToRemove.getValue();

DatabaseManager dm = new DatabaseManager();

HashMap<Integer, Integer> sessions = dm.getSessions();

// Use enhanced for loop
for(int id : clientIDs) {
    int currentSessions = sessions.get(id) == null ? 0 : sessions.get(id);
    String query = dm.getSessionsUpdateQuery(id,  sessions: currentSessions  - toRemove);
    dm.writeToDB(query);
}

// Run setup again to update table values
runSetup();
```

User Interface/GUI Work:
The program utilizes Java's Swing tools to provide interacteractable graphic interfaces for the user. The following components are used:
- JTextFields
- JCombobox
- JButtons
- JLabels
- JTabbedPane
- JCheckBox
- JTable
- JCalendar

Software Tools Used:
The Netbeans IDE (Integrated Development Environment) was utilized to code this program. This tool was selected as it is a user-friendly IDE that allows the programmer to develop and create functional GUI interfaces with drag-and-drop features. Five different libraries were used for a variety of functionality. SQLite-JDBC for the database, javax.mail for sending emails, swingx for the GUI, javazoom for sound (effects), and Jcalendar for associating the date off of a calendar.

Planning The SQLite Database:
Following the interview, the client requested to keep track of the following data[1]:
- First Name
- Last Name
- Email Address
- Age
- Occupation (Teacher/Student)
- Date Joined
- Amount of sessions purchased
- Payment Status

Given these inputs, the ideal way to store this information would be into three separate tables.
- Table 1: Current Clients
  - Client ID (Integer)
  - First Name (String)
  - Last Name (String)
  - Is student (Boolean represented by Integer)*
  - Is Teacher (Boolean represented by Integer)*
  - Age (Integer)
  - Email Address (String)
  - Date Joined (String)
- Table 2: Client Sessions

[1] Tony Hemsley, interview by author, Bangkok, January 13, 2022, Interview #1

    - Client ID (Integer)
    - Sessions Count (Integer)
- Table 3: Payments
    - Client ID (Integer)
    - Payment Date (String)
    - Is Paid (Boolean represented by Integer)*

*Booleans will be represented by Integers as SQLite does not support booleans.

The final SQLite Schema can be seen below

```
CREATE TABLE paymentsTable (
   clientID INTEGER,
   paymentDate TEXT,
   isPaid INTEGER
);


CREATE TABLE clientsTable (
   id INTEGER PRIMARY KEY AUTOINCREMENT,
   firstName TEXT,
   lastName TEXT,
   isStudent INTEGER,
   isTeacher INTEGER,
   age INTEGER,
   emailAddress TEXT,
   dateJoined TEXT
);

CREATE TABLE sessionsTable (
   clientID INTEGER,
   sessionsCount INTEGER
);
```

Word Count: 541