Appendix 2

Account Class

```
import org.sqlite.SQLiteDataSource;


import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class Account
{
    private String myUsername;
    private String myPassword;
    private String myEmail;
    private Database myUserDatabase;
    private String myAdmin;


    public Account(String inputName, String inputPassword, String
inputEmail, String inputAdmin)
    {
        myUsername = inputName;
        myPassword = inputPassword;
        myEmail = inputEmail;
        myUserDatabase = new Database();
        myAdmin = inputAdmin;
    }


    //returns myUsername
    public String getUsername()
    {
```

```
        return myUsername;
    }

    //returns myPassword
    public String getPassword()
    {
        return myPassword;
    }

    //returns myEmail
    public String getEmail()
    {
        return myEmail;
    }

    //returns true if row in Users uses myUsername as value
    //returns false if not.
    public boolean usernameInUse()
    {
        boolean result = false;

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myUserDatabase.getDS();

        String query = "SELECT * FROM Users WHERE Username= '" +
myUsername + "'";

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
```

```
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
            {

                result = true;
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        return result;

    }

    //returns true if row in Users uses myEmail
    //returns false if not
    public boolean emailInUse()
    {
        boolean result = false;

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myUserDatabase.getDS();

        String query = "SELECT * FROM Users WHERE Email='" + myEmail +
"'";
```

```
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
            {
                result = true;
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

        return result;

    }

    //Enters a row containing myUsername,myPassword,myEmail, and
myAdmin into Users
    public void enterAccount()
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myUserDatabase.getDS();
```

```
        String query = "INSERT INTO Users (Username, Password, Email,
Admin) VALUES ('" + myUsername + "','" + myPassword + "','" + myEmail
+ "', '" + myAdmin + "')";


        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }


    //Searches User table for a row where the username, password, and
email match this myUsername and myPassword
    //Returns true if said row exists, and false if it does not
    public boolean existsInDatabase()
    {
        boolean result = false;


        SQLiteDataSource source = myUserDatabase.getDS();


        //gets SQLiteDataSource from JapaneseClub
        String query = "SELECT * FROM Users WHERE Username='" +
myUsername + "' AND Password ='" + myPassword + "'";
```

```
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
                result = true;
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

        return result;
    }


    //Searches Users for row where Username, Password, and Email are
equal to myUsername,myPassword,and myEmail
    //If row exists, checks Admin column
    //If Admin is "A", user is an edmin, and this method returns true
    //If Admin is "N", user is not an admin, and this method returns
false
    public boolean isAdmin()
    {
        boolean result = false;

        //gets SQLiteDataSource from JapaneseClub
```

```
        SQLiteDataSource source = myUserDatabase.getDS();


        String query = "SELECT * FROM Users WHERE Username='" +
myUsername + "' AND Password ='" + myPassword + "'";


        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
                if((rs.getString("Admin")).equals("A"))
                    result = true;
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        return result;
    }


    public void createAdmin()
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myUserDatabase.getDS();
        String initialQuery = "SELECT * FROM Users WHERE Username='" +
myUsername + "' AND Password ='" + myPassword + "' AND Email='" +
myEmail + "'";
```

```
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(initialQuery);
            if(rs.next())
            {
                String updateQuery = "UPDATE Users SET Admin = 'A'
WHERE ID='" + rs.getInt("ID") + "'";
                stat.execute(updateQuery);
            }
            else
                enterAccount();
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }
}
```

AdminViewCalendar Class

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import org.sqlite.SQLiteDataSource;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import com.toedter.calendar.JCalendar;
import javax.swing.JTextField;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;

public class AdminViewCalendar extends JFrame
{
    private JPanel contentPane;
    private DefaultTableModel allStudentModel;
    private JTable allStudents;
    private JScrollPane allStudentPane;
    private DefaultTableModel studentModel;
    private DefaultTableModel eventModel;
    private JTable attendingStudents;
    private JTable currentEvents;
    private JScrollPane studentPane;
    private JScrollPane currentEventPanel;
    private Database eventBase;
    private JTextField eventNameTxtFld;
```

```
    private JTextField fromTxtFld;
    private JTextField untilTxtFld;
    private JTextField yearTxtFld;
    private JTextField monthTxtFld;
    private JTextField dayTxtFld;
    private JTextField locationTxtFld;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        AdminViewCalendar frame = new AdminViewCalen-
dar();
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
    }

    /**
     * Create the frame.
     */
    public AdminViewCalendar()
    {
        eventBase = new Database(); //creates new Database object

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 660, 739);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        //creates eventPanel
        JPanel eventPanel =new JPanel();
        eventPanel.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        eventPanel.setBounds(33, 132, 442, 510);
        contentPane.add(eventPanel);
        eventPanel.setLayout(null);
        eventPanel.setVisible(false);
```

```
        JLabel eventDate = new JLabel("Date");
        eventDate.setHorizontalAlignment(SwingConstants.CENTER);
        eventDate.setFont(new Font("Tahoma", Font.PLAIN, 30));
        eventDate.setBounds(33, 13, 323, 29);
        eventPanel.add(eventDate);

        JLabel eventName = new JLabel("Name");
        eventName.setHorizontalAlignment(SwingConstants.CENTER);
        eventName.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventName.setBounds(38, 52, 334, 29);
        eventPanel.add(eventName);

        JLabel eventTimes = new JLabel("From - Until");
        eventTimes.setHorizontalAlignment(SwingConstants.CENTER);
        eventTimes.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventTimes.setBounds(90, 91, 229, 25);
        eventPanel.add(eventTimes);

        JLabel eventLocation = new JLabel("Location");
        eventLocation.setHorizontalAlignment(SwingConstants.CENTER);
        eventLocation.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventLocation.setBounds(80, 126, 276, 32);
        eventPanel.add(eventLocation);

        JLabel attendingStudentsLbl = new JLabel("Attending Mem-
bers:");
        attendingStudentsLbl.setHorizontalAlignment(SwingCon-
stants.CENTER);
        attendingStudentsLbl.setFont(new Font("Tahoma", Font.PLAIN,
20));
        attendingStudentsLbl.setBounds(5, 157, 197, 29);
        eventPanel.add(attendingStudentsLbl);

        //creates panel which will list currently selected events
        JPanel allEventPanel = new JPanel();
        allEventPanel.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        allEventPanel.setBounds(23, 373, 595, 255);
        contentPane.add(allEventPanel);
        allEventPanel.setLayout(null);

        JLabel allEventLbl = new JLabel("Current Events");
        allEventLbl.setHorizontalAlignment(SwingConstants.CENTER);
        allEventLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
        allEventLbl.setBounds(0, 0, 595, 59);
        allEventPanel.add(allEventLbl);
```

```
        //initializes JCalendar
        final JCalendar currentCalendar = new JCalendar();
        currentCalendar.addPropertyChangeListener(new PropertyChange-
Listener() {
                public void propertyChange(PropertyChangeEvent evt) {
                    eventPanel.setVisible(false);
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    String convertedDate = convertDate(selectedYear,
selectedMonth,selectedDay);
                    if(currentEventPanel != null)
                    {
                        allEventPanel.remove(currentEventPanel);
                    }
                    //fills currentEvents with all events on the given
day
                    fillCurrentEvents(convertedDate);
                    currentEvents = new JTable(eventModel);
                    //adds event handler so that when currentEvents
table is clicked, loads event pop-up
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                allEventPanel.setVisible(false);
                                currentCalendar.setVisible(false);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;
```

```
                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
```

```
                                studentPane.setBounds(15, 236, 403,
211);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);
                }
            });
        currentCalendar.getMonthChooser().getSpinner().setFont(new
Font("Tahoma", Font.PLAIN, 20));
        currentCalendar.getYearChooser().getSpinner().setFont(new
Font("Tahoma", Font.PLAIN, 20));
        currentCalendar.getMonthChooser().getSpinner().setBack-
ground(new Color(0, 206, 209));
        currentCalendar.getMonthChooser().getComboBox().setFont(new
Font("Tahoma", Font.PLAIN, 20));
        currentCalendar.setBounds(23, 130, 595, 206);
        contentPane.add(currentCalendar);

        JButton eventCloseBtn = new JButton("Close");
        eventCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                 //clears the eventPanel for the next time it's open
                    studentPane.removeAll();
                    eventPanel.remove(studentPane);
                    eventPanel.setVisible(false);
                    //re-opens the calendar and the currentEvent table
                    currentCalendar.setVisible(true);
                    allEventPanel.setVisible(true);
                }
            });
        eventCloseBtn.setBackground(new Color(0, 206, 209));
        eventCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventCloseBtn.setBounds(20, 457, 95, 29);
        eventPanel.add(eventCloseBtn);

        JPanel addStudentPopup = new JPanel();
```

```
        addStudentPopup.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        addStudentPopup.setBounds(104, 236, 479, 176);
        contentPane.add(addStudentPopup);
        addStudentPopup.setVisible(false);

        //initializes panel that asks user if they want to remove
Event from JapaneseClub
        JPanel removeEventPanel = new JPanel();
        removeEventPanel.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        removeEventPanel.setBounds(135, 411, 456, 107);
        contentPane.add(removeEventPanel);
        removeEventPanel.setLayout(null);
        removeEventPanel.setVisible(false);

        JLabel removeLbl = new JLabel("Are you sure you want to remove
this event?");
        removeLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        removeLbl.setBounds(20, 5, 422, 26);
        removeEventPanel.add(removeLbl);

        //removes Event from Events
        JButton removeYesBtn = new JButton("Yes");
        removeYesBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {

                    //gets date from JCalendar
                    String date = eventDate.getText();
                    int eventID = searchEvent(date.substring(6));

                    //deletes event from Events, then deletes all rows
in EventStudents with same EventID
                    removeEvent(eventID);
                    removeStudents(eventID);
                    removeEventPanel.setVisible(false);
                    eventPanel.setVisible(false);

                    if(currentEvents != null)
                    {
                    allEventPanel.remove(currentEventPanel);
                    }
                    fillCurrentEvents("Show All");
                    currentEvents = new JTable(eventModel);
```

```
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                allEventPanel.setVisible(false);
                                currentCalendar.setVisible(false);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;

                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
                                catch(SQLException e1)
```

```
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(15, 236, 403,
211);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);
                    currentCalendar.setVisible(true);
                    allEventPanel.setVisible(true);
                }
            });
        removeYesBtn.setBackground(new Color(0, 206, 209));
        removeYesBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        removeYesBtn.setBounds(54, 54, 131, 37);
        removeEventPanel.add(removeYesBtn);

        JButton removeNoBtn = new JButton("No");
        removeNoBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
```

```
                //closes event panel and makes the currentEvent table
and calendar visible
                    removeEventPanel.setVisible(false);
                    allEventPanel.setVisible(true);
                    currentCalendar.setVisible(true);
                }
            });
        removeNoBtn.setBackground(new Color(0, 206, 209));
        removeNoBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        removeNoBtn.setBounds(244, 56, 123, 33);
        removeEventPanel.add(removeNoBtn);

        //creates errorPopup
        JPanel errorPopup = new JPanel();
        errorPopup.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        errorPopup.setBounds(140, 196, 469, 88);
        contentPane.add(errorPopup);
        errorPopup.setLayout(null);
        errorPopup.setVisible(false);

        JLabel errorLbl = new JLabel("Please make sure all fields were
entered correctly.");
        errorLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        errorLbl.setBounds(10, 10, 459, 25);
        errorPopup.add(errorLbl);

        JButton errorCloseBtn = new JButton("Close");
        errorCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    errorPopup.setVisible(false);
                }
            });
        errorCloseBtn.setBackground(new Color(0, 206, 209));
        errorCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        errorCloseBtn.setBounds(172, 45, 85, 33);
        errorPopup.add(errorCloseBtn);

        //creates addEventPanel, which doubles as the panel where us-
ers can edit events
        JPanel addEventPanel = new JPanel();
        addEventPanel.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        addEventPanel.setBounds(205, 300, 390, 328);
        contentPane.add(addEventPanel);
        addEventPanel.setLayout(null);
```

```
        addEventPanel.setVisible(false);

        JButton removeBtn = new JButton("Delete Event");
        removeBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    removeEventPanel.setVisible(true);
                    }
            });
        removeBtn.setBackground(new Color(0, 206, 209));
        removeBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        removeBtn.setBounds(261, 457, 157, 29);
        eventPanel.add(removeBtn);

        JButton removeStudentBtn = new JButton("Remove Member");
        removeStudentBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    int rowNum = attendingStudents.getSelectedRow();
                    String date = eventDate.getText();
                    int eventID = searchEvent(date.substring(6));
                    String last = (String)attendingStudents.get-
ValueAt(rowNum, 1);
                    String first = (String) attendingStudents.get-
ValueAt(rowNum,2);

                    Student tempStudent = new Student(last, first,
0,0);
                    int studentID = tempStudent.searchStudent(last,
first);

                    //gets SQLiteDataSource from JapaneseClub
                    SQLiteDataSource source = eventBase.getDS();
                    String query = "DELETE FROM EventStudents WHERE
EventID= " + eventID + " AND StudentID='" + studentID + "'";

                    //tries to connect to JapaneseClub and execute
query
                    try(Connection conn = source.getConnection();
                    Statement stat = conn.createStatement(); )
                    {
                        stat.execute(query);
                        conn.close();
                    }
                    catch(SQLException i)
                    {
```

```
                    i.printStackTrace();
                    System.exit(0);
                }

                //initializes attendingStudents as new JTable with
all students with EventID matching this event
                studentPane.removeAll();
                eventPanel.remove(studentPane);
                int attendance = fillAttendingStudents(eventID);
                attendingStudentsLbl.setText("Attending Students:
" + attendance);
                attendingStudents = new JTable(studentModel);
                attendingStudents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                studentPane = new JScrollPane(attendingStudents);
                studentPane.setBounds(15, 236, 403, 211);
                eventPanel.add(studentPane);
            }
        });
        removeStudentBtn.setBackground(new Color(0, 206, 209));
        removeStudentBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        removeStudentBtn.setBounds(188, 196, 230, 30);
        eventPanel.add(removeStudentBtn);

        //when editEventBtn clicked, shows addEditPanel (which is also
the panel where events are edited)
        //and fills in addEventPanel's text fields with the data from
JapaneseClub
        JButton editEventBtn = new JButton("Edit Event");
        editEventBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    addEventPanel.setVisible(true);
                    editEventBtn.setVisible(false);
                    removeBtn.setVisible(false);
                    removeStudentBtn.setVisible(false);
                    String currentDate = eventDate.getText();
                    int eventID = searchEvent(currentDate.sub-
string(6));
                    //checks if event exists
                    if(eventID!=-1)
                    {
                        //gets SQLiteDataSource from JapaneseClub
                        SQLiteDataSource source = eventBase.getDS();
                        String query = "SELECT * FROM Events WHERE
ID=" + eventID;
```

```
                        //tries to connect to JapaneseClub and execute
query
                        try(Connection conn = source.getConnection();
                        Statement stat = conn.createStatement();)
                        {
                            ResultSet rs = stat.executeQuery(query);
                            if(rs.next())
                            {
                                eventNam-
eTxtFld.setText(rs.getString("Name"));
                                String date = rs.getString("Date");
                                yearTxtFld.setText(date.sub-
string(0,4));
                                monthTxtFld.setText(date.sub-
string(5,7));
                                dayTxtFld.setText(date.substring(8));

fromTxtFld.setText(rs.getString("Start"));
                                un-
tilTxtFld.setText(rs.getString("End"));
                                loca-
tionTxtFld.setText(rs.getString("Location"));

                            }
                            conn.close();
                        }
                        catch(SQLException e1)
                        {
                            e1.printStackTrace();
                            System.exit(0);
                        }
                    }
                }
            });
        editEventBtn.setBackground(new Color(0, 206, 209));
        editEventBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        editEventBtn.setBounds(125, 457, 126, 29);
        eventPanel.add(editEventBtn);

        JButton saveNewEventBtn =     new JButton("Save");
        saveNewEventBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    //gets selected date from JCalendar
```

```
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    String convertedDate = convertDate(selectedYear,
selectedMonth,selectedDay);
                    int eventID = searchEvent(convertedDate);
                    String name = eventNameTxtFld.getText();
                    String year = yearTxtFld.getText();
                    String month = monthTxtFld.getText();
                    String day = dayTxtFld.getText();
                    String from = fromTxtFld.getText();
                    String until = untilTxtFld.getText();
                    String location = locationTxtFld.getText();

                    //if any fields are blank or not correctly format-
ted, show error message

if(name.equals("")||year.equals("")||month.equals("")||day.equals("")|
|from.equals("")||until.equals("") || location.contentEquals(""))
                    {
                        errorPopup.setVisible(true);
                    }
                    else if(year.length() != 4 || month.length() != 2
|| day.length() != 2)
                    {
                        errorPopup.setVisible(true);
                    }
                    else
                    {
                      editEventBtn.setVisible(true);
                      removeBtn.setVisible(true);
                      removeStudentBtn.setVisible(true);
                      addEventPanel.setVisible(false);
                        convertedDate = convertDate(Integer.val-
ueOf(year), Integer.valueOf(month),Integer.valueOf(day));
                        Event tempEvent = new Event(converted-
Date,name,from,until,location);

                        //if event exists in database, update all of
its values with new inputted values
                        //else do nothing
                        if(eventID != -1)
                        {
```

```
                            //gets SQLiteDataSource from JapaneseClub
                            SQLiteDataSource source = event-
Base.getDS();

                            String query = "SELECT * FROM Events WHERE
ID=" + eventID;

                            //tries to connect to JapaneseClub and ex-
ecuteQuery
                            try(Connection conn = source.getConnec-
tion();
                            Statement stat = conn.createStatement(); )
                            {
                                ResultSet rs = stat.exe-
cuteQuery(query);
                                if(rs.next())
                                {
                                    String date =
rs.getString("Date");
                                    int yearNum = Integer.par-
seInt(date.substring(0,4));
                                    int monthNum = Integer.par-
seInt(date.substring(5,7));
                                    int dayNum = Integer.par-
seInt(date.substring(8));
                                    tempEvent = new Event(year-
Num,monthNum,dayNum,
rs.getString("Name"),rs.getString("Start"),rs.getString("End"),rs.getS
tring("Location"));
                                }
                                conn.close();
                            }
                            catch(SQLException v)
                            {
                                v.printStackTrace();
                                System.exit(0);
                            }
                            //fills in the fields on the evnet panel
                            tempEvent.setName(name,eventID);
                            tempEvent.setStart(from,eventID);
                            tempEvent.setEnd(until, eventID);
                            tempEvent.updateLocation(location, even-
tID);
                            eventName.setText(name);
                            eventTimes.setText(from + "-" + until);
                            eventLocation.setText(location);
```

```
                        }
                        addEventPanel.setVisible(false);

                        //updates currentEvents to be new JTable in-
cluding newly created event
                        currentEventPanel.removeAll();
                        allEventPanel.remove(currentEventPanel);
                        fillCurrentEvents(convertedDate);
                        currentEvents = new JTable(eventModel);
                        currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                        currentEventPanel = new JScrollPane(currentE-
vents);
                        currentEventPanel.setBounds(10, 71, 575, 163);
                        allEventPanel.add(currentEventPanel);

                        //adds an event listener to the currentEvent
table so that the event panel is populated when an event is clicked
                        currentEvents.addMouseListener(new MouseA-
dapter() {
                                @Override
                                public void mouseClicked(MouseEvent e)
                                {
                                    //makes everything else not visi-
ble to clear up visual clutter
                                    eventPanel.setVisible(true);
                                    allEventPanel.setVisible(false);
                                    currentCalendar.setVisible(false);
                                    int rowNum = currentEvents.get-
SelectedRow();
                                    int eventID =
searchEvent((String)currentEvents.getValueAt(rowNum, 0));

                                    //gets SQLiteDataSource from Japa-
neseClub
                                    SQLiteDataSource source = event-
Base.getDS();
                                    String query = "SELECT * FROM
Events WHERE ID=" + eventID;

                                    //tries to connect to JapaneseClub
and execute query
                                    try(Connection conn =
source.getConnection();
                                    Statement stat = conn.createState-
ment();)
```

```
                                    {
                                        ResultSet rs = stat.exe-
cuteQuery(query);
                                        if(rs.next())
                                        {
                                            eventDate.setText("Date: "
+ rs.getString("Date"));
                                            eventName.setText("Name: "
+ rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                            eventLocation.setText("Lo-
cation: " + rs.getString("Location"));
                                        }
                                        conn.close();
                                    }
                                    catch(SQLException e1)
                                    {
                                        e1.printStackTrace();
                                        System.exit(0);
                                    }
                                    //initializes attendingStudents as
new JTable with all students with EventID matching this event
                                    if(studentPane != null)
                                    {
                                        eventPanel.remove(student-
Pane);
                                    }
                                    int attendance = fillAttendingStu-
dents(eventID);
                                    attendingStudentsLbl.setText("At-
tending Members: " + attendance);
                                    attendingStudents = new
JTable(studentModel);
                                    attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                    studentPane = new JScrollPane(at-
tendingStudents);
                                    studentPane.setBounds(15, 236,
403, 211);
                                    eventPanel.add(studentPane);
                                    contentPane.add(eventPanel);

                                }
                            });
```

```
                    }
                }
            });
        saveNewEventBtn.setBackground(new Color(0, 206, 209));
        saveNewEventBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        saveNewEventBtn.setBounds(123, 271, 108, 32);
        addEventPanel.add(saveNewEventBtn);

        //if all values are correct, adds newly created event to
Events
        //else shows error message
        JButton addNewEventBtn = new JButton("Add");
        addNewEventBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    String name = eventNameTxtFld.getText();
                    String year = yearTxtFld.getText();
                    String month = monthTxtFld.getText();
                    String day = dayTxtFld.getText();
                    String from = fromTxtFld.getText();
                    String until = untilTxtFld.getText();
                    String location = locationTxtFld.getText();


if(name.equals("")||year.equals("")||month.equals("")||day.equals("")|
|from.equals("")||until.equals("") || location.contentEquals(""))
                    {
                        errorPopup.setVisible(true);
                    }
                    else if(year.length() != 4 || month.length() != 2
|| day.length() != 2)
                    {
                        errorPopup.setVisible(true);
                    }
                    else
                    {
                      currentCalendar.setVisible(true);
                      allEventPanel.setVisible(true);
                      removeBtn.setVisible(true);
                      addNewEventBtn.setVisible(false);
                      saveNewEventBtn.setVisible(true);
                        String convertedDate = convertDate(Inte-
ger.valueOf(year), Integer.valueOf(month),Integer.valueOf(day));
                        Event tempEvent = new Event(converted-
Date,name,from,until,location);
                        tempEvent.enterEvent();
```

```
                        addEventPanel.setVisible(false);

                        //updates currentEvents to be new JTable in-
cluding newly created event
                        currentEventPanel.removeAll();
                        allEventPanel.remove(currentEventPanel);
                        fillCurrentEvents(convertedDate);
                        currentEvents = new JTable(eventModel);
                        currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                        currentEventPanel = new JScrollPane(currentE-
vents);
                        currentEventPanel.setBounds(10, 71, 575, 163);
                        allEventPanel.add(currentEventPanel);

                        //adds an event listener to currentEvents so
that the events panel is populated when an event is clicked
                        currentEvents.addMouseListener(new MouseA-
dapter() {
                                @Override
                                public void mouseClicked(MouseEvent e)
                                {
                                    //makes everything else not visi-
ble to clear up visual clutter
                                    eventPanel.setVisible(true);
                                    allEventPanel.setVisible(false);
                                    currentCalendar.setVisible(false);
                                    int rowNum = currentEvents.get-
SelectedRow();
                                    int eventID =
searchEvent((String)currentEvents.getValueAt(rowNum, 0));

                                    //gets SQLiteDataSource from Japa-
neseClub
                                    SQLiteDataSource source = event-
Base.getDS();
                                    String query = "SELECT * FROM
Events WHERE ID=" + eventID;

                                    //tries to connect to JapaneseClub
and execute query
                                    try(Connection conn =
source.getConnection();
                                    Statement stat = conn.createState-
ment();)
                                    {
```

```
                                        ResultSet rs = stat.exe-
cuteQuery(query);
                                        if(rs.next())
                                        {
                                            eventDate.setText("Date: "
+ rs.getString("Date"));
                                            eventName.setText("Name: "
+ rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                            eventLocation.setText("Lo-
cation: " + rs.getString("Location"));
                                        }
                                        conn.close();
                                    }
                                    catch(SQLException e1)
                                    {
                                        e1.printStackTrace();
                                        System.exit(0);
                                    }

                                    //initializes attendingStudents as
new JTable with all students with EventID matching this event
                                    if(studentPane != null)
                                    {
                                        eventPanel.remove(student-
Pane);
                                    }
                                    int attendance = fillAttendingStu-
dents(eventID);
                                    attendingStudentsLbl.setText("At-
tending Members: " + attendance);
                                    attendingStudents = new
JTable(studentModel);
                                    attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                    studentPane = new JScrollPane(at-
tendingStudents);
                                    studentPane.setBounds(15, 236,
403, 211);
                                    eventPanel.add(studentPane);
                                    contentPane.add(eventPanel);

                                }
                            });
```

```
                }

            }
        });
        addNewEventBtn.setBackground(new Color(0, 206, 209));
        addNewEventBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addNewEventBtn.setBounds(10, 272, 85, 31);
        addEventPanel.add(addNewEventBtn);
        addNewEventBtn.setVisible(false);



        //When "See Event Details" is clicked, eventPanel
        //displays the Event's Date, Name, Start time, End Time, Loca-
tion, and Attending Members
        //from Events and EventStudents
        JButton showAllBtn =new JButton("Show All Events");
        showAllBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    eventPanel.setVisible(false);
                    allEventPanel.remove(currentEventPanel);
                    fillCurrentEvents("Show All");
                    currentEvents = new JTable(eventModel);
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                allEventPanel.setVisible(false);
                                currentCalendar.setVisible(false);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;
```

```
                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(15, 236, 403,
211);
```

```
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);
                }
            });
        showAllBtn.setBackground(new Color(0, 206, 209));
        showAllBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        showAllBtn.setBounds(241, 64, 196, 49);
        contentPane.add(showAllBtn);

        //creates JPanel with all members of club
        JButton addStudentsBtn = new JButton("Add Member");
        addStudentsBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    addStudentPopup.setVisible(true);
                    addStudentsBtn.setVisible(false);
                }
            });
        addStudentsBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addStudentsBtn.setBackground(new Color(0, 206, 209));
        addStudentsBtn.setBounds(15, 196, 152, 30);
        eventPanel.add(addStudentsBtn);

        //when "Add Member" button clicked, a row from allStudents has
been selected
        //EventStudents is updated with a new row with the eventID and
the studentID from the selectedRow
        //This adds the student to the JTable attendingStudents, which
is re-initalized with them on it
        JButton newStudentSaveBtn = new JButton("Add Member");
        newStudentSaveBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                 addStudentsBtn.setVisible(true);
                    int rowNum = allStudents.getSelectedRow();

                    //gets date from JCalendar
```

```
                    String date = eventDate.getText();
                    int eventID = searchEvent(date.substring(6));

                    //gets the student ID
                    Student tempStudent = new Student((String)allStu-
dents.getValueAt(rowNum, 0),(String)allStudents.getValueAt(rowNum,
1),-1,-1);
                    int studentID = tempStudent.searchStudent(tempStu-
dent.getMyLastName(), tempStudent.getMyFirstName());

                    //gets SQLiteDataSource from JapaneseClub
                    SQLiteDataSource source = eventBase.getDS();

                    String query = "INSERT INTO EventStudents (Even-
tID, StudentID) VALUES ('" + eventID + "','" + studentID + "')";

                    //tries to connect to JapaneseClub and execute
query
                    try(Connection conn = source.getConnection();
                    Statement stat = conn.createStatement(); )
                    {
                        stat.execute(query);
                        conn.close();
                    }
                    catch(SQLException k)
                    {
                        k.printStackTrace();
                        System.exit(0);
                    }

                    //initializes attendingStudents as new JTable with
all students with EventID matching this event
                    studentPane.removeAll();
                    eventPanel.remove(studentPane);
                    addStudentPopup.setVisible(false);
                    int attendance = fillAttendingStudents(eventID);
                    attendingStudentsLbl.setText("Attending Students:
" + attendance);
                    attendingStudents = new JTable(studentModel);
                    attendingStudents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentPane = new JScrollPane(attendingStudents);
                    studentPane.setBounds(15, 236, 403, 211);
                    eventPanel.add(studentPane);

                }
```

```
    });
addStudentPopup.setLayout(null);
newStudentSaveBtn.setBackground(new Color(0, 206, 209));
newStudentSaveBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
newStudentSaveBtn.setBounds(10, 133, 185, 33);
addStudentPopup.add(newStudentSaveBtn);

//adds labels and text fields to addEventPanel
JLabel newEventName = new JLabel("Event Name: ");
newEventName.setFont(new Font("Tahoma", Font.PLAIN, 20));
newEventName.setBounds(10, 22, 120, 19);
addEventPanel.add(newEventName);

eventNameTxtFld = new JTextField();
eventNameTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
eventNameTxtFld.setBounds(133, 10, 245, 35);
addEventPanel.add(eventNameTxtFld);
eventNameTxtFld.setColumns(10);

JLabel fromLbl = new JLabel("From:");
fromLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
fromLbl.setBounds(10, 196, 68, 19);
addEventPanel.add(fromLbl);

fromTxtFld = new JTextField();
fromTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
fromTxtFld.setBounds(69, 191, 119, 31);
addEventPanel.add(fromTxtFld);
fromTxtFld.setColumns(10);

JLabel untilLbl = new JLabel("Until:");
untilLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
untilLbl.setBounds(198, 195, 61, 20);
addEventPanel.add(untilLbl);

untilTxtFld = new JTextField();
untilTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
untilTxtFld.setBounds(252, 188, 126, 35);
addEventPanel.add(untilTxtFld);
untilTxtFld.setColumns(10);

JLabel yearLbl = new JLabel("Year (YYYY):");
yearLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
yearLbl.setBounds(10, 67, 120, 19);
addEventPanel.add(yearLbl);
```

```
        yearTxtFld = new JTextField();
        yearTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        yearTxtFld.setBounds(133, 55, 245, 35);
        addEventPanel.add(yearTxtFld);
        yearTxtFld.setColumns(10);

        JLabel monthLbl = new JLabel("Month (MM):");
        monthLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        monthLbl.setBounds(10, 108, 120, 19);
        addEventPanel.add(monthLbl);

        monthTxtFld = new JTextField();
        monthTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        monthTxtFld.setBounds(140, 100, 238, 35);
        addEventPanel.add(monthTxtFld);
        monthTxtFld.setColumns(10);

        JLabel dayLbl = new JLabel("Day (DD) :");
        dayLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        dayLbl.setBounds(10, 146, 108, 23);
        addEventPanel.add(dayLbl);

        dayTxtFld = new JTextField();
        dayTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        dayTxtFld.setBounds(111, 143, 257, 38);
        addEventPanel.add(dayTxtFld);
        dayTxtFld.setColumns(10);

        JLabel locationLbl = new JLabel("Location:");
        locationLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        locationLbl.setBounds(10, 227, 135, 26);
        addEventPanel.add(locationLbl);

        locationTxtFld = new JTextField();
        locationTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        locationTxtFld.setBounds(113, 232, 265, 29);
        addEventPanel.add(locationTxtFld);
        locationTxtFld.setColumns(10);

        JButton newEventCloseBtn = new JButton("Close");
        newEventCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                 //closes the event panel and makes the calendar and
currentEvents visible
                 eventPanel.setVisible(false);
```

```
                addEventPanel.setVisible(false);
                currentCalendar.setVisible(true);
                allEventPanel.setVisible(true);
                editEventBtn.setVisible(true);
                removeStudentBtn.setVisible(true);
                removeBtn.setVisible(true);
                addNewEventBtn.setVisible(false);
                saveNewEventBtn.setVisible(true);
            }
        });
        newEventCloseBtn.setBackground(new Color(0, 206, 209));
        newEventCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        newEventCloseBtn.setBounds(252, 272, 126, 31);
        addEventPanel.add(newEventCloseBtn);

        JButton newStudentCloseBtn = new JButton("Close");
        newStudentCloseBtn.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                addStudentPopup.setVisible(false);
                addStudentsBtn.setVisible(true);
            }
        });
        newStudentCloseBtn.setBackground(new Color(0, 206, 209));
        newStudentCloseBtn.setFont(new Font("Tahoma", Font.PLAIN,
20));
        newStudentCloseBtn.setBounds(348, 133, 111, 33);
        addStudentPopup.add(newStudentCloseBtn);

        JLabel eventCalendarLbl = new JLabel("Event Calendar");
        eventCalendarLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
        eventCalendarLbl.setBounds(21, 64, 409, 49);
        contentPane.add(eventCalendarLbl);

        //goes to home page window
        JButton homeBtn = new JButton("Home");
        homeBtn.addMouseListener(new MouseAdapter()
        {
            @Override
            public void mouseClicked(MouseEvent e)
            {
                AdminViewHomepage home = new AdminViewHomepage();
                home.setVisible(true);
                setVisible(false);
            }
        });
```

```
        homeBtn.setBackground(new Color(0, 206, 209));
        homeBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        homeBtn.setBounds(23, 659, 163, 33);
        contentPane.add(homeBtn);

        JButton logoutBtn = new JButton("Log Out");
        logoutBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    LogoutScreen logout = new LogoutScreen();
                    logout.setVisible(true);
                    setVisible(false);
                }
            });
        logoutBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        logoutBtn.setBackground(new Color(0, 206, 209));
        logoutBtn.setBounds(422, 659, 196, 33);
        contentPane.add(logoutBtn);

        //goes to Roster window
        JButton rosterBtn = new JButton("Roster");
        rosterBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    AdminViewRoster roster = new AdminViewRoster();
                    roster.setVisible(true);
                    setVisible(false);
                }
            });
        rosterBtn.setBackground(new Color(0, 206, 209));
        rosterBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        rosterBtn.setBounds(224, 660, 144, 31);
        contentPane.add(rosterBtn);

        //intializes addEventPanel with all text fields blank
        JButton addEventBtn = new JButton("Add Event");
        addEventBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                 currentCalendar.setVisible(false);
                 allEventPanel.setVisible(false);
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
```

```
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    eventNameTxtFld.setText("");
                    yearTxtFld.setText("" + selectedYear);
                    monthTxtFld.setText("" + selectedMonth);
                    if(selectedDay - 10 > 0)
                    {
                      dayTxtFld.setText("" + selectedDay);
                    }
                    else
                    {
                      dayTxtFld.setText("0" + selectedDay);
                    }
                    fromTxtFld.setText("");
                    untilTxtFld.setText("");
                    locationTxtFld.setText("");
                    addEventPanel.setVisible(true);
                    addNewEventBtn.setVisible(true);
                    saveNewEventBtn.setVisible(false);
                    removeBtn.setVisible(false);
                }
            });
        addEventBtn.setBackground(new Color(0, 206, 209));
        addEventBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addEventBtn.setBounds(455, 64, 164, 49);
        contentPane.add(addEventBtn);

        //when the window is opened, the JTable currentEvents is ini-
tialized with all of the events
        //in JapaneseClub. Then it is added to allEventPanel.
        //In addition, the panel with all of the students for adding
members is initialized
        //and added to the addStudentPopup.
        addWindowListener(new WindowAdapter() {
                @Override
                public void windowOpened(WindowEvent e) {
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    String convertedDate = convertDate(selectedYear,
selectedMonth,selectedDay);
```

```
                    currentEventPanel.removeAll();
                    allEventPanel.remove(currentEventPanel);
                    fillCurrentEvents(convertedDate);
                    currentEvents = new JTable(eventModel);
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);

                    //adds event listener to currentEvents so event
panel is populated when an event panel is clicked
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                allEventPanel.setVisible(false);
                                currentCalendar.setVisible(false);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;

                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
```

```
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(15, 236, 403,
211);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });
                    //refills allStudents table with all students
                    getAllStudents();
                    allStudents = new JTable(allStudentModel);
                    allStudents.setFont(new Font("Tahoma", Font.PLAIN,
20));
```

```
                    allStudents.setRowSelectionAllowed(true);
                    allStudentPane = new JScrollPane(allStudents);
                    allStudentPane.setBounds(10, 10, 449, 113);
                    addStudentPopup.add(allStudentPane);
                }
            });
    }
    //searches for Event in Events table in database
    //if the event exists in the table, returns that event's ID
    //if not, returns -1
    public int searchEvent(String date)
    {
        int result = -1;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        String query = "SELECT * FROM Events WHERE Date='" + date +
"'";

        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement();)
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
                result = rs.getInt("ID");
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        return result;
    }

    //converts integers into YYYY-MM-DD format
    public String convertDate(int year, int month, int day)
    {
        String newMonth = "0";
        String newDay = "0";

        if((month - 10) < 0)
        {
            newMonth += Integer.toString(month);
        }
        else
        {
```

```
            newMonth = Integer.toString(month);
        }

        if((day - 10) < 0)
        {
            newDay += Integer.toString(day);
        }
        else
        {
            newDay = Integer.toString(day);
        }
        return year + "-" + newMonth + "-" + newDay;
    }

    //gets all rows from EventStudents where EventID = eventID
    //then adds the last and first names of all Students whose IDs
match the studentIDs from the selected rows
    //to StudentModel
    public int fillAttendingStudents(int eventID)
    {
        studentModel = new DefaultTableModel();
        studentModel.addColumn("");
        studentModel.addColumn("Last Name");
        studentModel.addColumn("First Name");
        int studentCount = 0;

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        String query = "SELECT Students.LastName, Students.FirstName
FROM EventStudents JOIN Students ON EventStudents.eventID=" + eventID
+ " AND eventStudents.studentID=Students.ID";

        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
                studentCount++;
                String[] result = {Integer.toString(studentCount),
rs.getString("LastName"),rs.getString("FirstName")};
                studentModel.insertRow(studentModel.getRowCount(), re-
sult);
            }
            conn.close();
        }
```

```
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

        return studentCount;
    }

    //deletes event from database
    public void removeEvent(int id)
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();

        String query = "DELETE FROM Events WHERE ID=" + id;

        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

    }

    //removes all rows from EventStudents where EventID = id
    //essentially removes all students who would attend the event
    public void removeStudents(int id)
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();

        String removeStudentQuery = "DELETE FROM EventStudents WHERE
EventID=" + id;

        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
```

```
            stat.execute(removeStudentQuery);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

    //gets data from rows in Events where the Date is the same as date
    //and initializes eventModel with the Date and Name from every row
    //if date is "Show All", then all rows in Events are selected
    public void fillCurrentEvents(String date)
    {
        String query;
        eventModel = new DefaultTableModel();
        eventModel.addColumn("Date");
        eventModel.addColumn("Name");

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        if(date.equals("Show All"))
        {
            query = "SELECT * FROM Events ORDER BY DATE";
        }
        else
        {
            query = "SELECT * FROM Events WHERE Date='" + date + "'
ORDER BY Date";
        }

        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
                String[] result = {rs.getString("Date"),
rs.getString("Name")};
                eventModel.insertRow(eventModel.getRowCount(), re-
sult);
            }
            conn.close();
        }
```

```
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

    //gets all rows from Students
    //and initializes allStudentModel with ID, Last Name, and First
Name from every row
    public void getAllStudents()
    {
        allStudentModel = new DefaultTableModel();
        allStudentModel.addColumn("Last Name");
        allStudentModel.addColumn("First Name");

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        String query = "SELECT * FROM Students";

        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
                Object[] data = {rs.getString("Last-
Name"),rs.getString("FirstName")};
                allStudentModel.insertRow(allStudentModel.getRow-
Count(),data);
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }
}
```

AdminViewHomepage Class

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.JTextArea;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.border.LineBorder;

import org.sqlite.SQLiteDataSource;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class AdminViewHomepage extends JFrame
{
    private JPanel contentPane;
    private JTextField newContactInfoTxtFld;
    private Database homeBase;
    private JTextField enterUsernameTxtFld;
    private JTextField enterPasswordTxtFld;
    private JTextField enterEmailTxtfld;

    /**
     * Launch the application.
     */
    public static void main(String[] args)
    {
        EventQueue.invokeLater(new Runnable()
```

```
            {
                public void run()
                {
                    try
                    {
                        AdminViewHomepage frame = new AdminViewHomep-
age();
                        frame.setVisible(true);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                }
            });
    }

    /**
     * Create the frame.
     * @throws SQLException
     */
    public AdminViewHomepage()
    {
        homeBase = new Database(); //creates new Database object

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 921, 809);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        final JTextArea FAQTxtBox = new JTextArea();
        FAQTxtBox.setEditable(false);
        FAQTxtBox.setFont(new Font("Tahoma", Font.PLAIN, 20));
        FAQTxtBox.setText("Frequently Asked Questions");
        FAQTxtBox.setBounds(27, 175, 748, 455);
        contentPane.add(FAQTxtBox);

        //creates errorPopup
        final JPanel errorPopup = new JPanel();
        errorPopup.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        errorPopup.setBounds(201, 191, 398, 139);
        contentPane.add(errorPopup);
        errorPopup.setLayout(null);
        errorPopup.setVisible(false);
```

```
        JLabel errorText = new JLabel("Please make sure all fields
have entries");
        errorText.setHorizontalAlignment(SwingConstants.CENTER);
        errorText.setFont(new Font("Tahoma", Font.PLAIN, 20));
        errorText.setBounds(0, 10, 396, 41);
        errorPopup.add(errorText);

        JButton closeErrorBtn = new JButton("Close");
        closeErrorBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    errorPopup.setVisible(false);
                }
            });
        closeErrorBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        closeErrorBtn.setBackground(new Color(0, 206, 209));
        closeErrorBtn.setBounds(134, 72, 115, 41);
        errorPopup.add(closeErrorBtn);

        //creates the popup for creating new admins
        JPanel createAdminPopup = new JPanel();
        createAdminPopup.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        createAdminPopup.setBounds(260, 340, 372, 343);
        contentPane.add(createAdminPopup);
        createAdminPopup.setLayout(null);
        createAdminPopup.setVisible(false);

        JLabel createAdminLabel = new JLabel("Create Admin");
        createAdminLabel.setFont(new Font("Tahoma", Font.PLAIN, 20));
        createAdminLabel.setBounds(112, 10, 194, 42);
        createAdminPopup.add(createAdminLabel);

        enterUsernameTxtFld = new JTextField();
        enterUsernameTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        enterUsernameTxtFld.setBounds(10, 79, 337, 42);
        createAdminPopup.add(enterUsernameTxtFld);
        enterUsernameTxtFld.setColumns(10);

        JLabel enterUsername = new JLabel("Enter Username");
        enterUsername.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterUsername.setBounds(10, 43, 337, 34);
        createAdminPopup.add(enterUsername);
```

```
        JLabel enterPassword = new JLabel("Enter Password");
        enterPassword.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterPassword.setBounds(10, 128, 337, 34);
        createAdminPopup.add(enterPassword);

        enterPasswordTxtFld = new JTextField();
        enterPasswordTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        enterPasswordTxtFld.setBounds(10, 159, 337, 42);
        createAdminPopup.add(enterPasswordTxtFld);
        enterPasswordTxtFld.setColumns(10);

        JLabel enterEmailLbl = new JLabel("Enter Email");
        enterEmailLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterEmailLbl.setBounds(10, 211, 318, 34);
        createAdminPopup.add(enterEmailLbl);

        enterEmailTxtfld = new JTextField();
        enterEmailTxtfld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterEmailTxtfld.setBounds(10, 242, 337, 42);
        createAdminPopup.add(enterEmailTxtfld);
        enterEmailTxtfld.setColumns(10);

        //if all fields are correct, creates new admin
        //else displays error
        JButton saveAdminBtn = new JButton("Save");
        saveAdminBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    String username = enterUsernameTxtFld.getText();
                    String password = enterPasswordTxtFld.getText();
                    String email = enterEmailTxtfld.getText();
                    if(username.equals("")||pass-
word.equals("")||email.equals(""))
                    {
                        errorPopup.setVisible(true);
                    }
                    else
                    {
                        Account newAccount = new Account(username,
password, email, "A");
                        newAccount.createAdmin();
                        createAdminPopup.setVisible(false);
                        FAQTxtBox.setVisible(true);
                    }
                }
```

```
            });
        saveAdminBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        saveAdminBtn.setBackground(new Color(0, 206, 209));
        saveAdminBtn.setBounds(252, 293, 95, 37);
        createAdminPopup.add(saveAdminBtn);

        JButton adminCloseBtn =new JButton("Close");
        adminCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    createAdminPopup.setVisible(false);
                    FAQTxtBox.setVisible(true);
                }
            });
        adminCloseBtn.setBackground(new Color(0, 206, 209));
        adminCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        adminCloseBtn.setBounds(10, 294, 142, 34);
        createAdminPopup.add(adminCloseBtn);

        final JLabel contactInfo = new JLabel("Contact Information");
        contactInfo.setFont(new Font("Tahoma", Font.ITALIC, 20));
        contactInfo.setHorizontalAlignment(SwingConstants.LEFT);
        contactInfo.setBounds(355, 95, 511, 37);
        contentPane.add(contactInfo);

        //when this window opens, set the Contact Info and FAQ to be
the values from the Home database
        addWindowListener(new WindowAdapter() {
                @Override
                public void windowOpened(WindowEvent e) {
                    //gets SQLiteDataSource from JapaneseClub
                    SQLiteDataSource source = homeBase.getDS();
                    String query = "SELECT * FROM Home";

                    //Creates connection to JapaneseClub and tries to
execute query
                    try(Connection conn = source.getConnection();
                    Statement stat = conn.createStatement(); )
                    {
                        ResultSet rs = stat.executeQuery(query);
                        contactInfo.setText(rs.getString("Contact"));
                        FAQTxtBox.setText(rs.getString("FAQ"));
                        conn.close();
                    }
                    catch(SQLException k)
                    {
```

```
                    k.printStackTrace();
                    System.exit(0);
                }
            }
        });

        final JPanel newFAQPopup = new JPanel();
        newFAQPopup.setBounds(27, 175, 778, 497);
        contentPane.add(newFAQPopup);
        newFAQPopup.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        newFAQPopup.setLayout(null);
        newFAQPopup.setVisible(false);

        JButton FAQCloseBtn = new JButton("Close");
        FAQCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    newFAQPopup.setVisible(false);
                    FAQTxtBox.setVisible(true);
                }
            });
        FAQCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        FAQCloseBtn.setBackground(new Color(0, 206, 209));
        FAQCloseBtn.setBounds(566, 439, 202, 45);
        newFAQPopup.add(FAQCloseBtn);

        final JTextPane newText = new JTextPane();
        newText.setBounds(10, 76, 758, 353);
        newFAQPopup.add(newText);
        newText.setFont(new Font("Tahoma", Font.PLAIN, 20));

        //if all fields are correct, updates Home with new values for
FAQ
        //else displays error
        JButton FAQSaveBtn = new JButton("Save Changes");
        FAQSaveBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    String info = newText.getText();
                    if(info.equals(""))
                        errorPopup.setVisible(true);
                    else
                    {
                        FAQTxtBox.setText(info);
                        //gets SQLiteDataSource from JapaneseClub
                        SQLiteDataSource source = homeBase.getDS();
```

```
                        String query = "UPDATE Home SET FAQ = '" +
info + "' WHERE ID=1";

                        //Creates connection to JapaneseClub and tries
to execute query
                        try(Connection conn = source.getConnection();
                        Statement stat = conn.createStatement(); )
                        {
                            stat.execute(query);
                            conn.close();
                        }
                        catch(SQLException i)
                        {
                            i.printStackTrace();
                            System.exit(0);
                        }
                        newFAQPopup.setVisible(false);
                        FAQTxtBox.setVisible(true);
                    }
                }
            });
        FAQSaveBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        FAQSaveBtn.setBackground(new Color(0, 206, 209));
        FAQSaveBtn.setBounds(20, 439, 202, 45);
        newFAQPopup.add(FAQSaveBtn);

        JLabel enterNewTxtLbl = new JLabel("Enter New Text");
        enterNewTxtLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterNewTxtLbl.setBounds(10, 24, 259, 31);
        newFAQPopup.add(enterNewTxtLbl);

        JButton logoutBtn = new JButton("Log-Out");
        logoutBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    LogoutScreen logout = new LogoutScreen();
                    logout.setVisible(true);
                    setVisible(false);
                }
            });
        logoutBtn.setBackground(new Color(0, 206, 209));
        logoutBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        logoutBtn.setBounds(27, 704, 159, 49);
        contentPane.add(logoutBtn);
```

```
JButton calendarBtn =    new JButton("Go to Calendar");
calendarBtn.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            AdminViewCalendar cal = new AdminViewCalendar();
            cal.setVisible(true);
            setVisible(false);
        }
    });
calendarBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
calendarBtn.setBackground(new Color(0, 206, 209));
calendarBtn.setBounds(402, 704, 270, 49);
contentPane.add(calendarBtn);

JButton editFAQBtn = new JButton("Edit");
editFAQBtn.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            newFAQPopup.setVisible(true);
            FAQTxtBox.setVisible(false);
            newText.setText(FAQTxtBox.getText());
        }
    });
editFAQBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
editFAQBtn.setBackground(new Color(64, 224, 208));
editFAQBtn.setBounds(796, 173, 89, 33);
contentPane.add(editFAQBtn);

JButton rosterBtn = new JButton("Roster");
rosterBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            AdminViewRoster roster = new AdminViewRoster();
            roster.setVisible(true);
            setVisible(false);
        }
    });
rosterBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
rosterBtn.setBackground(new Color(0, 206, 209));
rosterBtn.setBounds(218, 704, 150, 49);
contentPane.add(rosterBtn);

JButton createAdminBtn = new JButton("Create Admin");
createAdminBtn.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
```

```
                createAdminPopup.setVisible(true);
                FAQTxtBox.setVisible(false);
            }
        });
        createAdminBtn.setBackground(new Color(0, 206, 209));
        createAdminBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        createAdminBtn.setBounds(694, 704, 191, 49);
        contentPane.add(createAdminBtn);

        final JPanel contactInfoEdit = new JPanel();
        contactInfoEdit.setBounds(479, 10, 418, 156);
        contentPane.add(contactInfoEdit);
        contactInfoEdit.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        contactInfoEdit.setLayout(null);
        contactInfoEdit.setVisible(false);

        JButton editContactInfoBtn = new JButton("Edit");
        editContactInfoBtn.setBackground(new Color(64, 224, 208));
        editContactInfoBtn.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editContactInfoBtn.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                    contactInfoEdit.setVisible(true);
                    editContactInfoBtn.setVisible(false);
                    contactInfo.setVisible(false);
                    newContactInfoTxtFld.setText(con-
tactInfo.getText());
                }
            });
        editContactInfoBtn.setBounds(600, 97, 89, 33);
        contentPane.add(editContactInfoBtn);

        newContactInfoTxtFld = new JTextField();
        newContactInfoTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        newContactInfoTxtFld.setBounds(27, 49, 353, 47);
        contactInfoEdit.add(newContactInfoTxtFld);
        newContactInfoTxtFld.setColumns(10);

        JButton closeContactBtn = new JButton("Close");
        closeContactBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
```

```
                    //closes the edit contact info window and makes
the buttons and label visible
                    contactInfoEdit.setVisible(false);
                    editContactInfoBtn.setVisible(true);
                    contactInfo.setVisible(true);
                }
            });
        closeContactBtn.setBackground(new Color(0, 206, 209));
        closeContactBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        closeContactBtn.setBounds(27, 109, 114, 33);
        contactInfoEdit.add(closeContactBtn);

        //if all fields are correct, saves new contact information
into Home
        //else displays error
        JButton contactSaveChanges = new JButton("Save Changes");
        contactSaveChanges.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    String info = newContactInfoTxtFld.getText();
                    if(info.equals(""))
                        errorPopup.setVisible(true);
                    else
                    {
                        contactInfo.setText(info);

                        //gets SQLiteDataSource from JapaneseClub
                        SQLiteDataSource source = homeBase.getDS();
                        String query = "UPDATE Home SET Contact = '" +
info + "' WHERE ID=1";

                        //Creates connection to JapaneseClub and tries
to execute query
                        try(Connection conn = source.getConnection();
                        Statement stat = conn.createStatement(); )
                        {
                            stat.execute(query);
                            conn.close();
                        }
                        catch(SQLException m)
                        {
                            m.printStackTrace();
                            System.exit(0);
                        }
                        contactInfoEdit.setVisible(false);
                        editContactInfoBtn.setVisible(true);
```

```
                        contactInfo.setVisible(true);
                    }
                }
            });
        contactSaveChanges.setFont(new Font("Tahoma", Font.PLAIN,
20));
        contactSaveChanges.setBackground(new Color(0, 206, 209));
        contactSaveChanges.setBounds(218, 106, 162, 42);
        contactInfoEdit.add(contactSaveChanges);

        JLabel enterNewInfoLbl = new JLabel("Enter New Contact Infor-
mation");
        enterNewInfoLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        enterNewInfoLbl.setBounds(27, 10, 285, 25);
        contactInfoEdit.add(enterNewInfoLbl);

        JLabel jpnClubLbl = new JLabel("Japanese Club");
        jpnClubLbl.setHorizontalAlignment(SwingConstants.CENTER);
        jpnClubLbl.setFont(new Font("Tahoma", Font.PLAIN, 40));
        jpnClubLbl.setBounds(183, 28, 489, 49);
        contentPane.add(jpnClubLbl);

        JLabel contactInfoStartLbl = new JLabel("Club Contact Info:
");
        contactInfoStartLbl.setFont(new Font("Tahoma", Font.ITALIC,
20));
        contactInfoStartLbl.setBounds(162, 99, 230, 29);
        contentPane.add(contactInfoStartLbl);
    }
}
```

AdminViewRoster Class

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;

import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;
import java.awt.event.ActionEvent;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.time.Year;

import javax.swing.JTextField;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableModel;

import org.sqlite.SQLiteDataSource;

import javax.swing.JComboBox;
import java.util.*;

public class AdminViewRoster extends JFrame {

    private JPanel contentPane;
    private DefaultTableModel studentModel;
    private JTable studentDataTable;
    private JTextField lastNameTxtFld;
    private JTextField firstNameTxtFld;
    private JTextField gradeTxtFld;
```

```
    private JTextField schoolYearTxtFld;
    private JTextField editLastNameTxtFld;
    private JTextField editFirstNameTxtFld;
    private JTextField editGraduationYearTxtFld;
    private JTextField editSchoolYearTxtFld;
    private Database studentBase;
    private int currentYear;
    private JScrollPane scrollPane;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        AdminViewRoster frame = new AdminViewRoster();
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
    }

    /**
     * Create the frame.
     */
    public AdminViewRoster() {
        currentYear = Year.now().getValue();
        studentBase = new Database();
        contentPane = new JPanel();
        initializeDataTable();

        Integer[] schoolYears = fillYears(2020);

        final JComboBox<Object> years = new JComboBox<Ob-
ject>(schoolYears);
        years.setFont(new Font("Tahoma", Font.PLAIN, 20));

        final JPanel addStudentsPopup = new JPanel();
        addStudentsPopup.setBounds(21, 92, 310, 344);
        contentPane.add(addStudentsPopup);
        addStudentsPopup.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        addStudentsPopup.setVisible(false);
```

```
        addStudentsPopup.setLayout(null);

        JPanel removeStudentPopup = new JPanel();
        removeStudentPopup.setBounds(711, 410, 224, 90);
        contentPane.add(removeStudentPopup);
        removeStudentPopup.setBorder(new LineBorder(new Color(0, 0,
0), 2));
        removeStudentPopup.setLayout(null);
        removeStudentPopup.setVisible(false);

        JLabel removeStudentLabel = new JLabel("Are you sure you
want");
        removeStudentLabel.setFont(new Font("Tahoma", Font.PLAIN,
20));
        removeStudentLabel.setBounds(10, 0, 236, 33);
        removeStudentPopup.add(removeStudentLabel);

        JLabel removeStudentLabel2 = new JLabel("to remove this stu-
dent?");
        removeStudentLabel2.setFont(new Font("Tahoma", Font.PLAIN,
20));
        removeStudentLabel2.setBounds(10, 25, 236, 33);
        removeStudentPopup.add(removeStudentLabel2);

        //creates edit studentPopup
        final JPanel editStudentsPopup = new JPanel();
        editStudentsPopup.setBounds(340, 188, 455, 364);
        contentPane.add(editStudentsPopup);
        editStudentsPopup.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        editStudentsPopup.setLayout(null);
        editStudentsPopup.setVisible(false);

        //removes row from Students where ID = studentID
        JButton removeStudentsYesBtn = new JButton("Yes");
        removeStudentsYesBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    int rowNum = studentDataTable.getSelectedRow();
                    String last = (String)studentDataTable.get-
ValueAt(rowNum,0);
                    String first = (String)studentDataTable.get-
ValueAt(rowNum, 1);
                    int year = (Integer)studentDataTable.get-
ValueAt(rowNum, 2);
```

```
                    int graduationYear = (Integer)student-
DataTable.getValueAt(rowNum, 3);
                    Student newStudent = new Stu-
dent(last,first,year,graduationYear);
                    int studentID = newStudent.searchStudent(last,
first);
                    newStudent.deleteStudent(studentID);
                    //reinitializes studentDataTable to have all stu-
dents except deleted students
                    initializeDataTable();
                    fillStudentMatrix();
                    studentDataTable = new JTable(studentModel);
                    studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentDataTable.setRowSelectionAllowed(true);
                    studentDataTable.addMouseListener(new MouseA-
dapter() {
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                scrollPane.setVisible(false);
                                addStudentsPopup.setVisible(false);
                                years.setVisible(false);
                                editStudentsPopup.setVisible(true);
                                int rowNum = studentDataTable.get-
SelectedRow();
                                editLastNameTxtFld.setText((String)
studentDataTable.getValueAt(rowNum,0));
                                editFirstNam-
eTxtFld.setText((String)studentDataTable.getValueAt(rowNum, 1));
                                editGradua-
tionYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,2)));
                                ed-
itSchoolYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,3)));

                            }
                        });
                    scrollPane = new JScrollPane(studentDataTable);
                    scrollPane.setBounds(10, 92, 925, 460);
                    contentPane.add(scrollPane);
                    removeStudentPopup.setVisible(false);
                    editStudentsPopup.setVisible(false);
```

```
            }
        });
        removeStudentsYesBtn.setFont(new Font("Tahoma", Font.PLAIN,
20));
        removeStudentsYesBtn.setBackground(new Color(0, 206, 209));
        removeStudentsYesBtn.setBounds(20, 56, 73, 23);
        removeStudentPopup.add(removeStudentsYesBtn);

        JButton removeStudentNoBtn = new JButton("No");
        removeStudentNoBtn.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                removeStudentPopup.setVisible(false);
            }
        });
        removeStudentNoBtn.setFont(new Font("Tahoma", Font.PLAIN,
20));
        removeStudentNoBtn.setBackground(new Color(0, 206, 209));
        removeStudentNoBtn.setBounds(128, 56, 73, 23);
        removeStudentPopup.add(removeStudentNoBtn);

        editLastNameTxtFld = new JTextField();
        editLastNameTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editLastNameTxtFld.setColumns(10);
        editLastNameTxtFld.setBounds(21, 81, 419, 31);
        editStudentsPopup.add(editLastNameTxtFld);

        editFirstNameTxtFld = new JTextField();
        editFirstNameTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editFirstNameTxtFld.setColumns(10);
        editFirstNameTxtFld.setBounds(21, 145, 419, 31);
        editStudentsPopup.add(editFirstNameTxtFld);

        editGraduationYearTxtFld = new JTextField();
        editGraduationYearTxtFld.setFont(new Font("Tahoma",
Font.PLAIN, 20));
        editGraduationYearTxtFld.setColumns(10);
        editGraduationYearTxtFld.setBounds(21, 209, 419, 31);
        editStudentsPopup.add(editGraduationYearTxtFld);

        editSchoolYearTxtFld = new JTextField();
        editSchoolYearTxtFld.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editSchoolYearTxtFld.setColumns(10);
```

```
        editSchoolYearTxtFld.setBounds(21, 280, 419, 31);
        editStudentsPopup.add(editSchoolYearTxtFld);

        //reinitializes studentDataTable with all values from all rows
of Students
        JButton showAllBtn =new JButton("Show All");
        showAllBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    initializeDataTable();
                    fillStudentMatrix();
                    studentDataTable = new JTable(studentModel);
                    studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentDataTable.setRowSelectionAllowed(true);
                    //adds event listener that opens the edit student
pop-up and populates it with values in table
                    studentDataTable.addMouseListener(new MouseA-
dapter() {
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                editStudentsPopup.setVisible(true);
                                int rowNum = studentDataTable.get-
SelectedRow();
                                editLastNameTxtFld.setText((String)
studentDataTable.getValueAt(rowNum,0));
                                editFirstNam-
eTxtFld.setText((String)studentDataTable.getValueAt(rowNum, 1));
                                editGradua-
tionYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,2)));
                                ed-
itSchoolYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,3)));

                            }
                        });
                    scrollPane = new JScrollPane(studentDataTable);
                    scrollPane.setBounds(10, 92, 925, 460);

                    contentPane.add(scrollPane);
                }
            });
        showAllBtn.setBackground(new Color(0, 206, 209));
        showAllBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
```

```
        showAllBtn.setBounds(550, 36, 156, 32);
        contentPane.add(showAllBtn);

        //allows user to select year from dropdown
        //reinitializes studentDataTable with rows from Students where
membership year matches the selected year
        years.addItemListener(new ItemListener()
            {
                public void itemStateChanged(ItemEvent e)
                {
                    int year = (int)years.getSelectedItem();
                    initializeDataTable();
                    loadStudentsByYear(year);
                    studentDataTable = new JTable(studentModel);
                    studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentDataTable.setRowSelectionAllowed(true);
                    //adds event listener that opens the edit student
pop-up and populates it with values in table
                    studentDataTable.addMouseListener(new MouseA-
dapter() {
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                editStudentsPopup.setVisible(true);
                            }
                        });
                    scrollPane = new JScrollPane(studentDataTable);
                    scrollPane.setBounds(10, 92, 925, 460);
                    contentPane.add(scrollPane);
                    years.setVisible(false);
                    scrollPane.setVisible(true);
                }
            });

        years.setBackground(new Color(0, 206, 209));
        years.setBounds(813, 92, 101, 33);
        contentPane.add(years);
        years.setVisible(false);

        final JPanel addStudentsErrorPopup =new JPanel();
        addStudentsErrorPopup.setBorder(new LineBorder(new Color(0, 0,
0), 2));
        addStudentsErrorPopup.setBounds(475, 99, 305, 98);
        contentPane.add(addStudentsErrorPopup);
        addStudentsErrorPopup.setVisible(false);
```

```
        addStudentsErrorPopup.setLayout(null);

        JLabel addStudentsErrorText = new JLabel("Please fill out all
fields correctly");
        addStudentsErrorText.setBounds(11, 7, 284, 25);
        addStudentsErrorText.setFont(new Font("Tahoma", Font.PLAIN,
20));
        addStudentsErrorPopup.add(addStudentsErrorText);

        JButton addStudentsErrorCloseBtn = new JButton("Close");
        addStudentsErrorCloseBtn.addActionListener(new Action-
Listener() {
                public void actionPerformed(ActionEvent e) {
                }
            });
        addStudentsErrorCloseBtn.setBounds(112, 43, 81, 33);
        addStudentsErrorCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    addStudentsErrorPopup.setVisible(false);
                }
            });
        addStudentsErrorCloseBtn.setBackground(new Color(0, 206,
209));
        addStudentsErrorCloseBtn.setFont(new Font("Tahoma",
Font.PLAIN, 20));
        addStudentsErrorPopup.add(addStudentsErrorCloseBtn);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 958, 655);
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel rosterLbl = new JLabel("Club Member Roster");
        rosterLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
        rosterLbl.setBounds(21, 37, 285, 23);
        contentPane.add(rosterLbl);

        JButton homeBtn = new JButton("Home");
        homeBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
```

```
                AdminViewHomepage home = new AdminViewHomepage();
                home.setVisible(true);
                setVisible(false);
            }
        });
        homeBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        homeBtn.setBackground(new Color(0, 206, 209));
        homeBtn.setBounds(10, 566, 208, 41);
        contentPane.add(homeBtn);

        JButton calendarBtn = new JButton("Calendar");
        calendarBtn.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                AdminViewCalendar calendar = new AdminViewCalen-
dar();
                calendar.setVisible(true);
                setVisible(false);
            }
        });
        calendarBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        calendarBtn.setBackground(new Color(0, 206, 209));
        calendarBtn.setBounds(376, 566, 193, 41);
        contentPane.add(calendarBtn);

        JButton logoutBtn = new JButton("Log-Out");
        logoutBtn.addMouseListener(new MouseAdapter()
        {
            @Override
            public void mouseClicked(MouseEvent e)
            {
                LogoutScreen log = new LogoutScreen();
                log.setVisible(true);
                setVisible(false);
            }
        });
        logoutBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        logoutBtn.setBackground(new Color(0, 206, 209));
        logoutBtn.setBounds(721, 562, 214, 41);
        contentPane.add(logoutBtn);

        JButton loadStudentsBtn = new JButton("Filter By Year");
        loadStudentsBtn.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e)
            {
```

```
            years.setVisible(true);
            addStudentsPopup.setVisible(false);
            editStudentsPopup.setVisible(false);
        }
    });
loadStudentsBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
loadStudentsBtn.setBackground(new Color(0, 206, 209));
loadStudentsBtn.setBounds(732, 36, 203, 32);
contentPane.add(loadStudentsBtn);

JButton addStudentsBtn = new JButton("Add");
addStudentsBtn.addMouseListener(new MouseAdapter()
    {
        @Override
        public void mouseClicked(MouseEvent e)
        {
            addStudentsPopup.setVisible(true);
            editStudentsPopup.setVisible(false);
            years.setVisible(false);
        }
    });
addStudentsBtn.setBackground(new Color(0, 206, 209));
addStudentsBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
addStudentsBtn.setBounds(349, 36, 156, 32);
contentPane.add(addStudentsBtn);

JLabel editStudentsLbl = new JLabel("Edit Student");
editStudentsLbl.setFont(new Font("Tahoma", Font.PLAIN, 25));
editStudentsLbl.setBounds(155, 10, 201, 31);
editStudentsPopup.add(editStudentsLbl);

JButton removeStudentsBtn = new JButton("Remove Student");
removeStudentsBtn.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e)
        {
            removeStudentPopup.setVisible(true);
        }
    });
removeStudentsBtn.setFont(new Font("Tahoma", Font.PLAIN, 15));
removeStudentsBtn.setBackground(new Color(0, 206, 209));
removeStudentsBtn.setBounds(287, 325, 153, 31);
editStudentsPopup.add(removeStudentsBtn);

JLabel editStudentLastLbl = new JLabel("Student Last Name");
```

```
        editStudentLastLbl.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editStudentLastLbl.setBounds(21, 52, 259, 19);
        editStudentsPopup.add(editStudentLastLbl);

        JLabel editStudentFirstLbl = new JLabel("Student First Name");
        editStudentFirstLbl.setFont(new Font("Tahoma", Font.PLAIN,
20));
        editStudentFirstLbl.setBounds(21, 122, 259, 19);
        editStudentsPopup.add(editStudentFirstLbl);

        JLabel editStudentGraduationYear = new JLabel("Graduation
Year");
        editStudentGraduationYear.setFont(new Font("Tahoma",
Font.PLAIN, 20));
        editStudentGraduationYear.setBounds(21, 186, 259, 24);
        editStudentsPopup.add(editStudentGraduationYear);

        JLabel editStudentMembershipYearLbl = new JLabel("Membership
Year");
        editStudentMembershipYearLbl.setFont(new Font("Tahoma",
Font.PLAIN, 20));
        editStudentMembershipYearLbl.setBounds(21, 250, 259, 31);
        editStudentsPopup.add(editStudentMembershipYearLbl);

        JButton editStudentsCloseBtn = new JButton("Close");
        editStudentsCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    editStudentsPopup.setVisible(false);
                    scrollPane.setVisible(true);
                }
            });
        editStudentsCloseBtn.setBackground(new Color(0, 206, 209));
        editStudentsCloseBtn.setFont(new Font("Tahoma", Font.PLAIN,
15));
        editStudentsCloseBtn.setBounds(21, 324, 78, 33);
        editStudentsPopup.add(editStudentsCloseBtn);

        //if text fields are correct, existing student is updated with
new values
        //if not, errorPopup
        JButton editStudentsSaveBtn =new JButton("Save Changes");
        editStudentsSaveBtn.setBounds(120, 325, 137, 31);
        editStudentsPopup.add(editStudentsSaveBtn);
        editStudentsSaveBtn.addMouseListener(new MouseAdapter() {
```

```
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    String lastName = editLastNameTxtFld.getText();
                    String firstName = editFirstNameTxtFld.getText();
                    String grade = editGraduationYearTxtFld.getText();
                    String year = editSchoolYearTxtFld.getText();

                    if(lastName.equals("")||first-
Name.equals("")||grade.equals("")||year.equals(""))
                        addStudentsErrorPopup.setVisible(true);
                    else if(grade.length() != 4 || year.length() != 4)
                        addStudentsErrorPopup.setVisible(true);
                    else if(Integer.valueOf(year) > Integer.val-
ueOf(grade))
                        addStudentsErrorPopup.setVisible(true);
                    else
                    {
                        lastName = editLastNameTxtFld.getText();
                        firstName = editFirstNameTxtFld.getText();
                        grade = editGraduationYearTxtFld.getText();
                        year = editSchoolYearTxtFld.getText();
                        Student newStudent = new Student (last-
Name,firstName,Integer.valueOf(year),Integer.valueOf(grade));

                        int rowNum = studentDataTable.getSelect-
edRow();
                        int studentId = newStudent.searchStu-
dent((String)studentDataTable.getValueAt(rowNum,0), (String)student-
DataTable.getValueAt(rowNum, 1));
                        newStudent.updateMyLastName(lastName,studen-
tId);
                        newStudent.updateMyFirstName(firstName,studen-
tId);
                        newStudent.updateMyYear(Integer.val-
ueOf(year),studentId);
                        newStudent.updateMyGraduationYear(Integer.val-
ueOf(grade),studentId);
                        editStudentsPopup.setVisible(false);

                        //reinitializes studentDataTable with all stu-
dents, including student with updated values
                        initializeDataTable();
                        fillStudentMatrix();
                        studentDataTable = new JTable(studentModel);
```

```
                        studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                        studentDataTable.setRowSelectionAllowed(true);
                        //adds event listener that opens the edit stu-
dent pop-up and populates it with values in table
                        studentDataTable.addMouseListener(new MouseA-
dapter() {
                                @Override
                                public void mouseClicked(MouseEvent e)
                                {
                                    //makes everything else not visi-
ble to clear up visual clutter
                                    scrollPane.setVisible(false);
                                    addStudentsPopup.setVisi-
ble(false);
                                    years.setVisible(false);

                                    editStudentsPopup.setVisi-
ble(true);
                                    int rowNum = studentDataTable.get-
SelectedRow();
                                    editLastNam-
eTxtFld.setText((String) studentDataTable.getValueAt(rowNum,0));
                                    editFirstNam-
eTxtFld.setText((String)studentDataTable.getValueAt(rowNum, 1));
                                    editGradua-
tionYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,2)));
                                    ed-
itSchoolYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,3)));

                                }
                            });
                        scrollPane = new JScrollPane(student-
DataTable);
                        scrollPane.setBounds(10, 92, 925, 460);
                        contentPane.add(scrollPane);
                    }
                }
            });
        editStudentsSaveBtn.setBackground(new Color(0, 206, 209));
        editStudentsSaveBtn.setFont(new Font("Tahoma", Font.PLAIN,
15));
```

```
        //if studentDataTable exists, then add event handler that
shows editStudentsPopup when clicked
        if(studentDataTable != null)
        {

            studentDataTable.addMouseListener(new MouseAdapter() {
                    @Override
                    public void mouseClicked(MouseEvent e)
                    {
                        editStudentsPopup.setVisible(true);
                    }
                });

        }

        JLabel addStudentsLbl = new JLabel("Add Student");
        addStudentsLbl.setFont(new Font("Tahoma", Font.PLAIN, 28));
        addStudentsLbl.setBounds(72, 7, 165, 34);
        addStudentsPopup.add(addStudentsLbl);

        lastNameTxtFld = new JTextField();
        lastNameTxtFld.setBounds(33, 72, 243, 31);
        lastNameTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addStudentsPopup.add(lastNameTxtFld);
        lastNameTxtFld.setColumns(10);

        firstNameTxtFld = new JTextField();
        firstNameTxtFld.setBounds(33, 131, 243, 31);
        firstNameTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        firstNameTxtFld.setColumns(10);
        addStudentsPopup.add(firstNameTxtFld);

        gradeTxtFld = new JTextField();
        gradeTxtFld.setBounds(33, 188, 243, 31);
        gradeTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        gradeTxtFld.setColumns(10);
        addStudentsPopup.add(gradeTxtFld);

        schoolYearTxtFld = new JTextField();
        schoolYearTxtFld.setBounds(33, 249, 243, 31);
        schoolYearTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
        schoolYearTxtFld.setColumns(10);
        addStudentsPopup.add(schoolYearTxtFld);

        //if all fields are correct, adds new student to Students
        //if not, display error
```

```
        JButton addStudentSaveBtn = new JButton("Save Changes");
        addStudentSaveBtn.setBounds(119, 301, 157, 33);
        addStudentSaveBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    String lastName = lastNameTxtFld.getText();
                    String firstName = firstNameTxtFld.getText();
                    String grade = gradeTxtFld.getText();
                    String year = schoolYearTxtFld.getText();
                    if(lastName.equals("")||first-
Name.equals("")||grade.equals("")||year.equals(""))
                        addStudentsErrorPopup.setVisible(true);
                    else if(grade.length() != 4 || year.length() != 4)
                    {
                        addStudentsErrorPopup.setVisible(true);
                    }
                    else if(Integer.valueOf(year) > Integer.val-
ueOf(grade))
                        addStudentsErrorPopup.setVisible(true);
                    else
                    {
                        Student newStudent = new Student(lastName,
firstName, Integer.parseInt(year), Integer.parseInt(grade));
                        newStudent.enterStudent();
                        addStudentsPopup.setVisible(false);

                        //reinitializes studentDataTable with all stu-
dents, including student with updated values
                        initializeDataTable();
                        fillStudentMatrix();
                        studentDataTable = new JTable(studentModel);
                        studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                        studentDataTable.setRowSelectionAllowed(true);
                        studentDataTable.addMouseListener(new MouseA-
dapter() {
                                @Override
                                public void mouseClicked(MouseEvent e)
                                {
                                    //makes everything else not visi-
ble to clear up visual clutter
                                    scrollPane.setVisible(false);
                                    addStudentsPopup.setVisi-
ble(false);
```

```
                                    years.setVisible(false);

                                    editStudentsPopup.setVisi-
ble(true);
                                    int rowNum = studentDataTable.get-
SelectedRow();
                                    editLastNam-
eTxtFld.setText((String) studentDataTable.getValueAt(rowNum,0));
                                    editFirstNam-
eTxtFld.setText((String)studentDataTable.getValueAt(rowNum, 1));
                                    editGradua-
tionYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,2)));
                                    ed-
itSchoolYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,3)));

                                }
                            });
                        scrollPane = new JScrollPane(student-
DataTable);
                        scrollPane.setBounds(10, 92, 925, 460);
                        contentPane.add(scrollPane);
                    }
                }
            });
        addStudentSaveBtn.setBackground(new Color(0, 206, 209));
        addStudentSaveBtn.setFont(new Font("Tahoma", Font.PLAIN, 18));
        addStudentsPopup.add(addStudentSaveBtn);

        JButton addStudentsCloseBtn = new JButton("Close");
        addStudentsCloseBtn.setBounds(30, 301, 81, 33);
        addStudentsCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    addStudentsPopup.setVisible(false);
                    scrollPane.setVisible(true);
                }
            });
        addStudentsCloseBtn.setBackground(new Color(0, 206, 209));
        addStudentsCloseBtn.setFont(new Font("Tahoma", Font.PLAIN,
18));
        addStudentsPopup.add(addStudentsCloseBtn);

        JLabel addLastNameLbl = new JLabel("Student Last Name");
```

```
        addLastNameLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addLastNameLbl.setBounds(33, 51, 243, 17);
        addStudentsPopup.add(addLastNameLbl);

        JLabel addFirstName = new JLabel("Student First Name");
        addFirstName.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addFirstName.setBounds(33, 113, 243, 17);
        addStudentsPopup.add(addFirstName);

        JLabel addGraduationYear = new JLabel("Graduation Year");
        addGraduationYear.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addGraduationYear.setBounds(33, 172, 243, 17);
        addStudentsPopup.add(addGraduationYear);

        JLabel addMembershipYear = new JLabel("Membership Year");
        addMembershipYear.setFont(new Font("Tahoma", Font.PLAIN, 20));
        addMembershipYear.setBounds(33, 229, 243, 17);
        addStudentsPopup.add(addMembershipYear);

        //when the window is opened, initializes studentDataTable to
have all data from Students
        addWindowListener(new WindowAdapter() {
                @Override
                public void windowOpened(WindowEvent e)
                {
                    fillStudentMatrix();
                    studentDataTable = new JTable(studentModel);
                    studentDataTable.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentDataTable.setRowSelectionAllowed(true);
                    studentDataTable.addMouseListener(new MouseA-
dapter() {
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                //makes everything else not visible to
clear up visual clutter
                                scrollPane.setVisible(false);
                                addStudentsPopup.setVisible(false);
                                years.setVisible(false);
                                editStudentsPopup.setVisible(true);
                                int rowNum = studentDataTable.get-
SelectedRow();
                                editLastNameTxtFld.setText((String)
studentDataTable.getValueAt(rowNum,0));
```

```
                                editFirstNam-
eTxtFld.setText((String)studentDataTable.getValueAt(rowNum, 1));
                                editGradua-
tionYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,2)));
                                ed-
itSchoolYearTxtFld.setText(String.valueOf(studentDataTable.get-
ValueAt(rowNum,3)));

                            }
                        });
                    scrollPane = new JScrollPane(studentDataTable);
                    scrollPane.setBounds(10, 92, 925, 460);

                    contentPane.add(scrollPane);
                }
            });
    }

    /*For every row in the Students data table, fills studentModel
with a row of Objects
     * consisting of the ID, LastName, FirstName, Year, and Gradua-
tionYear gathered from Students.
     * The end result is studentModel being a filled DefaultTableModel
with the information from
     * every student currently registered in the database, which can
then be displayed with other
     * methods.
     */
    public void fillStudentMatrix()
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = studentBase.getDS();
        String query = "SELECT * FROM Students ORDER BY LastName";

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
                Object[] data = {rs.getString("Last-
Name"),rs.getString("FirstName"),rs.getInt("GraduationYear"),rs.get-
Int("Year")};
```

```
                studentModel.insertRow(studentModel.getRow-
Count(),data);
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

    //gets rows from Students where membership year = year
    public void loadStudentsByYear(int year)
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = studentBase.getDS();
        String query = "SELECT * FROM Students  WHERE YEAR='" + year +
"' ORDER BY LastName";

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
                Object[] data = {rs.getString("Last-
Name"),rs.getString("FirstName"),rs.getInt("GraduationYear"),rs.get-
Int("Year")};
                studentModel.insertRow(studentModel.getRow-
Count(),data);
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

    }

    //initializes studentModel with ID, Last Name, First Name, Gradua-
tion Year, and Membership Year columns
```

```
    public void initializeDataTable()
    {
        if(scrollPane != null)
        {
            scrollPane.removeAll();
            contentPane.remove(scrollPane);
        }
        studentModel = new DefaultTableModel();
        studentModel.addColumn("Last Name");
        studentModel.addColumn("First Name");
        studentModel.addColumn("Graduation Year");
        studentModel.addColumn("Membership Year");
    }

    //fills the years JComboBox with all years from starting years un-
til the current year
    public Integer[] fillYears(int startingYear)
    {
        //initializes an ArrayList the holds the values of the years
        ArrayList<Integer> pastYears = new ArrayList<Integer>();
        int current = currentYear; //contains the value of the current
year when roster opened
        //so long as the currentYear is greater than or equal to the
starting year
        //adds the year to pastYears, then increments current by -1
        while(current >= startingYear)
        {
            pastYears.add(current);
            current--;
        }

        int size = pastYears.size();
        //initializes the array that holds all of the values for year
        Integer[] yearList = new Integer[size];
        //copies values in pastYears into yearList array
        for(int i = 0; i < size; i++)
        {
            yearList[i] = pastYears.get(i);
        }

        return yearList;
    }
}
```

Database Class

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import org.sqlite.SQLiteDataSource;

public class Database
{
      private SQLiteDataSource ds;

      public Database()
      {
         try
         {
             ds = new SQLiteDataSource();
             ds.setUrl("jdbc:sqlite:JapaneseClub.db");
         }
         catch ( Exception e )
         {
             e.printStackTrace();
             System.exit(0);
         }
      }

      public SQLiteDataSource getDS()
      {
            return ds;
      }
}
```

Event Class

```
import org.sqlite.SQLiteDataSource;
import java.sql.*;

public class Event
{
    private String myDate;
    private String myName;
    private String myStart;
    private String myEnd;
    private String myLocation;
    private Database myEventBase;

    public Event(int year, int month, int day, String name, String
start,String end, String location)
    {
        myDate = year + "-" + month + "-" + day;
        myName = name;
        myStart = start;
        myEnd = end;
        myLocation = location;
        myEventBase = new Database();
    }

    public Event(String date, String name, String start, String end,
String location)
    {
        myDate = date;
        myName = name;
        myStart = start;
        myEnd = end;
        myLocation = location;
        myEventBase = new Database();
    }

    public String getDate()
    {
        return myDate;
    }

    public String getName() {
        return myName;
    }
```

```
    //updates value of myName and value of Name in Events
    public void setName(String name, int eventID) {
        myName = name;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myEventBase.getDS();
        String query = "UPDATE Events SET Name='" + myName + "' WHERE
ID=" + eventID;

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

    public String getStart() {
        return myStart;
    }

    //updates value of myStart and value of start time in Events
    public void setStart(String start, int eventID) {
        myStart = start;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myEventBase.getDS();
        String query = "UPDATE Events SET Start='" + myStart + "'
WHERE ID=" + eventID;

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
```

```
            System.exit(0);
        }
    }

    public String getEnd() {
        return myEnd;
    }

    public void setEnd(String end, int eventID) {
        myEnd = end;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myEventBase.getDS();
        String query = "UPDATE Events SET End='" + myEnd + "' WHERE
ID=" + eventID;

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

    }

    public String getLocation()
    {
        return myLocation;
    }

    //updates myLocation and Location in Events
    public void updateLocation(String location, int eventID)
    {
        myLocation = location;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myEventBase.getDS();
        String query = "UPDATE Events SET Location='" + myLocation +
"' WHERE ID=" + eventID;
```

```
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

    public void enterEvent()
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myEventBase.getDS();
        String query = "INSERT INTO Events (Date, Name, Start, End,
Location) VALUES ('" + myDate + "','" + myName + "','" + myStart +
"','" + myEnd + "','" + myLocation + "')";

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement();)
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }

}
```

JapanAcctCreation Class

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JTextPane;
import javax.swing.border.LineBorder;
import java.util.*;
import java.io.*;

	public class JapanAcctCreation extends JFrame {

	private JPanel contentPane;
	private JTextField usernameTxtFld;
	private JTextField passwordTxtFld;
	private JTextField emailTxtFld;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					JapanAcctCreation frame = new JapanAcctCre-
ation();
					frame.setVisible(true);
				} catch (Exception e)
```

```
                    {
                        e.printStackTrace();
                    }
                }
            });
        }

        /**
         * Create the frame.
         */
        public JapanAcctCreation()
        {

            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 630, 663);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            //no Username pop-up code
                        JPanel errorPopUp = new JPanel();
                        errorPopUp.setBorder(new LineBorder(new
Color(139, 0, 0), 2));
                        errorPopUp.setBounds(10, 156, 569, 286);
                        contentPane.add(errorPopUp);
                        errorPopUp.setLayout(null);

                        JLabel errorText = new JLabel("");
                        errorText.setFont(new Font("Tahoma", Font.PLAIN,
20));
                        errorText.setBounds(23, 21, 513, 166);
                        errorPopUp.add(errorText);

                        JButton errorCloseBtn = new JButton("Close");
                        errorCloseBtn.setBackground(new Color(0, 206,
209));
                        errorCloseBtn.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                        errorCloseBtn.setBounds(204, 215, 141, 45);
                        errorPopUp.add(errorCloseBtn);
                        errorCloseBtn.addMouseListener(new MouseAdapter()
                        {
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
```

```
                              errorPopUp.setVisible(false);
                        }
                  });

                  addWindowListener(new WindowAdapter()
                  {
                        @Override
                        public void windowOpened(WindowEvent e)
                        {
                              errorPopUp.setVisible(false);
                        }
                  });

            JLabel acctCreationLbl = new JLabel("Account Creation");
            acctCreationLbl.setHorizontalAlignment(SwingConstants.CEN-
TER);
            acctCreationLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
            acctCreationLbl.setBounds(133, 26, 307, 76);
            contentPane.add(acctCreationLbl);

            JLabel usernameInstructionsLbl = new JLabel("Please enter a
username.");
            usernameInstructionsLbl.setFont(new Font("Tahoma",
Font.PLAIN, 20));
            usernameInstructionsLbl.setBounds(10, 136, 521, 48);
            contentPane.add(usernameInstructionsLbl);

            usernameTxtFld = new JTextField();
            usernameTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
            usernameTxtFld.setBounds(10, 184, 569, 48);
            contentPane.add(usernameTxtFld);
            usernameTxtFld.setColumns(10);

            JLabel passwordInstructionsLbl = new JLabel("Please enter a
password.");
            passwordInstructionsLbl.setFont(new Font("Tahoma",
Font.PLAIN, 20));
            passwordInstructionsLbl.setBounds(10, 267, 461, 37);
            contentPane.add(passwordInstructionsLbl);

            passwordTxtFld = new JTextField();
            passwordTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
            passwordTxtFld.setBounds(10, 304, 569, 48);
            contentPane.add(passwordTxtFld);
            passwordTxtFld.setColumns(10);
```

```
            //if no fields are empty and the username and email are not
in use, creates Account
            //otherwise displays error
            JButton createAcctBtn = new JButton("Create Account");
            createAcctBtn.addMouseListener(new MouseAdapter()
            {
                  @Override
                  public void mouseClicked(MouseEvent e)
                  {
                        JapanLogin login = new JapanLogin();
                        String username = usernameTxtFld.getText();
                        String password = passwordTxtFld.getText();
                        String email = emailTxtFld.getText();
                        Account newAcct = new Account(username, password,
email, "N");

                        if(username.equals(""))
                        {
                              errorPopUp.setVisible(true);
                              errorText.setText("Need username to create
Account.");
                        }
                        else if(password.equals(""))
                        {
                              errorPopUp.setVisible(true);
                              errorText.setText("Need password to create
Account.");
                        }
                        else if(email.equals("") )
                        {
                              errorPopUp.setVisible(true);
                              errorText.setText("Need email to create Ac-
count.");
                        }
                        else
                        {
                              if(newAcct.usernameInUse() ||
newAcct.emailInUse())
                              {
                                    errorPopUp.setVisible(true);
                                    errorText.setText("This username or
email is already in use. Please try again.");
                              }
                              else
                              {
                                    newAcct.enterAccount();
```

```
                                    login.setVisible(true);
                                    setVisible(false);

                               }

                          }
                     }
          });
               createAcctBtn.setBackground(new Color(0, 206, 209));
               createAcctBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
               createAcctBtn.setBounds(10, 563, 217, 48);
               contentPane.add(createAcctBtn);

               emailTxtFld = new JTextField();
               emailTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
               emailTxtFld.setBounds(10, 422, 569, 48);
               contentPane.add(emailTxtFld);
               emailTxtFld.setColumns(10);

               JLabel lblPleaseEnterAn = new JLabel("Please enter an
email.");
               lblPleaseEnterAn.setFont(new Font("Tahoma", Font.PLAIN,
20));
               lblPleaseEnterAn.setBounds(10, 386, 461, 37);
               contentPane.add(lblPleaseEnterAn);

               JButton closeBtn = new JButton("Close");
               closeBtn.addMouseListener(new MouseAdapter() {
                    @Override
                    public void mouseClicked(MouseEvent e) {
                         JapanLogin login = new JapanLogin();
                         login.setVisible(true);
                         setVisible(false);
                    }
               });
               closeBtn.setBackground(new Color(0, 206, 209));
               closeBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
               closeBtn.setBounds(439, 562, 140, 51);
               contentPane.add(closeBtn);
          }
}
```

JapanLogin Class

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JPasswordField;
import javax.swing.border.EmptyBorder;
import javax.swing.JTextPane;
import java.awt.Font;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import net.miginfocom.swing.MigLayout;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.sql.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.*;
import javax.swing.JTextArea;
import javax.swing.border.LineBorder;
import java.awt.SystemColor;

public class JapanLogin extends JFrame
{
	private JPanel contentPane;
	private JTextField usernameTxtFld;
	private JPasswordField passwordTxtFld;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
```

```
                    try
                    {
                        JapanLogin frame = new JapanLogin();
                        frame.setVisible(true);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                }
            });
        }

        /**
         * Create the frame.
         */
        public JapanLogin()
        {
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(400, 400, 743, 533);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            getContentPane().setLayout(null);

            JPanel errorPopup = new JPanel();
            errorPopup.setBorder(new LineBorder(Color.BLACK, 2));
            errorPopup.setBackground(SystemColor.control);
            errorPopup.setBounds(177, 156, 330, 159);
            contentPane.add(errorPopup);
            errorPopup.setLayout(null);

            addWindowListener(new WindowAdapter()
            {
                @Override
                public void windowOpened(WindowEvent e)
                {
                    errorPopup.setVisible(false);
                }
            });

            JLabel errorMessage1 = new JLabel("Information entered is
incorrect.");
            errorMessage1.setFont(new Font("Tahoma", Font.PLAIN, 20));
            errorMessage1.setHorizontalAlignment(SwingConstants.CENTER);
            errorMessage1.setBounds(0, 10, 337, 37);
```

```
            errorPopup.add(errorMessage1);

            JLabel errorMessage2 = new JLabel("Please try again.");
            errorMessage2.setHorizontalAlignment(SwingConstants.CENTER);
            errorMessage2.setFont(new Font("Tahoma", Font.PLAIN, 20));
            errorMessage2.setBounds(-10, 42, 330, 42);
            errorPopup.add(errorMessage2);

            JButton errorCloseBtn = new JButton("Close");
            errorCloseBtn.addMouseListener(new MouseAdapter() {
                  @Override
                  public void mouseClicked(MouseEvent e) {
                        errorPopup.setVisible(false);
                  }
            });
            errorCloseBtn.setBackground(new Color(0, 206, 209));
            errorCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
            errorCloseBtn.setBounds(109, 106, 117, 33);
            errorPopup.add(errorCloseBtn);

            JLabel welcomeLbl = new JLabel("Welcome!");
            welcomeLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
            welcomeLbl.setBounds(266, 10, 151, 88);
            getContentPane().add(welcomeLbl);

            usernameTxtFld = new JTextField();
            usernameTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
            usernameTxtFld.setBounds(41, 141, 644, 50);
            getContentPane().add(usernameTxtFld);
            usernameTxtFld.setColumns(10);

            passwordTxtFld = new JPasswordField();
            passwordTxtFld.setFont(new Font("Tahoma", Font.PLAIN, 20));
            passwordTxtFld.setBounds(41, 269, 644, 50);
            getContentPane().add(passwordTxtFld);
            passwordTxtFld.setColumns(10);

            //if User with matching username, password, and email exists
in Users, checks if admin
            //if admin, show AdminHomePage
            //if not admin, show RegularUserHomePage
            //if does not exist, show error
            JButton loginBtn = new JButton("Log-In");
            loginBtn.addMouseListener(new MouseAdapter()
            {
                  @Override
```

```
                public void mouseClicked(MouseEvent e)
                {
                      String username = usernameTxtFld.getText();
                      String password = passwordTxtFld.getText();
                      Account logAcct = new Account(username, password,
"", "N");
                      if(logAcct.existsInDatabase())
                      {
                            if(logAcct.isAdmin())
                            {
                                  AdminViewHomepage homepage = new Ad-
minViewHomepage();
                                  homepage.setVisible(true);
                            }
                            else
                            {
                                  RegularUserHomepage home = new Regula-
rUserHomepage();
                                  home.setVisible(true);
                            }
                            setVisible(false);
                      }
                      else
                            errorPopup.setVisible(true);

                }
            });
            loginBtn.setBackground(new Color(0, 206, 209));
            loginBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
            loginBtn.setBounds(41, 417, 184, 45);
            getContentPane().add(loginBtn);

            JButton createAcctBtn = new JButton("Create Account");
            createAcctBtn.addMouseListener(new MouseAdapter()
            {
                  @Override
                  public void mouseClicked(MouseEvent e)
                  {
                        JapanAcctCreation creation = new JapanAcctCrea-
tion();
                        creation.setVisible(true); //setVisible (true);
                        setVisible(false);
                  }
            });
            createAcctBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
            createAcctBtn.setBackground(new Color(0, 206, 209));
```

```
            createAcctBtn.setBounds(344, 417, 341, 45);
            getContentPane().add(createAcctBtn);

            JLabel passwordLbl = new JLabel("Password");
            passwordLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
            passwordLbl.setBounds(41, 235, 139, 24);
            contentPane.add(passwordLbl);

            JLabel usernameLbl = new JLabel("Username");
            usernameLbl.setFont(new Font("Tahoma", Font.PLAIN, 20));
            usernameLbl.setBounds(41, 107, 113, 24);
            contentPane.add(usernameLbl);

        }
}
```

LogoutScreen Class

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextPane;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class LogoutScreen extends JFrame {

	private JPanel contentPane;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
				{
					LogoutScreen frame = new LogoutScreen();
					frame.setVisible(true);
					frame.setExtendedState(frame.getExtended-
State() | JFrame.MAXIMIZED_BOTH); //makes frame maximum size upon
opening
				}
				catch (Exception e)
				{
					e.printStackTrace();
				}
			}
		});
	}
```

```
	/**
	 * Create the frame.
	 */
	public LogoutScreen() {
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(100, 100, 744, 675);
		contentPane = new JPanel();
		contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
		setContentPane(contentPane);
		contentPane.setLayout(null);

		JButton btnNewButton = new JButton("Log-In");
		btnNewButton.addMouseListener(new MouseAdapter()
		{
			@Override
			public void mouseClicked(MouseEvent e)
			{
				JapanLogin login = new JapanLogin();
				login.setVisible(true);
				setVisible(false);
			}
		});
		btnNewButton.setBackground(new Color(0, 206, 209));
		btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 20));
		btnNewButton.setBounds(298, 410, 107, 33);
		contentPane.add(btnNewButton);

		JLabel logoutLbl = new JLabel("You Have Been Logged Out.");
		logoutLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
		logoutLbl.setBounds(159, 193, 421, 44);
		contentPane.add(logoutLbl);
	}
}
```

RegularUserCalendar Class

```
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import org.sqlite.SQLiteDataSource;

import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import com.toedter.calendar.JCalendar;
import java.beans.PropertyChangeListener;
import java.beans.PropertyChangeEvent;

public class RegularUserCalendar extends JFrame
{

    private JPanel contentPane;
    private DefaultTableModel studentModel;
    private JTable attendingStudents;
    private JScrollPane studentPane;
    private DefaultTableModel eventModel;
    private JTable currentEvents;
    private JScrollPane currentEventPanel;
    private Database eventBase;

    /**
     * Launch the application.
     */
```

```
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        RegularUserCalendar frame = new RegularUs-
erCalendar();
                        frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
    }

    /**
     * Create the frame.
     */
    public RegularUserCalendar()
    {
        eventBase = new Database(); //creates new Database object

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 643, 752);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel eventCalendarLbl = new JLabel("Event Calendar");
        eventCalendarLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
        eventCalendarLbl.setBounds(25, 64, 409, 49);
        contentPane.add(eventCalendarLbl);

        JButton homeBtn = new JButton("Home");
        homeBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    RegularUserHomepage home = new RegularUserHomep-
age();
                    home.setVisible(true);
                    setVisible(false);
                }
            });
        homeBtn.setBackground(new Color(0, 206, 209));
```

```
        homeBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        homeBtn.setBounds(23, 659, 273, 33);
        contentPane.add(homeBtn);

        JButton logoutBtn = new JButton("Log Out");
        logoutBtn.addMouseListener(new MouseAdapter()
            {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    LogoutScreen logout = new LogoutScreen();
                    logout.setVisible(true);
                    setVisible(false);
                }
            });
        logoutBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        logoutBtn.setBackground(new Color(0, 206, 209));
        logoutBtn.setBounds(353, 659, 265, 33);
        contentPane.add(logoutBtn);

        //creates panel which will list all of the events in Japa-
neseClub
        JPanel allEventPanel = new JPanel();
        allEventPanel.setBorder(new LineBorder(new Color(0, 0, 0),
2));
        allEventPanel.setBounds(23, 373, 595, 255);
        contentPane.add(allEventPanel);
        allEventPanel.setLayout(null);

        JLabel allEventLbl = new JLabel("Current Events");
        allEventLbl.setHorizontalAlignment(SwingConstants.CENTER);
        allEventLbl.setFont(new Font("Tahoma", Font.PLAIN, 30));
        allEventLbl.setBounds(0, 0, 595, 59);
        allEventPanel.add(allEventLbl);

        JPanel eventPanel =new JPanel();
        eventPanel.setBounds(83, 122, 428, 368);
        contentPane.add(eventPanel);
        eventPanel.setBorder(new LineBorder(new Color(0, 0, 0), 2));
        eventPanel.setLayout(null);
        eventPanel.setVisible(false);

        JLabel eventDate = new JLabel("Date");
        eventDate.setHorizontalAlignment(SwingConstants.CENTER);
        eventDate.setFont(new Font("Tahoma", Font.PLAIN, 25));
        eventDate.setBounds(55, 10, 323, 29);
```

```
        eventPanel.add(eventDate);

        JLabel eventName = new JLabel("Name");
        eventName.setHorizontalAlignment(SwingConstants.CENTER);
        eventName.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventName.setBounds(54, 49, 334, 29);
        eventPanel.add(eventName);

        JLabel eventTimes = new JLabel("From - Until");
        eventTimes.setHorizontalAlignment(SwingConstants.CENTER);
        eventTimes.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventTimes.setBounds(102, 88, 229, 25);
        eventPanel.add(eventTimes);

        JLabel eventLocation = new JLabel("Location");
        eventLocation.setHorizontalAlignment(SwingConstants.CENTER);
        eventLocation.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventLocation.setBounds(80, 123, 276, 32);
        eventPanel.add(eventLocation);

        JLabel attendingStudentsLbl = new JLabel("Attending Mem-
bers:");
        attendingStudentsLbl.setHorizontalAlignment(SwingCon-
stants.CENTER);
        attendingStudentsLbl.setFont(new Font("Tahoma", Font.PLAIN,
20));
        attendingStudentsLbl.setBounds(108, 168, 197, 29);
        eventPanel.add(attendingStudentsLbl);

        //initializes JCalendar
        final JCalendar currentCalendar = new JCalendar();
        currentCalendar.addPropertyChangeListener(new PropertyChange-
Listener() {
                public void propertyChange(PropertyChangeEvent evt) {
                    eventPanel.setVisible(false);
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    String convertedDate = convertDate(selectedYear,
selectedMonth,selectedDay);
                    if(currentEventPanel != null)
                    {
                        allEventPanel.remove(currentEventPanel);
```

```
                    }
                    fillCurrentEvents(convertedDate);
                    currentEvents = new JTable(eventModel);
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                currentCalendar.setVisible(false);
                                allEventPanel.setVisible(false);
                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;

                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
```

```
                                    conn.close();
                                }
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(32, 209, 378,
94);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);
                }
            });
        currentCalendar.getMonthChooser().getSpinner().setFont(new
Font("Tahoma", Font.PLAIN, 20));
        currentCalendar.getYearChooser().getSpinner().setFont(new
Font("Tahoma", Font.PLAIN, 20));
        currentCalendar.getMonthChooser().getSpinner().setBack-
ground(new Color(0, 206, 209));
        currentCalendar.getMonthChooser().getComboBox().setFont(new
Font("Tahoma", Font.PLAIN, 20));
```

```
        currentCalendar.setBounds(23, 130, 595, 206);
        contentPane.add(currentCalendar);

        JButton eventCloseBtn = new JButton("Close");
        eventCloseBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    studentPane.removeAll();
                    eventPanel.remove(studentPane);
                    eventPanel.setVisible(false);
                    currentCalendar.setVisible(true);
                    allEventPanel.setVisible(true);
                }
            });
        eventCloseBtn.setBackground(new Color(0, 206, 209));
        eventCloseBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        eventCloseBtn.setBounds(156, 325, 133, 29);
        eventPanel.add(eventCloseBtn);

        //when the window is opened, the JTable currentEvents is ini-
tialized with all events on the current day
        //in JapaneseClub. Then it is added to allEventPanel
        addWindowListener(new WindowAdapter() {
                @Override
                public void windowOpened(WindowEvent e) {
                    int selectedDay = currentCalendar.getDay-
Chooser().getDay();
                    int selectedMonth = currentCalen-
dar.getMonthChooser().getMonth() + 1;
                    int selectedYear = currentCalendar.getYear-
Chooser().getYear();
                    String convertedDate = convertDate(selectedYear,
selectedMonth,selectedDay);

                    currentEventPanel.removeAll();
                    allEventPanel.remove(currentEventPanel);
                    fillCurrentEvents(convertedDate);
                    currentEvents = new JTable(eventModel);
                    currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    currentEventPanel = new JScrollPane(currentE-
vents);
                    currentEventPanel.setBounds(10, 71, 575, 163);
                    allEventPanel.add(currentEventPanel);
```

```
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                currentCalendar.setVisible(false);
                                allEventPanel.setVisible(false);

                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;

                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
```

```
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(32, 209, 378,
94);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
                        });

                }
            });

        //in case not previously initialized, initializes currentE-
vents
        currentEvents = new JTable(eventModel);
        currentEvents.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e)
                {
                    currentCalendar.setVisible(false);
                    allEventPanel.setVisible(false);

                    //makes everything else not visible to clear up
visual clutter
                    eventPanel.setVisible(true);
                    int rowNum = currentEvents.getSelectedRow();
```

```
                    int eventID = searchEvent((String)currentE-
vents.getValueAt(rowNum, 0));

                    //gets SQLiteDataSource from JapaneseClub
                    SQLiteDataSource source = eventBase.getDS();
                    String query = "SELECT * FROM Events WHERE ID=" +
eventID;

                    //tries to connect to JapaneseClub and execute
query
                    try(Connection conn = source.getConnection();
                    Statement stat = conn.createStatement();)
                    {
                        ResultSet rs = stat.executeQuery(query);
                        if(rs.next())
                        {
                            eventDate.setText("Date: " +
rs.getString("Date"));
                            eventName.setText("Name: " +
rs.getString("Name"));
                            eventTimes.setText(rs.getString("Start") +
" - " + rs.getString("End"));
                            eventLocation.setText("Location: " +
rs.getString("Location"));
                        }
                        conn.close();
                    }
                    catch(SQLException e1)
                    {
                        e1.printStackTrace();
                        System.exit(0);
                    }
                    //initializes attendingStudents as new JTable with
all students with EventID matching this event
                    if(studentPane != null)
                    {
                        eventPanel.remove(studentPane);
                    }
                    int attendance = fillAttendingStudents(eventID);
                    attendingStudentsLbl.setText("Attending Members: "
+ attendance);
                    attendingStudents = new JTable(studentModel);
                    attendingStudents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                    studentPane = new JScrollPane(attendingStudents);
                    studentPane.setBounds(32, 209, 378, 94);
```

```
                    eventPanel.add(studentPane);
                    contentPane.add(eventPanel);

                }
            });

        //When a row on the table is clicked,
        //displays the Event's Date, Name, Start time, End Time, Loca-
tion, and Attending Members
        //from Events and EventStudents
        JButton showAllBtn = new JButton("Show All Events");
        showAllBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    eventPanel.setVisible(false);
                    allEventPanel.remove(currentEventPanel);
                    fillCurrentEvents("Show All");
                    currentEvents = new JTable(eventModel);
                    currentEvents.addMouseListener(new MouseAdapter()
{
                            @Override
                            public void mouseClicked(MouseEvent e)
                            {
                                currentCalendar.setVisible(false);
                                allEventPanel.setVisible(false);

                                //makes everything else not visible to
clear up visual clutter
                                eventPanel.setVisible(true);
                                int rowNum = currentEvents.getSelect-
edRow();
                                int eventID = searchEvent((String)cur-
rentEvents.getValueAt(rowNum, 0));

                                //gets SQLiteDataSource from Japa-
neseClub
                                SQLiteDataSource source = event-
Base.getDS();
                                String query = "SELECT * FROM Events
WHERE ID=" + eventID;

                                //tries to connect to JapaneseClub and
execute query
                                try(Connection conn = source.getCon-
nection();
```

```
                                Statement stat = conn.createState-
ment();)
                                {
                                    ResultSet rs = stat.exe-
cuteQuery(query);
                                    if(rs.next())
                                    {
                                        eventDate.setText("Date: " +
rs.getString("Date"));
                                        eventName.setText("Name: " +
rs.getString("Name"));

eventTimes.setText(rs.getString("Start") + " - " +
rs.getString("End"));
                                        eventLocation.setText("Loca-
tion: " + rs.getString("Location"));
                                    }
                                    conn.close();
                                }
                                catch(SQLException e1)
                                {
                                    e1.printStackTrace();
                                    System.exit(0);
                                }
                                //initializes attendingStudents as new
JTable with all students with EventID matching this event
                                if(studentPane != null)
                                {
                                    eventPanel.remove(studentPane);
                                }
                                int attendance = fillAttendingStu-
dents(eventID);
                                attendingStudentsLbl.setText("Attend-
ing Members: " + attendance);
                                attendingStudents = new JTable(stu-
dentModel);
                                attendingStudents.setFont(new
Font("Tahoma", Font.PLAIN, 20));
                                studentPane = new JScrollPane(attend-
ingStudents);
                                studentPane.setBounds(32, 209, 378,
94);
                                eventPanel.add(studentPane);
                                contentPane.add(eventPanel);

                            }
```

```
                    });
                currentEvents.setFont(new Font("Tahoma",
Font.PLAIN, 20));
                currentEventPanel = new JScrollPane(currentE-
vents);
                currentEventPanel.setBounds(10, 71, 575, 163);
                allEventPanel.add(currentEventPanel);
            }
        });
        showAllBtn.setBackground(new Color(0, 206, 209));
        showAllBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
        showAllBtn.setBounds(364, 64, 254, 49);
        contentPane.add(showAllBtn);

    }

    //searches for Event in Events table in database
    //if the event exists in the table, returns that event's ID
    //if not, returns -1
    public int searchEvent(String date)
    {
        int result = -1;
        SQLiteDataSource source = eventBase.getDS();
        String query = "SELECT * FROM Events WHERE Date='" + date +
"'";

        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement();)
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
                result = rs.getInt("ID");
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        return result;
    }

    //converts integers into YYYY-MM-DD format
    public String convertDate(int year, int month, int day)
    {
        String newMonth = "0";
```

```
        String newDay = "0";

        if((month - 10) < 0)
        {
            newMonth += Integer.toString(month);
        }
        else
        {
            newMonth = Integer.toString(month);
        }

        if((day - 10) < 0)
        {
            newDay += Integer.toString(day);
        }
        else
        {
            newDay = Integer.toString(day);
        }
        return year + "-" + newMonth + "-" + newDay;
    }

    //gets all rows from EventStudents where EventID = eventID
    //then adds the last and first names of all Students whose IDs
match the studentIDs from the selected rows
    //to StudentModel
    public int fillAttendingStudents(int eventID)
    {
        studentModel = new DefaultTableModel();
        studentModel.addColumn("");
        studentModel.addColumn("Last Name");
        studentModel.addColumn("First Name");
        int studentCount = 0;

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        String query = "SELECT Students.LastName, Students.FirstName
FROM EventStudents JOIN Students ON EventStudents.eventID=" + eventID
+ " AND eventStudents.studentID=Students.ID";

        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
```

```
                studentCount++;
                String[] result = {Integer.toString(studentCount),
rs.getString("LastName"),rs.getString("FirstName")};
                studentModel.insertRow(studentModel.getRowCount(), re-
sult);
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

        return studentCount;
    }

    //gets data from rows in Events where the Date is the same as date
    //and initializes eventModel with the Date and Name from every row
    //if date is "Show All", then all rows in Events are selected
    public void fillCurrentEvents(String date)
    {
        String query;
        eventModel = new DefaultTableModel();
        eventModel.addColumn("Date");
        eventModel.addColumn("Name");

        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = eventBase.getDS();
        if(date.equals("Show All"))
        {
            query = "SELECT * FROM Events ORDER BY DATE";
        }
        else
        {
            query = "SELECT * FROM Events WHERE Date='" + date + "'
ORDER BY Date";
        }

        //tries to connect to JapaneseClub and execute query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            while(rs.next())
            {
```

```
                String[] result = {rs.getString("Date"),
rs.getString("Name")};
                eventModel.insertRow(eventModel.getRowCount(), re-
sult);
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
    }
}
```

RegularUserHomepage Class

```
import java.awt.BorderLayout;
import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;

import org.sqlite.SQLiteDataSource;

import javax.swing.JLabel;
import java.awt.Font;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.SwingConstants;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.JTextArea;
import javax.swing.JButton;
import java.awt.Color;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class RegularUserHomepage extends JFrame
{

	private JPanel contentPane;
	private Database homeBase;

	/**
	 * Launch the application.
	 */
	public static void main(String[] args)
	{
		EventQueue.invokeLater(new Runnable()
		{
			public void run()
			{
				try
```

```
                    {
                        RegularUserHomepage frame = new RegularUs-
erHomepage();
                        frame.setVisible(true);
                    }
                    catch (Exception e)
                    {
                        e.printStackTrace();
                    }
                }
            });
    }

    /**
     * Create the frame.
     */
    public RegularUserHomepage()
    {
        homeBase = new Database();//creates new Database object

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 806, 766);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        final JLabel contactInfo = new JLabel("Contact Infor-
mation");
        contactInfo.setFont(new Font("Tahoma", Font.ITALIC, 20));
        contactInfo.setHorizontalAlignment(SwingConstants.CENTER);
        contactInfo.setBounds(10, 110, 748, 37);
        contentPane.add(contactInfo);

        final JTextArea FAQTxtBox = new JTextArea();
        FAQTxtBox.setEditable(false);
        FAQTxtBox.setFont(new Font("Tahoma", Font.PLAIN, 20));
        FAQTxtBox.setText("Frequently Asked Questions");
        FAQTxtBox.setBounds(23, 180, 748, 455);
        contentPane.add(FAQTxtBox);

        //when this window opens, set the Contact Info and FAQ to be
the values from the Home database
        addWindowListener(new WindowAdapter() {
            @Override
            public void windowOpened(WindowEvent e) {
```

```
                    //gets SQLiteDataSource from JapaneseClub
                    SQLiteDataSource source = homeBase.getDS();
                    String query = "SELECT * FROM Home";

                    //Creates connection to JapaneseClub and tries to
execute query
                    try(Connection conn = source.getConnection();
                    Statement stat = conn.createStatement(); )
                    {
                        ResultSet rs = stat.executeQuery(query);
                        contactInfo.setText(rs.getString("Contact"));
                        FAQTxtBox.setText(rs.getString("FAQ"));
                        conn.close();
                    }
                    catch(SQLException k)
                    {
                        k.printStackTrace();
                        System.exit(0);
                    }
                }
            });

            JLabel jpnClubLbl = new JLabel("Japanese Club");
            jpnClubLbl.setHorizontalAlignment(SwingConstants.CENTER);
            jpnClubLbl.setFont(new Font("Tahoma", Font.PLAIN, 40));
            jpnClubLbl.setBounds(206, 28, 368, 49);
            contentPane.add(jpnClubLbl);

            JButton logoutBtn = new JButton("Log-Out");
            logoutBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
                    LogoutScreen logout = new LogoutScreen();
                    logout.setVisible(true);
                    setVisible(false);
                }
            });
            logoutBtn.setBackground(new Color(0, 206, 209));
            logoutBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
            logoutBtn.setBounds(23, 673, 198, 49);
            contentPane.add(logoutBtn);

            JButton calendarBtn = new JButton("Go to Calendar");
            calendarBtn.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseClicked(MouseEvent e) {
```

```
                    RegularUserCalendar cal = new RegularUserCalen-
dar();
                    cal.setVisible(true);
                    setVisible(false);
               }
          });
          calendarBtn.setFont(new Font("Tahoma", Font.PLAIN, 20));
          calendarBtn.setBackground(new Color(0, 206, 209));
          calendarBtn.setBounds(536, 673, 235, 49);
          contentPane.add(calendarBtn);
     }
}
```

Student Class

```
import org.sqlite.SQLiteDataSource;
import java.sql.*;

public class Student
{
    private String myLastName;
    private String myFirstName;
    private int myYear;
    private int myGraduationYear;
    private Database myStudentBase;

    public Student(String lastName, String firstName, int year, int
graduationYear)
    {
        myLastName = lastName;
        myFirstName = firstName;
        myYear = year;
        myGraduationYear = graduationYear;
        myStudentBase = new Database();
    }

    public Student(String lastName, String firstName, int year, int
graduationYear, int marker)
    {
        myLastName = lastName;
        myFirstName = firstName;
        myYear = year;
        myGraduationYear = graduationYear;
        myStudentBase = new Database();
    }

    public String getMyLastName() {
        return myLastName;
    }

    //updates value of myLastName and value of LastName in Students
where ID = studentID
    public void updateMyLastName(String lastName, int studentID) {
        myLastName = lastName;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS(); //gets the DS
of the database
```

```
        String query = "UPDATE Students SET LastName = '" + myLastName
+ "' WHERE ID=" + studentID;

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); ) //creates connec-
tion to the database
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e1)
        {
            e1.printStackTrace();
            System.exit(0);
        }

    }

    public String getMyFirstName() {
        return myFirstName;
    }

    //updates value of myFirstName and value of FirstName in Students
where ID = studentID
    public void updateMyFirstName(String firstName, int studentID) {
        myFirstName = firstName;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();
        String query = "UPDATE Students SET FirstName = '" + myFirst-
Name + "' WHERE ID=" + studentID;
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e1)
        {
            e1.printStackTrace();
            System.exit(0);
        }
    }
```

```
    public int getMyYear() {
        return myYear;
    }

    //updates Students table so that row where ID = studentID has its
Year value set to the new value of myYear
    public void updateMyYear(int year, int studentID) {
        myYear = year;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();
        String query = "UPDATE Students SET Year = '" + myYear + "'
WHERE ID=" + studentID;
        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e1)
        {
            e1.printStackTrace();
            System.exit(0);
        }
    }

    public int getMyGraduationYear() {
        return myGraduationYear;
    }

    //updates value of myGraduationYear and value of GraduationYear in
Students where ID = studentID
    public void updateMyGraduationYear(int graduationYear, int studen-
tID) {
        myGraduationYear = graduationYear;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();
        String query = "UPDATE Students SET GraduationYear = '" +
myGraduationYear + "' WHERE ID=" + studentID;
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
```

```
        }
        catch(SQLException e1)
        {
            e1.printStackTrace();
            System.exit(0);
        }
    }

    //enters new row into Students with myLastName, myFirstName,
myYear, and myGraduationYear
    public void enterStudent()
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();

        String query = "INSERT INTO Students (LastName,First-
Name,Year,GraduationYear) VALUES ('" + myLastName + "','" + myFirst-
Name + "','" + myYear + "','" + myGraduationYear + "')";

        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }

    }

    //returns ID value of student if found
    //returns -1 if Student does not exist in database
    public int searchStudent(String last, String first, int id)
    {
        int result = -1;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();

        String query = "SELECT * FROM Students WHERE LastName='" +
last + "' AND FirstName='" + first + "' AND ID=" + id;

        //Creates connection to JapaneseClub and tries to execute
query
```

```
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
            {
                result = rs.getInt("ID");
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        return result;
    }

    //returns ID value of student if found
    //returns -1 if Student does not exist in database
    public int searchStudent(String last, String first)
    {
        int result = -1;
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();

        String query = "SELECT * FROM Students WHERE LastName='" +
last + "' AND FirstName='" + first + "'";

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            ResultSet rs = stat.executeQuery(query);
            if(rs.next())
            {
                result = rs.getInt("ID");
            }
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
```

```
        return result;
    }

    //deletes row from Student where ID=id
    public void deleteStudent(int id)
    {
        //gets SQLiteDataSource from JapaneseClub
        SQLiteDataSource source = myStudentBase.getDS();

        String query = "DELETE FROM Students WHERE ID=" + id;

        //Creates connection to JapaneseClub and tries to execute
query
        try(Connection conn = source.getConnection();
        Statement stat = conn.createStatement(); )
        {
            stat.execute(query);
            conn.close();
        }
        catch(SQLException e)
        {
            e.printStackTrace();
            System.exit(0);
        }
        System.out.print("deleted student");

    }
}
```