ICS4U7-3B Internal Assessment Appendix C

Registration Code:

```
/* Notes:
 * Status: FINISHED
 * Currently handles the saving of multiple users & pws */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class Registration extends Frame implements ActionListener {
        private JFrame f;
        private JPanel panelUser, panelPw, panelButton;
        private JPanel emptySpace, emptySpace1, emptySpace2, panelMsg;
        private JLabel userL, pwL;
        private JTextArea success;
        private JLabel blank, blank1;
        private JTextField userTF, pwTF;
        private JButton createAccount;
        public String username, password;

        public Registration() {
```

```
// New frame
f = new JFrame("Registration");

// New panels
emptySpace = new JPanel();
emptySpace.setBackground(Color.gray);

panelUser = new JPanel();
panelUser.setLayout(new GridLayout(1, 2));
panelUser.setBackground(Color.gray);

emptySpace1 = new JPanel();
emptySpace1.setBackground(Color.gray);

panelPw = new JPanel();
panelPw.setLayout(new GridLayout(1, 2));
panelPw.setBackground(Color.gray);

emptySpace2 = new JPanel();
emptySpace2.setBackground(Color.gray);

panelButton = new JPanel();
panelButton.setLayout(new GridLayout(1, 3));
```

```
panelButton.setBackground(Color.gray);

panelMsg = new JPanel();
panelMsg.setBackground(Color.gray);

// New labels
blank = new JLabel("");
blank.setBackground(Color.gray);

blank1 = new JLabel("");
blank1.setBackground(Color.gray);

userL = new JLabel("Username:");
userL.setBackground(Color.gray);
userL.setForeground(Color.white);

pwL = new JLabel("Password:");
pwL.setBackground(Color.gray);
pwL.setForeground(Color.white);

success = new JTextArea();
success.setBackground(Color.gray);
success.setForeground(Color.white);
```

```
success.setVisible(false);
success.setEditable(false);

// New textfields
userTF = new JTextField("");

pwTF = new JTextField("");

// New button
createAccount = new JButton("Create Account");
createAccount.setBackground(Color.green);
createAccount.addActionListener(this);

// Adding components to panel
panelUser.add(userL);
panelUser.add(userTF);

panelPw.add(pwL);
panelPw.add(pwTF);

panelButton.add(blank);
panelButton.add(createAccount);
panelButton.add(blank1);
```

```
        panelMsg.add(success);

        // Adding panel to frame
        f.add(emptySpace);
        f.add(panelUser);
        f.add(emptySpace1);
        f.add(panelPw);
        f.add(emptySpace2);
        f.add(panelButton);
        f.add(panelMsg);
        f.setLayout(new GridLayout(7, 1));
        f.setSize(400, 400);
        f.setVisible(true);
}

public void savingInfo(String u, String p) throws IOException {
        File directory = new File(u);

        if (u.equals("") || p.equals("")) {
                success.setText("Please create a valid username and password.");
                success.setVisible(true);
        }
```

```
                else if (directory.isDirectory()) {
                        success.setText("User already exists.\nPlease sign in, or make a different
account.");
                        success.setVisible(true);
                }
                else {
                        // Creating new directory
                        directory.mkdir();

                        // Creating new file
                        FileWriter fw = new FileWriter(directory + "/login_info.txt");

                        // Writing to the file
                        for (int i = 0; i < u.length(); i++) {
                                fw.write(u.charAt(i));
                        }

                        fw.write("\n");

                        for (int i = 0; i < p.length(); i++) {
                                fw.write(p.charAt(i));
                        }
```

```
                    // Closing the file reader
                    fw.close();

                    // Closes entire frame
                    f.dispose();
                }
            }

            public void actionPerformed(ActionEvent e) {
                username = userTF.getText();
                password = pwTF.getText();

                try {
                    // Calls saving method
                    savingInfo(username, password);
                }
                catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }
}
```

Login Validation Code:

```
/* Notes:
```

```
 * Status: FINISHED
 * Currently checks for validity of login information */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class LoginValidation extends Frame implements ActionListener {
	private JFrame f;
	private JPanel panelUser, panelPw, panelButtons, panelMsg;
	private JPanel emptySpace, emptySpace1, emptySpace2, emptySpace3;
	private JLabel userL, pwL;
	private JLabel blank, blank1, blank2;
	private JTextArea success;
	private JTextField userTF, pwTF;
	private JButton signIn, register;
	private boolean correct = false;
	protected static String username, password;

	public void loginValidation() {
		// New frame
		f = new JFrame("Login");
```

```
// New panels
emptySpace = new JPanel();
emptySpace.setBackground(Color.gray);

panelUser = new JPanel();
panelUser.setLayout(new GridLayout(1, 2));
panelUser.setBackground(Color.gray);

emptySpace1 = new JPanel();
emptySpace1.setBackground(Color.gray);

panelPw = new JPanel();
panelPw.setLayout(new GridLayout(1, 2));
panelPw.setBackground(Color.gray);

emptySpace2 = new JPanel();
emptySpace2.setBackground(Color.gray);

panelButtons = new JPanel();
panelButtons.setLayout(new GridLayout(1, 3));
panelButtons.setBackground(Color.gray);
```

```
panelMsg = new JPanel();
panelMsg.setBackground(Color.gray);

// New labels
blank = new JLabel("");
blank.setBackground(Color.gray);

blank1 = new JLabel("");
blank1.setBackground(Color.gray);

blank2 = new JLabel("");
blank2.setBackground(Color.gray);

userL = new JLabel("Username:");
userL.setBackground(Color.gray);
userL.setForeground(Color.white);

pwL = new JLabel("Password:");
pwL.setBackground(Color.gray);
pwL.setForeground(Color.white);

success = new JTextArea();
success.setBackground(Color.gray);
```

```
success.setForeground(Color.white);
success.setVisible(false);
success.setEditable(false);

// New textfields
userTF = new JTextField("");

pwTF = new JTextField("");

// New buttons
register = new JButton("Register");
register.setBackground(Color.green);
register.addActionListener(this);

signIn = new JButton("Sign In");
signIn.setBackground(Color.green);
signIn.addActionListener(this);

// Adding components to panel
panelUser.add(userL);
panelUser.add(userTF);

panelPw.add(pwL);
```

```
        panelPw.add(pwTF);


        panelButtons.add(blank);
        panelButtons.add(register);
        panelButtons.add(blank1);
        panelButtons.add(signIn);
        panelButtons.add(blank2);


        panelMsg.add(success);


        // Adding panel to frame
        f.add(emptySpace);
        f.add(panelUser);
        f.add(emptySpace1);
        f.add(panelPw);
        f.add(emptySpace2);
        f.add(panelButtons);
        f.add(panelMsg);
        f.setLayout(new GridLayout(7, 1));
        f.setSize(430, 400);
        f.setVisible(true);
}
```

```
public String getPassword(FileReader fr) {
	// Variable declaration
	int ch;
	String actualPw = "";

	// Process
	try {
		// Reading the username
		while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
			actualPw += (char) ch;
		}

		actualPw = "";

		// Reading the password
		while ((ch = fr.read()) != -1) {
			actualPw += (char) ch;
		}

		return actualPw;
	}
	catch (IOException ioe) {
		ioe.printStackTrace();
```

```
            return null;
        }
}

public void validation(String username, String password) throws IOException {
        // Variable declaration
        String actualPw;

        // Declaring directory
        File directory = new File(username);

        if (username.equals("") || password.equals("")) {
            success.setText("Please enter your username and password.");
            success.setVisible(true);
        }
        else {
            // Determining whether the user exists
            if (directory.isDirectory()) {
                // Declaring file reader
                FileReader fr = new FileReader(directory + "/login_info.txt");

                // Getting password
```

```
                actualPw = getPassword(fr);

                if (!(password.equals(actualPw))) { // Incorrect password

                        success.setText("Incorrect password.\n"
                                        + "Please try again.");
                        success.setVisible(true);

                        // Closing file
                        fr.close();
                }
                else { // Correct user & pw
                        correct = true;

                        // Closing file
                        fr.close();
                }
        }
        else {
                // Username incorrect or user does not exist
                success.setText("Your username does not match any existing
records.\n"
                                + "Please try again, or create a new account.");
```

```
                    success.setVisible(true);
                }
            }
        }

        public void setLogin(String username, String password) {
            LoginValidation.username = username;
            LoginValidation.password = password;
        }

        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == signIn) {
                try {
                    // Calls validation method
                    validation(userTF.getText(), pwTF.getText());

                    if (correct == true) {
                        setLogin(userTF.getText(), pwTF.getText());
                        f.dispose();
                        new MainProgram();
                    }
                }
                catch (IOException ioe) {
```

```
                        ioe.printStackTrace();
                    }
                }
                else {
                    new Registration();
                }
            }
}
```

Main Program Code:

```
/* Notes:
 * Status: FINISHED */

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import javax.swing.*;

public class MainProgram extends LoginValidation implements MouseListener {
        // ALL MAIN PROGRAM VARIABLES BELOW
        protected JFrame fMain;
        protected JPanel panelStart, panelCal, panelLists, panelRecords, panelFinance;
        protected JButton buttonCal, buttonFinance, buttonLists, buttonRecords, buttonNotepad,
buttonHelp, signOut;
```

```
protected Color c = new Color(180, 200, 255);

public MainProgram() {
        // New frame
        fMain = new JFrame("Organizer for Music Teachers");

        // New panels
        panelStart = new JPanel();
        panelStart.setLayout(new GridLayout(1, 7));
        panelStart.setBackground(Color.gray);

        // Calendar button
        buttonCal = new JButton("Calendar");
        buttonCal.setBackground(c);
        buttonCal.addActionListener(this);

        buttonLists = new JButton("Student Lists");
        buttonLists.setBackground(c);
        buttonLists.addActionListener(this);

        buttonFinance = new JButton("Financial Records");
        buttonFinance.setBackground(c);
        buttonFinance.addActionListener(this);
```

```
buttonRecords = new JButton("Student Records");
buttonRecords.setBackground(c);
buttonRecords.addActionListener(this);

buttonNotepad = new JButton("Notepad");
buttonNotepad.setBackground(c);
buttonNotepad.addActionListener(this);

buttonHelp = new JButton("Help");
buttonHelp.setBackground(c);
buttonHelp.addActionListener(this);

signOut = new JButton("Sign Out");
signOut.setBackground(c);
signOut.addActionListener(this);

// Adding components to panel
panelStart.add(buttonCal);
panelStart.add(buttonLists);
panelStart.add(buttonRecords);
panelStart.add(buttonFinance);
panelStart.add(buttonNotepad);
```

```
        panelStart.add(buttonHelp);
        panelStart.add(signOut);

        // Adding panel to frame
        fMain.add(panelStart);
        fMain.setSize(1200, 200);
        fMain.setVisible(true);
}

public void mouseClicked(MouseEvent e) {}

public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}

public void mousePressed(MouseEvent e) {}

public void mouseReleased(MouseEvent e) {}

public void actionPerformed(ActionEvent e) {
        // ALL MAIN PROGRAM EVENTS
        if (e.getSource() == buttonCal) {
                new Calendar().calendar();
```

```
            fMain.dispose();
        }
        else if (e.getSource() == buttonLists) {
            try {
                new StudentLists().studentLists();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonFinance) {
            try {
                new FinancialRecords().finance();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonRecords) {
            try {
                new StudentRecords().studentRecords();
```

```
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonNotepad) {
            try {
                new Notepad();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonHelp) {
            new Help();
        }
        else if (e.getSource() == signOut) {
            fMain.dispose();
        }
}
```

}

Calendar Code:

```
/* Notes:
 * Status: FINISHED
 * Currently functional
 * Currently saves repeating events for up to 1 month
 * Bugged for days that overlap (ex: December 24/31)
 * Missing multiple events function */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

import java.io.*;

import java.time.LocalDate;
import java.time.DayOfWeek;
import java.time.format.DateTimeFormatter;
import java.time.format.TextStyle;

public class Calendar extends MainProgram {
        private JFrame fCalEvents, fDisplay, fDisplayOrAdd;
        private JPanel panelEvent, panelDisplay, panelDisplayOrAdd;
```

```
private JLabel month, monday, tuesday, wednesday, thursday, friday, saturday, sunday;
private JButton leftArrow, rightArrow, addEvent, cancel, addNewEvent, display, delete;
private JTextField eventName, description, eventSTime, eventETime;
private JCheckBox daily = new JCheckBox("Daily", false);
private JCheckBox weekly = new JCheckBox("Weekly", false);
private String eventDate, endString = "";
private Object object;

private File directory = new File(username + "/Calendar");;

private DateTimeFormatter year = DateTimeFormatter.ofPattern("yyyy");
private DateTimeFormatter monthNum = DateTimeFormatter.ofPattern("MM");
private DateTimeFormatter day = DateTimeFormatter.ofPattern("dd");
private LocalDate today = LocalDate.now();
private LocalDate selectedDate;
private DayOfWeek weekDay;
private String monthName = "";
private String dayName;
private int[][] days;

public void calendar() {
        // New panel
        panelCal = new JPanel();
```

```
panelCal.setLayout(new GridLayout(8, 7));
panelCal.setBackground(Color.gray);

calendarDates(today);

// Month label
month = new JLabel(monthName + ", " + year.format(today));
month.setOpaque(true);
month.setBackground(new Color(150, 250, 200));
month.setForeground(Color.black);

// Days of the week labels
sunday = new JLabel("Sunday");
sunday.setForeground(Color.white);

monday = new JLabel("Monday");
monday.setOpaque(true);
monday.setBackground(Color.black);
monday.setForeground(Color.white);

tuesday = new JLabel("Tuesday");
tuesday.setForeground(Color.white);
```

```
wednesday = new JLabel("Wednesday");
wednesday.setOpaque(true);
wednesday.setBackground(Color.black);
wednesday.setForeground(Color.white);

thursday = new JLabel("Thursday");
thursday.setForeground(Color.white);

friday = new JLabel("Friday");
friday.setOpaque(true);
friday.setBackground(Color.black);
friday.setForeground(Color.white);

saturday = new JLabel("Saturday");
saturday.setForeground(Color.white);

// New blanks labels for formatting
JLabel blank1 = new JLabel("");
blank1.setOpaque(true);
blank1.setBackground(new Color(150, 250, 200));

JLabel blank2 = new JLabel("");
blank2.setOpaque(true);
```

```
blank2.setBackground(new Color(150, 250, 200));

JLabel blank3 = new JLabel("");
blank3.setOpaque(true);
blank3.setBackground(new Color(150, 250, 200));

JLabel blank4 = new JLabel("");
blank4.setOpaque(true);
blank4.setBackground(new Color(150, 250, 200));

// New buttons
leftArrow = new JButton("<");
leftArrow.setBackground(new Color(150, 250, 200));
leftArrow.addActionListener(this);

rightArrow = new JButton(">");
rightArrow.setBackground(new Color(150, 250, 200));
rightArrow.addActionListener(this);

// Disable calendar button
buttonCal.setEnabled(false);
buttonCal.setBackground(Color.pink);
```

```
// Adding components to panel
panelCal.add(buttonCal);
panelCal.add(buttonLists);
panelCal.add(buttonRecords);
panelCal.add(buttonFinance);
panelCal.add(buttonNotepad);
panelCal.add(buttonHelp);
panelCal.add(signOut);

panelCal.add(blank1);
panelCal.add(blank2);
panelCal.add(leftArrow);
panelCal.add(month);
panelCal.add(rightArrow);
panelCal.add(blank3);
panelCal.add(blank4);

panelCal.add(sunday);
panelCal.add(monday);
panelCal.add(tuesday);
panelCal.add(wednesday);
panelCal.add(thursday);
panelCal.add(friday);
```

```
panelCal.add(saturday);

// New textfields
JTextField[][] calendarDays = new JTextField[5][7];

// Initializing the textfields
for (int rows = 0; rows < days.length; rows++) {
    for (int columns = 0; columns < 7; columns++) {
        if (rows == 5) {
            if (dayName.equals("Friday") && days[rows][columns] ==
31) {

calendarDays[rows-1][columns].setText(calendarDays[rows-1][columns].getText() + "/" +
Integer.toString(days[rows][columns]));
            }
            else if (dayName.equals("Saturday") &&
days[rows][columns] == 30) {

calendarDays[rows-1][columns].setText(calendarDays[rows-1][columns].getText() + "/" +
Integer.toString(days[rows][columns]));
            }
            else if (dayName.equals("Saturday") &&
days[rows][columns] == 31) {
```

```
calendarDays[rows-1][columns].setText(calendarDays[rows-1][columns].getText() + "/" +
Integer.toString(days[rows][columns]));
                                        }
                                }
                                else {
                                        calendarDays[rows][columns] = new JTextField();

calendarDays[rows][columns].setText(Integer.toString(days[rows][columns]));

                                        // Grey out the days that aren't part of the month
                                        if (calendarDays[rows][columns].getText().equals("0")) {
                                                calendarDays[rows][columns].setText(null);
                                        }
                                }
                        }
                }

                // Adding the textfields to the panel
                for (int rows = 0; rows <= 4; rows++) {
                        for (int columns = 0; columns <= 6; columns++) {
                                if (!(calendarDays[rows][columns].getText().equals(""))) {
                                        calendarDays[rows][columns].addMouseListener(this);
```

```
                    }
                    calendarDays[rows][columns].setEditable(false);
                    panelCal.add(calendarDays[rows][columns]);
                }
            }

            // Adding panel to frame
            fMain.add(panelCal);
            fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
            fMain.setVisible(true);
            fMain.remove(panelStart);
    }

    public void calendarDates(LocalDate today) {
            int todaysDate = (int) Integer.parseInt(day.format(today));
            int totalDays = 0;

            // Determining name of current month & total # of days
            switch (monthNum.format(today)) {
                case "01" :
                    monthName = "January";
                    totalDays = 31;
                    break;
```

```
case "02" :
        monthName = "February";
        if (today.isLeapYear() == true) {
                totalDays = 29;
        }
        else {
                totalDays = 28;
        }
        break;
case "03" :
        monthName = "March";
        totalDays = 31;
        break;
case "04" :
        monthName = "April";
        totalDays = 30;
        break;
case "05" :
        monthName = "May";
        totalDays = 31;
        break;
case "06" :
        monthName = "June";
```

```
            totalDays = 30;
            break;
        case "07" :
            monthName = "July";
            totalDays = 31;
            break;
        case "08" :
            monthName = "August";
            totalDays = 31;
            break;
        case "09" :
            monthName = "September";
            totalDays = 30;
            break;
        case "10" :
            monthName = "October";
            totalDays = 31;
            break;
        case "11" :
            monthName = "November";
            totalDays = 30;
            break;
        case "12" :
```

```
                monthName = "December";
                totalDays = 31;
                break;
}

// First day of the month
LocalDate firstDay = today.minusDays(todaysDate-1);

// Determining the week day of the first day
weekDay = DayOfWeek.from(firstDay);
dayName = weekDay.getDisplayName(TextStyle.FULL, getLocale());

int indexNum = 0;
int day = 0;

switch (dayName) {
        case "Sunday" :
                indexNum = 7;
                days = new int[5][7];
                day = 1;
                break;
        case "Monday" :
                indexNum = 1;
```

```
            days = new int[5][7];
            break;
case "Tuesday" :
            indexNum = 2;
            days = new int[5][7];
            break;
case "Wednesday" :
            indexNum = 3;
            days = new int[5][7];
            break;
case "Thursday" :
            indexNum = 4;
            days = new int[5][7];
            break;
case "Friday" :
            indexNum = 5;
            if (totalDays == 31) {
                        days = new int[6][7];
            }
            else {
                        days = new int[5][7];
            }
            break;
```

```
        case "Saturday" :
                indexNum = 6;
                if (totalDays == 31 || totalDays == 30) {
                        days = new int[6][7];
                }
                else {
                        days = new int[5][7];
                }
                break;
}

// Inputting the days into the calendar
int count = 0;

for (int rows = 0; rows < days.length; rows++) {
        for (int columns = 0; columns < 7; columns++) {
                if (indexNum != 7) { // not starting on Sunday
                        if (!(rows == 0 && columns < indexNum-1)) {
                                days[rows][columns] = day;
                                if (day < totalDays) {
                                        day++;
                                }
                                if (day == totalDays) {
```

```
                                        count++;
                                        if (count == 2) {
                                                break;
                                        }
                                }
                        }
                }
                else { // starting on Sunday
                        days[rows][columns] = day;
                        if (day < totalDays) {
                                day++;
                        }
                        if (day == totalDays) {
                                count++;
                                if (count == 2) {
                                        break;
                                }
                        }
                }
        }
}
```

```
	public void displayOrAdd() {
		eventDate = monthName + " " + ((JTextField) object).getText() + ", " +
year.format(today);

		// New frame
		fDisplayOrAdd = new JFrame(eventDate);

		// New panel
		panelDisplayOrAdd = new JPanel();
		panelDisplayOrAdd.setLayout(new GridLayout(1, 2));
		panelDisplayOrAdd.setBackground(Color.gray);

		// New buttons
		display = new JButton("Display Events");
		display.addActionListener(this);

		addNewEvent = new JButton("New Event");
		addNewEvent.addActionListener(this);

		// Adding buttons to panel
		panelDisplayOrAdd.add(display);
		panelDisplayOrAdd.add(addNewEvent);
```

```
        // Adding panel to frame
        fDisplayOrAdd.add(panelDisplayOrAdd);
        fDisplayOrAdd.setSize(400, 200);
        fDisplayOrAdd.setVisible(true);
}

public void addCalEvent(Object object) {
        int num = 0;

        // Determining selected month name
        switch (monthName) {
                case "January" :
                        num = 1;
                        break;
                case "February" :
                        num = 2;
                        break;
                case "March" :
                        num = 3;
                        break;
                case "April" :
                        num = 4;
                        break;
```

```
case "May" :
        num = 5;
        break;
case "June" :
        num = 6;
        break;
case "July" :
        num = 7;
        break;
case "August" :
        num = 8;
        break;
case "September" :
        num = 9;
        break;
case "October" :
        num = 10;
        break;
case "November" :
        num = 11;
        break;
case "December" :
        num = 12;
```

```
                    break;
            }

            // Determining selected date
            selectedDate = LocalDate.of(Integer.parseInt(year.format(today)), num,
Integer.parseInt(((JTextField) object).getText()));

            LocalDate endDate = selectedDate.plusWeeks(4);
            DateTimeFormatter end = DateTimeFormatter.ofPattern("MM");

            // Determining name of current month & total # of days
            switch (end.format(endDate)) {
                case "01" :
                    endString = "January";
                    break;
                case "02" :
                    endString = "February";
                    break;
                case "03" :
                    endString = "March";
                    break;
                case "04" :
                    endString = "April";
```

```
            break;
        case "05" :
            endString = "May";
            break;
        case "06" :
            endString = "June";
            break;
        case "07" :
            endString = "July";
            break;
        case "08" :
            endString = "August";
            break;
        case "09" :
            endString = "September";
            break;
        case "10" :
            endString = "October";
            break;
        case "11" :
            endString = "November";
            break;
        case "12" :
```

```
                endString = "December";
                break;
}

// Determining day
int date;

switch(day.format(endDate)) {
        case "01":
                date = 1;
                break;
        case "02":
                date = 2;
                break;
        case "03":
                date = 3;
                break;
        case "04":
                date = 4;
                break;
        case "05":
                date = 5;
                break;
```

```
                case "06":
                        date = 6;
                        break;
                case "07":
                        date = 7;
                        break;
                case "08":
                        date = 8;
                        break;
                case "09":
                        date = 9;
                        break;
                default:
                        date = Integer.parseInt(day.format(endDate));
                        break;
                }

                endString += " " + date + ", " + year.format(endDate);

                // New frame
                eventDate = monthName + " " + ((JTextField) object).getText() + ", " +
year.format(today);
                fCalEvents = new JFrame(eventDate);
```

```
// New panel
panelEvent = new JPanel();
panelEvent.setLayout(new GridLayout(6, 2));
panelEvent.setBackground(Color.gray);

// New labels
JLabel name = new JLabel("Event Name:");
JLabel des = new JLabel("Description:");
JLabel sTime = new JLabel("Start Time:");
JLabel eTime = new JLabel("End Time:");

// New textfields
eventName = new JTextField();
description = new JTextField();
eventSTime = new JTextField("Ex: 9:00 AM");
eventETime = new JTextField("Ex: 5:00 PM");

// New button
addEvent = new JButton("Add Event");
addEvent.addActionListener(this);

cancel = new JButton("Cancel");
```

```
        cancel.addActionListener(this);

        // Adding components to panel
        panelEvent.add(name);
        panelEvent.add(eventName);
        panelEvent.add(des);
        panelEvent.add(description);
        panelEvent.add(sTime);
        panelEvent.add(eventSTime);
        panelEvent.add(eTime);
        panelEvent.add(eventETime);
        panelEvent.add(daily);
        panelEvent.add(weekly);
        panelEvent.add(addEvent);
        panelEvent.add(cancel);

        // Adding panel to frame
        fCalEvents.add(panelEvent);
        fCalEvents.setSize(400, 400);
        fCalEvents.setVisible(true);
}

public void saveCalEvent(JTextField eventName, JTextField description,
```

```
            JTextField eventSTime, JTextField eventETime, JCheckBox daily,
            JCheckBox weekly) throws IOException {
        // Creating directory
        directory.mkdir();

        // Creating new file
        FileWriter fw = new FileWriter(directory + "/" + eventDate + ".txt");

        // ensuring empty events aren't saved
        if (!(eventName.getText().equals("") && description.getText().equals("") &&
eventSTime.getText().equals("") && eventETime.getText().equals(""))) {
            // Writing to the file
            // event name
            for (int i = 0; i < eventName.getText().length(); i++) {
                fw.write(eventName.getText().charAt(i));
            }

            fw.write("\n");

            // description
            for (int i = 0; i < description.getText().length(); i++) {
                fw.write(description.getText().charAt(i));
            }
```

```
fw.write("\n");

// start time
for (int i = 0; i < eventSTime.getText().length(); i++) {
        fw.write(eventSTime.getText().charAt(i));
}

fw.write("\n");

// end time
for (int i = 0; i < eventETime.getText().length(); i++) {
        fw.write(eventETime.getText().charAt(i));
}

fw.write("\n");

// repeats daily
if (daily.isSelected()) {
        fw.write("true");

        // recursion
        if (!(eventDate.equals(endString))) {
```

```
selectedDate = selectedDate.plusDays(1);

// Determining name of current month
switch (monthNum.format(selectedDate)) {
        case "01" :
                monthName = "January";
                break;
        case "02" :
                monthName = "February";
                break;
        case "03" :
                monthName = "March";
                break;
        case "04" :
                monthName = "April";
                break;
        case "05" :
                monthName = "May";
                break;
        case "06" :
                monthName = "June";
                break;
        case "07" :
```

```
                monthName = "July";
                break;
            case "08" :
                monthName = "August";
                break;
            case "09" :
                monthName = "September";
                break;
            case "10" :
                monthName = "October";
                break;
            case "11" :
                monthName = "November";
                break;
            case "12" :
                monthName = "December";
                break;
    }

    // Determining day
    int date;

    switch(day.format(selectedDate)) {
```

```
case "01":
        date = 1;
        break;
case "02":
        date = 2;
        break;
case "03":
        date = 3;
        break;
case "04":
        date = 4;
        break;
case "05":
        date = 5;
        break;
case "06":
        date = 6;
        break;
case "07":
        date = 7;
        break;
case "08":
        date = 8;
```

```
                                        break;
                                    case "09":
                                        date = 9;
                                        break;
                                    default:
                                        date =
Integer.parseInt(day.format(selectedDate));
                                        break;
                                }

                                eventDate = monthName + " " + date + ", " +
year.format(selectedDate);

                                saveCalEvent(eventName, description, eventSTime,
eventETime, daily, weekly);
                            }
                        }
                        else {
                            fw.write("false");
                        }

                        fw.write("\n");
```

```
// repeats weekly
if (weekly.isSelected()) {
        fw.write("true");

        // recursion
        if (!(eventDate.equals(endString))) {
                selectedDate = selectedDate.plusWeeks(1);

                // Determining name of current month
                switch (monthNum.format(selectedDate)) {
                        case "01" :
                                monthName = "January";
                                break;
                        case "02" :
                                monthName = "February";
                                break;
                        case "03" :
                                monthName = "March";
                                break;
                        case "04" :
                                monthName = "April";
                                break;
                        case "05" :
```

```
            monthName = "May";
            break;
        case "06" :
            monthName = "June";
            break;
        case "07" :
            monthName = "July";
            break;
        case "08" :
            monthName = "August";
            break;
        case "09" :
            monthName = "September";
            break;
        case "10" :
            monthName = "October";
            break;
        case "11" :
            monthName = "November";
            break;
        case "12" :
            monthName = "December";
            break;
```

```
}

// Determining day
int date;

switch(day.format(selectedDate)) {
        case "01":
                date = 1;
                break;
        case "02":
                date = 2;
                break;
        case "03":
                date = 3;
                break;
        case "04":
                date = 4;
                break;
        case "05":
                date = 5;
                break;
        case "06":
                date = 6;
```

```
                    break;
                case "07":
                    date = 7;
                    break;
                case "08":
                    date = 8;
                    break;
                case "09":
                    date = 9;
                    break;
                default:
                    date =
Integer.parseInt(day.format(selectedDate));
                    break;
            }

            eventDate = monthName + " " + date + ", " +
year.format(selectedDate);

            saveCalEvent(eventName, description, eventSTime,
eventETime, daily, weekly);
        }
    }
```

```
else {
        fw.write("false");
}

// Resetting month name
switch (monthNum.format(today)) {
        case "01" :
                monthName = "January";
                break;
        case "02" :
                monthName = "February";
                break;
        case "03" :
                monthName = "March";
                break;
        case "04" :
                monthName = "April";
                break;
        case "05" :
                monthName = "May";
                break;
        case "06" :
                monthName = "June";
```

```
                    break;
                case "07" :
                    monthName = "July";
                    break;
                case "08" :
                    monthName = "August";
                    break;
                case "09" :
                    monthName = "September";
                    break;
                case "10" :
                    monthName = "October";
                    break;
                case "11" :
                    monthName = "November";
                    break;
                case "12" :
                    monthName = "December";
                    break;
            }
        }

        // Closing the file writer
```

```
                fw.close();
        }

        public void displayEvents() throws IOException {
                eventDate = monthName + " " + ((JTextField) object).getText() + ", " +
year.format(today);

                // Variable declaration
                int ch;
                String eventName = "", description = "", sTime = "", eTime = "";
                JLabel display = new JLabel();
                JLabel name = new JLabel();
                JLabel des = new JLabel();
                JLabel start = new JLabel();
                JLabel end = new JLabel();
                JLabel repeats = new JLabel();;
                delete = new JButton("Delete event");

                fDisplay = new JFrame(eventDate);
                panelDisplay = new JPanel();
                panelDisplay.setBackground(Color.gray);

                // Creating new file
```

```
FileReader fr;
File f = new File(directory + "/" + eventDate + ".txt");

try {
        fr = new FileReader(f);

        if (f.length() == 0) {
                display.setText("No events found for this day.");

                display.setForeground(Color.white);
                panelDisplay.add(display);

                panelDisplay.setLayout(new GridLayout(2, 1));

                delete.addActionListener(this);
                panelDisplay.add(delete);

                // Adding panel to frame
                fDisplay.add(panelDisplay);
                fDisplay.setSize(400, 400);
                fDisplay.setVisible(true);
        }
        else {
```

```
// Reading event name
while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
        eventName += (char) ch;
}

// description
while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
        description += (char) ch;
}

// start time
while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
        sTime += (char) ch;
}

// end time
while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
        eTime += (char) ch;
}

// repeats daily
if (this.daily.isSelected()) {
        String daily = "Repeats daily";
```

```
            // New labels
            name.setText("Event Name: " + eventName);
            des.setText("Description: " + description);
            start.setText("Start Time: " + sTime);
            end.setText("End Time: " + eTime);
            repeats.setText(daily);
        }
        else if (this.weekly.isSelected()) {
            String weekly = "Repeats weekly";

            // New labels
            name.setText("Event Name: " + eventName);
            des.setText("Description: " + description);
            start.setText("Start Time: " + sTime);
            end.setText("End Time: " + eTime);
            repeats.setText(weekly);
        }
        else {
            // New labels
            name.setText("Event Name: " + eventName);
            des.setText("Description: " + description);
            start.setText("Start Time: " + sTime);
```

```
            end.setText("End Time: " + eTime);
        }

        name.setForeground(Color.white);
        des.setForeground(Color.white);
        start.setForeground(Color.white);
        end.setForeground(Color.white);
        repeats.setForeground(Color.white);

        // Adding components to panel
        panelDisplay.add(name);
        panelDisplay.add(des);
        panelDisplay.add(start);
        panelDisplay.add(end);
        if (!repeats.getText().equals(null)) {
            panelDisplay.add(repeats);
            panelDisplay.setLayout(new GridLayout(6, 1));
        }
        else {
            panelDisplay.setLayout(new GridLayout(5, 1));
        }

        delete.addActionListener(this);
```

```
                panelDisplay.add(delete);

                // Adding panel to frame
                fDisplay.add(panelDisplay);
                fDisplay.setSize(400, 400);
                fDisplay.setVisible(true);
            }
        }
        catch (FileNotFoundException fnfe) {
            fnfe.printStackTrace();
            display.setText("No events found for this day.");

            display.setForeground(Color.white);
            panelDisplay.add(display);

            panelDisplay.setLayout(new GridLayout(2, 1));

            delete.addActionListener(this);
            panelDisplay.add(delete);

            // Adding panel to frame
            fDisplay.add(panelDisplay);
            fDisplay.setSize(400, 400);
```

```
                fDisplay.setVisible(true);
            }
        }

        public void deleteEvent() throws IOException {
            // Declaring file name
            eventDate = monthName + " " + ((JTextField) object).getText() + ", " +
year.format(today);

            // Declaring file writer
            FileWriter fw;

            try {
                fw = new FileWriter(directory + "/" + eventDate + ".txt");
                fw.write("");

                // Closing file writer
                fw.close();
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
        public void mouseClicked(MouseEvent e) {
                // ALL CALENDAR TAB EVENTS
                object = e.getSource();
                displayOrAdd();
}

        public void mouseEntered(MouseEvent e) {}

        public void mouseExited(MouseEvent e) {}

        public void mousePressed(MouseEvent e) {}

        public void mouseReleased(MouseEvent e) {}

        public void actionPerformed(ActionEvent e) {
                // ALL MAIN PROGRAM EVENTS
                if (e.getSource() == buttonCal) {
                        new Calendar().calendar();
                        fMain.dispose();
                }
                else if (e.getSource() == buttonLists) {
                        try {
```

```
                new StudentLists().studentLists();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonFinance) {
            try {
                new FinancialRecords().finance();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonRecords) {
            try {
                new StudentRecords().studentRecords();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
```

```
            }
}
else if (e.getSource() == buttonNotepad) {
            try {
                        new Notepad();
                        fMain.dispose();
            }
            catch (IOException ioe) {
                        ioe.printStackTrace();
            }
}
else if (e.getSource() == buttonHelp) {
            new HelpCal();
}
else if (e.getSource() == signOut) {
            fMain.dispose();
}

// ALL CALENDAR TAB EVENTS
if (e.getSource() == leftArrow) {
            today = today.minusMonths(1);
            fMain.remove(panelCal);
            calendar();
```

```
}
if (e.getSource() == rightArrow) {
        today = today.plusMonths(1);
        fMain.remove(panelCal);
        calendar();
}

// displayOrAdd buttons
if (e.getSource() == addNewEvent) {
        fDisplayOrAdd.dispose();
        addCalEvent(object);
}
else if (e.getSource() == display) {
        try {
                fDisplayOrAdd.dispose();
                displayEvents();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == delete) {
        try {
```

```
                    fDisplay.dispose();
                    deleteEvent();
                }
                catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }

            // addCalEvent buttons
            if (e.getSource() == addEvent) {
                try {
                    saveCalEvent(eventName, description, eventSTime, eventETime,
daily, weekly);
                    fCalEvents.dispose();
                }
                catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }
            else if (e.getSource() == cancel) {
                fCalEvents.dispose();
            }
        }
```

```
}
```

Student Lists Code:

```
/* Notes:
 * Status: FINISHED
 * Currently saves multiple students' contact information
 * Currently loads previously saved students' info
 * Currently deletes students' info
 * Currently displays newly added student info
 * Currently sorts first names & last names from A-Z */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class StudentLists extends MainProgram {
        private JFrame fNewStudent, listOfNames;
        private JPanel mainButtons, listsButtons, panelNewStu, buttons;
        private JTextField[][] students = new JTextField[8][5];
        private JTextField[] studentNames;
        private JTextField fName, lName, phone, mail;
        private JButton sortAZ_First, sortAZ_Last, addStudent, deleteStudent, addNew, cancel;
        private JButton delete, cancelDelete;
```

```
private Object object;
private File directory = new File(username + "/StudentLists");

public void studentLists() throws IOException {
        // New panel
        mainButtons = new JPanel();
        mainButtons.setLayout(new GridLayout(1, 7));
        mainButtons.setBackground(Color.gray);

        listsButtons = new JPanel();
        listsButtons.setLayout(new GridLayout(1, 4));

        panelLists = new JPanel();
        panelLists.setLayout(new GridLayout(8, 5));

        // New big panel
        buttons = new JPanel();
        buttons.setLayout(new GridLayout(2, 1));

        // New buttons
        sortAZ_First = new JButton("Sort A-Z (First Names)");
        sortAZ_First.setBackground(c);
        sortAZ_First.addActionListener(this);
```

```
sortAZ_Last = new JButton("Sort A-Z (Last Names)");
sortAZ_Last.setBackground(c);
sortAZ_Last.addActionListener(this);

addStudent = new JButton("Add a Student");
addStudent.setBackground(c);
addStudent.addActionListener(this);

deleteStudent = new JButton("Delete a Student");
deleteStudent.setBackground(c);
deleteStudent.addActionListener(this);

// New textfields
for (int rows = 0; rows < 8; rows++) {
        for (int columns = 0; columns < 5; columns++) {
                students[rows][columns] = new JTextField();
                students[rows][columns].setEditable(false);
                panelLists.add(students[rows][columns]);
        }
}

// Updating textfields if there are pre-saved students already
```

```
try {
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        int ch;
        int r = 0, c = 0;
        boolean maxCol = false;
        String s = "";

        if (fr.ready()) {
            do {
                while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
                    s += (char) ch;
                }

                // ensures that empty lines (deleted students) are not added
                if (!(s.equals(""))) {
                    students[r][c].setText(s);

                    s = "";

                    if (c == 4) {
                        maxCol = true;
                    }
```

```
                    if (maxCol == false) {
                        c++;
                    }
                    else {
                        if (r == 7) {
                            break;
                        }
                        else {
                            r++;
                            c = 0;
                            maxCol = false;
                        }
                    }
                }
            } while (fr.ready());
        }

        fr.close();
    }
    catch (FileNotFoundException fnfe) {
        fnfe.printStackTrace();
    }
```

```
// Adding components to main buttons panel
mainButtons.add(buttonCal);
mainButtons.add(buttonLists);
mainButtons.add(buttonRecords);
mainButtons.add(buttonFinance);
mainButtons.add(buttonNotepad);
mainButtons.add(buttonHelp);
mainButtons.add(signOut);

buttonLists.setEnabled(false);
buttonLists.setBackground(Color.pink);

// Adding components to lists buttons panel
listsButtons.add(sortAZ_First);
listsButtons.add(sortAZ_Last);
listsButtons.add(addStudent);
listsButtons.add(deleteStudent);

// Adding all buttons to buttons panel
buttons.add(mainButtons);
buttons.add(listsButtons);
```

```
        // Adding panel to frame
        fMain.add(buttons);
        fMain.add(panelLists);
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void studentListsUpdated() {
        // Clearing panels and frame
        panelLists.removeAll();
        fMain.remove(panelLists);

        // New textfields
        for (int rows = 0; rows < 8; rows++) {
                for (int columns = 0; columns < 5; columns++) {
                        panelLists.add(students[rows][columns]);
                }
        }

        // Adding panel to frame
        fMain.add(panelLists);
```

```
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void addAStudent() {
        // New frame
        fNewStudent = new JFrame("Add Student Information");

        // New panel
        panelNewStu = new JPanel();
        panelNewStu.setLayout(new GridLayout(5, 2));

        // New labels
        JLabel firstName = new JLabel("First Name:");
        JLabel lastName = new JLabel("Last Name:");
        JLabel phoneNum = new JLabel("Phone Number:");
        JLabel email = new JLabel("Email:");

        // New textfields
        fName = new JTextField();
        lName = new JTextField();
```

```
phone = new JTextField();
mail = new JTextField();

// New button
addNew = new JButton("Add Student");
addNew.addActionListener(this);

cancel = new JButton("Cancel");
cancel.addActionListener(this);

// Adding components to panel
panelNewStu.add(firstName);
panelNewStu.add(fName);
panelNewStu.add(lastName);
panelNewStu.add(lName);
panelNewStu.add(phoneNum);
panelNewStu.add(phone);
panelNewStu.add(email);
panelNewStu.add(mail);
panelNewStu.add(addNew);
panelNewStu.add(cancel);

// Adding panel to frame
```

```
            fNewStudent.add(panelNewStu);
            fNewStudent.setSize(400, 400);
            fNewStudent.setVisible(true);
        }

        public void savingStudent() throws IOException {
            // Creating directory
            directory.mkdir();

            // ensure that an empty student will not be added
            if (!(fName.getText().equals("") && lName.getText().equals("") &&
phone.getText().equals("") && mail.getText().equals(""))) {
                // Declaring file writer
                String s = fName.getText() + lName.getText();
                s = s.replaceAll("\\s", "");

                FileWriter fw = new FileWriter(directory + "/" + s + ".txt");

                // Writing to the file
                // first name
                for (int i = 0; i < fName.getText().length(); i++) {
                    fw.write(fName.getText().charAt(i));
                }
```

```
fw.write("\n");

// last name
for (int i = 0; i < lName.getText().length(); i++) {
        fw.write(lName.getText().charAt(i));
}

fw.write("\n");

// phone number
for (int i = 0; i < phone.getText().length(); i++) {
        fw.write(phone.getText().charAt(i));
}

fw.write("\n");

// email
for (int i = 0; i < mail.getText().length(); i++) {
        fw.write(mail.getText().charAt(i));
}

// Closing file writer
```

```
            fw.close();
}

// SAVING ALL STUDENTS
// Declaring file writer
FileWriter fw = new FileWriter(directory + "/Students_MasterList.txt", true);

// Writing to the file
// first name
for (int i = 0; i < fName.getText().length(); i++) {
            fw.append(fName.getText().charAt(i));
}

fw.append(' ');

// last name
for (int i = 0; i < lName.getText().length(); i++) {
            fw.append(lName.getText().charAt(i));
}

fw.append('\n');

// Closing file writer
```

```
            fw.close();
}

public void deletePopUp() {
            // New frame
            listOfNames = new JFrame("Delete a Student");

            // New panels
            JPanel names = new JPanel();

            JPanel userPrompt = new JPanel();
            userPrompt.setBackground(Color.gray);

            JPanel buttons = new JPanel();
            buttons.setLayout(new GridLayout(1, 2));

            // New label
            JLabel prompt = new JLabel("Please select a student to delete.");

            // New buttons
            delete = new JButton("Delete");
            delete.addActionListener(this);
```

```
cancelDelete = new JButton("Cancel");
cancelDelete.addActionListener(this);

// Adding prompt to prompt panel
userPrompt.add(prompt);

// Determining how many students there are currently
int count = 0;

for (int rows = 0; rows < 8; rows++) {
        for (int columns = 0; columns < 5; columns++) {
                if (!(students[rows][columns].getText().equals(""))) {
                        count++;
                }
        }
}

// Adding names to names panel
int newCount = 0;

studentNames = new JTextField[count];

for (int rows = 0; rows < 8; rows++) {
```

```
                    for (int columns = 0; columns < 5; columns++) {
                        if (!(students[rows][columns].getText().equals(""))) {
                            studentNames[newCount] = new JTextField();
                            studentNames[newCount].setEditable(false);

studentNames[newCount].setText(students[rows][columns].getText());
                            studentNames[newCount].addMouseListener(this);
                            names.add(studentNames[newCount]);
                            newCount++;
                        }
                    }
                }

                names.setLayout(new GridLayout(count, 1));

                // New scrollable content pane
                JScrollPane scrPane = new JScrollPane(names);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDE
D);
```

```
        // Adding buttons to buttons panel
        buttons.add(delete);
        buttons.add(cancelDelete);

        // Adding panels to frame
        listOfNames.add(userPrompt);
        listOfNames.add(scrPane);
        listOfNames.add(buttons);

        listOfNames.setLayout(new GridLayout(3, 1));
        listOfNames.setSize(400, 400);
        listOfNames.setVisible(true);
}

public void deleteStudent() throws IOException {
        // Declaring file writer
        FileWriter fw;

        // Determining file name
        String originalName = ((JTextField) object).getText();
        String s = originalName.replaceAll("\\s", "");

        try {
```

```
                fw = new FileWriter(directory + "/" + s + ".txt");
                fw.write("");

                // Determining index of original name
                int indexRows = 0, indexColumns = 0;
                boolean found = false;

                for (int rows = 0; rows < 8; rows++) {
                        for (int columns = 0; columns < 5; columns++) {
                                if
(students[rows][columns].getText().equals(originalName) && found == false) {
                                        indexRows = rows;
                                        indexColumns = columns;
                                        found = true;
                                }
                        }
                }

                // Deleting the name from the display
                for (int rows = indexRows; rows < 8; rows++) {
                        for (int columns = indexColumns; columns < 5; columns++) {
                                if (columns != 4) {
```

```
students[rows][columns].setText(students[rows][columns + 1].getText());
                                        }
                                        else if (columns == 4 && rows != 7) {
                                                students[rows][columns].setText(students[rows +
1][0].getText());
                                        }
                                        else if (columns == 4 && rows == 7) {
                                                students[rows][columns].setText("");
                                        }
                                }
                        }

                        // Closing file writer
                        fw.close();

                        // Editing master list
                        FileReader frMaster = new FileReader(directory +
"/Students_MasterList.txt");

                        int ch;
                        String nameToDelete = "";
```

```
                while ((ch = frMaster.read()) != -1) {
                        nameToDelete += (char) ch;
                }

                frMaster.close();

                nameToDelete = nameToDelete.replace(originalName + "\n",
"").replace("\r", "");

                System.out.print(nameToDelete);

                FileWriter fwMaster = new FileWriter(directory +
"/Students_MasterList.txt");

                fwMaster.write(nameToDelete);

                fwMaster.close();
            }
            catch (Exception e) {
                e.printStackTrace();
            }
        }
```

```
	public void display() throws IOException {
		boolean updated = false;

		for (int rows = 0; rows < 8; rows++) {
			for (int columns = 0; columns < 5; columns++) {
				if (!(fName.getText().equals("") && lName.getText().equals(""))
&& updated == false) {
					if (students[rows][columns].getText().equals("")) {
						students[rows][columns].setText(fName.getText() +
" " + lName.getText());
						updated = true;
					}
				}
			}
		}

		updated = false;
	}

	public void sortFirst() {
		int count = 0, min_index;
		String temp;
		String[] array = new String[40];
```

```
// Creating a temporary 1D array
for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 5; j++) {
                array[count] = students[i][j].getText();
                count++;
        }
}

// Sorting process
// Moving the lower boundary of the unsorted section
for (int i = 0; i < array.length - 1; i++) {
        // Finding min element in the unsorted section
        min_index = i;

        for (int j = i + 1; j < array.length; j++) {
                if (!(array[j].equals(""))) {
                        if (array[j].compareTo(array[min_index]) < 0) {
                                min_index = j;
                        }
                }
        }
```

```
                // Swapping the found minimum element w/ first element
                temp = array[min_index];
                array[min_index] = array[i];
                array[i] = temp;
        }

        // Updating students lists
        count = 0;

        for (int i = 0; i < 8; i++) {
                for (int j = 0; j < 5; j++) {
                        students[i][j].setText(array[count]);
                        count++;
                }
        }
}

public void sortLast() throws IOException {
        int min_index, ch, count = 0;
        String lastName = "";
        String temp;
        String[] array = new String[40];
        String[] names = new String[40];
```

```
try {
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        // Determining how many students there are currently
        int ch2;
        String studentName = "";

        while (fr.ready()) {
                while ((ch2 = fr.read()) != -1 && ((char) ch2 != '\n')) {
                        studentName += (char) ch2;
                }

                if (!(studentName.equals(""))) {
                        count++;
                }

                studentName = "";
        }

        fr.close();
}
catch (FileNotFoundException fnfe) {
```

```
                fnfe.printStackTrace();
            }

            try {
                FileReader fr1 = new FileReader(directory + "/Students_MasterList.txt");

                // Creating temporary 1D arrays
                for (int i = 0; i < 40; i++) {
                    String fileName = "";

                    while ((ch = fr1.read()) != -1 && ((char) ch != '\n')) {
                        fileName += (char) ch;
                    }

                    if (!(fileName.equals(""))) {
                        names[i] = fileName;

                        lastName = fileName.substring(fileName.indexOf(" ") +
1).toLowerCase();

                        array[i] = lastName;
                    }
                }
```

```
fr1.close();


// Sorting process
// Moving the lower boundary of the unsorted section
for (int i = 0; i < count - 1; i++) {
        // Finding min element in the unsorted section
        min_index = i;


        for (int j = i + 1; j < count; j++) {
                if (array[j].compareTo(array[min_index]) < 0) {
                        min_index = j;
                }
                else if (array[j].compareTo(array[min_index]) == 0) {
                        if (names[j].compareTo(names[min_index]) < 0) {
                                min_index = j;
                        }
                }
        }


        // Swapping the found minimum element w/ first element
        temp = array[min_index];
        array[min_index] = array[i];
```

```
                    array[i] = temp;


                    temp = names[min_index];
                    names[min_index] = names[i];
                    names[i] = temp;
                }
            }
            catch (FileNotFoundException fnfe){
                fnfe.printStackTrace();
            }

            // Updating students lists
            count = 0;

            for (int i = 0; i < 8; i++) {
                for (int j = 0; j < 5; j++) {
                    students[i][j].setText(names[count]);
                    count++;
                }
            }
        }

        public void mouseClicked(MouseEvent e) {
```

```
            // ALL STUDENT LISTS EVENTS
            object = e.getSource();
}

    public void mouseEntered(MouseEvent e) {}

    public void mouseExited(MouseEvent e) {}

    public void mousePressed(MouseEvent e) {}

    public void mouseReleased(MouseEvent e) {}

    public void actionPerformed(ActionEvent e) {
            // ALL MAIN PROGRAM EVENTS
            if (e.getSource() == buttonCal) {
                    new Calendar().calendar();
                    fMain.dispose();
            }
            else if (e.getSource() == buttonLists) {
                    try {
                            new StudentLists().studentLists();
                            fMain.dispose();
                    }
```

```
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonFinance) {
            try {
                new FinancialRecords().finance();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonRecords) {
            try {
                new StudentRecords().studentRecords();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonNotepad) {
```

```
            try {
                    new Notepad();
                    fMain.dispose();
            }
            catch (IOException ioe) {
                    ioe.printStackTrace();
            }
    }
    else if (e.getSource() == buttonHelp) {
            new HelpLists();
    }
    else if (e.getSource() == signOut) {
            fMain.dispose();
    }

    // ALL STUDENT LISTS EVENTS
    if (e.getSource() == addStudent) {
            addAStudent();
    }
    else if (e.getSource() == deleteStudent) {
            deletePopUp();
    }
    else if (e.getSource() == sortAZ_First) {
```

```
            sortFirst();
            fMain.dispose();
            studentListsUpdated();
    }
    else if (e.getSource() == sortAZ_Last){
            try {
                    sortLast();
                    fMain.dispose();
                    studentListsUpdated();
            }
            catch (IOException ioe) {
                    ioe.printStackTrace();
            }
    }

    // Adding new student events
    if (e.getSource() == addNew) {
            try {
                    savingStudent();
                    display();
                    fNewStudent.dispose();
                    fMain.dispose();
                    studentListsUpdated();
```

```
            }
            catch (IOException ioe) {
                  ioe.printStackTrace();
            }
}
else if (e.getSource() == cancel) {
            fNewStudent.dispose();
}

// Deleting student events
if (e.getSource() == delete) {
            try {
                  deleteStudent();
                  listOfNames.dispose();
                  fMain.dispose();
                  studentListsUpdated();
            }
            catch (IOException ioe) {
                  ioe.printStackTrace();
            }
}
else if (e.getSource() == cancelDelete) {
            listOfNames.dispose();
```

```
                }
        }
}
```

Student Records Code:

```
/* Notes:
 * Status: FINISHED
 * Currently displays all students' names
 * Currently has empty, editable textfields under labels for each student
 * Currently saves inputed text
 * Currently has scrollbar for both names and textfields
 * Currently sorts first names A-Z and last names A-Z
 * Currently searches for first occurrence of student name */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class StudentRecords extends MainProgram {
        private JFrame fSearch;
        private JLabel lResult = new JLabel();

        private JPanel mainButtons, top, names, editableArea;
```

```
private JButton sortAZ_First, sortAZ_Last, search, save, find;
private JTextField name;
private JTextField[] studentNames;
private JTextArea[][] textbox;
private int count = 0;

private File directory = new File(username + "/StudentLists");
private File newDirectory = new File(username + "/StudentRecords");

public void studentRecords() throws IOException {
        // New panel
        mainButtons = new JPanel();
        mainButtons.setLayout(new GridLayout(1, 7));

        // Adding components to main buttons panel
        mainButtons.add(buttonCal);
        mainButtons.add(buttonLists);
        mainButtons.add(buttonRecords);
        mainButtons.add(buttonFinance);
        mainButtons.add(buttonNotepad);
        mainButtons.add(buttonHelp);
        mainButtons.add(signOut);
```

```
buttonRecords.setEnabled(false);
buttonRecords.setBackground(new Color(150, 250, 200));

// New buttons
sortAZ_First = new JButton("Sort A-Z (F/N)");
sortAZ_First.setBackground(c);
sortAZ_First.addActionListener(this);

sortAZ_Last = new JButton("Sort A-Z (L/N)");
sortAZ_Last.setBackground(c);
sortAZ_Last.addActionListener(this);

search = new JButton("Search");
search.setBackground(c);
search.addActionListener(this);

save = new JButton("Save");
save.setBackground(c);
save.addActionListener(this);

// New panels
JPanel sidebar = new JPanel();
sidebar.setLayout(new GridLayout(1, 1));
```

```
names = new JPanel();

JPanel buttonsLabels = new JPanel();
buttonsLabels.setLayout(new GridLayout(1, 2));

JPanel buttons = new JPanel();
buttons.setLayout(new GridLayout(1, 4));

JPanel labels = new JPanel();
labels.setLayout(new GridLayout(1, 4));

// Adding buttons to panel
buttons.add(sortAZ_First);
buttons.add(sortAZ_Last);
buttons.add(search);
buttons.add(save);

// New labels
JLabel contactInfo = new JLabel("Contact Information");
contactInfo.setVisible(true);

JLabel currentRep = new JLabel("Current Repertoire");
```

```
currentRep.setVisible(true);

JLabel lessonDate = new JLabel("Lesson Date");
lessonDate.setVisible(true);

JLabel notes = new JLabel("Notes");
notes.setVisible(true);

// Adding labels to panel
labels.add(contactInfo);
labels.add(currentRep);
labels.add(lessonDate);
labels.add(notes);

// Adding panels to panel
buttonsLabels.add(buttons);
buttonsLabels.add(labels);

// New textfields
try {
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        // Determining how many students there are currently
```

```
int ch;
String studentName = "";

while (fr.ready()) {
	while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
		studentName += (char) ch;
	}

	count++;
	studentName = "";
}

fr.close();

String[] students = new String[count];
count = 0;
studentName = "";

FileReader fr2 = new FileReader(directory + "/Students_MasterList.txt");

while (fr2.ready()) {
	while ((ch = fr2.read()) != -1 && ((char) ch != '\n')) {
		studentName += (char) ch;
```

```
            }

            if (!(studentName.equals(""))) {
                students[count] = studentName;
                count++;
            }
            studentName = "";
        }

        fr2.close();

        // Adding names to names panel
        studentNames = new JTextField[count];

        for (int i = 0; i < count; i++) {
            studentNames[i] = new JTextField();
            studentNames[i].setEditable(false);
            studentNames[i].setText((i + 1) + ". " + students[i]);
            names.add(studentNames[i]);
        }

        names.setLayout(new GridLayout(count, 1));
```

```
                    // New scrollable content pane
                    JScrollPane scrPane = new JScrollPane(names);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                    sidebar.add(scrPane);
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    JLabel na = new JLabel("No students' records available.");
                    na.setVisible(true);

                    names.add(na);
                    // Smth like "No students available"

                    sidebar.add(names);
                }

                // New panel
```

```
editableArea = new JPanel();
editableArea.setLayout(new GridLayout(count, 4));

// New textfields
textbox = new JTextArea[count][4];

// Reading info from text files
for (int rows = 0; rows < count; rows++) {
    for (int columns = 0; columns < 4; columns++) {
        String fileName = "";

        for (int i = 0; i < studentNames[rows].getText().length(); i++) {
            if (studentNames[rows].getText().charAt(i) >= 65) {
                fileName +=
studentNames[rows].getText().charAt(i);
            }
        }

        fileName = fileName.replace(" ", "").replace("\r", "");

        if (columns == 0) {
            try {
                // Reading contact info
```

```
                    FileReader frContact = new FileReader(directory +
"/" + fileName + ".txt");
                    int chContact;
                    String contactInput = "";

                    while ((chContact = frContact.read()) != -1) {
                        contactInput += (char) chContact;
                    }

                    frContact.close();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(contactInput);
                    textbox[rows][columns].setEditable(true);
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
```

```
                }
                else if (columns == 1){
                        try {
                                // Reading other boxes
                                FileReader frMain = new FileReader(newDirectory
+ "/" + fileName + "Rep.txt");
                                int newCh;
                                String input = "";

                                while ((newCh = frMain.read()) != -1) {
                                        input += (char) newCh;
                                }

                                // Setting textbox
                                textbox[rows][columns] = new JTextArea();
                                textbox[rows][columns].setText(input);
                                textbox[rows][columns].setEditable(true);

                                frMain.close();
                        }
                        catch (FileNotFoundException fnfe) {
                                fnfe.printStackTrace();
```

```
                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setEditable(true);
                    }
                }
                else if (columns == 2){
                    try {
                        // Reading other boxes
                        FileReader frMain2 = new
FileReader(newDirectory + "/" + fileName + "LessonDate.txt");
                        int newCh;
                        String input = "";

                        while ((newCh = frMain2.read()) != -1) {
                            input += (char) newCh;
                        }

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setText(input);
                        textbox[rows][columns].setEditable(true);

                        frMain2.close();
```

```
                }
                catch (FileNotFoundException fnfe) {
                        fnfe.printStackTrace();

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setEditable(true);
                }
        }
        else {
                try {
                        // Reading other boxes
                        FileReader frMain3 = new
FileReader(newDirectory + "/" + fileName + "Notes.txt");
                        int newCh;
                        String input = "";

                        while ((newCh = frMain3.read()) != -1) {
                                input += (char) newCh;
                        }

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
```

```
                                textbox[rows][columns].setText(input);
                                textbox[rows][columns].setEditable(true);

                                frMain3.close();
                        }
                        catch (FileNotFoundException fnfe) {
                                fnfe.printStackTrace();

                                // Setting textbox
                                textbox[rows][columns] = new JTextArea();
                                textbox[rows][columns].setEditable(true);
                        }
                }
        }
}

for (int i = 0; i < count; i++) {
        for (int j = 0; j < 4; j++) {
                if (i % 2 == 0) {
                        if (j % 2 == 0) {
                                textbox[i][j].setBackground(Color.pink);
                        }
                }
```

```
                    else {
                        if (j % 2 != 0) {
                            textbox[i][j].setBackground(Color.pink);
                        }
                    }
                    editableArea.add(textbox[i][j]);
                }
            }

            // New scrollable content pane
            JScrollPane scrPane2 = new JScrollPane(editableArea);

scrPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);

            top = new JPanel();
            top.setLayout(new GridLayout(2, 1));
            top.add(mainButtons);
            top.add(buttonsLabels);

            panelRecords = new JPanel();
```

```
        panelRecords.setLayout(new GridLayout(1, 2));
        panelRecords.add(sidebar);
        panelRecords.add(scrPane2);

        // Adding components frame
        fMain.add(top);
        fMain.add(panelRecords);
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void updateDisplay() throws IOException {
        // Clearing bottom half of the frame
        panelRecords.removeAll();
        names.removeAll();
        editableArea.removeAll();
        fMain.remove(panelRecords);

        // New textfields
        for (int i = 0; i < count; i++) {
                names.add(studentNames[i]);
```

```
        }

        // New textfields
        textbox = new JTextArea[count][4];

        // Reading info from text files
        for (int rows = 0; rows < count; rows++) {
            for (int columns = 0; columns < 4; columns++) {
                String fileName = "";

                for (int i = 0; i < studentNames[rows].getText().length(); i++) {
                    if (studentNames[rows].getText().charAt(i) >= 65) {
                        fileName +=
studentNames[rows].getText().charAt(i);
                    }
                }

                fileName = fileName.replace(" ", "").replace("\r", "");

                if (columns == 0) {
                    try {
                        // Reading contact info
```

```
                    FileReader frContact = new FileReader(directory +
"/" + fileName + ".txt");
                    int chContact;
                    String contactInput = "";

                    while ((chContact = frContact.read()) != -1) {
                        contactInput += (char) chContact;
                    }

                    frContact.close();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(contactInput);
                    textbox[rows][columns].setEditable(true);
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
```

```
                }
                else if (columns == 1){
                        try {
                                // Reading other boxes
                                FileReader frMain = new FileReader(newDirectory
+ "/" + fileName + "Rep.txt");
                                int newCh;
                                String input = "";

                                while ((newCh = frMain.read()) != -1) {
                                        input += (char) newCh;
                                }

                                // Setting textbox
                                textbox[rows][columns] = new JTextArea();
                                textbox[rows][columns].setText(input);
                                textbox[rows][columns].setEditable(true);

                                frMain.close();
                        }
                        catch (FileNotFoundException fnfe) {
                                fnfe.printStackTrace();
```

```
                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setEditable(true);
                    }
                }
                else if (columns == 2){
                    try {
                        // Reading other boxes
                        FileReader frMain2 = new
FileReader(newDirectory + "/" + fileName + "LessonDate.txt");
                        int newCh;
                        String input = "";

                        while ((newCh = frMain2.read()) != -1) {
                            input += (char) newCh;
                        }

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setText(input);
                        textbox[rows][columns].setEditable(true);

                        frMain2.close();
```

```
                }
                catch (FileNotFoundException fnfe) {
                        fnfe.printStackTrace();

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setEditable(true);
                }
        }
        else {
                try {
                        // Reading other boxes
                        FileReader frMain3 = new
FileReader(newDirectory + "/" + fileName + "Notes.txt");
                        int newCh;
                        String input = "";

                        while ((newCh = frMain3.read()) != -1) {
                                input += (char) newCh;
                        }

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
```

```
                                textbox[rows][columns].setText(input);
                                textbox[rows][columns].setEditable(true);

                                frMain3.close();
                        }
                        catch (FileNotFoundException fnfe) {
                                fnfe.printStackTrace();

                                // Setting textbox
                                textbox[rows][columns] = new JTextArea();
                                textbox[rows][columns].setEditable(true);
                        }
                }
        }
}

for (int i = 0; i < count; i++) {
        for (int j = 0; j < 4; j++) {
                if (i % 2 == 0) {
                        if (j % 2 == 0) {
                                textbox[i][j].setBackground(Color.pink);
                        }
                }
```

```
                else {
                    if (j % 2 != 0) {
                        textbox[i][j].setBackground(Color.pink);
                    }
                }
                editableArea.add(textbox[i][j]);
            }
        }

        // New scrollable content pane
        JScrollPane scrPane2 = new JScrollPane(editableArea);

scrPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);

        // Adding panels to panel
        panelRecords.add(names);
        panelRecords.add(editableArea);

        // Adding panel to frame
        fMain.add(panelRecords);
```

```
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void sortFirstNames() throws IOException {
        int min_index;
        String temp;
        String[] array = new String[studentNames.length];
        int ch, count = 0;
        String studentName = "";

        // Temporary 1D array
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        while (fr.ready()) {
                while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
                        studentName += (char) ch;
                }

                if (!(studentName.equals(""))) {
                        array[count] = studentName;
```

```
                count++;
            }
            studentName = "";
        }

        fr.close();

        // Sorting process
        // Moving the lower boundary of the unsorted section
        for (int i = 0; i < array.length - 1; i++) {
            // Finding min element in the unsorted section
            min_index = i;

            for (int j = i + 1; j < array.length; j++) {
                if (array[j].compareTo(array[min_index]) < 0) {
                    min_index = j;
                }
            }

            // Swapping the found minimum element w/ first element
            temp = array[min_index];
            array[min_index] = array[i];
            array[i] = temp;
```

```
        }

        // Updating display
        for (int i = 0; i < count; i++) {
            studentNames[i].setText((i + 1) + ". " + array[i]);
        }
    }

    public void sortLastNames() throws IOException {
        int min_index, ch;
        String lastName = "";
        String temp;
        String[] array = new String[40];
        String[] names = new String[40];

        try {
            FileReader fr1 = new FileReader(directory + "/Students_MasterList.txt");

            // Creating temporary 1D arrays
            for (int i = 0; i < count; i++) {
                String fileName = "";

                while ((ch = fr1.read()) != -1 && ((char) ch != '\n')) {
```

```
                fileName += (char) ch;
            }

            if (!(fileName.equals(""))) {
                names[i] = fileName;

                lastName = fileName.substring(fileName.indexOf(" ") +
1).toLowerCase();

                array[i] = lastName;
            }
        }

        fr1.close();

        // Sorting process
        // Moving the lower boundary of the unsorted section
        for (int i = 0; i < count - 1; i++) {
            // Finding min element in the unsorted section
            min_index = i;

            for (int j = i + 1; j < count; j++) {
                if (array[j].compareTo(array[min_index]) < 0) {
```

```
                        min_index = j;
                    }
                    else if (array[j].compareTo(array[min_index]) == 0) {
                        if (names[j].compareTo(names[min_index]) < 0) {
                            min_index = j;
                        }
                    }
                }

                // Swapping the found minimum element w/ first element
                temp = array[min_index];
                array[min_index] = array[i];
                array[i] = temp;

                temp = names[min_index];
                names[min_index] = names[i];
                names[i] = temp;
            }
        }
        catch (FileNotFoundException fnfe){
            fnfe.printStackTrace();
        }
```

```
        // Updating display
        for (int i = 0; i < count; i++) {
                studentNames[i].setText((i + 1) + ". " + names[i]);
        }
}

public void save() throws IOException, NullPointerException {
        // Creating new directory
        newDirectory.mkdir();

        // Writing info to text files
        for (int rows = 0; rows < studentNames.length; rows++) {
                String original = "";
                String fileName;

                for (int k = 0; k < studentNames[rows].getText().length(); k++) {
                        if (studentNames[rows].getText().charAt(k) >= 65) {
                                original += studentNames[rows].getText().charAt(k);
                        }
                }

                fileName = original.replace(" ", "").replace("\r", "");
```

```
                    for (int columns = 0; columns < 4; columns++) {
                        // Editing student lists to update contact info
                        if (columns == 0) {
                            FileWriter fwContact = new FileWriter(directory + "/" +
fileName + ".txt");
                            fwContact.write(textbox[rows][columns].getText());
                            fwContact.close();

                            FileReader master = new FileReader(directory +
"/Students_MasterList.txt");
                            String s = "", newName = "";
                            int ch;
                            int num = 0;

                            // Reading old lists
                            while ((ch = master.read()) != -1) {
                                s += (char) ch;
                            }

                            master.close();

                            for (int i = 0; i <
textbox[rows][columns].getText().length(); i++) {
```

```
                                        if (textbox[rows][columns].getText().charAt(i) ==
'\n') {
                                                num++;

                                                if (num < 2) {
                                                        newName += " ";
                                                }
                                        }
                                        else {
                                                newName +=
textbox[rows][columns].getText().charAt(i);
                                        }

                                        if (num >= 2) {
                                                i =
textbox[rows][columns].getText().length() - 1;
                                                }
                                }

                                s = s.replaceAll(original, newName);

                                FileWriter fwMaster = new FileWriter(directory +
"/Students_MasterList.txt");
```

```
                    fwMaster.write(s);


                    fwMaster.close();
                }
                // Writing other inputed info to file
                else if (columns == 1) {
                    FileWriter fw = new FileWriter(newDirectory + "/" +
fileName + "Rep.txt");


                    fw.write(textbox[rows][columns].getText());


                    fw.close();
                }
                else if (columns == 2) {
                    FileWriter fw2 = new FileWriter(newDirectory + "/" +
fileName + "LessonDate.txt");


                    fw2.write(textbox[rows][columns].getText());


                    fw2.close();
                }
                else {
```

```
                                        FileWriter fw3 = new FileWriter(newDirectory + "/" +
fileName + "Notes.txt");

                                        fw3.write(textbox[rows][columns].getText());

                                        fw3.close();
                                    }
                                }
                            }
                        }

                        public void search() {
                            // New frame
                            fSearch = new JFrame("Search for Student");
                            fSearch.setSize(200, 200);
                            fSearch.setLayout(new GridLayout(3, 1));
                            fSearch.setVisible(true);

                            // New panel
                            JPanel prompt = new JPanel();

                            JPanel process = new JPanel();
                            process.setLayout(new GridLayout(1, 2));
```

```
JPanel result = new JPanel();

// New labels
JLabel lPrompt = new JLabel("Enter the name to search for:");

// New textfield
name = new JTextField();
name.setEditable(true);

// New button
find = new JButton("Search");
find.addActionListener(this);

// Adding components to panels
prompt.add(lPrompt);
process.add(name);
process.add(find);
result.add(lResult);

// Adding panels to frame
fSearch.add(prompt);
fSearch.add(process);
```

```
            fSearch.add(result);
        }

        public void find() {
            int num = 0;
            boolean found = false;

            while (found != true && num < count) {
                if
((studentNames[num].getText().toLowerCase()).contains(name.getText().toLowerCase())) {
                    lResult.setText(name.getText() + " is number " + (num + 1) + ".");
                    found = true;
                }
                else {
                    lResult.setText("Student not found.");
                }
                num++;
            }
        }

        public void actionPerformed(ActionEvent e) {
            // ALL MAIN PROGRAM EVENTS
            if (e.getSource() == buttonCal) {
```

```
            try {
                    save();
                    new Calendar().calendar();
                    fMain.dispose();
            }
            catch (IOException ioe) {
                    ioe.printStackTrace();
                    new Calendar().calendar();
                    fMain.dispose();
            }
            catch (NullPointerException npe) {
                    npe.fillInStackTrace();
                    new Calendar().calendar();
                    fMain.dispose();
            }
    }
    else if (e.getSource() == buttonLists) {
            try {
                    save();

                    try {
                            new StudentLists().studentLists();
                            fMain.dispose();
```

```
            }
            catch (IOException ioe) {
                  ioe.printStackTrace();
            }
      }
      catch (IOException ioe) {
            ioe.printStackTrace();

            try {
                  new StudentLists().studentLists();
                  fMain.dispose();
            }
            catch (IOException ioe2) {
                  ioe2.printStackTrace();
            }
      }
      catch (NullPointerException npe) {
            npe.fillInStackTrace();

            try {
                  new StudentLists().studentLists();
                  fMain.dispose();
            }
```

```
                catch (IOException ioe) {
                        ioe.printStackTrace();
                }
        }
}
else if (e.getSource() == buttonFinance) {
        try {
                new FinancialRecords().finance();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == buttonRecords) {
        try {
                new StudentRecords().studentRecords();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
```

```
else if (e.getSource() == buttonNotepad) {
        try {
                new Notepad();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == buttonHelp) {
        new HelpRecords();
}
else if (e.getSource() == signOut) {
        try {
                save();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
                fMain.dispose();
        }
        catch (NullPointerException npe) {
                npe.fillInStackTrace();
```

```
                fMain.dispose();
        }
}

// ALL STUDENT RECORDS EVENTS
if (e.getSource() == sortAZ_First) {
        try {
                sortFirstNames();
                fMain.dispose();
                updateDisplay();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == sortAZ_Last) {
        try {
                sortLastNames();
                fMain.dispose();
                updateDisplay();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
```

```
                }
        }
        else if (e.getSource() == save) {
                try {
                        save();
                }
                catch (IOException ioe) {
                        ioe.printStackTrace();
                }
        }
        else if (e.getSource() == search) {
                search();
        }

        // Search button
        if (e.getSource() == find) {
                find();
                fSearch.dispose();
                search();
        }
}

public void windowClosing(WindowEvent e) {
```

```
                try {
                        save();
                        fMain.dispose();
                }
                catch (IOException ioe) {
                        ioe.printStackTrace();
                        fMain.dispose();
                }
                catch (NullPointerException npe) {
                        npe.fillInStackTrace();
                        fMain.dispose();
                }
        }
}
```

Financial Records Code:

```
/* Notes:
 * Status: FINISHED
 * Currently displays all students' names
 * Currently has empty, editable textfields under labels for each student
 * Currently saves inputed text
 * Currently has scrollbar for both names and textfields
 * Currently sorts first names A-Z
 * Currently searches for first occurrence of student name
```

```
 * Currently has functional calculator */


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.text.DecimalFormat;


public class FinancialRecords extends MainProgram {
        private JFrame fSearch, fCalculator;
        private JPanel text, numberButtons, operations, buttons;
        private JButton one, two, three, four, five, six, seven, eight, nine, zero, decimal;
        private JButton add, subtract, multiply, divide, equal, ans, leftP, rightP, clear;
        private JLabel displayArea = new JLabel(), result = new JLabel();
        private double ansDouble = 0;


        private JPanel mainButtons, top, names, editableArea;
        private JButton sortAZ_First, calculator, search, save, find;
        private JLabel lResult = new JLabel();
        private JCheckBox[][] frequencies;
        private JTextField name;
        private JTextField[] studentNames;
        private JTextArea[][] textbox;
```

```
private int count = 0;
private File directory = new File(username + "/StudentLists");
private File newDirectory = new File(username + "/FinancialRecords");

public void finance() throws IOException {
        // New panel
        mainButtons = new JPanel();
        mainButtons.setLayout(new GridLayout(1, 7));

        // Adding components to main buttons panel
        mainButtons.add(buttonCal);
        mainButtons.add(buttonLists);
        mainButtons.add(buttonRecords);
        mainButtons.add(buttonFinance);
        mainButtons.add(buttonNotepad);
        mainButtons.add(buttonHelp);
        mainButtons.add(signOut);

        buttonFinance.setEnabled(false);
        buttonFinance.setBackground(Color.pink);

        // New buttons
        sortAZ_First = new JButton("Sort A-Z (F/N)");
```

```
sortAZ_First.setBackground(c);
sortAZ_First.addActionListener(this);

calculator = new JButton("Calculator");
calculator.setBackground(c);
calculator.addActionListener(this);

search = new JButton("Search");
search.setBackground(c);
search.addActionListener(this);

save = new JButton("Save");
save.setBackground(c);
save.addActionListener(this);

// New panels
JPanel sidebar = new JPanel();
sidebar.setLayout(new GridLayout(1, 1));

names = new JPanel();

JPanel buttonsLabels = new JPanel();
buttonsLabels.setLayout(new GridLayout(1, 2));
```

```
JPanel buttons = new JPanel();
buttons.setLayout(new GridLayout(1, 4));

JPanel labels = new JPanel();
labels.setLayout(new GridLayout(1, 4));

// Adding buttons to panel
buttons.add(sortAZ_First);
buttons.add(calculator);
buttons.add(search);
buttons.add(save);

// New labels
JLabel payFreq = new JLabel("Payment Frequency");
payFreq.setVisible(true);

JLabel payAmount = new JLabel("Payment Amount");
payAmount.setVisible(true);

JLabel payDay = new JLabel("Next Payment Date");
payDay.setVisible(true);
```

```
JLabel notes = new JLabel("Notes");
notes.setVisible(true);

// Adding labels to panel
labels.add(payFreq);
labels.add(payAmount);
labels.add(payDay);
labels.add(notes);

// Adding panels to panel
buttonsLabels.add(buttons);
buttonsLabels.add(labels);

// New textfields
try {
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        // Determining how many students there are currently
        int ch;
        String studentName = "";

        while (fr.ready()) {
                while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
```

```
                studentName += (char) ch;
        }

        count++;
        studentName = "";
}

fr.close();

String[] students = new String[count];
count = 0;
studentName = "";

FileReader fr2 = new FileReader(directory + "/Students_MasterList.txt");

while (fr2.ready()) {
        while ((ch = fr2.read()) != -1 && ((char) ch != '\n')) {
                studentName += (char) ch;
        }

        if (!(studentName.equals(""))) {
                students[count] = studentName;
                count++;
```

```
            }
            studentName = "";
        }

        fr2.close();

        // Adding names to names panel
        studentNames = new JTextField[count];

        for (int i = 0; i < count; i++) {
            studentNames[i] = new JTextField();
            studentNames[i].setEditable(false);
            studentNames[i].setText((i + 1) + ". " + students[i]);
            names.add(studentNames[i]);
        }

        names.setLayout(new GridLayout(count, 1));

        // New scrollable content pane
        JScrollPane scrPane = new JScrollPane(names);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
```

```
scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                sidebar.add(scrPane);
            }
            catch (FileNotFoundException fnfe) {
                fnfe.printStackTrace();

                JLabel na = new JLabel("No students' records available.");
                na.setVisible(true);

                names.add(na);
                // Smth like "No students available"

                sidebar.add(names);
            }

            // New panel
            editableArea = new JPanel();
            editableArea.setLayout(new GridLayout(count, 4));

            // New textfields
```

```
textbox = new JTextArea[count][4];

// Setting new checkbox string array
frequencies = new JCheckBox[count][3];

for (int i = 0; i < count; i++) {
    for (int j = 0; j < 3; j++) {
        if (j == 0) {
            frequencies[i][j] = new JCheckBox("upon lesson date",
false);
        }
        else if (j == 1) {
            frequencies[i][j] = new JCheckBox("weekly", false);
        }
        else {
            frequencies[i][j] = new JCheckBox("monthly", false);
        }

        frequencies[i][j].setOpaque(false);
    }
}

// Reading info from text files
```

```
                for (int rows = 0; rows < count; rows++) {
                        for (int columns = 0; columns < 4; columns++) {
                                String fileName = "";

                                for (int i = 0; i < studentNames[rows].getText().length(); i++) {
                                        if (studentNames[rows].getText().charAt(i) >= 65) {
                                                fileName +=
studentNames[rows].getText().charAt(i);
                                        }
                                }

                                fileName = fileName.replace(" ", "").replace("\r", "");

                                if (columns == 0) {
                                        // Setting textbox
                                        textbox[rows][columns] = new JTextArea();
                                        textbox[rows][columns].setLayout(new GridLayout(3, 1));

                                        try {
                                                FileReader frMain = new FileReader(newDirectory
+ "/" + fileName + ".txt");
                                                int newCh;
                                                String input = "";
```

```
                while ((newCh = frMain.read()) != -1) {
                        input += (char) newCh;
                }

                frMain.close();

                if (input.equals("upon lesson date")) {
                        frequencies[rows][0].setSelected(true);

                        for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                        }
                }
                else if (input.equals("weekly")) {
                        frequencies[rows][1].setSelected(true);

                        for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                        }
                }
```

```
                else if (input.equals("monthly")) {
                        frequencies[rows][2].setSelected(true);

                        for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                        }
                }
                else {
                        for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                        }
                }

                        textbox[rows][columns].setEditable(false);
        }
        catch (FileNotFoundException fnfe) {
                for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                }
```

```
                    textbox[rows][columns].setEditable(false);
                }
            }
            else if (columns == 1){
                try {
                    // Reading other boxes
                    FileReader frMain1 = new
FileReader(newDirectory + "/" + fileName + "Amount.txt");
                    int newCh;
                    String input = "";

                    while ((newCh = frMain1.read()) != -1) {
                        input += (char) newCh;
                    }

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(input);
                    textbox[rows][columns].setEditable(true);

                    frMain1.close();
                }
                catch (FileNotFoundException fnfe) {
```

```
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
            }
            else if (columns == 2){
                try {
                    // Reading other boxes
                    FileReader frMain2 = new
FileReader(newDirectory + "/" + fileName + "Date.txt");
                    int newCh;
                    String input = "";

                    while ((newCh = frMain2.read()) != -1) {
                        input += (char) newCh;
                    }

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(input);
                    textbox[rows][columns].setEditable(true);
```

```
                    frMain2.close();
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
            }
            else {
                try {
                    // Reading other boxes
                    FileReader frMain3 = new
FileReader(newDirectory + "/" + fileName + "Notes.txt");
                    int newCh;
                    String input = "";

                    while ((newCh = frMain3.read()) != -1) {
                        input += (char) newCh;
                    }
```

```
                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(input);
                    textbox[rows][columns].setEditable(true);

                    frMain3.close();
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
            }
        }
    }

for (int i = 0; i < count; i++) {
        for (int j = 0; j < 4; j++) {
            if (i % 2 == 0) {
                if (j % 2 == 0) {
```

```
                                        textbox[i][j].setBackground(new Color(150, 250,
200));
                                }
                        }
                        else {
                                if (j % 2 != 0) {
                                        textbox[i][j].setBackground(new Color(150, 250,
200));
                                }
                        }
                        editableArea.add(textbox[i][j]);
                }
        }

        // New scrollable content pane
        JScrollPane scrPane2 = new JScrollPane(editableArea);

scrPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);

        top = new JPanel();
```

```
        top.setLayout(new GridLayout(2, 1));
        top.add(mainButtons);
        top.add(buttonsLabels);

        panelFinance = new JPanel();
        panelFinance.setLayout(new GridLayout(1, 2));
        panelFinance.add(sidebar);
        panelFinance.add(scrPane2);

        // Adding components frame
        fMain.add(top);
        fMain.add(panelFinance);
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void updateDisplay() throws IOException {
        // Clearing bottom half of the frame
        panelFinance.removeAll();
        names.removeAll();
        editableArea.removeAll();
```

```
fMain.remove(panelFinance);

// New textfields
for (int i = 0; i < count; i++) {
	names.add(studentNames[i]);
}

// New textfields
textbox = new JTextArea[count][4];

// Reading info from text files
for (int rows = 0; rows < count; rows++) {
	for (int columns = 0; columns < 4; columns++) {
		String fileName = "";

		for (int i = 0; i < studentNames[rows].getText().length(); i++) {
			if (studentNames[rows].getText().charAt(i) >= 65) {
				fileName +=
studentNames[rows].getText().charAt(i);
			}
		}

		fileName = fileName.replace(" ", "").replace("\r", "");
```

```
                    if (columns == 0) {
                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setLayout(new GridLayout(3, 1));

                        try {
                            FileReader frMain = new FileReader(newDirectory
+ "/" + fileName + ".txt");
                            int newCh;
                            String input = "";

                            while ((newCh = frMain.read()) != -1) {
                                input += (char) newCh;
                            }

                            frMain.close();

                            if (input.equals("upon lesson date")) {
                                frequencies[rows][0].setSelected(true);
                                frequencies[rows][1].setSelected(false);
                                frequencies[rows][2].setSelected(false);
```

```
                    for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                    }
                }
                else if (input.equals("weekly")) {
                    frequencies[rows][0].setSelected(false);
                    frequencies[rows][1].setSelected(true);
                    frequencies[rows][2].setSelected(false);

                    for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
                    }
                }
                else if (input.equals("monthly")) {
                    frequencies[rows][0].setSelected(false);
                    frequencies[rows][1].setSelected(false);
                    frequencies[rows][2].setSelected(true);

                    for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
```

```
                }
            }
            else {
                for (int i = 0; i < 3; i++) {

frequencies[rows][0].setSelected(false);

frequencies[rows][1].setSelected(false);

frequencies[rows][2].setSelected(false);



textbox[rows][columns].add(frequencies[rows][i]);
                }
            }

            textbox[rows][columns].setEditable(false);
        }
        catch (FileNotFoundException fnfe) {
            for (int i = 0; i < 3; i++) {

textbox[rows][columns].add(frequencies[rows][i]);
            }
```

```
                    textbox[rows][columns].setEditable(false);
                }
            }
            else if (columns == 1){
                try {
                    // Reading other boxes
                    FileReader frMain1 = new
FileReader(newDirectory + "/" + fileName + "Amount.txt");
                    int newCh;
                    String input = "";

                    while ((newCh = frMain1.read()) != -1) {
                        input += (char) newCh;
                    }

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(input);
                    textbox[rows][columns].setEditable(true);

                    frMain1.close();
                }
```

```
                catch (FileNotFoundException fnfe) {
                        fnfe.printStackTrace();

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setEditable(true);
                }
        }
        else if (columns == 2){
                try {
                        // Reading other boxes
                        FileReader frMain2 = new
FileReader(newDirectory + "/" + fileName + "Date.txt");
                        int newCh;
                        String input = "";

                        while ((newCh = frMain2.read()) != -1) {
                                input += (char) newCh;
                        }

                        // Setting textbox
                        textbox[rows][columns] = new JTextArea();
                        textbox[rows][columns].setText(input);
```

```
                    textbox[rows][columns].setEditable(true);


                    frMain2.close();
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();


                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
            }
            else {
                try {
                    // Reading other boxes
                    FileReader frMain3 = new
FileReader(newDirectory + "/" + fileName + "Notes.txt");
                    int newCh;
                    String input = "";


                    while ((newCh = frMain3.read()) != -1) {
                        input += (char) newCh;
                    }
```

```
                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setText(input);
                    textbox[rows][columns].setEditable(true);

                    frMain3.close();
                }
                catch (FileNotFoundException fnfe) {
                    fnfe.printStackTrace();

                    // Setting textbox
                    textbox[rows][columns] = new JTextArea();
                    textbox[rows][columns].setEditable(true);
                }
            }
        }
    }

    for (int i = 0; i < count; i++) {
        for (int j = 0; j < 4; j++) {
            if (i % 2 == 0) {
                if (j % 2 != 0) {
```

```
                                        textbox[i][j].setBackground(new Color(150, 250,
200));
                                }
                        }
                        else {
                                if (j % 2 == 0) {
                                        textbox[i][j].setBackground(new Color(150, 250,
200));
                                }
                        }
                        editableArea.add(textbox[i][j]);
                }
        }

        // New scrollable content pane
        JScrollPane scrPane2 = new JScrollPane(editableArea);

scrPane2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);

        // Adding panels to panel
```

```
        panelFinance.add(names);
        panelFinance.add(editableArea);

        // Adding panel to frame
        fMain.add(panelFinance);
        fMain.setLayout(new GridLayout(2, 1));
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void sortFirstNames() throws IOException {
        int min_index;
        String temp;
        String[] array = new String[studentNames.length];
        int ch, count = 0;
        String studentName = "";

        // Temporary 1D array
        FileReader fr = new FileReader(directory + "/Students_MasterList.txt");

        while (fr.ready()) {
                while ((ch = fr.read()) != -1 && ((char) ch != '\n')) {
```

```
                studentName += (char) ch;
            }

            if (!(studentName.equals(""))) {
                array[count] = studentName;
                count++;
            }
            studentName = "";
        }

        fr.close();

        // Sorting process
        // Moving the lower boundary of the unsorted section
        for (int i = 0; i < array.length - 1; i++) {
            // Finding min element in the unsorted section
            min_index = i;

            for (int j = i + 1; j < array.length; j++) {
                if (array[j].compareTo(array[min_index]) < 0) {
                    min_index = j;
                }
            }
```

```
                // Swapping the found minimum element w/ first element
                temp = array[min_index];
                array[min_index] = array[i];
                array[i] = temp;
        }

        // Updating display
        for (int i = 0; i < count; i++) {
                studentNames[i].setText((i + 1) + ". " + array[i]);
        }
}

public void save() throws IOException, NullPointerException {
        // Creating new directory
        newDirectory.mkdir();

        // Writing info to text files
        for (int rows = 0; rows < studentNames.length; rows++) {
                String fileName = "";

                for (int k = 0; k < studentNames[rows].getText().length(); k++) {
                        if (studentNames[rows].getText().charAt(k) >= 65) {
```

```
                    fileName += studentNames[rows].getText().charAt(k);
                }
            }

            fileName = fileName.replace(" ", "").replace("\r", "");

            for (int columns = 0; columns < 4; columns++) {
                // Writing checkbox name selected
                if (columns == 0) {
                    FileWriter fw = new FileWriter(newDirectory + "/" +
fileName + ".txt");

                    if (frequencies[rows][0].isSelected()) {
                        fw.write("upon lesson date");
                    }
                    else if (frequencies[rows][1].isSelected()) {
                        fw.write("weekly");
                    }
                    else if (frequencies[rows][2].isSelected()) {
                        fw.write("monthly");
                    }
                    else {
                    }
```

```
                    fw.close();
                }
                // Writing other inputed info to file
                else if (columns == 1) {
                    FileWriter fw1 = new FileWriter(newDirectory + "/" +
fileName + "Amount.txt");

                    fw1.write(textbox[rows][columns].getText());

                    fw1.close();
                }
                else if (columns == 2) {
                    FileWriter fw2 = new FileWriter(newDirectory + "/" +
fileName + "Date.txt");

                    fw2.write(textbox[rows][columns].getText());

                    fw2.close();
                }
                else {
                    FileWriter fw3 = new FileWriter(newDirectory + "/" +
fileName + "Notes.txt");
```

```
                                fw3.write(textbox[rows][columns].getText());


                                fw3.close();
                        }
                }
        }
}

public void search() {
        // New frame
        fSearch = new JFrame("Search for Student");
        fSearch.setSize(200, 200);
        fSearch.setLayout(new GridLayout(3, 1));
        fSearch.setVisible(true);

        // New panel
        JPanel prompt = new JPanel();

        JPanel process = new JPanel();
        process.setLayout(new GridLayout(1, 2));

        JPanel result = new JPanel();
```

```
        // New labels
        JLabel lPrompt = new JLabel("Enter the name to search for:");

        // New textfield
        name = new JTextField();
        name.setEditable(true);

        // New button
        find = new JButton("Search");
        find.addActionListener(this);

        // Adding components to panels
        prompt.add(lPrompt);
        process.add(name);
        process.add(find);
        result.add(lResult);

        // Adding panels to frame
        fSearch.add(prompt);
        fSearch.add(process);
        fSearch.add(result);
}
```

```
	public void find() {
		int num = 0;
		boolean found = false;

		while (found != true && num < count) {
			if
((studentNames[num].getText().toLowerCase()).contains(name.getText().toLowerCase())) {
				lResult.setText(name.getText() + " is number " + (num + 1) + ".");
				found = true;
			}
			else {
				lResult.setText("Student not found.");
			}
			num++;
		}
	}

	public void calculator() {
		// New frame
		fCalculator = new JFrame("Calculator");
		fCalculator.setLayout(new GridLayout(2, 1));
		fCalculator.setSize(500, 500);
```

```
fCalculator.setVisible(true);

// New panels
text = new JPanel();
text.setLayout(new GridLayout(2, 1));

numberButtons = new JPanel();
numberButtons.setLayout(new GridLayout(4, 3));

operations = new JPanel();
operations.setLayout(new GridLayout(4, 2));

buttons = new JPanel();
buttons.setLayout(new GridLayout(1, 2));

// New labels
displayArea.setBackground(Color.gray);
text.add(displayArea);

result.setBackground(Color.white);
text.add(result);

// New buttons
```

```
// Numbers
one = new JButton("1");
one.setEnabled(true);
one.addActionListener(this);
numberButtons.add(one);

two = new JButton("2");
two.setEnabled(true);
two.addActionListener(this);
numberButtons.add(two);

three = new JButton("3");
three.setEnabled(true);
three.addActionListener(this);
numberButtons.add(three);

four = new JButton("4");
four.setEnabled(true);
four.addActionListener(this);
numberButtons.add(four);

five = new JButton("5");
five.setEnabled(true);
```

```
five.addActionListener(this);
numberButtons.add(five);

six = new JButton("6");
six.setEnabled(true);
six.addActionListener(this);
numberButtons.add(six);

seven = new JButton("7");
seven.setEnabled(true);
seven.addActionListener(this);
numberButtons.add(seven);

eight = new JButton("8");
eight.setEnabled(true);
eight.addActionListener(this);
numberButtons.add(eight);

nine = new JButton("9");
nine.setEnabled(true);
nine.addActionListener(this);
numberButtons.add(nine);
```

```
zero = new JButton("0");
zero.setEnabled(true);
zero.addActionListener(this);
numberButtons.add(zero);

decimal = new JButton(".");
decimal.setEnabled(true);
decimal.addActionListener(this);
numberButtons.add(decimal);

clear = new JButton("AC");
clear.setEnabled(true);
clear.setBackground(Color.red);
clear.setForeground(Color.white);
clear.addActionListener(this);
numberButtons.add(clear);

// Operations
leftP = new JButton("(");
leftP.setEnabled(true);
leftP.addActionListener(this);
operations.add(leftP);
```

```
rightP = new JButton(")");
rightP.setEnabled(true);
rightP.addActionListener(this);
operations.add(rightP);

multiply = new JButton("×");
multiply.setEnabled(true);
multiply.addActionListener(this);
operations.add(multiply);

divide = new JButton("÷");
divide.setEnabled(true);
divide.addActionListener(this);
operations.add(divide);

add = new JButton("+");
add.setEnabled(true);
add.addActionListener(this);
operations.add(add);

subtract = new JButton("-");
subtract.setEnabled(true);
subtract.addActionListener(this);
```

```
        operations.add(subtract);

        ans = new JButton("Ans");
        ans.setEnabled(true);
        ans.addActionListener(this);
        operations.add(ans);

        equal = new JButton("=");
        equal.setEnabled(true);
        equal.addActionListener(this);
        operations.add(equal);

        // Adding panels to panel
        buttons.add(numberButtons);
        buttons.add(operations);

        // Adding panels to frame
        fCalculator.add(text);
        fCalculator.add(buttons);
}

public void updateEquation(String num) {
        displayArea.setText(displayArea.getText() + num);
```

```
}

public String convert(String equation) {
        equation = displayArea.getText().replaceAll("Ans", "" + ansDouble);
        return equation;
}

public double calculate(final String str) {
        return new Object() {
                // Variable declaration
        int index = -1, ch;

        // Saving next char in equation
        void nextChar() {
                if (++index < str.length()) {
                ch = str.charAt(index);
                }
        else {
                ch = -1;
        }
      }

      // Determining saved char
```

```
boolean eat(int charToEat) {
    while (ch == ' ') {
            // Continue reading
            nextChar();
    }
    if (ch == charToEat) {
            // Read the next char
        nextChar();
        return true;
    }
    else {
            return false;
    }
}

// Addition & subtraction
double parseExpression() {
    // Obtaining result from multiplication & division first
    double x = parseTerm();

    // While loop repeats process until equation is finished
    while (true) {
        if (eat('+') == true) { //plus
```

```
            x += parseTerm();
        }
        else if (eat('-') == true) { //minus
            x -= parseTerm();
        }
        else {
            return x;
        }
    }
}

// Multiplication & division
double parseTerm() {
    // Obtaining result of addition/subtraction for factors first
    double x = parseFactor();

    // While loop repeats process until equation is finished
    while (true) {
        if (eat('×') == true) {
            x *= parseFactor(); //times
        }
        else if (eat('÷') == true) {
            x /= parseFactor(); //divide
```

```
        }
        else {
                return x;
        }
    }
}

// Addition/subtraction of factors of multiplication/division
double parseFactor() {
    if (eat('+') == true) {
                return parseFactor(); // unary plus
    }
    if (eat('-') == true) {
                return -parseFactor(); // unary minus
    }

    // Variable declaration
    double x;
    int startPos = index;

    // Parentheses
    if (eat('(') == true) {
                // Evaluate inside of parentheses
```

```
            x = parseExpression();

            // Keep reading until closing parenthesis reached
            eat(')');
        }
        // Numbers & decimal
        else if ((ch >= '0' && ch <= '9') || ch == '.') {
            while ((ch >= '0' && ch <= '9') || ch == '.') {
                    // Keep reading until an operand character is reached
                    nextChar();
            }
            x = Double.parseDouble(str.substring(startPos, index));
        }
        else {
                    result.setText("SYNTAX ERROR");
            throw new RuntimeException("Unexpected: " + (char)ch);
        }

        return x;
    }

    // Evaluation
    double parse() {
```

```
            // Read next char
            nextChar();

            // Evaluate current char
            double x = parseExpression();

            // Exception catching
            if (index < str.length()) {
                    result.setText("SYNTAX ERROR");
                    throw new RuntimeException("Unexpected: " + (char)ch);
            }
            else {
                    return x;
            }
        }
    }.parse();
}

public void updateResult(double ans) {
            if (ans == Double.POSITIVE_INFINITY) {
                    result.setText("MATH ERROR");
            }
            else if (ans == Double.NEGATIVE_INFINITY) {
```

```
                result.setText("MATH ERROR");
        }
        else {
                // Formatting result
                DecimalFormat df = new DecimalFormat("#.#########");

                result.setText("" + df.format(ans));
                ansDouble = ans;
        }
}

public void actionPerformed(ActionEvent e) {
        // ALL MAIN PROGRAM EVENTS
        if (e.getSource() == buttonCal) {
                try {
                        save();
                        new Calendar().calendar();
                        fMain.dispose();
                }
                catch (IOException ioe) {
                        ioe.printStackTrace();
                        new Calendar().calendar();
                        fMain.dispose();
```

```
            }
            catch (NullPointerException npe) {
                  npe.fillInStackTrace();
                  new Calendar().calendar();
                  fMain.dispose();
            }
}
else if (e.getSource() == buttonLists) {
            try {
                  save();

                  try {
                        new StudentLists().studentLists();
                        fMain.dispose();
                  }
                  catch (IOException ioe) {
                        ioe.printStackTrace();
                  }
            }
            catch (IOException ioe) {
                  ioe.printStackTrace();

                  try {
```

```
                    new StudentLists().studentLists();
                    fMain.dispose();
                }
                catch (IOException ioe2) {
                    ioe2.printStackTrace();
                }
            }
            catch (NullPointerException npe) {
                npe.fillInStackTrace();

                try {
                    new StudentLists().studentLists();
                    fMain.dispose();
                }
                catch (IOException ioe) {
                    ioe.printStackTrace();
                }
            }
        }
        else if (e.getSource() == buttonFinance) {
            try {
                new FinancialRecords().finance();
                fMain.dispose();
```

```
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonRecords) {
            try {
                new StudentRecords().studentRecords();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonNotepad) {
            try {
                new Notepad();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
```

```
else if (e.getSource() == buttonHelp) {
        new HelpFinance();
}
else if (e.getSource() == signOut) {
        try {
                save();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
                fMain.dispose();
        }
        catch (NullPointerException npe) {
                npe.fillInStackTrace();
                fMain.dispose();
        }
}

// ALL FINANCIAL RECORDS EVENTS
if (e.getSource() == sortAZ_First) {
        try {
                sortFirstNames();
                fMain.dispose();
```

```
                updateDisplay();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == calculator) {
            calculator();
        }
        else if (e.getSource() == save) {
            try {
                save();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == search) {
            search();
        }

        // Search button
        if (e.getSource() == find) {
```

```
                find();
                fSearch.dispose();
                search();
            }

            // Calculator buttons
            if (e.getSource() == one || e.getSource() == two || e.getSource() == three
                    || e.getSource() == four || e.getSource() == five || e.getSource() ==
six
                    || e.getSource() == seven || e.getSource() == eight || e.getSource()
== nine
                    || e.getSource() == zero || e.getSource() == decimal || e.getSource()
== leftP
                    || e.getSource() == rightP || e.getSource() == ans) {
                updateEquation(((JButton) e.getSource()).getText());

                fCalculator.dispose();
                calculator();
            }
            else if (e.getSource() == multiply || e.getSource() == divide || e.getSource() ==
add
                    || e.getSource() == subtract) {
                updateEquation(" " + ((JButton) e.getSource()).getText() + " ");
```

```
                fCalculator.dispose();
                calculator();
        }
        else if (e.getSource() == equal) {
                updateResult(calculate(convert(displayArea.getText())));
                updateEquation(" " + ((JButton) e.getSource()).getText() + " ");

                fCalculator.dispose();
                calculator();
        }
        else if (e.getSource() == clear) {
                displayArea.setText("");
                result.setText("");
                fCalculator.dispose();
                calculator();
        }
}

public void windowClosing(WindowEvent e) {
        try {
                save();
                fMain.dispose();
```

```
                }
                catch (IOException ioe) {
                        ioe.printStackTrace();
                        fMain.dispose();
                }
                catch (NullPointerException npe) {
                        npe.fillInStackTrace();
                        fMain.dispose();
                }
        }
}
```

Notepad Code:

```
/* Notes:
 * Status: FINISHED
 * Currently saves notes
 * Missing text formatting */

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;

public class Notepad extends MainProgram {
```

```
private JTextArea note = new JTextArea();
private JPanel mainButtons, pad, panel;
private File directory = new File(username + "/Notepad");

public Notepad() throws IOException {
        // Creating new directory
        directory.mkdir();

        // New panel
        mainButtons = new JPanel();
        mainButtons.setLayout(new GridLayout(1, 7));

        // Adding components to main buttons panel
        mainButtons.add(buttonCal);
        mainButtons.add(buttonLists);
        mainButtons.add(buttonRecords);
        mainButtons.add(buttonFinance);
        mainButtons.add(buttonNotepad);
        mainButtons.add(buttonHelp);
        mainButtons.add(signOut);

        buttonNotepad.setEnabled(false);
        buttonNotepad.setBackground(Color.pink);
```

```
// New colour
Color c = new Color(255, 255, 200);

// New panel
pad = new JPanel();
pad.setBackground(c);

// New text area
note.setBackground(c);

try {
        FileReader fr = new FileReader(directory + "/notepad.txt");
        int ch;
        String input = "";

        while ((ch = fr.read()) != -1) {
                input += (char) ch;
        }

        fr.close();

        if (!(input.equals(""))) {
```

```
                note.setText(input);
            }
            else {
                note.setText("Type your notes here.");
            }
        }
        catch (IOException ioe) {
            ioe.printStackTrace();
            note.setText("Type your notes here.");
        }

        note.setEditable(true);

        // Adding text area to panel
        pad.add(note);

        // New panel
        panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

        panel.add(mainButtons);
        panel.add(pad);
```

```
        // Adding components frame
        fMain.add(panel);
        fMain.setExtendedState(JFrame.MAXIMIZED_BOTH);
        fMain.setVisible(true);
        fMain.remove(panelStart);
}

public void save() throws IOException {
        FileWriter fw = new FileWriter(directory + "/notepad.txt");

        fw.write(note.getText());

        fw.close();
}

public void actionPerformed(ActionEvent e) {
        // ALL MAIN PROGRAM EVENTS
        if (e.getSource() == buttonCal) {
                try {
                        save();
                        new Calendar().calendar();
                        fMain.dispose();
                }
```

```
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonLists) {
            try {
                save();
                new StudentLists().studentLists();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
        else if (e.getSource() == buttonFinance) {
            try {
                new FinancialRecords().finance();
                fMain.dispose();
            }
            catch (IOException ioe) {
                ioe.printStackTrace();
            }
        }
```

```
else if (e.getSource() == buttonRecords) {
        try {
                save();
                new StudentRecords().studentRecords();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == buttonNotepad) {
        try {
                new Notepad();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
}
else if (e.getSource() == buttonHelp) {
        new HelpNotes();
}
else if (e.getSource() == signOut) {
```

```
                try {
                        save();
                        fMain.dispose();
                }
                catch (IOException ioe) {
                        ioe.printStackTrace();
                }
        }
    }

    public void windowClosing(WindowEvent e) {
        try {
                save();
                fMain.dispose();
        }
        catch (IOException ioe) {
                ioe.printStackTrace();
        }
    }
}
```

Help Code:

```
/* Notes:
 * Status: FINISHED
```

```
 * Currently displays instructions & FAQs */

import java.awt.*;
import javax.swing.*;

public class Help extends Frame {
	public JFrame fHelp;
	private JPanel panelHelp;
	private JTextArea iText, fText;
	private JTextArea iHeader, fHeader;

	public Help() {
		// New frame
		fHelp = new JFrame("Help");

		// New panel
		panelHelp = new JPanel();
		panelHelp.setBackground(Color.pink);
		panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));

		// New text area
		iHeader = new JTextArea();
		iText = new JTextArea();
```

```
fHeader = new JTextArea();

fText = new JTextArea();


// New font

Font headers = new Font("times new roman", Font.BOLD, 18);

Font text = new Font("times new roman", Font.PLAIN, 16);


// Setting the text

iHeader.setText("Instructions");

iHeader.setFont(headers);


iText.setText("Welcome to your personal Organizer for Music Teachers!\n\n"

                + "For more tab-specific help, click the help button after\n"

                + "opening a specific tab!\n\n"

                + "Get started by creating your personal scheduler\n"

                + "in the Calender tab, or keep track of your students\n"

                + "with the Student Lists.\n\n"

                + "You can organize your finances with the\n"

                + "Financial Records tab, or document your students'\n"

                + "progress in the Student Records.\n\n"

                + "Lastly, the Notepad function allows you a\n"

                + "convenient storage of your notes.\n");
```

```
iText.setFont(text);

fHeader.setText("Frequently Asked Questions (FAQs)");
fHeader.setFont(headers);

fText.setText("How do I add multiple events to a day?\n\n"
                + "Unfortunately, this app only allows you to\n"
                + "add one event per day.\n\n\n"
                + "If I forget to press the save button,\n"
                + "will all my records be deleted?\n\n"
                + "No, this app tries its best to autosave whatever content\n"
                + "you have entered upon the switching of tabs or closure\n"
                + "of the app. However, please remember that clicking\n"
                + "the save button is the best way to prevent data loss.");
fText.setFont(text);

iHeader.setOpaque(false);
fHeader.setOpaque(false);
iHeader.setEditable(false);
iText.setEditable(false);

// Adding text area to panel
panelHelp.add(iHeader);
```

```
                panelHelp.add(iText);
                panelHelp.add(fHeader);
                panelHelp.add(fText);
                panelHelp.setAlignmentX(LEFT_ALIGNMENT);

                // New scrollable content pane
                JScrollPane scrPane = new JScrollPane(panelHelp);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 500);
                fHelp.setVisible(true);
        }
}
/* Notes:
 * Status: FINISHED
 * Currently displays instructions */
```

```
import java.awt.*;
import javax.swing.*;

public class HelpCal extends Frame {
	public JFrame fHelp;
	private JPanel panelHelp;
	private JTextArea iText;
	private JTextArea iHeader;

	public HelpCal() {
		// New frame
		fHelp = new JFrame("Calendar Help");

		// New panel
		panelHelp = new JPanel();
		panelHelp.setBackground(Color.pink);
		panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));

		// New text area
		iHeader = new JTextArea();
		iText = new JTextArea();

		// New font
```

```
Font headers = new Font("times new roman", Font.BOLD, 18);
Font text = new Font("times new roman", Font.PLAIN, 16);

// Setting the text
iHeader.setText("Instructions");
iHeader.setFont(headers);

iText.setText("This is your Calendar.\n"
                + "You can:\n\n"
                + "- Scroll through dates using the arrow keys\n"
                + "- Add events by clicking on dates and pressing add event\n"
                + "- Display events by clicking on a specific day\n"
                + "- Delete events by displaying them, then clicking delete");
iText.setFont(text);

iHeader.setOpaque(false);
iHeader.setEditable(false);
iText.setEditable(false);

// Adding text area to panel
panelHelp.add(iHeader);
panelHelp.add(iText);
panelHelp.setAlignmentX(LEFT_ALIGNMENT);
```

```
                // New scrollable content pane
                JScrollPane scrPane = new JScrollPane(panelHelp);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 230);
                fHelp.setVisible(true);
        }
}
/* Notes:
 * Status: FINISHED
 * Currently displays instructions */

import java.awt.*;
import javax.swing.*;

public class HelpFinance extends Frame {
```

```
public JFrame fHelp;
private JPanel panelHelp;
private JTextArea iText;
private JTextArea iHeader;

public HelpFinance() {
        // New frame
        fHelp = new JFrame("Financial Records Help");

        // New panel
        panelHelp = new JPanel();
        panelHelp.setBackground(Color.pink);
        panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));

        // New text area
        iHeader = new JTextArea();
        iText = new JTextArea();

        // New font
        Font headers = new Font("times new roman", Font.BOLD, 18);
        Font text = new Font("times new roman", Font.PLAIN, 16);

        // Setting the text
```

```
iHeader.setText("Instructions");
iHeader.setFont(headers);

iText.setText("These are your Financial Records.\n"
                + "You can:\n\n"
                + "- Enter and save financial information about each student\n"
                + "in the text-fields next to their names\n"
                + "- Sort students by their first names\n"
                + "- Use the calculator to perform basic operations\n"
                + "- Search for students' row position\n\n"
                + "IMPORTANT REMINDER: Please remember to always\n"
                + "save your information before switching to a new tab or\n"
                + "closing the application.");
iText.setFont(text);

iHeader.setOpaque(false);
iHeader.setEditable(false);
iText.setEditable(false);

// Adding text area to panel
panelHelp.add(iHeader);
panelHelp.add(iText);
panelHelp.setAlignmentX(LEFT_ALIGNMENT);
```

```
                // New scrollable content pane
                JScrollPane scrPane = new JScrollPane(panelHelp);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 320);
                fHelp.setVisible(true);
        }
}
/* Notes:
 * Status: FINISHED
 * Currently displays instructions */

import java.awt.*;
import javax.swing.*;

public class HelpLists extends Frame {
```

```
public JFrame fHelp;
private JPanel panelHelp;
private JTextArea iText;
private JTextArea iHeader;

public HelpLists() {
        // New frame
        fHelp = new JFrame("Student Lists Help");

        // New panel
        panelHelp = new JPanel();
        panelHelp.setBackground(Color.pink);
        panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));

        // New text area
        iHeader = new JTextArea();
        iText = new JTextArea();

        // New font
        Font headers = new Font("times new roman", Font.BOLD, 18);
        Font text = new Font("times new roman", Font.PLAIN, 16);

        // Setting the text
```

```
iHeader.setText("Instructions");
iHeader.setFont(headers);

iText.setText("This is your Student Lists.\n"
                + "You can:\n\n"
                + "- Add new students and info by clicking \"Add a Student\"\n"
                + "- Delete students by clicking \"Delete a Student\"\n"
                + "(select a name, then press delete)\n"
                + "- Sort students by their first names\n"
                + "- Sort students by their last names");
iText.setFont(text);

iHeader.setOpaque(false);
iHeader.setEditable(false);
iText.setEditable(false);

// Adding text area to panel
panelHelp.add(iHeader);
panelHelp.add(iText);
panelHelp.setAlignmentX(LEFT_ALIGNMENT);

// New scrollable content pane
JScrollPane scrPane = new JScrollPane(panelHelp);
```

```
scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 230);
                fHelp.setVisible(true);
        }
}
/* Notes:
 * Status: FINISHED
 * Currently displays instructions */

import java.awt.*;
import javax.swing.*;

public class HelpNotes extends Frame {
        public JFrame fHelp;
        private JPanel panelHelp;
        private JTextArea iText;
```

```
private JTextArea iHeader;


public HelpNotes() {
        // New frame
        fHelp = new JFrame("Calendar Help");


        // New panel
        panelHelp = new JPanel();
        panelHelp.setBackground(Color.pink);
        panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));


        // New text area
        iHeader = new JTextArea();
        iText = new JTextArea();


        // New font
        Font headers = new Font("times new roman", Font.BOLD, 18);
        Font text = new Font("times new roman", Font.PLAIN, 16);


        // Setting the text
        iHeader.setText("Instructions");
        iHeader.setFont(headers);
```

```
                iText.setText("This is your Notepad.\n\n"
                                + "You can store quick notes here, and everything will be\n"
                                + "saved upon switching to another tab or closure of\n"
                                + "the application.");
                iText.setFont(text);

                iHeader.setOpaque(false);
                iHeader.setEditable(false);
                iText.setEditable(false);

                // Adding text area to panel
                panelHelp.add(iHeader);
                panelHelp.add(iText);
                panelHelp.setAlignmentX(LEFT_ALIGNMENT);

                // New scrollable content pane
                JScrollPane scrPane = new JScrollPane(panelHelp);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDE
D);
```

```
                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 170);
                fHelp.setVisible(true);
        }
}
/* Notes:
 * Status: FINISHED
 * Currently displays instructions */

import java.awt.*;
import javax.swing.*;

public class HelpRecords extends Frame {
        public JFrame fHelp;
        private JPanel panelHelp;
        private JTextArea iText;
        private JTextArea iHeader;

        public HelpRecords() {
                // New frame
                fHelp = new JFrame("Student Records Help");
```

```
// New panel
panelHelp = new JPanel();
panelHelp.setBackground(Color.pink);
panelHelp.setLayout(new BoxLayout(panelHelp, BoxLayout.Y_AXIS));

// New text area
iHeader = new JTextArea();
iText = new JTextArea();

// New font
Font headers = new Font("times new roman", Font.BOLD, 18);
Font text = new Font("times new roman", Font.PLAIN, 16);

// Setting the text
iHeader.setText("Instructions");
iHeader.setFont(headers);

iText.setText("These are your Student Records.\n"
                + "You can:\n\n"
                + "- Enter and save information about each student\n"
                + "in the text-fields next to their names\n"
                + "- Sort students by their first names\n"
                + "- Sort students by their last names\n"
```

```
                + "- Search for students' row position\n\n"
                + "IMPORTANT REMINDER: Please remember to always\n"
                + "save your information before switching to a new tab or\n"
                + "closing the application.");
        iText.setFont(text);

        iHeader.setOpaque(false);
        iHeader.setEditable(false);
        iText.setEditable(false);

        // Adding text area to panel
        panelHelp.add(iHeader);
        panelHelp.add(iText);
        panelHelp.setAlignmentX(LEFT_ALIGNMENT);

        // New scrollable content pane
        JScrollPane scrPane = new JScrollPane(panelHelp);

scrPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scrPane.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
```

```
                // Adding panel to frame
                fHelp.add(scrPane);
                fHelp.setSize(400, 320);
                fHelp.setVisible(true);
        }
}
```

Run Code:

```
public class Run {

        public static void main(String[] args) {
                new LoginValidation().loginValidation();
        }

}
```