**Appendix: Java Code**

Appointment.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.\*;

import java.util.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import javax.sound.sampled.LineUnavailableException;

import javax.swing.\*;

import sun.audio.\*;

/\*\*

\*

\* @author 061264

\*/

public class Appointment extends TimerTask {

String myName;

private int numApptParam = 5; //number of parameters for the date in an appointmetn

private int[] myApptInfo = new int[5]; //holds the date information of an appointment

private Date myStartTimeDate; //represents the the start time and the date of an appointment

public Appointment(String name, int year, int month, int date, int startTime, int endTime) {

myName = name;

myApptInfo[0] = year;

myApptInfo[1] = month;

myApptInfo[2] = date;

myApptInfo[3] = startTime;

myApptInfo[4] = endTime;

myStartTimeDate = new Date(myApptInfo[0] - 1900, myApptInfo[1] - 1, myApptInfo[2], myApptInfo[3] / 100, myApptInfo[3] % 100);

}

public Appointment() {

} // empty Appointment object

//task that runs when a reminder is sent for an appointment

@SuppressWarnings("empty-statement")

public void run() {

Reminder remind = new Reminder();

remind.setVisible(true);

}

public String getName() {

return myName;

}

public int[] getApptInfo() {

return myApptInfo;

} //returns the date appt information in an array

public int getYear() {

return myApptInfo[0];

}

public int getMonth() {

return myApptInfo[1];

}

public int getDayOfMonth() {

return myApptInfo[2];

}

public static int convertToMilit(int hrs, int min, String timeOfDay) { //converts a time to 0-2300 hr format

if (timeOfDay.equalsIgnoreCase("AM") && hrs == 12) //if its 12:00 AM hrs is zero. returns the minutes only

{

return min;

} else if (timeOfDay.equalsIgnoreCase("Pm") && hrs != 12) { //converts afternoon times

return (hrs + 12) \* 100 + min;

} else {

return (hrs \* 100) + min; //returns

}

}

public String getStartTime() {

int hrs = myApptInfo[3] / 100;

String timeOfDay; //represents the time of day of start time: "AM" or "PM"

if (hrs > 12) { // is the start time past noon

hrs = hrs - 12;

timeOfDay = "pm";

} else if (hrs == 12) { //is the start time noon

timeOfDay = "pm";

} else { // appointment is in the morning

timeOfDay = "am";

}

if (hrs == 0) {

hrs = 12;

}

if (myApptInfo[3] % 100 < 10) // are the minutes less than ten

{

return hrs + ":" + "0" + myApptInfo[3] % 100 + timeOfDay; //adds a zero to the minutes if it is. Ex if minutes = 3 adds a 0 so 12:03 is displayed instead of 12:3

} else {

return hrs + ":" + myApptInfo[3] % 100 + " " + timeOfDay;

}

}

public String getStartHours() {

int index = getStartTime().indexOf(":"); //gets hours by getting the index of the ":" from the start time string

return getStartTime().substring(0, index); // returns the hours by getting the substring of the start time up to the ":"

}

public String getStartMinutes() {

int index = getStartTime().indexOf(":"); // gets the index of the ":" in the start tiem

return getStartTime().substring(index + 1, index + 3); // returns the minutes by getting the substring after the colon and before the time of day

}

public String getEndTime() {

int hrs = myApptInfo[4] / 100;

String timeOfDay;

if (hrs > 12) {

hrs = hrs - 12;

timeOfDay = "pm";

} else if (hrs == 12) {

timeOfDay = "pm";

} else {

timeOfDay = "am";

}

if (hrs == 0) {

hrs = 12;

}

if (myApptInfo[4] % 100 < 10) {

return hrs + ":" + "0" + myApptInfo[4] % 100 + timeOfDay;

} else {

return hrs + ":" + myApptInfo[4] % 100 + " " + timeOfDay;

}

}

public String getEndHours() {

int index = getEndTime().indexOf(":");

return getEndTime().substring(0, index);

}

public String getEndMinutes() {

int index = getEndTime().indexOf(":");

return getEndTime().substring(index, index + 2);

}

//returns a date object that represents start time of appt.

public Date getDate() {

return myStartTimeDate;

}

//returns the date of the appointment information in an []

public int[] allInfo() {

return myApptInfo;

}

public void setYear(int year) {

myApptInfo[0] = year;

myStartTimeDate.setYear(year);

}

public void setMonth(int month) {

myApptInfo[1] = month;

myStartTimeDate.setMonth(month - 1);

}

public void setDate(int date) {

myApptInfo[2] = date;

myStartTimeDate.setDate(date);

}

public void setStartTimeHrs(int startTimeHrs) {

myApptInfo[3] = startTimeHrs;

myStartTimeDate.setHours(startTimeHrs);

}

public void setStartTimeMin(int startTimeMin) {

myApptInfo[4] = startTimeMin;

myStartTimeDate.setMinutes(startTimeMin);

}

public void setEndTimeHrs(int endTimeHrs) {

myApptInfo[5] = endTimeHrs;

}

public void setEndTimeMin(int endTimeMin) {

myApptInfo[3] = endTimeMin;

}

public int compareTo(Appointment appt) {

return this.getDate().compareTo(appt.getDate());

} //compares two appointments based on their dates

public String appointMentInfo() {

String apptInfo = myName + " ";

apptInfo = apptInfo + getMonth() + "/" + getDayOfMonth() + "/" + getYear() + " ";

apptInfo = apptInfo + getStartTime() + " to " + getEndTime();

return apptInfo;

}

public static void quickSort(ArrayList<Appointment> info, int first, int last) {

if (first < 0 || last < 0) {

return;

}

int f = first;

int l = last;

int midIndex = (first + last) / 2;

Appointment obj = (Appointment) info.get(midIndex);

do {

while (((Appointment) info.get(f)).compareTo(obj) < 0) {

f++;

}

while (((Appointment) info.get(l)).compareTo(obj) > 0) {

l--;

}

if (f <= l) {

swap(info, f, l);

f++;

l--;

}

} while (f < l);

if (l > first) {

quickSort(info, first, l);

}

if (f < last) {

quickSort(info, f, last);

}

}

private static void swap(ArrayList<Appointment> info, int x, int y) {

Appointment ex = (Appointment) info.get(x);

info.set(x, info.get(y));

info.set(y, ex);

}

}

ControlSystem.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.io.\*;

import java.time.\*;

import java.util.\*;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author 061264

\*/

public class ControlSystem extends Timer{

//creates all privates

private Scanner in;

private static ArrayList<Appointment> myTimerTasks = new ArrayList<Appointment>(); //holds all the reminders for the appts that will be executed

private ArrayList<Student> myStudents; //holds all student information in alphabetical order

private static ArrayList<Appointment> myFutureAppointments; //holds all upcoming appointments

private ArrayList<Appointment> myPastAppointments; //holds all past appointments in order( closest appointments first)

private ArrayList<Appointment> myAllAppointments; //holds all appointments

private static Timer myTimer; // timer object to schedule reminders for the appointments

public ControlSystem() throws Exception {

//initializes all privates

myTimer = new Timer();

myStudents = new ArrayList<Student>();

myPastAppointments = new ArrayList<Appointment>();

myFutureAppointments = new ArrayList<Appointment>();

myAllAppointments = new ArrayList<Appointment>();

open(); //adds existing student and appt info to list

}

public void open() throws Exception {

readStudentInfo(); //adds student to arraylist

readAppointmentInfo(); //adds appts to arraylist

if(myFutureAppointments.size()>0) //schedules reminders for upcoming appts if there ar any

scheduleReminders(myFutureAppointments);

}

public static void scheduleReminders(ArrayList<Appointment> appts) throws Exception{

for (int x = 0; x < appts.size(); x++) { //traverses the arraylist to schedule reminders for each appt

scheduleReminders(appts.get(x));

}

}

public static void scheduleReminders(Appointment appt) throws Exception {

Date date = scheduleDateForReminder(appt); //schedules a reminder for the appt a day earlier

Appointment task = new Appointment();

myTimerTasks.add(task);

myTimer.schedule(task, date);

}

private static Date scheduleDateForReminder(Appointment appt) {

Date date1 = appt.getDate();

date1.setDate(date1.getDate()-1);

return date1;

}

//reads student info from txt file

private void readStudentInfo() throws Exception {

/\*attempts to find the file with name "studentInfo.txt" \*/

try {

in = new Scanner(new File("studentInfo.txt"));

} catch (Exception e) {

File f = new File("studentInfo.txt"); //creates new file if there is no existing "studentInfo.txt" file

f.createNewFile();

return; //ends the method because the new file will obviously be empty

}

/\*begins to read from file\*/

while (in.hasNext()) { //reads the parametes for students until txt hits blank line

String firstName = in.next();

String lastName = in.next();

String phoneNumber = in.next();

String emailAddress = in.next();

myStudents.add(new Student(firstName, lastName, phoneNumber, emailAddress)); //creates student obj and adds to arraylist

}

if (!myStudents.isEmpty()) {

Student.quickSort(myStudents, 0, myStudents.size() - 1); //sorts arraylist alphabetically

}

}

//reads appointment information from "appointmentInfo.txt" file

private void readAppointmentInfo() throws Exception {

try { //attempts to find the file with name "appointmentInfo.txt"

in = new Scanner(new File("appointmentInfo.txt"));

} catch (Exception e) { //creates new "appointment.txt" file

File f = new File("appointmentInfo.txt");

f.createNewFile();

return; //Ends method. No reason to read since file will be empty

}

while (in.hasNext()) { // Reads the paramters for appointments until scanner encounters blank line

String name = in.nextLine();

int year = Integer.parseInt(in.nextLine());

int month = Integer.parseInt(in.nextLine());

int dayOfMonth = Integer.parseInt(in.nextLine());

int startTime = Integer.parseInt(in.nextLine());

int endTime = Integer.parseInt(in.nextLine());

Date apptDate = new Date(year - 1900, month-1, dayOfMonth, startTime/100, startTime%100); //creates a date object using the information that was read

LocalDateTime date = LocalDateTime.now(); //represents the local time

Date currentDate = Date.from(date.atZone(ZoneId.systemDefault()).toInstant()); //date object to represent local time

Appointment appt = new Appointment(name, year, month, dayOfMonth, startTime, endTime); //creates appointment

if (currentDate.compareTo(apptDate)<0) //compares the local date and appt's date. If date is in future add to myFutureAppts

myFutureAppointments.add(appt);

else myPastAppointments.add(appt); //add to myPastAppts if in past

myAllAppointments.add(appt);

}

//sort each arraylist chronologically

Appointment.quickSort(myAllAppointments, 0, myAllAppointments.size() - 1);

Appointment.quickSort(myPastAppointments, 0, myPastAppointments.size() - 1);

Appointment.quickSort(myFutureAppointments, 0, myFutureAppointments.size() - 1);

}

//returns arraylist of students

public ArrayList<Student> getAllStudents() {

return myStudents;

}

//returns arraylist of all appts

public ArrayList<Appointment> getAllAppointments() {

return myAllAppointments;

}

public static void editFuturesList(Appointment appt){

int indexOf = myFutureAppointments.indexOf(appt);

if(indexOf !=-1){

myTimerTasks.remove(indexOf);

myFutureAppointments.remove(indexOf);

}

}

//returns arraylist of past appts

public ArrayList<Appointment> getPastAppointments() {

return myPastAppointments;

}

//returns arraylist of upcoming appts

public ArrayList<Appointment> getFutureAppointments() {

return myFutureAppointments;

}

//saves any changes made to arraylist to the text file

public void close() throws Exception {

FileWriter rt = new FileWriter("studentInfo.txt"); //creates filewriter and erases file

saveStudentData(rt);

rt = new FileWriter("appointmentInfo.txt");

saveAppointmentData(rt); //closes to save changes to arraylist

}

private void saveStudentData(FileWriter rt) throws Exception {

for (int x = 0; x < myStudents.size(); x++) { //starts traversing arraylist of students

Student student = myStudents.get(x); //Student object at index x

try {

for(int y = 0;y<student.getStudentInfo().length;y++) //traverses through the array holding student's information

rt.write(student.getStudentInfo()[y] + "\n"); //writes the information of Student at x and the information at y

} catch (IOException e) {

}

}

rt.close(); //closes to save changes to arraylist

}

private void saveAppointmentData(FileWriter rt) throws Exception {

for (int x = 0; x < myAllAppointments.size(); x++) { //starts traversing arraylist of appointments

rt.write(myAllAppointments.get(x).getName() + "\n"); //Appointment object at index x

try {

Appointment appt = myAllAppointments.get(x);

for(int y = 0;y<appt.getApptInfo().length;y++){ //traverses through the array holding appointment's information

rt.write(Integer.toString((appt.getApptInfo())[y]) + "\n"); //writes the information of Appointment at x and the information at y

}

} catch (IOException e) {

}

}

rt.close(); //closes to save changes to arraylist

}

}

List.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.util.\*;

import javax.swing.\*;

import javax.swing.table.\*;

/\*\*

\*

\* @author 061264

\*/

public class List extends javax.swing.JFrame {

/\*\*

\* Creates new form List

\*/

private ArrayList<Student> myStudents; //student list to be displayed

private ArrayList<Student> myAllStudents; //reference to all students. Needed in case a student is removed or edited

private ArrayList<Appointment> myAppointments; //Appointment list to be displayed

private ArrayList<Appointment> myAllAppointments; //reference to all Appointment. Needed in case an appointmnet is removed or edited

private JTable myTable; //Table that displays the list of objects

private String myListType;

//constructor used if displaying appts

public List(String listType, ArrayList<Student> students, ArrayList<Student> allStudents) {

initComponents();

myListType = listType;

ObjectType.setText(myListType); //changes Label at the top of the form to "Students

myStudents = students;

myAllStudents = allStudents;

initializeList();

Student.quickSort(myStudents, 0, students.size() - 1);

myAppointments = null;

myAllAppointments = null;

}

//constructor used if displaying students

public List(ArrayList<Appointment> appointments, ArrayList<Appointment> allAppointments) {

initComponents();

myAppointments = appointments;

myAllAppointments = allAppointments;

myListType = "Appointments";

ObjectType.setText("Appointments");

initializeList();

Appointment.quickSort(appointments, 0, appointments.size() - 1);

myAllStudents = null;

myStudents = null;

}

public void initializeList() {

if (myStudents != null) addStudentsToList();

else addApptsToList();

}

public void addApptsToList() {

Object columnLabels[] = {"Student Name", "Year", "Month", "Day", "Start time", "End Time"};

Object rowData[][] = new Object[myAppointments.size()][columnLabels.length];

for (int apptNumber = myAppointments.size()-1, row = 0; row <myAppointments.size();apptNumber-- ,row++) { //begins traversing through the appointment list

rowData[row][0] = myAppointments.get(apptNumber).getName();

rowData[row][1] = (myAppointments.get(apptNumber).getYear());

rowData[row][2] = (myAppointments.get(apptNumber).getMonth());

rowData[row][3] = (myAppointments.get(apptNumber).getDayOfMonth());

rowData[row][4] = (myAppointments.get(apptNumber).getStartTime());

rowData[row][5] = (myAppointments.get(apptNumber).getEndTime());

}

createTable(rowData, columnLabels);

}

public void addStudentsToList() {

String rowData[][] = new String[myStudents.size()][4];

String columnLabels[] = {"First Name", "Last Name", "Phone number", "Email address"};

for (int row = 0; row < myStudents.size(); row++) {

for (int col = 0; col < 4; col++) {

rowData[row][col] = (myStudents.get(row).getStudentInfo())[col];

}

}

createTable(rowData, columnLabels);

}

public void createTable(Object[][] rowData, Object[] columnLabels) {

DefaultTableModel tableModel = new DefaultTableModel(rowData, columnLabels) {

@Override

public boolean isCellEditable(int row, int column) {

//all cells false

return false;

}

}; //creates a table model to hold JTable

myTable = new JTable(tableModel);

jScrollPane1.getViewport().add(myTable); //places table in a scroll pane and makes it visible

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

ObjectType = new javax.swing.JLabel();

Remove = new javax.swing.JButton();

jScrollPane1 = new javax.swing.JScrollPane();

edit = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

ObjectType.setText("jLabel1");

Remove.setText("Remove");

Remove.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

remove(evt);

}

});

edit.setText("Edit");

edit.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

editData(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(21, 21, 21)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 813, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addComponent(Remove)

.addGap(30, 30, 30)

.addComponent(edit))

.addGroup(layout.createSequentialGroup()

.addComponent(ObjectType)

.addGap(366, 366, 366)

.addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel1)))

.addContainerGap(16, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(ObjectType)

.addComponent(jLabel1)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 39, Short.MAX\_VALUE)

.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED\_SIZE, 509, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(Remove)

.addComponent(edit))

.addGap(35, 35, 35))

);

pack();

}// </editor-fold>

private void remove(java.awt.event.ActionEvent evt) {

int firstRowSelected = myTable.getSelectedRow();

int lastRowSelected = myTable.getSelectedRowCount() + firstRowSelected; //gets last row selected by gettin row count and adding to index of first row

if (myListType.equalsIgnoreCase("Appointments")) { //determines if this is a list of appts or students

removeAppts(firstRowSelected, lastRowSelected);

} else {

removeStudents(firstRowSelected, lastRowSelected);

}

initializeList(); //recreates list so removed objects are not visible

}

private void editData(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

int rowSelected = myTable.getSelectedRow();

if(rowSelected>0){

if (myStudents != null) {

ModifyStudent student = new ModifyStudent(myAllStudents,myStudents.get(rowSelected), "edit");

student.setVisible(true);

student.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

}

else {

ModifyAppointment appt = new ModifyAppointment(myAllAppointments, myAppointments.get(rowSelected), "Edit");

appt.setVisible(true);

appt.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

}

jLabel2.setText("Please close and reopen this list to show updated changes");

initializeList();

}

else jLabel2.setText("Please select a row to edit");

}

private void removeAppts(int firstRowSelected, int lastRowSelected) {

for (int x = lastRowSelected - 1; x >= firstRowSelected; x--) { //traverses through the appointments between the first and last row selected inclusive

Appointment appt = myAppointments.get(x);

myAppointments.remove(x); //removes from display

myAllAppointments.remove(appt); //removes from whole program

ControlSystem.editFuturesList(appt);

}

}

private void removeStudents(int firstRowSelected, int lastRowSelected) {

for (int x = lastRowSelected - 1; x >= firstRowSelected; x--) { //traverses through the students between the first and last row selected inclusive

Student stud = myStudents.get(x);

myStudents.remove(x); //removes student from list that is displayed

myAllStudents.remove(stud); //removes any reference to this student from the whole program

}

}

/\*\*

\* @param args the command line arguments

\*/

public void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(List.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

if (myStudents == null) {

new List(myAppointments, myAppointments).setVisible(true);

} else {

new List(myListType, myStudents, myAllStudents).setVisible(true);

}

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel ObjectType;

private javax.swing.JButton Remove;

private javax.swing.JButton edit;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JScrollPane jScrollPane1;

// End of variables declaration

}

MainActivityClass.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.awt.event.\*;

import java.io.\*;

import java.util.\*;

import java.util.logging.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author 061264

\*/

public class MainActivityClass extends javax.swing.JFrame {

/\*\*

\* Creates new form MainActivityClass

\*/

private ControlSystem myControlSystem;

private static final String myStudentLabel = "Student";

private static final String myAppointmentLabel = "Appointment";

public MainActivityClass() throws Exception {

initComponents();

myControlSystem = new ControlSystem();

addWindowListener(new WindowAdapter() {

@Override

public void windowClosing(WindowEvent e) {

try {

myControlSystem.close();

} catch (Exception ex) {

}

}

});

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

add = new javax.swing.JButton();

jLabel2 = new javax.swing.JLabel();

search = new javax.swing.JButton();

jComboBox1 = new javax.swing.JComboBox<String>();

displayAll = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

add.setText("Add");

add.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

add(evt);

}

});

jLabel2.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel2.setText("Tutoring System");

search.setText("Search");

search.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

search(evt);

}

});

jComboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--------------", "Student", "Appointment" }));

displayAll.setText("Display All");

displayAll.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

displayAll(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(35, 35, 35)

.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(search, javax.swing.GroupLayout.PREFERRED\_SIZE, 153, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(97, 97, 97)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 134, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(displayAll, javax.swing.GroupLayout.PREFERRED\_SIZE, 147, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addComponent(add, javax.swing.GroupLayout.PREFERRED\_SIZE, 153, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGap(0, 0, Short.MAX\_VALUE)))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGap(0, 0, Short.MAX\_VALUE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 19, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(search, javax.swing.GroupLayout.PREFERRED\_SIZE, 62, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(8, 8, 8)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(displayAll, javax.swing.GroupLayout.PREFERRED\_SIZE, 71, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(add, javax.swing.GroupLayout.PREFERRED\_SIZE, 70, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void add(java.awt.event.ActionEvent evt) {

String str = (String) jComboBox1.getSelectedItem();

if (str.equals(myStudentLabel)) {

ModifyStudent add = new ModifyStudent(myControlSystem.getAllStudents(), "Add");

add.setVisible(true);

add.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

} else if (str.equals(myAppointmentLabel)) {

ModifyAppointment add = new ModifyAppointment(myControlSystem.getAllAppointments(), "Add");

add.setVisible(true);

add.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

} else {

displayErrorDialogBox("Select search by student or appointment");

}

}

//opens search frame based on option

private void search(java.awt.event.ActionEvent evt) {

String str = (String) jComboBox1.getSelectedItem(); //reads selected item

if (str.equals(myStudentLabel)) {

ModifyStudent search = new ModifyStudent(myControlSystem.getAllStudents(), "Search");

search.setVisible(true);

search.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

} else if (str.equals(myAppointmentLabel)) {

SearchAppt search = new SearchAppt(myControlSystem.getAllAppointments());

search.setVisible(true);

search.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

} else {

displayErrorDialogBox("Select search by student or appointment");

}

}

private void displayAll(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

String str = (String) jComboBox1.getSelectedItem();

List list;

if (str.equals(myStudentLabel)) {

if (myControlSystem.getAllStudents().size() == 0) {

displayErrorDialogBox("You have no current students in your list.");

}

else{

list = new List("student", myControlSystem.getAllStudents(), myControlSystem.getAllStudents());

list.setVisible(true);

list.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

}

} else if (str.equals(myAppointmentLabel)) {

list = new List(myControlSystem.getAllAppointments(), myControlSystem.getAllAppointments());

list.setVisible(true);

list.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

} else {

displayErrorDialogBox("Select search by student or appointment");

}

}

public static void displayErrorDialogBox(String error) {

javax.swing.JDialog searchErrorDialog = new javax.swing.JDialog();

searchErrorDialog.add(new javax.swing.JLabel(error));

searchErrorDialog.setSize(450, 200);

searchErrorDialog.setVisible(true);

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(MainActivityClass.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(MainActivityClass.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(MainActivityClass.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(MainActivityClass.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

try {

new MainActivityClass().setVisible(true);

} catch (FileNotFoundException ex) {

Logger.getLogger(MainActivityClass.class.getName()).log(Level.SEVERE, null, ex);

} catch (IOException ex) {

Logger.getLogger(MainActivityClass.class.getName()).log(Level.SEVERE, null, ex);

} catch (Exception ex) {

Logger.getLogger(MainActivityClass.class.getName()).log(Level.SEVERE, null, ex);

}

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton add;

private javax.swing.JButton displayAll;

private javax.swing.JComboBox<String> jComboBox1;

private javax.swing.JLabel jLabel2;

private javax.swing.JButton search;

// End of variables declaration

}

ModifyAppointment.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.awt.event.\*;

import java.io.FileNotFoundException;

import java.time.\*;

import java.util.\*;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

public class ModifyAppointment extends javax.swing.JFrame {

private ArrayList<Appointment> myAppointments;

private ArrayList<String> myParameters;

private boolean isSearch;

private String myOption; //represent if this appt form is for adding or editing

private static final int myNumParams = 11;

private Appointment myApptToEdit;

//constructor that is used if adding an appt

public ModifyAppointment(ArrayList<Appointment> appointments, String option) {

initComponents();

myOption = option;

myAppointments = appointments;

myParameters = new ArrayList<String>(myNumParams);

myApptToEdit = null;

myApptToEdit = null;

jLabel4.setText("");

}

//constructor that is used if editing an appt

public ModifyAppointment(ArrayList<Appointment> appointments, Appointment apptToEdit, String option){

initComponents();

myOption = option;

myAppointments = appointments;

myParameters = new ArrayList<String>(myNumParams);

myApptToEdit = apptToEdit;

modifyAppt.setText("Save changes");

jLabel4.setText("Please fill out the appointment form to make any changes"); //requests user to reenter all the appt information with desired changes

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jComboBox2 = new javax.swing.JComboBox<String>();

jComboBox3 = new javax.swing.JComboBox<String>();

jLabel1 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

modifyAppt = new javax.swing.JButton();

jComboBox4 = new javax.swing.JComboBox<String>();

jComboBox5 = new javax.swing.JComboBox<String>();

jComboBox6 = new javax.swing.JComboBox<String>();

jComboBox7 = new javax.swing.JComboBox<String>();

jComboBox8 = new javax.swing.JComboBox<String>();

jComboBox9 = new javax.swing.JComboBox<String>();

firstName = new javax.swing.JTextField();

lastName = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jLabel8 = new javax.swing.JLabel();

jLabel11 = new javax.swing.JLabel();

jLabel12 = new javax.swing.JLabel();

jLabel13 = new javax.swing.JLabel();

jLabel14 = new javax.swing.JLabel();

jLabel15 = new javax.swing.JLabel();

jLabel9 = new javax.swing.JLabel();

jTextField1 = new javax.swing.JTextField();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jComboBox2.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12" }));

jComboBox3.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31" }));

jLabel1.setText("Start Time");

jLabel5.setText("End Time");

modifyAppt.setText("Add");

modifyAppt.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

modifyAppt(evt);

}

});

jComboBox4.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", " " }));

jComboBox5.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58", "59" }));

jComboBox6.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--", "AM", "PM" }));

jComboBox7.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", " " }));

jComboBox8.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "00", "01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31", "32", "33", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44", "45", "46", "47", "48", "49", "50", "51", "52", "53", "54", "55", "56", "57", "58", "59" }));

jComboBox9.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--", "AM", "PM" }));

jLabel2.setText("First Name");

jLabel3.setText("Last Name");

jLabel6.setText("Month");

jLabel7.setText("Date");

jLabel8.setText("Hour");

jLabel11.setText("Minutes");

jLabel12.setText("Time of Day");

jLabel13.setText("Hour");

jLabel14.setText("Minutes");

jLabel15.setText("Time of Day");

jLabel9.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel9.setText("Appointment Form");

jTextField1.setText("Year");

jTextField1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jTextField1ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1)

.addGap(0, 0, Short.MAX\_VALUE))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(0, 55, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel11)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jLabel12))

.addGroup(layout.createSequentialGroup()

.addGap(60, 60, 60)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jLabel4)

.addGap(51, 51, 51))

.addComponent(jLabel9, javax.swing.GroupLayout.Alignment.TRAILING))))

.addGap(15, 15, 15))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()

.addComponent(modifyAppt)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addGap(47, 47, 47)

.addComponent(jLabel14))

.addGroup(layout.createSequentialGroup()

.addComponent(jComboBox7, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jComboBox8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel15)

.addComponent(jComboBox9, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(50, 50, 50)))

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 51, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel6))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel7)

.addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addComponent(jLabel8)

.addGroup(layout.createSequentialGroup()

.addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jComboBox5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(jComboBox6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(jLabel13))

.addGap(44, 44, 44)))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel3)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 5, Short.MAX\_VALUE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 3, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(firstName, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 114, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(lastName, javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.PREFERRED\_SIZE, 114, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addContainerGap())))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel9)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel4)

.addGap(39, 39, 39)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel6)

.addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED\_SIZE, 14, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox2, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox3, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(7, 7, 7)

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel8)

.addComponent(jLabel11)

.addComponent(jLabel12))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jComboBox4, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox5, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox6, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel13)

.addComponent(jLabel14)

.addComponent(jLabel15))

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 16, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addComponent(firstName, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(9, 9, 9)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jComboBox8, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox7, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jComboBox9, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addGroup(layout.createSequentialGroup()

.addGap(15, 15, 15)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3)

.addComponent(lastName, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(modifyAppt)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

//responds when button is pressed

private void modifyAppt(java.awt.event.ActionEvent evt) {

if (myOption.equals("Add")) { //if the appt form is for adding an appt

try {

add(); //add appt

} catch (Exception e) {

}

} else {

try {

edit(); //edit appt

} catch (Exception ex) {

}

}

}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

//adds all the params by storing a String type of the input entered in the drop down

private void addParams() {

myParameters.clear(); //clears list in case there was a previous entry

myParameters.add((String) jTextField1.getText());

myParameters.add((String) jComboBox2.getSelectedItem());

myParameters.add((String) jComboBox3.getSelectedItem());

myParameters.add((String) jComboBox4.getSelectedItem());

myParameters.add((String) jComboBox5.getSelectedItem());

myParameters.add((String) jComboBox6.getSelectedItem());

myParameters.add((String) jComboBox7.getSelectedItem());

myParameters.add((String) jComboBox8.getSelectedItem());

myParameters.add((String) jComboBox9.getSelectedItem());

}

private void add() throws Exception {

addParams();

if (!isValidAppt()) //ends the method if the appt is not valid

{

return;

}

String name = firstName.getText() + " " + lastName.getText(); //reads all parameters for the date

int years = Integer.parseInt(myParameters.get(0));

int month = Integer.parseInt(myParameters.get(1));

int dayOfMonth = Integer.parseInt(myParameters.get(2));

int startTime = Appointment.convertToMilit(Integer.parseInt(myParameters.get(3)), Integer.parseInt(myParameters.get(4)), myParameters.get(5)); //converst start time to military time because that is how the constructor requires it

int endTime = Appointment.convertToMilit(Integer.parseInt(myParameters.get(6)), Integer.parseInt(myParameters.get(7)), myParameters.get(8)); //converst end time to military time because that is how the constructor requires it

Appointment appt = new Appointment(name, years, month, dayOfMonth, startTime, endTime);

if (checkConflict(appt)) { //if there is no overlap between any appts

myAppointments.add(appt);

Appointment.quickSort(myAppointments, 0, myAppointments.size() - 1);

Date date = appt.getDate();

ControlSystem.scheduleReminders(appt); //schedules reminder for the appt

jLabel4.setText(name +"'s Appointment has been added");

}

else

jLabel4.setText("");

}

private void edit() throws Exception {

myAppointments.remove(myApptToEdit); //editing an appointment is essentially creating a new one. current one is therefore removed

add();

}

//returns true if the appt is a valid appt

private boolean isValidAppt() {

return (isValidParam() && isValidDate());

}

//returns true if all params are entered

private boolean isValidParam() {

for (int x = 0; x < myParameters.size(); x++) { //traverses through the user input which is stored in myParameters. \*myParameters only includes the date and time info of appt

if (myParameters.get(x).equals("---")) { //did the user not enter a parameter

MainActivityClass.displayErrorDialogBox("Please enter all parameters"); //error displays to warn user that a parameter is missing

return false;

}

}

if (firstName.getText().isEmpty() && lastName.getText().isEmpty()) { //did the user enter at least a first name or a last name

MainActivityClass.displayErrorDialogBox("Please enter all parameters"); //error if both are empty

return false;

}

return true; //returns true if all required parameters are entere

}

//returns true if date is in the future

private boolean isValidDate() {

LocalDateTime localDate = LocalDateTime.now(); //local time to represent current time and date

Instant instant = localDate.atZone(ZoneId.systemDefault()).toInstant(); //converts the local time to an instant. Instant represents an instant in time

Date local = Date.from(instant); //creates a date object using the instant

Date apptDate = createDate(); //creates a date object based on the user input stored in myParameters

if (local.compareTo(createDate()) < 0) //is the appt date in the future

{

return true;

} else {

MainActivityClass.displayErrorDialogBox("This date has alread passed."); //displays error if the appt date is in past

return false;

}

}

//returns true if there is no overlap with future appts

private boolean checkConflict(Appointment appt) {

for (int x = 0; x < myAppointments.size(); x++) { //traverses the list of appts

Appointment thisAppt = myAppointments.get(x);

boolean isSameDate = true;

for (int y = 0; y < 3; y++) { //checks if the appt at this index is in same year, month and day

if (appt.getApptInfo()[y] != thisAppt.getApptInfo()[y]) {

isSameDate = false;

y=3;

}

}

if(isSameDate){

if(!checkTimeOverLap(appt,thisAppt))

return false;

}

}

return true;

}

private boolean checkTimeOverLap(Appointment appt, Appointment thisAppt) {

//is appt's start time betweent thisAppt's start time and end time

if (appt.getApptInfo()[3] >= thisAppt.getApptInfo()[3] && appt.getApptInfo()[3] <= thisAppt.getApptInfo()[4]) {

MainActivityClass.displayErrorDialogBox("This appointment conflicts with another appoinment on the same day at: " + thisAppt.getStartTime());

return false;

}

//is appt's end time between thisAppt's start time and endtime

if (appt.getApptInfo()[4] >= thisAppt.getApptInfo()[3] && appt.getApptInfo()[4] <= thisAppt.getApptInfo()[4]) {

MainActivityClass.displayErrorDialogBox("This appointment conflicts with another appoinment on the same day at: " + thisAppt.getStartTime());

return false;

}

return true;

}

//creates a temporary date object using the information filled out in the JFrame

private Date createDate() {

int year = Integer.parseInt(myParameters.get(0)) - 1900;

int month = Integer.parseInt(myParameters.get(1)) - 1;

int dayOfMonth = Integer.parseInt(myParameters.get(2));

int hrs = Integer.parseInt(myParameters.get(3));

int min = Integer.parseInt(myParameters.get(4));

int startTime = Appointment.convertToMilit(hrs, min, myParameters.get(5));

return new Date(year, month, dayOfMonth, startTime / 100, startTime % 100);

}

/\*\*

\* @param args the command line arguments

\*/

public void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ModifyAppointment.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ModifyAppointment.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ModifyAppointment.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ModifyAppointment.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

if(myOption.equals("Add"))

new ModifyAppointment(myAppointments, myOption).setVisible(true);

else

new ModifyAppointment(myAppointments, myApptToEdit, myOption).setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JTextField firstName;

private javax.swing.JComboBox<String> jComboBox2;

private javax.swing.JComboBox<String> jComboBox3;

private javax.swing.JComboBox<String> jComboBox4;

private javax.swing.JComboBox<String> jComboBox5;

private javax.swing.JComboBox<String> jComboBox6;

private javax.swing.JComboBox<String> jComboBox7;

private javax.swing.JComboBox<String> jComboBox8;

private javax.swing.JComboBox<String> jComboBox9;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel11;

private javax.swing.JLabel jLabel12;

private javax.swing.JLabel jLabel13;

private javax.swing.JLabel jLabel14;

private javax.swing.JLabel jLabel15;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JLabel jLabel8;

private javax.swing.JLabel jLabel9;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField lastName;

private javax.swing.JButton modifyAppt;

// End of variables declaration

}

ModifyStudent.java

package internalassesment;

import java.awt.Color;

import java.util.ArrayList;

public class ModifyStudent extends javax.swing.JFrame {

private ArrayList<Student> myStudents;

private ArrayList<String> myParams = new ArrayList<String>(4);

private final String myOption;

private Student myStudentToEdit;

//constructor if user wants to add student

//Only arraylist of students is reqeuired

public ModifyStudent(ArrayList<Student> students, String option) {

myStudents = students;

myOption = option;

initComponents();

if (myOption.equals("Search")) {

jLabel1.setText("Please enter at least one field");

modStdnt.setText("Search");

} else{

jLabel1.setText("Please enter at least a first or last name");

modStdnt.setText("Add");

}

}

public ModifyStudent(ArrayList<Student> students,Student student, String option) {

initComponents();

modStdnt.setText("SaveChanges");

myStudents = students;

myOption = option;

myStudentToEdit = student;

jTextField1.setText(myStudentToEdit.getFirstName());

jTextField2.setText(myStudentToEdit.getLastName());

jTextField3.setText(myStudentToEdit.getEmailAddress());

jTextField4.setText(myStudentToEdit.getPhoneNumber());

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jTextField1 = new javax.swing.JTextField();

jTextField2 = new javax.swing.JTextField();

jTextField3 = new javax.swing.JTextField();

jLabel1 = new javax.swing.JLabel();

modStdnt = new javax.swing.JButton();

jTextField4 = new javax.swing.JTextField();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jLabel1.setText("Enter at least One");

modStdnt.setText("Add");

modStdnt.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

modifyStudent(evt);

}

});

jLabel2.setText("First Name:");

jLabel3.setText("Last Name:");

jLabel4.setText("Email Address: ");

jLabel5.setText("Phone Numbe:");

jLabel6.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel6.setText("Student Form");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(modStdnt, javax.swing.GroupLayout.PREFERRED\_SIZE, 70, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jLabel5)

.addComponent(jLabel4)

.addComponent(jLabel3)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jTextField1)

.addComponent(jTextField2)

.addComponent(jTextField4, javax.swing.GroupLayout.DEFAULT\_SIZE, 126, Short.MAX\_VALUE)

.addComponent(jTextField3)))

.addComponent(jLabel7)

.addComponent(jLabel1)))

.addGroup(layout.createSequentialGroup()

.addGap(64, 64, 64)

.addComponent(jLabel6)))

.addContainerGap(39, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jLabel1)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel2))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel4)

.addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED\_SIZE, 38, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(9, 9, 9)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED\_SIZE, 45, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel5))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jLabel7)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(modStdnt, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap())

);

pack();

}// </editor-fold>

private void modifyStudent(java.awt.event.ActionEvent evt) {

myParams.clear();

if(!addParams())

return;

if (myOption.equals("Add")) {

add();

} else if (myOption.equals("Search"))

{

search();

} else {

edit();

}

myParams.removeAll(myParams);

}

private void edit() {

int x = myStudents.indexOf(myStudentToEdit);

addParams();

myStudentToEdit.setStudentFirstName(myParams.get(0));

myStudentToEdit.setStudentLastName(myParams.get(1));

myStudentToEdit.setEmailAddress(myParams.get(2));

myStudentToEdit.setPhoneNumber(myParams.get(3));

myStudents.remove(x);

myStudents.add(myStudentToEdit);

Student.quickSort(myStudents, 0, myStudents.size() - 1);

}

//adds parameters to arrayList

private boolean addParams() {

String firstName = jTextField1.getText();

String lastName = jTextField2.getText();

String emailAddress = jTextField3.getText();

String phoneNumber = jTextField4.getText();

myParams.add(firstName);

myParams.add(lastName);

myParams.add(phoneNumber);

myParams.add(emailAddress);

int y = 0;

if(myOption.equals("Add") &&( myParams.get(0).length()==0 && myParams.get(1).length()==0)){

MainActivityClass.displayErrorDialogBox("Enter a first or last name for student");

return false;

}

for (int x = 0; x < myParams.size(); x++) { //changes any null value to "---"

if (myParams.get(x).isEmpty()) {

y++;

myParams.set(x, "---");

}

}

if (y == 4) {

MainActivityClass.displayErrorDialogBox("You have not entered any of the fields");

return false;

}

return true;

}

private void search() {

ArrayList<Student> validStudents = findValidStudents(); /\*arraylist that hold Student objects that meet required parameters\*/

if(validStudents.size() == 0) {

/\*if there are no students with required parameters\*/

javax.swing.JDialog error = new javax.swing.JDialog();

error.add(new javax.swing.JLabel("No student found. Try again"));

error.setSize(300,300);

error.setVisible(true);

} else { //else display list of valid students

List studentList = new List("Student", validStudents, myStudents);

studentList.setVisible(true);

studentList.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

}

}

//adds student if parameters are valid

private void add() {

myStudents.add(new Student(myParams.get(0), myParams.get(1), myParams.get(2), myParams.get(3)));

Student.quickSort(myStudents, 0, myStudents.size() - 1);

jLabel7.setText("Student has been added");

jTextField1.setText(null);

jTextField2.setText(null);

jTextField3.setText(null);

jTextField4.setText(null);

}

//finds Student objects that fit the requred parameters

private ArrayList<Student> findValidStudents() {

ArrayList<Student> validStudents = new ArrayList<Student>();

/\*araylist that will hold Student objects meeting required parameters \*/

for (int x = 0; x < myStudents.size(); x++) { //traverses through all the Student objects

boolean isValidStudent = true;

for (int y = 0; y < myParams.size(); y++) {

if (!myParams.get(y).equals("---")) { //compare if their is a parameter

if (myParams.get(y).compareTo(myStudents.get(x).getStudentInfo()[y]) != 0) { //if that parameter does not match to the student info

isValidStudent = false; //not a valid student

System.out.println(myStudents.get(x).getStudentInfo()[y]);

y = myParams.size();

}

}

}

if (isValidStudent) { // if all the parameters matched student's info

validStudents.add(myStudents.get(x));

}

}

return validStudents;

}

/\*\*

\* @param args the command line arguments

\*/

public void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ModifyStudent.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ModifyStudent.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ModifyStudent.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ModifyStudent.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ModifyStudent(myStudents, myOption).setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JTextField jTextField1;

private javax.swing.JTextField jTextField2;

private javax.swing.JTextField jTextField3;

private javax.swing.JTextField jTextField4;

private javax.swing.JButton modStdnt;

// End of variables declaration

}

Reminder.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.File;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

/\*\*

\*

\* @author Alay

\*/

public class Reminder extends javax.swing.JFrame{

/\*\*

\* Creates new form Reminder

\*/

private String myReminderMessage = "You have an appointment tomorrow";

private Clip myClip; //audio clip that plays when the JFrame opens

public Reminder() {

initComponents();

jLabel1.setText(myReminderMessage); //adds the remind message to the JFrame

try {

myClip = AudioSystem.getClip();

myClip.open(AudioSystem.getAudioInputStream(new File("AlarmClock.wav"))); //opens the file that audio clip should play

myClip.start();

} catch (Exception ex) {

myClip = null;

}

setVisible(true);

setSize(300,200);

setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

addWindowListener(new WindowAdapter() { //adds window listener to detect when program closes

@Override

public void windowClosing(WindowEvent e) { //overrides the default method for when window is closin

try {

myClip.stop(); //stops audio while the JFrame is losing

} catch (Exception ex) {

return ;

}

}

});

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jLabel1.setText("jLabel1");

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addContainerGap(170, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(jLabel1)

.addContainerGap(52, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Reminder.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Reminder.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Reminder.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Reminder.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Reminder().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel1;

// End of variables declaration

}

SearchAppt.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import com.sun.glass.events.KeyEvent;

import java.util.\*;

import javax.swing.JOptionPane;

/\*\*

\*

\* @author Alay

\*/

public class SearchAppt extends javax.swing.JFrame {

private ArrayList<Appointment> myAppointments;

private ArrayList<String> myParams;

/\*\*

\* Creates new form SearchAppt

\*/

public SearchAppt(ArrayList<Appointment> appointments) {

initComponents();

myAppointments = appointments;

myParams = new ArrayList<String>();

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

myMonth = new javax.swing.JComboBox<String>();

myDate = new javax.swing.JComboBox<String>();

jLabel2 = new javax.swing.JLabel();

name = new javax.swing.JTextField();

search = new javax.swing.JButton();

myYear = new javax.swing.JTextField();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

myMonth.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "----", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12" }));

myDate.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "--", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29", "30", "31" }));

jLabel2.setText("Name");

name.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

nameActionPerformed(evt);

}

});

search.setText("Search");

search.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

search(evt);

}

});

myYear.addKeyListener(new java.awt.event.KeyAdapter() {

public void keyTyped(java.awt.event.KeyEvent evt) {

myYearKeyTyped(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(myYear, javax.swing.GroupLayout.PREFERRED\_SIZE, 68, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(myMonth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addComponent(myDate, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel2)

.addGap(18, 18, 18)

.addComponent(name, javax.swing.GroupLayout.PREFERRED\_SIZE, 112, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addComponent(search))

.addContainerGap(55, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(32, 32, 32)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(myMonth, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(myDate, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(myYear, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(name, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(search)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

pack();

}// </editor-fold>

private void search(java.awt.event.ActionEvent evt) {

if(!addParams()){

return;

}

ArrayList <Appointment> validAppts = findValidAppointments(); //adds appt to an arraylist of valid appts if valid

if(validAppts.size()!=0){ //displays the arraylist of valid appts if its not empty

List list = new List(validAppts, myAppointments);

list.setVisible(true);

list.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

} else MainActivityClass.displayErrorDialogBox("No appointments found"); //displays error if it is empty

}

private ArrayList<Appointment> findValidAppointments(){

ArrayList<Appointment> validAppts = new ArrayList<Appointment>();

for(int x = 0;x<myAppointments.size();x++){ //begins traversing myAppointments

boolean isValidAppt = true; //assumes that the current appointment in arraylist is valid

Appointment appt = myAppointments.get(x);

for(int y = 0;y<3;y++){ //traverses thru first 3 pieces of appt information: Year, month, date

if(!myParams.get(y).contains("--") && myParams.get(y).length()!=0){ //ignores any empty parameters

if(appt.getApptInfo()[y]!= (Integer.parseInt(myParams.get(y)))){ //comapres two appts using one piece of their information(year,month,date)

isValidAppt = false; //not a valid appt if the information is not equal

y=3;

}

}

}

if(name.getText().length()!=0){ //separately checks if names are equal because name is not part of the apptInfo[] in appt class

System.out.println(name.getText() + "\n" + appt.getName());

if(name.getText().equals(appt.getName()) && isValidAppt){

System.out.println(1);

isValidAppt = true; //not a valid appt if names are not equal

}

else {

System.out.println(2);

isValidAppt = false;

}

}

System.out.println(isValidAppt);

if(isValidAppt) validAppts.add(myAppointments.get(x));

}

return validAppts;

}

private boolean addParams(){

myParams.clear();

myParams.add(myYear.getText());

myParams.add((String)myMonth.getSelectedItem());

myParams.add((String) myDate.getSelectedItem());

return isValidParams();

}

private void myYearKeyTyped(java.awt.event.KeyEvent evt) {

// TODO add your handling code here:

char letter = evt.getKeyChar();

if(!(Character.isDigit(letter)) || (letter == KeyEvent.VK\_BACKSPACE) || (letter == KeyEvent.VK\_DELETE))

evt.consume();

}

private void nameActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

}

private boolean isValidParams(){

//check if all params are empty including name;

String errorMessage = "Please enter at least the year";

boolean isValidAppt = false;

if(name.getText().length()>0)

isValidAppt = true;

else{

for(int x = 0;x<myParams.size();x++){

if(myParams.get(x).length()!=0 || !myParams.get(x).contains("--")){

isValidAppt = true;

x = myParams.size();

}

}

}

//check if year and date is entered without month

if(((String)myYear.getText()).length()>0){

if(((String)myMonth.getSelectedItem()).contains("--") && !((String)myDate.getSelectedItem()).contains("--")){

errorMessage = "Please enter month to search for a specific day.";

isValidAppt = false;

}

else isValidAppt = true;

}

else { //check if month and/or date is entered without year

if(!((String)myMonth.getSelectedItem()).contains("--") || !((String)myDate.getSelectedItem()).contains("--")){

errorMessage = "Please enter the year to narrow your search";

isValidAppt = false;

}

else isValidAppt = true;

}

if(!isValidAppt)

MainActivityClass.displayErrorDialogBox(errorMessage);

return isValidAppt;

}

/\*\*

\* @param args the command line arguments

\*/

public void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(SearchAppt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(SearchAppt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(SearchAppt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(SearchAppt.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new SearchAppt(myAppointments).setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JLabel jLabel2;

private javax.swing.JComboBox<String> myDate;

private javax.swing.JComboBox<String> myMonth;

private javax.swing.JTextField myYear;

private javax.swing.JTextField name;

private javax.swing.JButton search;

// End of variables declaration

}

Stdent.java

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package internalassesment;

import java.util.ArrayList;

/\*\*

\*

\* @author 061264

\*/

public class Student {

//parameters of student object stored in an array

private String [] myStudentInfo = new String[4];

//student information is always stored in the array in this order

public Student(String firstName, String lastName,String phoneNumber,String emailAddress){

myStudentInfo[0] = firstName;

myStudentInfo[1] = lastName;

myStudentInfo[2] = phoneNumber;

myStudentInfo[3] = emailAddress;

}

public String getFirstName(){

return myStudentInfo[0];

}

public String getLastName(){

return myStudentInfo[1];

}

public String getPhoneNumber(){

return myStudentInfo[2];

}

public String getEmailAddress(){

try{

return myStudentInfo[3];

}

catch(NullPointerException e){

return "This student does not have an email address or has not been added";

}

}

public String [] getStudentInfo(){

return myStudentInfo;

}

public void setStudentFirstName(String firstName){

myStudentInfo[0]= firstName;

}

public void setStudentLastName(String lastName){

myStudentInfo[1] = lastName;

}

public void setPhoneNumber(String phoneNumber){

myStudentInfo[2] = phoneNumber;

}

public void setEmailAddress(String emailAddress){

myStudentInfo[3] = emailAddress;

}

public String toString(){

String info = getFirstName() + " " + getLastName() + " ";

if(!getEmailAddress().equals("---"))

info += getEmailAddress() + " ";

if(!getPhoneNumber().equals("---"))

info += getPhoneNumber();

return info;

}

public int compareTo(Student student){

for(int x =0;x<myStudentInfo.length;x++){

if(myStudentInfo[x].compareToIgnoreCase(student.getStudentInfo()[x])!=0)

return myStudentInfo[x].compareToIgnoreCase(student.getStudentInfo()[x]);

}

return 0;

}

/\*sorts any arraylist of students using quicksort

algorithm and puts them in alphabetical order \*/

public static void quickSort(ArrayList<Student> info, int first,int last){

int f = first;

int l = last;

int midIndex = (first+last)/2;

Student obj = (Student) info.get(midIndex);

do{

while(((Student)info.get(f)).compareTo(obj)<0){

f++;

}

while(((Student)info.get(l)).compareTo(obj)>0){

l--;

}

if(f<=l){

swap(info,f,l);

f++;

l--;

}

}while(f<l);

if(l>first){

quickSort(info,first,l);

}

if(f<last){

quickSort(info,f,last);

}

}

private static void swap(ArrayList<Student> info, int x, int y){

Student ex = (Student) info.get(x);

info.set(x,info.get(y));

info.set(y,ex);

}

}