תומר קצב (ת.ז 302175716)

ניר מקין (301734158)

**מסדי נתונים – פרויקט סוף:**

package main;  
  
import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.sql.\*;  
import java.util.Vector;  
  
public class guiCheck extends JFrame {  
  
 static final String JDBC\_Driver = "com.mysql.jdbc.Driver"; // jdbc driver and db url  
 static final String DB\_URL = "jdbc:mysql://localhost:3306/college";  
 static final String UserName = "root"; // DB login information  
 static final String Password = "Ktzv3404";  
 private JPanel panelA;  
 private JTextField buildingTextField;  
 private JTextField classNumTextField;  
 private JTextField floorTextField;  
 private JTable classesTable;  
 private JTextField firstNameTextField;  
 private JLabel lastNameLabel;  
 private JLabel firstNameLabel;  
 private JTextField lastNameTextField;  
 private JLabel phoneLabel;  
 private JTextField phoneTextField;  
 private JLabel adressLabel;  
 private JTextField addressTextField;  
 private JComboBox dayComboBox;  
 private JComboBox monthComboBox;  
 private JComboBox yearCombo;  
 private JLabel courseIDLabel;  
 private JLabel subjectLabel;  
 private JTextField courseNumTextField;  
 private JLabel semesterLabel;  
 private JComboBox semesterComboBox;  
 private JLabel yearLabel;  
 private JComboBox yearComboBox;  
 private JLabel weeklyHoursLabel;  
 private JTextField weeklyHoursTextField;  
 private JButton classesButton;  
 private JTable coursesTable;  
 private JTable lecturersTable;  
 private JButton connectButton;  
 private JComboBox classesBox;  
 private JButton closeConnectionButton;  
 private JTextField consoleTextField;  
 private JButton coursesButton;  
 private JComboBox coursesComboBox;  
 private JTextField subjectTextField;  
 private JButton lecturersButton;  
 private JComboBox lecturersComboBox;  
 private JTextField idTextField;  
 private JTable phonesTable;  
 private JTable schedulerTable;  
 private JRadioButton classesQueryRadioButton;  
 private JRadioButton lecturersQueryRadioButton;  
 private JRadioButton timeRangeQueryRadioButton;  
 private JButton schedulerButton;  
 private JRadioButton schedulerRadioButton;  
 private JComboBox daysFromComboBox;  
 private JComboBox daysToComboBox;  
 private JComboBox hoursFromComboBox;  
 private JComboBox hoursToComboBox;  
 private JComboBox yearComboBox2;  
 private JTextField classesQueryTextField;  
 private JTextField lecturersQueryTextField;  
 Connection connection = null;  
 Statement statement = null;  
 ResultSet resultQuery = null;  
  
 public guiCheck() {  
 super("College Scheduler");  
 run();  
 }  
  
 public void run() {  
 classesTable.setAutoCreateRowSorter(true);  
 classesTable.setFillsViewportHeight(true);  
 coursesTable.setAutoCreateRowSorter(true);  
 coursesTable.setFillsViewportHeight(true);  
 lecturersTable.setAutoCreateRowSorter(true);  
 lecturersTable.setFillsViewportHeight(true);  
 consoleTextField.setText("Disconnected");  
 this.setSize(1280, 717);  
 classesQueryTextField.setForeground(Color.GRAY);  
 classesQueryTextField.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 if (classesQueryTextField.getText().equals("Enter Class Number")) {  
 classesQueryTextField.setText("");  
 classesQueryTextField.setForeground(Color.BLACK);  
 }  
 }  
 @Override  
 public void focusLost(FocusEvent e) {  
 if (classesQueryTextField.getText().isEmpty()) {  
 classesQueryTextField.setForeground(Color.GRAY);  
 classesQueryTextField.setText("Enter Class Number");  
 }  
 }  
 });  
 lecturersQueryTextField.setForeground(Color.GRAY);  
 lecturersQueryTextField.addFocusListener(new FocusListener() {  
 @Override  
 public void focusGained(FocusEvent e) {  
 if (lecturersQueryTextField.getText().equals("Enter Lecturer's Name")) {  
 lecturersQueryTextField.setText("");  
 lecturersQueryTextField.setForeground(Color.BLACK);  
 }  
 }  
 @Override  
 public void focusLost(FocusEvent e) {  
 if (lecturersQueryTextField.getText().isEmpty()) {  
 lecturersQueryTextField.setForeground(Color.GRAY);  
 lecturersQueryTextField.setText("Enter Lecturer's Name");  
 }  
 }  
 });  
 setContentPane(panelA);  
 setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 connectButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 createConnection();  
 }  
 });  
 classesButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 editClasses();  
 }  
 });  
 closeConnectionButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 closeConnection();  
 }  
 });  
 coursesButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 editCourses();  
 }  
 });  
 lecturersButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 editLecturers();  
 }  
 });  
 schedulerButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 runSelectedQuery();  
 disableCheckBoxes();  
 }  
 });  
 timeRangeQueryRadioButton.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 daysFromComboBox.setEnabled(true);  
 daysToComboBox.setEnabled(true);  
 hoursFromComboBox.setEnabled(true);  
 hoursToComboBox.setEnabled(true);  
 }  
  
 });  
  
 classesQueryRadioButton.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 }  
  
 });  
  
 classesQueryRadioButton.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 disableCheckBoxes();  
 }  
  
 });  
  
 lecturersQueryRadioButton.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 disableCheckBoxes();  
 }  
  
 });  
 setResizable(false);  
 setVisible(true);  
 createConnection();  
 }  
  
 public void disableCheckBoxes(){  
 daysFromComboBox.setEnabled(false);  
 daysToComboBox.setEnabled(false);  
 hoursFromComboBox.setEnabled(false);  
 hoursToComboBox.setEnabled(false);  
 }  
  
 public DefaultTableModel createNewTable(ResultSet rs) throws SQLException {  
 Vector<String> columnNames = new Vector<String>();  
 Vector<Vector<Object>> tableData = new Vector<Vector<Object>>();  
 int columnCount;  
 ResultSetMetaData rsmd = rs.getMetaData();  
 columnCount = rsmd.getColumnCount();  
 for (int i = 1; i <= columnCount; i++)  
 columnNames.add(rsmd.getColumnName(i));  
 while (rs.next()) {  
 Vector<Object> vector = new Vector<Object>();  
 for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {  
 vector.add(rs.getObject(columnIndex));  
 }  
 tableData.add(vector);  
 }  
 return new DefaultTableModel(tableData, columnNames);  
 }  
  
 private void createRows(Statement statement) {  
 try {  
 statement.executeUpdate("INSERT INTO `Classes`" +  
 "VALUES (2105, 'Mitchell', 1)");  
 statement.executeUpdate("INSERT INTO `Classes`" +  
 "VALUES (2204, 'Mitchell', 2)");  
 statement.executeUpdate("INSERT INTO `Classes`" +  
 "VALUES (246, 'Fernik', 3)");  
 statement.executeUpdate("INSERT INTO Classes " +  
 "VALUES (247, 'Fernik', 3), (35, 'Gallery', 0), (66, 'Library', 5), (61, 'Mitchell', 0), (0, 'Fernik', -1), " +  
 "(23, 'Center', 25), (2104, 'Mitchell', 1);");  
 statement.executeUpdate("INSERT INTO `Lecturers`" +  
 "VALUES (302175716, 'Tomer Katzav', 27, 'Poleg 1', '04/04/1989')");  
 statement.executeUpdate("INSERT INTO `Lecturers`" +  
 "VALUES (301734158, 'Nir Mekin', 27, 'Levi 1', '07/05/1989')");  
 statement.executeUpdate("INSERT INTO `Lecturers`" +  
 "VALUES (456102333, 'Moshe Levi', 40, 'Sokolov 3', '30/02/1976')");  
 statement.executeUpdate("INSERT INTO Lecturers " +  
 "VALUES (123456789, 'Yogev Hezkia', 19, 'Ana Frank 12', '04/08/1997'), (994534123, 'Yotam Akshota', 26, 'Levontin 5', '10/10/1990');");  
 statement.executeUpdate("INSERT INTO Lecturers " +  
 "VALUES (111123000, 'Shamir Kritzler', 34, 'Osishkin 20', '08/12/1982'), (456000456, 'Alexander Djura', 40, 'Usha 7', '31/02/1976');");  
 statement.executeUpdate("INSERT INTO Lecturers " +  
 "VALUES (123456739, 'Yossi Efraim', 15, 'Pisnker 1', '04/01/2001'), (999534023, 'Shuli Cohen', 52, 'Brurya 12', '09/11/1964');");  
 statement.executeUpdate("INSERT INTO Lecturers " +  
 "VALUES (555444111, 'Natalie Levy', 22, 'Alenby 20', '12/06/1994');");  
 statement.executeUpdate("INSERT INTO `Courses`" +  
 "VALUES (31, 'Histroy', 'A', '2nd', 4)");  
 statement.executeUpdate("INSERT INTO `Courses`" +  
 "VALUES (6, 'Mathematics', 'B', '1st', 8)");  
 statement.executeUpdate("INSERT INTO `Courses`" +  
 "VALUES (50, 'Arts', 'Summer', '3rd', 2)");  
 statement.executeUpdate("INSERT INTO Courses " +  
 "VALUES (23, 'Physics', 'A', '1st', 6), (52, 'Algebra', 'B', '2nd', 3), (5, 'Sports', 'Summer', '4th', 2);");  
 statement.executeUpdate("INSERT INTO Courses " +  
 "VALUES (15, 'Algorithms', 'B', '2nd', 5), (10, 'Computer Science', 'A', '1st', 7), (200, 'Programming Languages', 'Summer', '3rd', 4);");  
 statement.executeUpdate("INSERT INTO Courses " +  
 "VALUES (76, 'Statistics', 'A', '4th', 2);");  
 statement.executeUpdate("INSERT INTO Phones " +  
 "VALUES (0521111111, 302175716), (0521111112, 302175716)");  
 statement.executeUpdate("INSERT INTO Phones " +  
 "VALUES (0521111113, 301734158), (0521111114, 301734158)");  
 statement.executeUpdate("INSERT INTO Phones " +  
 "VALUES (0521111115, 456102333), (0521111116, 456102333)");  
 statement.executeUpdate("INSERT INTO `Scheduler` " +  
 "VALUES (302175716, 2105, 31, 'Monday', '08:00')");  
 statement.executeUpdate("INSERT INTO `Scheduler` " +  
 "VALUES (301734158, 2204, 6, 'Wednesday', '12:00')");  
 statement.executeUpdate("INSERT INTO `Scheduler` " +  
 "VALUES (456102333, 246, 50, 'Tuesday', '09:30')");  
 statement.executeUpdate("INSERT INTO Scheduler " +  
 "VALUES (994534123, 0, 5, 'Friday', '10:00'), (123456739, 35, 10, 'Sunday', '16:00'), (123456789, 61, 15, 'Monday', '07:00'), (999534023, 66, 23, 'Thursday', '11:00');");  
 statement.executeUpdate("INSERT INTO Scheduler " +  
 "VALUES (111123000, 247, 52, 'Monday', '12:00'), (555444111, 23, 76, 'Wednesday', '16:30'), (456000456, 2104, 200, 'Friday', '08:00');");  
 statement.executeUpdate("INSERT INTO WeekDays " +  
 "VALUES (1,'Sunday'), (2,'Monday'), (3,'Tuesday'), (4,'Wednesday'), (5,'Thursday'), (6,'Friday'), (7,'Saturday')");  
 } catch (SQLException se) {  
 System.out.println("SQL Exception");  
 se.printStackTrace();  
 } catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 private void createTables(Statement statement) {  
 try {  
 statement.execute("CREATE TABLE IF NOT EXISTS Lecturers " +  
 "(ID INTEGER, " +  
 "Name VARCHAR(25), " +  
 "Age INTEGER, " +  
 "Address VARCHAR(25), " +  
 "BirthDate VARCHAR(25), " +  
 "PRIMARY KEY ( ID ))");  
 statement.execute("CREATE TABLE IF NOT EXISTS Phones " +  
 "(PhoneNum INTEGER, " +  
 "ID INTEGER NOT NULL, " +  
 "FOREIGN KEY (ID) REFERENCES Lecturers (ID) " +  
 "ON DELETE CASCADE " +  
 "ON UPDATE CASCADE)");  
 statement.execute("CREATE TABLE IF NOT EXISTS Courses " +  
 "(CourseNum INTEGER, " +  
 "Subject VARCHAR(25), " +  
 "Semester VARCHAR(25), " +  
 "Year VARCHAR(25), " +  
 "Weekly\_Hours INTEGER, " +  
 "PRIMARY KEY ( CourseNum ))");  
 statement.execute("CREATE TABLE IF NOT EXISTS Classes " +  
 "(ClassNum INTEGER, " +  
 "Building VARCHAR(25), " +  
 "Floor INTEGER, " +  
 "PRIMARY KEY ( ClassNum ))");  
 statement.execute("CREATE TABLE IF NOT EXISTS Scheduler " +  
 "(ID INTEGER, " +  
 "ClassNum INTEGER, " +  
 "CourseNum INTEGER, " +  
 "Day VARCHAR(25), " +  
 "Hour VARCHAR(25), " +  
 "FOREIGN KEY (ID) REFERENCES Lecturers (ID) " +  
 "ON DELETE CASCADE " +  
 "ON UPDATE CASCADE, " +  
 "FOREIGN KEY (CourseNum) REFERENCES Courses (CourseNum) " +  
 "ON DELETE CASCADE " +  
 "ON UPDATE CASCADE, " +  
 "FOREIGN KEY (ClassNum) REFERENCES Classes (ClassNum) " +  
 "ON DELETE CASCADE " +  
 "ON UPDATE CASCADE)");  
 statement.execute("CREATE TABLE IF NOT EXISTS WeekDays " +  
 "(DayNum INT PRIMARY KEY, " +  
 "Day VARCHAR(25))");  
 statement.execute("CREATE TABLE updateTimeTableClasses ( " +  
 " tableName VARCHAR(25), " +  
 " lastUpdate timestamp" +  
 ");");  
 statement.execute("CREATE TRIGGER classes\_trigger AFTER INSERT ON classes " +  
 " FOR EACH ROW INSERT INTO updateTimeTableClasses VALUES (\"Classes Table\",DEFAULT );");  
 resultQuery = statement.executeQuery("SELECT \* FROM Scheduler");  
 if (!resultQuery.first()) {  
 System.out.println("Adding default tuples");  
 createRows(statement);  
 }  
 } catch (SQLException se) {  
 System.out.println("SQL Exception");  
 se.printStackTrace();  
 } catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 public void createConnection() {  
 try {  
 Class.forName("com.mysql.jdbc.Driver");  
 System.out.println("Connecting to the selected database... Please hold on");  
 connection = DriverManager.getConnection(DB\_URL, UserName, Password);  
 System.out.println("You are now successfully connected to the database!");  
 statement = connection.createStatement();  
 createTables(statement);  
 System.out.println(statement);  
 ResultSet rsClasses = statement.executeQuery("SELECT \* FROM Classes");  
 classesTable.setModel(createNewTable(rsClasses));  
 ResultSet rsCourses = statement.executeQuery("SELECT \* FROM Courses");  
 coursesTable.setModel(createNewTable(rsCourses));  
 ResultSet rsLecturers = statement.executeQuery("SELECT \* FROM Lecturers");  
 lecturersTable.setModel(createNewTable(rsLecturers));  
 ResultSet rsPhones = statement.executeQuery("SELECT \* FROM Phones");  
 phonesTable.setModel(createNewTable(rsPhones));  
// ResultSet rsScheduler = statement.executeQuery("SELECT \* FROM Scheduler");  
// schedulerTable.setModel(createNewTable(rsScheduler));  
 phonesTable.getColumnModel().getColumn(1).setPreferredWidth(35);  
 lecturersTable.getColumnModel().getColumn(2).setPreferredWidth(20);  
 lecturersTable.getColumnModel().getColumn(3).setPreferredWidth(90);  
 coursesTable.getColumnModel().getColumn(3).setPreferredWidth(25);  
 coursesTable.getColumnModel().getColumn(2).setPreferredWidth(45);  
 consoleTextField.setText("Connected to " + statement);  
 //addButtons();  
 } catch (SQLException se) {  
 System.out.println("SQL Exception");  
 se.printStackTrace();  
 } catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 public void closeConnection() {  
 try{  
 if (statement != null)  
 connection.close();  
 }catch(SQLException se){  
 System.out.println("SQL Exception while disconnecting");  
 se.printStackTrace();  
 }  
 System.out.println("The program will now end!");  
 consoleTextField.setText("Disconnected");  
 }  
  
  
 public void editClasses(){  
 PreparedStatement pStmt = null;  
 connection = null;  
 statement = null;  
 try{  
 connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/college","root","Ktzv3404");  
 connection.setAutoCommit(false);  
 statement = connection.createStatement();  
 if (classesBox.getSelectedItem() == "Create") {  
 if (buildingTextField.getText().equals("") || classNumTextField.getText().equals("") || floorTextField.getText().equals("")) {  
 JOptionPane.showMessageDialog(null, "Missing Required Fields! No Entry Was Created!");  
 }  
 else {  
 pStmt = connection.prepareStatement("INSERT INTO Classes " +  
 "VALUES (?, ?, ?)");  
 pStmt.setInt(1, Integer.parseInt(classNumTextField.getText()));  
 pStmt.setString(2, buildingTextField.getText());  
 pStmt.setInt(3, Integer.parseInt(floorTextField.getText()));  
 }  
 }  
 else if (classesBox.getSelectedItem() == "Delete") {  
 if (classNumTextField.getText().equals(""))  
 JOptionPane.showMessageDialog(null, "Missing a Class Number Primary Key");  
 else {  
 pStmt = connection.prepareStatement("DELETE FROM Classes " +  
 "WHERE ClassNum = (?)");  
 pStmt.setInt(1, Integer.parseInt(classNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "Class # " + classNumTextField.getText() + " Was Removed From The Database!");  
 }  
 }  
 else if (classesBox.getSelectedItem() == "Edit") {  
 if (classNumTextField.getText().equals(""))  
 JOptionPane.showMessageDialog(null, "Missing a Class Number Primary Key");  
 else {  
 if (buildingTextField.getText().equals("") && floorTextField.getText().equals("")) {  
 JOptionPane.showMessageDialog(null, "No Values To Update Were Entered");  
 }  
 else if (buildingTextField.getText() != null && floorTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Classes " +  
 "SET Building = (?) " +  
 "WHERE ClassNum = (?)");  
 pStmt.setString(1, buildingTextField.getText());  
 pStmt.setInt(2, Integer.parseInt(classNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "You Successfully Edited an Entry");  
 }  
 else if (buildingTextField.getText().equals("") && floorTextField.getText() != null) {  
 pStmt = connection.prepareStatement("UPDATE Classes " +  
 "SET Floor = (?) " +  
 "WHERE ClassNum = (?)");  
 pStmt.setInt(1, Integer.parseInt(floorTextField.getText()));  
 pStmt.setInt(2, Integer.parseInt(classNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "You Successfully Edited an Entry");  
 }  
 else {  
 pStmt = connection.prepareStatement("UPDATE Classes " +  
 "SET Building = (?), " +  
 "Floor = (?) " +  
 "WHERE ClassNum = (?)");  
 pStmt.setString(1, buildingTextField.getText());  
 pStmt.setInt(2, Integer.parseInt(floorTextField.getText()));  
 pStmt.setInt(3, Integer.parseInt(classNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "You Successfully Edited an Entry");  
 }  
 }  
 }  
 pStmt.executeUpdate();  
 connection.commit();  
 resultQuery = statement.executeQuery("SELECT \* FROM Classes");  
 classesTable.setModel(createNewTable(resultQuery));  
 }catch(SQLException ex){  
 try {  
 if (connection != null)  
 connection.rollback();  
 } catch (SQLException ex2) {  
 JOptionPane.showMessageDialog(null, ex2.getMessage());  
 ex2.printStackTrace();  
 }  
 JOptionPane.showMessageDialog(null, ex.getMessage());  
 ex.printStackTrace();  
 }catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 public void editCourses() {  
 PreparedStatement pStmt = null;  
 connection = null;  
 statement = null;  
 try {  
 connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/college", "root", "Ktzv3404");  
 connection.setAutoCommit(false);  
 statement = connection.createStatement();  
 if (coursesComboBox.getSelectedItem() == "Create") {  
 if (courseNumTextField.getText().equals("") || subjectTextField.getText().equals("") || weeklyHoursTextField.getText().equals("")){  
 JOptionPane.showMessageDialog(null, "Missing Required Fields! No Entry Was Created!");  
 }  
 else {  
 pStmt = connection.prepareStatement("INSERT INTO Courses " +  
 "VALUES (?, ?, ?, ?, ?)");  
 pStmt.setInt(1, Integer.parseInt(courseNumTextField.getText()));  
 pStmt.setString(2, subjectTextField.getText());  
 pStmt.setString(3, semesterComboBox.getSelectedItem().toString());  
 pStmt.setString(4, yearComboBox.getSelectedItem().toString());  
 pStmt.setInt(5, Integer.parseInt(weeklyHoursTextField.getText()));  
 }  
 }  
 else if (coursesComboBox.getSelectedItem() == "Delete") {  
 if (courseNumTextField.getText().equals(""))  
 JOptionPane.showMessageDialog(null, "Missing a Course Number Primary Key");  
 else {  
 pStmt = connection.prepareStatement("DELETE FROM Courses " +  
 "WHERE CourseNum = (?)");  
 pStmt.setInt(1, Integer.parseInt(courseNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "Course #" + courseNumTextField.getText() + " Was Removed From The Database!");  
 }  
 }  
 else if (coursesComboBox.getSelectedItem() == "Edit") {  
 if (courseNumTextField.equals(""))  
 JOptionPane.showMessageDialog(null, "Missing a Course Number Primary Key");  
 else {  
 if (subjectTextField.getText().equals("") && weeklyHoursTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Courses " +  
 "SET Semester = (?), " +  
 "Year = (?) " +  
 "WHERE CourseNum = (?)");  
 pStmt.setString(1, semesterComboBox.getSelectedItem().toString());  
 pStmt.setString(2, yearComboBox.getSelectedItem().toString());  
 pStmt.setInt(3, Integer.parseInt(courseNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "Only Semester and Year Were Changed!");  
 }  
 else if (subjectTextField.getText() != null && weeklyHoursTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Courses " +  
 "SET Subject = (?), " +  
 "Semester = (?), " +  
 "Year = (?) " +  
 "WHERE CourseNum = (?)");  
 pStmt.setString(1, subjectTextField.getText());  
 pStmt.setString(2, semesterComboBox.getSelectedItem().toString());  
 pStmt.setString(3, yearComboBox.getSelectedItem().toString());  
 pStmt.setInt(4, Integer.parseInt(courseNumTextField.getText()));  
 JOptionPane.showMessageDialog(null, "Course #" + courseNumTextField.getText() + ": Subject, Semester and Year Were Modified!");  
 }  
 else if (subjectTextField.getText().equals("") && weeklyHoursTextField.getText() != null) {  
 pStmt = connection.prepareStatement("UPDATE Courses " +  
 "SET Semester = (?), " +  
 "Year = (?), " +  
 "Weekly\_Hours = (?) " +  
 "WHERE CourseNum = (?)");  
 pStmt.setString(1, semesterComboBox.getSelectedItem().toString());  
 pStmt.setString(2, yearComboBox.getSelectedItem().toString());  
 pStmt.setString(3, weeklyHoursTextField.getText());  
 pStmt.setString(4, courseNumTextField.getText());  
 JOptionPane.showMessageDialog(null, "Course #" + courseNumTextField.getText() + ": Semester, Year and Weekly Hours Were Modified!");  
 }  
 else {  
 pStmt = connection.prepareStatement("UPDATE Courses " +  
 "SET Subject = (?), " +  
 "Semester = (?), " +  
 "Year = (?), " +  
 "Weekly\_Hours = (?) " +  
 "WHERE CourseNum = (?)");  
 pStmt.setString(1, subjectTextField.getText());  
 pStmt.setString(2, semesterComboBox.getSelectedItem().toString());  
 pStmt.setString(3, yearComboBox.getSelectedItem().toString());  
 pStmt.setString(4, weeklyHoursTextField.getText());  
 pStmt.setString(5, courseNumTextField.getText());  
 JOptionPane.showMessageDialog(null, "Course #" + courseNumTextField.getText() + ": Subject, Semester, Year and Weekly Hours Were Modified!");  
 }  
 }  
 }  
 pStmt.executeUpdate();  
 connection.commit();  
 resultQuery = statement.executeQuery("SELECT \* FROM Courses");  
 coursesTable.setModel(createNewTable(resultQuery));  
 } catch (SQLException ex) {  
 try {  
 if (connection != null)  
 connection.rollback();  
 } catch (SQLException ex2) {  
 ex2.printStackTrace();  
 }  
 JOptionPane.showMessageDialog(null, ex.getMessage());  
 ex.printStackTrace();  
 } catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 public void editLecturers() {  
 PreparedStatement pStmt = null;  
 PreparedStatement pStmtPhone = null;  
 connection = null;  
 statement = null;  
 try{  
 String name = (firstNameTextField.getText() + " " + lastNameTextField.getText());  
 int age = (2016 - Integer.parseInt((String)yearComboBox2.getSelectedItem()));  
 String bDate = dayComboBox.getSelectedItem().toString() + "/" + monthComboBox.getSelectedItem().toString() + "/" + yearComboBox2.getSelectedItem().toString();  
 connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/college","root","Ktzv3404");  
 connection.setAutoCommit(false);  
 statement = connection.createStatement();  
 if (lecturersComboBox.getSelectedItem() == "Create") {  
 if (idTextField.getText().equals("") || phoneTextField.getText().equals("") || addressTextField.getText().equals("")) {  
 if (firstNameTextField.getText().equals("") && lastNameTextField.getText().equals("") && addressTextField.getText().equals("") && idTextField.getText() != null && phoneTextField.getText() != null) {  
 System.out.println("test2");  
 pStmtPhone = connection.prepareStatement("INSERT INTO Phones " +  
 "VALUES (?, ?)");  
 pStmtPhone.setInt(1, Integer.parseInt(phoneTextField.getText()));  
 pStmtPhone.setInt(2, Integer.parseInt(idTextField.getText()));  
 }  
 else  
 JOptionPane.showMessageDialog(null, "Missing Required Fields! No Entry Was Created!");  
 }  
 else {  
 System.out.println("test1");  
 pStmt = connection.prepareStatement("INSERT INTO Lecturers " +  
 "VALUES (?, ?, ?, ?, ?)");  
 pStmt.setInt(1, Integer.parseInt(idTextField.getText()));  
 pStmt.setString(2, name);  
 pStmt.setInt(3, age);  
 pStmt.setString(4, addressTextField.getText());  
 pStmt.setString(5, bDate);  
 pStmtPhone = connection.prepareStatement("INSERT INTO Phones " +  
 "VALUES (?, ?)");  
 pStmtPhone.setInt(1, Integer.parseInt(phoneTextField.getText()));  
 pStmtPhone.setInt(2, Integer.parseInt(idTextField.getText()));  
 }  
 }  
 else if (lecturersComboBox.getSelectedItem() == "Delete") {  
 if (idTextField.getText().equals(""))  
 JOptionPane.showMessageDialog(null, "Missing an ID Primary Key");  
 else {  
 pStmt = connection.prepareStatement("DELETE FROM Lecturers " +  
 "WHERE ID = (?)");  
 pStmt.setInt(1, Integer.parseInt(idTextField.getText()));  
 }  
 }  
 else if (lecturersComboBox.getSelectedItem() == "Edit") {  
 if (idTextField.getText().equals("")) {  
 JOptionPane.showMessageDialog(null, "Missing an ID Primary Key");  
 }  
 else {  
 if (firstNameTextField.getText().equals("") && lastNameTextField.getText().equals("") && phoneTextField.getText().equals("") && addressTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "SET Age = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setInt(1, age);  
 pStmt.setString(2, bDate);  
 pStmt.setInt(3, Integer.parseInt(idTextField.getText()));  
 }  
 else if ((firstNameTextField.getText() != null || lastNameTextField.getText() != null) && phoneTextField.getText().equals("") && addressTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "Set Name = (?), " +  
 "Age = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setString(1, name);  
 pStmt.setInt(2, age);  
 pStmt.setString(3, bDate);  
 pStmt.setInt(4, Integer.parseInt(idTextField.getText()));  
 }  
 else if ((firstNameTextField.getText().equals("") && lastNameTextField.getText().equals("")) && phoneTextField.getText() != null && addressTextField.getText().equals("")) {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "SET Age = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setInt(1, age);  
 pStmt.setString(2, bDate);  
 pStmt.setInt(3, Integer.parseInt(idTextField.getText()));  
 System.out.println("got in here");  
 pStmtPhone = connection.prepareStatement("UPDATE Phones " +  
 "SET PhoneNum = (?) " +  
 "WHERE ID = (?)" +  
 "LIMIT 1");  
 pStmtPhone.setInt(1, Integer.parseInt(phoneTextField.getText()));  
 pStmtPhone.setInt(2, Integer.parseInt(idTextField.getText()));  
 System.out.println("and here too");  
 }  
 else if ((firstNameTextField.getText().equals("") && lastNameTextField.getText().equals("")) && phoneTextField.getText().equals("") && addressTextField.getText() != null) {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "SET Age = (?), " +  
 "Address = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setInt(1, age);  
 pStmt.setString(2, addressTextField.getText());  
 pStmt.setString(3, bDate);  
 pStmt.setInt(4, Integer.parseInt(idTextField.getText()));  
 }  
 else if ((firstNameTextField.getText().equals("") && lastNameTextField.getText().equals("")) && phoneTextField.getText() != null && addressTextField.getText() != null) {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "SET Age = (?), " +  
 "Address = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setInt(1, age);  
 pStmt.setString(2, addressTextField.getText());  
 pStmt.setString(3, bDate);  
 pStmt.setInt(4, Integer.parseInt(idTextField.getText()));  
 pStmtPhone = connection.prepareStatement("UPDATE Phones " +  
 "SET PhoneNum = (?) " +  
 "WHERE ID = (?) " +  
 "LIMIT 1");  
 pStmtPhone.setInt(1, Integer.parseInt(phoneTextField.getText()));  
 pStmtPhone.setInt(2, Integer.parseInt(idTextField.getText()));  
 }  
 else {  
 pStmt = connection.prepareStatement("UPDATE Lecturers " +  
 "SET Name = (?), " +  
 "Age = (?), " +  
 "Address = (?), " +  
 "BirthDate = (?) " +  
 "WHERE ID = (?)");  
 pStmt.setString(1, name);  
 pStmt.setInt(2, age);  
 pStmt.setString(3, addressTextField.getText());  
 pStmt.setString(4, bDate);  
 pStmt.setInt(5, Integer.parseInt(idTextField.getText()));  
 pStmtPhone = connection.prepareStatement("UPDATE Phones " +  
 "SET PhoneNum = (?) " +  
 "WHERE ID = (?) " +  
 "LIMIT 1");  
 pStmtPhone.setInt(1, Integer.parseInt(phoneTextField.getText()));  
 pStmtPhone.setInt(2, Integer.parseInt(idTextField.getText()));  
 }  
 }  
 }  
 if (pStmt != null)  
 pStmt.executeUpdate();  
 if (pStmtPhone != null)  
 pStmtPhone.executeUpdate();  
 connection.commit();  
 resultQuery = statement.executeQuery("SELECT \* FROM Lecturers ");  
 lecturersTable.setModel(createNewTable(resultQuery));  
 resultQuery = statement.executeQuery("SELECT \* FROM Phones ");  
 phonesTable.setModel(createNewTable(resultQuery));  
 }catch(SQLException ex){  
 try {  
 if (connection != null)  
 connection.rollback();  
 } catch (SQLException ex2) {  
 ex2.printStackTrace();  
 }  
 JOptionPane.showMessageDialog(null, ex.getMessage());  
 ex.printStackTrace();  
 }catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
  
 public void runSelectedQuery() {  
 connection = null;  
 statement = null;  
 try {  
 connection = DriverManager.getConnection(DB\_URL, UserName, Password);  
 statement = connection.createStatement();  
 if (schedulerRadioButton.isSelected()) {  
 resultQuery = statement.executeQuery("SELECT \* FROM Scheduler");  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 else if (timeRangeQueryRadioButton.isSelected()) {  
 if (daysFromComboBox.getSelectedIndex() != -1 && daysToComboBox.getSelectedIndex() != -1 && hoursFromComboBox.getSelectedIndex() != -1 && hoursToComboBox.getSelectedIndex() != -1) {  
 if (daysFromComboBox.getSelectedIndex() > daysToComboBox.getSelectedIndex()) {  
 JOptionPane.showMessageDialog(null, "Days or hours are not accurate!");  
 }  
 else {  
// resultQuery = statement.executeQuery("SELECT ID as LecturerID, ClassNum, CourseNum, b.Day, Hour " +  
// "FROM Scheduler AS b " +  
// "INNER JOIN WeekDays AS a " +  
// "ON a.Day = b.Day " +  
// "WHERE DayNum BETWEEN "+ (daysFromComboBox.getSelectedIndex()+1) +" AND " + (daysToComboBox.getSelectedIndex()+1) +  
// " AND b.Hour IN (SELECT b.Hour FROM Scheduler WHERE b.Hour BETWEEN '"+ hoursFromComboBox.getSelectedItem().toString() +"' AND '"+hoursToComboBox.getSelectedItem().toString()+"') " +  
// "ORDER BY b.Day ASC, b.Hour ASC");  
  
 resultQuery = statement.executeQuery("SELECT \* FROM Scheduler " +  
 "WHERE Day IN (SELECT Day FROM WeekDays WHERE dayNum = " + (daysFromComboBox.getSelectedIndex()+1) + " and Hour >= '" + hoursFromComboBox.getSelectedItem().toString() + "') " +  
 "OR Day IN (SELECT Day FROM WeekDays WHERE dayNum BETWEEN " + daysFromComboBox.getSelectedIndex()+2 + " And " + daysToComboBox.getSelectedIndex() + ") " +  
 "OR Day IN (SELECT Day FROM WeekDays WHERE dayNum = " + (daysToComboBox.getSelectedIndex()+1) + " and Hour <= '" + hoursToComboBox.getSelectedItem().toString() + "') " +  
 "ORDER BY Day ASC,Hour ASC;");  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 }  
 else {  
 JOptionPane.showMessageDialog(null, "One of the required fields is missing!");  
 }  
 }  
 else if (classesQueryRadioButton.isSelected()) {  
 if (classesQueryTextField.getText().equals("Enter Class Number")) {  
 resultQuery = statement.executeQuery("SELECT ClassNum, s.CourseNum, Subject, s.ID AS LecturerID, l.Name " +  
 "FROM Scheduler AS s " +  
 "INNER JOIN Courses AS c " +  
 "INNER JOIN Lecturers AS l " +  
 "ON s.CourseNum = c.CourseNum AND s.ID = l.ID");  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 else {  
 resultQuery = statement.executeQuery("SELECT ClassNum, s.CourseNum, Subject, s.ID AS LecturerID, l.name " +  
 "From Scheduler AS s " +  
 "INNER JOIN Courses AS c " +  
 "INNER JOIN Lecturers AS l " +  
 "ON s.CourseNum = c.CourseNum AND s.ID = l.ID " +  
 "WHERE ClassNum = " + classesQueryTextField.getText());  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 }  
 else if (lecturersQueryRadioButton.isSelected()) {  
 if (lecturersQueryTextField.getText().equals("Enter Lecturer's Name")) {  
 resultQuery = statement.executeQuery("SELECT l.Name, l.ID, ClassNum, s.CourseNum, Subject, Day, Hour " +  
 "From Scheduler AS s " +  
 "INNER JOIN Lecturers AS l " +  
 "INNER join Courses AS c " +  
 "ON s.ID = l.ID AND c.CourseNum = s.CourseNum");  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 else {  
 resultQuery = statement.executeQuery("SELECT l.Name, l.ID, ClassNum, s.CourseNum, Subject, Day, Hour " +  
 "FROM Scheduler AS s " +  
 "INNER JOIN Lecturers AS l " +  
 "INNER join Courses AS c " +  
 "ON s.ID = l.ID AND c.CourseNum = s.CourseNum " +  
 "WHERE Name LIKE '%" + lecturersQueryTextField.getText() + "%';");  
 schedulerTable.setModel(createNewTable(resultQuery));  
 }  
 schedulerTable.getColumnModel().getColumn(0).setPreferredWidth(115);  
 schedulerTable.getColumnModel().getColumn(1).setPreferredWidth(100);  
 schedulerTable.getColumnModel().getColumn(4).setPreferredWidth(140);  
 schedulerTable.getColumnModel().getColumn(3).setPreferredWidth(90);  
 schedulerTable.getColumnModel().getColumn(6).setPreferredWidth(60);  
 }  
 }catch(SQLException ex){  
 JOptionPane.showMessageDialog(null, ex.getMessage());  
 ex.printStackTrace();  
 }catch (Exception e) {  
 System.out.println("Exception");  
 e.printStackTrace();  
 }  
 }  
}