Joseph Kim

ITMD 411

11DEC22

Java Project Files

**Login.java**

**package** javaapplication1;

**import** java.awt.GridLayout; //useful for layouts

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

//controls-label text fields, button

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPasswordField;

**import** javax.swing.JTextField;

@SuppressWarnings("serial")

**public** **class** Login **extends** JFrame {

Dao conn;

**public** Login() {

**super**("IIT HELP DESK LOGIN");

conn = **new** Dao();

conn.createTables();

setSize(400, 210);

setLayout(**new** GridLayout(4, 2));

setLocationRelativeTo(**null**); // centers window

// SET UP CONTROLS

JLabel lblUsername = **new** JLabel("Username", JLabel.***LEFT***);

JLabel lblPassword = **new** JLabel("Password", JLabel.***LEFT***);

JLabel lblStatus = **new** JLabel(" ", JLabel.***CENTER***);

// JLabel lblSpacer = new JLabel(" ", JLabel.CENTER);

//Login cred text

JTextField txtUname = **new** JTextField(10);

JPasswordField txtPassword = **new** JPasswordField();

//Login and Exit button

JButton btn = **new** JButton("Login");

JButton btnExit = **new** JButton("Exit");

// constraints

lblStatus.setToolTipText("Contact help desk to unlock password");

lblUsername.setHorizontalAlignment(JLabel.***CENTER***);

lblPassword.setHorizontalAlignment(JLabel.***CENTER***);

// ADD OBJECTS TO FRAME

add(lblUsername); // 1st row filler

add(txtUname);

add(lblPassword); // 2nd row

add(txtPassword);

add(btn); // 3rd row

add(btnExit);

add(lblStatus); // 4th row

btn.addActionListener(**new** ActionListener() {

**int** count = 0; // count agent

@Override

**public** **void** actionPerformed(ActionEvent e) {

**boolean** adminStatus = **false**;

**int** admin = 0;

count = count + 1;

// verify credentials of user (MAKE SURE TO CHANGE TO YOUR TABLE NAME BELOW)

String query = "SELECT \* FROM jkim\_users WHERE uname = ? and upass = ?;";

**try** (PreparedStatement stmt = conn.getConnection().prepareStatement(query)) {

stmt.setString(1, txtUname.getText());

stmt.setString(2, txtPassword.~~getText~~());

ResultSet rs = stmt.executeQuery();

**if** (rs.next()) {

**int** isAdmin = rs.getInt("admin"); // get table column value

**if** (isAdmin == 1) {

adminStatus = **true**;

System.***out***.println("Admin Access Approved");

}

**else** {

System.***out***.println("User Access Approved");

}

**new** Tickets(adminStatus); //open Tickets file / GUI interface

setVisible(**false**); // HIDE THE FRAME

dispose(); // CLOSE OUT THE WINDOW

} **else**

lblStatus.setText("Try again! " + (3 - count) + " / 3 attempt(s) left");

} **catch** (SQLException ex) {

ex.printStackTrace();

}

}

});

btnExit.addActionListener(e -> System.*exit*(0));

setVisible(**true**); // SHOW THE FRAME

}

**public** **static** **void** main(String[] args) {

**new** Login();

}

}

**Tickets.java**

**package** javaapplication1;

**import** java.awt.Color;

**import** java.awt.Component;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.WindowAdapter;

**import** java.awt.event.WindowEvent;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** javax.swing.JFrame;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

@SuppressWarnings("serial")

**public** **class** Tickets **extends** JFrame **implements** ActionListener {

// class level member objects

Dao dao = **new** Dao(); // for CRUD operations

Boolean chkIfAdmin = **null**;

// Main menu object items

**private** JMenu mnuFile = **new** JMenu("File");

**private** JMenu mnuAdmin = **new** JMenu("Admin");

**private** JMenu mnuTickets = **new** JMenu("Tickets");

// Sub menu item objects for all Main menu item objects

JMenuItem mnuItemExit;

JMenuItem mnuItemRefresh;

JMenuItem mnuItemUpdate;

JMenuItem mnuItemDelete;

JMenuItem mnuItemOpenTicket;

JMenuItem mnuItemViewTicket;

JMenuItem mnuItemSelectTicket;

**private** JTable jt;

**public** Tickets(Boolean isAdmin) {

**if** (isAdmin != chkIfAdmin) {

System.***out***.println("Admin Approved");

}

chkIfAdmin = isAdmin;

createMenu();

prepareGUI();

}

**private** **void** createMenu() {

/\* Initialize sub menu items \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// initialize sub menu item for File main menu

mnuItemExit = **new** JMenuItem("Exit");

// add to File main menu item

mnuFile.add(mnuItemExit);

// initialize first sub menu item for Tickets main menu

mnuItemOpenTicket = **new** JMenuItem("Open Ticket");

// add to Ticket Main menu item

mnuTickets.add(mnuItemOpenTicket);

// initialize second sub menu item for Tickets main menu

mnuItemViewTicket = **new** JMenuItem("View All Ticket");

// add to Ticket Main menu item

mnuTickets.add(mnuItemViewTicket);

// initialize any more desired sub menu items below

// initialize sub menu item for File main menu

mnuItemRefresh = **new** JMenuItem("Refresh");

// add to File main menu item

mnuFile.add(mnuItemRefresh);

//only show this tab if the program is accessed through admin credential

**if** (chkIfAdmin == **true**) {

// initialize first sub menu items for Admin main menu

mnuItemUpdate = **new** JMenuItem("Update Ticket");

// add to Admin main menu item

mnuAdmin.add(mnuItemUpdate);

// initialize second sub menu items for Admin main menu

mnuItemDelete = **new** JMenuItem("Delete Ticket");

// add to Admin main menu item

mnuAdmin.add(mnuItemDelete);

}

// initialize second sub menu item for Tickets main menu

mnuItemSelectTicket = **new** JMenuItem("Select Ticket");

// add to Ticket Main menu item

mnuTickets.add(mnuItemSelectTicket);

/\* Add action listeners for each desired menu item \*\*\*\*\*\*\*\*\*\*\*\*\*/

mnuItemExit.addActionListener(**this**);

mnuItemOpenTicket.addActionListener(**this**);

mnuItemViewTicket.addActionListener(**this**);

/\*

\* continue implementing any other desired sub menu items (like

\* for update and delete sub menus for example) with similar

\* syntax & logic as shown above

\*/

mnuItemRefresh.addActionListener(**this**);

**if** (chkIfAdmin == **true**) {

mnuItemUpdate.addActionListener(**this**);

mnuItemDelete.addActionListener(**this**);

}

mnuItemSelectTicket.addActionListener(**this**);

}

**private** **void** prepareGUI() {

// create JMenu bar

JMenuBar bar = **new** JMenuBar();

bar.add(mnuFile); // add main menu items in order, to JMenuBar

**if** (chkIfAdmin == **true**) { //only show this to admin

bar.add(mnuAdmin);

}

bar.add(mnuTickets);

// add menu bar components to frame

setJMenuBar(bar);

addWindowListener(**new** WindowAdapter() {

// define a window close operation

**public** **void** windowClosing(WindowEvent wE) {

System.*exit*(0);

}

});

// set frame options

setSize(400, 400);

getContentPane().setBackground(Color.***LIGHT\_GRAY***);

setLocationRelativeTo(**null**);

**try** {

jt = **new** JTable(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.setBounds(30, 40, 200, 400);

JScrollPane sp = **new** JScrollPane(jt);

add(sp);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

setVisible(**true**);

}

@Override

**public** **void** actionPerformed(ActionEvent e) {

// implement actions for sub menu items

//if Exit btn is pressed then the system will exit

**if** (e.getSource() == mnuItemExit) {

System.*exit*(0);

}

//if Open Ticket btn is pressed the system will add new ticket to the system

**else** **if** (e.getSource() == mnuItemOpenTicket) {

openTicket();

}

//If View Ticket is pressed the system will display existing tickets

**else** **if** (e.getSource() == mnuItemViewTicket) {

viewAll();

}

**else** **if** (e.getSource() == mnuItemRefresh) {

viewAll();

}

**else** **if** (e.getSource() == mnuItemSelectTicket) {

selectTicket();

}

**else** **if** (e.getSource() == mnuItemUpdate) {

updateTicket();

}

**else** **if** (e.getSource() == mnuItemDelete) {

deleteTicket();

}

}

**private** **void** openTicket() {

// get ticket information

String ticketName = JOptionPane.*showInputDialog*(**null**, "Enter your name");

String ticketDesc = JOptionPane.*showInputDialog*(**null**, "Enter a ticket description");

// insert ticket information to database

**if**(ticketName == **null** || (ticketName != **null** && ("".equals(ticketName))) || ticketDesc == **null** || (ticketDesc != **null** && ("".equals(ticketDesc))))

{

JOptionPane.*showMessageDialog*(**null**, "Ticket creation failed: empty name / description.");

System.***out***.println("Ticket creation failed: empty name / description.");

} **else** {

**int** id = dao.insertRecords(ticketName, ticketDesc);

// display results if successful or not to console / dialog box

**if** (id != 0) {

System.***out***.println("Ticket ID : " + id + " created successfully!!!");

JOptionPane.*showMessageDialog*(**null**, "Ticket id: " + id + " created");

} **else** {

System.***out***.println("Ticket cannot be created!!!");

}

}

}

**private** **void** deleteTicket() {

String deleteTicketId = JOptionPane.*showInputDialog*(**null**, "Enter the ticket id to delete");

**if**(deleteTicketId == **null** || (deleteTicketId != **null** && ("".equals(deleteTicketId)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket view failed: empty id.");

}

**else** {

**int** tid = Integer.*parseInt*(deleteTicketId);

**int** reply = JOptionPane.*showConfirmDialog*(**null**, "Are you sure you want to delete ticket " + tid + "?", "Warning!", JOptionPane.***YES\_NO\_OPTION***);

**if** (reply == JOptionPane.***YES\_OPTION***) {

dao.deleteRecords(tid);

}

**else** {

JOptionPane.*showMessageDialog*(**null**, "Ticket " + tid + " was not deleted.");

}

}

}

**private** **void** updateTicket() {

String updateTicketID = JOptionPane.*showInputDialog*(**null**, "Please enter id of the ticket to update");

String ticketDesc = JOptionPane.*showInputDialog*(**null**, "Append to the ticket description");

String updateTicketStatus = JOptionPane.*showInputDialog*(**null**, "Update the ticket status");

**if**(updateTicketID == **null** || ticketDesc == **null** || updateTicketStatus == **null** || (updateTicketID != **null** && ("".equals(updateTicketID))) || (ticketDesc != **null** && ("".equals(ticketDesc))) || (updateTicketStatus != **null** && ("".equals(updateTicketStatus)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket view failed: empty id.");

}

**else** {

**int** tid = Integer.*parseInt*(updateTicketID);

dao.updateRecords(tid, ticketDesc, updateTicketStatus);

}

}

**private** **void** viewAll() {

// retrieve all tickets details for viewing in JTable

**try** {

// Use JTable built in functionality to build a table model and

// display the table model off your result set!!!

jt.setModel(ticketsJTable.*buildTableModel*(dao.readRecords()));

jt.revalidate();

jt.repaint();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

}

//If Select Ticket has been pressed, the system will show a dialog asking user to input specific ticket id and display the information

**private** **void** selectTicket() {

String ticketId = JOptionPane.*showInputDialog*(**null**, "Enter the ticket ID");

**if**(ticketId == **null** || (ticketId != **null** && ("".equals(ticketId)))) {

JOptionPane.*showMessageDialog*(**null**, "Ticket view failed: empty id.");

}

**else** {

**int** tid = Integer.*parseInt*(ticketId);

**try** {

jt.setModel(ticketsJTable.*buildTableModel*(dao.selectRecord(tid)));

jt.revalidate();

jt.repaint();

}

**catch** (SQLException e1) {

e1.printStackTrace();

}

}

}

}

**Dao.java**

**package** javaapplication1;

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileReader;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.time.LocalDateTime;

**import** java.time.format.DateTimeFormatter;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**public** **class** Dao {

// instance fields

**static** Connection *connect* = **null**;

Statement statement = **null**;

// constructor

**public** Dao() {

}

**public** Connection getConnection() {

// Setup the connection with the DB

**try** {

*connect* = DriverManager

.*getConnection*("jdbc:mysql://www.papademas.net:3307/tickets?autoReconnect=true&useSSL=false"

+ "&user=fp411&password=411");

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** *connect*;

}

// CRUD implementation

**public** **void** createTables() {

// variables for SQL Query table creations

**final** String createTicketsTable = "CREATE TABLE jkim\_tickets1(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_issuer VARCHAR(30), ticket\_description VARCHAR(200), open\_time VARCHAR(30), status VARCHAR(15))";

**final** String createUsersTable = "CREATE TABLE jkim\_users(uid INT AUTO\_INCREMENT PRIMARY KEY, uname VARCHAR(30), upass VARCHAR(30), admin int)";

**try** {

// execute queries to create tables

statement = getConnection().createStatement();

statement.executeUpdate(createTicketsTable);

statement.executeUpdate(createUsersTable);

System.***out***.println("Created tables in given database...");

// end create table

// close connection/statement object

statement.close();

*connect*.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

// add users to user table

addUsers();

}

**public** **void** addUsers() {

// add list of users from userlist.csv file to users table

// variables for SQL Query inserts

String sql;

Statement statement;

BufferedReader br;

List<List<String>> array = **new** ArrayList<>(); // list to hold (rows & cols)

// read data from file

**try** {

br = **new** BufferedReader(**new** FileReader(**new** File("./userlist.csv")));

String line;

**while** ((line = br.readLine()) != **null**) {

array.add(Arrays.*asList*(line.split(",")));

}

} **catch** (Exception e) {

System.***out***.println("There was a problem loading the file");

}

**try** {

// Setup the connection with the DB

statement = getConnection().createStatement();

// create loop to grab each array index containing a list of values

// and PASS (insert) that data into your User table

**for** (List<String> rowData : array) {

sql = "insert into jkim\_users(uname,upass,admin) " + "values('" + rowData.get(0) + "'," + " '"

+ rowData.get(1) + "','" + rowData.get(2) + "');";

statement.executeUpdate(sql);

}

System.***out***.println("Inserts completed in the given database...");

// close statement object

statement.close();

} **catch** (Exception e) {

System.***out***.println(e.getMessage());

}

}

**public** **int** insertRecords(String ticketName, String ticketDesc) {

**int** id = 0;

**try** {

**final** LocalDateTime myDateObj = LocalDateTime.*now*();

**final** DateTimeFormatter myFormatObj = DateTimeFormatter.*ofPattern*("dd-MM-yyyy HH:mm:ss");

**final** String formattedDate = myDateObj.format(myFormatObj);

**final** String status = "OPEN";

statement = getConnection().createStatement();

statement.executeUpdate("Insert into jkim\_tickets1" + "(ticket\_issuer, ticket\_description, open\_time, status) values(" + " '"

+ ticketName + "','" + ticketDesc + "','" + formattedDate + "','" + status + "')", Statement.***RETURN\_GENERATED\_KEYS***);

// retrieve ticket id number newly auto generated upon record insertion

ResultSet resultSet = **null**;

resultSet = statement.getGeneratedKeys();

**if** (resultSet.next()) {

// retrieve first field in table

id = resultSet.getInt(1);

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** id;

}

**public** ResultSet readRecords() {

ResultSet results = **null**;

**try** {

statement = *connect*.createStatement();

results = statement.executeQuery("SELECT \* FROM jkim\_tickets1");

//connect.close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

// continue coding for updateRecords implementation

**public** **void** updateRecords(**int** tid, String ticketDesc, String updateTicketStatus) {

**try** {

// statement = connect.createStatement();

statement = getConnection().createStatement();

//String command = "UPDATE `jkim\_tickets1` SET `ticket\_id`=[value-1],`ticket\_issuer`=[value-2],`ticket\_description`=[value-3] WHERE 1)", Statement.RETURN\_GENERATED\_KEYS;

String command = "Update jkim\_tickets1 set ticket\_description = '" + ticketDesc + "', status = '" + updateTicketStatus + "' where ticket\_id = " + tid;

statement.executeUpdate(command);

//connect.close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

}

// continue coding for deleteRecords implementation

**public** **void** deleteRecords(**int** tid) {

**try** {

statement = *connect*.createStatement();

String command = "DELETE FROM jkim\_tickets1 WHERE ticket\_id = " + tid;

statement.executeUpdate(command);

//connect.close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

}

**public** ResultSet selectRecord(**int** tid) {

ResultSet results = **null**;

**try** {

statement = *connect*.createStatement();

results = statement.executeQuery("SELECT \* FROM jkim\_tickets1 WHERE ticket\_id = " + tid);

//connect.close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

}

**ticketsJTable.java**

**package** javaapplication1;

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.SQLException;

**import** java.util.Vector;

**import** javax.swing.table.DefaultTableModel;

**public** **class** ticketsJTable {

@SuppressWarnings("unused")

**private** **final** DefaultTableModel tableModel = **new** DefaultTableModel();

**public** **static** DefaultTableModel buildTableModel(ResultSet rs) **throws** SQLException {

ResultSetMetaData metaData = rs.getMetaData();

// names of columns

Vector<String> columnNames = **new** Vector<String>();

**int** columnCount = metaData.getColumnCount();

**for** (**int** column = 1; column <= columnCount; column++) {

columnNames.add(metaData.getColumnName(column));

}

// data of the table

Vector<Vector<Object>> data = **new** Vector<Vector<Object>>();

**while** (rs.next()) {

Vector<Object> vector = **new** Vector<Object>();

**for** (**int** columnIndex = 1; columnIndex <= columnCount; columnIndex++) {

vector.add(rs.getObject(columnIndex));

}

data.add(vector);

}

// return data/col.names for JTable

**return** **new** DefaultTableModel(data, columnNames);

}

}