**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.LocalDate;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import javax.swing.JOptionPane;

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 ttran\_tickets(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_issuer VARCHAR(30), ticket\_description VARCHAR(200), start\_date DATE, status VARCHAR(10))";

final String createUsersTable = "CREATE TABLE ttran\_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 ttran\_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, LocalDate startDate) {

int id = 0;

try {

statement = getConnection().createStatement();

statement.executeUpdate("Insert into ttran\_tickets" + "(ticket\_issuer, ticket\_description, start\_date) values(" + " '"

+ ticketName + "','" + ticketDesc +"','" + startDate + "')", 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 ttran\_tickets");

//connect.close();

} catch (SQLException e1) {

e1.printStackTrace();

}

return results;

}

// continue coding for updateRecords implementation

public void updateRecords(int ticketId) {

try {

String status = null;

String newDesc = null;

int response1 = JOptionPane.showConfirmDialog(null, "Do you want to update the ticket description for ticket ID: " + ticketId + "?", "Confirmation" , JOptionPane.YES\_NO\_OPTION);

int response2 = JOptionPane.showConfirmDialog(null, "Want to close ticket ID: " + ticketId + "?", "Confirmation" , JOptionPane.YES\_NO\_OPTION);

if (response1 == JOptionPane.YES\_OPTION && response2 == JOptionPane.YES\_OPTION) {

newDesc = JOptionPane.showInputDialog(null, "What is your new description for ticket ID: " + ticketId + " to be?");

status = "CLOSED";

PreparedStatement update = connect.prepareStatement("UPDATE ttran\_tickets SET ticket\_description = ?, status = ? WHERE ticket\_id = ?");

update.setString(1,newDesc);

update.setString(2, status);

update.setLong(3,ticketId);

update.executeUpdate();

update.close();

System.out.println("Ticket ID : " + ticketId + " updated successfully!");

JOptionPane.showMessageDialog(null, "Ticket Id " +ticketId+ "was updated");

System.out.println("Ticket ID : " + ticketId + " was closed.");

JOptionPane.showMessageDialog(null, "Ticket Id " +ticketId+ "was closed");

}

else if (response1 == JOptionPane.YES\_OPTION && response2 == JOptionPane.NO\_OPTION) {

newDesc = JOptionPane.showInputDialog(null, "What is your new description for ticket ID: " + ticketId + " to be?");

PreparedStatement update = connect.prepareStatement("UPDATE ttran\_tickets SET ticket\_description = ? WHERE ticket\_id = ?");

update.setString(1,newDesc);

update.setLong(2,ticketId);

update.executeUpdate();

update.close();

}

else if (response1 == JOptionPane.NO\_OPTION && response2 == JOptionPane.YES\_OPTION) {

status = "CLOSED";

PreparedStatement update = connect.prepareStatement("UPDATE ttran\_tickets status = ? WHERE ticket\_id = ?");

update.setString(1,status);

update.setLong(2,ticketId);

update.executeUpdate();

update.close();

}

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public int deleteRecords(int ticketID) {

try {

// Execute delete query

PreparedStatement delete = connect.prepareStatement("DELETE FROM ttran\_tickets WHERE ticket\_id = ?");

delete.setLong(1,ticketID);

delete.executeUpdate();

delete.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return ticketID;

}

}

// continue coding for deleteRecords implementation

**Tickets.java**

package javaapplication1;

import java.awt.Color;

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 java.time.LocalDate;

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 mnuItemUpdate;

JMenuItem mnuItemDelete;

JMenuItem mnuItemOpenTicket;

JMenuItem mnuItemViewTicket;

JMenuItem mnuItemCloseTicket;

public Tickets(Boolean isAdmin) {

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 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 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 Ticket");

// add to Ticket Main menu item

mnuTickets.add(mnuItemViewTicket);

// initialize second sub menu item for Tickets main menu

mnuItemCloseTicket = new JMenuItem("Close Ticket");

// add to Ticket Main menu item

mnuTickets.add(mnuItemCloseTicket);

// initialize any more desired sub menu items below

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

mnuItemExit.addActionListener(this);

mnuItemUpdate.addActionListener(this);

mnuItemDelete.addActionListener(this);

mnuItemOpenTicket.addActionListener(this);

mnuItemViewTicket.addActionListener(this);

mnuItemCloseTicket.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\*

\*/

}

private void prepareGUI() {

// create JMenu bar

JMenuBar bar = new JMenuBar();

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

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);

setVisible(true);

}

@Override

public void actionPerformed(ActionEvent e) {

// implement actions for sub menu items

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

System.exit(0);

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

// get ticket information

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

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

LocalDate startDate = LocalDate.now();

// insert ticket information to database

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

// 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!!!");

}

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

// 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!!!

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

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

JScrollPane sp = new JScrollPane(jt);

add(sp);

setVisible(true); // refreshes or repaints frame on screen

} catch (SQLException e1) {

e1.printStackTrace();

}

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

String ticketId = JOptionPane.showInputDialog("Enter the ticket ID you want to update");

int id = Integer.parseInt(ticketId);

dao.updateRecords(id);

if (id == 0) {

JOptionPane.showMessageDialog(null, "You can't update a ticket that doesn't exist");

}

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

String ticketID = JOptionPane.showInputDialog(null, "Enter the ticket ID you want to delete");

int ticketID1 = Integer.parseInt(ticketID);

int response = JOptionPane.showConfirmDialog(null,

"Are you sure you want to delete the ticket with the id: " + ticketID1 + " Warning, you will not be able to revert changes" );

dao.deleteRecords(ticketID1);

if (response == JOptionPane.YES\_OPTION) {

System.out.println("Ticket ID : " + ticketID1 + " deleted successfully!");

} else {

JOptionPane.showMessageDialog(null, "Your actions were cancelled");

}

}

/\*

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

\* delete sub menus for example) with similar syntax & logic as shown above

\*/

}

}

**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);

}

}

**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);

JTextField txtUname = new JTextField(10);

JPasswordField txtPassword = new JPasswordField();

JButton btn = new JButton("Submit");

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 admin = false;

count = count + 1;

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

String query = "SELECT \* FROM ttran\_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()) {

admin = rs.getBoolean("admin"); // get table column value

new Tickets(admin);

setVisible(false); // HIDE THE FRAME

dispose(); // CLOSE OUT THE WINDOW

} else

lblStatus.setText("Try again! " + (3 - count) + " / 3 attempts 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();

}

}