**ITMD 411: Submission Document**

**Group Members:** [Yaqub Farook](mailto:yfarook@hawk.iit.edu) **and** [Griselda Pasillas](mailto:Gpasillas@hawk.iit.edu)

**Code for Every class:**

**Dao class**

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.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import javax.swing.JOptionPane;

import java.util.Date;

import java.time.format.DateTimeFormatter;

import java.time.LocalDate;

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. what they had

final String createTicketsTable = "CREATE TABLE farpas\_tickets(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_issuer VARCHAR(30), ticket\_description VARCHAR(200), start\_date VARCHAR(30))";

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

final String createReturnTable = "CREATE TABLE farpas\_returned(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, ticket\_issuer VARCHAR(30), ticket\_description VARCHAR(200), start\_date VARCHAR(30), end\_date VARCHAR(30))";

try {

// execute queries to create tables

statement = getConnection().createStatement();

statement.executeUpdate(createReturnTable);

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 farpas\_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 initialdate) {

int id = 0;

try {

statement = getConnection().createStatement();

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

+ ticketName + "','" + ticketDesc + "','" + initialdate +"')", 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;

}

//Admin View

public ResultSet readRecords() {

ResultSet results = null;

try {

statement = connect.createStatement();

results = statement.executeQuery("SELECT \* FROM farpas\_tickets");

System.out.println("Hello, Administrative user!");

//connect.close();

} catch (SQLException e) {

e.printStackTrace();

}

return results;

}

//\*\*\*\*

// continue coding for updateRecords implementation

// continue coding for deleteRecords implementation

//User View

// public ResultSet readUserRecords() {

// ResultSet results = null;

// try {

// statement = connect.createStatement();

// results = statement.executeQuery("SELECT \* FROM farpas\_tickets WHERE admin= '" + Login.adminid + "';");

// System.out.println("Hello!");

// //connect.close();

// } catch (SQLException e) {

// e.printStackTrace();

// }

// return results;

// }

// continue coding for deleteRecords implementation

//We were able to find a tutorial to help us at this link: https://www.geeksforgeeks.org/files-delete-method-in-java-with-examples/

public void deleteRecords(int id) throws SQLException {

// Execute delete query

System.out.println("Hold on... we are making a delete statement...");

statement = connect.createStatement();

String sql = "DELETE FROM farpas\_tickets " + "WHERE ticket\_id = '" + id + "'" ;

String sqlOne="DELETE FROM farpas\_returned " + "WHERE ticket\_id = '" + id + "'" ;

int answer = JOptionPane.showConfirmDialog(null, "Would you like us to remove your ticket number " + id + "?", "Please confirm", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if (answer == JOptionPane.NO\_OPTION) {

System.out.println("There have been no records deleted");

JOptionPane.showMessageDialog(null, "None of the tickets have been deleted.");

} else if (answer == JOptionPane.YES\_OPTION) {

statement.executeUpdate(sql);

statement.executeUpdate(sqlOne);

JOptionPane.showMessageDialog(null, "The ticket has now been deleted.");

System.out.println("File discarded");

} else if (answer == JOptionPane.CLOSED\_OPTION) {

JOptionPane.showMessageDialog(null, "The ticket has now been cancelled.");

System.out.println("Cancelling Request");

}

}

// continue coding for updateRecords implementation

public void updateRecords(int id, String ticketDesc) throws SQLException {

// Execute update query

System.out.println("Hold on... we are making an update statement...");

statement = connect.createStatement();

String sql = "UPDATE farpas\_tickets " + "SET ticket\_description = '" + ticketDesc + "' WHERE ticket\_id = '" + id +"'";

statement.executeUpdate(sql);

JOptionPane.showMessageDialog(null, "the following update is successful!.");

}

//close ticket

public void endingRecords(int id, LocalDate initialdate) throws SQLException {

// Execute close ticket query

System.out.println("Hold on... we're making the final ticket statement to close...");

statement = connect.createStatement();

String sql = "INSERT INTO farpas\_returned SELECT \* FROM farpas\_tickets WHERE ticket\_id = '" + id + "'";

String sqlOne="DELETE FROM farpas\_tickets " + "WHERE ticket\_id = '" + id + "'" ;

String sqlTwo = "UPDATE farpas\_returned " + "SET end\_date = '" + initialdate + "' WHERE ticket\_id = '" + id +"'";

int answer = JOptionPane.showConfirmDialog(null, "Would you like us to close your ticket number " + id + "?", "Please confirm before proceeding ahead", JOptionPane.YES\_NO\_OPTION, JOptionPane.QUESTION\_MESSAGE);

if (answer == JOptionPane.NO\_OPTION) {

System.out.println("There are no records closed");

JOptionPane.showMessageDialog(null, "No tickets have been closed yet.");

} else if (answer == JOptionPane.YES\_OPTION) {

statement.executeUpdate(sql);

statement.executeUpdate(sqlOne);

statement.executeUpdate(sqlTwo);

JOptionPane.showMessageDialog(null, "Closing the ticket");

System.out.println("The ticket has been closed successfully.");

} else if (answer == JOptionPane.CLOSED\_OPTION) {

JOptionPane.showMessageDialog(null, "Error: Oops! Unfortuantley your tickt didn't close.");

System.out.println("The request has now been cancelled...");

}

}

//Display returned tickets

public ResultSet readReturned() {

ResultSet results = null;

try {

statement = connect.createStatement();

results = statement.executeQuery("SELECT \* FROM farpas\_returned");

//connect.close();

} catch (SQLException e) {

e.printStackTrace();

}

return results;

}

}

**Login class**

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;

import java.util.Date;

import java.awt.Color;

import java.awt.Font;

import java.awt.Image;

import javax.swing.ImageIcon;

//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;

import javax.swing.JComboBox;

import javax.swing.JComponent;

import javax.swing.border.Border;

@SuppressWarnings("serial")

public class Login extends JFrame {

Dao conn;

boolean adminid;

public Login() {

super("IIT HELP DESK LOGIN");

conn = new Dao();

conn.createTables();

setSize(400, 210);

setLayout(new GridLayout(4, 2));

//ADD

setLocationRelativeTo(null); // centers window.

// SET UP CONTROLS

JLabel lblRightHeading = new JLabel("Help Desk", JLabel.RIGHT);

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

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

JLabel lblUserType = new JLabel("Select Type", JLabel.LEFT);

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

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

lblRightHeading.setForeground(new Color(255, 255, 255));

lblUsername.setForeground(new Color(0, 102, 255));

lblPassword.setForeground(new Color(0, 102, 255));

lblStatus.setForeground(new Color(0, 102, 255));

lblUserType.setForeground(new Color(0, 102, 255));

lblRightHeading.setFont(new Font("Arial", Font.BOLD, 25));

lblUsername.setFont(new Font("Georgia", Font.BOLD, 20));

lblPassword.setFont(new Font("Georgia", Font.BOLD, 20));

lblUserType.setFont(new Font("Georgia", Font.BOLD, 20));

JTextField txtUname = new JTextField(10);

JPasswordField txtPassword = new JPasswordField();

JButton btn = new JButton("Submit");

JButton btnExit = new JButton("Exit");

btn.setFont(new Font("Geneva", Font.PLAIN, 19));

btnExit.setFont(new Font("Geneva", Font.PLAIN, 19));

btnExit.setBackground(new Color(102, 153, 153));

btnExit.setForeground(new Color(102, 153, 153));

btn.setBackground(new Color(255, 51, 51));

btn.setForeground(new Color(153, 204, 255));

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

lblRightHeading.setHorizontalAlignment(JLabel.CENTER);

lblUsername.setHorizontalAlignment(JLabel.CENTER);

lblPassword.setHorizontalAlignment(JLabel.CENTER);

lblUserType.setHorizontalAlignment(JLabel.CENTER);

//Dropdown list choices

String choices[]= { "Administrator", "Guest User" };

JComboBox select = new JComboBox(choices);

// ADD OBJECTS TO FRAME

add(lblRightHeading); // 1st row filler

add(lblUsername); // 2nd row

add(txtUname);

add(lblPassword);// 3rd row

add(txtPassword);

add(lblUserType); // 4th row

//add(txtType);

add(select);

add(btn); // 5th row

add(btnExit);

add(lblStatus); // 6th row

btn.addActionListener(new ActionListener() {

int count = 0; // count agent

@SuppressWarnings("deprecation") //might need to trouble shoot from this line down

@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 farpas\_users WHERE uname = ? and upass = ?;";

String uname;

String upass;

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

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

uname = txtUname.getText();//comeback to this

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

upass = txtPassword.getText(); //comeback 2 this

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

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

new Tickets(admin); //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();

}

//getting the user. verifying administrator creidentials

String queryOne = "SELECT admin FROM farpas\_users WHERE uname = ? and upass = ?;";

try (PreparedStatement stmt2 = conn.getConnection().prepareStatement(queryOne)) {

stmt2.setString(1,txtUname.getText()); // uname

stmt2.setString(2,txtPassword.getText());// upass);

ResultSet rs = stmt2.executeQuery();

if(rs.next()) {

adminid = rs.getBoolean("admin");

} 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) {

System.out.println("[ Ticketing system made by Yaqub Farook and Griselda Pasillas ]\n");

new Login();

}

}

**Tickets class**

package javaapplication1;

import java.awt.Color;

//import java.awt.Image; comeback to thius

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

//import java.sql.SQLException;

import java.sql.SQLException;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.util.Date;

//might delete lines 4 lines above

//import javax.swing.ImageIcon; comeback to this

import javax.swing.JFrame;

import javax.swing.JLabel;

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 mnuHaltTicket;

JMenuItem mnuItemReturned;

JMenuItem mnuItemOpenTicket;

JMenuItem mnuItemViewTicket;

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);

//-----------------------------------------------

// initialization for 3rd sub menu items for main menu of admin

mnuItemReturned = new JMenuItem("See Resolved Tickets");

// main menu item--adding to admin

mnuAdmin.add(mnuItemReturned);

// initialization for sub menu items for main menu of admin

mnuHaltTicket = new JMenuItem("Close Ticket");

// main menu item--adding to ticket

mnuAdmin.add(mnuHaltTicket);

// initialization for main menu of Tickets (1st sub menu item)

mnuItemOpenTicket = new JMenuItem("Open Ticket");

// main menu item--adding to ticket

mnuTickets.add(mnuItemOpenTicket);

//// initialization for main menu of Tickets (2nd sub menu item)

mnuItemViewTicket = new JMenuItem("View Ticket");

// main menu item--adding to ticket

mnuTickets.add(mnuItemViewTicket);

//---------------------------------------------------------

// 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);

mnuItemReturned.addActionListener(this);

mnuItemOpenTicket.addActionListener(this);

mnuItemViewTicket.addActionListener(this);

mnuHaltTicket.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); //we might need to keep change

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");

String initialdate = JOptionPane.showInputDialog("Create booking for the following date: ");

// insert ticket information to database

int id = dao.insertRecords(ticketName, ticketDesc, LocalDate.parse(initialdate));

// 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);

//getContentPane().setBackground(new Color(10, 44, 92)); this is what they had. Take it out

getContentPane().setBackground(Color.DARK\_GRAY);

JScrollPane sp = new JScrollPane(jt);

add(sp);

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

} catch (SQLException sqe) {

sqe.printStackTrace(); //we might've changed e to e in the Dao class

}

}

/\*

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

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

\*/

//----------------------------------------------------------------------------------------------

//when the user clicks on the delete button

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

try {

String idval = JOptionPane.showInputDialog(null, "Out of all the ticket\_id 's which one do you want to delete?");

int id = Integer.parseInt(idval);

dao.deleteRecords(id);

} catch (SQLException sqe) {

sqe.printStackTrace();

}

}

//when the user clicks on the closing ticket button

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

try {

String idval = JOptionPane.showInputDialog(null, "Out of all the ticket\_id 's which one do you want to close?");

int id = Integer.parseInt(idval);

LocalDate initialdate = null;

dao.endingRecords(id, initialdate);//come back to this at the end

} catch (SQLException sqe) {

sqe.printStackTrace();

}

}

//When the user clicks on the updates button

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

try {

String idval = JOptionPane.showInputDialog(null, "Out of all the ticket\_id 's which one do you want to update?");

int id = Integer.parseInt(idval);

String ticketDesc = JOptionPane.showInputDialog(null, "Changing the following ticket description to: ");

dao.updateRecords(id, ticketDesc);

//e

} catch (SQLException sqe) {

sqe.printStackTrace();

}

}

//When the user clicks on the resolved button

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

// collecting all of returned tickets so that the details can be viewed in the JTable

try {

// Using the JTable to make a table and display the model off the results

JTable jt = new JTable(ticketsJtable.buildTableModel(dao.readReturned()));

jt.setBounds(25, 35, 150, 350);

JScrollPane sp = new JScrollPane(jt);

add(sp);

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

} catch (SQLException sqe) {

sqe.printStackTrace();

}

}

}}

**ticketsJtable**

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);

}

}

**Returned table class**

package javaapplication1;

public class Returned {

}

**EXTRA CREDIT**

1. Github Link

<https://github.com/gpasillas01/ITMD411_FinalProject.git>

1. Outstanding GUI: As asked, we made a GUI that incorporated the following: fluid navigation, buttons , menus, table view actions, event handling. Attached below are screen shots:







