**Shahaan Mirza**

**Papademas**

**ITMD 411**

**Final Project Code**

GitHub Repository

[/shahaanmirza/ITMD411/](https://github.com/shahaanmirza/ITMD411/tree/main/FinalProject_smirza3)

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 smirza3\_users\_test 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); //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.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.awt.event.WindowAdapter;  
import java.awt.event.WindowEvent;  
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 mnuItemUpdate;  
 JMenuItem mnuItemDelete;  
 JMenuItem mnuItemOpenTicket;  
 JMenuItem mnuItemViewTicket;  
  
 //add an option for users to close their ticket if they fix their own issue  
 JMenuItem mnuItemCloseTicket;  
  
 public Tickets(Boolean isAdmin) {  
  
 chkIfAdmin = isAdmin; // checks if the user that logged in is admin (1) or not (0)  
 createMenu(); // runs the create menu method to add menu items  
 prepareGUI(); // creates a gui  
  
 }  
  
 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  
 mnuAdmin.add(mnuItemViewTicket);  
  
 // initialize any more desired sub menu items below  
 // ADDED:  
 mnuItemCloseTicket = new JMenuItem("Close Ticket");  
 //add to Ticket Main menu item  
 mnuTickets.add(mnuItemCloseTicket);  
  
  
 /\* Add action listeners for each desired menu item \*\*\*\*\*\*\*\*\*\*\*\*\*/  
 mnuItemExit.addActionListener(this);  
 mnuItemUpdate.addActionListener(this);  
 mnuItemDelete.addActionListener(this);  
 mnuItemOpenTicket.addActionListener(this);  
 mnuItemViewTicket.addActionListener(this);  
 //added:  
 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  
 if(chkIfAdmin == true) {  
 bar.add(mnuAdmin);  
 }  
 else {  
 //empty  
 }  
 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");  
  
 // insert ticket information to database  
  
 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!!!");  
 }  
  
 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() == mnuItemDelete) {  
  
 // delete a ticket  
 try {  
 // get id of ticket  
 String ticketID = JOptionPane.*showInputDialog*(null,  
 "Enter ticket ID to be deleted");  
  
 //check to see if user wants to continue with delete  
 int response = JOptionPane.*showConfirmDialog*(null,  
 "Do you want to continue with this action?",  
 "Delete This Ticket",  
 JOptionPane.*YES\_NO\_OPTION*);  
 if(response == JOptionPane.*YES\_OPTION*) {  
 // delete ticket associated with ID  
 dao.deleteRecords(ticketID);  
 JOptionPane.*showMessageDialog*(null,  
 "Ticket ID: " + ticketID + " was deleted.");  
 System.*out*.println("Record has been deleted");  
 }  
 else {  
 dispose();  
 }  
 }  
 catch (Exception se) {  
 se.printStackTrace();  
 }  
 }  
  
 else if (e.getSource() == mnuItemUpdate) {  
  
 // update a ticket  
 try {  
  
 String ticketID = JOptionPane.*showInputDialog*(null, "Enter ticket ID to be updated");  
  
 //check for which parameter wants to be updated  
 // array with parameter choices  
 String[] choices = {"Update Name", "Update Description"};  
 String response = (String) JOptionPane.*showInputDialog*(null,  
 "What would you like to update?",  
 "Update a Ticket",  
 JOptionPane.*QUESTION\_MESSAGE*,  
 null,  
 choices,  
 choices[0]);  
  
 //set input to oldparam value  
 String oldParam;  
 String newParam;  
 if(response.equals("Update Name")) {  
 oldParam = "ticket\_issuer";  
 newParam = JOptionPane.*showInputDialog*(null, "Enter new name");  
 }  
 else {  
 oldParam = "ticket\_description";  
 newParam = JOptionPane.*showInputDialog*(null, "Enter new description");  
 }  
  
 // run updateRecords  
 dao.updateRecords(ticketID, oldParam, newParam);  
 System.*out*.println("Changes were made to ticket " + ticketID + "...");  
 JOptionPane.*showMessageDialog*(null, "Ticket ID: " + ticketID + " was updated.");  
 }  
 catch (Exception se) {  
 se.printStackTrace();  
 }  
  
 }  
  
 else if (e.getSource() == mnuItemCloseTicket) {  
 try {  
 // get id of ticket  
 String ticketID = JOptionPane.*showInputDialog*(null, "Enter Ticket ID to be closed");  
  
 //check to see if user wants to continue with close  
 int response = JOptionPane.*showConfirmDialog*(null,  
 "Do you want to continue with this action?",  
 "Close This Ticket",  
 JOptionPane.*YES\_NO\_OPTION*);  
 if (response == JOptionPane.*YES\_OPTION*) {  
 // run close records  
 dao.closeRecords(ticketID);  
 JOptionPane.*showMessageDialog*(null,  
 "Ticket ID: " + ticketID + " was closed.");  
 System.*out*.println("Record has been closed");  
 }  
 else {  
 dispose();  
 }  
 }  
 catch (Exception se) {  
 se.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.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;  
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 smirza3\_tickets\_test(ticket\_id INT AUTO\_INCREMENT PRIMARY KEY, " +  
 "ticket\_issuer VARCHAR(30), " +  
 "ticket\_description VARCHAR(200), " +  
 "start\_date TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP, " +  
 "end\_date TIMESTAMP NULL," +  
 "status int DEFAULT 1)";  
 final String createUsersTable = "CREATE TABLE smirza3\_users\_test(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 smirza3\_users\_test(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 {  
 statement = getConnection().createStatement();  
 statement.executeUpdate("INSERT INTO smirza3\_tickets\_test" + "(ticket\_issuer, ticket\_description) VALUES(" + " '"  
 + ticketName + "','" + ticketDesc + "')", 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 = getConnection().createStatement();  
 results = statement.executeQuery("SELECT \* FROM smirza3\_tickets\_test");  
 //connect.close();  
 } catch (SQLException e1) {  
 e1.printStackTrace();  
 }  
 return results;  
 }  
  
 // continue coding for updateRecords implementation  
 public void updateRecords(String id, String oldParameter, String updatedParameter) {  
 try {  
 statement = getConnection().createStatement();  
  
 //set update target to be the tickets table  
 statement.executeUpdate("UPDATE smirza3\_tickets\_test SET " + oldParameter + " = '" + updatedParameter + "' WHERE ticket\_id = " + id + ";");  
 //apply updates to selected parameter (ticket\_id, ticket\_issuer, or ticket\_description)  
 //statement.executeUpdate("SET " + oldParameter + " = '" + updatedParameter + "'");  
 //set update target to the ticket entry with provided id  
 //statement.executeUpdate("WHERE ticket\_id = " + id);  
  
 //print to console  
 System.*out*.println("Record " + id + " has been updated...");  
  
 //close objects  
 statement.close();  
 *connect*.close();  
 }  
  
 catch (SQLException se) {  
 se.printStackTrace();  
 System.*out*.println("There was a problem updating the record");  
 System.*out*.println(se.getMessage());  
 }  
  
 }  
  
 // continue coding for deleteRecords implementation  
 public void deleteRecords(String id) {  
 try {  
 statement = getConnection().createStatement();  
  
 //delete record entry in the tickets table with given id  
 statement.executeUpdate("DELETE FROM smirza3\_tickets\_test WHERE ticket\_id = " + id);  
  
 //print to console  
 System.*out*.println("Record " + id + " has been deleted...");  
  
 //close objects  
 statement.close();  
 *connect*.close();  
 }  
 catch (SQLException se) {  
 se.printStackTrace();  
 System.*out*.println("There was a problem deleting the record");  
 System.*out*.println(se.getMessage());  
 }  
 }  
  
 // close records implementation  
 public void closeRecords(String id) {  
 try {  
 statement = getConnection().createStatement();  
  
 // set table target for update  
 statement.executeUpdate("UPDATE tickets.smirza3\_tickets\_test SET status = '0', end\_date = current\_timestamp() WHERE ticket\_id = " + id + ";");  
  
 // update the status of the ticket and add an end\_date timestamp  
 //statement.executeUpdate("");  
  
 // set ticket selection criteria for update  
 //statement.executeUpdate("WHERE ticket\_id = " + id);  
  
 // print to console  
 System.*out*.println("Ticket ID: " + id + " has been closed...");  
  
 //close objects  
 statement.close();  
 *connect*.close();  
 }  
 catch (SQLException se) {  
 se.printStackTrace();  
 System.*out*.println("There was a problem closing the record");  
 System.*out*.println(se.getMessage());  
 }  
 }  
}

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);   
  
 }  
  
}

userlist.csv

*admin*,*password*,*1  
Shahaan Mirza*,*smirza3*,*0  
Bruce Wayne*,*batman*,*0  
Peter Parker*,*spiderman*,*0*