**Dao.java**

package javaapplication1**;**import javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**import java.awt.event.WindowAdapter**;**import java.awt.event.WindowEvent**;**import java.sql.SQLException**;**@SuppressWarnings("serial")  
public class Tickets extends JFrame implements ActionListener {  
  
 // Main menu object items  
 private final JMenu mnuFile = new JMenu("File")**;** private final JMenu mnuAdmin = new JMenu("Admin")**;** private final JMenu mnuTickets = new JMenu("Tickets")**;** // class level member objects  
 Dao dao = new Dao()**;** // for CRUD operations  
 Boolean chkIfAdmin = null**;** // Sub menu item objects for all Main menu item objects  
 JMenuItem mnuItemExit**;** JMenuItem mnuItemUpdate**;** JMenuItem mnuItemDelete**;** JMenuItem mnuItemOpenTicket**;** JMenuItem mnuItemViewTicket**;** // Sub menu item objects to close ticket  
 JMenuItem mnuItemCloseTicket**;** public Tickets(Boolean isAdmin) {  
  
 // Remove Admin button if user is not admin  
 if (chkIfAdmin = isAdmin) {  
 createMenu()**;** prepareGUI()**;** } else {  
 createMenu()**;** prepareGUIUser()**;** }  
  
 }  
  
 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 any more desired sub menu items below  
  
 // NEW ADDITION:  
 // initialize first sub menu item for Tickets main menu  
 mnuItemCloseTicket = new JMenuItem("Close Ticket")**;** // add to Admin Main menu item  
 mnuAdmin.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)**;** // NEW ADDITION:  
 mnuItemCloseTicket.addActionListener(this)**;** }  
  
 private void prepareGUIUser() {  
 // create JMenu bar  
 JMenuBar bar = new JMenuBar()**;** bar.add(mnuFile)**;** // add main menu items in order, to JMenuBar  
 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)**;** }  
  
 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")**;** // 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() == mnuItemUpdate) {  
  
 try {  
 // Ask user for the ticket ID that they want to update  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to update")**;** // Prompt user what they'd like to update about the ticket  
 String[] options = {"Update Ticket Name"**,** "Update Ticket Description"}**;** String input = (String) JOptionPane.*showInputDialog* (null**,** "What property would you like to update?"**,** "Ticket Update"**,** JOptionPane.*QUESTION\_MESSAGE***,** null**,** options**,** options[**0**])**;** // Parse variables to replace information  
 String oldContent = null**;** String newContent = null**;** if (input.equals("Update Ticket Name")) {  
 oldContent = "ticket\_issuer"**;** newContent = JOptionPane.*showInputDialog*(null**,** "Please provide the new name")**;** } else if (input.equals("Update Ticket Description")) {  
 oldContent = "ticket\_description"**;** newContent = JOptionPane.*showInputDialog*(null**,** "Please enter the new description")**;** } else {  
 System.*out*.println("ERROR. Selections are not able to recognized.")**;** }  
  
 // Request update by calling upon updateRecords  
 dao.updateRecords(ticketnum**,** oldContent**,** newContent)**;** JOptionPane.*showMessageDialog*(null**,** "Ticket # " + ticketnum + " was updated")**;** System.*out*.println("Ticket # " + ticketnum + " was updated")**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 } else if (e.getSource() == mnuItemDelete) {  
 try {  
 // Ask user for the ticket ID that they want to delete  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to update")**;** // identify ticket  
 int input = JOptionPane.*showConfirmDialog*(null**,** "Are you sure you'd like to delete ticket #" + ticketnum + "?"**,** "Confirm"**,** JOptionPane.*YES\_NO\_OPTION***,** JOptionPane.*QUESTION\_MESSAGE*)**;** if (input == JOptionPane.*YES\_OPTION*) {  
 dao.deleteRecords(ticketnum)**;** JOptionPane.*showMessageDialog*(null**,** "Ticket # " + ticketnum + " was deleted")**;** System.*out*.println("The ticket #" + ticketnum + "is deleted")**;** } else if (input == JOptionPane.*NO\_OPTION*) {  
 System.*out*.println("Record Not Deleted")**;** } else if (input == JOptionPane.*CLOSED\_OPTION*) {  
 System.*out*.println("Nothing Deleted.")**;** }  
 dao.deleteRecords(ticketnum)**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 } else if (e.getSource() == mnuItemCloseTicket) {  
 try {  
 // Ask user for the ticket ID that they want to update  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to close")**;** int number = Integer.*parseInt*(ticketnum)**;** String newStatus = null**;** String open = "OPEN"**;** String closed = "CLOSED"**;** // Prompt user what they'd like to update about the ticket  
 String[] options = {"OPEN"**,** "CLOSE"}**;** String input = (String) JOptionPane.*showInputDialog* (null**,** "Would you like to open or close this ticket?"**,** "Ticket Status Change"**,** JOptionPane.*QUESTION\_MESSAGE***,** null**,** options**,** options[**0**])**;** if (input.equals("OPEN")) {  
 newStatus = open**;** } else if (input.equals("CLOSE")) {  
 newStatus = closed**;** } else {  
 System.*out*.println("ERROR. Selections are not able to recognized.")**;** }  
  
 // Request update by calling upon closeRecords  
 dao.closeRecords(ticketnum**,** newStatus)**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 }  
  
 }  
  
}

**Login.java**

package javaapplication1**;**import javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**import java.sql.PreparedStatement**;**import java.sql.ResultSet**;**import java.sql.SQLException**;**@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 jmelc\_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)**;** //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 javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**import java.awt.event.WindowAdapter**;**import java.awt.event.WindowEvent**;**import java.sql.SQLException**;**@SuppressWarnings("serial")  
public class Tickets extends JFrame implements ActionListener {  
  
 // Main menu object items  
 private final JMenu mnuFile = new JMenu("File")**;** private final JMenu mnuAdmin = new JMenu("Admin")**;** private final JMenu mnuTickets = new JMenu("Tickets")**;** // class level member objects  
 Dao dao = new Dao()**;** // for CRUD operations  
 Boolean chkIfAdmin = null**;** // Sub menu item objects for all Main menu item objects  
 JMenuItem mnuItemExit**;** JMenuItem mnuItemUpdate**;** JMenuItem mnuItemDelete**;** JMenuItem mnuItemOpenTicket**;** JMenuItem mnuItemViewTicket**;** // Sub menu item objects to close ticket  
 JMenuItem mnuItemCloseTicket**;** public Tickets(Boolean isAdmin) {  
  
 // Remove Admin button if user is not admin  
 if (chkIfAdmin = isAdmin) {  
 createMenu()**;** prepareGUI()**;** } else {  
 createMenu()**;** prepareGUIUser()**;** }  
  
 }  
  
 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 any more desired sub menu items below  
  
 // NEW ADDITION:  
 // initialize first sub menu item for Tickets main menu  
 mnuItemCloseTicket = new JMenuItem("Close Ticket")**;** // add to Admin Main menu item  
 mnuAdmin.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)**;** // NEW ADDITION:  
 mnuItemCloseTicket.addActionListener(this)**;** }  
  
 private void prepareGUIUser() {  
 // create JMenu bar  
 JMenuBar bar = new JMenuBar()**;** bar.add(mnuFile)**;** // add main menu items in order, to JMenuBar  
 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)**;** }  
  
 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")**;** // 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() == mnuItemUpdate) {  
  
 try {  
 // Ask user for the ticket ID that they want to update  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to update")**;** // Prompt user what they'd like to update about the ticket  
 String[] options = {"Update Ticket Name"**,** "Update Ticket Description"}**;** String input = (String) JOptionPane.*showInputDialog* (null**,** "What property would you like to update?"**,** "Ticket Update"**,** JOptionPane.*QUESTION\_MESSAGE***,** null**,** options**,** options[**0**])**;** // Parse variables to replace information  
 String oldContent = null**;** String newContent = null**;** if (input.equals("Update Ticket Name")) {  
 oldContent = "ticket\_issuer"**;** newContent = JOptionPane.*showInputDialog*(null**,** "Please provide the new name")**;** } else if (input.equals("Update Ticket Description")) {  
 oldContent = "ticket\_description"**;** newContent = JOptionPane.*showInputDialog*(null**,** "Please enter the new description")**;** } else {  
 System.*out*.println("ERROR. Selections are not able to recognized.")**;** }  
  
 // Request update by calling upon updateRecords  
 dao.updateRecords(ticketnum**,** oldContent**,** newContent)**;** JOptionPane.*showMessageDialog*(null**,** "Ticket # " + ticketnum + " was updated")**;** System.*out*.println("Ticket # " + ticketnum + " was updated")**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 } else if (e.getSource() == mnuItemDelete) {  
 try {  
 // Ask user for the ticket ID that they want to delete  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to update")**;** // identify ticket  
 int input = JOptionPane.*showConfirmDialog*(null**,** "Are you sure you'd like to delete ticket #" + ticketnum + "?"**,** "Confirm"**,** JOptionPane.*YES\_NO\_OPTION***,** JOptionPane.*QUESTION\_MESSAGE*)**;** if (input == JOptionPane.*YES\_OPTION*) {  
 dao.deleteRecords(ticketnum)**;** JOptionPane.*showMessageDialog*(null**,** "Ticket # " + ticketnum + " was deleted")**;** System.*out*.println("The ticket #" + ticketnum + "is deleted")**;** } else if (input == JOptionPane.*NO\_OPTION*) {  
 System.*out*.println("Record Not Deleted")**;** } else if (input == JOptionPane.*CLOSED\_OPTION*) {  
 System.*out*.println("Nothing Deleted.")**;** }  
 dao.deleteRecords(ticketnum)**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 } else if (e.getSource() == mnuItemCloseTicket) {  
 try {  
 // Ask user for the ticket ID that they want to update  
 String ticketnum = JOptionPane.*showInputDialog*(null**,** "Enter the ticket ID you want to close")**;** int number = Integer.*parseInt*(ticketnum)**;** String newStatus = null**;** String open = "OPEN"**;** String closed = "CLOSED"**;** // Prompt user what they'd like to update about the ticket  
 String[] options = {"OPEN"**,** "CLOSE"}**;** String input = (String) JOptionPane.*showInputDialog* (null**,** "Would you like to open or close this ticket?"**,** "Ticket Status Change"**,** JOptionPane.*QUESTION\_MESSAGE***,** null**,** options**,** options[**0**])**;** if (input.equals("OPEN")) {  
 newStatus = open**;** } else if (input.equals("CLOSE")) {  
 newStatus = closed**;** } else {  
 System.*out*.println("ERROR. Selections are not able to recognized.")**;** }  
  
 // Request update by calling upon closeRecords  
 dao.closeRecords(ticketnum**,** newStatus)**;** } catch (Exception e1) {  
 e1.printStackTrace()**;** }  
 }  
  
 }  
  
}

**ticketsJTable.java**

package javaapplication1**;**import javax.swing.table.DefaultTableModel**;**import java.sql.ResultSet**;**import java.sql.ResultSetMetaData**;**import java.sql.SQLException**;**import java.util.Vector**;**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,<P%9R5vy,1  
jmelchor,StrongPassword,1  
Joe User,123,0  
tom Thumb,abc,0  
Peter Parker,xzy,0