**Emma Prager**

**ITMD 411**

**Final Project**

**May 5, 2019**

Create a trouble ticket system for IIT. A database is used to store users and tickets.

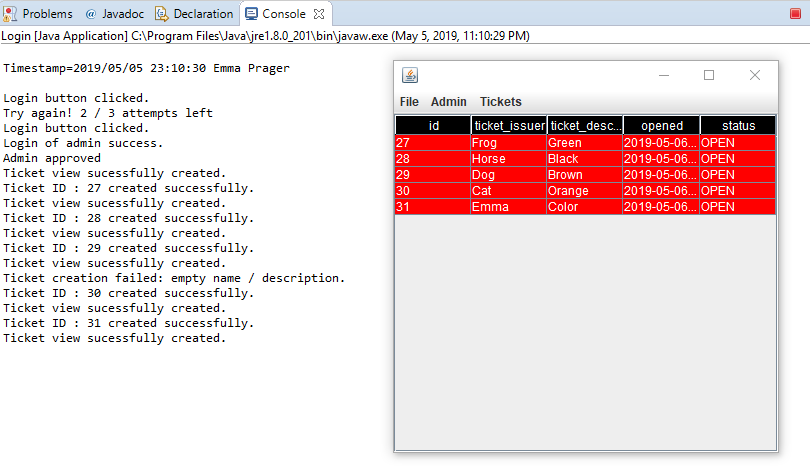
* Login.java: Login GUI authenticates both admin and regular users in the system. USER LOGIN: joe, user and ADMIN LOGIN: admin, admin1
* Tickets.java: Use to open, view, delete, and edit the tickets in the system.
* Dao.java where Dao stands for Data Access Object. This will allow for database connectivity and CRUD (Create Read Update Delete) like operations including insert, update, read, and delete tickets.
* ticketsJTable.java incorporates JTable structure and data

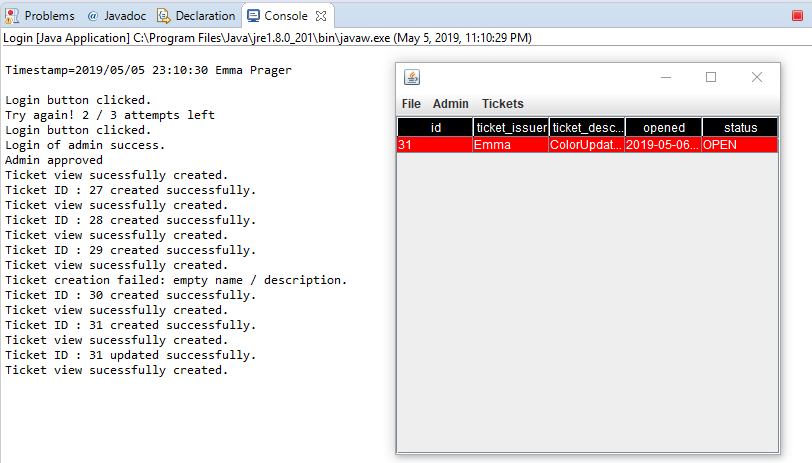
The user has access to basic functionality including inserting and viewing tickets, while admins can additionally update and delete tickets. The functionality of closing a ticket is incorporated into the status of updating a ticket.

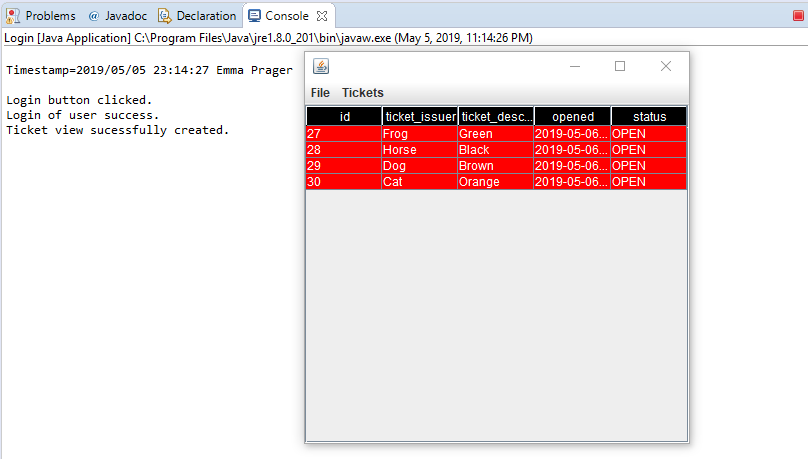
Testing process:

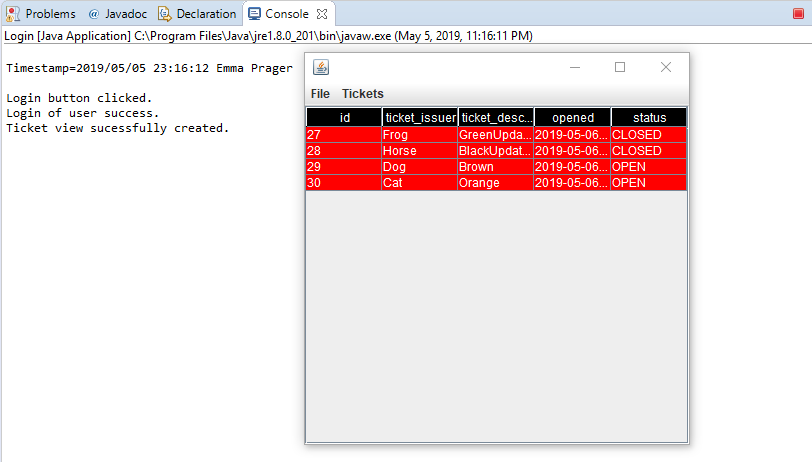
1. Insert at least 5 tickets into the DB table. *Include a record with your name* into the table.
2. Update your record by changing your ticket description.
3. Show a view of your updated ticket.
4. Delete your ticket from the DB.
5. Close two existing tickets.
6. Lastly, show a table view of all of your tickets.

The results are displayed to the console:



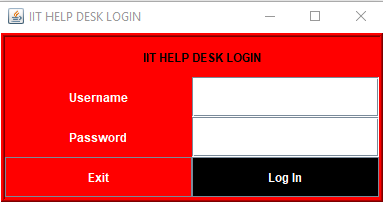


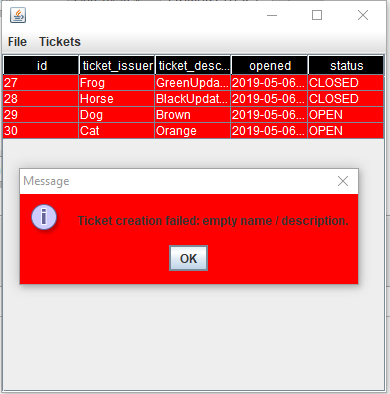




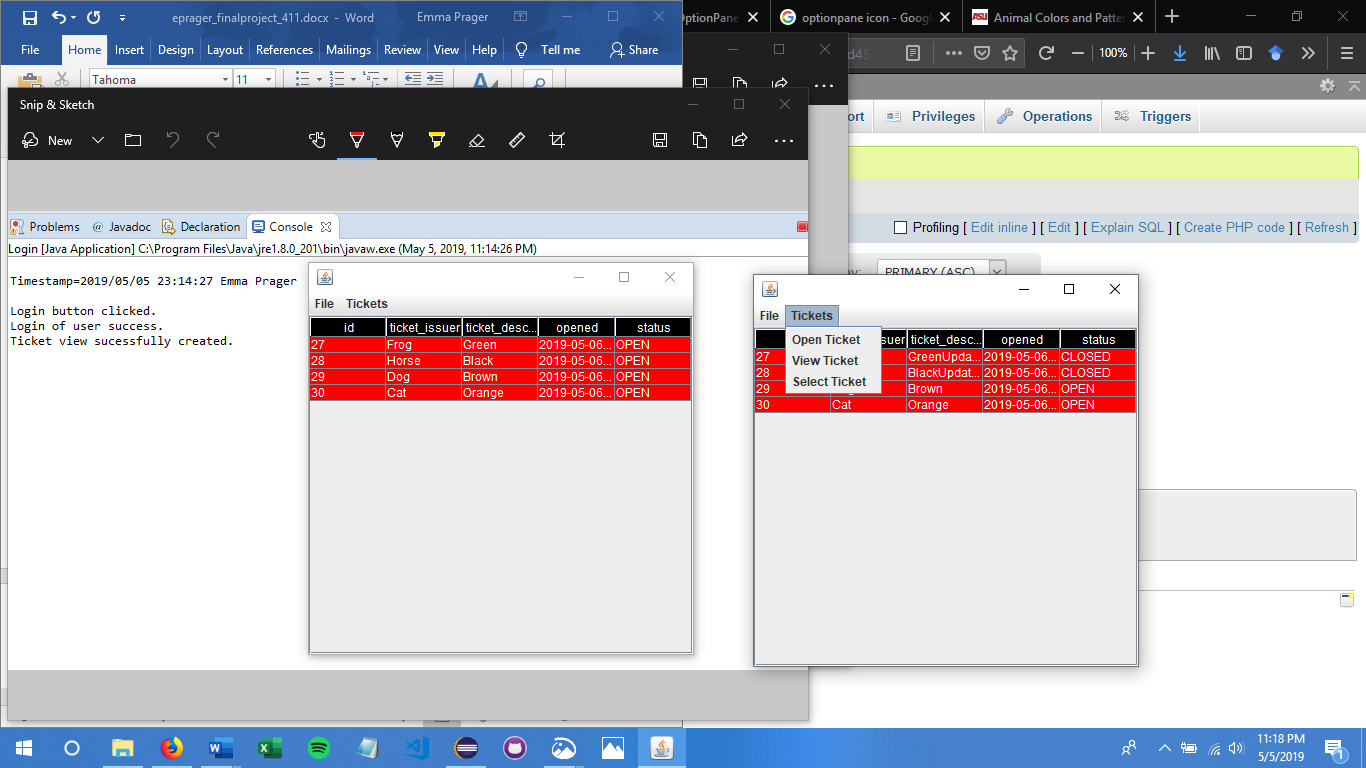
Extra Credit:

1. Outstanding GUI i.e., fluid navigation, buttons , menus, table view actions, event handling

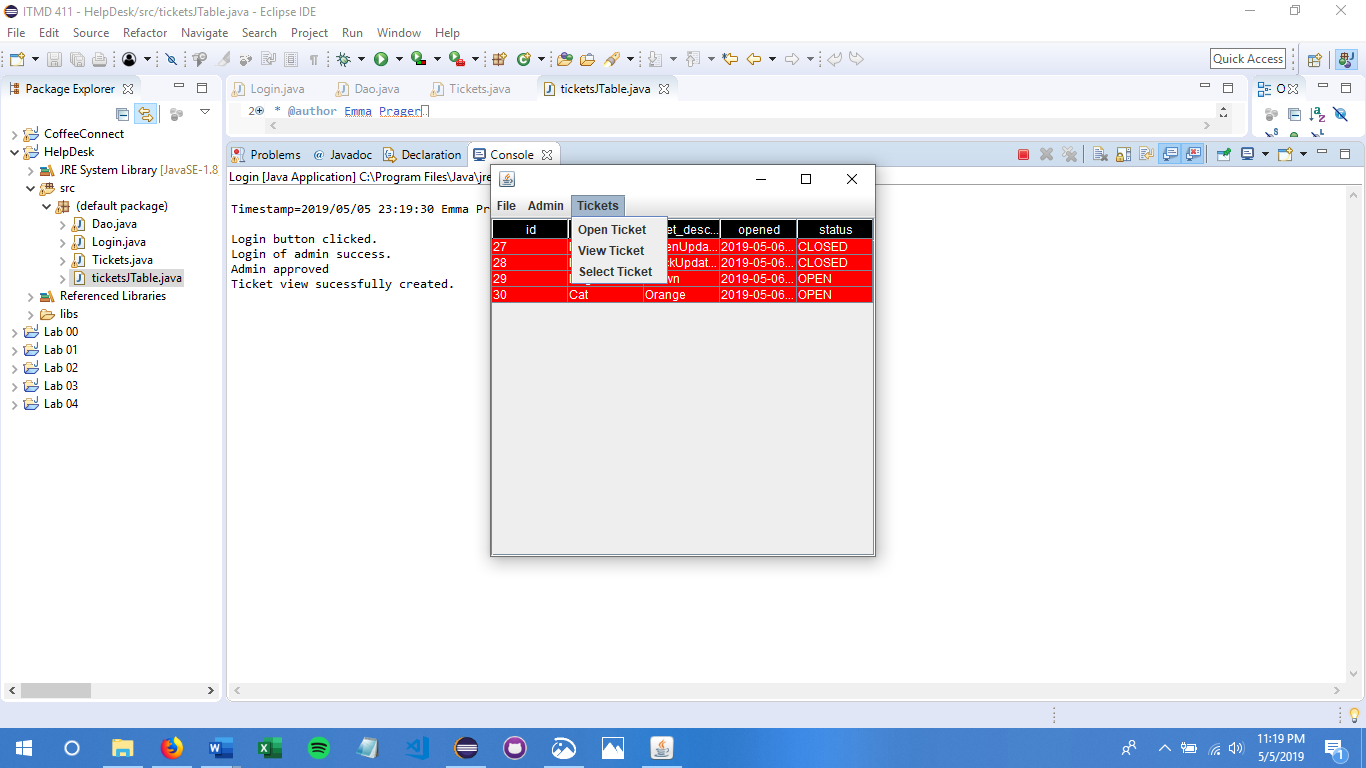




User view options



Additional admin options



14. Posting all project files to Github (<https://github.com/eprager/itmd411-final>)

Previous Labs:

<https://github.com/eprager/itmd411-lab01>

<https://github.com/eprager/itmd411-lab02>

<https://github.com/eprager/itmd411-lab03>

<https://github.com/eprager/itmd411-lab04>

Source Code

/\*\*

\* **@author** Emma Prager

\* **@date** 05 May 2019

\* **@title** Final Project

\* **@file** Login.java

\*/

**import** java.awt.Color;

**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.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** javax.swing.BorderFactory;

**import** javax.swing.JButton; //controls-label text fields, button

**import** javax.swing.JComponent;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JPasswordField;

**import** javax.swing.JTextField;

/\*\*

\* Login GUI authenticates both admin and regular users in the system.

\* USER LOGIN: joe, user

\* ADMIN LOGIN: admin, admin1

\*/

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

Dao conn = **new** Dao();

**public** Login() {

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

conn = **new** Dao();

setSize(400, 210);

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

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

//set background color

getContentPane().setBackground(Color.***red***);

//create border

((JComponent) getContentPane()).setBorder(BorderFactory.*createCompoundBorder*(BorderFactory.*createLineBorder*(Color.***white***), BorderFactory.*createCompoundBorder*(BorderFactory.*createRaisedBevelBorder*(), BorderFactory.*createLoweredBevelBorder*())));

JLabel title = **new** JLabel("IIT HELP ", JLabel.***RIGHT***);

JLabel title1 = **new** JLabel("DESK LOGIN", JLabel.***LEFT***);

//set text color

title.setForeground(Color.***black***);

title1.setForeground(Color.***black***);

// 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);

//set text color

lblUsername.setForeground(Color.***white***);

lblPassword.setForeground(Color.***white***);

//space for username and password

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

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

//color for text boxes

txtUname.setBackground(Color.***white***);

txtPassword.setBackground(Color.***white***);

txtUname.setForeground(Color.***black***);

txtPassword.setForeground(Color.***black***);

//button to login and exit

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

JButton btn = **new** JButton("Log In");

//make exit button blend in so login is the focus

btnExit.setBackground(Color.***red***);

btn.setBackground(Color.***black***);

btnExit.setForeground(Color.***white***);

btn.setForeground(Color.***white***);

// constraints

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

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

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

// ADD OBJECTS TO FRAME

add(title);

add(title1);

add(lblUsername);// 1st row filler

add(txtUname);

add(lblPassword); // 2nd row

add(txtPassword);// 3rd row

add(btnExit);

add(btn);

//listen for button to be clicked

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

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

@Override

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

System.***out***.println("Login button clicked.");

**boolean** adminFlag = **false**;

**int** admin = 0;

count = count + 1;

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

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

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

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

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

ResultSet rs = stmt.executeQuery();

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

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

**int** isAdmin = rs.getInt("admin");

**if** (isAdmin == 1)

{

adminFlag = **true**;

System.***out***.println("Login of admin success.");

}

**else** {

System.***out***.println("Login of user success.");

}

**new** Tickets(adminFlag);

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

dispose(); // CLOSE OUT THE WINDOW

} **else** {

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

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

// code for time stamp

String timeStamp = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.*getInstance*().getTime());

System.***out***.println("\nTimestamp=" + timeStamp + " Emma Prager\n");

}

}

/\*\*

\* **@author** Emma Prager

\* **@date** 05 May 2019

\* **@title** Final Project

\* **@file** Dao.java

\*/

**import** java.sql.CallableStatement;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.PreparedStatement;

**import** java.sql.SQLException;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

/\*\*

\* Dao stands for Data Access Object. This will allow for database connectivity and

\* CRUD (Create Read Update Delete) like operations including insert, update, read, and delete tickets.

\*/

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

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

Statement statement = **null**;

// Code database URL

**static** **final** String ***DB\_URL*** = "jdbc:mysql://www.papademas.net:3307/tickets?autoReconnect=true&useSSL=false";

// Database credentials

**static** **final** String ***USER*** = "fp411", ***PASS*** = "411";

**public** Connection connect() **throws** SQLException {

**return** DriverManager.*getConnection*(***DB\_URL***, ***USER***, ***PASS***);

}

// CRUD implementation

//INSERT

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

**int** id = 0;

**try** {

statement = connect().createStatement();

String timeStamp = **new** SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(Calendar.*getInstance*().getTime());

statement.executeUpdate("Insert into eprag\_tickets" + "(ticket\_issuer, ticket\_description, opened, status) values(" + " '" + ticketName + "','" + ticketDesc + "','" + timeStamp + "','" + "OPEN" + "')", 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;

}

//VIEW RECORDS

**public** ResultSet readRecords() {

ResultSet results = **null**;

**try** {

statement = connect().createStatement();

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

connect().close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

//VIEW a RECORD

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

ResultSet results = **null**;

**try** {

statement = connect().createStatement();

results = statement.executeQuery("SELECT \* FROM eprag\_tickets WHERE id = " + tid);

connect().close();

} **catch** (SQLException e1) {

e1.printStackTrace();

}

**return** results;

}

// continue coding for updateRecords implementation

//UPDATE

**public** **void** updateRecords(String tid, String desc, String status) {

//UPDATE `eprag\_tickets` SET `ticket\_description` = 'This thing is broken and I need it to be fixed.

//\r\n\r\nUpdate: fixed', `status` = 'CLOSED' WHERE `eprag\_tickets`.`id` = 7;

**try** {

Statement stmt = connect().createStatement();

ResultSet rs = stmt.executeQuery("SELECT ticket\_description FROM eprag\_tickets WHERE id = " + tid);

connect().close();

String results = **null**;

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

results = rs.getString("ticket\_description");

}

PreparedStatement ps = connect().prepareStatement("UPDATE eprag\_tickets SET ticket\_description = ?, status = ? WHERE id = ?");

// set the preparedstatement parameters

String descUp = results + "\nUpdate: " + desc;

ps.setString(1,descUp);

ps.setString(2,status);

ps.setString(3,tid);

ps.executeUpdate();

ps.close();

} **catch** (SQLException e) {

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

e.printStackTrace();

}

}

// continue coding for deleteRecords implementation

//DELETE

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

**try** {

statement = connect().createStatement();

String sql = "DELETE FROM eprag\_tickets WHERE eprag\_tickets.id = " + tid;

statement.executeUpdate(sql);

} **catch** (SQLException e) {

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

e.printStackTrace();

}

**return** tid;

}

}

/\*\*

\* **@author** Emma Prager

\* **@date** 05 May 2019

\* **@title** Final Project

\* **@file** Tickets.java

\*/

**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** javax.swing.ImageIcon;

**import** javax.swing.JFrame;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JMenuItem;

**import** java.sql.SQLException;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTable;

**import** javax.swing.LookAndFeel;

**import** javax.swing.UIManager;

/\*\*

\* Use to open, view, delete, and edit the tickets in the system.

\*/

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

// class level member objects

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

Boolean chkIfAdmin = **false**;

**public** Tickets(**boolean** isAdmin) {

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

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

// show the tables at startup

**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);

jt.setBackground(Color.***red***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

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

add(sp);

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

System.***out***.println("Ticket view sucessfully created.");

chkIfAdmin = **true**;

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed.");

e1.printStackTrace();

}

}

createMenu();

prepareGUI();

}

// 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** **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 sub menu item for File main menu

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

// add to File main menu item

mnuFile.add(mnuItemRefresh);

**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 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

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**);

mnuItemRefresh.addActionListener(**this**);

**if** (chkIfAdmin == **true**) { //on show these on admin

mnuItemUpdate.addActionListener(**this**);

mnuItemDelete.addActionListener(**this**);

}

mnuItemOpenTicket.addActionListener(**this**);

mnuItemViewTicket.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.***RED***);

setLocationRelativeTo(**null**);

setVisible(**true**);

}

@Override

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

UIManager.*put*("OptionPane.background", (Color.***red***));

UIManager.*put*("Panel.background", (Color.***red***));

UIManager.*put*("OptionPane.foreground", (Color.***white***));

UIManager.*put*("Panel.foreground", (Color.***white***));

// implement actions for sub menu items

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

System.***out***.println("Ticket system sucessfully exited.");

System.*exit*(0);

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

// get ticket information

//UIManager.put("

//ImageIcon icon = new ImageIcon("icon.jpg");

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

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

**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** {

// 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");

**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);

jt.setBackground(Color.***red***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

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

add(sp);

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

System.***out***.println("Ticket view sucessfully created.");

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed.");

e1.printStackTrace();

}

} **else**

System.***out***.println("Ticket creation failed.");

}

}

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

// 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);

jt.setBackground(Color.***red***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

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

add(sp);

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

System.***out***.println("Ticket view sucessfully created.");

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed.");

e1.printStackTrace();

}

}

//select specific ticket to view

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

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

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

{

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

System.***out***.println("Ticket view failed: empty id.");

}

**else** {

// retrieve tickets details for viewing in JTable

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

**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.selectRecords(tid)));

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

jt.setBackground(Color.***red***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

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

add(sp);

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

System.***out***.println("Ticket view sucessfully created.");

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed.");

e1.printStackTrace();

}

}

}

/\*

\* continue implementing any other desired sub menu items

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

\* syntax & logic as shown above

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

JMenuItem mnuItemUpdate;

JMenuItem mnuItemDelete;

\*/

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

{

// get ticket information

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

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

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

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

{

JOptionPane.*showMessageDialog*(**null**, "Ticket update failed: empty id / status.");

System.***out***.println("Ticket update failed: empty id / status.");

}

**else** {

// insert ticket information to database

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

//go to dao to update records

dao.updateRecords(ticketId, ticketDesc, ticketStatus);

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

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

System.***out***.println("Ticket ID : " + tid + " updated successfully.");

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

} **else**

System.***out***.println("Ticket update failed.");

**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.selectRecords(tid)));

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

jt.setBackground(Color.***red***);

jt.setForeground(Color.***white***);

jt.getTableHeader().setBackground(Color.***BLACK***);

jt.getTableHeader().setForeground(Color.***white***);

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

add(sp);

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

System.***out***.println("Ticket view sucessfully created.");

} **catch** (SQLException e1) {

System.***out***.println("Ticket view failed.");

e1.printStackTrace();

}

}

}

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

{

// get ticket information

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

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

{

JOptionPane.*showMessageDialog*(**null**, "Ticket deletion failed: empty tid.");

System.***out***.println("Ticket deletion failed: empty tid.");

}

**else** {

// check ticket information to database

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

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

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

**int** id = dao.deleteRecords(tid);

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

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

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

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

} **else**

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

}

**else** {

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

}

}

}

}

}

/\*\*

\* **@author** Emma Prager

\* **@date** 05 May 2019

\* **@title** Final Project

\* **@file** ticketsJTable.java

\*/

**import** java.sql.ResultSet;

**import** java.sql.ResultSetMetaData;

**import** java.sql.SQLException;

**import** java.util.Vector;

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

/\*\*

\* JTable structure and data

\*/

**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);

}

}