Grace Sopha

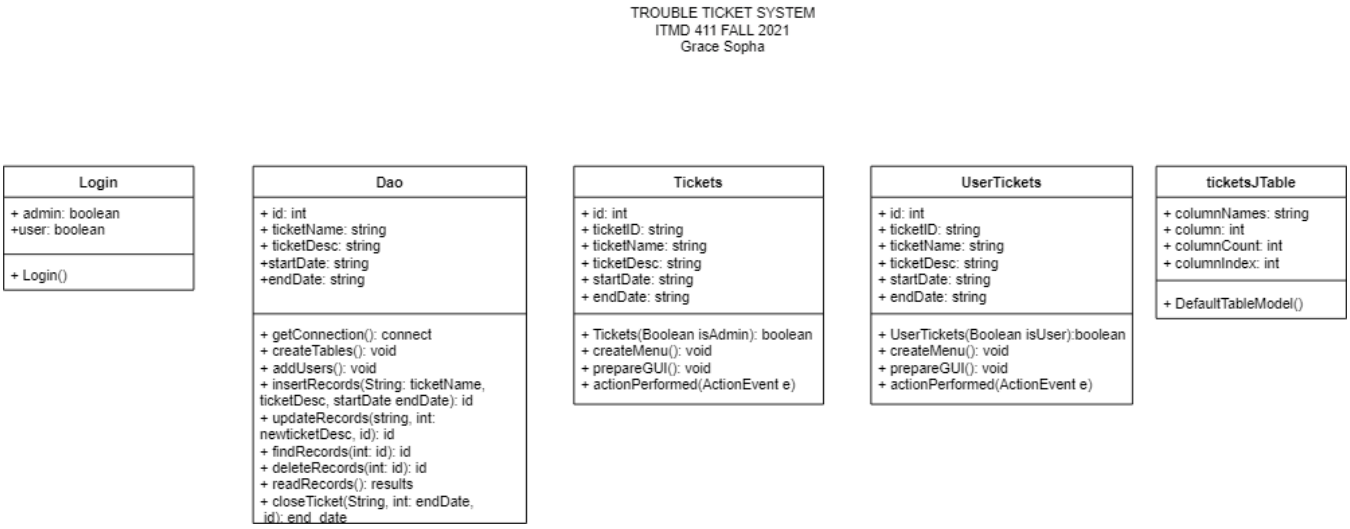
ITMD 411

Fall 2021

Final Project: Trouble Ticket System

This program utilizes the JDBC specifications to create a trouble ticket system. It will allow for an admin at the IIT Help Desk to see users' tickets that have been submitted to them for help. The admin will be allowed all CRUD activities, such as updating, deleting and closing a ticket. Users will be limited to only submitting and viewing tickets. Demonstrations are listed below.

UML Diagram



LOGIN CREDENTIALS

ADMIN:

- Username: admin
- Password: admin1

USERS:

- Username: Joe User
- Password: 123

- Username: Tom Thumb
- Password: abc
- Username: Peter Parker
- Password: xzy

Source Code

Login:

```
//This program will simulate a trouble ticket system

//ITMD 411 FALL 2021

//Made by Grace Sopha

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 {
```

```
//set up connection
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;
                boolean user = false;
                count = count + 1;
                // verify credentials of user (MAKE SURE TO CHANGE TO YOUR
TABLE NAME BELOW)
                String query = "SELECT * FROM gsoph_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();
```

```
//open the admin gui
if (rs.next()) {
        admin = rs.getBoolean("admin"); // get table
column value
        if (admin) {
                new Tickets(admin); //open Tickets file /
GUI interface
                setVisible(false); // HIDE THE FRAME
                dispose(); // CLOSE OUT THE WINDOW
        }
        //open the user gui
        else {
                        new UserTickets(user); //open
UserTickets file / GUI interface
                        dispose(); // CLOSE OUT THE
WINDOW
        }
}

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

}
catch (SQLException ex) {
ex.printStackTrace();
System.out.println("Cannot login");
}
```

```
                }
            });
            btnExit.addActionListener(e -> System.exit(0));

            setVisible(true); // SHOW THE FRAME
        }

        public static void main(String[] args) {

            new Login();
        }
}
```

Dao:

```
//This program will simulate a trouble ticket system
//ITMD 411 FALL 2021
//Made by Grace Sopha
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.sql.Connection;
import java.sql.Date;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.List;

import javax.swing.JOptionPane;

import com.mysql.cj.util.Util;

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();
                System.out.println("Cannot connect to database");
            }
            return connect;
        }

        // CRUD implementation

        public void createTables() {
            // variables for SQL Query table creations
            //table for tickets
            //add in ticket id, ticket name, ticket description, start date, end date
            final String createTicketsTable = "CREATE TABLE gsoph_tickets1(ticket_id INT
AUTO_INCREMENT PRIMARY KEY, ticket_issuer VARCHAR(30), ticket_description
VARCHAR(200), start_date VARCHAR(30), end_date VARCHAR(30))";
            //table for users
            final String createUsersTable = "CREATE TABLE gsoph_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());
                System.out.println("Could not create tables in given database");
        }
        // add users to user table
        addUsers();
}
// add list of users from userlist.csv file to users table
public void addUsers() {

        // 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 gsoph_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());
```

```
		}
	}
	//insert a ticket into database by name, desc, and start date
	public int insertRecords(String ticketName, String ticketDesc, String startDate, String endDate) {
		int id = 0;
		try {
			statement = getConnection().createStatement();
			statement.executeUpdate("Insert into gsoph_tickets1" + "(ticket_issuer, ticket_description, start_date, end_date) values(" + " '"
								+ ticketName + "','" + ticketDesc + "','" + startDate + "', '" + endDate + "')", 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;

	}
```

```
//read certain record from database
public ResultSet findRecords(int id) {
	// use statement to find certain ticket
	String SQL = "SELECT * FROM gsoph_tickets1 WHERE ticket_id = '" + id + "'";
	ResultSet results = null;
	try {
		//connect to database to find ticket
		System.out.println("Connecting to database to read records...");
		statement = connect.createStatement();
		results = statement.executeQuery(SQL);
	} catch (SQLException e1) {
		e1.printStackTrace();
		System.out.println("Cannot find ticket from database");
	}
	return results;
}



//read all records from database
public ResultSet readRecords() {
	//make blank resultset
	ResultSet results = null;
	try {
		System.out.println("Connecting to database to read records...");
		statement = connect.createStatement();
		results = statement.executeQuery("SELECT * FROM gsoph_tickets1");
```

```
                //connect.close();
            } catch (SQLException e1) {
                e1.printStackTrace();
                System.out.println("Cannot read from database");
            }
            return results;
        }

        // continue coding for updateRecords implementation
        public int updateRecords(String newticketDesc, int id) {
            // use prepared statement to update
            String SQL = "Update gsoph_tickets1 SET ticket_description = ? WHERE ticket_id
= ?";

            try (PreparedStatement pstmt = getConnection().prepareStatement(SQL)) {
                pstmt.setString(1, newticketDesc);
                pstmt.setInt(2, id);
                pstmt.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            return id;
        }

        // continue coding for deleteRecords implementation
        public int deleteRecords(int id) {
            //use a prepared statement to delete
```

```
        String SQL = "DELETE FROM gsoph_tickets1 WHERE ticket_id = ?";
        try (PreparedStatement pstmt = getConnection().prepareStatement(SQL)){
                //delete record of given ticket id
                pstmt.setLong(1, id);
                pstmt.executeUpdate();
        }
        catch (SQLException e) {
                e.printStackTrace();
                System.out.println("Cannot delete from database");
        }
        return id;
}

//close a ticket - admin
//end date
public String closeTicket(String endDate, int id) {
        String SQL = "UPDATE gsoph_tickets1 SET end_date = ? WHERE ticket_id = ?";
        try (PreparedStatement pstmt = getConnection().prepareStatement(SQL)) {
                pstmt.setString(1, endDate);
                pstmt.setInt(2, id);
                pstmt.executeUpdate();
        } catch (SQLException e) {
                e.printStackTrace();
                System.out.println("Cannot close the ticket");
        }
        return endDate;
```

```
        }

}
```

Tickets:

```
//This program will simulate a trouble ticket system
//ITMD 411 FALL 2021
//Made by Grace Sopha
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.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.time.Instant;
import java.util.Calendar;

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;
```

```
import javax.swing.JTextField;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import java.awt.Font;
import java.awt.HeadlessException;

import javax.swing.SwingConstants;

@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 mnuItemUpdateTicket;
	JMenuItem mnuItemDeleteTicket;
	JMenuItem mnuItemOpenTicket;
	JMenuItem mnuItemCloseTicket;
	JMenuItem mnuItemViewTicket;
```

```
JMenuItem mnuItemFindTicket;
private JLabel welcomeLabel;

//create view for admin account
public Tickets(Boolean isAdmin) {
	chkIfAdmin = isAdmin;
	createMenu();
	prepareGUI();
}

//admin menu
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
	mnuItemUpdateTicket = new JMenuItem("Update Ticket");
	// add to Admin main menu item
	mnuAdmin.add(mnuItemUpdateTicket);

	// initialize second sub menu items for Admin main menu
	mnuItemDeleteTicket = new JMenuItem("Delete Ticket");
	// add to Admin main menu item
	mnuAdmin.add(mnuItemDeleteTicket);
```

```
// initialize first sub menu item for Tickets main menu
mnuItemOpenTicket = new JMenuItem("Open Ticket");
// add to Ticket Main menu item
mnuTickets.add(mnuItemOpenTicket);

//add to admin sub menu
//let the admin close ticket
mnuItemCloseTicket = new JMenuItem("Close Ticket");
// add to Ticket Main menu item
mnuAdmin.add(mnuItemCloseTicket);

// initialize second sub menu item for Tickets main menu
mnuItemViewTicket = new JMenuItem("View Tickets");
// add to Ticket Main menu item
mnuTickets.add(mnuItemViewTicket);

// initialize second sub menu item for Tickets main menu
mnuItemFindTicket = new JMenuItem("Find Ticket");
// add to Ticket Main menu item
mnuTickets.add(mnuItemFindTicket);


// initialize any more desired sub menu items below

/* Add action listeners for each desired menu item *************/
mnuItemExit.addActionListener(this);
```

```
        mnuItemUpdateTicket.addActionListener(this);
        mnuItemDeleteTicket.addActionListener(this);
        mnuItemOpenTicket.addActionListener(this);
        mnuItemCloseTicket.addActionListener(this);
        mnuItemViewTicket.addActionListener(this);
        mnuItemFindTicket.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*
         */
}

//admin gui
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, 200);
	getContentPane().setBackground(Color.LIGHT_GRAY);

	welcomeLabel = new JLabel("Welcome, Admin!");
	welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
	welcomeLabel.setFont(new Font("Tahoma", Font.PLAIN, 24));
	getContentPane().add(welcomeLabel, BorderLayout.CENTER);
	setLocationRelativeTo(null);
	setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
	// implement actions for sub menu items
	//exit button
	if (e.getSource() == mnuItemExit) {
		System.exit(0);
	}
	//open a new ticket button
	else if (e.getSource() == mnuItemOpenTicket) {

		//get rid of welcome label
```

```
welcomeLabel.setText(" ");


try {
        //set up timestamp
        Timestamp timestamp = new
Timestamp(System.currentTimeMillis());
    //returns a LocalDateTime object which represents the same date-time
value as this Timestamp
    String str=timestamp.toString();


        // get ticket information
        String ticketName = JOptionPane.showInputDialog(null, "Enter
your name");
        String ticketDesc = JOptionPane.showInputDialog(null, "Enter a
ticket description");
        String startDate = str.toString();
        String endDate = "";


        //do not allow blank entries
        if (ticketName.isEmpty() && ticketDesc.isEmpty()) {
            //notify user
            System.out.println("Ticket cannot be created!");
            JOptionPane.showMessageDialog(null, "Ticket cannot be
created! Please make sure all fields are filled in.");
        }
        //allow entries that are filled in to be entered to database
        else {
            // insert ticket information to database
```

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


                    // 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");
                    }
                }
            } catch (HeadlessException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
                System.out.println("Ticket cannot be created!");
                System.out.println("Ticket cannot be created! Please make sure
all fields are filled in.");
            }
        }
        // view ticket button
        else if (e.getSource() == mnuItemViewTicket) {
            //get rid of welcome label
            welcomeLabel.setText(" ");


            // 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, 50, 300, 400);
JScrollPane sp = new JScrollPane(jt);
getContentPane().add(sp);
setVisible(true); // refreshes or repaints frame on screen



}
catch (SQLException e1) {
e1.printStackTrace();
System.out.println("Tickets cannot be viewed!");
}
}

//find ticket button
else if (e.getSource() == mnuItemFindTicket) {
//get rid of welcome label
welcomeLabel.setText(" ");

// retrieve certain ticket details for viewing in JTable
try {
String ticketID = JOptionPane.showInputDialog(null, "Please enter
the ticket id to find: ");
int id = Integer.parseInt(ticketID);

//show results of database with j table
```

```
JTable jt = new
JTable(ticketsJTable.buildTableModel(dao.findRecords(id)));
jt.setBounds(30, 50, 300, 400);
JScrollPane sp = new JScrollPane(jt);
getContentPane().add(sp);
setVisible(true); // refreshes or repaints frame on screen
// display results if successful or not to console / dialog box
if (id != 0) {
System.out.println("Ticket ID : " + id + " found
successfully!");
JOptionPane.showMessageDialog(null, "Ticket id: " + id + "
found");
}

}
catch (SQLException e1) {
e1.printStackTrace();
System.out.println("Tickets cannot be found!");
}
}

//update ticket button
else if (e.getSource() == mnuItemUpdateTicket) {
//get rid of welcome label
welcomeLabel.setText(" ");

// retrieve all tickets details for viewing in JTable
try {
```

```
//enter the ticket id for the ticket to update
String ticketID = JOptionPane.showInputDialog(null, "Please enter the ticket id to update: ");
String newticketDesc = JOptionPane.showInputDialog(null, "Please enter the new description: ");
int id = Integer.parseInt(ticketID);
//confirm update
int ans = JOptionPane.showConfirmDialog(null, "Update ticket id: " + id + " with new description?", "WARNING", JOptionPane.YES_NO_OPTION);
//choose yes or no option
if (ans == JOptionPane.YES_OPTION) {
    //update database
    int update = dao.updateRecords(newticketDesc, id);
    //notify the update has been completed
    System.out.println("Ticket ID : " + id + " updates successfully!");
    JOptionPane.showMessageDialog(null, "Ticket id: " + id + " updated");
    //it can't be 0
    if (id == 0) {
        JOptionPane.showMessageDialog(null, "The ticket cannot be updated");
    }
}
else if (ans == JOptionPane.NO_OPTION) {
    System.out.println("Ticket ID : " + id + " was not updated");
    JOptionPane.showMessageDialog(null, "Ticket id: " + id + " was not updated");
```

```
                }

            } catch (Exception e1) {
                e1.printStackTrace();
                System.out.println("The ticket cannot be updated");
            }

        }
        // delete ticket button
        else if (e.getSource() == mnuItemDeleteTicket) {
            //get rid of welcome label
            welcomeLabel.setText(" ");

            //choose a ticket to delete
            try {
                String ticketID = JOptionPane.showInputDialog(null,
"Please enter the ticket id to delete: ");
                int id = Integer.parseInt(ticketID);

                // confirm that user wants to delete ticket
                int ans = JOptionPane.showConfirmDialog(null, "Delete
ticket id: " + id + "?", "WARNING", JOptionPane.YES_NO_OPTION);
                //choose yes or no option
                if (ans == JOptionPane.YES_OPTION) {
                    //update database
                    int delete = dao.deleteRecords(id);
                    //notify the update has been completed
```

```
                    System.out.println("Ticket ID : " + id + " deleted
successfully!");
                    JOptionPane.showMessageDialog(null, "Ticket id: "
+ id + " deleted");
                //it can't be 0
                    if (id == 0) {
                        JOptionPane.showMessageDialog(null, "The
ticket cannot be deleted");
                                }
                }
                else if (ans == JOptionPane.NO_OPTION) {
                    System.out.println("Ticket ID : " + id + " was not
deleted");
                    JOptionPane.showMessageDialog(null, "Ticket id: "
+ id + " was not deleted");
                }

            } catch (Exception e1) {
                e1.printStackTrace();
                System.out.println("The ticket cannot be deleted");
            }

        }
            // close ticket button
            else if (e.getSource() == mnuItemCloseTicket) {
                //get rid of welcome label
                welcomeLabel.setText(" ");

                try {
```

```
//set up timestamp
//enter ticket to close
String ticketID = JOptionPane.showInputDialog(null,
"Please enter the ticket id to close: ");
int id = Integer.parseInt(ticketID);
Timestamp timestamp = new
Timestamp(System.currentTimeMillis());
//returns a LocalDateTime object which represents the same date-
time value as this Timestamp
String str=timestamp.toString();

// get ticket information
String endDate = str.toString();

// confirm that user wants to delete ticket
int ans = JOptionPane.showConfirmDialog(null, "Close
ticket id: " + id + "?", "WARNING", JOptionPane.YES_NO_OPTION);
//choose yes or no option
if (ans == JOptionPane.YES_OPTION) {
    // insert ticket information to database
    String close = dao.closeTicket(endDate, id);
    //notify the update has been completed
    System.out.println("Ticket ID : " + id + " closed
successfully!");
    JOptionPane.showMessageDialog(null, "Ticket id: "
+ id + " closed");
    //it can't be 0
    if (id == 0) {
```

```
                                                JOptionPane.showMessageDialog(null, "The
ticket cannot be closed");
                                        }
                                }
                                else if (ans == JOptionPane.NO_OPTION) {
                                        System.out.println("Ticket ID : " + id + " was not
closed");
                                        JOptionPane.showMessageDialog(null, "Ticket id: "
+ id + " was not closed");
                                        }
                        } catch (HeadlessException e1) {
                                // TODO Auto-generated catch block
                                e1.printStackTrace();
                                System.out.println("Ticket cannot be created!");
                        }
                }
                /*
                 * continue implementing any other desired sub menu items (like for update and
                 * delete sub menus for example) with similar syntax & logic as shown above
                 */
        }
}
```

UserTickets:

```
//This program will simulate a trouble ticket system
//ITMD 411 FALL 2021
//Made by Grace Sopha
import java.awt.BorderLayout;
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.HeadlessException;
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.Timestamp;

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;
import javax.swing.SwingConstants;

@SuppressWarnings("serial")
public class UserTickets extends JFrame implements ActionListener {

	// class level member objects
	Dao dao = new Dao(); // for CRUD operations
	Boolean chkIfUser = null;
```

```
// Main menu object items
private JMenu mnuFile = new JMenu("File");
private JMenu mnuUser = new JMenu("User");
private JMenu mnuTickets = new JMenu("Tickets");

// Sub menu item objects for all Main menu item objects
JMenuItem mnuItemExit;
JMenuItem mnuItemOpenTicket;
JMenuItem mnuItemViewTicket;
private JLabel welcomeLabel;

//create view for regular user account
	public UserTickets(Boolean isUser) {
		chkIfUser = isUser;
		createMenu();
		prepareGUI();
	}

//user menu
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 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 Tickets");
        // add to Ticket Main menu item
        mnuTickets.add(mnuItemViewTicket);

        // initialize any more desired sub menu items below

        /* Add action listeners for each desired menu item *************/
        mnuItemExit.addActionListener(this);
        mnuItemOpenTicket.addActionListener(this);
        mnuItemViewTicket.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*
         */
}

//user gui
private void prepareGUI() {
```

```
        // create JMenu bar
        JMenuBar bar = new JMenuBar();
        bar.add(mnuFile); // add main menu items in order, to JMenuBar
        bar.add(mnuUser);
        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, 200);
        getContentPane().setBackground(Color.LIGHT_GRAY);

        welcomeLabel = new JLabel("Welcome to the IIT Help Desk!");
        welcomeLabel.setHorizontalAlignment(SwingConstants.CENTER);
        welcomeLabel.setFont(new Font("Tahoma", Font.PLAIN, 24));
        getContentPane().add(welcomeLabel, BorderLayout.CENTER);
        setLocationRelativeTo(null);
        setVisible(true);
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
	// implement actions for sub menu items
	//exit button
	if (e.getSource() == mnuItemExit) {
		System.exit(0);
	}
	//open a new ticket button
	else if (e.getSource() == mnuItemOpenTicket) {
		//get rid of welcome label
		welcomeLabel.setText(" ");


		try {
			//set up timestamp
			Timestamp timestamp = new
Timestamp(System.currentTimeMillis());
		//returns a LocalDateTime object which represents the same date-time
value as this Timestamp
		String str=timestamp.toString();


			// get ticket information
			String ticketName = JOptionPane.showInputDialog(null, "Enter
your name");
			String ticketDesc = JOptionPane.showInputDialog(null, "Enter a
ticket description");
			String startDate = str.toString();
			String endDate = "";
```

```
//do not allow blank entries
if (ticketName.isEmpty() && ticketDesc.isEmpty()) {
	//notify user
	System.out.println("Ticket cannot be created!");
	JOptionPane.showMessageDialog(null, "Ticket cannot be created! Please make sure all fields are filled in.");
}
//allow entries that are filled in to be entered to database
else {
	// insert ticket information to database
	int id = dao.insertRecords(ticketName, ticketDesc, startDate, endDate);


	// 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");
	}
}
} catch (HeadlessException e1) {
	// TODO Auto-generated catch block
	e1.printStackTrace();
	System.out.println("Ticket cannot be created!");
	System.out.println("Ticket cannot be created! Please make sure all fields are filled in.");
}
```

```
		}
		// view ticket button
		else if (e.getSource() == mnuItemViewTicket) {
			//get rid of welcome label
			welcomeLabel.setText(" ");

			// 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, 50, 300, 400);
				JScrollPane sp = new JScrollPane(jt);
				getContentPane().add(sp);
				setVisible(true); // refreshes or repaints frame on screen

			}
			catch (SQLException e1) {
				e1.printStackTrace();
				System.out.println("Tickets cannot be viewed!");
			}
			}
	}
}
```

ticketsJTable:

```
//This program will simulate a trouble ticket system
//ITMD 411 FALL 2021
//Made by Grace Sopha
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));
		}
```

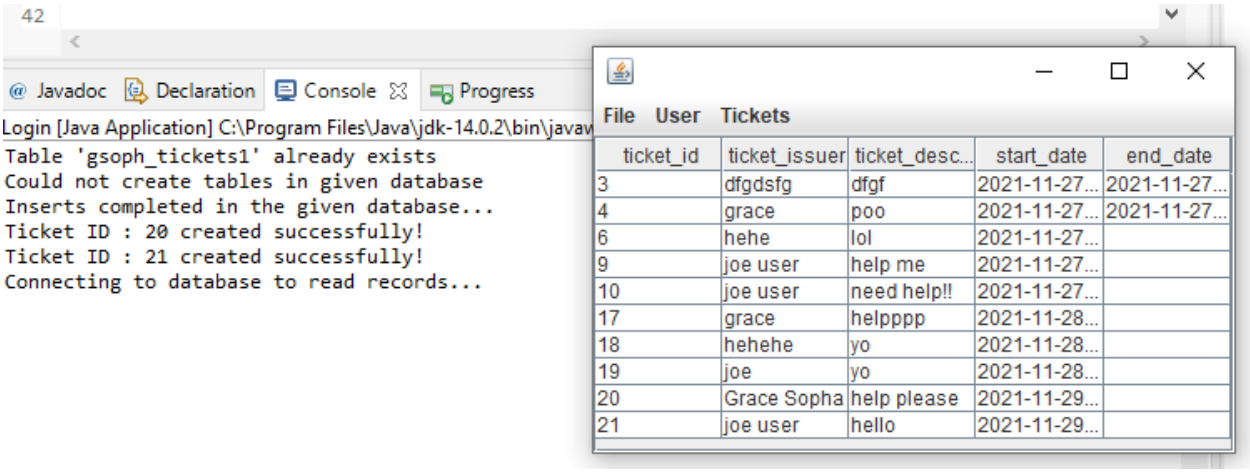
```
            // data of the table
            Vector<Vector<Object>> data = new Vector<Vector<Object>>();
            while (rs.next()) {
                Vector<Object> vector = new Vector<Object>();
                for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++)
{
                    vector.add(rs.getObject(columnIndex));
                }
                data.add(vector);
            }
            // return data/col.names for JTable
            return new DefaultTableModel(data, columnNames);

        }

}
```

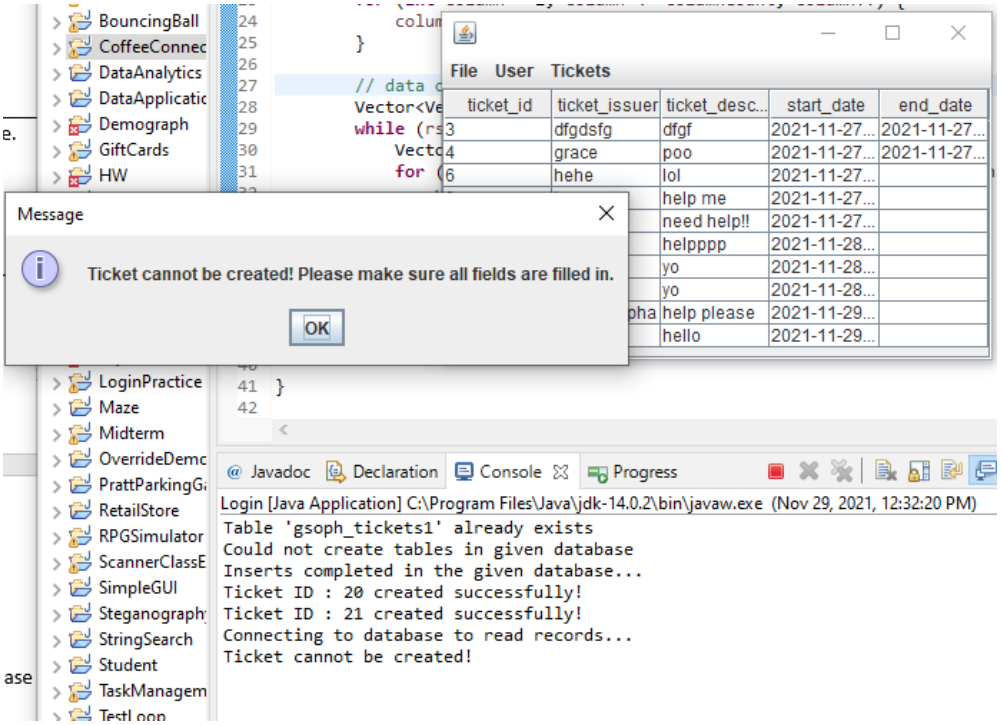
<u>Outputs</u>

Insert a ticket:

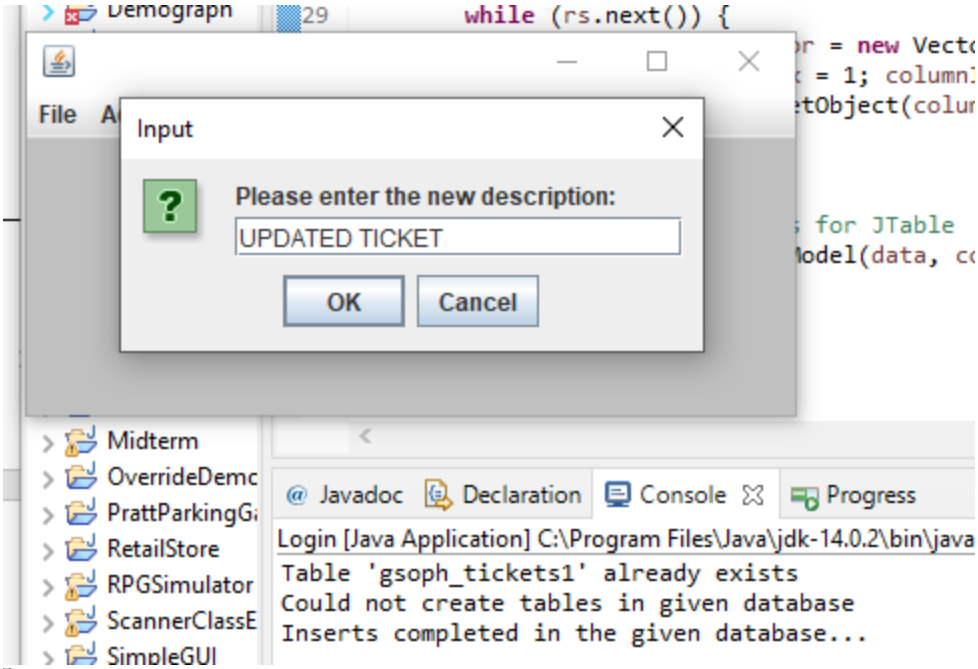


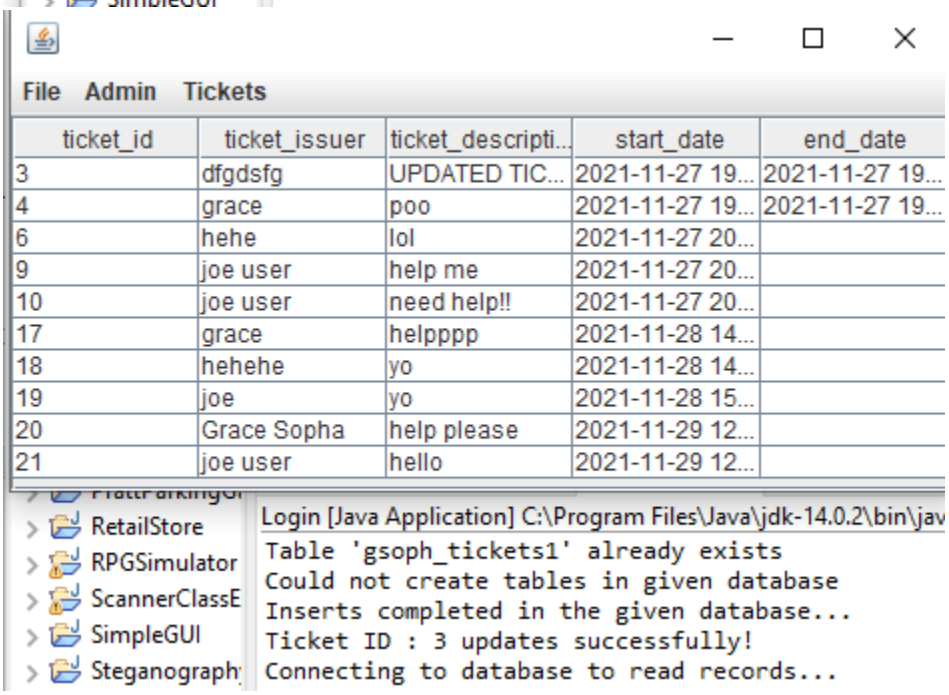
ticket_id	ticket_issuer	ticket_desc...	start_date	end_date
3	dfgdsfg	dfgf	2021-11-27...	2021-11-27...
4	grace	poo	2021-11-27...	2021-11-27...
6	hehe	lol	2021-11-27...	
9	joe user	help me	2021-11-27...	
10	joe user	need help!!	2021-11-27...	
17	grace	helpppp	2021-11-28...	
18	hehehe	yo	2021-11-28...	
19	joe	yo	2021-11-28...	
20	Grace Sopha	help please	2021-11-29...	
21	joe user	hello	2021-11-29...	

If fields are null/empty

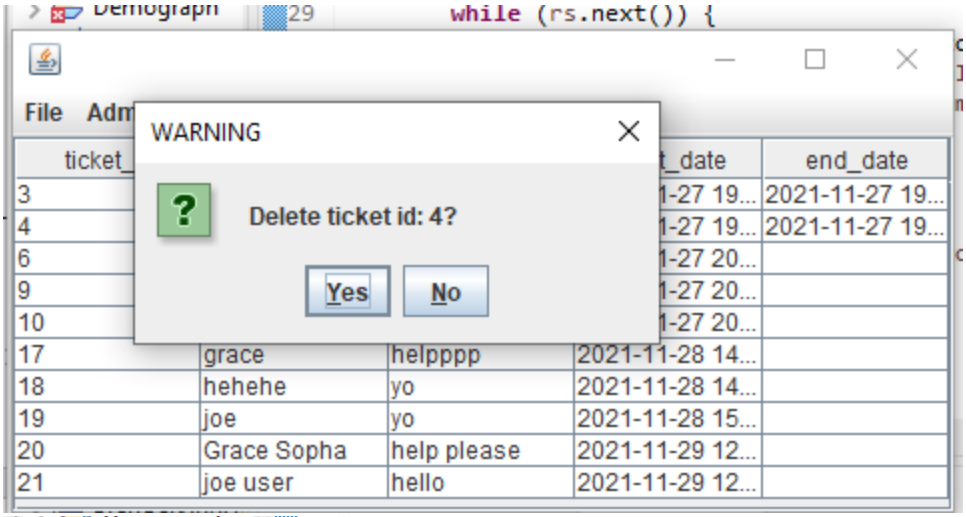


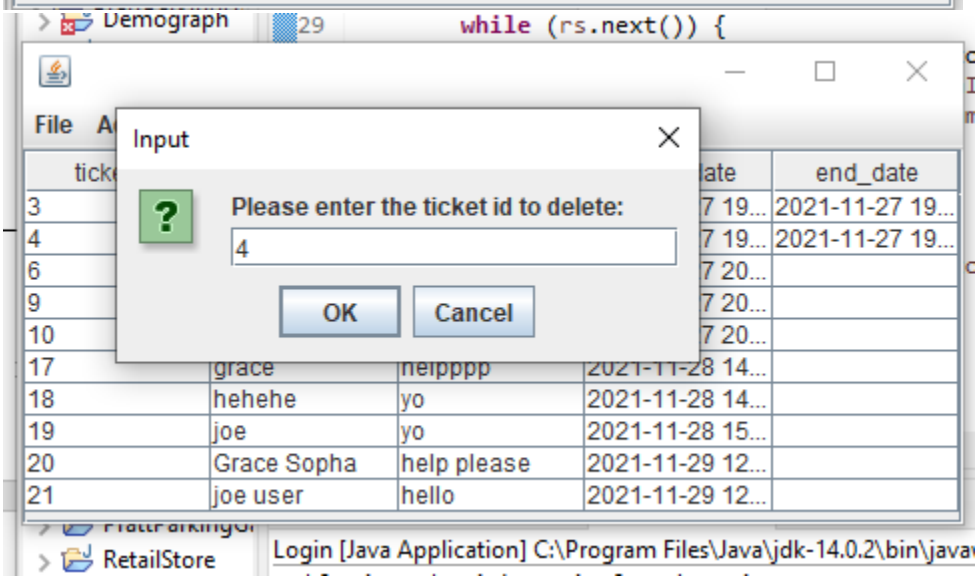
Update a record:

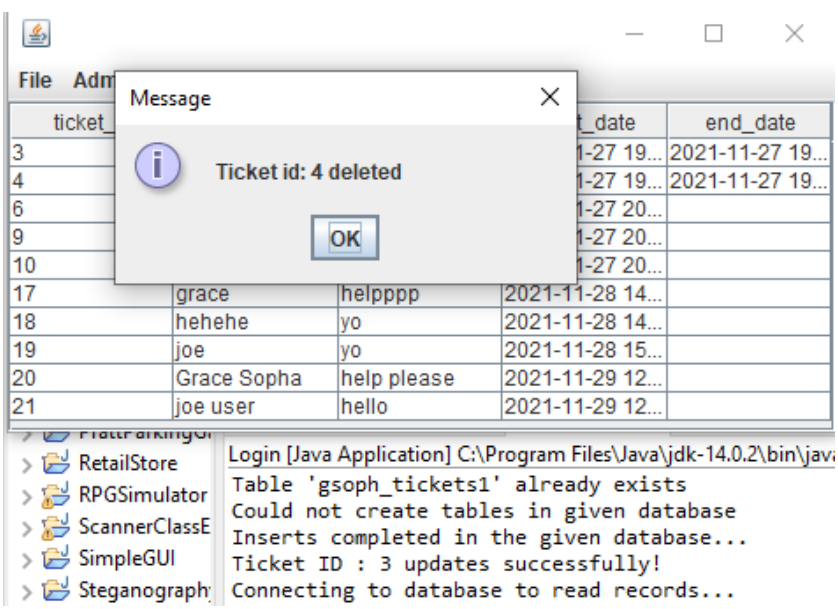




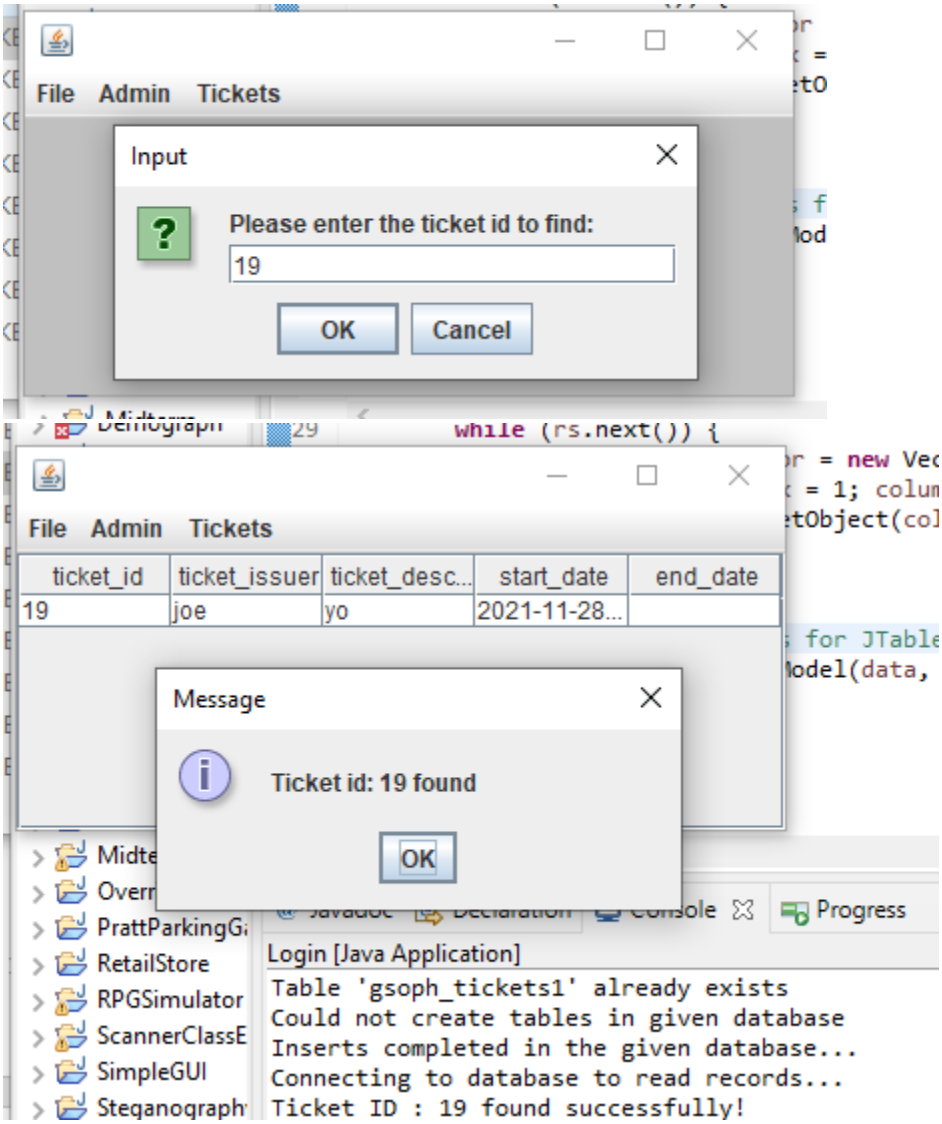
Delete a record:


while (rs.next()) {
File
WARNING
Delete ticket id: 4?
Yes
No
end_date
2021-11-27 19...
2021-11-27 19...
2021-11-27 19...
2021-11-27 19...
grace
helpppp
2021-11-28 14...
hehehe
yo
2021-11-28 14...
joe
yo
2021-11-28 15...
Grace Sopha
help please
2021-11-29 12...
joe user
hello
2021-11-29 12...

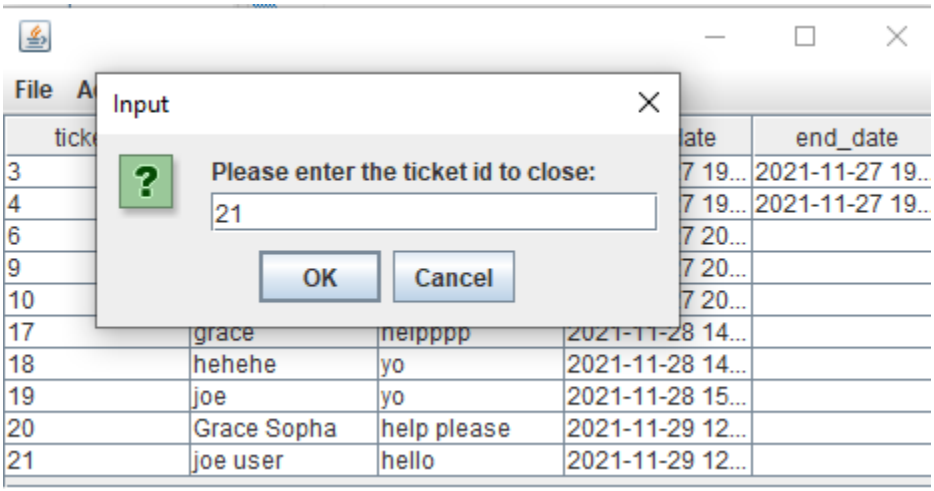

while (rs.next()) {
File
Input
Please enter the ticket id to delete:
4
OK
Cancel
end_date
2021-11-27 19...
2021-11-27 19...
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello
RetailStore
Login [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\java

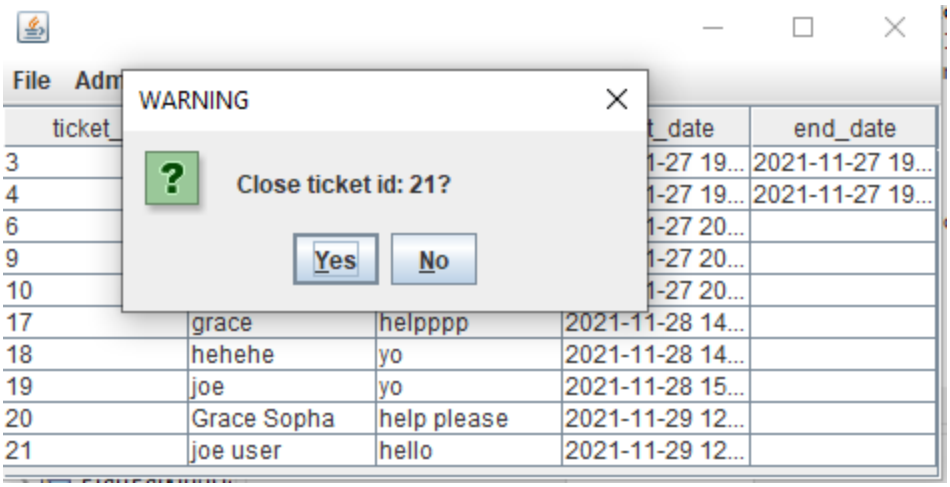

File
Message
Ticket id: 4 deleted
OK
end_date
2021-11-27 19...
2021-11-27 19...
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello
RetailStore
RPGSimulator
SimpleGUI
Login [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\jav
Table 'gsoph_tickets1' already exists
Could not create tables in given database
Inserts completed in the given database...
Ticket ID : 3 updates successfully!
Connecting to database to read records...

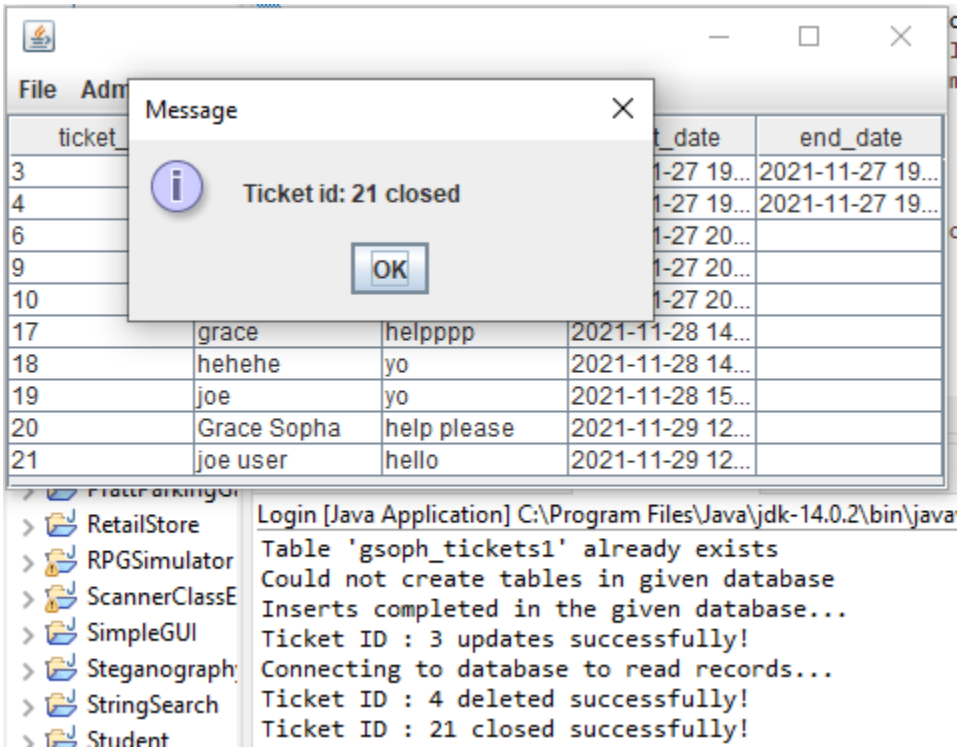
View/find a record:

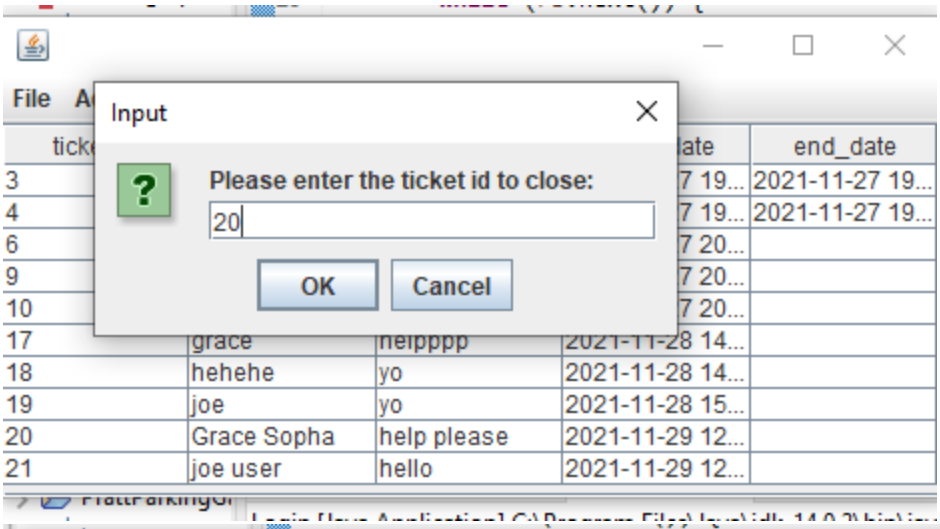


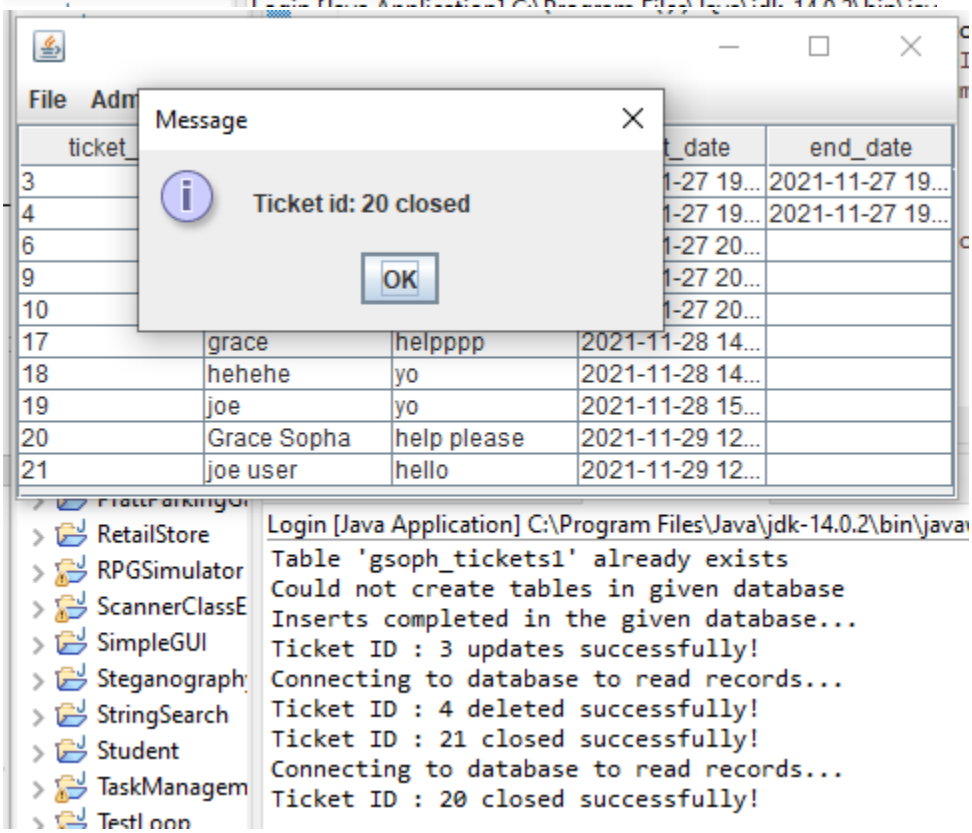
Close a ticket:

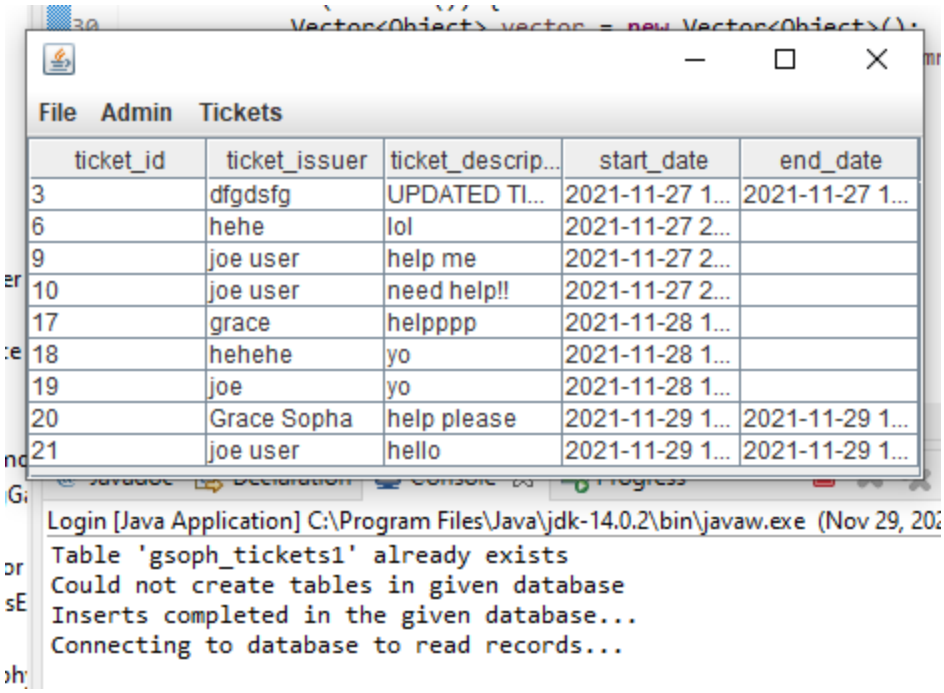



File
WARNING
Close ticket id: 21?
Yes
No
ticket_
t_date
end_date
3
4
6
9
10
17
18
19
20
21
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello
2021-11-28 14...
2021-11-28 15...
2021-11-29 12...
2021-11-27 19...
2021-11-27 20...


File
Message
Ticket id: 21 closed
OK
ticket_
t_date
end_date
3
4
6
9
10
17
18
19
20
21
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello
2021-11-28 14...
2021-11-28 15...
2021-11-29 12...
2021-11-27 19...
2021-11-27 20...
RetailStore
RPGSimulator
ScannerClassE
SimpleGUI
Steganograph
StringSearch
Student
Login [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\java
Table 'gsoph_tickets1' already exists
Could not create tables in given database
Inserts completed in the given database...
Ticket ID : 3 updates successfully!
Connecting to database to read records...
Ticket ID : 4 deleted successfully!
Ticket ID : 21 closed successfully!


File
Input
Please enter the ticket id to close:
20
OK
Cancel
end_date
2021-11-27 19...
2021-11-27 19...
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello


File
Message
Ticket id: 20 closed
OK
end_date
2021-11-27 19...
2021-11-27 19...
grace
helpppp
hehehe
yo
joe
yo
Grace Sopha
help please
joe user
hello
RetailStore
RPGSimulator
ScannerClassE
SimpleGUI
Steganograph
StringSearch
Student
TaskManagem
Login [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\java
Table 'gsoph_tickets1' already exists
Could not create tables in given database
Inserts completed in the given database...
Ticket ID : 3 updates successfully!
Connecting to database to read records...
Ticket ID : 4 deleted successfully!
Ticket ID : 21 closed successfully!
Connecting to database to read records...
Ticket ID : 20 closed successfully!


File Admin Tickets
ticket_id | ticket_issuer | ticket_descrip... | start_date | end_date
3 | dfgdsfg | UPDATED TI... | 2021-11-27 1... | 2021-11-27 1...
6 | hehe | lol | 2021-11-27 2...
9 | joe user | help me | 2021-11-27 2...
10 | joe user | need help!! | 2021-11-27 2...
17 | grace | helpppp | 2021-11-28 1...
18 | hehehe | yo | 2021-11-28 1...
19 | joe | yo | 2021-11-28 1...
20 | Grace Sopha | help please | 2021-11-29 1... | 2021-11-29 1...
21 | joe user | hello | 2021-11-29 1... | 2021-11-29 1...
Login [Java Application] C:\Program Files\Java\jdk-14.0.2\bin\javaw.exe (Nov 29, 202
Table 'gsoph_tickets1' already exists
Could not create tables in given database
Inserts completed in the given database...
Connecting to database to read records...