

## Laboratory 6

Title of the Laboratory Exercise: interface to the system

### 1. Introduction and Purpose of Experiment

A database connection is the means by which a database server and its client software communicate with each other. The client and the server can be on the same machine or on different machines. The client uses a database connection to send commands to and receive replies from the server. A database is stored as a file or a set of files on magnetic disk or tape, optical disk, or some other secondary storage device. By doing this lab, students will be able to connect the developed application with the database.

### 2. Aim and Objectives

Aim

- To design an interface and connect to the database

Objectives

At the end of this lab, the student will be able to

- Design and implement an interface for the application
- Connect the developed application with the database

### 3. Experimental Procedure

- Analyse the problem statement
- Design an interface for the given problem statement
- Connect the application with the database
- Test the implemented program
- Document the Results
- Analyse and discuss the outcomes of your experiment

### 4. Questions

Consider the problem statement that you selected in Laboratory 2. Design a GUI with operations such as insert, delete, update and display of record(s) in the relational schema. Use appropriate components to display the page(s).

## 5. Calculations/Computations/Algorithms

**Note : Add this Jar files to your project**

1. JDBC (For Making a Database Connection)
2. javax.swing.JFrame (To Use JFrame)

### Java Program

#### Global Objects (Home.java)

```
You, 2 hours ago | 1 author (You)
public class Home extends javax.swing.JFrame {
    private static final long serialVersionUID = 1L;
    public Home() {
        initComponents();
        viewtable();
    }
    Connection con;
    PreparedStatement pst;
    ResultSet rs;
```

Figure 1 Java Program with Swing JFrame and Global Objects

#### ViewTable Function (Home.java)

```
public void viewtable() {
    int c;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab_6", "root", "Sri123");
        pst = con.prepareStatement("select * from customer");
        rs = pst.executeQuery();

        ResultSetMetaData rsd = rs.getMetaData();
        c = rsd.getColumnCount();
        DefaultTableModel dft = (DefaultTableModel) jTable1.getModel();
        dft.setRowCount(0);

        while (rs.next()) {
            Vector v2 = new Vector();
            for (int i = 1; i <= c; i++) {
                v2.add(rs.getString("Customer_ID"));
                v2.add(rs.getString("Name"));
                v2.add(rs.getString("Phone_Number"));
            }
            dft.addRow(v2);
        }
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

Figure 2 Java Program for Viewtable() Function into a table frame

**Book Button (Home.java)**

```

private void Book_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Book_ButtonActionPerformed
    // Book Button Logic
    String Name;
    String Phone_Number;
    Name = Customer_Name_Input.getText();
    Phone_Number = Customer_Phone_Input.getText();
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab_6", "root", "Sri123");
        pst = con.prepareStatement("insert into customer(Name,Phone_Number) values (?,?)");
        pst.setString(1, Name);
        pst.setString(2, Phone_Number);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Data Has Been Entered Sucessfully");
        Customer_Name_Input.setText("");
        Customer_Phone_Input.setText("");
        Customer_Name_Input.requestFocus();
        viewtable();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(Home.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Figure 3 Java Program for Book Button (Action)

**Edit Button (Home.java)**

```

private void Edit_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Edit_ButtonActionPerformed
    Edit ed = new Edit();
    ed.setVisible(true);
} //GEN-LAST:event_Edit_ButtonActionPerformed

```

Figure 4 Java Program for Edit Button (Action)

**Global Objects (Edit.java)**

You, 2 hours ago | 1 author (You)

```

public class Edit extends javax.swing.JFrame {

    public Edit() {
        initComponents();
        viewtable();
    }

    Connection con;
    PreparedStatement pst;
    ResultSet rs;
}

```

Figure 5 Java Program with Swing JFrame and Global Objects

**Edit Button (Edit.java)**

```

private void Edit_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Book_ButtonActionPerformed
    // Edit Button Logic

    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    int index = jTable1.getSelectedRow();
    int Customer_ID = Integer.parseInt(model.getValueAt(index, 0).toString());
    String Name;
    String Phone_Number;
    Name = Customer_Name_Input.getText();
    Phone_Number = Customer_Phone_Input.getText();
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab_6", "root", "Sri123");
        pst = con.prepareStatement("update customer set Name = ?,Phone_Number=? where Customer_ID = ?");
        pst.setString(1, Name);
        pst.setString(2, Phone_Number);
        pst.setInt(3, Customer_ID);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Data Has Been Updated Sucessfully");
        Customer_Name_Input.setText("");
        Customer_Phone_Input.setText("");
        Customer_Name_Input.requestFocus();
        viewtable();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Edit.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(Edit.class.getName()).log(Level.SEVERE, null, ex);
    }
} //GEN-LAST:event_Book_ButtonActionPerformed

```

Figure 6 Java Program for Edit Button (Action)

**Delete Button (Edit.java)**

```

private void Delete_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Delete_ButtonActionPerformed
    try {
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        int index = jTable1.getSelectedRow();
        int Customer_ID = Integer.parseInt(model.getValueAt(index, 0).toString());
        int dialogresult = JOptionPane.showConfirmDialog(null, "Are you sure that you want to delete the data ?", "Warning", JOptionPane.YES_NO_OPTION);
        if (dialogresult == JOptionPane.YES_NO_OPTION) {
            Class.forName("com.mysql.cj.jdbc.Driver");
        }
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/lab_6", "root", "Sri123");
        pst = con.prepareStatement("delete from customer where Customer_ID = ?");
        pst.setInt(1, Customer_ID);
        pst.executeUpdate();
        JOptionPane.showMessageDialog(this, "Data Has Been Deleted Sucessfully");
        viewtable();
        Customer_Name_Input.setText("");
        Customer_Phone_Input.setText("");
        Customer_Name_Input.requestFocus();
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(Edit.class.getName()).log(Level.SEVERE, null, ex);
    } catch (SQLException ex) {
        Logger.getLogger(Edit.class.getName()).log(Level.SEVERE, null, ex);
    }
} //GEN-LAST:event_Delete_ButtonActionPerformed

```

Figure 7 Java Program for Delete Button (Action)

**On Click for Table (Edit.java)**

```
private void jTable1MouseClicked(java.awt.event.MouseEvent evt) {GEN-FIRST:event_jTable1MouseClicked
    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    int index = jTable1.getSelectedRow();
    Customer_Name_Input.setText(model.getValueAt(index, 1).toString());
    Customer_Phone_Input.setText(model.getValueAt(index, 2).toString());
}GEN-LAST:event_jTable1MouseClicked
```

Figure 8 Java Program for Selecting a row with index number (Click Action)

**Back Button (Edit.java)**

```
private void Back_ButtonActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_Back_ButtonActionPerformed
    Home hom = new Home();
    hom.setVisible(true);
}GEN-LAST:event_Back_ButtonActionPerformed
```

Figure 9 Java Program for Back Button (Action)

**SQL Query****To Create Customer Table from Lab 3 ER Diagram (Updated)**

```
>>create table Customer(Customer_ID int auto_increment,Name
varchar(40) not null,Phone_Number bigint not null,primary
key(Customer_ID));
```

**Metadata for Customer Table**

```
Database changed
mysql> desc customer;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra      |
+-----+-----+-----+-----+-----+-----+
| Customer_ID | int       | NO   | PRI | NULL    | auto_increment |
| Name        | varchar(40) | NO   |     | NULL    |              |
| Phone_Number | bigint    | NO   |     | NULL    |              |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)
```

Figure 10 MySQL Metadata for Table "Customer"

## 6. Presentation of Results

### Scenario 1 ( Insert )

#### UI/JFrame

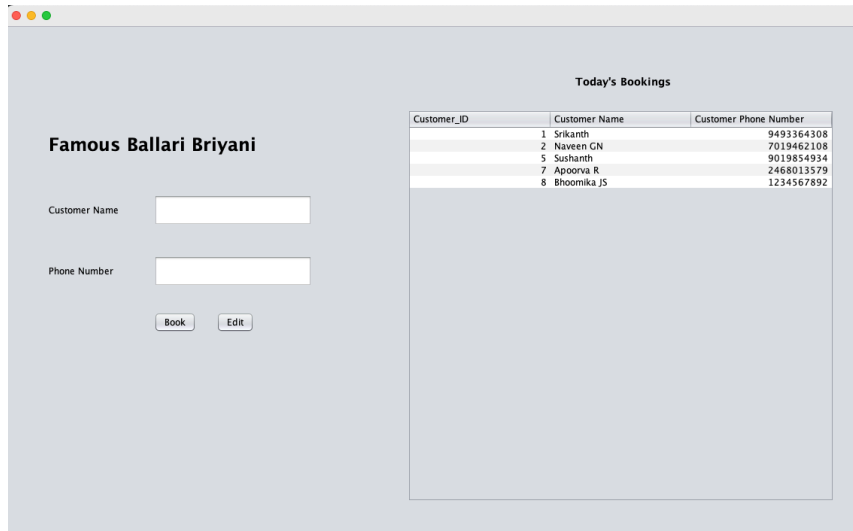


Figure 11 Swing UI using Java (Home Page)

This is the basic UI layout (Home Page) where we can see that it has **two text fields** where the user can enter his/her **name and phone number** and two buttons which are **Book** which adds data to the list on the side and when you click on **Edit** which takes you to the next page or JFrame where you will see this **UI Layout ( Image 15 )**

Now let's enter the data into the Text Fields. Our **Customer Name is "Kushgra Gandhi"** and his **Phone Number is "56793210101"** and hit **"Book"** which adds the data to the list

**Note : Customer ID is Auto Incremented**



Figure 12 Inserting data into database using Swing UI

Now when we hit **“Book”** button we will get a Pop-up window which says that **“Data Has Been Entered Successfully”**

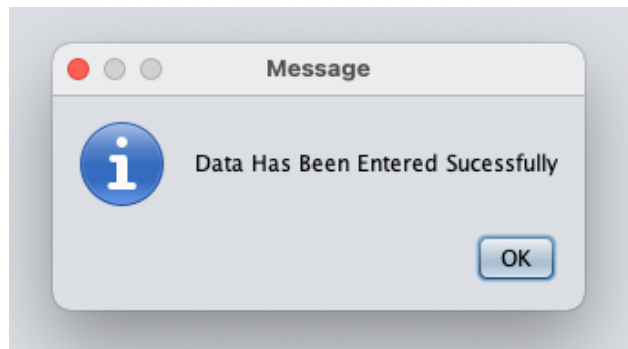


Figure 13 Pop-up window when data is entered

As we can see from image 14 that our data (**Customer Name is “Kushgra Gandhi”** and his **Phone Number is “56793210101”**) has been enter successfully.

Today's Bookings		
Customer_ID	Customer Name	Customer Phone Number
1	Srikanth	9493364308
2	Naveen GN	7019462108
5	Sushanth	9019854934
7	Apoorva R	2468013579
8	Bhoomika JS	1234567892
9	Kushgra Gandhi	56793210101

Figure 14 Updated Table when the data is inserted

- To verify from MySQL terminal to check if the data is entered or not into the database Image 22

## Scenario 2 (Update)

### UI/JFrame

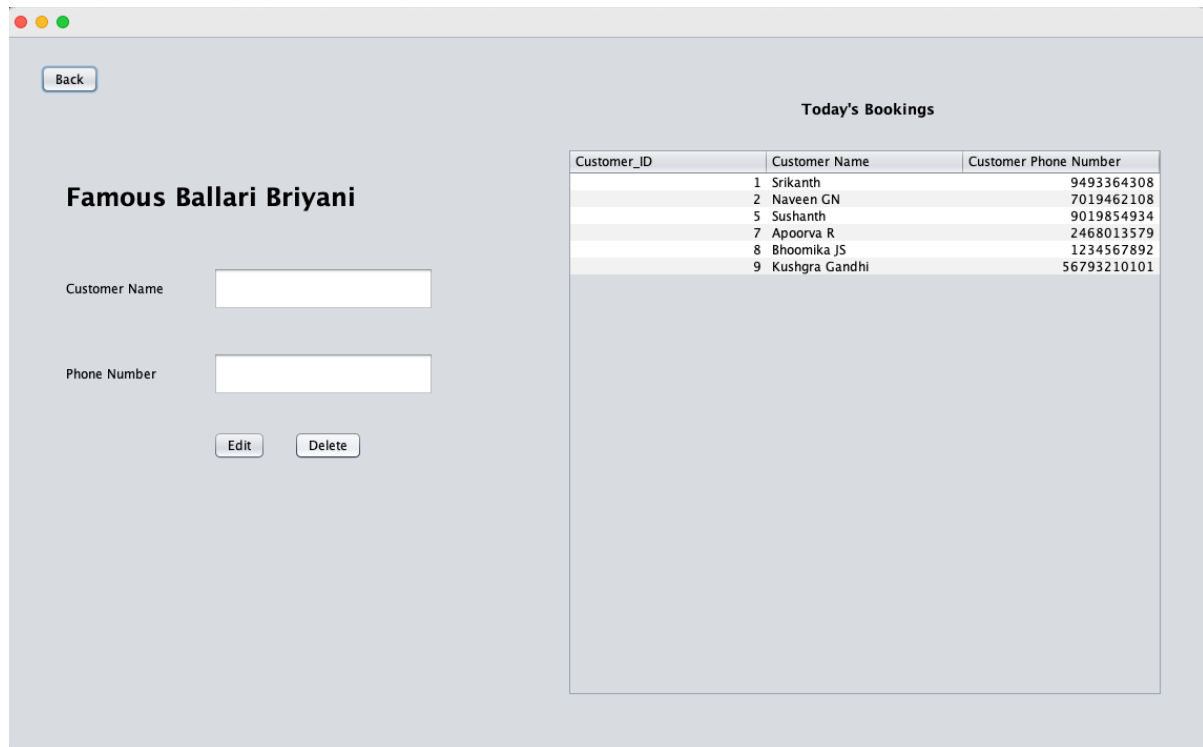


Figure 15 Swing UI using Java (Edit Page)

**This is the UI layout to Edit / Delete the data** where we can see that it have **two text fields** which are **name and phone number** when you select the a row from the table on the side the data is fetched into the text field respectively and there are three buttons

#### 1. Edit

After the user clicks on a row the data is fetched into the text field respectively now the user can edit the data and they can click on the edit button and the data will be update

#### 2. Delete

After the user clicks on a row the data is fetched into the text field respectively now the user can hit delete and the data is deleted

#### 3. Back (on Top)

If the user wants to go back to homepage they can hit **Back button** which will go back to **Home Page**



Now let's update the data for the customer (Customer Name is "Kushgra Gandhi" and his Phone Number is "56793210101"). Lets update the Phone Number to "1234567890" .



The screenshot shows a web application titled "Famous Ballari Briyani". It contains two input fields: "Customer Name" with the value "Kushgra Gandhi" and "Phone Number" with the value "12345678910". Below the fields are two buttons: "Edit" and "Delete". The "Phone Number" field is highlighted with a blue border, indicating it is the focus of the update operation.

Figure 16 Updating Data for the Customer Name is "Kushgra Gandhi"

Now when we hit "Edit" button we will get a Pop-up window which says that "Data Has Been Updated Successfully"

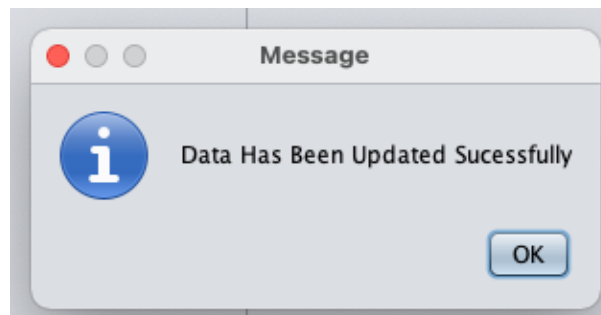


Figure 17 Pop-up window when data is Updated

As we can see from Image 18 that our data (Customer Name is "Kushgra Gandhi" and his Phone Number is "56793210101") has been updated to the Phone Number "1234567890".

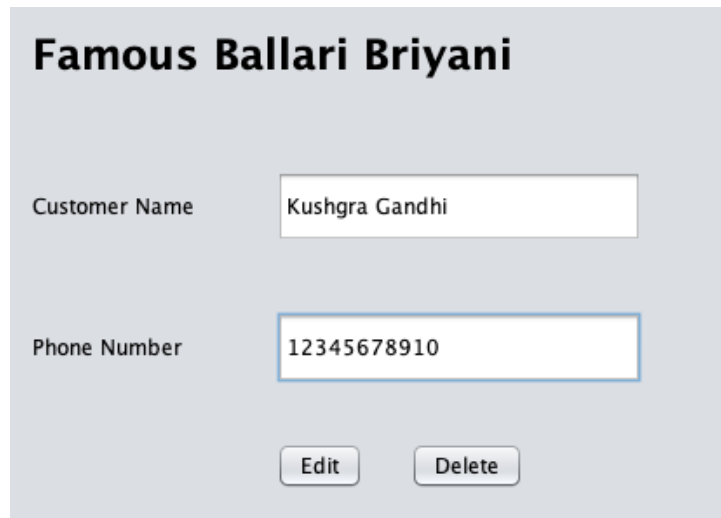
Today's Bookings		
Customer_ID	Customer Name	Customer Phone Number
1	Srikanth	9493364308
2	Naveen GN	7019462108
5	Sushanth	9019854934
7	Apoorva R	2468013579
8	Bhoomika JS	1234567892
9	Kushgra Gandhi	12345678910

Figure 18 Updated Table when the data is updated

- To verify from MySQL terminal to check if the data is Updated or not in the database Image 23

**Scenario 3 (Delete)****From Edit UI Frame**

To delete the data from the table lets select the customer (**Customer Name is “Kushgra Gandhi”** and his **Phone Number is “1234567890”**).



The screenshot shows a web application titled "Famous Ballari Briyani". It has two input fields: "Customer Name" with the value "Kushgra Gandhi" and "Phone Number" with the value "12345678910". Below the fields are two buttons: "Edit" and "Delete".

Figure 19 Deleting Data from the Database where **Customer Name is “Kushgra Gandhi”**

Now when we hit **“Delete”** button we will get a Confirm Pop-up window which says that **“Are you sure that you want to delete the data ?”** if the user wants to delete it then he/she can click on **“Yes”** or if not then **“No”**.

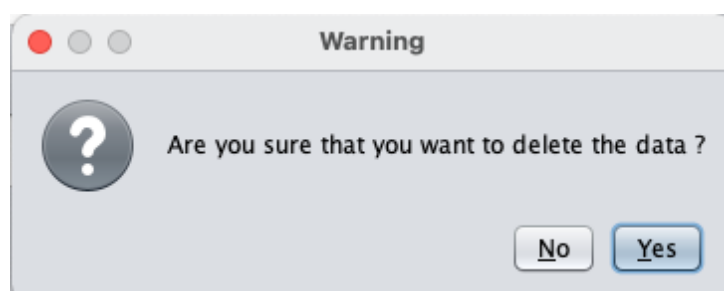


Figure 20 Confirmation Pop-up window when data is being deleted

As we can see from Image 21 that our data (**Customer Name is “Kushgra Gandhi”** and his **Phone Number “1234567890”**) has been deleted.

Today's Bookings		
Customer_ID	Customer Name	Customer Phone Number
1	Srikanth	9493364308
2	Naveen GN	7019462108
5	Sushanth	9019854934
7	Apoorva R	2468013579
8	Bhoomika JS	1234567892

Figure 21 Updated Table when the data is deleted

- To verify from MySQL terminal to check if the data is Deleted or not from the database Image 24

### SQL Query

#### Scenario 1 (Inserted)

```
[mysql> select * from customer;
+-----+-----+-----+
| Customer_ID | Name       | Phone_Number |
+-----+-----+-----+
| 1 | Srikanth   | 9493364308 |
| 2 | Naveen GN  | 7019462108 |
| 5 | Sushanth   | 9019854934 |
| 7 | Apoorva R  | 2468013579 |
| 8 | Bhoomika JS | 1234567892 |
| 9 | Kushgra Gandhi | 56793210101 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Figure 22 MySQL Database Display for Scenario 1

#### Scenario 2 (Updated)

```
Database changed
[mysql> select * from customer;
+-----+-----+-----+
| Customer_ID | Name       | Phone_Number |
+-----+-----+-----+
| 1 | Srikanth   | 9493364308 |
| 2 | Naveen GN  | 7019462108 |
| 5 | Sushanth   | 9019854934 |
| 7 | Apoorva R  | 2468013579 |
| 8 | Bhoomika JS | 1234567892 |
| 9 | Kushgra Gandhi | 12345678910 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

Figure 23 MySQL Database Display for Scenario 2

**Scenario 3 (Deleted)**

```
[mysql> select * from customer;
```

Customer_ID	Name	Phone_Number
1	Srikanth	9493364308
2	Naveen GN	7019462108
5	Sushanth	9019854934
7	Apoorva R	2468013579
8	Bhoomika JS	1234567892

```
5 rows in set (0.00 sec)
```

Figure 24 MySQL Database Display for Scenario 3

**7. Conclusions**

Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as **JFrame**,  **JButton**,  **JTextField**,  **JTextArea**,  **JRadioButton**,  **JCheckbox**,  **JMenu**,  **JColorChooser** and lot more.

Ps: The Restaurant Name seemed nice as i had a good memory about it and is intentionally written 😊

**8. Comments****Learning happened**

Learned About Java Swing and how we connect it to database using JDBC.