

## **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**

- i. Analyse the problem statement
- ii. Design an interface for the given problem statement
- iii. Connect the application with the database
- iv. Test the implemented program
- v. Document the Results
- vi. Analyse and discuss the outcomes of your experiment

#### **4. Questions**

- a. Consider the problem statement that you selected in Laboratory 2. Design a GUI with provision for insertion, deletion and display of a particular record in the database. Use appropriate components to display the page.

## 5. Calculations/Computations/Algorithms

### SQL QUERIES

```
1  drop table if exists client_details;
2
3  create table client_details (
4      client_id Char,
5      client_name char,
6      client_phno char,
7      client_Mgr_id char,
8      client_address char);
9
10
```

### Clientdetails.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Deepak R
 */
public class Clientdetails extends javax.swing.JFrame {
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;

    /**
     * Creates new form Clientdetails
     */
    public Clientdetails() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        client_name = new javax.swing.JTextField();
        Client_name = new javax.swing.JLabel();
        Client_ID = new javax.swing.JLabel();
        client_id = new javax.swing.JTextField();
        Client_Address = new javax.swing.JLabel();
        client_address = new javax.swing.JTextField();
        Client_Phno = new javax.swing.JLabel();
        client_Phno = new javax.swing.JTextField();
        Client_Mgr_Id = new javax.swing.JLabel();
        client_Mgr_id = new javax.swing.JTextField();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        Insert = new javax.swing.JButton();
        ViewData = new javax.swing.JButton();
        Clear = new javax.swing.JButton();

    }
}
```

```
Exit = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

Client_name.setText("client_name:");

Client_ID.setText("Client_ID: ");

client_id.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        client_idActionPerformed(evt);
    }
});

Client_Address.setText("Client_Address:");

client_address.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        client_addressActionPerformed(evt);
    }
});

Client_Phno.setText("Client_Phno:");

client_Phno.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        client_PhnoActionPerformed(evt);
    }
});

Client_Mgr_Id.setText("Client_Mgr_id:");

client_Mgr_id.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        client_Mgr_idActionPerformed(evt);
    }
});

JTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null},
        {null, null, null, null, null}
    },
    new String [] {
        "Client_ID", "Client_Name", "Client_Phno", "Client_Address", "Client_Mgr_Id"
    }
) {
    Class[] types = new Class [] {
        java.lang.String.class, java.lang.String.class, java.lang.String.class, java.lang.String.class, java.lang.String.class
    };

    public Class getColumnClass(int columnIndex) {
        return types [columnIndex];
    }
});

JScrollPane1.setViewportViewView(JTable1);

Insert.setText("Insert");

Insert.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        InsertActionPerformed(evt);
    }
});
```

```
    }  
    });  
  
    ViewData.setText("View Data");  
    ViewData.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            ViewDataActionPerformed(evt);  
        }  
    });  
  
    Clear.setText("Clear");  
    Clear.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            ClearActionPerformed(evt);  
        }  
    });  
  
    Exit.setText("Exit");  
    Exit.addActionListener(new java.awt.event.ActionListener() {  
        public void actionPerformed(java.awt.event.ActionEvent evt) {  
            ExitActionPerformed(evt);  
        }  
    });  
  
    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());  
    getContentPane().setLayout(layout);  
    layout.setHorizontalGroup(  
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
            .addGroup(layout.createSequentialGroup()  
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                    .addGroup(layout.createSequentialGroup()  
                        .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                            .add(layout.createSequentialGroup()  
                                .addContainerGap()  
                                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,  
javax.swing.GroupLayout.PREFERRED_SIZE))  
                            .add(layout.createSequentialGroup()  
                                .addGap(39, 39, 39)  
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)  
                                    .addComponent(Exit)  
                                    .addGroup(layout.createSequentialGroup()  
                                        .addComponent(Insert)  
                                        .addGap(78, 78, 78)  
                                        .addComponent(ViewData)))  
                                .addGap(82, 82, 82)  
                                .addComponent(Clear))  
                            .add(layout.createSequentialGroup()  
                                .addGap(25, 25, 25)  
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                                    .addComponent(Client_Phno)  
                                    .addComponent(Client_name)  
                                    .addGroup(layout.createSequentialGroup()  
                                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                                            .addComponent(Client_Address)  
                                            .addComponent(Client_Mgr_Id)  
                                            .addComponent(Client_ID))  
                                        .addGap(27, 27, 27)  
                                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)  
                                            .addComponent(client_id, javax.swing.GroupLayout.DEFAULT_SIZE, 240, Short.MAX_VALUE)  
                                            .addComponent(client_name)  
                                            .addComponent(client_Phno)  
                                            .addComponent(client_address)  
                                            .addComponent(client_Mgr_id))))))  
                                .addContainerGap(228, Short.MAX_VALUE))  
                        .addGap(25, 25, 25)  
                        .addComponent(Client_Phno)  
                        .addComponent(Client_name)  
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                            .addGroup(layout.createSequentialGroup()  
                                .addComponent(Client_Address)  
                                .addComponent(Client_Mgr_Id)  
                                .addComponent(Client_ID))  
                            .addGroup(layout.createSequentialGroup()  
                                .addComponent(client_id, javax.swing.GroupLayout.DEFAULT_SIZE, 240, Short.MAX_VALUE)  
                                .addComponent(client_name)  
                                .addComponent(client_Phno)  
                                .addComponent(client_address)  
                                .addComponent(client_Mgr_id))))))  
                .addGap(25, 25, 25)  
                .addComponent(Client_Phno)  
                .addComponent(Client_name)  
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                    .addGroup(layout.createSequentialGroup()  
                        .addComponent(Client_Address)  
                        .addComponent(Client_Mgr_Id)  
                        .addComponent(Client_ID))  
                    .addGroup(layout.createSequentialGroup()  
                        .addComponent(client_id, javax.swing.GroupLayout.DEFAULT_SIZE, 240, Short.MAX_VALUE)  
                        .addComponent(client_name)  
                        .addComponent(client_Phno)  
                        .addComponent(client_address)  
                        .addComponent(client_Mgr_id))))  
            .addGap(25, 25, 25)  
            .addComponent(Client_Phno)  
            .addComponent(Client_name)  
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
                .addGroup(layout.createSequentialGroup()  
                    .addComponent(Client_Address)  
                    .addComponent(Client_Mgr_Id)  
                    .addComponent(Client_ID))  
                .addGroup(layout.createSequentialGroup()  
                    .addComponent(client_id, javax.swing.GroupLayout.DEFAULT_SIZE, 240, Short.MAX_VALUE)  
                    .addComponent(client_name)  
                    .addComponent(client_Phno)  
                    .addComponent(client_address)  
                    .addComponent(client_Mgr_id))))  
    );
```

```

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(15, 15, 15)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Client_ID)
                .addComponent(client_id, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(47, 47, 47)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Client_name)
                .addComponent(client_name, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Client_Phno)
                .addComponent(client_Phno, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(12, 12, 12)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(client_address, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(Client_Address))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Client_Mgr_Id)
                .addComponent(client_Mgr_id, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 168, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(66, 66, 66)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Insert)
                .addComponent(ViewData, javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(Clear))
            .addGap(60, 60, 60)
            .addComponent(Exit)
            .addGap(100, 100, 100))
        );

pack();
} // </editor-fold>

private void ViewDataActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
try {
    Class.forName("com.mysql.jdbc.Driver");
    Connection con = DriverManager.getConnection(
        "jdbc:mysql://localhost/bankclientdetails", "root",
        "root");
    String query = "SELECT * FROM client_details;";
    ResultSet rs = stmt.executeQuery(query);
    while (rs.next()) {
        String Client_Id = rs.getString("Client_Id");
        String Client_Name = rs.getString("Client_name");
        String Client_Phno = rs.getString("Client_Phno");
        String Client_Mgr_id = rs.getString("Client_Mgr_id");
        String Client_address = rs.getString("Client_address");
        model.addRow(
            new Object[] { Client_Id, Client_Name, Client_Phno, Client_Mgr_id, Client_address});
    }
}

```

```
rs.close();
stmt.close();
con.close();
}
catch (Exception e) {
    JOptionPane.showMessageDialog(this,"Error In Connectivity");
}
}

private void InsertActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost/bankclientdetails", "root", "root");
        stmt = con.createStatement();
        String Client_ID = client_id.getText();
        String Client_Name = client_name.getText();
        String Client_Phno = client_Phno.getText();
        String Client_mgr_id = client_Mgr_id.getText();
        String Client_address = client_address.getText();
        String INSERT = "INSERT INTO RECORD VALUES("
            + Client_ID + "," + Client_Name + ","
            + Client_Phno + "," + Client_mgr_id + "," + Client_address
            + ")";
        stmt.executeUpdate(INSERT);
        JOptionPane.showMessageDialog(
            this, "Record Added Successfully");
        Insert.setEnabled(true);
    }
    catch (Exception e) {
        JOptionPane.showMessageDialog(
            this, "Error In Connectivity");
    }
}

private void ExitActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    System.exit(0);
}

private void ClearActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    client_name.setText("");
    client_id.setText("");
    client_address.setText("");
    client_Phno.setText("");
    client_Mgr_id.setText("");
    DefaultTableModel dm = (DefaultTableModel)jTable1.getModel();
    dm.getDataVector().removeAllElements();
    jTable1.repaint();
}

private void client_PhnoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void client_addressActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

private void client_Mgr_idActionPerformed(java.awt.event.ActionEvent evt) {
```

```
// TODO add your handling code here:
}

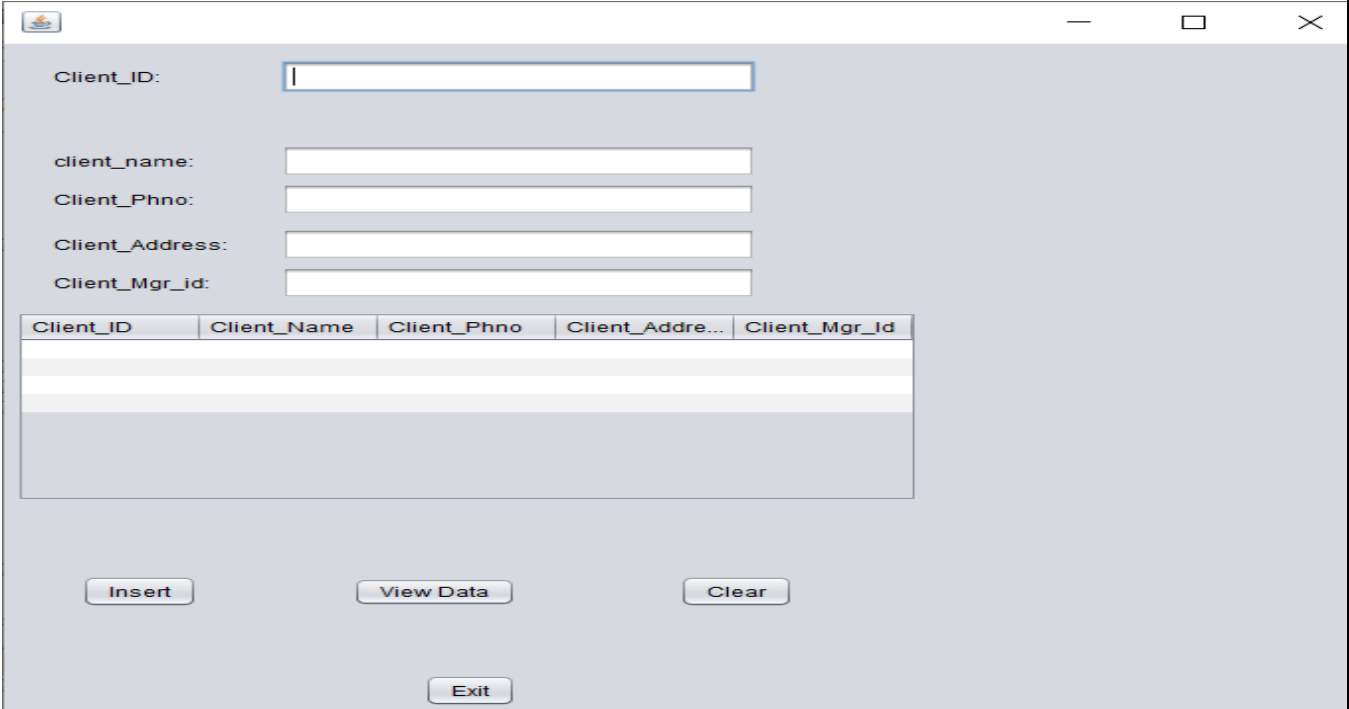
private void client_idActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {
        java.util.logging.Logger.getLogger(Clientdetails.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {
        java.util.logging.Logger.getLogger(Clientdetails.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {
        java.util.logging.Logger.getLogger(Clientdetails.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger(Clientdetails.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
}
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Clientdetails().setVisible(true);
    }
});
}

// Variables declaration - do not modify
private javax.swing.JButton Clear;
private javax.swing.JLabel Client_Address;
private javax.swing.JLabel Client_ID;
private javax.swing.JLabel Client_Mgr_Id;
private javax.swing.JLabel Client_Phno;
private javax.swing.JLabel Client_name;
private javax.swing.JButton Exit;
private javax.swing.JButton Insert;
private javax.swing.JButton ViewData;
private javax.swing.JTextField client_Mgr_id;
private javax.swing.JTextField client_Phno;
private javax.swing.JTextField client_address;
private javax.swing.JTextField client_id;
private javax.swing.JTextField client_name;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}
```

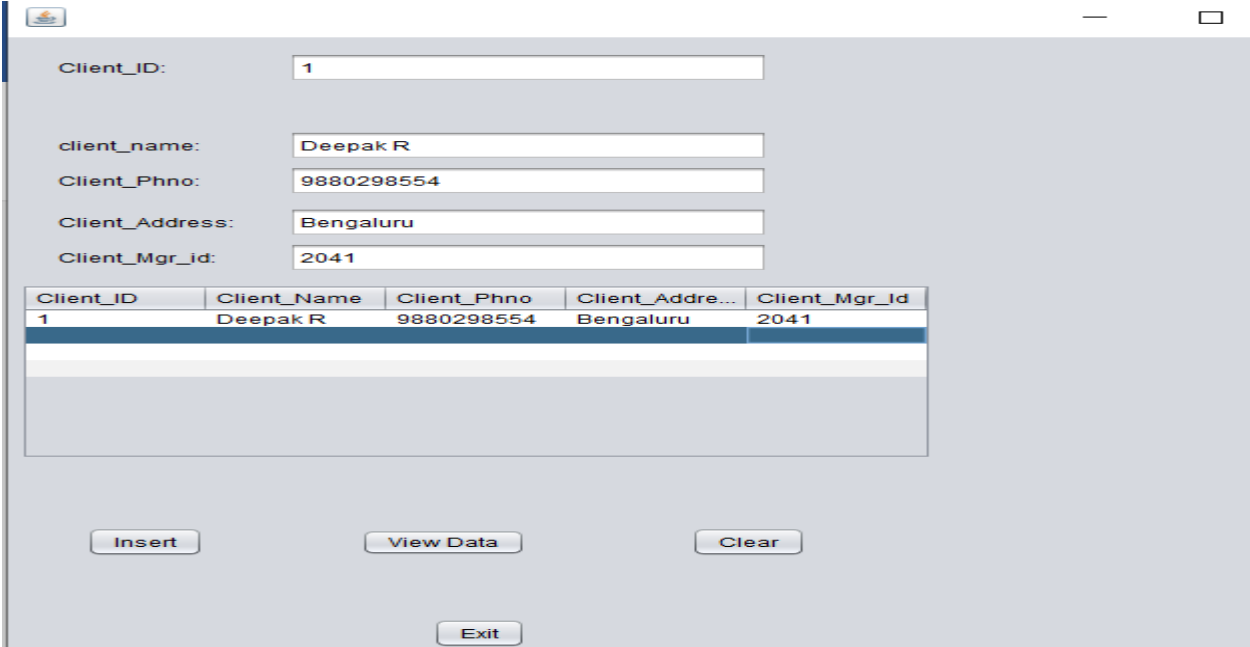
## 6. Presentation of Results



The screenshot shows a GUI window titled "Clientdetails". It contains five input fields for client information: Client\_ID, client\_name, Client\_Phno, Client\_Address, and Client\_Mgr\_id. Below these fields is a table with five columns: Client\_ID, Client\_Name, Client\_Phno, Client\_Addre..., and Client\_Mgr\_Id. The table is currently empty. At the bottom of the window are four buttons: Insert, View Data, Clear, and Exit.

Client_ID	Client_Name	Client_Phno	Client_Addre...	Client_Mgr_Id
-----------	-------------	-------------	-----------------	---------------

*Figure 1 Gui of Clientdetails*



The screenshot shows the same GUI window after an insert operation. The input fields now contain the following data: Client\_ID: 1, client\_name: Deepak R, Client\_Phno: 9880298554, Client\_Address: Bengaluru, and Client\_Mgr\_id: 2041. The table now has one row with the same data. The buttons remain the same.

Client_ID	Client_Name	Client_Phno	Client_Addre...	Client_Mgr_Id
1	Deepak R	9880298554	Bengaluru	2041

*Figure2 Insert operation of Clientdetails*



Client_ID	Client_Name	Client_Phno	Client_Addre...	Client_Mgr_Id
1	Deepak R	9880298554	Bengaluru	2041
2	Virat	9110496542	Delhi	2087

***Figure3 ViewData operation of Clientdetails***

```
run:
BUILD SUCCESSFUL (total time: 4 seconds)
```

**Figure4 Exit operation of Clientdetails**

## **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. The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

The java.sql and javax.sql are the primary packages for JDBC 4.0. This is the latest JDBC version at the time of writing this tutorial. It offers the main classes for interacting with your data sources.

## **8. Comments**

### **1. Limitations of Experiments**

- The database designed needs additional features like hashing for safely storing passwords
- The JDBC connector is not secured, and the transmission of data happened unsecurely

### **2. Limitations of Results**

- Swing components that look like native components might not act exactly like native components
- To develop the application in swing, the individual has to be very careful with programming
- It can be slower than AWT

### **3. Learning happened**

- We learnt how to use Java Swing to create GUI's in Java and connect it to the database using JDBC Driver.

### **4. Recommendations**

- Be careful when using Swing Components as they behave differently in the editor and when run.