

TUGAS PERTEMUAN 2

PRAKTIKUM PBO

Dosen Pengampu : Nurfiah, S.ST, M.Kom



Disusun Oleh :

Yusuf Fatha Mubina Alexander (2211533017)

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

2024

A. Pendahuluan

XAMPP yaitu paket software yang terdiri dari Apache HTTP Server, MySQL, PHP dan Perl yang bersifat open source, xampp biasanya digunakan sebagai development environment dalam pengembangan aplikasi berbasis web secara localhost.

Apache berfungsi sebagai web server yang digunakan untuk menjalankan halaman web, MySQL digunakan untuk manajemen basis data dalam melakukan manipulasi data, PHP digunakan sebagai Bahasa pemrograman untuk membuat aplikasi berbasis web.

MySQL adalah sebuah relational database management system (RDBMS) open-source yang digunakan dalam pengelolaan database suatu aplikasi, MySQL ini dapat digunakan untuk menyimpan, mengelola dan mengambil data dalam format table.

MySQL Connection/j adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Beberapa fungsi MySQL connector yaitu :

- Membuka koneksi ke database MySQL
- Mengirimkan permintaan SQL ke server MySQL
- Menerima hasil dari permintaan SQL
- Menutup koneksi ke database MySQL

DAO (Data Access Object) merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update). Tujuan penggunaan DAO yaitu :

- Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga memudahkan untuk dikelola
- Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
- Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

Interface dalam Bahasa java yaitu mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

CRUD (Create, Read, Update, Delete) merupakan fungsi dasar atau umum yang ada pada sebuah aplikasi yang mana fungsi ini dapat membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

B. Tujuan

Tujuan dari praktikum ini adalah sebagai berikut:

1. Mahasiswa mampu membuat table user pada database MySQL
2. Mahasiswa mampu membuat koneksi Java dengan database MySQL
3. Mahasiswa mampu membuat tampilan GUI CRUD user
4. Mahasiswa mampu membuat dan mengimplementasikan interface

5. Mahasiswa mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikannya.
6. Mahasiswa mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek.

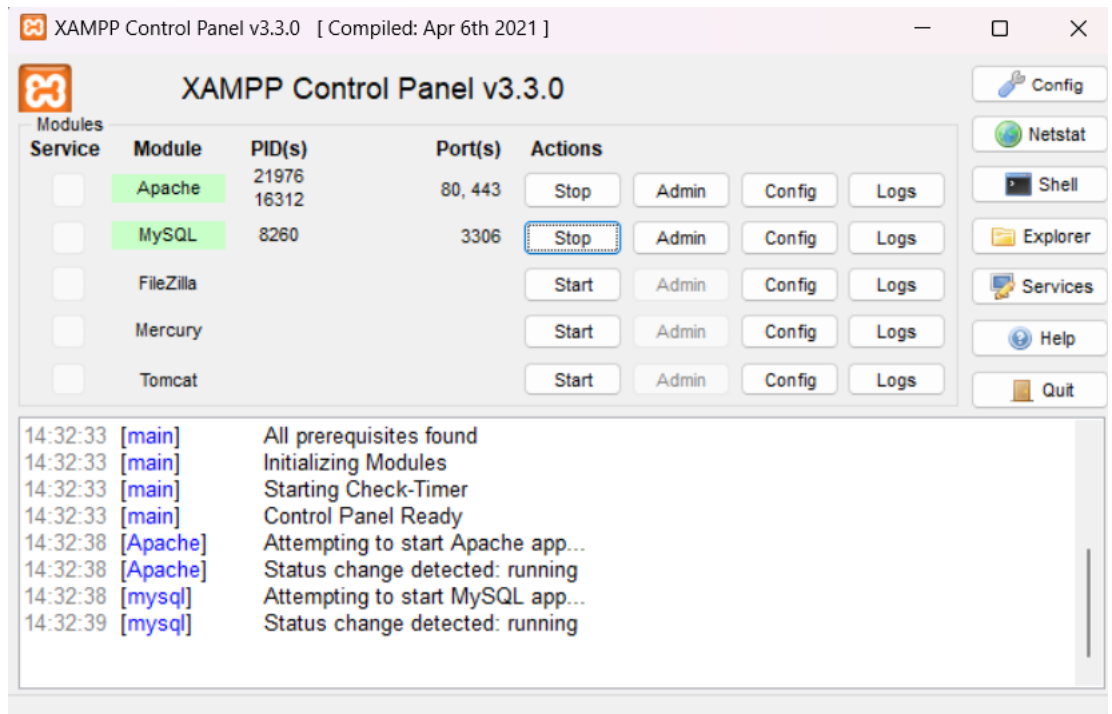
C. Langkah Kerja Praktikum

1. Instalasi XAMPP

Jika masih belum memiliki desktop xampp, unduh melalui website <https://www.apachefriends.org/>

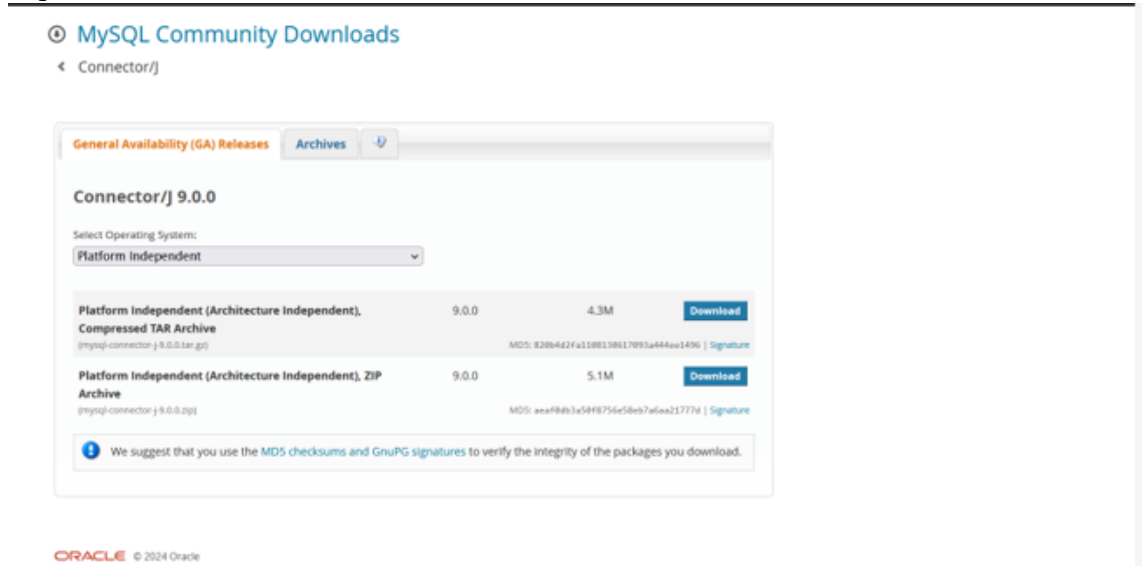


Setelah berhasil melakukan instalasi aplikasi XAMPP, buka aplikasi tersebut lalu aktifkan modul Apache dan MySQL.

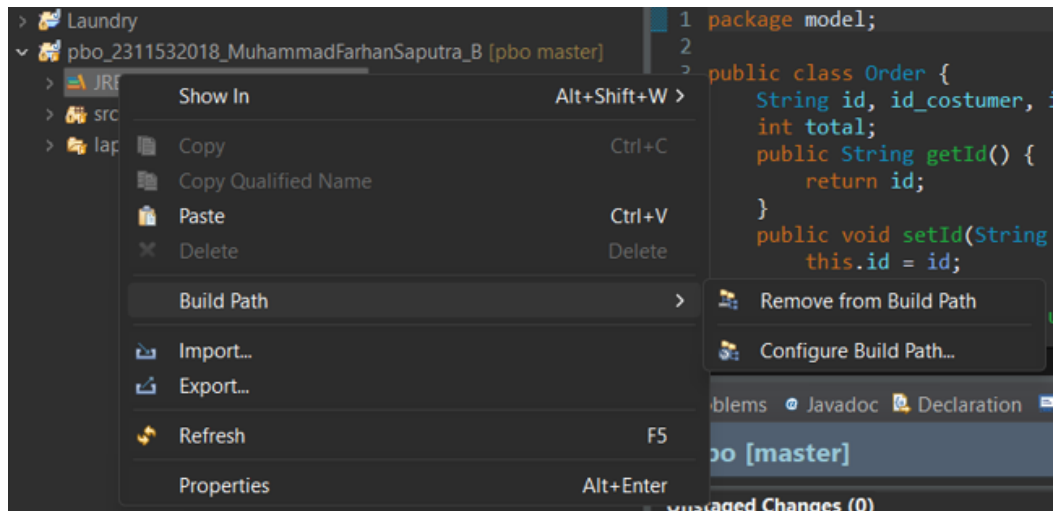


2. Menambahkan MySQL Connector

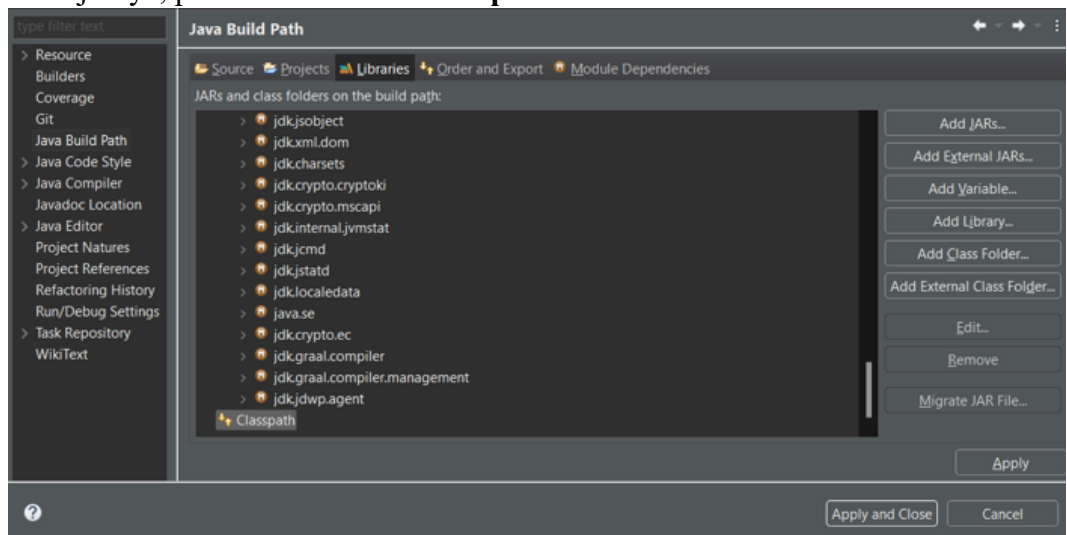
Aplikasi java agar dapat terhubung dengan database MySQL membutuhkan sebuah driver yaitu MySQL Connection, berikut ini Langkah-langkah membuat koneksi Database MySQL. Download MySQL connection melalui website <https://dev.mysql.com/downloads/connector/j/> Pilihlah Operating System Platform Independent, lalu download yang extensi .zip



Setelah didownload file, tambahkan MySQL Connector kedalam project dengan cara klik kanan directory **JRE System Library** → **Built Path** → **Configure Build Path**

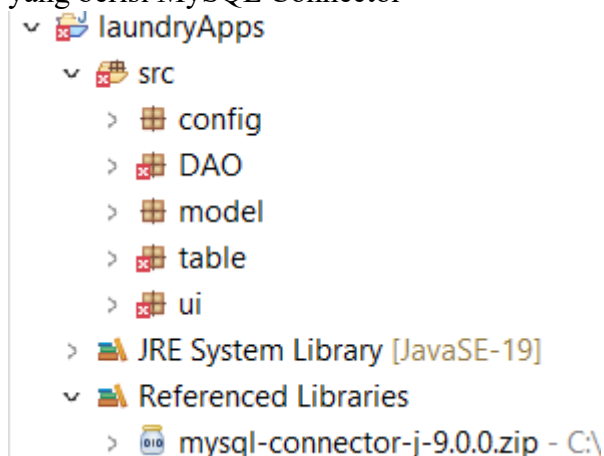


Selanjutnya, pilih **Libraries** → **Classpath**.



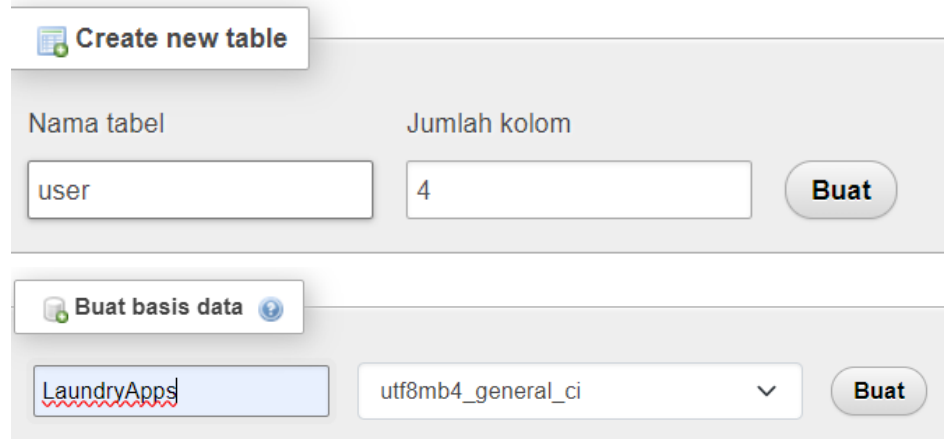
Setelah itu pilih Add External JARs dan pilih file dengan ekstensi .jar yang telah didownload dan pilih Apply and Close.

Jika berhasil, maka akan generate folder Referenced Libraries pada project yang berisi MySQL Connector



3. Membuat Database dan Table User

Buka <http://localhost/phpmyadmin> dan klik new dan buat database dengan nama laundryapps.



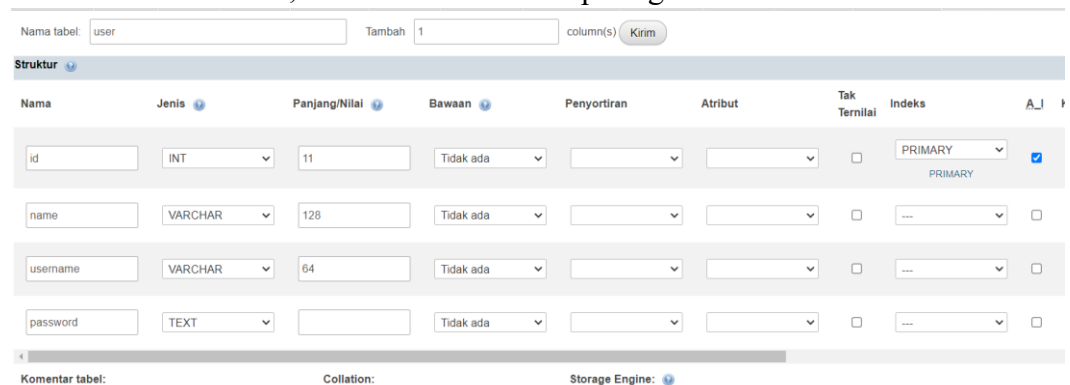
Create new table

Nama tabel: Jumlah kolom: **Buat**

Buat basis data

Buat

Setelah itu klik create, maka akan muncul seperti gambar berikut.



Nama tabel: Tambah column(s) **Kirim**

Struktur

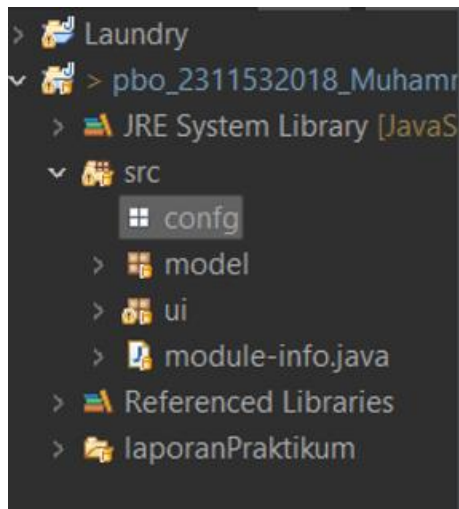
Nama	Jenis	Panjang/Nilai	Bawaan	Penyortiran	Atribut	Tak Ternilai	Indeks	A_I
<input type="text" value="id"/>	<input type="text" value="INT"/>	<input type="text" value="11"/>	<input type="text" value="Tidak ada"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="text" value="PRIMARY"/>	<input checked="" type="checkbox"/>
<input type="text" value="name"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="128"/>	<input type="text" value="Tidak ada"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="text" value=""/>	<input type="checkbox"/>
<input type="text" value="username"/>	<input type="text" value="VARCHAR"/>	<input type="text" value="64"/>	<input type="text" value="Tidak ada"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="text" value=""/>	<input type="checkbox"/>
<input type="text" value="password"/>	<input type="text" value="TEXT"/>	<input type="text" value=""/>	<input type="text" value="Tidak ada"/>	<input type="text" value=""/>	<input type="text" value=""/>	<input type="checkbox"/>	<input type="text" value=""/>	<input type="checkbox"/>

Komentar tabel: Collation: Storage Engine:

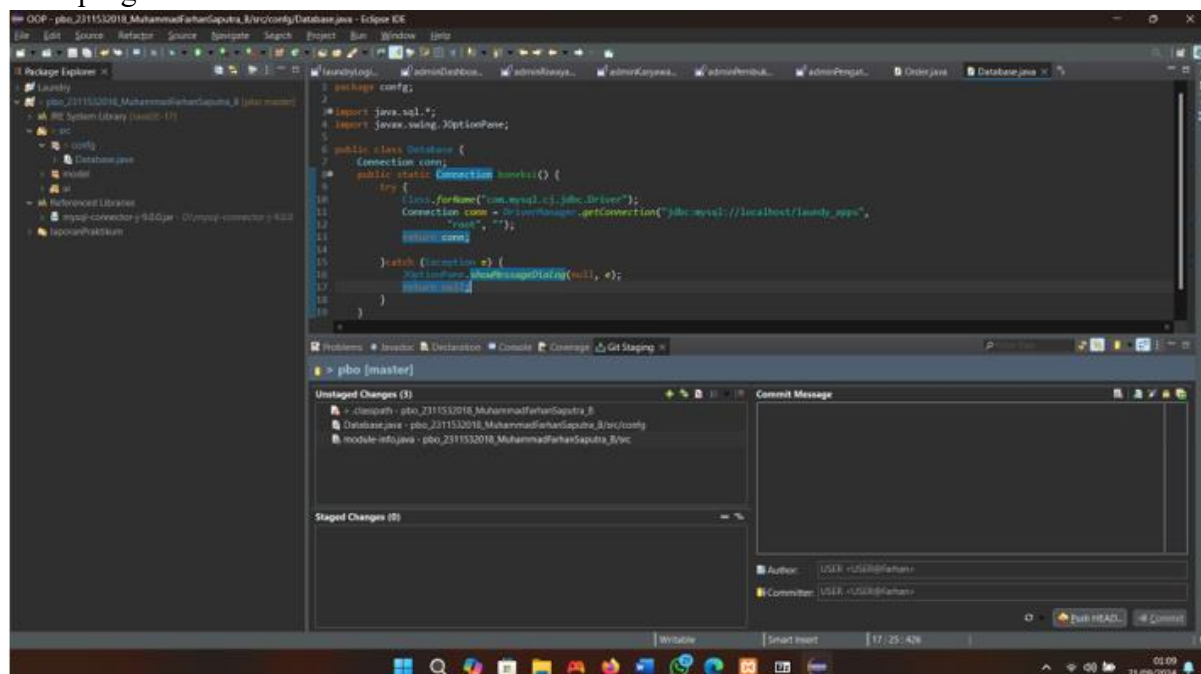
Setelah diisi semua kolom, maka klik save.

4. Koneksi ke Database MySQL

Buat package baru dengan nama config, package ini yang akan digunakan untuk membuat konfigurasi aplikasi yang akan dibuat termasuk dengan konfigurasi database.



Buat class baru dengan nama Database, kemudian konfigurasi sesuai dengan kode program berikut.



Penjelasan Kodingan:

- Import `java.sql.*` digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method `Connection` dengan nama koneksi, yang mana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi database, jika koneksi berhasil maka akan mengembalikan nilai `Connection`
- Line 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan `JOptionPane`.

5. Membuat Tampilan CRUD User

Buat file baru menggunakan `JFrame` pada package `ui` dengan nama `UserFrame` seperti gambar berikut ini.

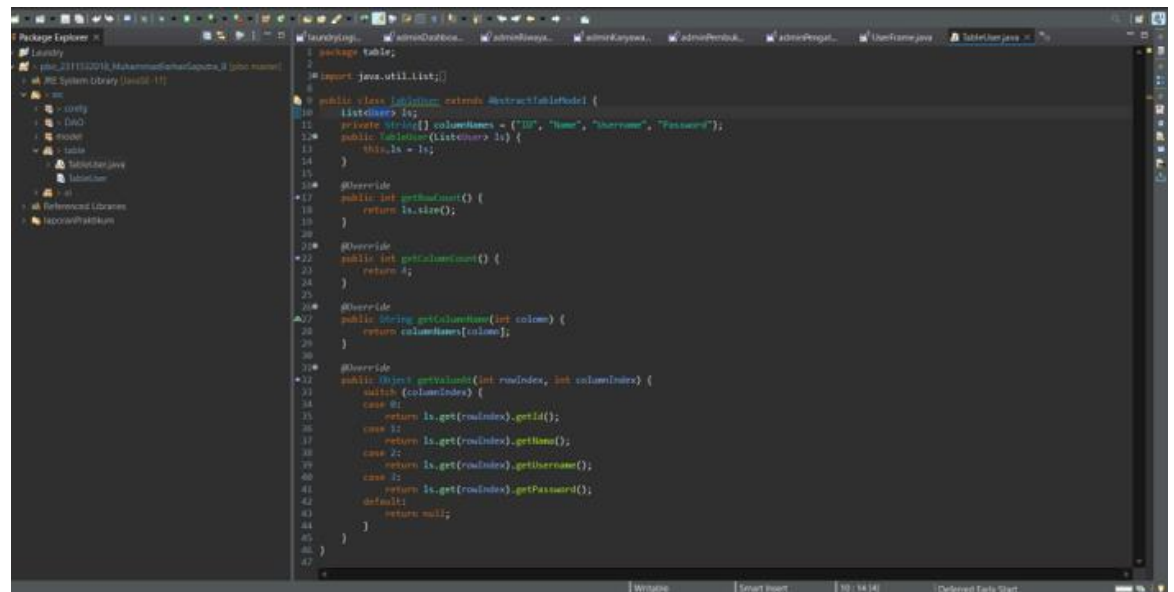
Keterangan Variabel

Component	Variable	Keterangan
TextField	txtName	Name
TextField	txtUsername	Username
TextField	txtPassword	Password
JButton	btnSave	Save
JButton	btnUpdate	Update
JButton	btnDelete	Delete
JButton	BtnCancel	Cancel
JTable	tableUsers	Table Users

6. Membuat Tabel Model

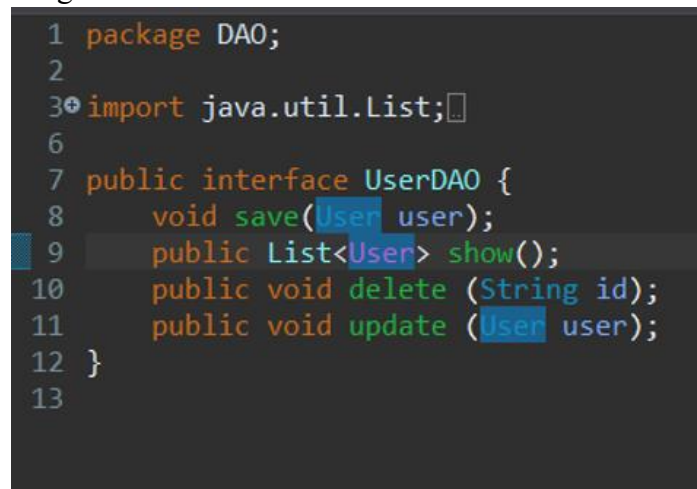
Tabel model user berguna untuk mengambil data dari database dan ditampilkan kedalam table.

Pertama, buat package baru dengan nama table, lalu buat class dengan nama TableUser.



7. Membuat Fungsi DAO

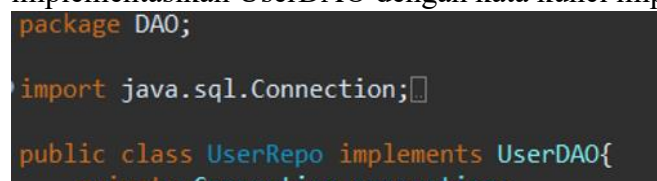
Buat package baru dengan nama DAO, kemudian buat class interface baru dengan nama UserDAO.



Terdapat method save, show, delete dan update. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

8. Menggunakan Fungsi DAO

Buat class baru pada package DAO dengan nama UserRepo yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat, lalu implementasikan UserDAO dengan kata kunci implements.



Membuat instaniasi Connection, membuat constructor dan membuat String untuk melakukan manipulas database.

```
private Connection connection;  
final String insert = "INSERT INTO user (name, username, password) VALUES (?, ?, ?);";  
final String select = "SELECT * FROM user;";  
final String delete = "DELETE FROM user WHERE id = ?;";  
final String update = "UPDATE user SET name = ?, username = ?, password = ? WHERE id = ?;";  
  
public UserRepo() {  
    connection = Database.koneksi();  
}
```

Membuat method save

```
@Override  
public void save(User user) {  
    PreparedStatement st = null;  
    try {  
        st = connection.prepareStatement(insert);  
        st.setString(1, user.getNama());  
        st.setString(2, user.getUsername());  
        st.setString(3, user.getPassword());  
        st.executeUpdate();  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            st.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

Membuat method show untuk mengambil data dari database.

```

public List<User> show(){
    List<User> ls = null;
    try {
        ls = new ArrayList<User>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            User user = new User();
            user.setId(rs.getString("id"));
            user.setNama(rs.getString("name"));
            user.setUsername(rs.getString("username"));
            user.setPassword(rs.getString("password"));
            ls.add(user);
        }
    } catch (SQLException e) {
        Logger.getLogger(UserDAO.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}

```

Membuat method update yang digunakan untuk mengubah data.

```

@Override
public void update (User user) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, user.getNama());
        st.setString(2, user.getUsername());
        st.setString(3, user.getPassword());
        st.setString(4, user.getId());
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            st.close();
        } catch (SQLException e){
            e.printStackTrace();
        }
    }
}

```

Membuat method delete yang digunakan untuk menghapus data

```
public void delete(String id) {  
    PreparedStatement st = null;  
    try {  
        st = connection.prepareStatement(delete);  
        st.setString(1, id);  
        st.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        try {  
            st.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

9. Menggunakan Fungsi CRUD DAO pada GUI.
Buat method reset pada JFrame.

```
public void reset() {  
    txtName.setText("");  
    txtUsername.setText("");  
    txtPassword.setText("");  
}
```

Membuat instance pada UserFrame.

```
UserRepo usr = new UserRepo();  
List<User> ls;  
public String id;
```

10. Membuat Create User

```

JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        usr.save(user);
        reset();
        loadTable();
    }
});
btnSave.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnSave.setBounds(165, 193, 85, 40);
panel.add(btnSave);

```

11. Membuat Read User

Buat method dengan nama loadTable().

```

public void loadTable() {
    ls = usr.show();
    TableUser tu = new TableUser(ls);
    tableUsers.setModel(tu);
    tableUsers.getTableHeader().setVisible(true);
}

```

Lalu memanggil method pada class main.

```

public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                UserFrame frame = new UserFrame();
                frame.setVisible(true);
                frame.loadTable();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

```

12. Membuat Update User

```

tableUsers = new JTable();
tableUsers.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableUsers.getValueAt(tableUsers.getSelectedRow(), 0).toString();
        txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 1).toString());
        txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 2).toString());
        txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 3).toString());
    }
});
scrollPane.setViewportViewView(tableUsers);

```


Buat Tombol Update

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        user.setId(id);
        usr.update(user);
        reset();
        loadTable();
    }
});
btnUpdate.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnUpdate.setBounds(271, 193, 85, 40);
panel.add(btnUpdate);
```

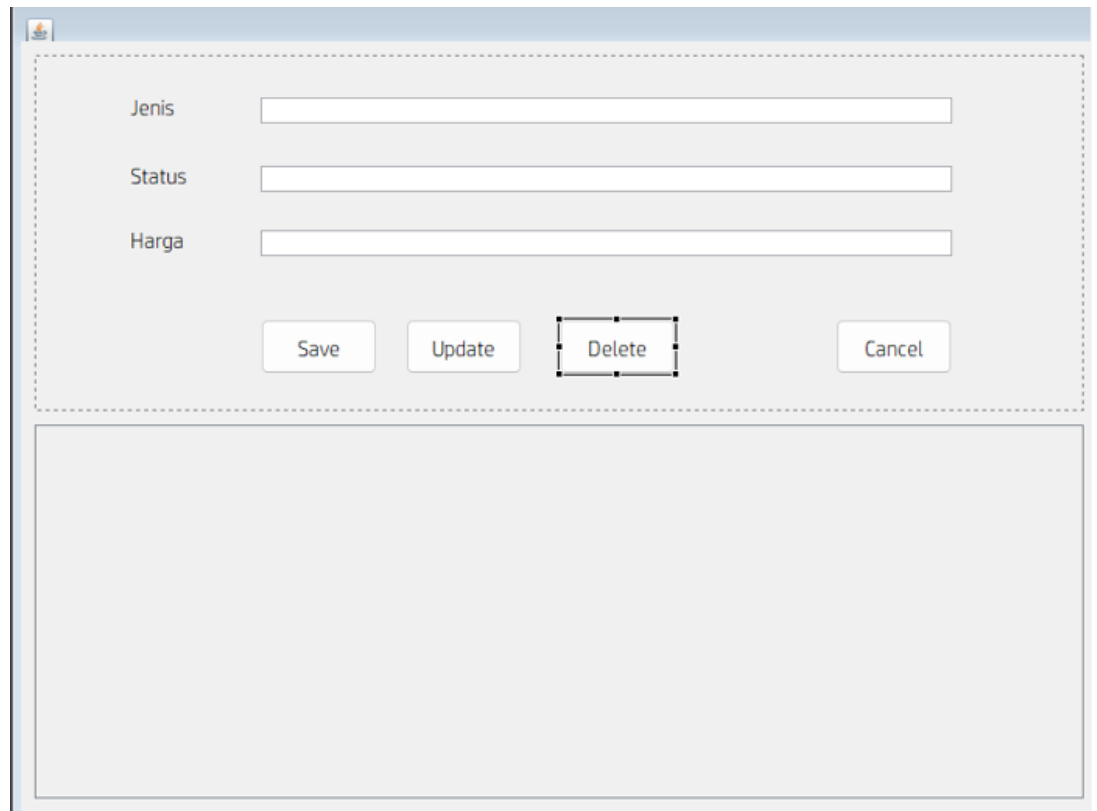
13. Membuat Delete User

```
JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            usr.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
        }
    }
});
btnDelete.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnDelete.setBounds(383, 193, 85, 40);
panel.add(btnDelete);
```

TUGAS

A. Fungsi CRUD untuk Layanan

1. Membuat UI ServiceFrame



2. Membuat ServiceDAO dan ServiceRepo.

```
package DAO;

import java.util.List;

import model.Service;

public interface ServiceDAO {
    void save(Service service);
    public List<Service> show();
    public void delete (String id);
    public void update ([Service service]);
}
```

```

package DAO;

import java.sql.Connection;

public class ServiceRepo implements ServiceDAO{
    private Connection connection;
    final String insert = "INSERT INTO service (jenis, status, harga) VALUES (?, ?, ?);";
    final String select = "SELECT * FROM service;";
    final String delete = "DELETE FROM service WHERE id = ?;";
    final String update = "UPDATE service SET jenis = ?, status = ?, harga = ? WHERE id = ?;";

    public ServiceRepo() {
        connection = Database.koneksi();
    }

    @Override
    public void save(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, service.getJenis());
            st.setString(2, service.getStatus());
            st.setInt(3, service.getHarga());
            st.executeUpdate();

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

    @Override
    public List<Service> show() {
        List<Service> ls = null;
        try {
            ls = new ArrayList<Service>();
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while(rs.next()) {
                Service service = new Service();
                service.setId(rs.getString("id"));
                service.setJenis(rs.getString("jenis"));
                service.setStatus(rs.getString("status"));
                service.setHarga(rs.getInt("harga"));
                ls.add(service);
            }
        } catch (SQLException e) {
            Logger.getLogger(ServiceDAO.class.getName()).log(Level.SEVERE, null, e);
        }
        return ls;
    }

    @Override
    public void delete(String id) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(delete);
            st.setString(1, id);
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```



```

@Override
public void update(Service service) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, service.getJenis());
        st.setString(2, service.getStatus());
        st.setInt(3, service.getHarga());
        st.setString(4, service.getId());
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
}

```

Ada sedikit perubahan di bagian getHarga. Karena jenis variable getHarga adalah integer, maka dibutuhkan setInt untuk mengambil nilai dari table service.

3. Membuat TableService

```

public class TableService extends AbstractTableModel{
    List<Service> ls;
    private String[] columnNames = {"ID", "Jenis", "Status", "Harga"};
    public TableService(List<Service> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 4;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch (columnIndex) {
            case 0:
                return ls.get(rowIndex).getId();
            case 1:
                return ls.get(rowIndex).getJenis();
            case 2:
                return ls.get(rowIndex).getStatus();
            case 3:
                return ls.get(rowIndex).getHarga();
            default:
                return null;
        }
    }
}

```

4. Memasukkan fungsi ServiceDAO kedalam tombol ServiceFrame.

- Tombol Save

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Service service = new Service();
        service.setJenis(txtJenis.getText());
        service.setStatus(txtStatus.getText());
        service.setHarga(Integer.parseInt(txtHarga.getText()));
        srv.save(service);
        reset();
        loadTable();
    }
});
btnSave.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnSave.setBounds(165, 193, 85, 40);
panel.add(btnSave);
```

- Tombol Update

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Service service = new Service();
        service.setJenis(txtJenis.getText());
        service.setStatus(txtStatus.getText());
        service.setHarga(Integer.parseInt(txtHarga.getText()));
        service.setId(id);
        srv.update(service);
        reset();
        loadTable();
    }
});
btnUpdate.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnUpdate.setBounds(271, 193, 85, 40);
panel.add(btnUpdate);
```

- Tombol Delete

```
JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            srv.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
        }
    }
});
btnDelete.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnDelete.setBounds(383, 193, 85, 40);
panel.add(btnDelete);
```

- Membuat table dapat diklik dan masuk kedalam JTextField.

```
tableService = new JTable();
tableService.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableService.getValueAt(tableService.getSelectedRow(), 0).toString();
        txtJenis.setText(tableService.getValueAt(tableService.getSelectedRow(), 1).toString());
        txtStatus.setText(tableService.getValueAt(tableService.getSelectedRow(), 2).toString());
        txtHarga.setText(tableService.getValueAt(tableService.getSelectedRow(), 3).toString());
    }
});
scrollPane.setViewportView(tableService);
```

- Menampilkan table, memasukkan interface, dan reset.

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                ServiceFrame frame = new ServiceFrame();
                frame.setVisible(true);
                frame.loadTable();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

public void reset() {
    txtJenis.setText("");
    txtStatus.setText("");
    txtHarga.setText("");
}

ServiceRepo srv = new ServiceRepo();
List<Service> ls;
public String id;

public void loadTable() {
    ls = srv.show();
    TableService ts = new TableService(ls);
    tableService.setModel(ts);
    tableService.getTableHeader().setVisible(true);
}
}
```

- B. Fungsi CRUD untuk Pelanggan
1. Membuat UI CostumerFrame

Nama Pelanggan
 Alamat
 No. HP
 Save Update Delete Cancel

2. Membuat TableCostumer

```

1 package table;
2
3 import java.util.List;
4
5
6
7
8
9 public class TableCostumer extends AbstractTableModel {
10     List<Costumer> ls;
11     private String[] columnNames = {"ID", "Nama", "Alamat", "Nohp"};
12     public TableCostumer(List<Costumer> ls) {
13         this.ls = ls;
14     }
15
16     @Override
17     public int getRowCount() {
18         return ls.size();
19     }
20
21     @Override
22     public int getColumnCount() {
23         return 4;
24     }
25
26     @Override
27     public String getColumnName(int column) {
28         return columnNames[column];
29     }
30
31     @Override
32     public Object getValueAt(int rowIndex, int columnIndex) {
33         switch (columnIndex) {
34             case 0:
35                 return ls.get(rowIndex).getId();
36             case 1:
37                 return ls.get(rowIndex).getNama();
38             case 2:
39                 return ls.get(rowIndex).getAlamat();
40             case 3:
41                 return ls.get(rowIndex).getNohp();
42             default:
43                 return null;
44         }
45     }
46 }

```

3. Membuat CostumerDAO dan CostumerRepo

```
package DAO;

import java.util.List;

import model.Costumer;

public interface CostumerDAO {
    void save(Costumer costumer);
    public List<Costumer> show();
    public void delete (String id);
    public void update (Costumer costumer);
}
```

```
public class CostumerRepo implements CostumerDAO{
    private Connection connection;
    final String insert = "INSERT INTO costumer (nama, alamat, nohp) VALUES (?, ?, ?);";
    final String select = "SELECT * FROM costumer;";
    final String delete = "DELETE FROM costumer WHERE id = ?;";
    final String update = "UPDATE costumer SET nama = ?, alamat = ?, nohp = ? WHERE id = ?;";

    public CostumerRepo() {
        connection = Database.koneksi();
    }

    @Override
    public void save(Costumer costumer) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, costumer.getNama());
            st.setString(2, costumer.getAlamat());
            st.setString(3, costumer.getNohp());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```



```

public List<Costumer> show(){
    List<Costumer> ls = null;
    try {
        ls = new ArrayList<Costumer>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            Costumer costumer = new Costumer();
            costumer.setId(rs.getString("id"));
            costumer.setNama(rs.getString("nama"));
            costumer.setAlamat(rs.getString("alamat"));
            costumer.setNohp(rs.getString("nohp"));
            ls.add(costumer);
        }
    }catch(SQLException e) {
        Logger.getLogger(CostumerDAO.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}

@Override
public void update (Costumer costumer) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, costumer.getNama());
        st.setString(2, costumer.getAlamat());
        st.setString(3, costumer.getNohp());
        st.setString(4, costumer.getId());
    }catch(SQLException e) {
        e.printStackTrace();
    }finally {
        try {
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }
    }
}
}

```

```

public void delete(String id) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(delete);
        st.setString(1, id);
        st.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();
    }finally {
        try {
            st.close();
        }catch(SQLException e) {
            e.printStackTrace();
        }
    }
}
}

```

4. Membuat Fungsi CRUD di JFrame
 - Tombol Save

```

JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Costumer costumer = new Costumer();
        costumer.setNama(txtNama.getText());
        costumer.setAlamat(txtAlamat.getText());
        costumer.setNohp(txtNohp.getText());
        cst.save(costumer);
        reset();
        loadTable();
    }
});
btnSave.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnSave.setBounds(165, 193, 85, 40);
panel.add(btnSave);

```

- Tombol Update

```

JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Costumer costumer = new Costumer();
        costumer.setNama(txtNama.getText());
        costumer.setAlamat(txtAlamat.getText());
        costumer.setNohp(txtNohp.getText());
        costumer.setId(id);
        cst.update(costumer);
        reset();
        loadTable();
    }
});
btnUpdate.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnUpdate.setBounds(271, 193, 85, 40);
panel.add(btnUpdate);

```

- Tombol Delete

```

JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            cst.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
        }
    }
});
btnDelete.setFont(new Font("HP Simplified Jpan Light", Font.PLAIN, 16));
btnDelete.setBounds(383, 193, 85, 40);
panel.add(btnDelete);

```

- Membuat fungsi klik pada table

```

tableCostumer = new JTable();
tableCostumer.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 0).toString();
        txtNama.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 1).toString());
        txtAlamat.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 2).toString());
        txtNohp.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 3).toString());
    }
});
scrollPane.setViewportView(tableCostumer);

```

- Menampilkan table saat pertama kali run, method reset, dan memasukkan interface.

```

public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                CostumerFrame frame = new CostumerFrame();
                frame.setVisible(true);
                frame.loadTable();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

public void reset() {
    txtNama.setText("");
    txtAlamat.setText("");
    txtNohp.setText("");
}

CostumerRepo cst = new CostumerRepo();
List<Costumer> ls;
public String id;

public void loadTable() {
    ls = cst.show();
    TableCostumer tc = new TableCostumer(ls);
    tableCostumer.setModel(tc);
    tableCostumer.getTableHeader().setVisible(true);
}

```