LAPORAN PRATIKUM PBO

MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete)
USER , LAYANAN DAN PELANGGAN

OLEH:

HABIBI PUTRA RIZQULLAH

2411531001

MATA KULIAH : PEMOGRAMAN BERIOENTASI OBJEK

DOSEN PENGAMPU : NURFIAH, S.ST, M.KOM

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Pemrograman Berorientasi Objek (OOP) adalah cara membuat program komputer yang menggunakan kelas dan objek sebagai komponen utama dalam membangun perangkat lunak. OOP memudahkan pengembang dalam menyusun program secara terstruktur, menjaga keamanan data, serta memanfaatkan kembali kode melalui konsep penting seperti enkapsulasi, pewarisan, dan polimorfisme.

Dalam pengembangan aplikasi berbasis Java, OOP sering dipadukan dengan database untuk mengelola data secara lebih terorganisir. Salah satu pendekatan yang umum digunakan adalah Data Access Object (DAO), yaitu lapisan khusus yang memisahkan logika akses data dari logika bisnis. DAO berfungsi sebagai jembatan antara aplikasi dan database sehingga proses Create, Read, Update, dan Delete (CRUD) dapat dilakukan dengan lebih terstruktur, mudah dikembangkan sesuai dengan prinsip OOP.

B. Tujuan

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Mahasiswa mampu memahami konsep dasar Data Access Object (DAO) dalam pengembangan aplikasi Java berbasis database.

2. Mahasiswa mampu merancang tampilan GUI untuk melakukan operasi CRUD data user dengan menerapkan konsep Pemrograman Berorientasi Objek.

3. Mahasiswa mampu membuat serta mengimplementasikan interface dan fungsi DAO (Data Access Object) untuk mengelola proses CRUD secara terstruktur dan modular.

C. Teori Dasar

1. Object-Oriented Programming (OOP) adalah cara pemrograman yang berfokus pada penggunaan objek sebagai representasi dari entitas dunia nyata. Objek memiliki atribut (data) dan method (fungsi) yang saling berkaitan, sehingga program dapat dibuat lebih terstruktur, terorganisir, dan mudah dikembangkan.

2. Data Access Object (DAO) adalah pola desain yang digunakan untuk memisahkan logika akses data dari logika bisnis aplikasi. Dengan pendekatan ini, operasi seperti menyimpan, mengambil, memperbarui, dan menghapus data dapat dilakukan secara lebih teratur melalui antarmuka khusus.

3. CRUD (Create, Read, Update, Delete) merupakan empat operasi dasar yang digunakan untuk mengelola data dalam aplikasi. Operasi ini menjadi fondasi utama dalam proses pengembangan perangkat lunak yang berhubungan dengan data.

4. Java Database Connectivity (JDBC) adalah API yang memungkinkan aplikasi Java terhubung dengan database. Dengan JDBC, aplikasi dapat menjalankan perintah CRUD secara langsung melalui kode program, sehingga mendukung penerapan OOP dan pola desain DAO dalam pengelolaan data.

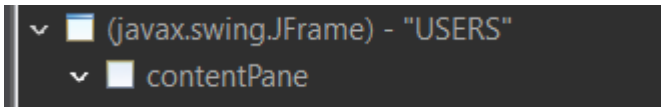
D. Langkah – Langkah Pengerjaan

1. Membuat Tampilan CRUD User

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

- Atur nama JFrame menjadi "USERS" untuk judul pada frame.

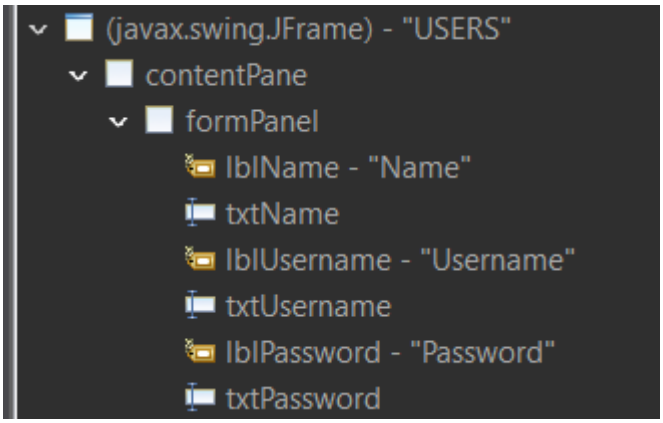
- Di dalam JFrame UserFrame, tambahkan sebuah panel utama dengan nama contentPane sebagai wadah semua komponen.



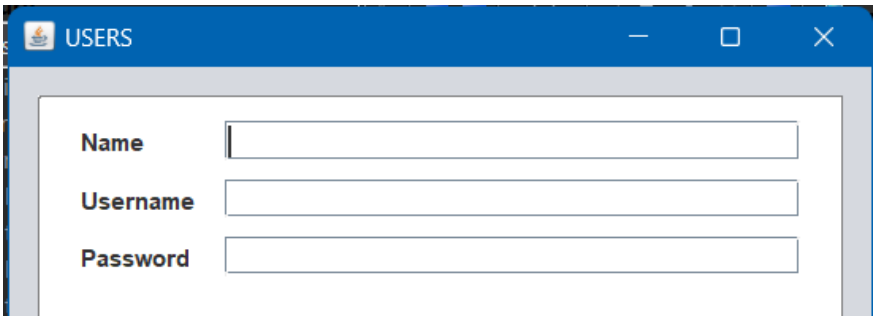
- Tambahkan panel bernama formPanel ke dalam contentPane. Panel ini berfungsi sebagai tempat input data user.

Komponen-komponen yang ditambahkan di formPanel antara lain:

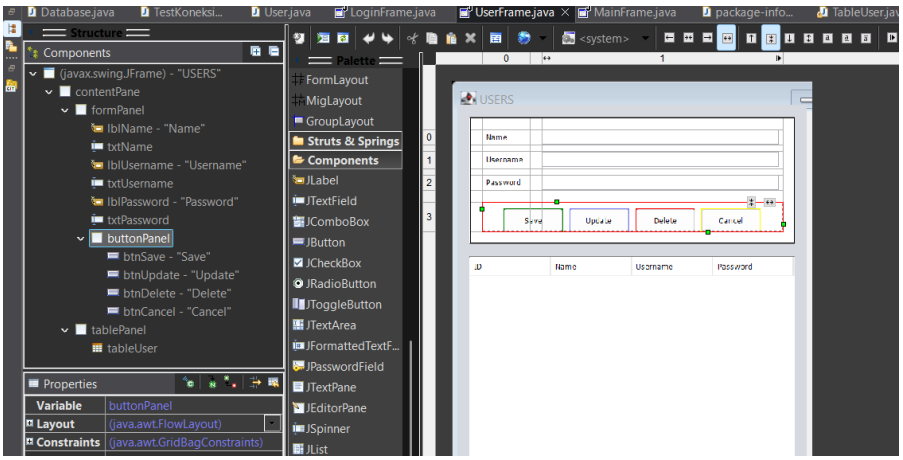
- JLabel lblName dan JTextField txtName untuk menginput nama.
- JLabel lblUsername dan JTextField txtUsername untuk menginput username.
- JLabel lblPassword dan JTextField txtPassword untuk menginput password.



Maka hasilnya akan seperti ini

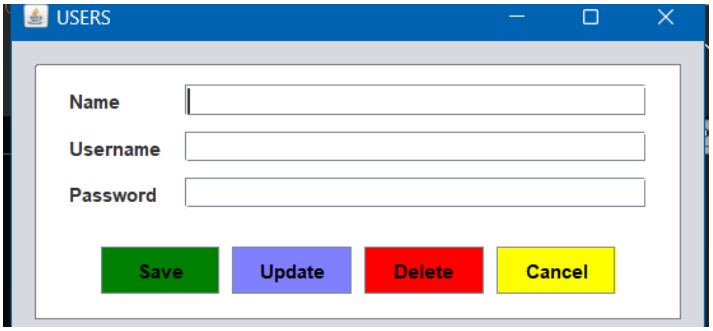


- Selanjutnya tambahkan panel bernama buttonPanel di bawah formPanel. Panel ini berisi 4 buah tombol (JButton) sebagai aksi utama:
  - btnSave untuk menyimpan data baru ke database.
  - btnUpdate untuk memperbarui data yang dipilih dari tabel.
  - btnDelete untuk menghapus data yang dipilih dari tabel.
  - btnCancel untuk membatalkan input atau mengosongkan field.

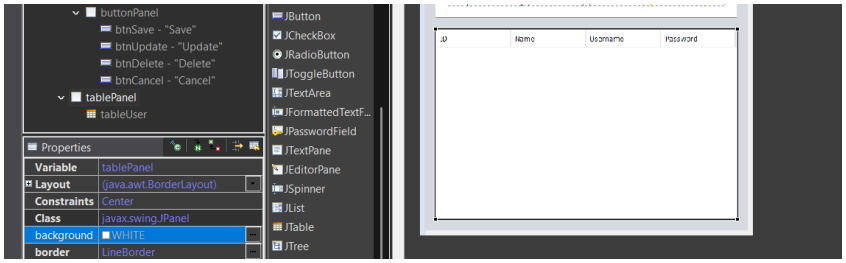


- Selanjutnya masing-masing tombol diatur warnanya untuk membedakan fungsi secara visual. Sebagai contoh warna yang saya gunakan :
  - Hijau → Save (tambah data)
  - Magenta → Update (ubah data)
  - Merah → Delete (hapus data)
  - Kuning → Cancel (reset/batal)

Maka hasilnya seperti ini :



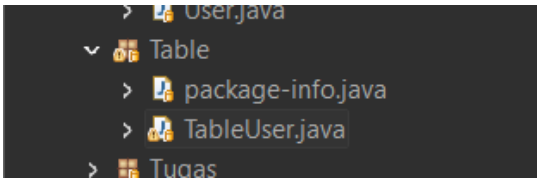
- Tambahkan panel bernama tablePanel di bagian bawah contentPane. Panel ini berisi sebuah JTable bernama tableUser untuk menampilkan data user dalam bentuk tabel.



2. Membuat Table Model

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

- Buat package baru dengan nama table

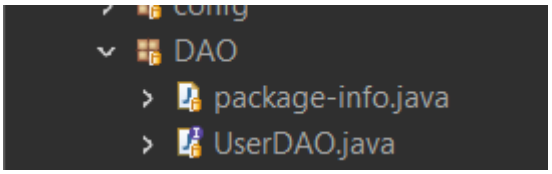


- Buat file baru didalam package table dengan nama TableUser, kemudian isikan dengan kode program berikut.

```java
public class TableUser extends AbstractTableModel {
    private List<User> ls;
    private String[] columnNames = {"ID", "Name", "Username", "Password"};

    public TableUser(List<User> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        return 4;
    }

    @Override
    public String getColumnName(int column) {
        return columnNames[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
            case 0:
                return ls.get(rowIndex).getId();
            case 1:
                return ls.get(rowIndex).getNama();
            case 2:
                return ls.get(rowIndex).getUsername();
            case 3:
                return ls.get(rowIndex).getPassword();
            default:
                return null;
        }
    }
}
```

3. Membuat Fungsi DAO

- Buat package baru dengan nama DAO

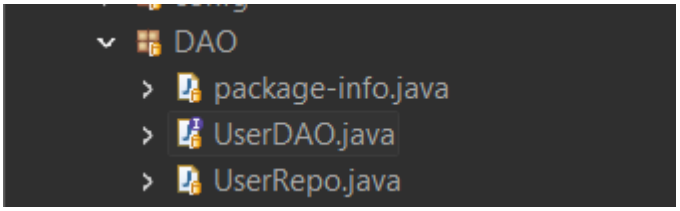



```java
package DAO;

import java.util.List;

import model.User;

public interface UserDAO {
    void save(User user);
    List<User> show();
    void delete(String id);
    void update(User user);
}
```

- Buat class Interface baru dengan nama UserDAO, kemudian isikan dengan kode program berikut. Terdapat method save, show, delete dan update. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

4. Menggunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama UserRepo yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat.



- Implementasikan UserDao dengan kata kunci implements

```
10
11 public class UserRepo implements UserDAO {
12
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database.

```
12
13     private final Connection connection;
14     private static final String INSERT = "INSERT INTO user (name, username, password) VALUES (?,?,?)";
15     private static final String SELECT = "SELECT * FROM user";
16     private static final String UPDATE = "UPDATE user SET name=?, username=?, password=? WHERE id=?";
17     private static final String DELETE = "DELETE FROM user WHERE id=?";
18
19     public UserRepo() {
20         connection = Database.koneksi();
21     }
22
```

- Selanjutnya membuat method save, isikan dengan kode program berikut

```
22
23     @Override
24     public void save(User user) {
25         PreparedStatement st = null;
26         try {
27             st = connection.prepareStatement(INSERT);
28             st.setString(1, user.getNama());
29             st.setString(2, user.getUsername());
30             st.setString(3, user.getPassword());
31             st.executeUpdate();
32
33         } catch (SQLException e) {
34             e.printStackTrace();
35         }finally {
36             try {
37                 st.close();
38             }catch(SQLException e) {
39                 e.printStackTrace();
40             }
41         }
42     }
43
```

- Lalu membuat method show untuk mengambil data dari database

```java
@Override
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;
}
```

- Selanjutnya kita membuat method update yang digunakan untuk mengubah data

```java
@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());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();

    }finally {
        try {
            st.close();

        }catch(SQLException e) {
            e.printStackTrace();
        }
    }
}
```

- Dan terakhir kita Membuat method delete yang digunakan untuk menghapus data.

```java
@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();
        }
    }
}
}
```

5. Menggunakan Fungsi CRUD DAO pada GUI

- Method digunakan untuk menghapus value inputan Ketika suatu proses berhasil dilakukan, buat method reset pada JFrame seperti kode program dibawah ini

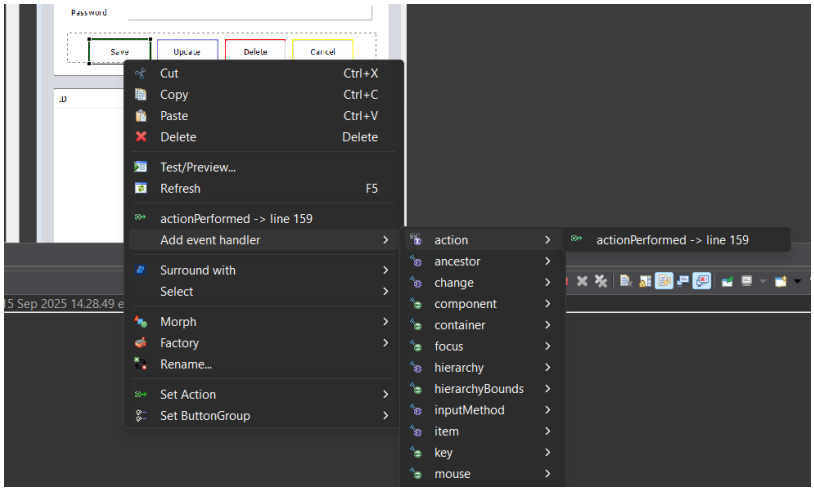
```
271     public TableUser getTableModel() {
272         return tableModel;
273     }
274
275     public void reset() {
276         txtName.setText("");
277         txtUsername.setText("");
278         txtPassword.setText("");
279     }
280
281     public void loadTable() {
```

- Membuat Instance pada UserFrame

```
43
44      private JTable tableUser;
45      private TableUser tableModel;
46
47  // user repo
48      UserRepo usr = new UserRepo() ;
49      List<User> ls;
50      public String id;
51
```

6. Create User

- Klik kanan pada tombol save → add event handlers → actionPerformed kemdian isi dengan kode program berikut.



```
        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();
            }
        });
        btnSave.setPreferredSize(new Dimension(75, 30));
        btnSave.setBackground(new Color(0, 128, 0));
```

7. Read User

- Buat method dengan nama loadTable() kemudian isikan dengan kode program berikut.

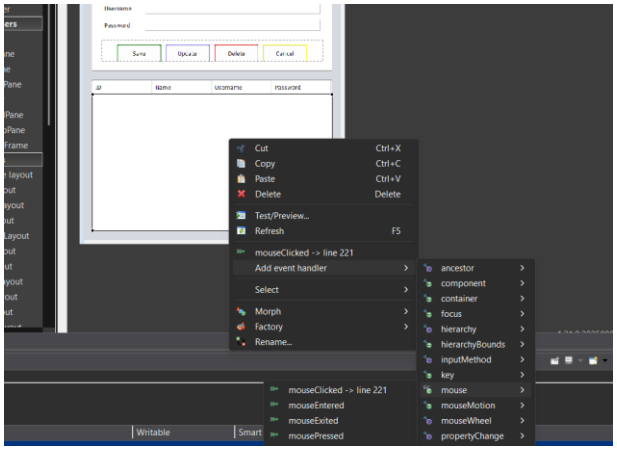
```
280
281     public void loadTable() {
282         ls = usr.show();
283         TableUser tu = new TableUser(ls);
284         tableUser.setModel(tu);
285         tableUser.getTableHeader().setVisible(true);
286
287     }
288 }
```

- Selanjutnya memanggil method pada class main, sehingga ketika pertama kali program LoadTable akan dipanggil.

```
        UserFrame frame = new UserFrame();
        frame.setVisible(true);
        frame.loadTable();
```

- Kode tersebut berfungsi untuk membuat objek UserFrame, menampilkan jendela tersebut, lalu memanggil method loadTable() agar data langsung ditampilkan ke dalam tabel saat program dijalankan. Jadi alurnya adalah: ketika objek UserFrame dibuat, frame ditampilkan ke layar menggunakan setVisible(true), kemudian method loadTable() dipanggil untuk mengambil data dari database dan mengisinya ke tabel pada form. Dengan cara ini, pengguna langsung melihat data yang sudah ter-load ketika aplikasi dibuka.

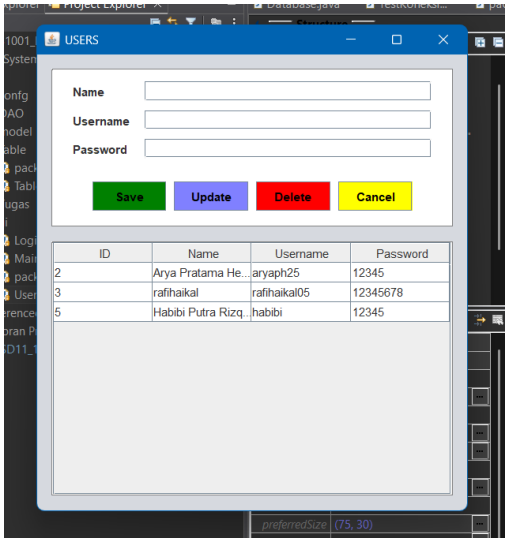
- Lalu tambahkan EventHandler pada JTable yang diset untuk **mouseclicked.** Di dalam eventhandler, **assign** data dari baris yang dipilih oleh penggguna pada tabel **tableUser.**



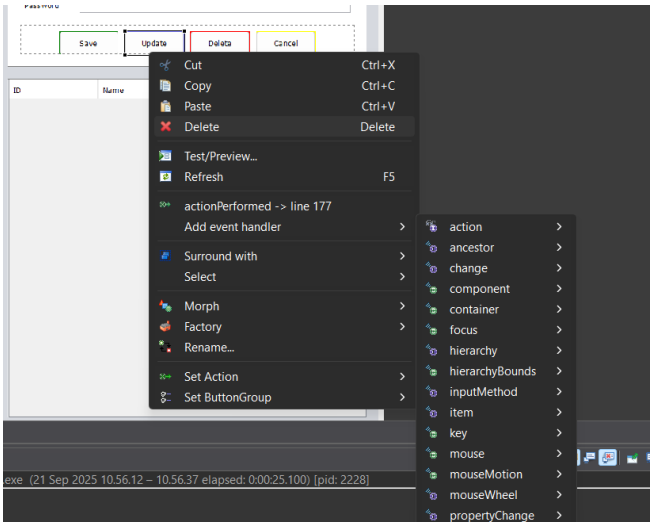
- Kemudian, data dari masing-masing kolom Jtable pada setiap JTextField dengan menggunakan **method** setText() dimana data setiap kolom diambil dari baris yang ditekan, dan pada **method** getValueAt() ditentukan index yang dari kolom yang dipilih.

```
    public void mouseClicked(MouseEvent e) {
        id = tableUser.getValueAt(tableUser.getSelectedRow(), 0).toString();
        txtName.setText(tableUser.getValueAt(tableUser.getSelectedRow(), 1).toString());
        txtUsername.setText(tableUser.getValueAt(tableUser.getSelectedRow(), 2).toString());
        txtPassword.setText(tableUser.getValueAt(tableUser.getSelectedRow(), 3).toString());
    }
);
```

- Sehingga ketika kita run akan menampilkan seperti ini



- Selanjutnya tambahkan actionListener pada tombol **update** dari GUI, dengan melakukan **right click** pada tombol-> **add event handler -> action -> actionPerformed**



- Lalu isikan dengan kode program berikut

```
btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
                User user = new User();
                user.setId(id);
                user.setNama(txtName.getText());
                user.setUsername(txtUsername.getText());
                user.setPassword(txtPassword.getText());
                usr.update(user);
                reset();
                loadTable();
```

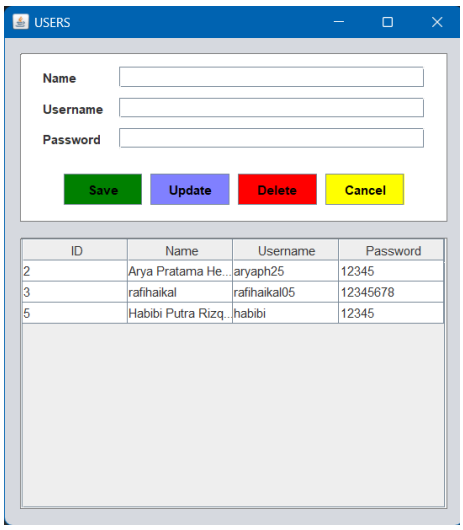
- Di dalam method actionPerformed, tombol Update digunakan untuk memperbarui data yang ada pada tabel. Pertama, dibuat objek baru dari kelas User, kemudian nilai id diatur menggunakan setId(id). Selanjutnya, data yang dimasukkan pada form teks seperti nama, username, dan password diambil menggunakan method getText() dan disimpan ke dalam objek user melalui setNama(), setUsername(), dan setPassword(). Setelah semua data terisi, method update(user) dipanggil untuk memperbarui data di database. Terakhir, method reset() dijalankan untuk mengosongkan form input, lalu loadTable() dipanggil agar tabel menampilkan data terbaru.

- Selanjutnya klik kanan tombol **delete → add event handler → action → actionPerformed** dan isikan dengan kode program berikut.

```
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 di hapus");
        }
    }
```

- Di dalam method actionPerformed, tombol Delete dijalankan dengan memeriksa apakah variabel id bernilai null atau tidak. Jika id ada, maka data akan dihapus dengan usr.delete(id), form dikembalikan ke kondisi awal menggunakan method reset(), dan tabel diperbarui melalui method loadTable(). Namun, jika id bernilai null, maka akan muncul pop-up pesan menggunakan JOptionPane yang berisi teks "Silahkan pilih data yang akan dihapus".

- Berikut Hasil GUI dari UserFrame yang sudah terkoneksi database.



E. Tugas / Latihan

1. Fungsi CRUD untuk Pelanggan (Customer)

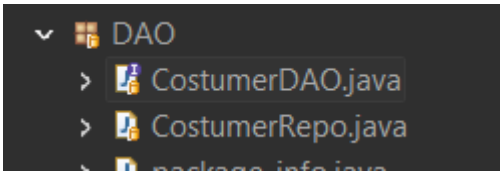
1) Membuat Fungsi DAO

- Pada package bernama DAO
- Buat class Interface baru dengan nama CostumerDAO, kemudian isikan dengan kode program berikut. Terdapat method save, show, delete dan update. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

```
1  package DAO;
2
3  import model.Costumer;
4  import java.util.List;
5
6  public interface CostumerDAO {
7      void save(Costumer costumer);
8      List<Costumer> show();
9      void update(Costumer costumer);
10     void delete(String id);
```

2) Menggunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama CustomerRepo yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat.



- Implementasikan CustomerDao dengan kata kunci implements

```
11
12 public class CostumerRepo implements CostumerDAO {
13
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database.

```
14     private final Connection connection;
15     private static final String INSERT = "INSERT INTO costumer (id, nama, alamat, nomor_hp) VALUES (?,?,?,?)";
16     private static final String SELECT = "SELECT * FROM costumer";
17     private static final String UPDATE = "UPDATE costumer SET nama=?, alamat=?, nomor_hp=? WHERE id=?";
18     private static final String DELETE = "DELETE FROM costumer WHERE id=?";
19
20     public CostumerRepo() {
21         connection = Database.koneksi();
22     }
```

- Selanjutnya membuat method save, isikan dengan kode program berikut

```
    @Override
    public void save(Costumer costumer) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(INSERT);
            st.setString(1, costumer.getId());
            st.setString(2, costumer.getNama());
            st.setString(3, costumer.getAlamat());
            st.setString(4, costumer.getNomorHp());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

- Lalu membuat method show untuk mengambil data dari database

```
    @Override
    public List<Costumer> show() {
        List<Costumer> list = new ArrayList<>();
        Statement st = null;
        ResultSet rs = null;
        try {
            st = connection.createStatement();
            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.setNomorHp(rs.getString("nomor_hp"));
                list.add(costumer);
            }
        } catch (SQLException e) {
            Logger.getLogger(CostumerRepo.class.getName()).log(Level.SEVERE, null, e);
        } finally {
            try {
                if (rs != null) rs.close();
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return list;
    }
```

- Selanjutnya kita membuat method update yang digunakan untuk mengubah data

```
    }

    @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.getNomorHp());
            st.setString(4, costumer.getId());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

- Dan terakhir kita Membuat method delete yang digunakan untuk menghapus data.

```
    }

    @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 {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

2. Fungsi CRUD untuk Layanan (Service)

1) Membuat Fungsi DAO

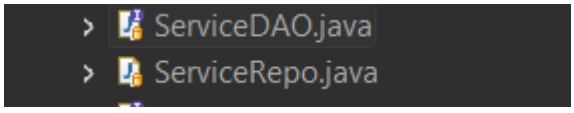
- Pada package bernama DAO

- Buat class Interface baru dengan nama ServiceDAO, kemudian isikan dengan kode program berikut. Terdapat method save, show, delete dan update. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

```
las_B master] ge DAO;

import model.Service;
import java.util.List;

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

2) Menggunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama ServiceRepo yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat.



- Implementasikan ServiceDao dengan kata kunci implements

```
public class ServiceRepo implements ServiceDAO {
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database.

```
    private final Connection connection;
    private static final String INSERT = "INSERT INTO service (id, jenis, harga, status) VALUES (?,?,?,?)";
    private static final String SELECT = "SELECT * FROM service";
    private static final String UPDATE = "UPDATE service SET jenis=?, harga=?, status=? WHERE id=?";
    private static final String DELETE = "DELETE FROM service WHERE id=?";

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

- Selanjutnya membuat method save, isikan dengan kode program berikut

```
    @Override
    public void save(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(INSERT);
            st.setString(1, service.getId());
            st.setString(2, service.getJenis());
            st.setDouble(3, service.getHarga());
            st.setString(4, service.getStatus());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

- Lalu membuat method show untuk mengambil data dari database

```
    @Override
    public List<Service> show() {
        List<Service> list = new ArrayList<>();
        Statement st = null;
        ResultSet rs = null;

        try {
            st = connection.createStatement();
            rs = st.executeQuery(SELECT);
            while (rs.next()) {
                Service service = new Service();
                service.setId(rs.getString("id"));
                service.setJenis(rs.getString("jenis"));
                service.setHarga(rs.getDouble("harga"));
                service.setStatus(rs.getString("status"));
                list.add(service);
            }
        } catch (SQLException e) {
            Logger.getLogger(ServiceRepo.class.getName()).log(Level.SEVERE, null, e);
        } finally {
            try {
                if (rs != null) rs.close();
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
```

- Selanjutnya kita membuat method update yang digunakan untuk mengubah data

```
    }

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

```

- Dan terakhir kita Membuat method delete yang digunakan untuk menghapus data.

```

    @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 {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}

```

F. KESIMPULAN

Tujuan dari praktikum ini adalah agar saya mampu memahami dan menerapkan pembuatan sistem CRUD (Create, Read, Update, Delete) pada data user menggunakan bahasa pemrograman Java yang terhubung dengan database MySQL. Melalui praktikum ini, saya belajar bagaimana membuat tabel user pada MySQL serta menghubungkannya dengan aplikasi Java melalui koneksi database. Selain itu, saya juga bertujuan untuk merancang antarmuka pengguna (GUI) yang mendukung proses CRUD dengan menerapkan konsep Pemrograman Berorientasi Objek (OOP). Tak hanya itu, saya juga mempelajari cara mengimplementasikan interface dan fungsi DAO (Data Access Object) sebagai pendekatan yang modular dan terstruktur untuk mengelola proses CRUD agar lebih efisien dan mudah dikembangkan ke depannya.