LAPORAN PRATIKUM PBO

MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete) USER
DENGAN DATABASE MYSQL

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 untuk membangun perangkat lunak. Metode ini ternyata lebih baik daripada cara pemrograman lama, dengan ide utamanya adalah menempatkan informasi dan tindakan yang menggunakan informasi tersebut di satu tempat, sehingga hanya tindakan tertentu yang dapat mengaksesnya. Dengan melakukan ini, OOP memastikan data aman dan terorganisir dengan baik, sekaligus membantu programmer membuat program yang terpisah dan terstruktur dengan baik.

OOP telah berkembang menjadi salah satu metode paling umum dalam bahasa pemrograman saat ini, seperti Java, karena mendukung ide-ide penting seperti menjaga privasi, mewariskan sesuatu, memiliki banyak bentuk, dan menggunakan kembali kode. Manfaat-manfaat ini membuat pembuatan perangkat lunak lebih mudah ditangani, diperbarui, dan dapat diubah untuk memenuhi kebutuhan yang mendesak. Oleh karena itu, memahami OOP sangat penting, terutama dalam praktik ini, karena memberikan Anda dasar-dasar untuk membuat program yang mengikuti aturan pembangunan perangkat lunak modern.

B. Tujuan

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

1. Mahasiswa mampu membuat tabel user pada MySQL dan menghubungkannya dengan aplikasi Java melalui koneksi 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. Alat dan Bahan

- Computer / laptop yang telah terinstall JDK dan Eclipse
- MySQL / XAMPP
- MySQL connector atau Connector/J

D. Teori Dasar

XAMPP adalah paket software open-source yang berisi Apache HTTP Server, MySQL, PHP, dan Perl, digunakan sebagai *development environment* berbasis *localhost*. Apache berfungsi sebagai web server, MySQL sebagai sistem manajemen basis data relasional, sedangkan PHP digunakan untuk membangun aplikasi web dinamis.

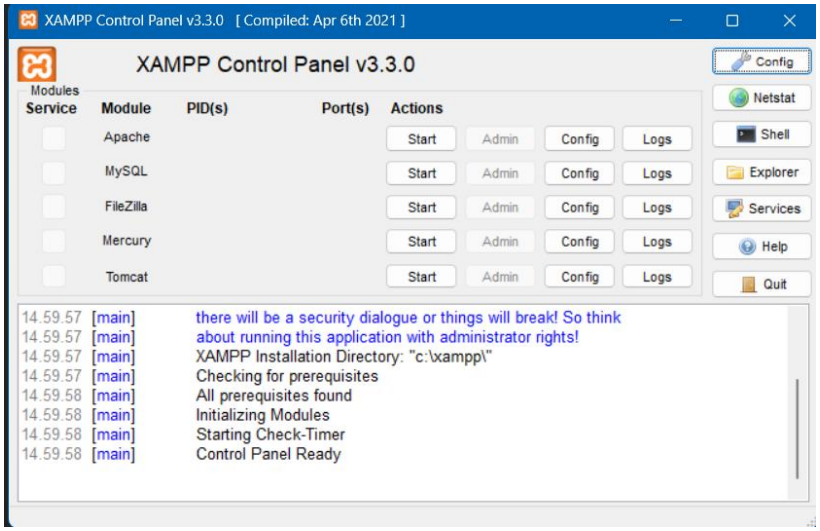
MySQL menyimpan data dalam bentuk tabel dan mendukung SQL (Structured Query Language) untuk membuat, membaca, memperbarui, dan menghapus data. Operasi ini disebut CRUD (Create, Read, Update, Delete) yang merupakan fungsi dasar dalam pengelolaan data

Java dapat terhubung ke database melalui JDBC (Java Database Connectivity) untuk mengelola data secara langsung. Dalam pengembangannya digunakan konsep Pemrograman Berorientasi Objek (PBO), termasuk pembuatan GUI untuk operasi CRUD, penerapan *interface* sebagai kontrak antar kelas, serta pola desain DAO (*Data Access Object*) untuk memisahkan logika akses data dari logika bisnis aplikasi.

E. Langkah – Langkah Pengerjaan

1. Install Xampp

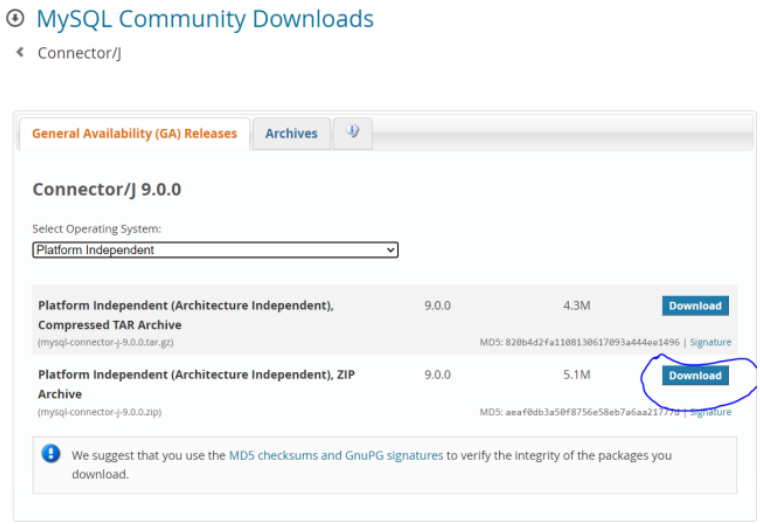
- Download XAMPP dari pada link berikut : https://www.apachefriends.org/
- Setelah didownload install xampp pada computer/laptop masing-masing
- Jalankan xampp dan aktifkan apache dan mysql



2. Menambahkan MySQL Connector

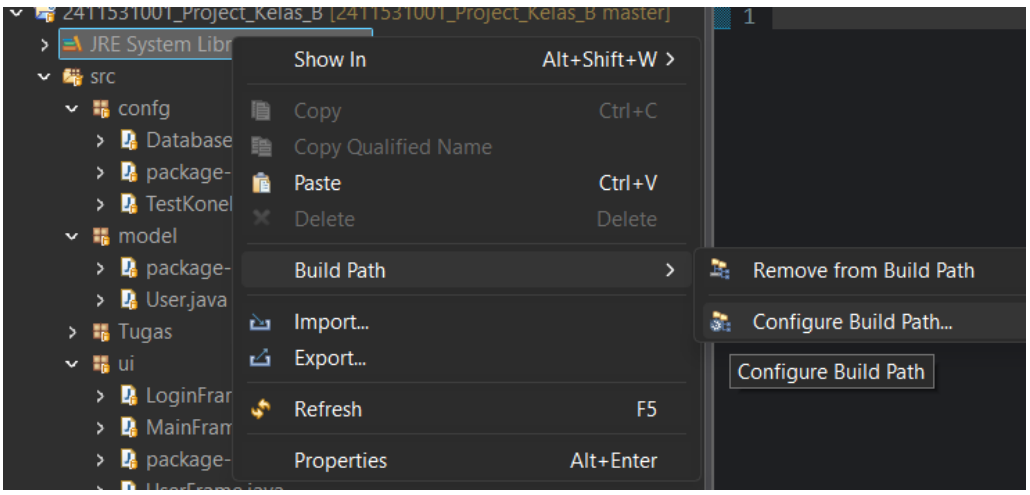
Aplikasi java agar dapat terhubung dengan database MySQL membutuhkan sebuah driver yaitu MySQL Connection, berikut ini Langkah-langkah membuat koneksi Database MySQL :

- Download MySQL connection pada link berikut https://dev.mysql.com/downloads/connector/j/
- Pilih file yang berekstensi .zip

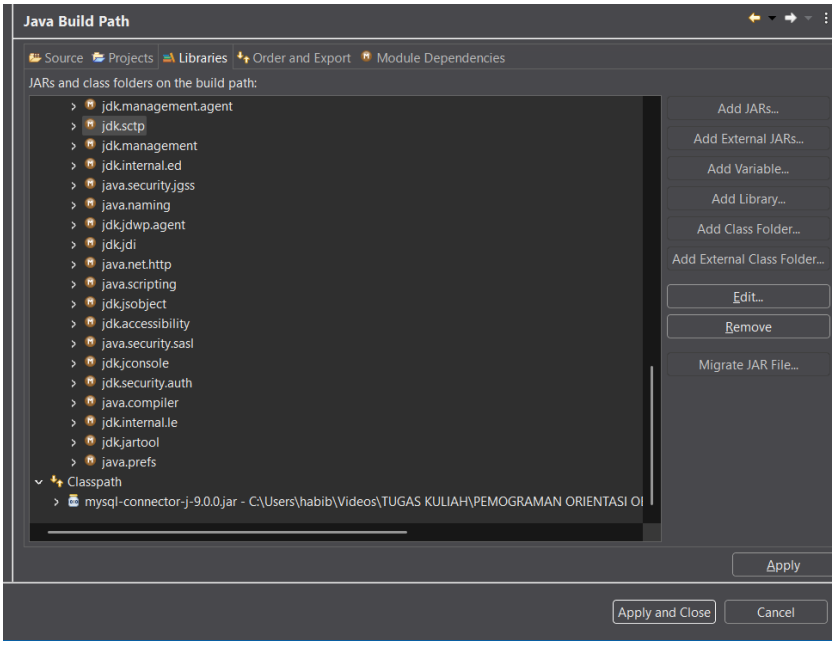


3. Menambahkan MySQL Connector ke dalam project

- klik kanan directory JRE System Library → Built Path → Configure Build Path

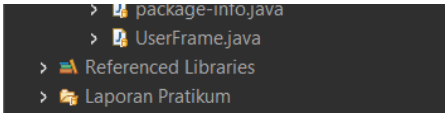


- Selanjutnya pilih Libraries → Classpath
- Tambahkan file MySQL Connector dengan cara klik Add External JARs dan pilih file yang telah didownlod dan pilih Apply and



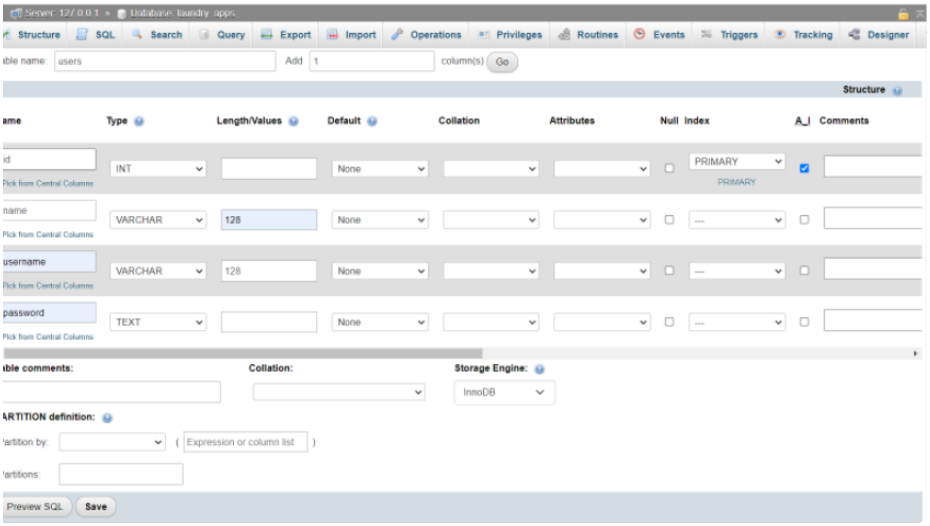
Close.

- Jika berhasil menambahkan MySQL Connector maka akan generate folder Referenced Libraries pada project yang berisi MySQL Connector.



4. Membuat Database dan Table User

- Buka http://localhost/phpmyadmin
- Klik new dan buat database dengan nama laundry_apps
- Buat table user dengan cara klik database laundry_apps dan buat table dengan nama user
- Klik create maka akan muncul seperti gambar dibawah ini.



- Isi seperti gambar diatas dan klik save

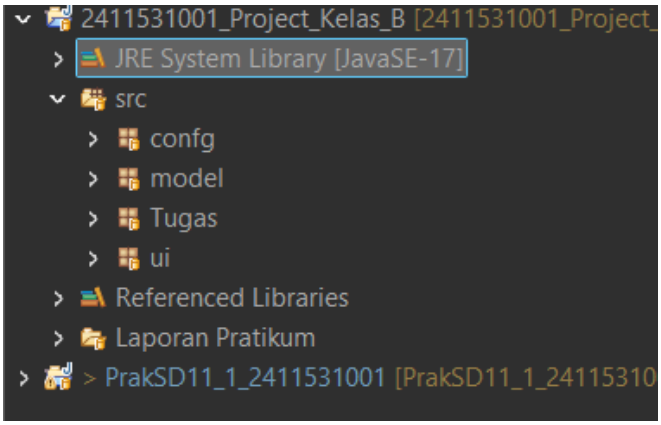
- Cara lain membuat table user pada database laundry_apps, klik SQL pada phpMyAdmin dan ketikan sql seperti gambar dibawah ini.

```sql
CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `username` varchar(64) NOT NULL,
  `password` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

5. **Membuat Koneksi ke Database MySQL**

Setelah menambahkan MySQL Connector maka dapat membuat koneksi ke database MySQL, berikut Langkah-langkahnya :

- 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.

```java
package confg;

import java.sql.*;

public class Database {
    Connection conn;

    public static Connection koneksi() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps",
                "root","");
            return conn;
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
    }
}
```

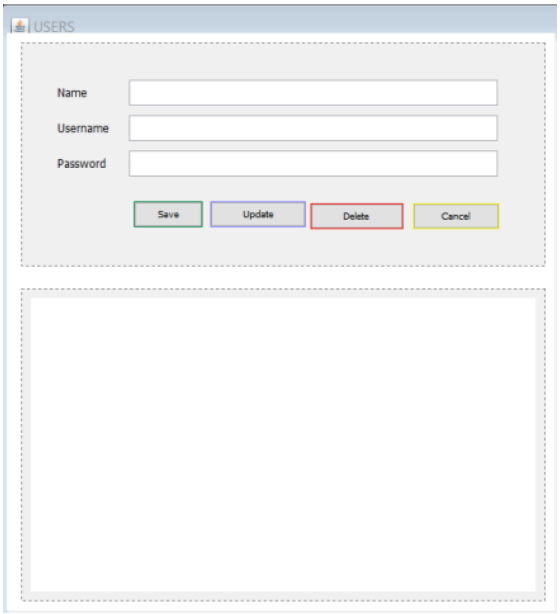
Penjelasan :

- import java.sql.* digunakan untuk mengimpor seluruh class dan method yang berkaitan dengan koneksi dan operasi database SQL.
- import javax.swing.JOptionPane digunakan untuk menampilkan pesan dialog jika terjadi kesalahan.

- Class Database berisi method koneksi() yang bertipe static agar dapat dipanggil tanpa membuat objek.
- Di dalam try:
    - Class.forName("com.mysql.cj.jdbc.Driver") memuat driver MySQL Connector/J agar Java dapat terhubung ke MySQL melalui JDBC.
    - DriverManager.getConnection(...) membuat koneksi ke database laundry_apps dengan user root dan password kosong.
    - Jika berhasil, mengembalikan objek Connection.
- Jika terjadi error, catch (Exception e) akan menangkap kesalahan dan menampilkannya dalam kotak dialog menggunakan JOptionPane, lalu mengembalikan null.

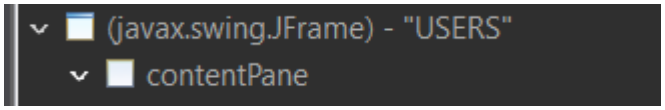
6. Membuat Tampilan CRUD User

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



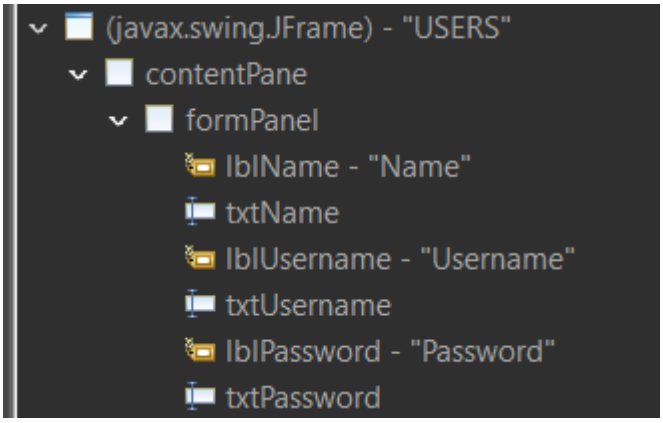
- Atur nama JFrame menjadi "USERS" untuk judul pada frame.

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

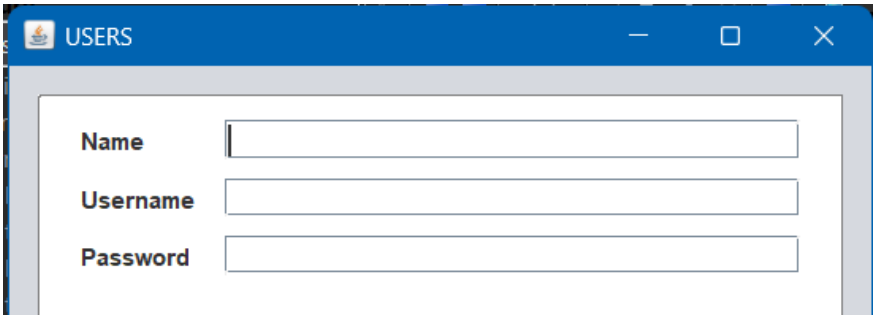


- Tambahkan panel bernama formPanel ke dalam contentPane. Panel ini berfungsi sebagai tempat input data user.

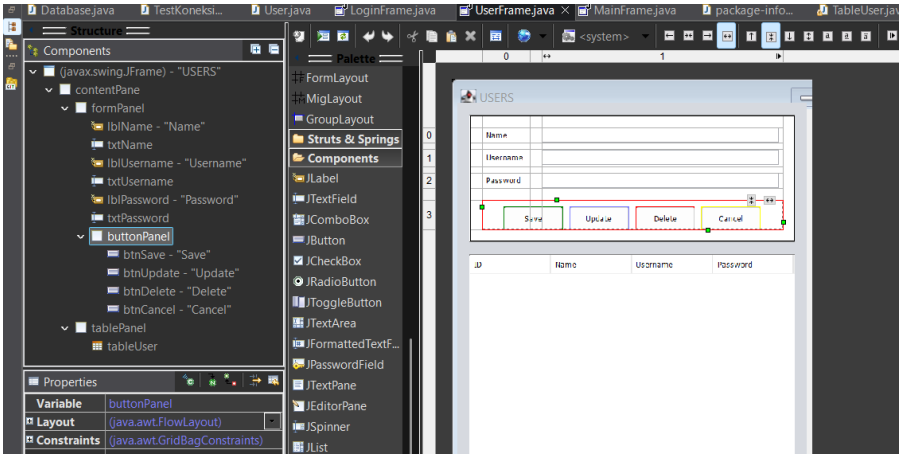
Komponen-komponen yang ditambahkan di formPanel antara lain:
  - JLabel lblName dan JTextField txtName untuk menginput nama.
  - JLabel lblUsername dan JTextField txtUsername untuk menginput username.
  - JLabel lblPassword dan JTextField txtPassword untuk menginput password.



Maka hasilnya akan seperti ini

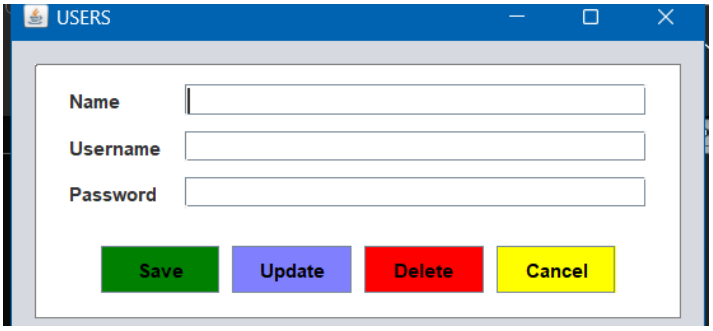


- Selanjutnya tambahkan panel bernama buttonPanel di bawah formPanel.
  Panel ini berisi 4 buah tombol (JButton) sebagai aksi utama:
  - btnSave untuk menyimpan data baru ke database.
  - btnUpdate untuk memperbarui data yang dipilih dari tabel.
  - btnDelete untuk menghapus data yang dipilih dari tabel.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

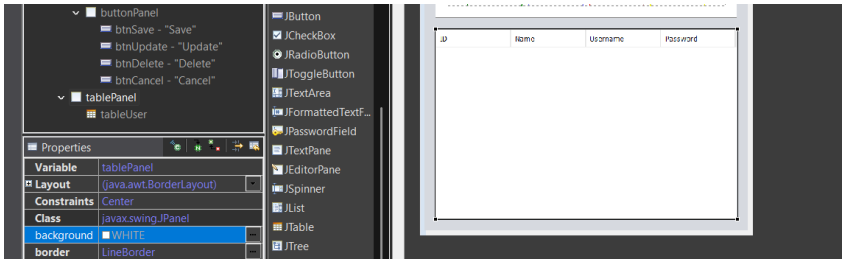


- Selanjutnya masing-masing tombol diatur warnanya untuk membedakan fungsi secara visual. Sebagai contoh warna yang saya gunakan :
  - Hijau → Save (tambah data)
  - Magenta → Update (ubah data)
  - Merah → Delete (hapus data)
  - Kuning → Cancel (reset/batal)

  Maka hasilnya seperti ini :



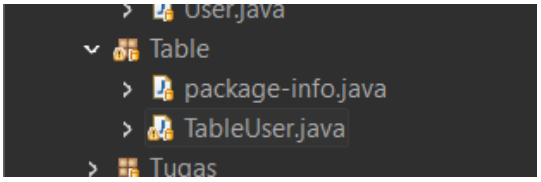
- Tambahkan panel bernama tablePanel di bagian bawah contentPane. Panel ini berisi sebuah JTable bernama tableUser untuk menampilkan data user dalam bentuk tabel.



7. Membuat Table Model

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

- Buat package baru dengan nama table

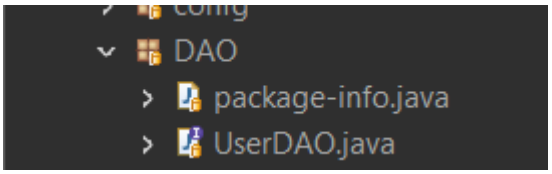


- Buat file baru didalam package table dengan nama TableUser, kemudian isikan dengan kode program berikut.

```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;
        }
    }
}
```

8. Membuat Fungsi DAO

- Buat package baru dengan nama DAO

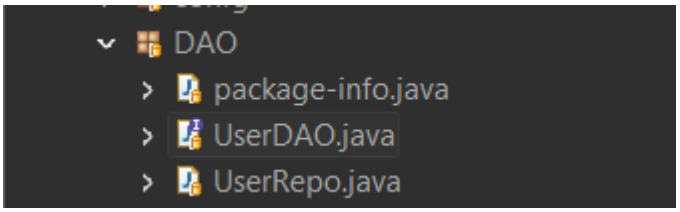



```java
package DAO;

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.

9. 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();
        }
    }
}
}
```

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.