LAPORAN PRAKTIKUM PEMOGRAMAN BERORIENTASI OBJEK

MEMBUAT FUNGSI CRUD USER DENGAN DATABASE MYSQL

OLEH:

AHMAD ZAHRAN

2411532004

DOSEN PENGAMPU:

NURFIAH, S.ST. M.KOM

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Pemrograman Berorientasi Objek (PBO) merupakan paradigma pemrograman yang memodelkan program berdasarkan objek dari dunia nyata dengan memanfaatkan atribut dan metode. Konsep ini penting dipahami karena banyak bahasa pemrograman modern, termasuk Java, menerapkannya sebagai dasar.

Pada praktikum ini dibuat aplikasi Laundry Apps menggunakan Java dengan menerapkan konsep class dan object untuk memodelkan tiga entitas utama: Customer**,** Service**,** dan Order. Aplikasi dilengkapi antarmuka grafis (GUI) berbasis Swing menggunakan JTabbedPane untuk memudahkan interaksi pengguna.

Tujuan pembuatan aplikasi ini adalah melatih pemahaman konsep dasar OOP, mulai dari pembuatan class dan object, penggunaan setter-getter.

B. Tujuan

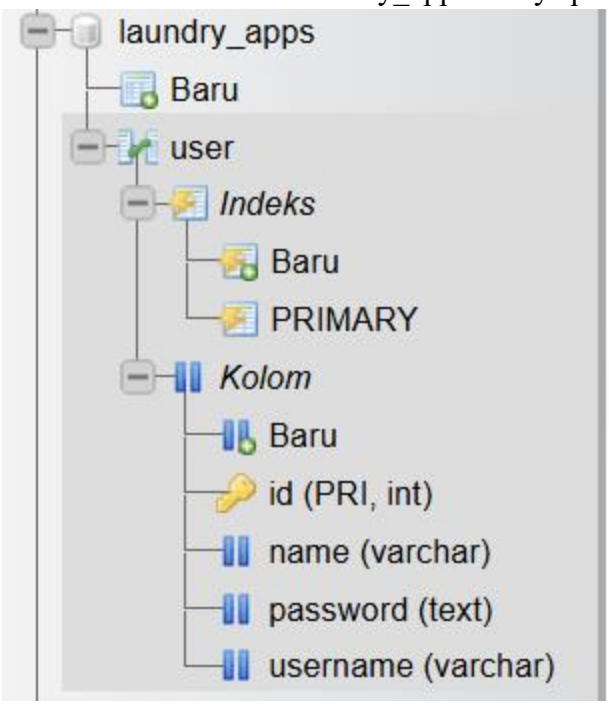
Tujuan dilakukannya praktikum ini adalah sebagai berikut:

1. Memahami cara koneksi database mysql dengan Java
2. Mengimplementasikan konsep CRUD (Create, Read, Update, Delete)
3. Membuat desain aplikasi Laundry

C. Langkah Kerja Praktikum

a. Membuat database

1. Pertama buat database laundry_apps di mysql



b. Kelas Database

1. Pertama deklarasi kelas Database dan deklarasi Connection bernama conn untuk menyimpan objek koneksi database lalu deklarasi method koneksi

```
public class Database {
    Connection conn;
    public static Connection koneksi() {
```

2. Selanjutnya buat koneksi ke mysql

```
try {
    Class.forName("com.mysql.cj.jdc.Driver");
    Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps",
            "root", "");
    return conn;
```

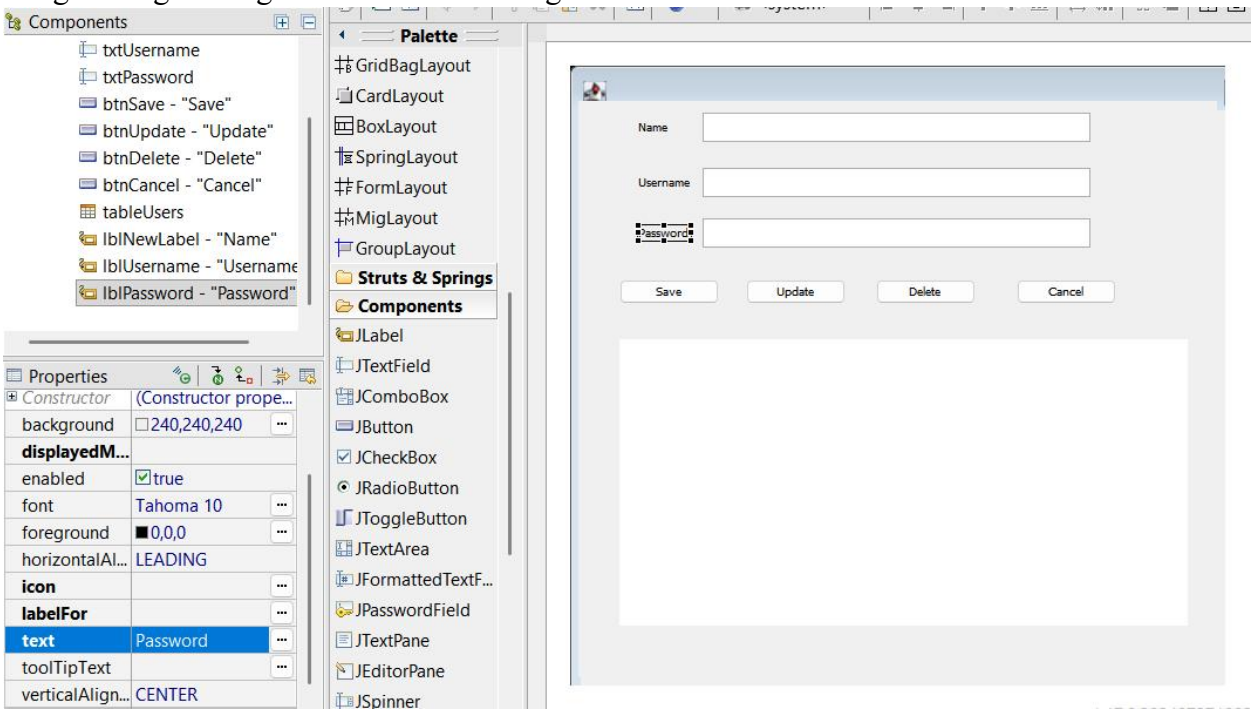
3. Dan buat catch untuk menampilkan pesan eror

```
}catch(Exception e) {
    JOptionPane.showMessageDialog(null, e);
    return null;
```

4. Fungsi dari kelas database ini secara keseluruhan adalah Menyediakan method koneksi() agar bagian lain dari program bisa langsung memanggil Database.koneksi() untuk mendapatkan koneksi ke MySQL.Jika berhasil mengembalikan objek Connection. Jika gagalmenampilkan pesan error dan mengembalikan null.

c. Kelas Userframe

1. Disini kita membuat tempilan untuk user menginputkan data yang mana disini terdiri dari beberapa tombol yaitu save, update, delete, dan cancel
   Yang masing-masing tombol memiliki fungsi tersendiri



d. Kelas TableUser

1. Pertama deklarasi kelas TableUser yang extends pada Abstract Table Model
   Artinya, kelas ini dipakai sebagai model data untuk ditampilkan di tabel (JTable).
   List<User> ls; menyimpan data user dalam bentuk List. Dan buat array yang berisi nama-nama kolom tabel (ID, Name, Username, Password).

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

2. Buat constructor untuk menerima List dari user yang mana disimpan di ls agra bisa ditampilkan di table

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

3. Buat method untuk table yang berisi jumlah dari baris dan kolom dan juga buat nama table

```java
@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 String getColumnName(int column) {
    // TODO Auto-generated method stub
    return columnNames[column];
}
```

4. Membuat method getValue untuk menentukan isi pada baris dan kolom table

```java
@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).getNama();
        case 2:
            return ls.get(rowIndex).getUsername();
        case 3:
            return ls.get(rowIndex).getPassword();
        default:
            return null;
```

e. Interface userDAO

1. Interface userDAO ini hanya berisi daftar method yang ingin di implementasikan nantinya, jadi kelas manapun yang ingin mengimplementasikan interface ini harus membuat semua method yang ada di interface ini yang mana disini ada method save, show, delete, dan update

```
package DAO;

import model.User;
import java.util.List;

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

f. Kelas UserRepo

1. Pertama deklarasi kelas UserRepo ynag mengimplementasi interface UserDAO dan buat objek koneksi ke database yaitu connection;

```
public class UserRepo implements UserDAO{
    private Connection connection;
```

2. Lalu buat query SQL untuk operasi CRUD yaitu ada insert, select, delete, update

```
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 FROM user SET name=?, username=?, password=? WHERE id=?;";
```

3. Selanjutnya buat method save

```
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 PreparedStatement berdasarkan query INSERT. Mengisi parameter (?) dengan nilai dari objek User. Menjalankan executeUpdate() untuk mengeksekusi INSERT. Menutup statement di blok finally.

4. Lalu buat method show

```
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 Statement untuk menjalankan query SELECT.ResultSet rs menyimpan hasil query. Looping while(rs.next()) untuk membaca setiap baris hasil

query.Membuat objek User, mengisi field dari kolom database, lalu menambahkannya ke List<User>.Return ls (list semua user).

5. Buat method update

```
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();
        }
```

6. Dan terakhit method delete

```
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();
        }
```