LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

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

OLEH :

DEDE BINTANG GAFENDI

2411533010

DOSEN PENGAMPU:

NURFIAH, S.ST, M.Kom..,

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

Pendahuluan

Dalam pengembangan aplikasi modern, pengelolaan data merupakan salah-satu aspek fundamental. Operasi dasar pengelolaan data yang dikenal dengan istilah CRUD (Create, Read, Update, Delete) menjadi inti dari hampir semua sistem informasi yang berinteraksi dengan basis data. Praktikum ini berfokus pada implementasi fungsi CRUD menggunakan bahasa pemrograman Java dengan konsep Pemrograman Berorientasi Objek (PBO).

DAO (Data Access Object). Pola DAO bertujuan memisahkan logika akses data dari logika bisnis aplikasi, sehingga kode menjadi lebih terstruktur, modular, dan mudah dikelola (reusable). Dengan demikian, aplikasi yang dibangun tidak hanya fungsional tetapi juga memiliki arsitektur yang baik dan mudah untuk dikembangkan lebih lanjut.

Tujuan

Tujuan praktikum ini yaitu mahasiswa mampu membuat fungsi CRUD data user menggunakan database MySQL, Adapun poin-poin praktikum yaitu :

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

Langkah langkah

1. Method ini berfungsi untuk membersihkan kembali nilai pada input field setelah sebuah proses berhasil dijalankan. Untuk itu, dibuatlah method reset pada JFrame seperti contoh kode berikut.

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

Membuat instance pada UserFrame

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

2. Berikan perintah seperti dibawah ini pada btn_save

```
 User user = new User();
user.setNama(txtName.getText());
user.setUsername(txtUsername.getText());
user.setPassword(txtPassword.getText());
usr.save(user);
reset();
loadTable(); // refresh table setelah simpan
        }
```

3. Buatlah sebuah method bernama loadTable(), lalu lengkapi isinya dengan potongan kode program berikut

```
public void loadTable() {
    ls = usr.show();
```

```
    TableUser tu = new TableUser(ls);
    tableUsers.setModel(tu);
    tableUsers.getTableHeader().setVisible(true);
}
```

4. Panggil method tersebut pada class main, sehingga saat program pertama kali dijalankan, method loadTable akan otomatis dieksekusi.

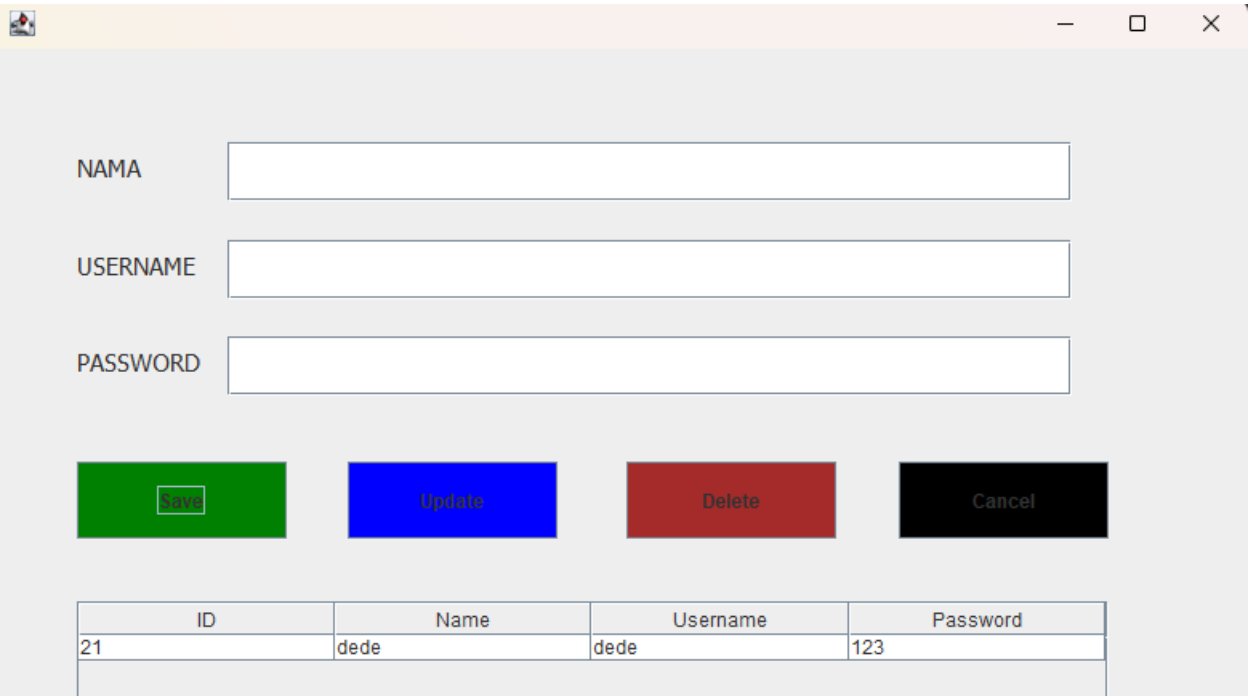
```
        UserFrame window = new UserFrame();
window.frame.setVisible(true);
window.loadTable();
```

5. Klik kanan pada JTable → add event handler → mouse → mouseClicked

```
                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());
```

Kode program diatas berfungsi yaitu mengambil id user dan menyimpannya kedalam variable id kemudian mengambil data nama, username dan password dan ditampilkan kedalam form inputan.

6. Jalankan dan input data



7. Klik kanan tombol update → add event handler → action → actionPeformed dan isikan dengan kode program berikut.

```
User user = new User();
user.setNama(txtName.getText());
user.setUsername(txtUsername.getText());
user.setPassword(txtPassword.getText());
user.setId(id);
usr.update(user);
reset();
loadTable();
```

8. Klik kanan tombol delete → add event handler → action → actionPerformed dan isikan dengan kode program berikut.

```
                if (id != null)   {
                        usr.delete(id);
                        reset();
                        loadTable();
                } else {
                        JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di
hapus bang");
```

Latihan

1. Buat table customer dan service pada database laundry_apps

```
CREATE TABLE customer (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    email VARCHAR(100)
);

CREATE TABLE service (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(100),
    harga DOUBLE
);
```

2. Tambahkan class customer dan generate setter & getter masing masing attribute

```java
package model;

public class Customer {
    String id, nama, alamat, nomorhp;
```

3. Buat DAO Interface pada package DAO

```java
package DAO;

import java.util.List;

public interface ServiceDao {
    void save(Service service);
    List<Service> show();
    void delete(String id);
    void update(Service service);
    List<Service> show1();
}
```

```java
package DAO;

import java.util.List;

public interface CustomerDao {
    void save(Customer customer);
    List<Customer> show();
    void delete(String id);
    void update(Customer customer);
}
```

4. Pada kelas CustomerRepo, koneksi database diinisialisasi melalui constructor. Method save(), show(), update(), dan delete() digunakan untuk mengelola data pelanggan (nama, alamat, nomorhp) dengan dukungan try-catch-finally agar resource selalu tertutup dengan baik.

```java
public class CustomerRepo implements CustomerDao {
  private Connection connection;
```

```java
public void save(Customer customer) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(insert);
        st.setString(1, customer.getNama());
        st.setString(2, customer.getAlamat());
        st.setString(3, customer.getNomorhp());
        st.executeUpdate();
    } catch(SQLException e) {
        e.printStackTrace();
```

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

```java
public void update(Customer customer) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, customer.getNama());
        st.setString(2, customer.getAlamat());
        st.setString(3, customer.getNomorhp());
        st.setString(4, customer.getId());
        st.executeUpdate();
    } catch(SQLException e) {
        e.printStackTrace();
```

```java
public List<Customer> show() {
    List<Customer> list = new ArrayList<>();
    Statement st = null;
    ResultSet rs = null;
    try {
        st = connection.createStatement();
        rs = st.executeQuery(select);
        while(rs.next()) {
            Customer customer = new Customer();
            customer.setId(rs.getString("id"));
            customer.setNama(rs.getString("nama"));
            customer.setAlamat(rs.getString("alamat"));
            customer.setNomorhp(rs.getString("nomorhp"));
            list.add(customer);
        }
    } catch(SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if(rs != null) rs.close();
            if(st != null) st.close();
        } catch(SQLException e) {
            e.printStackTrace();
        }
    }
    return list;
```

5. Pada kelas **ServiceRepo**, koneksi database diinisialisasi melalui constructor. Method save(), show(), update(), dan delete() digunakan untuk mengelola data layanan berupa jenis, harga, dan status, dengan setiap operasi dilengkapi pengelolaan resource menggunakan try-catch-finally.

```java
public class ServiceRepo implements ServiceDao {
    private Connection connection;
```

```java
public void save(Service service) {
    try (PreparedStatement st = connection.prepareStatement(insert)) {
        st.setString(1, service.getJenis());
        st.setString(2, service.getHarga());
        st.setString(3, service.getStatus());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
```

```java
public List<Service> show() {
    List<Service> list = new ArrayList<>();
    try (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.setHarga(rs.getString("harga"));
            service.setStatus(rs.getString("status"));
            list.add(service);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
```

```java
public void update(Service service) {
    try (PreparedStatement st = connection.prepareStatement(update)) {
        st.setString(1, service.getJenis());
        st.setString(2, service.getHarga());
        st.setString(3, service.getStatus());
        st.setString(4, service.getId());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

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

6. Buat table untuk customer dan service

Codingan TableCustomer

Menunjukkan bahwa TableCustomer adalah turunan AbstractTableModel dengan daftar data dan nama kolom.

```java
public class TableCustomer extends AbstractTableModel {

    private List<Customer> list;
    private String[] columnNames = {"ID", "Nama", "Alamat", "No. HP"};

    public TableCustomer(List<Customer> list) {
        this.list = list;
    }
```

Menghitung jumlah baris berdasarkan banyaknya data customer.

```java
public int getRowCount() {
    return list.size();
}
```

Menentukan jumlah kolom yang digunakan dalam tabel.

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

Mengambil nilai data customer sesuai baris dan kolom tabel.

```java
@Override
public Object getValueAt(int rowIndex, int columnIndex) {
    Customer customer = list.get(rowIndex);
    switch(columnIndex) {
        case 0: return customer.getId();
        case 1: return customer.getNama();
        case 2: return customer.getAlamat();
        case 3: return customer.getNomorhp();
        default: return null;
    }
}
```

Memberikan nama pada tiap kolom tabel.

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

Codingan TableService

Pada bagian ini dibuat class TableService yang merupakan turunan dari AbstractTableModel. Class ini memiliki variabel list untuk menampung data Service dan columnNames untuk menyimpan nama kolom yang akan ditampilkan di tabel.

```java
public class TableService extends AbstractTableModel {
    private List<Service> list;
    private String[] columnNames = {"ID", "Jenis", "Harga", "Status"};
```

Konstruktor ini berfungsi untuk menerima data berupa List<Service> dari luar class dan menyimpannya ke variabel list. Dengan cara ini, tabel dapat menampilkan data sesuai yang diberikan.

```java
public TableService(List<Service> list) {
    this.list = list;
}
```

Method getRowCount() mengembalikan jumlah baris yang akan ditampilkan pada tabel, sesuai jumlah data Service yang tersedia di list.

```java
public int getRowCount() {
    return list.size();
}
```

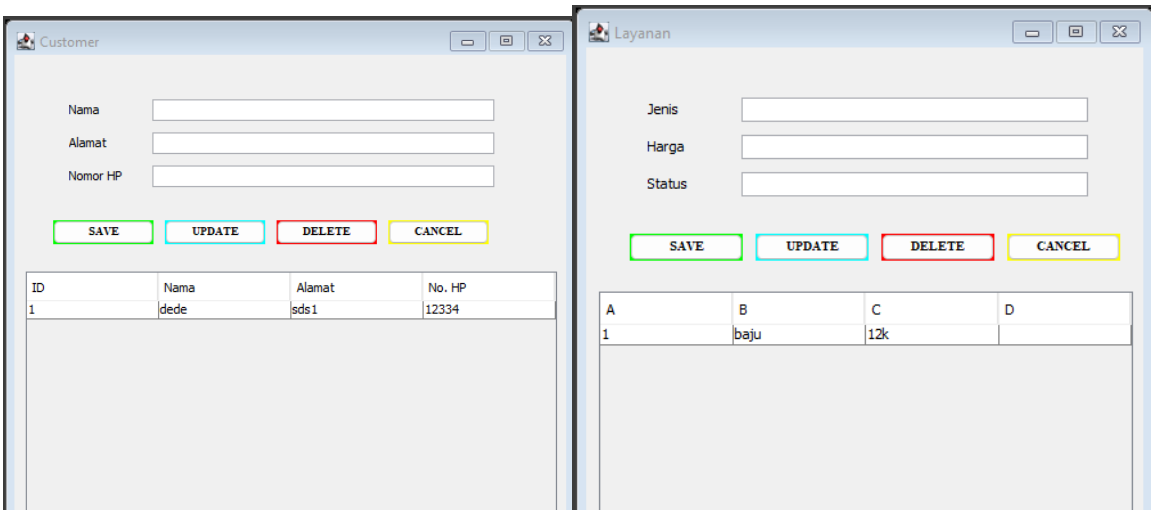
Method getColumnCount() menentukan jumlah kolom tabel, di sini ada 4 kolom: ID, Jenis, Harga, dan Status.

```
public int getColumnCount() {
    return 4;
}
```

Method getValueAt() digunakan untuk mengambil nilai yang akan ditampilkan pada setiap sel tabel berdasarkan indeks baris dan kolom. switch digunakan untuk memilih atribut Service yang sesuai dengan kolom yang dimaksud.

```
public Object getValueAt(int rowIndex, int columnIndex) {
    Service s = list.get(rowIndex);
    switch (columnIndex) {
        case 0: return s.getId();
        case 1: return s.getJenis();
        case 2: return s.getHarga();
        case 3: return s.getStatus();
        default: return null;
    }
}
```

7. Buat GUI untuk Customer dan Service, dan jalankan dengan menginputkan sesuatu pada program GUI



Penutup

disimpulkan bahwa fungsi CRUD (Create, Read, Update, Delete) untuk data pengguna telah berhasil diimplementasikan pada aplikasi Java. Proses ini melibatkan beberapa tahapan kunci, mulai dari perancangan dan pembuatan tabel user di database MySQL, hingga berhasil membangun koneksi antara aplikasi Java dan database menggunakan MySQL Connector/J.

Penerapan pola desain DAO (Data Access Object) terbukti efektif dalam memisahkan logika akses data dengan logika bisnis, yang terlihat dari pembuatan interface UserDAO dan kelas implementasinya UserRepo. Hal ini menjadikan kode program lebih terstruktur dan modular. Seluruh fungsionalitas CRUD—mulai dari menyimpan data baru, menampilkan data pada JTable, memperbarui, hingga menghapus data—berhasil diintegrasikan dengan antarmuka pengguna grafis (GUI) yang telah dirancang. Dengan demikian, praktikum ini telah mencapai tujuannya untuk membangun sebuah aplikasi PBO yang mampu mengelola data pengguna secara dinamis.