LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

MENGIMPLEMENTASIKAN FUNGSI CRUD (Create, Read, Update, Delete) PADA USERFRAME GUI

Oleh :

DERIEL CHAERAHMAN

NIM 2411533007

DOSEN PENGAMPU :

Nurfiah, S.ST., M.Kom

PROGRAM STUDI S1-INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

SEPTEMBER 2025

A. Pendahuluan

Praktikum ini dilakukan untuk membuat database aplikasi loundry dengan menerapkan CRUD (Create, Read, Update, Delete) dengan membuat database localhost mneggunakan XAMPP sebagai server, lalu dihubungkan ke java project loundry apps lewat MySQL Connector/J. Menerapkan konsep DAO (Data Access Object) dan Interface unutk logika akses data dan interface/GUI pada aplikasi dengan menerapkan konsep Pemrograman berorientasi objek pada java.

1. XAMPP
   Merupakan paket software teridir dari Apache HTTP Server, MySQL,PHP dan Perl yang bersifat open source, xampp biasanya digunakan sebagai deleveopment 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.

2. MySQL
   Merupakan sistem manajemen basis data relasional (RDBMS) open-source yang menggunakan Structured Query Language (SQL) sebagai bahasa untuk mengelola dan memanipulasi data dalam database. SQL adalah bahasa yang digunakan untuk melakukan berbagai operasi seperti menyimpan, mengambil, memperbarui, dan menghapus data dalam basis data. Sistem ini juga mendukung operasi CRUD (Create, Read, Update, Delete) untuk memanipulasi data dalam format table.

3. MySQL Connection/j
   Merupakan driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Berfungsi :
   - Membuka koneksi ke database MySQL
   - Mengirimkan permintaan SQL ke server MySQL
   - Menerima hasil dari permintaan SQL
   - Menutup koneksi ke database MySQL

4. 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). Bertujuan unutk :
   - Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga
   - Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
   - Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

5. Interface (Antarmuka)
   Mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya. Merupakan blue print dari class, berfungsi

menghubungkan antar objek. Interface bersifat abstrak, sehingga interface tidak bisa dibuat objek instance dengan kunci “new”. Interface menerapkan prinsip Abstraksi dan Enakpsulasi.

6. CRUD (Create, Read, Update, Delete)
   Merupakan fungsi dasar aplikasi yang berfungsi unutk membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

B. Tujuan

Tujuan dari dilakukannya praktikum ini adalah :

1. Mampu membuat table user pada database MySQL
2. Mampu membuat koneksi Java dengan database MySQL
3. Mampu membuat tampilan GUI CRUD user
4. Mampu membuat dan mengimplementasikan interface
5. Mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikan
6. Mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek.

C. Langkah kerja praktikum

a. Alat

1. Perangkat computer atau laptop
2. Jaringan internet
3. IDE (Integreted Development Environment) direkomendasikan Eclipse IDE
4. Java JDK (Java Development Kit)
5. MySQL & XAMPP
6. MySQL connector atau Connector/J

b. Menggunakan Fungsi CRUD DAO pada GUI

1. Buat method reset pada GUI UserFrame untuk menghapus value inputan ketika suatu proses berhasil dilakukan.

```
147    public void reset() {
148        txtName.setText("");
149        txtUsername.setText("");
150        txtPassword.setText("");
151    }
```

   Method reset berfungsi set input nama, username dan password menjadi teks kosong inputannya/mengosongkan ruang input.

2. Buat instasiasi pada UserFrame GUI.

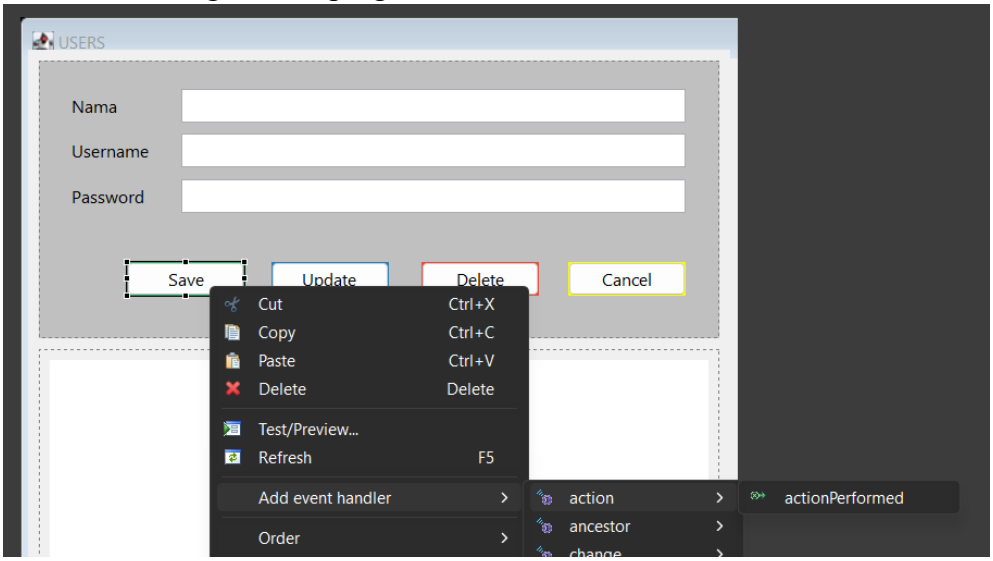
```
153    UserRepo usr = new UserRepo();
154    List<User> ls;
155    public String id;
```

3. Penjelasan :
   Membuat objek dari dari class UserRepo yang bertugas unutk mengelola data User (berisi method unutk terkoneksi ke database, method save, show, delete, update). Disini objek yang dibuat dengan nama “usr” dari class UserRepo.
4. Kemudian mendeklarasikan variabel “ls” unutk menyimpan daftar (list) berisi objek dari class User/wadah untuk kumpulan akun/data user

5. Mendeklarasikan variabel instance (field) String dengan akses public. Berfungsi unutk menyimpan ID user, dan karena public jadi dapat diakses dari luar class.

c. Create User

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



2. Tulis kode program sebagai berikut.

```
81          JButton btnSave = new JButton("Save");
82          btnSave.addActionListener(new ActionListener() {
83              public void actionPerformed(ActionEvent e) {
84                  User user = new User();
85                  user.setNama(txtName.getText());
86                  user.setUsername(txtUsername.getText());
87                  user.setPassword(txtPassword.getText());
88                  usr.save(user);
89                  reset();
90              }
91          });
```

3. Penjelasan : Membuat objek “User” baru untuk menyimpan data input dari form. Lalu mengambil input dari field name, username, password, yang kemudian menggunakan objek “usr” (yang dibuat sebelumnya) unutk menyimpan data user yang baru dibuat, untuk ditambahkan ke dalam list lalu ke database. Terakhir memanggil method reset() untuk mengosongkan form.

d. Read Users

1. Buat method dengan nama “loadTable”(), lalu tuliskan kode porgram berikut

```
157     public void loadTable() {
158         ls = usr.show();
159         TableUser tu = new TableUser(ls);
160         tableUsers.setModel(tu);
161         tableUsers.getTableHeader().setVisible(true);
162     }
```

2. Penjelasan : Memanggil method show() dari usr (yaitu objek dari class UserRepo) berisi semua dari user dari database lalu disimpan pada variabel ls.

3. Membuat objek baru bernama “tu” dari class TableUser dengan parameter “ls” yang berisi list user unutk dikirimkan agar tabel tahu data yang perlu ditampilkan.
4. Memanggil method setModel dari class tableUser dengan parameter “tu” unutk dihubngkan dengan tableUser yang sudah berisi/menyimpan data.
5. Menampilkan header kolom dengan set visible menjadi true.
6. Jadi, method loadtable() berfungsi unutk mengirim data ke Jtable dengan data user dari database.
7. Setelah form ditampilkan, method loadtable() dipanggil agar tabel diisi data dari database.

```
34    public static void main(String[] args) {
35        EventQueue.invokeLater(new Runnable() {
36            public void run() {
37                try {
38                    UserFrame frame = new UserFrame();
39                    frame.setVisible(true);
40                    frame.loadTable();
```

e. Update User

1. Klik kanan pada JTable → add event handler → mouse → mouseClicked.
2. Tulis kode berikut.

```
144         tableUsers = new JTable();
145         tableUsers.addMouseListener(new MouseAdapter() {
146             @Override
147             public void mouseClicked(MouseEvent e) {
148                 id = tableUsers.getValueAt(tableUsers.getSelectedRow(),0).toString();
149                 txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),1).toString());
150                 txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),2).toString());
151                 txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),3).toString());
152             }
153         });
```

3. Penjelasan : Berfungsi untuk mengambil id user dan menyimpannya ke dalam variabel id, lalu mengambil (get) data nama, username, dan password dan ditampilkan ke dalam form inputan.
4. Klik kanan tombol update → add event handler → action → actionPeformed dan isikan dengan kode program berikut.

```
100         JButton btnUpdate = new JButton("Update");
101         btnUpdate.addActionListener(new ActionListener() {
102             public void actionPerformed(ActionEvent e) {
103                 User user = new User();
104                 user.setNama(txtName.getText());
105                 user.setUsername(txtUsername.getText());
106                 user.setPassword(txtPassword.getText());
107                 user.setId(id);
108                 usr.update(user);
109                 reset();
110                 loadTable();
111             }
112         });
```

5. Penjelasan : Membuat objek “user” dari class user, kemudian mendapatkan (getText) data input dari form, lalu menggunakna method update() dari isntasiasi objek usr (UserRepo). Kemudian memanggil method reset() unutk mengosongkan hasil input dan memanggil method loadTable () untuk memuat ulang data pada tabelUsers/pada GUI.

f. Delete User

1. Klik kanan tombol delete → add event handler → action → actionPerformed dan tulis kode berikut.

```
121         JButton btnDelete = new JButton("Delete");
122         btnDelete.addActionListener(new ActionListener() {
123             public void actionPerformed(ActionEvent e) {
124                 if(id != null) {
125                     usr.delete(id);
126                     reset();
127                     loadTable();
128                 } else {
129                     JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
130                 }
131             }
132         });
```

2. Penjelasan : Berfungsi unutk menghapus data user dari jtable.

g. Membuat CRUD Layanan

1. Buat new class pada package model dengan nama “Layanan”.
2. Tulis kode sebagai berikut.

```
 1 package model;
 2
 3 public class Layanan {
 4     private String id;
 5     private String nama;
 6     private double harga;
 7
 8     // getter & setter
 9     public String getId() { return id; }
10     public void setId(String id) { this.id = id; }
11
12     public String getNama() { return nama; }
13     public void setNama(String nama) { this.nama = nama; }
14
15     public double getHarga() { return harga; }
16     public void setHarga(double harga) { this.harga = harga; }
17 }
```

3. Buat new class pada package DAO dengan nama “LayananRepo”.
4. Tulis kode sebagai berikut, import model.Layanan juga.

```
 1 package DAO;
 2
 3 import java.sql.*;
 4 import java.util.*;
 5 import java.util.logging.*;
 6 import model.Layanan;
 7
 8 public class LayananRepo {
 9     Connection connection;
10     String select = "SELECT * FROM layanan";
11
12     public LayananRepo(Connection conn) {
13         this.connection = conn;
14     }
15
16     // CREATE
17     public void save(Layanan layanan) {
18         try {
19             String sql = "INSERT INTO layanan(id, nama, harga) VALUES(?,?,?)";
20             PreparedStatement ps = connection.prepareStatement(sql);
21             ps.setString(1, layanan.getId());
22             ps.setString(2, layanan.getNama());
23             ps.setDouble(3, layanan.getHarga());
24             ps.executeUpdate();
25         } catch (SQLException e) {
26             Logger.getLogger(LayananRepo.class.getName()).log(Level.SEVERE, null, e);
27         }
28     }
```

```
    // READ
    public List<Layanan> show() {
        List<Layanan> list = new ArrayList<>();
        try {
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while (rs.next()) {
                Layanan l = new Layanan();
                l.setId(rs.getString("id"));
                l.setNama(rs.getString("nama"));
                l.setHarga(rs.getDouble("harga"));
                list.add(l);
            }
        } catch (SQLException e) {
            Logger.getLogger(LayananRepo.class.getName()).log(Level.SEVERE, null, e);
        }
        return list;
    }
    // UPDATE
    public void update(Layanan layanan) {
        try {
            String sql = "UPDATE layanan SET nama=?, harga=? WHERE id=?";
            PreparedStatement ps = connection.prepareStatement(sql);
            ps.setString(1, layanan.getNama());
            ps.setDouble(2, layanan.getHarga());
            ps.setString(3, layanan.getId());
            ps.executeUpdate();
        } catch (SQLException e) {
            Logger.getLogger(LayananRepo.class.getName()).log(Level.SEVERE, null, e);
        }
    }

    // DELETE
    public void delete(String id) {
        try {
            String sql = "DELETE FROM layanan WHERE id=?";
            PreparedStatement ps = connection.prepareStatement(sql);
            ps.setString(1, id);
            ps.executeUpdate();
        } catch (SQLException e) {
            Logger.getLogger(LayananRepo.class.getName()).log(Level.SEVERE, null, e);
        }
    }
}
```

Penjelasna : Class Layanan di package model berisi attribut id, nama, dan harga serta setter/getter unutk memanipulasi data. Class LayananRepo di package DAO berfungsi untuk mengelola data layanan dengan menyediakan method CRUD.

h. Membuat CRUD Layanan

1. Buat new class pada package model dengan nama “Pelanggan”.
2. Tulis kode sebagai berikut.

```
package model;

public class Pelanggan {
    private String id;
    private String nama;
    private String alamat;
    private String telepon;

    // getter & setter
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getNama() { return nama; }
    public void setNama(String nama) { this.nama = nama; }

    public String getAlamat() { return alamat; }
    public void setAlamat(String alamat) { this.alamat = alamat; }

    public String getTelepon() { return telepon; }
    public void setTelepon(String telepon) { this.telepon = telepon; }
}
```

3. Buat new class pada package DAO dengan nama “PelangganRepo”.
4. Tulis kode sebagai berikut, import model.Pelanggan juga

```java
package DAO;

import java.sql.*;
import java.util.*;
import java.util.logging.*;
import model.Pelanggan;

public class PelangganRepo {
    Connection connection;
    String select = "SELECT * FROM pelanggan";

    public PelangganRepo(Connection conn) {
        this.connection = conn;
    }

    // CREATE
    public void save(Pelanggan pelanggan) {
        try {
            String sql = "INSERT INTO pelanggan(id, nama, alamat, telepon) VALUES(?,?,?,?)";
            PreparedStatement ps = connection.prepareStatement(sql);
            ps.setString(1, pelanggan.getId());
            ps.setString(2, pelanggan.getNama());
            ps.setString(3, pelanggan.getAlamat());
            ps.setString(4, pelanggan.getTelepon());
            ps.executeUpdate();
        } catch (SQLException e) {
            Logger.getLogger(PelangganRepo.class.getName()).log(Level.SEVERE, null, e);
        }
    }
```

```java
    // READ
    public List<Pelanggan> show() {
        List<Pelanggan> list = new ArrayList<>();
        try {
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while (rs.next()) {
                Pelanggan p = new Pelanggan();
                p.setId(rs.getString("id"));
                p.setNama(rs.getString("nama"));
                p.setAlamat(rs.getString("alamat"));
                p.setTelepon(rs.getString("telepon"));
                list.add(p);
            }
        } catch (SQLException e) {
            Logger.getLogger(PelangganRepo.class.getName()).log(Level.SEVERE, null, e);
        }
        return list;
    }
```

```
51      // UPDATE
52      public void update(Pelanggan pelanggan) {
53          try {
54              String sql = "UPDATE pelanggan SET nama=?, alamat=?, telepon=? WHERE id=?";
55              PreparedStatement ps = connection.prepareStatement(sql);
56              ps.setString(1, pelanggan.getNama());
57              ps.setString(2, pelanggan.getAlamat());
58              ps.setString(3, pelanggan.getTelepon());
59              ps.setString(4, pelanggan.getId());
60              ps.executeUpdate();
61          } catch (SQLException e) {
62              Logger.getLogger(PelangganRepo.class.getName()).log(Level.SEVERE, null, e);
63          }
64      }
65
66      // DELETE
67      public void delete(String id) {
68          try {
69              String sql = "DELETE FROM pelanggan WHERE id=?";
70              PreparedStatement ps = connection.prepareStatement(sql);
71              ps.setString(1, id);
72              ps.executeUpdate();
73          } catch (SQLException e) {
74              Logger.getLogger(PelangganRepo.class.getName()).log(Level.SEVERE, null, e);
75          }
76      }
77  }
```

Penjelasan : Class pelanggan di package model berfungsi unutk mempresentasikan data (entity) dari objek pelanggan yang berisi attribut berupa id, nama, alamat dan setter/getter unutk mengakses/merubah nilai attribut. Class PelngganRepo di package DAO berfugnsi unutk mengelola data pelanggan, berupa menyediakan method CRUD unutk menyimpan, mengambil, memperbarui atau menghapus data pelanggan dari database.