LAPORAN PRAKTIKUM 2

PEMROGRAMAN BERORIENTASI OBJEK (PBO)

OLEH

Jovantri Immanuel Gulo

2411532014

DOSEN PENGAMPU

Nurfiah, S.ST, M.Kom.

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

A. Pendahuluan

Pemrograman Berorientasi Object (PBO) saat ini banyak diterapkan dalam berbagai macam penerapan aplikasi-aplikasi dalam Java, dengan adanya PBO ini kita semakin dimudahkan dengan kode program yang lebih terstruktur dan mudah untuk dibaca, ini juga memudahkan kita dalam menemukan kesalahan atau error yang terjadi, dikarenakan kode program yang tersusun dengan baik dan teratur. Dalam aplikasi laundry yang dibuat, kita menerapkan berbagai macam bentuk PBO di antaranya seperti adanya implements dari interface dan juga inheritance dan lainnya. Semua ini bertujuan untuk pembelajaran dan mempermudah serta memperluas pengetahuan kita dalam bidang pemrograman, terkhususnya Java.

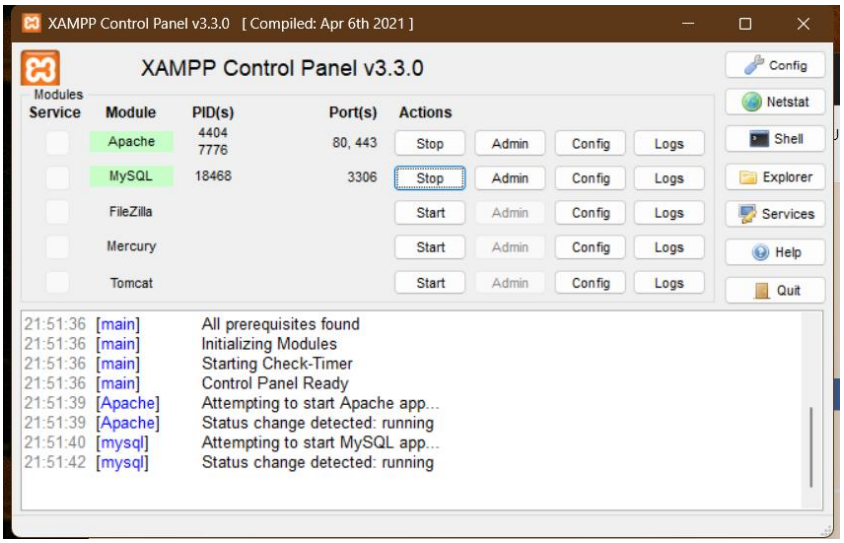
B. Tujuan

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

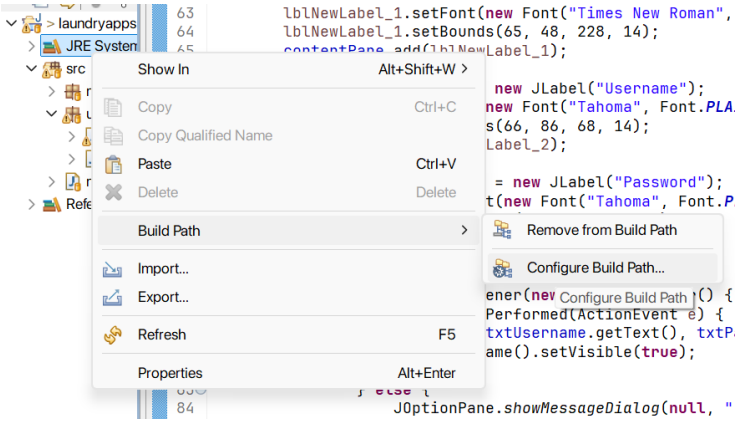
1. Mahasiswa mampu membuat table user pada database MySQL
2. Mahasiswa mampu membuat koneksi Java dengan database MySQL
3. Mahasiswa mampu membuat tampilan GUI CRUD user
4. Mahasiswa mampu membuat dan mengimplementasikan interface
5. Mahasiswa mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikannya.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

C. Langkah-langkah

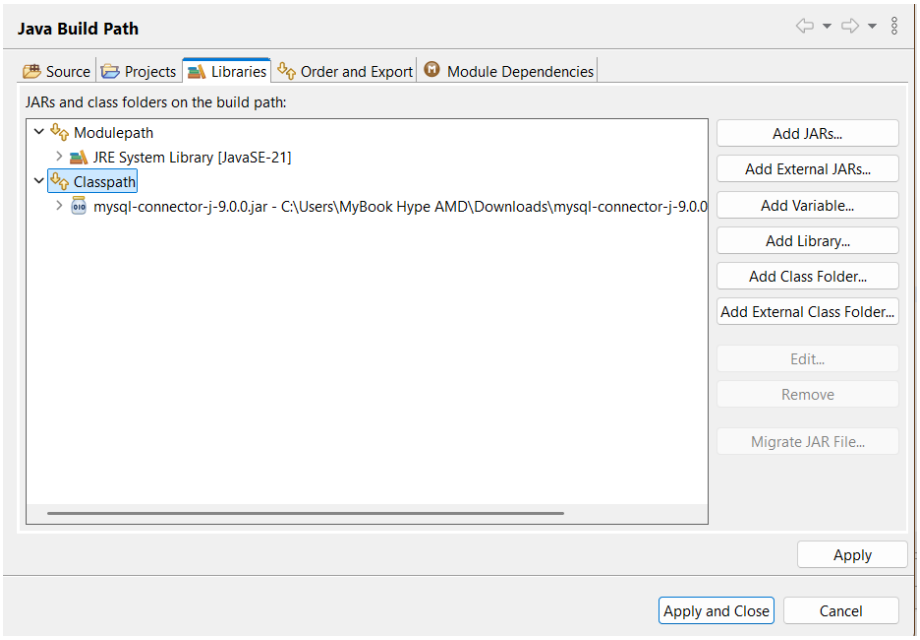
1. Install XAMPP dan jalankan Apache dan MySQL



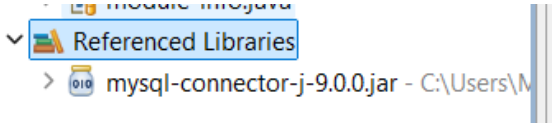
2. Install MySQL connector, dan letakkan pada build path, sebagai berikut



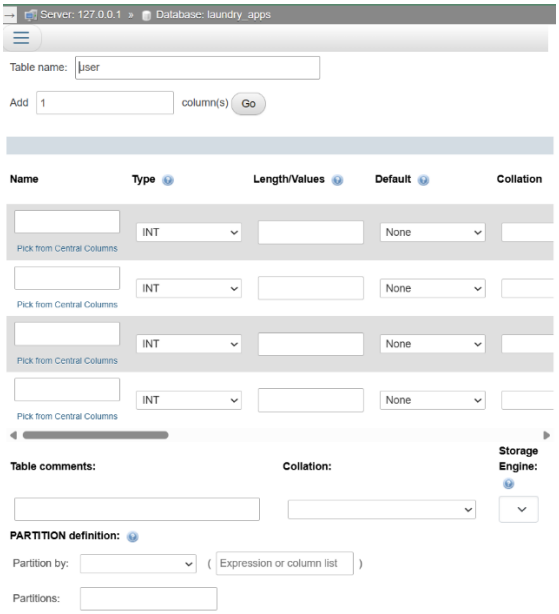
3. Pilih Libraries dan Classpath dan tambahkan file jar pada External JARs dan pilih jar dari file MySQL connector tadi.



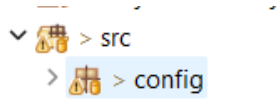
4. Maka akan muncul file jar mysql connector pada Referenced Libraries



5. Buka http://localhost/phpmyadmin dan klik new dan buat database dengan nama **laundry_apps**
6. Buat tabel user dengan cara klik database laundry_apps dan buat tabel dengan nama **user**.
7. Klik create dan akan muncul seperti gambar di bawah ini, lalu klik save.



8. Buat package baru dengan nama **config**



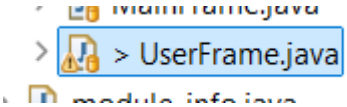
9. Buat class baru dengan nama **Database** kemudian kita konfigurasikan

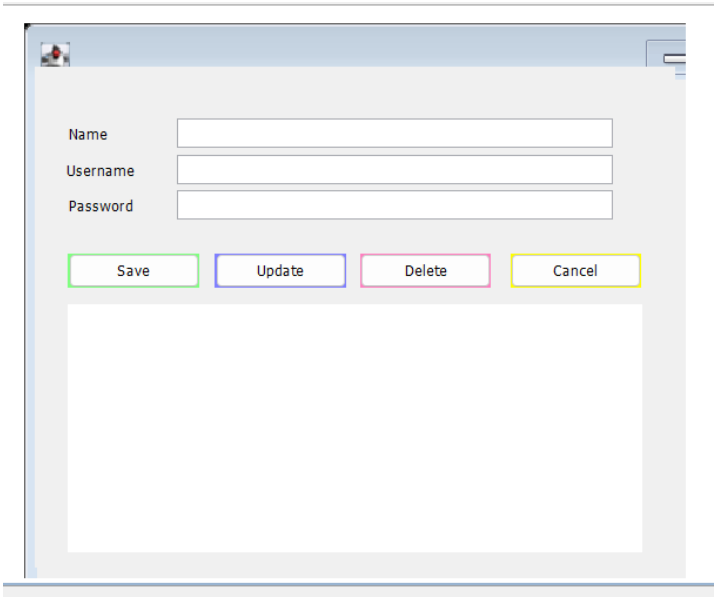
```
1  package config;
2
3  import java.sql.*;
5
6  public class Database {
7      Connection conn;
8      public static Connection koneksi() {
9          try {
10             Class.forName("com.mysql.cj.jdbc.Driver");
11             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps", "root", "");
12             return conn;
13         } catch(Exception e) {
14             JOptionPane.showMessageDialog(null, e);
15             return null;
16         }
17     }
18 }
19
```

Penjelasan :

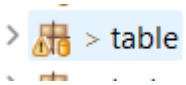
- Import java.sql.* digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method Connection dengan nama koneksi, yang mana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi database, jika koneksi berhasil maka akan mengembalikkan nilai Connection
- Line 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan JOptionPane.

10. Pada package ui, buat class baru dengan nama **UserFrame** dan desain seperti gambar berikut ini





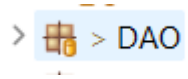
11. Buat package baru dengan nama **table**



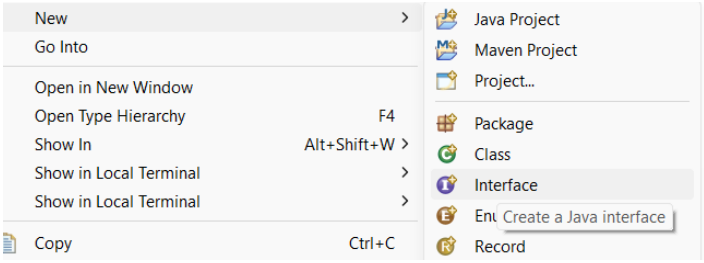
12. Buat class baru dalam package table dengna nama **TableUser**, kemudian isikan kode program berikut

```java
package table;

import java.util.List;

public class TableUser extends AbstractTableModel {
	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) {
	  User user = ls.get(rowIndex);
	  switch (columnIndex) {
	    case 0:
	      return user.getId();
	    case 1:
	      return user.getNama();
	    case 2:
	      return user.getUsername();
	    case 3:
	      return user.getPassword();
	    default:
	      return null;
	  }
	}
}
```

13. Buat package baru dengan nama **DAO**



14. Buat **class Interface** baru dengan nama **UserDAO** kemudian isikan dengan kode program tersebut. Terdapat method save, show, delete, dan update.




```java
package DAO;

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);
}
```

15. Pada package DAO, buat file class baru dengan nama **UserRepo** yang digunakan untuk mengimplementasikan DAO yang telah dibuat.
    a. Implementasikan UserDao dengan kata kunci **implements**

```
1  package DAO;
2
3  import java.sql.Connection;
15
16 public class UserRepo implements UserDAO {
```

    b. Membuat instanisasi connection

```
17     private Connection connection;
18     final String insert = "INSERT INTO user (name, username, password) VALUES (?, ?, ?)";
19     final String select = "SELECT * FROM user";
20     final String delete = "DELETE FROM user WHERE id = ?";
21     final String update = "UPDATE user SET name = ?, username = ?, password = ? WHERE id = ?";
22
23     public UserRepo() {
24         connection = Database.koneksi();
25     }
```

    c. Membuat method save

```
27     @Override
28     public void save(User user) {
29         PreparedStatement st = null;
30         try {
31             st = connection.prepareStatement(insert);
32             st.setString(1, user.getNama());
33             st.setString(2, user.getUsername());
34             st.setString(3, user.getPassword());
35             st.executeUpdate();
36         } catch (SQLException e) {
37             e.printStackTrace();
38         } finally {
39             try {
40                 st.close();
41             } catch (SQLException e) {
42                 e.printStackTrace();
43             }
44         }
45     }
```

    d. Membuat method show

```
47     @Override
48     public List<User> show() {
49         List<User> ls = null;
50         try {
51             ls = new ArrayList<User>();
52             Statement st = connection.createStatement();
53             ResultSet rs = st.executeQuery(select);
54             while (rs.next()) {
55                 User user = new User();
56                 user.setId(rs.getString("id"));
57                 user.setNama(rs.getString("name"));
58                 user.setUsername(rs.getString("username"));
59                 user.setPassword(rs.getString("password"));
60                 ls.add(user);
61             }
62         } catch (SQLException e) {
63             Logger.getLogger(UserDAO.class.getName()).log(Level.SEVERE, null, e);
64         }
65         return ls;
66     }
```

    e. Membuat method update

```
86      @Override
87      public void update(User user) {
88          PreparedStatement st = null;
89          try {
90              st = connection.prepareStatement(update);
91              st.setString(1, user.getNama());
92              st.setString(2, user.getUsername());
93              st.setString(3, user.getPassword());
94              st.setString(4, user.getId());
95              st.executeUpdate();
96          } catch (SQLException e) {
97              e.printStackTrace();
98          } finally {
99              try {
100                 st.close();
101             } catch (SQLException e) {
102                 e.printStackTrace();
103             }
104         }
105     }
```

f. Membuat method delete

```
68      @Override
69      public void delete(String id) {
70          PreparedStatement st = null;
71          try {
72              st = connection.prepareStatement(delete);
73              st.setString(1, id);
74              st.executeUpdate();
75          } catch (SQLException e) {
76              e.printStackTrace();
77          } finally {
78              try {
79                  st.close();
80              } catch (SQLException e) {
81                  e.printStackTrace();
82              }
83          }
84      }
85
```

D. Kesimpulan

Dari praktikum yang telah dikerjakan, dapat diambil kesimpulan bahwa pemrograman berorientasi objek memudahkan kita dalam mengelompokkan sesuatu yang memiliki attribut dan sifat, sehingga untuk menggunakannya, kita tinggal memanggilnya sesuai dengan nama classnya, ini sangat mempermudah kita, ditambah lagi dengan adanya sifat inheritance dan implements, semakin mempermudah kita dalam membangun aplikasi-aplikasi di lingkungan bahasa pemrograman Java.