LAPORAN PRAKTIKUM 2

PEMROGRAMAN BERORIENTASI OBJEK

Oleh:

Nama : Arkan Ubaidillah Warman

NIM : 2411537001

Dosen Pengampu : Nurfiah,S.ST,M.Kom

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERITAS ANDALAS

PADANG

A. PENDAHULUAN

1. Latar Belakang

Perkembangan teknologi informasi menuntut adanya sistem yang mampu mengelola data secara efisien, cepat, dan terstruktur. Salah satu kebutuhan utama dalam pembangunan aplikasi adalah manajemen data pengguna (*user management*). Proses manajemen data umumnya mencakup penambahan data baru, menampilkan data, memperbarui data, serta menghapus data. Empat proses dasar ini dikenal dengan istilah CRUD (*Create, Read, Update, Delete*).

Dalam praktik pengembangan aplikasi, integrasi antara bahasa pemrograman dan sistem manajemen basis data (DBMS) merupakan hal penting agar data dapat dikelola secara efektif. MySQL sebagai salah satu DBMS populer, sering digunakan karena bersifat open source, ringan, dan mendukung integrasi dengan berbagai bahasa pemrograman, termasuk Java.

Eclipse sebagai *Integrated Development Environment (IDE)* mendukung pengembangan aplikasi Java dengan mudah melalui fitur debugging, integrasi library, serta dukungan konektivitas database. Oleh karena itu, membuat fungsi CRUD dengan database MySQL di Eclipse menjadi salah satu dasar penting untuk memahami proses pengelolaan data secara terstruktur dalam aplikasi berbasis Java.

2. Tujuan Praktikum

- Mampu membuat tabel user pada database MYSQL
- Mampu membuat koneksi java dengan database MYSQL
- Mampu membuat tampilan GUI CRUD user
- Mampu membuat dan mengimplementasikan interface
- Mampu membuat fungsi DAO dan mengimplementasikannya
- Mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

3. Landasan teori

- **CRUD (Create, Read, Update, Delete)**

  CRUD merupakan empat operasi dasar dalam manajemen data:

a. **Create**: menambahkan data baru ke dalam database.

b. **Read**: menampilkan atau membaca data yang sudah ada.

c. **Update**: memperbarui data tertentu sesuai kebutuhan.

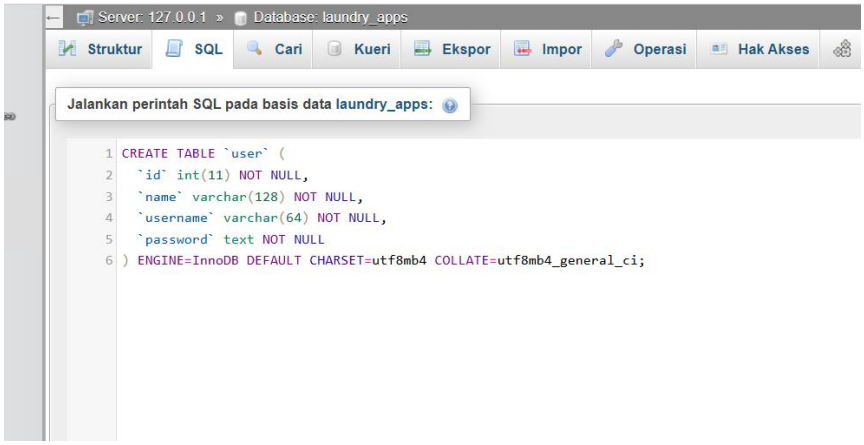
d. **Delete**: menghapus data dari database.

- **Database MySQL**
  MySQL adalah sistem manajemen basis data relasional (*Relational Database Management System/RDBMS*) yang menggunakan bahasa SQL (*Structured Query Language*). Kelebihan MySQL adalah bersifat open source, mendukung skala besar, cepat, serta kompatibel dengan berbagai bahasa pemrograman.
- **Java Database Connectivity (JDBC)**
  JDBC merupakan API (Application Programming Interface) dalam Java yang digunakan untuk menghubungkan program dengan database. Dengan JDBC, aplikasi Java dapat mengirim query SQL ke database MySQL dan memproses hasilnya.
- **Eclipse IDE**
  Eclipse adalah IDE populer untuk pengembangan aplikasi Java. Fasilitas yang disediakan, seperti integrasi library JDBC dan fitur debugging, memudahkan pengembang dalam membuat aplikasi berbasis database.

B. LANGKAH-LANGKAH

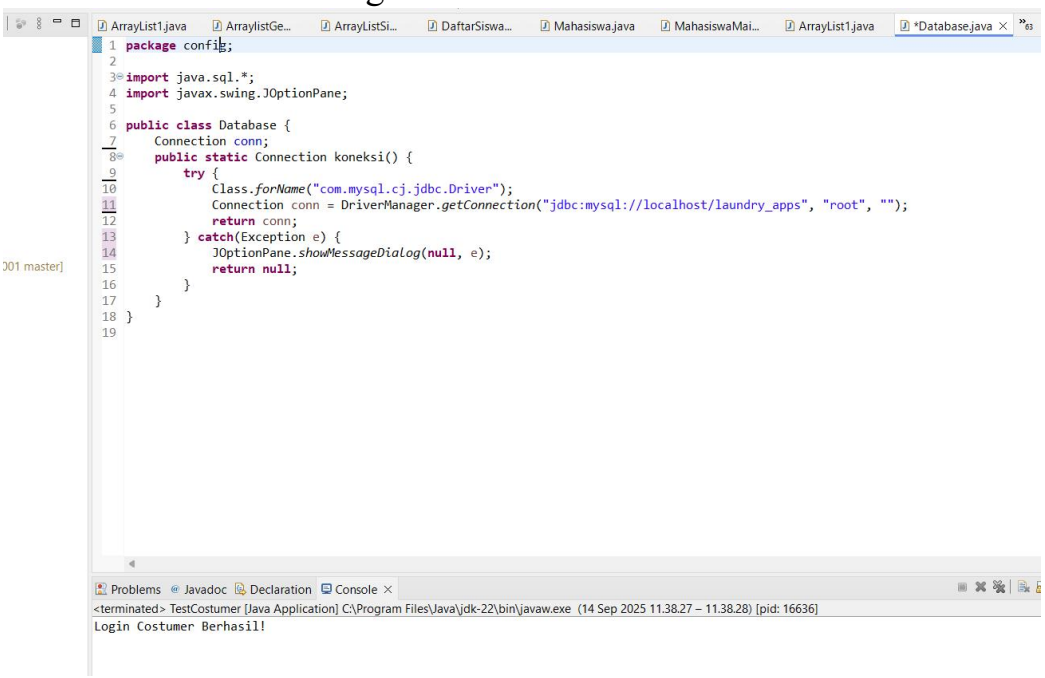
1. Download beberapa aplikasi seperti mysql connection
2. Tambahkan MySQL connector dalam projek eclipse.

3. Membuat Table User.



4. Membuat koneksi ke database mysql

- Buat package baru dengan nama config
- Lalu buat class baru dengan nama Database

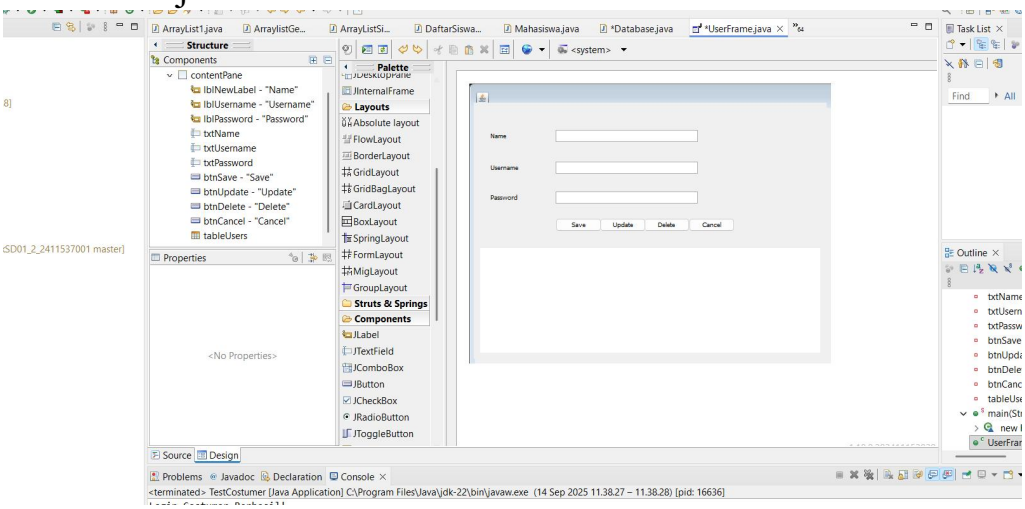


Penjelasan :

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

5. Membuat tampilan CRUD user.

Buat file baru menggunakan JFrame pada package UI Dengan nama UserFrame.java



6. Membuat tabel model

- Buat package baru dengan nama table
- Buat class baru menggunakan nama TableUser

```java
package table;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.User;

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) {
        switch (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;
        }
    }
}
```

7. Membuat fungsi DAO

- Buat package baru dengan nama DAO
- Buat class interface baru dengan nama UserDAO

ArrayList1.java | ArraylistGe... | ArrayListSi... | DaftarSiswa... | *Database.java | *UserFrame.java | *TableUser.java | *UserDAO.java

```java
package DAO;

import java.util.List;

import model.User;

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

}
```

Terdapat method save, show, delete dan update. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

8. Menggunakan Fungsi DAO.

- Buat class baru dengan nama UserRepo
- Implementasikan UserDAO dengan kata kunci implements
- Membuat instanisasi Connection,constructor dan String untuk melakukan manipulasi database

```java
package DAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import config.Database;
import model.User;

public class UserRepo implements UserDAO {

    private Connection connection;
    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 user SET name=?, username=?, password=? WHERE id=?;";

    public UserRepo() {
        connection = Database.koneksi();
    }

    @Override
    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();
            }
        }
    }

    @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;
    }

    @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();
            }
        }
    }

    @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();
            }
        }
    }


}
```

C. KESIMPULAN

Setiap fungsi CRUD diimplementasikan secara fungsional:

a. Create: Menambahkan data user baru ke tabel user dengan memanfaatkan PreparedStatement.
b. Read: Menampilkan seluruh data user dari database ke dalam tabel GUI menggunakan ResultSet.
c. Update: Memperbarui informasi user berdasarkan ID yang dipilih.
d. Delete: Menghapus data user secara spesifik dengan parameter ID.

Koneksi database berhasil dilakukan melalui driver MySQL Connector/J.