LAPORAN PRAKTIKUM 3
PEMROGRAMAN BERBASIS OBJEK

Disusun Oleh :

SASKIA ALIFAH (2411531002)

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

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

Tahun 2025

LaundryApps

A. TUJUAN PRAKTIKUM

1. Memahami cara mengintegrasikan Java dengan MySQL Java Database Connectivitymenggunakan MySQL Connector.
2. Menerapkan operasi CRUD (Create, Read, Update, Delete)
3. Membangun aplikasi sederhana berbasis database yang dapat mengelola data pengguna (user) dengan tampilan (UI), logika program (DAO), dan database.

B. PENDAHULUAN

Dalam pembuatan aplikasi berbasis Java, penggunaan database menjadi hal penting untuk menyimpan serta mengelola data agar lebih terstruktur. Java sendiri dapat diintegrasikan dengan berbagai jenis database melalui JDBC (Java Database Connectivity), salah satunya dengan MySQL. Agar aplikasi Java dapat terhubung dengan MySQL, diperlukan driver tambahan berupa MySQL Connector yang berfungsi sebagai jembatan komunikasi antara aplikasi dan database. Dengan koneksi tersebut, program mampu melakukan operasi dasar pengolahan data seperti menambah, menampilkan, memperbarui, dan menghapus data (CRUD).

Database yang digunakan pada praktikum ini adalah MySQL yang dijalankan melalui XAMPP dengan bantuan phpMyAdmin sebagai interface untuk memudahkan pembuatan database dan tabel. Pada contoh ini, dibuat database dengan nama **laundry_apps** dan tabel **user** yang memiliki beberapa field seperti *id, name*, *username,* dan *password*. Tabel inilah yang akan menjadi media penyimpanan data user dan selanjutnya dihubungkan ke aplikasi Java. Dengan begitu, mahasiswa dapat memahami proses kerja sistem penyimpanan data dan cara menghubungkannya ke dalam sebuah aplikasi.

Selain itu, untuk menampilkan data dari database ke dalam aplikasi, digunakan komponen JTable yang dipadukan dengan AbstractTableModel sehingga data dapat ditampilkan secara rapi dalam bentuk tabel pada GUI. Melalui latihan ini, mahasiswa tidak hanya mempelajari cara membuat koneksi database menggunakan class Database dan

mengujinya dengan TestKoneksi, tetapi juga memahami konsep implementasi CRUD dalam aplikasi berbasis Java Swing

C. METODE PRAKTIKUM

1. Menambahkan method reset

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

Metode reset() berfungsi untuk mengembalikan seluruh kolom input pada form ke keadaan awal atau kosong setelah dilakukan suatu proses seperti penyimpanan, pembaruan, atau penghapusan data. Di dalam metode ini, setiap komponen input teks seperti txtName, txtUsername, dan txtPassword diatur ulang dengan memberikan nilai string kosong melalui perintah setText(""). Dengan demikian, seluruh data yang sebelumnya dimasukkan oleh pengguna akan dihapus dari kolom input, sehingga form siap digunakan kembali untuk memasukkan data baru.

2. Membuat instance pada Userframe

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

UserRepo usr = new UserRepo(); digunakan untuk membuat objek usr dari kelas UserRepo, yang berfungsi sebagai penghubung antara aplikasi dan data pengguna, termasuk operasi seperti menyimpan, menampilkan, memperbarui, dan menghapus data user. Selanjutnya, List<User> ls; adalah sebuah variabel daftar (list) yang nantinya menampung kumpulan objek User sehingga data pengguna dapat diolah dan ditampilkan dalam bentuk tabel. Sementara itu, public String id; digunakan untuk menyimpan nilai **ID** dari data pengguna yang sedang dipilih pada tabel, sehingga ID tersebut bisa menjadi acuan saat melakukan proses update maupun delete data.

3. Kode simpan data user

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        usr.save(user);
        reset();
        loadTable();
    }
});
```

Kode tersebut membuat sebuah tombol bernama **Save** (btnSave). Saat tombol ditekan, akan dijalankan sebuah **ActionListener** yang berisi logika penyimpanan data. Pertama, objek User baru dibuat, lalu nilai dari form input (txtName, txtUsername, txtPassword) diambil dan dimasukkan ke objek user dengan method setter. Setelah itu, data user disimpan ke database melalui usr.save(user). Untuk menjaga form tetap bersih, dipanggil method reset() agar input kosong kembali, kemudian loadTable() digunakan untuk memperbarui tampilan tabel sehingga data yang baru disimpan langsung muncul.

4. **Proses Memuat Data ke Tabel**

```
public void loadTable() {
    ls = usr.show();
    TableUser tu = new TableUser(ls);
    tableUsers.setModel(tu);
    tableUsers.getTableHeader().setVisible(true);
}
```

menampilkan data ke dalam tabel pada aplikasi Java Swing. Pertama, data ditarik melalui pemanggilan usr.show() dan disimpan ke dalam variabel ls. Data ini kemudian dimasukkan ke dalam objek TableUser yang berfungsi sebagai model tabel. Selanjutnya, model tersebut diatur ke komponen tableUsers menggunakan setModel(tu), sehingga tabel menampilkan data sesuai model. Terakhir, baris tableUsers.getTableHeader().setVisible(true); memastikan bahwa header tabel tetap ditampilkan agar kolom-kolom dapat terlihat jelas.

5. **Menjalankan Frame dan Memuat Data Tabel**

```
public void run() {
    try {
        UserFrame frame = new UserFrame();
        frame.setVisible(true);
        frame.loadTable();
```

implementasi dari metode run() yang digunakan untuk menampilkan antarmuka pengguna. Di dalam blok try, dibuat objek baru dari kelas UserFrame yang merepresentasikan jendela utama aplikasi. Setelah itu, perintah frame.setVisible(true); dipanggil agar jendela tersebut tampil di layar. Selanjutnya, frame.loadTable(); digunakan untuk memuat data ke dalam tabel yang ada di dalam frame tersebut. Dengan demikian, saat program dijalankan, antarmuka pengguna langsung muncul dan tabel di dalamnya otomatis terisi dengan data yang tersedia.

6. Menampilkan Data Tabel ke TextField Saat Baris Diklik

```
public void mouseClicked(MouseEvent e) {
    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());
}
```

Potongan kode tersebut berfungsi untuk mengambil data dari tabel ketika salah satu baris diklik oleh pengguna. Metode mouseClicked menangkap event klik pada tabel, kemudian mengambil nilai dari baris yang dipilih menggunakan getSelectedRow() dan kolom tertentu melalui getValueAt(). Nilai pada kolom pertama disimpan ke variabel id, sedangkan nilai pada kolom berikutnya digunakan untuk mengisi txtName, txtUsername, dan txtPassword. Dengan demikian, saat pengguna memilih baris tertentu pada tabel, data secara otomatis ditampilkan di dalam field teks yang sesuai.

7. Proses Update Data Pengguna

```
btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setPassword(txtPassword.getText());
        user.setId(id);
        usr.update(user);
        reset();
        loadTable();
    }
});
```

mengatur fungsi dari tombol **Update** pada aplikasi. Ketika tombol ditekan, sistem membuat objek User baru, lalu mengisi atributnya dengan data yang diambil dari field input seperti txtName, txtUsername, dan txtPassword. Selain itu, nilai id juga diatur agar aplikasi mengetahui data mana yang sedang diperbarui. Setelah semua data terkumpul, method usr.update(user) dipanggil untuk menyimpan perubahan ke dalam database atau daftar data. Kemudian, fungsi reset() membersihkan field input agar siap

digunakan kembali, dan loadTable() dijalankan untuk memuat ulang tabel sehingga menampilkan data terbaru yang sudah diperbarui.

8. Proses Hapus Data Pengguna

```
btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            usr.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
        }
    }
});
```

mengatur fungsi tombol **Delete** pada aplikasi. Saat tombol ditekan, sistem akan memeriksa apakah ada id data yang dipilih dari tabel. Jika id tidak bernilai null, maka method usr.delete(id) dipanggil untuk menghapus data sesuai id tersebut, kemudian reset() dijalankan untuk mengosongkan field input, dan loadTable() dipanggil kembali agar tabel langsung menampilkan data terbaru setelah penghapusan. Namun, jika tidak ada data yang dipilih, maka akan muncul dialog peringatan menggunakan JOptionPane.showMessageDialog dengan pesan *"Silahkan pilih data yang akan dihapus"*, sehingga pengguna diarahkan untuk memilih data terlebih dahulu sebelum melakukan penghapusan.

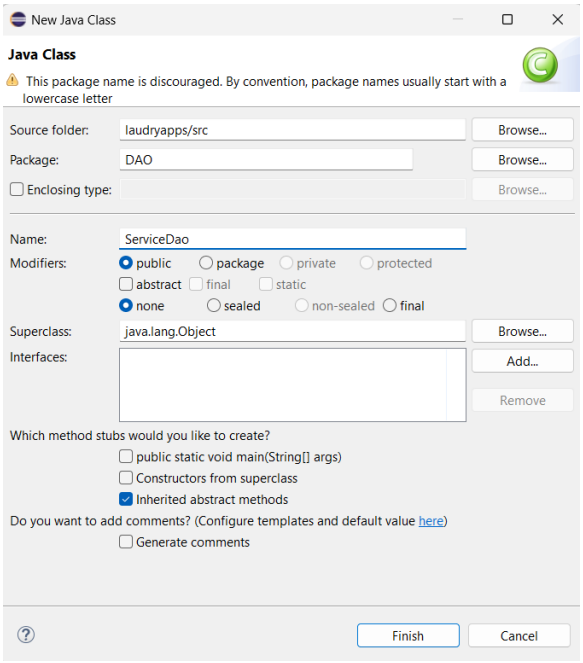
Latihan/Tugas

Membuat fungsi CRUD untuk layanan dan pelanggan

1. Buka phpMyAdmin lalu buat database untuk tabel layanan dan pelanggan.
   - Tabel pelanggan

   - Tabel layanan

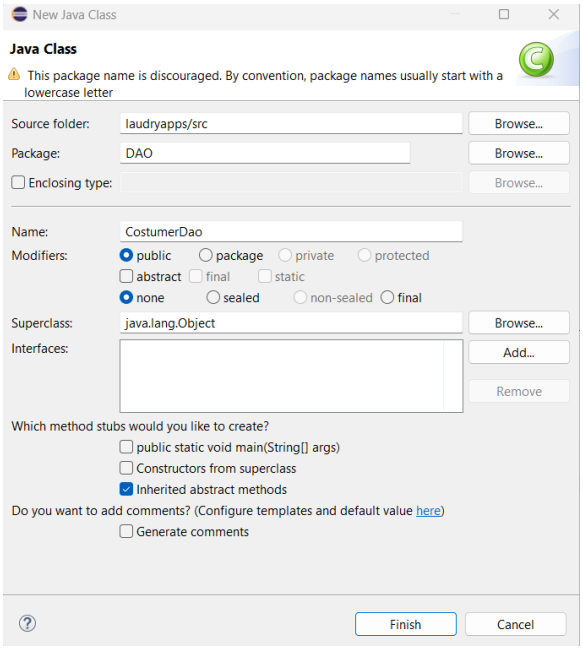
2. Selanjutnya pada package DAO buat interface baru dengan nama ServiceDAO.




```java
package DAO;

import java.util.List;

import model.Service;

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

merupakan **interface ServiceDao** yang berfungsi sebagai kontrak (blueprint) untuk operasi CRUD (Create, Read, Update, Delete) pada entitas **Service** di aplikasi Java. Interface ini berada pada package **DAO** dan mendefinisikan empat method yang wajib diimplementasikan oleh class repository atau DAO yang mengakses database. Method save(Service service) digunakan untuk menyimpan data service baru ke dalam database. Method show() mengembalikan daftar semua objek Service yang tersimpan, sehingga bisa ditampilkan misalnya di tabel pada GUI. Method delete(String id) digunakan untuk menghapus data service berdasarkan **id** tertentu, sedangkan method update(Service service) berfungsi untuk memperbarui data service yang sudah ada dengan nilai baru yang dikirim melalui objek Service.

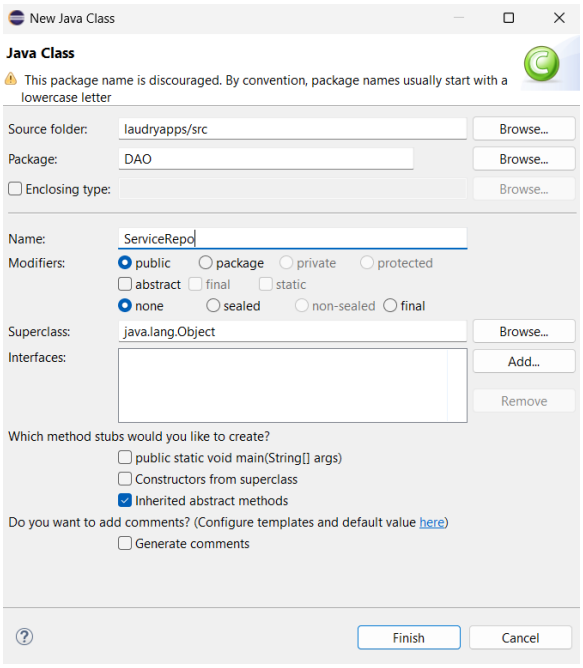
3. Pada package DAO buat interface baru dengan nama CostumerDAO.




```java
package DAO;

import java.util.List;

import model.Costumer;

public interface CostumerDao {
    void save(Costumer costumer);
    public List<Costumer> show();
    public void delete (String id);
    public void update (Costumer costumer);
}
```

merupakan **interface CostumerDao** yang berfungsi sebagai kontrak untuk operasi CRUD (Create, Read, Update, Delete) pada entitas **Costumer** di aplikasi Java. Interface ini berada dalam package **DAO** dan mendefinisikan empat method abstrak yang wajib diimplementasikan oleh class repository atau DAO yang menangani akses data costumer ke database. Method save(Costumer costumer) digunakan untuk menyimpan data costumer baru ke database. Method show() mengembalikan daftar semua objek Costumer yang tersimpan, misalnya untuk ditampilkan pada JTable di GUI. Method delete(String id) berfungsi untuk menghapus data costumer berdasarkan **id** tertentu, sedangkan method update(Costumer costumer) digunakan untuk memperbarui data costumer yang sudah ada sesuai informasi terbaru yang dikirim melalui objek Costumer.

4. Lalu pada package DAO buat class dengan nama ServiceRepo




```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.Service;


public class ServiceRepo implements ServiceDao{
    private Connection connection;
    final String insert = "INSERT INTO service (jenis, status, harga) VALUES (?,?,?);";
    final String select = "SELECT * FROM service;" ;
    final String delete = "DELETE FROM service WHERE id = ?;";
    final String update = "UPDATE service SET jenis = ?, status = ?, harga = ? WHERE id = ?;";

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

    @Override
    public void save(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, service.getJenis());
            st.setString(2, service.getStatus());
            st.setDouble(3, service.getHarga());
            st.executeUpdate();

        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
    @Override
    public List<Service> show() {
        List<Service> ls = null;
        try {
            ls = new ArrayList<Service>();
            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.setStatus(rs.getString("status"));
                service.setHarga(rs.getDouble("harga"));
                ls.add(service);
            }
        }catch(SQLException e) {
            Logger.getLogger(ServiceDao.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(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(update);
            st.setString(1, service.getJenis());
            st.setString(2, service.getStatus());
            st.setDouble(3, service.getHarga());
            st.setString(4, service.getId());
            st.executeUpdate();
        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e){
                e.printStackTrace();
            }
        }
    }
}
```

merupakan implementasi class **ServiceRepo** yang mengelola operasi **CRUD** (Create, Read, Update, Delete) untuk entitas **Service** pada database menggunakan JDBC. Class ini mengimplementasikan interface **ServiceDao** dan berada dalam package **DAO**.

1. **DeklarasidanKoneksi**

   Variabel connection menyimpan objek koneksi database yang diperoleh melalui Database.koneksi(). Class juga mendefinisikan query SQL untuk operasi INSERT, SELECT, DELETE, dan UPDATE sebagai variabel String.

2. **Methodsave(Serviceservice)**

   Method ini digunakan untuk menambahkan data service baru ke database. Menggunakan PreparedStatement, nilai **jenis**, **status**, dan **harga** diambil dari objek Service dan dieksekusi melalui executeUpdate(). Bagian finally memastikan statement ditutup agar tidak terjadi memory leak.

3. **Methodshow()**

   Method ini mengambil seluruh data service dari database. Menggunakan Statement untuk mengeksekusi query SELECT, kemudian setiap baris hasil (ResultSet) diubah menjadi objek Service dan ditambahkan ke list ls. List ini dikembalikan untuk ditampilkan di JTable.

4. **Methoddelete(Stringid)**

   Method ini menghapus data service berdasarkan **id** tertentu. Menggunakan PreparedStatement untuk mengeksekusi query DELETE.

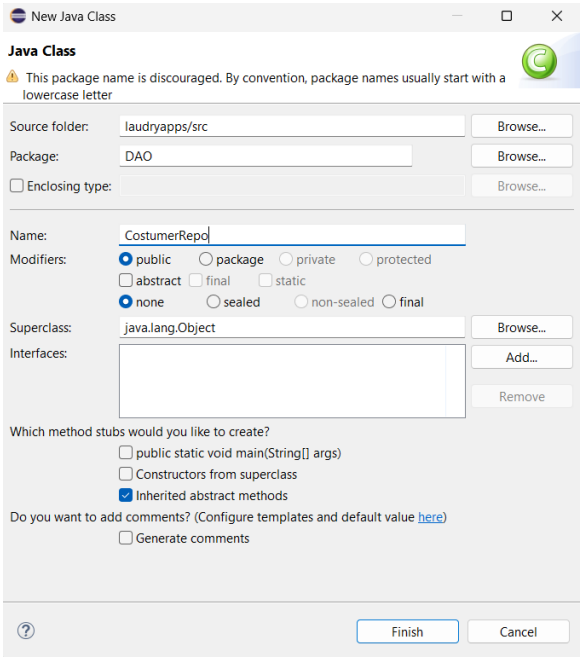
5. **Methodupdate(Serviceservice)**

   Method ini memperbarui data service yang sudah ada berdasarkan **id** tertentu. Menggunakan PreparedStatement, field **jenis**, **status**, dan **harga** di-update sesuai data baru dari objek Service.

Secara keseluruhan, class ini bertanggung jawab untuk mengelola interaksi langsung dengan database terkait data service. Dengan struktur DAO, logika akses data

dipisahkan dari GUI dan logika bisnis, memudahkan pemeliharaan kode dan implementasi prinsip **Data Access Object (DAO)**.

5. Lalu pada package DAO buat class baru untuk pelanggan dengan nama CoatumerRepo.




```
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.Costumer;


public class CostumerRepo implements CostumerDao{
    private Connection connection;
    final String insert = "INSERT INTO costumer (nama, alamat, nohp) VALUES (?,?,?);";
    final String select = "SELECT * FROM costumer;" ;
    final String delete = "DELETE FROM costumer WHERE id = ?;";
    final String update = "UPDATE costumer SET nama=?, alamat=?, nohp=? WHERE id=?;";

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

    @Override
    public void save(Costumer costumer) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, costumer.getName());
            st.setString(2, costumer.getAlamat());
            st.setString(3, costumer.getNomorHp());
            st.executeUpdate();

        }catch(SQLException e) {
            e.printStackTrace();
        }finally {
            try {
                st.close();
            }catch(SQLException e) {
```

```
            e.printStackTrace();
        }
    }
}

public List<Costumer> show(){
    List<Costumer> ls = null;
    try {
        ls = new ArrayList<Costumer>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            Costumer costumer = new Costumer();
            costumer.setId(rs.getString("id"));
            costumer.setNama(rs.getString("nama"));
            costumer.setAlamat(rs.getString("alamat"));
            costumer.setNomorHp(rs.getString("nohp"));
            ls.add(costumer);
        }
    }catch(SQLException e) {
        Logger.getLogger(CostumerDao.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}

@Override
public void update (Costumer costumer) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, costumer.getName());
        st.setString(2, costumer.getAlamat());
        st.setString(3, costumer.getNomorHp());
        st.setString(4, costumer.getId());
        st.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();
    }finally {
        try {
            st.close();
        }catch(SQLException e){
            e.printStackTrace();
        }

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();
        }
    }
}

}
```

merupakan implementasi class **CostumerRepo** yang mengelola operasi CRUD (Create, Read, Update, Delete) untuk entitas **Costumer** pada database menggunakan JDBC. Class ini mengimplementasikan interface **CostumerDao** dan berada di package **DAO**.

1. **DeklarasidanKoneksi**

   Variabel connection menyimpan objek koneksi database yang diperoleh melalui Database.koneksi(). Class juga mendefinisikan query SQL sebagai String untuk operasi insert, select, delete, dan update.

2. **Methodsave(Costumercostumer)**

   Method ini digunakan untuk menyimpan data pelanggan baru ke database. Menggunakan PreparedStatement, nilai **nama**, **alamat**, dan **nohp** diambil dari objek Costumer dan dieksekusi dengan executeUpdate(). Bagian finally memastikan statement ditutup setelah eksekusi untuk mencegah memory leak.

3. **Methodshow()**

   Method ini mengambil seluruh data pelanggan dari database. Menggunakan Statement untuk mengeksekusi query SELECT, kemudian setiap baris hasil (ResultSet) diubah menjadi objek Costumer dan ditambahkan ke list ls. List ini dikembalikan untuk ditampilkan di JTable.

4. **Method update(Costumer costumer)**

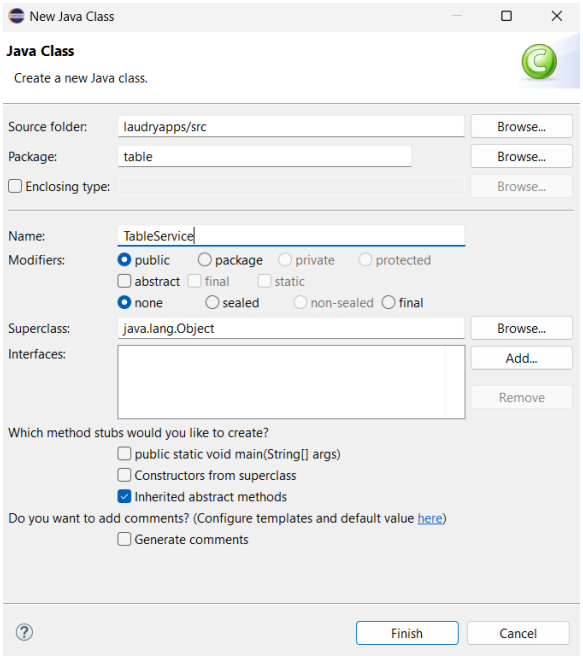
   Method ini memperbarui data pelanggan berdasarkan id tertentu. Menggunakan PreparedStatement, field **nama**, **alamat**, dan **nohp** di-update sesuai data baru dari objek Costumer.

5. **Methoddelete(Stringid)**

   Method ini menghapus data pelanggan berdasarkan id. Menggunakan PreparedStatement untuk mengeksekusi query DELETE.

Secara keseluruhan, class ini bertanggung jawab untuk komunikasi langsung dengan database untuk data pelanggan, memisahkan logika database dari GUI atau logika bisnis, sehingga memudahkan pemeliharaan kode dan implementasi prinsip **DAO (Data Access Object)**.

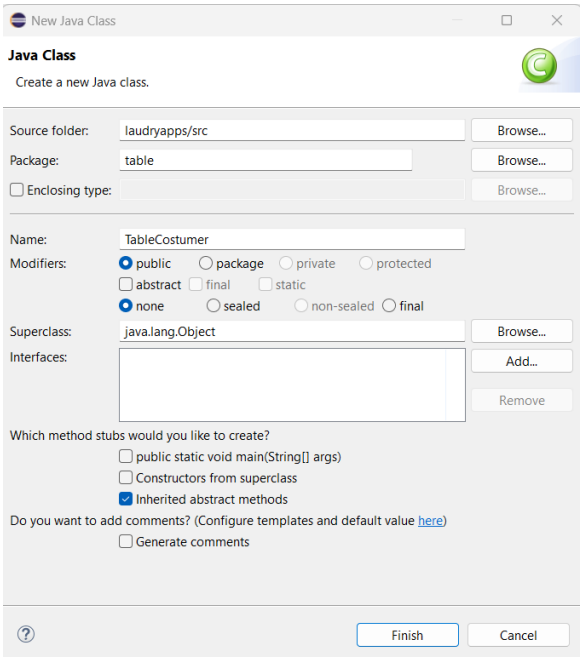
6. Selanjutnya buat class TableService pada package table.



```java
package table;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import model.Service;

public class TableService extends AbstractTableModel{
    List<Service> ls;
    private String[] columnNames = {"ID", "Jenis", "Status", "Harga"};
    public TableService(List<Service> ls) {
        this.ls = ls;
    }

    @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 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).getJenis();
        case 2:
            return ls.get(rowIndex).getStatus();
        case 3:
            return ls.get(rowIndex).getHarga();
        default:
            return null;
        }
    }
```

merupakan class **TableService** yang berfungsi sebagai model untuk menampilkan data **Service** pada JTable di Java Swing. Class ini berada dalam package **table** dan meng-extend AbstractTableModel, sehingga dapat digunakan untuk mengatur baris, kolom, dan nilai yang ditampilkan di tabel. Variabel ls menyimpan daftar objek Service, sedangkan columnNames berisi judul kolom: ID, Jenis, Status, dan Harga. Method getRowCount() mengembalikan jumlah baris sesuai jumlah data Service, getColumnCount() mengembalikan jumlah kolom (4), dan getValueAt(int rowIndex, int columnIndex) menentukan nilai yang ditampilkan pada setiap sel berdasarkan baris dan kolom.

7. Selanjutnya pada package table juga, buat class untuk TableCostumer.

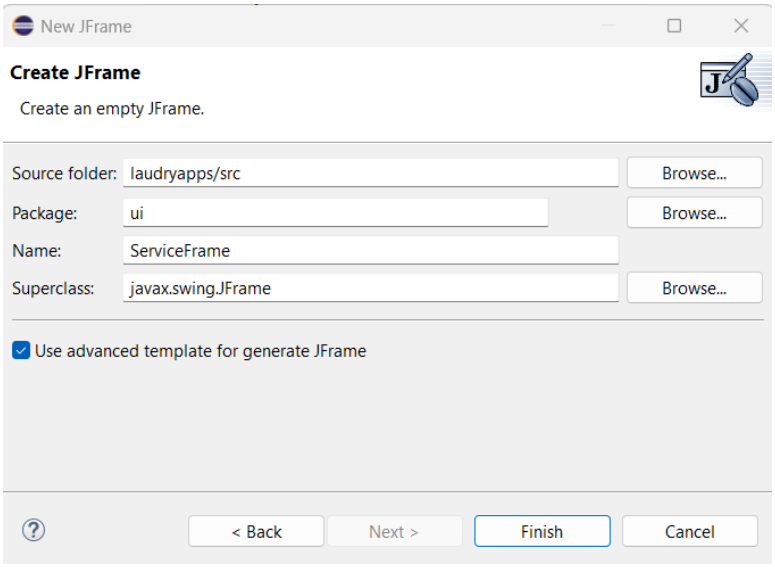


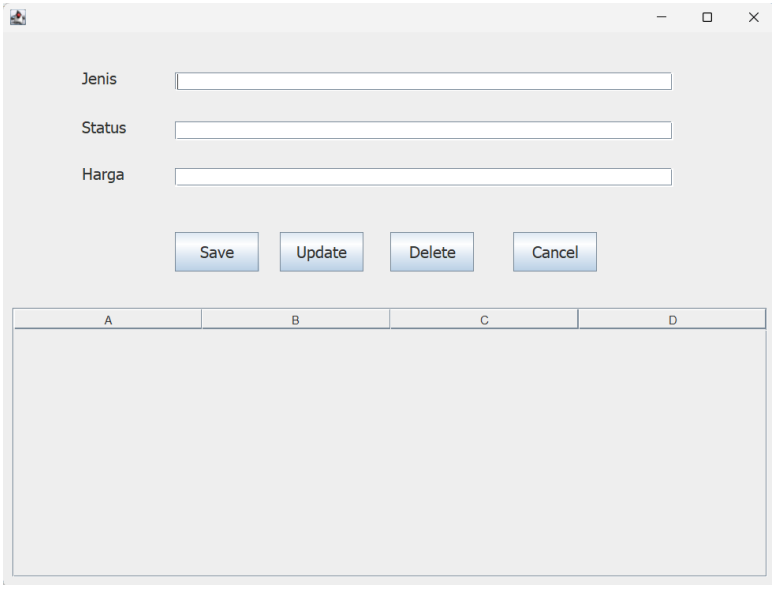

```java
package table;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.Costumer;
public class TableCostumer extends AbstractTableModel {
    List<Costumer> ls;
    private String[] columnNames = {"ID", "Nama", "Alamat", "Nohp"};
    public TableCostumer(List<Costumer> ls) {
        this.ls = ls;
    }
        @Override
    public int getRowCount() {
        return ls.size();
    }
        @Override
    public int getColumnCount() {
        return 4;
    }
        @Override
    public String getColumnName(int colomn) {
        return columnNames[colomn];
    }
        @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
        case 0:
            return ls.get(rowIndex).getId();
        case 1:
            return ls.get(rowIndex).getName();
        case 2:
            return ls.get(rowIndex).getAlamat();
        case 3:
            return ls.get(rowIndex).getNomorHp();
        default:
            return null;
        }
    }
        public Costumer getCostumerAt(int rowIndex) {
        return ls.get(rowIndex);

    }
}
```

merupakan class **TableCostumer** yang dibuat untuk menampilkan data **Costumer** pada komponen JTable di Java Swing. Class ini berada dalam package **table** dan meng-extend AbstractTableModel, sehingga dapat digunakan sebagai model data untuk JTable. Variabel ls menyimpan daftar objek Costumer yang akan ditampilkan, sedangkan columnNames berisi judul kolom tabel: ID, Nama, Alamat, dan NoHP. Method getRowCount() mengembalikan jumlah baris sesuai jumlah data, getColumnCount() mengembalikan jumlah kolom, getColumnName(int column) memberikan nama kolom berdasarkan index, dan getValueAt(int rowIndex, int columnIndex) mengatur data yang ditampilkan di setiap sel tabel. Terdapat juga method

getCostumerAt(int rowIndex) untuk mengambil objek Costumer pada baris tertentu, yang memudahkan pengambilan data ketika user memilih baris di JTable.

8. Selanjutnya Untuk tampilan GUI nya silakan menggunakan ini.
9. Buat pada package ui JFrame baru dengan nama LayananFrame





CostumerFrame (CRUD Costumer)

Bagian ini merupakan langkah untuk membuat tampilan CRUD (Create, Read, Update, Delete) Costumer menggunakan JFrame pada package ui dengan nama class CostumerFrame. Tampilan ini adalah antarmuka (GUI) yang digunakan untuk mengelola data pelanggan di aplikasi.

Form Input Data

- JTextField txtNama → digunakan untuk memasukkan nama pelanggan.

- JTextField txtAlamat → digunakan untuk memasukkan alamat pelanggan.
- JTextField txtNomorHP → digunakan untuk memasukkan nomor HP pelanggan.

Ketiga textfield ini menjadi komponen utama untuk menambahkan data baru maupun mengedit data yang sudah ada.

Tombol Aksi (CRUD Buttons)

- *JButton btnSave*
- Berfungsi untuk menyimpan data costumer baru ke database.
- ActionListener:
    - Membuat objek baru Costumer.
    - Mengambil data dari textfield (txtNama, txtAlamat, txtNomorHP).
    - Memanggil method cst.save(costumer) dari CostumerRepo untuk menyimpan ke database.
    - Memanggil reset() untuk mengosongkan form.
    - Memanggil loadTable() untuk menampilkan data terbaru di tabel.
- *JButton btnUpdate*
- Berfungsi untuk memperbarui data costumer yang sudah ada.
- ActionListener:
    - Mengecek apakah ada baris yang dipilih di tabel (id != null).
    - Membuat objek Costumer baru dan mengambil nilai dari textfield.
    - Mengatur id costumer yang dipilih untuk update.
    - Memanggil cst.update(costumer) untuk memperbarui data di database.
    - Reset form dan reload tabel.
    - Jika belum memilih data, menampilkan JOptionPane: "Silahkan pilih data yang akan dihapus".
- *JButton btnDelete*
- Berfungsi untuk menghapus data costumer tertentu.
- ActionListener:
    - Mengecek apakah ada baris yang dipilih (id != null).
    - Memanggil cst.delete(id) untuk menghapus data dari database.
    - Reset form dan reload tabel.
    - Jika belum memilih data, menampilkan JOptionPane peringatan.

JButton btnCancel

- Berfungsi untuk membatalkan input atau keluar dari frame saat ini.
- ActionListener:
    - Membuka MainFrame.
    - Menutup (dispose()) frame CostumerFrame.

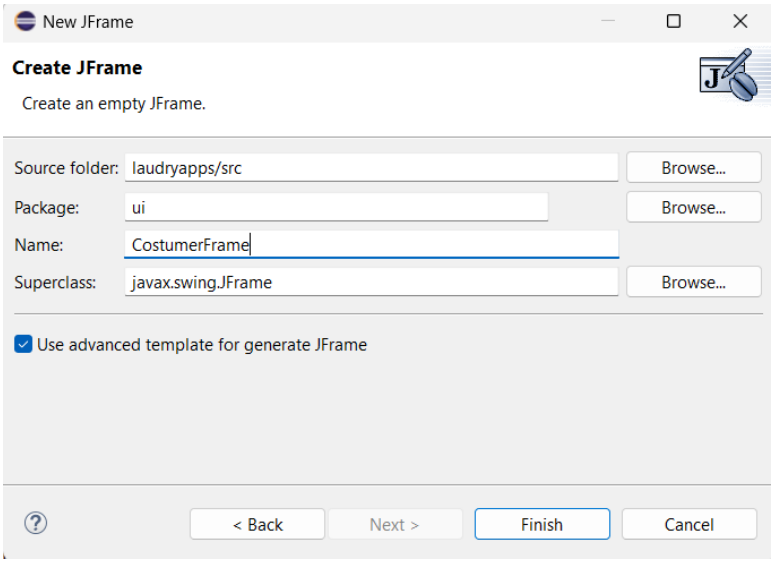
Tabel Data

- JTable tableCostumer → digunakan untuk menampilkan seluruh data pelanggan yang tersimpan di database.
- Data yang ditampilkan di tabel dapat dipilih untuk diedit atau dihapus.
- Tabel ditempatkan di dalam JScrollPane agar bisa discroll jika data banyak.

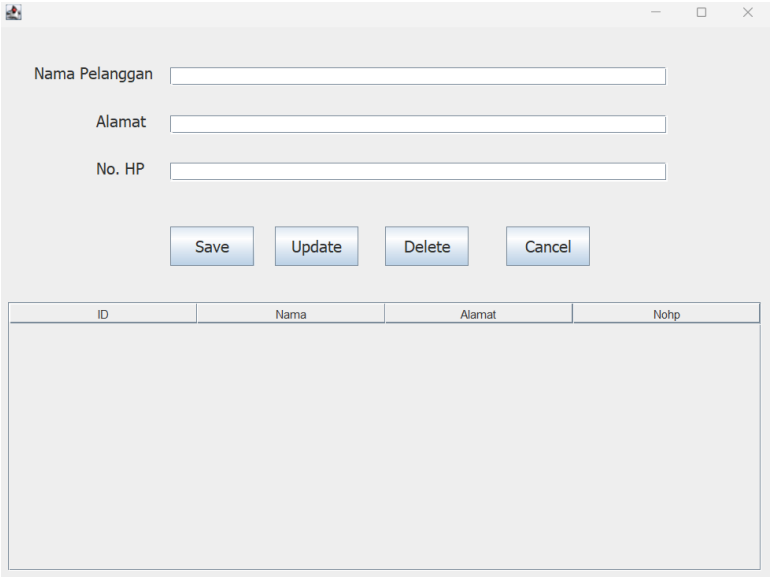
MouseListener

- Digunakan agar ketika user klik baris tabel:
    - id disimpan dari kolom 0 (ID costumer).
    - Textfield txtNama, txtAlamat, dan txtNomorHP otomatis terisi sesuai data baris yang diklik.
- Ini mempermudah user untuk melakukan update atau delete.

10. Selanjutnya untuk pelanggan buat JFrame baru pada package ui dengan nama CostumerFrame.



11. Lalu desain seperti ini.



Bagian ini merupakan langkah untuk membuat tampilan CRUD (Create, Read, Update, Delete) Service menggunakan JFrame Tabel ditempatkan di **JScrollPane** agar bisa discroll jika pada package ui dengan nama class ServiceFrame. Tampilan ini adalah antarmuka (GUI) yang akan digunakan untuk mengelola data service di dalam aplikasi.

Form Input Data

- JTextField txtJenis → digunakan untuk menginput jenis service.
- JTextField txtStatus → digunakan untuk menginput status service.
- JTextField txtHarga → digunakan untuk menginput harga service.

Ketiga text field ini menjadi komponen utama untuk memasukkan data service baru maupun mengedit data service yang sudah ada.

Tombol Aksi (CRUD Buttons)

- JButton btnSave → berfungsi untuk menyimpan data service baru ke dalam database.
    - Saat diklik, ActionListener akan membuat objek Service baru, mengambil data dari textfield, menyimpannya melalui ServiceRepo.save(), mereset form, dan memuat ulang tabel.
- JButton btnUpdate → digunakan untuk memperbarui data service yang sudah ada.
    - Mengecek apakah user sudah memilih baris di tabel (id != null), lalu update data melalui ServiceRepo.update().
- JButton btnDelete → digunakan untuk menghapus data service tertentu.

- Mengecek id yang dipilih, lalu menghapus data melalui ServiceRepo.delete().

JButton btnCancel → berfungsi untuk membatalkan input atau menutup frame saat ini, lalu kembali ke MainFrame.

Keempat tombol ini merupakan implementasi dari fungsi CRUD yang terhubung ke logika database.

Tabel Data

- JTable tableService → berfungsi untuk menampilkan seluruh data service yang sudah tersimpan di database.
- Data yang ditampilkan pada tabel dapat dipilih untuk kemudian diedit atau dihapus.
- MouseListener digunakan agar saat user klik baris tabel, textfield otomatis terisi dengan data baris tersebut.

D. KESIMPULAN

Berdasarkan hasil praktikum, dapat disimpulkan bahwa proses menghubungkan aplikasi Java dengan database MySQL membutuhkan konfigurasi yang tepat, mulai dari menambahkan MySQL Connector hingga membuat database dan tabel melalui phpMyAdmin. Dengan adanya koneksi tersebut, aplikasi dapat melakukan operasi dasar pengolahan data (CRUD) seperti menambahkan, menampilkan, memperbarui, dan menghapus data. Hal ini menunjukkan pentingnya pemahaman konsep JDBC sebagai jembatan komunikasi antara aplikasi Java dan database.

Selain itu, penerapan JTable dengan AbstractTableModel memudahkan penyajian data secara terstruktur dalam aplikasi berbasis GUI. Mahasiswa juga dapat memahami bagaimana alur kerja program dari mulai input data, penyimpanan ke database, hingga menampilkannya kembali ke layar aplikasi. Praktikum ini menjadi dasar penting dalam pembuatan aplikasi yang lebih kompleks, karena mampu melatih pemahaman mahasiswa tentang integrasi Java dengan database serta penerapan prinsip pemrograman berorientasi objek dalam membangun sistem informasi sederhana.