LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete) USER

DENGAN DATABASE MYSQL

OLEH :

Abdullah Al Ramadhani

2411533016

DOSEN PENGAMPU:

NURFIAH, S.ST, M.Kom,

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

PADANG 2025

A. Pendahuluan

Di era digital saat ini, pengembangan aplikasi perangkat lunak memegang peranan krusial dalam berbagai sektor industri. Java, sebagai salah satu bahasa pemrograman yang paling populer dan serbaguna, sering digunakan untuk membangun aplikasi, mulai dari skala kecil hingga sistem enterprise yang kompleks. Salah satu tantangan terbesar dalam pengembangan perangkat lunak adalah mengelola kompleksitas kode agar tetap terstruktur, mudah dipelihara (*maintainable*), dan dapat dikembangkan lebih lanjut (*scalable*). Tanpa arsitektur yang dirancang dengan baik, aplikasi dapat menjadi monolitik dan sulit untuk dimodifikasi tanpa menimbulkan masalah baru.

Untuk mengatasi tantangan ini, para pengembang perangkat lunak mengadopsi berbagai pola arsitektur (*architectural patterns*) dan pola desain (*design patterns*). Pola-pola ini menyediakan solusi teruji untuk masalah-masalah umum dalam desain perangkat lunak. Beberapa prinsip yang paling fundamental adalah Pemisahan Tanggung Jawab (Separation of Concerns), yang diwujudkan melalui pola seperti Model-View-Controller (MVC) dan Data Access Object (DAO). Pola-pola ini bertujuan untuk memisahkan logika yang berkaitan dengan data, logika bisnis, dan antarmuka pengguna, sehingga setiap komponen dapat dikembangkan dan diuji secara independen.

B. Tujuan

1) Menjelaskan implementasi arsitektur perangkat lunak yang menerapkan prinsip Pemisahan Tanggung Jawab.
2) Mendemonstrasikan cara kerja operasi CRUD (Create, Read, Update, Delete).

C. Langkah-langkah

1) Buat fungsi reset(), yang dimana fungsi ini biasanya dipanggil setelah pengguna berhasil menyimpan data atau ketika pengguna menekan tombol **Batal** atau **Reset**, sehingga formulir kembali ke keadaan awal yang bersih dan siap untuk diisi kembali.

```
public void reset() {
    txtNama.setText("");
    txtEmail.setText("");
    txtTelepon.setText("");
    selectedId = null;
}
```

2) Buat instance pada UserFrame

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

kode ini menyiapkan tiga hal penting untuk mengelola data user: alat untuk mengatur data (UserRepo), tempat untuk menyimpan daftar user (List<User>), dan variabel untuk menampung ID user yang sedang aktif (id).

3) Untuk mengaktifkan tombol save, pertama-tama klik kanan pada tombol tersebut. Dari menu yang tampil, arahkan kursor ke add event handlers dan pilih opsi actionPerformed. Terakhir, masukkan kode program yang telah disediakan ke dalam blok fungsi yang muncul.

```
User user = new User();
user.setNama(txtName.getText());
user.setUsername(txtUsername.getText());
user.setPassword(txtPassword.getText());
usr.save(user);
reset();
```

Codingan diatas mengambil data dari formulir, menyimpannya ke database, lalu langsung mengosongkan formulirnya kembali.

4) Langkah berikutnya adalah mendeklarasikan sebuah method baru. Beri nama method tersebut loadTable() dan letakkan kode program berikut di dalamnya.

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

ini adalah untuk mengambil data pengguna dari database dan menampilkannya ke dalam sebuah tabel di antarmuka aplikasi.

5) Agar tabel langsung terisi saat aplikasi dibuka, panggil method loadTable() dari dalam *constructor* kelas utama. Dengan begitu, method ini akan otomatis berjalan setiap kali program dimulai.

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

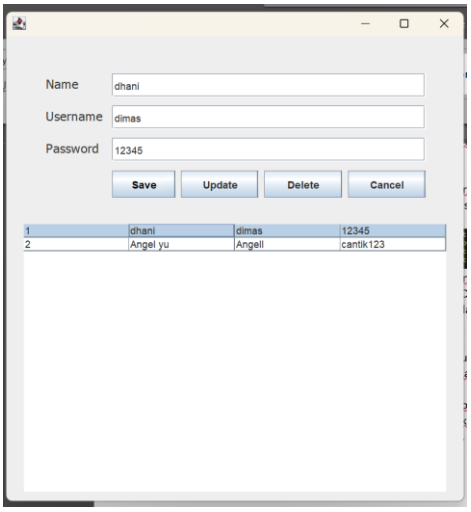
Codingan ini berfungsi untuk menjalankan dan menampilkan jendela utama aplikasi kepada pengguna.

6) buat tabelnya agar bisa bereaksi saat di-klik. Caranya, klik kanan di atas tabel, cari menu add event handler, lanjutkan ke pilihan mouse, kemudian klik mouseClicked, lalu isikan dengan codingan dibawah ini.

```
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 ini berfungsi untuk menampilkan detail pengguna ke formulir. Saat salah satu pengguna dipilih, program akan menyimpan ID-nya, lalu memakai ID itu untuk menemukan nama, username, serta password, dan menampilkannya di kolom yang sesuai.

7) Jika sudah selesai menulis codingan tersebut, run codingan tersebut lalu mengklik salah satu table maka secara otomatis menampilkan pada form inputan



8) Selanjutnya untuk memberikan fungsi pada tombol update, klik kanan tombol tersebut. Dari menu yang muncul, navigasikan ke add event handler, lanjutkan ke submenu action, dan pilih opsi actionPerformed. Terakhir, masukkan kode program berikut ke dalam method yang telah dibuat.

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

9) Pilih salah satu data pada JTable.

10) Kemudian, buat *event* untuk tombol 'delete' dengan cara klik kanan tombolnya, lalu pilih add event handler, lanjutkan ke action, dan klik actionPerformed. Masukkan skrip berikut ke dalam blok kode yang tersedia.

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

- **Tugas/Latihan**

a) **Service**

1) Buat class baru pada package DAO, dengan nama **ServiceRepo**

```java
package DAO;

import java.sql.Connection;

public class ServiceRepo {

    public void save(Service service) {
        String sql = "INSERT INTO services (service_name, description, price) VALUES (?, ?, ?)";
        try (Connection conn = Database.koneksi();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, service.getServiceName());
            pstmt.setString(2, service.getDescription());
            pstmt.setDouble(3, service.getPrice());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public List<Service> show() {
        List<Service> services = new ArrayList<>();
        String sql = "SELECT * FROM services";
        try (Connection conn = Database.koneksi();
             PreparedStatement pstmt = conn.prepareStatement(sql);
             ResultSet rs = pstmt.executeQuery()) {
            while (rs.next()) {
                Service service = new Service();
                service.setId(rs.getString("id"));
                service.setServiceName(rs.getString("service_name"));
                service.setDescription(rs.getString("description"));
                service.setPrice(rs.getDouble("price"));
                services.add(service);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return services;
    }

    public void update(Service service) {
        String sql = "UPDATE services SET service_name = ?, description = ?, price = ? WHERE id = ?";
        try (Connection conn = Database.koneksi();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, service.getServiceName());
            pstmt.setString(2, service.getDescription());
            pstmt.setDouble(3, service.getPrice());
            pstmt.setString(4, service.getId());
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void delete(String id) {
        String sql = "DELETE FROM services WHERE id = ?";
        try (Connection conn = Database.koneksi();
             PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, id);
            pstmt.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

- **save(Service service):** Untuk menyimpan objek Service baru ke dalam tabel services.
- **show():** Untuk mengambil semua data dari tabel services dan menampilkannya.
- **update(Service service):** Untuk memperbarui data sebuah layanan yang sudah ada di database.
- **delete(String id):** Untuk menghapus sebuah data layanan dari database berdasarkan id-nya.

2) Buat class baru dengan nama **Service,** pada package **Model**

```java
package model;

public class Service {
    private String id;
    private String serviceName;
    private String description;
    private double price;

    // Konstruktor
    public Service() {}

    public Service(String id, String serviceName, String description, double price) {
        this.id = id;
        this.serviceName = serviceName;
        this.description = description;
        this.price = price;
    }

    // Getters and Setters
    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getServiceName() {
        return serviceName;
    }

    public void setServiceName(String serviceName) {
        this.serviceName = serviceName;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }
}
```

- **id, serviceName, description, dan price:** adalah properti atau data yang dimiliki oleh setiap objek Service.
- **public Service(...):** Ini adalah konstruktor dengan parameter. Berguna untuk membuat objek Service dan langsung mengisinya dengan semua data yang dibutuhkan dalam satu baris kode.
- **Getters, contohnya: getId(), getServiceName()):** Ini adalah metode publik untuk mengambil atau membaca nilai dari atribut yang bersifat private.
- **Setters, contohnya: setId(...), setServiceName(...)):** Ini adalah metode publik untuk mengatur atau mengubah nilai dari atribut yang bersifat private.

3) Buat class baru dengan nama **TableService,** pada package **Table.**

```java
package Table;

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

public class TableService extends AbstractTableModel {
    private List<Service> services;
    private final String[] columnNames = {"ID", "Service Name", "Description", "Price"};

    public TableService(List<Service> services) {
        this.services = services;
    }

    @Override
    public int getRowCount() {
        return services.size();
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;
    }

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

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        Service service = services.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return service.getId();
            case 1:
                return service.getServiceName();
            case 2:
                return service.getDescription();
            case 3:
                return service.getPrice();
            default:
                return null;
        }
    }
}
```

- **TableService(List<Service> services):** Ini adalah konstruktor yang menerima daftar data (List<Service>) yang ingin ditampilkan. Data ini disimpan dalam variabel services**.**
- **getRowCount():** untuk menentukan jumlah baris dalam tabel.
- **getColumnCount():** Untuk menentukan jumlah kolom dalam tabel.
- **getColumnName(int column):** Untuk memberikan nama atau judul untuk setiap kolom.
- **getValueAt(int rowIndex, int columnIndex):** Fungsinya adalah mengambil nilai spesifik untuk ditampilkan di sel tertentu (berdasarkan baris dan kolom).

4) Buat JFrame baru dengan nama **ServiceFrame,** pada package **ui.**

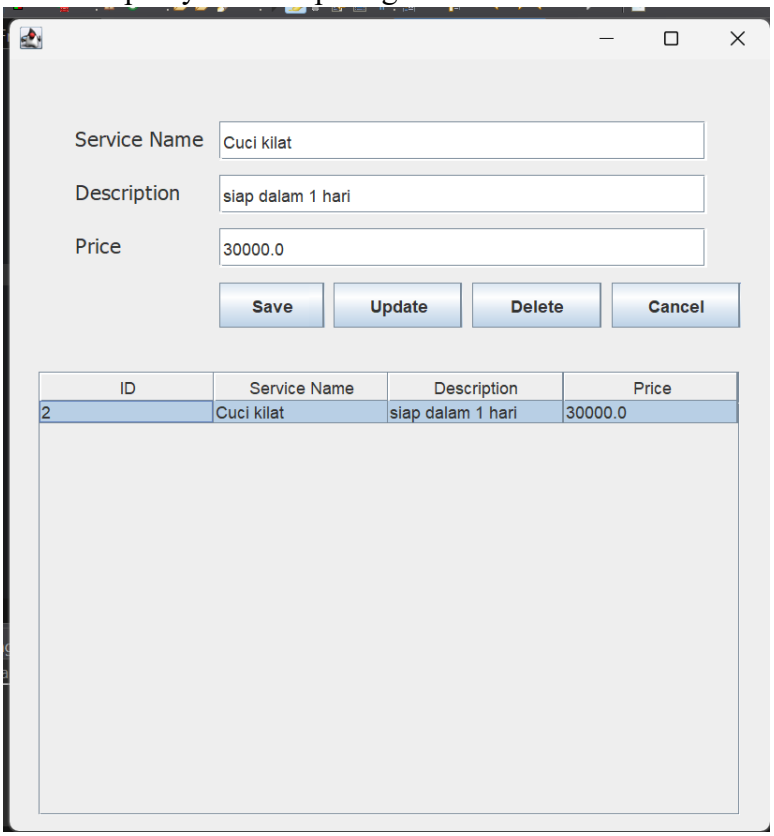
```java
package ui;

import java.awt.EventQueue;

public class ServiceFrame extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField txtServiceName;
    private JTextField txtDescription;
    private JTextField txtPrice;
    private JTable tableServices;
    private ServiceRepo serviceRepo = new ServiceRepo();
    private List<Service> ls;
    private String selectedId;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                ServiceFrame frame = new ServiceFrame();
                frame.setVisible(true);
                frame.loadTable();
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public void reset() {
        txtServiceName.setText("");
        txtDescription.setText("");
        txtPrice.setText("");
        selectedId = null;
    }

    public void loadTable() {
        ls = serviceRepo.show();
        TableService tu = new TableService(ls);
        tableServices.setModel(tu);
        tableServices.getTableHeader().setVisible(true);
    }

    public ServiceFrame() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 556, 586);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblServiceName = new JLabel("Service Name");
        lblServiceName.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblServiceName.setBounds(47, 40, 100, 32);
        contentPane.add(lblServiceName);
```

```
        JLabel lblDescription = new JLabel("Description");
        lblDescription.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblDescription.setBounds(47, 78, 92, 32);
        contentPane.add(lblDescription);

        JLabel lblPrice = new JLabel("Price");
        lblPrice.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblPrice.setBounds(47, 116, 70, 32);
        contentPane.add(lblPrice);

        txtServiceName = new JTextField();
        txtServiceName.setBounds(150, 44, 347, 28);
        contentPane.add(txtServiceName);
        txtServiceName.setColumns(10);

        txtDescription = new JTextField();
        txtDescription.setColumns(10);
        txtDescription.setBounds(150, 82, 347, 28);
        contentPane.add(txtDescription);

        txtPrice = new JTextField();
        txtPrice.setColumns(10);
        txtPrice.setBounds(150, 120, 347, 28);
        contentPane.add(txtPrice);
```

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(e -> {
    Service service = new Service();
    service.setServiceName(txtServiceName.getText());
    service.setDescription(txtDescription.getText());
    try {
        service.setPrice(Double.parseDouble(txtPrice.getText()));
        serviceRepo.save(service);
        reset();
        loadTable();
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(null, "Harga harus berupa angka.", "Input Error", JOptionPane.ERROR_MESSAGE);
    }
});
btnSave.setBounds(150, 159, 75, 32);
contentPane.add(btnSave);

JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(e -> {
    if (selectedId != null) {
        Service service = new Service();
        service.setServiceName(txtServiceName.getText());
        service.setDescription(txtDescription.getText());
        try {
            service.setPrice(Double.parseDouble(txtPrice.getText()));
            service.setId(selectedId);
            serviceRepo.update(service);
            reset();
            loadTable();
            loadTable();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Harga harus berupa angka.", "Input Error", JOptionPane.ERROR_MESSAGE);
        }
    } else {
        JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di-update");
    }
});
btnUpdate.setBounds(231, 159, 92, 32);
contentPane.add(btnUpdate);

JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(e -> {
    if (selectedId != null) {
        int confirmation = JOptionPane.showConfirmDialog(null, "Anda yakin ingin menghapus data ini?", "Konfirmasi Hapus", JOption
        if (confirmation == JOptionPane.YES_OPTION) {
            serviceRepo.delete(selectedId);
            reset();
            loadTable();
        }
    } else {
        JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
    }
});
btnDelete.setBounds(330, 159, 92, 32);
contentPane.add(btnDelete);

JButton btnCancel = new JButton("Cancel");
btnCancel.addActionListener(e -> reset());
btnCancel.setBounds(429, 159, 92, 32);
```

```
158         contentPane.add(btnCancel);
159
160         JScrollPane scrollPane = new JScrollPane();
161         scrollPane.setBounds(21, 222, 500, 316);
162         contentPane.add(scrollPane);
163
164         tableServices = new JTable();
165         tableServices.addMouseListener(new MouseAdapter() {
166             @Override
167             public void mouseClicked(MouseEvent e) {
168                 int row = tableServices.getSelectedRow();
169                 if (row >= 0) {
170                     selectedId = tableServices.getValueAt(row, 0).toString();
171                     txtServiceName.setText(tableServices.getValueAt(row, 1).toString());
172                     txtDescription.setText(tableServices.getValueAt(row, 2).toString());
173                     txtPrice.setText(tableServices.getValueAt(row, 3).toString());
174                 }
175             }
176         });
177         scrollPane.setViewportView(tableServices);
178
179         loadTable();
180     }
181 }
```

Maka outputnya akan seperti gambar dibawah ini:



b) Pelanggan

1) Buat class baru dengan nama **PelangganRepo,** pada package **DAO.**

```java
package DAO;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import model.Pelanggan;

public class PelangganRepo {

    private static List<Pelanggan> db = new ArrayList<>();
    private static boolean isDataLoaded = false;

    public PelangganRepo() {
        if (!isDataLoaded) {
            Pelanggan p1 = new Pelanggan();
            p1.setId(UUID.randomUUID().toString());
            p1.setNama("Budi Santoso");
            p1.setEmail("budi.s@mail.com");
            p1.setTelepon("081234567890");
            db.add(p1);

            Pelanggan p2 = new Pelanggan();
            p2.setId(UUID.randomUUID().toString());
            p2.setNama("Citra Lestari");
            p2.setEmail("citra.l@mail.com");
            p2.setTelepon("082345678901");
            db.add(p2);

            isDataLoaded = true;
        }
    }
    public void save(Pelanggan pelanggan) {
        pelanggan.setId(UUID.randomUUID().toString());
        db.add(pelanggan);
        System.out.println("Data pelanggan berhasil disimpan!");
    }

    public List<Pelanggan> show() {
        return db;
    }

    public void update(Pelanggan pelanggan) {
        for (int i = 0; i < db.size(); i++) {
            if (db.get(i).getId().equals(pelanggan.getId())) {
                db.set(i, pelanggan);
                System.out.println("Data pelanggan berhasil diupdate!");
                return;
            }
        }
    }

    public void delete(String id) {
        db.removeIf(p -> p.getId().equals(id));
        System.out.println("Data pelanggan berhasil dihapus!");
    }
}
```

- **PelangganRepo():** Mengisi "database" dengan dua data
- **save(Pelanggan pelanggan):** Menambahkan objek pelanggan baru ke dalam List db.
- **show():** Mengembalikan seluruh isi List db, menampilkan semua data pelanggan yang tersimpan.
- **update(Pelanggan pelanggan):** Mencari pelanggan di dalam List yang memiliki id yang sama dengan id dari objek pelanggan yang diberikan.
- **delete(String id):** Menghapus pelanggan dari List berdasarkan id yang diberikan.

2) Buat class baru dengan nama **Pelanggan,** pada package **model.**

```java
package model;

public class Pelanggan {

    private String id;
    private String nama;
    private String email;
    private String telepon;

    // Getters and Setters
    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 getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getTelepon() {
        return telepon;
    }

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

3) Buat class baru dengan nama **TablePelanggan,** pada package **Table.**

```java
package Table;

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

public class TablePelanggan extends AbstractTableModel {

    private List<Pelanggan> list;

    public TablePelanggan(List<Pelanggan> list) {
        this.list = list;
    }

    @Override
    public int getRowCount() {
        return list.size();
    }

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

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
            case 0: return list.get(rowIndex).getId();
            case 1: return list.get(rowIndex).getNama();
            case 2: return list.get(rowIndex).getEmail();
            case 3: return list.get(rowIndex).getTelepon();
            default: return null;
        }
    }

    @Override
    public String getColumnName(int column) {
        switch (column) {
            case 0: return "ID";
            case 1: return "Nama Pelanggan";
            case 2: return "Email";
            case 3: return "No. Telepon";
            default: return null;
        }
    }
}
```

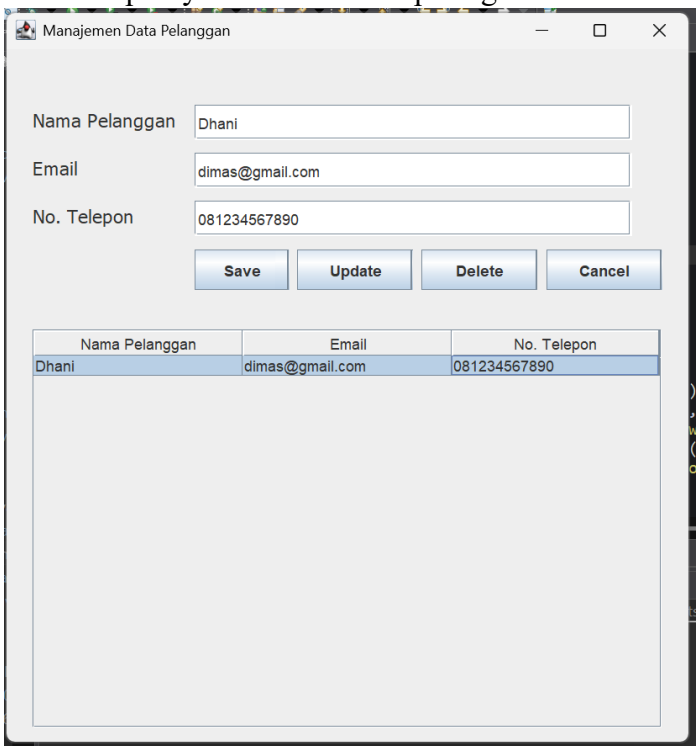
4) Buat JFrame baru dengan nama **PelangganFrame**, pada package **ui**

```java
package ui;

import java.awt.EventQueue;

public class PelangganFrame extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField txtNama;
    private JTextField txtEmail;
    private JTextField txtTelepon;
    private JTable tablePelanggan;
    private PelangganRepo pelangganRepo = new PelangganRepo();
    private List<Pelanggan> ls;
    private String selectedId;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                PelangganFrame frame = new PelangganFrame();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        });
    }

    public void reset() {
        txtNama.setText("");
        txtEmail.setText("");
        txtTelepon.setText("");
        selectedId = null;
    }

    public void loadTable() {
        ls = pelangganRepo.show();
        TablePelanggan tableModel = new TablePelanggan(ls);
        tablePelanggan.setModel(tableModel);
        tablePelanggan.getColumnModel().getColumn(0).setMinWidth(0);
        tablePelanggan.getColumnModel().getColumn(0).setMaxWidth(0);
        tablePelanggan.getColumnModel().getColumn(0).setWidth(0);
    }

    public PelangganFrame() {
        setTitle("Manajemen Data Pelanggan");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 556, 586);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNama = new JLabel("Nama Pelanggan");
        lblNama.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblNama.setBounds(21, 40, 120, 32);
        contentPane.add(lblNama);

        JLabel lblEmail = new JLabel("Email");
        lblEmail.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblEmail.setBounds(21, 78, 92, 32);
```

```
        contentPane.add(lblEmail);

        JLabel lblTelepon = new JLabel("No. Telepon");
        lblTelepon.setFont(new Font("Tahoma", Font.PLAIN, 15));
        lblTelepon.setBounds(21, 116, 100, 32);
        contentPane.add(lblTelepon);

        txtNama = new JTextField();
        txtNama.setBounds(150, 44, 347, 28);
        contentPane.add(txtNama);
        txtNama.setColumns(10);

        txtEmail = new JTextField();
        txtEmail.setColumns(10);
        txtEmail.setBounds(150, 82, 347, 28);
        contentPane.add(txtEmail);

        txtTelepon = new JTextField();
        txtTelepon.setColumns(10);
        txtTelepon.setBounds(150, 120, 347, 28);
        contentPane.add(txtTelepon);

        JButton btnSave = new JButton("Save");
        btnSave.addActionListener(e -> {
            Pelanggan p = new Pelanggan();
            p.setNama(txtNama.getText());
            p.setEmail(txtEmail.getText());
            p.setTelepon(txtTelepon.getText());
            pelangganRepo.save(p);
            reset();
            loadTable();
            JOptionPane.showMessageDialog(null, "Data berhasil disimpan!");
        });
        btnSave.setBounds(150, 159, 75, 32);
        contentPane.add(btnSave);

        JButton btnUpdate = new JButton("Update");
        btnUpdate.addActionListener(e -> {
            if (selectedId != null) {
                Pelanggan p = new Pelanggan();
                p.setId(selectedId);
                p.setNama(txtNama.getText());
                p.setEmail(txtEmail.getText());
                p.setTelepon(txtTelepon.getText());
                pelangganRepo.update(p);
                reset();
                loadTable();
                JOptionPane.showMessageDialog(null, "Data berhasil diupdate!");
            } else {
                JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di-update");
            }
        });
        btnUpdate.setBounds(231, 159, 92, 32);
        contentPane.add(btnUpdate);

        JButton btnDelete = new JButton("Delete");
        btnDelete.addActionListener(e -> {
            if (selectedId != null) {
                int confirmation = JOptionPane.showConfirmDialog(null, "Anda yakin ingin menghapus data ini?", "Konfirmasi Hapus", JOption
                if (confirmation == JOptionPane.YES_OPTION) {
                    pelangganRepo.delete(selectedId);
                    reset();
                    loadTable();
                    JOptionPane.showMessageDialog(null, "Data berhasil dihapus!");
                }
            } else {
                JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
            }
        });
        btnDelete.setBounds(330, 159, 92, 32);
        contentPane.add(btnDelete);

        JButton btnCancel = new JButton("Cancel");
        btnCancel.addActionListener(e -> reset());
        btnCancel.setBounds(429, 159, 92, 32);
        contentPane.add(btnCancel);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(21, 222, 500, 316);
        contentPane.add(scrollPane);

        tablePelanggan = new JTable();
        tablePelanggan.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                int row = tablePelanggan.getSelectedRow();
                if (row >= 0) {
                    selectedId = tablePelanggan.getValueAt(row, 0).toString();
                    txtNama.setText(tablePelanggan.getValueAt(row, 1).toString());
                    txtEmail.setText(tablePelanggan.getValueAt(row, 2).toString());
```

```
                    Object teleponObj = tablePelanggan.getValueAt(row, 3);
                    txtTelepon.setText(teleponObj != null ? teleponObj.toString() : "");
                }
            }
        });
        scrollPane.setViewportView(tablePelanggan);

        loadTable();
    }
}
```

Maka outputnya akan terlihat seperti gambar dibawah ini:



D. Kesimpulan

Kesimpulannya Adalah Kode ini menunjukkan cara membuat aplikasi Java yang rapi dengan membagi tugas secara jelas. Ada bagian khusus untuk mengatur data (Model), bagian untuk urusan database (DAO), dan bagian untuk tampilan (TableModel). Kode ini juga mencontohkan dua cara penyimpanan: menggunakan database asli yang permanen dan memakai daftar sementara di memori untuk latihan. Cara ini membuat programnya jadi gampang diurus, dites, dan dikembangkan.