Laporan Praktikum Pekan 3
Pemrograman Berorientasi Objek

Disusun Oleh:

Nama: Fadil Insanus Siddik

NIM: 2411532013

Dosen Pengampu:

Nurfiah Bin Sutarwo

Program Studi Informatika

Fakultas Teknologi Informatika

Universitas Andalas

Padang, 2025

1. Menghapus Fungsi CRUD DAO pada GUI

   - Method digunakan utuk menghapus value inputan ketika suatu proses berhasil dilakukan, buat method **reset** pada JFrame seperti kodeprogram di bawah ini.

```
54  public void reset() {
55      txtName.setText("");
56      txtUsername.setText("");
57      txtPassword.setText("");
58  }
59
```

   - Membuat instance pada UserFrame

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

2. CREATE USER

   - Klik kanan pada tombol **save → add event handlers → actionPerformed** kemudian isi dengan kode program berikut.

```
User user = new User();
user.setNama(nama);
user.setUsername(usernameFormatted);
user.setPassword(password);
usr.save(user);
reset();
```

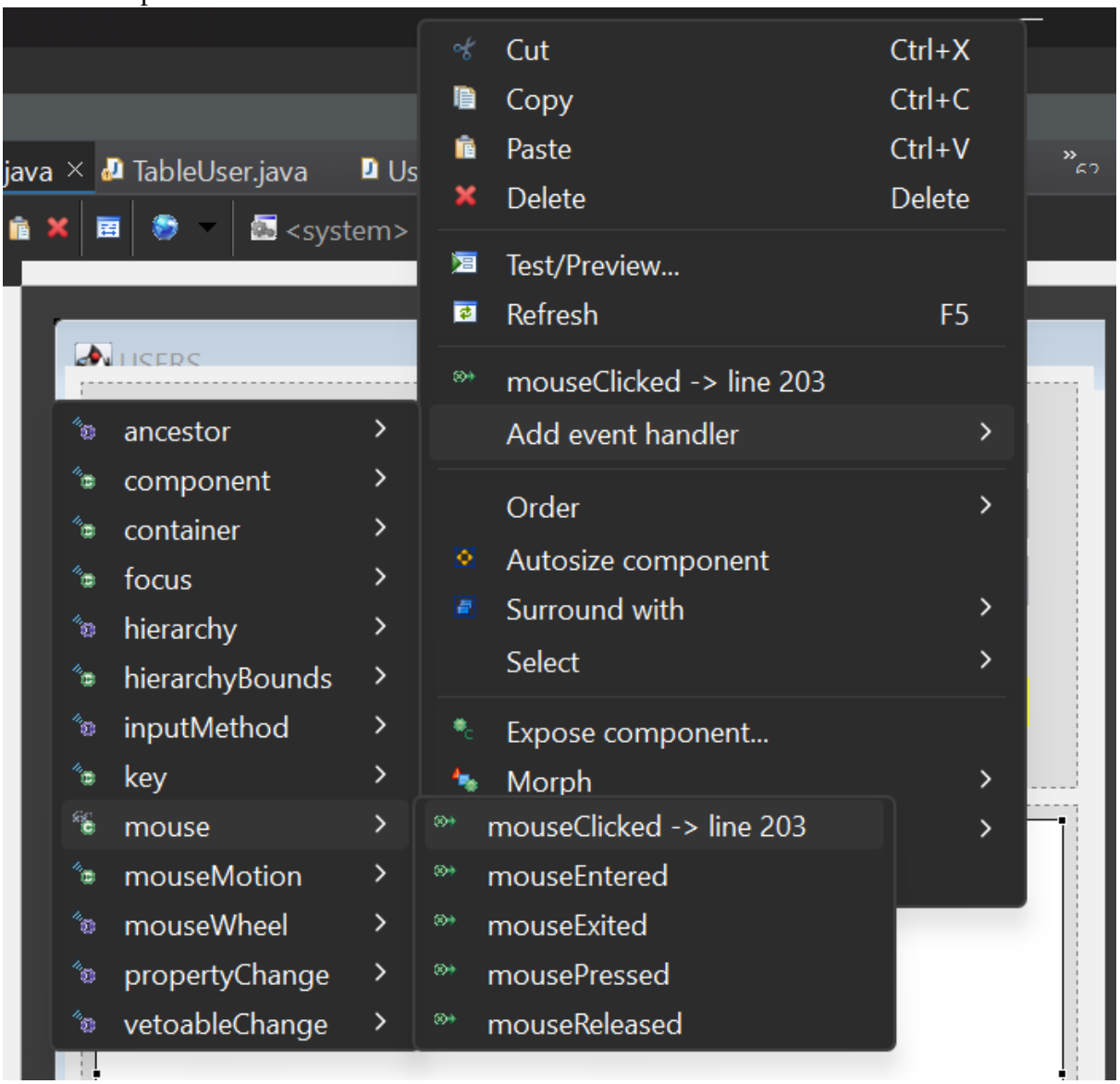
3. READ USERS

   - Buat method dengan nama **loadTable()** kemudian sisikan dengan kode program berikut.

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

4. UPDATE USER

- Klik kanan pada **JTable → add event handler → mouse → mouseClicked**

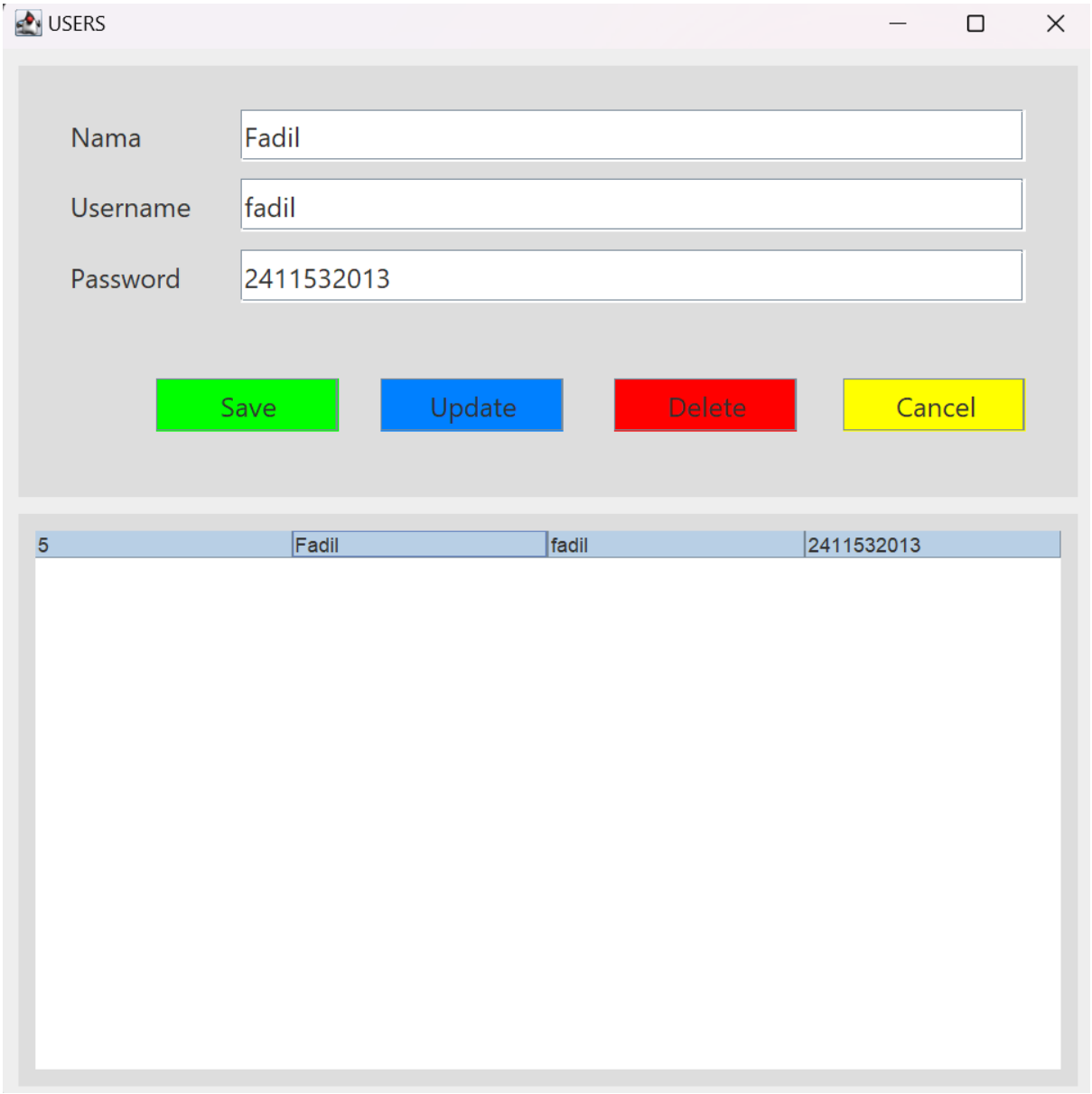


Lalu, isikan dengan kode program di bawah ini.

```
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());
```

Kode program di atas berfungsi untuk mengambil id user dan menyimpannya ke dalam variable **id** kemudian mengambil data nama, username dan password dan ditampilkan ke dalam form inputan.

- Klik salah satu isi table maka akan secara otomatis tampil pada form inputan.



- Klik kanan tombol **update → add event handler → action → actionPerformed** dan isikan dengan kode program berikut.

```
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setNama(txtName.getText());

        String usernameFormatted = txtUsername.getText().toLowerCase().replaceAll("\\s+", "");
        user.setUsername(usernameFormatted);

        user.setPassword(txtPassword.getText());
        user.setId(id);
        usr.update(user);
        reset();
        loadTable();
    }
});
```

5. DELETE USER

- Klik salah satu data pada JTable
- Klik kanan tombol **delete → add event handler → action → actionPerformed** dan isikan dengan kode program berikut.

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

LATIHAN

Soal: Membuat fungsi CRUD untuk layanan dan pelanggan

1. Buat class baru pada package DAO dengan nama Pelangganrepo
   Lalu isi dengan kode program seperti di bawah ini.

```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("Fadil Asril");
            p1.setEmail("fadila@gmail.com");
            p1.setTelepon("081234567890");
            db.add(p1);

            Pelanggan p2 = new Pelanggan();
            p2.setId(UUID.randomUUID().toString());
            p2.setNama("Anggun");
            p2.setEmail("anggun123@gmail.com");
            p2.setTelepon("089876543210");
            db.add(p2);

            isDataLoaded = true;
        }
    }
```

```java
    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!");
    }
}
```

Kode program tersebut memiliki beberapa fungsi, diantaranya:

- **db →** tempat penyimpanan data pelanggan (pakai Arraylist, mirip database sementara).
- **isDataLoaded →** supaya data contoh (dummy) hanya dimuat sekali.
- **Constructor →** otomatis menambahkan 2 data pelanggan contoh saat pertama kali dibuat.
- **save() →** menyimpandata pelanggan bar uke **db** dengan ID unik (UUID).
- **show →** menampilkan seluruh data pelanggan dari **db.**
- **update() →** mengganti data pelanggan lama dengan data baru berdasarkan ID.
- **Delete() →** menghapus data pelanggan dari **db** berdasarkan ID.

2. Buat class baru pada package DAO dengan nama Servicerepo
   Lalu isikan kode program seperti di bawah ini.

```
package DAO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import model.Service;
import confg.Database;

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();
```

Berikut penjelasan dari class dengan nama Servicerepo:

- Tujuan utama: Mengelola data services di database menggunakan JDBC (CRUD).
- **save(Service service)** → Menyimpan data layanan baru ke table **services.**
- **show()** → Mengambil semua data dari table **services** dan mengubahnya menjadi list **Service.**
- **update(Service service)** → Mengubah data layanan **id** (nama, deskripsi, harga).
- **delete(String id)** → Menghapusdata layanan dari table **services** berdasarkan **id.**
- **Database.koneksi()** → Digunakan untuk membuat koneksi ke database.
- **Try-with-resources** → Menjamin **Connection, PreparedStatement,** dan **ResultSet** otomatis ditutup setelah digunakan.

3. Buat class baru dengan nama Pelanggan pada package model
   Lalu isikan kode program seperti di bawah ini.

```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;
    }
```

Berikut penjelasan mengenai kode program di atas:

- Class **Pelanggan** → merepresentasikan data pelanggan (model).
- Memiliki 4 atribut (field):
  - **Id** → identitas unik pelanggan.
  - **nama** → nama pelanggan.
  - **email** → alamat email pelanggan.
  - **telepon** → nomor telepon pelanggan.

- **Getter dan Setter** disediakan untuk tiap atribut → digunakan untuk membaca (get) dan mengubah (set) nilai field secara aman.
- Termasuk dalam konsep **POJO (Plain Old Java Object)** → hanya menyimpan data, tanpa logika bisnis.

4. Buat class baru dengan nama TablePelanggan pada package table
   Lalu isi kode program seperti di bawah ini.

```
package table;

import java.util.List;

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;
        }
    }
```

Berikut penjelasan mengenai kode program di atas:

- Class **TablePelanggan** → turunan dari **AbstractTableModel**, digunakan untuk menampilkan data pelanggan di **JTable** (Swing).
- **list** → menampung data pelanggan dalam bentuk **List<Pelanggan>.**
- **Constructor** → menerima **List<pelanggan**> dan menyimpannya ke variabel **list.**
- **getRowCount()** → mengembalikan jumlah baris table (sesuai jumlah data pelanggan).
- **getColumnCount()** → mengembalikan jumlah kolom table (4 kolom: ID, Nama, Email, Telepon).
- **getValueAt(int rowIndex, int columnIndex)** → menentukan isi sel table berdasarkan baris dan kolom.
- **getColumnName(int column)** → memberi nmaa kolom table (“ID”, “Nama Pelanggan”, “Email”, “No. Telepon”).

5. Buat kelas baru degan nama TablesService dan di extends dengan AbstractTableModel pada package table
   Lalu isikan kode program seperti berikut.

```java
package table;

import java.util.List;

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;
        }
    }
}
```

Berikut penjelasan mengenai kode program di atas:

- Class **TableService** → turunan **AbstractTableModel**, dipakai untuk menampilkan data layanan (**Service**) di **JTable.**
- **service** → menyimpan daftar data layanan dalam bentuk **List<Service>.**
- **columnNames** → array yang berisi nama-nama kolom table (**ID, Service, Name, Description, Price).**
- **Constructor** → menerima list layanan (**services)** dan menyimpanan ke variabel internal.
- **getRowCount()** → mengembalikan jumlah baris (sesuai banyaknya data layanan).

- **getColumnCount()** → mengembalikan jumlah kolom (4, sesuai Panjang **columnNames**).
- **getColumnName(int column)** → memberikan nama kolom berdasarkan **columnNames**.
- **getValueAt(int rowIndex, int columnIndex)** → mengambil nilai data yang akan ditampilkan di sel table sesuai baris dan kolom (ID, nama layanan, deskripsi, harga).

6. Buat kelas baru pada package ui dengan nama Pelangganframe
   Lalu isikan kode program seperti berikut

```java
package ui;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import DAO.Pelangganrepo;
import table.TablePelanggan;
import model.Pelanggan;
import javax.swing.JLabel;
import java.awt.Font;
import java.util.List;
import javax.swing.JTextField;
import javax.swing.JScrollPane;
import javax.swing.JButton;
import javax.swing.JTable;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JOptionPane;

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 {
```

```
            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");
```

```
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.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
            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);
```

Berikut penjelasan mengenai kode program di atas:

- Class **Pelangganframe →** turunan **JFrame**, membuat GUI untuk manajemen data pelanggan.
- Komponen utama GUI:
  - **JTextField →** input untuk Nama, Email, dan Telepon.
  - **JButton →** aksi CRUD (Save, Update, Delete, Cancel).
  - **JTable →** menampilkan daftar pelanggan.
  - **JScrollPane →** membungkus table agar bisa di-scroll.
- **Pelangganrepo →** dipakai untuk operasi CRUD pada data pelanggan.
- **LoadTable() →** memuat data pelanggan ke dalam table menggunakan **TablePelanggan.**
- **Reset() →** mengosongkan input form dan reset **selectedId.**
- Fungsi tombol:
  - **Save →** menambah pelanggan baru.
  - **Update →** mengubah data pelanggan terpilih.
  - **Delete →** menghapus data pelanggan terpilih (dengan konfirmasi).
  - **Cancel →** membersihkan input form.
- **MouseListener** pada JTable → saat baris di klik, data ditampilkan di text field agar bisa diupdate/hapus.
- **main() →** menjalankan aplikasi GUI dengan membuka **Pelangganframe.**

7. Buat class baru pada package ui dengan nama Serviceframe
   Lalu isi kode program seperti di bawah ini

```java
package ui;

import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import DAO.Servicerepo;
import table.TableService;
import model.Service;
import javax.swing.JLabel;
import java.awt.Font;
import java.util.List;
import javax.swing.JTextField;
import javax.swing.JScrollPane;
import javax.swing.JButton;
import java.awt.Color;
import javax.swing.JTable;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JOptionPane;

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);
```

```
    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
    }
});
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();
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Harga harus berupa angka.", "Input E
        }
    } 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
        if (confirmation == JOptionPane.YES_OPTION) {
            serviceRepo.delete(selectedId);
            reset();
            loadTable();
        }
    } else {
        JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapu
    }
});
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);

tableServices = new JTable();
tableServices.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int row = tableServices.getSelectedRow();
        if (row >= 0) {
```

```
    public void mouseClicked(MouseEvent e) {
        int row = tableServices.getSelectedRow();
        if (row >= 0) {
            selectedId = tableServices.getValueAt(row, 0).toString();
            txtServiceName.setText(tableServices.getValueAt(row, 1).toString());
            txtDescription.setText(tableServices.getValueAt(row, 2).toString());
            txtPrice.setText(tableServices.getValueAt(row, 3).toString());
        }
    }
});
scrollPane.setViewportView(tableServices);

loadTable();
```

Berikut penjelasan mengenai kode program di atas:

- Class **Serviceframe** → turunan **JFrame**, GUI untuk manajemen data service (layanan).
- Komponen utama GUI:
  - **JTextField** → inpt untuk Service Name, Description, dan Price.
  - **JButton** → aksi CRUD (Save, Update, Delete, Cancel).
  - **JTable** → menampilkan daftar layanan, dibungkus **JScrollPane**.
- **Servicerepo** → digunakan untuk operasi CRUD ke database tabel **services.**

- **loadTable() →** memuat data layanan ke **JTable** menggunakan **TableService**.
- **reset() →** mengosongkan input dan mereset **selectedId**.
- Fungsi tombol:
  - **Save →** menambah layanan baru (validasi harga harus angka).
  - **Update →** mengubah data layanan terpilih.
  - **Delete →** menghaspu layanan terpilih (dengan konfirmasi).
  - **Cancel →** mengosongkan form input.
- **MouseListener** pada JTable → saat baris dipilih, data layanan ditampilkan di text field untuk di edit/ hapus.
- **main() →** menjalankan aplikasi dengan membuka fram **Serviceframe.**