LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

OLEH :

AHMAD MUHAIMIN KAMIL

2311533013

DOSEN PENGAMPU : Nurfiah, S.ST., M.Kom.

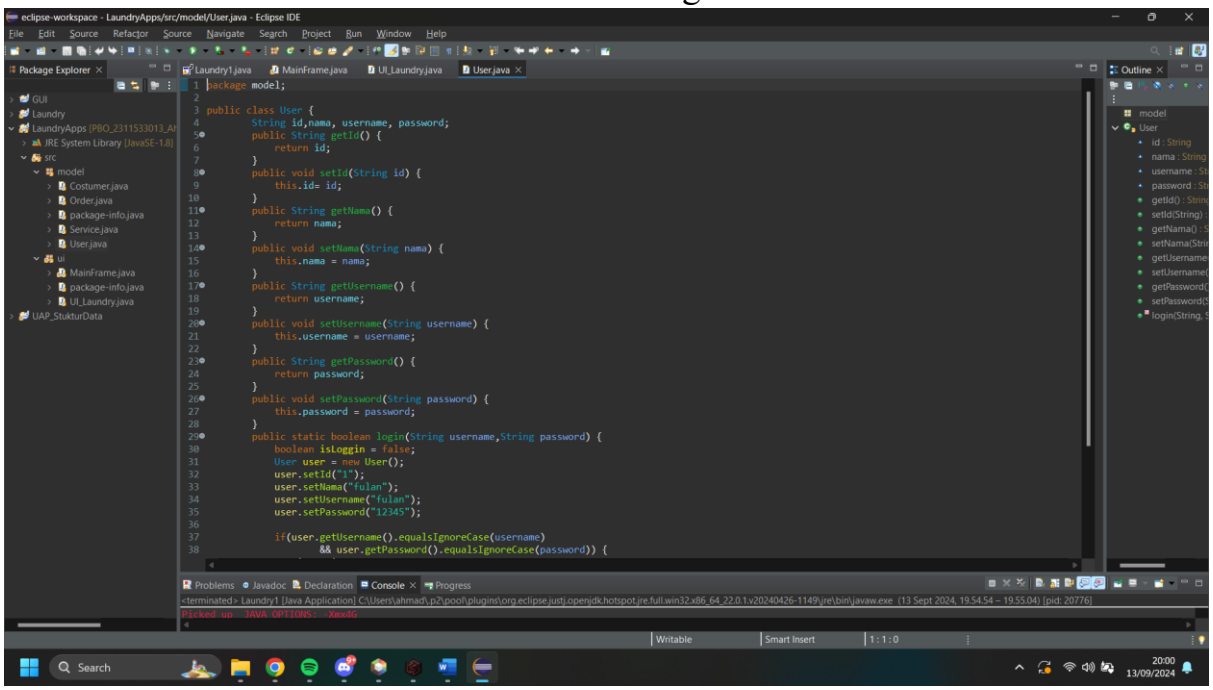
PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

Aplikasi Laundry

1.Membuat attribute class user dan membuat getter & setter.\

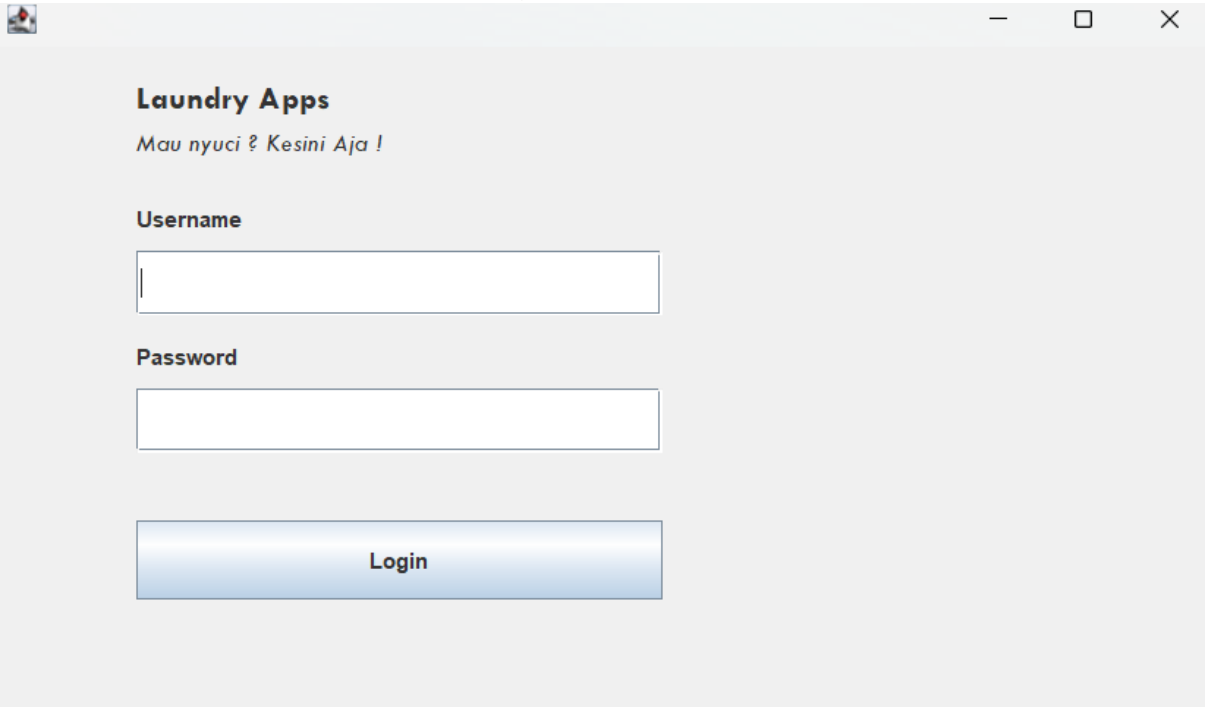


Terdiri dari id, nama, username, password.

2. Method login yang nantinya akan digunakan Ketika pengguna akan login ke Aplikasi

```
    public static boolean login(String username,String password) {
        boolean isLoggin = false;
        User user = new User();
        user.setId("1");
        user.setNama("fulan");
        user.setUsername("fulan");
        user.setPassword("12345");

        if(user.getUsername().equalsIgnoreCase(username)
                && user.getPassword().equalsIgnoreCase(password)) {
            isLoggin = true;
        }else {
            isLoggin = false;
        }
        return isLoggin;
    }

}
```

3.Membuat JFrame baru untuk Login



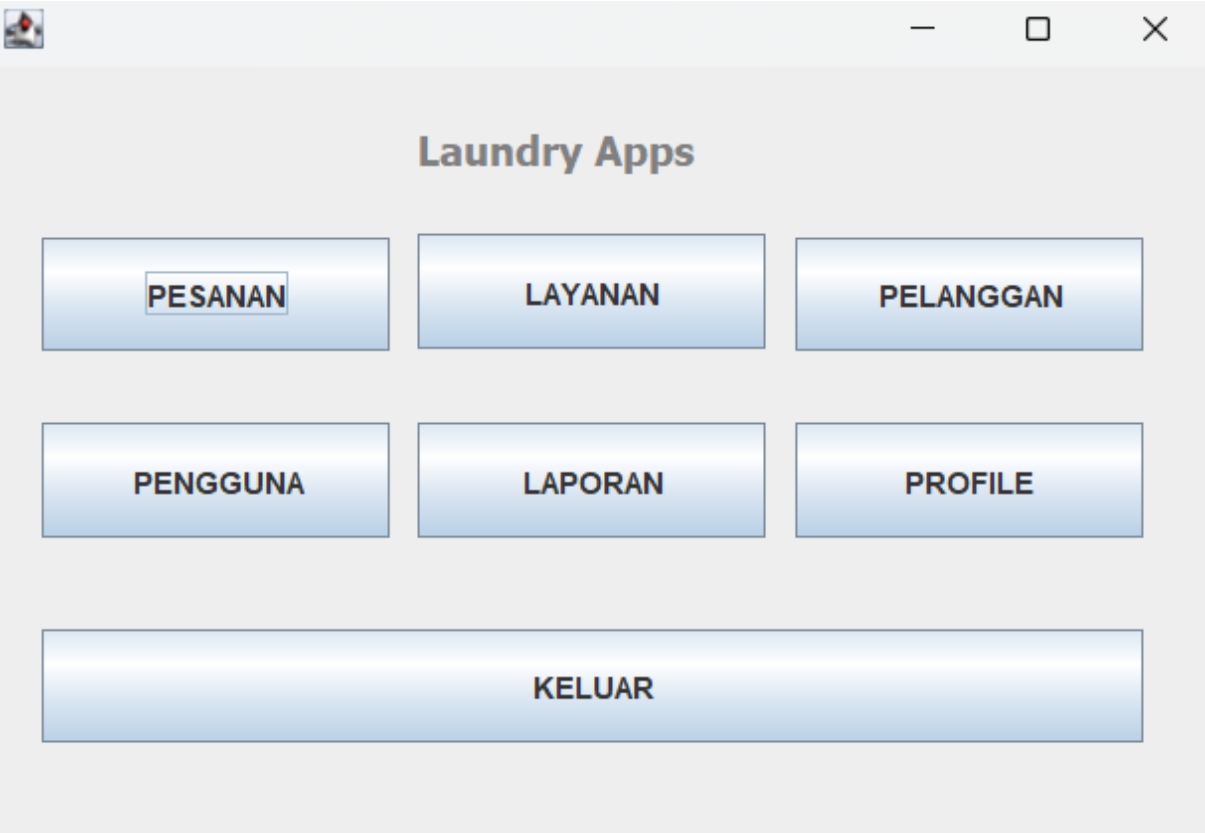
4.Memanggil method login pada class User

```
JButton btnLogin = new JButton("Login");
btnLogin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(User.login(txtUsername.getText(), txtPassword.getText())) {
            new MainFrame().setVisible(true);
            dispose();
        }else {
            JOptionPane.showMessageDialog(null, "Login Gagal");
        }
    }
});
btnLogin.setBounds(78, 259, 287, 43);
contentPane.add(btnLogin);
```

Kode di atas merupakan event listener yang dipasang pada tombol btnLogin. Event listener ini berfungsi untuk mendeteksi ketika tombol diklik dan menjalankan aksi yang ditentukan dalam metode actionPerformed. Saat tombol btnLogin diklik, metode actionPerformed akan dijalankan. Di dalam metode ini, pertama-tama dilakukan pengecekan terhadap hasil dari metode User.login(txtUsername.getText(), txtPassword.getText()). Metode User.login() ini mungkin memeriksa apakah kombinasi username dan password yang dimasukkan oleh pengguna valid.

Jika login berhasil (nilai true dikembalikan), objek MainFrame akan dibuat dan ditampilkan dengan menggunakan new MainFrame().setVisible(true). Pada saat yang bersamaan, frame saat ini akan ditutup menggunakan dispose(). Namun, jika login gagal (nilai false dikembalikan), pesan kesalahan akan ditampilkan kepada pengguna dengan menggunakan JOptionPane.showMessageDialog(null, "Login Gagal").

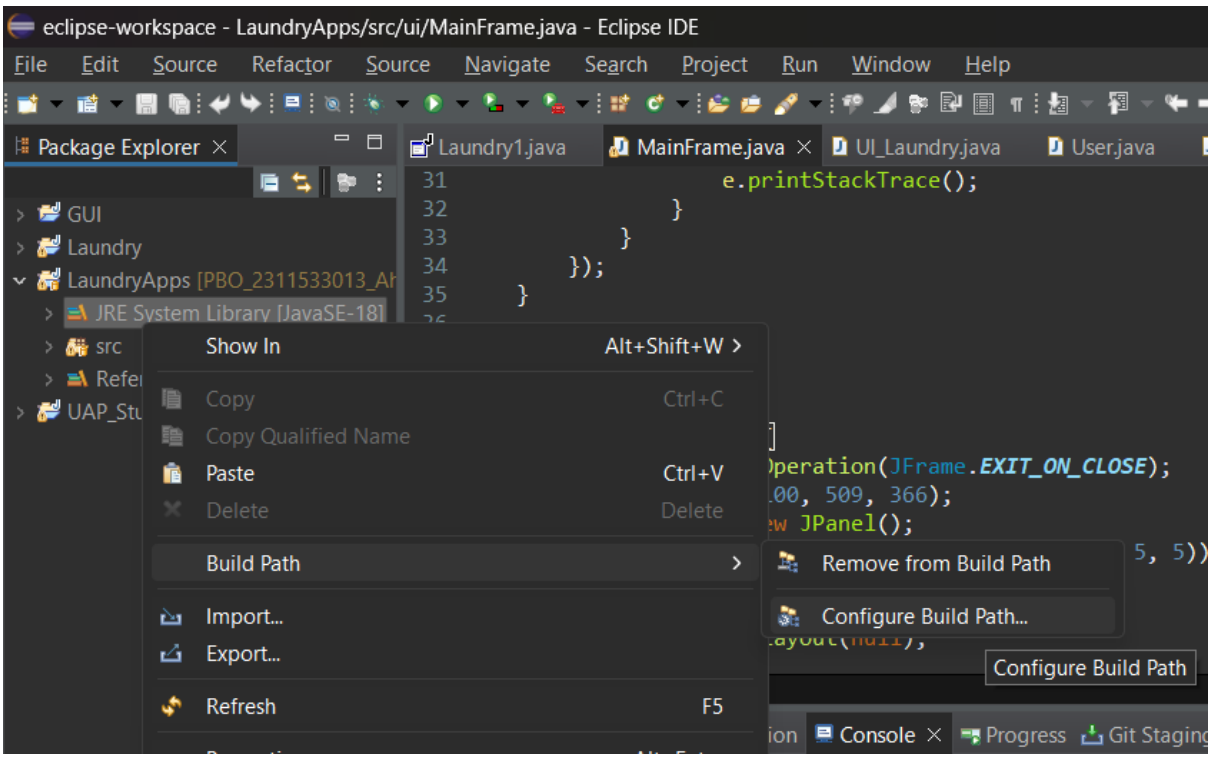
5.Mendesain tampilan Halaman Utama



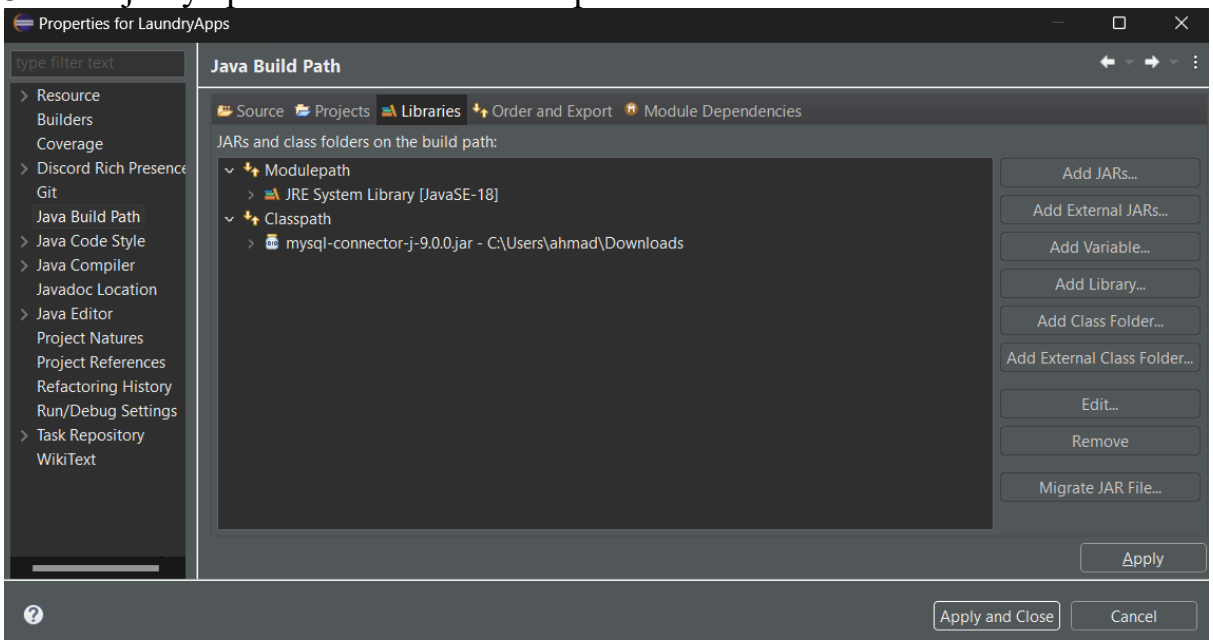
MEMBUAT FUNGSI CRUD USER DENGAN DATABASE MYSQL

1.Mendownload MySQL Connector pada link berikut ini :
https://dev.mysql.com/downloads/connector/j/

2. Menambahkan MySQL Connector kedalam project dengan cara klik kanan directory JRE System Library → Built Path → Configure Build Path

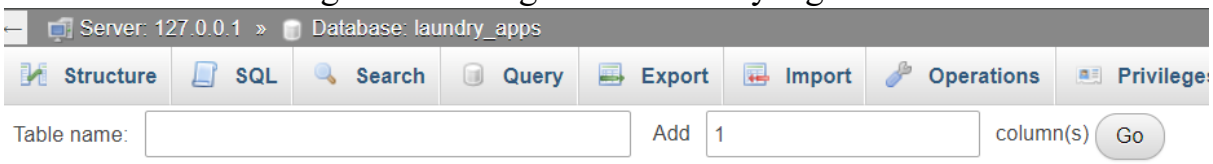


3. Selanjutnya pilih Libraries → Classpath

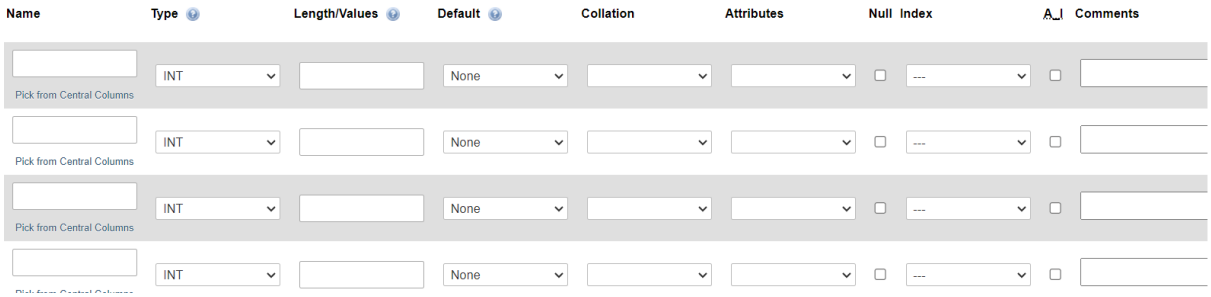


4.Membuat Database dan Table User
Run XAMPP , dan klik Start pada Apache dan MySQL
5.Ketik link berikut pada browser yang digunakan
localhost/phpMyAdmin
6.Klik new dan buat database dengan nama laundry_apps

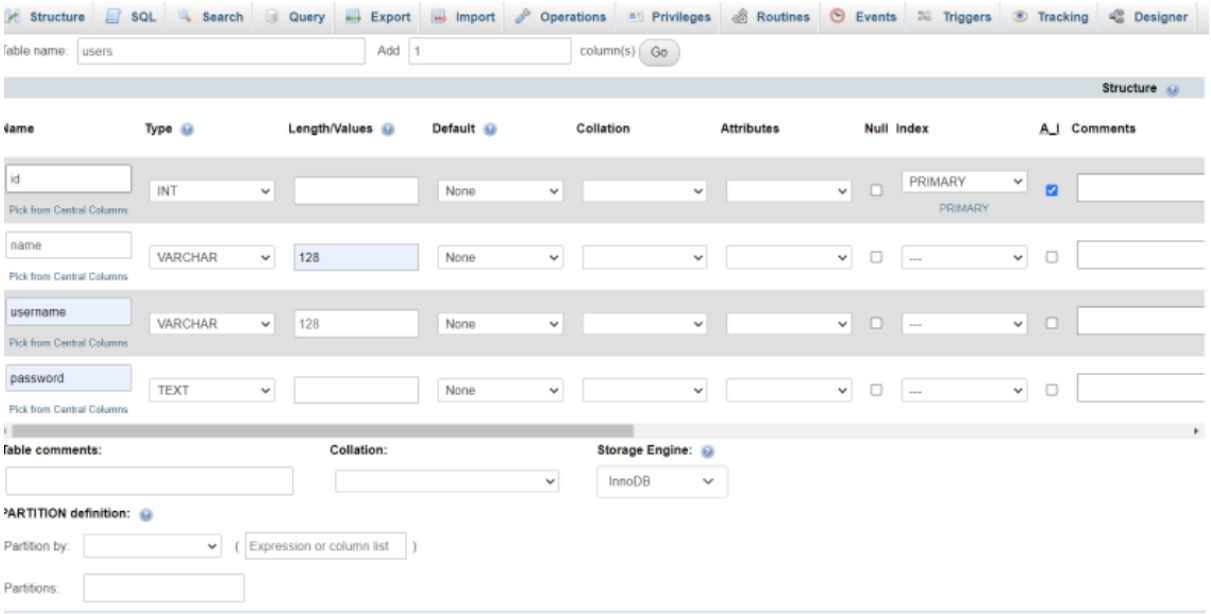
7.Buat table user dengan cara meng klik database yang sudah di buat tadi



8.Klik Create dan akan muncul seperti ini :



Dan isi seperti berikut ini :

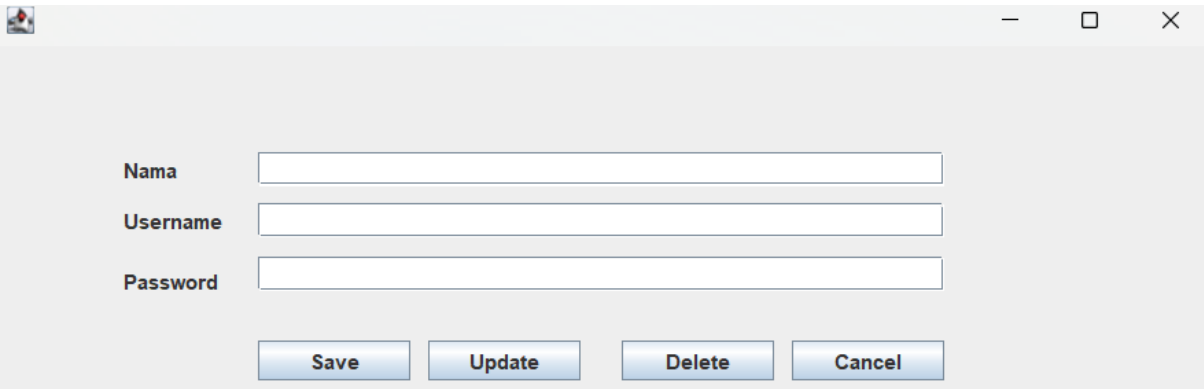


9.Pada Eclipse buat Package baru dengan nama config

10.Buat Class baru dengan nama Database dan ketikkan kode seperti gambar dibawah ini:

```
package config;

import java.sql.*;

public class Database {
    Connection conn;
    public static Connection koneksi() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps", "root", "");
            return conn;

        }catch(Exception e) {
            JOptionPane.showMessageDialog(null, e);
            return null;
        }
    }

}
```

11.Buat JFrame baru dengan nama UserFrame dan desain tampilannya



12.Membuat Table Model , buat package baru dengan nama table dan buat class baru didalamnya dengan nama TableUser dan ketikkan kode seperti gambar dibawah :

```java
package table;

import javax.swing.table.AbstractTableModel;

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) {
        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;
        }
    }

}
```

13.Membuat fungsi DAO, buat package baru dengan nama DAO , dan buat class UserDAO didalamnya , dan ketikkan kode seperti dibawah ini

```java
package DAO;

import java.util.List;

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

14.Buat class baru pada package DAO bernama UserRepo , dan ketikkan kode seperti dibawah , Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database.

```java
package DAO;

import java.util.logging.Logger;

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();
    }
```

15.Membuat method save , ketikkan kode seperti gambar dibawah ini

```java
    @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();
                }
            }
    }
```

16.Membuat method show , ketikkan kode seperti gambar dibawah ini :

```java
    @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;
}
```

17.Membuat method update, dan ketikkan kode seperti gambar dibawah ini :

```java
    @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();
        }
    }
}
```

18.Membuat method delete, dan ketikkan kode seperti dibawah ini:

```java
    @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();
            }
        }
    }

}
```

19.Menggunakan fungsi CRUD DAO pada GUI

Method digunakan untuk menghapus value inputan Ketika suatu proses berhasil dilakukan, buat method reset pada JFrame seperti kode program dibawah ini :

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

20. Membuat instance pada UserFrame

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

21. Untuk membuat tombol save berfungsi, Klik kanan pada tombol save → add event handlers → actionPerformed kemdian isi dengan kode program berikut :

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

22. Buat method dengan nama loadTable() kemudian isikan dengna kode program berikut:

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

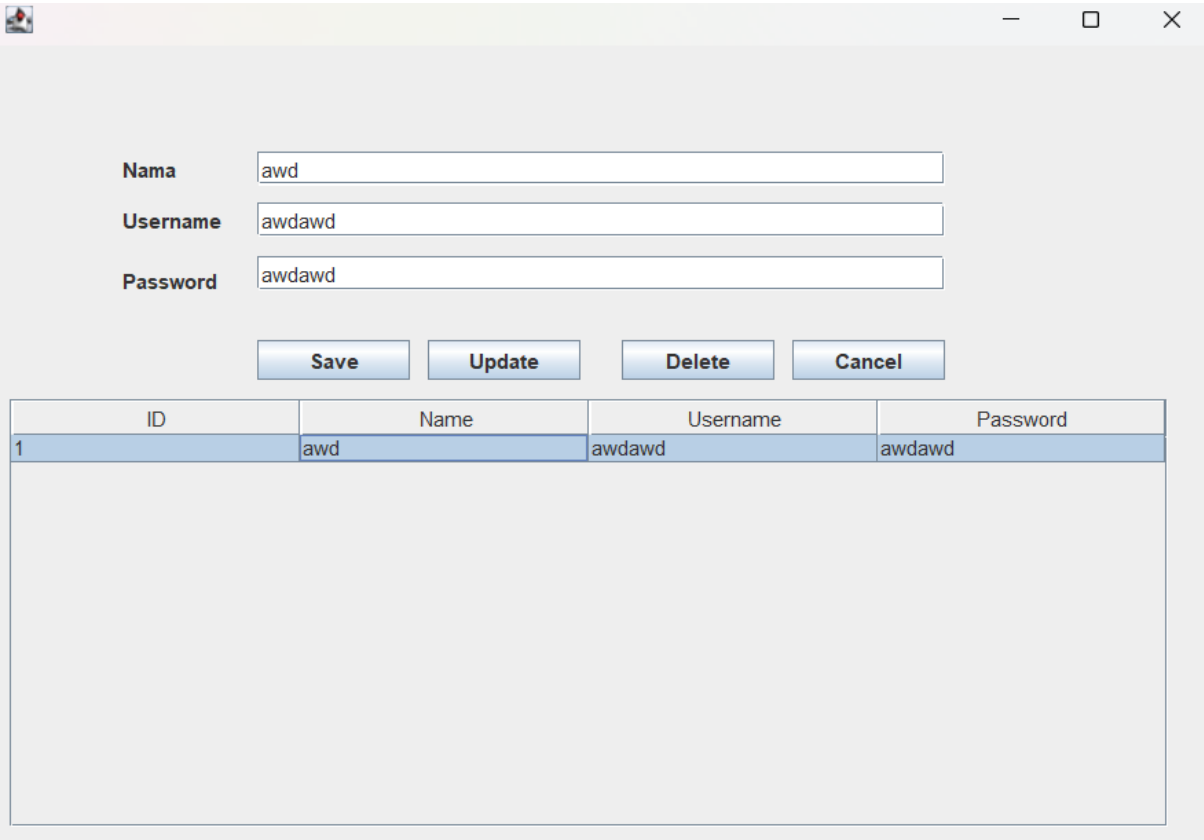
23. Memanggil method pada class main, sehingga Ketika pertama kali program dijalankan maka loadTable akan dipanggil.

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

24.Membuat UpdateUser , Klik kanan pada JTable → add event handler → mouse → mouseClicked

```
@Override
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());
}
```

25.Jika code benar, hasilnya akan seperti ini :



26.Untuk membuat button Update berfungsi, Klik kanan tombol update → add event handler → action → actionPeformed dan isikan dengan kode program berikut:

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

27.Membuat Delete User, Klik salah satu data pada table , Klik kanan tombol delete → add event handler → action → actionPerformed dan isikan dengan kode program berikut.

```
    if(id != null) {
        usr.delete(id);
        reset();
        loadTable();
    }else {
        JOptionPane.showMessageDialog(null, "Silahkan Pilih Data yang Akan di Hapus");
    }
```

MEMBUAT FUNGSI CRUD SERVICE DAN COSTUMER

1.Membuat table service di database MySQL seperti ini :

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
☐	1	**id**	int(11)			No	*None*		AUTO_INCREMENT
☐	2	**jenis**	varchar(255)	utf8mb4_general_ci		No	*None*		
☐	3	**harga**	double			No	*None*		
☐	4	**status**	varchar(50)	utf8mb4_general_ci		No	*None*		

2.Membuat class ServiceRepo dan codenya seperti ini :

```java
package DAO;

import java.sql.Connection;

public class ServiceRepo implements ServiceDAO {
    private Connection connection;


    final String insert = "INSERT INTO service(id, jenis, harga, status) VALUES(?,?,?,?);";
    final String select = "SELECT * FROM service;";
    final String delete = "DELETE FROM service WHERE id=?;";
    final String update = "UPDATE service SET jenis=?, harga=?, status=? 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.getId());
            st.setString(2, service.getJenis());
            st.setString(3, service.getHarga());
            st.setString(4, service.getStatus());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

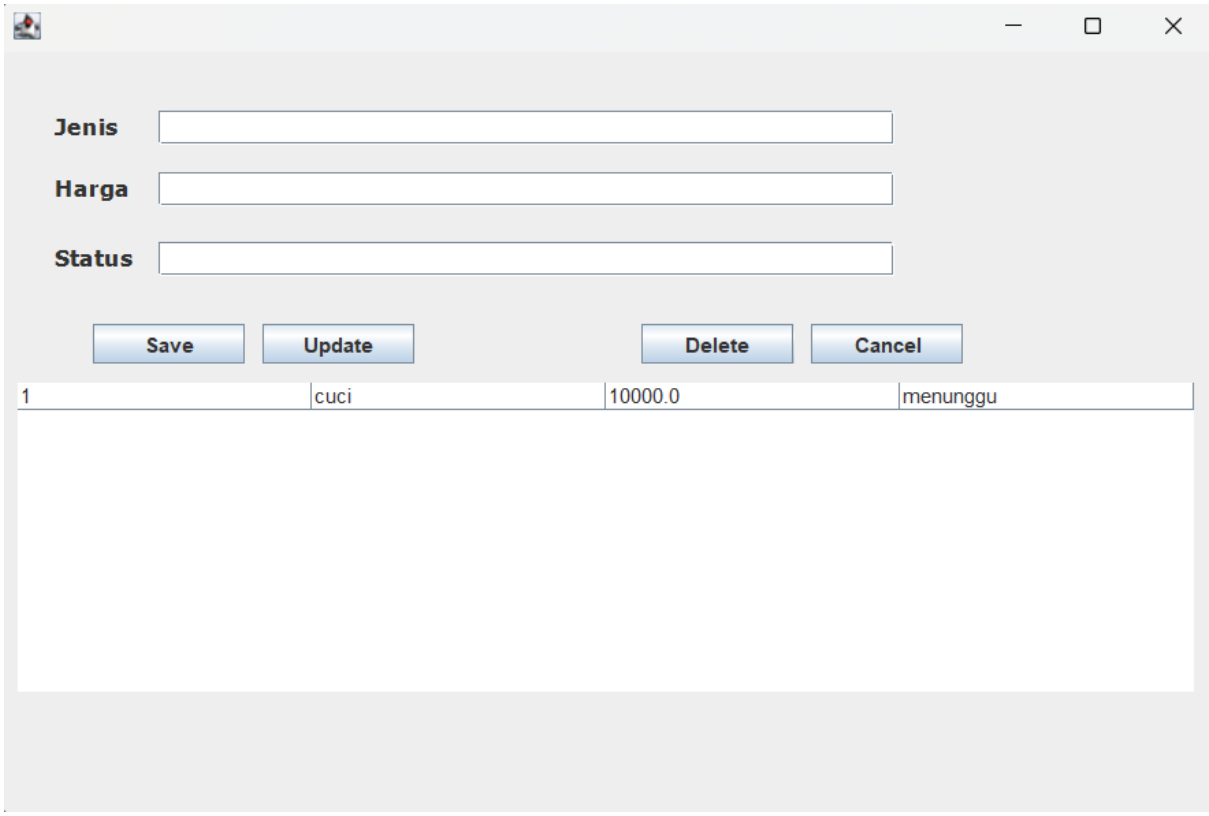
```java
    @Override
    public List<Service> show() {
        List<Service> services = new ArrayList<>();
        Statement st = null;
        ResultSet rs = null;
        try {
            st = connection.createStatement();
            rs = st.executeQuery(select);
            while (rs.next()) {
                Service service = new Service();
                service.setId(rs.getString("id"));
                service.setJenis(rs.getString("jenis"));
                service.setHarga(rs.getString("harga"));
                service.setStatus(rs.getString("status"));
                services.add(service);
            }
        } catch (SQLException e) {
            Logger.getLogger(ServiceDAO.class.getName()).log(Level.SEVERE, null, e);
        } finally {
            try {
                if (rs != null) rs.close();
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        return services;
    }

    @Override
    public void update(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(update);
            st.setString(1, service.getJenis());
            st.setString(2, service.getHarga());
            st.setString(3, service.getStatus());
            st.setString(4, service.getId());
```

```java
    @Override
    public void update(Service service) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(update);
            st.setString(1, service.getJenis());
            st.setString(2, service.getHarga());
            st.setString(3, service.getStatus());
            st.setString(4, service.getId());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    @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 {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

3.Membuat class ServiceDAO dan code seperti ini :

```java
package DAO;

import java.util.List;

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

4.Membuat JFrame untuk Service seperti ini :



5.Membuat method reset pada JFrame seperti ini :

```
public void reset() {
    txtJenis.setText("");
    txtHarga.setText("");
    txtStatus.setText("");
}
```

6.Membuat instance pada JFrame ServiceFrame

```
ServiceRepo srvc = new ServiceRepo();
List<Service> ls;
public String id;
```

7.Membuat fungsi CREATE pada ServiceFrame

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Service service = new Service();
        service.setJenis(txtJenis.getText());
        service.setHarga(txtHarga.getText());
        service.setStatus(txtStatus.getText());
        srvc.save(service);
        reset();
        loadTable();
    }
});
btnSave.setBounds(54, 160, 89, 23);
contentPane.add(btnSave);
```

8.Membuat fungsi READ pada ServiceFrame

```
public void loadTable() {
    ls = srvc.show();
    TableService tu = new TableService(ls);
    tableService.setModel(tu);
    tableService.getTableHeader().setVisible(true);
}
```

9.Membuat fungsi UPDATE pada ServiceFrame

```
tableService = new JTable();
tableService.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableService.getValueAt(tableService.getSelectedRow(), 0).toString();
        txtJenis.setText(tableService.getValueAt(tableService.getSelectedRow(), 1).toString());
        txtHarga.setText(tableService.getValueAt(tableService.getSelectedRow(), 2).toString());
        txtStatus.setText(tableService.getValueAt(tableService.getSelectedRow(), 3).toString());
    }
});
tableService.setBounds(10, 194, 688, 181);
contentPane.add(tableService);
```

Untuk mengupdate table atau mengubah inputan, ketik code dibawah :

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Service service = new Service();
        service.setJenis(txtJenis.getText());
        service.setHarga(txtHarga.getText());
        service.setStatus(txtStatus.getText());
        srvc.update(service);
        reset();
        loadTable();
    }
});
btnUpdate.setBounds(153, 160, 89, 23);
contentPane.add(btnUpdate);
```

10.Membuat fungsi DELETE pada ServiceFrame :

```
JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            srvc.delete(id);
            reset();
            loadTable();
        }else {
            JOptionPane.showMessageDialog(null, "Silahkan Pilih Data yang Akan di Hapus");
        }
    }
});
btnDelete.setBounds(375, 160, 89, 23);
contentPane.add(btnDelete);
```

11.Membuat table Costumer di database MySQL seperti ini :

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**id**	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	**nama**	varchar(128)	utf8mb4_general_ci		No	*None*			Change Drop More
☐	3	**alamat**	text	utf8mb4_general_ci		No	*None*			Change Drop More
☐	4	**nohp**	varchar(15)	utf8mb4_general_ci		No	*None*			Change Drop More

12.Membuat class CostumerRepo dan codenya seperti ini :

```java
package DAO;

import java.sql.Connection;

public class CustomerRepo implements CustomerDAO {

    private Connection connection;

    final String insert = "INSERT INTO costumer(nama, alamat, nohp) VALUES(?, ?, ?);";
    final String select = "SELECT * FROM costumer;";
    final String update = "UPDATE costumer SET nama=?, alamat=?, nohp=? WHERE id=?;";
    final String delete = "DELETE FROM costumer WHERE id=?;";

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

    @Override
    public void save(Costumer costumer) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, costumer.getNama());
            st.setString(2, costumer.getAlamat());
            st.setString(3, costumer.getNohp());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) {
                    st.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

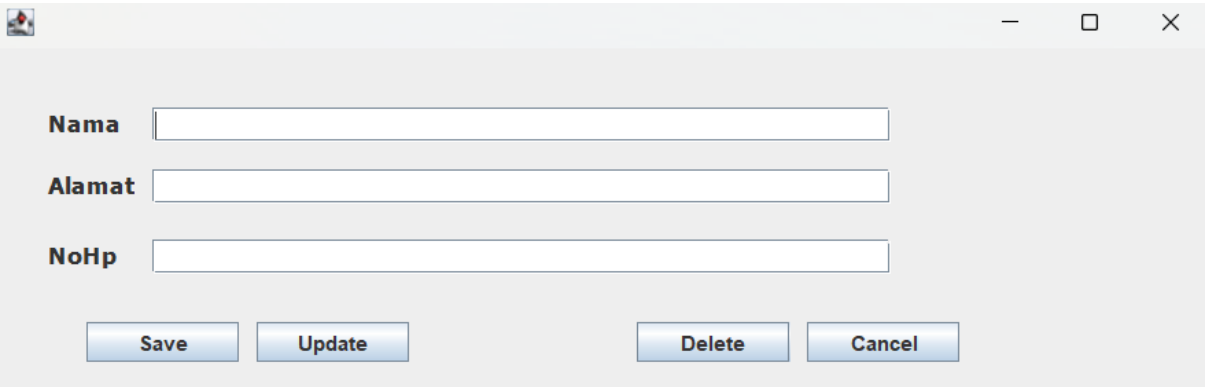
```java
    @Override
    public List<Costumer> show() {
        List<Costumer> ls = new ArrayList<>();
        try {
            Statement st = connection.createStatement();
            ResultSet rs = st.executeQuery(select);
            while (rs.next()) {
                Costumer costumer = new Costumer();
                costumer.setId(rs.getString("id"));
                costumer.setNama(rs.getString("nama"));
                costumer.setAlamat(rs.getString("alamat"));
                costumer.setNohp(rs.getString("nohp"));
                ls.add(costumer);
            }
        } catch (SQLException e) {
            Logger.getLogger(CustomerDAO.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.getNama());
            st.setString(2, costumer.getAlamat());
            st.setString(3, costumer.getNohp());
            st.setString(4, costumer.getId());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) {
                    st.close();
                }
            } catch (SQLException e) {
```

```
            st.setString(2, costumer.getAlamat());
            st.setString(3, costumer.getNohp());
            st.setString(4, costumer.getId());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) {
                    st.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    @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 {
                if (st != null) {
                    st.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

13.Membuat class CostumerDAO dan code seperti ini :

```java
package DAO;

import java.util.List;

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

```

14.Membuat JFrame untuk Costumer seperti ini :



15.Membuat method reset pada JFrame seperti ini :

```java
public void reset() {
    txtNama.setText("");
    txtAlamat.setText("");
    txtNohp.setText("");
}
```

16.Membuat instance pada JFrame ServiceFrame

```java
CustomerRepo cstmr = new CustomerRepo();
List<Costumer> ls;
public String id;
```

17.Membuat fungsi CREATE pada ServiceFrame

```
        JButton btnSave = new JButton("Save");
        btnSave.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Costumer costumer = new Costumer();
                costumer.setNama(txtNama.getText());
                costumer.setAlamat(txtAlamat.getText());
                costumer.setNohp(txtNohp.getText());
                cstmr.save(costumer);
                reset();
                loadTable();
            }
        });
        btnSave.setBounds(54, 160, 89, 23);
        contentPane.add(btnSave);
```

18.Membuat fungsi READ pada ServiceFrame

```
    public void loadTable() {
        ls = cstmr.show();
        TableCustomer tu = new TableCustomer(ls);
        tableCostumer.setModel(tu);
        tableCostumer.getTableHeader().setVisible(true);
    }
}
```

19.Membuat fungsi UPDATE pada ServiceFrame

```
        tableCostumer = new JTable();
        tableCostumer.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                id = tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 0).toString();
                txtNama.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 1).toString());
                txtAlamat.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 2).toString());
                txtNohp.setText(tableCostumer.getValueAt(tableCostumer.getSelectedRow(), 3).toString());
            }
        });
        tableCostumer.setBounds(10, 194, 688, 228);
        contentPane.add(tableCostumer);
```

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Costumer costumer = new Costumer();
        costumer.setNama(txtNama.getText());
        costumer.setAlamat(txtAlamat.getText());
        costumer.setNohp(txtNohp.getText());
        cstmr.update(costumer);
        reset();
        loadTable();
    }
});
btnUpdate.setBounds(153, 160, 89, 23);
contentPane.add(btnUpdate);
```

20.Membuat fungsi DELETE pada ServiceFrame :

```
JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            cstmr.delete(id);
            reset();
            loadTable();
        }else {
            JOptionPane.showMessageDialog(null, "Silahkan Pilih Data yang Akan di Hapus");
        }
    }
});
btnDelete.setBounds(375, 160, 89, 23);
contentPane.add(btnDelete);
```

21.Tampilan pada DATABASE :

←T→				id	jenis	harga	status
☐	Edit	Copy	Delete	1	cuci	10000	menunggu

Membuat Fungsi CRUD untuk OrderDetailFrame

1.Membuat Class OrderDetailDAO

```java
package DAO;

import java.util.List;

public interface OrderDetailDAO {
    void save(OrderDetail orderDetail);
    List<OrderDetail> findAll();
    void update(OrderDetail orderDetail);
    void delete(String idOrderDetail);
    List<OrderDetail> show();
}
```

2.Membuat Class OrderDetailRepo

```java
package DAO;

import java.sql.Connection;

public class OrderDetailRepo implements OrderDetailDAO {
    private Connection connection;

    final String insert = "INSERT INTO order_detail(id_order_detail, id_order, id_layanan, jumlah, total) VALUES(?,?,?,?,?);";
    final String select = "SELECT * FROM order_detail;";
    final String delete = "DELETE FROM order_detail WHERE id_order_detail=?;";
    final String update = "UPDATE order_detail SET id_order=?, id_layanan=?, jumlah=?, total=? WHERE id_order_detail=?;";



    public OrderDetailRepo() {
        connection = Database.Koneksi();
    }
    @Override
    public void save(OrderDetail orderDetail) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(insert);
            st.setString(1, orderDetail.getIdOrderDetail());
            st.setString(2, orderDetail.getIdOrder());
            st.setString(3, orderDetail.getIdLayanan());
            st.setInt(4, orderDetail.getJumlah());
            st.setDouble(5, orderDetail.getTotal());
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

```java
@Override
public List<OrderDetail> show() {
    List<OrderDetail> orderDetails = new ArrayList<>();
    Statement st = null;
    ResultSet rs = null;
    try {
        st = connection.createStatement();
        rs = st.executeQuery(select);
        while (rs.next()) {
            OrderDetail orderDetail = new OrderDetail();
            orderDetail.setIdOrderDetail(rs.getString("id_order_detail"));
            orderDetail.setIdOrder(rs.getString("id_order"));
            orderDetail.setIdLayanan(rs.getString("id_layanan"));
            orderDetail.setJumlah(rs.getInt("jumlah"));
            orderDetail.setTotal(rs.getDouble("total"));
            orderDetails.add(orderDetail);
        }
    } catch (SQLException e) {
        Logger.getLogger(OrderDetailDAO.class.getName()).log(Level.SEVERE, null, e);
    } finally {
        try {
            if (rs != null) rs.close();
            if (st != null) st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    return orderDetails;
}

@Override
public void update(OrderDetail orderDetail) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, orderDetail.getIdOrder());
        st.setInt(2, orderDetail.getJumlah());
        st.setDouble(3, orderDetail.getTotal());
        st.setString(4, orderDetail.getIdOrderDetail());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            if (st != null) st.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```java
    @Override
    public void delete(String idOrderDetail) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(delete);
            st.setString(1, idOrderDetail);
            st.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public List<OrderDetail> findAll() {
        return null;
    }
}
```

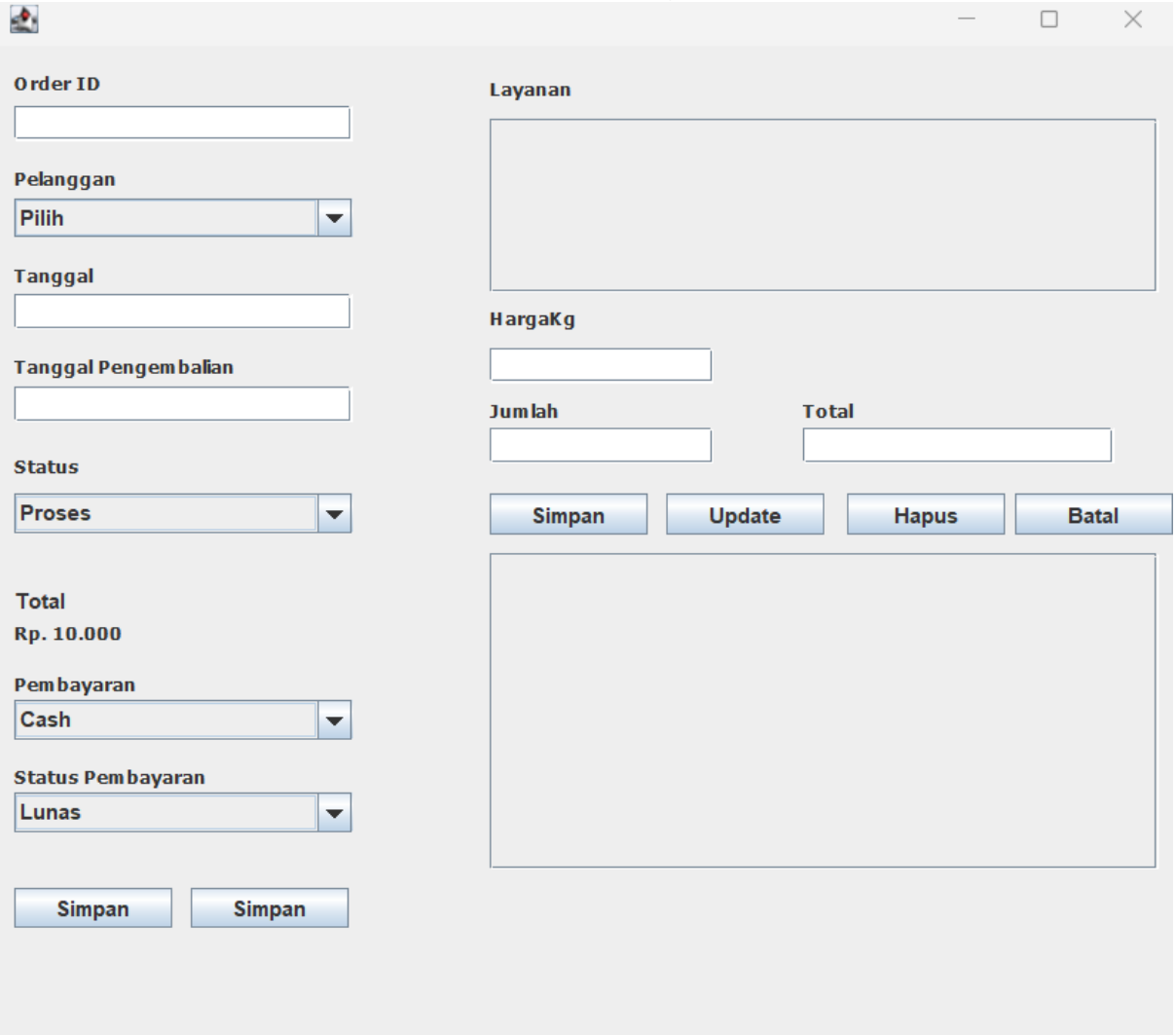
3.Membuat Class OrderDetail pada Package model

```java
package model;

public class OrderDetail {
    private String idOrderDetail;
    private String idOrder;
    private int jumlah;
    private double total;
    private String idLayanan;

    public String getIdLayanan() {
        return idLayanan;
    }

    public void setIdLayanan(String idLayanan) {
        this.idLayanan = idLayanan;
    }

    public String getIdOrderDetail() {
        return idOrderDetail;
    }

    public void setIdOrderDetail(String idOrderDetail) {
        this.idOrderDetail = idOrderDetail;
    }

    public String getIdOrder() {
        return idOrder;
    }

    public void setIdOrder(String idOrder) {
        this.idOrder = idOrder;
    }

    public int getJumlah() {
        return jumlah;
    }

    public void setJumlah(int jumlah) {
        this.jumlah = jumlah;
    }

    public double getTotal() {
        return total;
    }

    public void setTotal(double total) {
        this.total = total;
    }
}
```

4.Membuat Class TableOrderDetail pada Package table

```java
package table;

import javax.swing.table.AbstractTableModel;

public class TableOrderDetail extends AbstractTableModel {
    List<OrderDetail> OrderDetail;
    private String[] columnNames = {"ID Order Detail", "ID Order", "ID Layanan", "Jumlah", "Total"};

    public TableOrderDetail(List<OrderDetail> OrderDetail) {
        this.OrderDetail = OrderDetail;
    }

    @Override
    public int getRowCount() {
        return OrderDetail.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) {
        OrderDetail orderDetail = OrderDetail.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return orderDetail.getIdOrderDetail();
            case 1:
                return orderDetail.getIdOrder();
            case 2:
                return orderDetail.getIdLayanan();
            case 3:
                return orderDetail.getJumlah();
            case 4:
                return orderDetail.getTotal();
            default:
                return null;
        }
    }
}
```

5.Membuat Class OrderDetailFrame di Package ui



6.Membuat Logika pada TextField yang Bervariable textJumlah

```
textJumlah = new JTextField();
textJumlah.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        double hargaKg = Double.parseDouble(textHargakg.getText());
        int jumlah = Integer.parseInt(textJumlah.getText());
        double total = hargaKg * jumlah;
        textTotal.setText(String.valueOf(total));
    }
});
textJumlah.setColumns(10);
textJumlah.setBounds(277, 215, 126, 20);
contentPane.add(textJumlah);
```

7.Membuat Fungsi Create pada btnSave

```
JButton btnSave = new JButton("Simpan");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OrderDetail detail = new OrderDetail();
        detail.setIdLayanan(id);
        detail.setIdOrder(txtidOrder.getText());
        int jumlah = Integer.parseInt(textJumlah.getText());
        detail.setJumlah(jumlah);
        double total = Double.parseDouble(textTotal.getText());
        detail.setTotal(total);
        orderDetailRepo.save(detail);
        reset();
        loadTableOrderDetail();
    }
});
btnSave.setBounds(277, 252, 89, 23);
contentPane.add(btnSave);
```

8.Membuat Fungsi Update pada btnUpdate

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OrderDetail detail = new OrderDetail();
        detail.setIdLayanan(id);
        detail.setIdOrder(txtidOrder.getText());
        int jumlah = Integer.parseInt(textJumlah.getText());
        detail.setJumlah(jumlah);
        double total = Double.parseDouble(textTotal.getText());
        detail.setTotal(total);
        orderDetailRepo.save(detail);
        reset();
        loadTableOrderDetail();
    }
});
btnUpdate.setBounds(376, 252, 89, 23);
contentPane.add(btnUpdate);
```

8.Membuat Fungsi Delete pada btnDelete

```
JButton btnDelete = new JButton("Hapus");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id1 != null) {
            orderDetailRepo.delete(id1);
            reset();
            loadTableOrderDetail();
        }else {
            JOptionPane.showMessageDialog(null, "Silahkan Pilih Data yang Akan di Hapus");
        }
    }
});
btnDelete.setBounds(478, 252, 89, 23);
contentPane.add(btnDelete);
```

10.Membuat Fungsi pada tableLayanan

```
tableLayanan = new JTable();
tableLayanan.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableLayanan.getValueAt(tableLayanan.getSelectedRow(), 0).toString();
            textHargakg.setText(tableLayanan.getValueAt(tableLayanan.getSelectedRow(), 2).toString());
    }
});
tableLayanan.setBounds(276, 45, 323, 100);
contentPane.add(tableLayanan);
```

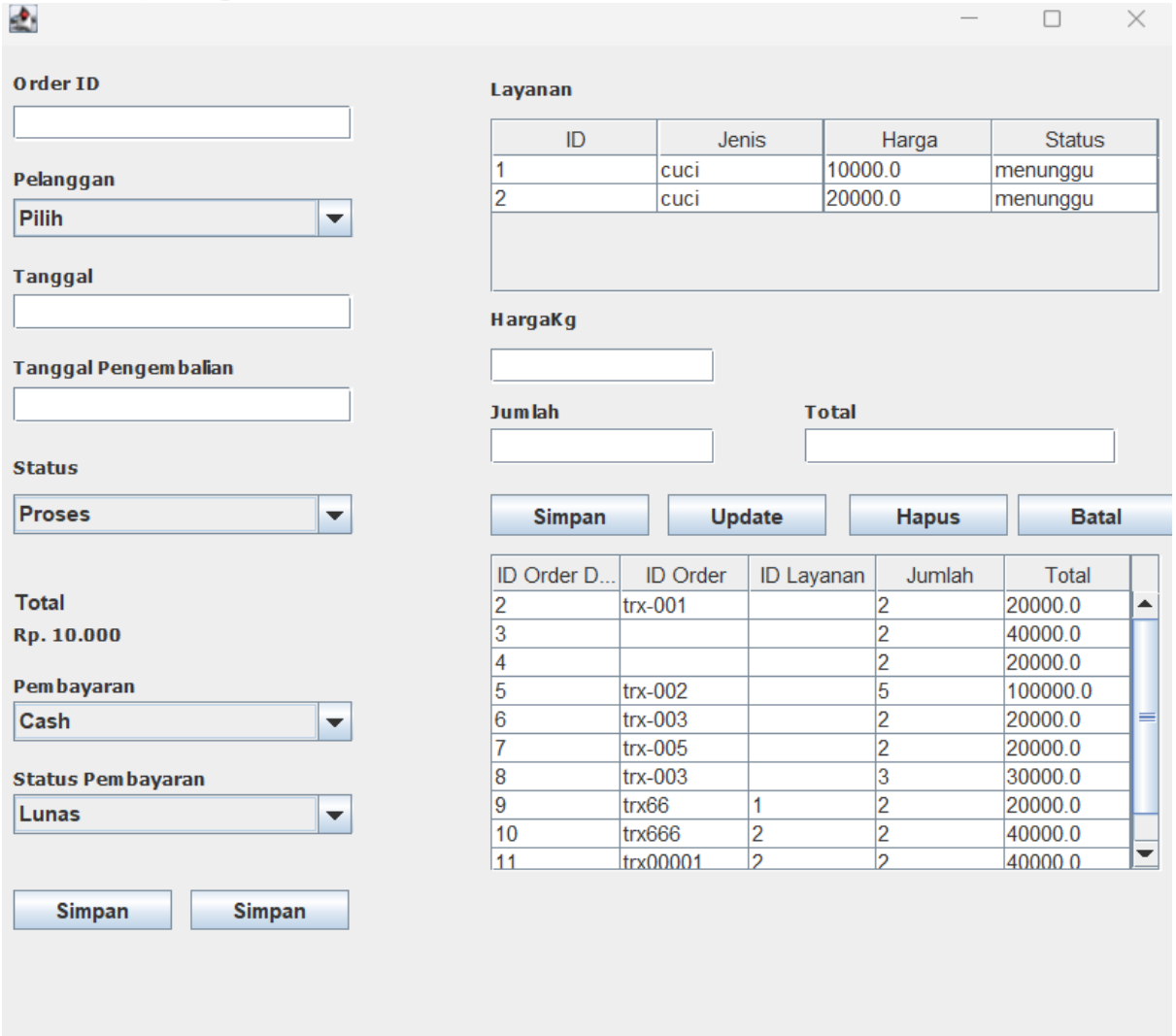
11.Membuat Fungsi untuk tableOrderDetail

```
tableOrderDetail = new JTable();
tableOrderDetail.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableLayanan.getValueAt(tableLayanan.getSelectedRow(), 2).toString();
        id1 = tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(), 0).toString();
        textJumlah.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(), 3).toString());
        textTotal.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(), 4).toString());
        txtidOrder.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(), 1).toString());
    }
});
tableOrderDetail.setBounds(277, 304, 323, 138);
contentPane.add(tableOrderDetail);
```

12.Method tambahan untuk Load Table dari Database

```
public void reset() {
    txtidOrder.setText("");
    textJumlah.setText("");
    textTotal.setText("");
}
ServiceRepo srvc = new ServiceRepo();
List<Service> ls;
public String id;

OrderDetailRepo orderDetailRepo = new OrderDetailRepo();
List<OrderDetail> ls1;
public String id1;
private JTable tableOrderDetail;

public void loadTableOrderDetail() {
    ls1 = orderDetailRepo.show();
    TableOrderDetail tu = new TableOrderDetail(ls1);
    tableOrderDetail.setModel(tu);
    tableOrderDetail.getTableHeader().setVisible(true);
}

public void loadTable() {
    ls = srvc.show();
    TableService tu = new TableService(ls);
    tableLayanan.setModel(tu);
    tableLayanan.getTableHeader().setVisible(true);
```

13.Hasilnya Seperti ini :



Membuat Method Total untuk Kolom total di tableOrderDetail untuk OrderDetailFrame dan Membuat OrderFrame serta CRUD nya

1.Membuat Method pada class OrderDetailRepo seperti Code pada gambar :

```java
@Override
public String total(String id_order) {
    String query_total = "SELECT sum(total) as total from order_detail WHERE id_order= "+"'"+id_order+"'"+";";
    Statement st;
    ResultSet rs;
    String result="";
    try {
        st = connection.createStatement();
        rs = st.executeQuery(query_total);
        if(rs.next()) {
            result = "" +rs.getDouble(1);
        }else {
            result = "0";
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return result;


}
```

2.Menambahkan Method pada btnSave pada class OrderDetailFrame

```java
JButton btnSave = new JButton("Simpan");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String orderId = txtidOrder.getText().trim();
        System.out.println("ID Order sebelum memanggil total: " + orderId);
        if (orderId.isEmpty()) {
            JOptionPane.showMessageDialog(null, "ID Order tidak boleh kosong!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        OrderDetail detail = new OrderDetail();
        detail.setIdLayanan(id);
        detail.setIdOrder(orderId);
        int jumlah = Integer.parseInt(textJumlah.getText());
        detail.setJumlah(jumlah);
        double total = Double.parseDouble(textTotal.getText());
        detail.setTotal(total);
        orderDetailRepo.save(detail);
        reset();
        loadTableOrderDetail();
        lblRp.setText("Rp. " + orderDetailRepo.total(orderId));
    }
});
btnSave.setBounds(277, 252, 89, 23);
contentPane.add(btnSave);
```

3. Menambahkan Method pada btnUpdate pada class OrderDetailFrame

```java
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (id1 == null) {
            JOptionPane.showMessageDialog(null, "Silahkan pilih detail yang akan diupdate", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        OrderDetail detail = new OrderDetail();
        detail.setIdOrderDetail(id1);
        detail.setIdLayanan(id);
        detail.setIdOrder(txtidOrder.getText());
        int jumlah = Integer.parseInt(textJumlah.getText());
        detail.setJumlah(jumlah);
        double total = Double.parseDouble(textTotal.getText());
        detail.setTotal(total);

        orderDetailRepo.update(detail);

        String orderId = txtidOrder.getText();
        lblRp.setText("Rp. " + orderDetailRepo.total(orderId));
        reset();
        loadTableOrderDetail();
    }
});
btnUpdate.setBounds(376, 252, 89, 23);
contentPane.add(btnUpdate);
```

4. Menambahkan Method pada btnDetele pada class OrderDetailFrame

```java
JButton btnDelete = new JButton("Hapus");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (id1 != null) {
            orderDetailRepo.delete(id1);
            String orderId = txtidOrder.getText();
            lblRp.setText("Rp. " + orderDetailRepo.total(orderId));
            reset();
            loadTableOrderDetail();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan Pilih Data yang Akan di Hapus");
        }
    }
});
btnDelete.setBounds(478, 252, 89, 23);
contentPane.add(btnDelete);
```

5.Membuat Class OrderDAO pada package DAO

```java
package DAO;

import java.util.List;

public interface OrderDAO {
    void save(Order order);
    List<Order> show();
    void delete(String id);
    void update(Order order);
}
```

6.Membuat Class OrderRepo pada package DAO

```java
import java.sql.Connection;


public class OrderRepo implements OrderDAO {
    private Connection connection;

    final String insert = "INSERT INTO `order` (id_order, nama, pembayaran, total, tanggal, tanggal_peng, status, status_pembayaran) VALUES (?, ?, ?, ?,
    final String select = "SELECT * FROM `order`;";
    final String delete = "DELETE FROM `order` WHERE id_order=?;";
    final String update = "UPDATE `order` SET nama=?, pembayaran=?, total=?, tanggal=?, tanggal_peng=?, status=?, status_pembayaran=? WHERE id_order=?;";

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

    @Override
    public void save(Order order) {
        PreparedStatement st = null;
        try {

            System.out.println("Menyimpan Order: ");
            System.out.println("ID: " + order.getId());
            System.out.println("Nama Customer: " + order.getNama_costumer());
            System.out.println("Pembayaran: " + order.getpembayaran());
            System.out.println("Tanggal: " + order.getTanggal());
            System.out.println("Tanggal Peng: " + order.getTanggal_selesai());
            System.out.println("Status: " + order.getStatus());
            System.out.println("Status Pembayaran: " + order.getStatus_pembayaran());
            System.out.println("Total sebelum konversi: " + order.getTotal());

```

```java
            int total = Integer.parseInt(order.getTotal().replace("Rp. ", "").replace(".", "").trim());
            System.out.println("Total setelah konversi: " + total);

            st = connection.prepareStatement(insert);
            st.setString(1, order.getId());
            st.setString(2, order.getNama_costumer());
            st.setString(3, order.getpembayaran());
            st.setInt(4, total);
            st.setString(5, order.getTanggal());
            st.setString(6, order.getTanggal_selesai());
            st.setString(7, order.getStatus());
            st.setString(8, order.getStatus_pembayaran());
            st.executeUpdate();
            System.out.println("Order berhasil disimpan.");

        } catch (SQLException e) {
            e.printStackTrace();
        } catch (NumberFormatException e) {
            System.out.println("Format total tidak valid: " + order.getTotal());
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

```java
    @Override
    public List<Order> show() {
        List<Order> orders = new ArrayList<>();
        Statement st = null;
        ResultSet rs = null;
        try {
            st = connection.createStatement();
            rs = st.executeQuery(select);
            while (rs.next()) {
                Order order = new Order();
                order.setId(rs.getString("id_order"));
                order.setNama_costumer(rs.getString("nama"));
                order.setTotal(rs.getString("total"));
                order.setTanggal(rs.getString("tanggal"));
                order.setTanggal_selesai(rs.getString("tanggal_peng"));
                order.setStatus(rs.getString("status"));
                order.setStatus_pembayaran(rs.getString("status_pembayaran"));
                orders.add(order);
            }
        } catch (SQLException e) {
            Logger.getLogger(OrderDAO.class.getName()).log(Level.SEVERE, null, e);
        } finally {
            try {
                if (rs != null) rs.close();
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
```

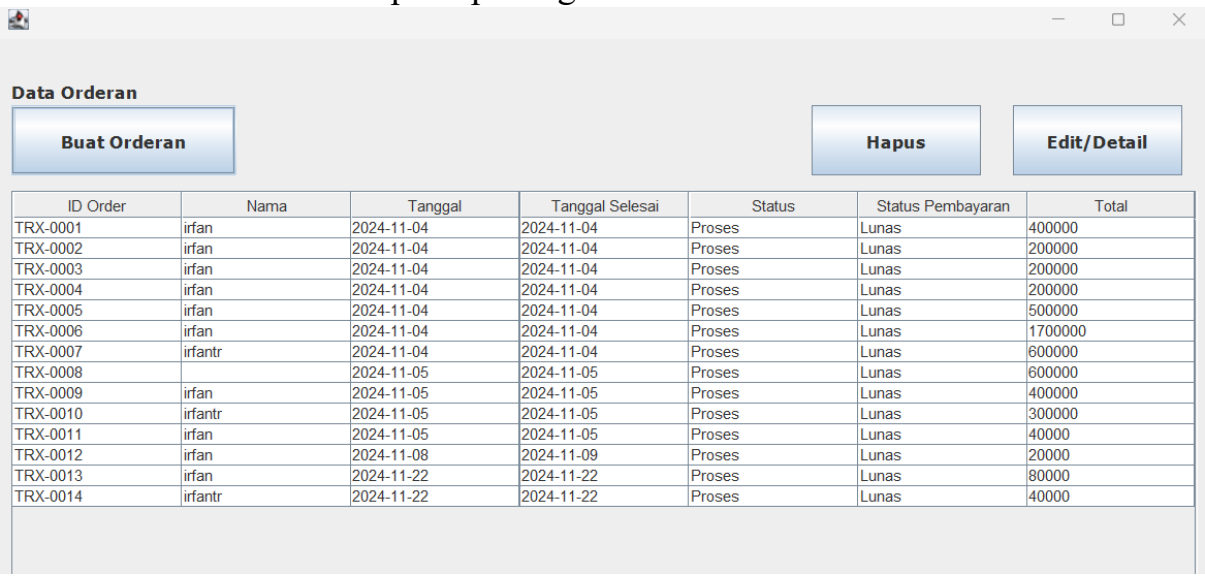
```java
        }
        return orders;
    }

    @Override
    public void update(Order order) {
        PreparedStatement st = null;
        try {

            System.out.println("Memperbarui Order: ");
            System.out.println("ID: " + order.getId());
            System.out.println("Nama Customer: " + order.getNama_costumer());
            System.out.println("Pembayaran: " + order.getpembayaran());
            System.out.println("Tanggal: " + order.getTanggal());
            System.out.println("Tanggal Peng: " + order.getTanggal_selesai());
            System.out.println("Status: " + order.getStatus());
            System.out.println("Status Pembayaran: " + order.getStatus_pembayaran());
            System.out.println("Total sebelum update: " + order.getTotal());

            // Konversi total
            int total = Integer.parseInt(order.getTotal().replace("Rp. ", "").replace(".", "").trim());
            System.out.println("Total setelah konversi: " + total);

            st = connection.prepareStatement(update);
            st.setString(1, order.getNama_costumer());
            st.setString(2, order.getpembayaran());
            st.setInt(3, total); // Total
            st.setString(4, order.getTanggal());
            st.setString(5, order.getTanggal_selesai());
            st.setString(6, order.getStatus());
```

```java
            st.setString(7, order.getStatus_pembayaran());
            st.setString(8, order.getId());
            st.executeUpdate();
            System.out.println("Order berhasil diperbarui.");

        } catch (SQLException e) {
            e.printStackTrace();
        } catch (NumberFormatException e) {
            System.out.println("Format total tidak valid: " + order.getTotal());
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    @Override
    public void delete(String id) {
        PreparedStatement st = null;
        try {
            st = connection.prepareStatement(delete);
            st.setString(1, id);
            st.executeUpdate();
            System.out.println("Order dengan ID " + id + " berhasil dihapus.");
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
```

```java
        } finally {
            try {
                if (st != null) st.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }

    public String getLastOrderId() {
        String lastOrderId = null;
        String query = "SELECT id_order FROM `order` ORDER BY id_order DESC LIMIT 1";

        try (PreparedStatement statement = connection.prepareStatement(query);
             ResultSet resultSet = statement.executeQuery()) {
            if (resultSet.next()) {
                lastOrderId = resultSet.getString("id_order");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }

        return lastOrderId;
    }


    public String generateNewOrderId() {
        String lastId = getLastOrderId();
        if (lastId != null && lastId.startsWith("TRX-")) {
            int nextId = Integer.parseInt(lastId.substring(4)) + 1;
```

```java
    public String generateNewOrderId() {
        String lastId = getLastOrderId();
        if (lastId != null && lastId.startsWith("TRX-")) {
            int nextId = Integer.parseInt(lastId.substring(4)) + 1;
            return String.format("TRX-%04d", nextId);
        } else {
            return "TRX-0001";
        }
    }
}
```

7.Buat Class OrderFrame pada package ui



ID Order	Nama	Tanggal	Tanggal Selesai	Status	Status Pembayaran	Total
TRX-0001	irfan	2024-11-04	2024-11-04	Proses	Lunas	400000
TRX-0002	irfan	2024-11-04	2024-11-04	Proses	Lunas	200000
TRX-0003	irfan	2024-11-04	2024-11-04	Proses	Lunas	200000
TRX-0004	irfan	2024-11-04	2024-11-04	Proses	Lunas	200000
TRX-0005	irfan	2024-11-04	2024-11-04	Proses	Lunas	500000
TRX-0006	irfan	2024-11-04	2024-11-04	Proses	Lunas	1700000
TRX-0007	irfantr	2024-11-04	2024-11-04	Proses	Lunas	600000
TRX-0008		2024-11-05	2024-11-05	Proses	Lunas	600000
TRX-0009	irfan	2024-11-05	2024-11-05	Proses	Lunas	400000
TRX-0010	irfantr	2024-11-05	2024-11-05	Proses	Lunas	300000
TRX-0011	irfan	2024-11-05	2024-11-05	Proses	Lunas	40000
TRX-0012	irfan	2024-11-08	2024-11-09	Proses	Lunas	20000
TRX-0013	irfan	2024-11-22	2024-11-22	Proses	Lunas	80000
TRX-0014	irfantr	2024-11-22	2024-11-22	Proses	Lunas	40000

8.Method untuk Load TableOrder

```java
    public void loadTable() {
        orders = orderRepo.show();
        String[] columnNames = {"ID Order", "Nama", "Tanggal", "Tanggal Selesai", "Status", "Status Pembayaran", "Total"};
        DefaultTableModel model = new DefaultTableModel(columnNames, 0);

        for (Order order : orders) {
            System.out.println("Nama Costumer: " + order.getNama_costumer());
            Object[] rowData = {
                order.getId(),
                order.getNama_costumer(),
                order.getTanggal(),
                order.getTanggal_selesai(),
                order.getStatus(),
                order.getStatus_pembayaran(),
                order.getTotal()
            };
            model.addRow(rowData);
        }

        tableOrder.setModel(model);
        tableOrder.getTableHeader().setVisible(true);
    }
}
```

9.Generate Order ID secara otomatis

tambahkan code pada class OrderRepo

```
public String getLastOrderId() {
    String lastOrderId = null;
    String query = "SELECT id_order FROM `order` ORDER BY id_order DESC LIMIT 1";

    try (PreparedStatement statement = connection.prepareStatement(query);
         ResultSet resultSet = statement.executeQuery()) {
        if (resultSet.next()) {
            lastOrderId = resultSet.getString("id_order");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return lastOrderId;
}


public String generateNewOrderId() {
    String lastId = getLastOrderId();
    if (lastId != null && lastId.startsWith("TRX-")) {
        int nextId = Integer.parseInt(lastId.substring(4)) + 1;
        return String.format("TRX-%04d", nextId);
    } else {
        return "TRX-0001";
    }
}
```

Menambhakan code berikut pada btnSimpan

```
String idOrder = orderRepo.generateNewOrderId();
```

10.Menambahkan JSpinner untuk mengatur tanggal order dan tanggal pengembalian

import seperti code dibawah :

```
10 import java.text.SimpleDateFormat;
11 import javax.swing.JSpinner;
12 import javax.swing.SpinnerDateModel;
13 import java.util.Date;
```

Lalu untuk menggunakannya, tambahkan code dibawah :

```
spinnerTanggal = new JSpinner(new SpinnerDateModel());
JSpinner.DateEditor editorTanggal = new JSpinner.DateEditor(spinnerTanggal, "dd/MM/yyyy");
spinnerTanggal.setEditor(editorTanggal);
spinnerTanggal.setBounds(10, 140, 190, 20);
contentPane.add(spinnerTanggal);

spinnerTanggalPeng = new JSpinner(new SpinnerDateModel());
JSpinner.DateEditor editorTanggalPeng = new JSpinner.DateEditor(spinnerTanggalPeng, "dd/MM/yyyy");
spinnerTanggalPeng.setEditor(editorTanggalPeng);
spinnerTanggalPeng.setBounds(10, 192, 190, 20);
contentPane.add(spinnerTanggalPeng);
```

Hasilnya :

11.Method Edit/Detail pada OrderFrame

Tambahkan code ini pada OrderRepo

```
public Order getOrderById(String id) {
    String query = "SELECT * FROM `order` WHERE id_order = ?";
    Order order = null;
    try (PreparedStatement st = connection.prepareStatement(query)) {
        st.setString(1, id);
        try (ResultSet rs = st.executeQuery()) {
            if (rs.next()) {
                order = new Order();
                order.setId(rs.getString("id_order"));
                order.setNama_costumer(rs.getString("nama"));
                order.setTotal(rs.getString("total"));
                order.setTanggal(rs.getString("tanggal"));
                order.setTanggal_selesai(rs.getString("tanggal_peng"));
                order.setStatus(rs.getString("status"));
                order.setStatus_pembayaran(rs.getString("status_pembayaran"));
                order.setpembayaran(rs.getString("pembayaran"));
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return order;
}
```

Method untuk load data dan frame OrderDetailFrame :

```
btnEdit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int selectedRow = tableOrder.getSelectedRow();
        if (selectedRow >= 0) {
            String orderId = tableOrder.getValueAt(selectedRow, 0).toString();
            Order selectedOrder = orderRepo.getOrderById(orderId);
            if (selectedOrder != null) {
                OrderDetailFrame detailFrame = new OrderDetailFrame(selectedOrder);
                detailFrame.setVisible(true);
                detailFrame.addWindowListener(new java.awt.event.WindowAdapter() {
                    @Override
                    public void windowClosed(java.awt.event.WindowEvent windowEvent) {
                        loadTable();
                    }
                });
            } else {
                JOptionPane.showMessageDialog(null, "Data order tidak ditemukan!", "Error", JOptionPane.ERROR_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog(null, "Pilih baris data terlebih dahulu!", "Warning", JOptionPane.WARNING_MESSAGE);
        }
    }
});
```

Constructor dan method loadorderdetails untuk load dari OrderFrame ke OrderDetailFrame

```
public OrderDetailFrame(Order order) {
    this();
    loadOrderDetails(order);
    loadTable();
}

private void loadOrderDetails(Order order) {
    txtidOrder.setText(order.getId());
    txtPelanggan.setText(order.getNama_costumer());
    txtTanggal.setText(order.getTanggal());
    txtTanggalPeng.setText(order.getTanggal_selesai());
    comboboxStatus.setSelectedItem(order.getStatus());
    comboboxPembayaran.setSelectedItem(order.getpembayaran());
    comboBoxStatusPembayaran.setSelectedItem(order.getStatus_pembayaran());
}
```