

**LAPORAN AKHIR PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK**



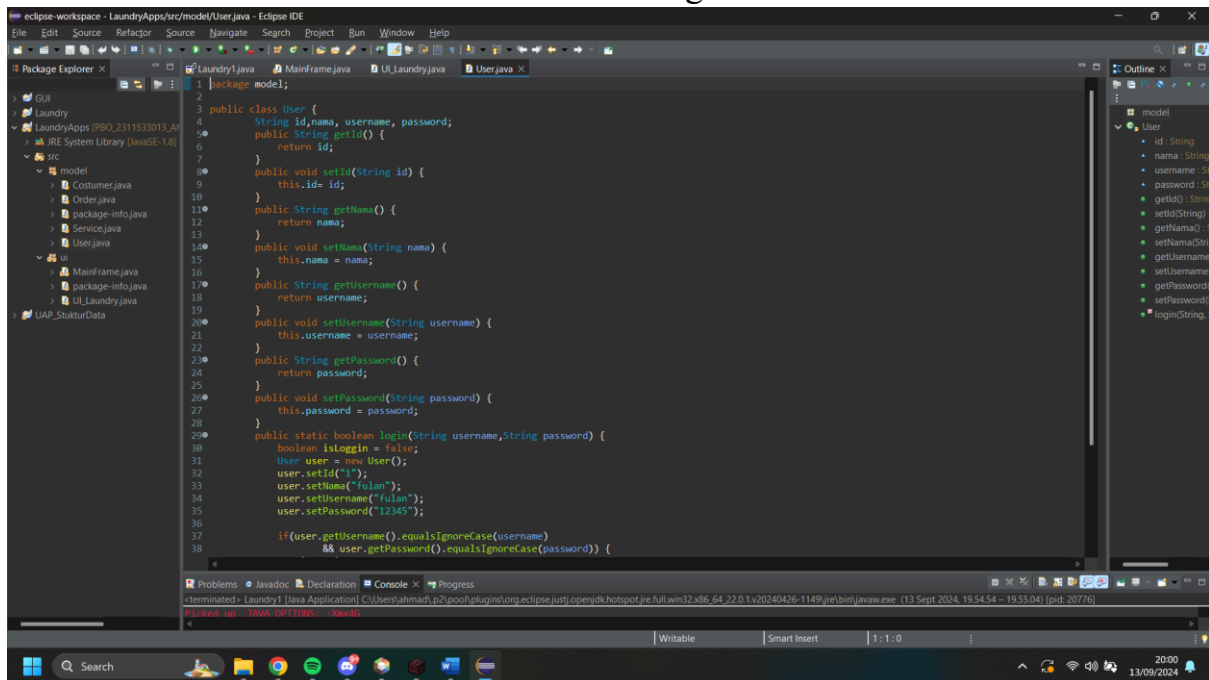
OLEH :  
AHMAD MUHAJMIN KAMIL  
2311533013

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.\



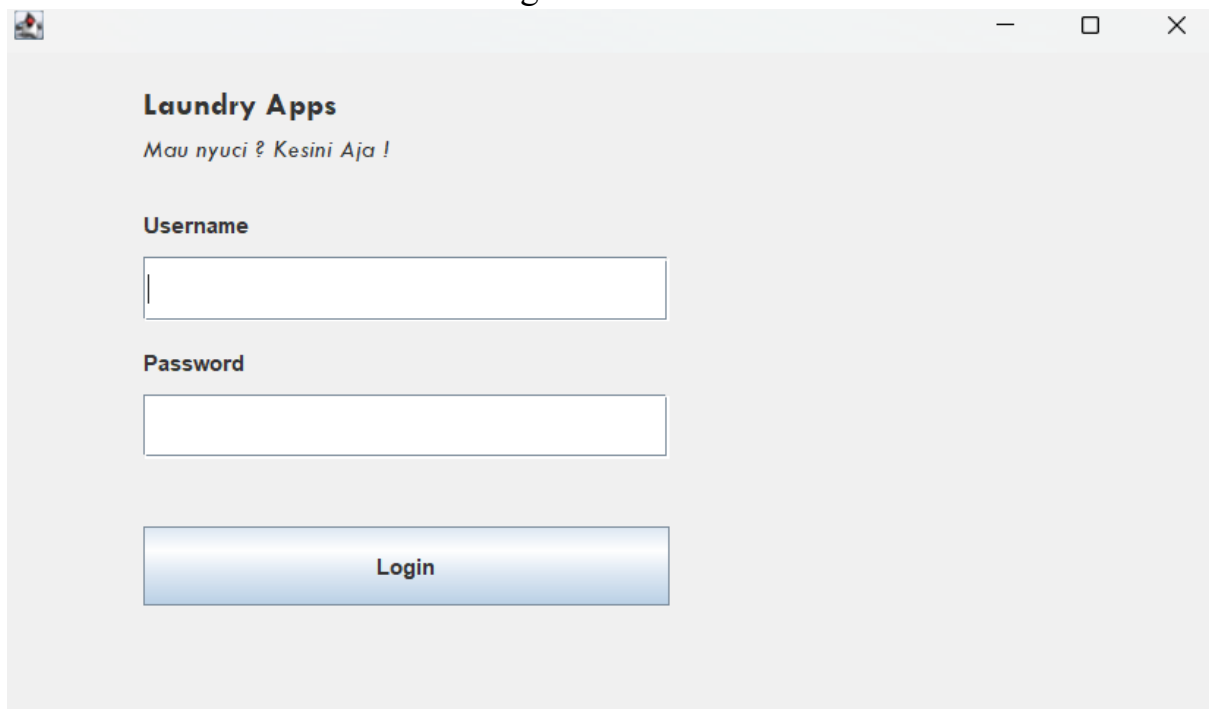
Terdiri dari id, nama, username, password.

## 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.

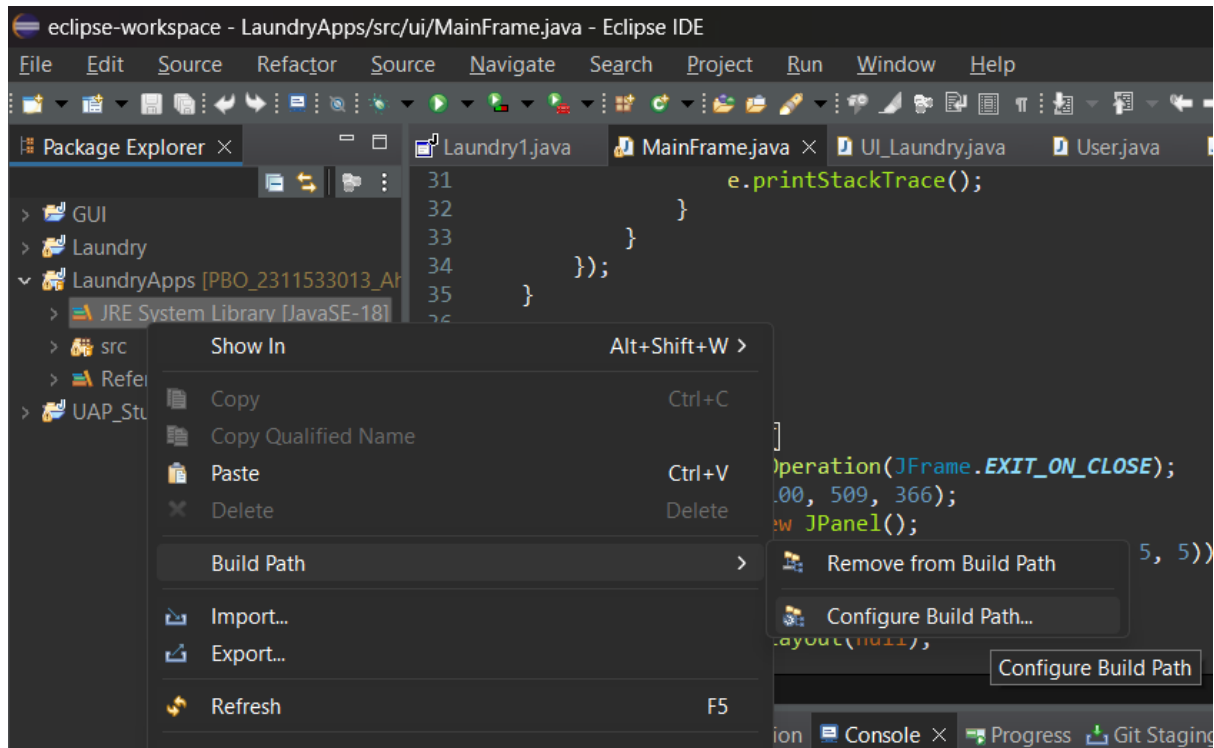
Jika login berhasil (nilai true dikembalikan), objek MainFrame akan dibuat dan ditampilkan dengan menggunakan `new MainFrame().setVisible(true)`. Pada saat yang bersamaan, frame saat ini akan ditutup menggunakan `dispose()`. Namun, jika login gagal (nilai false dikembalikan), pesan kesalahan akan ditampilkan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

## 5.Mendesain tampilan Halaman Utama

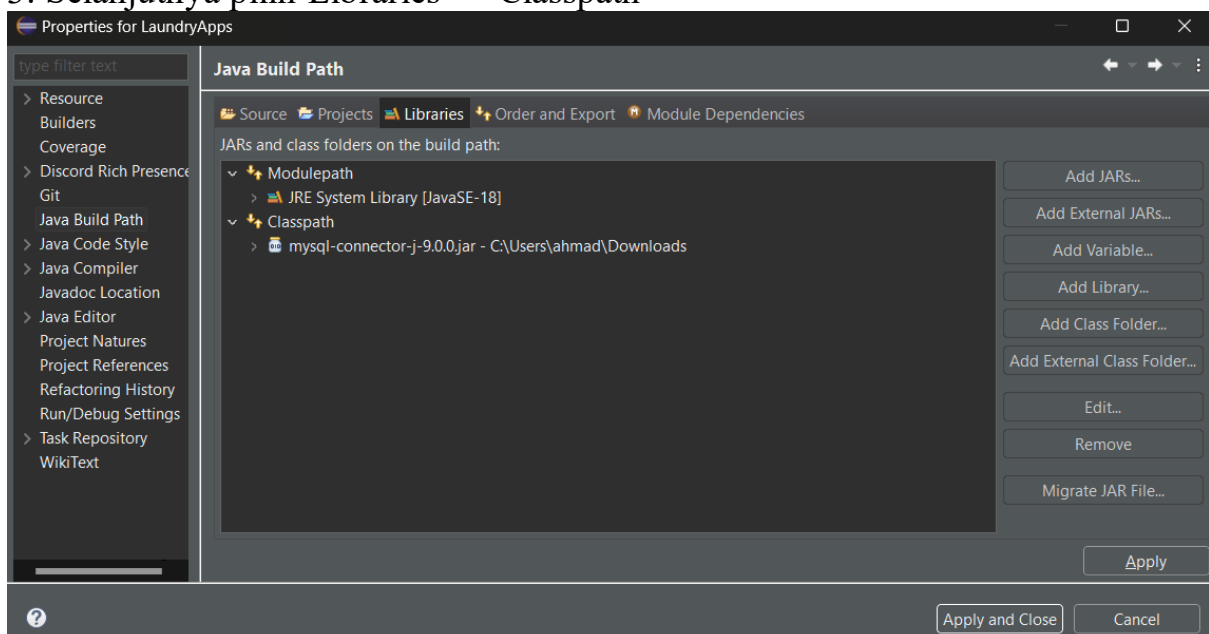


## MEMBUAT FUNGSI CRUD USER DENGAN DATABASE MYSQL

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

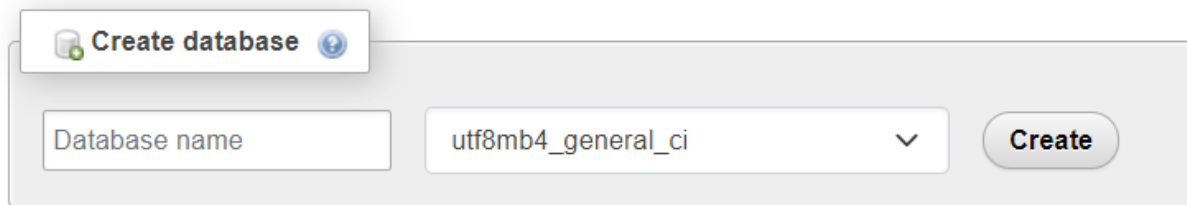
Run XAMPP , dan klik Start pada Apache dan MySQL

#### 5. Ketik link berikut pada browser yang digunakan

localhost/phpMyAdmin

#### 6. Klik new dan buat database dengan nama laundry\_apps

## Databases



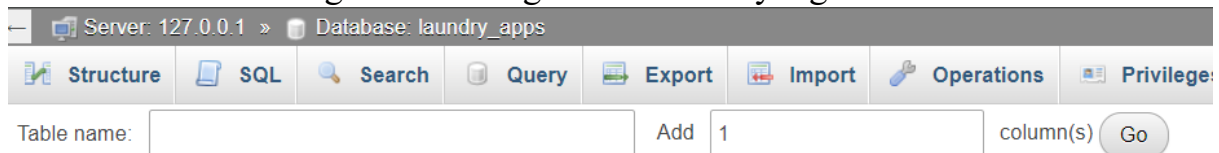
Create database

Database name

utf8mb4\_general\_ci

Create

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

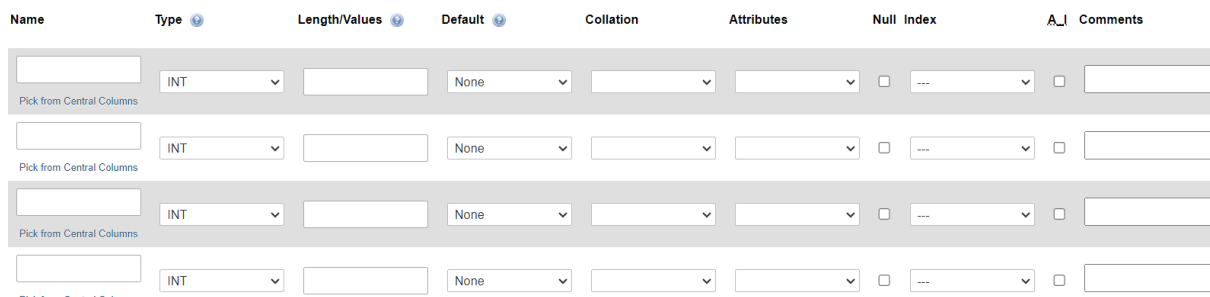


Server: 127.0.0.1 » Database: laundry\_apps

Structure SQL Search Query Export Import Operations Privileges

Table name: Add 1 column(s) Go

#### 8. Klik Create dan akan muncul seperti ini :



Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	Comments
	INT		None			<input type="checkbox"/>	---	
<small>Pick from Central Columns</small>								
	INT		None			<input type="checkbox"/>	---	
<small>Pick from Central Columns</small>								
	INT		None			<input type="checkbox"/>	---	
<small>Pick from Central Columns</small>								
	INT		None			<input type="checkbox"/>	---	
<small>Pick from Central Columns</small>								

Dan isi seperti berikut ini :

The screenshot shows the MySQL Workbench Structure tab for a table named 'users'. The table has four columns: 'id' (INT, PRIMARY), 'name' (VARCHAR(128)), 'username' (VARCHAR(128)), and 'password' (TEXT). The table is using the InnoDB storage engine. The interface includes a toolbar with various tools like Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, Tracking, and Designer. Below the column list, there are fields for table comments, collation, and storage engine. At the bottom, there is a section for partitioning with fields for 'Partition by' and 'Partitions'.

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

The screenshot shows a Java Swing JFrame window titled 'UserFrame'. It contains three text input fields labeled 'Nama', 'Username', and 'Password'. Below the input fields, there are four buttons: 'Save', 'Update', 'Delete', and 'Cancel'. The window has a standard title bar with minimize, maximize, and close buttons.

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

```
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

```
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 instansiasi Connection, membuat constructor dan membuat String untuk melakukan manipulasi database.



```

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

```

@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 :

```

@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 :

```

@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:

```

@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 kemudian 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 dengan kode program berikut:

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

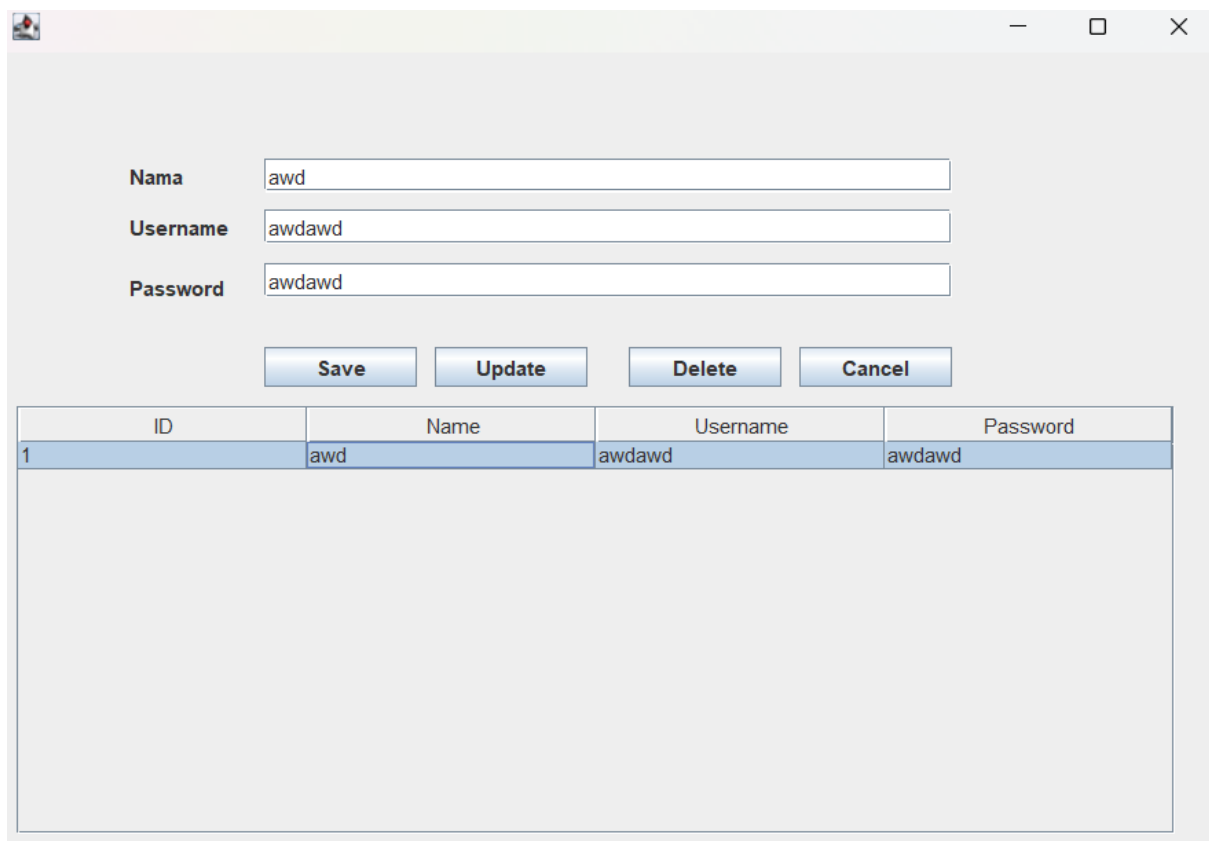
23. Memanggil method pada class main, sehingga Ketika pertama kali program dijalankan maka loadTable akan dipanggil.

```
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 :



The screenshot shows a Java Swing window with a light gray background. At the top, there are three standard window control buttons (minimize, maximize, close). Below the controls, there are three text input fields with labels to their left: "Nama" (containing "awd"), "Username" (containing "awdawd"), and "Password" (containing "awdawd"). Below these fields are four buttons: "Save", "Update", "Delete", and "Cancel". At the bottom of the window is a JTable with four columns: "ID", "Name", "Username", and "Password". The first row of the table contains the values "1", "awd", "awdawd", and "awdawd". The table has a blue header and a light blue selected row.

ID	Name	Username	Password
1	awd	awdawd	awdawd

26. Untuk membuat button Update berfungsi, Klik kanan tombol update → add event handler → action → actionPerformed 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
<input type="checkbox"/>	1	id	int(11)			No	None		AUTO_INCREMENT
<input type="checkbox"/>	2	jenis	varchar(255)	utf8mb4_general_ci		No	None		
<input type="checkbox"/>	3	harga	double			No	None		
<input type="checkbox"/>	4	status	varchar(50)	utf8mb4_general_ci		No	None		

## 2.Membuat class ServiceRepo dan codenya seperti ini :

```
1 package DAO;
2
3 import java.sql.Connection;
4
15 public class ServiceRepo implements ServiceDAO {
16     private Connection connection;
17
18     final String insert = "INSERT INTO service(id, jenis, harga, status) VALUES(?,?,?,?);";
19     final String select = "SELECT * FROM service;";
20     final String delete = "DELETE FROM service WHERE id=?;";
21     final String update = "UPDATE service SET jenis=?, harga=?, status=? WHERE id=?;";
22
23     public ServiceRepo() {
24         connection = Database.koneksi();
25     }
26
27     @Override
28     public void save(Service service) {
29         PreparedStatement st = null;
30         try {
31             st = connection.prepareStatement(insert);
32             st.setString(1, service.getId());
33             st.setString(2, service.getJenis());
34             st.setString(3, service.getHarga());
35             st.setString(4, service.getStatus());
36             st.executeUpdate();
37         } catch (SQLException e) {
38             e.printStackTrace();
39         } finally {
40             try {
41                 if (st != null) st.close();
42             } catch (SQLException e) {
43                 e.printStackTrace();
44             }
45         }
46     }
47
48     @Override
49     public List<Service> show() {
50         List<Service> services = new ArrayList<>();
51         Statement st = null;
52         ResultSet rs = null;
53         try {
54             st = connection.createStatement();
55             rs = st.executeQuery(select);
56             while (rs.next()) {
57                 Service service = new Service();
58                 service.setId(rs.getString("id"));
59                 service.setJenis(rs.getString("jenis"));
60                 service.setHarga(rs.getString("harga"));
61                 service.setStatus(rs.getString("status"));
62                 services.add(service);
63             }
64         } catch (SQLException e) {
65             Logger.getLogger(ServiceDAO.class.getName()).log(Level.SEVERE, null, e);
66         } finally {
67             try {
68                 if (rs != null) rs.close();
69                 if (st != null) st.close();
70             } catch (SQLException e) {
71                 e.printStackTrace();
72             }
73         }
74         return services;
75     }
76
77     @Override
78     public void update(Service service) {
79         PreparedStatement st = null;
80         try {
81             st = connection.prepareStatement(update);
82             st.setString(1, service.getJenis());
83             st.setString(2, service.getHarga());
84             st.setString(3, service.getStatus());
85             st.setString(4, service.getId());
86         }
```

```

78  @Override
79  public void update(Service service) {
80      PreparedStatement st = null;
81      try {
82          st = connection.prepareStatement(update);
83          st.setString(1, service.getJenis());
84          st.setString(2, service.getHarga());
85          st.setString(3, service.getStatus());
86          st.setString(4, service.getId());
87          st.executeUpdate();
88      } catch (SQLException e) {
89          e.printStackTrace();
90      } finally {
91          try {
92              if (st != null) st.close();
93          } catch (SQLException e) {
94              e.printStackTrace();
95          }
96      }
97  }
98
99  @Override
100 public void delete(String id) {
101     PreparedStatement st = null;
102     try {
103         st = connection.prepareStatement(delete);
104         st.setString(1, id);
105         st.executeUpdate();
106     } catch (SQLException e) {
107         e.printStackTrace();
108     } finally {
109         try {
110             if (st != null) st.close();
111         } catch (SQLException e) {
112             e.printStackTrace();
113         }
114     }
115 }
116 }

```

### 3.Membuat class ServiceDAO dan code seperti ini :

```

1 package DAO;
2
3 import java.util.List;
4
5
6 public interface ServiceDAO {
7     void save(Service service);
8     List<Service> show();
9     void delete(String id);
10    void update(Service service);
11 }
12

```

### 4.Membuat JFrame untuk Service seperti ini :



1	cuci	10000.0	menunggu
---	------	---------	----------

**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
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change  Drop  More
<input type="checkbox"/>	2 nama	varchar(128)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	3 alamat	text	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	4 nohp	varchar(15)	utf8mb4_general_ci		No	None			Change  Drop  More

**12.Membuat class CostumerRepo dan codenya seperti ini :**

```

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();
            }
        }
    }
}

```

```

@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();
        }
    }
}
}
}

```

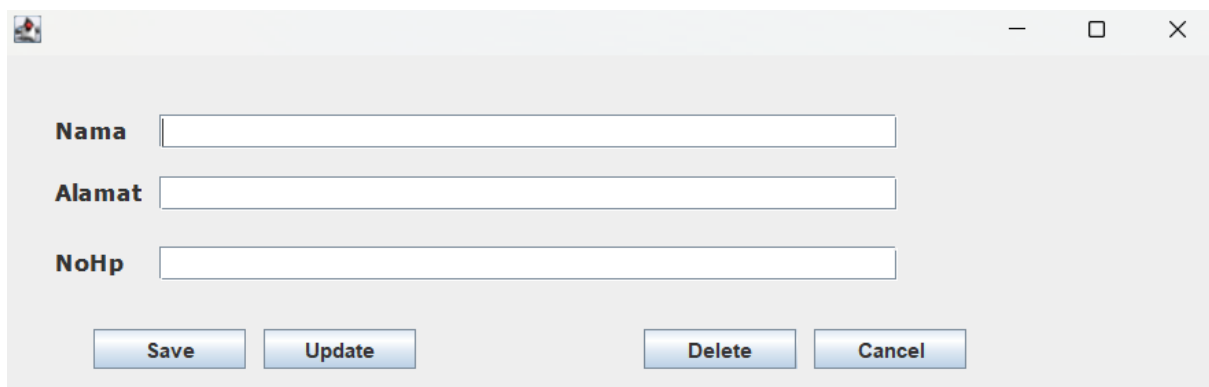
**13.Membuat class CostumerDAO dan code seperti ini :**

```

1 package DAO;
2
3 import java.util.List;
4
5
6 public interface CustomerDAO {
7     void save(Costumer costumer);
8     List<Costumer> show();
9     void update(Costumer costumer);
10    void delete(String id);
11 }
12

```

**14.Membuat JFrame untuk Costumer seperti ini :**



The screenshot shows a standard Java Swing window with a title bar containing a maximize button, a close button, and a window icon. The main content area has a light gray background. It features three text input fields stacked vertically, each preceded by a label: 'Nama', 'Alamat', and 'NoHp'. At the bottom of the window, there are four buttons arranged horizontally: 'Save', 'Update', 'Delete', and 'Cancel'.

**15.Membuat method reset pada JFrame seperti ini :**

```

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

```

**16.Membuat instance pada JFrame ServiceFrame**

```

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 :

<div> <div>← T →</div> <div>▼</div> </div>					id	jenis	harga	status
<div> <input type="checkbox"/> <input type="checkbox"/> Edit                 <input type="checkbox"/> Copy                 <input type="checkbox"/> Delete             </div>					1	cuci	10000	menunggu

# Membuat Fungsi CRUD untuk OrderDetailFrame

## 1. Membuat Class OrderDetailDAO

```
1 package DAO;
2
3 import java.util.List;
4
5
6 public interface OrderDetailDAO {
7     void save(OrderDetail orderDetail);
8     List<OrderDetail> findAll();
9     void update(OrderDetail orderDetail);
10    void delete(String idOrderDetail);
11    List<OrderDetail> show();
12 }
```

## 2. Membuat Class OrderDetailRepo

```
1 package DAO;
2
3 import java.sql.Connection;
4
5
6 public class OrderDetailRepo implements OrderDetailDAO {
7     private Connection connection;
8
9     final String insert = "INSERT INTO order_detail(id_order_detail, id_order, id_layanan, jumlah, total) VALUES(?, ?, ?, ?, ?)";
10    final String select = "SELECT * FROM order_detail";
11    final String delete = "DELETE FROM order_detail WHERE id_order_detail=?";
12    final String update = "UPDATE order_detail SET id_order=?, id_layanan=?, jumlah=?, total=? WHERE id_order_detail=?";
13
14
15    public OrderDetailRepo() {
16        connection = Database.koneksi();
17    }
18
19    @Override
20    public void save(OrderDetail orderDetail) {
21        PreparedStatement st = null;
22        try {
23            st = connection.prepareStatement(insert);
24            st.setString(1, orderDetail.getIdOrderDetail());
25            st.setString(2, orderDetail.getIdOrder());
26            st.setString(3, orderDetail.getIdLayanan());
27            st.setInt(4, orderDetail.getJumlah());
28            st.setDouble(5, orderDetail.getTotal());
29            st.executeUpdate();
30        } catch (SQLException e) {
31            e.printStackTrace();
32        } finally {
33            try {
34                if (st != null) st.close();
35            } catch (SQLException e) {
36                e.printStackTrace();
37            }
38        }
39    }
40
41    @Override
42    public List<OrderDetail> show() {
43        List<OrderDetail> orderDetails = new ArrayList<>();
44        Statement st = null;
45        ResultSet rs = null;
46        try {
47            st = connection.createStatement();
48            rs = st.executeQuery(select);
49            while (rs.next()) {
50                OrderDetail orderDetail = new OrderDetail();
51                orderDetail.setIdOrderDetail(rs.getString("id_order_detail"));
52                orderDetail.setIdOrder(rs.getString("id_order"));
53                orderDetail.setIdLayanan(rs.getString("id_layanan"));
54                orderDetail.setJumlah(rs.getInt("jumlah"));
55                orderDetail.setTotal(rs.getDouble("total"));
56                orderDetails.add(orderDetail);
57            }
58        } catch (SQLException e) {
59            Logger.getLogger(OrderDetailDAO.class.getName()).log(Level.SEVERE, null, e);
60        } finally {
61            try {
62                if (rs != null) rs.close();
63                if (st != null) st.close();
64            } catch (SQLException e) {
65                e.printStackTrace();
66            }
67        }
68        return orderDetails;
69    }
70
71    @Override
72    public void update(OrderDetail orderDetail) {
73        PreparedStatement st = null;
74        try {
75            st = connection.prepareStatement(update);
76            st.setString(1, orderDetail.getIdOrder());
77            st.setInt(2, orderDetail.getJumlah());
78            st.setDouble(3, orderDetail.getTotal());
79            st.setString(4, orderDetail.getIdOrderDetail());
80            st.executeUpdate();
81        } catch (SQLException e) {
82            e.printStackTrace();
83        } finally {
84            try {
85                if (st != null) st.close();
86            } catch (SQLException e) {
87                e.printStackTrace();
88            }
89        }
90    }
91 }
```

```

@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

```

1 package model;
2
3 public class OrderDetail {
4     private String idOrderDetail;
5     private String idOrder;
6     private int jumlah;
7     private double total;
8     private String idLayanan;
9
10    public String getIdLayanan() {
11        return idLayanan;
12    }
13
14    public void setIdLayanan(String idLayanan) {
15        this.idLayanan = idLayanan;
16    }
17
18    public String getIdOrderDetail() {
19        return idOrderDetail;
20    }
21
22    public void setIdOrderDetail(String idOrderDetail) {
23        this.idOrderDetail = idOrderDetail;
24    }
25
26    public String getIdOrder() {
27        return idOrder;
28    }
29
30    public void setIdOrder(String idOrder) {
31        this.idOrder = idOrder;
32    }
33
34    public int getJumlah() {
35        return jumlah;
36    }
37
38    public void setJumlah(int jumlah) {
39        this.jumlah = jumlah;
40    }
41
42    public double getTotal() {
43        return total;
44    }
45
46    public void setTotal(double total) {
47        this.total = total;
48    }
49 }
50

```

### 4. Membuat Class TableOrderDetail pada Package table

```

1 package table;
2
3 import javax.swing.table.AbstractTableModel;
4
5 public class TableOrderDetail extends AbstractTableModel {
6     List<OrderDetail> orderDetail;
7     private String[] columnNames = {"ID Order Detail", "ID Order", "ID Layanan", "Jumlah", "Total"};
8
9     public TableOrderDetail(List<OrderDetail> orderDetail) {
10        this.orderDetail = orderDetail;
11    }
12
13    @Override
14    public int getRowCount() {
15        return orderDetail.size();
16    }
17
18    @Override
19    public int getColumnCount() {
20        return columnNames.length;
21    }
22
23    @Override
24    public String getColumnName(int column) {
25        return columnNames[column];
26    }
27
28    @Override
29    public Object getValueAt(int rowIndex, int columnIndex) {
30        OrderDetail orderDetail = orderDetail.get(rowIndex);
31        switch (columnIndex) {
32            case 0:
33                return orderDetail.getIdOrderDetail();
34            case 1:
35                return orderDetail.getIdOrder();
36            case 2:
37                return orderDetail.getIdLayanan();
38            case 3:
39                return orderDetail.getJumlah();
40            case 4:
41                return orderDetail.getTotal();
42            default:
43                return null;
44        }
45    }
46 }
47

```

## 5. Membuat Class OrderDetailFrame di Package ui

Order ID

Pelanggan

Tanggal

Tanggal Pengembalian

Status

Total

Rp. 10.000

Pembayaran

Status Pembayaran

Layanan

HargaKg

Jumlah

Total

Simpan

Update

Hapus

Batal

Simpan

Simpan

## 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.setLayanan(textIdOrder.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 ts = new TableService(ls);
    tableLayanan.setModel(ts);
    tableLayanan.getTableHeader().setVisible(true);
}
```

13.Hasilnya Seperti ini :

Order ID

Pelanggan

Pilih

Tanggal

Tanggal Pengembalian

Status

Proses

Total

Rp. 10.000

Pembayaran

Cash

Status Pembayaran

Lunas

Simpan

Simpan

Layanan

ID	Jenis	Harga	Status
1	cuci	10000.0	menunggu
2	cuci	20000.0	menunggu

HargaKg

Jumlah

Total

Simpan

Update

Hapus

Batal

ID Order D...	ID Order	ID Layanan	Jumlah	Total
2	trx-001		2	20000.0
3			2	40000.0
4			2	20000.0
5	trx-002		5	100000.0
6	trx-003		2	20000.0
7	trx-005		2	20000.0
8	trx-003		3	30000.0
9	trx66	1	2	20000.0
10	trx666	2	2	40000.0
11	trx00001	2	2	40000.0

## 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 :

```
@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

```
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

```
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 btnDelete pada class OrderDetailFrame

```

        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

```

1 package DAO;
2
3 import java.util.List;
4
5
6 public interface OrderDAO {
7     void save(Order order);
8     List<Order> show();
9     void delete(String id);
10    void update(Order order);
11 }
12
```

#### 6. Membuat Class OrderRepo pada package DAO

```

30 import java.sql.Connection;
31
32 public class OrderRepo implements OrderDAO {
33     private Connection connection;
34
35     final String insert = "INSERT INTO `order` (id_order, nama, pembayaran, total, tanggal, tanggal_peng, status, status_pembayaran) VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
36     final String select = "SELECT * FROM `order`";
37     final String delete = "DELETE FROM `order` WHERE id_order=?";
38     final String update = "UPDATE `order` SET nama=?, pembayaran=?, total=?, tanggal=?, tanggal_peng=?, status=?, status_pembayaran=? WHERE id_order=?";
39
40     public OrderRepo() {
41         connection = Database.koneksi();
42     }
43
44     @Override
45     public void save(Order order) {
46         PreparedStatement st = null;
47         try {
48             System.out.println("Menyimpan Order: ");
49             System.out.println("ID: " + order.getId());
50             System.out.println("Nama Customer: " + order.getNama_costumer());
51             System.out.println("Pembayaran: " + order.getpembayaran());
52             System.out.println("Tanggal: " + order.getTanggal());
53             System.out.println("Tanggal Peng: " + order.getTanggal_selesai());
54             System.out.println("Status: " + order.getStatus());
55             System.out.println("Status Pembayaran: " + order.getStatus_pembayaran());
56             System.out.println("Total sebelum konversi: " + order.getTotal());
57
58             int total = Integer.parseInt(order.getTotal().replace("Rp. ", "").replace(".", "").trim());
59             System.out.println("Total setelah konversi: " + total);
60
61             st = connection.prepareStatement(insert);
62             st.setString(1, order.getId());
63             st.setString(2, order.getNama_costumer());
64             st.setString(3, order.getpembayaran());
65             st.setInt(4, total);
66             st.setString(5, order.getTanggal());
67             st.setString(6, order.getTanggal_selesai());
68             st.setString(7, order.getStatus());
69             st.setString(8, order.getStatus_pembayaran());
70             st.executeUpdate();
71             System.out.println("Order berhasil disimpan.");
72
73         } catch (SQLException e) {
74             e.printStackTrace();
75         } catch (NumberFormatException e) {
76             System.out.println("Format total tidak valid: " + order.getTotal());
77         } finally {
78             try {
79                 if (st != null) st.close();
80             } catch (SQLException e) {
81                 e.printStackTrace();
82             }
83         }
84     }
85 }
86
```



```

71
72● @Override
73 public List<Order> show() {
74     List<Order> orders = new ArrayList<>();
75     Statement st = null;
76     ResultSet rs = null;
77     try {
78         st = connection.createStatement();
79         rs = st.executeQuery(select);
80         while (rs.next()) {
81             Order order = new Order();
82             order.setId(rs.getString("id_order"));
83             order.setNama_costumer(rs.getString("nama"));
84             order.setTotal(rs.getString("total"));
85             order.setTanggal(rs.getString("tanggal"));
86             order.setTanggal_selesai(rs.getString("tanggal_peng"));
87             order.setStatus(rs.getString("status"));
88             order.setStatus_pembayaran(rs.getString("status_pembayaran"));
89             orders.add(order);
90         }
91     } catch (SQLException e) {
92         Logger.getLogger(OrderDAO.class.getName()).log(Level.SEVERE, null, e);
93     } finally {
94         try {
95             if (rs != null) rs.close();
96             if (st != null) st.close();
97         } catch (SQLException e) {
98             e.printStackTrace();
99         }
100     }
101 }
102 return orders;
103 }
104● @Override
105 public void update(Order order) {
106     PreparedStatement st = null;
107     try {
108         System.out.println("Memperbarui Order: ");
109         System.out.println("ID: " + order.getId());
110         System.out.println("Nama Customer: " + order.getNama_costumer());
111         System.out.println("Pembayaran: " + order.getpembayaran());
112         System.out.println("Tanggal: " + order.getTanggal());
113         System.out.println("Tanggal Peng: " + order.getTanggal_selesai());
114         System.out.println("Status: " + order.getStatus());
115         System.out.println("Status Pembayaran: " + order.getStatus_pembayaran());
116         System.out.println("Total sebelum update: " + order.getTotal());
117
118         // Konversi total
119         int total = Integer.parseInt(order.getTotal().replace("Rp. ", "").replace(".", "").trim());
120         System.out.println("Total setelah konversi: " + total);
121
122         st = connection.prepareStatement(update);
123         st.setString(1, order.getNama_costumer());
124         st.setString(2, order.getpembayaran());
125         st.setInt(3, total); // Total
126         st.setString(4, order.getTanggal());
127         st.setString(5, order.getTanggal_selesai());
128         st.setString(6, order.getStatus());
129         st.setString(7, order.getStatus_pembayaran());
130         st.setString(8, order.getId());
131         st.executeUpdate();
132         System.out.println("Order berhasil diperbarui.");
133
134     } catch (SQLException e) {
135         e.printStackTrace();
136     } catch (NumberFormatException e) {
137         System.out.println("Format total tidak valid: " + order.getTotal());
138     } finally {
139         try {
140             if (st != null) st.close();
141         } catch (SQLException e) {
142             e.printStackTrace();
143         }
144     }
145 }
146
147
148● @Override
149 public void delete(String id) {
150     PreparedStatement st = null;
151     try {
152         st = connection.prepareStatement(delete);
153         st.setString(1, id);
154         st.executeUpdate();
155         System.out.println("Order dengan ID " + id + " berhasil dihapus.");
156     } catch (SQLException e) {
157         e.printStackTrace();
158     } finally {

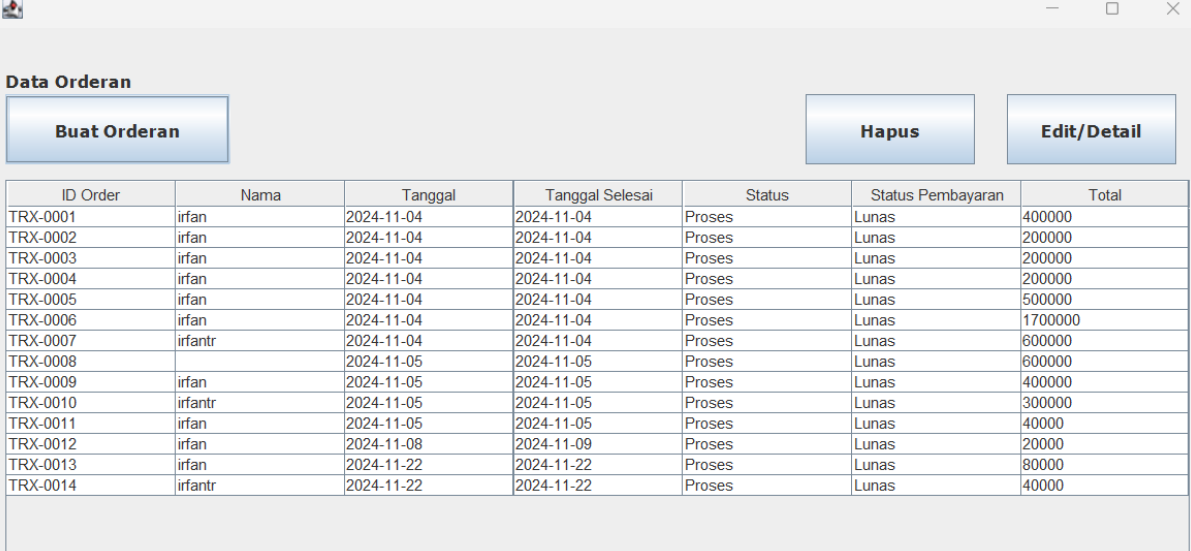
```

```

158     } finally {
159         try {
160             if (st != null) st.close();
161         } catch (SQLException e) {
162             e.printStackTrace();
163         }
164     }
165 }
166
167 public String getLastOrderId() {
168     String lastOrderId = null;
169     String query = "SELECT id_order FROM `order` ORDER BY id_order DESC LIMIT 1";
170
171     try (PreparedStatement statement = connection.prepareStatement(query);
172          ResultSet resultSet = statement.executeQuery()) {
173         if (resultSet.next()) {
174             lastOrderId = resultSet.getString("id_order");
175         }
176     } catch (SQLException e) {
177         e.printStackTrace();
178     }
179
180     return lastOrderId;
181 }
182
183 public String generateNewOrderId() {
184     String lastId = getLastOrderId();
185     if (lastId != null && lastId.startsWith("TRX-")) {
186         int nextId = Integer.parseInt(lastId.substring(4)) + 1;
187
188         public String generateNewOrderId() {
189             String lastId = getLastOrderId();
190             if (lastId != null && lastId.startsWith("TRX-")) {
191                 int nextId = Integer.parseInt(lastId.substring(4)) + 1;
192                 return String.format("TRX-%04d", nextId);
193             } else {
194                 return "TRX-0001";
195             }
196         }
197     }
198 }

```

## 7. Buat Class OrderFrame pada package ui



The screenshot shows a Java Swing window titled "Data Orderan". It contains three buttons at the top: "Buat Orderan", "Hapus", and "Edit/Detail". Below the buttons is a table with 7 columns: ID Order, Nama, Tanggal, Tanggal Selesai, Status, Status Pembayaran, and Total. The table contains 14 rows of order data.

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

```

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";  
    }  
}
```

Menambahkan 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 :



Tanggal

22/11/2024

Tanggal Pengembalian

22/11/2024

## 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());
}
```