

**MODUL PRAKTIKUM**  
**“PEMROGRAMAN BERORIENTASI OBJEK (PBO)”**  
**(DIF61119)**



Oleh  
DOSEN : NURFIAH, S.ST, M.KOM

**PROGRAM STUDI S1-INFORMATIKA**  
**FAKULTAS TEKNOLOGI INFORMASI**  
**UNIVERSITAS ANDALAS**  
**2024**

# PRAKTIKUM 1

## CLASS, OBJECT, CONSTRUCTOR dan METHOD

### 1.1 Tujuan

Tujuan praktikum ini yaitu mahasiswa mampu membuat class, object, encapsulation, constructor dan method menggunakan Bahasa java dan menerapkannya pada aplikasi manajemen laundry, Adapun pada praktikum ini mahasiswa akan mempelajari beberapa poin yaitu :

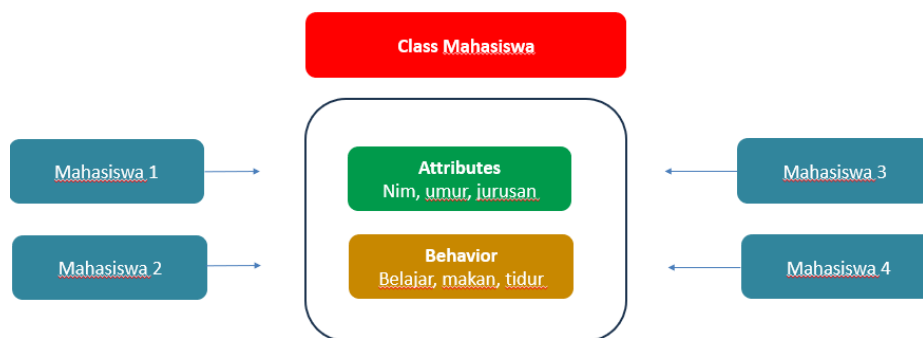
- Membuat class, object, encapsulation, constructor dan method untuk aplikasi laundry seperti class User, Customer, Service dan Order
- Membuat desain antarmuka aplikasi Laundry Login dan Halaman Utama
- Mampu menggunakan method pada JFrame/Tampilan aplikasi
- Mampu membuat fungsi berpindah dari tampilan login ke halaman utama

### 1.2 Alat

- Computer / laptop yang telah terinstall JDK dan Eclipse

### 1.3 Teori

**Class** merupakan sekumpulan objek yang memiliki karakteristik/sifat/behavior dan properties/atribut yang sama, class juga bisa disebut sebagai template/blueprint dari objek yang akan dibuat. Misalkan misalkan ada class mahasiswa, yaitu nama-nama mahasiswa misalkan fulan merupakan sebuah object.



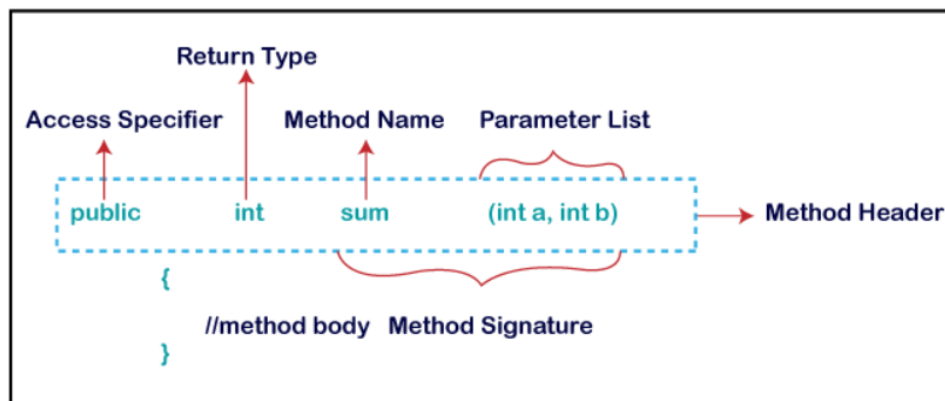
Sebuah class bukan sebuah entitas didalam dunia nyata melainkan sebagai blueprint untuk membuat suatu objek, class didalam java dapat berisi Data Member, Method, Constructor, Nested Class dan Interface.

**Object** merupakan representasi entitis yang ada pada dunia nyata, sebuah object terdiri dari :

- State merepresentasikan attribute dari sebuah object
- Behavior merepresentasikan method dari sebuah object
- Identity nama yang unik sebuah objek dan digunakan untuk interaksi dengan objek lain.



**Method** merupakan sebuah blok kode yang dapat dijalankan berulang kali, sehingga lebih terorganisir dan dapat digunakan Kembali, contoh method yang disediakan oleh java adalah method main(), equals(), toString() dll. Pola sebuah method dapat dilihat seperti gambar berikut :



Aturan pembuatan method pada java :

- Penamaan method harus menggunakan **verb** dan dimulai dengan **lowercase** letter
- Jika nama method lebih dari satu kata maka kata pertama harus **verb** dan selanjutnya **adjective** atau **noun** seperti `sum()`, `area()`
- jika nama method terdiri lebih dari 1 kata, maka kata pertama **lowercase** dan kata selanjutnya **uppercase** seperti `areaOfCircle()`

Method pada java terdiri dari beberapa jenis, berikut jenis-jenis method yang ada pada Java.

- **Predefined Method** (standard library method or built-in method) method yang sudah disediakan oleh java seperti `length()`, `equals()`, `compareTo()`, `print()`
- **User-defined Method** yaitu method yang dibuat oleh user atau Programmer, method dibuat sesuai dengan kebutuhan
- **Static method** yaitu Method yang menggunakan `static` sebelum nama method, keutamaan method `static` yaitu dapat memanggil/menggunakan method tanpa harus membuat object terlebih dahulu.

- Instance method merupakan method non static yaitu harus membuat object terlebih dahulu Ketika akan menggunakannya. Instance method terdiri dari 2 jenis yaitu accessor method dan mutator method.
  - Accessor Method digunakan untuk membaca instance, menggunakan kata kunci **get** atau disebut **getter**.

```
public String getNim() {
    return nim;
}
```

- Mutator Method digunakan untuk membaca dan mengubah nilai, menggunakan kata kunci **set** atau disebut juga dengan **setter** dan akan mengembalikan value yang bersifat private.

```
public void setNim(String nim) {
    this.nim = nim;
}
```

- Abstract Method yaitu method yang tidak memiliki body, dideklarasikan didalam class abstract dengna menggunakan kata kunci abstract.
- Factory Method yaitu method yang digunakan untuk mengembalikan object ke class yang bersangkutan, semua method static disebut factory method contohnya **NumberFormat**  
**obj = NumberFormat.getNumberInstance();**

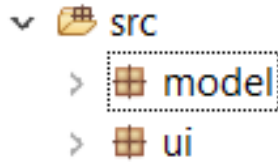
**Constructor** yaitu blok kode yang mirip dengan method, menggunakan kata kunci new Ketika akan membuat sebuah object, nama constructor harus sama dengan nama class.

```
public Costumer(String id, String nama, String alamat, String hp) {
    super();
    this.id = id;
    this.nama = nama;
    this.alamat = alamat;
    this.hp = hp;
}
```

## 1.4 Langkah-langkah

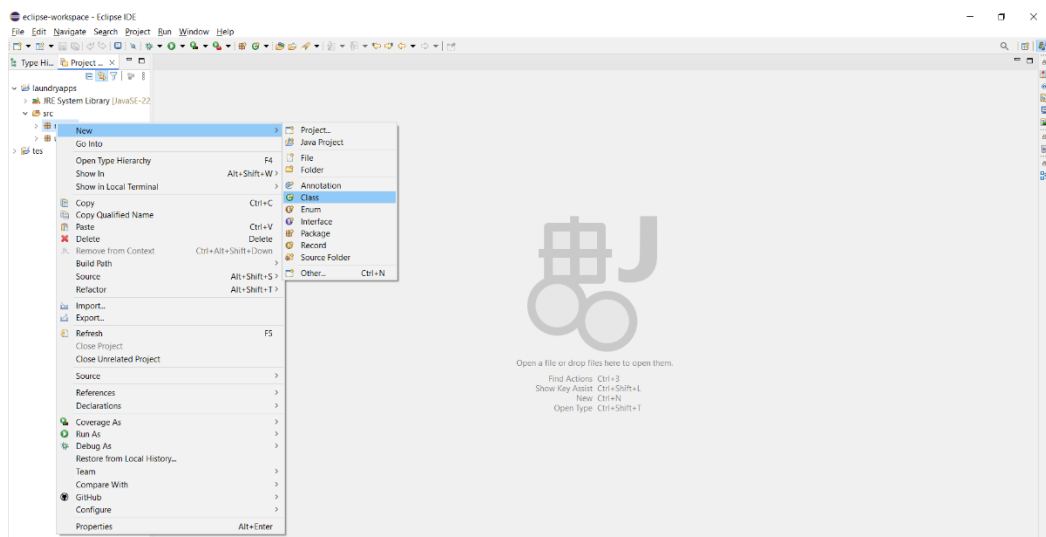
### Membuat Class User

- Buat project baru pada eclipse dengna nama **laundryapps**
- Selanjutnya buat 2 buah package pada directory src dengan nama **model** dan **ui**



Package **model** berfungsi untuk menyimpan class-class yang diperlukan untuk pembuatan aplikasi laundry, sedangkan package **ui** digunakan untuk menyimpan tampilan atau antar muka aplikasi laundry.

- Selanjutnya buat class baru pada package model dengan nama **User**.



- Membuat attribute class user yaitu id, nama, username dan password

```
String id, nama, username, password;
```

- Membuat **setter** dan **getter** attribute user, method ini berfungsi untuk memasukkan/mengubah dan menampilkan value dari suatu object.

```
public String getId() {
    return id;
}

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

public String getNama() {
    return nama;
}

public void setNama(String nama) {
    this.nama = nama;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}
```

- Membuat 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;
}
```

Method login ini merupakan method static dikarenakan menggunakan kata kunci static yang mana dapat diakses dari class lain tanpa harus membuat sebuah object terlebih

dahulu. Method login ini menggunakan tipe data Boolean yang mana akan mengembalikan nilai Boolean Ketika digunakan, kemudian memiliki 2 buah parameter yaitu username dan password. Pertama akan variable isLogin diberikan nilai false, kemudian dibuatkan sebuah object user baru, selanjutnya akan dilakukan pencocokan antara data username dan password pada object dengan data username dan password yang dikirimkan dari class lain, jika cocok maka isLogin akan diubah menjadi true dan terakhir method akan mengembalikan nilai isLogin.

Source code lengkap class User seperti gambar dibawah ini :

```

package model;

public class User {
    String id, nama, username, password;

    public String getId() {
        return id;
    }

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

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    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");

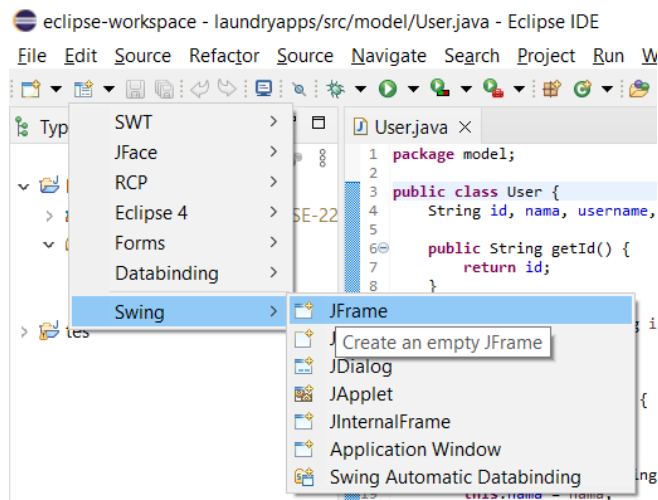
        if(user.getUsername().equalsIgnoreCase(username)
            && user.getPassword().equalsIgnoreCase(password)) {
            isLoggin = true;
        }else {
            isLoggin = false;
        }
        return isLoggin;
    }
}

```

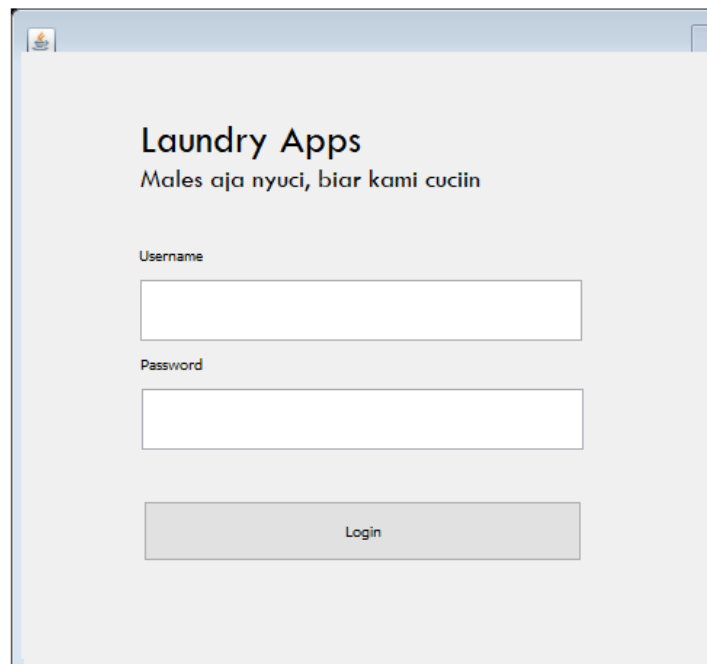


## Membuat Tampilan Login menggunakan JFrame

- buat JFrame baru pada package ui dengan nama LoginFrame



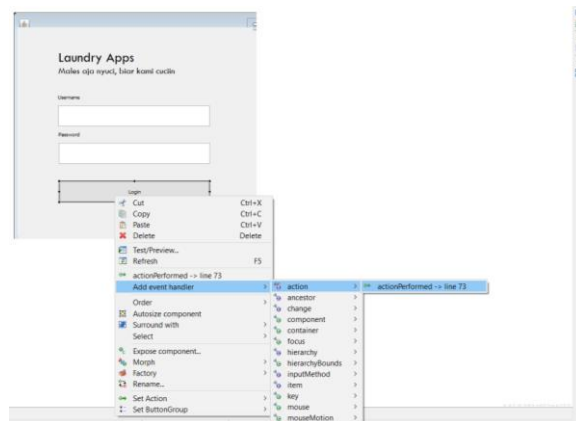
- kemudian desain tampilan login seperti gambar dibawah ini.



Ubah id JTextField username, password dan JButton seperti table dibawah ini.

Component	Id	Keterangan
JTextField	txtUsername	Username
JTextField	txtPassword	Password
JButton	btnLogin	Login

- selanjutnya klik kanan pada button login, pilih add event handler → action → actionPerformed

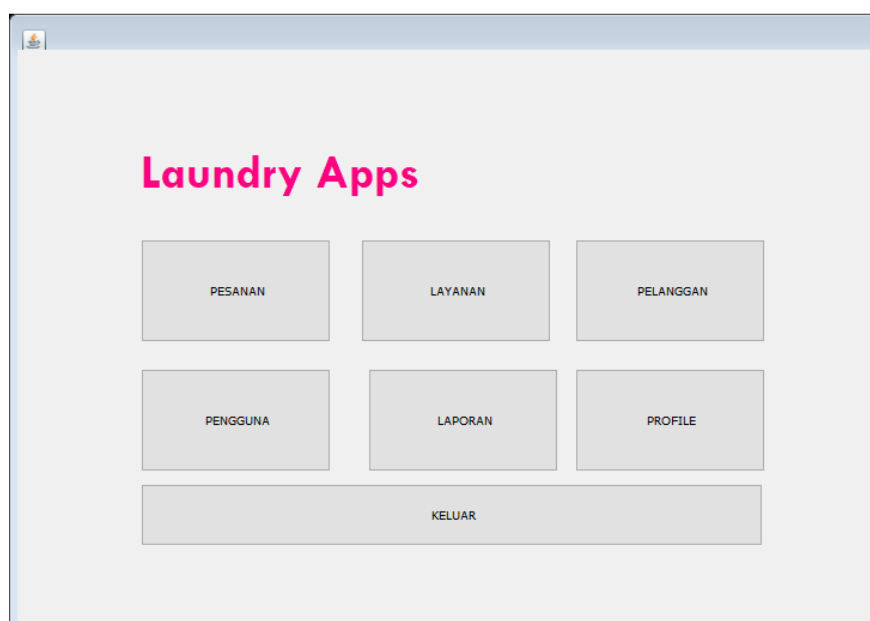


Kemudian panggil method login pada class User dengan mengirimkan parameter yang diambil dari txtUsername dan txtPassword, jika cocok maka akan tampil halaman utama.

```
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");
        }
    }
});
```

Membuat Tampilan Halaman Utama dengan JFrame

- buat sebuah JFrame baru pada package ui dengan nama MainFrame
- selanjutnya desain halaman utama seperti gambar berikut :



## 1.5 Latihan /Tugas

- Buatlah class dengan nama Costumer dengan attribute id, nama, Alamat dan nomor hp, buatlah setter dan getter untuk menambahkan object Costumer
- Buatlah class dengan nama Service dengan attribute id, jenis, harga dan status, buatlah setter dan getter untuk menambahkan object Service
- Buatlah class dengan nama Order dengan attribute id, id\_costumer, id\_service, id\_user, total, tanggal, tanggal\_selesai, status, status\_pembayaran , buatlah setter dan getter untuk menambahkan object Order

# PRAKTIKUM 2 dan 3

## MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete) USER DENGAN DATABASE MYSQL

### 1.1 Tujuan

Tujuan praktikum ini yaitu mahasiswa mampu membuat fungsi CRUD data user menggunakan database MySQL, Adapun poin-poin praktikum yaitu :

- Mahasiswa mampu membuat table user pada database MySQL
- Mahasiswa mampu membuat koneksi Java dengan database MySQL
- Mahasiswa mampu membuat tampilan GUI CRUD user
- Mahasiswa mampu membuat dan mengimplementasikan interface
- Mahasiswa mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikannya.
- Mahasiswa mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek

### 1.2 Alat

- Computer / laptop yang telah terinstall JDK dan Eclipse
- MySQL / XAMPP
- MySQL connector atau Connector/J

### 1.3 Teori

**XAMPP** yaitu paket software yang terdiri dari Apache HTTP Server, MySQL, PHP dan Perl yang bersifat open source, xampp biasanya digunakan sebagai development environment dalam pengembangan aplikasi berbasis web secara localhost.

Apache berfungsi sebagai web server yang digunakan untuk menjalankan halaman web, MySQL digunakan untuk manajemen basis data dalam melakukan manipulasi data, PHP digunakan sebagai Bahasa pemrograman untuk membuat aplikasi berbasis web.

**MySQL** adalah sebuah relational database management system (RDBMS) open-source yang digunakan dalam pengelolaan database suatu aplikasi, MySQL ini dapat digunakan untuk menyimpan, mengelola dan mengambil data dalam format table.



Logo MySQL

**MySQL Connection/j** adalah driver yang digunakan untuk menghubungkan aplikasi berbasis java dengan database MySQL sehingga dapat berinteraksi seperti menyimpan, mengubah, mengambil dan menghapus data. Beberapa fungsi MySQL connector yaitu :

- Membuka koneksi ke database MySQL
- Mengirimkan permintaan SQL ke server MySQL
- Menerima hasil dari permintaan SQL
- Menutup koneksi ke database MySQL

**DAO (Data Access Object)** merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update). Tujuan penggunaan DAO yaitu :

- Meningkatkan modularitas yaitu memisahkan logika akses data dengan logika bisnis sehingga memudahkan untuk dikelola
- Meningkatkan reusabilitas yaitu DAO dapat digunakan Kembali
- Perubahan pada logika akses data dapat dilakukan tanpa mempengaruhi logika bisnis.

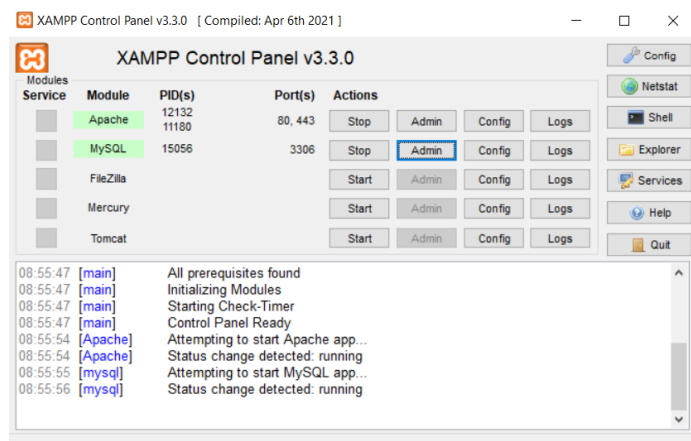
**Interface** dalam Bahasa java yaitu mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

**CRUD** (Create, Read, Update, Delete) merupakan fungsi dasar atau umum yang ada pada sebuah aplikasi yang mana fungsi ini dapat membuat, membaca, mengubah dan menghapus suatu data pada database aplikasi.

## 1.4 Langkah-langkah

### Install XAMPP

- Download XAMPP dari pada link berikut : <https://www.apachefriends.org/>
- Setelah didownload install xampp pada computer/laptop masing-masing
- Jalankan xampp dan aktifkan apache dan mysql



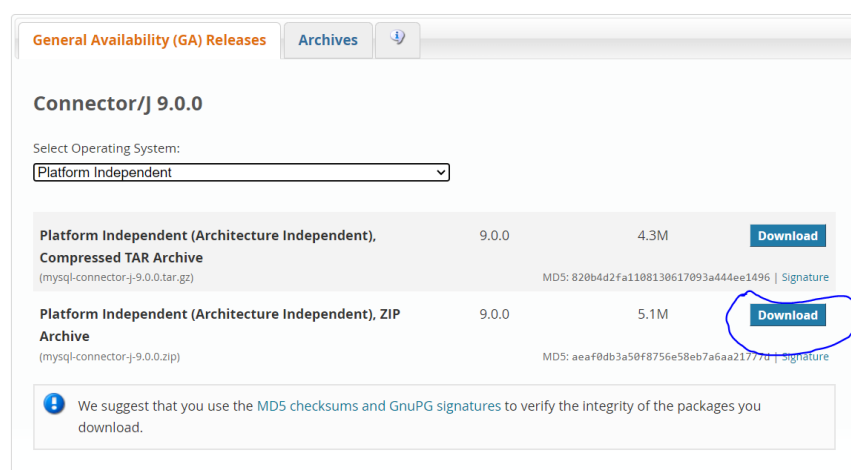
### Menambahkan MySQL Connector

Aplikasi java agar dapat terhubung dengan database MySQL membutuhkan sebuah driver yaitu MySQL Connection, berikut ini Langkah-langkah membuat koneksi Database MySQL.

- Download MySQL connection pada link berikut  
<https://dev.mysql.com/downloads/connector/j/>

#### MySQL Community Downloads

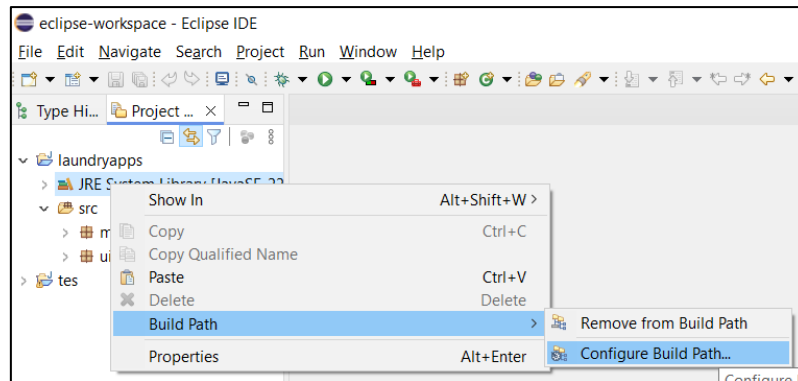
Connector/J



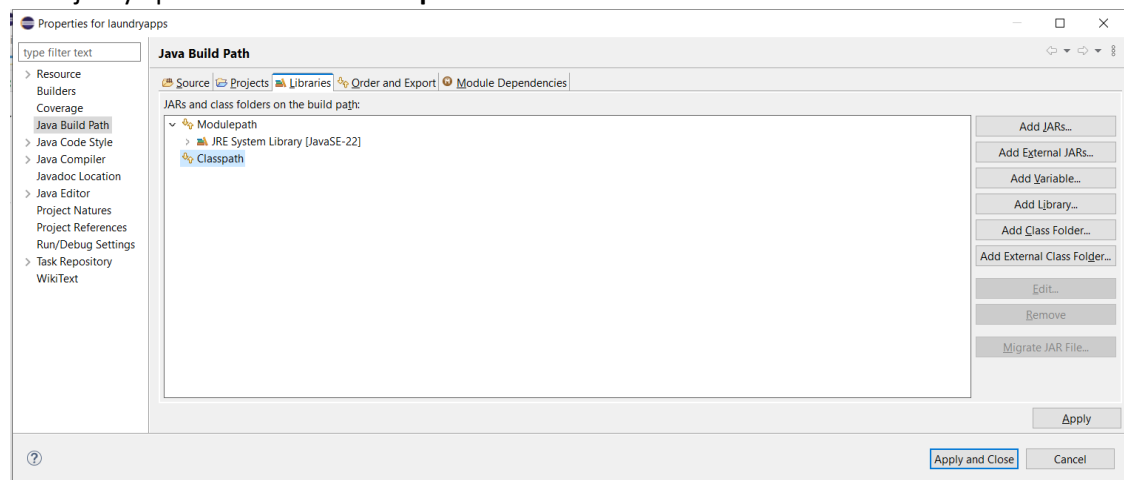
Pilih file yang berekstensi .zip

- Menambahkan MySQL Connector kedalam project dengan cara klik kanan directory **JRE**

**System Library → Built Path → Configure Build Path**

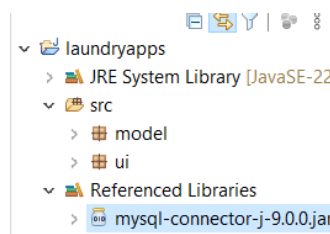


Selanjutnya pilih **Libraries → Classpath**



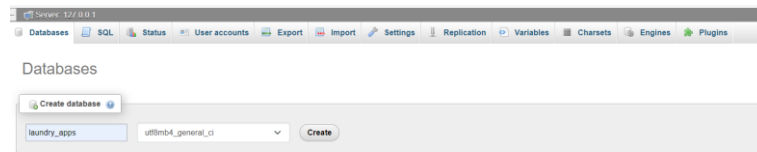
Tambahkan file MySQL Connector dengan cara klik **Add External JARs** dan pilih file yang telah didownload dan pilih **Apply and Close**.

Jika berhasil menambahkan MySQL Connector maka akan generate folder Referenced Libraries pada project yang berisi MySQL Connector.

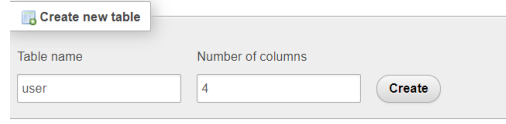


## Membuat Database dan Table User

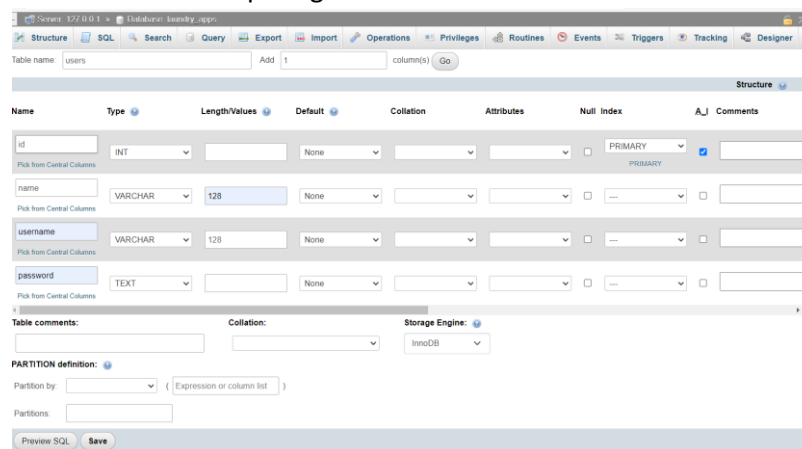
- Buka <http://localhost/phpmyadmin>
- Klik new dan buat database dengan nama laundry\_apps



- Buat table user dengan cara klik database laundry\_apps dan buat table dengan nama user



Klik create maka akan muncul seperti gambar dibawah ini.



Isi seperti gambar diatas dan klik save.

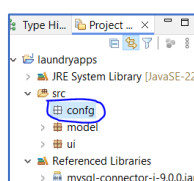
- Cara lain membuat table user pada database laundry\_apps, klik SQL pada phpMyAdmin dan ketikan sql seperti gambar dibawah ini.

```
CREATE TABLE `user` (
  `id` int(11) NOT NULL,
  `name` varchar(128) NOT NULL,
  `username` varchar(64) NOT NULL,
  `password` text NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

## Membuat Koneksi ke Database MySQL

Setelah berhasil menambahkan MySQL Connector maka dapat membuat koneksi ke database MySQL, berikut Langkah-langkahnya.

- Buat package baru dengan nama config, package ini yang akan digunakan untuk membuat konfigurasi aplikasi yang akan dibuat termasuk dengan konfigurasi database.





- Buat class baru dengan nama Database, kemudian konfigurasi sesuai dengan kode program berikut.

```

1 package config;
2
3 import java.sql.*;
4 import javax.swing.JOptionPane;
5
6 public class Database {
7     Connection conn;
8     public static Connection koneksi() {
9         try {
10             Class.forName("com.mysql.cj.jdbc.Driver");
11             Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/laundry_apps",
12                 "root", "");
13             return conn;
14         } catch (Exception e) {
15             JOptionPane.showMessageDialog(null, e);
16             return null;
17         }
18     }
19 }

```

Penjelasan :

- Import java.sql.\* digunakan untuk import seluruh fungsi-fungsi SQL
- Line 8 membuka method Connection dengan nama koneksi, yang mana method ini akan digunakan untuk membuka koneksi ke database
- Line 10-13 membuat koneksi database, jika koneksi berhasil maka akan mengembalikan nilai Connection
- Line 15-16 jika koneksi gagal maka akan ditampilkan pesan error menggunakan JOptionPane.

#### Membuat Tampilan CRUD User

- Buat file baru menggunakan JFrame pada package ui dengan nama UserFrame seperti gambar berikut ini.

Keterangan :

Component	Variable	Keterangan
TextField	txtName	Name
TextField	txtUsername	Username
TextField	txtPassword	Password
Button	btnSave	Save
Button	btnUpdate	Update
Button	btnDelete	Delete
Button	btnCancel	Cancel
JTable	tableUsers	Table Users

### Membuat Table Model

Table model user ini berguna untuk mengambil data dari database dan ditampilkan kedalam table.

- Buat package baru dengan nama **table**
- Buat file baru didalam package table dengan nama **TableUser**, kemudian isikan dengan kode program berikut.

```
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() {
        // TODO Auto-generated method stub
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 4;
    }

    @Override
    public String getColumnName(int column) {
        // TODO Auto-generated method stub
        return columnNames[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch (columnIndex) {
            case 0:
                return ls.get(rowIndex).getId();
            case 1:
                return ls.get(rowIndex).getNama();
            case 2:
                return ls.get(rowIndex).getUsername();
            case 3:
                return ls.get(rowIndex).getPassword();
            default:
                return null;
        }
    }
}
```

### Membuat Fungsi DAO

- Buat package baru dengan nama DAO
- Buat class Interface baru dengan nama **UserDAO**, kemudian isikan dengan kode program berikut.

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

Terdapat method **save**, **show**, **delete** dan **update**. Method pada class interface digunakan sebagai method utama yang wajib diimplementasikan pada class yang menggunakannya.

### Menggunakan Fungsi DAO

- Buat class baru pada package DAO dengan nama **UserRepo** yang mana akan digunakan untuk mengimplementasikan DAO yang telah dibuat.
- Implementasikan UserDao dengan kata kunci **implements**

```
public class UserRepo implements UserDao{
```

- Membuat instanisasi Connection, membuat constructor dan membuat String untuk melakukan manipulas database.

```
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();  
}
```

- Membuat method **save**, isikan dengan kode program berikut.

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

- Membuat method **show** untuk mengambil data dari database

```
@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;
}
```

- Membuat method **update** yang digunakan untuk mengubah data.

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

- Membuat method **delete** yang digunakan untuk menghapus data.

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

## 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("");  
}
```

- Membuat instance pada UserFrame

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

## CREATE USER

- 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();
```

## READ USERS

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

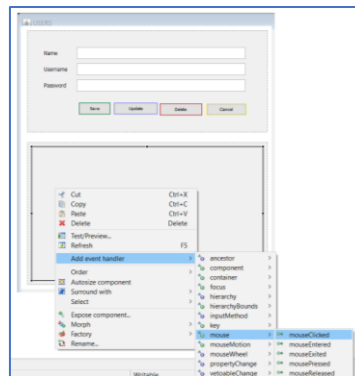
```
public void loadTable() {  
    ls = usr.show();  
    TableUser tu = new TableUser(ls);  
    tableUsers.setModel(tu);  
    tableUsers.getTableHeader().setVisible(true);  
}
```

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

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

## UPDATE USER

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

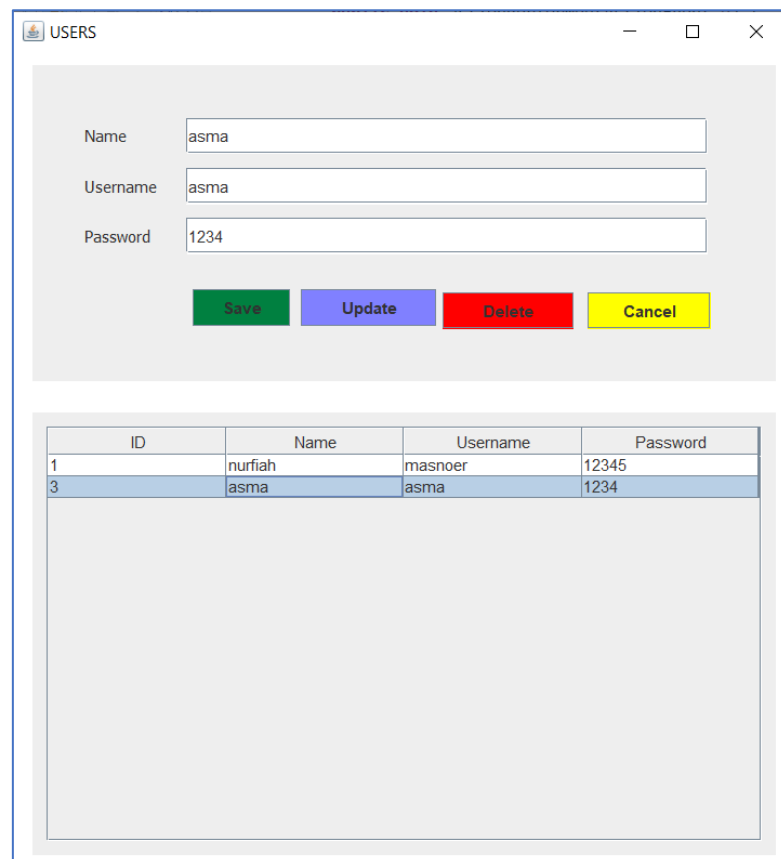


Isikan dengan kode program dibawah ini.

```
id = tableUsers.getValueAt(tableUsers.getSelectedRow(),0).toString();
txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),1).toString());
txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),2).toString());
txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(),3).toString());
```

Kode program diatas berfungsi yaitu mengambil id user dan menyimpannya kedalam variable **id** kemudian mengambil data nama, username dan password dan ditampilkan kedalam form inputan.

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



ID	Name	Username	Password
1	nurfiah	masnoer	12345
3	asma	asma	1234

- 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();
```

#### DELETE USER

- Klik salah satu data pada JTable
- 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");
}
```

#### 1.5 Latihan /Tugas

- Buat fungsi CRUD untuk layanan dan pelanggan



## PRAKTIKUM 4 dan 5

### MEMBUAT FUNGSI ORDER LAUNDRY

#### 1.1 Tujuan

Tujuan praktikum ini yaitu mahasiswa mampu membuat desain interface dan fungsi untuk melakukan order laundry.

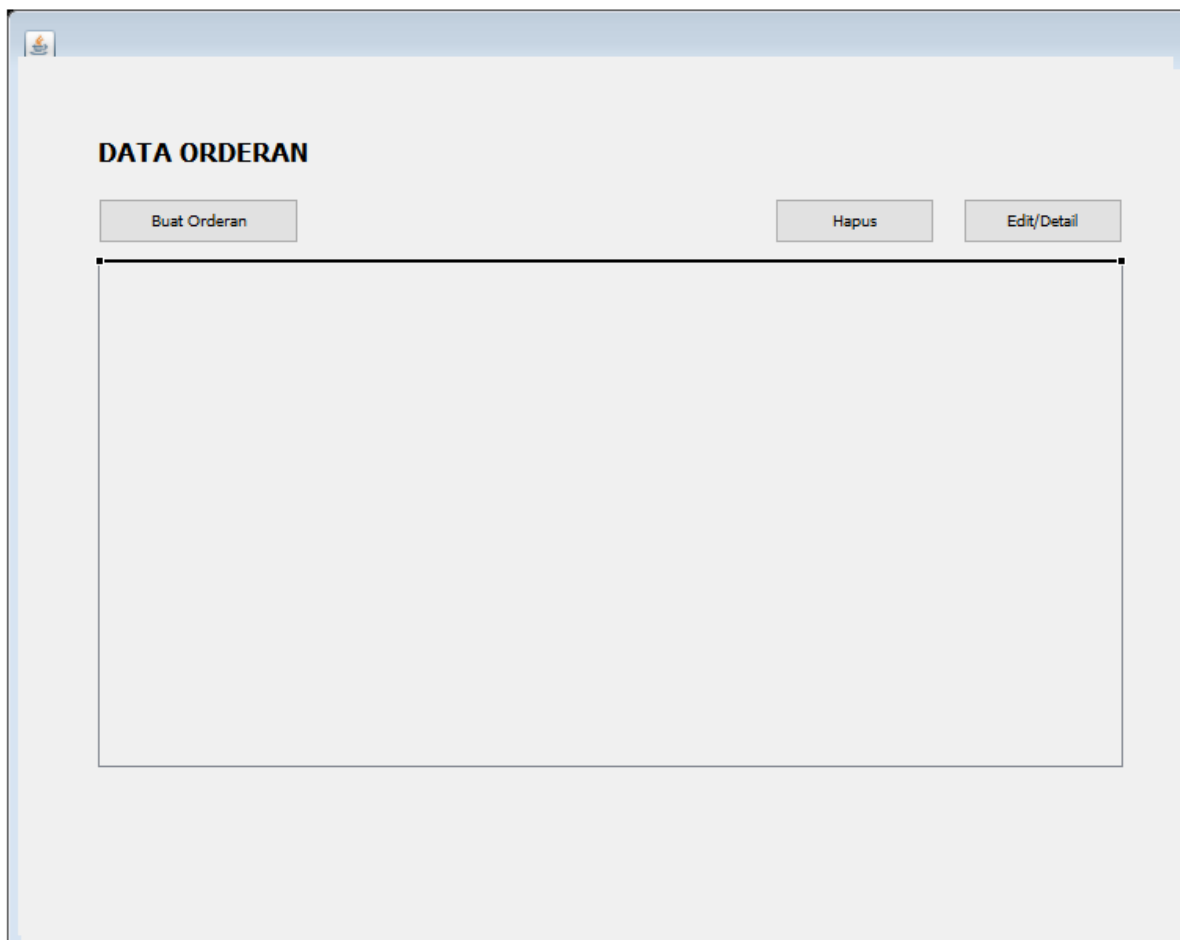
- Mahasiswa mampu membuat UI untuk melakukan order laundry
- Mahasiswa mampu membuat fungsi pemesanan multi layanan laundry baik menambahkan, mengubah dan menghapusnya
- Mahasiswa mampu membuat fungsi pemesanan laundry

#### 1.2 Alat

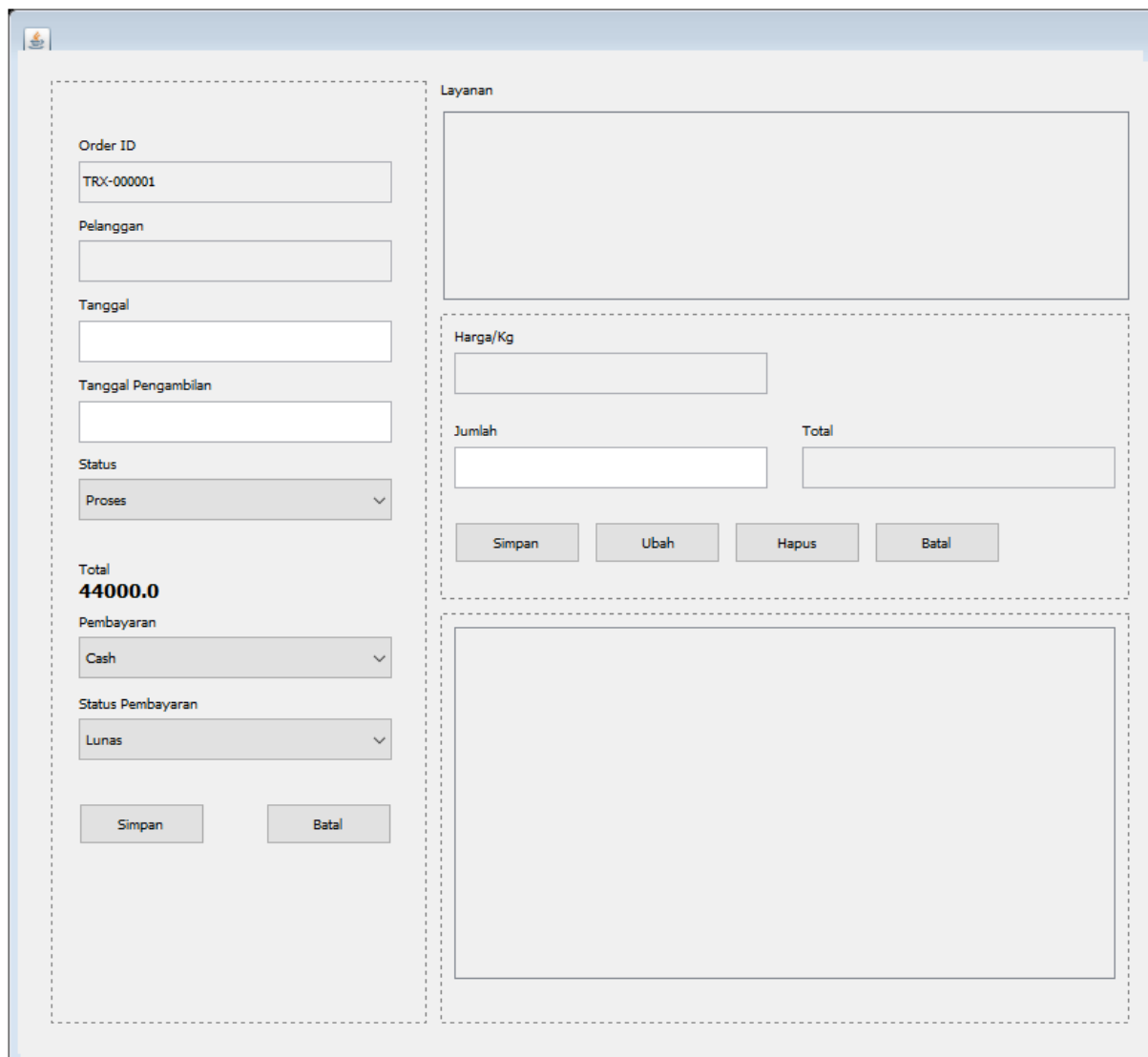
- Computer / laptop yang telah terinstall JDK dan Eclipse

#### 1.3 Teori

Berikut ini rancangan tampilan interface order laundry



Desain UI Order



The image shows a UI design for an 'Order Detail' form. It is divided into two main sections by a dashed line. The left section contains input fields for 'Order ID' (with value 'TRX-000001'), 'Pelanggan', 'Tanggal', 'Tanggal Pengambilan', 'Status' (dropdown with 'Proses'), 'Total' (displaying '44000.0'), 'Pembayaran' (dropdown with 'Cash'), and 'Status Pembayaran' (dropdown with 'Lunas'). At the bottom of this section are 'Simpan' and 'Batal' buttons. The right section is titled 'Layanan' and contains a large empty box. Below this is a dashed box containing 'Harga/Kg' (input), 'Jumlah' (input), and 'Total' (input). Below these are 'Simpan', 'Ubah', 'Hapus', and 'Batal' buttons. At the bottom of the right section is another large empty box.

Desain UI Order Detail

Penjelasan :

- Pertama-tama akan tampil UI Order yang mana pada tampilan ini akan ditampilkan data orderan laundry, tombol untuk menambah orderan, menghapus dan melakukan perubahan data order.
- Pada UI Order Detail akan ditampilkan layanan-layanan yang tersedia pada aplikasi laundry misalkan cuci, cuci + setrika, setrika dll
- Operator akan memilih jenis layanan kemudian secara otomatis pada form harga/kg akan terisi dengan harga sesuai dengan layanan yang dipilih.
- Operator akan menginputkan jumlah dari laundry yang dipesan maka secara otomatis total akan terisi dengan hasil perkalian harga dengan jumlah.

- Selanjutnya operator akan klik tombol simpan untuk menyimpan pesanan, jika pelanggan akan memesan layanan lain maka dapat dilakukan dengan cara yang sama diatas.
- Jika ada perubahan terhadap pesanan maka operator dapat memilik pesanan maka otomatis terisi kolom harga, jumlah dan total
- Untuk melakukan perubahan klik ubah, untuk menghapus klik hapus dan jika ingin membatalkan klik batal.
- Selanjutnya jika pesanan sudah selesai, operator akan memilih nama pelanggan, tanggal pemesanan, tanggal pengambilan, status, jenis pembayaran dan status pembayaran kemudian klik simpan untuk menyimpan pemesanan laundry.

#### 1.4 Langkah-langkah

Buat Desain UI

Langkah pertama silahkan buat frame baru dengan nama OrderDetail Frame dan buat tampilan seperti berikut :

The image shows a UI design for an 'OrderDetail' frame. It is divided into two main sections: a form on the left and a table on the right.

**Form Section (Left):**

- Order ID:** A text input field containing 'TRX-000001'.
- Pelanggan:** A text input field.
- Tanggal:** A text input field.
- Tanggal Pengambilan:** A text input field.
- Status:** A dropdown menu with 'Proses' selected.
- Total:** A text input field displaying '44000.0'.
- Pembayaran:** A dropdown menu with 'Cash' selected.
- Status Pembayaran:** A dropdown menu with 'Lunas' selected.
- Buttons:** 'Simpan' and 'Batal' buttons at the bottom.


**Table Section (Right):**

- Table 1 (Top):** A table with the word 'TABLE' in red text, representing a list of services.
- Form Section (Middle):** A dashed box containing:
  - Harga/Kg:** A text input field.
  - Jumlah:** A text input field.
  - Total:** A text input field.
  - Buttons:** 'Simpan', 'Ubah', 'Hapus', and 'Batal' buttons.
- Table 2 (Bottom):** A table with the word 'TABLE' in red text, representing a list of items or details.

Komponen	Variabel	Keterangan
TextField	txtTrx	Order ID
TextField	txtPelanggan	Pelanggan
TextField	txtTanggal	Tanggal
TextField	txtTanggalPengambilan	Tanggal Pengambilan
ComboBox	cbxStatus	Status
Label	txtTotalOrder	Total Order
ComboBox	cbxPembayaran	Pembayaran
ComboBox	cbxStatusPembayaran	Status Pembayaran
Button	btnSimpanOrder	Simpan Order
Button	btnBatalOrder	Batal Order
Table	tableService	Table Layanan
TextField	txtHarga	Harga/Kg
TextField	txtJumlah	Jumlah
TextField	txtTotal	Total
Button	btnSimpanDetail	Simpan Order Per Layanan
Button	btnUbahDetail	Ubah Order Per Layanan
Button	btnHapusDetail	Hapus Order Per Layanan
Button	btnBatalDetail	Batal Order Per Layanan
Table	tableOrderDetail	Table Detail Order

### Membuat Table Detail Order

Langkah selanjutnya yaitu menambahkan table untuk menyimpan detail order, silahkan buat table baru dengan nama order detail dan struktur table seperti gambar dibawah ini :

#	Name	Type	Collation
<input type="checkbox"/> 1	<b>id</b> 	int(11)	
<input type="checkbox"/> 2	<b>order_id</b>	varchar(32)	utf8mb4_general_ci
<input type="checkbox"/> 3	<b>service_id</b>	int(11)	
<input type="checkbox"/> 4	<b>harga</b>	double	
<input type="checkbox"/> 5	<b>jumlah</b>	double	
<input type="checkbox"/> 6	<b>total</b>	double	
<input type="checkbox"/> 7	<b>status</b>	enum('draft', 'finish')	utf8mb4_general_ci

### Menampilkan Daftar Layanan

- Tambahkan kode program berikut pada Source **OrderDetailFrame**

```
ServiceRepo sr = new ServiceRepo();  
List<Service> ls_service;  
public String id_service;  
public static String txt_pelanggan="";
```

- Buat method untuk menampilkan data layanan

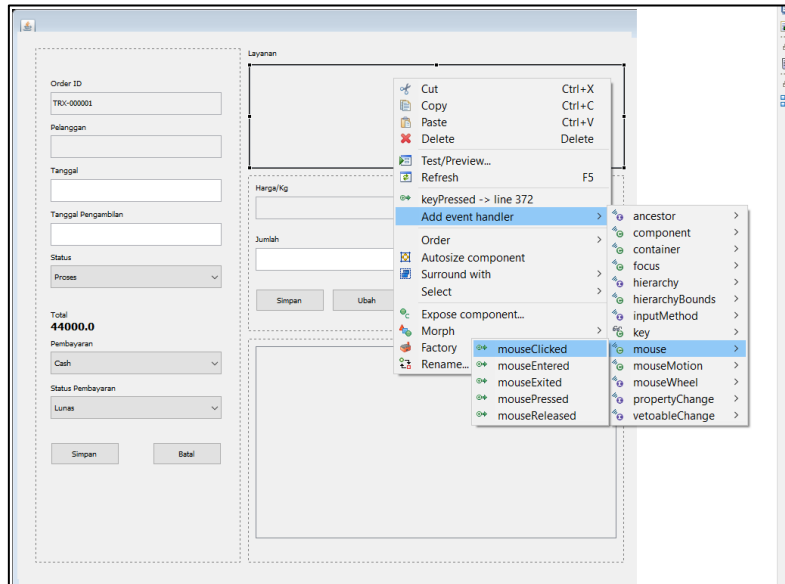
```
// load table service  
public void loadTableService() {  
    ls_service = sr.show();  
    TableService tu = new TableService(ls_service);  
    tableService.setModel(tu);  
    tableService.getTableHeader().setVisible(true);  
}
```

- Selanjutnya panggil method **loadTableService** kedalam main Frame agar data layanan muncul kedalam table Ketika frame di load.

```
public static void main(String[] args) {  
    EventQueue.invokeLater(new Runnable() {  
        public void run() {  
            try {  
                OrderDetailFrame frame = new OrderDetailFrame();  
                frame.setVisible(true);  
                frame.loadTableService();  
                frame.loadTableDetail();  
            } catch (Exception e) {  
                e.printStackTrace();  
            }  
        }  
    });  
}
```

Menampilkan Harga Ketika layanan dipilih

Selanjutnya agar form harga terisi otomatis Ketika layanan dipilih, klik kanan pada table **service** → **Add event handler** → **mouse** → **mouseClicked**



Kemudian tambahkan kode program berikut :

```
tableService.addMouseListener(new MouseAdapter() {  
    @Override  
    public void mouseClicked(MouseEvent e) {  
        id_service = tableService.getValueAt(tableService.getSelectedRow(),0).toString();  
        txtHarga.setText(tableService.getValueAt(tableService.getSelectedRow(),3).toString());  
  
        if(!txtJumlah.getText().isEmpty()) {  
            txtTotal.setText(""+total(txtJumlah.getText()));  
        }  
    }  
});
```

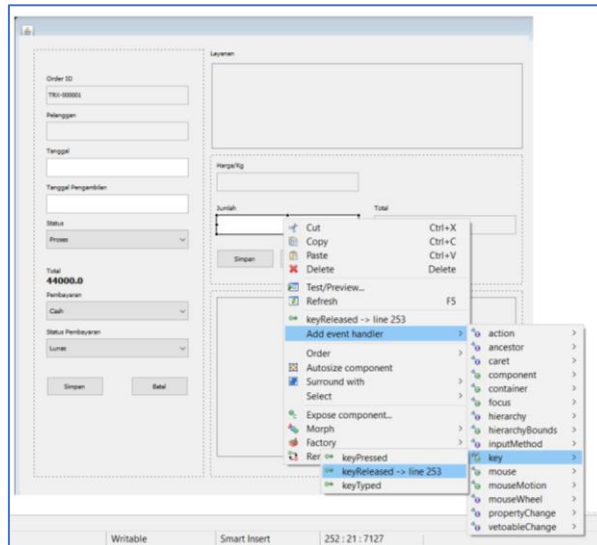
Membuat Method Hitung Total

Ketika form harga sudah terisi, maka Ketika operator memasukkan jumlah maka form total akan terisi secara otomatis, berikut ini Langkah-langkahnya.

- Buat method untuk menghitung total seperti tampilan berikut :

```
public double total(String jumlah) {  
    double result =0;  
    if(jumlah.isEmpty()) {  
        result=0;  
    }else {  
        result = Double.parseDouble(jumlah) * Double.parseDouble(txtHarga.getText());  
    }  
    return result;  
}
```

- Klik kanan **TextField jumlah** → **Add event handler** → **key** → **keyReleased**



- Kemudian tambahkan kode program berikut.

```
txtJumlah.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        String value_jumlah = txtJumlah.getText().toString();
        txtTotal.setText(""+total(value_jumlah));
    }
});
```

Membuat method reset

Method ini berfungsi akan mereset inputan menjadi kosong Ketika order detail berhasil disimpan, tambahkan kode program berikut :

```
public void reset() {
    txtHarga.setText("");
    txtJumlah.setText("");
    txtTotal.setText("");
    id_service=null;
    id_order_detail=null;
}
```

Fungsi CRUD Order Detail

- Tambahkan file baru pada package **dao** dengan nama **OrderDetailDao** kemudian tambahkan kode program berikut :

```
public interface OrderDetailDao {
    void save(OrderDetail od);
    public List<OrderDetail> show(String order_id);
    public void delete(String id);
    public void update(OrderDetail od);
    public String total(String order_id);
}
```

Pada potongan kode program diatas terdapat 5 buah method yaitu :

- Save digunakan untuk menyimpan data order detail dengan Argument Model OrderDetail
  - Show digunakan untuk menampilkan data OrderDetail dengan argument/parameter order\_id
  - Delete digunakan untuk menghapus data
  - Update digunakan untuk melakukan perubahan terhadap data
  - Total digunakan untuk menghitung jumlah total order detail berdasarkan order\_id
- Tambahkan file baru pada package **model** dengan nama **OrderDetail** kemudian tambahkan kode program berikut.

```
public class OrderDetail {
    String id, order_id, service_id, harga, jumlah, total;

    public String getId() {
        return id;
    }

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

    public String getOrder_id() {
        return order_id;
    }

    public void setOrder_id(String order_id) {
        this.order_id = order_id;
    }

    public String getService_id() {
        return service_id;
    }

    public void setService_id(String service_id) {
        this.service_id = service_id;
    }

    public String getHarga() {
        return harga;
    }

    public void setHarga(String harga) {
        this.harga = harga;
    }

    public String getJumlah() {
        return jumlah;
    }

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

    public String getTotal() {
        return total;
    }

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



- Tambahkan file baru pada package dao dengan nama **OrderDetailRepo**, pada file ini akan dilakukan proses manipulasi data. File ini akan **mengimplements** file **OrderDetailDao**

```
private Connection connection;
final String insert = "INSERT INTO order_detail (order_id, service_id, harga,jumlah,total) VALUES (?,?,,?,?)";
final String delete = "DELETE FROM order_detail WHERE id=?";
final String update = "UPDATE order_detail SET order_id=?, service_id=?, harga=?, jumlah=?, total=? WHERE id=?";
```

### Simpan Order Detail

```
@Override
public void save(OrderDetail cs) {

    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(insert);
        st.setString(1, cs.getOrder_id());
        st.setString(2, cs.getService_id());
        st.setString(3, cs.getHarga());
        st.setString(4, cs.getJumlah());
        st.setString(5, cs.getTotal());
        st.executeUpdate();

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

### Menampilkan Detail Order

```
@Override
public List<OrderDetail> show(String order_id) {
    List<OrderDetail> ls=null;
    try {
        ls = new ArrayList<OrderDetail>();
        Statement st = connection.createStatement();
        String select = "SELECT * FROM order_detail where order_id='"+order_id+"'";
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            OrderDetail cs = new OrderDetail();
            cs.setId(rs.getString("id"));
            cs.setOrder_id(rs.getString("order_id"));
            cs.setService_id(rs.getString("service_id"));
            cs.setHarga(rs.getString("harga"));
            cs.setJumlah(rs.getString("jumlah"));
            cs.setTotal(rs.getString("total"));
            ls.add(cs);
        }

    }catch(SQLException e) {
        Logger.getLogger(OrderDetailDao.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}
```

### Menghapus Detail Order

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

### Mengubah Detail Order

```
@Override
public void update(OrderDetail cs) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, cs.getOrder_id());
        st.setString(2, cs.getService_id());
        st.setString(3, cs.getHarga());
        st.setString(4, cs.getJumlah());
        st.setString(5, cs.getTotal());
        st.setString(6, cs.getId());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            st.close();

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

## Menghitung Total Detail Order

```
@Override
public String total(String order_id) {
    String query_total = "SELECT sum(total) as total from order_detail WHERE order_id='"+order_id+"'";
    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) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return result;
}
```

- Tambahkan kode program berikut pada source **DetailOrderFrame**

```
OrderDetailRepo repo_od = new OrderDetailRepo();
List<OrderDetail> ls_od;
public String id_order_detail;
```

- Fungsi **Simpan Order Detail** tambahkan kode program berikut pada **event handler btnSimpanDetail**.

```
btnSimpanDetail.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id_order_detail == null) {
            OrderDetail od = new OrderDetail();
            od.setOrder_id(txtTrx.getText());
            od.setService_id(id_service);
            od.setHarga(txtHarga.getText());
            od.setJumlah(txtJumlah.getText());
            od.setTotal(txtTotal.getText());
            repo_od.save(od);
            JOptionPane.showMessageDialog(null, "berhasil disimpan");
            loadTableDetail();
            reset();
            txtTotalOrder.setText(""+repo_od.total(txtTrx.getText()));
        }
    }
});
```

Kode program diatas berfungsi untuk menyimpan data order detail kedalam database.

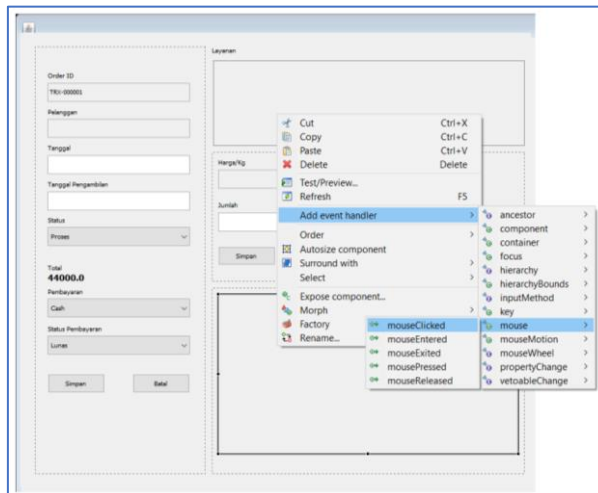
- Fungsi **Menampilkan Order Detail** tambahkan kode program berikut.

```
public void loadTableDetail() {
    ls_od = repo_od.show();
    TableOrderDetail tu = new TableOrderDetail(ls_od);
    tableOrderDetail.setModel(tu);
    tableOrderDetail.getTableHeader().setVisible(true);
}
```

Selanjutnya panggil method pada class main seperti kode berikut.

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                OrderDetailFrame frame = new OrderDetailFrame();
                frame.setVisible(true);
                frame.loadTableService();
                frame.loadTableDetail();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

- Fungsi Mengubah Order Detail klik kanan pada table `tableOrderDetail` → mouse → `mouseClicked`



Tambahkan kode program berikut :

```
tableOrderDetail.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id_order_detail = tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(),0).toString();
        id_service = tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(),2).toString();
        txtHarga.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(),3).toString());
        txtTotal.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(),5).toString());
        txtJumlah.setText(tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(),4).toString());
    }
});
```

Kemudian tambahkan **event handler** pada **btnUbahDetail** seperti kode program berikut.

```
btnUbahDetail.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        if(id_order_detail != null) {  
            OrderDetail od = new OrderDetail();  
            od.setOrder_id(txtTrx.getText());  
            od.setService_id(id_service);  
            od.setHarga(txtHarga.getText());  
            od.setJumlah(txtJumlah.getText());  
            od.setTotal(txtTotal.getText());  
            od.setId(id_order_detail);  
            repo_od.update(od);  
            loadTableDetail();  
            reset();  
            txtTotalOrder.setText(""+repo_od.total(txtTrx.getText()));  
        }else{  
            JOptionPane.showMessageDialog(null, "silahkan pilih order terlebih dahulu");  
        }  
    }  
});
```

- **Fungsi Hapus Detail Order** tambahkan kode program berikut pada event Handler **btnHapusDetail**

```
btnHapusDetail.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        if(id_order_detail != null) {  
            repo_od.delete(id_order_detail);  
            reset();  
            loadTableDetail();  
            txtTotalOrder.setText(""+repo_od.total(txtTrx.getText()));  
        }else {  
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");  
        }  
    }  
});
```

- **Fungsi Batal Detail Order** tambahkan kode program berikut pada event handler **btnBatalDetail**

```
btnBatalDetail.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        dispose();  
    }  
});
```

Sampai pada tahapan ini, kita sudah dapat melakukan proses menambah detail order, mengubah, menghapus dan membatalkannya berdasarkan dengan order id, berikut ini tampilan aplikasi Ketika melakukan proses CRUD data.

The screenshot shows a software window titled "Order Detail Frame". On the left, there are several form fields: "Order ID" (containing "TRX-000001"), "Pelanggan" (empty), "Tanggal" (empty), "Tanggal Pengambilan" (empty), "Status" (a dropdown menu with "Proses" selected), "Total" (displaying "0.0"), "Pembayaran" (a dropdown menu with "Cash" selected), and "Status Pemba..." (a dropdown menu with "Lunas" selected). At the bottom left are "Simpan" and "Batal" buttons. On the right, under the heading "Layanan", is a table with 4 columns: ID, Jenis, Satuan, and Harga. The table contains three rows: (1, Cuci, kg, 4000.0), (2, Setrika, kg, 4000.0), and (3, Cuci + Setrika, kg, 6000.0). Below the table are input fields for "Harga/Kg", "Jumlah", and "Total", each with a corresponding button: "Simpan", "Ubah", "Hapus", and "Batal". At the bottom right is another table with 6 columns: ID, Order ID, Service ID, Harga, Jumlah, and Total, which is currently empty.

Order Detail Frame

Ketika pertama kali OrderDetailFrame dijalankan maka form Order ID akan terisi secara otomatis, ini kedepannya akan dibuatkan generate Order ID secara otomatis, akan tetapi pada praktikum kali ini masih manual, selanjutnya akan tampilan daftar layanan yang tersedia jika dipilih salah satu layanan maka form harga akan terisi otomatis, seperti gambar berikut :

This screenshot shows the same "Order Detail Frame" but with the "Layanan" table selected. The first row, "1 Cuci kg 4000.0", is highlighted in blue. The "Harga/Kg" input field now contains the value "4000.0". The "Jumlah" and "Total" input fields remain empty. The "Simpan", "Ubah", "Hapus", and "Batal" buttons are still present at the bottom.

Pilih Layanan

Pada gambar diatas kita memilih layanan cuci maka secara otomatis form harga terisi dengan harga layanan yaitu 4.000/kg, selanjutnya operator akan menginputkan jumlah cuciannya kedalam form jumlah dan secara otomatis akan menghitung totalnya yaitu harga x jumlah dengan memanggil method total yang sudah dibuat.

**Layanan**

ID	Jenis	Satuan	Harga
1	Cuci	kg	4000.0
2	Setrika	kg	4000.0
3	Cuci + Setrika	kg	6000.0

**Harga/Kg**

4000.0

**Jumlah**

4

**Total**

16000.0

Input Jumlah

Pada gambar diatas diinputkan jumlahnya yaitu 4 maka secara otomatis totalnya menjadi 16.000, selanjutnya klik tombol simpan untuk menyimpannya, jika data berhasil disimpan maka data akan ditampilkan pada table detail order seperti gambar berikut :

ID	Order ID	Service ID	Harga	Jumlah	Total
18	TRX-000001	1	4000.0	4.0	16000.0

Detail Order

Jika ingin mengubah atau menghapus data dapat dilakukan dengan cara klik data order kemudian secara otomatis form harga, jumlah dan total akan terisi dengan data-data sesuai dengan yang dipilih, selanjutnya klik tombol ubah untuk mengubah data atau klik tombol hapus untuk menghapus data.

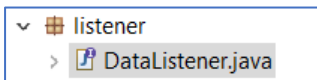
Langkah selanjutnya setelah berhasil menambahkan data order detail maka selanjutnya membuat pesanan, Langkah pertama adalah memilih pelanggan, menginputkan tanggal, tanggal pengambilan, memilih status, metode pembayaran dan status pembayaran.

## Membuat JDialog Pelanggan

JDialog berfungsi untuk menampilkan nama-nama pelanggan yang ada pada aplikasi, Ketika operator klik form pelanggan maka akan ditampilkan daftar pelanggan, selanjutnya operator memilih pelanggan maka secara otomatis form pelanggan akan terisi oleh data pelanggan tersebut.

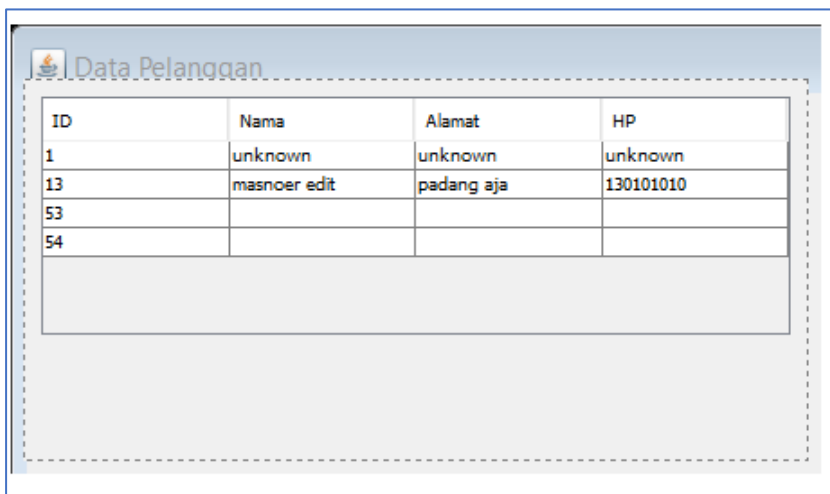
Ada 2 buah parameter yang dikirimkan oleh JDialog Ketika memilih pelanggan yaitu id dan nama, id merupakan id pelanggan yang nantinya akan disimpan sementara pada variable `id_pelanggan`, kemudian nama akan ditampilkan pada form nama, berikut ini Langkah-langkahnya.

- Buat package baru dengan nama **listener** kemudian tambahkan file java baru dengan nama **DataListener.java** file ini akan digunakan sebagai **interface** yang berfungsi menangkap data yang dikirimkan dari **JDialog Pelanggan**, tambahkan kode program dibawah ini.



```
public interface DataListener {  
    void onDataReceived(String id, String nama);  
}
```

- Tambahkan **JDialog** pada package **ui** dengan nama **DialogPelanggan.java**, buat desain UI seperti gambar berikut :





Tambahkan kode program berikut pada **DialogPelanggan.java**

```
public class DialogPelanggan extends JDialog {
    private DataListener listener;
    CostumerRepo usr = new CostumerRepo();
    List<Costumer> ls;
    public String id;
    private JTable tablePelanggan;

    public DialogPelanggan(DataListener listener) {
        this.listener = listener;
        setModal(true);
        setModalityType(ModalityType.APPLICATION_MODAL);
        setSize(450, 249);
        setLocationRelativeTo(null);
        setTitle("Data Pelanggan");
        getContentPane().setLayout(null);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
            }
        });
        scrollPane.setBounds(10, 10, 416, 132);
        getContentPane().add(scrollPane);

        tablePelanggan = new JTable();
        tablePelanggan.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                listener.onDataReceived(tablePelanggan.getValueAt(tablePelanggan.getSelectedRow(),0).toString(), tablePelanggan.getValueAt(tablePelanggan.getSelectedRow(),1).toString());
                getData();
                dispose();
            }
        });
        scrollPane.setViewportView(tablePelanggan);
        loadTable();
    }

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

- Menampilkan JDialog Ketika form pelanggan diklik, tambahkan event handler berikut pada **txtPelanggan**.

```
txtPelanggan.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        DialogPelanggan dialog = new DialogPelanggan(OrderDetailFrame.this);
        dialog.setVisible(true);
    }
});
```

- Tambahkan **implements DataListener**

```
public class OrderDetailFrame extends JFrame implements DataListener {
```

Maka akan meng-override method **onDataReceived** yang berfungsi menerima data id customer dan nama customer.

```
@Override
public void onDataReceived(String id, String nama) {
    txtPelanggan.setText(nama);
    id_pelanggan=id;
}
```

## Menyimpan Data Order

Setelah proses CRUD order detail selesai selanjutnya menyimpan data Order, berikut ini Langkah-langkahnya.

- Buat table baru dengan nama **Orders** dan Struktur Table seperti berikut.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	varchar(32)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	2 id_pelanggan	int(11)			No	None			Change  Drop  More
<input type="checkbox"/>	3 tanggal	date			No	None			Change  Drop  More
<input type="checkbox"/>	4 tanggal_pengambilan	date			No	None			Change  Drop  More
<input type="checkbox"/>	5 status	varchar(32)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	6 pembayaran	varchar(32)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	7 status_pembayaran	varchar(32)	utf8mb4_general_ci		No	None			Change  Drop  More
<input type="checkbox"/>	8 total	varchar(32)	utf8mb4_general_ci		No	None			Change  Drop  More

- Tambahkan model baru pada package **model** dengan nama **Order** dan tambahkan kode program berikut.

```
public class Order {  
  
    String id, id_pelanggan, tanggal, tanggal_pengambilan, status, pembayaran, status_pembayaran, total;  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public String getId_pelanggan() {  
        return id_pelanggan;  
    }  
  
    public void setId_pelanggan(String id_pelanggan) {  
        this.id_pelanggan = id_pelanggan;  
    }  
  
    public String getTanggal() {  
        return tanggal;  
    }  
  
    public void setTanggal(String tanggal) {  
        this.tanggal = tanggal;  
    }  
  
    public String getTanggal_pengambilan() {  
        return tanggal_pengambilan;  
    }  
  
    public void setTanggal_pengambilan(String tanggal_pengambilan) {  
        this.tanggal_pengambilan = tanggal_pengambilan;  
    }  
  
    public String getStatus() {  
        return status;  
    }  
  
    public void setStatus(String status) {  
        this.status = status;  
    }  
  
    public String getPembayaran() {  
        return pembayaran;  
    }  
  
    public void setPembayaran(String pembayaran) {  
        this.pembayaran = pembayaran;  
    }  
  
    public String getStatus_pembayaran() {  
        return status_pembayaran;  
    }  
  
    public void setStatus_pembayaran(String status_pembayaran) {  
        this.status_pembayaran = status_pembayaran;  
    }  
  
    public String getTotal() {  
        return total;  
    }  
  
    public void setTotal(String total) {  
        this.total = total;  
    }  
}
```

- Tambahkan **DAO** pada package **dao** dengan nama **OrderDao** dan tambahkan kode program berikut.

```
public interface OrderDao{
    void save(Order cs);
    public List<Order> show();
    public void delete(Order id);
    public void update(Order cs);
}
```

- Tambahkan **OrderRepo** pada package **dao** dan tambahkan kode program berikut ini.

```
private Connection connection;
final String insert = "INSERT INTO orders (id, id_pelanggan, tanggal, tanggal_pengambilan, status, pembayaran, status_pembayaran, total) "
+ "VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
final String select = "SELECT * FROM orders;";
final String delete = "DELETE FROM orders WHERE id=?;";
final String delete_detail = "DELETE FROM order_detail WHERE order_id=?;";
final String update = "UPDATE orders SET id_pelanggan=?, tanggal=?, tanggal_pengambilan=?, status=?, "
+ "pembayaran=?, status_pembayaran=?, total=? WHERE id=?;";

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

#### Menambah Data Order

```
@Override
public void save(Order cs) {
    // TODO Auto-generated method stub
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(insert);
        st.setString(1, cs.getId());
        st.setString(2, cs.getId_pelanggan());
        st.setString(3, cs.getTanggal());
        st.setString(4, cs.getTanggal_pengambilan());
        st.setString(5, cs.getStatus());
        st.setString(6, cs.getPembayaran());
        st.setString(7, cs.getStatus_pembayaran());
        st.setString(8, cs.getTotal());
        st.executeUpdate();

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

## Menampilkan Data Order

```
@Override
public List<Order> show() {
    // TODO Auto-generated method stub
    List<Order> ls=null;
    try {
        ls = new ArrayList<Order>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            Order cs = new Order();
            cs.setId(rs.getString("id"));
            cs.setId_pelanggan(rs.getString("id_pelanggan"));
            cs.setTanggal(rs.getString("tanggal"));
            cs.setTanggal_pengambilan(rs.getString("tanggal_pengambilan"));
            cs.setStatus(rs.getString("status"));
            cs.setPembayaran(rs.getString("pembayaran"));
            cs.setStatus_pembayaran(rs.getString("status_pembayaran"));
            cs.setTotal(rs.getString("total"));
            ls.add(cs);
        }
    } catch (SQLException e) {
        Logger.getLogger(UserDao.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}
```

## Menghapus Data Order

```
@Override
public void delete(String id) {
    // TODO Auto-generated method stub
    PreparedStatement st = null;
    PreparedStatement st_detail = null;
    try {
        st = connection.prepareStatement(delete);
        st.setString(1, id);
        st.executeUpdate();

        st_detail = connection.prepareStatement(delete_detail);
        st_detail.setString(1, id);
        st_detail.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            st.close();
            st_detail.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

### Mengubah Data Order

```
@Override
public void update(Order cs) {
    // TODO Auto-generated method stub
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(update);
        st.setString(1, cs.getId_pelanggan());
        st.setString(2, cs.getTanggal());
        st.setString(3, cs.getTanggal_pengambilan());
        st.setString(4, cs.getStatus());
        st.setString(5, cs.getPembayaran());
        st.setString(6, cs.getStatus_pembayaran());
        st.setString(7, cs.getTotal());
        st.setString(8, cs.getId());
        st.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }

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

Sebelum menyimpan data Order maka harus dipastikan terlebih dahulu data pelanggan, tanggal, tanggal pengambilan, status, pembayaran dan status pembayaran baru diklik tombol simpan.

Order ID

TRX-000001

Pelanggan

nurfiah

Tanggal

2024-11-18

Tanggal Pengambilan

2024-11-21

Status

Proses

Total

16000.0

Pembayaran

Cash

Status Pemba...

Lunas

Simpan

Batal

Layanan

ID	Jenis	Satuan	Harga
1	Cuci	kg	4000.0
2	Setrika	kg	4000.0
3	Cuci + Setrika	kg	8000.0

Harga/Kg

Jumlah

Total

Simpan

Ubah

Hapus

Batal

ID	Order ID	Service ID	Harga	Jumlah	Total
18	TRX-000001	1	4000.0	4.0	16000.0

Tambahkan kode program berikut ini pada tombol simpan.

```

btnSimpanOrder.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OrderRepo order_repo = new OrderRepo();

        if(id_pelanggan != "") {
            Order order = new Order();
            order.setId(txtTrx.getText());
            order.setId_pelanggan(id_pelanggan);
            order.setTanggal(txtTanggal.getText());
            order.setTanggal_pengambilan(txtTanggalPengambilan.getText());
            order.setStatus(cbxBStatus.getSelectedItem().toString());
            order.setStatus_pembayaran(cbxBStatusPembayaran.getSelectedItem().toString());
            order.setPembayaran(cbxBPembayaran.getSelectedItem().toString());
            order.setTotal(txtTotalOrder.getText());
            order_repo.save(order);
            JOptionPane.showMessageDialog(null, "Order Berhasil Disimpan");
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih Pelanggan terlebih dahulu");
        }
    }
});

```

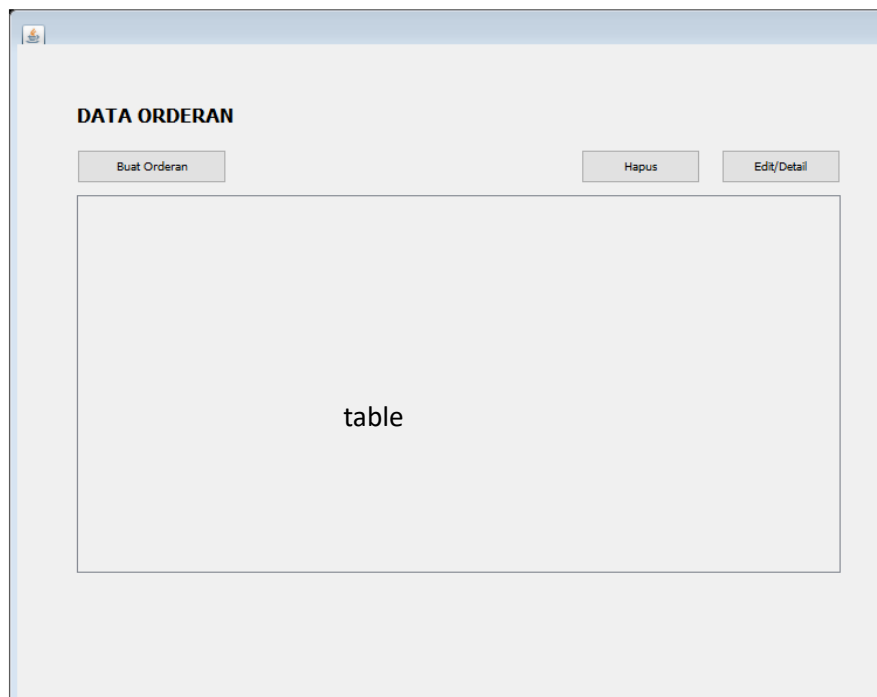
## Menampilkan Data Order

Setelah data order berhasil disimpan, selanjutnya menampilkan data order.

- Pertama-tama buatlah terlebih dahulu class untuk table order, tambahkan class baru pada package table dengan nama **TableOrder** tambahkan kode program berikut :

```
public class TableOrder extends AbstractTableModel {
    List<Order> ls;
    private String[] columnNames = {"Order ID", "ID Pelanggan", "Tanggal", "Status", "Status Pembayaran", "Total"};
    public TableOrder(List<Order> ls) {
        this.ls = ls;
    }
    @Override
    public int getRowCount() {
        // TODO Auto-generated method stub
        return ls.size();
    }
    @Override
    public int getColumnCount() {
        // TODO Auto-generated method stub
        return 6;
    }
    @Override
    public String getColumnName(int column) {
        // TODO Auto-generated method stub
        return columnNames[column];
    }
    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        // TODO Auto-generated method stub
        switch (columnIndex) {
            case 0:
                return ls.get(rowIndex).getId();
            case 1:
                return ls.get(rowIndex).getId_pelanggan();
            case 2:
                return ls.get(rowIndex).getTanggal();
            case 3:
                return ls.get(rowIndex).getStatus();
            case 4:
                return ls.get(rowIndex).getStatus_pembayaran();
            case 5:
                return ls.get(rowIndex).getTotal();
            default:
                return null;
        }
    }
}
```

- Kemudian tambahkan JFrame baru dengan nama OrderFrame, buat desainnya seperti gambar berikut.



Keterangan :

Komponen	Variabel	Keterangan
JButton	btnOrder	Buat Orderan
JButton	btnHapus	Hapus
JButton	btnEditDetail	Edit/Detail
JTable	tblOrder	Table Order

- Selanjutnya tambahkan kode program berikut pada **OrderFrame**

```
OrderRepo repo_od = new OrderRepo();
List<Order> ls_od;
String order_id="";
```

- Buat method untuk menampilkan data order

```
public void loadTableOrder() {
    ls_od = repo_od.show();
    TableOrder tu = new TableOrder(ls_od);
    tblOrder.setModel(tu);
    tblOrder.getTableHeader().setVisible(true);
}
```

- Panggil method loadTableOrder() pada main frame

```
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                OrderFrame frame = new OrderFrame();
                frame.setVisible(true);
                frame.loadTableOrder();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```



- Tampilan Data Order

**DATA ORDERAN**

Buat Orderan Hapus Edit/Detail

Order ID	ID Pelanggan	Tanggal	Status	Status Pembayaran	Total
TRX-000001	61	2024-11-18	Proses	Lunas	16000.0

- Tambahkan kode program berikut pada **event handler Buat Orderan** sehingga akan tampil form **OrderDetailFrame** dan dapat menambahkan data order.

```

JButton btnOrder = new JButton("Buat Orderan");
btnOrder.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        OrderDetailFrame odf = new OrderDetailFrame();
        odf.setVisible(true);
        odf.loadTableDetail();
        odf.loadTableService();
    }
});

```

- Tambahkan kode program berikut pada event handler pada tombol hapus.

```

btnHapus.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(order_id != "") {
            repo_od.delete(order_id);
        } else {
            JOptionPane.showMessageDialog(null, "Pilih data yang akan dihapus");
        }
    }
});

```

Untuk menghapus data, pilih data order dan klik tombol hapus.

## 1.5 Latihan /Tugas

Sampai pada bagian akhir modul ini sudah dapat melakukan pemesanan laundry, akan tetapi masih perlu penambahan beberapa fitur, untuk itu sebagai Latihan/tugas maka silakan tambahkan fitur-fitur berikut :

- Order ID masih manual, untuk itu buatlah sebuah fungsi yang dapat generate Order ID secara otomatis dan unik, sehingga tidak ada duplikasi Order ID Ketika membuat pesananan/orderan.
- Tanggal pesanan dan tanggal pengambilan silahkan ditambahkan menggunakan JCalender
- Pada saat tombol Edit/Detail pada OrderFrame diklik maka akan memanggil OrderDetailFrame dan menampilkan data-data orderan sesuai dengan data yang dipilih dan dapat dilakukan fungsi untuk edit orderan.