Laporan Praktikum

Pemrograman Berorientasi Objek



Disusun Oleh :

**Ahda Rindang Al-Amin (2311531003)**

Dosen Pengampu : Nurfiah, S.ST, M.Kom.

Departemen Informatika

Fakultas Teknologi Informasi

Universitas Andalas

Tahun 2024

**Membuat Fungsi CRUD (Create, Read, Update, Delete) Service dan Customer dengan Database MySQL**

1. **Tujuan Praktikum**
2. Mampu membuat table service dan table customer pada database MySQL
3. Mampu membuat tampilan GUI CRUD service dan customer
4. Mampu membuat dan mengimplementasikan interface
5. Mampu membuat fungsi DAO (Data Access Object)dan mengimplementasikannya.
6. Mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek
7. **Pendahuluan**

**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.

**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

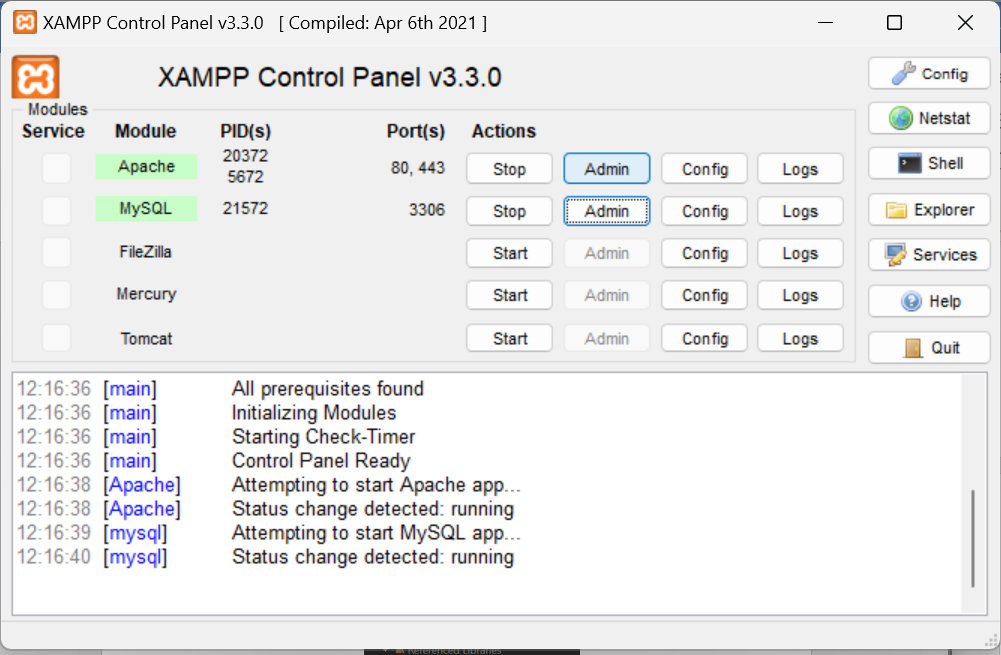
**DAO (Data Access Object)** merupakan object yang menyediakan abstract interface terhadap beberapa method yang berhubungan dengan database seperti mengambil data (read), menyimpan data(create), menghapus data (delete), mengubah data(update). Tujuan penggunaan DAO yaitu :

* 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.

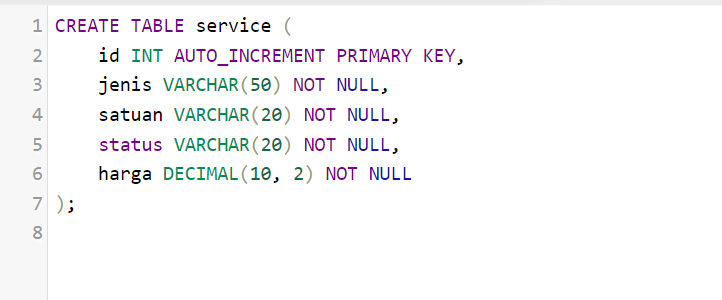
**Interface** dalam Bahasa java yaitu mendefinisikan beberapa method abstrak yang harus diimplementasikan oleh class yang akan menggunakannya.

**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.

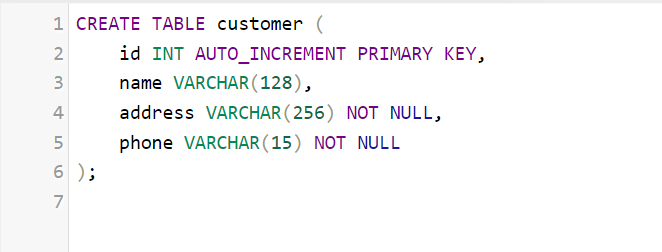
1. **Metode Praktikum**
2. **Membuat Table pada Database**
3. Buka XAMPP dan jalankan Apache dan MySQL



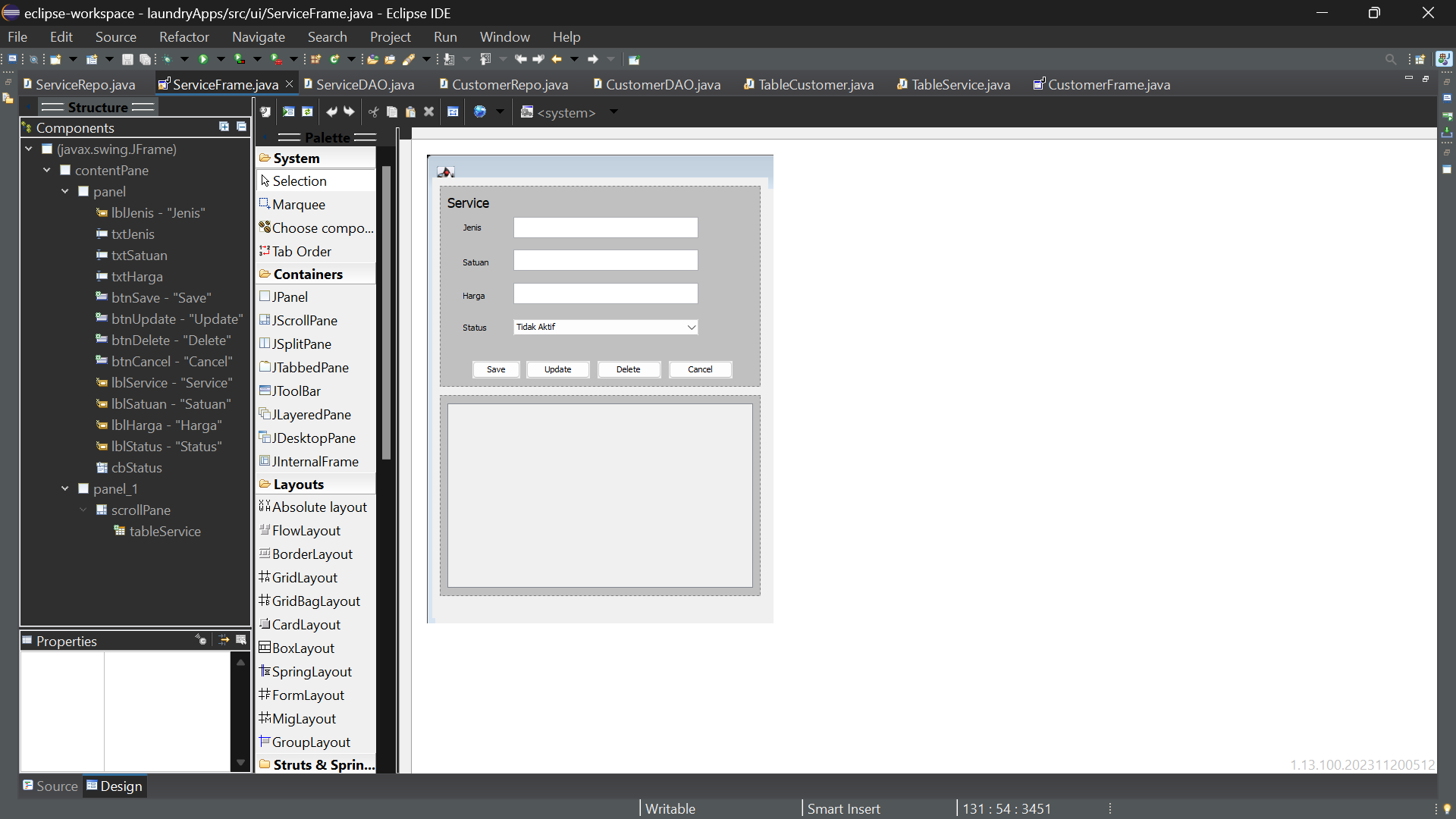
1. Buka phpMyAdmin, buat tabel service dan customer pada database laundry\_apps
2. Buat tabel service dengan SQL sebagai berikut



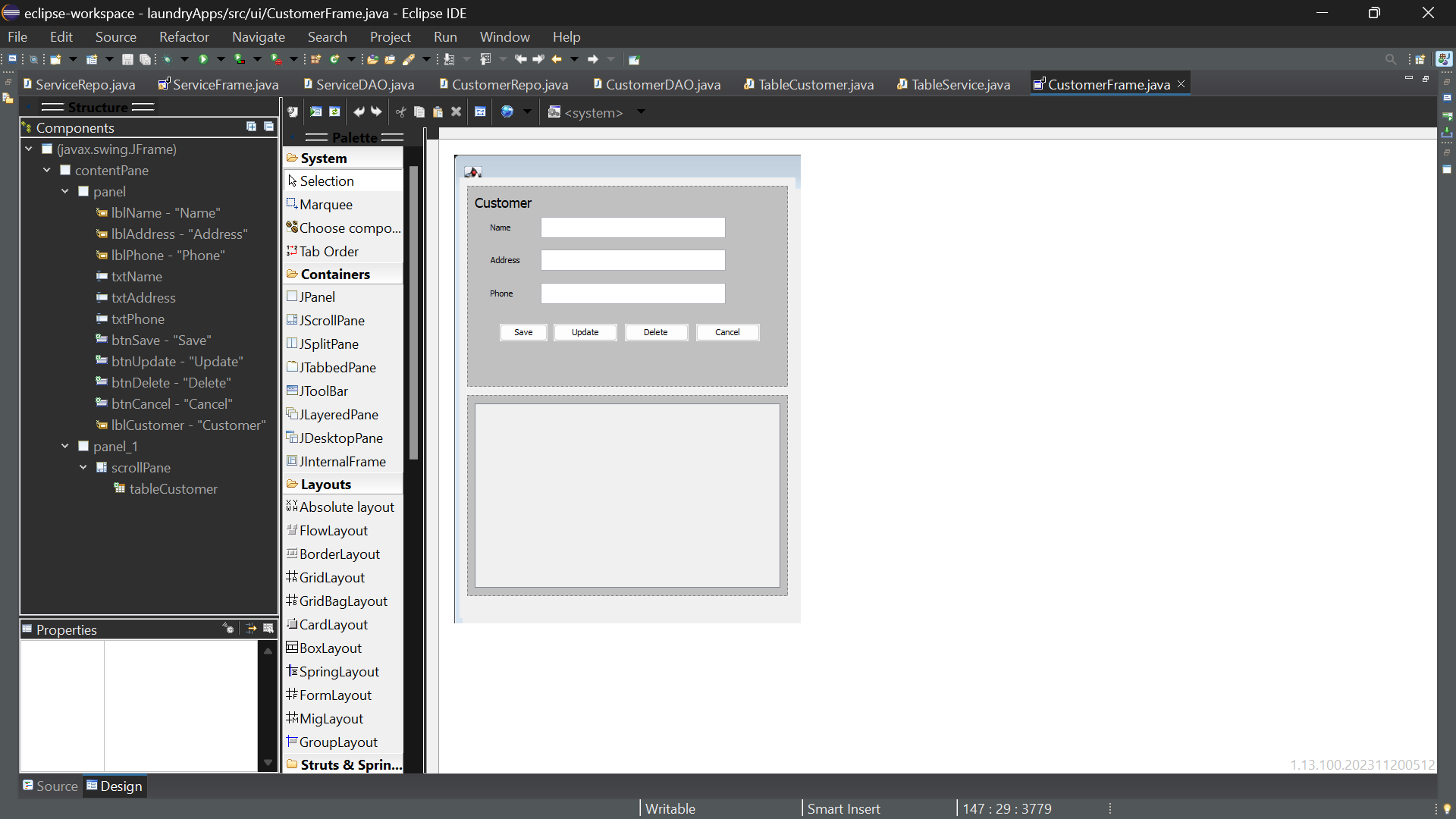
1. Buat tabel customer dengan SQL sebagai berikut



1. **Membuat tampilan CRUD**
2. Buat frame baru untuk service menggunakan JFrame pada package ui dengan nama ServiceFrame.



1. Buat frame baru untuk customer menggunakan JFrame pada package ui dengan nama CustomerFrame.



1. **Membuat Table Model**
2. Buat class baru untuk table model service dalam package table dengan nama TableService

package table;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import model.Service;

public class TableService extends AbstractTableModel {

List<Service> ls;

private String[] columnNames = {"ID", "Jenis", "Satuan", "Harga", "Status"};

public TableService(List<Service> 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 5;

}

public String getColumnName(int column) {

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).getJenis();

case 2:

return ls.get(rowIndex).getSatuan();

case 3:

return ls.get(rowIndex).getHarga();

case 4:

return ls.get(rowIndex).getStatus();

default:

return null;

}

}

}

1. Buat class baru untuk table model customer dalam package table dengan nama TableCustomer

package table;

import java.util.List;

import javax.swing.table.AbstractTableModel;

import model.Customer;

public class TableCustomer extends AbstractTableModel {

List<Customer> ls;

private String[] columnNames = {"ID", "Name", "Address", "Phone"};

public TableCustomer(List<Customer> 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;

}

public String getColumnName(int column) {

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).getAlamat();

case 3:

return ls.get(rowIndex).getNoHP();

default:

return null;

}

}

}

1. **Membuat Fungsi DAO**
2. Buat interface baru untuk service dengan nama ServiceDAO

public interface ServiceDAO {

void save(Service service);

public List<Service> show();

public void delete(String id);

public void update(Service service);

}

1. Buat interface baru untuk customer dengan nama CustomerDAO

public interface CustomerDAO {

void save(Customer customer);

public List<Customer> show();

public void delete(String id);

public void update(Customer customer);

}

1. **Menggunakan Fungsi DAO**
2. Buat class baru pada package DAO dengan nama ServiceDAO yang digunakan untuk mengimplementasikan interface ServiceDAO

public class ServiceRepo implements ServiceDAO{

private Connection connection;

final String insert = "INSERT INTO service (jenis, satuan, harga, status) VALUES (?,?,?,?);";

final String select = "SELECT \* FROM service;";

final String delete = "DELETE FROM service WHERE id=?;";

final String update = "UPDATE service SET jenis=?, satuan=?, harga=?, status=? WHERE id=?;";

public ServiceRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(Service service) {

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, service.getJenis());

st.setString(2, service.getSatuan());

st.setInt(3, service.getHarga());

st.setString(4, service.getStatus());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<Service> show() {

List<Service> ls = null;

try {

ls = new ArrayList<Service>();

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

Service service = new Service();

service.setId(rs.getString("id"));

service.setJenis(rs.getString("jenis"));

service.setSatuan(rs.getString("satuan"));

service.setHarga(rs.getInt("harga"));

service.setStatus(rs.getString("status"));

ls.add(service);

}

} catch(SQLException e) {

Logger.*getLogger*(ServiceDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@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();

}

}

}

*@Override*

public void update(Service service) {

PreparedStatement st = null;

try {

st= connection.prepareStatement(update);

st.setString(1, service.getJenis());

st.setString(2, service.getSatuan());

st.setInt(3, service.getHarga());

st.setString(4, service.getStatus());

st.setString(5, service.getId());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}}

}

1. Buat class baru pada package DAO dengan nama CustomerRepo yang digunakan untuk mengimplementasikan interface CustomerDAO

public class CustomerRepo implements CustomerDAO{

private Connection connection;

final String insert = "INSERT INTO customer (name, address, phone) VALUES (?,?,?);";

final String select = "SELECT \* FROM customer;";

final String delete = "DELETE FROM customer WHERE id=?;";

final String update = "UPDATE customer SET name=?, address=?, phone=? WHERE id=?;";

public CustomerRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(Customer customer) {

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, customer.getNama());

st.setString(2, customer.getAlamat());

st.setString(3, customer.getNoHP());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<Customer> show() {

List<Customer> ls = null;

try {

ls = new ArrayList<Customer>();

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

Customer customer = new Customer();

customer.setId(rs.getString("id"));

customer.setNama(rs.getString("name"));

customer.setAlamat(rs.getString("address"));

customer.setNoHP(rs.getString("phone"));

ls.add(customer);

}

} catch(SQLException e) {

Logger.*getLogger*(CustomerDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@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();

}

}

}

*@Override*

public void update(Customer customer) {

PreparedStatement st = null;

try {

st= connection.prepareStatement(update);

st.setString(1, customer.getNama());

st.setString(2, customer.getAlamat());

st.setString(3, customer.getNoHP());

st.setString(4, customer.getId());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

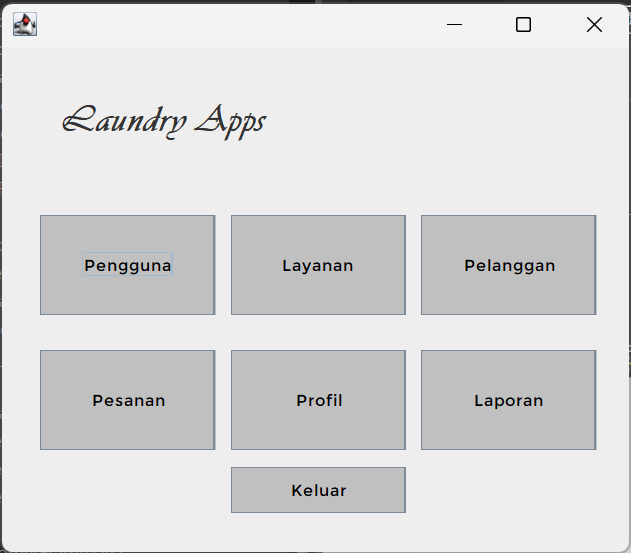
e.printStackTrace();

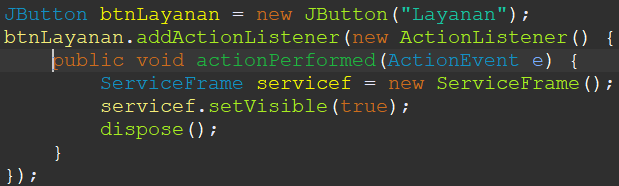
}

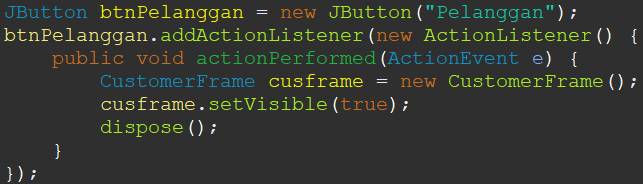
}}

}

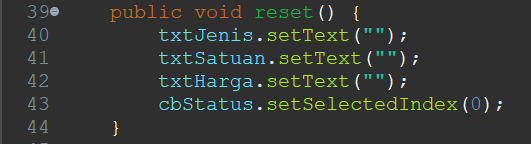
1. **Hubungkan GUI Service dan Customer dengan MainFrame**



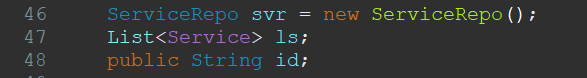




1. **Menggunakan Fungsi CRUD Service pada GUI**
2. Buat method reset pada ServiceFrame untuk menghapus value inputan ketika suatu proses berhasil dilaksanakan



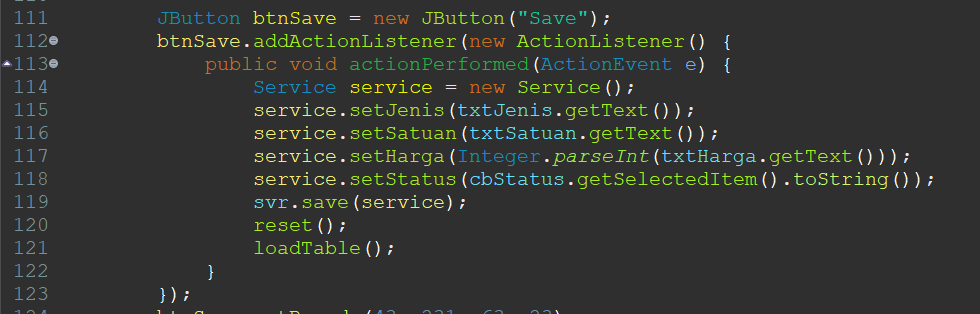
1. Buat Instance pada ServiceFrame



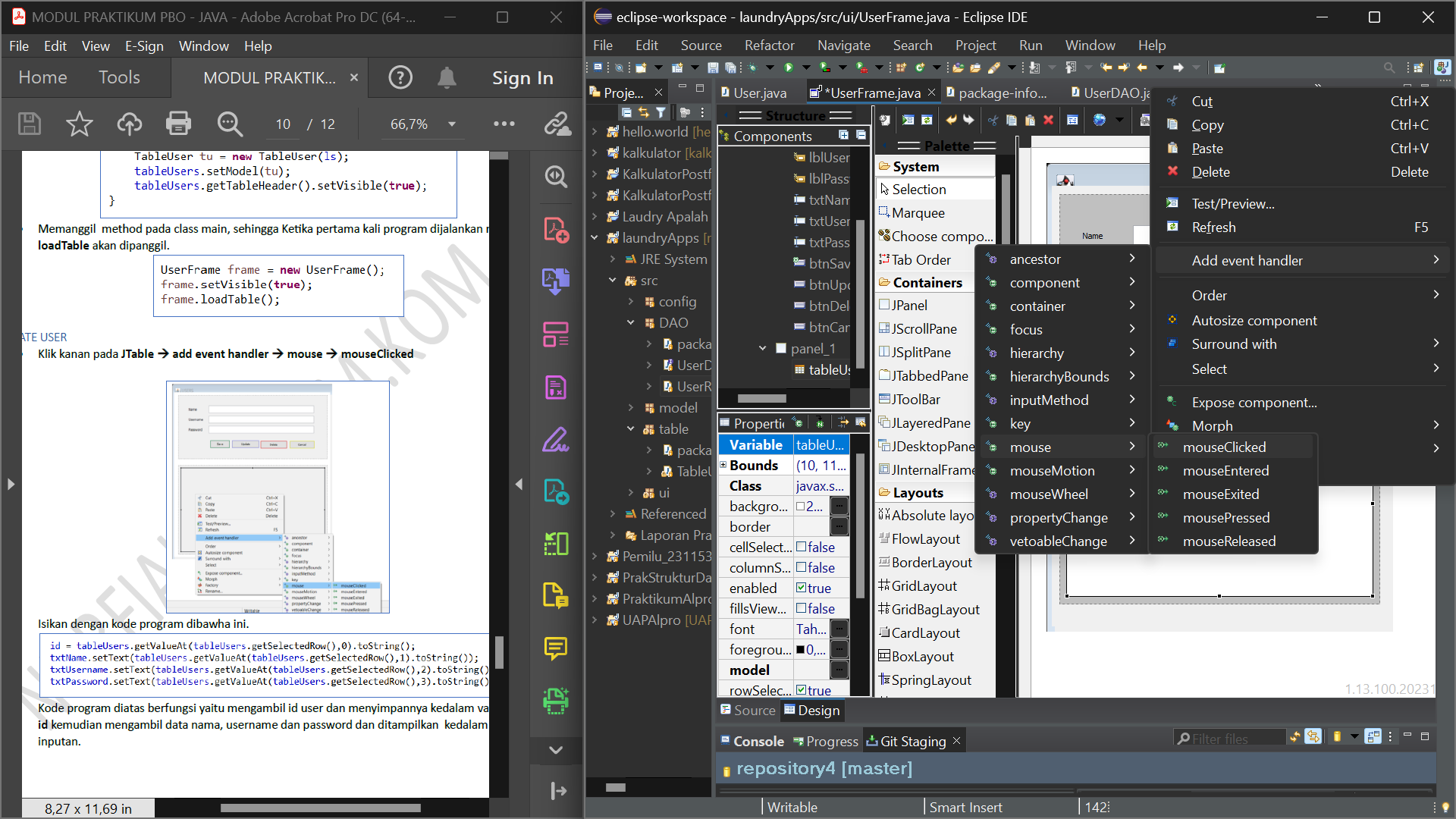
1. Buat method loadTable , lalu panggil pada method main



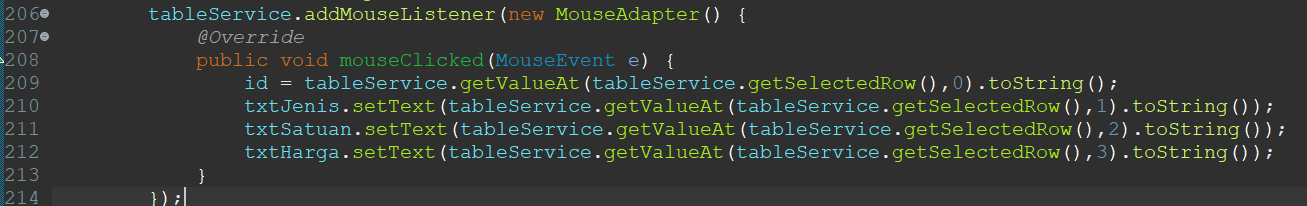
1. Klik dua kali pada tombol save untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



1. Klik kanan pada JTable > add event handler > mouse > mouseClicked

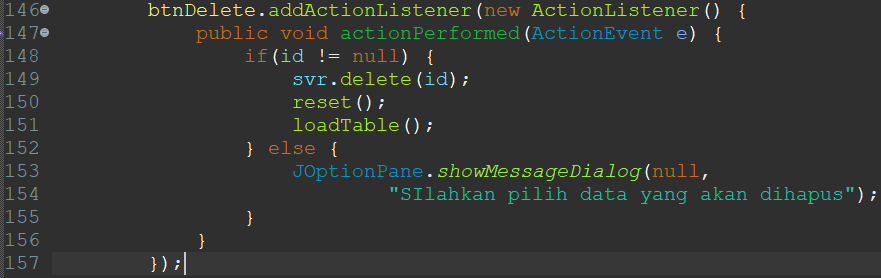


1. Isi mouseClicked dengan program berikut

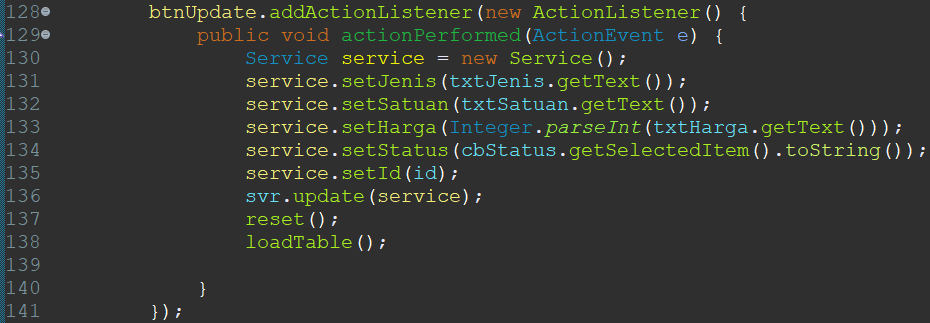


Kode program diatas berfungsi yaitu mengambil id service dan menyimpannya kedalam variable id kemudian mengambil data jenis, satuan dan harga dan ditampilkan kedalam form inputan.

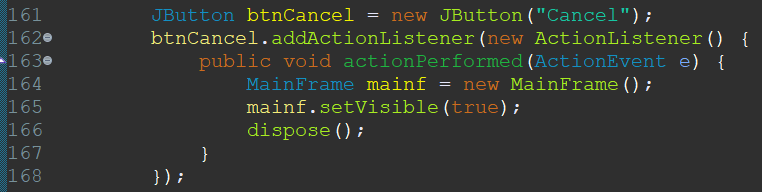
1. Klik dua kali pada tombol delete untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



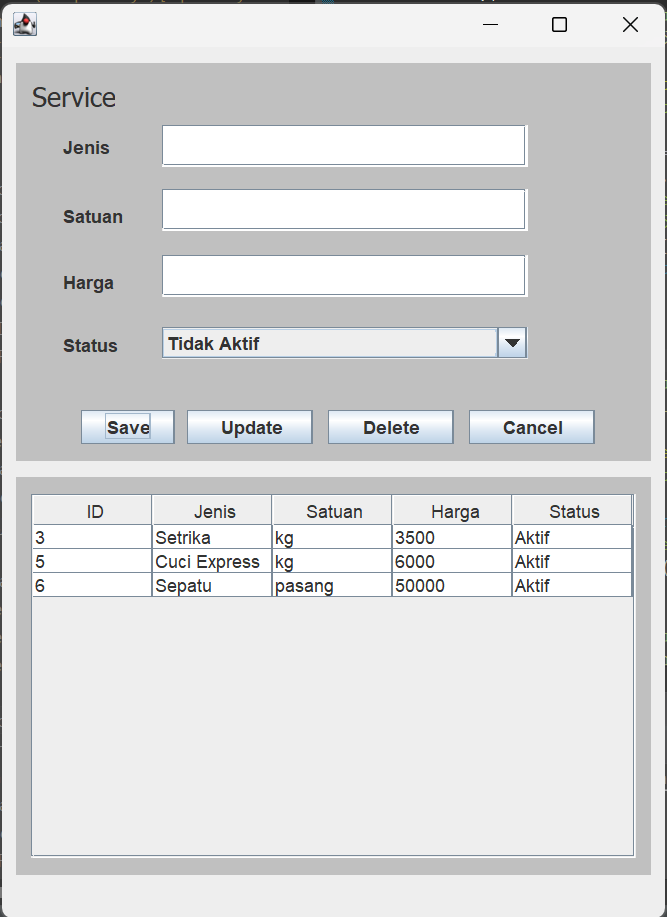
1. Klik dua kali pada tombol update untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



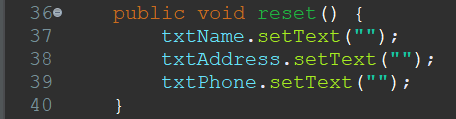
1. Klik dua kali pada tombol cancel untuk menambahkan event handlers actionPerformed, kemudian isi dengan perintah dispose() untuk menutup GUI ketika tombol cancel ditekan dan kembali ke MainFrame.



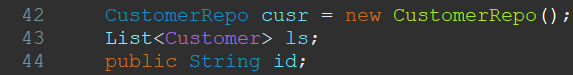
1. Fungsi CRUD Service sudah bisa digunakan



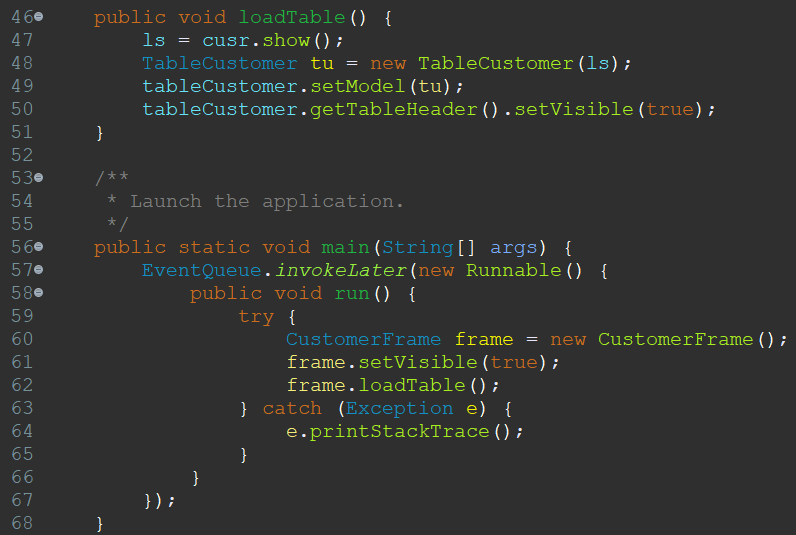
1. **Menggunakan Fungsi CRUD Customer pada GUI**
2. Buat method reset pada CustomerFrame untuk menghapus value inputan ketika suatu proses berhasil dilaksanakan



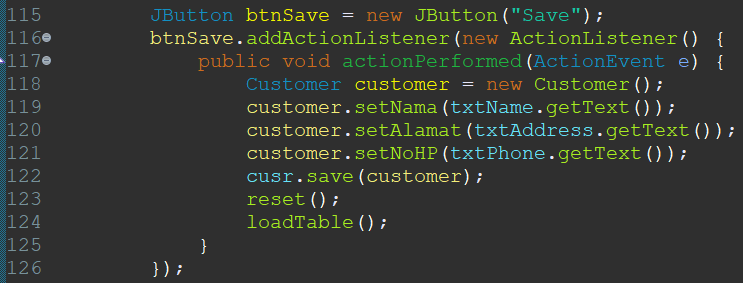
1. Buat Instance pada CustomerFrame



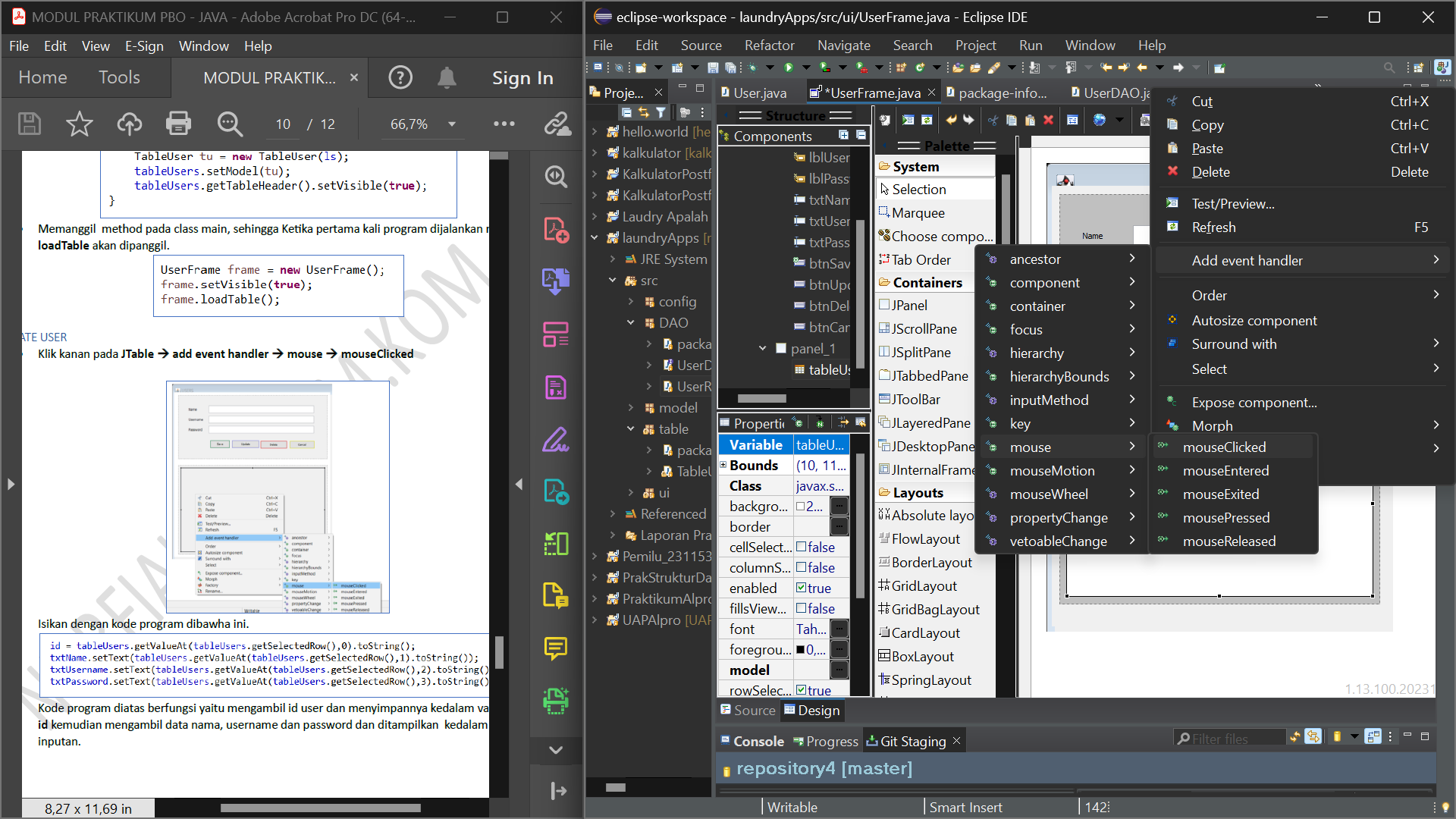
1. Buat method loadTable , lalu panggil pada method main



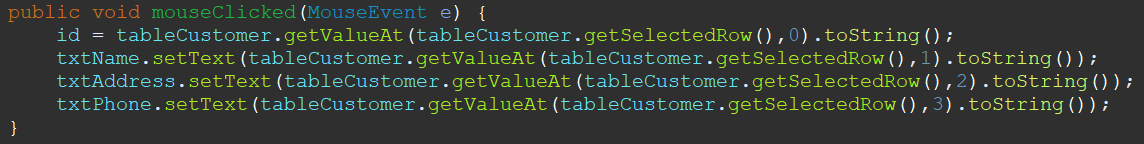
1. Klik dua kali pada tombol save untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



1. Klik kanan pada JTable > add event handler > mouse > mouseClicked

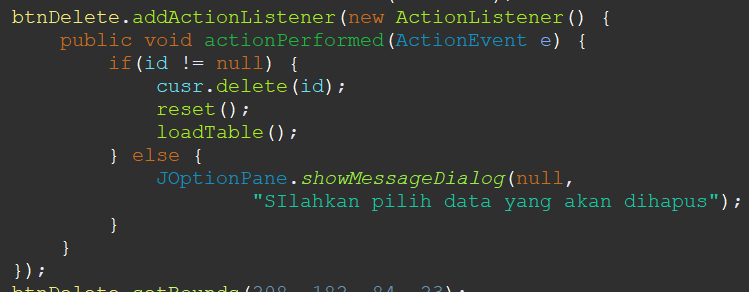


1. Isi mouseClicked dengan program berikut

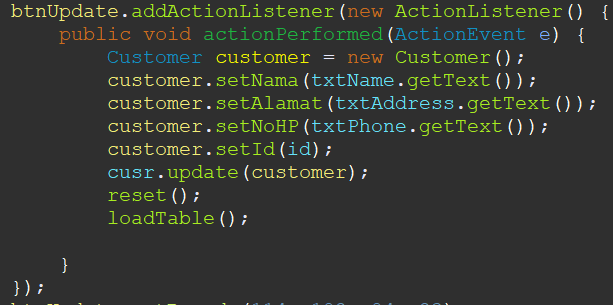


Kode program diatas berfungsi yaitu mengambil id customer dan menyimpannya kedalam variable id kemudian mengambil data nama, alamat dan telepon dan ditampilkan kedalam form inputan.

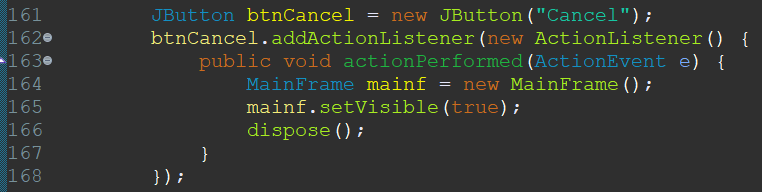
1. Klik dua kali pada tombol delete untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



1. Klik dua kali pada tombol update untuk menambahkan event handlers actionPerformed, kemudian isi dengan program berikut



1. Klik dua kali pada tombol cancel untuk menambahkan event handlers actionPerformed, kemudian isi dengan perintah dispose() untuk menutup GUI ketika tombol cancel ditekan dan kembali ke MainFrame.



1. Fungsi CRUD Customer sudah bisa digunakan

