

PEMROGRAMAN BERORIENTASI OBJEK

PRAKTIKUM 3

MEMBUAT FUNGSI CRUD (Create, Read, Update, Delete)

CUSTOMER & SERVICE DENGAN DATABASE MYSQL



Nama: Siti Fadhilah Rahmi

NIM: 2311532003

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

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2024

A. PENDAHULUAN

1. Tujuan Praktikum

- a. Mampu membuat table customer dan service pada database MySQL
- b. mampu membuat tampilan GUI CRUD customer dan service
- c. mampu membuat dan mengimplementasikan interface
- d. mampu membuat fungsi DAO (Data Access Object) dan mengimplementasikannya.
- e. mampu membuat fungsi CRUD dengan menggunakan konsep Pemrograman Berorientasi Objek.

2. Kajian Teori

a. MySQL

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.

b. MySQL Connection/j

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

c. DAO (Data Access Object)

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).

d. Interface

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

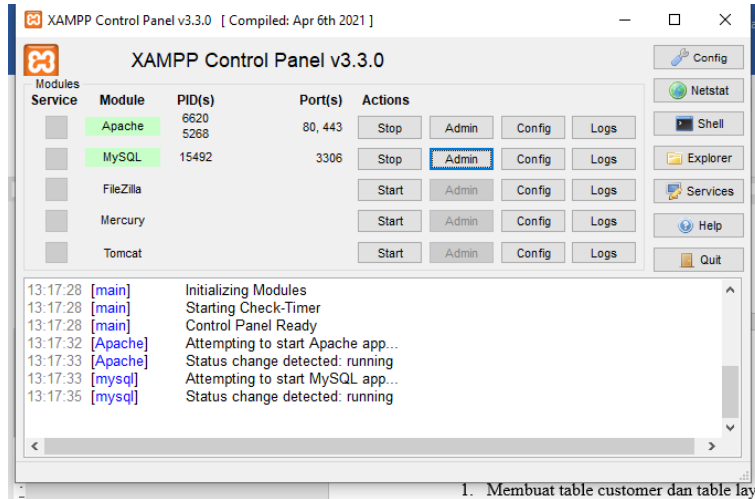
e. CRUD (Create, Read, Update, Delete)

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.

B. PRAKTIKUM

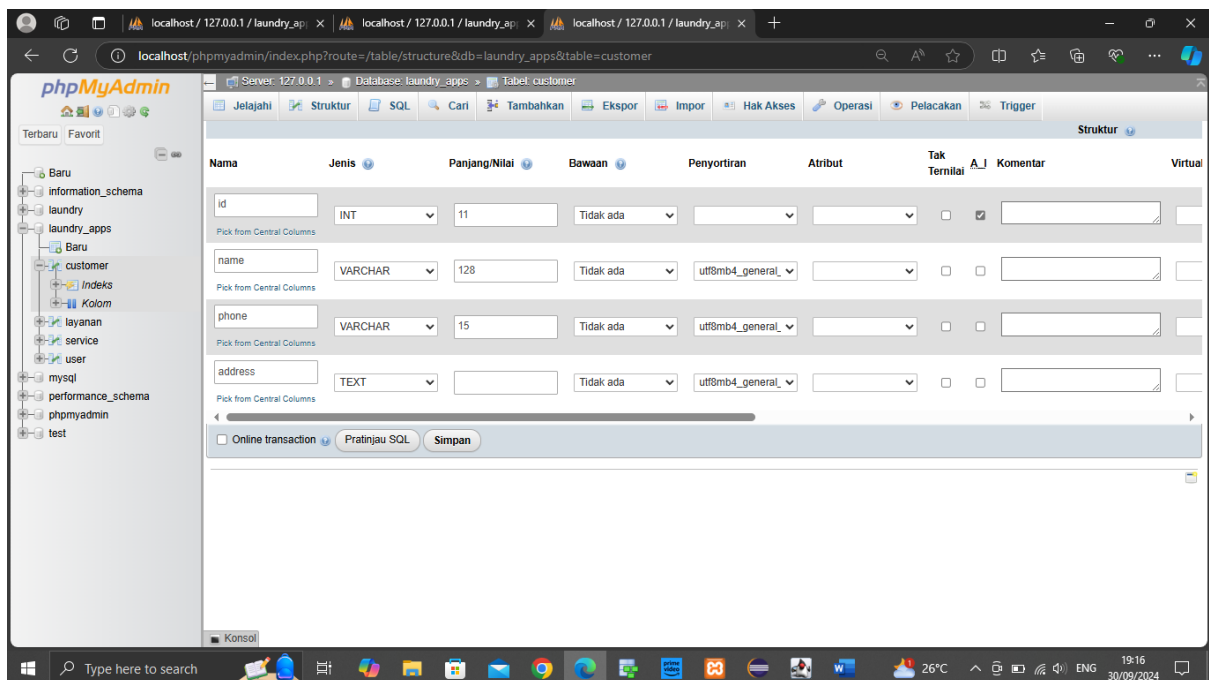
1. Membuat table customer dan table layanan di database

a. Aktifkan Apache dan MySQL di XAMPP dan masuk ke phpmyadmin

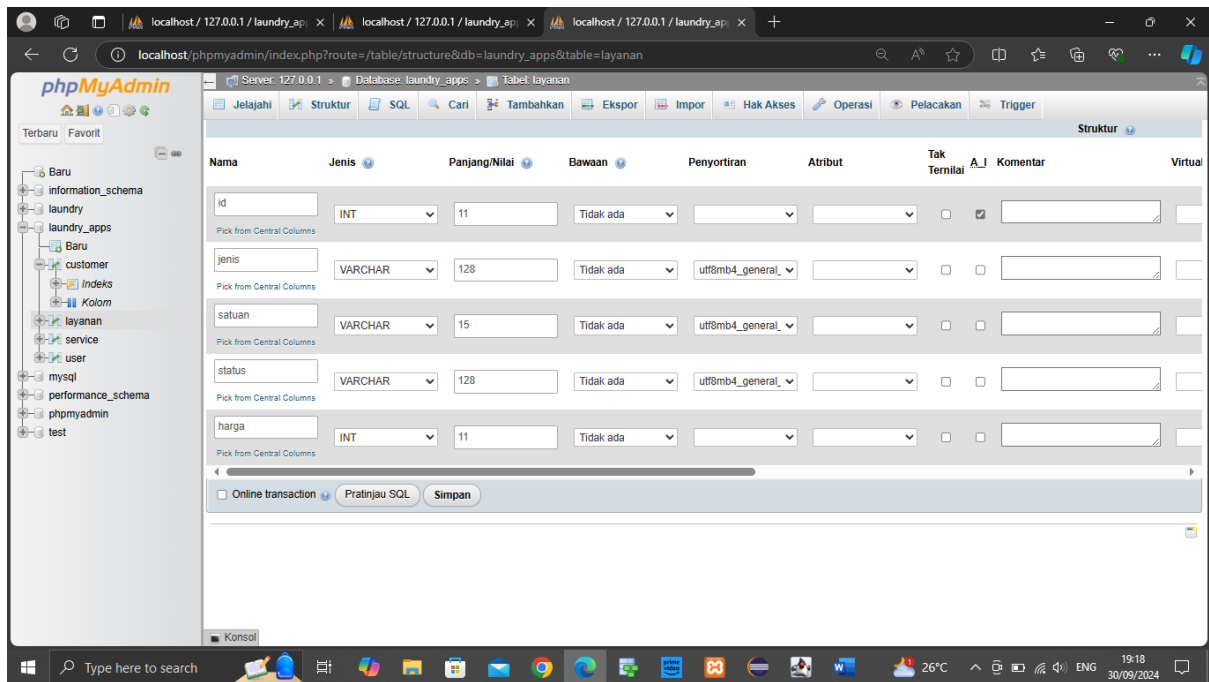


1. Membuat table customer dan table layanan

b. Buat table customer di database laundry_apps yang telah dibuat pada praktikum sebelumnya

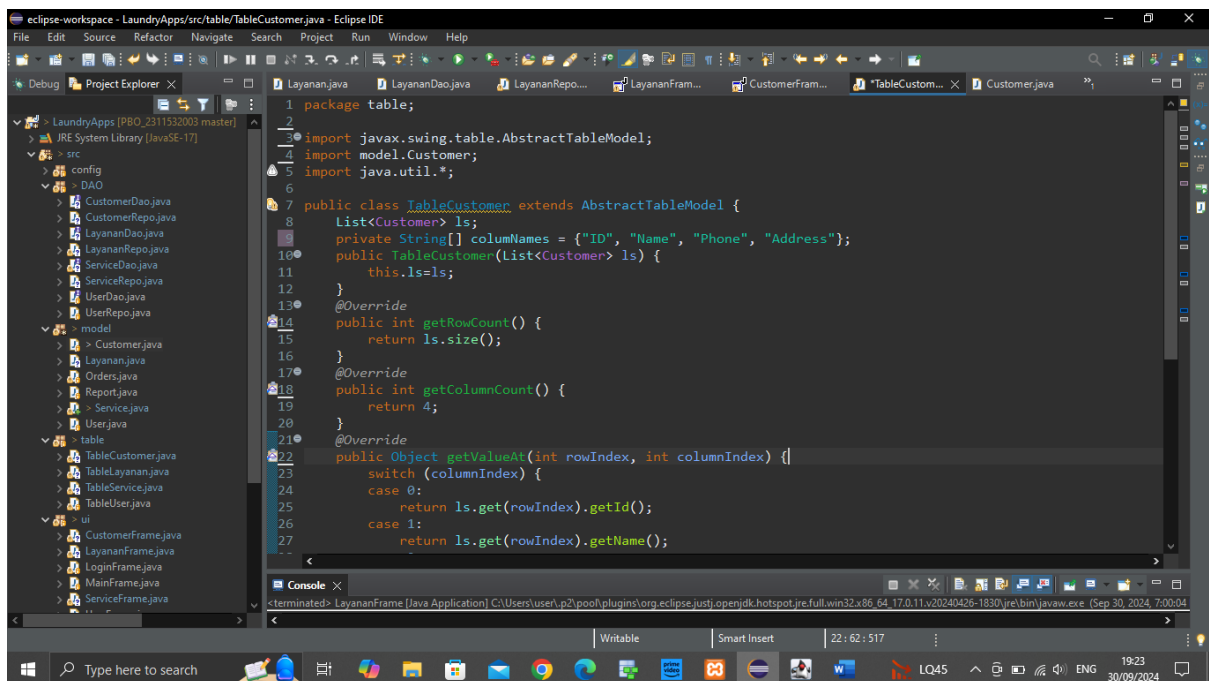


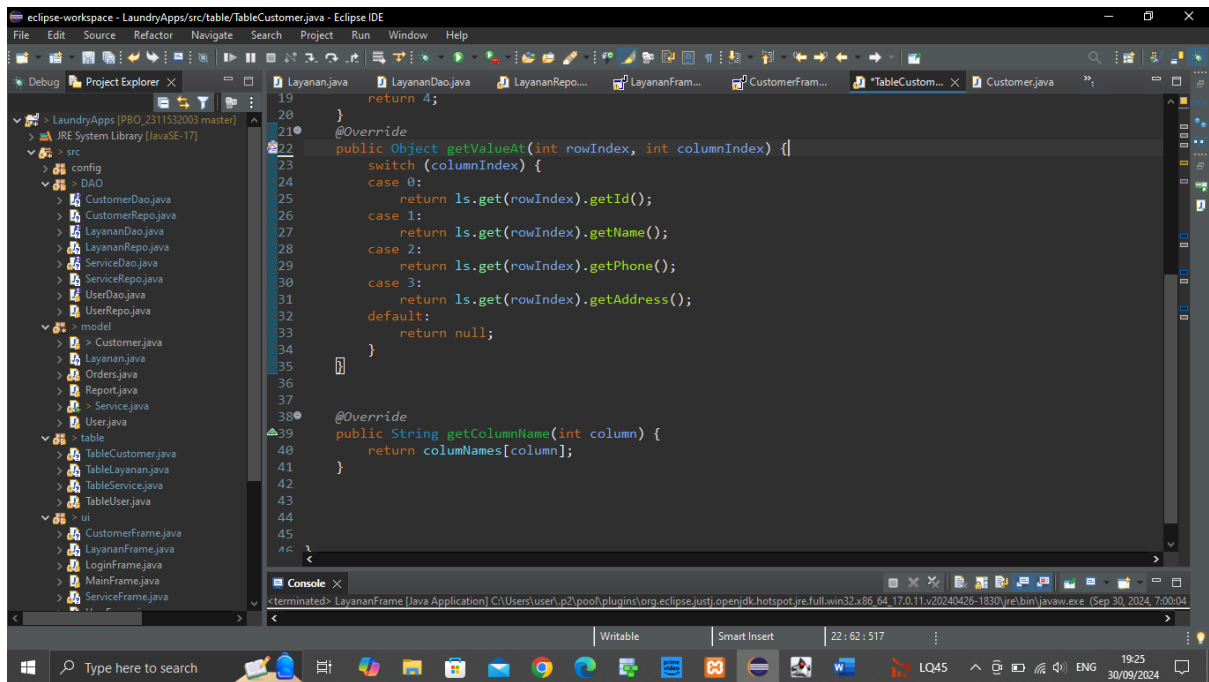
c. Buat juga table layanan



2. Membuat CRUD Customer

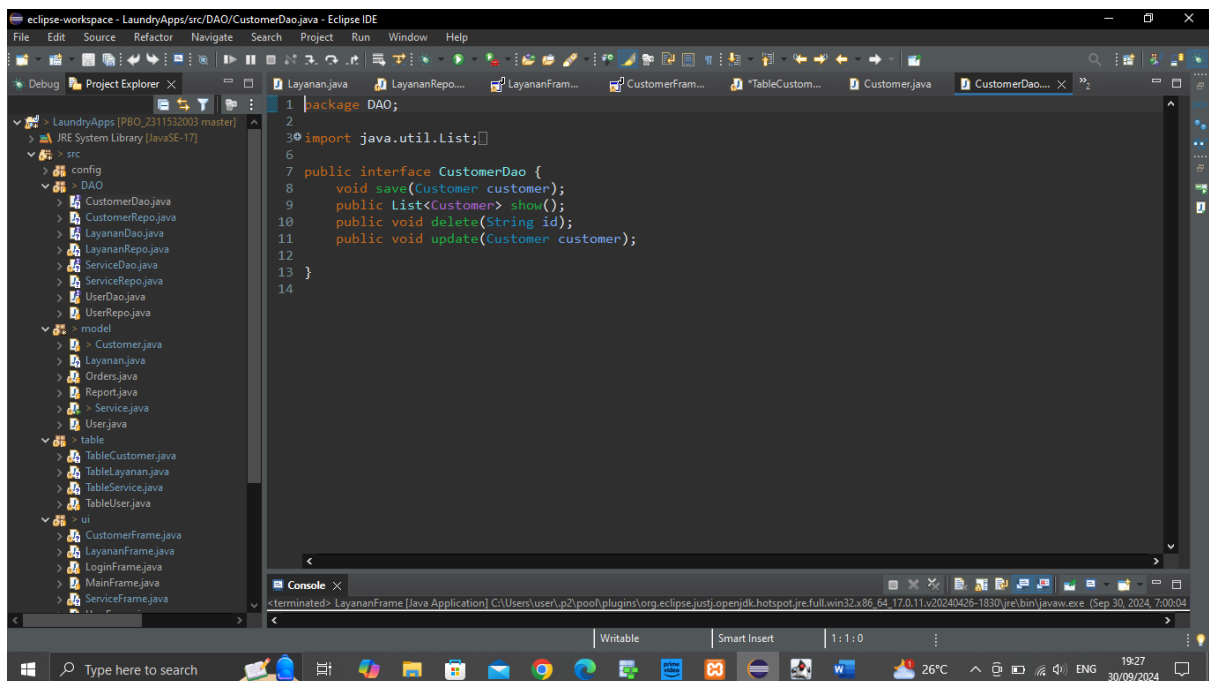
- Membuat class Table Model dengan nama TableCustomer seperti Table Model untuk User pada praktikum sebelumnya





b. Membuat Interface DAO

Buat interface baru pada package DAO dengan nama CustomerDao yang mana akan digunakan pada CustomerRepo



c. Membuat Class Repo untuk Customer dengan nama CustomerRepo dan implementasikan interface CustomerDao

d. Inisialisasi koneksi ke database dan buat command yang akan dimasukkan ke database untuk insert, select, update, dan delete

```

1 package DAO;
2
3 import java.sql.Connection;
4
5
6
7 public class CustomerRepo implements CustomerDao {
8     private Connection connection;
9     final String insert = "INSERT INTO customer (name, phone, address) VALUES (?, ?, ?)";
10    final String select = "SELECT * FROM customer;";
11    final String delete = "DELETE FROM customer WHERE id=?;";
12    final String update = "UPDATE customer SET name=?, phone=?, address=? WHERE id=?;";
13
14    public CustomerRepo() {
15        connection = Database.koneksi();
16    }
17
18 }

```

e. Buat method save menggunakan override

```

27
28 @Override
29 public void save(Customer customer) {
30     PreparedStatement st = null;
31     try {
32         st = connection.prepareStatement(insert);
33         st.setString(1, customer.getName());
34         st.setString(2, customer.getPhone());
35         st.setString(3, customer.getAddress());
36         st.executeUpdate();
37     } catch (SQLException e) {
38         e.printStackTrace();
39     } finally {
40         if (st != null) {
41             try {
42                 st.close();
43             } catch (SQLException e) {
44                 e.printStackTrace();
45             }
46         }
47     }
48 }

```

f. Buat method show menggunakan override

```

50
51 @Override
52 public List<Customer> show() {
53     List<Customer> ls = null;
54     try {
55         ls = new ArrayList<Customer>();
56         Statement st = connection.createStatement();
57         ResultSet rs = st.executeQuery(select);
58         while (rs.next()) {
59             Customer customer = new Customer();
60             customer.setId(rs.getString("id"));
61             customer.setName(rs.getString("name"));
62             customer.setPhone(rs.getString("phone"));
63             customer.setAddress(rs.getString("address"));
64             ls.add(customer);
65         }
66     } catch (SQLException e) {
67         Logger.getLogger(CustomerDao.class.getName()).log(Level.SEVERE, null, e);
68     }
69     return ls;
70 }
71

```

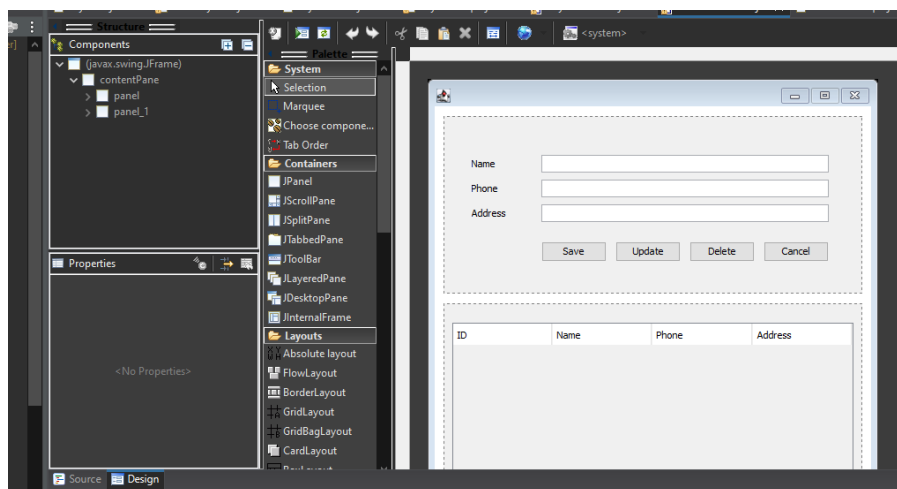
g. Buat juga method delete dan update

```

71
72 @Override
73 public void delete(String id) {
74     PreparedStatement st = null;
75     try {
76         st = connection.prepareStatement(delete);
77         st.setString(1, id);
78         st.executeUpdate();
79     } catch (SQLException e) {
80         e.printStackTrace();
81     } finally {
82         try {
83             st.close();
84         } catch (SQLException e) {
85             e.printStackTrace();
86         }
87     }
88 }
89
90 @Override
91 public void update(Customer customer) {
92     PreparedStatement st = null;
93     try {
94         st = connection.prepareStatement(update);
95         st.setString(1, customer.getName());
96         st.setString(2, customer.getPhone());
97         st.setString(3, customer.getAddress());
98         st.setString(4, customer.getId());
99         st.executeUpdate();
100     } catch (SQLException e) {
101         e.printStackTrace();
102     } finally {
103         try {
104             st.close();
105         } catch (SQLException e) {
106             e.printStackTrace();
107         }
108     }
109 }

```

h. Buat CustomerFrame menggunakan JFrame dan buat desainnya



i. Buat method dengan nama loadTable untuk memuat data dari database dan menampilkannya pada Jtable

```

55
56 CustomerRepo cus = new CustomerRepo();
57 List<Customer> lc;
58 public String id;
59
60 public void loadTable() {
61     lc = cus.show();
62     TableCustomer tc = new TableCustomer(lc);
63     tableCustomers.setModel(tc);
64     tableCustomers.getTableHeader().setVisible(true);
65 }
66

```

j. Buat method reset untuk mengosongkan field setelah selesai di proses

```

198 public void reset() {
199     txtName.setText("");
200     txtPhone.setText("");
201     txtAddress.setText("");
202 }

```

k. Tambahkan action untuk tombol save

```

    JButton btnSave = new JButton("Save");
    btnSave.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            Customer customer = new Customer();
            customer.setName(txtName.getText());
            customer.setPhone(txtPhone.getText());
            customer.setAddress(txtAddress.getText());
            cus.save(customer);
            reset();
            loadTable();
        }
    });

```

l. Tambahkan action untuk tombol update, delete, dan cancel

```

125
126
127 JButton btnUpdate = new JButton("Update");
128 btnUpdate.addActionListener(new ActionListener() {
129     public void actionPerformed(ActionEvent e) {
130         Customer customer = new Customer();
131         customer.setName(txtName.getText());
132         customer.setPhone(txtPhone.getText());
133         customer.setAddress(txtAddress.getText());
134         customer.setId(id);
135         cus.update(customer);
136         reset();
137         loadTable();
138     }
139 });
140 btnUpdate.setBounds(195, 142, 74, 23);
141 panel.add(btnUpdate);
142 JButton btnDelete = new JButton("Delete");
143 btnDelete.addActionListener(new ActionListener() {
144     public void actionPerformed(ActionEvent e) {
145         if(id != null) {
146             cus.delete(id);
147             reset();
148             loadTable();
149         } else {
150             JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
151         }
152     }
153 });
154 btnDelete.setBounds(279, 142, 74, 23);
155 panel.add(btnDelete);
156 JButton btnCancel = new JButton("Cancel");
157 btnCancel.addActionListener(new ActionListener() {
158     public void actionPerformed(ActionEvent e) {
159         reset();
160     }
161 });

```

m. Tambahkan event handler untuk mouseclicked pada Jtable untuk mengambil data dari tabel ketika di klik dan menampilkannya di field

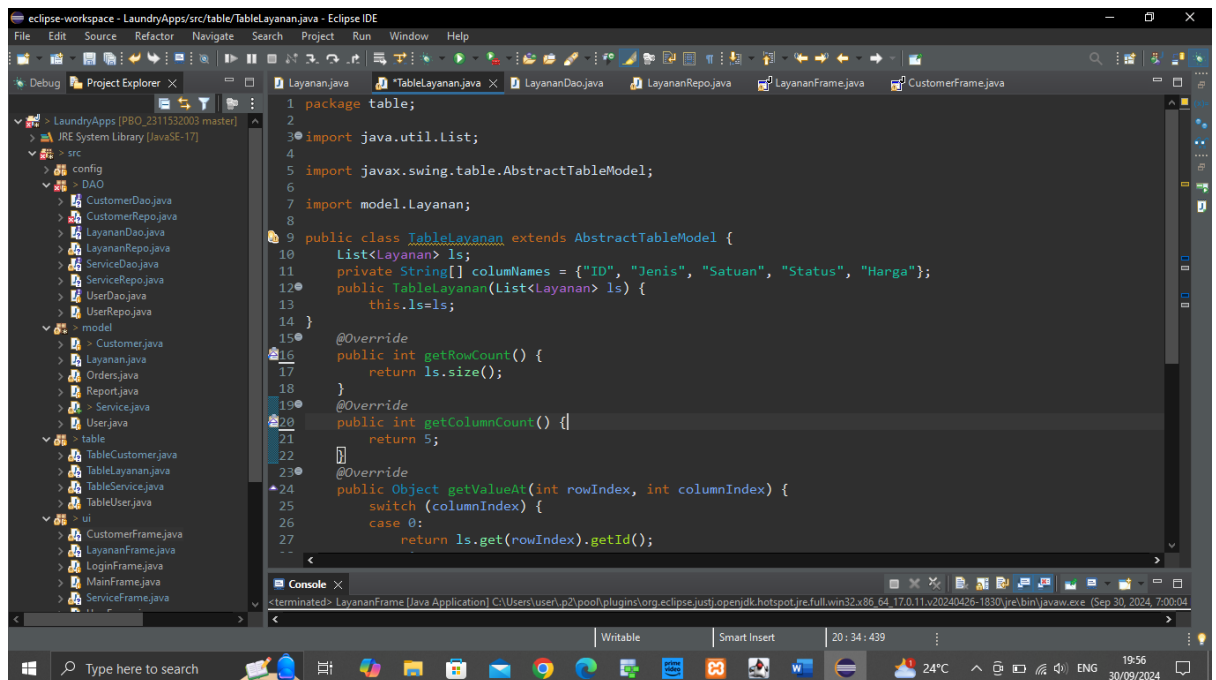
```

170 tableCustomers = new JTable();
171 tableCustomers.addMouseListener(new MouseAdapter() {
172     @Override
173     public void mouseClicked(MouseEvent e) {
174         id = tableCustomers.getValueAt(tableCustomers.getSelectedRow(),0).toString();
175         txtName.setText(tableCustomers.getValueAt(tableCustomers.getSelectedRow(), 1).toString());
176         txtPhone.setText(tableCustomers.getValueAt(tableCustomers.getSelectedRow(),2).toString());
177         txtAddress.setText(tableCustomers.getValueAt(tableCustomers.getSelectedRow(),3).toString());
178     }
179 }
180 });

```


3. Membuat CRUD Layanan

a. Membuat class Table Model dengan nama TableLayanan



The screenshot shows the Eclipse IDE with the 'TableLayanan.java' file open. The code defines a class that extends 'AbstractTableModel'. It includes imports for 'java.util.List' and 'javax.swing.table.AbstractTableModel'. The class has a private list 'ls' and a private array 'columnNames' with values 'ID', 'Jenis', 'Satuan', 'Status', and 'Harga'. The constructor 'TableLayanan(List<Layanan> ls)' initializes 'this.ls=ls;'. The 'getRowCount()' method returns 'ls.size()'. The 'getColumnCount()' method returns '5'. The 'getValueAt(int rowIndex, int columnIndex)' method uses a switch statement to return the corresponding value from 'ls.get(rowIndex)' based on the column index: 0 for ID, 1 for Jenis, 2 for Satuan, 3 for Status, and 4 for Harga. If the column index is 0, it returns 'ls.get(rowIndex).getId()'. The 'main' method is not visible in this snippet.

```
1 package table;
2
3 import java.util.List;
4
5 import javax.swing.table.AbstractTableModel;
6
7 import model.Layanan;
8
9 public class TableLayanan extends AbstractTableModel {
10     List<Layanan> ls;
11     private String[] columnNames = {"ID", "Jenis", "Satuan", "Status", "Harga"};
12     public TableLayanan(List<Layanan> ls) {
13         this.ls=ls;
14     }
15
16     @Override
17     public int getRowCount() {
18         return ls.size();
19     }
20
21     @Override
22     public int getColumnCount() {
23         return 5;
24     }
25
26     @Override
27     public Object getValueAt(int rowIndex, int columnIndex) {
28         switch (columnIndex) {
29             case 0:
30                 return ls.get(rowIndex).getId();
31             case 1:
32                 return ls.get(rowIndex).getJenis();
33             case 2:
34                 return ls.get(rowIndex).getSatuan();
35             case 3:
36                 return ls.get(rowIndex).getStatus();
37             case 4:
38                 return ls.get(rowIndex).getHarga();
39             default:
40                 return null;
41         }
42     }
43
44     @Override
45     public String getColumnName(int column) {
46         return columnNames[column];
47     }
48 }
```

b. Buat LayananDao

```
1 package DAO;
2
3 import java.util.List;
4
5 import model.Layanan;
6
7 public interface LayananDao {
8     void save(Layanan layanan);
9     public List<Layanan> show();
10    public void delete(String id);
11    public void update(Layanan layanan);
12 }
13
14
```

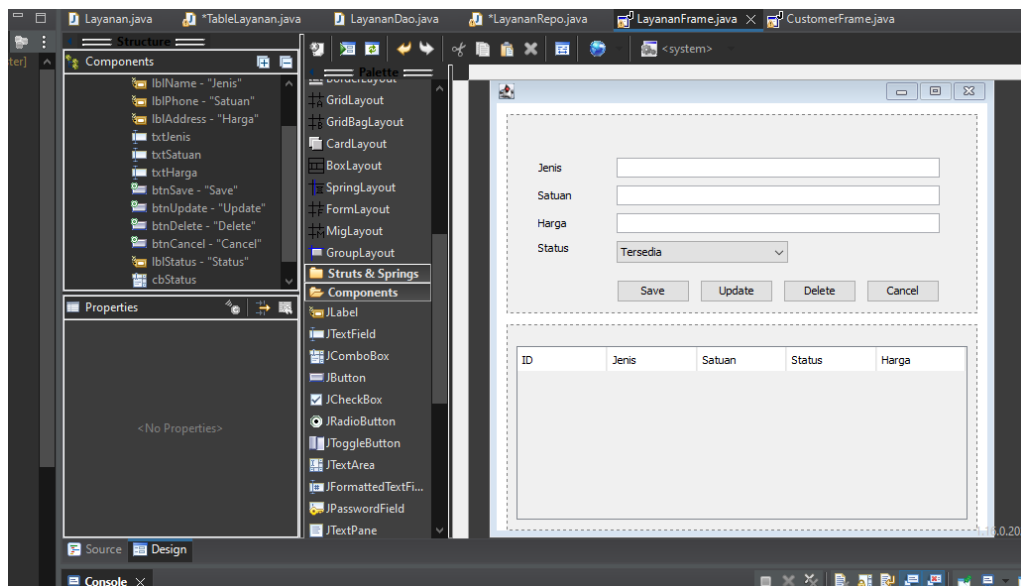
c. Buat LayananRepo seperti pada CustomerRepo

```
14 import model.Layanan;
15 public class LayananRepo implements LayananDao {
16     private Connection connection;
17     final String insert = "INSERT INTO layanan (jenis, satuan, status, harga) VALUES (?, ?, ?, ?)";
18     final String select = "SELECT * FROM layanan";
19     final String delete = "DELETE FROM layanan WHERE id=?";
20     final String update = "UPDATE layanan SET jenis=?, satuan=?, status=?, harga=? WHERE id=?";
21
22     public LayananRepo() {
23         connection = Database.koneksi();
24     }
25
26     @Override
27     public void save(Layanan layanan) {
28         PreparedStatement st = null;
29         try {
30             st = connection.prepareStatement(insert);
31             st.setString(1, layanan.getJenis());
32             st.setString(2, layanan.getSatuan());
33             st.setString(3, layanan.getStatus());
34             st.setDouble(4, layanan.getHarga());
35             st.executeUpdate();
36         } catch (SQLException e) {
37             e.printStackTrace();
38         } finally {
39             if (st != null) {
40                 try {
41                     st.close();
42                 } catch (SQLException e) {
43                     e.printStackTrace();
44                 }
45             }
46         }
47     }
48 }
```

```
Layanan.java *TableLayanan.java LayananDao.java *LayananRepo.java x LayananF
49
50 @Override
51 public List<Layanan> show() {
52     List<Layanan> ls=null;
53     try {
54         ls = new ArrayList<Layanan>();
55         Statement st = connection.createStatement();
56         ResultSet rs = st.executeQuery(select);
57         while(rs.next()) {
58             Layanan layanan = new Layanan();
59             layanan.setId(rs.getString("id"));
60             layanan.setJenis(rs.getString("jenis"));
61             layanan.setSatuan(rs.getString("satuan"));
62             layanan.setStatus(rs.getString("status"));
63             layanan.setHarga(rs.getDouble("harga"));
64             ls.add(layanan);
65         }
66     } catch (SQLException e) {
67         Logger.getLogger(LayananDao.class.getName()).log(Level.SEVERE,null,e);
68     }
69     return ls;
70 }
71 @Override
72 public void delete(String id) {
73     PreparedStatement st = null;
74     try {
75         st = connection.prepareStatement(delete);
76         st.setString(1, id);
77         st.executeUpdate();
78     } catch (SQLException e) {
79         e.printStackTrace();
80     } finally {
81         try {
82             st.close();
83         } catch (SQLException e) {
84             e.printStackTrace();
85         }
86     }
87 }
88
```

```
Layanan.java *TableLayanan.java LayananDao.java *LayananRepo.java x La
87
88
89 @Override
90 public void update(Layanan layanan) {
91     PreparedStatement st = null;
92     try {
93         st = connection.prepareStatement (update);
94         st.setString(1, layanan.getJenis());
95         st.setString(2, layanan.getSatuan());
96         st.setString(3, layanan.getStatus());
97         st.setDouble(4, layanan.getHarga());
98         st.setString(5, layanan.getId());
99         st.executeUpdate();
100     } catch (SQLException e) {
101         e.printStackTrace();
102     } finally {
103         try {
104             st.close();
105         } catch (SQLException e) {
106             e.printStackTrace();
107         }
108     }
109 }
110
111
112 }
113
```

d. Buat LayananFrame menggunakan JFrame dan buat desainnya



e. Buat method loadTable dan method reset seperti sebelumnya

```

52
53
54     LayananRepo cus = new LayananRepo();
55     List<Layanan> lc;
56     public String id;
57
58     public void loadTable() {
59         lc = cus.show();
60         TableLayanan tc = new TableLayanan(lc);
61         tableLayanan.setModel(tc);
62         tableLayanan.getTableHeader().setVisible(true);
63     }
64
65     public LayananFrame() {

```

```

20     public void reset() {
21         txtJenis.setText("");
22         txtSatuan.setText("");
23         cbStatus.setSelectedItem(0);
24         txtHarga.setText("");
25     }
26

```

f. Berikan action pada tombol save

```

105     txtHarga.setBounds(111, 100, 326, 20);
106     panel.add(txtHarga);
107
108     JButton btnSave = new JButton("Save");
109     btnSave.addActionListener(new ActionListener() {
110         public void actionPerformed(ActionEvent e) {
111             Layanan layanan = new Layanan();
112             layanan.setJenis(txtJenis.getText());
113             layanan.setSatuan(txtSatuan.getText());
114             layanan.setStatus(cbStatus.getSelectedItem().toString());
115             try {
116                 double harga = Double.parseDouble(txtHarga.getText());
117                 layanan.setHarga(harga);
118             } catch (NumberFormatException ex) {
119                 JOptionPane.showMessageDialog(null, "Input untuk harga tidak valid. "
120                     + "Masukkan angka yang benar.");
121             }
122             return;
123             cus.save(layanan);
124             reset();
125             loadTable();
126         }
127     });
128

```

g. Berikan action pada tombol upadte

```
132 JButton btnUpdate = new JButton("Update");
133 btnUpdate.addActionListener(new ActionListener() {
134     public void actionPerformed(ActionEvent e) {
135         Layanan layanan = new Layanan();
136         layanan.setJenis(txtJenis.getText());
137         layanan.setSatuan(txtSatuan.getText());
138         layanan.setStatus(cbStatus.getSelectedItem().toString());
139         try {
140             double harga = Double.parseDouble(txtHarga.getText());
141             layanan.setHarga(harga);
142         } catch (NumberFormatException ex) {
143             JOptionPane.showMessageDialog(null, "Input untuk harga tidak valid. "
144                 + "Masukkan angka yang benar.");
145             return;
146         }
147         layanan.setId(id);
148         cus.update(layanan);
149         reset();
150         loadTable();
151     }
152 });
153 btnUpdate.setBounds(195, 167, 74, 23);
```

h. Berikan action pada tombol delete dan cancel

```
155 JButton btnDelete = new JButton("Delete");
156 btnDelete.addActionListener(new ActionListener() {
157     public void actionPerformed(ActionEvent e) {
158         if(id != null) {
159             cus.delete(id);
160             reset();
161             loadTable();
162         } else {
163             JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan di hapus");
164         }
165     }
166 });
167 btnDelete.setBounds(279, 167, 74, 23);
168 panel.add(btnDelete);
169
170 JButton btnCancel = new JButton("Cancel");
171 btnCancel.addActionListener(new ActionListener() {
172     public void actionPerformed(ActionEvent e) {
173         reset();
174     }
175 });
176 btnCancel.setBounds(353, 167, 74, 23);
177
```

i. Tambahkan event handler untuk mouseclicked pada Jtable untuk mengambil data dari tabel ketika di klik dan menampilkannya di field

```
197
198 tableLayanan = new JTable();
199 tableLayanan.addMouseListener(new MouseAdapter() {
200     @Override
201     public void mouseClicked(MouseEvent e) {
202         id = tableLayanan.getValueAt(tableLayanan.getSelectedRow(),0).toString();
203         txtJenis.setText(tableLayanan.getValueAt(tableLayanan.getSelectedRow(), 1).toString());
204         txtSatuan.setText(tableLayanan.getValueAt(tableLayanan.getSelectedRow(),2).toString());
205         cbStatus.setSelectedItem(tableLayanan.getValueAt(tableLayanan.getSelectedRow(), 3).toString());
206         txtHarga.setText(tableLayanan.getValueAt(tableLayanan.getSelectedRow(),4).toString());
207     }
208 });
209
```

C. HASIL

1. Customer

Project Explorer X Layanana.java TableLayanan.j... LayananaD...

Name

Phone

Address

Save Update Delete Cancel

ID	Name	Phone	Address
2	fgdfgd	4453434	fgdd
3	Dila	081111111111	Padang

2. Layanan

Project Explorer X Layanana.java TableLayanan.j... LayananaD...

Jenis

Satuan

Harga

Status

Save Update Delete Cancel

ID	Jenis	Satuan	Status	Harga
1	cuci	KG	Tersedia	4000.0
2	cuci lipat	kg	Tersedia	4000.0
3	cuci lipat	satuan	Tersedia	4500.0
4	cuci setrika	kg	Tersedia	6000.0
5	Setrika saja	kg	Tersedia	4000.0