

LAPORAN PRAKTIKUM 8 PBO

DESIGN PATTERN



Disusun Oleh:

INDAH SYAHFITRI

2311532016

Dosen Pembimbing: Afdhal Dinilhak, S.Komp., M.Kom.

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

A. PENDAHULUAN

Design Patterns adalah solusi standar yang telah terbukti (teruji) untuk menyelesaikan masalah umum yang sering terjadi dalam pengembangan perangkat lunak. Design Patterns bukanlah hal yang bisa ditranslate ke dalam kode program secara langsung, melainkan hanya template atau panduan cara menyelesaikan masalah. Terdapat banyak sekali desain pattern, dan untuk implementasi ke dalam kode programnya mungkin akan sedikit berbeda tergantung teknologi dan bahasa pemrograman yang digunakan. Dengan memahami design pattern kita akan lebih mudah dalam mengatasi masalah yang terjadi dalam pembuatan aplikasi/software. Karena kita sudah tahu solusi-solusi umum yang harus dilakukan, berbeda jika belum mengerti design pattern maka kita mencari solusi dari permasalahan yang dihadapi dengan mencoba coba solusi versi kita sendiri padahal sebenarnya sudah ada solusi yang sudah terbukti untuk masalah tersebut. Adapun manfaat dari menggunakan design pattern dalam adalah sebagai berikut:

1. Reusability (Penggunaan Ulang), membantu mengurangi pengulangan logika karena solusi tersebut dapat diterapkan di berbagai situasi.
2. Efisiensi, membantu menyelesaikan masalah lebih cepat dengan solusi yang sudah teruji.
3. Kolaborasi, memberikan cara komunikasi yang lebih mudah antar software developer karena banyak software developer sudah memahami pola-pola ini.
4. Scalability, membantu membuat kode yang dapat diskalakan dan mudah diperluas.
5. Maintainability, membuat kode lebih mudah dipahami dan diperbaiki ketika terjadi perubahan.

Jenis-jenis Design Pattern yakni: 1) Creational Patterns, berfokus pada cara membuat objek agar lebih fleksibel dan terstruktur. Creational pattern terdiri dari Singleton, Factory Method, Abstract Factory, Builder, Prototype. 2) Structural Patterns, membantu menyusun class dan object agar mudah dikelola dan membentuk struktur yang kompleks. Structural pattern terdiri dari Adapter, Decorator, Proxy, Facade, Composite, Bridge. 3) Behavioral Patterns, berfokus pada cara object berinteraksi satu sama lain. Behavioral pattern terdiri dari Observer, Strategy, Command, State, Template Method, Mediator, Chain of Responsibility.

Singleton Pattern merupakan pattern yang berfungsi untuk memastikan hanya satu instance dari suatu class yang dapat diakses guna menghindari pembuatan instance duplikat.

Berikut adalah contoh penggunaan Singleton Pattern pada kasus insert data ke database di mana sebelum melakukan insert data kita perlu untuk membuat koneksi ke database terlebih dahulu. Dengan menggunakan Singleton Pattern kita hanya perlu membuat koneksi ke database satu kali saja dan untuk selanjutnya koneksi tersebut akan digunakan berkali-kali.

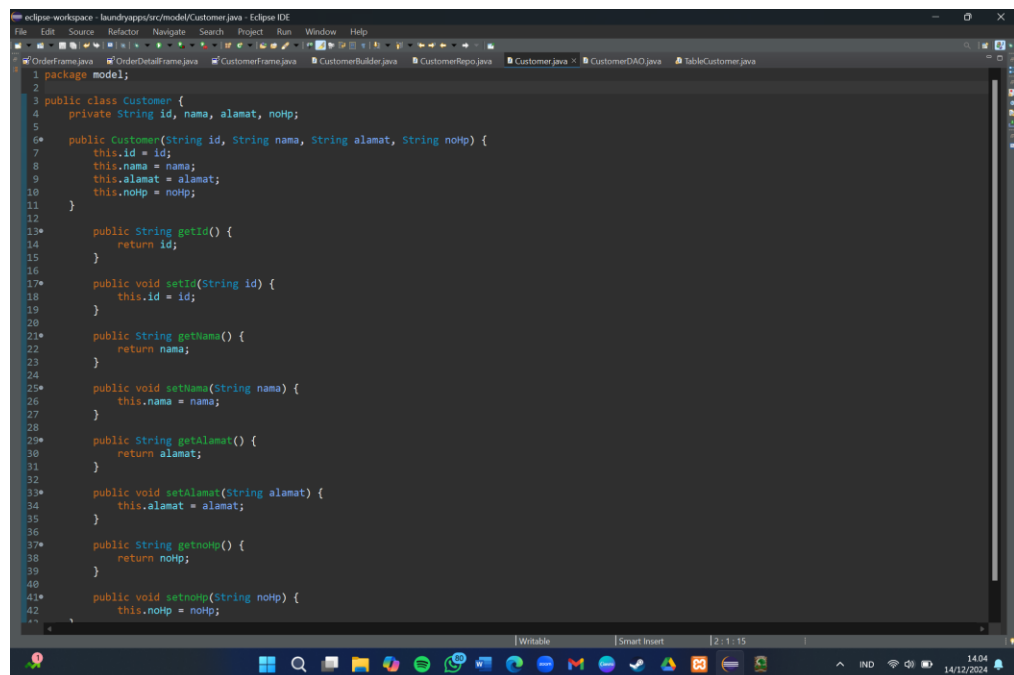
Builder pattern merupakan salah satu design pattern paling banyak digunakan dalam software development. Builder Pattern digunakan untuk memisahkan logika konstruksi sebuah objek dari representasi final objek tersebut. Builder Pattern dirancang untuk membantu dalam pembuatan objek kompleks dengan memberikan kontrol yang lebih besar atas proses konstruksinya dan memastikan bahwa objek tersebut dibuat secara konsisten. Pattern ini sangat berguna saat sebuah objek yang memiliki banyak atribut, beberapa di antaranya opsional, atau saat proses pembuatannya membutuhkan beberapa langkah.

B. TUJUAN PRAKTIKUM

1. Mahasiswa mampu memahami dan menggunakan beberapa jenis design pattern (singleton dan builder pattern) pada kasus CRUD sebuah aplikasi laundry.

C. LANGKAH-LANGKAH

1. Perbarui class Customer di dalam package model. Class Customer memiliki beberapa atribut dan sebuah constructor seperti berikut.

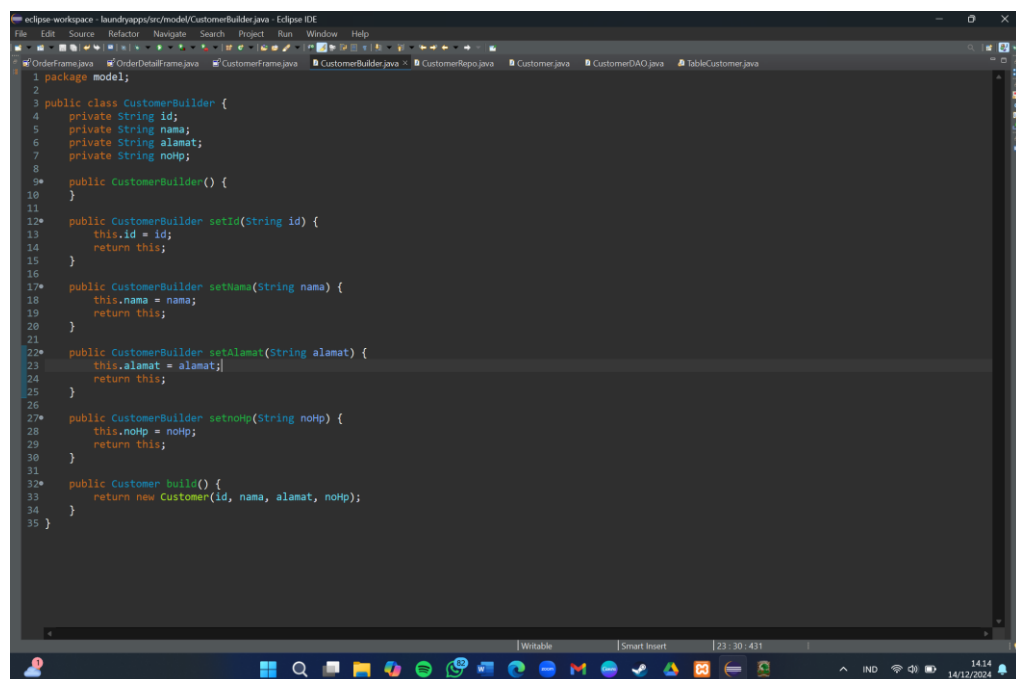


```
1 package model;
2
3 public class Customer {
4     private String id, nama, alamat, noHp;
5
6     public Customer(String id, String nama, String alamat, String noHp) {
7         this.id = id;
8         this.nama = nama;
9         this.alamat = alamat;
10        this.noHp = noHp;
11    }
12
13    public String getId() {
14        return id;
15    }
16
17    public void setId(String id) {
18        this.id = id;
19    }
20
21    public String getNama() {
22        return nama;
23    }
24
25    public void setNama(String nama) {
26        this.nama = nama;
27    }
28
29    public String getAlamat() {
30        return alamat;
31    }
32
33    public void setAlamat(String alamat) {
34        this.alamat = alamat;
35    }
36
37    public String getNoHp() {
38        return noHp;
39    }
40
41    public void setNoHp(String noHp) {
42        this.noHp = noHp;
43    }
44 }
```

Class Customer adalah representasi data pelanggan yang memiliki atribut id, nama, alamat, dan noHp untuk menyimpan informasi dasar pelanggan. Data ini

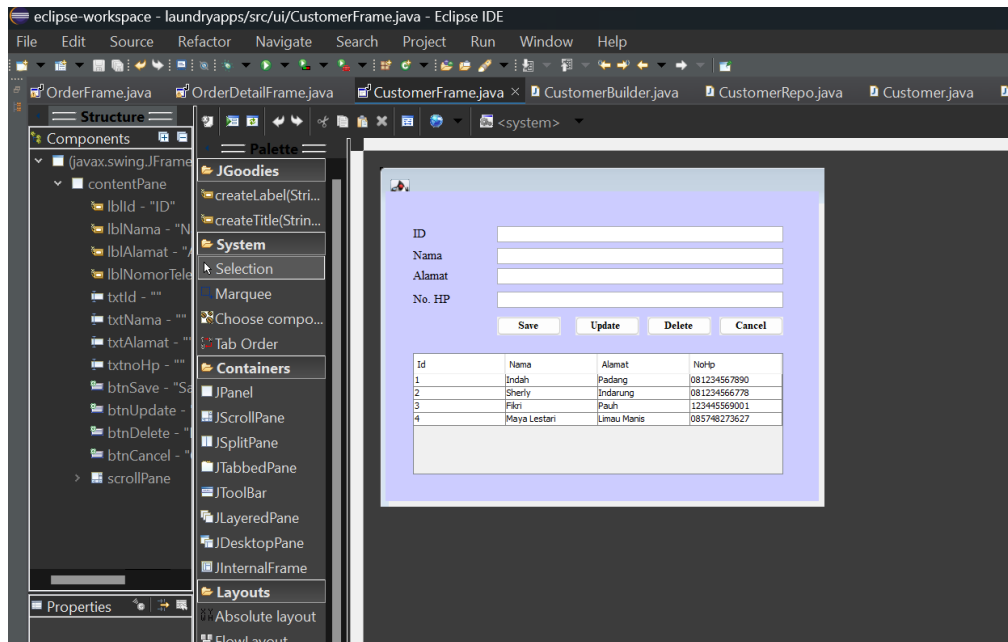
diinisialisasi melalui constructor dan bisa diakses atau diubah menggunakan metode getter dan setter. Class ini dirancang untuk mendukung operasi seperti tambah, ubah, hapus, atau ambil data pelanggan, sehingga memudahkan integrasi dengan database.

2. Buat sebuah class baru dengan nama CustomerBuilder di dalam package model. Class CustomerBuilder merupakan builder untuk membuat objek Customer. Dengan menggunakan Builder, kita dapat dengan mudah membuat objek yang memiliki kombinasi atribut yang berbeda tanpa harus membuat banyak constructor. Selanjutnya, copy semua atribut dari yang ada di class Customer ke dalam class CustomerBuilder dan buat method setter untuk masing-masing atribut tersebut. Di sini kita bisa memberikan default value untuk atribut yang ingin diberikan default value. Kemudian, buat sebuah method dengan nama build dengan return value adalah Customer.



```
1 package model;
2
3 public class CustomerBuilder {
4     private String id;
5     private String nama;
6     private String alamat;
7     private String noHp;
8
9     public CustomerBuilder() {
10    }
11
12    public CustomerBuilder setId(String id) {
13        this.id = id;
14        return this;
15    }
16
17    public CustomerBuilder setName(String nama) {
18        this.nama = nama;
19        return this;
20    }
21
22    public CustomerBuilder setAlamat(String alamat) {
23        this.alamat = alamat;
24        return this;
25    }
26
27    public CustomerBuilder setNoHp(String noHp) {
28        this.noHp = noHp;
29        return this;
30    }
31
32    public Customer build() {
33        return new Customer(id, nama, alamat, noHp);
34    }
35 }
```

3. Tampilan CRUD Customer menggunakan JFrame di dalam package ui yang diberi nama dengan CustomerFrame adalah sebagai berikut.



4. Pada package dao, buat interface dengan nama CustomerDao yang dapat mempermudah dalam melakukan proses CRUD Customer yang terdiri dari menampilkan data customer dari database, menambah, memperbarui, dan menghapus data customer.

```

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

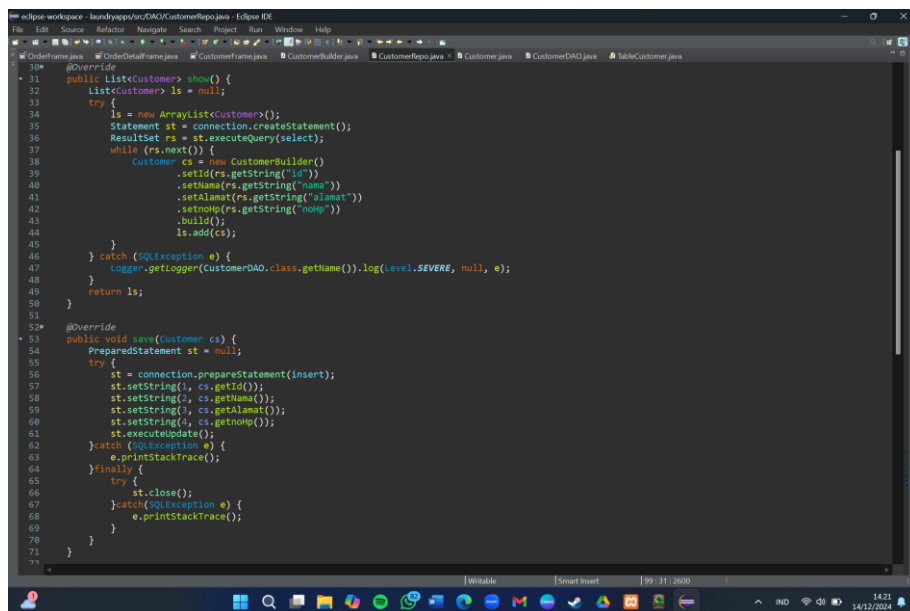
```

5. Perbarui class CustomerRepo di dalam package dao yang akan digunakan untuk mengimplementasikan CustomerDao yang telah dibuat sebelumnya.

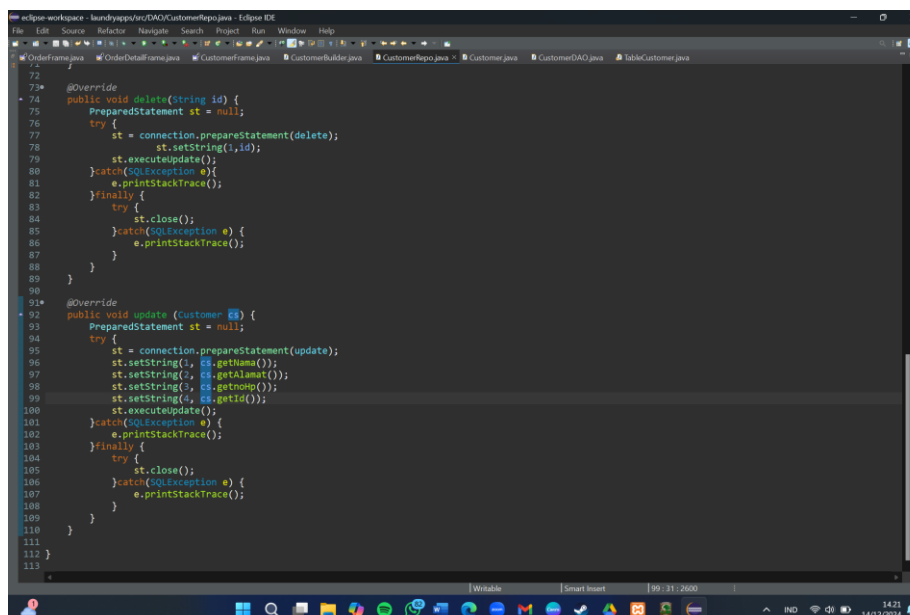
Tambahkan variabel bertipe string sebagai perintah query yang akan digunakan untuk proses CRUD dan juga instansiasi koneksi pada class constructor.

```
1 package DAO;
2
3 import java.sql.Connection;
17
18 public class CustomerRepo implements CustomerDAO {
19     private Connection connection;
20     final String insert = "INSERT INTO Customer (id, nama, alamat, noHp) VALUES (?, ?, ?, ?)";
21     final String select = "SELECT * FROM Customer;";
22     final String delete = "DELETE FROM Customer WHERE id=?";
23     final String update = "UPDATE Customer SET nama=?, alamat=?, noHp=? WHERE id=?";
24
25     public CustomerRepo() {
26         connection = Database.getConnection();
27     }
28 }
```

6. Selanjutnya, implementasikan masing-masing method yang terdapat pada interface CustomerDao yaitunya method save, update, delete, dan show. Berikut adalah implementasi method show pada class CustomerRepo.



```
30 @Override
31 public List<Customer> show() {
32     List<Customer> ls = null;
33     try {
34         ls = new ArrayList<Customer>();
35         Statement st = connection.createStatement();
36         ResultSet rs = st.executeQuery(select);
37         while (rs.next()) {
38             Customer cs = new CustomerBuilder()
39                 .setId(rs.getString("id"))
40                 .setName(rs.getString("nama"))
41                 .setAlamat(rs.getString("alamat"))
42                 .setNoHp(rs.getString("noHp"))
43                 .build();
44             ls.add(cs);
45         }
46     } catch (SQLException e) {
47         Logger.getLogger(CustomerDAO.class.getName()).log(Level.SEVERE, null, e);
48     }
49     return ls;
50 }
51
52 @Override
53 public void save(Customer cs) {
54     PreparedStatement st = null;
55     try {
56         st = connection.prepareStatement(insert);
57         st.setString(1, cs.getId());
58         st.setString(2, cs.getName());
59         st.setString(3, cs.getAlamat());
60         st.setString(4, cs.getNoHp());
61         st.executeUpdate();
62     } catch (SQLException e) {
63         e.printStackTrace();
64     } finally {
65         try {
66             st.close();
67         } catch (SQLException e) {
68             e.printStackTrace();
69         }
70     }
71 }
```



```
72
73 @Override
74 public void delete(String id) {
75     PreparedStatement st = null;
76     try {
77         st = connection.prepareStatement(delete);
78         st.setString(1, id);
79         st.executeUpdate();
80     } catch (SQLException e) {
81         e.printStackTrace();
82     } finally {
83         try {
84             st.close();
85         } catch (SQLException e) {
86             e.printStackTrace();
87         }
88     }
89 }
90
91 @Override
92 public void update(Customer cs) {
93     PreparedStatement st = null;
94     try {
95         st = connection.prepareStatement(update);
96         st.setString(1, cs.getName());
97         st.setString(2, cs.getAlamat());
98         st.setString(3, cs.getNoHp());
99         st.setString(4, cs.getId());
100         st.executeUpdate();
101     } catch (SQLException e) {
102         e.printStackTrace();
103     } finally {
104         try {
105             st.close();
106         } catch (SQLException e) {
107             e.printStackTrace();
108         }
109     }
110 }
111
112 }
113 }
```

Class ini menggunakan koneksi database dari Database dan menyediakan metode CRUD (Create, Read, Update, Delete). Metode save() digunakan untuk menyimpan data pelanggan baru ke database, sementara show() mengambil semua data pelanggan yang ada. Untuk menghapus data pelanggan berdasarkan ID, digunakan metode delete(). Sedangkan, metode update() memperbarui data pelanggan yang sudah ada dengan informasi baru. Semua operasi dilakukan menggunakan pernyataan SQL yang sudah disiapkan (PreparedStatement) untuk menjaga keamanan dan mengurangi risiko SQL Injection. Selain itu, setiap koneksi atau pernyataan ditutup di blok finally untuk menghindari kebocoran sumber daya. Class ini dirancang untuk mempermudah manipulasi data pelanggan dalam aplikasi.

7. Untuk menampilkan data customer dalam bentuk table pada CustomerFrame kita perlu membuat Class baru dengan nama TableCustomer pada package table. TableCustomer ini akan meng-extends class AbstractTableModel yang merupakan class bawaan dari Java. Berikut adalah kode program dari class TableCustomer.

```
1 package table;
2
3 import java.util.List;
4 import javax.swing.table.AbstractTableModel;
5 import model.Customer;
6
7 public class TableCustomer extends AbstractTableModel {
8     private List<Customer> ls;
9     private String[] columnNames = {"Id", "Nama", "Alamat", "NoHp"};
10
11     public TableCustomer(List<Customer> ls) {
12         this.ls = ls;
13     }
14
15     @Override
16     public int getRowCount() {
17         return ls.size();
18     }
19
20     @Override
21     public int getColumnCount() {
22         return 4;
23     }
24
25     @Override
26     public String getColumnName(int column) {
27         return columnNames[column];
28     }
29 }
```

```

30•  @Override
31  public Object getValueAt(int rowIndex, int columnIndex) {
32      switch (columnIndex) {
33          case 0:
34              return ls.get(rowIndex).getId();
35          case 1:
36              return ls.get(rowIndex).getNama();
37          case 2:
38              return ls.get(rowIndex).getAlamat();
39          case 3:
40              return ls.get(rowIndex).getnoHp();
41          default:
42              return null;
43      }
44  }
45•  public Customer getCostumerAt(int rowIndex) {
46
47      return ls.get(rowIndex);
48  }
49  }
50 }

```

Class ini mewarisi AbstractTableModel dan menggunakan daftar (List) dari objek Customer sebagai sumber datanya. Ada empat kolom dalam tabel ini: ID, Nama, Alamat, dan NoHp, yang didefinisikan dalam array columnNames. Metode getRowCount() dan getColumnCount() digunakan untuk menentukan jumlah baris dan kolom dalam tabel, sedangkan getColumnName() mengembalikan nama kolom berdasarkan indeksinya. Data untuk setiap sel diambil melalui getValueAt(), yang memetakan kolom dan baris ke atribut Customer. Selain itu, metode getCustomerAt() memungkinkan pengambilan objek Customer berdasarkan baris yang dipilih. Class ini dirancang untuk mempermudah pengelolaan dan penampilan data pelanggan dalam tabel.

8. Di CustomerFrame, tambahkan instance id, ls yang akan menampung data List Customer, dan objek customerRepo. Selain itu, kita juga perlu membuat method untuk load data customer dan memasukkannya ke TableCustomer, serta method untuk me-reset form input customer.


```

28 public class CustomerFrame extends JFrame {
29
30     private static final long serialVersionUID = 1L;
31     private JPanel contentPane;
32     private JTextField txtId;
33     private JTextField txtNama;
34     private JTextField txtAlamat;
35     private JTextField txtNoHp;
36     private JTable tableCustomer;
37
38     public String id;
39     List<Customer> ls;
40     CustomerRepo cst = new CustomerRepo();
41
42     public static void main(String[] args) {
43        .EventQueue.invokeLater(new Runnable() {
44             public void run() {
45                 try {
46                     CustomerFrame frame = new CustomerFrame();
47                     frame.setVisible(true);
48                     frame.loadTable();
49                 } catch (Exception e) {
50                     e.printStackTrace();
51                 }
52             }
53         });
54     }
55
56     public void loadTable() {
57         ls = cst.show();
58         TableCustomer tc = new TableCustomer(ls);
59         tableCustomer.setModel(tc);
60         tableCustomer.getTableHeader().setVisible(true);
61     }
62
63     public void reset() {
64         txtId.setText("");
65         txtNama.setText("");
66         txtAlamat.setText("");
67         txtNoHp.setText("");
68         id = null;

```

9. Tambahkan kode berikut pada tabel customer pada CustomerFrame untuk menampilkan data customer ke dalam table.

```

tableCustomer = new JTable();
scrollPane.setViewportViewView(tableCustomer);
tableCustomer.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int selectedRow = tableCustomer.getSelectedRow();
        if (selectedRow != -1) {
            id = tableCustomer.getValueAt(selectedRow, 0).toString();
            txtId.setText(id);
            txtNama.setText(tableCustomer.getValueAt(selectedRow, 1).toString());
            txtAlamat.setText(tableCustomer.getValueAt(selectedRow, 2).toString());
            txtNoHp.setText(tableCustomer.getValueAt(selectedRow, 3).toString());
        }
    }
});
loadTable();

```

Kode ini berfungsi untuk menampilkan data pelanggan dalam sebuah tabel tableCustomer yang berada dalam scrollPane. Ketika baris tertentu pada tabel diklik, sebuah event listener MouseListener akan memproses aksi tersebut. Metode mouseClicked() mengambil indeks baris yang diklik menggunakan getSelectedRow(). Jika ada baris yang dipilih (indeks tidak -1), data dari baris tersebut diambil berdasarkan kolomnya dengan getValueAt(). Data ini kemudian diisi ke dalam txtId, txtNama, txtAlamat, dan txtNoHp, yang merupakan input field pada form. Selain itu, ID pelanggan dari baris yang diklik juga disimpan dalam variabel id untuk mempermudah manipulasi data seperti update atau delete. Terakhir, loadTable() dipanggil untuk memuat data ke tabel saat aplikasi dijalankan.

10. Selanjutnya, untuk menambahkan customer baru dan menyimpan datanya ke database ketika tombol save diklik, silahkan tambahkan kode berikut.

```
JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Customer customer = new CustomerBuilder()
            .setId(txtId.getText())
            .setNama(txtNama.getText())
            .setAlamat(txtAlamat.getText())
            .setnoHp(txtnoHp.getText())
            .build();
        cst.save(customer);
        reset();
        loadTable();
    }
});
```

Kode ini membuat tombol Save yang, saat diklik, akan mengambil data dari field input, membuat objek Customer, lalu menyimpannya ke database. Setelah itu, form direset dan tabel data diperbarui. gPada proses ini kita menggunakan builder pattern untuk membuat object customer yang akan menampung data inputan pengguna sebelum disimpan ke database.

11. Untuk memperbarui data yang telah tersimpan ke database, tambahkan kode berikut.

```
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (id == null || id.isEmpty()) {
            JOptionPane.showMessageDialog(null, "Silakan pilih data yang ingin diupdate.");
            return;
        }
        Customer customer = new CustomerBuilder()
            .setId(id)
            .setNama(txtNama.getText())
            .setAlamat(txtAlamat.getText())
            .setnoHp(txtnoHp.getText())
            .build();
        cst.update(customer);
        reset();
        loadTable();
    }
});
```

Kode ini membuat tombol Update yang, saat diklik, memeriksa apakah data yang akan diupdate sudah dipilih (id tidak kosong). Jika sudah, data diambil dari form, objek Customer dibuat, dan data tersebut diperbarui di database. Setelah itu, form direset dan tabel data diperbarui. Jika data belum dipilih, pesan peringatan akan muncul.

12. Untuk menghapus data yang telah tersimpan, tambahkan kode berikut.

```

JButton btnDelete = new JButton("Delete");
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (id != null) {
            cst.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silakan pilih data yang ingin dihapus.");
        }
    }
});

```

Kode ini membuat tombol Delete yang berfungsi untuk menghapus data ketika diklik. Jika data dengan id sudah dipilih (id tidak null), maka data tersebut akan dihapus dari database menggunakan metode delete() dan kemudian memperbarui tabel serta mereset form. Jika tidak ada data yang dipilih, maka akan muncul pesan peringatan yang meminta pengguna untuk memilih data yang ingin dihapus.

D. SIMPULAN

Dalam praktikum ini, dilakukan penerapan dua jenis design pattern, yaitu Singleton Pattern dan Builder Pattern, dalam pengembangan aplikasi CRUD untuk data customer. Singleton Pattern digunakan untuk memastikan bahwa hanya ada satu instance dari kelas yang mengelola koneksi ke database, sehingga dapat menghindari pembuatan koneksi ganda yang tidak efisien. Dengan pendekatan ini, aplikasi menjadi lebih efisien dalam hal pengelolaan koneksi database, karena koneksi hanya dibuat sekali dan digunakan berulang kali.

Di sisi lain, Builder Pattern diterapkan untuk mempermudah pembuatan objek dengan banyak atribut. Dalam kasus ini, CustomerBuilder digunakan untuk membangun objek Customer dengan berbagai atribut seperti id, nama, alamat, dan nomor HP. Pattern ini memungkinkan pembuatan objek yang lebih fleksibel dan terstruktur, terutama ketika objek tersebut memiliki banyak variabel atau atribut opsional, sehingga meminimalkan penggunaan konstruktor yang kompleks.

Pengimplementasikan pattern dalam aplikasi GUI berbasis Frame dapat melakukan operasi CRUD pada data customer. Aplikasi ini menampilkan data dalam bentuk tabel menggunakan AbstractTableModel, dan memungkinkan pengguna untuk menambah, memperbarui, dan menghapus data customer. Dengan memahami dan mengimplementasikan Singleton dan Builder Pattern dapat mengembangkan aplikasi yang lebih efisien, mudah dikelola, dan lebih mudah dikembangkan di masa depan.