

LAPORAN PRAKTIKUM PBO
PEKAN KE – 3
INTEGRASI DATABASE



Oleh:

Daffa Aira Adrin

NIM 2411531006

MATA KULIAH PRAKTIKUM PEMROGRAMAN BERORIENTASI
OBJEK

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

FAKULTAS TEKNOLOGI INFORMASI
DEPARTEMEN INFORMATIKA
UNIVERSITAS ANDALAS
PADANG

A. Pendahuluan

a) Latar Belakang

Dalam pengembangan aplikasi berbasis Java, pengelolaan data yang tersimpan di dalam database merupakan aspek penting yang harus diperhatikan. Salah satu pendekatan yang umum digunakan adalah dengan memisahkan logika akses data ke dalam lapisan tersendiri, yaitu **Data Access Object (DAO)**. DAO berfungsi sebagai jembatan antara aplikasi dan database, sehingga proses penyimpanan, pengambilan, penghapusan, dan pembaruan data dapat dilakukan secara terstruktur dan terpisah dari logika bisnis.

b) Tujuan

Tujuan dari pelaksanaan praktikum ini sebagai berikut:

- 1) Memahami konsep dasar **Data Access Object (DAO)** dalam pengembangan aplikasi Java berbasis database.
- 2) Memahami, dan menerapkan konsep dasar **fungsi CRUD** pada bahasa pemrograman Java.
- 3) Mengimplementasikan koneksi database menggunakan **JDBC** untuk menghubungkan aplikasi dengan database MySQL.
- 4) Menerapkan prinsip **Object-Oriented Programming (OOP)** melalui pemisahan logika data dan tampilan menggunakan class User, interface UserDao, dan class UserRepo.

c) Landasan Teori

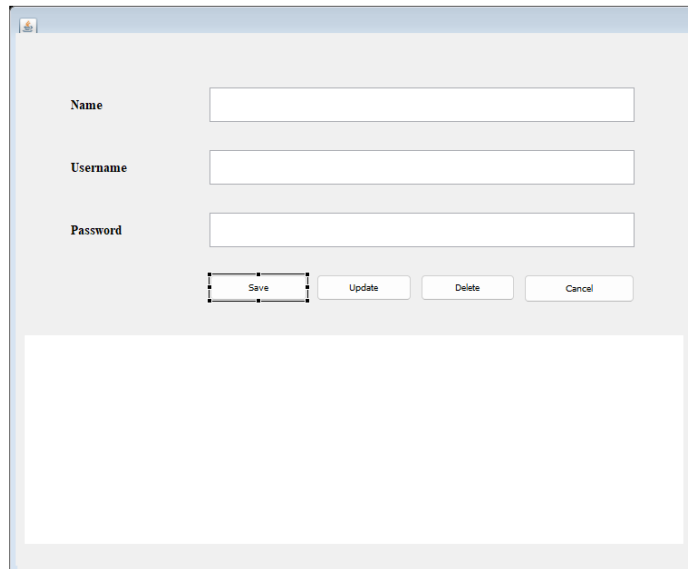
- 1) a Object-Oriented Programming (OOP) adalah paradigma pemrograman yang berfokus pada objek sebagai representasi dari entitas dunia nyata. Dalam OOP, objek memiliki atribut (data) dan method (fungsi) yang saling terkait.
- 2) Data Access Object (DAO) adalah pola desain yang digunakan untuk memisahkan logika akses data dari logika bisnis aplikasi. DAO menyediakan antarmuka untuk melakukan operasi terhadap database, seperti menyimpan, mengambil, memperbarui, dan menghapus data.
- 3) JDBC (Java Database Connectivity) adalah API dalam bahasa pemrograman Java yang digunakan untuk menghubungkan aplikasi dengan database. JDBC menyediakan berbagai class dan interface untuk melakukan koneksi, eksekusi query, dan pengelolaan hasil query dari database relasional seperti MySQL.

- 4) `AbstractTableModel` adalah class abstrak dalam pustaka `javax.swing.table` yang digunakan untuk membuat model data khusus untuk komponen `JTable`. Dengan meng-extend `AbstractTableModel`,
- 5) **Fungsi CRUD** (Create, Read, Update, Delete) merupakan empat operasi dasar dalam pengolahan data yang menjadi inti dari hampir semua aplikasi berbasis database. Dalam pemrograman Java, implementasi CRUD biasanya dilakukan melalui pola DAO (Data Access Object) dengan memanfaatkan perintah SQL, di mana operasi Create digunakan untuk menambahkan data baru ke dalam tabel, Read untuk membaca atau menampilkan data dari database, Update untuk memperbarui data yang sudah ada berdasarkan primary key, serta Delete untuk menghapus data tertentu dari tabel. Setiap operasi ini direpresentasikan dalam bentuk method Java seperti `save()`, `show()`, `update()`, dan `delete()`, yang kemudian dihubungkan dengan antarmuka pengguna (UI) menggunakan komponen seperti `JTable` pada Swing, sehingga pengguna dapat menambah, melihat, mengubah, dan menghapus data secara langsung melalui aplikasi.

B. Langkah-Langkah Pengerjaan

a. Tampilan untuk pengguna

1. Buat GUI mengandung 3 `Jtextfield` untuk menerima input, dan 4 tombol untuk melakukan operasi yang telah ditentukan pada modul. Berikan nama variabel yang telah ditentukan di modul. Buatlah sebuah `JTable` dengan meletakkannya pada sebuah `JScrollPane` agar menampilkan data yang diinputkan oleh pengguna.



2. Pada tab source code, instansiasikan objek UserRepo, List<User> dengan nama **ls**, dan deklarasikan atribut id dengan tipe data String.

```

Main.java × Database.java UserDao.java UserRepo.java LoginFram
1 package ui;
2
3 import java.awt.EventQueue;
23
24 public class UserFrame extends JFrame {
25
26     private static final long serialVersionUID = 1L;
27     private JPanel contentPane;
28     private JTextField txtName;
29     private JTextField txtUsername;
30     private JTextField txtPassword;
31     private JTable tableUsers;
32     UserRepo usr = new UserRepo();
33     List<User> ls;
34     public String id;
35
36

```

3. Deklarasikan method reset pada **class UserFrame**. Dalam method, gunakan pada **method** setText() pada setiap JTextField, dan masukkan “ ” agar JTextField dikosongkan setiap kali **method** dipanggil.

```

170
171     }
172 public void reset() {
173     txtName.setText("");
174     txtUsername.setText("");
175     txtPassword.setText("");
176 }
177

```

4. Tambahkan actionlistener pada tombol save untuk melakukan operasi terhadap input yang dimasukkan oleh pengguna ketika menekan tombol save.
5. Pada actionlistener, instansiasikan objek User, dan beri nama “user”. Gunakan method setter, dan jadikan input setiap JTextField sebagai parameter **method setter** dengan menggunakan method getText().

6. Gunakan method save pada objek usr, dan masukkan objek user sebagai input parameternya.

```
ContentPane.add(new JLabel("Save"),  
  
JButton btnSave = new JButton("Save");  
btnSave.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        User user = new User();  
        user.setNama(txtName.getText());  
        user.setUsername(txtUsername.getText());  
        user.setPassword(txtPassword.getText());  
        usr.save(user);  
        reset();  
    }  
});
```

7. Buat method dengan nama loadTable() pada **class JFrame**. Di dalam **method**, **assign** data yang telah ditampilkan dengan menggunakan method show() pada objek **usr** pada list **ls**.
8. Setelah itu, inialisasikan objek **class TableUser** dengan nama "tu", dan masukkan **ls** sebagai parameter inisial objek.

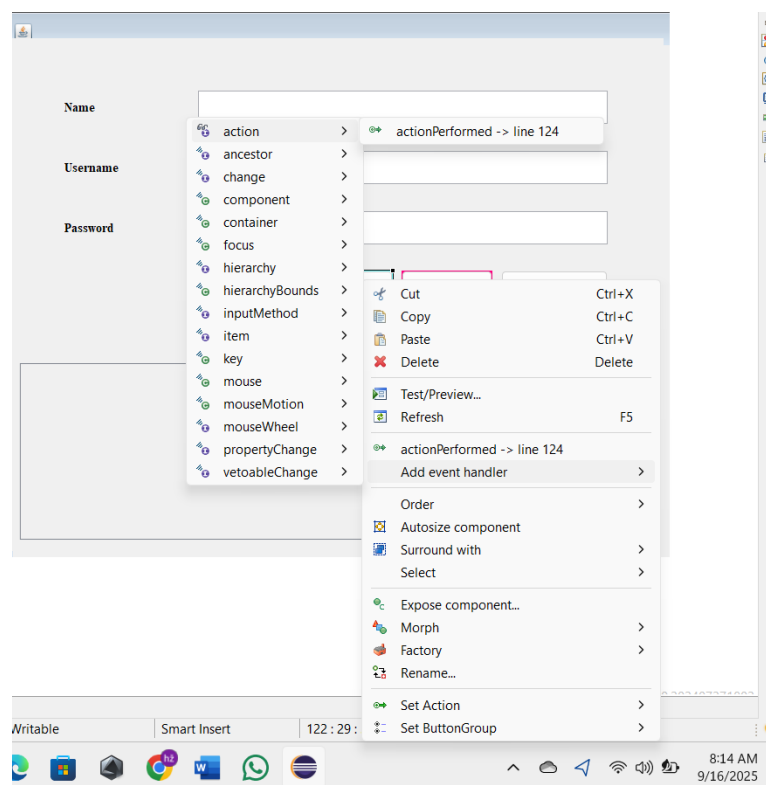
```
    }  
  
    public void loadTable() {  
        ls = usr.show();  
        TableUser tu = new TableUser(ls);  
    }  
  
    public void loadTable() {  
        ls = usr.show();  
        TableUser tu = new TableUser(ls);  
        tableUsers.setModel(tu);  
        tableUsers.getTableHeader().setVisible(true);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

11. Tambahkan EventHandler pada JTable yang diset untuk **mouseclicked**. Di dalam eventhandler, **assign** data dari baris yang dipilih oleh pengguna pada tabel **tableUsers**.

12. Kemudian, data dari masing-masing kolom JTable, dan ditampilkan pada setiap JTextField dengan menggunakan **method** `setText()` dimana data setiap kolom diambil dari baris yang ditekan, dan pada **method** `getValueAt()` ditentukan index yang dari kolom yang dipilih.

```
scrollPane.setViewportView(tableUsers);
tableUsers.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id = tableUsers.getValueAt(tableUsers.getSelectedRow(), 0).toString();
        txtName.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 1).toString());
        txtUsername.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 2).toString());
        txtPassword.setText(tableUsers.getValueAt(tableUsers.getSelectedRow(), 3).toString());
    }
});
```

13. Tambahkan `actionListener` pada tombol update dari GUI, dengan melakukan **right click** pada tombol-> **add event handler** -> **action** -> **actionPerformed**.



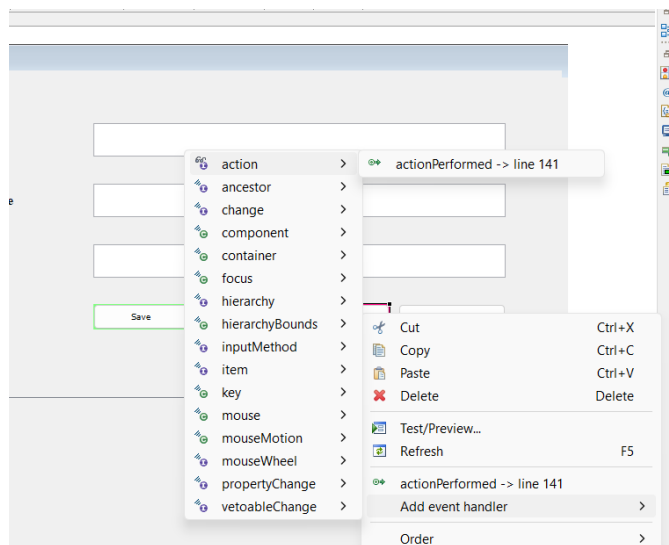
14. Pada **method** `actionPerformed`, inialisasikan objek dari **class** `User`, dan beri nama “user”. Setelah itu, gunakan **method** `setter` untuk menetapkan nama, username, password, dan id kepada objek dengan menggunakan **method** `getText()` dari masing-masing JTextField yang dijadikan sebagai input dari parameter **method** `setter`.
15. Pada objek `UserRepo` `usr`, gunakan **method** `update`, dan inputkan objek `user` ke dalam parameternya. Kemudian, panggil **method** `reset()`, dan `loadTable()`.

```

btnUpdate.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        User user = new User();
        user.setName(txtName.getText());
        user.setUsername(txtUsername.getText());
        user.setId(id);
        usr.update(user);
        reset();
        loadTable();
    }
});

```

16. Tambahkan actionListener pada tombol delete, dengan melakukan **right-click** pada tombol delete -> add event handler -> action -> actionPerformed.



17. Di dalam method actionPerformed, gunakan if statement untuk menerapkan operasi tombol **delete**. Operasi dimulai dengan diperiksa apakah id memiliki nilai null atau tidaknya, jika mengembalikan nilai true, maka id akan dihapus objek **usr**, JTextField akan direset dengan menggunakan method **reset()**, dan method **loadTable()** untuk memberikan tampilan baru pada tabel akan dipanggil.
18. Jika if statement mengembalikan nilai **False** (yaitu, jika pengguna tidak menekan kolom yang ada di tabel), akan ditampilkannya sebuah pop-up dengan menggunakan JOptionPane yang menampilkan tulisan "Silahkan pilih data yang akan dihapus".

```

JButton btnDelete = new JButton("Delete");
btnDelete.setBackground(new Color(255, 0, 128));
btnDelete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            usr.delete(id);
            reset();
            loadTable();
        } else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
        }
    }
});

```

19. Berikut tampilan GUI dari userFrame yang sudah terhubung dengan database.

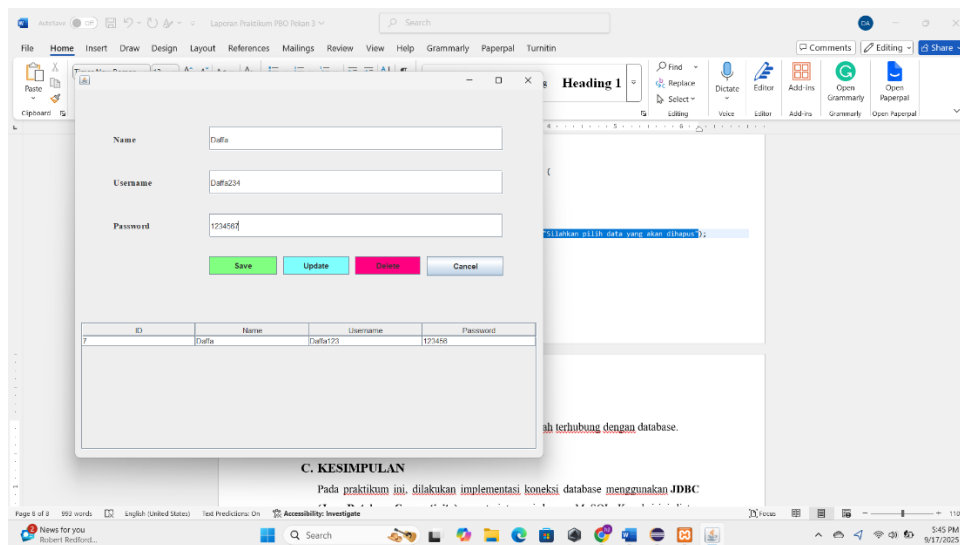


Figure 1 Memasukkan data pengguna

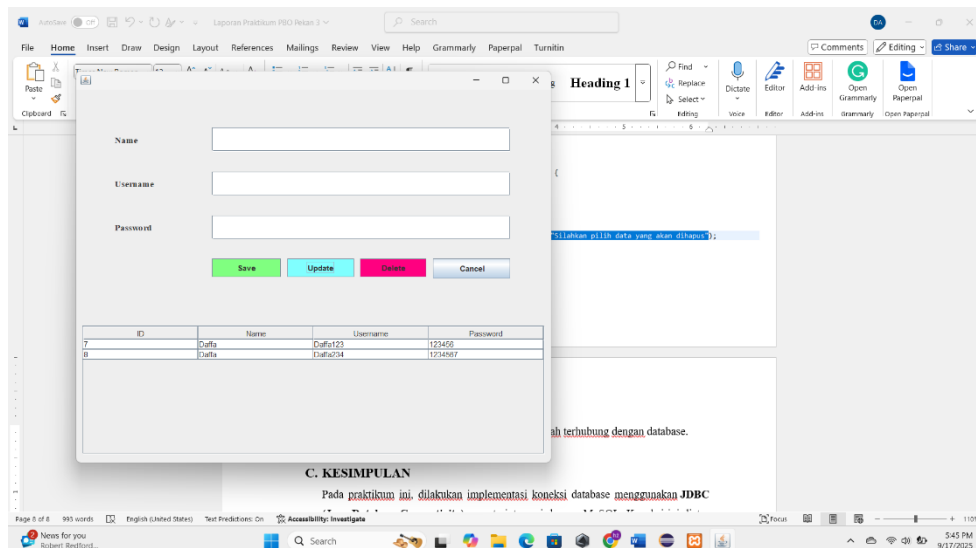


Figure 2 Tabel diperbaharui berdasarkan data yang dimasukkan

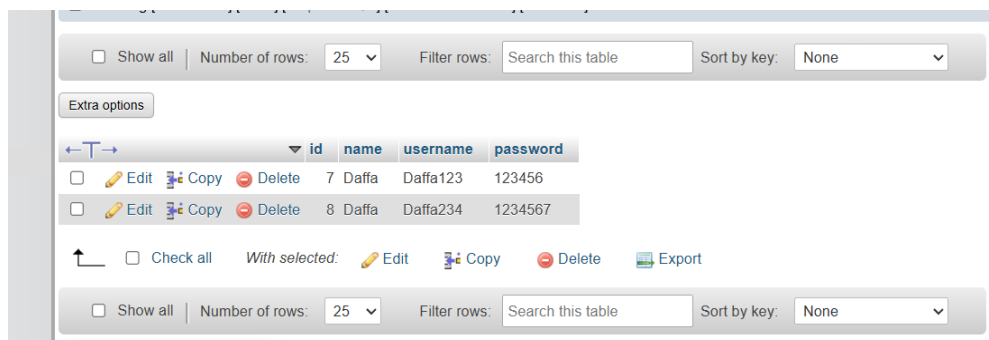
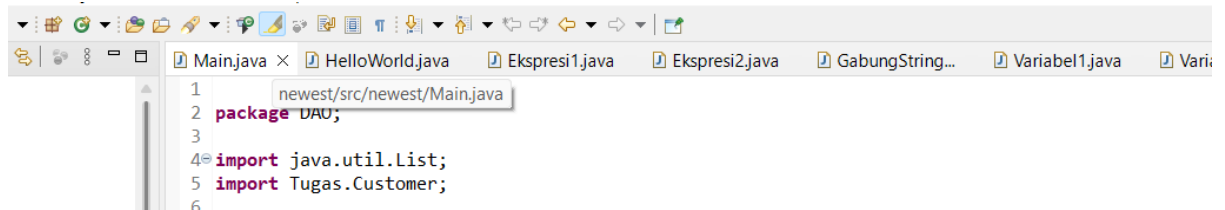


Figure 3 Database menyimpan data yang diinputkan pada GUI

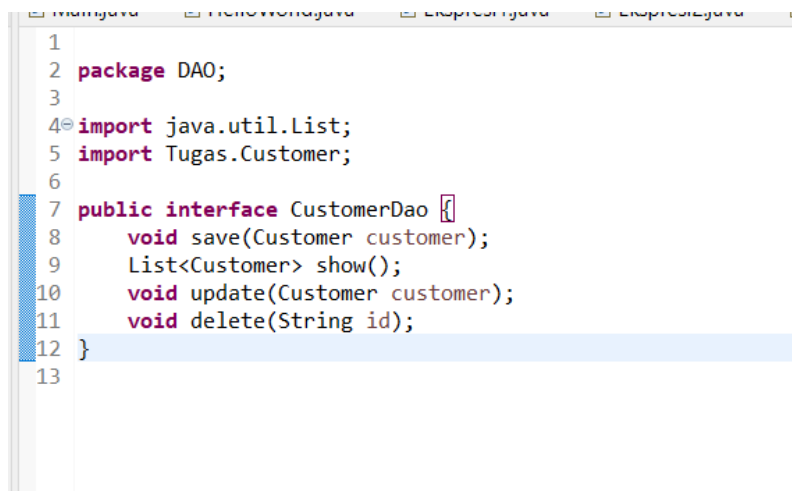
b. Tugas (Fungsi CRUD untuk pelanggan)

1. Di dalam package DAO, buat sebuah interface baru dengan nama CustomerDao.
2. Tambahkan import untuk java.util.List karena interface ini akan menggunakan tipe data List untuk menampung kumpulan objek Customer.
3. Tambahkan juga import untuk class Customer dari package Tugas.



```
1 newest/src/newest/Main.java
2 package DAO;
3
4 import java.util.List;
5 import Tugas.Customer;
6
```

4. Di dalam interface CustomerDao, deklarasikan method-method berikut:
 - a) save(Customer customer) untuk menyimpan data customer ke dalam database.
 - b) show() yang mengembalikan List<Customer> untuk menampilkan seluruh data customer.
 - c) update(Customer customer) untuk memperbarui data customer.
 - d) delete(String id) untuk menghapus data customer berdasarkan ID.



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

5. Buat sebuah class baru dengan nama CustomerRepo di dalam package DAO.
6. Pastikan class CustomerRepo mengimplementasikan interface CustomerDao.
7. Tambahkan import untuk library berikut:
 - a) java.sql.* untuk koneksi dan manipulasi database.
 - b) java.util.* untuk penggunaan List dan ArrayList.
 - c) config.Database untuk memanggil method koneksi database.
 - d) Tugas.Customer untuk menggunakan class Customer.

```

1 // DAO/CustomerRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Customer;
6 import java.sql.*;
7 import java.util.*;
8

```

8. Deklarasikan atribut connection bertipe Connection.
9. Buat beberapa query SQL sebagai final String:
 - a) insert untuk menyimpan data customer.
 - b) select untuk menampilkan semua data customer.
 - c) delete untuk menghapus data customer berdasarkan ID.
 - d) update untuk memperbarui data customer berdasarkan ID.

```

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

```

10. Buat constructor CustomerRepo() yang akan menginisialisasi koneksi database dengan memanggil Database.koneksi().

```

6 import java.sql.*;
7 import java.util.*;
8
9 public class CustomerRepo implements CustomerDao {
10     private Connection connection;
11     final String insert = "INSERT INTO customer (nama, alamat, no_hp) VALUES (?, ?, ?);";
12     final String select = "SELECT * FROM customer;";
13     final String delete = "DELETE FROM customer WHERE id=?;";
14     final String update = "UPDATE customer SET nama=?, alamat=?, no_hp=? WHERE id=?;";
15
16     public CustomerRepo() {
17         connection = Database.koneksi();
18     }
19

```

11. Implementasikan method save(Customer customer) menggunakan PreparedStatement untuk query insert.

```

15
16 public CustomerRepo() {
17     connection = Database.koneksi();
18 }
19
20 @Override
21 public void save(Customer customer) {
22     try (PreparedStatement st = connection.prepareStatement(insert)) {
23         st.setString(1, customer.getNama());
24         st.setString(2, customer.getAlamat());
25         st.setString(3, customer.getNomorHp());
26         st.executeUpdate();
27     } catch (SQLException e) { e.printStackTrace(); }
28 }
29

```

12. Implementasikan method show() menggunakan Statement untuk query select, lalu mapping hasil ResultSet ke objek Customer.

```

30
31 }
32
33 @Override
34 public List<Customer> show() {
35     List<Customer> list = new ArrayList<>();
36     try (Statement st = connection.createStatement()) {
37         ResultSet rs = st.executeQuery(select);
38         while (rs.next()) {
39             Customer c = new Customer();
40             c.setId(rs.getString("id"));
41             c.setNama(rs.getString("nama"));
42             c.setAlamat(rs.getString("alamat"));
43             c.setNomorHp(rs.getString("no_hp"));
44             list.add(c);
45         }
46     } catch (SQLException e) { e.printStackTrace(); }
47     return list;
48 }
49

```

13. Implementasikan method update(Customer customer) menggunakan PreparedStatement untuk query update.

```

44     return list;
45 }
46
47 @Override
48 public void update(Customer customer) {
49     try (PreparedStatement st = connection.prepareStatement(update)) {
50         st.setString(1, customer.getNama());
51         st.setString(2, customer.getAlamat());
52         st.setString(3, customer.getNomorHp());
53         st.setString(4, customer.getId());
54         st.executeUpdate();
55     } catch (SQLException e) { e.printStackTrace(); }
56 }
57

```

14. Implementasikan method delete(String id) menggunakan PreparedStatement untuk query delete.

15. Tambahkan blok try-catch di setiap method untuk menangani SQLException.

16. Tutup PreparedStatement atau Statement dengan try-with-resources agar tidak terjadi kebocoran koneksi.

```

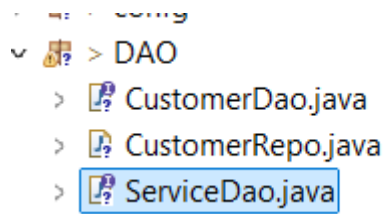
@Override
public void update(Customer customer) {
    try (PreparedStatement st = connection.prepareStatement(update)) {
        st.setString(1, customer.getNama());
        st.setString(2, customer.getAlamat());
        st.setString(3, customer.getNomorHp());
        st.setString(4, customer.getId());
        st.executeUpdate();
    } catch (SQLException e) { e.printStackTrace(); }
}

@Override
public void delete(String id) {
    try (PreparedStatement st = connection.prepareStatement(delete)) {
        st.setString(1, id);
        st.executeUpdate();
    } catch (SQLException e) { e.printStackTrace(); }
}
}

```

c. TUGAS (FUNGSI CRUD UNTUK PELAYANAN)

1. Di dalam package DAO, buat sebuah interface baru dengan nama ServiceDao.



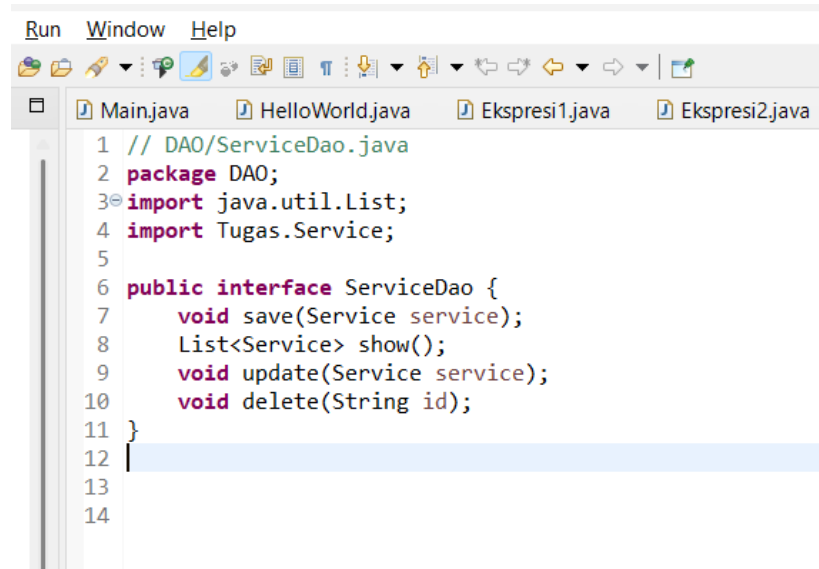
2. Tambahkan import untuk java.util.List karena interface ini akan menggunakan tipe data List untuk menampung kumpulan objek Service.
3. Tambahkan juga import untuk class Service dari package Tugas.

```

Main.java HelloWorld.java Ekspresi1.java
1 // DAO/ServiceDao.java
2 package DAO;
3 import java.util.List;
4 import Tugas.Service;
5

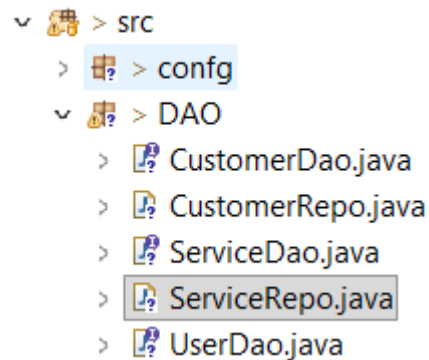
```

4. Di dalam interface ServiceDao, deklarasikan method-method berikut:
 - a) save(Service service) untuk menyimpan data service ke dalam database.
 - b) show() yang mengembalikan List<Service> untuk menampilkan seluruh data service.
 - c) update(Service service) untuk memperbarui data service.
 - d) delete(String id) untuk menghapus data service berdasarkan ID.

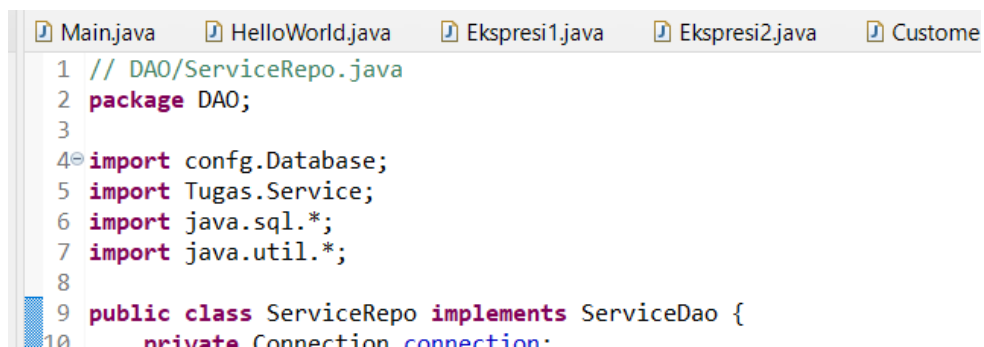


```
1 // DAO/ServiceDao.java
2 package DAO;
3 import java.util.List;
4 import Tugas.Service;
5
6 public interface ServiceDao {
7     void save(Service service);
8     List<Service> show();
9     void update(Service service);
10    void delete(String id);
11 }
12
13
14
```

5. Buat sebuah class baru dengan nama ServiceRepo di dalam package DAO.



6. Pastikan class ServiceRepo mengimplementasikan interface ServiceDao.
7. Tambahkan import untuk library berikut:
- a) `java.sql.*` untuk koneksi dan manipulasi database.
 - b) `java.util.*` untuk penggunaan List dan ArrayList.
 - c) `config.Database` untuk memanggil method koneksi database.
 - d) `Tugas.Service` untuk menggunakan class Service.



```
1 // DAO/ServiceRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Service;
6 import java.sql.*;
7 import java.util.*;
8
9 public class ServiceRepo implements ServiceDao {
10     private Connection connection;
```

8. Deklarasikan atribut connection bertipe Connection.

```

1 // DAO/ServiceRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Service;
6 import java.sql.*;
7 import java.util.*;
8
9 public class ServiceRepo implements ServiceDao {
10     private Connection connection;
11     // ...
12 }

```

9. Buat beberapa query SQL sebagai final String:
 - a) insert untuk menyimpan data service.
 - b) select untuk menampilkan semua data service.
 - c) delete untuk menghapus data service berdasarkan ID.
 - d) update untuk memperbarui data service berdasarkan ID.

```

1 // DAO/ServiceRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Service;
6 import java.sql.*;
7 import java.util.*;
8
9 public class ServiceRepo implements ServiceDao {
10     private Connection connection;
11     final String insert = "INSERT INTO service (jenis, harga, status) VALUES (?, ?, ?);";
12     final String select = "SELECT * FROM service;";
13     final String delete = "DELETE FROM service WHERE id=?;";
14     final String update = "UPDATE service SET jenis=?, harga=?, status=? WHERE id=?;";
15 }

```

10. Buat constructor ServiceRepo() yang akan menginisialisasi koneksi database dengan memanggil Database.koneksi().

```

1 // DAO/ServiceRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Service;
6 import java.sql.*;
7 import java.util.*;
8
9 public class ServiceRepo implements ServiceDao {
10     private Connection connection;
11     final String insert = "INSERT INTO service (jenis, harga, status) VALUES (?, ?, ?);";
12     final String select = "SELECT * FROM service;";
13     final String delete = "DELETE FROM service WHERE id=?;";
14     final String update = "UPDATE service SET jenis=?, harga=?, status=? WHERE id=?;";
15
16     public ServiceRepo() {
17         connection = Database.koneksi();
18     }
19 }

```

11. Implementasikan method save(Service service) dengan PreparedStatement untuk query insert.

```

1 // DAO/ServiceRepo.java
2 package DAO;
3
4 import config.Database;
5 import Tugas.Service;
6 import java.sql.*;
7 import java.util.*;
8
9 public class ServiceRepo implements ServiceDao {
10     private Connection connection;
11     final String insert = "INSERT INTO service (jenis, harga, status) VALUES (?, ?, ?);";
12     final String select = "SELECT * FROM service;";
13     final String delete = "DELETE FROM service WHERE id=?;";
14     final String update = "UPDATE service SET jenis=?, harga=?, status=? WHERE id=?;";
15
16     public ServiceRepo() {
17         connection = Database.koneksi();
18     }
19
20     @Override
21     public void save(Service service) {
22         try (PreparedStatement st = connection.prepareStatement(insert)) {
23             st.setString(1, service.getJenis());
24             st.setDouble(2, service.getHarga());
25             st.setString(3, service.getStatus());
26             st.executeUpdate();
27         } catch (SQLException e) { e.printStackTrace(); }
28     }
29 }

```

12. Implementasikan method show() dengan Statement untuk query select, lalu mapping hasil ResultSet ke objek Service.

```

    }

    @Override
    public List<Service> show() {
        List<Service> list = new ArrayList<>();
        try (Statement st = connection.createStatement()) {
            ResultSet rs = st.executeQuery(select);
            while (rs.next()) {
                Service s = new Service();
                s.setId(rs.getString("id"));
                s.setJenis(rs.getString("jenis"));
                s.setHarga(rs.getDouble("harga"));
                s.setStatus(rs.getString("status"));
                list.add(s);
            }
        } catch (SQLException e) { e.printStackTrace(); }
        return list;
    }
}

```

13. Implementasikan method update(Service service) dengan PreparedStatement untuk query update.

```

    }

    @Override
    public void update(Service service) {
        try (PreparedStatement st = connection.prepareStatement(update)) {
            st.setString(1, service.getJenis());
            st.setDouble(2, service.getHarga());
            st.setString(3, service.getStatus());
            st.setString(4, service.getId());
            st.executeUpdate();
        } catch (SQLException e) { e.printStackTrace(); }
    }
}

```

14. Implementasikan method delete(String id) dengan PreparedStatement untuk query delete.

15. Tambahkan blok try-catch di setiap method untuk menangani SQLException.

16. Gunakan try-with-resources agar resource seperti PreparedStatement dan Statement otomatis tertutup setelah digunakan.

```

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

    @Override
    public void delete(String id) {
        try (PreparedStatement st = connection.prepareStatement(delete)) {
            st.setString(1, id);
            st.executeUpdate();
        } catch (SQLException e) { e.printStackTrace(); }
    }
}

```

C. KESIMPULAN

Pada praktikum ini, dilakukan implementasi koneksi database menggunakan **JDBC (Java Database Connectivity)** yang terintegrasi dengan MySQL. Koneksi ini diatur melalui class Database, yang bertanggung jawab untuk membuka akses ke database laundry_apps. Selanjutnya, dibuat interface UserDao yang mendefinisikan operasi dasar terhadap data pengguna, seperti save, show, delete, dan update.

Implementasi dari interface tersebut dilakukan dalam class UserRepo, yang berisi fungsi **CRUD** untuk berinteraksi langsung dengan tabel user di database. Selain itu, untuk menampilkan data pengguna dalam bentuk tabel pada GUI, digunakan class TableUser yang merupakan turunan dari AbstractTableModel.