

**TUGAS PRAKTIKUM PEMOGRAMAN II**



Novan Ramdan (223040031)

**FAKULTAS TEKNIK  
UNIVERSITAS PASUNDAN  
BANDUNG 2024**

## 1. Person.java

```
Untitled-1

package model;

public class Person {
    private int id;
    private String nama;
    private String alamat;
    private int umur;

    public int getId() {
        return id;
    }

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

    public String getNama() {
        return nama;
    }

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

    public String getAlamat() {
        return alamat;
    }

    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }

    public int getUmur() {
        return umur;
    }

    public void setUmur(int umur) {
        this.umur = umur;
    }
}
```

Terdapat setter dan getter untuk: id,nama,Alamat dan umur

## 2. MySqlConnection.java

```
private static final String URL = "jdbc:mysql://localhost:3306/pp2_biodata";
private static final String USER = "root";
private static final String PASSWORD = "";
```

URL: Menyimpan URL database, termasuk nama database pp2\_biodata.

USER: Nama pengguna untuk mengakses database yaitu root.

PASSWORD: Kata sandi untuk pengguna root (tidak memakai password).

```
public static Connection getConnection() {
    Connection conn = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection(URL, USER, PASSWORD);
        System.out.println("Koneksi berhasil ke database pp2_biodata.");
    } catch (ClassNotFoundException e) {
        System.err.println("Driver MySQL tidak ditemukan. Pastikan driver sudah ditambahkan ke proyek.");
        e.printStackTrace();
    } catch (SQLException e) {
        System.err.println("Koneksi gagal: " + e.getMessage());
        e.printStackTrace();
    }
    return conn;
}
```

Fungsi dari metode ini adalah untuk membuka koneksi ke database dengan parameter URL, USER, dan PASSWORD yang telah ditentukan.

- Langkah pertama adalah memuat driver MySQL menggunakan `Class.forName("com.mysql.cj.jdbc.Driver")`
- Setelah driver ditemukan, metode `DriverManager.getConnection(URL, USER, PASSWORD)` mencoba menghubungkan aplikasi ke database.
- Jika koneksi berhasil, pesan "Koneksi berhasil ke database pp2\_biodata" akan ditampilkan di konsol.
- Jika gagal, kode akan menangani ada dua jenis error:

- `ClassNotFoundException`: Jika driver MySQL tidak ditemukan, menunjukkan bahwa driver belum ditambahkan ke proyek.
- `SQLException`: Jika koneksi gagal karena alasan lain.

### 3. PersonDao.java

```
public int insert(Person person) {  
    int result = -1;  
    try (Connection connection = MySQLConnection.getConnection()) {  
        PreparedStatement statement = connection.prepareStatement(  
            "INSERT INTO person (nama, alamat, umur) VALUES (?, ?, ?)");  
        statement.setString(1, person.getNama());  
        statement.setString(2, person.getAlamat());  
        statement.setInt(3, person.getUmur());  
        result = statement.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return result;  
}
```

#### Method insert

- Menambahkan data baru ke tabel person.
- Menggunakan perintah SQL `INSERT INTO person (nama, alamat, umur) VALUES (?, ?, ?)` untuk menambah data.
- Mengembalikan jumlah baris yang terpengaruh jika berhasil, atau -1 jika gagal.

```

    public int update(Person person) {
        int result = -1;
        try (Connection connection = MySQLConnection.getConnection()) {
            PreparedStatement statement = connection.prepareStatement(
                "UPDATE person SET nama = ?, alamat = ?, umur = ? WHERE id = ?");
            statement.setString(1, person.getNama());
            statement.setString(2, person.getAlamat());
            statement.setInt(3, person.getUmur());
            statement.setInt(4, person.getId());
            result = statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result;
    }

```

### Method update

- Memperbarui data pada tabel person berdasarkan id.
- Menggunakan perintah SQL UPDATE person SET nama = ?, alamat = ?, umur = ? WHERE id = ? untuk memperbarui data.
- Mengembalikan jumlah baris yang diperbarui jika berhasil, atau -1 jika gagal.

```

    public int delete(Person person) {
        int result = -1;
        try (Connection connection = MySQLConnection.getConnection()) {
            PreparedStatement statement = connection.prepareStatement(
                "DELETE FROM person WHERE id = ?");
            statement.setInt(1, person.getId());
            result = statement.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return result;
    }

```

### Method delete

- Menghapus data dari tabel person berdasarkan id.
- Menggunakan perintah SQL DELETE FROM person WHERE id = ? untuk menghapus data.
- Mengembalikan jumlah baris yang dihapus jika berhasil, atau -1 jika gagal.

```
Untitled-1

public List<Person> findAll() {
    List<Person> list = new ArrayList<>();
    try (Connection connection = MySqlConnection.getConnection();
        PreparedStatement statement = connection.prepareStatement("SELECT * FROM person");
        ResultSet resultSet = statement.executeQuery()) {

        while (resultSet.next()) {
            Person person = new Person();
            person.setId(resultSet.getInt("id"));
            person.setName(resultSet.getString("nama"));
            person.setAlamat(resultSet.getString("alamat"));
            person.setUmur(resultSet.getInt("umur"));
            list.add(person);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return list;
}
```

### Method findall

- Mengambil semua data dari tabel person.
- Menggunakan perintah SQL SELECT \* FROM person untuk mendapatkan seluruh data.
- Mengembalikan daftar List<Person> yang berisi semua data Person dari tabel person.

#### 4. PersonFrame.java

```
Untitled-1

setLayout(new GridBagLayout());
GridBagConstraints gbc = new GridBagConstraints();
gbc.insets = new Insets(5, 5, 5, 5);
gbc.fill = GridBagConstraints.HORIZONTAL;
```

Menggunakan GridBagLayout untuk mengatur komponen dalam grid, yang memberikan fleksibilitas dalam mengatur posisi dan ukuran komponen di JFrame.

```
Untitled-1

// Label Nama
gbc.gridx = 0;
gbc.gridy = 0;
gbc.weightx = 0.3;
add(new JLabel("Nama:"), gbc);

// Field Nama
gbc.gridx = 1;
gbc.weightx = 0.7;
namaField = new JTextField(20);
add(namaField, gbc);
```

Setiap label dan JTextField ditempatkan menggunakan koordinat gridx dan gridy. weightx digunakan untuk mengatur distribusi ruang antar-komponen

```
Untitled-1

JButton simpanButton = new JButton("Simpan");
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 2;
add(simpanButton, gbc);
```

Menambahkan tombol "Simpan" dan mengatur posisi serta ukuran tombol. Membuat ActionListener untuk menangani aksi saat tombol diklik.

```

Untitled-1

simpanButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        Person person = new Person();
        person.setNama(namaField.getText());
        person.setAlamat(alamatField.getText());
        try {
            person.setUmur(Integer.parseInt(umurField.getText()));
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(null, "Umur harus berupa angka.", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        personDao.insert(person);
        JOptionPane.showMessageDialog(null, "Data berhasil disimpan!");

        namaField.setText("");
        alamatField.setText("");
        umurField.setText("");
    }
});

```

Ketika tombol "Simpan" diklik:

- Membuat objek Person dan mengambil data dari namaField, alamatField, dan umurField.
- Mengecek tipe umur. Jika umur tidak valid (bukan angka), pesan error akan ditampilkan.
- Menyimpan data ke database menggunakan personDao.insert(person).
- Menampilkan konfirmasi setelah data berhasil disimpan dan membersihkan semua field input.