# Laporan Tugas Pertemuan ke-7

## Identitas

* NPM : 223040147
* Nama : Galuh Wikri Ramadhan
* Kelas : B
* URL Repository Github : <https://github.com/GaluhWikri/pp2_223040147_B/tree/main/Tugas%207>

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **Member Dao** |
|  |
| **Penjelasan** |
| 1. **insert(Member member)**  * Fungsi ini menyisipkan data Member baru ke dalam tabel member. Menggunakan PreparedStatement untuk menghindari SQL injection, fungsi ini memasukkan nilai id, nama, dan jenis\_member\_id dari objek Member ke dalam database. * Mengembalikan jumlah baris yang terpengaruh (result), dengan nilai -1 jika terjadi kesalahan.  1. **update(Member member)**  * Fungsi ini mengupdate data Member berdasarkan id dari objek Member yang diberikan. * Menyusun query update dengan PreparedStatement, menetapkan nilai baru untuk nama dan jenis\_member\_id sesuai data yang baru. * Hasilnya adalah jumlah baris yang diperbarui, atau -1 jika terjadi error.  1. **delete(Member member)**  * Fungsi ini menghapus data Member berdasarkan id. * Query delete dieksekusi menggunakan PreparedStatement, dengan parameter id dari objek Member. * Mengembalikan jumlah baris yang dihapus atau -1 jika terjadi error.  1. **findAll()**  * Fungsi ini mengembalikan daftar semua data Member dalam bentuk List<Member>. * Menjalankan query select menggunakan Statement, mengambil data id, nama dari member, serta jenis\_member\_id dan jenis\_member\_nama dari tabel jenis\_member. * Menggunakan ResultSet untuk membaca hasil query dan membuat objek Member serta JenisMember sebelum menambahkannya ke dalam List<Member> yang dikembalikan |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **Jenis Member Dao** |
|  |
| **Penjelasan** |
| 1. **insert(JenisMember jenisMember)**  * Fungsi ini menambahkan data JenisMember baru ke tabel jenis\_member di database. * Menggunakan PreparedStatement untuk menyisipkan nilai id dan nama dari objek JenisMember. * Mengembalikan jumlah baris yang terpengaruh (result), atau -1 jika terjadi kesalahan.  1. **update(JenisMember jenisMember)**  * Fungsi ini memperbarui data JenisMember yang sudah ada berdasarkan id yang diberikan. * Menjalankan query update menggunakan PreparedStatement, mengubah kolom nama berdasarkan id. * Mengembalikan jumlah baris yang diperbarui, atau -1 jika terjadi error.  1. **delete(JenisMember jenisMember)**  * Fungsi ini menghapus data JenisMember dari tabel jenis\_member berdasarkan id. * Menggunakan query delete dengan PreparedStatement dan id sebagai parameter. * Mengembalikan jumlah baris yang dihapus atau -1 jika ada kesalahan.  1. **findAll()**  * Fungsi ini mengambil seluruh data dari tabel jenis\_member dan mengembalikannya dalam bentuk List<JenisMember>. * Menjalankan query select \* from jenis\_member menggunakan Statement, dan setiap baris hasilnya digunakan untuk membuat objek JenisMember yang baru. * Menggunakan ResultSet untuk mengambil nilai id dan nama dari setiap baris, lalu menambahkannya ke dalam List<JenisMember> yang dikembalikan. |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **Model jenis member dan member** |
|  |
| **Penjelasan** |
| 1. **Deklarasi Atribut**  * id: Menyimpan ID unik dari biodata, digunakan sebagai pengenal utama. * nama: Menyimpan nama dari biodata. * alamat: Menyimpan alamat dari biodata.  1. **Getter dan Setter**  * **Getter** (getId, getNama, getAlamat): Method untuk mengambil nilai dari masing-masing atribut id, nama, dan alamat. * **Setter** (setId, setNama, setAlamat): Method untuk menetapkan atau mengubah nilai masing-masing atribut. |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **View Jenis Member** |
|  |
| **Penjelasan** |
| **1. JenisMemberButtonSimpanActionListener**  Kelas ini bertindak sebagai ActionListener untuk tombol "Simpan" pada JenisMemberFrame.   * **Constructor**:   + Menerima parameter JenisMemberFrame dan JenisMemberDao yang digunakan untuk berinteraksi dengan komponen tampilan dan DAO. * **Method actionPerformed**:   + Dipanggil saat tombol "Simpan" diklik.   + Mengambil input nama dari JenisMemberFrame, membuat JenisMember baru dengan id acak (UUID) dan nama sesuai input.   + Menambahkan JenisMember baru ke tabel di GUI dengan memanggil addJenisMember pada JenisMemberFrame dan menyimpannya ke database dengan jenisMemberDao.insert.   **2. JenisMemberFrame**  Kelas ini adalah tampilan utama untuk fitur JenisMember, dengan komponen input dan tabel untuk menampilkan data.   * **Atribut**:   + jenisMemberList: Menyimpan daftar JenisMember yang diambil dari database.   + textFieldNama: JTextField untuk input nama JenisMember.   + tableModel: Model tabel khusus (JenisMemberTableModel) untuk mengatur data JenisMember di tabel.   + jenisMemberDao: Objek JenisMemberDao untuk interaksi dengan database. * **Constructor**:   + Memuat data JenisMember dari database dengan jenisMemberDao.findAll.   + Menyiapkan komponen tampilan (label, input, tombol, dan tabel) dan menempatkannya pada posisi yang ditentukan.   + Membuat instance JenisMemberButtonSimpanActionListener untuk tombol "Simpan", yang mengatur aksi saat tombol diklik. * **Method getNama**:   + Mengembalikan teks yang diinput di textFieldNama. * **Method addJenisMember**:   + Menambahkan objek JenisMember baru ke tableModel agar muncul di tabel.   + Mengosongkan textFieldNama setelah data disimpan.   **Struktur Komponen**   * **Komponen Input**:   + Label (labelInput) untuk memberi keterangan.   + textFieldNama untuk menginput nama JenisMember. * **Tombol Simpan**:   + Memiliki ActionListener yang dipicu saat diklik. * **Tabel**:   + Menggunakan JTable dengan model JenisMemberTableModel untuk menampilkan data JenisMember dalam bentuk tabel.   + Dibungkus dengan JScrollPane agar bisa di-scroll. |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **View Member** |
|  |
| **Penjelasan** |
|  |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **View Main** |
|  |
| **Penjelasan** |
|  |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **Biodata Dao** |
|  |
| **Penjelasan** |
| 1. **Koneksi Database**    * BiodataDao menginisialisasi koneksi database menggunakan MySqlConnection.getConnection() saat objek BiodataDao dibuat. Ini memungkinkan penggunaan koneksi yang sama untuk semua operasi dalam DAO ini. 2. **addBiodata** (Menambahkan Data Biodata)    * Method ini menerima objek Biodata, lalu menjalankan query INSERT untuk menambahkan data ke tabel biodata. Data id, nama, dan alamat dari objek Biodata dimasukkan ke kolom tabel yang sesuai. 3. **getAllBiodata** (Mengambil Semua Data Biodata)    * Method ini mengambil semua data dari tabel biodata. Data hasil query disimpan dalam List<Biodata> dengan mengisi setiap objek Biodata dengan id, nama, dan alamat dari setiap baris di tabel. 4. **updateBiodata** (Memperbarui Data Biodata)    * Method ini menerima objek Biodata, lalu menjalankan query UPDATE untuk memperbarui nama dan alamat berdasarkan id. Hanya kolom yang berubah yang di-update di tabel. 5. **deleteBiodata** (Menghapus Data Berdasarkan ID)    * Method ini menerima id dan menjalankan query DELETE untuk menghapus data dari tabel biodata yang memiliki id tersebut. |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **Db MySqlConnection** |
|  |
| **Penjelasan** |
| 1. **Deklarasi URL, Username, dan Password Database**  * DB\_URL: URL untuk menghubungkan aplikasi ke database MySQL. Dalam contoh ini, URL mengarah ke localhost pada port 3306 dan database bernama pp2\_membership. * DB\_USER: Username untuk mengakses database (dalam hal ini, root). * DB\_PASS: Password untuk username tersebut (di sini kosong, namun sebaiknya diatur jika dibutuhkan).  1. **Static Connection**  * connection: Objek Connection yang digunakan untuk menyimpan koneksi database agar dapat digunakan ulang di seluruh aplikasi tanpa harus membuka koneksi baru setiap kali dibutuhkan.  1. **Method getConnection**  * Method ini akan memeriksa apakah connection bernilai null (tidak ada koneksi yang aktif). * Jika connection kosong, method akan mencoba membuat koneksi baru dengan DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASS). * Jika koneksi berhasil, pesan “Koneksi ke database berhasil!” dicetak ke console. * Jika gagal, akan menampilkan pesan error dan stack trace untuk membantu proses debugging. |

## Penjelasan Kode

Jika **berkas memiliki output**

|  |
| --- |
| **Model Biodata** |
|  |
| **Penjelasan** |
| 1. **Deklarasi Atribut**  * id: Menyimpan ID unik dari biodata, digunakan sebagai pengenal utama. * nama: Menyimpan nama dari biodata. * alamat: Menyimpan alamat dari biodata.  1. **Getter dan Setter**  * **Getter** (getId, getNama, getAlamat): Method untuk mengambil nilai dari masing-masing atribut id, nama, dan alamat. * **Setter** (setId, setNama, setAlamat): Method untuk menetapkan atau mengubah nilai masing-masing atribut. |

## Penjelasan Kode

Jika berkas **tidak memiliki output**, gunakan tabel ini.

|  |
| --- |
| **View Biodata** |
|  |
| **Penjelasan** |
| 1. **Deklarasi Atribut**    * biodataDao: Objek untuk mengakses data biodata dari database.    * table, tableModel: Untuk menampilkan data biodata dalam bentuk tabel.    * txtId, txtNama, txtAlamat: Field input untuk memasukkan ID, nama, dan alamat.    * btnAdd, btnUpdate, btnDelete: Tombol untuk menambah, memperbarui, dan menghapus data. 2. **Konstruktor**    * **BiodataView()**: Konstruktor ini membuat instance BiodataDao, memanggil metode initComponents() untuk inisialisasi komponen GUI, dan memuat data dari database ke dalam tabel. 3. **Metode initComponents()**    * Menyiapkan GUI aplikasi.    * Membuat panel input untuk memasukkan data biodata, panel tombol untuk menambah, memperbarui, dan menghapus data, serta panel tabel untuk menampilkan data.    * Menambahkan event listener ke setiap tombol dan ke tabel untuk menangani aksi pengguna. 4. **Metode loadBiodataData()**    * Mengambil semua data biodata dari database menggunakan biodataDao dan menampilkannya dalam tabel. 5. **Metode addBiodata()**    * Membuat objek Biodata dengan data dari field input.    * Memanggil biodataDao.addBiodata() untuk menambahkan data ke database.    * Memuat ulang data dalam tabel dan mengosongkan field input. 6. **Metode updateBiodata()**    * Mengambil data dari field input untuk memperbarui objek Biodata.    * Memanggil biodataDao.updateBiodata() untuk memperbarui data di database.    * Memuat ulang tabel dan mengosongkan field input. 7. **Metode deleteBiodata()**    * Mengambil id dari baris yang dipilih di tabel.    * Memanggil biodataDao.deleteBiodata() untuk menghapus data dari database.    * Memuat ulang tabel dan mengosongkan field input. 8. **Metode fillInputFields()**    * Mengisi field input berdasarkan baris yang dipilih di tabel, sehingga data dapat diedit atau dihapus. 9. **Metode clearInputFields()**    * Mengosongkan semua field input setelah penambahan, pembaruan, atau penghapusan data. 10. **Metode main()**  * Memulai aplikasi dengan membuat instance BiodataView dan menampilkannya. |
| **Output** |
|  |

Catatan:

* Silakan modifikasi dan tambahkan informasi sesuai dengan proyek yang Anda kerjakan baik pada tanda [kalimat].
* Sesuaikan nama file laporan dengan format **Tugas[Pertemuan ke-X]\_PP1\_NPM**

Contoh: Tugas[P10]\_PP1\_203040104

* Sesuaikan judul laporan dengan format **Laporan Tugas Pertemuan ke-X**

Contoh: Laporan Tugas Pertemuan ke-10