Laporan Tugas Proyek
Pemrograman Berorientasi Objek

Aplikasi Travel

Disusun Oleh :

11322003	Adinda Hutasoit
11322008	Maria Pangaribuan
11322013	Felix Aldi I Simanjuntak
11322018	Maranatha Siahaan

Institut Teknologi Del
Fakultas Vokasi
DIII- Teknologi Informasi
Laguboti

2023

1. FUNCTIONAL DESCRIPTION

Pada sub bab ini akan dijelaskan fungsi-fungsi utama sistem yang dibuat untuk kebutuhan *Admin* antara lain:

1. Fungsi Login

   Fungsi *Login* digunakan oleh *Admin* untuk dapat masuk ke dalam sistem untuk dapat mengaksesnya.

2. Fungsi Mengupdate Data Kendaraan

   Fungsi ini digunakan oleh *Admin* untuk dapat memperbarui informasi kendaraan yang ada dalam sistem saat adanya perubahan data yang dibutuhkan.

3. Fungsi Membaca Data Kendaraan

   Fungsi ini digunakan oleh *Admin* untuk dapat melihat informasi kendaraan yang tersimpan dalam sistem.

4. Fungsi Menambah Data Kendaraan

   Fungsi ini digunakan oleh *Admin* untuk dapat menambahkan informasi kendaraan baru ke dalam sistem. Adapun Data yang dibutuhkan adalah dengan mengisi form nama dan jumlah kendaraan yang dibutuhkan.

5. Fungsi Menghapus Data Kendaraan:

   Fungsi ini digunakan oleh *Admin* untuk dapat menghapus informasi kendaraan dari sistem yang sebelumnya telah diisi.

6. Fungsi Membaca Data Pegawai:

   Fungsi ini digunakan oleh *Admin* untuk dapat melihat informasi pegawai yang terdaftar dalam sistem.

7. Fungsi Menghapus Data Pegawai:

   Fungsi ini digunakan oleh *Admin* untuk dapat menghapus data atau informasi pegawai dari sistem.

8. Fungsi Mengedit Data Pegawai:

   Fungsi ini digunakan oleh *Admin* untuk dapat mengubah atau memperbarui informasi pegawai yang sudah ada dalam sistem.

9. Fungsi Menambah Data Pegawai:

   Fungsi ini digunakan oleh Admin untuk dapat menambahkan data pegawai baru ke dalam sistem. Adapun Data yang dibutuhkan adalah dengan mengisi form ID, Nama dan Tanggal Lahir.

2. DESIGN TABLE STRUCTURE

	Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
☐	kendaraan	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_general_ci	16.0 KB	-
☐	pegawai	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	5	InnoDB	utf8mb4_general_ci	16.0 KB	-
☐	user	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_general_ci	16.0 KB	-
	3 tabel	Jumlah	6	InnoDB	utf8mb4_general_ci	48.0 KB	0 B

Terdapat tiga tabel yang disebutkan dalam struktur basis data `travel` yaitu `kendaraan`, `pegawai`, dan `user`.

1. Tabel `kendaraan`:

- Kolom-kolom:
- `id` (int, primary key, auto-increment): ID unik untuk setiap kendaraan.
- `nama` (varchar(50), not null): Nama kendaraan (maksimum 50 karakter).
- `jumlah` (varchar(50), not null): Jumlah kendaraan (maksimum 50 karakter).
- Engine: InnoDB
- Charset: utf8mb4

2. Tabel `pegawai`:

- Kolom-kolom:
- `id` (int, primary key, auto-increment): ID unik untuk setiap pegawai.
- `nama` (varchar(50), not null): Nama pegawai (maksimum 50 karakter).
- `birth` (varchar(50), not null): Tanggal lahir pegawai (maksimum 50 karakter).
- Engine: InnoDB
- Charset: utf8mb4

3. Tabel `user`:

- Kolom-kolom:

- `id` (int, primary key, auto-increment): ID unik untuk setiap pengguna.

- `nama` (varchar(50), not null): Nama pengguna (maksimum 50 karakter).

- `email` (varchar(50), not null): Alamat email pengguna (maksimum 50 karakter).

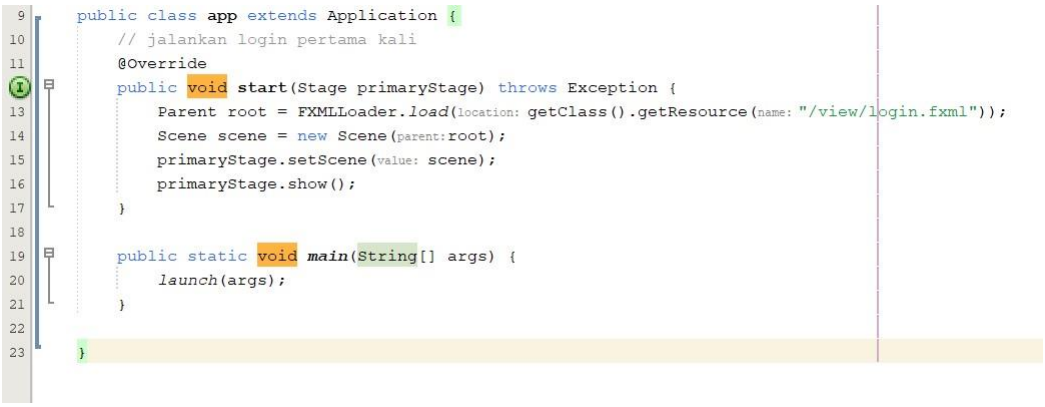
- `password` (varchar(50), not null): Kata sandi pengguna (maksimum 50 karakter).

- Engine: InnoDB

- Charset: utf8mb4

3. OOP IMPLEMENTATION a. Abstraction

Abstraksi adalah konsep dasar dalam pemrograman yang melibatkan penyederhanaan sistem kompleks dengan memodelkan kelas berdasarkan fitur-fitur pokoknya dan mengabaikan detail yang tidak relevan. Ini memungkinkan para pemrogram fokus pada aspek-aspek yang diperlukan dari objek atau sistem sambil menyembunyikan kompleksitas yang tidak perlu.



```
9   public class app extends Application {
10      // jalankan login pertama kali
11      @Override
12      public void start(Stage primaryStage) throws Exception {
13          Parent root = FXMLLoader.load(location: getClass().getResource(name: "/view/login.fxml"));
14          Scene scene = new Scene(parent:root);
15          primaryStage.setScene(value: scene);
16          primaryStage.show();
17      }
18
19      public static void main(String[] args) {
20          launch(args);
21      }
22
23  }
```

Abstraksi dalam konteks ini terkait dengan penyembunyian detail implementasi tampilan dan logika dari pengguna dan hanya mengekspos fungsionalitas yang relevan melalui antarmuka pengguna (UI).

Berikut adalah beberapa poin abstraksi yang dapat diidentifikasi dalam kode:

- FXMLLoader:

Penggunaan FXMLLoader adalah bentuk abstraksi yang memungkinkan kami mendefinisikan antarmuka pengguna menggunakan file FXML (XML-based markup language) tanpa perlu menulis kode Java secara langsung untuk setiap elemen UI.

- Parent, Scene, dan Stage:

Penggunaan Parent, Scene, dan Stage adalah bagian dari abstraksi JavaFX untuk menangani elemen-elemen tampilan dan pengaturan jendela. Pengguna hanya perlu berurusan dengan konsep seperti Stage (jendela) dan Scene (lingkungan tampilan), tanpa perlu memahami detail implementasinya.

- launch(args):

Metode launch(args) digunakan untuk memulai aplikasi JavaFX dan menyediakan abstraksi dari detail inisialisasi dan pengelolaan siklus hidup aplikasi.

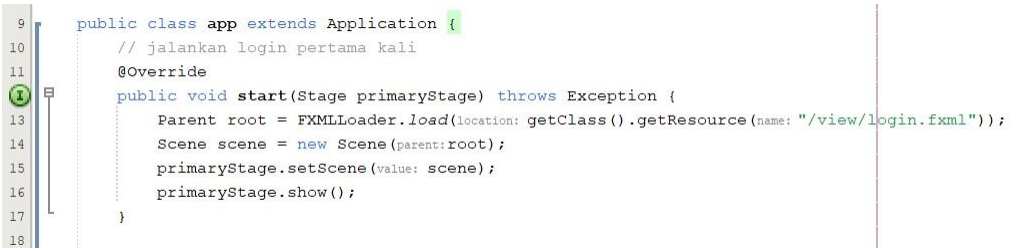
Dengan menggunakan abstraksi ini, pengguna dari kelas app dapat dengan mudah memulai aplikasi dan menampilkan antarmuka pengguna tanpa harus terlalu terlibat dengan detail-detail implementasi dari JavaFX. Ini adalah contoh bagaimana abstraksi membantu menyembunyikan kompleksitas dan mempermudah penggunaan suatu kerangka kerja atau library.

b. Encapsulation

Encapsulation diimplementasikan melalui penggunaan aksesibilitas private pada properti statement dan result, serta penggunaan metode get dan set untuk mengakses dan mengubah nilai properti tersebut. Berikut adalah bagian dari kode yang menunjukkan implementasi encapsulation:

c. Inheritance

Pewarisan adalah mekanisme dalam pemrograman berorientasi objek yang memungkinkan sebuah kelas (subclass atau kelas turunan) untuk mewarisi properti dan perilaku dari kelas lain (kelas dasar atau superclass). Ini meningkatkan kemungkinan penggunaan kembali kode dan membentuk hubungan antar kelas, di mana subclass dapat menggunakan fitur-fitur yang telah ada dalam superclass.

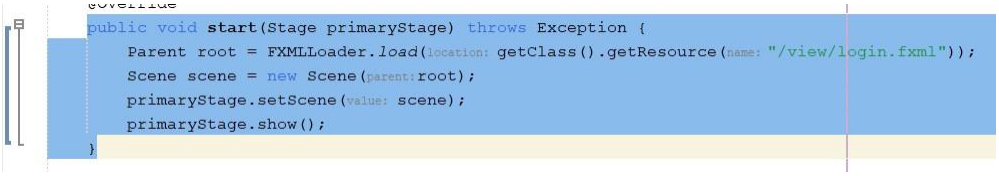


```
9   public class app extends Application {
10      // jalankan login pertama kali
11      @Override
12      public void start(Stage primaryStage) throws Exception {
13          Parent root = FXMLLoader.load(location: getClass().getResource(name: "/view/login.fxml"));
14          Scene scene = new Scene(parent:root);
15          primaryStage.setScene(value: scene);
16          primaryStage.show();
17      }
18
```

d. Polymorphism

Polimorfisme adalah kemampuan objek untuk mengambil bentuk yang berbeda, atau istilah lainnya, untuk merespon panggilan metode dengan cara yang sesuai dengan jenis objek tersebut. Polimorfisme dapat berupa polimorfisme compile-time (overloading) dan polimorfisme runtime (overriding).

Polimorfisme dalam konteks ini terjadi karena metode start diambil dari antarmuka (Application) dan di-override oleh kelas app. Metode ini sebenarnya merupakan polimorfisme karena dapat mengambil berbagai bentuk (implementasi) tergantung pada kelas yang mengimplementasikannya.



```
    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(location: getClass().getResource(name: "/view/login.fxml"));
        Scene scene = new Scene(parent:root);
        primaryStage.setScene(value: scene);
        primaryStage.show();
    }
```

e. Interface

Interface mendefenisikan sebuah template yang menyatakan metode-metode yang harus diimplementasikan oleh kelas yang menggunakan interface tersebut. Pada model sudah tersedia interface.

```
public final StringProperty namaProperty() {
    return nama;
}

public final String getnama() {
    return namaProperty().get();
}

public final void setnama(String newnama) {
    namaProperty().set(value: newnama);
}

public final StringProperty birthProperty() {
    return birth;
}

public final String getbirth() {
    return birthProperty().get();
}

public final void setbirth(String birth) {
    birthProperty().set(value: birth);
}

public final int getId() {
    return id.get();
}
```

f. Exception handling

Penanganan pengecualian adalah proses menanggapi dan mengelola kondisi abnormal atau kesalahan yang dapat terjadi selama eksekusi program. Dengan menggunakan mekanisme penanganan pengecualian, programmer dapat mengidentifikasi, menangani, dan memberikan respon yang sesuai terhadap situasi-situasi yang tidak terduga atau bermasalah.

```
        try {
            if (nama.isEmpty() || jumlah.isEmpty()) {
                showAlert(nama: "Warning", header:"Isian Kosong", content: "Pastikan semua isian terisi.");
            } else {
                pst = con.prepareStatement("insert into kendaraan(nama, jumlah) values (?, ?)");

                pst.setString(i: 1, string:nama);
                pst.setString(i: 2, string:jumlah);
                pst.executeUpdate();

                showAlert(nama: "successfully", header:"Penambahan kendaraan", content: "kendaraan has been successfully add

                table();

                isi_nama.clear();
                isi_nama.requestFocus();
                isi_jumlah.clear();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            showAlert(nama: "Error", header:"Database Error", "Error adding kendaraan: " + ex.getMessage());
        }
    }
```

```
        try {
            if (email.getText().isEmpty() || password.getText().isEmpty()) {
                emptyFieldErrorMessage(message: "Harap isi semua field");
            } else {
                String selectData = "SELECT * FROM user WHERE email=? AND password=?";

                prepare = connect.prepareStatement(selectData);
                prepare.setString(i: 1, string:email.getText());
                prepare.setString(i: 2, string:password.getText());
                result = prepare.executeQuery();

                if (result.next()) {
                    successMessage(message: "Login Berhasil");
                    Parent root = FXMLLoader.load(location: getClass().getResource(name: "/view/DashboardView.fxml"));

                    Stage stage = new Stage();
                    stage.setTitle(value: "Dashboard");
                    stage.setScene(new Scene(parent:root));

                    stage.show();
                    btn_login.getScene().getWindow().hide();

                } else {
                    emptyFieldErrorMessage(message: "email atau Password Salah");
                }

            }
        } catch (Exception e) {

        }

    }
```

4. LINK YOUTUBE

https://youtu.be/DVeiKkKtfeg?si=Vodz27qe3jwb9M7P