

**LAPORAN**  
**UJIAN AKHIR SEMESTER (UAS)**  
**PERANCANGAN APLIKASI PEMBELIAN SEPEDA**

Dosen Pengampu: Taufik Ridwan, S.T., M.T



Disusun Oleh:

Kelompok 4 :

Mubtaghi Ridho Robbi	2310631250065
Nayaka Al Fikri Januar	2310631250070
Faridzal Nur Fadlillah	2310631250090
Irzha Adji Prabowo	2310631250093

**PROGRAM STUDI SISTEM INFORMASI**  
**FAKULTAS ILMU KOMPUTER**  
**UNIVERSITAS SINGAPERBANGSA KARAWANG**  
**2025**

## KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Allah SWT atas segala rahmat, taufik, dan hidayah-Nya sehingga kami dapat menyelesaikan Laporan Ujian Akhir Semester (UAS) Mata Kuliah Perancangan Aplikasi dengan judul “Aplikasi Pembelian Sepeda” dengan baik dan tepat waktu.

Laporan ini disusun sebagai bentuk pertanggungjawaban atas tugas akhir pada mata kuliah Perancangan Aplikasi, yang bertujuan untuk mengasah kemampuan mahasiswa dalam merancang dan mengimplementasikan sebuah aplikasi sesuai dengan kebutuhan pengguna. Dalam laporan ini, kami merancang aplikasi pembelian sepeda berbasis GUI (Graphical User Interface) menggunakan bahasa pemrograman Java, yang diharapkan dapat menjadi solusi praktis dalam proses transaksi penjualan dan pembelian sepeda secara online.

Kami menyampaikan terima kasih yang sebesar-besarnya kepada Bapak Taufik Ridwan, S.T., M.T., selaku dosen pengampu mata kuliah, atas bimbingan, ilmu, dan kesempatan yang telah diberikan selama perkuliahan berlangsung. Ucapan terima kasih juga kami sampaikan kepada semua pihak yang telah memberikan dukungan dan bantuan, baik secara langsung maupun tidak langsung, dalam penyusunan laporan ini.

Kami menyadari bahwa laporan ini masih jauh dari sempurna karena keterbatasan pengetahuan dan pengalaman kami. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun demi penyempurnaan laporan ini di masa yang akan datang.

Akhir kata, semoga laporan ini dapat memberikan manfaat dan menambah wawasan bagi pembaca, serta menjadi referensi dalam pengembangan aplikasi sejenis di masa mendatang.

Karawang, Mei 2025

Penyusun

Kelompok 4

## DAFTAR ISI

<b>KATA PENGANTAR.....</b>	i
<b>DAFTAR ISI.....</b>	ii
<b>BAB I PENDAHULUAN.....</b>	1
1.1.    Latar Belakang .....	1
1.2.    Rumusan Masalah.....	2
1.3.    Tujuan .....	2
<b>BAB II LANDASAN TEORI.....</b>	4
2.1    Fitur – Fitur .....	4
2.2    Konsep OOP dalam Kode Program .....	5
2.3    Unifield Modeling Language (UML) .....	9
2.3.1    Use Case Diagram.....	9
2.3.2    Sequence Diagram .....	11
2.3.3    Class Diagram.....	17
<b>BAB III IMPLEMENTASI DAN PENGUJIAN KODE PROGRAM .....</b>	19
3.1    Implementasi Kode Program .....	19
3.1.1    Pendahuluan & Arsitektur Umum .....	19
3.1.2    Struktur Data (Model).....	23
3.1.3    Konektivitas Database .....	31
3.1.4    Alur Pengguna (View) .....	41
3.2    Implementasi Pengujian.....	78
<b>BAB IV PENUTUP .....</b>	87
4.1    Kesimpulan .....	87

# **BAB I**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Pada era digital saat ini, teknologi informasi memainkan peran yang sangat penting dalam meningkatkan efisiensi dan efektivitas di berbagai sektor, seperti pertanian, pendidikan, kesehatan, hingga sektor perdagangan. Salah satu penerapan teknologi informasi yang semakin berkembang adalah pemanfaatan antarmuka pengguna grafis (Graphical User Interface/GUI) dalam sistem komputerisasi. GUI memungkinkan interaksi antara manusia dan komputer menjadi lebih mudah dan intuitif, sehingga sangat cocok diterapkan dalam dunia bisnis, termasuk pada bisnis penjualan sepeda.

Bisnis penjualan sepeda merupakan salah satu sektor yang dapat mengambil manfaat besar dari digitalisasi, khususnya dalam pengelolaan data produk, transaksi pembelian, dan stok barang. Sistem penjualan sepeda yang masih dilakukan secara konvensional sering kali menimbulkan berbagai kendala, seperti kesalahan pencatatan transaksi, kurangnya transparansi data stok, serta keterbatasan dalam akses informasi produk. Oleh karena itu, dibutuhkan suatu solusi berbasis aplikasi yang mampu mengintegrasikan seluruh proses tersebut secara efisien dan sistematis.

Sebagai bagian dari tugas akhir dalam Mata Kuliah Pemrograman Berorientasi Objek (PBO), kami mengembangkan sebuah aplikasi penjualan sepeda berbasis Java GUI yang terintegrasi dengan database MySQL. Proyek ini dipilih tidak hanya karena relevansinya dengan kebutuhan digitalisasi di dunia nyata, tetapi juga sebagai sarana untuk menerapkan konsep-konsep dasar pemrograman berorientasi objek secara praktis dan nyata. Aplikasi ini dirancang untuk mempermudah pengguna, baik admin maupun pelanggan, dalam melakukan berbagai aktivitas, seperti mengelola data sepeda, melihat katalog produk, melakukan transaksi pembelian, dan memantau riwayat transaksi dengan antarmuka yang user-friendly dan interaktif.

Dalam pengembangan aplikasi ini, kami menerapkan berbagai prinsip OOP seperti enkapsulasi, inheritance, polimorfisme, dan exception handling, agar struktur kode menjadi modular, terstruktur, dan mudah dikembangkan. Admin memiliki fitur untuk menambahkan, mengedit, dan menghapus data sepeda, termasuk spesifikasi, harga, dan gambar. Sementara

itu, pengguna dapat menjelajahi katalog sepeda, melakukan pencarian berdasarkan kriteria tertentu, serta melakukan pembelian yang langsung tercatat dalam sistem. Setiap komponen dalam aplikasi dibangun menggunakan Java Swing sebagai GUI utama, dengan pendekatan event-driven programming untuk memastikan pengalaman pengguna yang responsif.

Melalui proyek ini, kami tidak hanya belajar menerapkan teori pemrograman secara nyata, tetapi juga mengembangkan kemampuan dalam merancang dan membangun sistem yang dapat menyelesaikan permasalahan di dunia industri. Diharapkan, aplikasi ini dapat menjadi contoh sederhana dari implementasi teknologi informasi dalam dunia perdagangan, serta menjadi langkah awal kami dalam memahami dan mengembangkan solusi digital berbasis pemrograman berorientasi objek secara profesional.

## 1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam pengembangan proyek aplikasi ini adalah sebagai berikut:

1. Bagaimana membangun sebuah aplikasi penjualan sepeda yang interaktif dan mudah digunakan dengan antarmuka grafis berbasis Java GUI?
2. Bagaimana menerapkan konsep pemrograman berorientasi objek (Object-Oriented Programming) seperti enkapsulasi, inheritance, polimorfisme, dan exception handling dalam pengembangan aplikasi penjualan sepeda?
3. Bagaimana cara mengintegrasikan database MySQL dengan aplikasi Java GUI untuk memastikan pengelolaan data yang terstruktur dan aman?

## 1.3. Tujuan

Adapun tujuan dari pembangunan sistem penjualan sepeda berbasis Java GUI ini adalah sebagai berikut:

1. Mengembangkan aplikasi penjualan sepeda berbasis Java GUI yang memiliki tampilan antarmuka yang intuitif, responsif, dan mudah digunakan.
2. Menerapkan prinsip-prinsip dasar pemrograman berorientasi objek (OOP) seperti enkapsulasi, inheritance, polimorfisme, dan exception handling dalam pengembangan aplikasi.
3. Merancang dan mengimplementasikan sistem database menggunakan MySQL untuk mengelola data produk sepeda, data pelanggan, dan riwayat transaksi secara terorganisir.
4. Menyediakan fitur login berdasarkan jenis pengguna (admin dan pelanggan) untuk

membedakan hak akses dan fungsi sistem sesuai peran masing-masing.

5. Menjadi media pembelajaran praktis bagi mahasiswa dalam mengimplementasikan teori PBO dalam bentuk sistem nyata yang dapat diterapkan dalam dunia industri.

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Fitur – Fitur**

Sistem toko sepeda berbasis Java GUI + SQLite ini dikembangkan untuk mempermudah proses pemesanan dan transaksi produk sepeda oleh pengguna. Antarmuka dirancang menggunakan Java Swing, dengan penyimpanan data menggunakan database SQLite melalui koneksi JDBC. Berikut ini adalah penjelasan fitur-fitur utama:

##### **1. Login dan Registrasi Pengguna**

Sistem menyediakan halaman registrasi dan login untuk pengguna baru maupun yang sudah memiliki akun.

- Registrasi: Pengguna dapat mendaftarkan diri dengan mengisi nama lengkap, email, dan password. Validasi diterapkan untuk memastikan seluruh input diisi.
- Login: Autentikasi dilakukan dengan mencocokkan email dan password terhadap database SQLite. Jika berhasil, pengguna diarahkan ke halaman utama aplikasi (Main).

##### **2. Dashboard Produk**

Setelah login, pengguna akan melihat daftar produk berupa sepeda yang dikelompokkan berdasarkan kategori seperti “Sepeda Gunung”, “Sepeda Lipat”, dan “Sepeda Balap”.

Fitur yang tersedia antara lain:

- Tampilan daftar produk per kategori
- Informasi produk ditampilkan menggunakan komponen visual seperti JLabel dan JPanel
- Tombol “Beli” tersedia untuk setiap produk, yang akan mengarahkan ke halaman pembayaran

##### **3. Proses Pembayaran**

- Pengguna dapat memilih metode pembayaran (Virtual Account atau COD) dan kurir pengiriman.
- Sistem menampilkan detail pembayaran, termasuk alamat pengiriman dan nomor Virtual Account (jika dipilih).
- Setelah pembayaran berhasil, sistem menampilkan struk pembayaran.

##### **4. Antarmuka Pengguna (GUI) Interaktif dan Modern**

- Seluruh sistem dirancang menggunakan Java Swing dengan layout yang rapi dan konsisten.
- Warna, font, dan elemen visual disesuaikan untuk pengalaman pengguna yang nyaman.

## 5. Validasi dan Penanganan Kesalahan

- Setiap form dilengkapi dengan validasi, seperti input wajib diisi dan pengecekan stok produk.
- Sistem menerapkan exception handling untuk menghindari crash akibat kesalahan input atau koneksi database.

## 6. Koneksi Database SQLite

- Sistem terhubung dengan database SQLite melalui koneksi JDBC.
- Data pengguna, produk, dan pesanan disimpan dalam database dan dikelola melalui query SQL.

## 2.2 Konsep OOP dalam Kode Program

Object-Oriented Programming (OOP) atau Pemrograman Berbasis Objek adalah paradigma pemrograman yang berfokus pada penggunaan objek dan class untuk memodelkan solusi dari suatu masalah. Dalam sistem penjualan sepeda ini, seluruh struktur program dibangun berdasarkan prinsip-prinsip OOP yang meliputi:

### 1. Class dan Object

- Class adalah blueprint atau template yang mendefinisikan atribut (variabel) dan method (fungsi) dari suatu entitas.
- Object adalah instansiasi nyata dari class, yang memiliki nilai unik untuk atributnya.
- Contoh Implementasi:
  - Class Produk (Produk.java):

```
public class Produk {
    private int productId;
    private String name;
    // ... atribut lainnya ...

    public Produk(int productId, String name, ...) { // Constructor
        this.productId = productId;
        this.name = name;
        // ... inisialisasi lainnya ...
    }
    // ... method getter dan setter ...
}
```

- Class ini mendefinisikan struktur data untuk produk sepeda, termasuk atribut seperti

productId, name, dan method seperti getProductId().

- Pembuatan Object (DatabaseHelper.java):

```
Produk produk = new Produk(1, "BMX Pro X1", "BMX", 1500000, "...", "bmx1.png", 10);
```

- Baris kode di atas membuat objek produk dari class Produk dengan nilai atribut yang spesifik.

## 2. Method

- Method merupakan fungsi yang dimiliki class untuk menjalankan logika tertentu.
- Contoh implementasi:
  - getNamaProduk() dan getHarga() pada Produk.java
  - actionPerformed() pada setiap class yang mengimplementasikan ActionListener
  - prosesPembayaran() pada HalamanPembayaran.java
  - main() di Main.java sebagai entry point aplikasi

## 3. Enkapsulasi (Encapsulation)

- Enkapsulasi adalah teknik menyembunyikan detail internal object dengan membatasi akses langsung ke atribut melalui modifier private dan menyediakan method public (getter/setter) untuk interaksi.
- Contoh Implementasi:

- Atribut private dan Getter/Setter (Produk.java):

```
private int productId; // Atribut private

// Getter untuk productId
public int getProductId() {
    return productId;
}

// Setter untuk productId (tidak ada dalam kode, contoh tambahan)
public void setProductId(int productId) {
    if (productId > 0) { // Validasi sederhana
        this.productId = productId;
    }
}
```

- Atribut productId hanya bisa diakses melalui method getProductId(), mencegah modifikasi sembarangan.
- Penerapan di User.java:

```
private String password; // Password disembunyikan
public String getPassword() { return password; } // Hanya getter, tidak ada setter untuk keamanan.
```

## 4. Pewarisan (Inheritance)

- Inheritance memungkinkan class anak (subclass) mewarisi atribut dan method dari

class induk (superclass), sehingga menghindari duplikasi kode.

- Contoh Implementasi:

- Class JFrame sebagai Superclass (semua file GUI):

```
public class HalamanLogin extends JFrame { // HalamanLogin mewarisi JFrame
    public HalamanLogin() {
        setTitle("Login - CyclePro");
        // ... konfigurasi JFrame ...
    }
}
```

- Class HalamanLogin mewarisi semua fitur dari JFrame (seperti window management) dan menambahkan fungsionalitas spesifik.

## 5. Polimorfisme (Polymorphism)

- Polimorfisme memungkinkan method yang sama memiliki perilaku berbeda tergantung pada class yang mengimplementasikannya.
- Contoh Implementasi:

- Override Method `toString()` (`Produk.java`):

```
@Override
public String toString() {
    return name + " - Rp" + String.format("%.0f", price); // Format tampilan produk
}
```

- Method `toString()` dari class `Object` di-override untuk menampilkan informasi produk secara custom.
  - Polimorfisme dalam GUI (`KategoriSepeda.java`):

```
JButton backButton = new JButton("Kembali");
backButton.addActionListener(e -> { // Lambda expression untuk perilaku unik
    new Dashboard().setVisible(true);
    dispose();
});
```

- Method `addActionListener` bisa menerima perilaku berbeda tergantung konteks.

## 6. Abstraction

- Abstraction adalah konsep menyembunyikan detail kompleks dan hanya menampilkan fungsionalitas esensial.
- Contoh Implementasi:

- Method `createOrder` di `DatabaseHelper.java`:

```
public static boolean createOrder(int userId, int productId, ...) {
    // Proses kompleks: koneksi DB, eksekusi SQL, dll.
    // Pengguna hanya perlu tahu "createOrder berhasil/gagal".
}
```

- Pengguna class tidak perlu tahu detail bagaimana order disimpan ke database.

## 7. Exception Handling

- Exception handling digunakan untuk menangani error secara elegan tanpa menghentikan program.
- Contoh Implementasi:

- Try-Catch di DatabaseHelper.java:

```
try (Connection conn = connect()) {
    // Eksekusi query SQL
} catch (SQLException e) {
    System.out.println("Error: " + e.getMessage()); // Pesan error jelas
}
```

- Jika koneksi database gagal, program tetap berjalan dan menampilkan pesan error.
- Validasi Input di Registrasi.java:

```
if (username.isEmpty() || password.isEmpty()) {
    JOptionPane.showMessageDialog(..., "Field tidak boleh kosong!");
}
```

## 8. GUI dan Event-Driven Programming

- Sistem menggunakan Java Swing untuk antarmuka pengguna dan merespons event (seperti klik tombol).
- Contoh Implementasi:

- Event Listener di HalamanLogin.java:

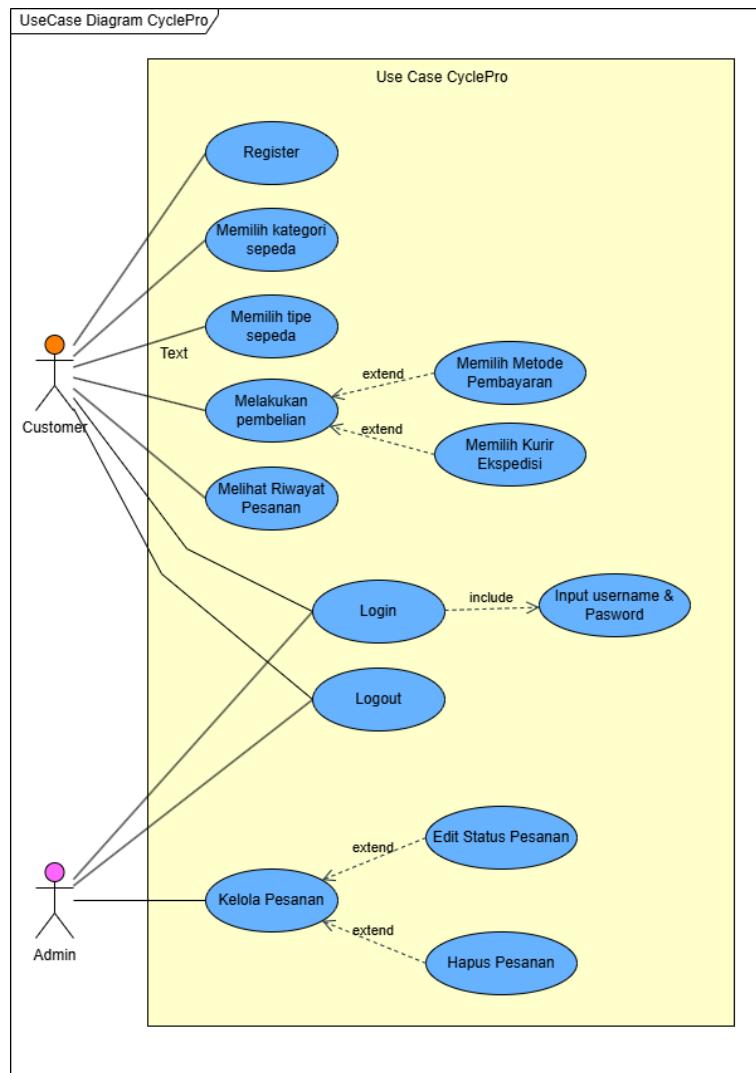
```
loginButton.addActionListener(e -> {
    String username = usernameField.getText();
    // ... proses login ...
});
```

- Kode di atas memicu aksi saat tombol login diklik.
- Komponen GUI di KategoriSepeda.java:

```
JPanel bikeItemPanel = new JPanel(); // Panel untuk setiap produk
bikeItemPanel.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        // Buka halaman pembayaran saat produk diklik
    }
});
```

## 2.3 Unifield Modeling Language (UML)

### 2.3.1 Use Case Diagram



Use case diagram merupakan gambaran fungsionalitas sistem yang berinteraksi langsung dengan aktor yang terlibat, baik pengguna maupun administrator. Pada sistem CyclePro, terdapat dua aktor utama, yaitu Customer dan Admin. Masing-masing aktor memiliki hak akses dan fungsi yang berbeda sesuai perannya dalam sistem.

#### 1. Aktor: Customer

Aktor ini merupakan pengguna umum yang dapat mengakses fitur-fitur utama dari sistem CyclePro. Adapun aktivitas atau use case yang dapat dilakukan oleh Customer antara lain:

- Register: Proses pendaftaran akun baru, dilakukan jika Customer belum memiliki akun untuk mengakses sistem.
- Login: Proses masuk ke dalam sistem yang memiliki hubungan include dengan use

case Input Username & Password, yang berarti proses login harus mencakup pengisian username dan password secara valid.

- Memilih Kategori Sepeda dan Memilih Tipe Sepeda: Customer dapat menjelajahi berbagai kategori dan tipe sepeda yang tersedia sebelum melakukan pembelian.
- Melakukan Pembelian: Proses transaksi pembelian sepeda yang memiliki hubungan extend dengan dua use case tambahan, yaitu:
  - Memilih Metode Pembayaran
  - Memilih Kurir Ekspedisi

Relasi extend menunjukkan bahwa kedua proses ini merupakan bagian opsional atau tambahan yang hanya dijalankan jika diperlukan oleh sistem.

- Melihat Riwayat Pesanan: Customer dapat melihat daftar pesanan yang pernah dilakukan untuk keperluan pelacakan atau bukti transaksi.
- Logout: Digunakan untuk keluar dari sistem. Untuk masuk kembali, Customer hanya perlu melakukan login tanpa perlu register ulang.

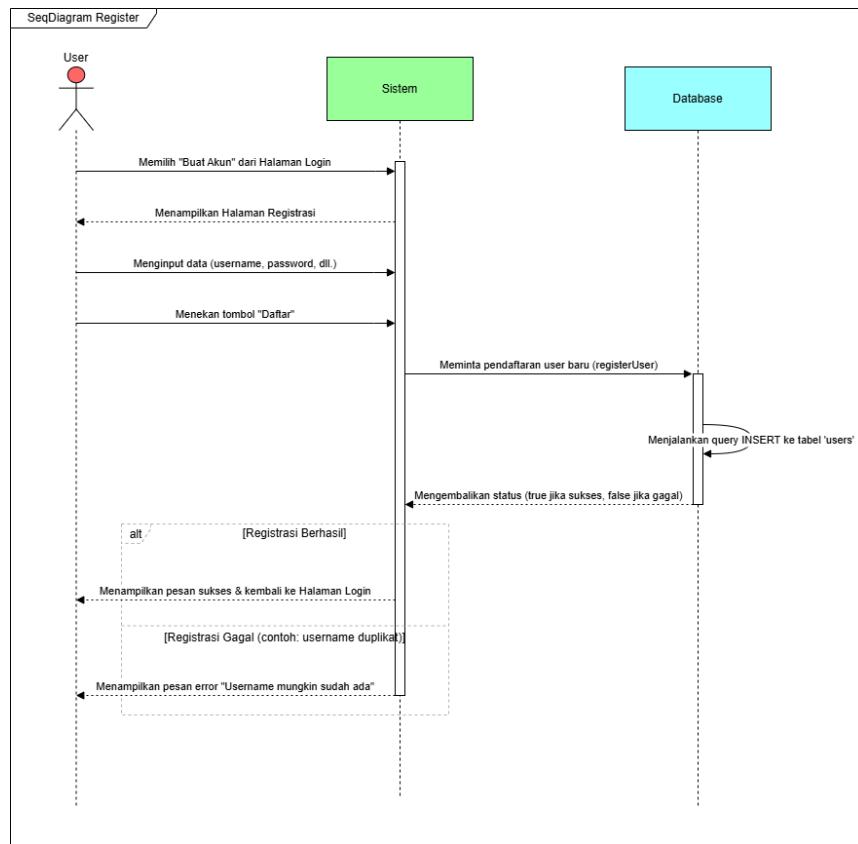
## 2. Aktor: Admin

Admin memiliki tanggung jawab untuk mengelola data dan memantau transaksi yang terjadi dalam sistem. Adapun aktivitas atau use case yang dapat dilakukan oleh Admin antara lain:

- Kelola Pesanan: Merupakan proses utama yang dapat diperluas dengan dua use case tambahan melalui relasi extend, yaitu:
  - Edit Status Pesanan: Admin dapat memperbarui status pesanan (misalnya: diproses, dikirim, selesai).
  - Hapus Pesanan: Admin dapat menghapus pesanan yang tidak valid atau bermasalah dari sistem.

### 2.3.2 Sequence Diagram

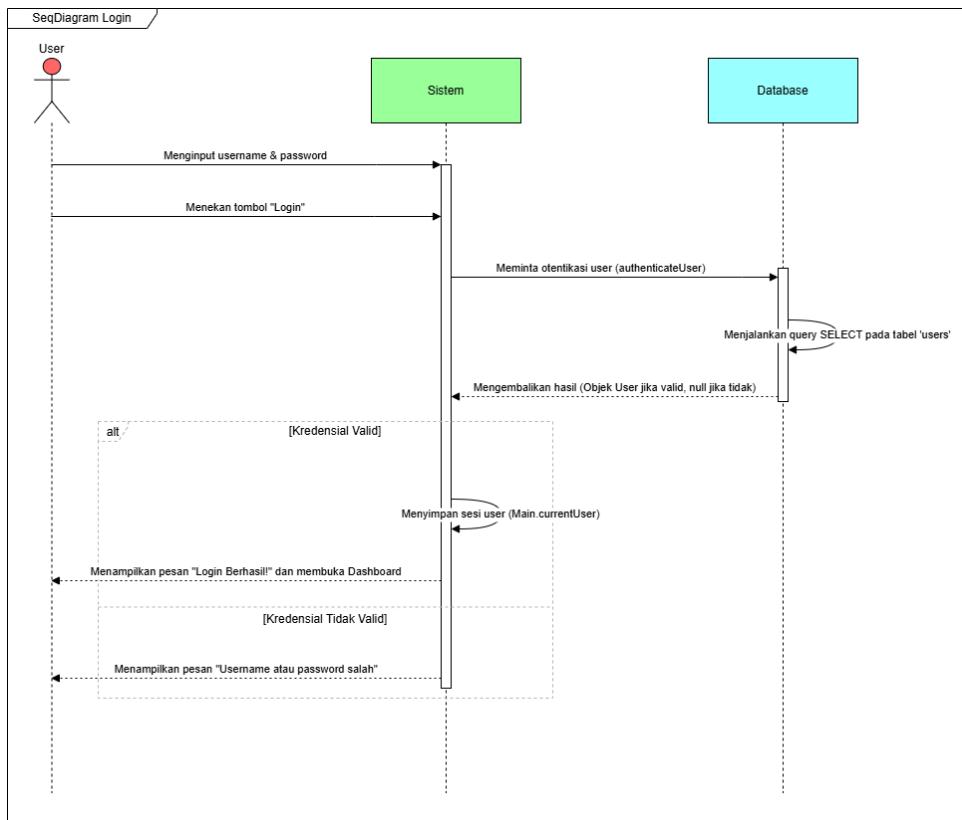
#### a. Register



1. Proses dimulai ketika User memilih opsi "Buat Akun" dari halaman login.
2. Sistem kemudian menampilkan halaman registrasi yang berisi form isian seperti username, password, dan informasi pendukung lainnya.
3. Setelah mengisi data yang diperlukan, User menekan tombol "Daftar".
4. Sistem menerima input dari user dan kemudian mengirim permintaan pendaftaran user baru ke database melalui fungsi registerUser.
5. Database menjalankan query INSERT untuk menyimpan data user ke dalam tabel users.
  - Jika query berhasil dijalankan (tidak ada konflik seperti username ganda), maka status yang dikembalikan ke sistem adalah true.
  - Jika terjadi kesalahan (misalnya, username sudah digunakan), maka status dikembalikan sebagai false.
6. Setelah mendapatkan respon dari database, sistem akan:
  - Menampilkan pesan sukses jika registrasi berhasil, dan mengarahkan user kembali ke halaman login.

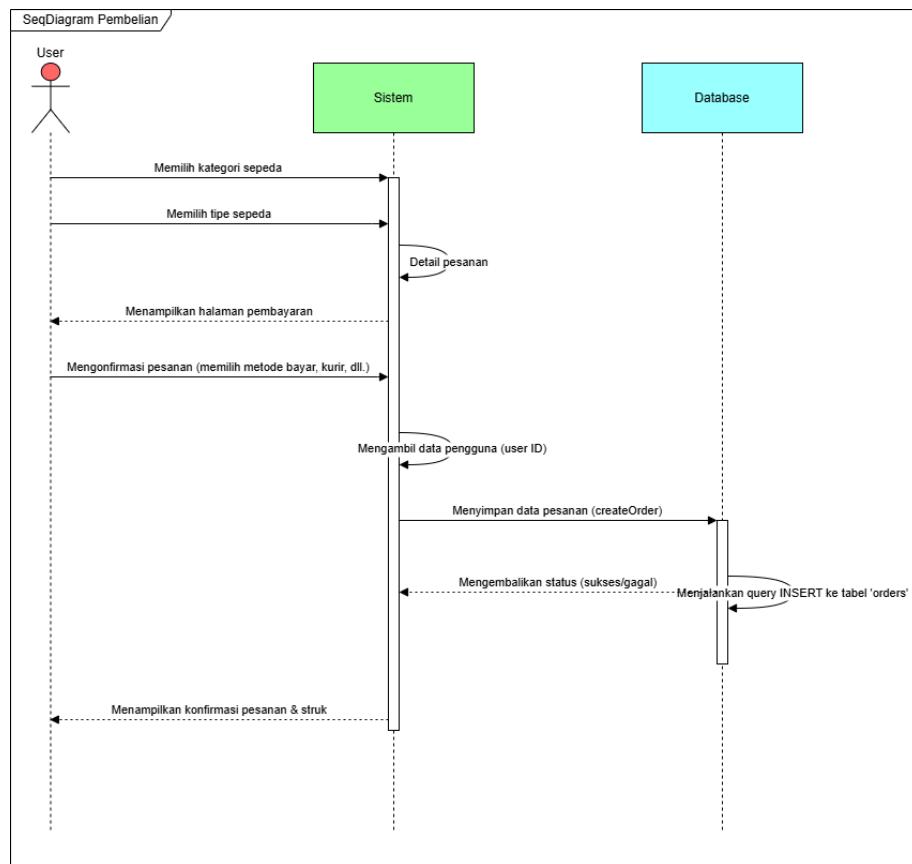
- Menampilkan pesan error seperti "Username mungkin sudah ada" jika proses gagal, tanpa mengarahkan ke halaman login.

## b. Login



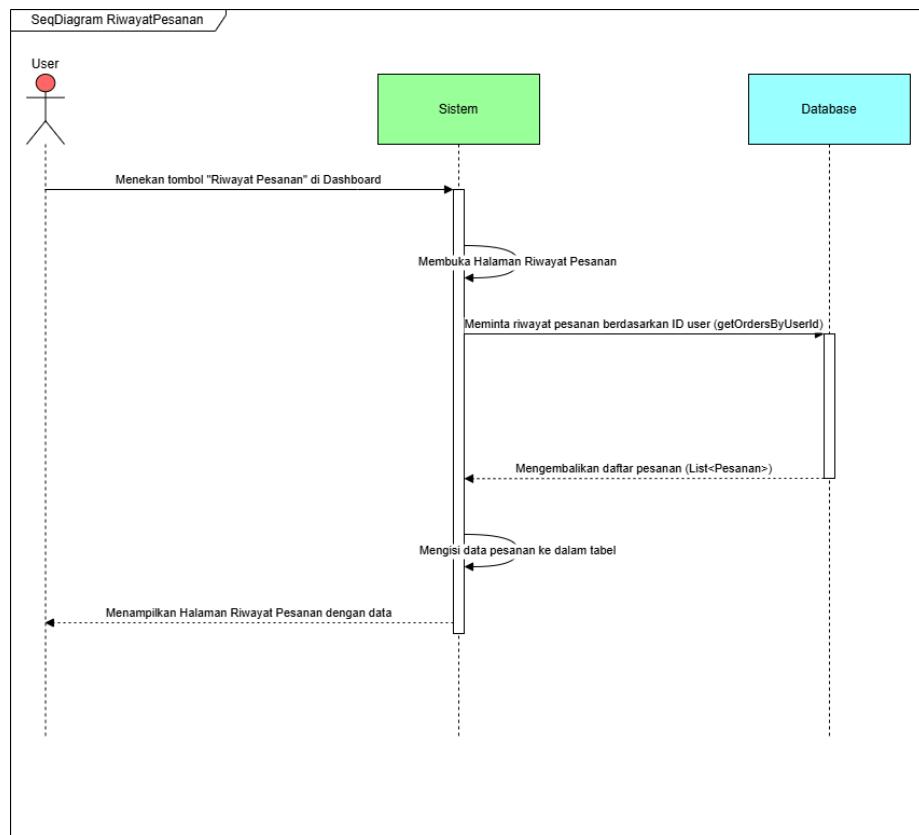
- Proses dimulai ketika User menginputkan username dan password pada halaman login.
- User kemudian menekan tombol "Login", yang mengirimkan data ke sistem.
- Sistem mengirim permintaan autentikasi ke database melalui fungsi authenticateUser.
- Database menjalankan query SELECT untuk mencari kecocokan username dan password pada tabel users.
- Database kemudian mengembalikan hasil:
  - Objek User jika data valid.
  - null jika data tidak ditemukan (tidak valid).
- Setelah mendapatkan hasil:
  - Jika kredensial valid, sistem akan menyimpan sesi user dan menampilkan pesan "Login Berhasil" serta mengarahkan ke dashboard.
  - Jika kredensial tidak valid, sistem akan menampilkan pesan error, misalnya "Username atau password salah".

### c. Pembelian



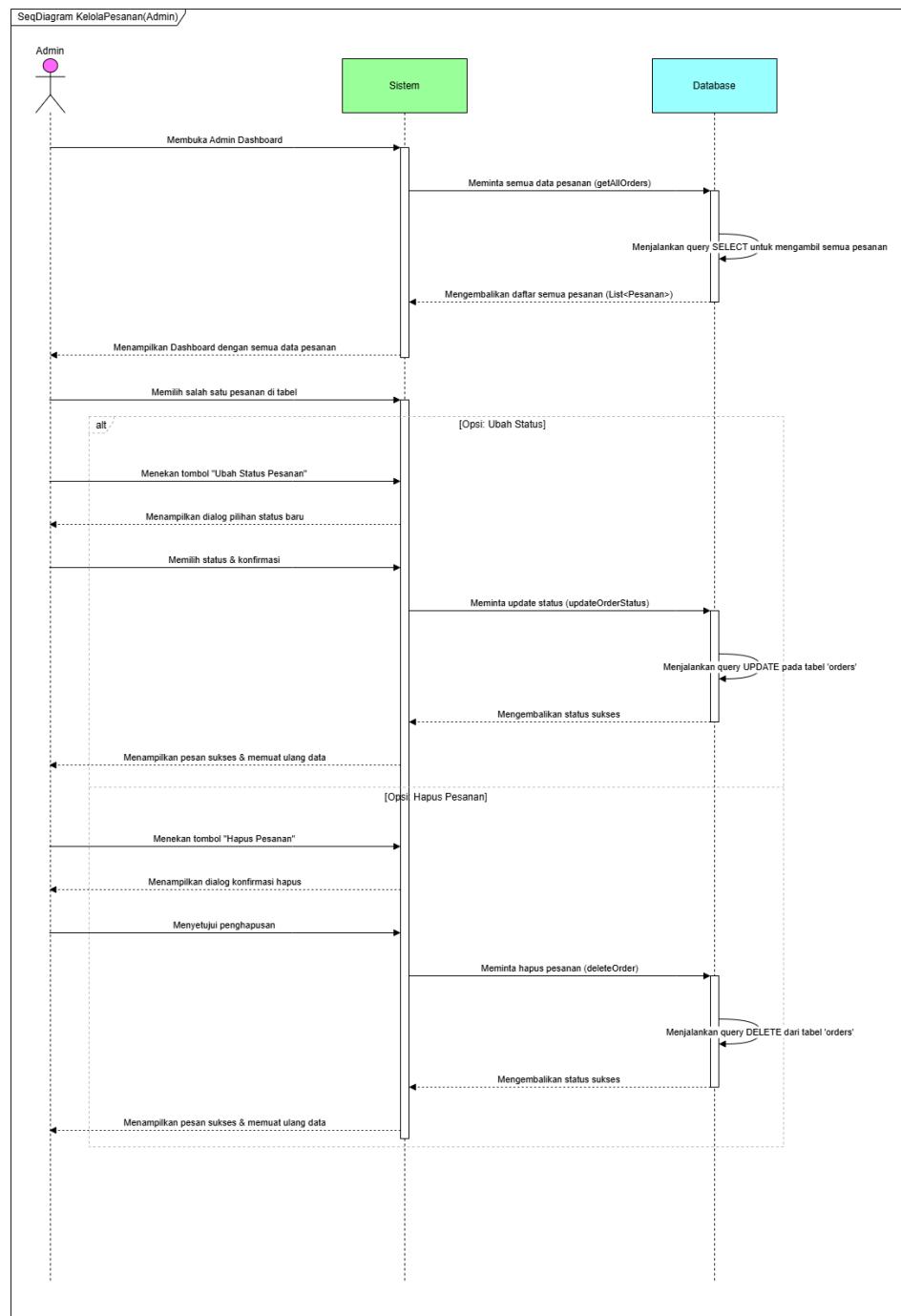
1. User memilih kategori sepeda, kemudian memilih tipe sepeda yang diinginkan.
2. Sistem menerima pilihan dan menampilkan detail pesanan untuk dikonfirmasi.
3. Sistem kemudian menampilkan halaman pembayaran, yang mencakup pilihan metode bayar dan jasa pengiriman.
4. Setelah itu, user melakukan konfirmasi pesanan.
5. Sistem mengambil data pengguna (user ID) untuk dicatat sebagai pemilik pesanan.
6. Sistem kemudian mengirim data pesanan (createOrder) ke database.
7. Database menjalankan query INSERT untuk menyimpan pesanan ke dalam tabel orders.
8. Setelah proses berhasil atau gagal, sistem mengembalikan status pemesanan (sukses/gagal).
9. Terakhir, sistem akan menampilkan konfirmasi pesanan dan struk kepada user.

#### d. Melihat Riwayat Pesanan



1. User menekan tombol "Riwayat Pesanan" di dashboard aplikasi.
2. Sistem kemudian membuka halaman Riwayat Pesanan.
3. Sistem mengirim permintaan getOrdersById ke database dengan menyertakan ID user sebagai parameter.
4. Database menjalankan query SELECT untuk mengambil semua pesanan yang dimiliki oleh user tersebut.
5. Hasil berupa daftar pesanan (List<Pesanan>) dikembalikan ke sistem.
6. Sistem mengisi data pesanan ke dalam tabel pada halaman.
7. Akhirnya, halaman Riwayat Pesanan ditampilkan kepada user dengan data yang telah dimuat.

## e. Kelola Pesanan (Admin)



1. Admin membuka dashboard, lalu sistem meminta semua data pesanan ke database melalui fungsi getAllOrders.
2. Database menjalankan query SELECT untuk mengambil seluruh isi dari tabel orders.
3. Hasilnya adalah daftar semua pesanan yang ditampilkan pada dashboard.

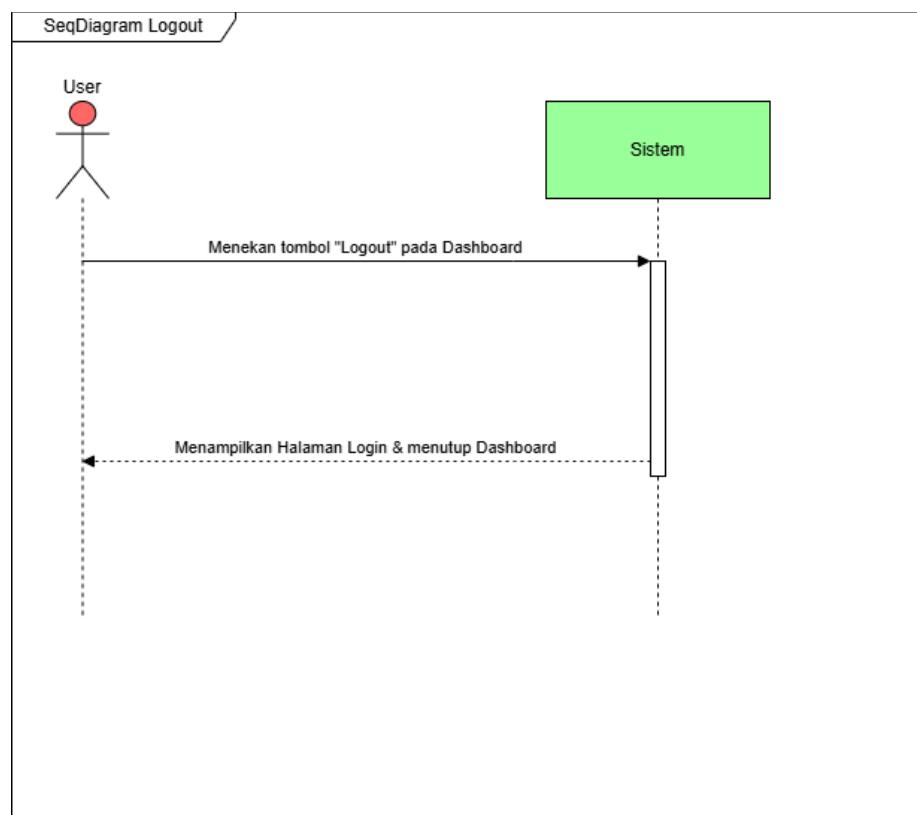
### ❖ Ubah Status Pesanan

4. Admin memilih salah satu pesanan, lalu menekan tombol "Ubah Status Pesanan".
5. Sistem menampilkan dialog pilihan status baru.
6. Setelah admin memilih status dan konfirmasi:
7. Sistem mengirim permintaan updateOrderStatus ke database.
8. Database menjalankan query UPDATE pada tabel orders.
9. Sistem menerima status sukses dan menampilkan pesan sukses + memuat ulang data.

❖ Hapus Pesanan

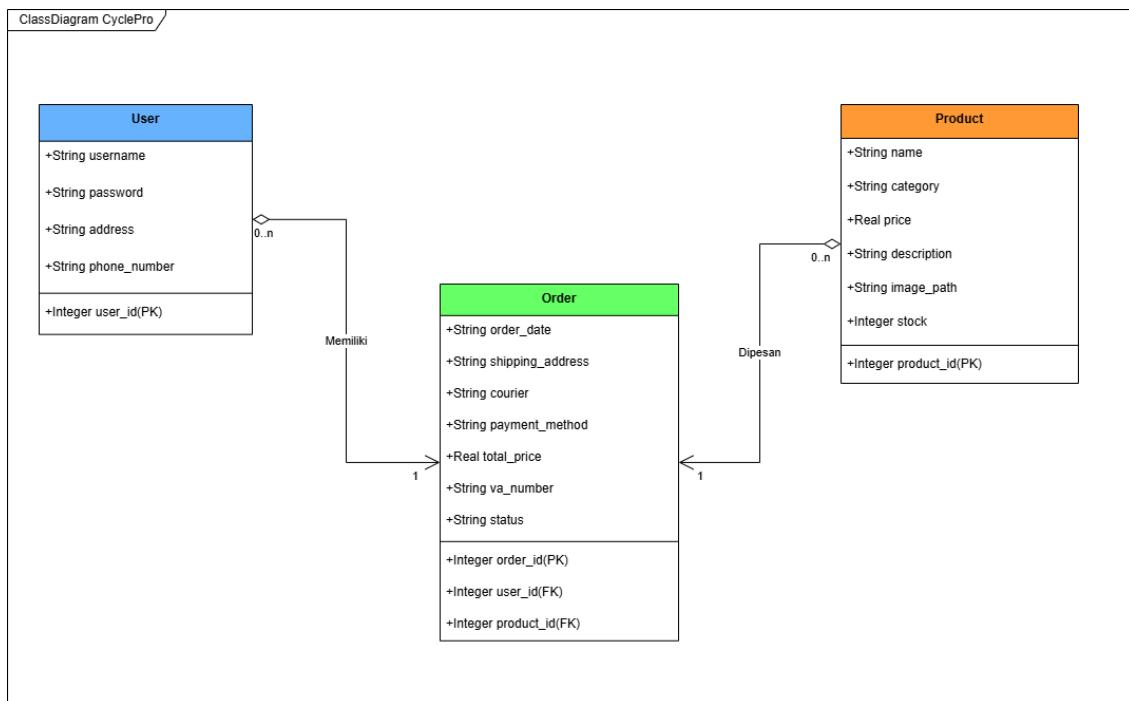
10. Admin menekan tombol "Hapus Pesanan".
11. Sistem menampilkan dialog konfirmasi.
12. Admin menyetujui penghapusan:
13. Sistem mengirim permintaan deleteOrder ke database.
14. Database menjalankan query DELETE dari tabel orders.
15. Sistem menerima status sukses dan kembali memuat ulang data + menampilkan pesan.

**f. Logout**



1. User menekan tombol "Logout" yang tersedia di dashboard atau menu navigasi.
2. Sistem menghapus data sesi pengguna.
3. Sistem mengarahkan pengguna kembali ke halaman login.
4. Halaman login ditampilkan, dan pengguna harus memasukkan kredensial lagi untuk mengakses sistem.

### 2.3.3 Class Diagram



Class diagram untuk sistem e-commerce CyclePro terdiri dari tiga kelas utama: User, Product, dan Order. Kelas User digunakan untuk menyimpan data mengenai pengguna yang terdaftar dalam aplikasi. Setiap entri memiliki atribut user\_id yang berfungsi sebagai primary key, username dan password untuk login, serta address dan phone\_number untuk data pribadi dan pengiriman.

Kelas Product menyimpan data mengenai produk sepeda yang dijual. Kelas ini memiliki atribut product\_id sebagai primary key, name yang menyimpan nama produk, category, price, description, image\_path untuk detail produk, dan stock untuk jumlah persediaan.

Kelas Order mencatat semua transaksi pembelian yang dilakukan oleh pengguna. Setiap pesanan memiliki order\_id sebagai primary key, serta atribut lain seperti order\_date, shipping\_address, courier, payment\_method, total\_price, va\_number, dan status pesanan. Kelas ini juga memiliki dua foreign key: user\_id yang merujuk ke kelas User dan product\_id yang merujuk ke kelas Product. Ini menunjukkan bahwa setiap

pesanan terhubung dengan satu pengguna dan satu produk tertentu.

Relasi antar kelas dalam diagram ini memastikan data transaksi terstruktur dengan baik. Misalnya, setiap Order harus terhubung dengan satu User tertentu melalui user\_id. Demikian pula, setiap Order harus terhubung dengan satu Product tertentu melalui product\_id. Struktur ini memungkinkan pengelolaan data pesanan yang efisien dan terintegrasi, menghubungkan siapa yang membeli dengan apa yang dibeli dalam sistem e-commerce CyclePro.

## BAB III

### IMPLEMENTASI DAN PENGUJIAN

#### 3.1 Implementasi Kode Program

##### 3.1.1 Pendahuluan & Arsitektur Umum

###### 1. Main.java

```
1  import javax.swing.SwingUtilities;
2
3  public class Main {
4      public static User currentUser;
5
6      Run | Debug
7      public static void main(String[] args) {
8          DatabaseHelper.createNewDatabase();
9          DatabaseHelper.createTables();
10         DatabaseHelper.addDefaultAdmin();
11
12         SwingUtilities.invokeLater(() -> {
13             HalamanLogin loginPage = new HalamanLogin();
14             loginPage.setVisible(b:true);
15         });
16     }
}
```

###### Penjelasan Kode Program

Kelas Main.java adalah kelas yang paling fundamental dalam proyek ini karena berfungsi sebagai titik awal (*entry point*) dari aplikasi CyclePro.

- Mendeklarasikan public class Main untuk menjadikan kelas ini sebagai kelas utama yang akan dieksekusi pertama kali.
- Terdapat atribut public static User currentUser. Atribut ini berfungsi sebagai *session manager* sederhana yang menyimpan data pengguna yang sedang login. Karena bersifat public static, data ini dapat diakses secara global dari kelas mana pun untuk mengetahui siapa pengguna yang aktif.
- Terdapat metode public static void main(String[] args) yang merupakan metode utama yang dijalankan saat aplikasi pertama kali dibuka.
- Di dalam metode main, terdapat tiga panggilan metode statis dari DatabaseHelper:
  - DatabaseHelper.createNewDatabase(): Memastikan file database telah dibuat.
  - DatabaseHelper.createTables(): Memastikan semua tabel yang dibutuhkan (seperti users, produk, pesanan) telah ada di dalam database.
  - DatabaseHelper.addDefaultAdmin(): Menambahkan akun admin default ke dalam database, yang berguna untuk login pertama kali.
- Terdapat pemanggilan SwingUtilities.invokeLater(). Ini adalah praktik terbaik dalam

Swing untuk memastikan bahwa semua kode yang berhubungan dengan antarmuka pengguna (GUI), seperti pembuatan HalamanLogin, dijalankan pada *thread* yang aman dan terpisah (Event Dispatch Thread).

- Di dalam invokeLater, sebuah *instance* dari HalamanLogin dibuat dan kemudian ditampilkan (setVisible(true)) kepada pengguna sebagai jendela pertama aplikasi.

### Penjelasan Konsep OOP

- Menggunakan konsep Desain Berorientasi Objek (*Object-Oriented Design*), yaitu:
  - Kelas Main menunjukkan pemisahan tanggung jawab (*separation of concerns*). Tidak menangani logika database atau tampilan secara langsung, melainkan mendelegasikan tugas-tugas tersebut ke objek lain, yaitu DatabaseHelper untuk urusan database dan HalamanLogin untuk urusan tampilan.
- Menggunakan Anggota Statis (*Static Members*), yaitu:
  - Atribut currentUser dan semua metode dari DatabaseHelper dideklarasikan sebagai static. Hal ini memungkinkan mereka untuk diakses secara langsung melalui nama kelasnya (misalnya, Main.currentUser atau DatabaseHelper.createTables()) tanpa perlu membuat objek dari kelas tersebut. Ini adalah pendekatan yang efektif untuk data global dan fungsionalitas utilitas.

## 2. Colors.java

```
1 import java.awt.Color;
2
3 public class Colors {
4
5     private Colors() {}
6
7     // Warna Latar Belakang Utama Aplikasi
8     public static final Color BACKGROUND_PRIMARY = new Color(rgb:0xF4F6F8);
9
10    // Warna Latar Belakang Sekunder
11    public static final Color BACKGROUND_SECONDARY = Color.WHITE;
12
13    // Warna untuk Komponen seperti Navbar atau Panel Header
14    public static final Color NAVBAR_BACKGROUND = new Color(rgb:0xD4AF37);
15    public static final Color NAVBAR_BACKGROUND_DARKER = new Color(rgb:0xB59874);
16    public static final Color NAVBAR_TEXT = Color.WHITE;
17
18    // Warna untuk Tombol Utama
19    public static final Color BUTTON_PRIMARY_BACKGROUND = new Color(rgb:0x3498DB);
20    public static final Color BUTTON_PRIMARY_TEXT = Color.WHITE;
21    public static final Color BUTTON_PRIMARY_HOVER_BACKGROUND = new Color(rgb:0x2980B9);
22    public static final Color BUTTON_GOLD_BACKGROUND = new Color(rgb:0xD4AF37);
23
24    // Warna untuk Tombol Sekunder
25    public static final Color BUTTON_SECONDARY_BACKGROUND = new Color(rgb:0x95A5A6);
26    public static final Color BUTTON_SECONDARY_TEXT = Color.WHITE;
27    public static final Color BUTTON_SECONDARY_HOVER_BACKGROUND = new Color(rgb:0x7F8C8D);
28
29    // Warna untuk Tombol Aksi Penting
30    public static final Color BUTTON_SUCCESS_BACKGROUND = new Color(rgb:0xECC71);
31    public static final Color BUTTON_SUCCESS_TEXT = Color.WHITE;
32    public static final Color BUTTON_SUCCESS_HOVER_BACKGROUND = new Color(rgb:0x27AE60);
33
34    // Warna untuk Tombol Peringatan atau Batal
35    public static final Color BUTTON_DANGER_BACKGROUND = new Color(rgb:0xE74C3C);
36    public static final Color BUTTON_DANGER_TEXT = Color.WHITE;
37    public static final Color BUTTON_DANGER_HOVER_BACKGROUND = new Color(rgb:0xC0392B);
38
39    // Warna Teks Utama
40    public static final Color TEXT_PRIMARY = new Color(rgb:0x333333);
41
42    // Warna Teks Sekunder
43    public static final Color TEXT_SECONDARY = new Color(rgb:0x777777);
44
45    // Warna untuk Border atau Garis Pemisah
46    public static final Color BORDER_COLOR = new Color(rgb:0xBD3C7);
47
48    public static final Color TEXT_LINK = new Color(rgb:0x3498DB);
49
50    // Warna baru untuk gradasi dan background input field
51    public static final Color BACKGROUND_LIGHT_GREY_GRADIENT = new Color(r:240, g:240, b:240);
52    public static final Color INPUT_FIELD_LIGHT_BACKGROUND = new Color(r:250, g:250, b:250);
53 }
```

### Penjelasan Kode Program

Kelas Colors.java adalah sebuah kelas utilitas (*utility class*) yang dirancang khusus untuk menjadi pusat pengelolaan semua definisi warna yang digunakan di dalam aplikasi CyclePro.

- Mendeklarasikan public class Colors untuk menjadikan kelas ini dapat diakses dari kelas lain di dalam proyek.
- Terdapat konstruktor private Colors(). Konstruktor ini sengaja dibuat privat untuk mencegah kelas lain membuat objek dari kelas Colors. Ini adalah pola desain umum untuk kelas utilitas yang semua anggotanya bersifat statis.
- Terdapat serangkaian atribut yang dideklarasikan sebagai public static final Color.

Kombinasi *modifier* ini menciptakan konstanta warna global yang tidak dapat diubah.

- public: Memungkinkan atribut diakses dari kelas mana pun.
- static: Menandakan bahwa atribut tersebut milik kelas Colors itu sendiri, bukan milik objek. Ini memungkinkan pemanggilan langsung seperti Colors.BACKGROUND\_PRIMARY tanpa perlu membuat *instance*.
- final: Memastikan bahwa nilai warna yang telah ditetapkan tidak dapat diubah selama program berjalan.
- Konstanta-konstanta warna ini dikelompokkan secara logis berdasarkan fungsinya (misalnya, BACKGROUND\_PRIMARY, NAVBAR\_BACKGROUND, BUTTON\_SUCCESS\_BACKGROUND, TEXT\_PRIMARY) untuk meningkatkan keterbacaan dan mempermudah pemeliharaan kode.

### Penjelasan Konsep OOP

- Menggunakan konsep Enkapsulasi, yaitu:
  - Konstruktor kelas Colors dideklarasikan sebagai private. Ini adalah bentuk penyembunyian informasi (*information hiding*) yang mencegah pembuatan objek dari kelas ini, sehingga menegakkan perannya sebagai kelas utilitas murni.
- Menggunakan konsep Abstraksi, yaitu:
  - Kelas ini menyembunyikan detail implementasi dari nilai-nilai warna (misalnya, kode heksadesimal 0xF4F6F8) di balik nama-nama yang deskriptif dan mudah dipahami (seperti BACKGROUND\_PRIMARY).
  - Dengan memusatkan semua definisi warna dalam satu file, aplikasi menjadi lebih mudah dikelola. Jika ada perubahan tema visual, programmer hanya perlu mengubah nilai di dalam kelas Colors tanpa harus mencari dan mengganti nilai warna di banyak file berbeda.
- Menggunakan Anggota Statis (*Static Members*), yaitu:
  - Semua atribut warna dideklarasikan sebagai static, sehingga mereka dapat diakses langsung melalui nama kelasnya (Colors.NAMA\_WARNA). Ini adalah pendekatan yang efisien untuk menyediakan nilai-nilai konstanta global di seluruh aplikasi.

### 3.1.2 Struktur Data (Model)

#### 3. User.java

```
1  public class User {
2      private int userId;
3      private String username;
4      private String password;
5      private String address;
6      private String phoneNumber;
7
8      public User(int userId, String username, String password, String address, String phoneNumber) {
9          this.userId = userId;
10         this.username = username;
11         this.password = password;
12         this.address = address;
13         this.phoneNumber = phoneNumber;
14     }
15
16     // Getter
17     public int getUserId() {
18         return userId;
19     }
20     public String getUsername() {
21         return username;
22     }
23     public String getPassword() {
24         return password;
25     }
26     public String getAddress() {
27         return address;
28     }
29     public String getPhoneNumber() {
30         return phoneNumber;
31     }
32 }
```

#### Penjelasan Kode Program

Kelas User.java adalah sebuah *Plain Old Java Object* (POJO) yang berfungsi sebagai model atau cetak biru (*blueprint*) untuk merepresentasikan data pengguna dalam sistem CyclePro.

- Mendeklarasikan public class User untuk menjadikan kelas ini sebagai tipe data yang dapat diakses dan digunakan oleh kelas lain.
- Terdapat beberapa atribut yang dideklarasikan dengan hak akses private, seperti userId, username, password, address, dan phoneNumber. Hak akses private berarti atribut-atribut ini hanya dapat diakses dari dalam kelas User itu sendiri, menyembunyikan data dari akses luar.
- Terdapat public User(...) yang merupakan sebuah konstruktur. Metode ini dipanggil saat sebuah objek User baru dibuat dan berfungsi untuk menginisialisasi semua nilai atribut (userId, username, dll.) berdasarkan parameter yang diberikan.
- Terdapat serangkaian metode *getter* publik (seperti getUserId(), getUsername(), getAddress()). Metode-metode ini menyediakan akses *read-only* (hanya baca) ke nilai-nilai atribut privat dari luar kelas.
- Dalam kelas ini tidak terdapat metode *setter* (misalnya, setIdUser()). Ini berarti

setelah sebuah objek User dibuat melalui konstruktor, data di dalamnya tidak dapat diubah (bersifat *immutable*).

### Penjelasan Konsep OOP

- Menggunakan konsep Enkapsulasi
  - Semua atribut (userId, username, dll.) dideklarasikan sebagai private untuk menyembunyikan data dan melindunginya dari modifikasi langsung dari luar kelas.
  - Metode *getter* (getUserId(), getUsername(), dll.) dideklarasikan sebagai public untuk menyediakan akses yang terkontrol dan aman ke atribut-atribut privat tersebut.
  - Dengan membungkus data (atribut) dan metode yang beroperasi padanya (konstruktor dan *getter*) ke dalam satu unit tunggal (kelas User), kelas ini menerapkan prinsip dasar enkapsulasi.
- Menggunakan Konstruktor
  - Metode public User(int userId, ...) adalah konstruktor yang digunakan untuk membuat dan menginisialisasi sebuah objek User baru dengan nilai-nilai spesifik. Setiap kali new User(...) dipanggil, konstruktor inilah yang dieksekusi.
- Menggunakan konsep Abstraksi
  - Kelas User menyajikan model sederhana dari "pengguna" di dunia nyata, dengan hanya menyertakan atribut dan perilaku yang relevan untuk aplikasi ini (userId, username, dll.). Detail-detail lain yang tidak relevan diabaikan.

#### 4. Admin.java

```
1 ~ public class Admin extends User {
2     private String role;
3
4 ~     public Admin(int userId, String username, String password, String address, String phoneNumber) {
5         super(userId, username, password, address, phoneNumber);
6         this.role = "admin";
7     }
8
9 ~     public String getRole() {
10        return role;
11    }
12 }
```

### Penjelasan Kode Program

Kelas Admin.java adalah model data yang merepresentasikan seorang administrator dalam sistem. Kelas ini merupakan spesialisasi dari kelas User, yang berarti seorang Admin memiliki semua atribut dan perilaku seorang User, ditambah dengan properti spesifik admin.

- Mendeklarasikan public class Admin extends User. Pernyataan extends User menandakan bahwa kelas Admin adalah kelas turunan (subclass) dari kelas User (superclass). Ini adalah implementasi dari konsep pewarisan (*Inheritance*).
- Terdapat atribut private String role. Atribut ini khusus untuk kelas Admin dan berfungsi untuk menyimpan peran pengguna, yang dalam hal ini selalu diatur sebagai "admin".
- Terdapat public Admin(...) yang merupakan sebuah konstruktor.
  - Di dalam konstruktor, terdapat pemanggilan super(...). Perintah super ini digunakan untuk memanggil konstruktor dari kelas induknya (User) dan meneruskan semua parameter yang diperlukan (userId, username, dll.). Ini memastikan bahwa semua atribut yang diwarisi dari User diinisialisasi dengan benar.
  - Setelah itu, atribut role diinisialisasi dengan nilai "admin".
- Terdapat metode *getter* public String getRole(). Metode ini menyediakan akses hanya-baca (*read-only*) ke nilai atribut role.

### **Penjelasan Konsep OOP**

- Menggunakan konsep Pewarisan (*Inheritance*)
  - Kelas Admin mewarisi semua atribut (seperti userId, username) dan metode (seperti getUsername(), getAddress()) dari kelas User. Ini menghilangkan kebutuhan untuk menulis ulang kode yang sama dan menciptakan hubungan "is-a" (seorang Admin *adalah* seorang User).
- Menggunakan konsep Enkapsulasi
  - Atribut role dideklarasikan sebagai private, menyembunyikan detail implementasi dari luar kelas.
  - Akses ke atribut role dikontrol melalui metode public String getRole(), yang menyediakan cara yang aman untuk membaca nilainya.
- Menggunakan super untuk Memanggil Konstruktor Induk:
  - Kata kunci super digunakan di dalam konstruktor Admin untuk memanggil konstruktor dari kelas User. Ini adalah mekanisme standar dalam OOP untuk memastikan bahwa inisialisasi pada kelas induk tetap dijalankan saat membuat objek dari kelas turunan.



## 5. Produk.java

```
1  public class Produk {
2      private int productId;
3      private String name;
4      private String category;
5      private double price;
6      private String description;
7      private String imagePath;
8      private int stock;
9
10     public Produk(int productId, String name, String category, double price, String description, String imagePath, int stock) {
11         this.productId = productId;
12         this.name = name;
13         this.category = category;
14         this.price = price;
15         this.description = description;
16         this.imagePath = imagePath;
17         this.stock = stock;
18     }
19
20     // Getters
21     public int getProductId() { return productId; }
22     public String getName() { return name; }
23     public String getCategory() { return category; }
24     public double getPrice() { return price; }
25     public String getDescription() { return description; }
26     public String getImagePath() { return imagePath; }
27     public int getStock() { return stock; }
28
29     @Override
30     public String toString() {
31         return name + " - Rp" + String.format(format:"%.0f", price);
32     }
33 }
```

### Penjelasan Kode Program

Kelas Produk.java berfungsi sebagai model atau cetak biru (*blueprint*) untuk merepresentasikan data produk sepeda dalam sistem CyclePro.

- Mendeklarasikan public class Produk untuk menjadikan kelas ini sebagai tipe data yang dapat diakses dan digunakan oleh kelas lain.
- Terdapat beberapa atribut yang dideklarasikan dengan hak akses private, seperti productId, name, category, price, description, imagePath, dan stock. Hak akses private memastikan data ini hanya dapat diakses dari dalam kelas Produk itu sendiri.
- Terdapat public Produk(...) yang merupakan sebuah konstruktur. Metode ini digunakan saat sebuah objek Produk baru dibuat dan berfungsi untuk menginisialisasi semua nilai atributnya berdasarkan parameter yang diterima.
- Terdapat serangkaian metode *getter* publik (seperti getProductId(), getName(), getPrice()). Metode-metode ini menyediakan akses *read-only* (hanya baca) ke nilai-nilai atribut privat dari luar kelas.
- Terdapat metode @Override public String toString(). Metode ini meng-*override* (menimpa) metode toString() bawaan dari kelas Object. Tujuannya adalah untuk memberikan representasi String yang lebih informatif dan mudah dibaca dari sebuah objek Produk, yang sangat berguna saat menampilkan objek ini di komponen GUI seperti JComboBox atau JList.

## Penjelasan Konsep OOP

- Menggunakan konsep Enkapsulasi
  - Semua atribut data produk dideklarasikan sebagai private untuk menyembunyikan detail internal dan melindungi data dari akses langsung yang tidak sah.
  - Metode *getter* publik menyediakan "jendela" akses yang terkontrol untuk membaca nilai atribut tanpa memberikan izin untuk mengubahnya.
- Menggunakan Konstruktor
  - Metode public Produk(...) adalah konstruktor yang bertugas untuk memastikan setiap objek Produk yang dibuat berada dalam keadaan yang valid dan lengkap sejak awal, dengan semua atributnya telah diinisialisasi.
- Menggunakan konsep Abstraksi
  - Kelas Produk merupakan abstraksi dari produk sepeda di dunia nyata. Ia hanya menyertakan atribut dan perilaku yang relevan dan penting untuk kebutuhan aplikasi, seperti nama, harga, dan stok, sambil mengabaikan detail lain yang tidak diperlukan.
- Menggunakan konsep Polimorfisme
  - Penggunaan anotasi @Override pada metode `toString()` adalah contoh dari *method overriding*, salah satu bentuk polimorfisme. Kelas Produk menyediakan implementasi spesifiknya sendiri untuk metode `toString()` yang diwarisi dari kelas Object, sehingga perilaku metode tersebut berubah sesuai dengan konteks kelas Produk.

## 6. Pesanan.java

```
1  public class Pesanan {
2      private int orderId;
3      private int userId;
4      private int productId;
5      private String orderDate;
6      private String shippingAddress;
7      private String courier;
8      private String paymentMethod;
9      private double totalPrice;
10     private String vaNumber;
11     private String status;
12     private User user;
13     private Produk product;
14
15     // Konstruktor utama
16     public Pesanan(int orderId, int userId, int productId, String orderDate,
17                     String shippingAddress, String courier, String paymentMethod,
18                     double totalPrice, String vaNumber, String status) {
19         this.orderId = orderId;
20         this.userId = userId;
21         this.productId = productId;
22         this.orderDate = orderDate;
23         this.shippingAddress = shippingAddress;
24         this.courier = courier;
25         this.paymentMethod = paymentMethod;
26         this.totalPrice = totalPrice;
27         this.vaNumber = vaNumber;
28         this.status = status;
29     }
30
31     public Pesanan(int orderId, User user, Produk product, String orderDate,
32                     String shippingAddress, String courier, String paymentMethod,
33                     double totalPrice, String vaNumber, String status) {
34         this.orderId = orderId;
35         this.user = user;
36         this.product = product;
37         this.orderDate = orderDate;
38         this.shippingAddress = shippingAddress;
39         this.courier = courier;
40         this.paymentMethod = paymentMethod;
41         this.totalPrice = totalPrice;
42         this.vaNumber = vaNumber;
43         this.status = status;
44         this.userId = (user != null) ? user.getUserId() : 0;
45         this.productId = (product != null) ? product.getProductId() : 0;
46     }
47
48     // --- Getters ---
49     public int getOrderId() { return orderId; }
50     public int getUserId() { return userId; }
51     public int getProductId() { return productId; }
52     public String getOrderDate() { return orderDate; }
53     public String getShippingAddress() { return shippingAddress; }
54     public String getCourier() { return courier; }
55     public String getPaymentMethod() { return paymentMethod; }
56     public double getTotalPrice() { return totalPrice; }
57     public String getVaNumber() { return vaNumber; }
58     public String getStatus() { return status; }
59     public User getUser() { return user; }
60     public Produk getProduct() { return product; }
61
62     // --- Setters ---
63     public void setStatus(String status) { this.status = status; }
64     public void setShippingAddress(String shippingAddress) { this.shippingAddress = shippingAddress; }
65     public void setCourier(String courier) { this.courier = courier; }
66     public void setPaymentMethod(String paymentMethod) { this.paymentMethod = paymentMethod; }
67     public void setVaNumber(String vaNumber) { this.vaNumber = vaNumber; }
68
69
70     @Override
71     public String toString() {
72         return "Pesanan [orderId=" + orderId + ", userId=" + userId + ", productId=" + productId +
73             ", orderDate=" + orderDate + ", status=" + status + ", totalPrice=" + totalPrice + "]";
74     }
75 }
```

## Penjelasan Kode Program

Kelas Pesanan.java adalah sebuah model data yang berfungsi sebagai cetak biru (*blueprint*) untuk merepresentasikan data transaksi atau pesanan yang dibuat oleh pengguna dalam sistem CyclePro.

- Mendeklarasikan public class Pesanan untuk menjadikan kelas ini sebagai tipe data yang dapat diakses oleh kelas lain.
- Terdapat beberapa atribut yang dideklarasikan dengan hak akses private, seperti orderId, userId, productId, orderDate, shippingAddress, totalPrice, dan status. Selain itu, terdapat juga atribut yang bertipe objek, yaitu User user dan Produk product, untuk menyimpan referensi langsung ke objek pengguna dan produk yang terkait dengan pesanan tersebut.
- Terdapat dua buah konstruktor dengan parameter yang berbeda (*constructor overloading*).
  - Konstruktor pertama menerima parameter dasar termasuk userId dan productId. Ini berguna untuk membuat objek Pesanan dari data mentah yang diambil langsung dari database.
  - Konstruktor kedua menerima parameter berupa objek User dan Produk. Ini lebih mudah digunakan saat membuat pesanan baru di dalam aplikasi, karena objek User dan Produk yang relevan biasanya sudah tersedia. Konstruktor ini kemudian secara otomatis mengatur userId dan productId dari objek yang diterima.
- Terdapat serangkaian metode *getter* publik (seperti getOrderId(), getShippingAddress(), getUser()). Metode-metode ini menyediakan akses *read-only* (hanya baca) ke semua atribut privat.
- Terdapat beberapa metode *setter* publik (seperti setStatus(), setShippingAddress()). Berbeda dengan *getter*, *setter* hanya disediakan untuk atribut-atribut yang nilainya mungkin perlu diubah setelah objek pesanan dibuat, misalnya untuk memperbarui status pesanan.
- Terdapat metode @Override public String toString(). Metode ini menimpa metode toString() bawaan untuk memberikan representasi String dari objek Pesanan yang lebih ringkas dan informatif, yang berguna untuk keperluan *logging* atau *debugging*.

## Penjelasan Konsep OOP

- Menggunakan konsep Enkapsulasi
  - Semua atribut data pesanan dideklarasikan sebagai private untuk

- menyembunyikan detail implementasi dari dunia luar.
- Akses ke atribut dikontrol secara ketat melalui metode *getter* (untuk membaca) dan *setter* (untuk mengubah), yang menyediakan antarmuka publik yang aman.
  - Menggunakan konsep Polimorfisme
    - Adanya dua konstruktor dengan daftar parameter yang berbeda adalah contoh dari *constructor overloading*, salah satu bentuk polimorfisme statis (atau *compile-time polymorphism*).
    - Penggunaan anotasi @Override pada metode *toString()* adalah contoh dari *method overriding*, salah satu bentuk polimorfisme dinamis (*runtime polymorphism*).
  - Menggunakan konsep Abstraksi
    - Kelas Pesanan adalah sebuah abstraksi dari konsep pesanan di dunia nyata. Ia hanya merepresentasikan detail-detail yang penting dan relevan untuk aplikasi, seperti data pembeli, produk, alamat, dan status, sambil menyembunyikan kompleksitas lain yang tidak diperlukan.

### 3.1.3 Konektivitas Database

#### 7. DatabaseHelper.java

```

1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.PreparedStatement;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7  import java.util.ArrayList;
8  import java.util.List;
9
10 public class DatabaseHelper {
11     private static final String URL = "jdbc:sqlite:cyclepro.db";
12
13     public static Connection connect() {
14         Connection conn = null;
15         try {
16             conn = DriverManager.getConnection(URL);
17         } catch (SQLException e) {
18             System.out.println(e.getMessage());
19         }
20         return conn;
21     }
22
23     public static void createNewDatabase() {
24         try (Connection conn = connect()) {
25             if (conn != null) {
26                 System.out.println("A new database has been created or connection established.");
27             }
28         } catch (SQLException e) {
29             System.out.println(e.getMessage());
30         }
31     }
32 }
```

```

33     public static void createTables() {
34         String usersTable = "CREATE TABLE IF NOT EXISTS users (" +
35             + " user_id INTEGER PRIMARY KEY AUTOINCREMENT," +
36             + " username TEXT UNIQUE NOT NULL," +
37             + " password TEXT NOT NULL," +
38             + " address TEXT," +
39             + " phone_number TEXT NOT NULL," +
40             + " role TEXT DEFAULT 'user'" +
41             + ");";
42
43         String productsTable = "CREATE TABLE IF NOT EXISTS products (" +
44             + " product_id INTEGER PRIMARY KEY AUTOINCREMENT," +
45             + " name TEXT NOT NULL," +
46             + " category TEXT NOT NULL CHECK(category IN ('BMX', 'Gunung', 'Lipat'))," +
47             + " price REAL NOT NULL," +
48             + " description TEXT," +
49             + " image_path TEXT," +
50             + " stock INTEGER DEFAULT 10" +
51             + ");";
52
53         String ordersTable = "CREATE TABLE IF NOT EXISTS orders (" +
54             + " order_id INTEGER PRIMARY KEY AUTOINCREMENT," +
55             + " user_id INTEGER," +
56             + " product_id INTEGER," +
57             + " order_date TEXT DEFAULT CURRENT_TIMESTAMP," +
58             + " shipping_address TEXT," +
59             + " courier TEXT," +
60             + " payment_method TEXT," +
61             + " total_price REAL," +
62             + " va_number TEXT," +
63             + " status TEXT DEFAULT 'Pending'," +
64             + " FOREIGN KEY (user_id) REFERENCES users (user_id)," +
65             + " FOREIGN KEY (product_id) REFERENCES products (product_id)" +
66             + ");";
67
68     try (Connection conn = connect();
69          Statement stmt = conn.createStatement()) {
70         stmt.execute(usersTable);
71         stmt.execute(productsTable);
72         stmt.execute(ordersTable);
73         System.out.println("Tables created successfully (users table updated).");
74         addSampleProducts();
75     } catch (SQLException e) {
76         System.out.println("Error creating tables: " + e.getMessage());
77         e.printStackTrace();
78     }
79 }
```

```

80
81     public static void addSampleProducts() {
82         String checkProducts = "SELECT COUNT(*) AS count FROM products";
83         String insertProductSQL = "INSERT INTO products(name, category, price, description, image_path, stock) VALUES(?,?,?,?,?,?)";
84
85         try (Connection conn = connect();
86              Statement stmt = conn.createStatement();
87              ResultSet rs = stmt.executeQuery(checkProducts)) {
88
89             if (rs.next() && rs.getInt(columnLabel:"count") == 0) {
90                 System.out.println(x:"Adding sample products...");
91                 try (PreparedStatement pstmt = conn.prepareStatement(insertProductSQL)) {
92                     pstmt.setString(parameterIndex:1, x:"BMX Pro X1");
93                     pstmt.setString(parameterIndex:2, x:"BMX");
94                     pstmt.setDouble(parameterIndex:3, x:1500000);
95                     pstmt.setString(parameterIndex:4, x:"Sepeda BMX untuk freestyle");
96                     pstmt.setString(parameterIndex:5, x:"bmx1.png");
97                     pstmt.setInt(parameterIndex:6, x:10);
98                     pstmt.executeUpdate();
99
100                    pstmt.setString(parameterIndex:1, x:"BMX Street Master");
101                    pstmt.setString(parameterIndex:2, x:"BMX");
102                    pstmt.setDouble(parameterIndex:3, x:1200000);
103                    pstmt.setString(parameterIndex:4, x:"Sepeda BMX lincah di jalanan");
104                    pstmt.setString(parameterIndex:5, x:"bmx2.png");
105                    pstmt.setInt(parameterIndex:6, x:15);
106                    pstmt.executeUpdate();
107
108                    pstmt.setString(parameterIndex:1, x:"BMX Lite Fun");
109                    pstmt.setString(parameterIndex:2, x:"BMX");
110                    pstmt.setDouble(parameterIndex:3, x:900000);
111                    pstmt.setString(parameterIndex:4, x:"Sepeda BMX ringan untuk pemula");
112                    pstmt.setString(parameterIndex:5, x:"bmx3.png");
113                    pstmt.setInt(parameterIndex:6, x:20);
114                    pstmt.executeUpdate();
115
116                    pstmt.setString(parameterIndex:1, x:"BMX Racer Z");
117                    pstmt.setString(parameterIndex:2, x:"BMX");
118                    pstmt.setDouble(parameterIndex:3, x:1800000);
119                    pstmt.setString(parameterIndex:4, x:"Sepeda BMX untuk kecepatan");
120                    pstmt.setString(parameterIndex:5, x:"bmx4.png");
121                    pstmt.setInt(parameterIndex:6, x:8);
122                    pstmt.executeUpdate();
123
124                    pstmt.setString(parameterIndex:1, x:"BMX Park King");
125                    pstmt.setString(parameterIndex:2, x:"BMX");
126                    pstmt.setDouble(parameterIndex:3, x:1600000);
127                    pstmt.setString(parameterIndex:4, x:"Sepeda BMX untuk atraksi di taman");
128                    pstmt.setString(parameterIndex:5, x:"bmx5.png");
129                    pstmt.setInt(parameterIndex:6, x:12);
130                    pstmt.executeUpdate();
131
132                    pstmt.setString(parameterIndex:1, x:"BMX Dirt Jumper");
133                    pstmt.setString(parameterIndex:2, x:"BMX");
134                    pstmt.setDouble(parameterIndex:3, x:1700000);
135                    pstmt.setString(parameterIndex:4, x:"Sepeda BMX untuk trek tanah");
136                    pstmt.setString(parameterIndex:5, x:"bmx6.png");
137                    pstmt.setInt(parameterIndex:6, x:7);
138                    pstmt.executeUpdate();
139
140                    pstmt.setString(parameterIndex:1, x:"Gunung Everest Peak");
141                    pstmt.setString(parameterIndex:2, x:"Gunung");
142                    pstmt.setDouble(parameterIndex:3, x:3500000);
143                    pstmt.setString(parameterIndex:4, x:"Sepeda gunung tangguh semua medan");
144                    pstmt.setString(parameterIndex:5, x:"gunung1.png");
145                    pstmt.setInt(parameterIndex:6, x:5);
146                    pstmt.executeUpdate();
147
148                    pstmt.setString(parameterIndex:1, x:"Gunung TrailBlazer X");
149                    pstmt.setString(parameterIndex:2, x:"Gunung");
150                    pstmt.setDouble(parameterIndex:3, x:2800000);
151                    pstmt.setString(parameterIndex:4, x:"Sepeda gunung untuk petualangan trail");
152                    pstmt.setString(parameterIndex:5, x:"gunung2.png");
153                    pstmt.setInt(parameterIndex:6, x:8);
154                    pstmt.executeUpdate();
155
156                    pstmt.setString(parameterIndex:1, x:"Gunung RockHopper");
157                    pstmt.setString(parameterIndex:2, x:"Gunung");
158                    pstmt.setDouble(parameterIndex:3, x:2000000);
159                    PreparedStatement pstmt - DatabaseHelper.addSampleProducts() yaman";
160                    pstmt.setString(parameterIndex:5, x:"gunung3.png");
161                    pstmt.setInt(parameterIndex:6, x:12);
162                    pstmt.executeUpdate();
163

```

```

164     pstmt.setString(parameterIndex:1, x:"Gunung Summit Seeker");
165     pstmt.setString(parameterIndex:2, x:"Gunung");
166     pstmt.setDouble(parameterIndex:3, x:400000);
167     pstmt.setString(parameterIndex:4, x:"Sepeda gunung performa tinggi");
168     pstmt.setString(parameterIndex:5, x:"gunung4.png");
169     pstmt.setInt(parameterIndex:6, x:4);
170     pstmt.executeUpdate();
171
172     pstmt.setString(parameterIndex:1, x:"Gunung Forest Rider");
173     pstmt.setString(parameterIndex:2, x:"Gunung");
174     pstmt.setDouble(parameterIndex:3, x:250000);
175     pstmt.setString(parameterIndex:4, x:"Sepeda gunung untuk menjelajah hutan");
176     pstmt.setString(parameterIndex:5, x:"gunung5.png");
177     pstmt.setInt(parameterIndex:6, x:10);
178     pstmt.executeUpdate();
179
180     pstmt.setString(parameterIndex:1, x:"Gunung Canyon Carver");
181     pstmt.setString(parameterIndex:2, x:"Gunung");
182     pstmt.setDouble(parameterIndex:3, x:320000);
183     pstmt.setString(parameterIndex:4, x:"Sepeda gunung untuk medan curam");
184     pstmt.setString(parameterIndex:5, x:"gunung6.png");
185     pstmt.setInt(parameterIndex:6, x:6);
186     pstmt.executeUpdate();
187
188     pstmt.setString(parameterIndex:1, x:"Lipat CityCommute");
189     pstmt.setString(parameterIndex:2, x:"Lipat");
190     pstmt.setDouble(parameterIndex:3, x:180000);
191     pstmt.setString(parameterIndex:4, x:"Sepeda lipat praktis untuk perkotaan");
192     pstmt.setString(parameterIndex:5, x:"lipat1.png");
193     pstmt.setInt(parameterIndex:6, x:15);
194     pstmt.executeUpdate();
195
196     pstmt.setString(parameterIndex:1, x:"Lipat UrbanFold");
197     pstmt.setString(parameterIndex:2, x:"Lipat");
198     pstmt.setDouble(parameterIndex:3, x:150000);
199     pstmt.setString(parameterIndex:4, x:"Sepeda lipat ringkas dan stylish");
200     pstmt.setString(parameterIndex:5, x:"lipat2.png");
201     pstmt.setInt(parameterIndex:6, x:18);
202     pstmt.executeUpdate();
203
204     pstmt.setString(parameterIndex:1, x:"Lipat QuickFold");
205     pstmt.setString(parameterIndex:2, x:"Lipat");
206     pstmt.setDouble(parameterIndex:3, x:200000);
207     pstmt.setString(parameterIndex:4, x:"Sepeda lipat mudah dibawa");
208     pstmt.setString(parameterIndex:5, x:"lipat3.png");
209     pstmt.setInt(parameterIndex:6, x:10);
210     pstmt.executeUpdate();
211
212     pstmt.setString(parameterIndex:1, x:"Lipat Metro Lite");
213     pstmt.setString(parameterIndex:2, x:"Lipat");
214     pstmt.setDouble(parameterIndex:3, x:160000);
215     pstmt.setString(parameterIndex:4, x:"Sepeda lipat ringan dan efisien");
216     pstmt.setString(parameterIndex:5, x:"lipat4.png");
217     pstmt.setInt(parameterIndex:6, x:20);
218     pstmt.executeUpdate();
219
220     pstmt.setString(parameterIndex:1, x:"Lipat TravelMate");
221     pstmt.setString(parameterIndex:2, x:"Lipat");
222     pstmt.setDouble(parameterIndex:3, x:220000);
223     pstmt.setString(parameterIndex:4, x:"Sepeda lipat ideal untuk bepergian");
224     pstmt.setString(parameterIndex:5, x:"lipat5.png");
225     pstmt.setInt(parameterIndex:6, x:9);
226     pstmt.executeUpdate();
227
228     pstmt.setString(parameterIndex:1, x:"Lipat SwiftFold");
229     pstmt.setString(parameterIndex:2, x:"Lipat");
230     pstmt.setDouble(parameterIndex:3, x:190000);
231     pstmt.setString(parameterIndex:4, x:"Sepeda lipat cepat dan nyaman");
232     pstmt.setString(parameterIndex:5, x:"lipat6.png");
233     pstmt.setInt(parameterIndex:6, x:11);
234     pstmt.executeUpdate();
235     System.out.println(x:"Sample products added.");
236 }
237 }
238 } catch (SQLException e) {
239     System.out.println("Error adding sample products: " + e.getMessage());
240     e.printStackTrace();
241 }
242
243 }
```

```

244     public static Admin authenticateAdmin(String username, String password) {
245         String sql = "SELECT user_id, username, password, address, phone_number, role FROM users WHERE username = ? AND password = ? AND role = 'admin'";
246         System.out.println("DEBUG: Mencoba otentikasi admin dengan username: " + username + " dan password: " + password);
247         try (Connection conn = connect();
248              PreparedStatement pstmt = conn.prepareStatement(sql)) {
249             pstmt.setString(parameterIndex:1, username);
250             pstmt.setString(parameterIndex:2, password);
251             ResultSet rs = pstmt.executeQuery();
252             if (rs.next()) {
253                 System.out.println(":DEBUG: Data admin ditemukan di database!");
254                 System.out.println("DEBUG: Role yang ditemukan: " + rs.getString(columnLabel:"role"));
255                 return new Admin(
256                     rs.getInt(columnLabel:"user_id"),
257                     rs.getString(columnLabel:"username"),
258                     rs.getString(columnLabel:"password"),
259                     rs.getString(columnLabel:"address"),
260                     rs.getString(columnLabel:"phone_number")
261                 );
262             } else {
263                 System.out.println("DEBUG: Tidak ada data admin ditemukan untuk username: " + username + " dengan role 'admin'.");
264             }
265         } catch (SQLException e) {
266             System.out.println("Authentication error for admin: " + e.getMessage());
267             e.printStackTrace();
268         }
269         return null;
270     }
271
272     public static void addDefaultAdmin() {
273         String sql = "INSERT INTO users(username, password, address, phone_number, role) VALUES(?, ?, ?, ?, ?)";
274         String checkAdminSql = "SELECT COUNT(*) FROM users WHERE username = 'admin' AND role = 'admin'";
275         System.out.println(x:"DEBUG: Memeriksa apakah admin default sudah ada...");
276         try (Connection conn = connect();
277              Statement stmt = conn.createStatement();
278              ResultSet rs = stmt.executeQuery(checkAdminSql)) {
279             if (rs.next() && rs.getInt(columnIndex:1) == 0) {
280                 System.out.println(x:"DEBUG: Admin default belum ada, mencoba membuat...");
281                 try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
282                     pstmt.setString(parameterIndex:1, x:"admin");
283                     pstmt.setString(parameterIndex:2, x:"admin123");
284                     pstmt.setString(parameterIndex:3, x:"Admin Address");
285                     pstmt.setString(parameterIndex:4, x:"081122334455");
286                     pstmt.setString(parameterIndex:5, x:"admin");
287                     int rowsAffected = pstmt.executeUpdate();
288                     if (rowsAffected > 0) {
289                         System.out.println("Default admin user 'admin' created with password 'admin123'. Rows affected: " + rowsAffected);
290                     } else {
291                         System.out.println("WARNING: Admin user 'admin' not created. Rows affected: " + rowsAffected);
292                     }
293                 } catch (SQLException e) {
294                     System.out.println("Error adding default admin: " + e.getMessage());
295                     e.printStackTrace();
296                 }
297             } else {
298                 System.out.println(x:"DEBUG: Admin default sudah ada.");
299             }
300         }
301     }
302
303     public static User authenticateUser(String username, String password) {
304         String sql = "SELECT user_id, username, password, address, phone_number FROM users WHERE username = ? AND password = ?";
305         try (Connection conn = connect();
306              PreparedStatement pstmt = conn.prepareStatement(sql)) {
307             pstmt.setString(parameterIndex:1, username);
308             pstmt.setString(parameterIndex:2, password);
309             ResultSet rs = pstmt.executeQuery();
310             if (rs.next()) {
311                 return new User(rs.getInt(columnLabel:"user_id"), rs.getString(columnLabel:"username"), rs.getString(columnLabel:"password"),
312                               rs.getString(columnLabel:"address"), rs.getString("phone_number"));
313             }
314         } catch (SQLException e) {
315             System.out.println("Authentication error: " + e.getMessage());
316         }
317         return null;
318     }

```

```

319     public static boolean registerUser(String username, String password, String address, String phoneNumber) {
320         String sql = "INSERT INTO users(username, password, address, phone_number) VALUES(?, ?, ?, ?)";
321         try (Connection conn = connect();
322              | PreparedStatement pstmt = conn.prepareStatement(sql)) {
323             | pstmt.setString(parameterIndex:1, username);
324             | pstmt.setString(parameterIndex:2, password);
325             | pstmt.setString(parameterIndex:3, address);
326             | pstmt.setString(parameterIndex:4, phoneNumber);
327             | pstmt.executeUpdate();
328             | return true;
329         } catch (SQLException e) {
330             System.out.println("Registration error: " + e.getMessage());
331             return false;
332         }
333     }
334
335     public static List<Produk> getProductsByCategory(String category) {
336         List<Produk> products = new ArrayList<>();
337         String sql = "SELECT product_id, name, category, price, description, image_path, stock FROM products WHERE category = ?";
338         try (Connection conn = connect();
339              | PreparedStatement pstmt = conn.prepareStatement(sql)) {
340             | pstmt.setString(parameterIndex:1, category);
341             | ResultSet rs = pstmt.executeQuery();
342             while (rs.next()) {
343                 products.add(new Produk(
344                     rs.getInt(columnLabel:"product_id"),
345                     rs.getString(columnLabel:"name"),
346                     rs.getString(columnLabel:"category"),
347                     rs.getDouble(columnLabel:"price"),
348                     rs.getString(columnLabel:"description"),
349                     rs.getString(columnLabel:"image_path"),
350                     rs.getInt(columnLabel:"stock")
351                 ));
352             }
353         } catch (SQLException e) {
354             System.out.println("Error fetching products: " + e.getMessage());
355         }
356         return products;
357     }
358
359     public static boolean createOrder(int userId, int productId, String shippingAddress, String courier, String paymentMethod, double totalPrice, String vaNumber) {
360         String sql = "INSERT INTO orders(user_id, product_id, shipping_address, courier, payment_method, total_price, va_number, status) VALUES(?, ?, ?, ?, ?, ?, ?, ?)";
361         try (Connection conn = connect();
362              | PreparedStatement pstmt = conn.prepareStatement(sql)) {
363             | pstmt.setInt(parameterIndex:1, userId);
364             | pstmt.setInt(parameterIndex:2, productId);
365             | pstmt.setString(parameterIndex:3, shippingAddress);
366             | pstmt.setString(parameterIndex:4, courier);
367             | pstmt.setString(parameterIndex:5, paymentMethod);
368             | pstmt.setDouble(parameterIndex:6, totalPrice);
369             if (vaNumber != null) {
370                 pstmt.setString(parameterIndex:7, vaNumber);
371             } else {
372                 | PreparedStatement pstmt - DatabaseHelper.createOrder(int, int, String, String, String, double, String)
373                 | pstmt.setNull(parameterIndex:7, java.sql.Types.VARCHAR);
374             }
375             pstmt.executeUpdate();
376             return true;
377         } catch (SQLException e) {
378             System.out.println("Error creating order: " + e.getMessage());
379             return false;
380         }
381     }
382
383     public static List<Pesanan> getAllOrders() {
384         List<Pesanan> orders = new ArrayList<>();
385         String sql = "SELECT o.order_id, o.user_id, o.product_id, o.order_date, " +
386             "o.shipping_address, o.courier, o.payment_method, o.total_price, o.va_number, o.status, " +
387             "u.username AS user_username, u.address AS user_address, u.phone_number AS user_phone_number, " +
388             "p.name AS product_name, p.category AS product_category, p.price AS product_price, p.description AS
389             product_description, p.image_path AS product_image_path, p.stock AS product_stock " +
390             "FROM orders o " +
391             "JOIN users u ON o.user_id = u.user_id " +
392             "JOIN products p ON o.product_id = p.product_id";

```

```

393     try (Connection conn = connect();
394          PreparedStatement pstmt = conn.prepareStatement(sql); // Gunakan PreparedStatement meski tanpa parameter agar lebih aman
395          ResultSet rs = pstmt.executeQuery()) {
396
397     while (rs.next()) {
398         // Buat objek User
399         User user = new User(
400             rs.getInt(columnLabel:"user_id"),
401             rs.getString(columnLabel:"user_username"),
402             password:"",
403             rs.getString(columnLabel:"user_address"),
404             rs.getString(columnLabel:"user_phone_number")
405         );
406
407         // Buat objek Produk
408         Produk product = new Produk(
409             rs.getInt(columnLabel:"product_id"),
410             rs.getString(columnLabel:"product_name"),
411             rs.getString(columnLabel:"product_category"),
412             rs.getDouble(columnLabel:"product_price"),
413             rs.getString(columnLabel:"product_description"),
414             rs.getString(columnLabel:"product_image_path"),
415             rs.getInt(columnLabel:"product_stock")
416         );
417         // Buat objek Pesanan
418         Pesanan order = new Pesanan(
419             rs.getInt(columnLabel:"order_id"),
420             user,
421             product,
422             rs.getString(columnLabel:"order_date"),
423             rs.getString(columnLabel:"shipping_address"),
424             rs.getString(columnLabel:"courier"),
425             rs.getString(columnLabel:"payment_method"),
426             rs.getDouble(columnLabel:"total_price"),
427             rs.getString(columnLabel:"va_number"),
428             rs.getString(columnLabel:"status")
429         );
430         orders.add(order);
431     }
432 } catch (SQLException e) {
433     System.out.println("Error fetching all orders: " + e.getMessage());
434     e.printStackTrace();
435 }
436 return orders;
437 }
438
439
440 public static List<Pesanan> getOrdersByUserId(int userId) {
441     List<Pesanan> orders = new ArrayList<>();
442     String sql = "SELECT o.order_id, o.user_id, o.product_id, o.order_date, " +
443                 "o.shipping_address, o.courier, o.payment_method, o.total_price, o.va_number, o.status, " +
444                 "u.username AS user_username, u.address AS user_address, u.phone_number AS user_phone_number, " +
445                 "p.name AS product_name, p.category AS product_category, p.price AS product_price, p.description AS product_description,
446                 p.image_path AS product_image_path, p.stock AS product_stock " +
447                 "FROM orders o " +
448                 "JOIN users u ON o.user_id = u.user_id " +
449                 "JOIN products p ON o.product_id = p.product_id " +
450                 "WHERE o.user_id = ?";
451
452     try (Connection conn = connect();
453          PreparedStatement pstmt = conn.prepareStatement(sql)) {
454         pstmt.setInt(parameterIndex:1, userId);
455         try (ResultSet rs = pstmt.executeQuery()) {
456             while (rs.next()) {
457                 User user = new User(
458                     rs.getInt(columnLabel:"user_id"),
459                     rs.getString(columnLabel:"user_username"),
460                     password:"",
461                     rs.getString(columnLabel:"user_address"),
462                     rs.getString(columnLabel:"user_phone_number")
463                 );
464
465                 Produk product = new Produk(
466                     rs.getInt(columnLabel:"product_id"),
467                     rs.getString(columnLabel:"product_name"),
468                     rs.getString(columnLabel:"product_category"),
469                     rs.getDouble(columnLabel:"product_price"),
470                     rs.getString(columnLabel:"product_description"),
471                     rs.getString(columnLabel:"product_image_path"),
472                     rs.getInt(columnLabel:"product_stock")
473                 );
474             }
475         }
476     }
477 }

```

```

474     Pesanan order = new Pesanan(
475         rs.getInt(columnLabel:"order_id"),
476         user,
477         product,
478         rs.getString(columnLabel:"order_date"),
479         rs.getString(columnLabel:"shipping_address"),
480         rs.getString(columnLabel:"courier"),
481         rs.getString(columnLabel:"payment_method"),
482         rs.getDouble(columnLabel:"total_price"),
483         rs.getString(columnLabel:"va_number"),
484         rs.getString(columnLabel:"status")
485     );
486     orders.add(order);
487 }
488 }
489 catch (SQLException e) {
490     System.out.println("Error fetching orders by user ID: " + e.getMessage());
491     e.printStackTrace();
492 }
493 return orders;
494 }
495
496 public static boolean updateOrderStatus(int orderId, String newStatus) {
497     String sql = "UPDATE orders SET status = ? WHERE order_id = ?";
498     try (Connection conn = connect();
499          PreparedStatement pstmt = conn.prepareStatement(sql)) {
500         pstmt.setString(parameterIndex:1, newStatus);
501         pstmt.setInt(parameterIndex:2, orderId);
502         int rowsAffected = pstmt.executeUpdate();
503         return rowsAffected > 0;
504     } catch (SQLException e) {
505         System.out.println("Error updating order status: " + e.getMessage());
506         e.printStackTrace();
507         return false;
508     }
509 }
510
511 public static boolean deleteOrder(int orderId) {
512     String sql = "DELETE FROM orders WHERE order_id = ?";
513     try (Connection conn = connect();
514          PreparedStatement pstmt = conn.prepareStatement(sql)) {
515         pstmt.setInt(parameterIndex:1, orderId);
516         int rowsAffected = pstmt.executeUpdate();
517         return rowsAffected > 0;
518     } catch (SQLException e) {
519         System.out.println("Error deleting order: " + e.getMessage());
520         e.printStackTrace();
521         return false;
522     }
523 }
524 }

```

## Penjelasan Kode Program

Kelas DatabaseHelper.java adalah kelas utilitas yang sangat krusial, karena berfungsi sebagai satu-satunya jembatan antara aplikasi Java dengan database SQLite. Kelas ini menangani semua operasi terkait data, mulai dari koneksi, pembuatan tabel, hingga manipulasi data (Create, Read, Update, Delete - CRUD).

- Mendeklarasikan public class DatabaseHelper. Kelas ini tidak memiliki konstruktur publik dan semua metodenya bersifat static, menandakan bahwa ini adalah kelas utilitas yang tidak perlu dibuat objeknya.
- Terdapat atribut private static final String URL. Ini adalah konstanta yang menyimpan *connection string* ke file database SQLite (cyclepro.db).
- Terdapat metode public static Connection connect(). Metode ini bertanggung jawab untuk membuat dan mengembalikan sebuah objek Connection ke database.

- Terdapat metode public static void createNewDatabase() dan public static void createTables(). Metode-metode ini dipanggil saat aplikasi pertama kali dijalankan untuk memastikan file database dan semua tabel (users, products, orders) telah dibuat dengan struktur yang benar, termasuk kolom, tipe data, dan relasi *foreign key*.
- Terdapat metode public static void addSampleProducts() dan public static void addDefaultAdmin(). Ini adalah metode *seeding* yang berfungsi untuk mengisi data awal ke dalam database, seperti contoh-contoh produk dan akun admin default, agar aplikasi tidak kosong saat pertama kali dijalankan.
- Terdapat metode otentikasi: authenticateUser() dan authenticateAdmin(). Metode ini menerima username dan password sebagai input, lalu menjalankan query SQL untuk memeriksa apakah kredensial tersebut cocok dengan data yang ada di tabel users.
- Terdapat metode public static boolean registerUser(...). Metode ini menangani proses pendaftaran pengguna baru dengan menyimpan datanya ke dalam tabel users.
- Terdapat serangkaian metode untuk mengambil data (Read):
  - getProductsByCategory(): Mengambil daftar produk berdasarkan kategori tertentu.
  - getAllOrders(): Mengambil semua data pesanan yang ada (biasanya untuk admin).
  - getOrdersById(): Mengambil riwayat pesanan untuk satu pengguna spesifik.
- Terdapat serangkaian metode untuk memanipulasi data (Create, Update, Delete):
  - createOrder(): Menyimpan data pesanan baru ke dalam tabel orders.
  - updateOrderStatus(): Mengubah status sebuah pesanan (misalnya dari "Diproses" menjadi "Dikirim").
  - deleteOrder(): Menghapus data pesanan dari database.
- Penggunaan PreparedStatement di sebagian besar metode (misalnya authenticateUser, registerUser, createOrder) adalah praktik keamanan yang penting untuk mencegah serangan *SQL Injection*.
- Penggunaan blok try-with-resources (misalnya try (Connection conn = connect())) memastikan bahwa koneksi database dan sumber daya lainnya seperti Statement dan ResultSet selalu ditutup secara otomatis, bahkan jika terjadi *error*, untuk mencegah kebocoran sumber daya.

## **Penjelasan Konsep OOP**

- Menggunakan konsep Enkapsulasi
  - Kelas ini secara sempurna menyembunyikan semua detail kompleks dari interaksi database (seperti SQL query, driver JDBC, manajemen koneksi) dari bagian lain aplikasi.
  - Kelas lain seperti HalamanLogin atau Dashboard tidak perlu tahu bagaimana cara kerja database; mereka hanya perlu memanggil metode yang sudah disediakan, seperti DatabaseHelper.authenticateUser(...).
- Menggunakan konsep Abstraksi
  - Kelas ini menyediakan antarmuka (kumpulan metode publik) yang sederhana dan abstrak untuk melakukan operasi database. Misalnya, metode registerUser() adalah sebuah abstraksi dari serangkaian perintah SQL INSERT. Programmer dapat menggunakan metode ini tanpa perlu memahami bahasa SQL secara mendalam.
- Menggunakan Anggota Statis
  - Semua metode dalam kelas ini dideklarasikan sebagai static. Ini menjadikannya kelas utilitas murni yang berfungsi sebagai titik akses tunggal dan global untuk semua kebutuhan database, tanpa perlu membuat *instance* (new DatabaseHelper()).
- Menerapkan Prinsip Tanggung Jawab Tunggal
  - Kelas ini memiliki satu tanggung jawab yang jelas dan terfokus: mengelola semua komunikasi dengan database. Ini membuat arsitektur aplikasi menjadi lebih bersih, terorganisir, dan mudah untuk dipelihara atau dimodifikasi di kemudian hari.

### 3.1.4 Alur Pengguna (View)

## 8. HalamanLogin.java

```
1 import java.awt.*;
2 import java.awt.event.FocusAdapter;
3 import java.awt.event.FocusEvent;
4 import java.awt.event.MouseAdapter;
5 import java.awt.event.MouseEvent;
6 import java.awt.geom.RoundRectangle2D;
7 import javax.swing.*;
8 import javax.swing.border.EmptyBorder;
9 import javax.swing.text.JTextComponent;
10
11 public class HalamanLogin extends JFrame {
12     private RoundedTextField usernameField;
13     private RoundedJPasswordField passwordField;
14
15     public HalamanLogin() {
16         setTitle(title:"Login - CyclePro");
17         setSize(width:900, height:550);
18         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
19         setLocationRelativeTo(c:null);
20         setResizable(resizable:false);
21
22         JLayeredPane layeredPane = new JLayeredPane();
23         layeredPane.setPreferredSize(new Dimension(width:900, height:550));
24         add(layeredPane);
25
26         JLabel backgroundLabel = new JLabel();
27         try {
28             java.net.URL imgUrl = getClass().getResource(name:"/img/DesainBG.png");
29             if (imgUrl != null) {
30                 ImageIcon originalIcon = new ImageIcon(imgUrl);
31                 Image originalImage = originalIcon.getImage();
32                 Image scaledImage = originalImage.getScaledInstance(width:900, height:550, Image.SCALE_SMOOTH);
33                 backgroundLabel.setIcon(new ImageIcon(scaledImage));
34             } else {
35                 System.err.println(x:"Background image not found: /img/DesainBG.png");
36                 JPanel fallbackPanel = new JPanel();
37                 fallbackPanel.setBackground(Colors.BACKGROUND_PRIMARY);
38                 fallbackPanel.setBounds(x:0, y:0, width:900, height:550);
39                 layeredPane.add(fallbackPanel, JLayeredPane.DEFAULT_LAYER);
40             }
41         } catch (Exception e) {
42             System.err.println("Error loading background image: " + e.getMessage());
43             e.printStackTrace();
44             JPanel fallbackPanel = new JPanel();
45             fallbackPanel.setBackground(Colors.BACKGROUND_PRIMARY);
46             fallbackPanel.setBounds(x:0, y:0, width:900, height:550);
47             layeredPane.add(fallbackPanel, JLayeredPane.DEFAULT_LAYER);
48         }
49         backgroundLabel.setBounds(x:0, y:0, width:900, height:550);
50         layeredPane.add(backgroundLabel, JLayeredPane.DEFAULT_LAYER);
51
52         JPanel loginPanel = new JPanel();
53         loginPanel.setLayout(mgr:null);
54         loginPanel.setBackground(Color.WHITE);
55         loginPanel.setBorder(BorderFactory.createEmptyBorder());
56
57         int panelWidth = 400;
58         int panelHeight = 400;
59         int panelX = 900 - panelWidth - 40;
60         int panelY = (550 - panelHeight) / 2;
61         loginPanel.setBounds(panelX, panelY, panelWidth, panelHeight);
62         layeredPane.add(loginPanel, JLayeredPane.PALETTE_LAYER);
63
64         int xMargin = 50;
65         int currentY = 50;
66         int fieldHeight = 35;
67         int spacing = 20;
68
69         JLabel titleLabel = new JLabel(text:"USER LOGIN");
70         titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:22));
71         titleLabel.setForeground(Colors.TEXT_PRIMARY);
72         titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
73         titleLabel.setBounds(x:0, currentY, panelWidth, height:30);
74         loginPanel.add(titleLabel);
75         currentY += 50;
76     }
}
```

```

77     usernameField = new RoundedTextField(text:"Username", size:20);
78     usernameField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
79     usernameField.setBackground(Color.WHITE);
80     usernameField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
81     loginPanel.add(usernameField);
82     addPlaceholderAndFocusListeners(usernameField, placeholder:"Username", isPasswordField:false);
83     currentY += fieldHeight + spacing;
84
85     passwordField = new RoundedJPasswordField(text:"Masukkan Password", size:20);
86     passwordField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
87     passwordField.setBackground(Color.WHITE);
88     passwordField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
89     loginPanel.add(passwordField);
90     addPlaceholderAndFocusListeners(passwordField, placeholder:"Masukkan Password", isPasswordField:true);
91     currentY += fieldHeight + 20;
92
93     JButton loginButton = new JButton(text:"Login");
94     loginButton.setBackground(new Color(rgb:0xD4AF37));
95     loginButton.setForeground(Color.WHITE);
96     loginButton.setFont(new Font(name:"Arial", Font.BOLD, size:16));
97     loginButton.setFocusPainted(b:false);
98     loginButton.setBorder(new EmptyBorder(top:10, left:25, bottom:10, right:25));
99     loginButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
100    loginButton.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), height:40);
101    loginPanel.add(loginButton);
102    currentY += 40 + 20;
103
104    JLabel createAccountLabel = new JLabel(text:"Buat Akun");
105    createAccountLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:12));
106    createAccountLabel.setForeground(Colors.TEXT_LINK);
107    createAccountLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));
108    createAccountLabel.setHorizontalTextPosition(SwingConstants.CENTER);
109    createAccountLabel.setBounds(x:0, currentY, panelWidth, height:20);
110    loginPanel.add(createAccountLabel);
111    createAccountLabel.addMouseListener(new MouseAdapter() {
112        @Override
113        public void mouseClicked(MouseEvent e) {
114            new Registrasi().setVisible(b:true);
115            dispose();
116        }
117    });
118
119    loginButton.addActionListener(e -> [
120        String username = usernameField.getText();
121        String password = new String(passwordField.getPassword());
122
123        if (username.isEmpty() || username.equals(anObject:"Username") || password.isEmpty() || password.equals(anObject:"Password")) {
124            JOptionPane.showMessageDialog(this, message:"Username dan password tidak boleh kosong!", title:"Error", ERROR_MESSAGE);
125            return;
126        }
127
128        Admin adminUser = DatabaseHelper.authenticateAdmin(username, password);
129        if (adminUser != null) {
130            Main.currentUser = adminUser;
131            JOptionPane.showMessageDialog(this, message:"Login Admin Berhasil!");
132            new AdminDashboard().setVisible(b:true);
133            dispose();
134            return;
135        }
136
137        User regularUser = DatabaseHelper.authenticateUser(username, password);
138        if (regularUser != null) {
139            Main.currentUser = regularUser;
140            JOptionPane.showMessageDialog(this, message:"Login Pengguna Berhasil!");
141            new Dashboard().setVisible(b:true);
142            dispose();
143        } else {
144            JOptionPane.showMessageDialog(this, message:"Username atau password salah.", title:"Login Gagal", JOptionPane.ERROR_MESSAGE);
145        }
146    });
147
148

```

```

149 private void addPlaceholderAndFocusListeners(JTextComponent field, String placeholder, boolean isPasswordField) {
150     if (isPasswordField) {
151         JPasswordField pf = (JPasswordField) field;
152         pf.setText(placeholder);
153         pf.setEchoChar((char) 0);
154         pf.setForeground(Colors.TEXT_SECONDARY);
155     } else {
156         JTextField tf = (JTextField) field;
157         tf.setText(placeholder);
158         tf.setForeground(Colors.TEXT_SECONDARY);
159     }
160
161     field.addFocusListener(new FocusAdapter() {
162         @Override
163         public void focusGained(FocusEvent e) {
164             if (isPasswordField) {
165                 JPasswordField pf = (JPasswordField) field;
166                 if (new String(pf.getPassword()).equals(placeholder)) {
167                     pf.setText(t:"");
168                     pf.setEchoChar(c:'*');
169                     pf.setForeground(Colors.TEXT_PRIMARY);
170                 }
171             } else {
172                 JTextField tf = (JTextField) field;
173                 if (tf.getText().equals(placeholder)) {
174                     tf.setText(t:"");
175                     tf.setForeground(Colors.TEXT_PRIMARY);
176                 }
177             }
178             if (field instanceof RoundedTextField) {
179                 ((RoundedTextField) field).setBorderColor(Colors.BUTTON_PRIMARY_BACKGROUND);
180             } else if (field instanceof RoundedJPasswordField) {
181                 ((RoundedJPasswordField) field).setBorderColor(Colors.BUTTON_PRIMARY_BACKGROUND);
182             }
183             field.repaint();
184         }
185
186         @Override
187         public void focusLost(FocusEvent e) {
188             if (isPasswordField) {
189                 JPasswordField pf = (JPasswordField) field;
190                 if (new String(pf.getPassword()).isEmpty()) {
191                     pf.setEchoChar((char) 0);
192                     pf.setText(placeholder);
193                     pf.setForeground(Colors.TEXT_SECONDARY);
194                 }
195             } else {
196                 JTextField tf = (JTextField) field;
197                 if (tf.getText().isEmpty()) {
198                     tf.setText(placeholder);
199                     tf.setForeground(Colors.TEXT_SECONDARY);
200                 }
201             }
202             if (field instanceof RoundedTextField) {
203                 ((RoundedTextField) field).setBorderColor(Colors.BORDER_COLOR);
204             } else if (field instanceof RoundedJPasswordField) {
205                 ((RoundedJPasswordField) field).setBorderColor(Colors.BORDER_COLOR);
206             }
207             field.repaint();
208         }
209     });
210 }
211
212 static class RoundedTextField extends JTextField {
213     private Shape shape;
214     private int arcWidth = 20;
215     private int arcHeight = 20;
216     private Color borderColor = Colors.BORDER_COLOR;
217
218     public RoundedTextField(int size) {
219         super(size);
220         setOpaque(isOpaque:false);
221         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
222     }
223
224     public RoundedTextField(String text, int size) {
225         super(text, size);
226         setOpaque(isOpaque:false);
227         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
228     }
229 }

```

```

230     public void setBorderColor(Color color) {
231         this.borderColor = color;
232         repaint();
233     }
234
235     @Override
236     protected void paintComponent(Graphics g) {
237         Graphics2D g2 = (Graphics2D) g.create();
238         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
239         g2.setBackground(getBackground());
240         g2.fill(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
241         super.paintComponent(g2);
242         g2.dispose();
243     }
244
245     @Override
246     protected void paintBorder(Graphics g) {
247         Graphics2D g2 = (Graphics2D) g.create();
248         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
249         g2.setColor(borderColor);
250         g2.draw(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
251         g2.dispose();
252     }
253
254     @Override
255     public boolean contains(int x, int y) {
256         if (shape == null || !shape.getBounds().equals(getBounds())) {
257             shape = new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight);
258         }
259         return shape.contains(x, y);
260     }
261 }
262
263 static class RoundedJPasswordField extends JPasswordField {
264     private Shape shape;
265     private int arcWidth = 20;
266     private int arcHeight = 20;
267     private Color borderColor = Colors.BORDER_COLOR;
268
269     public RoundedJPasswordField(int size) {
270         super(size);
271         setOpaque(isOpaque:false);
272         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
273     }
274
275     public RoundedJPasswordField(String text, int size) {
276         super(text, size);
277         setOpaque(isOpaque:false);
278         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
279     }
280
281     public void setBorderColor(Color color) {
282         this.borderColor = color;
283         repaint();
284     }
285
286     @Override
287     protected void paintComponent(Graphics g) {
288         Graphics2D g2 = (Graphics2D) g.create();
289         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
290         g2.setColor(getBackground());
291         g2.fill(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
292         super.paintComponent(g2);
293         g2.dispose();
294     }
295
296     @Override
297     protected void paintBorder(Graphics g) {
298         Graphics2D g2 = (Graphics2D) g.create();
299         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
300         g2.setColor(borderColor);
301         g2.draw(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
302         g2.dispose();
303     }
304
305     @Override
306     public boolean contains(int x, int y) {
307         if (shape == null || !shape.getBounds().equals(getBounds())) {
308             shape = new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight);
309         }
310         return shape.contains(x, y);
311     }
312 }

```

## Penjelasan Kode Program

Kelas HalamanLogin.java adalah salah satu komponen antarmuka pengguna (GUI) utama yang berfungsi sebagai gerbang masuk bagi pengguna dan admin ke dalam aplikasi CyclePro.

- Mendeklarasikan public class HalamanLogin extends JFrame, yang berarti kelas ini adalah sebuah jendela utama aplikasi yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat atribut private untuk usernameField dan passwordField yang menggunakan komponen kustom RoundedTextField dan RoundedJPasswordField untuk tampilan input yang lebih modern.
- Menggunakan JLayeredPane untuk menumpuk komponen. Ini memungkinkan penempatan gambar latar belakang (backgroundLabel) di lapisan bawah dan panel login (loginPanel) di atasnya.
- Terdapat penanganan *error* saat memuat gambar latar. Jika gambar tidak ditemukan atau terjadi kesalahan, sebuah panel dengan warna solid akan ditampilkan sebagai *fallback*.
- Panel login (loginPanel) menggunakan setLayout(null) untuk penempatan komponen secara absolut dengan koordinat x dan y, memberikan kontrol penuh atas tata letak.
- Terdapat ActionListener yang ditambahkan pada loginButton. Ketika tombol diklik, logika berikut dieksekusi:
  1. Mengambil teks dari usernameField dan passwordField.
  2. Melakukan validasi untuk memastikan input tidak kosong atau masih berupa teks *placeholder*.
  3. Mencoba melakukan otentikasi sebagai admin dengan memanggil DatabaseHelper.authenticateAdmin().
  4. Jika gagal, mencoba melakukan otentikasi sebagai pengguna biasa dengan memanggil DatabaseHelper.authenticateUser().
  5. Berdasarkan hasil otentikasi, pengguna akan diarahkan ke AdminDashboard atau Dashboard, atau akan menerima pesan kesalahan jika login gagal.
- Terdapat MouseListener pada label "Buat Akun" (createAccountLabel) yang akan membuka jendela Registrasi saat diklik.
- Terdapat metode privat addPlaceholderAndFocusListeners(...). Metode *helper* ini digunakan untuk menambahkan fungsionalitas *placeholder* pada kolom input. Ia akan

menampilkan teks petunjuk saat kolom kosong dan mengubah warna *border* saat kolom mendapatkan fokus, sehingga meningkatkan pengalaman pengguna.

- Terdapat dua kelas dalam (*inner static class*), yaitu RoundedTextField dan RoundedJPasswordField. Kedua kelas kustom ini meng-*override* metode paintComponent() dan paintBorder() untuk menggambar bentuk persegi panjang dengan sudut membulat, memberikan tampilan yang unik pada kolom input.

### Penjelasan Konsep OOP

- Menggunakan konsep Enkapsulasi
  - Atribut usernameField dan passwordField dideklarasikan sebagai private, sehingga aksesnya terbatas hanya di dalam kelas HalamanLogin.
  - Kelas dalam RoundedTextField dan RoundedJPasswordField menyembunyikan detail kompleks tentang cara menggambar sudut membulat dan mengelola warna *border*. Kelas luar hanya perlu menggunakan komponen ini tanpa perlu tahu implementasi internalnya.
- Menggunakan konsep Pewarisan (*Inheritance*)
  - HalamanLogin adalah turunan dari JFrame.
  - RoundedTextField adalah turunan dari JTextField.
  - RoundedJPasswordField adalah turunan dari JPasswordField.
  - Pewarisan ini memungkinkan kelas-kelas tersebut untuk menggunakan kembali semua fungsionalitas dari kelas induknya sambil menambahkan atau memodifikasi perilaku tertentu.
- Menggunakan konsep Polimorfisme
  - Penggunaan @Override pada metode paintComponent dan paintBorder di kelas-kelas dalam adalah contoh dari *method overriding*.
  - Penggunaan MouseAdapter dan FocusAdapter juga merupakan bentuk polimorfisme, di mana kita hanya perlu meng-*override* metode *event* yang kita butuhkan saja (misalnya, mouseClicked atau focusGained).
- Menggunakan konsep Abstraksi
  - Kelas HalamanLogin menggunakan DatabaseHelper untuk melakukan otentifikasi tanpa perlu mengetahui detail query SQL atau cara koneksi ke database. DatabaseHelper menyediakan abstraksi dari logika akses data.

## 9. Registrasi.java

```
1 import java.awt.*;
2 import java.awt.event.FocusAdapter;
3 import java.awt.event.FocusEvent;
4 import java.awt.event.MouseAdapter;
5 import java.awt.event.MouseEvent;
6 import java.awt.geom.RoundRectangle2D;
7 import javax.swing.*;
8 import javax.swing.border.EmptyBorder;
9 import javax.swing.text.JTextComponent;
10
11 public class Registrasi extends JFrame {
12     private RoundedJTextField usernameField;
13     private RoundedJPasswordField passwordField;
14     private RoundedJTextField addressField;
15     private RoundedJTextField phoneField;
16
17     public Registrasi() {
18         setTitle("Buat Akun - CyclePro");
19         setSize(width:900, height:550);
20         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
21         setLocationRelativeTo(null);
22         setResizable(resizable:false);
23
24         JLayeredPane layeredPane = new JLayeredPane();
25         layeredPane.setPreferredSize(new Dimension(width:900, height:550));
26         add(layeredPane);
27
28         JLabel backgroundLabel = new JLabel();
29         try {
30             java.net.URL imgUrl = getClass().getResource(name:"/img/DesainBG.png");
31             if (imgUrl != null) {
32                 ImageIcon originalIcon = new ImageIcon(imgUrl);
33                 Image originalImage = originalIcon.getImage();
34                 Image scaledImage = originalImage.getScaledInstance(width:900, height:550, Image.SCALE_SMOOTH);
35                 backgroundLabel.setIcon(new ImageIcon(scaledImage));
36             } else {
37                 System.err.println("Background image not found: /img/DesainBG.png");
38                 JPanel fallbackPanel = new JPanel();
39                 fallbackPanel.setBackground(Colors.BACKGROUND_PRIMARY);
40                 fallbackPanel.setBounds(x:0, y:0, width:900, height:550);
41                 layeredPane.add(fallbackPanel, JLayeredPane.DEFAULT_LAYER);
42             }
43         } catch (Exception e) {
44             System.err.println("Error loading background image: " + e.getMessage());
45             e.printStackTrace();
46             JPanel fallbackPanel = new JPanel();
47             fallbackPanel.setBackground(Colors.BACKGROUND_PRIMARY);
48             fallbackPanel.setBounds(x:0, y:0, width:900, height:550);
49             layeredPane.add(fallbackPanel, JLayeredPane.DEFAULT_LAYER);
50         }
51         backgroundLabel.setBounds(x:0, y:0, width:900, height:550);
52         layeredPane.add(backgroundLabel, JLayeredPane.DEFAULT_LAYER);
53
54         JPanel registrasiPanel = new JPanel();
55         registrasiPanel.setLayout(mgr:null);
56         registrasiPanel.setBackground(Color.WHITE);
57         registrasiPanel.setBorder(BorderFactory.createEmptyBorder());
58
59         int panelWidth = 400;
60         int panelHeight = 550;
61         int panelX = 900 - panelWidth;
62         int panelY = 0;
63
64         registrasiPanel.setBounds(panelX, panelY, panelWidth, panelHeight);
65         layeredPane.add(registrasiPanel, JLayeredPane.PALETTE_LAYER);
66
67         int xMargin = 50;
68         int currentY = 30;
69         int fieldHeight = 35;
70         int spacing = 15;
71
72         JLabel titleLabel = new JLabel(text:"BUAT AKUN");
73         titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:22));
74         titleLabel.setForeground(Colors.TEXT_PRIMARY);
75         titleLabel.setHorizontalAlignment(SwingConstants.CENTER);
76         titleLabel.setBounds(x:0, currentY, panelWidth, height:30);
77         registrasiPanel.add(titleLabel);
78         currentY += 50;
79 }
```

```

80     usernameField = new RoundedJTextField(size:20);
81     usernameField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
82     usernameField.setBackground(Color.WHITE);
83     usernameField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
84     registrasiPanel.add(usernameField);
85     addPlaceholderAndFocusListeners(usernameField, placeholder:"Username", isPasswordField:false);
86     currentY += fieldHeight + spacing;
87
88     passwordField = new RoundedJPasswordField(size:20);
89     passwordField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
90     passwordField.setBackground(Color.WHITE);
91     passwordField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
92     registrasiPanel.add(passwordField);
93     addPlaceholderAndFocusListeners(passwordField, placeholder:"Password", isPasswordField:true);
94     currentY += fieldHeight + spacing;
95
96     addressField = new RoundedJTextField(size:20);
97     addressField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
98     addressField.setBackground(Color.WHITE);
99     addressField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
100    registrasiPanel.add(addressField);
101    addPlaceholderAndFocusListeners(addressField, placeholder:"Alamat", isPasswordField:false);
102    currentY += fieldHeight + spacing;
103
104    phoneField = new RoundedJTextField(size:20);
105    phoneField.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
106    phoneField.setBackground(Color.WHITE);
107    phoneField.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), fieldHeight);
108    registrasiPanel.add(phoneField);
109    addPlaceholderAndFocusListeners(phoneField, placeholder:"Nomor Telepon", isPasswordField:false);
110    currentY += fieldHeight + spacing + 10;
111
112    JButton registerButton = new JButton(text:"Daftar");
113    registerButton.setBackground(Colors.BUTTON_SUCCESS_BACKGROUND);
114    registerButton.setForeground(Colors.BUTTON_SUCCESS_TEXT);
115    registerButton.setFont(new Font(name:"Arial", Font.BOLD, size:16));
116    registerButton.setFocusPainted(b:false);
117    registerButton.setBorder(new EmptyBorder(top:10, left:25, bottom:10, right:25));
118    registerButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
119    registerButton.setBounds(xMargin, currentY, panelWidth - (2 * xMargin), height:40);
120    registrasiPanel.add(registerButton);
121    currentY += 40 + 20;
122
123    JLabel backToLoginLabel = new JLabel(text:"Kembali ke Login");
124    backToLoginLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:12));
125    backToLoginLabel.setForeground(Colors.TEXT_LINK);
126    backToLoginLabel.setCursor(new Cursor(Cursor.HAND_CURSOR));
127    backToLoginLabel.setHorizontalTextPosition(SwingConstants.CENTER);
128    backToLoginLabel.setBounds(x:0, currentY, panelWidth, height:20);
129    registrasiPanel.add(backToLoginLabel);
130
131    registerButton.addActionListener(e -> {
132        String username = usernameField.getText();
133        String password = new String(passwordField.getPassword());
134        String address = addressField.getText();
135        String phone = phoneField.getText();
136
137        if (username.isEmpty() || username.equals(anObject:"Username") ||
138            password.isEmpty() || password.equals(anObject:"Password") ||
139            address.isEmpty() || address.equals(anObject:"Alamat") ||
140            phone.isEmpty() || phone.equals(anObject:"Nomor Telepon"))
141        {
142            JOptionPane.showMessageDialog(this, message:"Semua field harus diisi!", title:"Error", JOptionPane.ERROR_MESSAGE);
143            return;
144        }
145
146        if (DatabaseHelper.registerUser(username, password, address, phone)) {
147            JOptionPane.showMessageDialog(this, message:"Registrasi berhasil! Silakan login.");
148            new HalamanLogin().setVisible(b:true);
149            dispose();
150        } else {
151            JOptionPane.showMessageDialog(this, message:"Registrasi gagal. Username mungkin sudah ada.", title:"Error", JOptionPane.ERROR_MESSAGE);
152        }
153    });
154

```

```

155     backToLoginLabel.addMouseListener(new MouseAdapter() {
156         @Override
157         public void mouseClicked(MouseEvent e) {
158             new HalamanLogin().setVisible(b:true);
159             dispose();
160         }
161     });
162 }
163
164 private void addPlaceholderAndFocusListeners(JTextComponent field, String placeholder, boolean isPasswordField) {
165     if (isPasswordField) {
166         JPasswordField pf = (JPasswordField) field;
167         pf.setText(placeholder);
168         pf.setEchoChar((char) 0);
169         pf.setForeground(Colors.TEXT_SECONDARY);
170     } else {
171         JTextField tf = (JTextField) field;
172         tf.setText(placeholder);
173         tf.setForeground(Colors.TEXT_SECONDARY);
174     }
175
176     field.addFocusListener(new FocusAdapter() {
177         @Override
178         public void focusGained(FocusEvent e) {
179             if (isPasswordField) {
180                 JPasswordField pf = (JPasswordField) field;
181                 if (new String(pf.getPassword()).equals(placeholder)) {
182                     pf.setText(t:"");
183                     pf.setEchoChar(c:'*');
184                     pf.setForeground(Colors.TEXT_PRIMARY);
185                 }
186             } else {
187                 JTextField tf = (JTextField) field;
188                 if (tf.getText().equals(placeholder)) {
189                     tf.setText(t:"");
190                     tf.setForeground(Colors.TEXT_PRIMARY);
191                 }
192             }
193             if (field instanceof RoundedJTextField) {
194                 ((RoundedJTextField) field).setBorderColor(Colors.BUTTON_PRIMARY_BACKGROUND);
195             } else if (field instanceof RoundedJPasswordField) {
196                 ((RoundedJPasswordField) field).setBorderColor(Colors.BUTTON_PRIMARY_BACKGROUND);
197             }
198             field.repaint();
199         }
200
201         @Override
202         public void focusLost(FocusEvent e) {
203             if (isPasswordField) {
204                 JPasswordField pf = (JPasswordField) field;
205                 if (new String(pf.getPassword()).isEmpty()) {
206                     pf.setEchoChar((char) 0);
207                     pf.setText(placeholder);
208                     pf.setForeground(Colors.TEXT_SECONDARY);
209                 }
210             } else {
211                 JTextField tf = (JTextField) field;
212                 if (tf.getText().isEmpty()) {
213                     tf.setText(placeholder);
214                     tf.setForeground(Colors.TEXT_SECONDARY);
215                 }
216             }
217             if (field instanceof RoundedJTextField) {
218                 ((RoundedJTextField) field).setBorderColor(Colors.BORDER_COLOR);
219             } else if (field instanceof RoundedJPasswordField) {
220                 ((RoundedJPasswordField) field).setBorderColor(Colors.BORDER_COLOR);
221             }
222             field.repaint();
223         }
224     });
225 }
226

```

```

-->
227 static class RoundedJTextField extends JTextField {
228     private Shape shape;
229     private int arcWidth = 20;
230     private int arcHeight = 20;
231     private Color borderColor = Colors.BORDER_COLOR;
232
233     public RoundedJTextField(int size) {
234         super(size);
235         setOpaque(isOpaque:false);
236         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
237     }
238
239     public RoundedJTextField(String text, int size) {
240         super(text, size);
241         setOpaque(isOpaque:false);
242         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
243     }
244
245     public void setBorderColor(Color color) {
246         this.borderColor = color;
247         repaint();
248     }
249
250     @Override
251     protected void paintComponent(Graphics g) {
252         Graphics2D g2 = (Graphics2D) g.create();
253         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
254
255         g2.setColor(getBackground());
256         g2.fill(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
257
258         super.paintComponent(g2);
259         g2.dispose();
260     }
261
262     @Override
263     protected void paintBorder(Graphics g) {
264         Graphics2D g2 = (Graphics2D) g.create();
265         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
266         g2.setColor(borderColor);
267         g2.draw(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
268         g2.dispose();
269     }
270
271     @Override
272     public boolean contains(int x, int y) {
273         if (shape == null || !shape.getBounds().equals(bounds)) {
274             shape = new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight);
275         }
276         return shape.contains(x, y);
277     }
278 }
279
280 static class RoundedJPasswordField extends JPasswordField {
281     private Shape shape;
282     private int arcWidth = 20;
283     private int arcHeight = 20;
284     private Color borderColor = Colors.BORDER_COLOR;
285
286     public RoundedJPasswordField(int size) {
287         super(size);
288         setOpaque(isOpaque:false);
289         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
290     }
291
292     public RoundedJPasswordField(String text, int size) {
293         super(text, size);
294         setOpaque(isOpaque:false);
295         setBorder(new EmptyBorder(top:5, left:10, bottom:5, right:10));
296     }
297
298     public void setBorderColor(Color color) {
299         this.borderColor = color;
300         repaint();
301     }
302 }

```

```

303     @Override
304     protected void paintComponent(Graphics g) {
305         Graphics2D g2 = (Graphics2D) g.create();
306         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
307
308         g2.setColor(backgroundColor);
309         g2.fill(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
310
311         super.paintComponent(g2);
312         g2.dispose();
313     }
314
315     @Override
316     protected void paintBorder(Graphics g) {
317         Graphics2D g2 = (Graphics2D) g.create();
318         g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
319         g2.setColor(borderColor);
320         g2.draw(new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight));
321         g2.dispose();
322     }
323
324     @Override
325     public boolean contains(int x, int y) {
326         if (shape == null || !shape.getBounds().equals(getBounds())) {
327             shape = new RoundRectangle2D.Double(x:0, y:0, getWidth() - 1, getHeight() - 1, arcWidth, arcHeight);
328         }
329         return shape.contains(x, y);
330     }
331 }
332 }
```

## Penjelasan Kode Program

Kelas Registrasi.java adalah antarmuka pengguna (GUI) yang menyediakan fungsionalitas bagi pengguna baru untuk membuat akun di dalam aplikasi CyclePro.

- Mendeklarasikan public class Registrasi extends JFrame, yang menandakan bahwa kelas ini adalah sebuah jendela utama yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat beberapa atribut private untuk menampung komponen input, yaitu usernameField, passwordField, addressField, dan phoneField, yang semuanya menggunakan komponen kustom untuk tampilan yang lebih modern.
- Sama seperti halaman login, kelas ini menggunakan JLayeredPane untuk menumpuk panel registrasi di atas sebuah gambar latar belakang, menciptakan tampilan yang konsisten secara visual.
- Panel registrasi (registrasiPanel) menggunakan setLayout(null) untuk memungkinkan penempatan komponen-komponen (seperti judul, kolom input, dan tombol) secara presisi menggunakan koordinat.
- Terdapat ActionListener yang ditambahkan pada registerButton. Ketika tombol "Daftar" diklik, logika berikut akan dijalankan:
  1. Mengambil semua data yang diinput oleh pengguna dari setiap kolom.
  2. Melakukan validasi sederhana untuk memastikan tidak ada kolom yang kosong.
  3. Memanggil metode DatabaseHelper.registerUser() untuk menyimpan data pengguna baru ke dalam database.

- 4. Jika registrasi berhasil, sebuah pesan konfirmasi akan ditampilkan dan pengguna akan diarahkan kembali ke HalamanLogin. Jika gagal (misalnya karena username sudah ada), pesan *error* akan ditampilkan.
- Terdapat MouseListener pada label "Kembali ke Login" (backToLoginLabel) yang berfungsi sebagai tautan untuk menutup jendela registrasi dan kembali ke halaman login.
- Kelas ini juga memanfaatkan metode *helper* addPlaceholderAndFocusListeners(...) untuk memberikan efek *placeholder* dan mengubah visual kolom input saat berinteraksi, serta menggunakan kelas dalam (*inner class*) RoundedTextField dan RoundedJPasswordField yang sama dengan HalamanLogin.

### **Penjelasan Konsep OOP**

- Menggunakan konsep Enkapsulasi
  - Atribut-atribut untuk kolom input dideklarasikan sebagai private, sehingga hanya dapat diakses dari dalam kelas Registrasi.
  - Kelas dalam RoundedTextField dan RoundedJPasswordField menyembunyikan detail implementasi dari proses penggambaran komponen, sehingga kelas luar hanya perlu menggunakannya.
- Menggunakan konsep Pewarisan (*Inheritance*)
  - Registrasi adalah turunan dari JFrame.
  - RoundedTextField adalah turunan dari JTextField.
  - RoundedJPasswordField adalah turunan dari JPasswordField.
  - Pewarisan memungkinkan penggunaan kembali fungsionalitas dari kelas induk sambil menambahkan fitur-fitur baru.
- Menggunakan konsep Polimorfisme
  - Penggunaan @Override pada metode-metode di dalam MouseAdapter dan FocusAdapter adalah contoh dari *method overriding*.
  - Kelas dalam RoundedTextField dan RoundedJPasswordField juga meng-*override* metode paintComponent dan paintBorder dari kelas induknya.
- Menggunakan konsep Abstraksi
  - Kelas Registrasi berinteraksi dengan DatabaseHelper melalui metode registerUser(). Ini adalah bentuk abstraksi, di mana Registrasi tidak perlu mengetahui detail kompleks tentang perintah SQL INSERT atau cara kerja koneksi database; ia hanya perlu tahu bahwa metode tersebut akan menyimpan

data pengguna.

## 10. Dashboard.java

```
1 import java.awt.*;
2 import java.awt.event.MouseAdapter;
3 import java.awt.event.MouseEvent;
4 import javax.swing.*;
5 import javax.swing.border.Border;
6 import javax.swing.border.EmptyBorder;
7
8 public class Dashboard extends JFrame {
9
10    public Dashboard() {
11        setTitle("Dashboard - CyclePro");
12        setSize(900, 700);
13        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14        setLocationRelativeTo(null);
15        setLayout(new BorderLayout(hgap:0, vgap:0));
16        getContentPane().setBackground(Colors.BACKGROUND_PRIMARY);
17
18        JPanel headerPanel = new JPanel();
19        headerPanel.setBackground(Colors.NAVBAR_BACKGROUND);
20        headerPanel.setLayout(new BorderLayout(hgap:15, vgap:0));
21        headerPanel.setBorder(new EmptyBorder(top:10, left:20, bottom:10, right:20));
22
23        JLabel appTitleLabel = new JLabel(text:"CYCLEPRO");
24        appTitleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:28));
25        appTitleLabel.setForeground(Colors.NAVBAR_TEXT);
26        headerPanel.add(appTitleLabel, BorderLayout.WEST);
27
28        JPanel userInfoPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, hgap:15, vgap:0));
29        userInfoPanel.setOpaque(isOpaque:false);
30
31        JButton historyButton = new JButton(text:"Riwayat Pesanan");
32        historyButton.setBackground(Colors.BUTTON_PRIMARY_BACKGROUND);
33        historyButton.setForeground(Colors.BUTTON_PRIMARY_TEXT);
34        historyButton.setFont(new Font(name:"Arial", Font.BOLD, size:12));
35        historyButton.setFocusPainted(b:false);
36        historyButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
37        historyButton.addActionListener(e -> {
38            if (Main.currentUser != null) {
39                new HalamanRiwayatPesanan().setVisible(b:true);
40                dispose();
41            } else {
42                JOptionPane.showMessageDialog(this, message:"Silakan login untuk melihat riwayat pesanan.", title:"Peringatan",
43                JOptionPane.WARNING_MESSAGE);
44            }
45        });
46        userInfoPanel.add(historyButton);
47
48        String username = (Main.currentUser != null) ? Main.currentUser.getUsername() : "Guest";
49        JLabel profileLabel = new JLabel("Halo, " + username + "!");
50        profileLabel.setForeground(Colors.NAVBAR_TEXT);
51        profileLabel.setFont(new Font(name:"Arial", Font.BOLD, size:14));
52        userInfoPanel.add(profileLabel);
53
54        JButton logoutButton = new JButton(text:"Logout");
55        logoutButton.setBackground(Colors.BUTTON_DANGER_BACKGROUND);
56        logoutButton.setForeground(Colors.BUTTON_DANGER_TEXT);
57        logoutButton.setFont(new Font(name:"Arial", Font.BOLD, size:12));
58        logoutButton.setFocusPainted(b:false);
59        logoutButton.setCursor(new Cursor(Cursor.HAND_CURSOR));
60        logoutButton.addActionListener(e -> {
61            Main.currentUser = null;
62            new HalamanLogin().setVisible(b:true);
63            dispose();
64        });
65        userInfoPanel.add(logoutButton);
66        headerPanel.add(userInfoPanel, BorderLayout.EAST);
67        add(headerPanel, BorderLayout.NORTH);
68
69        JPanel mainContentPanel = new JPanel(new BorderLayout());
70        mainContentPanel.setBackground(Colors.BACKGROUND_PRIMARY);
71        mainContentPanel.setBorder(new EmptyBorder(top:25, left:25, bottom:25, right:25));
72
73        JLabel chooseCategoryLabel = new JLabel(text:"Pilih Kategori Sepeda:");
74        chooseCategoryLabel.setFont(new Font(name:"Arial", Font.BOLD, size:20));
75        chooseCategoryLabel.setForeground(Colors.TEXT_PRIMARY);
76        mainContentPanel.add(chooseCategoryLabel, BorderLayout.NORTH);
77
78        JPanel categoriesPanel = new JPanel();
79        categoriesPanel.setLayout(new BoxLayout(categoriesPanel, BoxLayout.Y_AXIS));
80        categoriesPanel.setBackground(Colors.BACKGROUND_PRIMARY);
81        categoriesPanel.setBorder(new EmptyBorder(top:20, left:0, bottom:0, right:0));
```

```

82     String[] bikeTypes = {"Sepeda BMX", "Sepeda Gunung", "Sepeda Lipat"};
83     String[] categoriesDB = {"BMX", "Gunung", "Lipat"};
84     String[] imagePaths = {
85         "/img/imgDashboard/sepedaBMX.png",
86         "/img/imgDashboard/sepedaGunung.png",
87         "/img/imgDashboard/sepedaLIPAT.png"
88     };
89
90     for (int i = 0; i < bikeTypes.length; i++) {
91         final String categoryName = bikeTypes[i];
92         final String categoryDBName = categoriesDB[i];
93         final String imagePath = imagePaths[i];
94
95         JPanel categoryItemPanel = new JPanel(new BorderLayout(hgap:20, vgap:0));
96         categoryItemPanel.setBackground(Colors.BACKGROUND_SECONDARY);
97
98         Border defaultBorder = BorderFactory.createCompoundBorder(
99             BorderFactory.createLineBorder(Colors.BORDER_COLOR, thickness:1),
100            BorderFactory.createEmptyBorder(top:15, left:15, bottom:15, right:15)
101        );
102        categoryItemPanel.setBorder(defaultBorder);
103
104        categoryItemPanel.setMaximumSize(new Dimension(Integer.MAX_VALUE, height:150));
105
106        JLabel imageLabel = new JLabel();
107        imageLabel.setPreferredSize(new Dimension(width:120, height:120));
108        imageLabel.setHorizontalTextPosition(SwingConstants.CENTER);
109        imageLabel.setVerticalTextPosition(SwingConstants.CENTER);
110        try {
111            java.net.URL imgUrl = getClass().getResource(imagePath);
112            if (imgUrl != null) {
113                ImageIcon bikeIcon = new ImageIcon(imgUrl);
114                Image image = bikeIcon.getImage().getScaledInstance(width:100, height:100, Image.SCALE_SMOOTH);
115                imageLabel.setIcon(new ImageIcon(image));
116            } else {
117                System.err.println("Gambar kategori tidak ditemukan: " + imagePath);
118                imageLabel.setText(text:"Gambar Tidak Ditemukan");
119                imageLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:10));
120                imageLabel.setForeground(Color.RED);
121            }
122        } catch (Exception e) {
123            System.err.println("Error memuat gambar kategori: " + imagePath + " - " + e.getMessage());
124            imageLabel.setText(text:"Gagal Muat Gambar");
125            imageLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:10));
126            imageLabel.setForeground(Color.RED);
127        }
128        categoryItemPanel.add(imageLabel, BorderLayout.WEST);
129
130        JLabel categoryLabelText = new JLabel(categoryName);
131        categoryLabelText.setFont(new Font(name:"Arial", Font.BOLD, size:22));
132        categoryLabelText.setForeground(Colors.TEXT_PRIMARY);
133        categoryItemPanel.add(categoryLabelText, BorderLayout.CENTER);
134
135        categoryItemPanel.setCursor(new Cursor(Cursor.HAND_CURSOR));
136        categoryItemPanel.addMouseListener(new MouseAdapter() {
137            Color originalTextForeground = categoryLabelText.getForeground();
138
139            @Override
140            public void mouseClicked(MouseEvent e) {
141                new KategoriSepeda(categoryDBName).setVisible(b:true);
142                dispose();
143            }
144            @Override
145            public void mouseEntered(MouseEvent e) {
146                categoryItemPanel.setBackground(Colors.BUTTON_GOLD_BACKGROUND);
147                categoryLabelText.setForeground(Color.WHITE);
148
149                categoryItemPanel.setBorder(BorderFactory.createCompoundBorder(
150                    BorderFactory.createLineBorder(Colors.BUTTON_GOLD_BACKGROUND.darker(), thickness:3),
151                    BorderFactory.createEmptyBorder(top:15, left:15, bottom:15, right:15)
152                ));
153            }

```

```

154
155     }
156
157     @Override
158     public void mouseExited(MouseEvent e) {
159         categoryItemPanel.setBackground(Colors.BACKGROUND_SECONDARY);
160         categoryLabelText.setForeground(originalTextForeground);
161         categoryItemPanel.setBorder(defaultBorder);
162     });
163     categoriesPanel.add(categoryItemPanel);
164     categoriesPanel.add(Box.createRigidArea(new Dimension(width:0, height:20)));
165
166     JScrollPane scrollPane = new JScrollPane(categoriesPanel);
167     scrollPane.setBorder(BorderFactory.createEmptyBorder());
168     scrollPane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
169     scrollPane.getVerticalScrollBar().setUnitIncrement(unitIncrement:16);
170     mainContentPanel.add(scrollPane, BorderLayout.CENTER);
171
172     add(mainContentPanel, BorderLayout.CENTER);
173 }
174 }
```

## Penjelasan Kode Program

Kelas Dashboard.java adalah antarmuka pengguna (GUI) yang berfungsi sebagai halaman utama bagi pengguna setelah berhasil login. Halaman ini menjadi pusat navigasi di mana pengguna dapat melihat kategori produk, mengakses riwayat pesanan, dan keluar dari aplikasi.

- Mendeklarasikan public class Dashboard extends JFrame, yang menandakan bahwa kelas ini adalah sebuah jendela utama aplikasi yang mewarisi semua fungsionalitas dari kelas JFrame.
- Layout utama jendela menggunakan BorderLayout untuk membagi area menjadi bagian atas (NORTH) untuk *header* dan bagian tengah (CENTER) untuk konten utama.
- Panel *header* (headerPanel) berisi judul aplikasi "CYCLEPRO", dan sebuah panel info pengguna (userInfoPanel) di sebelah kanan.
- Panel info pengguna menampilkan tombol "Riwayat Pesanan", sapaan "Halo, [nama pengguna]", dan tombol "Logout". Data nama pengguna diambil secara dinamis dari variabel statis Main.currentUser.
- Konten utama (mainContentPanel) menampilkan daftar kategori sepeda. Daftar ini dibuat secara dinamis menggunakan sebuah *loop* yang membaca dari array bikeTypes, categoriesDB, dan imagePaths.
- Setiap kategori ditampilkan sebagai sebuah kartu (categoryItemPanel) yang berisi gambar dan nama kategori. Terdapat penanganan *error* jika file gambar tidak ditemukan.
- Terdapat MouseListener yang ditambahkan pada setiap kartu kategori untuk memberikan interaktivitas:

1. mouseClicked: Saat kartu diklik, aplikasi akan membuka jendela KategoriSepeda yang sesuai dan menutup jendela Dashboard saat ini.
  2. mouseEntered: Saat kursor masuk, warna latar belakang dan teks kartu berubah, dan *border* menjadi lebih tebal untuk memberikan efek *hover* sebagai umpan balik visual.
  3. mouseExited: Saat kursor keluar, tampilan kartu kembali ke kondisi semula.
- Seluruh panel kategori dibungkus di dalam JScrollPane untuk memungkinkan pengguna melakukan *scroll* jika daftar kategori melebihi tinggi layar.
  - Terdapat ActionListener pada tombol historyButton untuk membuka HalamanRiwayatPesanan dan pada logoutButton untuk keluar dari sesi pengguna dan kembali ke HalamanLogin.

### **Penjelasan Konsep OOP**

- Menggunakan konsep Pewarisan (*Inheritance*)
  - Dashboard adalah turunan dari JFrame, sehingga ia mewarisi semua sifat dan perilaku sebuah jendela aplikasi.
- Menggunakan konsep Enkapsulasi
  - Logika untuk membuat setiap bagian dari antarmuka (seperti *header* dan panel kategori) terbungkus rapi di dalam konstruktur. Detail pembuatan setiap kartu disembunyikan di dalam sebuah *loop*, membuat kode utama lebih bersih.
- Menggunakan konsep Polimorfisme
  - Penggunaan MouseAdapter adalah contoh polimorfisme. Daripada mengimplementasikan semua metode dari *interface* MouseListener, kita hanya meng-*override* metode yang relevan (mouseClicked, mouseEntered, mouseExited).
- Menggunakan konsep Abstraksi
  - Kelas Dashboard berinteraksi dengan HalamanRiwayatPesanan dan KategoriSepeda dengan cara yang sederhana (new KategoriSepeda(...)). Ia tidak perlu tahu detail kompleks tentang bagaimana halaman-halaman tersebut akan menampilkan datanya.
  - Penggunaan kelas Colors (misalnya Colors.NAVBAR\_BACKGROUND) juga merupakan bentuk abstraksi, di mana detail nilai warna disembunyikan di balik nama konstanta yang deskriptif.

## 11. AdminDashboard.java

```
1  import javax.swing.*;
2  import javax.swing.table.DefaultTableModel;
3  import javax.swing.border.EmptyBorder;
4  import java.awt.*;
5  import java.util.List;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.ActionListener;
8  import java.awt.event.MouseAdapter;
9  import java.awt.event.MouseEvent;
10
11 public class AdminDashboard extends JFrame {
12
13     private DefaultTableModel tableModel;
14     private JTable ordersTable;
15
16     public AdminDashboard() {
17         setTitle(title:"Admin Dashboard - Kelola Pesanan");
18         setSize(width:1000, height:700);
19         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20         setLocationRelativeTo(c:null);
21         setLayout(new BorderLayout(hgap:10, vgap:10));
22
23         getContentPane().setBackground(Colors.BACKGROUND_PRIMARY);
24
25         JPanel headerPanel = new JPanel(new BorderLayout(hgap:10, vgap:0));
26         headerPanel.setBackground(Colors.NAVBAR_BACKGROUND);
27         headerPanel.setBorder(new EmptyBorder(top:10, left:15, bottom:10, right:15));
28
29         JLabel titleLabel = new JLabel(text:"Data Pesanan");
30         titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:28));
31         titleLabel.setForeground(Colors.NAVBAR_TEXT);
32         headerPanel.add(titleLabel, BorderLayout.WEST);
33
34         JLabel adminInfoLabel = new JLabel("Admin: " + (Main.currentUser != null ? Main.currentUser.getUsername() : "N/A"));
35         adminInfoLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
36         adminInfoLabel.setForeground(Colors.NAVBAR_TEXT);
37         headerPanel.add(adminInfoLabel, BorderLayout.EAST);
38
39         add(headerPanel, BorderLayout.NORTH);
40
41         JPanel mainContentPanel = new JPanel(new BorderLayout());
42         mainContentPanel.setBorder(BorderFactory.createEmptyBorder(top:15, left:15, bottom:15, right:15));
43         mainContentPanel.setBackground(Colors.BACKGROUND_PRIMARY);
44
45         tableModel = new DefaultTableModel(new Object[]{"ID Pesanan", "User", "Produk", "Tanggal", "Total", "Status", "Alamat Pengiriman", "Kurir", "Metode Bayar", "VA Number"}, 0) {
46             @Override
47             public boolean isCellEditable(int row, int column) {
48                 return false;
49             }
50         };
51         ordersTable = new JTable(tableModel);
52         ordersTable.setFont(new Font(name:"Arial", Font.PLAIN, size:12));
53         ordersTable.setRowHeight(rowHeight:25);
54         ordersTable.getTableHeader().setFont(new Font(name:"Arial", Font.BOLD, size:12));
55         ordersTable.getTableHeader().setBackground(Colors.BUTTON_PRIMARY_BACKGROUND);
56         ordersTable.getTableHeader().setForeground(Colors.BUTTON_PRIMARY_TEXT);
57         ordersTable.setSelectionBackground(Colors.BUTTON_PRIMARY_BACKGROUND.brighter());
58         ordersTable.setSelectionForeground(Color.WHITE);
59
60         JScrollPane scrollPane = new JScrollPane(ordersTable);
61         scrollPane.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR, thickness:1));
62         mainContentPanel.add(scrollPane, BorderLayout.CENTER);
63
64         add(mainContentPanel, BorderLayout.CENTER);
65
66         JPanel actionPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT, hgap:15, vgap:10));
67         actionPanel.setBackground(Colors.BACKGROUND_PRIMARY);
68
69         JButton refreshButton = new JButton(text:"Refresh Data");
70         styleButton(refreshButton, Colors.BUTTON_PRIMARY_BACKGROUND, Colors.BUTTON_PRIMARY_TEXT);
71         refreshButton.addActionListener(new ActionListener() {
72             @Override
73             public void actionPerformed(ActionEvent e) {
74                 loadOrdersData();
75             }
76         });
77     }
78 }
```

```

77     actionPanel.add(refreshButton);
78
79     JButton editStatusButton = new JButton(text:"Ubah Status Pesanan");
80     styleButton(editStatusButton, Colors.BUTTON_SUCCESS_BACKGROUND, Colors.BUTTON_SUCCESS_TEXT);
81     editStatusButton.addActionListener(new ActionListener() {
82         @Override
83         public void actionPerformed(ActionEvent e) {
84             int selectedRow = ordersTable.getSelectedRow();
85             if (selectedRow != -1) {
86                 int orderId = (int) ordersTable.getValueAt(selectedRow, column:0);
87                 String currentStatus = (String) ordersTable.getValueAt(selectedRow, column:5);
88
89                 String[] statusOptions = {"Pending", "Diproses", "Dikirim", "Selesai", "Dibatalkan"};
90                 String newPassword = (String) JOptionPane.showInputDialog(
91                     AdminDashboard.this,
92                     "Pilih status baru untuk Pesanan ID: " + orderId + " (Status saat ini: " + currentStatus + ")",
93                     title:"Ubah Status Pesanan",
94                     JOptionPane.QUESTION_MESSAGE,
95                     icon:null,
96                     statusOptions,
97                     currentStatus
98                 );
99
100                if (newPassword != null && !newPassword.equals(currentStatus)) {
101                    if (DatabaseHelper.updateOrderStatus(orderId, newPassword)) {
102                        JOptionPane.showMessageDialog(AdminDashboard.this, message:"Status pesanan berhasil diperbarui!");
103                        loadOrdersData();
104                    } else {
105                        JOptionPane.showMessageDialog(AdminDashboard.this, message:"Gagal memperbarui status pesanan.", title:"Error", JOptionPane.ERROR_MESSAGE);
106                    }
107                } else {
108                    JOptionPane.showMessageDialog(AdminDashboard.this, message:"Pilih pesanan yang akan diubah statusnya terlebih dahulu.", title:"Peringatan", JOptionPane.WARNING_MESSAGE);
109                }
110            }
111        });
112    });
113    actionPanel.add(editStatusButton);
114
115    JButton deleteOrderButton = new JButton(text:"Hapus Pesanan");
116    styleButton(deleteOrderButton, Colors.BUTTON_DANGER_BACKGROUND, Colors.BUTTON_DANGER_TEXT);
117    deleteOrderButton.addActionListener(new ActionListener() {
118        @Override
119        public void actionPerformed(ActionEvent e) {
120            int selectedRow = ordersTable.getSelectedRow();
121            if (selectedRow != -1) {
122                int orderId = (int) ordersTable.getValueAt(selectedRow, column:0);
123                String userName = (String) ordersTable.getValueAt(selectedRow, column:1);
124                String productName = (String) ordersTable.getValueAt(selectedRow, column:2);
125
126                int confirm = JOptionPane.showConfirmDialog(AdminDashboard.this,
127                    "Yakin ingin menghapus pesanan ID: " + orderId + " dari " + userName + " untuk produk " + productName + "?",
128                    title:"Konfirmasi Hapus Pesanan", JOptionPane.YES_NO_OPTION, JOptionPane.WARNING_MESSAGE);
129
130                if (confirm == JOptionPane.YES_OPTION) {
131                    if (DatabaseHelper.deleteOrder(orderId)) {
132                        JOptionPane.showMessageDialog(AdminDashboard.this, message:"Pesanan berhasil dihapus!");
133                        loadOrdersData();
134                    } else {
135                        JOptionPane.showMessageDialog(AdminDashboard.this, message:"Gagal menghapus pesanan.", title:"Error", JOptionPane.ERROR_MESSAGE);
136                    }
137                } else {
138                    JOptionPane.showMessageDialog(AdminDashboard.this, message:"Pilih pesanan yang akan dihapus terlebih dahulu.", title:"Peringatan", JOptionPane.WARNING_MESSAGE);
139                }
140            }
141        });
142    });

```

```

143     actionPanel.add(deleteOrderButton);
144
145     JButton logoutButton = new JButton(text:"Logout Admin");
146     styleButton(logoutButton, Colors.BUTTON_SECONDARY_BACKGROUND, Colors.BUTTON_SECONDARY_TEXT);
147     logoutButton.addActionListener(new ActionListener() {
148         @Override
149         public void actionPerformed(ActionEvent e) {
150             Main.currentUser = null;
151             new HalamanLogin().setVisible(b:true);
152             dispose();
153         }
154     });
155     actionPanel.add(logoutButton);
156
157     add(actionPanel_BorderLayout.SOUTH);
158     void AdminDashboard.loadOrdersData()
159     loadOrdersData();
160
161 }
162
163 private void loadOrdersData() {
164     tableModel.setRowCount(rowCount:0);
165     List<Pesanan> orders = DatabaseHelper.getAllOrders();
166
167     if (orders.isEmpty()) {
168         System.out.println(x:"Tidak ada pesanan ditemukan.");
169     } else {
170         for (Pesanan order : orders) {
171             String userName = (order.getUser() != null) ? order.getUser().getUsername() : "N/A";
172             String productName = (order.getProduct() != null) ? order.getProduct().getName() : "N/A";
173
174             tableModel.addRow(new Object[]{
175                 order.getOrderId(),
176                 userName,
177                 productName,
178                 order.getOrderDate(),
179                 String.format(format:"Rp %,.0f", order.getTotalPrice()),
180                 order.getStatus(),
181                 order.getShippingAddress(),
182                 order.getCourier(),
183                 order.getPaymentMethod(),
184                 order.getVaNumber() != null ? order.getVaNumber() : "-"
185             });
186         }
187     }
188 }
189
190 private void styleButton(JButton button, Color bgColor, Color textColor) {
191     button.setBackground(bgColor);
192     button.setForeground(textColor);
193     button.setFont(new Font(name:"Arial", Font.BOLD, size:14));
194     button.setFocusPainted(b:false);
195     button.setBorder(BorderFactory.createEmptyBorder(top:8, left:15, bottom:8, right:15));
196     button.setCursor(new Cursor(Cursor.HAND_CURSOR));
197     button.addMouseListener(new MouseAdapter() {
198         @Override
199         public void mouseEntered(MouseEvent e) {
200             button.setBackground(bgColor.darker());
201         }
202
203         @Override
204         public void mouseExited(MouseEvent e) {
205             button.setBackground(bgColor);
206         }
207     });
208 }

```

```

209     public static void main(String[] args) {
210         DatabaseHelper.createNewDatabase();
211         DatabaseHelper.createTables();
212         DatabaseHelper.addDefaultAdmin();
213
214         Main.currentUser = DatabaseHelper.authenticateAdmin(username:"admin", password:"admin123");
215
216         SwingUtilities.invokeLater(() -> {
217             if (Main.currentUser instanceof Admin) {
218                 new AdminDashboard().setVisible(b:true);
219             } else {
220                 JOptionPane.showMessageDialog(parentComponent:null, message:"Anda harus login sebagai Admin untuk mengakses dashboard ini.", title:"Akses Ditolak", JOptionPane.ERROR_MESSAGE);
221                 new HalamanLogin().setVisible(b:true);
222             }
223         });
224     }
225 }

```

## Penjelasan Kode Program

Kelas AdminDashboard.java adalah antarmuka pengguna (GUI) yang berfungsi sebagai halaman utama khusus untuk administrator. Halaman ini menyediakan fungsionalitas untuk melihat, mengelola, dan memanipulasi semua data pesanan yang masuk ke dalam sistem.

- Mendeklarasikan public class AdminDashboard extends JFrame, yang berarti kelas ini adalah sebuah jendela utama yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat atribut private yaitu tableModel (sebuah DefaultTableModel) untuk mengelola data tabel, dan ordersTable (sebuah JTable) untuk menampilkan data tersebut dalam bentuk baris dan kolom. DefaultTableModel di-override agar sel tabel tidak bisa diedit langsung oleh admin.
- Layout utama menggunakan BorderLayout yang membagi jendela menjadi tiga bagian: headerPanel di utara (atas), mainContentPanel yang berisi tabel di tengah, dan actionPanel yang berisi tombol-tombol aksi di selatan (bawah).
- Terdapat metode privat private void loadOrdersData(). Metode ini bertanggung jawab untuk memuat atau menyegarkan data pada tabel. Ia akan mengosongkan tabel terlebih dahulu, kemudian memanggil DatabaseHelper.getAllOrders() untuk mengambil semua data pesanan, dan akhirnya mengisi setiap baris tabel dengan data tersebut.
- Terdapat metode privat private void styleButton(...). Ini adalah metode *helper* yang digunakan untuk memberikan gaya (seperti warna, font, dan efek *hover*) yang konsisten pada semua tombol aksi, menunjukkan penerapan prinsip DRY (*Don't Repeat Yourself*).
- Panel aksi (actionPanel) berisi beberapa tombol dengan ActionListener untuk fungsionalitas manajemen pesanan:

1. refreshButton: Memanggil `loadOrdersData()` untuk mengambil data pesanan terbaru dari database.
  2. editStatusButton: Mengambil baris yang dipilih pada tabel, menampilkan dialog `JOptionPane.showInputDialog` dengan pilihan status baru, lalu memanggil `DatabaseHelper.updateOrderStatus()` untuk menyimpan perubahan.
  3. deleteOrderButton: Mengambil baris yang dipilih, menampilkan dialog konfirmasi `JOptionPane.showConfirmDialog` untuk memastikan admin yakin, lalu memanggil `DatabaseHelper.deleteOrder()` untuk menghapus pesanan.
  4. logoutButton: Mengatur `Main.currentUser` menjadi null dan mengarahkan admin kembali ke HalamanLogin.
- Kelas ini juga memiliki metode `main` sendiri yang berfungsi untuk tujuan pengujian (*testing*), memungkinkan developer menjalankan hanya jendela ini secara terpisah dengan login sebagai admin default.

### **Penjelasan Konsep OOP**

- Menggunakan konsep Pewarisan (*Inheritance*)
  - `AdminDashboard` adalah turunan langsung dari `JFrame`, sehingga mewarisi semua sifat dan perilaku dasar dari sebuah jendela Swing.
- Menggunakan konsep Enkapsulasi
  - Atribut-atribut seperti `tableModel` dan `ordersTable` dideklarasikan sebagai `private` untuk membatasi akses langsung dari luar kelas.
  - Logika yang kompleks untuk memuat data dan menata gaya tombol dienkapsulasi ke dalam metode privatnya masing-masing (`loadOrdersData()` dan `styleButton()`), membuat kode utama di konstruktur lebih bersih dan mudah dibaca.
- Menggunakan konsep Abstraksi
  - Kelas ini berinteraksi dengan database secara eksklusif melalui kelas `DatabaseHelper`. `AdminDashboard` tidak perlu tahu tentang query SQL atau detail koneksi database; ia hanya memanggil metode abstrak seperti `getAllOrders()` atau `deleteOrder()`. Ini menyembunyikan kompleksitas implementasi database.
- Menggunakan konsep Polimorfisme
  - Penggunaan `ActionListener` dan `MouseAdapter` sebagai kelas anonim adalah bentuk polimorfisme. Kita menyediakan implementasi spesifik untuk metode `actionPerformed` atau `mouseEntered/mouseExited` yang sesuai dengan kebutuhan setiap tombol.

- Peng-*override*-an metode `isCellEditable` pada `DefaultTableModel` juga merupakan contoh polimorfisme, di mana kita mengubah perilaku standar dari kelas induknya.

## 12. KategoriSepeda.java

```

1  import java.awt.*;
2  import java.awt.event.MouseAdapter;
3  import java.awt.event.MouseEvent;
4  import java.util.List;
5  import javax.swing.*;
6  import javax.swing.border.Border;
7
8  public class KategoriSepeda extends JFrame {
9      private String category;
10
11     public KategoriSepeda(String category) {
12         this.category = category;
13         setTitle("Daftar Sepeda " + category + " - CyclePro");
14         setSize(width:900, height:700);
15         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
16         setLocationRelativeTo(null);
17         setLayout(new BorderLayout(hgap:0,vgap:0));
18         getContentPane().setBackground(Colors.BACKGROUND_PRIMARY);
19
20         JPanel headerPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, hgap:15, vgap:10));
21         headerPanel.setBackground(Colors.NAVBAR_BACKGROUND);
22
23         JButton backButton = new JButton(text:"Kembali ke Dashboard");
24         backButton.setBackground(Colors.BUTTON_SECONDARY_BACKGROUND);
25         backButton.setForeground(Colors.BUTTON_SECONDARY_TEXT);
26         backButton.setFont(new Font(name:"Arial", Font.BOLD, size:12));
27         headerPanel.add(backButton);
28
29         headerPanel.add(Box.createHorizontalStrut(width:20));
30
31         JLabel titleLabelText = new JLabel("Pilih Sepeda " + category);
32         titleLabelText.setFont(new Font(name:"Arial", Font.BOLD, size:24));
33         titleLabelText.setForeground(Colors.NAVBAR_TEXT);
34         headerPanel.add(titleLabelText);
35         add(headerPanel, BorderLayout.NORTH);
36
37         JPanel bikesPanelWrapper = new JPanel(new BorderLayout());
38         bikesPanelWrapper.setBorder(BorderFactory.createEmptyBorder(top:15, left:15, bottom:15, right:15));
39         bikesPanelWrapper.setBackground(Colors.BACKGROUND_PRIMARY);
40
41         JPanel bikesPanel = new JPanel(new GridLayout(rows:0, cols:3, hgap:15, vgap:15));
42         bikesPanel.setBackground(Colors.BACKGROUND_PRIMARY);
43         List<Produk> productList = DatabaseHelper.getProductsByCategory(category);
44
45     if (productList.isEmpty()) {
46         bikesPanel.setLayout(new FlowLayout(FlowLayout.CENTER));
47         JLabel noProductLabel = new JLabel(text:"Tidak ada produk untuk kategori ini.");
48         noProductLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:16));
49         noProductLabel.setForeground(Colors.TEXT_SECONDARY);
50         bikesPanel.add(noProductLabel);
51     } else {
52         for (Produk product : productList) {
53             JPanel bikeItemPanel = new JPanel();
54             bikeItemPanel.setLayout(new BoxLayout(bikeItemPanel, BoxLayout.Y_AXIS));
55             bikeItemPanel.setBackground(Colors.BACKGROUND_SECONDARY);
56
57             Border defaultBorder = BorderFactory.createCompoundBorder(
58                 BorderFactory.createLineBorder(Colors.BORDER_COLOR, thickness:1),
59                 BorderFactory.createEmptyBorder(top:10, left:10, bottom:10, right:10)
60             );
61             bikeItemPanel.setBorder(defaultBorder);
62             bikeItemPanel.setPreferredSize(new Dimension(width:250, height:320));
63
64             JLabel imageDisplayLabel = new JLabel();
65             imageDisplayLabel.setPreferredSize(new Dimension(width:200, height:150));
66             imageDisplayLabel.setHorizontalTextPosition(SwingConstants.CENTER);
67             imageDisplayLabel.setVerticalTextPosition(SwingConstants.CENTER);
68             imageDisplayLabel.setBorder(BorderFactory.createLineBorder(Color.GRAY));
69
70         }
71     }
72
73     add(bikesPanel, BorderLayout.CENTER);
74
75     pack();
76     setVisible(true);
77 }

```

```

70     String subfolder;
71     if ("BMX".equalsIgnoreCase(category)) {
72         subfolder = "imgBMX";
73     } else if ("Gunung".equalsIgnoreCase(category)) {
74         subfolder = "imgGunung";
75     } else if ("Lipat".equalsIgnoreCase(category)) {
76         subfolder = "imgLipat";
77     } else {
78         subfolder = "produk";
79     }
80
81     String resourcePath = "/img/" + subfolder + "/" + product.getImagePath();
82
83     try {
84         java.net.URL imgUrl = getClass().getResource(resourcePath);
85         if (imgUrl != null) {
86             ImageIcon originalIcon = new ImageIcon(imgUrl);
87             Image originalImage = originalIcon.getImage();
88             Image scaledImage = originalImage.getScaledInstance(width:200, height:150, Image.SCALE_SMOOTH);
89             imageDisplayLabel.setIcon(new ImageIcon(scaledImage));
90         } else {
91             System.out.println("Resource gambar tidak ditemukan: " + resourcePath);
92             imageDisplayLabel.setText("<html><div style='width:180px;height:130px;text-align:center;color:black;'>Gmbr  
Tdk<br>Ditemukan:<br>" + product.getImagePath() + "</div></html>");
93         }
94     } catch (Exception ex) {
95         System.out.println("Error memuat resource gambar: " + resourcePath + " - " + ex.getMessage());
96         imageDisplayLabel.setText(text:"Gagal Muat");
97         imageDisplayLabel.setForeground(Color.RED);
98     }
99
100    imageDisplayLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
101
102    JLabel nameLabel = new JLabel(product.getName());
103    nameLabel.setFont(new Font(name:"Arial", Font.BOLD, size:16));
104    nameLabel.setForeground(Colors.TEXT_PRIMARY);
105    nameLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
106
107    JLabel priceLabel = new JLabel(String.format(format:"Rp %,.0f", product.getPrice()));
108    priceLabel.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
109    priceLabel.setForeground(Colors.TEXT_PRIMARY);
110    priceLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
111
112    JLabel stockLabel = new JLabel("Stok: " + product.getStock());
113    stockLabel.setFont(new Font(name:"Arial", Font.ITALIC, size:12));
114    stockLabel.setForeground(product.getStock() > 0 ? Colors.TEXT_SECONDARY : Colors.BUTTON_DANGER_BACKGROUND);
115    stockLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
116
117    bikeItemPanel.add(Box.createVerticalStrut(height:10));
118    bikeItemPanel.add(imageDisplayLabel);
119    bikeItemPanel.add(Box.createVerticalStrut(height:10));
120    bikeItemPanel.add(nameLabel);
121    bikeItemPanel.add(priceLabel);
122    bikeItemPanel.add(stockLabel);
123    bikeItemPanel.add(Box.createVerticalStrut(height:10));
124
125    bikeItemPanel.setCursor(new Cursor(Cursor.HAND_CURSOR));
126    bikeItemPanel.addMouseListener(new MouseAdapter() {
127        Color originalBackground = bikeItemPanel.getBackground();
128        Color originalNameForeground = nameLabel.getForeground();
129        Color originalPriceForeground = priceLabel.getForeground();
130        Color originalStockForeground = stockLabel.getForeground();
131
132        @Override
133        public void mouseClicked(MouseEvent e) {
134            if (product.getStock() > 0) {
135                new HalamanPembayaran(product).setVisible(b:true);
136                dispose();
137            } else {
138                JOptionPane.showMessageDialog(KategoriSepeda.this, message:"Stok produk ini habis.", title:"Stok Habis", JOptionPane.WARNING_MESSAGE);
139            }
140        }
141    });

```

```

141
142     @Override
143     public void mouseEntered(MouseEvent e) {
144         bikeItemPanel.setBackground(Colors.BUTTON_GOLD_BACKGROUND);
145         nameLabel.setForeground(Color.WHITE);
146         priceLabel.setForeground(Color.WHITE);
147         stockLabel.setForeground(Color.WHITE);
148
149         bikeItemPanel.setBorder(BorderFactory.createCompoundBorder(
150             BorderFactory.createLineBorder(Colors.BUTTON_GOLD_BACKGROUND.darker(), thickness:3),
151             BorderFactory.createEmptyBorder(top:10, left:10, bottom:10, right:10)
152         ));
153     }
154
155     @Override
156     public void mouseExited(MouseEvent e) {
157         bikeItemPanel.setBackground(originalBackground);
158         nameLabel.setForeground(originalNameForeground);
159         priceLabel.setForeground(originalPriceForeground);
160         stockLabel.setForeground(originalStockForeground);
161         bikeItemPanel.setBorder(defaultBorder);
162     }
163     bikesPanel.add(bikeItemPanel);
164 }
165
166 bikesPanelWrapper.add(bikesPanel, BorderLayout.CENTER);
167 JScrollPane scrollPane = new JScrollPane(bikesPanelWrapper);
168 scrollPane.setBorder(BorderFactory.createEmptyBorder());
169 add(scrollPane, BorderLayout.CENTER);
170
171 backButton.addActionListener(e -> {
172     new Dashboard().setVisible(b:true);
173     dispose();
174 });
175
176 }

```

## Penjelasan Kode Program

Kelas KategoriSepeda.java adalah antarmuka pengguna (GUI) yang berfungsi untuk menampilkan daftar produk sepeda berdasarkan kategori yang dipilih oleh pengguna dari halaman Dashboard.

- Mendeklarasikan public class KategoriSepeda extends JFrame, yang berarti kelas ini adalah sebuah jendela utama yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat atribut private String category untuk menyimpan nama kategori (misalnya, "BMX", "Gunung") yang diterima dari halaman sebelumnya.
- Konstruktor public KategoriSepeda(String category) menerima nama kategori sebagai parameter. Judul jendela dan label di-set secara dinamis menggunakan nilai dari parameter ini.
- Terdapat headerPanel di bagian atas yang berisi tombol "Kembali ke Dashboard" dan judul halaman yang dinamis.
- Konten utama menggunakan GridLayout untuk menampilkan produk dalam format kartu (grid).
- Data produk diambil dari database dengan memanggil metode DatabaseHelper.getProductsByCategory(category). Ini menunjukkan interaksi antara lapisan View (antarmuka) dan lapisan Model/Controller (logika data).

- Terdapat penanganan kasus jika tidak ada produk yang ditemukan untuk kategori yang dipilih, di mana sebuah pesan akan ditampilkan kepada pengguna.
- Jika produk ditemukan, aplikasi akan melakukan *looping* untuk setiap objek Produk dan membuat sebuah kartu JPanel (bikeItemPanel) untuk masing-masing produk.
- Setiap kartu produk menampilkan gambar, nama, harga, dan jumlah stok. Path gambar produk ditentukan secara dinamis berdasarkan kategori untuk memuat dari subfolder yang benar.
- Terdapat MouseListener yang ditambahkan pada setiap kartu produk untuk memberikan interaktivitas:
  1. mouseClicked: Jika stok tersedia, mengklik kartu akan membuka HalamanPembayaran dengan membawa data produk yang dipilih. Jika stok habis, sebuah pesan peringatan akan muncul.
  2. mouseEntered / mouseExited: Mengimplementasikan efek *hover* visual, di mana tampilan kartu akan berubah saat kurSOR mouse berada di atasnya dan kembali normal saat kurSOR pergi.
- Seluruh panel produk dibungkus dalam JScrollPane agar pengguna dapat melakukan *scroll* jika jumlah produk melebihi ukuran layar.

### **Penjelasan Konsep OOP**

- Menggunakan konsep Pewarisan (*Inheritance*)
  - KategoriSepeda adalah turunan dari JFrame, sehingga ia dapat berfungsi sebagai jendela aplikasi.
- Menggunakan konsep Enkapsulasi:
  - Atribut category dideklarasikan sebagai private untuk menjaga data agar hanya dapat diakses dari dalam kelas.
  - Logika kompleks untuk memuat data dari database dan menyusunnya menjadi kartu-kartu produk terbungkus rapi di dalam konstruktur, menyembunyikan detail implementasi dari dunia luar.
- Menggunakan konsep Polimorfisme
  - Penggunaan MouseAdapter untuk menangani *event* klik dan *hover* adalah bentuk polimorfisme, di mana kita hanya perlu meng-*override* metode yang kita butuhkan dari kelas induknya.
- Menggunakan konsep Abstraksi
  - Kelas ini berinteraksi dengan DatabaseHelper melalui metode

getProductsByCategory(). KategoriSepeda tidak perlu tahu bagaimana DatabaseHelper mengambil data dari database; ia hanya perlu menerima hasilnya berupa List<Produk>.

- Saat kartu diklik, ia membuat objek new HalamanPembayaran(product). Ini adalah abstraksi di mana KategoriSepeda mendelegasikan proses pembayaran ke kelas lain tanpa perlu mengetahui detail implementasinya.

### 13. HalamanPembayaran.java

```
1 import javax.swing.*;
2 import javax.swing.border.EmptyBorder;
3 import java.awt.*;
4 import java.text.SimpleDateFormat;
5 import java.util.Date;
6
7 public class HalamanPembayaran extends JFrame {
8     private Produk selectedProduct;
9     private JComboBox<String> paymentMethodComboBox;
10    private JComboBox<String> courierComboBox;
11    private JLabel vaInfoLabel;
12    private JTextArea addressArea;
13    private JTextArea productDescriptionArea;
14
15    public HalamanPembayaran(Produk product) {
16        this.selectedProduct = product;
17        setTitle(title:"Pembayaran - CyclePro");
18        setSize(width:950, height:700);
19        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20        setLocationRelativeTo(c:null);
21        setLayout(new BorderLayout(hgap:0, vgap:0));
22        getContentPane().setBackground(Colors.BACKGROUND_PRIMARY);
23
24        JPanel topBannerPanel = new JPanel(new BorderLayout(hgap:10, vgap:0));
25        topBannerPanel.setBackground(Colors.NAVBAR_BACKGROUND);
26        topBannerPanel.setBorder(new EmptyBorder(top:5, left:15, bottom:5, right:15));
27
28        JLabel logoLabel = new JLabel(text:"CYCLEPRO");
29        logoLabel.setFont(new Font(name:"Arial", Font.BOLD, size:24));
30        logoLabel.setForeground(Colors.NAVBAR_TEXT);
31        topBannerPanel.add(logoLabel, BorderLayout.WEST);
32
33        JLabel taglineLabel = new JLabel(text:"Solusi Kebutuhan Sepeda Anda");
34        taglineLabel.setFont(new Font(name:"Arial", Font.ITALIC, size:14));
35        taglineLabel.setForeground(Colors.NAVBAR_TEXT);
36        taglineLabel.setHorizontalAlignment(SwingConstants.CENTER);
37        topBannerPanel.add(taglineLabel, BorderLayout.CENTER);
38        add(topBannerPanel, BorderLayout.NORTH);
39
40        JPanel mainPagePanel = new JPanel(new BorderLayout());
41        mainPagePanel.setBackground(Colors.BACKGROUND_PRIMARY);
42        add(mainPagePanel, BorderLayout.CENTER);
43
44
45        JPanel productInfoBarPanel = new JPanel(new BorderLayout());
46        productInfoBarPanel.setBackground(new Color(r:255, g:204, b:0));
47        productInfoBarPanel.setBorder(new EmptyBorder(top:10, left:20, bottom:10, right:20));
48
49        JLabel productNameLabelBar = new JLabel(selectedProduct.getName());
50        productNameLabelBar.setFont(new Font(name:"Arial", Font.BOLD, size:28));
51        productNameLabelBar.setForeground(Color.BLACK);
52        productInfoBarPanel.add(productNameLabelBar, BorderLayout.WEST);
53
54        JLabel productPriceLabelBar = new JLabel(String.format(format:"Rp %,.0f", selectedProduct.getPrice()));
55        productPriceLabelBar.setFont(new Font(name:"Arial", Font.BOLD, size:28));
56        productPriceLabelBar.setForeground(Color.BLACK);
57        productInfoBarPanel.add(productPriceLabelBar, BorderLayout.EAST);
58        mainPagePanel.add(productInfoBarPanel, BorderLayout.NORTH);
59
```

```

60 // Main Content Panel
61 JPanel mainContentPanel = new JPanel(new BorderLayout(hgap:15, vgap:15));
62 mainContentPanel.setBorder(new EmptyBorder(top:15, left:20, bottom:15, right:20));
63 mainContentPanel.setBackground(Colors.BACKGROUND_PRIMARY);
64 mainPagePanel.add(mainContentPanel, BorderLayout.CENTER);
65
66 // Panel Gambar Produk
67 JPanel productImagePanel = new JPanel(new BorderLayout());
68 productImagePanel.setBackground(Colors.BACKGROUND_SECONDARY);
69 productImagePanel.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
70 productImagePanel.setPreferredSize(new Dimension(width:400, height:0));
71
72 JLabel imageLabel = new JLabel();
73 imageLabel.setHorizontalTextPosition(SwingConstants.CENTER);
74
75 String subfolder;
76 String category = selectedProduct.getCategory();
77 if ("BMX".equalsIgnoreCase(category)) {
78     subfolder = "imgBMX";
79 } else if ("Gunung".equalsIgnoreCase(category)) {
80     subfolder = "imgGunung";
81 } else if ("Lipat".equalsIgnoreCase(category)) {
82     subfolder = "imgLipat";
83 } else {
84     subfolder = "produk";
85 }
86
87 String resourcePath = "/img/" + subfolder + "/" + selectedProduct.getImagePath();
88
89 try {
90     java.net.URL imgUrl = getClass().getResource(resourcePath);
91
92     if (imgUrl != null) {
93         ImageIcon bikeIcon = new ImageIcon(imgUrl);
94         Image image = bikeIcon.getImage().getScaledInstance(width:380, height:280, Image.SCALE_SMOOTH);
95         imageLabel.setIcon(new ImageIcon(image));
96     } else {
97         imageLabel.setText("<html><div style='width:380px;height:280px;background-color:#ECECEC;text-align:center;line-height:140px;color:black;border:1px solid #DADADA;'>Gambar:<br>" + selectedProduct.getName() + "<br>tidak ditemukan (" + resourcePath + ")</div></html>");
98         System.err.println("Resource gambar tidak ditemukan: " + resourcePath);
99     }
100 } catch (Exception e) {
101     imageLabel.setText(text:"Gagal memuat gambar produk.");
102     System.err.println("Error memuat resource gambar: " + resourcePath + " - " + e.getMessage());
103     e.printStackTrace();
104 }
105
106 productImagePanel.add(imageLabel, BorderLayout.CENTER);
107 mainContentPanel.add(productImagePanel, BorderLayout.EAST);
108
109 // Panel Detail dan Form
110 JPanel detailsFormPanel = new JPanel();
111 detailsFormPanel.setLayout(mgr:null);
112 detailsFormPanel.setBackground(Colors.BACKGROUND_SECONDARY);
113 detailsFormPanel.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
114 mainContentPanel.add(detailsFormPanel, BorderLayout.CENTER);
115
116 int yPos = 20;
117 int xMargin = 20;
118 int labelWidth = 150;
119 int fieldWidth = 250;
120 int componentHeight = 25;
121 int areaHeight = 70;
122 int spacing = 15;
123
124 // Deskripsi Produk
125 JLabel desclabel = new JLabel(text:"Deskripsi Produk:");
126 desclabel.setBounds(xMargin, yPos, labelWidth, componentHeight);
127 desclabel.setForeground(Colors.TEXT_PRIMARY);
128 desclabel.setFont(new Font(name:"Arial", Font.BOLD, size:14));
129 detailsFormPanel.add(desclabel);
130 yPos += componentHeight + 5;
131

```

```

132 productDescriptionArea = new JTextArea(selectedProduct.getDescription());
133 productDescriptionArea.setEditable(b:false);
134 productDescriptionArea.setLineWrap(wrap:true);
135 productDescriptionArea.setWrapStyleWord(word:true);
136 productDescriptionArea.setFont(new Font(name:"Arial", Font.PLAIN, size:13));
137 productDescriptionArea.setBackground(Colors.BACKGROUND_PRIMARY);
138 productDescriptionArea.setForeground(Colors.TEXT_SECONDARY);
139 JScrollPane descScrollPane = new JScrollPane(productDescriptionArea);
140 descScrollPane.setBounds(xMargin, yPos, fieldWidth + 100, areaHeight);
141 descScrollPane.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
142 detailsFormPanel.add(descScrollPane);
143 yPos += areaHeight + spacing;
144
145 // Metode Pembayaran
146 JLabel paymentMethodLabelText = new JLabel(text:"Metode Pembayaran:");
147 paymentMethodLabelText.setBounds(xMargin, yPos, labelWidth, componentHeight);
148 paymentMethodLabelText.setForeground(Colors.TEXT_PRIMARY);
149 detailsFormPanel.add(paymentMethodLabelText);
150
151 String[] paymentMethods = {"Virtual Account", "COD (Bayar di Tempat)"};
152 paymentMethodComboBox = new JComboBox<>(paymentMethods);
153 paymentMethodComboBox.setBounds(xMargin + labelWidth + 10, yPos, fieldWidth, componentHeight);
154 paymentMethodComboBox.setBackground(Color.WHITE);
155 paymentMethodComboBox.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
156 detailsFormPanel.add(paymentMethodComboBox);
157 yPos += componentHeight + 5;
158
159 vaInfoLabel = new JLabel(text:" ");
160 vaInfoLabel.setForeground(Colors.TEXT_LINK);
161 vaInfoLabel.setFont(new Font(name:"Arial", Font.BOLD, size:12));
162 vaInfoLabel.setBounds(xMargin + labelWidth + 10, yPos, fieldWidth + 50, componentHeight);
163 detailsFormPanel.add(vaInfoLabel);
164 yPos += componentHeight + spacing;
165
166 // Kurir Ekspedisi
167 JLabel courierLabelText = new JLabel(text:"Kurir Ekspedisi:");
168 courierLabelText.setBounds(xMargin, yPos, labelWidth, componentHeight);
169 courierLabelText.setForeground(Colors.TEXT_PRIMARY);
170 detailsFormPanel.add(courierLabelText);
171
172 String[] couriers = {"SiCepat", "JNE", "J&T Express"};
173 courierComboBox = new JComboBox<>(couriers);
174 courierComboBox.setBounds(xMargin + labelWidth + 10, yPos, fieldWidth, componentHeight);
175 courierComboBox.setBackground(Color.WHITE);
176 courierComboBox.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
177 detailsFormPanel.add(courierComboBox);
178 yPos += componentHeight + spacing;
179
180 // Alamat Pengiriman
181 JLabel addressLabelText = new JLabel(text:"Alamat Pengiriman:");
182 addressLabelText.setBounds(xMargin, yPos, labelWidth, componentHeight);
183 addressLabelText.setForeground(Colors.TEXT_PRIMARY);
184 detailsFormPanel.add(addressLabelText);
185 yPos += componentHeight + 5;
186
187 addressArea = new JTextArea(Main.currentUser != null ? Main.currentUser.getAddress() : "Alamat tidak tersedia");
188 addressArea.setEditable(b:false);
189 addressArea.setLineWrap(wrap:true);
190 addressArea.setWrapStyleWord(word:true);
191 addressArea.setBackground(Colors.BACKGROUND_PRIMARY);
192 addressArea.setForeground(Colors.TEXT_SECONDARY);
193 JScrollPane addressScrollPane = new JScrollPane(addressArea);
194 addressScrollPane.setBounds(xMargin, yPos, fieldWidth + 100, areaHeight - 20);
195 addressScrollPane.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR));
196 detailsFormPanel.add(addressScrollPane);
197 yPos += (areaHeight - 20) + spacing + 10;
198
199 // Tombol Aksi (Confirm & Cancel)
200 JButton confirmPaymentButton = new JButton(text:"BELI SEKARANG");
201 confirmPaymentButton.setBackground(Colors.BUTTON_SUCCESS_BACKGROUND);
202 confirmPaymentButton.setForeground(Colors.BUTTON_SUCCESS_TEXT);
203 confirmPaymentButton.setFont(new Font(name:"Arial", Font.BOLD, size:16));
204 confirmPaymentButton.setBounds(xMargin, yPos, fieldWidth + 100, height:40);
205 detailsFormPanel.add(confirmPaymentButton);
206 yPos += 40 + 10;
207
208 JButton cancelButton = new JButton(text:"Batal");
209 cancelButton.setBackground(Colors.BUTTON_DANGER_BACKGROUND);
210 cancelButton.setForeground(Colors.BUTTON_DANGER_TEXT);
211 cancelButton.setFont(new Font(name:"Arial", Font.PLAIN, size:14));
212 cancelButton.setBounds(xMargin, yPos, (fieldWidth + 100) / 2 - 5, height:30);
213 detailsFormPanel.add(cancelButton);
214

```

```

215     paymentMethodComboBox.addActionListener(e -> {
216         String selectedMethod = (String) paymentMethodComboBox.getSelectedItem();
217         if ("Virtual Account".equals(selectedMethod) && Main.currentUser != null) {
218             vaInfoLabel.setText("Kode VA: 727" + Main.currentUser.getPhoneNumber());
219         } else {
220             vaInfoLabel.setText(text: " ");
221         }
222     });
223     if (paymentMethodComboBox.getActionListeners().length > 0) {
224         paymentMethodComboBox.getActionListeners()[0].actionPerformed(e:null);
225     }
226
227     confirmPaymentButton.addActionListener(e -> {
228         String paymentMethod = (String) paymentMethodComboBox.getSelectedItem();
229         String courier = (String) courierComboBox.getSelectedItem();
230         String shippingAddress = addressArea.getText();
231         int userId = Main.currentUser != null ? Main.currentUser.getUserId() : -1;
232         String vaNumber = null;
233
234         if (userId == -1) {
235             JOptionPane.showMessageDialog(this, message:"Error: User tidak teridentifikasi. Silakan login ulang.", title:"Error",
236             JOptionPane.ERROR_MESSAGE);
237             return;
238         }
239
240         SimpleDateFormat formatter = new SimpleDateFormat(pattern:"dd/MM/yyyy HH:mm:ss");
241         Date date = new Date();
242         String orderDate = formatter.format(date);
243         String orderId = "ORD" + System.currentTimeMillis() / 1000;
244
245         if ("Virtual Account".equals(paymentMethod)) {
246             if (Main.currentUser == null || Main.currentUser.getPhoneNumber() == null || Main.currentUser.getPhoneNumber().isEmpty())
247             {
248                 JOptionPane.showMessageDialog(this, message:"Nomor telepon pengguna tidak ditemukan untuk Virtual Account.", title:"Error",
249                 JOptionPane.ERROR_MESSAGE);
250             }
251             valNumber = "727" + Main.currentUser.getPhoneNumber();
252
253             StringBuilder receiptMessage = new StringBuilder();
254             receiptMessage.append(str:<html><body style='width: 350px; font-family: Monospace; font-size: 10pt; padding: 10px;'>);
255             receiptMessage.append(str:<center><b>--- Struk Pembayaran CyclePro ---</b></center><br>');
256             receiptMessage.append(str:"-----<br>");
257             receiptMessage.append(str:"No. Pesanan : ").append(orderId).append(str:<br>');
258             receiptMessage.append(str:"Tanggal : ").append(orderDate).append(str:<br>');
259             receiptMessage.append(str:"Pelanggan : ").append(Main.currentUser.getUsername()).append(str:<br>');
260             receiptMessage.append(str:"-----<br>");
261             receiptMessage.append(str:<b>Produk:</b><br>');
262             receiptMessage.append(String.format(format:"%30.3s Rp%,10.0f", selectedProduct.getName(), selectedProduct.getPrice()));
263             receiptMessage.append(str:<br>);
264             receiptMessage.append(str:"-----<br>");
265             receiptMessage.append(String.format(format:"%30.3s <b>Rp%,10.0f</b>", ...args:"Total Harga:", selectedProduct.getPrice()));
266             receiptMessage.append(str:<br>);
267             receiptMessage.append(str:"-----<br>");
268             receiptMessage.append(str:"Metode Bayar: ").append(paymentMethod).append(str:<br>);
269             receiptMessage.append(str:<b>No. Virtual Account: </b>").append(valNumber).append(str:<b></b><br>');
270             receiptMessage.append(str:"Kurir : ").append(courier).append(str:<br>);
271             receiptMessage.append(str:"Alamat : ").append(shippingAddress.replaceAll(regex:"\\n", replacement:<br>));
272             receiptMessage.append(str:<br>);
273             receiptMessage.append(str:"-----<br>");
274             receiptMessage.append(str:<br><center><b>Silakan segera selesaikan pembayaran Anda.</b></center><br>');
275             receiptMessage.append(str:"Terima kasih telah berbelanja di CyclePro!</center><br>");
276             receiptMessage.append(str:</body></html>');
277
278             DatabaseHelper.createOrder(userId, selectedProduct.getProductId(), shippingAddress, courier, paymentMethod,
279             selectedProduct.getPrice(), vaNumber);
280
281             JEditorPane editorPane = new JEditorPane(type:"text/html", receiptMessage.toString());
282             editorPane.setEditable(b:false);
283             editorPane.setBackground(Colors.BACKGROUND_SECONDARY);
284             JScrollPane receiptScrollPane = new JScrollPane(editorPane);
285             receiptScrollPane.setPreferredSize(new Dimension(width:480, height:400));
286
287             JOptionPane.showMessageDialog(this, receiptScrollPane, title:"Konfirmasi Pembayaran Virtual Account", JOptionPane.
INFORMATION_MESSAGE);
288
289         } else { // COD
290             DatabaseHelper.createOrder(userId, selectedProduct.getProductId(), shippingAddress, courier, paymentMethod,
291             selectedProduct.getPrice(), vaNumber:null);
292         }
293     });

```

```

287     StringBuider codReceiptMessage = new StringBuider();
288     codReceiptMessage.append(str:"<html><body style='width: 300px; font-family: Monospace; font-size: 10pt; padding:10px; >");
289     codReceiptMessage.append(str:"<center><b>--- Konfirmasi Pesanan CyclePro ---</b></center><br>");
290     codReceiptMessage.append(str:"-----<br>");
291     codReceiptMessage.append(str:"No. Pesanan: ").append(orderId).append(str:"<br>");
292     codReceiptMessage.append(str:"Tanggal : ").append(orderDate).append(str:"<br>");
293     codReceiptMessage.append(str:"Pelanggan : ").append(Main.currentUser.getUsername()).append(str:"<br>");
294     codReceiptMessage.append(str:"-----<br>");
295     codReceiptMessage.append(str:"<b>Produk:</b><br>");
296     codReceiptMessage.append(String.format(format:"%-20.20s Rp%,10.0f", selectedProduct.getName(), selectedProduct.getPrice()
297     ())),append(str:"<br>");
298     codReceiptMessage.append(str:"-----<br>");
299     codReceiptMessage.append(String.format(format:"%-20.20s <b>Rp%,10.0f</b>", ...args:"Total Bayar:", selectedProduct.
300     getPrice())).append(str:"<br>");
301     codReceiptMessage.append(str:"-----<br>");
302     codReceiptMessage.append(str:"Metode Bayar: ").append(paymentMethod).append(str:"<br>");
303     codReceiptMessage.append(str:"Kurir : ").append(courier).append(str:"<br>");
304     codReceiptMessage.append(str:"Alamat : ").append(shippingAddress.replaceAll(regex:"\\n",
305     replacement:"<br> ")).append(str:"<br>");
306     codReceiptMessage.append(str:"-----<br>");
307     codReceiptMessage.append(str:"<br><center><b>Pesanan Anda akan segera diproses.</b><br>");
308     codReceiptMessage.append(str:"<br><center><b>Mohon siapkan pembayaran saat kurir tiba.</b><br>");
309     codReceiptMessage.append(str:"</body></html>");
310
311     JEditorPane editorPaneCod = new JEditorPane(type:"text/html", codReceiptMessage.toString());
312     editorPaneCod.setEditable(b:false);
313     editorPaneCod.setBackground(Colors.BACKGROUND_SECONDARY);
314     JScrollPane scrollPaneCod = new JScrollPane(editorPaneCod);
315     scrollPaneCod.setPreferredSize(new Dimension(width:430, height:350));
316
317     JOptionPane.showMessageDialog(this, scrollPaneCod, title:"Konfirmasi Pesanan COD", JOptionPane.INFORMATION_MESSAGE);
318 }
319 new Dashboard().setVisible(b:true);
320 dispose();
321 });
322 });
323 });
324 );
325 );

```

## Penjelasan Kode Program

Kelas HalamanPembayaran.java adalah antarmuka pengguna (GUI) yang berfungsi sebagai halaman *checkout*. Halaman ini ditampilkan setelah pengguna memilih sebuah produk dan bertanggung jawab untuk mengumpulkan informasi pengiriman dan pembayaran, serta menyelesaikan transaksi.

- Mendeklarasikan public class HalamanPembayaran extends JFrame, yang berarti kelas ini adalah sebuah jendela utama yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat atribut private Produk selectedProduct untuk menyimpan objek produk yang dipilih dari halaman sebelumnya. Data dari objek ini digunakan untuk menampilkan detail seperti nama, harga, deskripsi, dan gambar.
- Konstruktor public HalamanPembayaran(Produk product) menerima objek Produk sebagai parameter. Ini adalah mekanisme penting untuk mentransfer data antar jendela.
- Layout utama dibagi menjadi beberapa bagian, termasuk banner atas, bar info produk yang menonjol, panel gambar produk di sisi kanan, dan panel form detail di sisi kiri.
- Panel form (detailsFormPanel) menggunakan setLayout(null) untuk penempatan

komponen secara absolut. Panel ini berisi:

1. Area teks untuk deskripsi produk (tidak dapat diedit).
  2. JComboBox untuk memilih metode pembayaran ("Virtual Account" atau "COD").
  3. JComboBox untuk memilih kurir pengiriman.
  4. Area teks yang menampilkan alamat pengiriman pengguna, yang diambil secara otomatis dari Main.currentUser.
- Terdapat ActionListener pada paymentMethodComboBox. Jika "Virtual Account" dipilih, sebuah nomor VA akan dibuat secara dinamis menggunakan nomor telepon pengguna dan ditampilkan di label vaInfoLabel.
  - Terdapat ActionListener pada tombol "BELI SEKARANG" (confirmPaymentButton) yang menjalankan logika inti transaksi:
    1. Mengambil semua data yang dipilih (metode bayar, kurir) dan data pengguna yang sedang login.
    2. Membuat ID Pesanan dan tanggal pesanan secara dinamis.
    3. Terdapat logika kondisional: jika metode bayar adalah "Virtual Account", ia akan membuat nomor VA.
    4. Memanggil metode DatabaseHelper.createOrder() untuk menyimpan pesanan ke database.
    5. Membuat "struk" pembayaran dalam format HTML menggunakan StringBuilder dan menampilkannya di dalam JOptionPane menggunakan komponen JEditorPane. Ini memberikan umpan balik yang jelas kepada pengguna bahwa pesanan telah berhasil dibuat.
    6. Setelah transaksi selesai, jendela pembayaran ditutup dan pengguna diarahkan kembali ke Dashboard.
  - Terdapat ActionListener pada tombol "Batal" (cancelButton) yang akan menutup jendela saat ini dan mengembalikan pengguna ke halaman KategoriSepeda.

### Penjelasan Konsep OOP

- Menggunakan konsep Pewarisan (*Inheritance*)
  - HalamanPembayaran adalah turunan dari JFrame, mewarisi semua kapabilitas sebuah jendela aplikasi.
- Menggunakan konsep Enkapsulasi
  - Atribut-atribut seperti selectedProduct dan komponen Swing lainnya

- dideklarasikan sebagai private untuk membatasi aksesnya.
- Logika yang sangat kompleks untuk memproses pembayaran dan membuat struk dienkapsulasi sepenuhnya di dalam ActionListener dari tombol "BELI SEKARANG", menyembunyikan detail implementasi dari bagian lain kelas.
  - Menggunakan konsep Abstraksi
    - Kelas ini berinteraksi dengan DatabaseHelper melalui metode createOrder(). HalamanPembayaran tidak perlu tahu detail query SQL INSERT yang dijalankan; ia hanya perlu tahu bahwa metode tersebut akan menyimpan pesanan.
    - Penggunaan objek selectedProduct juga merupakan abstraksi. Kelas ini hanya memanggil metode seperti selectedProduct.getName() tanpa perlu tahu bagaimana data nama tersebut disimpan di dalam objek Produk.
  - Menggunakan konsep Polimorfisme
    - Penggunaan ActionListener sebagai kelas anonim adalah contoh dari polimorfisme, di mana kita menyediakan implementasi spesifik untuk metode actionPerformed sesuai dengan kebutuhan setiap tombol.

## 14. HalamanRiwayatPesanan.java

```

1  import javax.swing.*;
2  import javax.swing.table.DefaultTableModel;
3  import javax.swing.border.EmptyBorder;
4  import java.awt.*;
5  import java.util.List;
6  import java.awt.event.ActionEvent;
7  import java.awt.event.ActionListener;
8  import java.awt.event.MouseAdapter;
9  import java.awt.event.MouseEvent;
10
11 public class HalamanRiwayatPesanan extends JFrame {
12
13     private DefaultTableModel tableModel;
14     private JTable historyTable;
15
16     public HalamanRiwayatPesanan() {
17         setTitle("Riwayat Pesanan Anda");
18         setSize(width:900, height:600);
19         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
20         setLocationRelativeTo(null);
21         setLayout(new BorderLayout(hgap:10, vgap:10));
22
23         getContentPane().setBackground(Colors.BACKGROUND_PRIMARY);
24
25         // --- Panel Atas (Header) ---
26         JPanel headerPanel = new JPanel(new BorderLayout(hgap:10, vgap:0));
27         headerPanel.setBackground(Colors.NAVBAR_BACKGROUND);
28         headerPanel.setBorder(new EmptyBorder(top:10, left:15, bottom:10, right:15));
29
30         JLabel titleLabel = new JLabel(text:"Riwayat Pesanan Anda");
31         titleLabel.setFont(new Font(name:"Arial", Font.BOLD, size:24));
32         titleLabel.setForeground(Colors.NAVBAR_TEXT);
33         headerPanel.add(titleLabel, BorderLayout.WEST);
34

```

```

35     JButton backButton = new JButton(text:"Kembali ke Dashboard");
36     styleButton(backButton, Colors.BUTTON_SECONDARY_BACKGROUND, Colors.BUTTON_SECONDARY_TEXT);
37     backButton.addActionListener(new ActionListener() {
38         @Override
39         public void actionPerformed(ActionEvent e) {
40             new Dashboard().setVisible(true);
41             dispose();
42         }
43     });
44     headerPanel.add(backButton, BorderLayout.EAST);
45
46     add(headerPanel, BorderLayout.NORTH);
47
48 // --- Panel Tengah (Tabel Riwayat Pesanan) ---
49 JPanel mainContentPanel = new JPanel(new BorderLayout());
50 mainContentPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));
51 mainContentPanel.setBackground(Colors.BACKGROUND_PRIMARY);
52
53     tableModel = new DefaultTableModel(new Object[]{"ID Pesanan", "Produk", "Tanggal", "Total Harga", "Status", "Kurir", "Alamat Pengiriman"}, 0) {
54         @Override
55         public boolean isCellEditable(int row, int column) {
56             return false;
57         }
58     };
59     historyTable = new JTable(tableModel);
60     historyTable.setFont(new Font("Arial", Font.PLAIN, size:12));
61     historyTable.setRowHeight(rowHeight:25);
62     historyTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, size:12));
63     historyTable.getTableHeader().setBackground(Colors.BUTTON_PRIMARY_BACKGROUND);
64     historyTable.getTableHeader().setForeground(Colors.BUTTON_PRIMARY_TEXT);
65     historyTable.setSelectionBackground(Colors.BUTTON_PRIMARY_BACKGROUND.brighter());
66     historyTable.setSelectionForeground(Color.WHITE);
67
68     JScrollPane scrollPane = new JScrollPane(historyTable);
69     scrollPane.setBorder(BorderFactory.createLineBorder(Colors.BORDER_COLOR, thickness:1));
70     mainContentPanel.add(scrollPane, BorderLayout.CENTER);
71
72     add(mainContentPanel, BorderLayout.CENTER);
73
74     loadOrderHistory();
75 }
76
77 private void loadOrderHistory() {
78     tableModel.setRowCount(0);
79     if (Main.currentUser != null) {
80         List<Pesanan> userOrders = DatabaseHelper.getOrdersByUserId(Main.currentUser.getUserId());
81
82         if (userOrders.isEmpty()) {
83             System.out.println("Tidak ada riwayat pesanan ditemukan untuk user ini.");
84
85         } else {
86             for (Pesanan order : userOrders) {
87                 String productName = (order.getProduct() != null) ? order.getProduct().getName() : "N/A";
88                 tableModel.addRow(new Object[] {
89                     order.getId(),
90                     productName,
91                     order.getDate(),
92                     String.format(format:"Rp %,.0f", order.getTotalPrice()),
93                     order.getStatus(),
94                     order.getCourier(),
95                     order.getShippingAddress()
96                 });
97             }
98         }
99     } else {
100         JOptionPane.showMessageDialog(this, message:"Anda harus login untuk melihat riwayat pesanan.", title:"Error", JOptionPane.ERROR_MESSAGE);
101         System.out.println("Tidak ada user yang sedang login untuk melihat riwayat pesanan.");
102     }
103 }
104

```

```

105     private void styleButton(JButton button, Color bgColor, Color textColor) {
106         button.setBackground(bgColor);
107         button.setForeground(textColor);
108         button.setFont(new Font("Arial", Font.BOLD, size:14));
109         button.setFocusPainted(b:false);
110         button.setBorder(BorderFactory.createEmptyBorder(top:8, left:15, bottom:8, right:15));
111         button.setCursor(new Cursor(Cursor.HAND_CURSOR));
112         button.addMouseListener(new MouseAdapter() {
113             @Override
114             public void mouseEntered(MouseEvent e) {
115                 button.setBackground(bgColor.darker());
116             }
117
118             @Override
119             public void mouseExited(MouseEvent e) {
120                 button.setBackground(bgColor);
121             }
122         });
123     }
124
125     public static void main(String[] args) {
126         DatabaseHelper.createNewDatabase();
127         DatabaseHelper.createTables();
128         DatabaseHelper.addDefaultAdmin();
129
130         SwingUtilities.invokeLater(() -> {
131             if (Main.currentUser != null) {
132                 new HalamanRiwayatPesanan().setVisible(b:true);
133             } else {
134                 JOptionPane.showMessageDialog(parentComponent:null, message:"Login sebagai user untuk melihat riwayat pesanan.", title:"Info", JOptionPane.INFORMATION_MESSAGE);
135                 new HalamanLogin().setVisible(b:true);
136             }
137         });
138     }
139 
```

## Penjelasan Kode Program

Kelas HalamanRiwayatPesanan.java adalah antarmuka pengguna (GUI) yang berfungsi untuk menampilkan daftar riwayat transaksi atau pesanan yang telah dilakukan oleh pengguna yang sedang login.

- Mendeklarasikan public class HalamanRiwayatPesanan extends JFrame, yang menandakan bahwa kelas ini adalah sebuah jendela utama yang mewarisi semua fungsionalitas dari kelas JFrame.
- Terdapat atribut private yaitu tableModel (sebuah DefaultTableModel) dan historyTable (sebuah JTable). Kombinasi ini digunakan untuk menampilkan data riwayat pesanan dalam format tabel yang terstruktur. DefaultTableModel juga di-override agar sel tabel tidak dapat diedit oleh pengguna.
- Layout utama menggunakan BorderLayout yang membagi jendela menjadi headerPanel di bagian atas dan mainContentPanel yang berisi tabel di bagian tengah.
- Tombol "Kembali ke Dashboard" pada *header* dilengkapi ActionListener untuk menutup jendela saat ini dan membuka kembali halaman Dashboard.
- Terdapat metode privat private void loadOrderHistory(). Metode ini adalah inti dari fungsionalitas halaman ini. Ia bertugas untuk:
  1. Memastikan ada pengguna yang sedang login (Main.currentUser != null).
  2. Mengosongkan semua baris tabel yang ada saat ini

(tableModel.setRowCount(0)).

3. Memanggil metode DatabaseHelper.getOrdersById() dengan ID pengguna yang sedang aktif untuk mengambil data riwayat yang relevan dari database.
  4. Melakukan *looping* pada setiap objek Pesanan yang diterima dan menambahkannya sebagai baris baru ke dalam tableModel.
- Terdapat metode *helper* private void styleButton(...) yang digunakan untuk memberikan gaya visual yang konsisten pada tombol, menunjukkan penerapan prinsip DRY (*Don't Repeat Yourself*).
  - Kelas ini juga memiliki metode main sendiri yang berfungsi untuk tujuan pengujian (*testing*), memungkinkan developer untuk menjalankan jendela ini secara terpisah.

#### Penjelasan Konsep OOP

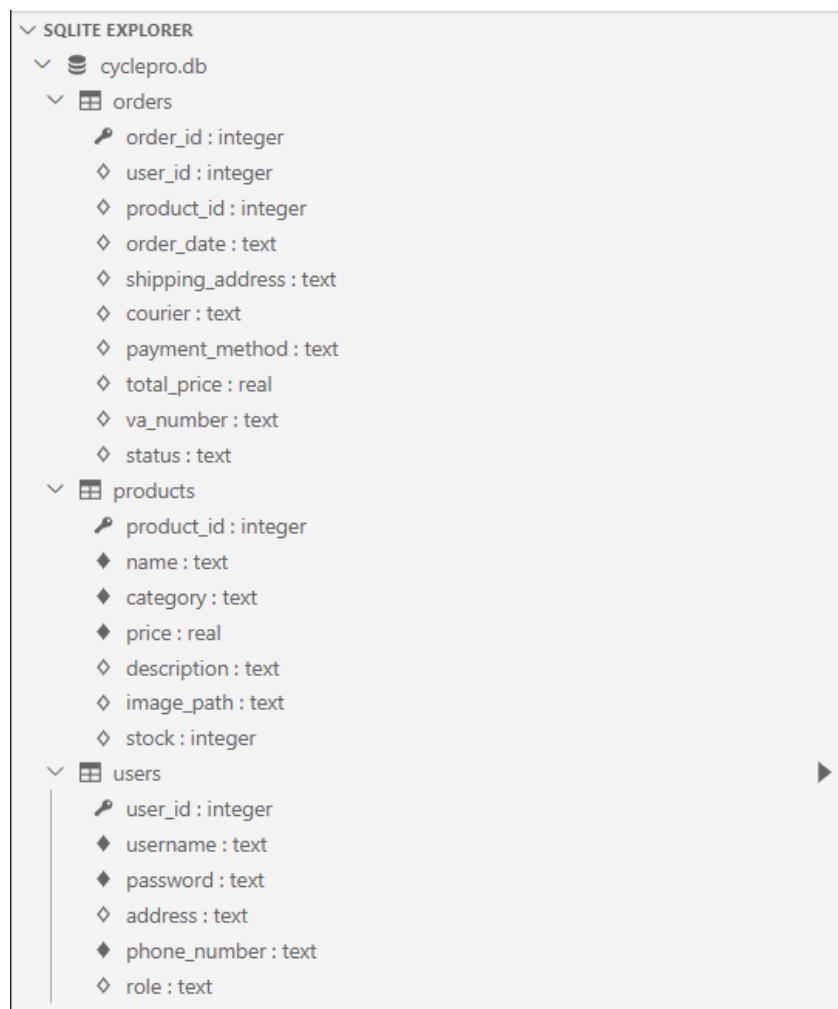
- Menggunakan konsep Pewarisan (*Inheritance*)
  - HalamanRiwayatPesanan adalah turunan dari JFrame, sehingga ia mewarisi semua sifat dan perilaku sebuah jendela aplikasi.
- Menggunakan konsep Enkapsulasi
  - Atribut-atribut tableModel dan historyTable dideklarasikan sebagai private, membatasi akses langsung dari luar kelas.
  - Logika untuk mengambil dan menampilkan data dienkapsulasi sepenuhnya di dalam metode privat loadOrderHistory(), sehingga membuat kode di konstruktur lebih rapi dan terfokus pada penyusunan komponen.
- Menggunakan konsep Abstraksi
  - Kelas ini berinteraksi dengan DatabaseHelper melalui metode getOrdersById(). HalamanRiwayatPesanan tidak perlu tahu detail tentang query SQL JOIN atau cara kerja koneksi database; ia hanya perlu menerima hasilnya berupa List<Pesanan>.
- Menggunakan konsep Polimorfisme
  - Penggunaan ActionListener dan MouseAdapter sebagai kelas anonim adalah bentuk polimorfisme, di mana kita menyediakan implementasi spesifik untuk metode actionPerformed atau mouseEntered/mouseExited.
  - Peng-override-an metode isCellEditable pada DefaultTableModel juga merupakan contoh polimorfisme, di mana kita mengubah perilaku standar dari kelas induknya untuk mencegah pengeditan sel.



## 3.2 Implementasi Pengujian

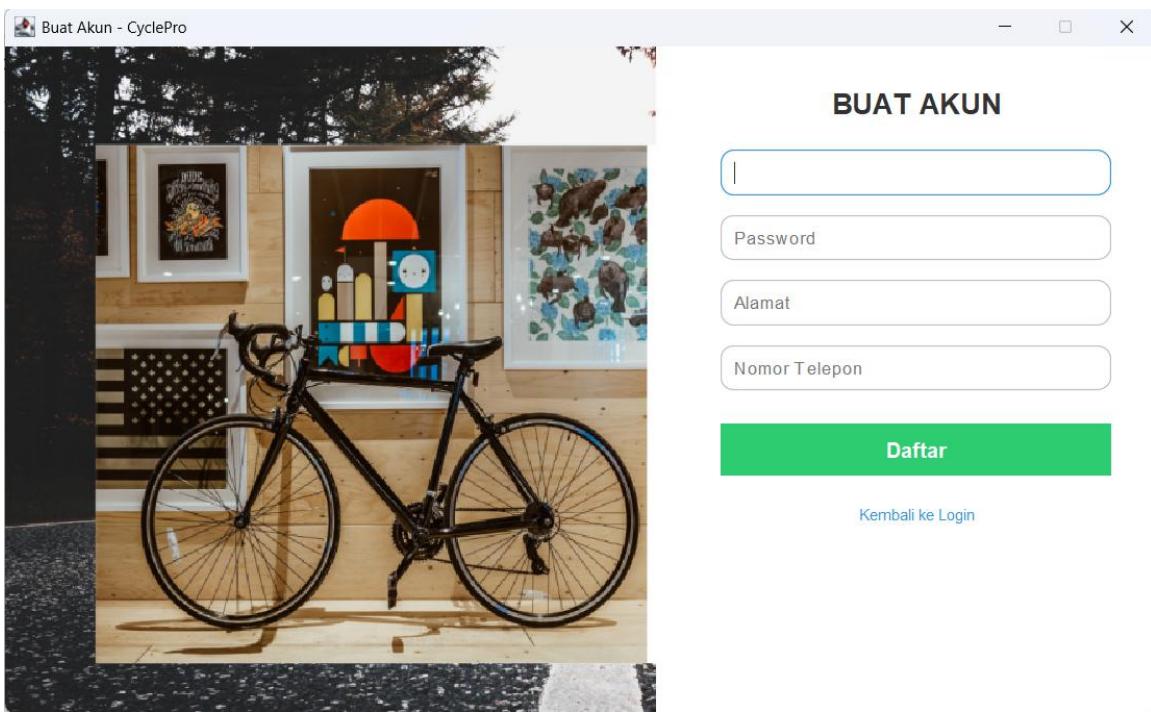
### 1. Koneksi Database

Koneksi database pada aplikasi CyclePro dikelola secara terpusat dan efisien melalui kelas utilitas DatabaseHelper.java, yang menggunakan teknologi database berbasis file SQLite dan dihubungkan melalui JDBC. Pendekatan ini mengenkapsulasi seluruh logika interaksi data, mulai dari pembuatan koneksi menggunakan string URL "jdbc:sqlite:cyclepro.db" hingga eksekusi query. Aspek paling krusial dari implementasi ini adalah penggunaan blok try-with-resources di setiap metode, yang secara otomatis memastikan bahwa setiap sesi koneksi ditutup dengan aman setelah operasi selesai. Hal ini menciptakan alur kerja yang efisien, di mana setiap operasi database (seperti otentikasi atau pengambilan data) mendapatkan koneksi baru yang langsung ditutup setelah digunakan, sehingga mencegah kebocoran sumber daya.



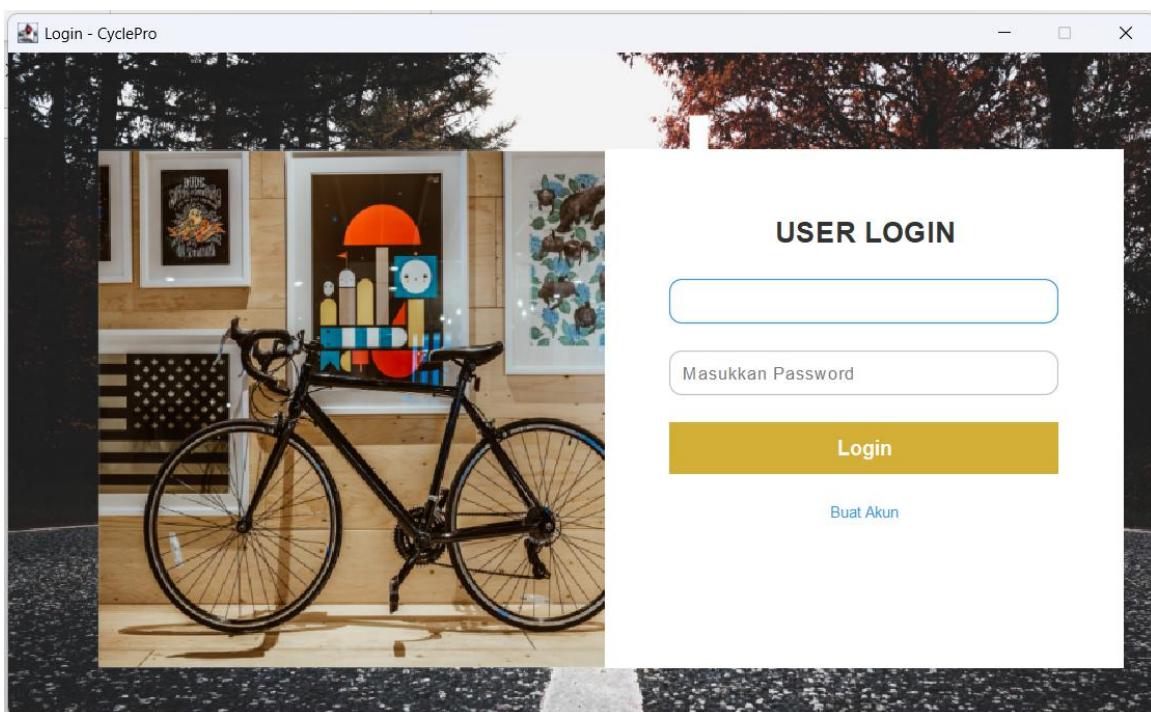
```
SQLite EXPLORER
cyclepro.db
  orders
    order_id : integer
    user_id : integer
    product_id : integer
    order_date : text
    shipping_address : text
    courier : text
    payment_method : text
    total_price : real
    va_number : text
    status : text
  products
    product_id : integer
    name : text
    category : text
    price : real
    description : text
    image_path : text
    stock : integer
  users
    user_id : integer
    username : text
    password : text
    address : text
    phone_number : text
    role : text
```

## 2. Register



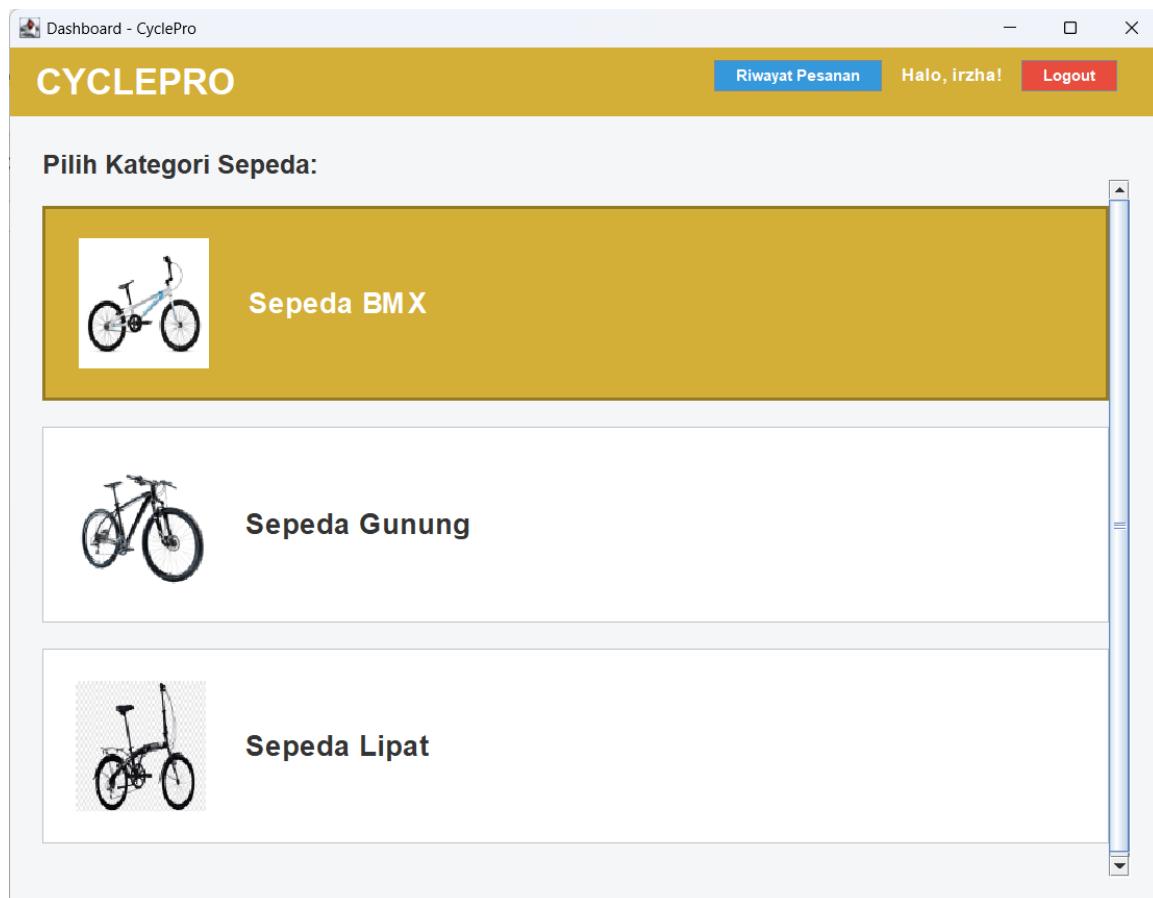
Calon pengguna diwajibkan untuk mengisi formulir yang terdiri dari data-data esensial seperti username, password, alamat, dan nomor telepon. Setelah semua data terisi dan tombol "Daftar" ditekan, informasi tersebut akan divalidasi dan disimpan ke dalam tabel users di database. Proses ini merupakan langkah awal untuk membuat akun pribadi di dalam sistem CyclePro.

## 3. Login



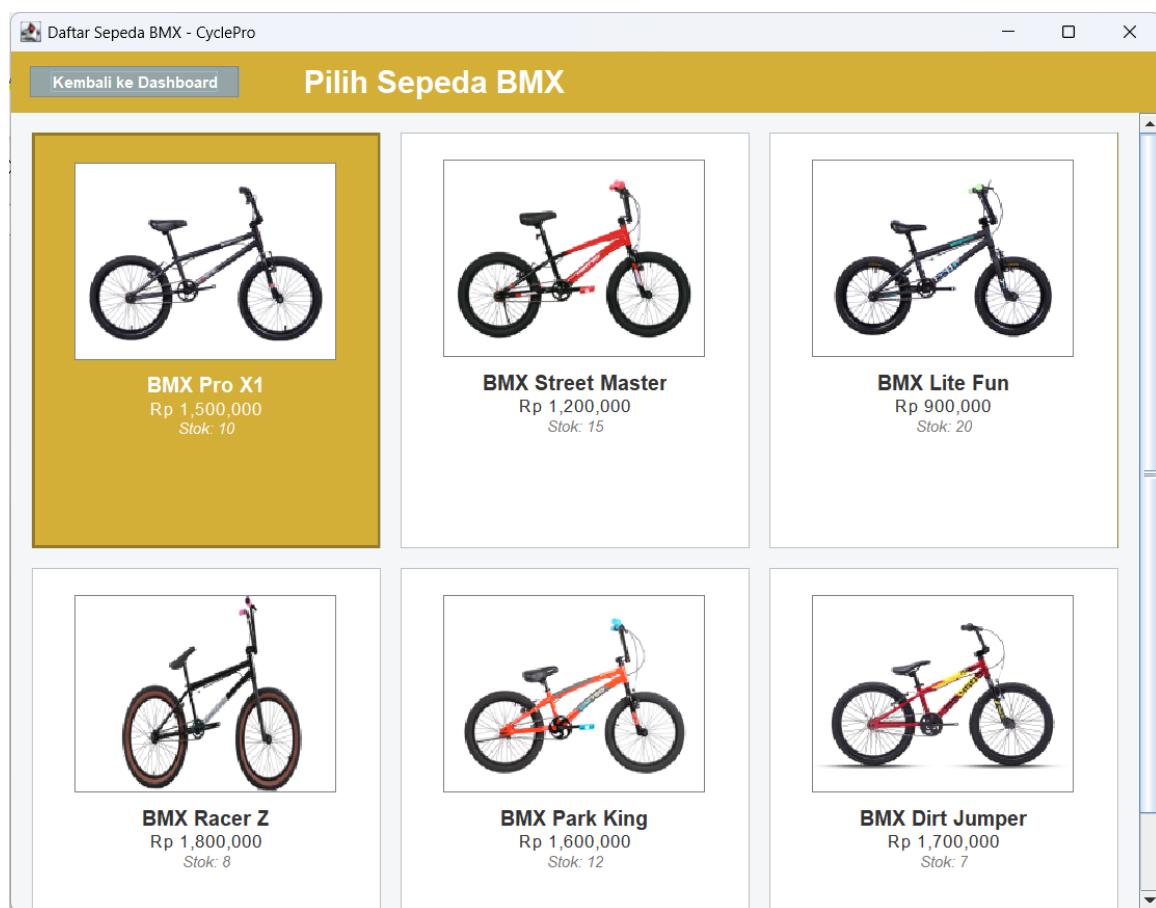
Setelah memiliki akun, pengguna dapat masuk melalui halaman Login. Pengguna memasukkan username dan password yang telah terdaftar. Sistem kemudian akan memvalidasi kredensial ini dengan data yang ada di database. Halaman ini berfungsi sebagai gerbang utama yang membedakan akses antara pengguna biasa dan administrator, yang akan diarahkan ke dasbornya masing-masing setelah login berhasil.

#### 4. Dashboard > Memilih kategori sepeda



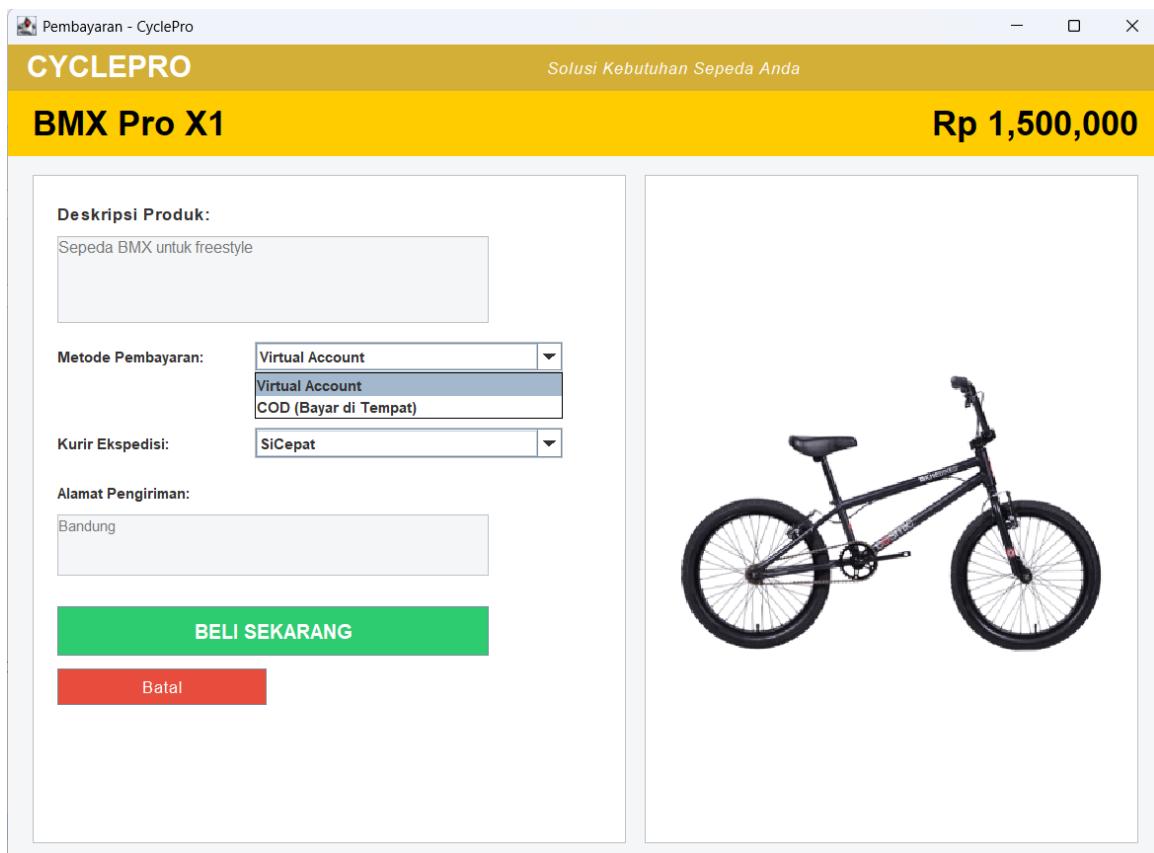
Setelah berhasil login, pengguna akan disambut di halaman Dashboard utama. Halaman ini secara dinamis menampilkan nama pengguna yang sedang aktif dan menyajikan menu utama dalam bentuk kartu-kartu visual yang interaktif. Setiap kartu merepresentasikan satu kategori sepeda yang tersedia, seperti "Sepeda BMX", "Sepeda Gunung", atau "Sepeda Lipat", mengundang pengguna untuk memulai proses penjelajahan produk.

## 5. Memilih Tipe Sepeda



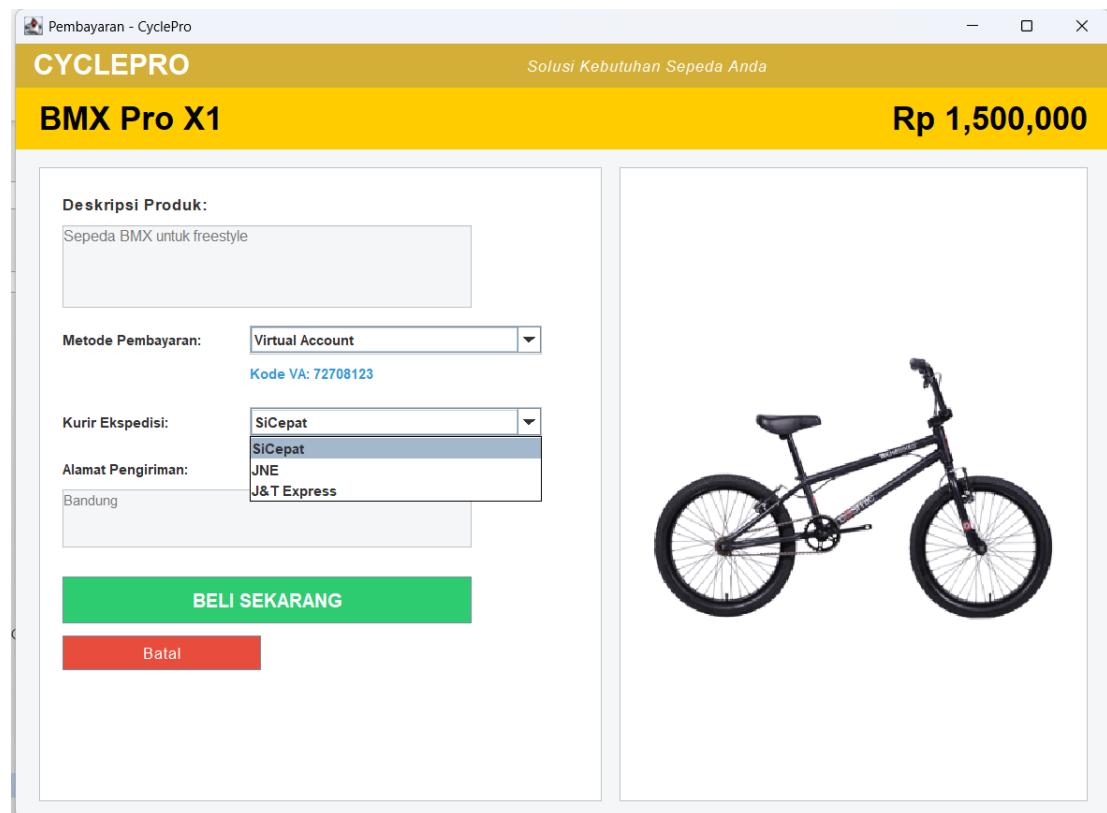
Ketika pengguna mengklik salah satu kategori di Dashboard, aplikasi akan beralih ke halaman Kategori Sepeda. Halaman ini menampilkan galeri produk yang lebih spesifik sesuai dengan kategori yang dipilih. Setiap produk ditampilkan dalam format kartu yang berisi gambar, nama, harga, dan informasi stok. Pengguna dapat melihat berbagai pilihan tipe sepeda dan memilih salah satu yang paling diminati untuk melanjutkan ke proses pembelian.

## 6. Pembayaran > Memilih metode pembayaran



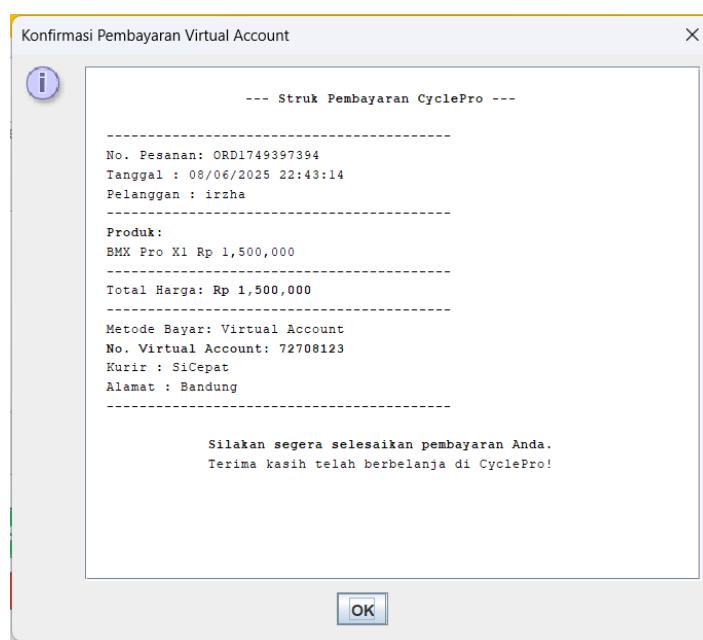
Setelah memilih produk, pengguna diarahkan ke halaman Pembayaran. Pada tahap ini, salah satu langkah pertama adalah memilih metode pembayaran yang diinginkan dari sebuah menu *dropdown*. Pilihan yang tersedia adalah "Virtual Account", yang akan menghasilkan nomor VA unik, dan "COD (Bayar di Tempat)" bagi pengguna yang ingin membayar secara tunai saat barang diterima.

## 7. Pembayaran > Memilih kurir ekspedisi



Masih di halaman Pembayaran, setelah memilih metode pembayaran, pengguna melanjutkan dengan memilih layanan kurir ekspedisi dari menu *dropdown* yang tersedia. Opsi seperti "SiCepat", "JNE", atau "J&T Express" dapat dipilih untuk menentukan jasa pengiriman yang akan digunakan untuk mengantar produk ke alamat pengguna yang sudah terisi secara otomatis.

## 8. Konfirmasi Pesanan > Cetak struk



Setelah semua detail pembayaran dan pengiriman diisi dan tombol "BELI SEKARANG" ditekan, aplikasi akan memproses transaksi. Sebagai konfirmasi, sebuah jendela dialog akan muncul menampilkan struk digital. Struk ini berisi semua detail pesanan seperti nomor order, tanggal, nama produk, total harga, metode pembayaran, dan alamat pengiriman dalam format yang rapi dan mudah dibaca, menandakan bahwa pesanan telah berhasil dibuat.

## 9. Riwayat Pesanan

ID Pesanan	Produk	Tanggal	Total Harga	Status	Kurir	Alamat Pengiriman
2	Gunung Forest Rider	2025-06-08 15:31:35	Rp 2,500,000	Diproses	SiCepat	Bandung
3	BMX Pro X1	2025-06-08 15:31:50	Rp 1,500,000	Diproses	SiCepat	Bandung
4	BMX Pro X1	2025-06-08 15:43:14	Rp 1,500,000	Diproses	SiCepat	Bandung

Pengguna dapat jedazeit meninjau kembali semua transaksi yang pernah mereka lakukan dengan mengakses halaman Riwayat Pesanan dari Dashboard. Halaman ini menyajikan data dalam bentuk tabel yang terstruktur, menampilkan informasi penting dari setiap pesanan seperti ID, nama produk, tanggal, total harga, dan status terakhir dari pesanan tersebut, memberikan transparansi penuh kepada pengguna.

## 10. Admin Dashboard

The screenshot shows a Windows application window titled "Admin Dashboard - Kelola Pesanan". The main title bar is yellow with the text "Data Pesanan" and "Admin: admin". Below the title bar is a table with the following columns: ID Pesanan, User, Produk, Tanggal, Total, Status, Alamat Pengir..., Kurir, Metode Bayar, and VA Number. There are three rows of data:

ID Pesanan	User	Produk	Tanggal	Total	Status	Alamat Pengir...	Kurir	Metode Bayar	VA Number
2	irzha	Gunung Forest ...	2025-06-08 15:3...	Rp 2,500,000	Diproses	Bandung	SiCepat	Virtual Account	72708123
3	irzha	BMX Pro X1	2025-06-08 15:3...	Rp 1,500,000	Diproses	Bandung	SiCepat	COD (Bayar di ...)	-
4	irzha	BMX Pro X1	2025-06-08 15:4...	Rp 1,500,000	Diproses	Bandung	SiCepat	Virtual Account	72708123

At the bottom of the dashboard are four buttons: "Refresh Data" (blue), "Ubah Status Pesanan" (green), "Hapus Pesanan" (red), and "Logout Admin" (grey).

Tampilan utamanya adalah sebuah tabel komprehensif yang menampilkan seluruh data pesanan dari semua pengguna. Dasbor ini berfungsi sebagai pusat kontrol bagi admin untuk memantau semua aktivitas transaksi yang terjadi di dalam aplikasi CyclePro.

## 11. Kelola Pesanan > Ubah status pesanan

The screenshot shows the same "Admin Dashboard - Kelola Pesanan" window. A modal dialog box titled "Ubah Status Pesanan" is open over the order list. The dialog has a question mark icon and a message: "Pilih status baru untuk Pesanan ID: 2 (Status saat ini: Diproses)". It contains a dropdown menu with the following options: Diproses, Pending, Dikirim, Selesai, and Dibatalkan. The "Diproses" option is currently selected.

At the bottom of the dashboard are four buttons: "Refresh Data" (blue), "Ubah Status Pesanan" (green), "Hapus Pesanan" (red), and "Logout Admin" (grey).

Dari Admin Dashboard, administrator memiliki kewenangan untuk mengelola

pesanannya. Dengan memilih salah satu baris pesanan dari tabel dan mengklik tombol "Ubah Status Pesanan", sebuah jendela dialog akan muncul. Dialog ini memungkinkan admin untuk mengubah status pesanan secara manual, misalnya dari "Diproses" menjadi "Dikirim", yang kemudian akan diperbarui di database dan dapat dilihat oleh pengguna di halaman riwayat pesanannya.

## 12. Kelola Pesanan > Hapus pesanan

The screenshot shows the Admin Dashboard with the title "Admin Dashboard - Kelola Pesanan". The main area displays a table titled "Data Pesanan" with the following data:

ID Pesanan	User	Produk	Tanggal	Total	Status	Alamat Pengiriman	Kurir	Metode Bayar	VA Number
2	irzha	Gunung Forest ...	2025-06-08 15:3...	Rp 2,500,000	Dikirim	Bandung	SiCepat	Virtual Account	72708123
3	irzha	BMX Pro X1	2025-06-08 15:3...	Rp 1,500,000	Diproses	Bandung	SiCepat	COD (Bayar di ...	-
4	irzha	BMX Pro X1	2025-06-08 15:4...	Rp 1,500,000	Diproses	Bandung	SiCepat	Virtual Account	72708123

A modal dialog box titled "Konfirmasi Hapus Pesanan" is displayed, containing the message "Yakin ingin menghapus pesanan ID: 4 dari irzha untuk produk BMX Pro X1?". It has "Yes" and "No" buttons.

At the bottom of the dashboard, there are buttons for "Refresh Data", "Ubah Status Pesanan" (highlighted in green), "Hapus Pesanan" (highlighted in red), and "Logout Admin".

Selain mengubah status, admin juga dapat menghapus data pesanan. Setelah memilih pesanan yang ingin dihapus, admin akan menekan tombol "Hapus Pesanan". Untuk mencegah kesalahan, sebuah jendela dialog konfirmasi akan muncul, menanyakan apakah admin yakin untuk melanjutkan tindakan tersebut. Jika "Yes" dipilih, data pesanan tersebut akan dihapus secara permanen dari database.

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Proyek aplikasi desktop "CyclePro" telah berhasil diimplementasikan sebagai sebuah sistem simulasi *e-commerce* untuk penjualan sepeda. Aplikasi ini dibangun menggunakan bahasa pemrograman Java dengan pustaka Swing untuk antarmuka pengguna (GUI) dan SQLite untuk manajemen database. Secara fungsional, aplikasi ini berhasil mencakup dua alur kerja utama: alur untuk pengguna (pembeli) dan alur untuk administrator. Pengguna dapat melakukan registrasi, login, melihat produk per kategori, melakukan pemesanan dengan memilih metode pembayaran dan kurir, hingga melihat riwayat transaksinya. Di sisi lain, administrator memiliki dasbor khusus untuk memantau dan mengelola semua pesanan yang masuk, termasuk mengubah status dan menghapus pesanan.

Selama pengembangan, beberapa konsep fundamental Pemrograman Berorientasi Objek (OOP) telah berhasil diterapkan secara efektif:

- Enkapsulasi: Diterapkan secara fundamental melalui penggunaan hak akses private pada atribut-atribut di kelas model (User, Admin, Produk, Pesanan) dan penyediaan metode public (*getter* dan *setter*) sebagai jalur akses yang aman dan terkontrol.
- Pewarisan (*Inheritance*): Digunakan dalam struktur kelas, di mana kelas Admin merupakan turunan dari kelas User, mewarisi semua propertinya. Selain itu, semua kelas antarmuka (HalamanLogin, Dashboard, dll.) merupakan turunan dari kelas JFrame Swing.
- Polimorfisme: Terwujud dalam penanganan *event* melalui penggunaan ActionListener dan MouseAdapter, di mana metode actionPerformed atau mouseClicked di-*override* untuk memberikan perilaku yang spesifik pada setiap komponen. *Method overriding* juga diterapkan pada metode toString() di kelas Produk untuk kustomisasi tampilan.
- Abstraksi: Kelas DatabaseHelper menjadi contoh utama abstraksi, di mana semua kompleksitas interaksi dengan database (query SQL, koneksi JDBC) disembunyikan dari kelas-kelas antarmuka, yang hanya perlu memanggil metode yang sudah disediakan.

Secara keseluruhan, proyek ini berhasil mencapai tujuannya untuk menciptakan aplikasi desktop yang fungsional dan terstruktur dengan baik, sekaligus menjadi implementasi praktis dari konsep-konsep inti Pemrograman Berorientasi Objek.