LAPORAN UJIAN AKHIR SEMESTER
PEMROGRAMAN BERORIENTASI OBJEK
"SISTEM PARKIR BERBASIS GUI JAVA"

Dosen Pengampu:

Taufik Ridwan, M.T.

Disusun oleh:

Kelompok 1 – 4A Sistem Informasi

1. Alifia Nur Huda : 2310631250005
2. Arya Cakra Ramadhan : 2310631250086
3. Daffa Abyan Hakim : 2310631250088
4. Yuuka Natasya Aji : 2310631250079

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan berkembangnya teknologi informasi, kebutuhan akan sistem yang dapat membantu aktivitas manusia dalam kehidupan sehari-hari menjadi semakin penting. Salah satu bidang yang mengalami transformasi signifikan adalah sistem pengelolaan parkir. Pengelolaan parkir secara manual sering kali menimbulkan berbagai permasalahan, seperti kesalahan pencatatan data kendaraan, waktu tunggu yang lama, serta ketidakefisienan dalam penghitungan tarif parkir. Oleh karena itu, diperlukan suatu sistem terkomputerisasi yang mampu mengelola data kendaraan secara otomatis, cepat, dan akurat.

Kampus sebagai institusi pendidikan tinggi yang memiliki mobilitas kendaraan tinggi, baik dari mahasiswa, dosen, staf, maupun pengunjung, menghadapi tantangan dalam pengelolaan area parkir yang efektif. Penggunaan sistem parkir manual di lingkungan kampus masih umum ditemukan, yang menyebabkan kurangnya akurasi data, kesulitan dalam pencarian riwayat kendaraan, serta tidak tersedianya bukti parkir dalam bentuk cetakan yang dapat dipertanggungjawabkan.

Sebagai solusi atas permasalahan tersebut, dirancang dan dibangun sebuah sistem parkir berbasis desktop menggunakan bahasa pemrograman Java dan database MySQL. Sistem ini dilengkapi dengan fitur pencatatan kendaraan masuk dan keluar, perhitungan otomatis biaya parkir berdasarkan durasi, serta pencetakan tiket dan struk dalam bentuk file PDF menggunakan pustaka iText. Sistem ini tidak hanya mendukung efisiensi operasional, tetapi juga meningkatkan profesionalisme dan akuntabilitas dalam pengelolaan parkir kampus.

Dengan adanya sistem ini, diharapkan pengelolaan parkir di lingkungan kampus UNSIKA dapat berjalan lebih tertib, terstruktur, dan modern. Selain itu, sistem ini juga dapat menjadi sarana pembelajaran dan implementasi nyata dari penguasaan teknologi informasi, khususnya dalam

pengembangan aplikasi berbasis Java dan penggunaan sistem manajemen basis data relasional.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam pembuatan sistem parkir ini adalah sebagai berikut:

1. Bagaimana merancang dan membangun sistem parkir berbasis desktop yang dapat mencatat data kendaraan masuk dan keluar secara otomatis dan akurat?
2. Bagaimana sistem dapat menghitung durasi parkir dan total biaya secara otomatis berdasarkan waktu masuk dan keluar kendaraan?
3. Bagaimana menghasilkan tiket parkir dan struk pembayaran dalam bentuk file PDF sebagai bukti transaksi yang sah?
4. Bagaimana sistem dapat mempermudah proses administrasi parkir di lingkungan kampus?

1.3 Tujuan Masalah

Tujuan dari penelitian dan pengembangan sistem parkir berbasis Java dan MySQL ini adalah:

1. Untuk merancang dan mengimplementasikan sistem parkir digital berbasis desktop yang dapat mencatat data kendaraan masuk dan keluar secara cepat dan tepat.
2. Untuk mengembangkan fitur perhitungan otomatis durasi parkir dan total biaya berdasarkan waktu kedatangan dan waktu keluar kendaraan.
3. Untuk menghasilkan tiket parkir dan struk pembayaran dalam bentuk file PDF sebagai dokumentasi dan bukti transaksi yang sah.
4. Untuk mempermudah petugas parkir dalam mengelola dan memantau aktivitas parkir kendaraan di lingkungan kampus.
5. Untuk meningkatkan efisiensi, akurasi, dan profesionalitas dalam pengelolaan sistem parkir kampus dengan memanfaatkan teknologi informasi.

BAB II
FITUR-FITUR APLIKASI

2.1 Login

Fitur login berfungsi sebagai gerbang awal akses ke dalam aplikasi sistem parkir. Melalui tampilan login, pengguna diminta untuk memasukkan username dan password yang telah ditentukan. Validasi dilakukan secara langsung, dan jika data yang dimasukkan sesuai, maka pengguna akan diarahkan menuju halaman Menu Utama. Sebaliknya, jika terdapat kesalahan dalam memasukkan username atau password, sistem akan memunculkan pop up bahwa data tidak valid. Saat ini, sistem login menggunakan data statis yaitu username “admin” dan password “1234”, namun dapat dikembangkan lebih lanjut agar terintegrasi dengan database untuk mendukung manajemen pengguna yang lebih kompleks.

2.2 Menu Utama

Setelah berhasil melakukan login, pengguna diarahkan ke tampilan Menu Utama. Pada halaman ini, disediakan tiga tombol utama yaitu “Parkir Masuk”, “Parkir Keluar”, dan “Logout”. Tombol “Parkir Masuk” digunakan untuk mencatat kendaraan yang masuk ke area parkir, sedangkan “Parkir Keluar” digunakan untuk mencatat data kendaraan yang keluar sekaligus menghitung biaya parkir. Tombol “Logout” digunakan untuk keluar dari sistem dan kembali ke halaman login. Tampilan Menu Utama dirancang secara sederhana dan responsif agar mudah digunakan oleh petugas parkir.

2.3 Parkir Masuk

Fitur ini digunakan untuk mencatat kendaraan yang baru saja memasuki area parkir. Petugas akan mengisi data seperti nomor tiket, nomor polisi kendaraan, dan jenis kendaraan. Sementara itu, tanggal dan jam masuk akan dicatat secara otomatis oleh sistem sesuai waktu saat penginputan. Setelah data diisi, informasi tersebut akan disimpan ke dalam database MySQL dan secara bersamaan sistem akan mencetak tiket parkir

dalam format PDF A6. Tiket ini berisi data kendaraan dan informasi waktu masuk, yang kemudian diberikan kepada pengguna kendaraan sebagai bukti parkir.

Selain mencatat data baru, tampilan Parkir Masuk juga dilengkapi dengan dua tombol tambahan yaitu Edit dan Hapus. Tombol Edit digunakan ketika petugas ingin memperbaiki data kendaraan yang sudah tercatat sebelumnya. Sebelum penghapusan dilakukan, sistem akan menampilkan pop-up konfirmasi terlebih dahulu guna menghindari kesalahan pengeditan. Setelah data diperbarui, sistem akan menampilkan pop-up notifikasi yang mengonfirmasi bahwa data berhasil diubah. Sementara itu, tombol Hapus digunakan untuk menghapus data kendaraan tertentu dari database yang juga menampilkan pop up konfirmasi penghapusan dan pop up berhasil dihapus. Dengan adanya fitur Edit dan Hapus ini, petugas parkir memiliki fleksibilitas dalam mengelola data kendaraan yang masuk dengan lebih akurat dan efisien.

2.4 Parkir Keluar

Fitur parkir keluar digunakan untuk mengelola data kendaraan yang keluar dari area parkir. Petugas akan memasukkan nomor tiket yang digunakan saat kendaraan masuk. Setelah itu, sistem akan mengambil data yang sesuai dari database dan menampilkan informasi kendaraan beserta waktu masuk. Selanjutnya, sistem akan mencatat waktu keluar secara otomatis, menghitung durasi parkir berdasarkan selisih waktu masuk dan keluar, serta menghitung total biaya yang harus dibayarkan. Setelah pembayaran dilakukan, sistem juga menghitung jumlah kembalian dan mencatat seluruh transaksi untuk dicetak dalam bentuk struk PDF.

2.5 Cetak Tiket Parkir (PDF)

Setiap kali kendaraan masuk ke area parkir, sistem akan secara otomatis mencetak tiket parkir sebagai bukti transaksi. Proses ini dilakukan oleh kelas CetakPdf.java yang menggunakan library iText untuk menghasilkan file PDF berukuran A6. Di dalam tiket tersebut tercantum

informasi penting seperti nomor tiket, nomor polisi kendaraan, jenis kendaraan, tanggal, dan jam masuk. Tiket ini menjadi acuan ketika kendaraan akan keluar dan sebagai identitas kendaraan selama berada di area parkir.

2.6 Cetak Struk Parkir (PDF)

Setelah kendaraan keluar dan proses pembayaran selesai dilakukan, sistem akan mencetak struk sebagai bukti pembayaran parkir. Fitur ini diatur oleh kelas CetakStruk.java, juga menggunakan library iText untuk mencetak file PDF. Struk yang dicetak mencakup informasi lengkap seperti nomor tiket, nomor kendaraan, jenis kendaraan, tanggal dan jam masuk, tanggal keluar, durasi parkir, total biaya, nominal pembayaran, dan kembalian. Struk ini dapat digunakan sebagai arsip atau bukti resmi bagi pengguna layanan parkir.

2.7 Koneksi Database

Aplikasi ini terhubung dengan database MySQL bernama parkir_db untuk menyimpan dan mengelola data parkir secara terstruktur. Koneksi ke database dikelola oleh kelas DatabaseConnection.java, yang menyederhanakan proses pengambilan dan penyimpanan data antara aplikasi dan server database. Dengan sistem ini, semua data kendaraan yang masuk dan keluar akan tercatat dengan baik, sehingga memudahkan proses rekapitulasi maupun pencarian data secara real-time.

BAB III
KONSEP OOP KODE PROGRAM

3.1 Object, Class, Method, Package

Dalam pemrograman berorientasi objek (OOP), class adalah blueprint atau cetakan yang mendefinisikan atribut (data) dan method (perilaku) dari suatu objek. Object merupakan instance atau perwujudan nyata dari sebuah class yang dapat digunakan dalam program. Method adalah fungsi yang didefinisikan di dalam class dan digunakan oleh objek untuk melakukan suatu aksi. Sementara itu, package adalah wadah atau namespace yang digunakan untuk mengelompokkan class-class agar lebih terorganisir, mencegah konflik nama, dan memudahkan pemeliharaan kode. Keempat konsep ini saling berkaitan dan menjadi dasar utama dalam pengembangan aplikasi berbasis OOP seperti Java.

1. File Login.java
    - Class: Mendefinisikan class Login, juga turunan dari JFrame, yang berisi tampilan form login serta validasi user.
    - Object: Objek JLabel, JTextField, JPasswordField, dan JButton digunakan untuk membangun GUI login. Objek koneksi database juga digunakan.
    - Method: Method loginUser() dan main() merupakan penerapan konsep fungsi/metode.
    - Package: Menggunakan javax.swing.* dan java.sql.* untuk komponen GUI dan koneksi database.
2. File ParkirMasuk.java
    - Class: File ini mendefinisikan sebuah class ParkirMasuk yang merupakan turunan dari JFrame, sehingga mewarisi semua properti dan metode GUI dari javax.swing.JFrame.
    - Object: Objek dari berbagai komponen GUI dideklarasikan, seperti JButton, JTextField, JTable, dan DefaultTableModel. Contoh: JButton btnTambah = new JButton("Tambah"); adalah objek dari class JButton.

- Method: Terdapat banyak method seperti tambahData(), editData(), hapusData(), muatDataDariDatabase(), dan buatNomorTiketOtomatis(). Ini menunjukkan penerapan metode dalam OOP untuk memisahkan logika fungsi program.
- Package: Program kemungkinan berada dalam package default atau dideklarasikan dengan package parkir; (jika disediakan). Library eksternal seperti javax.swing., java.sql., dan java.time.* di-import dari package standar Java.

3. File ParkirKeluar.java
    - Class: File ini berisi class ParkirKeluar yang juga merupakan turunan dari JFrame, memanfaatkan fitur GUI dari Java.
    - Object: Terdapat objek seperti JLabel, JButton, JTextField, dan objek koneksi database seperti Connection, PreparedStatement, dan ResultSet.
    - Method: Method seperti hitungDurasiParkir(), ambilDataMasuk(), dan prosesKeluar() adalah contoh implementasi fungsi berbasis OOP untuk modularitas program.
    - Package: Menggunakan package-package standar seperti java.sql.* dan javax.swing.*.
4. File CetakTiket.java
    - Class: Berisi class CetakTiket yang berisi method statis untuk mencetak tiket dalam bentuk PDF.
    - Object: Meskipun sebagian besar menggunakan method statis, di dalam method terdapat objek dari class lain seperti Document, PdfWriter, dan Paragraph dari library iText.
    - Method: Method public static void cetakTiket(...) adalah method statis yang bisa dipanggil langsung tanpa membuat objek.
    - Package: Menggunakan package eksternal iText (com.itextpdf.text.) dan standar seperti java.io..
5. File CetakStruk.java
    - Class: Mirip dengan CetakTiket, class CetakStruk merupakan class utilitas dengan method statis.

- Object: Objek yang digunakan seperti PdfWriter, Document, Paragraph, dan FileOutputStream digunakan untuk membuat dan menyimpan struk PDF.
- Method: Method public static void cetakStruk(...) adalah method utama dalam class ini.
- Package: Menggunakan package eksternal iText untuk PDF dan Java standard packages.

6. File DatabaseConnection.java
   - Class: Class DatabaseConnection berfungsi sebagai utilitas koneksi database. Ini adalah class dengan peran tunggal (single responsibility).
   - Object: Menggunakan objek Connection dari java.sql.
   - Method: Method public static Connection getConnection() mengembalikan objek koneksi database dan dapat digunakan oleh semua class lain.
   - Package: Menggunakan package java.sql.*.
7. File MenuUtamaParkir.java
   - Class: Class MenuUtamaParkir mewakili tampilan utama dari aplikasi dan merupakan turunan dari JFrame.
   - Object: Terdapat objek JButton yang merepresentasikan navigasi ke fitur-fitur lain seperti parkir masuk, parkir keluar, dan login.
   - Method: Biasanya hanya berisi constructor atau inisialisasi GUI serta method main() untuk menjalankan program.
   - Package: Memanfaatkan package standar Java GUI javax.swing.*.

3.2 Enkapsulasi

Enkapsulasi adalah konsep OOP yang menyembunyikan data internal suatu objek dan hanya memperbolehkan akses melalui method tertentu. Tujuannya adalah untuk melindungi data agar tidak diubah secara langsung dari luar class, serta menjaga keamanan dan integritas program.

1. File ParkirMasuk.java
    - Menggunakan private JTextField tfNomorTiket, tfNomorPolisi, tfTanggal untuk membatasi akses langsung ke inputan pengguna. Ini adalah bentuk enkapsulasi karena hanya method dalam class ini yang boleh memanipulasi nilai inputan.
    - Method tambahData(), editData(), dan hapusData() menangani proses logika CRUD, menjaga agar proses tersebut tidak diakses sembarangan dari luar class.
    - Akses koneksi database dilakukan melalui DatabaseConnection.getConnection() yang juga terenkapsulasi agar tidak dieksekusi langsung di luar class koneksi.
    - Penanganan aksi tombol menggunakan listener lambda, sehingga hanya method dalam class yang bisa merespons input pengguna — ini melindungi logika UI dari campur tangan luar.
2. File ParkirKeluar.java
    - Field seperti tfCariTiket, tfDurasi, dan tfBiaya dideklarasikan sebagai private, hanya dapat diakses dari method internal class tersebut.
    - Proses validasi dan penghitungan biaya dalam hitungDurasiDanBiaya() menjaga agar proses perhitungan hanya terjadi melalui method resmi, bukan dimanipulasi dari luar.
    - Method cariData(), hapusData(), dan hitungDurasiDanBiaya() menyembunyikan detail implementasi dari luar class. Ini memastikan pengguna class hanya menggunakan interface publik yang disediakan, bentuk nyata dari enkapsulasi.
3. File Login.java
    - Field tfUsername dan pfPassword diset private, agar hanya bisa diakses dan dimanipulasi dari dalam class itu sendiri.
    - Method cekLogin() mengenkapsulasi proses autentikasi. Semua logika koneksi dan validasi disimpan di method ini, melindungi bagian UI dari mengetahui detail proses verifikasi.

- Tombol login hanya memanggil cekLogin() tanpa tahu bagaimana cara login bekerja, bentuk pemisahan tanggung jawab dan perlindungan data internal.

4. File CetakTiket.java
   - Method cetakTiket(...) adalah static public, tapi parameter dan logika internal seperti pengaturan font dan format disimpan di dalam method tersebut — menjaga agar pengguna method hanya perlu tahu input-output, tanpa tahu proses detail pembuatan PDF.
   - Field dan method bantu di dalam class tidak dibuka ke luar selain method utama cetakTiket(), ini adalah bentuk enkapsulasi data dan operasi cetak PDF.
5. File CetakStruk.java
   - Hampir sama seperti CetakTiket, method cetakStruk(...) menyembunyikan detail pembuatan file PDF dari luar class. Komponen seperti pengaturan layout, font, dan isi PDF tidak bisa disentuh dari luar.
   - Hanya method cetakStruk(...) yang dibuka sebagai API publik — bentuk penerapan enkapsulasi yang jelas.
6. File DatabaseConnection.java
   - Menggunakan method static getConnection() sebagai satu-satunya cara mengakses koneksi database. Ini adalah contoh enkapsulasi resource, di mana objek Connection tidak dibuat di banyak tempat, tapi dikelola terpusat dan tersembunyi dari luar.
   - Tidak ada field atau method lain yang dibuka ke luar. Ini adalah class dengan tanggung jawab tunggal, penuh dengan prinsip enkapsulasi.
7. File MenuUtamaParkir.java
   - Semua komponen GUI seperti tombol disimpan sebagai field private.
   - Navigasi antar-form dilakukan melalui pemanggilan objek baru (new ParkirMasuk(), new ParkirKeluar(), dst), tanpa mengubah isi internal form lain.

- Aksi user seperti klik tombol tidak langsung memodifikasi objek lain, tapi hanya memicu alur kerja yang telah dienkapsulasi.

3.3 Inheritance

Inheritance (pewarisan) adalah konsep OOP di mana sebuah class (subclass) dapat mewarisi atribut dan method dari class lain (superclass), sehingga memungkinkan penggunaan kembali kode dan mempermudah pengembangan program yang lebih terstruktur dan efisien.

1. File ParkirMasuk.java
    - Kelas ParkirMasuk mewarisi dari JFrame, yaitu class dalam Java Swing. Ini adalah contoh penerapan inheritance, di mana ParkirMasuk mewarisi semua sifat dan method dari JFrame, seperti setVisible(), dispose(), dan lain-lain.
    - Dengan inheritance dari JFrame, kelas ini dapat berfungsi sebagai jendela utama GUI tanpa harus membuat ulang semua fungsi dasar jendela dari awal.
2. File ParkirKeluar.java
    - Sama seperti ParkirMasuk, class ParkirKeluar juga mewarisi dari JFrame, sehingga otomatis memiliki semua fungsi GUI window dari Swing.
    - Penerapan inheritance ini memungkinkan ParkirKeluar untuk memiliki method seperti setLayout(), setSize(), dan menangani event GUI tanpa harus menuliskannya ulang.
3. File Login.java
    - Kelas Login juga merupakan turunan dari JFrame, sehingga memiliki semua kemampuan GUI dasar dari class tersebut.
    - Dengan pewarisan ini, class Login dapat mengatur tampilan form login, menambahkan komponen GUI, dan menangani aksi pengguna menggunakan method dari superclass-nya.
4. File MenuUtamaParkir.java
    - Class ini juga mewarisi dari JFrame, artinya merupakan implementasi pewarisan dari class GUI Swing.

- Dengan inheritance ini, MenuUtamaParkir dapat berperilaku seperti jendela utama dan dapat menggunakan semua method GUI seperti add(), setDefaultCloseOperation(), dan sebagainya.

3.4 Polimorfisme

Polimorfisme adalah konsep OOP yang memungkinkan satu method, objek, atau class memiliki banyak bentuk atau perilaku berbeda, tergantung pada konteksnya. Ini memudahkan pengembangan program yang fleksibel dan mudah diperluas.

1. File ParkirMasuk.java
   - Pemanggilan addActionListener(e -> {...}) pada berbagai tombol merupakan contoh polimorfisme melalui anonymous class lambda expression. Objek ActionListener di-override dengan implementasi fungsional sesuai kebutuhan tombolnya.
   - Method setVisible(true) adalah contoh polimorfisme dinamis, karena objek ParkirMasuk mewarisi dari JFrame dan memanggil method bawaan dari superclass-nya.
2. File ParkirKeluar.java
   - Sama seperti ParkirMasuk, addActionListener(...) juga menggunakan polimorfisme dalam bentuk lambda yang merepresentasikan implementasi berbeda dari method actionPerformed() di setiap tombol.
   - Pemanggilan setVisible(true) dan dispose() merupakan contoh polimorfisme dinamis, karena dipanggil oleh subclass (ParkirKeluar) yang mewarisi method tersebut dari JFrame.
3. File Login.java
   - Tombol login menggunakan btnLogin.addActionListener(e -> {...}), yang menunjukkan polimorfisme melalui implementasi lambda terhadap method actionPerformed().
   - Pemanggilan new MenuUtamaParkir().setVisible(true) juga contoh polimorfisme dinamis, karena method tersebut dimiliki JFrame tetapi dijalankan dari subclass yang spesifik.

4. File MenuUtamaParkir.java
    - Penggunaan btnParkirMasuk.addActionListener(...), btnParkirKeluar.addActionListener(...) dan btnLogout.addActionListener(...) adalah bentuk polimorfisme lambda expression dari interface ActionListener.
    - Method GUI seperti setLayout() atau setDefaultCloseOperation() juga bentuk polimorfisme dinamis, dipanggil oleh objek subclass yang mewarisi dari JFrame.
5. File CetakTiket.java
    - Method Document document = new Document() lalu dipanggil document.open() merupakan contoh polimorfisme dinamis, karena Document bisa memiliki beberapa bentuk (subclass, ekstensi fungsi PDF).
    - Pemanggilan PdfWriter.getInstance(...) juga menunjukkan polimorfisme statis, di mana method overload tersedia untuk berbagai tipe parameter.
6. File CetakStruk.java
    - Sama seperti CetakTiket, penggunaan method open(), add(...), dan close() pada objek Document adalah bentuk polimorfisme dinamis, karena class Document mengimplementasikan antarmuka dan method umum yang bisa digunakan oleh berbagai objek dokumen.
    - Method PdfWriter.getInstance(...) juga menunjukkan polimorfisme melalui overloading.
7. File DatabaseConnection.java
    - Method getConnection() dapat dianggap sebagai polimorfisme dinamis ketika digunakan oleh class lain yang memanggil koneksi database, karena objek Connection yang dikembalikan bisa berupa berbagai implementasi tergantung driver JDBC.
    - Selain itu, objek DriverManager.getConnection(...) adalah bentuk lain dari polimorfisme, karena dapat menerima berbagai bentuk parameter string untuk berbagai jenis database.

3.5 Exception

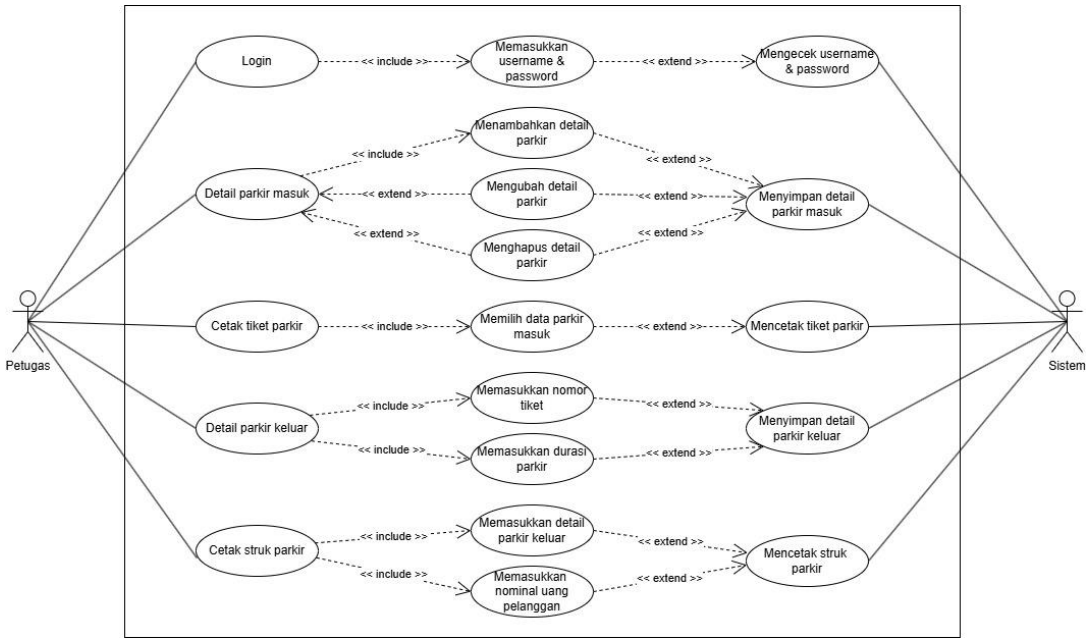
Exception adalah mekanisme dalam pemrograman untuk menangani kesalahan atau kondisi tak terduga saat program berjalan, agar tidak menyebabkan program berhenti secara tiba-tiba dan tetap bisa dikendalikan.

1. File ParkirMasuk.java
    - Blok try-catch digunakan saat melakukan penyimpanan data ke database menggunakan PreparedStatement, untuk menangani kemungkinan SQLException jika input tidak valid atau koneksi bermasalah.
    - Exception juga ditangani saat memuat data dari database dan saat membuat nomor tiket otomatis, dengan menampilkan pesan error lewat JOptionPane.
2. File ParkirKeluar.java
    - Digunakan blok try-catch saat mengambil data parkir berdasarkan nomor tiket untuk menghindari SQLException, terutama ketika tiket tidak ditemukan.
    - Exception juga ditangani saat menghitung biaya parkir dan menyimpan data keluar ke database, dengan menampilkan pesan gagal lewat JOptionPane.
3. File CetakTiket.java
    - Try-catch digunakan untuk menangani kemungkinan kesalahan saat mencetak tiket ke dalam file PDF dengan iText (misalnya kesalahan penulisan file atau file sedang digunakan aplikasi lain).
    - Exception ditangani dengan mencetak pesan kesalahan menggunakan e.printStackTrace() sebagai feedback debugging.
4. File CetakStruk.java
    - Blok try-catch digunakan untuk menangani kesalahan saat mencetak struk PDF, seperti error file output atau masalah font.
    - Penanganan kesalahan dilakukan dengan mencetak stack trace (e.printStackTrace()), sesuai standar debugging Java.

5. File DatabaseConnection.java
    - Exception ditangani di method getConnection() dengan try-catch untuk menangkap SQLException saat koneksi ke MySQL gagal.
    - Jika koneksi gagal, program mencetak pesan kesalahan ke konsol menggunakan System.out.println("Koneksi gagal").
6. File Login.java
    - Try-catch digunakan untuk menangani SQLException ketika melakukan query pencocokan username dan password terhadap tabel user.
    - Jika login gagal atau query error, program menampilkan pesan kesalahan melalui JOptionPane atau mencetak ke konsol.

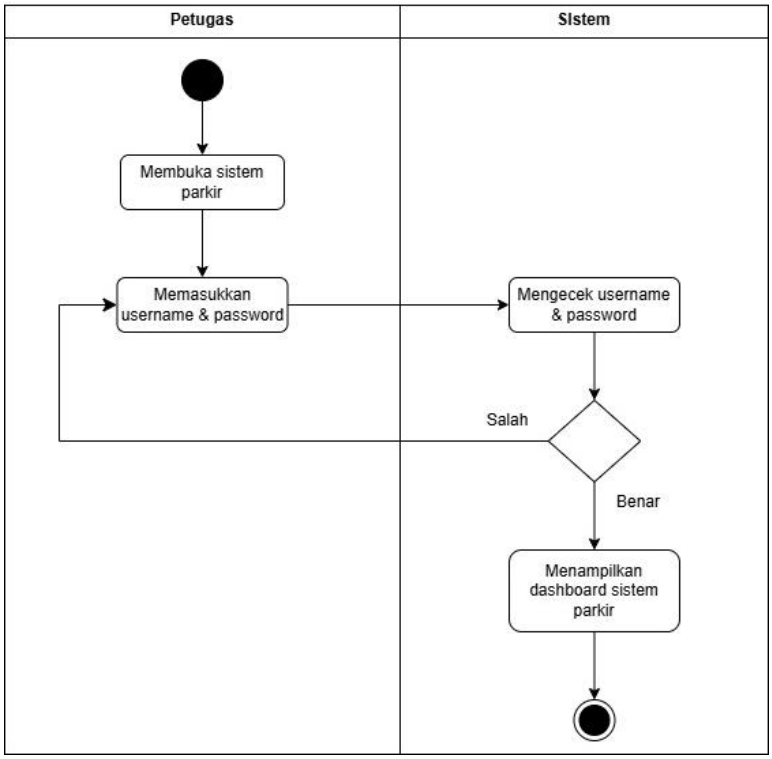
BAB IV
UNIFIED MODELING LANGUAGE

4.1 Use Case Diagram

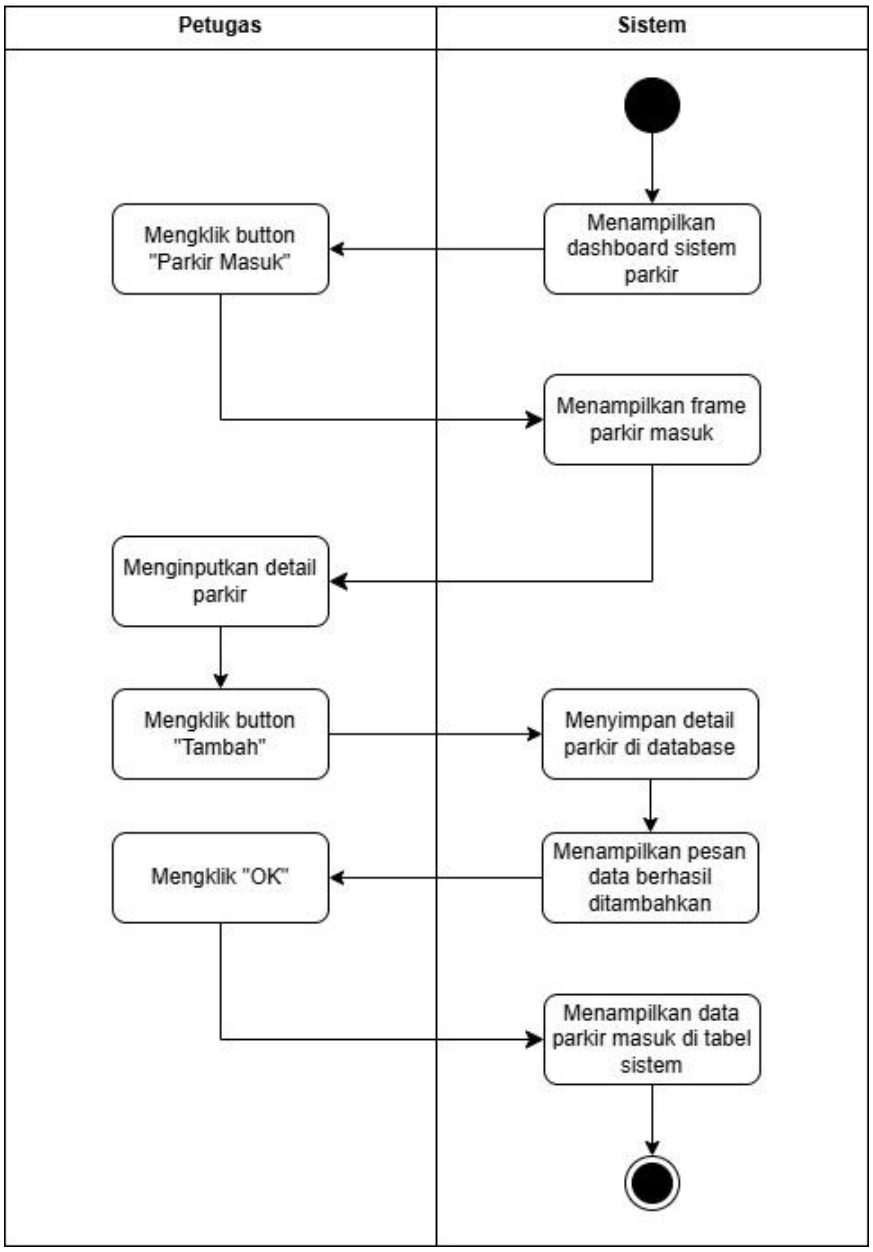


4.2 Activity Diagram

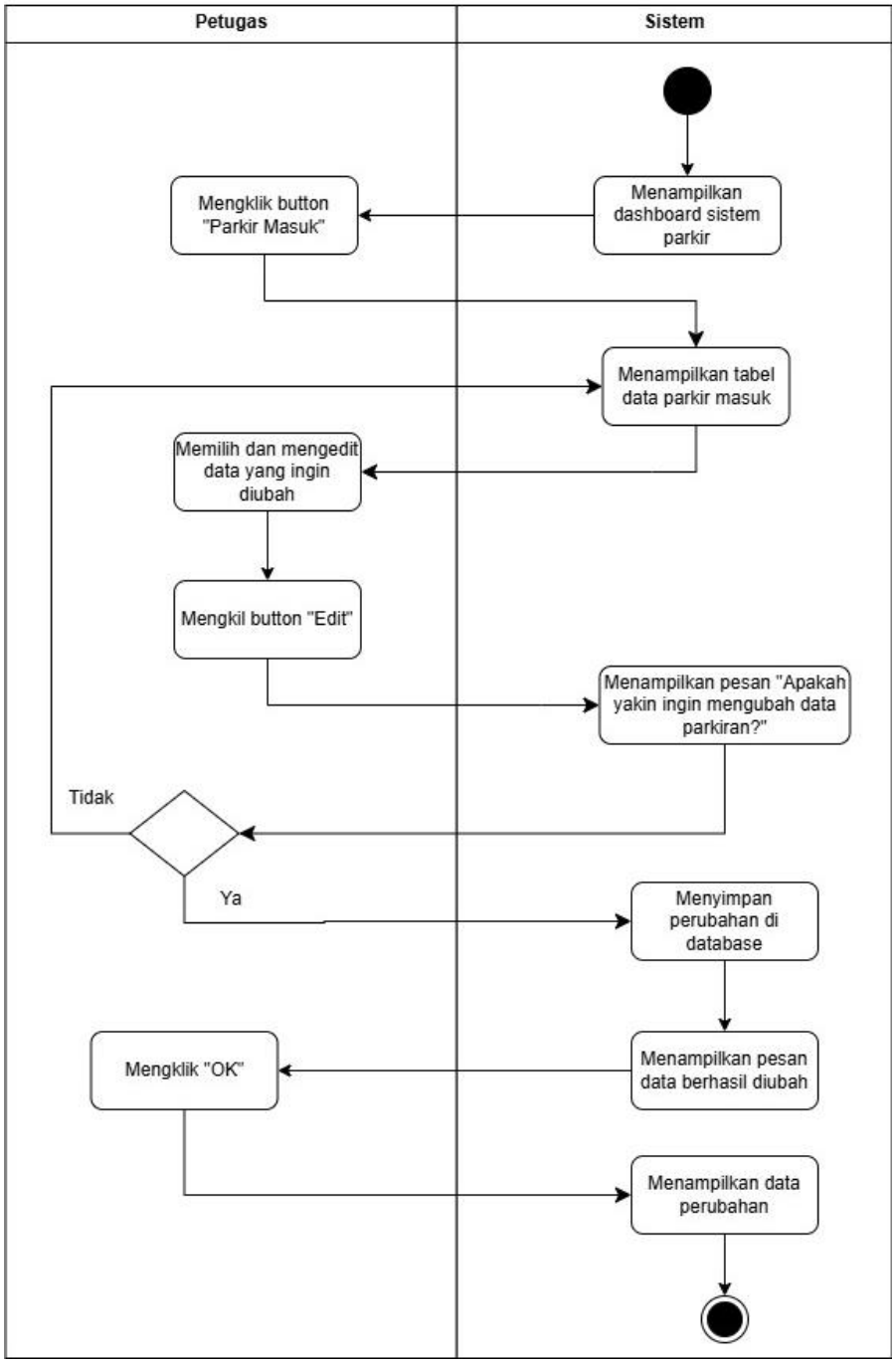
4.2.1 Login



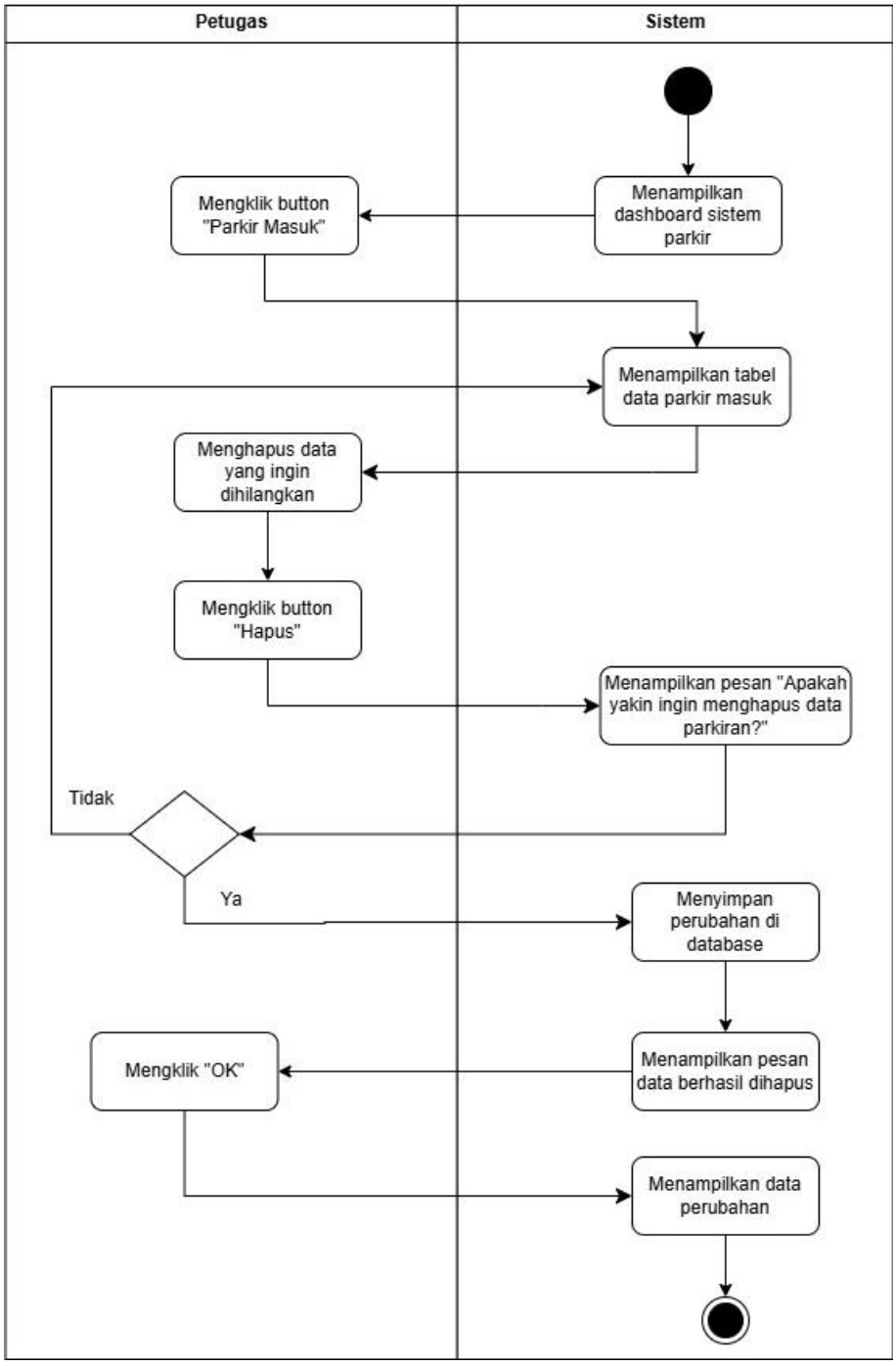
4.2.2 Menambahkan Detail Parkir Masuk



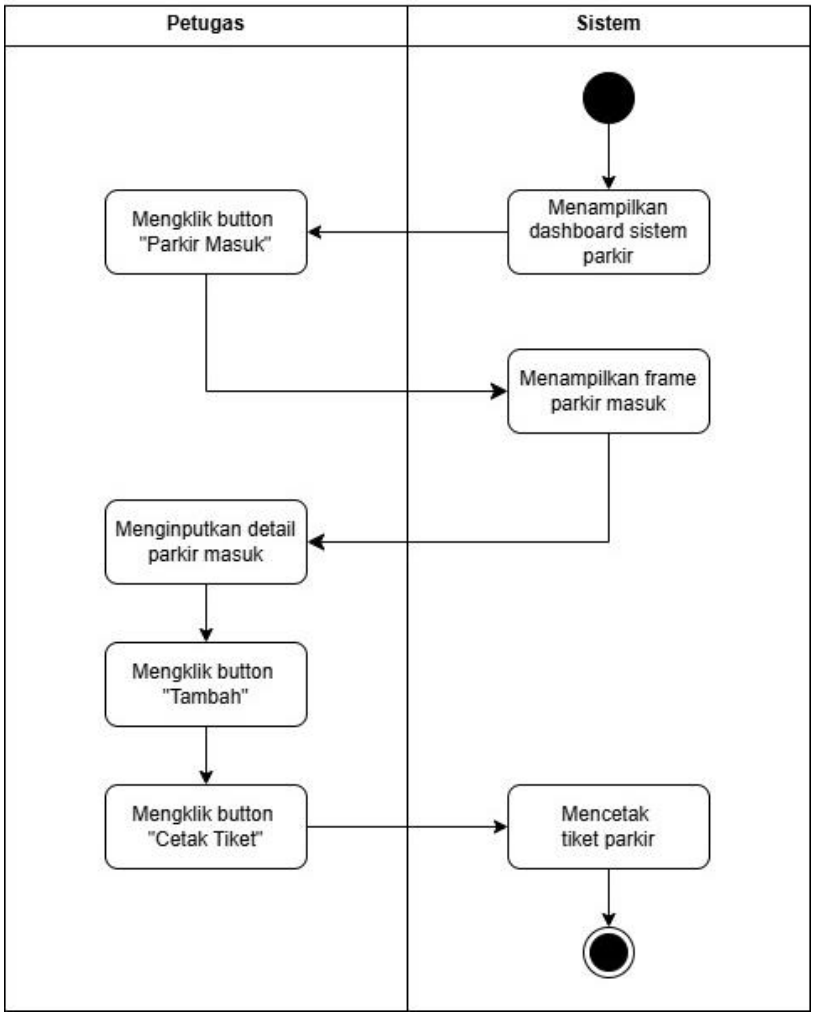
4.2.3 Mengubah Detail Parkir Masuk



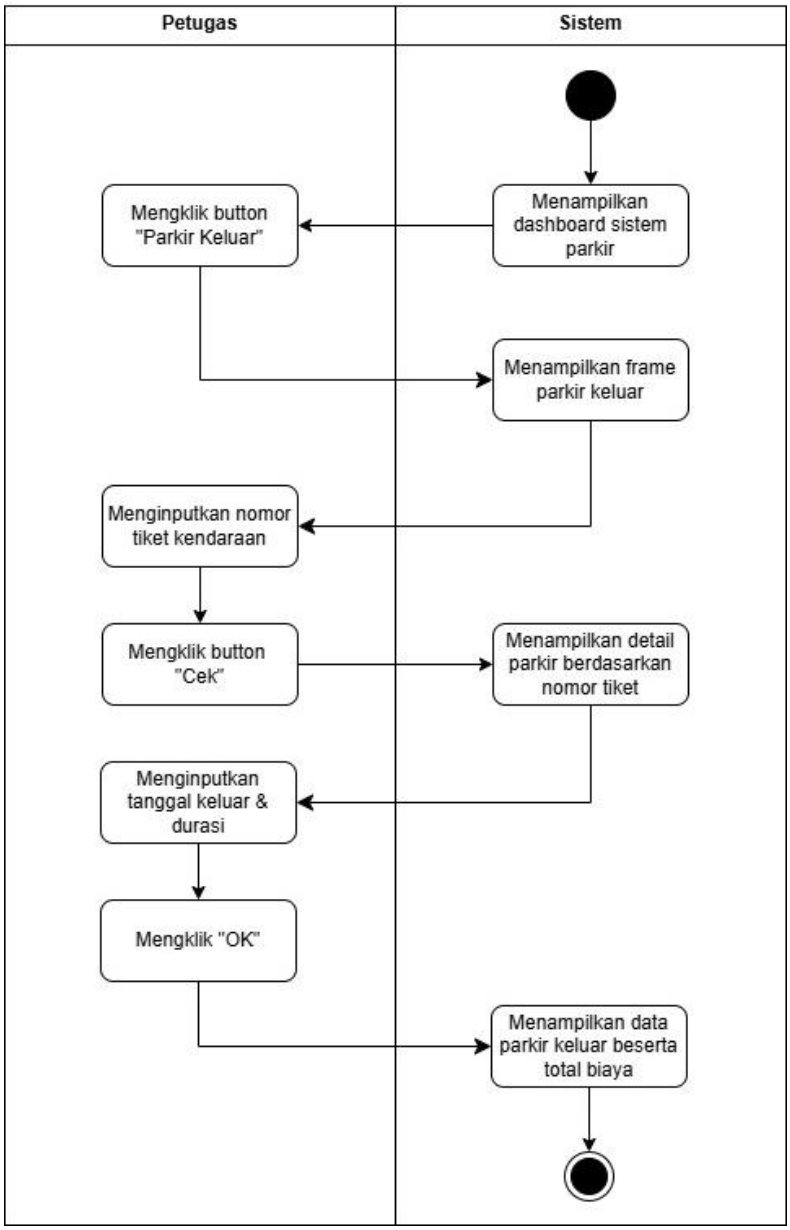
4.2.4 Menghapus Detail Parkir Masuk



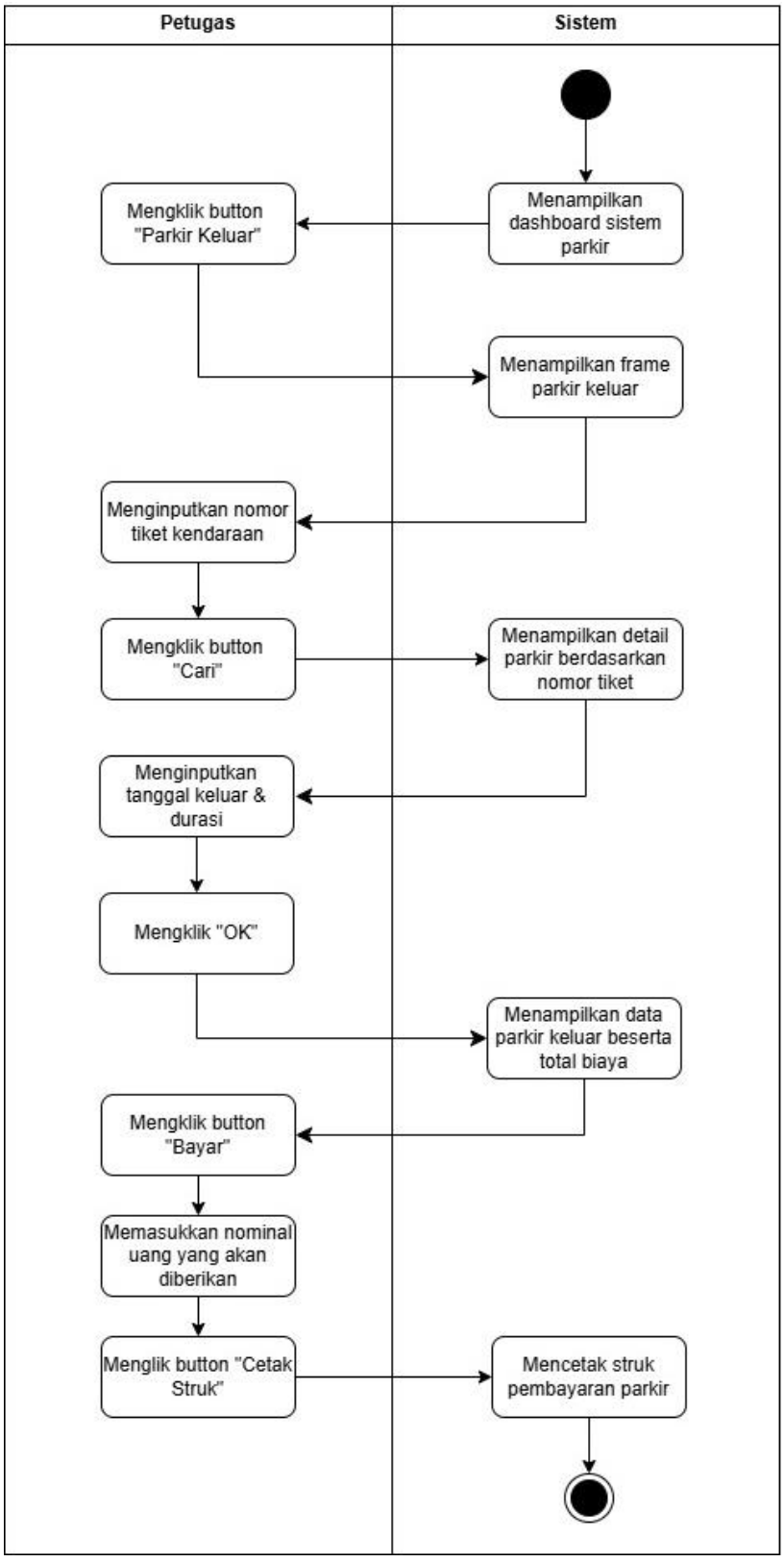
4.2.5 Mencetak Tiket Parkir



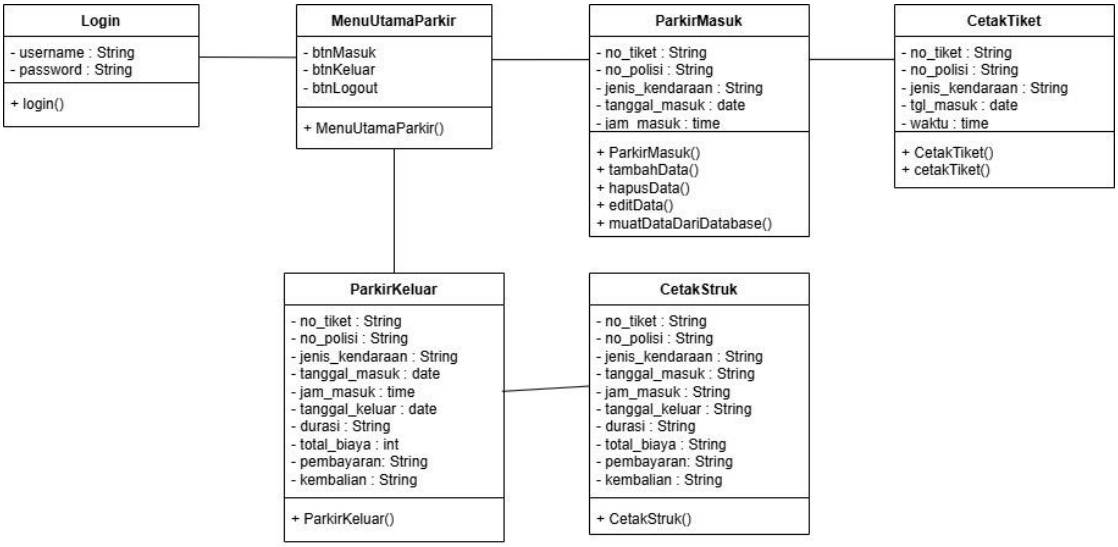
4.2.6 Menambahkan Detail Parkir Keluar



4.2.7 Mencetak Struk Pembayaran



4.3 Class Diagram



BAB V
IMPELEMENTASI KODE PROGRAM & PENGUJIAN

5.1 Implementasi Kode Program

5.1.1 Class Login.java

```
1   package Parkir1;
2
3   import java.awt.*;
4   import javax.swing.*;
5   import javax.swing.border.EmptyBorder;
6
```

Kelas Login merupakan antarmuka awal yang ditampilkan saat pengguna menjalankan aplikasi. Kode diawali dengan deklarasi package Parkir1;, menandakan bahwa kelas ini berada dalam paket tersebut. Beberapa library diimpor dari Java Swing dan AWT, seperti javax.swing.*, javax.swing.border.*, dan java.awt.*, untuk membangun antarmuka grafis. Komponen GUI seperti label, tombol, dan text field diatur melalui kombinasi BorderLayout dan GridBagLayout.

```
public class Login extends JFrame {
    private JTextField userField;
    private JPasswordField passField;

    public Login() {
        setTitle(title:"Login Parkir UNSIKA");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        setMinimumSize(new Dimension(width:800, height:500));
        setLocationRelativeTo(c:null); // Pusat saat pertama dibuka
```

Kelas Login diturunkan dari JFrame, yang merupakan jendela utama dalam aplikasi Swing. Di dalam konstruktor Login(), jendela diatur agar tampil dalam ukuran penuh dan memiliki ukuran minimum agar elemen UI tetap proporsional. Judul jendela diatur sebagai "Login Parkir UNSIKA", dan posisi jendela dipusatkan saat pertama kali dibuka. Panel utama (root) menggunakan BorderLayout, dengan latar belakang abu-abu terang dan padding 50 piksel di setiap sisi. Label judul "Parkir UNSIKA" diletakkan di

atas, ditampilkan dalam font besar dan berwarna biru tua sebagai identitas visual aplikasi.

```
17
18          // Root panel dengan BorderLayout
19          JPanel root = new JPanel(new BorderLayout());
20          root.setBackground(new Color(r:245, g:245, b:245));
21          root.setBorder(new EmptyBorder(top:50, left:50, bottom:50, right:50));
22
23          // Panel judul
24          JLabel title = new JLabel(text:"Parkir UNSIKA", SwingConstants.CENTER);
25          title.setFont(new Font(name:"Segoe UI", Font.BOLD, size:36));
26          title.setForeground(new Color(r:40, g:40, b:120));
27          root.add(title, BorderLayout.NORTH);
28
```

Panel utama (root) menggunakan tata letak BorderLayout dan diberi latar belakang berwarna abu-abu terang. Panel ini juga memiliki batas kosong sebesar 50 piksel di setiap sisi untuk memberikan ruang antara tepi jendela dan konten. Di bagian atas panel, terdapat label judul "Parkir UNSIKA" yang diformat dengan font besar dan warna biru tua, memberikan identitas yang jelas pada aplikasi.

```
29          // Panel tengah dengan GridBagLayout
30          JPanel center = new JPanel(new GridBagLayout());
31          center.setBackground(Color.WHITE);
32          center.setBorder(BorderFactory.createCompoundBorder(
33              BorderFactory.createLineBorder(new Color(r:100,g:100,b:150), thickness:2, rounded:true),
34              new EmptyBorder(top:30, left:30, bottom:30, right:30)
35          ));
36
37          GridBagConstraints gbc = new GridBagConstraints();
38          gbc.insets = new Insets(top:15, left:15, bottom:15, right:15);
39          gbc.fill = GridBagConstraints.HORIZONTAL;
40
```

Panel tengah (center) menggunakan GridBagLayout, memungkinkan penempatan komponen yang fleksibel dan terkontrol. Panel ini memiliki latar belakang putih dan diberi batas kombinasi antara garis berwarna biru dan ruang kosong sebesar 30 piksel di setiap sisi, menciptakan tampilan yang bersih dan profesional. Di dalam panel ini, terdapat label "LOGIN" yang menandai area untuk memasukkan kredensial pengguna.

```
41          // Label "Login"
42          JLabel loginLabel = new JLabel(text:"LOGIN", SwingConstants.CENTER);
43          loginLabel.setFont(new Font(name:"Segoe UI", Font.BOLD, size:24));
44          loginLabel.setForeground(new Color(r:60, g:60, b:100));
45          gbc.gridx = 0;
46          gbc.gridy = 0;
47          gbc.gridwidth = 2;
48          center.add(loginLabel, gbc);
49
```

Label loginLabel menampilkan teks "LOGIN" dengan font dan warna tertentu, ditempatkan di posisi atas tengah panel center. Properti gridx, gridy, dan gridwidth dari GridBagConstraints digunakan untuk menentukan posisi dan lebar komponen dalam grid.

```
50        // Username
51        gbc.gridwidth = 1;
52        gbc.gridy = 1;
53        center.add(new JLabel(text:"Username:"), gbc);
54        gbc.gridx = 1;
55        userField = new JTextField();
56        userField.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:16));
57        center.add(userField, gbc);
58
59        // Password
60        gbc.gridx = 0;
61        gbc.gridy = 2;
62        center.add(new JLabel(text:"Password:"), gbc);
63        gbc.gridx = 1;
64        passField = new JPasswordField();
65        passField.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:16));
66        center.add(passField, gbc);
67
```

Bidang teks untuk username (userField) dan password (passField) ditempatkan secara berurutan dengan label yang sesuai. Kedua bidang teks ini diformat dengan font yang konsisten untuk memastikan keterbacaan. Panel tombol (btnPanel) menggunakan GridLayout untuk menempatkan dua tombol, "Login" dan "Cancel", secara berdampingan dengan jarak 20 piksel di antara keduanya. Tombol "Login" diberi warna biru dengan teks putih, sedangkan tombol "Cancel" diberi warna merah dengan teks putih, memberikan indikasi visual yang jelas mengenai fungsinya.

```
68        // Panel tombol
69        JPanel btnPanel = new JPanel(new GridLayout(rows:1, cols:2, hgap:20, vgap:0));
70        btnPanel.setBackground(Color.WHITE);
71        JButton loginBtn = new JButton(text:"Login");
72        loginBtn.setFont(new Font(name:"Segoe UI", Font.BOLD, size:16));
73        loginBtn.setBackground(new Color(r:40, g:120, b:200));
74        loginBtn.setForeground(Color.WHITE);
75        JButton cancelBtn = new JButton(text:"Cancel");
76        cancelBtn.setFont(new Font(name:"Segoe UI", Font.BOLD, size:16));
77        cancelBtn.setBackground(new Color(r:200, g:40, b:40));
78        cancelBtn.setForeground(Color.WHITE);
79        btnPanel.add(loginBtn);
80        btnPanel.add(cancelBtn);
81
82        gbc.gridx = 0;
83        gbc.gridy = 3;
84        gbc.gridwidth = 2;
85        center.add(btnPanel, gbc);
86
```

Panel btnPanel menggunakan GridLayout untuk menempatkan dua tombol secara horizontal dengan jarak antar tombol. Tombol loginBtn dan cancelBtn masing-masing berfungsi untuk melakukan proses login dan

menghapus input. Warna latar belakang dan teks ditetapkan untuk membedakan fungsi tombol.

```
87          root.add(center, BorderLayout.CENTER);
88          setContentPane(root);
89
```

Panel center yang berisi formulir login dan tombol ditambahkan ke tengah panel root. Kemudian, root ditetapkan sebagai konten utama jendela.

```
90          // Aksi tombol
91          loginBtn.addActionListener(e -> handleLogin());
92          cancelBtn.addActionListener(e -> {
93              userField.setText(t:"");
94              passField.setText(t:"");
95          });
96      }
```

Aksi untuk tombol loginBtn dan cancelBtn ditetapkan menggunakan ActionListener. Saat tombol login diklik, metode handleLogin() dipanggil untuk memproses login. Saat tombol cancel diklik, input pada userField dan passField dikosongkan.

```
97
98      private void handleLogin() {
99          String user = userField.getText().trim();
100         String pass = new String(passField.getPassword());
101         if (user.equals(anObject:"admin") && pass.equals(anObject:"1234")) {
102             JOptionPane.showMessageDialog(this, message:"Login berhasil!", title:"Sukses", JOptionPane.INFORMATION_MESSAGE);
103             dispose();
104             new MenuUtamaParkir().setVisible(b:true);
105         } else {
106             JOptionPane.showMessageDialog(this, message:"Username atau Password salah.", title:"Error", JOptionPane.ERROR_MESSAGE);
107             userField.setText(t:"");
108             passField.setText(t:"");
109         }
110     }
111
```

Fungsi handleLogin() menangani logika saat tombol "Login" ditekan. Fungsi ini mengambil input dari bidang teks username dan password, kemudian memeriksa apakah keduanya sesuai dengan nilai yang telah ditentukan ("admin" dan "1234"). Jika sesuai, maka akan ditampilkan pesan bahwa login berhasil, jendela login ditutup, dan jendela utama aplikasi (MenuUtamaParkir) dibuka. Jika tidak sesuai, maka akan ditampilkan pesan kesalahan, dan bidang teks akan dikosongkan untuk memungkinkan pengguna mencoba kembali.

```
111
        Run | Debug | Run main | Debug main
112     public static void main(String[] args) {
113         SwingUtilities.invokeLater(() -> new Login().setVisible(b:true));
114     }
115 }
116
117
```

Fungsi main() menggunakan SwingUtilities.invokeLater() agar antarmuka dijalankan di thread yang sesuai dengan GUI, sesuai praktik terbaik Java Swing.

5.1.2 Class MenuUtamaParkir.java

```
1   package Parkir1;
2
3   import java.awt.*;
4   import javax.swing.*;
5
6   public class MenuUtamaParkir extends JFrame {
7
8       public MenuUtamaParkir() {
9           setTitle(title:"Menu Utama Parkir");
10          setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11          setSize(width:500, height:400); // Ukuran awal
12          setMinimumSize(new Dimension(width:400, height:300));
13          setLocationRelativeTo(c:null); // Pusat saat pertama dibuka
14          setResizable(resizable:true);
15
```

Kelas MenuUtamaParkir merupakan bagian dari package Parkir1 dan turunan dari JFrame, yang menjadikannya sebagai jendela utama dalam aplikasi GUI. Di dalam konstruktor MenuUtamaParkir(), jendela diatur dengan judul "Menu Utama Parkir", ukuran awal 500x400 piksel, ukuran minimum 400x300 piksel, dan ditempatkan di tengah layar. Jendela ini juga dapat diubah ukurannya oleh pengguna.

```
16          // Panel utama dengan layout GridBag untuk fleksibilitas
17          JPanel panelUtama = new JPanel(new GridBagLayout());
18          panelUtama.setBackground(Color.WHITE); // Latar belakang putih untuk tampilan bersih
19          panelUtama.setBorder(BorderFactory.createEmptyBorder(top:30, left:50, bottom:30, right:50));
20
21          GridBagConstraints gbc = new GridBagConstraints();
22          gbc.gridx = 0;
23          gbc.fill = GridBagConstraints.HORIZONTAL;
24          gbc.insets = new Insets(top:10, left:0, bottom:10, right:0); // Spasi antar komponen
25
```

Panel utama (panelUtama) menggunakan GridBagLayout, sebuah layout manager yang fleksibel yang memungkinkan penempatan komponen dalam grid dengan ukuran sel yang dapat bervariasi. Panel ini diberi latar belakang putih dan padding untuk memberikan ruang antara tepi jendela dan konten.

```
        // Label Judul
        JLabel labelJudul = new JLabel(text:"Parkir UNSIKA", SwingConstants.CENTER);
        labelJudul.setFont(new Font(name:"Segoe UI", Font.BOLD, size:30));
        labelJudul.setForeground(new Color(r:40, g:40, b:100)); // Biru tua
        gbc.gridy = 0;
        panelUtama.add(labelJudul, gbc);

        // Tombol-tombol
        JButton btnMasuk = new JButton(text:"Parkir Masuk");
        JButton btnKeluar = new JButton(text:"Parkir Keluar");
        JButton btnLogout = new JButton(text:"Logout");

        Dimension tombolUkuran = new Dimension(width:250, height:45);
        for (JButton btn : new JButton[]{btnMasuk, btnKeluar, btnLogout}) {
            btn.setPreferredSize(tombolUkuran);
            btn.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:16));
            gbc.gridy++;
            panelUtama.add(btn, gbc);
        }

        // Tambahkan ke frame
        setContentPane(panelUtama);
```

Di dalam panel utama, terdapat label judul "Parkir UNSIKA" yang ditampilkan di atas dengan font besar dan warna biru tua. Selanjutnya, tiga tombol utama ditambahkan secara vertikal: "Parkir Masuk", "Parkir Keluar", dan "Logout". Setiap tombol memiliki ukuran tetap 250x45 piksel dan font yang konsisten untuk tampilan yang seragam.

```
    // Aksi tombol
    btnMasuk.addActionListener(e -> {
        new ParkirMasuk().setVisible(b:true);
        dispose();
    });
    btnKeluar.addActionListener(e -> {
        new ParkirKeluar().setVisible(b:true);
        dispose();
    });
    btnLogout.addActionListener(e -> {
        int confirm = JOptionPane.showConfirmDialog(this, message:"Yakin ingin logout?", title:"Konfirmasi Logout", JOptionPane.YES_NO_OPTION);
        if (confirm == JOptionPane.YES_OPTION) {
            dispose();
            new Login().setVisible(b:true);
        }
    });
}
```

Setiap tombol diberikan aksi menggunakan addActionListener. Tombol "Parkir Masuk" membuka form ParkirMasuk dan menutup menu utama saat ini. Tombol "Parkir Keluar" membuka form ParkirKeluar dan juga menutup menu utama. Tombol "Logout" menampilkan dialog konfirmasi; jika pengguna memilih "Yes", maka jendela saat ini ditutup dan form Login ditampilkan.

```
    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new MenuUtamaParkir().setVisible(b:true));
    }
}
```

Metode main() menggunakan SwingUtilities.invokeLater untuk memastikan bahwa antarmuka pengguna dibuat dan dimodifikasi di thread Event Dispatch Thread (EDT), yang merupakan praktik terbaik dalam pengembangan aplikasi Swing. Metode ini membuat instance dari MenuUtamaParkir dan menampilkannya.

5.1.3 Class ParkirMasuk.java

```
1   package Parkir1;
2
3   import javax.swing.*;
4   import javax.swing.border.TitledBorder;
5   import javax.swing.table.DefaultTableCellRenderer;
6   import javax.swing.table.DefaultTableModel;
7   import javax.swing.table.JTableHeader;
8   import java.awt.*;
9   import java.awt.event.*;
10  import java.sql.*;
11  import java.time.LocalDate;
12  import java.time.LocalTime;
13  import java.time.format.DateTimeFormatter;
14
```

Baris 1 mendeklarasikan bahwa class ParkirMasuk berada dalam package Parkir1. Package digunakan dalam OOP untuk mengelompokkan class-class terkait agar lebih terstruktur dan modular. Baris 3-13 mengimpor berbagai library yang dibutuhkan. javax.swing.* digunakan untuk membangun antarmuka pengguna grafis (GUI), termasuk tombol, label, dan tabel. javax.swing.table berisi class-class yang digunakan untuk manipulasi tabel. java.awt.* dan java.awt.event.* digunakan untuk layout dan event handling. java.sql.* dibutuhkan untuk koneksi dan operasi dengan database, yang merupakan bagian penting dari sistem parkir ini. java.time.* digunakan untuk mengambil dan memformat tanggal serta waktu masuk kendaraan secara otomatis. Penggunaan pustaka ini menunjukkan penerapan reuse dan modularitas dalam OOP.

```
15  public class ParkirMasuk extends JFrame {
16      private JTextField tfNomorTiket, tfNomorPolisi, tfTanggal, tfJam;
17      private JComboBox<String> cbJenisKendaraan;
18      private JTable table;
19      private DefaultTableModel tableModel;
20      private Timer timer;
21
```

Baris 15 mendefinisikan class ParkirMasuk sebagai turunan (extends) dari JFrame, class utama untuk window di Java Swing. Ini merupakan contoh nyata dari inheritance, di mana class ParkirMasuk mewarisi semua sifat dari JFrame. Baris 16-21 terdapat deklarasi atribut-atribut yang merupakan komponen GUI dan objek-objek pendukung lainnya. Modifier private digunakan sebagai bentuk enkapsulasi, yaitu menyembunyikan detail implementasi dari luar class. Ini menjaga integritas data dan memastikan bahwa interaksi hanya melalui method yang disediakan class.

```
22      public ParkirMasuk() {
23          setTitle(title:"Parkir Masuk");
24          setExtendedState(JFrame.MAXIMIZED_BOTH);
25          setMinimumSize(new Dimension(width:850, height:500));
26          setLocationRelativeTo(c:null);
27          setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
28
```

Pada baris 22, konstruktor ParkirMasuk() didefinisikan, yaitu method khusus yang akan dipanggil saat objek dari kelas ParkirMasuk dibuat. Ini merupakan contoh konsep object. Baris 23 menggunakan setTitle() untuk memberikan judul pada window aplikasi, sebuah fitur dari class JFrame (yang diwarisi melalui konsep inheritance). Baris 24 menggunakan setExtendedState() untuk membuat jendela tampil dalam ukuran penuh saat dijalankan, dan baris 25 menentukan ukuran minimum menggunakan class Dimension, yang berasal dari package java.awt. Pada baris 26, setLocationRelativeTo(null) digunakan agar jendela muncul di tengah layar, dan baris 27 setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE) membuat window akan ditutup tanpa menghentikan aplikasi utama (jika ada frame lain).

```
29          JPanel panelUtama = new JPanel(new BorderLayout(hgap:20, vgap:20));
30          panelUtama.setBackground(Color.decode(nm:"#f9f9f9"));
31          panelUtama.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20,
            right:20));
32
```

Selanjutnya pada baris 29, sebuah panel utama (panelUtama) dibuat dengan layout manager BorderLayout, yang memungkinkan komponen diletakkan di bagian atas, bawah, kiri, kanan, dan tengah. Di baris 30, panel

ini diwarnai latar belakang abu-abu muda menggunakan Color.decode(), dan baris 31 menambahkan padding di sekeliling panel dengan setBorder().

```
33      JPanel panelForm = new JPanel(new GridBagLayout());
34      panelForm.setPreferredSize(new Dimension(width:400, height:600));
35      panelForm.setBorder(BorderFactory.createTitledBorder(
36              BorderFactory.createLineBorder(new Color(r:100, g:100, b:150), thickness:1,
                rounded:true),
37              title:"  DATA KENDARAAN MASUK  ",
38              TitledBorder.CENTER, TitledBorder.TOP,
39              new Font(name:"Segoe UI", Font.BOLD, size:15), new Color(r:40, g:40, b:120)
40      ));
41      panelForm.setBackground(Color.WHITE);
42      GridBagConstraints gbc = new GridBagConstraints();
43      gbc.insets = new Insets(top:10, left:10, bottom:10, right:10);
44      gbc.fill = GridBagConstraints.HORIZONTAL;
45      gbc.weightx = 1.0;
46
```

Pada baris 33, dibuat panelForm dengan layout GridBagLayout, sebuah layout fleksibel yang memungkinkan kontrol detail terhadap posisi komponen. Baris 34 menentukan ukurannya, lalu baris 35–40 menetapkan border dengan judul "DATA KENDARAAN MASUK" yang dihias dengan garis dan font tertentu — ini merupakan aspek dari GUI dalam Java. Di baris 41, panel diberi warna latar putih. Baris 42 hingga 45 mengatur objek GridBagConstraints yang akan digunakan untuk mengatur posisi dan perilaku komponen di dalam panel form — menunjukkan penggunaan enkapsulasi karena properti layout dikelola melalui objek gbc.

```
47      gbc.gridx = 0; gbc.gridy = 0;
48      panelForm.add(new JLabel(text:"Nomor Tiket:"), gbc);
49      tfNomorTiket = new JTextField(columns:10);
50      tfNomorTiket.setEditable(b:false);
51      gbc.gridx = 1;
52      panelForm.add(tfNomorTiket, gbc);
53
54      gbc.gridx = 0; gbc.gridy++;
55      panelForm.add(new JLabel(text:"Nomor Polisi:"), gbc);
56      tfNomorPolisi = new JTextField(columns:10);
57      gbc.gridx = 1;
58      panelForm.add(tfNomorPolisi, gbc);
59
60      gbc.gridx = 0; gbc.gridy++;
61      panelForm.add(new JLabel(text:"Jenis Kendaraan:"), gbc);
62      cbJenisKendaraan = new JComboBox<>(new String[]{"Motor", "M
63      gbc.gridx = 1;
64      panelForm.add(cbJenisKendaraan, gbc);
65
```

Baris-baris ini menyusun komponen-komponen input yang ada di form sebelah kiri jendela aplikasi. Di baris 47–48, label "Nomor Tiket:" diletakkan di posisi grid (0,0) menggunakan gbc, lalu di baris 49 tfNomorTiket dibuat sebagai JTextField untuk menampung nomor tiket secara otomatis. Di baris 50, field ini diset setEditable(false) karena nilai

tiket akan di-generate secara otomatis oleh sistem (OOP: prinsip enkapsulasi dan pengendalian akses ke data). Baris 51–52 menambahkan field tersebut ke panel di kolom ke-1 baris ke-0.

Baris 54–58 mengatur label dan field input untuk "Nomor Polisi", dan baris 60–64 mengatur "Jenis Kendaraan" dengan menggunakan JComboBox yang menyajikan dua pilihan: "Motor" dan "Mobil". Ini memudahkan pengguna untuk memilih dan mengurangi kesalahan input manual, mencerminkan prinsip user interface design dalam pengembangan GUI.

```
66          gbc.gridx = 0; gbc.gridy++;
67          panelForm.add(new JLabel(text:"Tanggal:"), gbc);
68          tfTanggal = new JTextField(columns:10);
69          tfTanggal.setEditable(b:false);
70          tfTanggal.setText(LocalDate.now().toString());
71          gbc.gridx = 1;
72          panelForm.add(tfTanggal, gbc);
73
74          gbc.gridx = 0; gbc.gridy++;
75          panelForm.add(new JLabel(text:"Jam:"), gbc);
76          tfJam = new JTextField(columns:10);
77          tfJam.setEditable(b:false);
78          gbc.gridx = 1;
79          panelForm.add(tfJam, gbc);
80
```

Baris 66–72 mengatur komponen input "Tanggal" menggunakan JTextField yang juga diset tidak bisa diedit karena nilainya diisi otomatis dari tanggal saat ini dengan LocalDate.now().toString() — ini merupakan contoh pemrosesan data secara otomatis yang dilakukan dalam konstruktor.

Baris 74–79 mengatur input "Jam" menggunakan JTextField yang juga diset non-editable, dan nilainya akan diisi melalui timer secara real-time. Ini mencerminkan prinsip OOP tentang otomatisasi proses melalui method dan pengelolaan waktu sistem yang bersifat dinamis.

```
81          // Jalankan timer untuk update jam realtime
82          timer = new Timer(delay:1000, e -> {
83              tfJam.setText(LocalTime.now().format(DateTimeFormatter.ofPattern
                (pattern:"HH:mm:ss")));
84          });
85          timer.start();
86
87          gbc.gridx = 0; gbc.gridy++;
88          JPanel panelButtonRow = new JPanel(new GridLayout(rows:1, cols:3, hgap:10, vgap:0));
89          panelButtonRow.setBackground(Color.WHITE);
90          JButton btnTambah = new JButton(text:"Tambah");
91          JButton btnEdit = new JButton(text:"Edit");
92          JButton btnHapus = new JButton(text:"Hapus");
93
```

Pada baris 81–85, sebuah objek Timer dari javax.swing.Timer dibuat untuk mengatur pembaruan jam setiap detik (1000 ms). Baris 82 menggunakan lambda expression (e -> { ... }) sebagai event handler, yang berarti setiap 1 detik, nilai dari tfJam akan diupdate dengan waktu saat ini menggunakan LocalTime.now() dan diformat dengan pola "HH:mm:ss". Ini menunjukkan pemanfaatan event-driven programming, yang umum dalam paradigma OOP saat membangun aplikasi GUI.

Baris 87–88 memulai pembuatan tiga tombol utama (Tambah, Edit, dan Hapus) dalam sebuah panel baris horizontal (panelButtonRow) yang diatur dengan GridLayout 1 baris 3 kolom. Warna latar belakang diset putih untuk konsistensi tampilan (baris 89). Tombol-tombol ini dibuat sebagai objek JButton (baris 90–92), yang nantinya akan dihubungkan ke method seperti tambahData(), editData(), dan hapusData().

```
94      Dimension uniformButtonSize = new Dimension(width:180, height:35);
95      btnTambah.setPreferredSize(uniformButtonSize);
96      btnEdit.setPreferredSize(uniformButtonSize);
97      btnHapus.setPreferredSize(uniformButtonSize);
98
99      panelButtonRow.add(btnTambah);
100     panelButtonRow.add(btnEdit);
101     panelButtonRow.add(btnHapus);
102
103     gbc.gridwidth = 2;
104     panelForm.add(panelButtonRow, gbc);
105     gbc.gridwidth = 1;
106
```

Baris 94–97 menetapkan ukuran tetap (Dimension(180, 35)) ke setiap tombol agar tampil dengan ukuran yang seragam. Ini membantu dalam menjaga konsistensi desain UI.

Pada baris 99–101, tombol-tombol tersebut ditambahkan ke panelButtonRow. Kemudian, baris 103–104 menambahkan panel tersebut ke dalam panelForm dengan mengatur lebar grid menjadi 2 kolom (gridwidth = 2) agar tombol-tombol menempati seluruh baris formulir, lalu dikembalikan ke 1 pada baris 105 untuk pengaturan selanjutnya.

Di baris 107–110, tombol "Cetak Tiket" dibuat sebagai objek JButton dengan ukuran lebar penuh (560 piksel) agar sejajar dengan elemen-elemen sebelumnya. Tombol ini akan digunakan untuk mencetak tiket berdasarkan data yang dipilih pada tabel. Baris 111–112 menambahkan

tombol "Cetak Tiket" ke dalam panelForm, lalu mengatur ulang gridwidth kembali ke 1 karena elemen setelahnya hanya akan mengisi satu kolom atau diatur berbeda.

```
114         gbc.gridx = 0; gbc.gridy++;
115         JButton btnKembali = new JButton(text:"Kembali Ke Menu Utama");
116         btnKembali.setPreferredSize(new Dimension(width:560, height:35));
117         btnKembali.setBackground(new Color(r:220, g:220, b:220));
118         btnKembali.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:14));
119         gbc.gridwidth = 2;
120         panelForm.add(btnKembali, gbc);
121
```

Baris 114–120 mengatur tombol "Kembali Ke Menu Utama" sebagai navigasi untuk kembali ke form utama. Tombol ini diatur agar memiliki ukuran yang konsisten (baris 116), diberi latar abu-abu terang (baris 117), serta font "Segoe UI" ukuran 14 (baris 118) agar tampak profesional dan mudah dibaca. gridwidth = 2 memastikan tombol mengisi dua kolom grid layout (baris 119).

```
122         String[] kolom = {"Nomor Tiket", "Nomor Polisi", "Jenis Kendaraan", "Tanggal Masuk", "Jam
            Masuk"};
123         tableModel = new DefaultTableModel(kolom, rowCount:0);
124         table = new JTable(tableModel);
125         table.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:13));
126         table.setRowHeight(rowHeight:28);
127         JTableHeader header = table.getTableHeader();
128         header.setFont(new Font(name:"Segoe UI", Font.BOLD, size:14));
129         ((DefaultTableCellRenderer) header.getDefaultRenderer()).setHorizontalAlignment(JLabel.
            CENTER);
130
```

Baris 122–124 mendefinisikan tabel untuk menampilkan data kendaraan parkir masuk. kolom adalah array String yang menentukan nama-nama kolom tabel. DefaultTableModel adalah model data untuk JTable, yang memungkinkan kita menambahkan dan menghapus baris data secara dinamis, memperlihatkan prinsip enkapsulasi dan abstraksi dalam pengelolaan data tampilan.

Baris 125–126 mengatur font dan tinggi baris dari tabel, sementara 127–129 memodifikasi tampilan header tabel: font dibuat tebal dan besar, dan teks header ditengahkan menggunakan DefaultTableCellRenderer, yang merupakan bagian dari kustomisasi tampilan komponen GUI Swing.

```
131         DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
132         centerRenderer.setHorizontalAlignment(JLabel.CENTER);
133         for (int i = 0; i < table.getColumnCount(); i++) {
134             table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);
135         }
136
137         table.setSelectionBackground(new Color(r:210, g:210, b:255));
138         JScrollPane scrollPane = new JScrollPane(table);
139         scrollPane.setPreferredSize(new Dimension(width:600, height:400));
140
```

Baris 131–135 mengatur seluruh kolom agar data di dalamnya dirender dengan perataan tengah (center alignment). Ini dilakukan melalui perulangan berdasarkan jumlah kolom dalam tabel, menampilkan penggunaan perulangan dan objek dalam manipulasi GUI berbasis OOP.

Baris 137–139 mengatur warna latar belakang ketika baris di tabel dipilih (SelectionBackground) dan membungkus tabel dalam JScrollPane untuk memungkinkan pengguliran jika data melampaui ukuran tampilan. Ukuran preferensi JScrollPane diatur menjadi 600x400 piksel untuk memberikan ruang tampil yang cukup pada bagian tengah aplikasi.

```
141         btnTambah.addActionListener(e -> tambahData());
142         btnEdit.addActionListener(e -> editData());
143         btnHapus.addActionListener(e -> hapusData());
144         btnKembali.addActionListener(e -> {
145             dispose();
146             new MenuUtamaParkir().setVisible(b:true);
147         });
148         btnCetak.addActionListener(e -> {
149             int selected = table.getSelectedRow();
150             if (selected == -1) {
151                 JOptionPane.showMessageDialog(this, message:"Pilih data yang ingin dicetak
                    tiketnya.");
152                 return;
153             }
154             String tiket = tableModel.getValueAt(selected, column:0).toString();
155             String polisi = tableModel.getValueAt(selected, column:1).toString();
156             String jenis = tableModel.getValueAt(selected, column:2).toString();
157             String tanggal = tableModel.getValueAt(selected, column:3).toString();
158             String jam = tableModel.getValueAt(selected, column:4).toString();
159             CetakTiket.cetakTiket(tiket, polisi, jenis, tanggal, jam);
160         });
161
```

Baris 141–144 menetapkan event handler menggunakan lambda expression (e ->) yang mengacu ke method tertentu saat tombol diklik. Konsep event-driven programming dalam GUI Swing diimplementasikan dengan pendekatan enkapsulasi, karena aksi-aksi ini dibungkus dalam method terpisah (tambahData(), editData(), hapusData()). Tombol-tombol di sini menjadi objek GUI yang memiliki perilaku khusus saat diklik, menunjukkan penerapan object-oriented design.

Baris 145–146 ketika tombol "Kembali" diklik, jendela saat ini ditutup (dispose()), lalu instance dari class MenuUtamaParkir dibuat dan ditampilkan. Ini menunjukkan instansiasi objek baru dan prinsip modularitas, karena navigasi dipisahkan ke dalam class GUI lain. Penerapan ini mencerminkan konsep inheritance dan enkapsulasi, di mana MenuUtamaParkir kemungkinan besar merupakan subclass dari JFrame.

Baris 148–160 menangani tombol "Cetak Tiket". Event handler tombol "Cetak Tiket" dimulai di sini. Program memeriksa apakah ada baris yang dipilih dalam tabel. Jika tidak ada (selected == -1), maka akan ditampilkan dialog peringatan menggunakan JOptionPane. Di sini konsep exception prevention diterapkan secara eksplisit, yaitu mencegah aksi tidak sah sebelum mengakses data array dari baris tabel.

Pada baris 154-160 data dari baris yang dipilih pada tabel diambil menggunakan getValueAt, lalu dikonversi menjadi String. Pendekatan ini merupakan contoh abstraksi data, karena tableModel menyembunyikan detail penyimpanan datanya dan menyediakan antarmuka yang konsisten.

```
162         muatDataDariDatabase();
163         buatNomorTiketOtomatis();
164
165         panelUtama.add(panelForm, BorderLayout.WEST);
166         panelUtama.add(scrollPane, BorderLayout.CENTER);
167         setContentPane(panelUtama);
168     }
169
```

Baris 162–163 memanggil dua method untuk memuat data dari database dan membuat nomor tiket baru secara otomatis. Keduanya merupakan contoh modularisasi kode dan penggunaan method dalam OOP untuk menjaga agar constructor tetap bersih dan ringkas.

Baris 165–168 mengatur tata letak antarmuka grafis, di mana panelForm ditaruh di sebelah kiri (WEST), scrollPane (yang membungkus tabel) ditaruh di tengah (CENTER), lalu semuanya dimasukkan ke dalam JFrame. Ini mencerminkan penggunaan pewarisan GUI Swing dan komposisi objek untuk membentuk antarmuka.

```
170     private void tambahData() {
171      String tiket = tfNomorTiket.getText();
172      String polisi = tfNomorPolisi.getText();
173      String jenis = cbJenisKendaraan.getSelectedItem().toString();
174      String tanggal = LocalDate.now().toString();
175      String jam = LocalTime.now().format(DateTimeFormatter.ofPattern(pattern:"HH:mm:ss"));
176
```

Baris 170–175 merupakan method untuk menambahkan data, mengambil input dari field GUI dan membentuk data yang akan disimpan. Penggunaan LocalDate dan LocalTime dari java.time adalah contoh library berbasis objek, dan pengambilan nilai dari GUI memperlihatkan enkapsulasi data antarmuka.

```
177     try (Connection conn = DatabaseConnection.getConnection()) {
178         String sql = "INSERT INTO parkir (nomor_tiket, nomor_polisi, jenis_kendaraan,
            tanggal_masuk, jam_masuk) VALUES (?, ?, ?, ?, ?)";
179         PreparedStatement pst = conn.prepareStatement(sql);
180         pst.setString(parameterIndex:1, tiket);
181         pst.setString(parameterIndex:2, polisi);
182         pst.setString(parameterIndex:3, jenis);
183         pst.setString(parameterIndex:4, tanggal);
184         pst.setString(parameterIndex:5, jam);
185         pst.executeUpdate();
186
```

Baris 177–185 menjalankan proses penyimpanan data ke database menggunakan koneksi dan PreparedStatement. Ini bagian dari exception handling dan enkapsulasi koneksi database, di mana kelas DatabaseConnection menyembunyikan detail implementasi koneksi.

```
187         tableModel.addRow(new Object[]{tiket, polisi, jenis, tanggal, jam});
188         tfNomorPolisi.setText(t:"");
189         buatNomorTiketOtomatis();
190         tfTanggal.setText(LocalDate.now().toString());
191
192         JOptionPane.showMessageDialog(this, message:"Data berhasil ditambahkan!");
193     } catch (SQLException e) {
194         JOptionPane.showMessageDialog(this, "Gagal menambahkan data: " + e.getMessage());
195     }
196 }
197
```

Baris 187–196 menambahkan data baru ke tabel dan mengatur ulang form input. Pesan dialog ditampilkan sebagai feedback ke pengguna. Ini menunjukkan interaksi antar objek dan exception handling yang baik jika proses gagal.

```
198     private void editData() {
199      int selected = table.getSelectedRow();
200      if (selected == -1) {
201          JOptionPane.showMessageDialog(this, message:"Pilih data terlebih dahulu.");
202          return;
203      }
204
```

Baris 198–203 merupakan method untuk mengedit data, memeriksa apakah baris pada tabel telah dipilih sebelum mengedit. Validasi ini penting dalam interaksi berbasis event dan mencegah error logika. Baris 201–203 memberikan konfirmasi kepada pengguna sebelum mengubah data. Ini merupakan bagian dari GUI interaktif dan good UX practice.

```
210     String tiket = tableModel.getValueAt(selected, column:0).toString();
211     String polisi = tfNomorPolisi.getText();
212     String jenis = cbJenisKendaraan.getSelectedItem().toString();
213
214     try (Connection conn = DatabaseConnection.getConnection()) {
215         String sql = "UPDATE parkir SET nomor_polisi=?, jenis_kendaraan=? WHERE nomor_tiket=?";
216         PreparedStatement pst = conn.prepareStatement(sql);
217         pst.setString(parameterIndex:1, polisi);
218         pst.setString(parameterIndex:2, jenis);
219         pst.setString(parameterIndex:3, tiket);
220         pst.executeUpdate();
221
222         tableModel.setValueAt(polisi, selected, column:1);
223         tableModel.setValueAt(jenis, selected, column:2);
224
225         JOptionPane.showMessageDialog(this, message:"Data berhasil diubah!");
226     } catch (SQLException e) {
227         JOptionPane.showMessageDialog(this, "Gagal mengedit data: " + e.getMessage());
228     }
229 }
230
```

Baris 210–229 adalah proses update data ke database dan update tampilan GUI. Ini menunjukkan koneksi database yang aman, manipulasi data GUI, dan pemisahan tanggung jawab dalam desain metode OOP.

```
231     private void hapusData() {
232     int selected = table.getSelectedRow();
233     if (selected == -1) {
234         JOptionPane.showMessageDialog(this, message:"Pilih data terlebih dahulu.");
235         return;
236     }
237
238     int konfirmasi = JOptionPane.showConfirmDialog(this, message:"Apakah yakin ingin menghapus
        data?", title:"Konfirmasi Hapus", JOptionPane.YES_NO_OPTION);
239     if (konfirmasi != JOptionPane.YES_OPTION) {
240         return;
241     }
242
```

Baris 231–241 merupakan method untuk menghapus data. Pada baris 232 kembali melakukan validasi apakah ada data yang dipilih sebelum menghapus. Baris 225–227 memberi peringatan konfirmasi sebelum penghapusan dilakukan.

```
243     String tiket = tableModel.getValueAt(selected, column:0).toString();
244
245     try (Connection conn = DatabaseConnection.getConnection()) {
246         String sql = "DELETE FROM parkir WHERE nomor_tiket=?";
247         PreparedStatement pst = conn.prepareStatement(sql);
248         pst.setString(parameterIndex:1, tiket);
249         pst.executeUpdate();
250
251         tableModel.removeRow(selected);
252
253         JOptionPane.showMessageDialog(this, message:"Data berhasil dihapus!");
254     } catch (SQLException e) {
255         JOptionPane.showMessageDialog(this, "Gagal menghapus data: " + e.getMessage());
256     }
257 }
258
```

Baris 243–257 menghapus data dari database dan tampilan. Konsep penghapusan data via SQL, sinkronisasi tampilan, dan exception handling digunakan di sini.

```
259     private void muatDataDariDatabase() {
260         try (Connection conn = DatabaseConnection.getConnection()) {
261             String sql = "SELECT nomor_tiket, nomor_polisi, jenis_kendaraan, tanggal_masuk,
                jam_masuk FROM parkir";
262             Statement stmt = conn.createStatement();
263             ResultSet rs = stmt.executeQuery(sql);
264             while (rs.next()) {
265                 tableModel.addRow(new Object[]{
266                         rs.getString(columnLabel:"nomor_tiket"),
267                         rs.getString(columnLabel:"nomor_polisi"),
268                         rs.getString(columnLabel:"jenis_kendaraan"),
269                         rs.getString(columnLabel:"tanggal_masuk"),
270                         rs.getString(columnLabel:"jam_masuk")
271                 });
272             }
273         } catch (SQLException e) {
274             JOptionPane.showMessageDialog(this, "Gagal memuat data: " + e.getMessage());
275         }
276     }
277
```

Kode program ini merupakan method untuk memuat data. Baris 260–276 membaca seluruh data dari database dan menampilkannya ke dalam tabel GUI. Ini menunjukkan abstraksi dan enkapsulasi: pengguna tidak tahu detail SQL, hanya melihat tabel terisi.

```
278     private void buatNomorTiketOtomatis() {
279         try (Connection conn = DatabaseConnection.getConnection()) {
280             String sql = "SELECT nomor_tiket FROM parkir ORDER BY id DESC LIMIT 1";
281             Statement stmt = conn.createStatement();
282             ResultSet rs = stmt.executeQuery(sql);
283             String nomorBaru = "TKT0001";
284             if (rs.next()) {
285                 String last = rs.getString(columnLabel:"nomor_tiket");
286                 int nextNum = Integer.parseInt(last.substring(beginIndex:3)) + 1;
287                 nomorBaru = String.format(format:"TKT%04d", nextNum);
288             }
289             tfNomorTiket.setText(nomorBaru);
290         } catch (SQLException e) {
291             JOptionPane.showMessageDialog(this, "Gagal membuat nomor tiket otomatis: " + e.
                getMessage());
292         }
293     }
294
```

Kode program ini merupakan method untuk membuat nomor tiket. Baris 279–293 menghasilkan nomor tiket otomatis berdasarkan tiket terakhir di database. Ini contoh logika otomatisasi ID unik dan manipulasi string, sesuai dengan praktik OOP dan SQL.

```
    Run | Debug
295 public static void main(String[] args) {
296     SwingUtilities.invokeLater(() -> new ParkirMasuk().setVisible(b:true));
297 }
298 }
```

Baris 295–298 adalah titik awal aplikasi. SwingUtilities.invokeLater() memastikan GUI berjalan di thread yang benar, dan pembuatan objek ParkirMasuk memanggil constructor. Ini contoh konsep objek dan inisialisasi GUI dalam OOP Java.

5.1.4 Class ParkirKeluar.java

```
1  package Parkir1;
2
3  import java.awt.*;
4  import java.awt.event.*;
5  import java.sql.*;
6  import java.time.*;
7  import java.time.format.DateTimeFormatter;
8  import javax.swing.*;
9  import javax.swing.border.TitledBorder;
10 import javax.swing.table.*;
11
12 public class ParkirKeluar extends JFrame {
13     private JTextField tfNomorTiket, tfNomorPolisi, tfJenisKendaraan, tfTanggalMasuk, tfJamMasuk;
14     private JTextField tfTanggalKeluar, tfDurasi, tfTotal, tfPembayaran;
15     private JTable table;
16     private DefaultTableModel tableModel;
17     private Timer timer;
18     private LocalDateTime waktuMasuk;
19     private long totalBiaya = 0;
20     private JButton btnBayar;
21
22
23     public ParkirKeluar() {
24         setTitle(title:"Parkir Keluar");
25         setExtendedState(JFrame.MAXIMIZED_BOTH);
26         setMinimumSize(new Dimension(width:850, height:500));
27         setLocationRelativeTo(c:null);
28         setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

Kelas ParkirKeluar digunakan untuk mencatat data kendaraan yang keluar. Sama seperti ParkirMasuk, kelas ini merupakan turunan dari JFrame, ditampilkan dalam ukuran penuh, dan dipusatkan di layar. Pada konstruktor, frame diatur dengan judul "Parkir Keluar", dimaksimalkan ukurannya, dan ditetapkan ukuran minimum. Lokasi frame diatur agar muncul di tengah layar, dan operasi default saat ditutup adalah DISPOSE_ON_CLOSE, yang berarti frame akan ditutup tanpa menghentikan seluruh aplikasi.

```
        JPanel panelUtama = new JPanel(new BorderLayout(hgap:20, vgap:20));
        panelUtama.setBackground(Color.decode(nm:"#f9f9f9"));
        panelUtama.setBorder(BorderFactory.createEmptyBorder(top:20, left:20, bottom:20, right:20));

        JPanel panelForm = new JPanel(new GridBagLayout());
        panelForm.setPreferredSize(new Dimension(width:400, height:600));
        panelForm.setBorder(BorderFactory.createTitledBorder(
                BorderFactory.createLineBorder(new Color(r:100, g:100, b:150), thickness:1, rounded:true),
                title:"  DATA KENDARAAN KELUAR  ",
                TitledBorder.CENTER, TitledBorder.TOP,
                new Font(name:"Segoe UI", Font.BOLD, size:15), new Color(r:40, g:40, b:120)
        ));
```

Panel utama bernama panelUtama yang menggunakan layout BorderLayout dengan jarak antar komponen 20 piksel. Panel ini diberi warna latar belakang putih keabu-abuan dan diberi margin kosong di keempat sisinya. Di dalamnya terdapat panel lain bernama panelForm, yang menggunakan layout GridBagLayout, layout ini dipilih karena fleksibel untuk menyusun form secara rapi dalam baris dan kolom. Ukuran panel form diatur dengan preferensi tinggi 600 dan lebar 400 piksel, serta diberi judul batas panel "DATA KENDARAAN KELUAR" yang diletakkan di tengah atas dengan font tebal dan warna biru gelap untuk menegaskan bahwa form ini khusus untuk data kendaraan yang akan keluar dari parkiran.

```
        panelForm.setBackground(Color.WHITE);
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(top:10, left:10, bottom:10, right:10);
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.weightx = 1.0;

        gbc.gridx = 0; gbc.gridy = 0;
        panelForm.add(new JLabel(text:"Nomor Tiket:"), gbc);
        tfNomorTiket = new JTextField(columns:10);
        gbc.gridx = 1;
        panelForm.add(tfNomorTiket, gbc);
        JButton btnCek = new JButton(text:"Cek");
        gbc.gridx = 2;
        panelForm.add(btnCek, gbc);

        gbc.gridx = 0; gbc.gridy++;
        panelForm.add(new JLabel(text:"Nomor Polisi:"), gbc);
        tfNomorPolisi = new JTextField(columns:10); tfNomorPolisi.setEditable(b:false);
        gbc.gridx = 1; gbc.gridwidth = 2;
        panelForm.add(tfNomorPolisi, gbc); gbc.gridwidth = 1;

        gbc.gridx = 0; gbc.gridy++;
        panelForm.add(new JLabel(text:"Jenis Kendaraan:"), gbc);
        tfJenisKendaraan = new JTextField(columns:10); tfJenisKendaraan.setEditable(b:false);
        gbc.gridx = 1; gbc.gridwidth = 2;
        panelForm.add(tfJenisKendaraan, gbc); gbc.gridwidth = 1;

        gbc.gridx = 0; gbc.gridy++;
        panelForm.add(new JLabel(text:"Tanggal Masuk:"), gbc);
        tfTanggalMasuk = new JTextField(columns:10); tfTanggalMasuk.setEditable(b:false);
        gbc.gridx = 1; gbc.gridwidth = 2;
        panelForm.add(tfTanggalMasuk, gbc); gbc.gridwidth = 1;

        gbc.gridx = 0; gbc.gridy++;
        panelForm.add(new JLabel(text:"Jam Masuk:"), gbc);
        tfJamMasuk = new JTextField(columns:10); tfJamMasuk.setEditable(b:false);
        gbc.gridx = 1; gbc.gridwidth = 2;
        panelForm.add(tfJamMasuk, gbc); gbc.gridwidth = 1;
```

Selanjutnya, diatur GridBagConstraints bernama gbc yang mengatur posisi, jarak antar komponen, dan cara pengisian ruang dalam form. Form dimulai dengan label "Nomor Tiket", diikuti oleh JTextField tempat pengguna menginput nomor tiket, lalu tombol "Cek" yang berfungsi untuk mengambil data kendaraan berdasarkan nomor tiket tersebut. Baris-baris berikutnya menampilkan data kendaraan seperti Nomor Polisi, Jenis Kendaraan, Tanggal Masuk, dan Jam Masuk, di mana semua field ini dalam keadaan non-editable karena hanya menampilkan data dari database, bukan untuk diisi langsung oleh pengguna.

```
80          gbc.gridx = 0; gbc.gridy++;
81          panelForm.add(new JLabel(text:"Tanggal Keluar:"), gbc);
82          tfTanggalKeluar = new JTextField(columns:10); tfTanggalKeluar.setEditable(b:false);
83          tfTanggalKeluar.setText(LocalDate.now().format(DateTimeFormatter.ofPattern(pattern:"yyyy-MM-dd")));
84          gbc.gridx = 1; gbc.gridwidth = 2;
85          panelForm.add(tfTanggalKeluar, gbc); gbc.gridwidth = 1;
86
87          gbc.gridx = 0; gbc.gridy++;
88          panelForm.add(new JLabel(text:"Durasi:"), gbc);
89          tfDurasi = new JTextField(columns:10); tfDurasi.setEditable(b:false);
90          gbc.gridx = 1;
91          panelForm.add(tfDurasi, gbc);
92          JButton btnOk = new JButton(text:"OK");
93          gbc.gridx = 2;
94          panelForm.add(btnOk, gbc);
95
```

Lalu terdapat field "Tanggal Keluar" yang otomatis terisi dengan tanggal hari ini menggunakan LocalDate.now() dan diformat ke dalam pola yyyy-MM-dd. Setelah itu, ada field "Durasi" yang akan menampilkan berapa lama kendaraan berada di dalam parkiran. Di sebelah field durasi terdapat tombol "OK" yang kemungkinan akan digunakan untuk memproses durasi parkir dan menghitung total biaya, meskipun fungsinya baru akan terlihat lengkap di bagian kode selanjutnya.

```
96          // Baris Total Biaya
97          gbc.gridx = 0;
98          gbc.gridy++;
99          panelForm.add(new JLabel(text:"Total Biaya:"), gbc);
100
101         // TextField untuk Total Biaya
102         tfTotal = new JTextField(columns:10);
103         tfTotal.setEditable(b:false);
104         gbc.gridx = 1;
105         panelForm.add(tfTotal, gbc);
```

Pada bagian ini, ditambahkan elemen antarmuka pengguna untuk menampilkan informasi total biaya. Sebuah label dengan teks "Total Biaya:" ditambahkan ke dalam panel form menggunakan GridBagConstraints untuk penataan posisi. Kemudian, sebuah JTextField (tfTotal) dibuat dengan lebar

10 kolom dan diset tidak dapat diedit (setEditable(false)), karena hanya digunakan untuk menampilkan informasi. GridBagConstraints digunakan lagi untuk menempatkan tfTotal di posisi yang sesuai dalam grid.

```
106
107         // Tombol Bayar di sebelah tfTotal (gridx = 2)
108         btnBayar = new JButton(text:"Bayar");
109         gbc.gridx = 2;
110         panelForm.add(btnBayar, gbc);
111
112         // Action untuk tombol Bayar
113         btnBayar.addActionListener(e -> {
114             String input = JOptionPane.showInputDialog(this, message:"Silahkan masukkan pembayaran:");
115             if (input != null && !input.trim().isEmpty()) {
116                 tfPembayaran.setText(input.trim());
117             }
118         });
119
```

Setelah tfTotal, sebuah tombol JButton dengan label "Bayar" ditambahkan ke dalam panel form. Tombol ini ditempatkan di sebelah tfTotal menggunakan GridBagConstraints. Tombol ini akan digunakan untuk memproses pembayaran setelah pengguna memverifikasi total biaya yang harus dibayar.

Sebuah ActionListener ditambahkan pada tombol "Bayar" untuk menangani aksi klik dari pengguna. Ketika tombol diklik, sebuah dialog input (JOptionPane.showInputDialog) muncul, meminta pengguna untuk memasukkan jumlah pembayaran. Jika pengguna memasukkan nilai dan menekan tombol OK, nilai tersebut akan disalin ke dalam tfPembayaran setelah menghapus spasi yang tidak diperlukan. Jika pengguna membatalkan atau menutup dialog, tidak ada perubahan yang terjadi.

Sebuah JTextField (tfPembayaran) ditambahkan ke dalam panel form untuk menampilkan jumlah pembayaran yang dimasukkan oleh pengguna. Field ini diset tidak terlihat (setVisible(false)) pada awalnya dan akan ditampilkan hanya setelah tombol "Bayar" diklik dan pengguna memasukkan jumlah pembayaran.

```
120         gbc.gridx = 0; gbc.gridy++;
121         tfPembayaran = new JTextField(columns:10);
122         tfPembayaran.setVisible(aFlag:false);
123         gbc.gridx = 1; gbc.gridwidth = 2;
124         panelForm.add(tfPembayaran, gbc); gbc.gridwidth = 1;
125         gbc.gridx = 0; gbc.gridy++;
126         gbc.gridwidth = 3;
127         JButton btnCetakStruk = new JButton(text:"Cetak Struk");
128         panelForm.add(btnCetakStruk, gbc);
129
```

Sebuah tombol JButton dengan label "Cetak Struk" ditambahkan ke dalam panel form. Tombol ini akan digunakan untuk mencetak struk pembayaran setelah transaksi selesai. Tombol ini ditempatkan di bawah tfPembayaran menggunakan GridBagConstraints.

```
130         gbc.gridy++;
131         JButton btnKembali = new JButton(text:"Kembali Ke Menu Utama");
132         btnKembali.setBackground(new Color(r:220, g:220, b:220));
133         btnKembali.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:14));
134         panelForm.add(btnKembali, gbc);
135         gbc.gridwidth = 1;
136
```

Sebuah tombol JButton dengan label "Kembali Ke Menu Utama" ditambahkan ke dalam panel form. Tombol ini memungkinkan pengguna untuk kembali ke menu utama aplikasi parkir. Tombol ini ditempatkan di bawah tombol "Cetak Struk" menggunakan GridBagConstraints.

```
137     String[] kolom = {"Nomor Tiket", "Nomor Polisi", "Jenis Kendaraan", "Tanggal Masuk", "Jam Masuk", "Tanggal Keluar", "Durasi"};
138     tableModel = new DefaultTableModel(kolom, rowCount:0);
139     table = new JTable(tableModel);
140     table.setFont(new Font(name:"Segoe UI", Font.PLAIN, size:13));
141     table.setRowHeight(rowHeight:28);
142     JTableHeader header = table.getTableHeader();
143     header.setFont(new Font(name:"Segoe UI", Font.BOLD, size:14));
144     ((DefaultTableCellRenderer) header.getDefaultRenderer()).setHorizontalAlignment(JLabel.CENTER);
145
```

Sebuah JTable dengan model data DefaultTableModel disiapkan untuk menampilkan data kendaraan yang telah keluar dari area parkir. Tabel ini memiliki kolom-kolom seperti "Nomor Tiket", "Nomor Polisi", "Jenis Kendaraan", "Tanggal Masuk", "Jam Masuk", "Tanggal Keluar", dan "Durasi". Font dan tinggi baris tabel disesuaikan untuk meningkatkan keterbacaan. Header tabel diberi font tebal dan disejajarkan ke tengah.

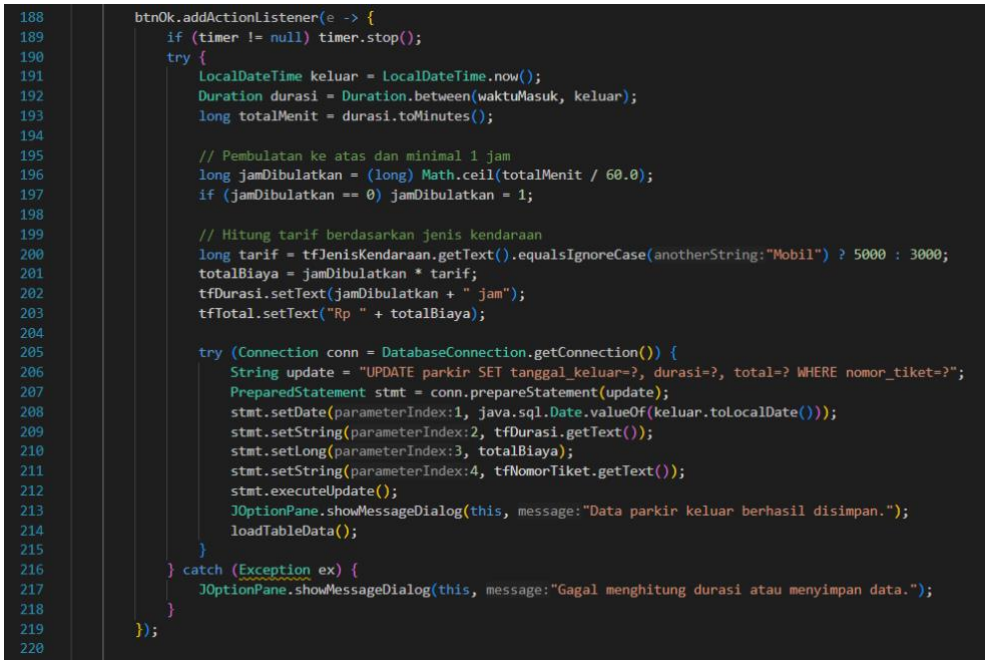
```
146     DefaultTableCellRenderer centerRenderer = new DefaultTableCellRenderer();
147     centerRenderer.setHorizontalAlignment(JLabel.CENTER);
148     for (int i = 0; i < table.getColumnCount(); i++) {
149         table.getColumnModel().getColumn(i).setCellRenderer(centerRenderer);
150     }
151
152     table.setSelectionBackground(new Color(r:210, g:210, b:255));
153     JScrollPane scrollPane = new JScrollPane(table);
154     scrollPane.setPreferredSize(new Dimension(width:600, height:400));
155
156     loadTableData();
157
```

Untuk memungkinkan scroll pada tabel jika data melebihi ukuran tampilan, sebuah JScrollPane ditambahkan di sekitar JTable. ScrollPane ini diset dengan ukuran preferensi tertentu dan ditambahkan ke dalam panel utama di bagian tengah.

```
btnCek.addActionListener(e -> {
    String tiket = tfNomorTiket.getText().trim();
    try (Connection conn = DatabaseConnection.getConnection()) {
        String query = "SELECT * FROM parkir WHERE nomor_tiket = ?";
        PreparedStatement stmt = conn.prepareStatement(query);
        stmt.setString(parameterIndex:1, tiket);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
            tfNomorPolisi.setText(rs.getString(columnLabel:"nomor_polisi"));
            tfJenisKendaraan.setText(rs.getString(columnLabel:"jenis_kendaraan"));
            tfTanggalMasuk.setText(rs.getDate(columnLabel:"tanggal_masuk").toString());
            tfJamMasuk.setText(rs.getTime(columnLabel:"jam_masuk").toString());
            waktuMasuk = LocalDateTime.of(rs.getDate(columnLabel:"tanggal_masuk").toLocalDate(), rs.getTime(columnLabel:"jam_masuk").toLocalTime());

            if (timer != null) timer.stop();
            timer = new Timer(delay:1000, evt -> {
                Duration durasi = Duration.between(waktuMasuk, LocalDateTime.now());
                long jam = durasi.toMinutes() / 60;
                long menit = durasi.toMinutes() % 60;
                tfDurasi.setText(jam + " jam " + menit + " menit");
            });
            timer.start();
        } else {
            JOptionPane.showMessageDialog(this, message:"Nomor Tiket tidak ditemukan.");
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, message:"Gagal mengambil data dari database.");
    }
});
```

Sebuah ActionListener ditambahkan pada tombol "Cek" untuk menangani aksi klik dari pengguna. Ketika tombol diklik, aplikasi akan mencari data kendaraan berdasarkan nomor tiket yang dimasukkan oleh pengguna. Jika data ditemukan, informasi seperti nomor polisi, jenis kendaraan, tanggal dan jam masuk akan ditampilkan di field yang sesuai. Jika data tidak ditemukan, sebuah pesan kesalahan akan ditampilkan kepada pengguna.



```
btnOk.addActionListener(e -> {
    if (timer != null) timer.stop();
    try {
        LocalDateTime keluar = LocalDateTime.now();
        Duration durasi = Duration.between(waktuMasuk, keluar);
        long totalMenit = durasi.toMinutes();

        // Pembulatan ke atas dan minimal 1 jam
        long jamDibulatkan = (long) Math.ceil(totalMenit / 60.0);
        if (jamDibulatkan == 0) jamDibulatkan = 1;

        // Hitung tarif berdasarkan jenis kendaraan
        long tarif = tfJenisKendaraan.getText().equalsIgnoreCase(anotherString:"Mobil") ? 5000 : 3000;
        totalBiaya = jamDibulatkan * tarif;
        tfDurasi.setText(jamDibulatkan + " jam");
        tfTotal.setText("Rp " + totalBiaya);

        try (Connection conn = DatabaseConnection.getConnection()) {
            String update = "UPDATE parkir SET tanggal_keluar=?, durasi=?, total=? WHERE nomor_tiket=?";
            PreparedStatement stmt = conn.prepareStatement(update);
            stmt.setDate(parameterIndex:1, java.sql.Date.valueOf(keluar.toLocalDate()));
            stmt.setString(parameterIndex:2, tfDurasi.getText());
            stmt.setLong(parameterIndex:3, totalBiaya);
            stmt.setString(parameterIndex:4, tfNomorTiket.getText());
            stmt.executeUpdate();
            JOptionPane.showMessageDialog(this, message:"Data parkir keluar berhasil disimpan.");
            loadTableData();
        }
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(this, message:"Gagal menghitung durasi atau menyimpan data.");
    }
});
```

Setelah data kendaraan ditampilkan, pengguna dapat menekan tombol "OK" untuk menghentikan timer dan menghitung durasi parkir serta total biaya berdasarkan jenis kendaraan. Durasi dihitung dalam jam, dengan pembulatan ke atas dan minimal satu jam. Tarif per jam ditentukan

berdasarkan jenis kendaraan: Rp5.000 untuk mobil dan Rp3.000 untuk motor. Total biaya kemudian ditampilkan di tfTotal. Tombol "Bayar" memungkinkan pengguna memasukkan jumlah pembayaran melalui dialog input. Jika pembayaran mencukupi, tombol "Cetak Struk" dapat digunakan untuk mencetak struk pembayaran, yang mencakup detail seperti nomor tiket, nomor polisi, jenis kendaraan, tanggal dan jam masuk serta keluar, durasi, total biaya, pembayaran, dan kembalian.

```
221         btnCetakStruk.addActionListener(e -> {
222             if (tfPembayaran.getText().isEmpty()) {
223                 JOptionPane.showMessageDialog(this, message:"Masukkan jumlah pembayaran!");
224                 return;
225             }
226 
227             try {
228                 long pembayaran = Long.parseLong(tfPembayaran.getText());
229                 if (pembayaran < totalBiaya) {
230                     JOptionPane.showMessageDialog(this, message:"Pembayaran kurang!");
231                     return;
232                 }
233 
234                 long kembalian = pembayaran - totalBiaya;
235                 CetakStruk.cetakStruk(
236                     tfNomorTiket.getText(),
237                     tfNomorPolisi.getText(),
238                     tfJenisKendaraan.getText(),
239                     tfTanggalMasuk.getText(),
240                     tfJamMasuk.getText(),
241                     tfTanggalKeluar.getText(),
242                     tfDurasi.getText(),
243                     tfTotal.getText(),
244                     String.format(format:"Rp %d", pembayaran),
245                     String.format(format:"Rp %d", kembalian)
246                 );
247             } catch (NumberFormatException ex) {
248                 JOptionPane.showMessageDialog(this, message:"Masukkan pembayaran dengan angka yang valid!");
249             }
250         });
251 
```

Sebuah ActionListener ditambahkan pada tombol "Cetak Struk" untuk menangani aksi klik dari pengguna. Ketika tombol diklik, aplikasi akan memeriksa apakah pengguna telah memasukkan jumlah pembayaran. Jika belum, sebuah pesan akan ditampilkan meminta pengguna untuk memasukkan jumlah pembayaran. Jika jumlah pembayaran kurang dari total biaya, sebuah pesan akan ditampilkan memberitahukan pengguna bahwa pembayaran kurang. Jika pembayaran cukup, aplikasi akan menghitung kembalian dan mencetak struk pembayaran menggunakan metode CetakStruk.cetakStruk.

```
252         btnKembali.addActionListener(e -> {
253             dispose();
254             new MenuUtamaParkir().setVisible(b:true);
255         });
256 
```

Sebuah ActionListener ditambahkan pada tombol "Kembali Ke Menu Utama" untuk menangani aksi klik dari pengguna. Ketika tombol diklik, jendela saat ini akan ditutup dan menu utama aplikasi parkir akan ditampilkan kembali.

```
257             panelUtama.add(panelForm, BorderLayout.WEST);
258             panelUtama.add(scrollPane, BorderLayout.CENTER);
259             setContentPane(panelUtama);
260         }
261
```

Panel form yang telah disiapkan sebelumnya ditambahkan ke dalam panel utama (panelUtama) di bagian kiri, sementara JScrollPane yang berisi tabel data ditambahkan ke dalam panel utama di bagian tengah. Dengan demikian, antarmuka pengguna akan menampilkan form input di sebelah kiri dan tabel data di sebelah kanan.

```
262     private void loadTableData() {
263         try (Connection conn = DatabaseConnection.getConnection()) {
264             tableModel.setRowCount(rowCount:0);
265             Statement stmt = conn.createStatement();
266             ResultSet rs = stmt.executeQuery(sql:"SELECT * FROM parkir");
267             while (rs.next()) {
268                 Object[] data = {
269                         rs.getString(columnLabel:"nomor_tiket"),
270                         rs.getString(columnLabel:"nomor_polisi"),
271                         rs.getString(columnLabel:"jenis_kendaraan"),
272                         rs.getString(columnLabel:"tanggal_masuk"),
273                         rs.getString(columnLabel:"jam_masuk"),
274                         rs.getString(columnLabel:"tanggal_keluar") != null ? rs.getString(columnLabel:"tanggal_keluar") : "",
275                         rs.getString(columnLabel:"durasi") != null ? rs.getString(columnLabel:"durasi") : ""
276                 };
277                 tableModel.addRow(data);
278             }
279         } catch (Exception e) {
280             JOptionPane.showMessageDialog(this, message:"Gagal memuat data parkir keluar dari database.");
281         }
282     }
```

Terakhir, jendela utama aplikasi (JFrame) disiapkan dengan judul "Parkir Keluar", ukuran maksimal, dan ukuran minimum tertentu. Jendela ini diset untuk menutup aplikasi saat ditutup dan ditampilkan di tengah layar. Panel utama yang telah disusun sebelumnya ditetapkan sebagai konten dari jendela, dan jendela ditampilkan kepada pengguna.

```
262     private void loadTableData() {
263         try (Connection conn = DatabaseConnection.getConnection()) {
264             tableModel.setRowCount(rowCount:0);
265             Statement stmt = conn.createStatement();
266             ResultSet rs = stmt.executeQuery(sql:"SELECT * FROM parkir");
267             while (rs.next()) {
268                 Object[] data = {
269                         rs.getString(columnLabel:"nomor_tiket"),
270                         rs.getString(columnLabel:"nomor_polisi"),
271                         rs.getString(columnLabel:"jenis_kendaraan"),
272                         rs.getString(columnLabel:"tanggal_masuk"),
273                         rs.getString(columnLabel:"jam_masuk"),
274                         rs.getString(columnLabel:"tanggal_keluar") != null ? rs.getString(columnLabel:"tanggal_keluar") : "",
275                         rs.getString(columnLabel:"durasi") != null ? rs.getString(columnLabel:"durasi") : ""
276                 };
277                 tableModel.addRow(data);
278             }
279         } catch (Exception e) {
280             JOptionPane.showMessageDialog(this, message:"Gagal memuat data parkir keluar dari database.");
281         }
282     }
```

Metode ini bertanggung jawab untuk memuat data kendaraan yang telah keluar dari database dan menampilkannya di tabel. Pertama, model tabel dikosongkan untuk menghindari duplikasi data. Kemudian, koneksi ke database dibuka, dan query dijalankan untuk mengambil semua data dari tabel parkir. Setiap hasil query diubah menjadi array objek dan ditambahkan ke model tabel. Jika terjadi kesalahan selama proses ini, pesan kesalahan akan ditampilkan kepada pengguna.

```
    Run | Debug | Run main | Debug main
284     public static void main(String[] args) {
285         SwingUtilities.invokeLater(() -> new ParkirKeluar().setVisible(b:true));
286     }
287 }
288
```

Metode main berfungsi sebagai titik masuk utama aplikasi. Di dalamnya, SwingUtilities.invokeLater digunakan untuk memastikan bahwa pembuatan dan penampilan jendela ParkirKeluar dilakukan di thread yang sesuai untuk GUI, yaitu Event Dispatch Thread (EDT).

5.1.5 Class CetakTiket.java

```
1   package Parkir1;
2
3   import com.itextpdf.text.*;
4   import com.itextpdf.text.pdf.*;
5   import java.io.FileOutputStream;
6   import java.time.LocalDate;
7   import java.time.LocalTime;
8   import java.time.format.DateTimeFormatter;
9
```

Baris 1 menyatakan bahwa file CetakTiket.java merupakan bagian dari package Parkir1. Baris 3-4 mengimpor class dari library iText, sebuah

pustaka pihak ketiga yang digunakan untuk membuat dan memanipulasi file PDF. Baris 5-8 mengimpor class untuk penanganan file (FileOutputStream) dan tanggal/waktu (LocalDate, LocalTime, dan DateTimeFormatter). Ini penting dalam konteks aplikasi parkir karena informasi waktu dan tanggal sangat krusial untuk mencetak tiket parkir yang akurat.

```
10  public class CetakTiket {
11      public static void cetakTiket(String nomorTiket, String nomorPolisi, String jenisKendaraan,
        String tanggal, String jam) {
12          Document document = new Document(PageSize.A6);
13          try {
14              String fileName = "Tiket_" + nomorTiket + ".pdf";
15              PdfWriter.getInstance(document, new FileOutputStream(fileName));
16              document.open();
17
```

Baris 10 mendefinisikan sebuah class publik bernama CetakTiket. Baris 11 merupakan method yang bersifat public agar bisa diakses dari class lain, dan static agar bisa dipanggil tanpa harus membuat objek CetakTiket. Ini menunjukkan konsep method sharing antar class. Konsep encapsulation diterapkan dengan hanya membatasi inputan melalui parameter method. Baris 12 membuat objek Document dari library iText dengan ukuran kertas A6, ukuran kecil yang umum digunakan untuk tiket. Baris 13 blok try digunakan untuk membungkus proses yang berpotensi menimbulkan error, seperti menulis ke file. Baris 14 membuat nama file PDF berdasarkan nomor tiket. Baris 15 menginisialisasi writer yang akan menulis isi dokumen ke file PDF. FileOutputStream digunakan sebagai medium output ke sistem file. Baris 16 membuka dokumen PDF agar bisa ditulis. Ini adalah method dari objek Document yang menerapkan prinsip OOP yaitu pengaksesan objek melalui method publik.

```
18              Font fontHeader = new Font(Font.FontFamily.COURIER, size:12,
                Font.BOLD);
19              Font fontText = new Font(Font.FontFamily.COURIER, size:10, Font.
                NORMAL);
20
```

Baris 18-19 membuat objek Font dengan berbagai ukuran dan ketebalan. Ini adalah bentuk penerapan object configuration, di mana satu class memiliki banyak properti yang bisa diatur saat instansiasi (polimorfisme melalui overloading constructor).

```
21          Paragraph title = new Paragraph(string:"TIKET PARKIR", fontHeader);
22          title.setAlignment(Element.ALIGN_CENTER);
23          document.add(title);
24
25          Paragraph subTitle = new Paragraph(string:"UNIVERSITAS SINGAPERBANGSA KARAWANG",
            fontHeader);
26          subTitle.setAlignment(Element.ALIGN_CENTER);
27          document.add(subTitle);
28
29          document.add(new Paragraph(string:"==================================",
            fontText));
30          document.add(new Paragraph("No       : " + nomorTiket, fontText));
31          document.add(new Paragraph("Plat     : " + nomorPolisi, fontText));
32          document.add(new Paragraph("Jenis    : " + jenisKendaraan, fontText));
33          document.add(new Paragraph("Tanggal  : " + tanggal, fontText));
34          document.add(new Paragraph("Jam      : " + jam, fontText));
35          document.add(new Paragraph(string:"==================================",
            fontText));
36
37          Paragraph thankYou = new Paragraph(string:"Terima Kasih", fontText);
38          thankYou.setAlignment(Element.ALIGN_CENTER);
39          document.add(thankYou);
40
```

Blok ini menyusun isi dari dokumen tiket. Baris 21–23 menambahkan judul “TIKET PARKIR” ke dokumen, dipusatkan dengan setAlignment. Baris 25–27 menambahkan subjudul “UNIVERSITAS SINGAPERBANGSA KARAWANG” dengan format serupa. Bagian ini menunjukkan pemanfaatan object reuse dan abstraction, di mana objek Paragraph dari iText mengelola teks secara modular. Baris 29–35 mencetak garis pemisah dan detail tiket (nomor tiket, plat nomor, jenis kendaraan, tanggal, jam) ke dalam dokumen. Setiap informasi dicetak sebagai Paragraph, dan seluruh proses ini mencerminkan konsep modularisasi dalam OOP. Baris 37–39 menambahkan ucapan “Terima Kasih” di akhir dokumen dan memposisikannya di tengah.

```
41              } catch (Exception e) {
42                  e.printStackTrace();
43              } finally {
44                  document.close();
45              }
46          }
47      }
```

Blok ini merupakan bagian akhir dari proses penulisan file PDF dan terdiri dari dua bagian penting: penanganan error (catch) dan penutupan sumber daya (finally). Baris 41 menggunakan catch (Exception e) untuk menangkap semua jenis kesalahan (misalnya kesalahan file system, error saat menulis ke file PDF, atau kesalahan internal dari library iText). Baris 42 mencetak detail error ke konsol dengan e.printStackTrace(), agar

programmer bisa menganalisis sumber masalah. Blok finally (baris 43–45) akan dieksekusi selalu, baik ketika proses try berhasil atau ketika terjadi exception. Di dalamnya, document.close() dipanggil untuk menutup file PDF yang telah dibuka sebelumnya.

5.1.6 Class CetakStruk.java

```
1   package Parkir1;
2
3   import com.itextpdf.text.*;
4   import com.itextpdf.text.pdf.*;
5
6   import java.io.FileOutputStream;
7
8   public class CetakStruk {
9       public static void cetakStruk(String nomorTiket, String nomorPolisi, String jenisKendaraan,
10                                    String tanggalMasuk, String jamMasuk, String tanggalKeluar,
11                                    String durasi, String total, String pembayaran, String kembalian) {
```

Kelas CetakStruk merupakan bagian dari package Parkir1 dan berfungsi untuk mencetak struk parkir dalam bentuk file PDF menggunakan library eksternal iText.

Di dalamnya terdapat metode statis cetakStruk() yang menerima sejumlah parameter seperti nomor tiket, nomor polisi, jenis kendaraan, tanggal dan jam masuk, tanggal keluar, durasi parkir, total biaya, jumlah pembayaran, dan kembalian. File PDF akan disimpan dengan nama "Struk_[nomorTiket].pdf" dan memiliki ukuran kertas A6, yang cocok untuk ukuran struk fisik.

```
12          Document document = new Document(PageSize.A6);
13          try {
14              String fileName = "Struk_" + nomorTiket + ".pdf";
15              PdfWriter.getInstance(document, new FileOutputStream(fileName));
16              document.open();
17
18              Font fontHeader = new Font(Font.FontFamily.COURIER, size:12, Font.BOLD);
19              Font fontText = new Font(Font.FontFamily.COURIER, size:10, Font.NORMAL);
```

Metode ini pertama-tama membuat objek Document dengan ukuran kertas A6, lalu PdfWriter.getInstance() digunakan untuk menghubungkan objek document dengan aliran keluaran file (FileOutputStream). Setelah itu, document.open() dipanggil untuk mulai menulis isi PDF. Dua jenis font ditentukan: satu untuk header (font Courier ukuran 12 dan tebal), dan satu untuk teks biasa (Courier ukuran 10).

```
21          Paragraph title = new Paragraph(string:"STRUK PARKIR", fontHeader);
22          title.setAlignment(Element.ALIGN_CENTER);
23          document.add(title);
24
25          Paragraph subTitle = new Paragraph(string:"UNIVERSITAS SINGAPERBANGSA KARAWANG", fontHeader);
26          subTitle.setAlignment(Element.ALIGN_CENTER);
27          document.add(subTitle);
```

Konten struk terdiri dari beberapa bagian. Pertama, terdapat judul "STRUK PARKIR" dan subjudul "UNIVERSITAS SINGAPERBANGSA KARAWANG" yang ditampilkan di tengah. Kemudian terdapat garis pembatas dan informasi kendaraan serta transaksi seperti nomor tiket, plat kendaraan, jenis kendaraan, waktu masuk dan keluar, durasi parkir, total biaya, pembayaran yang diberikan, dan jumlah kembalian. Semua informasi ini ditampilkan dengan perataan kiri menggunakan objek Paragraph yang ditambahkan satu per satu ke dalam dokumen.

```
29          document.add(new Paragraph(string:"==================================", fontText));
30          document.add(new Paragraph("No          : " + nomorTiket, fontText));
31          document.add(new Paragraph("Plat        : " + nomorPolisi, fontText));
32          document.add(new Paragraph("Jenis       : " + jenisKendaraan, fontText));
33          document.add(new Paragraph("Masuk       : " + tanggalMasuk + " " + jamMasuk, fontText));
34          document.add(new Paragraph("Keluar      : " + tanggalKeluar, fontText));
35          document.add(new Paragraph("Durasi      : " + durasi, fontText));
36          document.add(new Paragraph("Total       : " + total, fontText));
37          document.add(new Paragraph("Pembayaran  : " + pembayaran, fontText));
38          document.add(new Paragraph("Kembalian   : " + kembalian, fontText));
39          document.add(new Paragraph(string:"==================================", fontText));
40
41          Paragraph thankYou = new Paragraph(string:"Terima Kasih", fontText);
42          thankYou.setAlignment(Element.ALIGN_CENTER);
43          document.add(thankYou);
```

Setelah semua informasi ditambahkan, sistem juga menambahkan ucapan "Terima Kasih" yang diratakan ke tengah sebagai penutup struk.

```
44
45              } catch (Exception e) {
46                  e.printStackTrace();
47              } finally {
48                  document.close();
49              }
50          }
51      }
```

Apabila terjadi kesalahan saat proses penulisan, blok catch akan menangkap dan mencetak pesan error. Akhirnya, pada blok finally, dokumen ditutup secara eksplisit menggunakan document.close() agar file PDF tersimpan dengan baik.

5.1.7 Class DatabaseConnection.java

```
1  package Parkir1;
2
3  import java.sql.Connection;
4  import java.sql.DriverManager;
5  import java.sql.SQLException;
6
```

Baris 1 mendeklarasikan bahwa class DatabaseConnection berada dalam package bernama Parkir1. Baris 3-4 merupakan import yang memanggil class dari package java.sql yang dibutuhkan untuk mengelola koneksi ke database MySQL. Class Connection digunakan sebagai objek penghubung antara Java dengan database, DriverManager adalah class utilitas yang menyediakan metode statis untuk membuat koneksi, dan SQLException digunakan untuk menangani kesalahan (exception) yang berhubungan dengan database. Ini memperlihatkan penggunaan abstraksi dan exception handling dalam Java OOP.

```
7  public class DatabaseConnection {
8      private static final String URL = "jdbc:mysql://localhost:3306/parkir_db";
9      private static final String USER = "root";
10     private static final String PASSWORD = "";
11
```

Baris 5–8 mendefinisikan class DatabaseConnection, sebuah class utilitas berperan tunggal (single responsibility) untuk menyediakan koneksi ke database. Di dalamnya terdapat tiga field statis URL, USER, dan PASSWORD yang menyimpan informasi koneksi database. Akses modifier private dan keyword static final digunakan untuk menjaga keamanan dan konsistensi data (mendukung prinsip encapsulation), serta memungkinkan nilai tersebut digunakan langsung dalam method statis tanpa perlu instance.

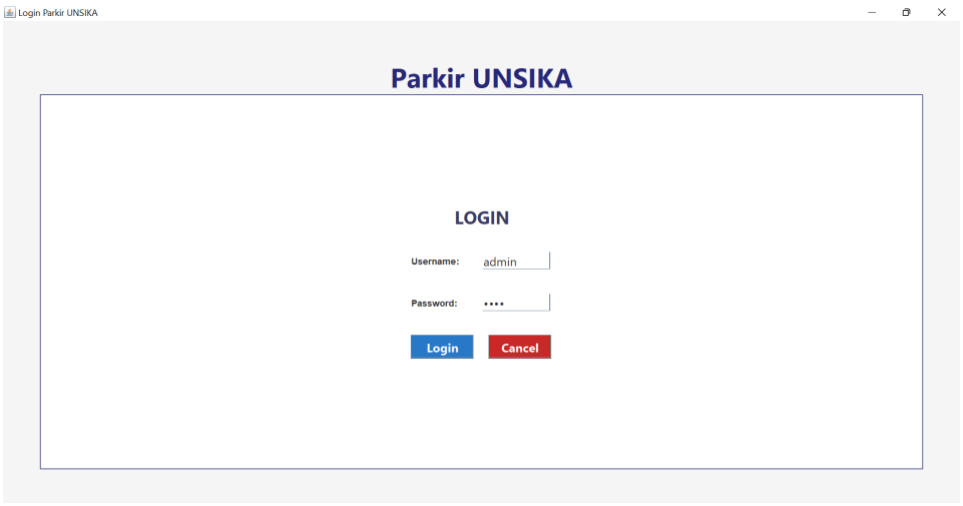
```
12     public static Connection getConnection() throws SQLException {
13         return DriverManager.getConnection(URL, USER, PASSWORD);
14     }
15 }
```

Baris 12-15, method getConnection() adalah method public static yang mengembalikan objek Connection. Karena static, method ini bisa dipanggil langsung tanpa membuat instance dari class DatabaseConnection. Di dalamnya, DriverManager.getConnection(...) dipanggil dengan tiga parameter (URL, USER, PASSWORD) untuk membuka koneksi ke

MySQL database. Bila koneksi berhasil, method ini akan mengembalikan objek Connection kepada pemanggilnya, memungkinkan class lain untuk mengakses dan berinteraksi dengan database.

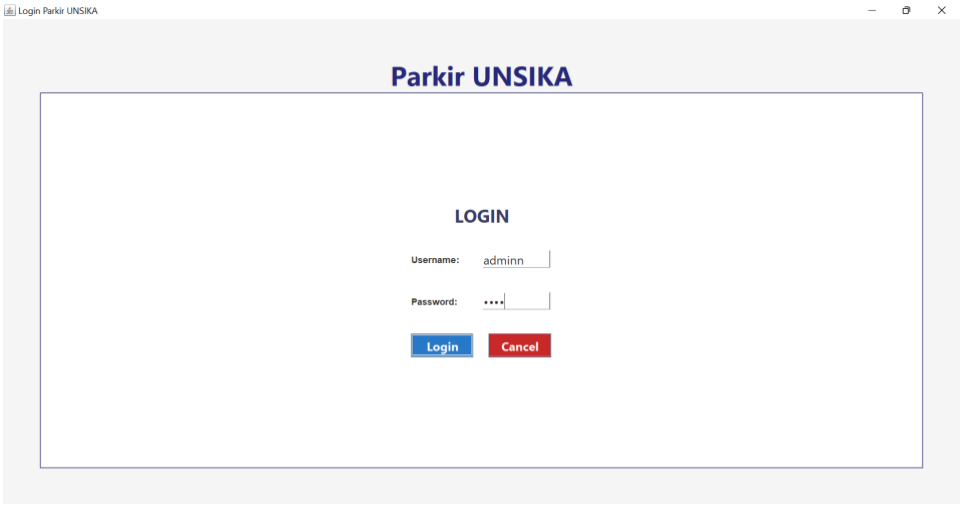
5.2 Pengujian

5.2.1 Login

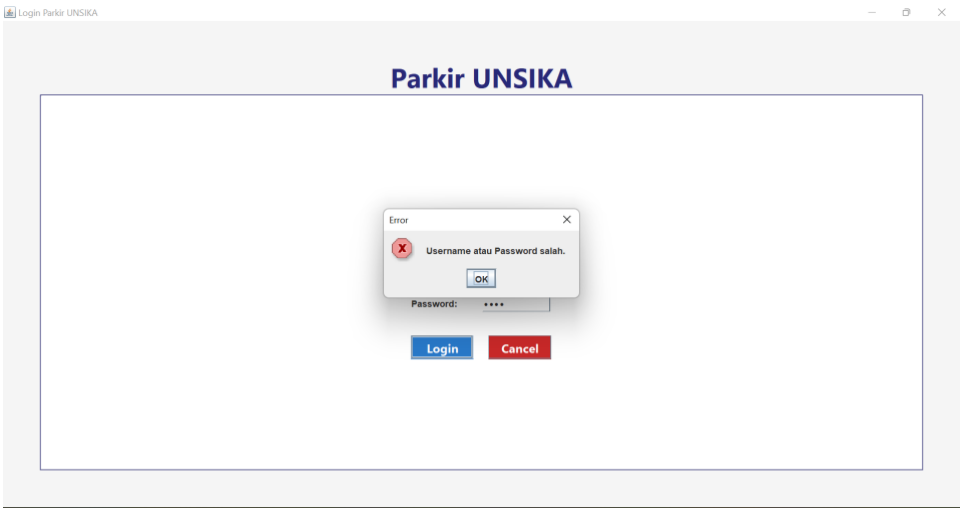


Gambar 5.2.1.1: Pengujian Login dengan Username & Password (Benar)

Gambar 5.2.1.2: Pop Up Login Berhasil

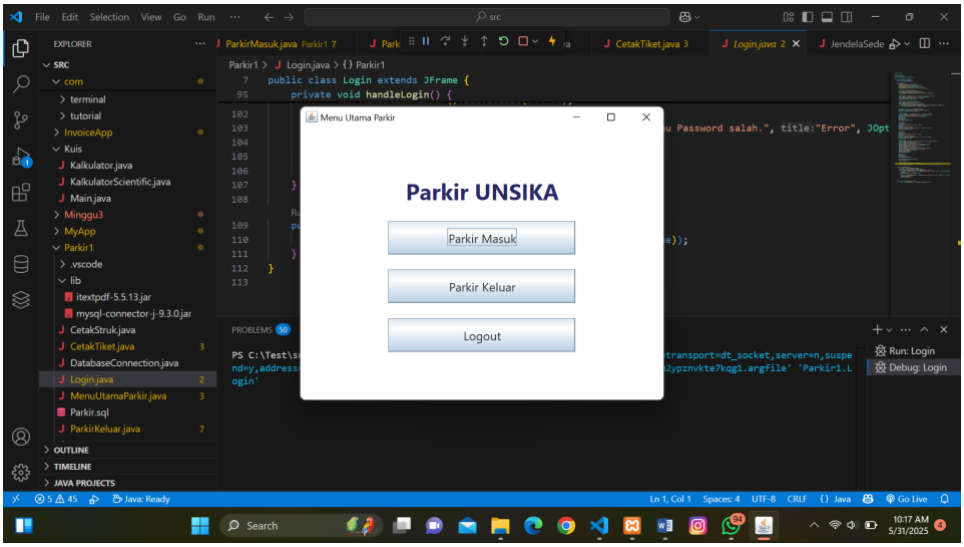


Gambar 5.2.1.3: Pengujian Login dengan Usrname & Password (Salah)



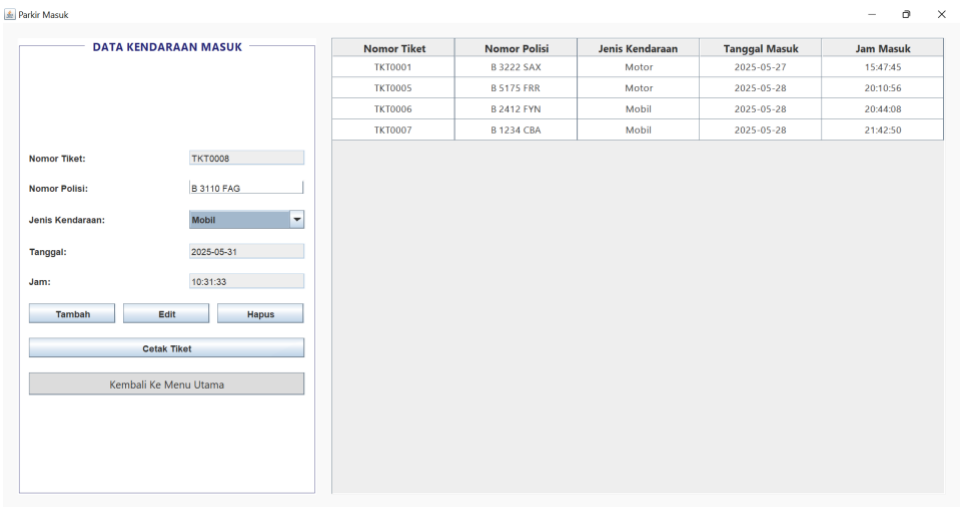
Gambar 5.2.1.4: Pop Up Login Gagal

5.2.2 Menu Utama Parkir

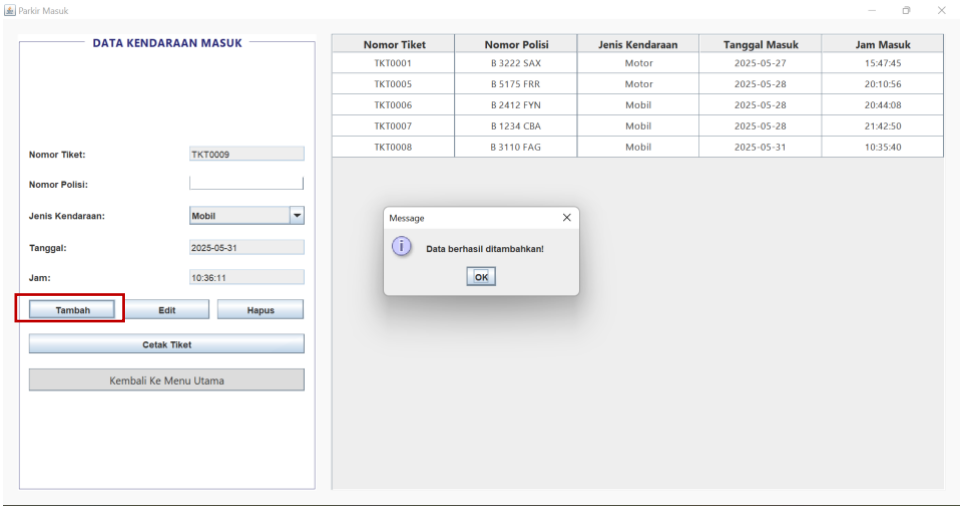


Gambar 5.2.2.1: Menu Utama Parkir

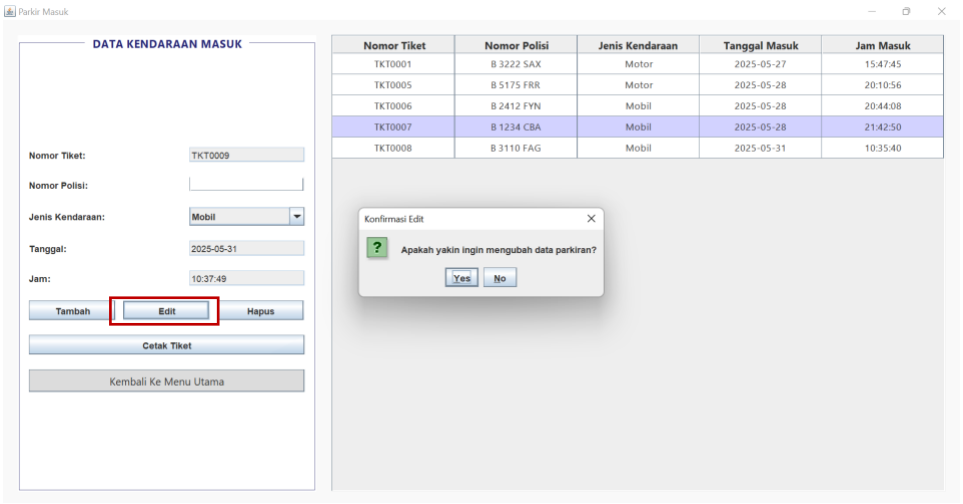
5.2.3 Parkir Masuk



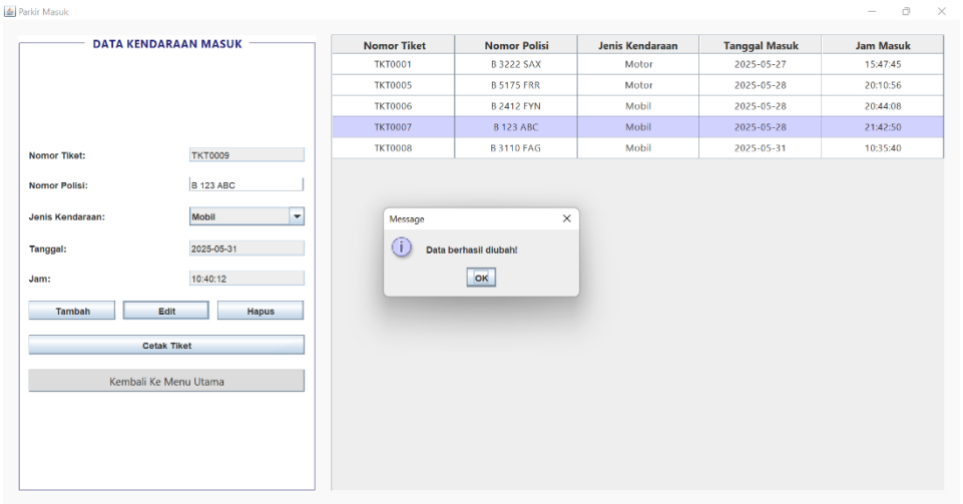
Gambar 5.2.3.1: Tampilan Parkir Masuk



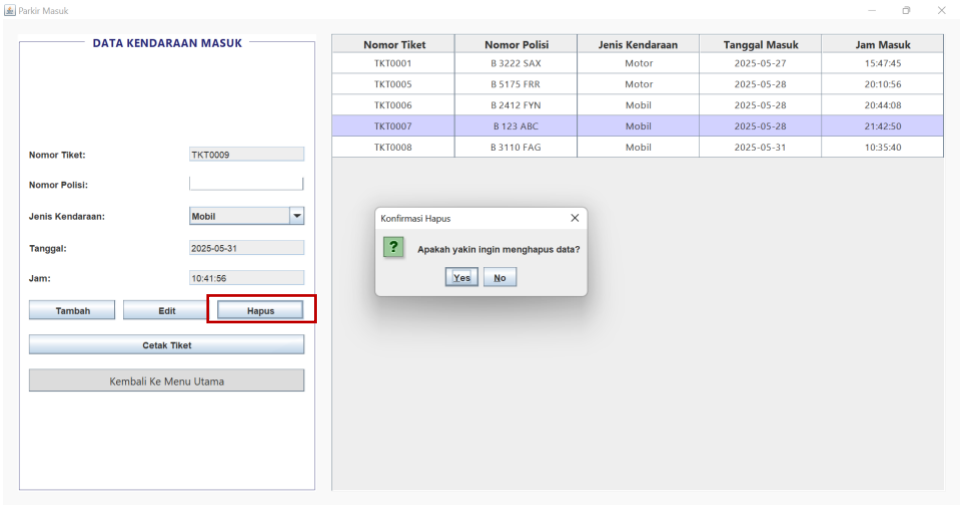
Gambar 5.2.3.2: Pop Up Data Berhasil ditambahkan Ketika Button Tambah diklik



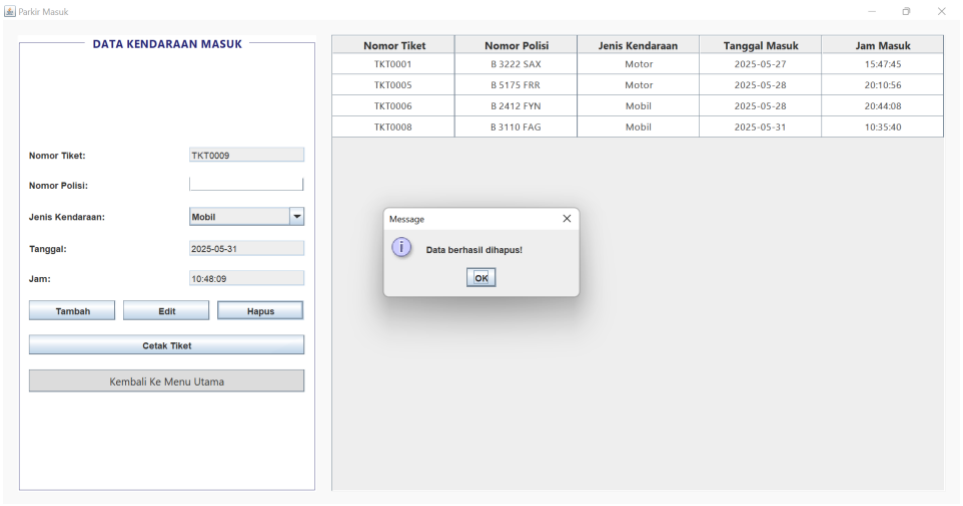
Gambar 5.2.3.3: Pop Up Konfirmasi Edit Ketika Button Edit diklik



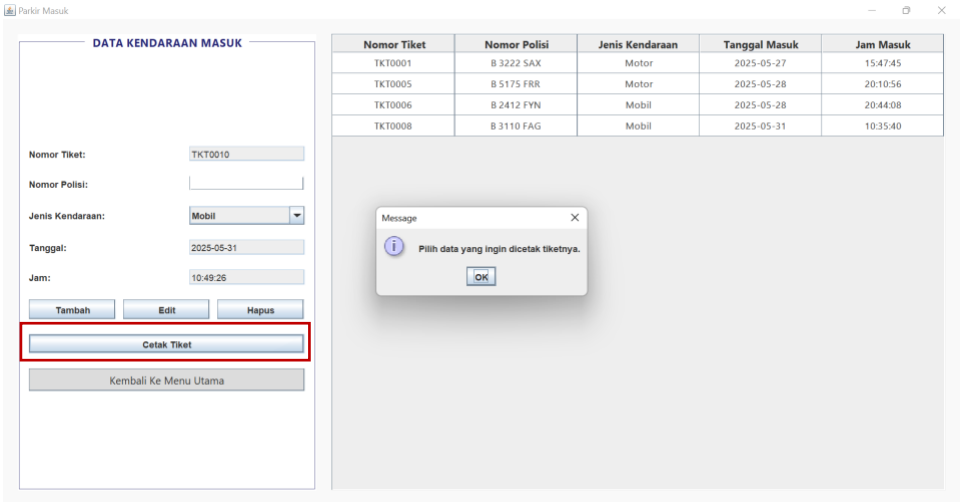
Gambar 5.2.3.4: Pop Up Data Berhasil diubah



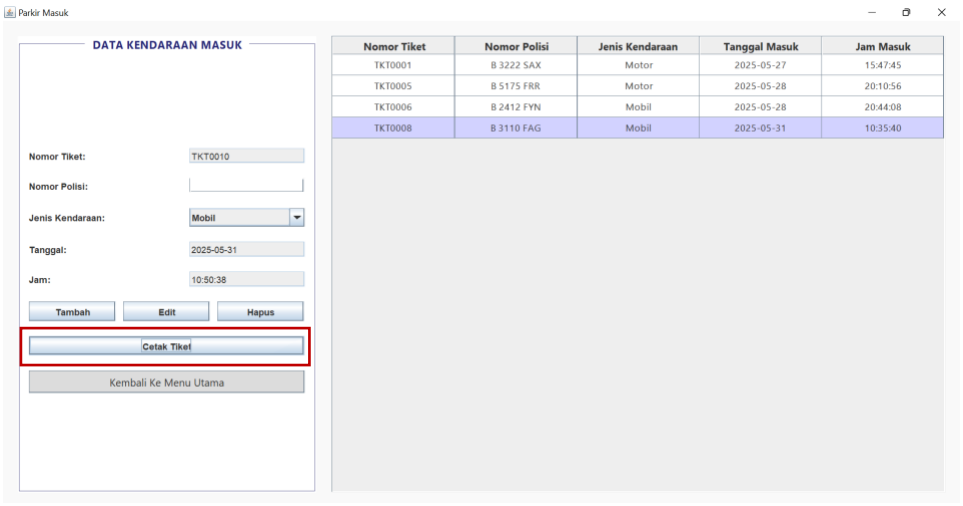
Gambar 5.2.3.5: Pop Up Konfirmasi Hapus Ketika Button Hapus diklik



Gambar 5.2.3.6: Pop Up Data Berhasil dihapus

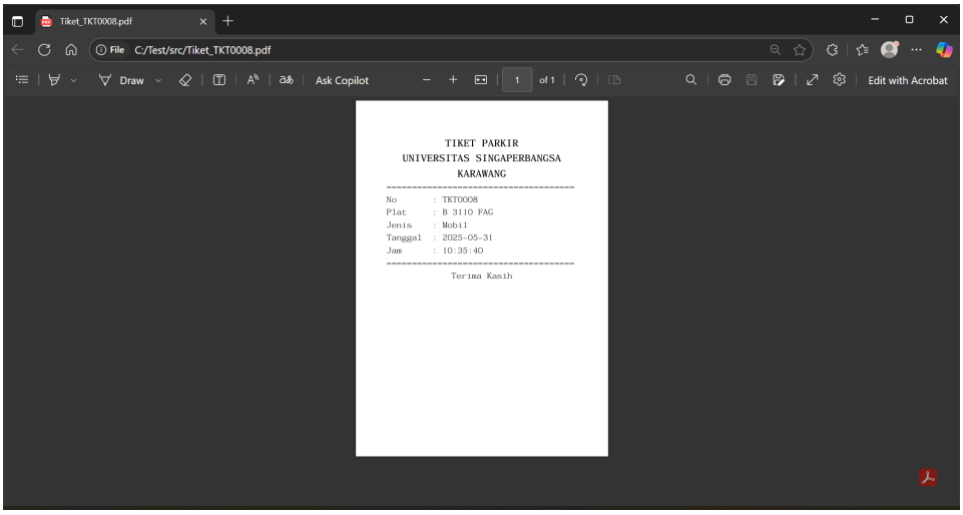


Gambar 5.2.3.7: Pop Up Pilih Data Ketika Button Cetak Tiket diklik dan Belum Ada Data yang dipilih



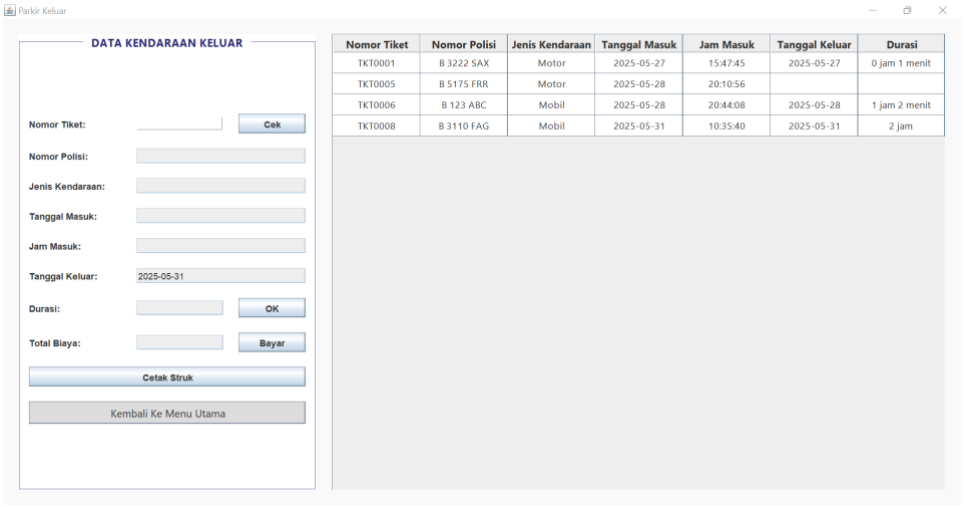
Gambar 5.2.3.8: Button Cetak Tiket untuk Menyimpan PDF Tiket Parkir

5.2.4 Tiket Parkir Masuk

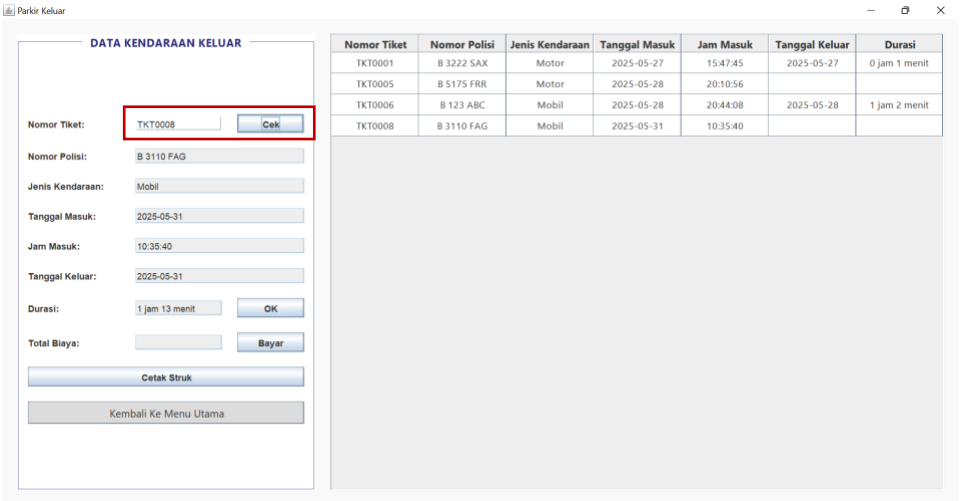


Gambar 5.2.4.1: Tampilan PDF Tiket Parkir

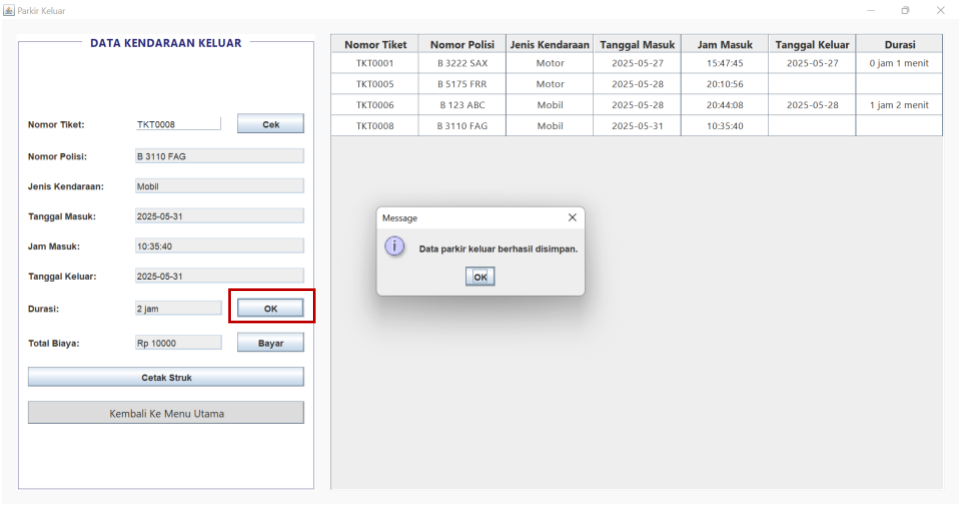
5.2.5 Parkir Keluar



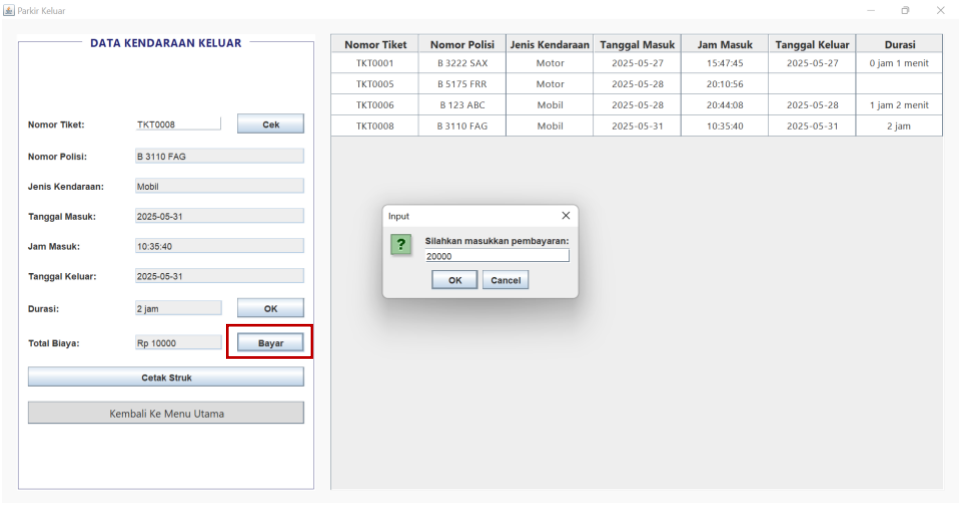
Gambar 5.2.5.1: Tampilan Parkir Keluar



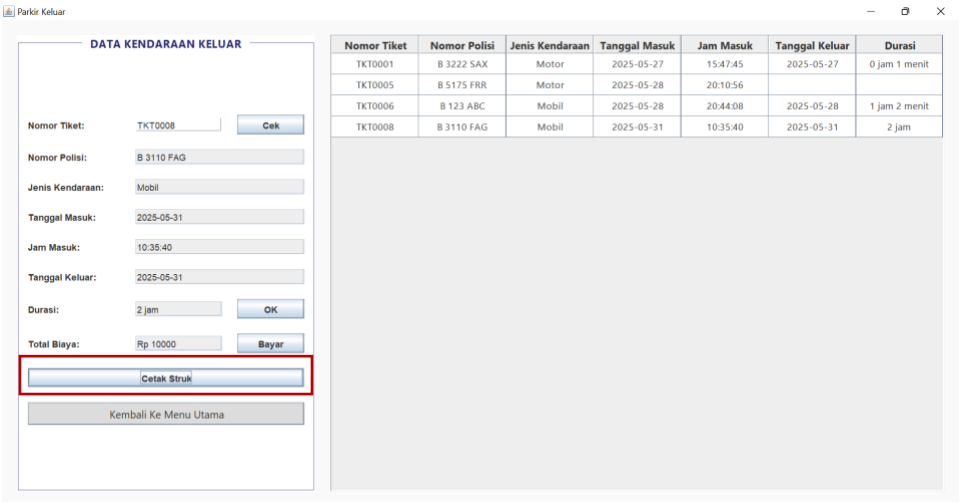
Gambar 5.2.5.2: Tampilan Parkir Keluar Ketika Mengisi Nomor Tiket dan Klik Button Cek



Gambar 5.2.5.3: Pop Up Data Berhasil disimpan ke Tabel Ketika Button OK diklik

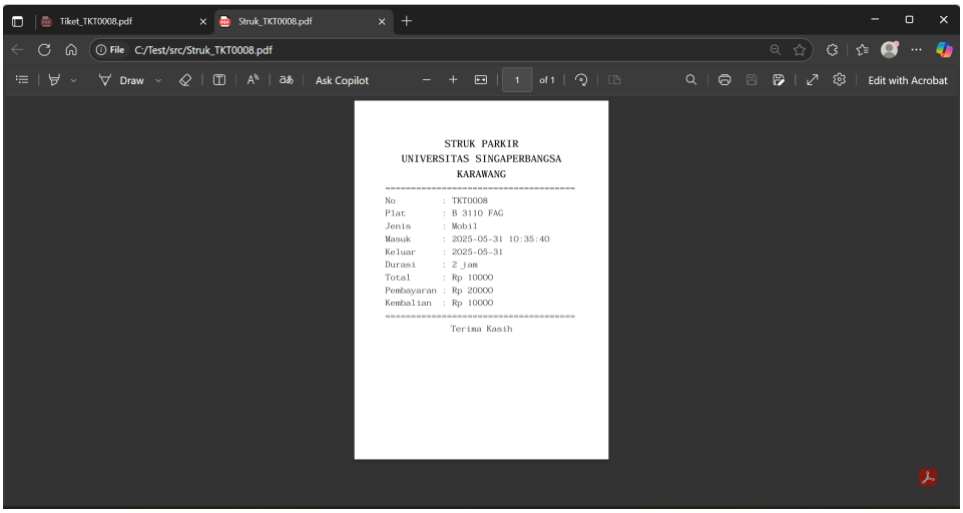


Gambar 5.2.5.4: Input Pembayaran Ketika Button Bayar diklik



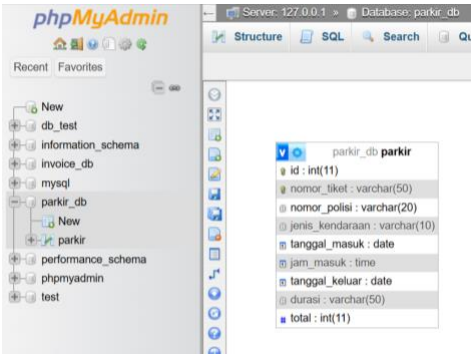
Gambar 5.2.5.5: Button Cetak Struk untuk Menyimpan PDF Struk Parkir

5.2.6 Struk Parkir

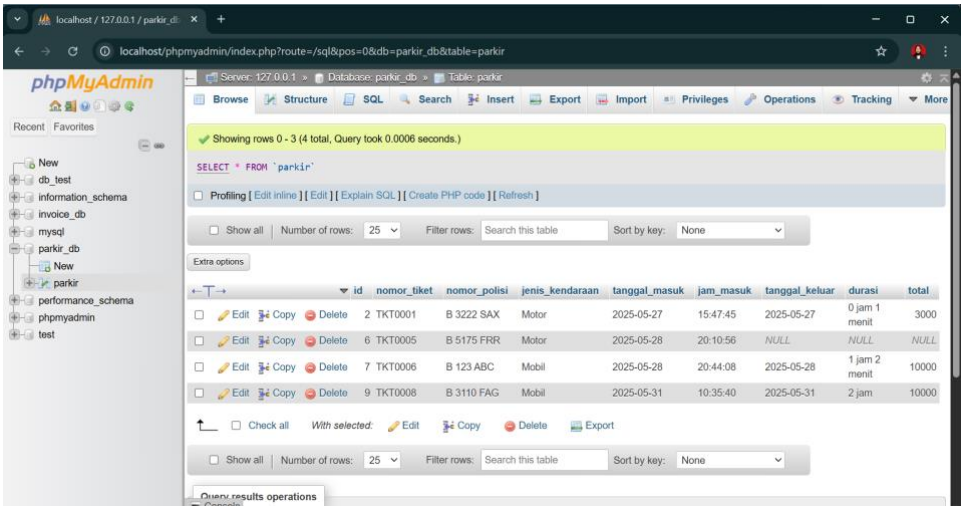


Gambar 5.2.6.1: Tampilan PDF Struk Parkir

5.2.7 Koneksi Database



Gambar 5.2.8.1: Tabel Parkir pada Database db_parkir



Gambar 5.2.8.2: Isi Tabel Parkir

BAB VI
KESIMPULAN

6.1 Kesimpulan

Berdasarkan laporan dan kode program yang telah dibuat, dapat disimpulkan bahwa aplikasi Sistem Parkir Berbasis Java GUI Terhubung ke Database MySQL ini berhasil diimplementasikan dengan baik menggunakan prinsip-prinsip Object Oriented Programming (OOP). Aplikasi ini terdiri dari berbagai fitur utama seperti pencatatan data kendaraan masuk dan keluar, penghitungan durasi parkir dan biaya, serta pencetakan tiket dan struk dalam format PDF.

Setiap file program memanfaatkan konsep OOP secara tepat, seperti penggunaan class untuk membungkus data dan logika, method untuk menjalankan fungsi tertentu, serta object sebagai representasi nyata dari entitas data dalam program. Selain itu, aplikasi ini juga menunjukkan penerapan enkapsulasi melalui penggunaan modifier akses pada variabel dan method, inheritance melalui pewarisan dari kelas bawaan Java seperti JFrame dan JPanel, serta polimorfisme dalam bentuk overriding method GUI dan event handling.

Penerapan exception handling juga dimanfaatkan untuk menangani kesalahan saat koneksi ke database dan proses input. Dengan desain antarmuka yang sederhana namun fungsional, aplikasi ini mampu memudahkan proses administrasi parkir secara efisien dan sistematis, serta memberikan pengalaman pengguna yang baik bagi operator parkir.