

**LAPORAN UJIAN AKHIR SEMESTER  
PEMROGRAMAN BERORIENTASI OBJEK**

**“Program Rental Property Event”**



**Dosen Pengampu:**

Taufik Ridwan., S.T., M.T.

**Disusun Oleh Kelompok 3:**

Dava Wahyu Erlangga	(2310631250011)
Erlin Dwi Haryanti	(2310631250050)
Faricha Dillia Dinda Thomas	(2310631250054)
Resya Hidayatunnisa	(2310631250104)

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

## DAFTAR ISI

<b>DAFTAR ISI .....</b>	<b>ii</b>
<b>BAB I .....</b>	<b>1</b>
<b>PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah .....	1
1.3 Tujuan Laporan .....	2
<b>BAB II.....</b>	<b>3</b>
<b>HASIL &amp; PEMBAHASAN .....</b>	<b>3</b>
2.1 Fitur Program Rental Property Event .....	3
2.1.1 General Section .....	3
2.1.2 Admin Section .....	3
2.1.3 Customer Section.....	4
2.2 Konsep OOP pada Kode Program Rental Property Event .....	5
2.2.1 General Section .....	5
2.2.2 Admin Section .....	7
2.2.3 Customer Section.....	17
2.3 Perancangan Program dengan UML .....	24
2.3.1 Use Case Diagram .....	24
2.3.2 Class Diagram .....	24
2.3.3 Entity Relationship Diagram .....	25
2.4 Implementasi Kode Program & Pengujian.....	26
2.4.1 General Section .....	26
2.4.2 Admin Section .....	36
2.4.3 Customer Section.....	46
<b>BAB III .....</b>	<b>73</b>
<b>KESIMPULAN .....</b>	<b>73</b>

## **BAB I**

### **PENDAHULUAN**

#### **1.1 Latar Belakang**

Penyelenggaraan acara seperti pernikahan, seminar, konser, dan pesta lainnya membutuhkan berbagai properti pendukung seperti tenda, kursi, meja, sound system, dan perlengkapan dekorasi. Karena sifat penggunaannya yang sementara, menyewa menjadi pilihan yang lebih efisien dibanding membeli. Hal ini menjadikan bisnis rental properti event semakin diminati dan berkembang pesat. Namun, dalam praktiknya, banyak penyedia layanan rental masih menggunakan sistem manual untuk mengelola data barang, pemesanan, dan pembayaran. Cara ini rentan menimbulkan berbagai masalah, seperti kesalahan pencatatan, ketidaksesuaian jadwal peminjaman, serta lambatnya proses verifikasi dan konfirmasi pesanan.

Untuk menjawab kebutuhan tersebut, dibutuhkan sistem rental berbasis komputer yang mampu mengelola seluruh proses penyewaan secara otomatis, cepat, dan terorganisir. Dengan pendekatan Pemrograman Berorientasi Objek (PBO), sistem dapat dirancang secara modular sehingga lebih mudah dikembangkan dan disesuaikan dengan kebutuhan pengguna. Pengembangan sistem ini diharapkan mampu meningkatkan efisiensi operasional, memberikan pengalaman pengguna yang lebih baik, serta menjadi sarana pembelajaran penerapan konsep OOP dalam konteks nyata dunia kerja dan industri event.

#### **1.2 Rumusan Masalah**

Berdasarkan latar belakang di atas, rumusan masalah dalam pengembangan sistem rental property event ini adalah:

- a. Bagaimana merancang sistem login yang aman dan dapat memberikan notifikasi jika login gagal?
- b. Bagaimana pengguna dapat melihat daftar dan detail barang serta memesan barang yang tersedia?
- c. Bagaimana sistem dapat menghitung total harga dan memeriksa ketersediaan barang secara otomatis sebelum pesanan dikonfirmasi?

- d. Bagaimana admin dapat mengelola data barang dan memverifikasi status pembayaran serta memperbarui status pesanan dengan mudah?

### **1.3 Tujuan Laporan**

Adapun tujuan dari pembuatan program rental property event berbasis objek ini adalah sebagai berikut:

- a. Membangun sistem rental berbasis web yang mendukung proses penyewaan secara digital dan terstruktur.
- b. Menerapkan konsep Pemrograman Berorientasi Objek (PBO) dalam pengembangan fitur-fitur utama sistem.
- c. Mempermudah pelanggan dalam mengakses informasi barang dan melakukan pemesanan.
- d. Menyediakan antarmuka bagi admin untuk mengelola inventaris, memverifikasi transaksi, dan memantau status penyewaan.

## **BAB II**

### **HASIL & PEMBAHASAN**

#### **2.1 Fitur Program Rental Property Event**

##### **2.1.1 Login and Register Section**

- Koneksi Database

Ini merupakan kode program yang berisikan logika yang mana digunakan untuk mengkoneksi java dengan database. Dan database yang digunakan adalah MySql

- Login

Fitur ini menampilkan laman untuk user ataupun admin login kedalam suatu sistem, adapun pada fitur login user atau admin diharuskan memasukkan username dan password yang telah terdaftar. Jika belum, user dipersilahkan untuk mendaftar pada laman register.

- Registration

Fitur ini akan menyajikan laman untuk user mendaftarkan diri, yang mana pada laman registrasi terdapat form nama lengkap, username, password, alamat dan no telepon, jika data pribadi telah diisi semua dan sudah lengkap, maka berikutnya pengguna bisa klik daftar. Jika daftar berhasil, maka pengguna bisa melakukan login pada laman login

- Forget Pasword

Fitur ini menampilkan laman jika pengguna lupa dengan password yang saat ini dimiliki, pengguna hanya perlu memasukkan username dan reset password saja, kemudian jika berhasil pengguna dapat login dengan password yang baru.

- Logout

Fitur ini ada pada dashboard, jika pengguna sudah selesai melakukan aktivitas sewa barang properti, pengguna bisa logout dari laman dashboard dan menutup sistem

##### **2.1.2 Admin Section**

- Dashboard Admin

Fitur ini menyajikan tampilan utama bagi admin yang berisi informasi profil perusahaan VenueVibe secara ringkas namun informatif. Melalui dashboard ini, admin dapat melihat gambaran umum perusahaan serta akses awal ke fitur-fitur utama lainnya dalam sistem.

- **Kelola Data Barang**

Fitur ini digunakan untuk mengelola seluruh data barang yang tersedia untuk disewakan kepada customer. Admin dapat menambahkan barang baru, mengedit informasi barang yang sudah ada, maupun menghapus barang yang tidak lagi tersedia.

- **Konfirmasi Pesanan**

Fitur ini memungkinkan admin untuk melihat daftar pesanan yang masuk dari customer, memproses pesanan tersebut, serta melakukan konfirmasi pembayaran. Setelah proses selesai, status pesanan akan diperbarui secara otomatis menjadi “Lunas” sebagai tanda bahwa transaksi telah berhasil diselesaikan.

#### 2.1.3 Customer Section

- **Dashboard Customer**

Fitur ini digunakan untuk menyajikan tampilan utama kepada pengguna (customer) setelah berhasil masuk ke aplikasi. Dashboard ini berisi informasi ringkas atau sambutan, memberikan gambaran umum tentang aplikasi VenueVibe, dan berfungsi sebagai titik awal akses ke fitur-fitur utama yang tersedia untuk customer.

- **Lihat Daftar Barang**

Fitur ini digunakan untuk memungkinkan customer menjelajahi dan memilih barang-barang yang tersedia untuk disewa dari VenueVibe. Fitur ini terbagi menjadi beberapa tahapan proses:

- **Daftar Barang Tersedia**

Tahapan ini menampilkan katalog lengkap dari semua barang yang dapat disewa oleh customer. Setiap item ditampilkan dengan detail seperti gambar, nama barang, kategori, harga sewa, dan informasi stok. Customer dapat melihat ketersediaan barang dan memilih barang yang ingin mereka sewa.

- Pesanan

Setelah customer memilih barang dari "Daftar Barang Tersedia" dan mengklik tombol "Sewa", fitur ini akan menampilkan halaman rincian pesanan. Customer dapat menentukan durasi sewa (dalam hari) dan jumlah unit yang diinginkan untuk barang yang dipilih. Sistem akan secara otomatis menghitung dan menampilkan total harga sewa berdasarkan pilihan customer.
- Pembayaran

Setelah customer mengonfirmasi detail pesanan pada tahap "Pesanan", fitur ini akan mengarahkan mereka ke halaman pembayaran. Di sini, customer dapat memilih metode pembayaran (misalnya E-Wallet atau Transfer Bank), memasukkan detail rekening, dan tanggal pembayaran. Setelah semua informasi diisi dan dikonfirmasi, sistem akan mencatat pesanan, mengurangi stok barang yang disewa, dan menyimpan detail pembayaran ke database, lalu mengarahkan customer ke riwayat pesanan mereka.

- Riwayat Pesanan

Fitur ini menampilkan daftar pesanan yang telah dilakukan oleh customer. Setiap pesanan menunjukkan informasi seperti ID pesanan, nama barang, tanggal pemesanan, status pembayaran, total harga, jumlah unit yang disewa, dan durasi sewa. Customer juga dapat melihat rincian pesanan dan melakukan aksi seperti mencetak struk pembelian jika pembelian sudah dikonfirmasi oleh admin dan berstatus Lunas.

## 2.2 Konsep OOP pada Kode Program Rental Property Event

### 2.2.1 Login and Register Section

#### A. Login

- Encapsulation

```
10  private JTextField emailField;  
11  private JPasswordField passwordField;  
12
```

Variabel emailField dan passwordField disimpan secara private, sehingga tidak bisa diakses langsung dari luar kelas. Ini adalah contoh enkapsulasi, untuk menjaga agar data hanya dapat diakses dan dimodifikasi oleh metode dalam kelas LoginFrame.

- Inheritance

```
public class LoginFrame extends JFrame {
```

LoginFrame mewarisi kelas JFrame dari library javax.swing. Artinya, LoginFrame bisa menggunakan seluruh fitur dari JFrame, seperti setSize(), setTitle(), setVisible(), dan lain-lain.

- Polymorphism

```
forgotPass.setCursor(new Cursor(Cursor.HAND_CURSOR));
forgotPass.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        new ForgotPasswordFrame();
    }
});
```

Pada kode ini, kita membuat sebuah fungsi khusus untuk menangani kejadian ketika pengguna mengklik label "Forgot Password?". Untuk melakukan ini, kita menggunakan sebuah kelas bernama MouseAdapter yang sudah menyediakan berbagai metode untuk merespon aksi mouse, salah satunya adalah mouseClicked(). Namun, secara default, metode mouseClicked() dalam MouseAdapter tidak melakukan apa-apa. Oleh karena itu, kita menggantikan (override) metode tersebut dengan versi kita sendiri, yaitu agar saat mouse diklik, program akan membuka jendela baru bernama ForgotPasswordFrame.

```
UserRoleHandler handler;
if (user.getRole().equalsIgnoreCase("admin")) {
    handler = new AdminHandler();
} else {
    handler = new CustomerHandler();
}
```

Di sini, objek handler bertipe UserRoleHandler, tetapi bisa menyimpan instance dari AdminHandler atau CustomerHandler. Meskipun objeknya berbeda, keduanya memiliki

method openDashboard(user). Ini merupakan implementasi polymorphism, objek yang berbeda (AdminHandler, CustomerHandler) dapat dipanggil melalui referensi tipe induknya (UserRoleHandler) dan metode yang dipanggil akan menyesuaikan dengan implementasinya masing-masing.

## B. Register

- Encapsulation

```
8     private JTextField namaField;
9     private JTextField usernameField;
10    private JPasswordField passwordField;
11    private JTextArea alamatArea;
12    private JTextField noTelpField;
```

Variabel variabel diatas, dibentuk dalam private sehingga tidak bisa diakses langsung dari luar kelas, yang mana ini merupakan penerapan dari enkapsulasi dalam integrasi kemanan.

- Inheritance

```
public class RegisterFrame extends JFrame {
```

RegisterFrame mewarisi kelas JFrame dari library javax.swing. Artinya, LoginFrame bisa menggunakan seluruh fitur dari JFrame, seperti setSize(), setTitle(), setVisible(), dan lain-lain.

- Polymorphism

```
loginBtn.addActionListener(e -> {
    new LoginFrame();
    dispose();
});
rightPanel.add(loginBtn);
```

addActionListener menerima objek dengan tipe ActionListener, interface yang memiliki method actionPerformed(). Lambda expression yang digunakan secara implisit **meng-override method tersebut**.

### 2.2.2 Admin Section

#### A. Dashboard

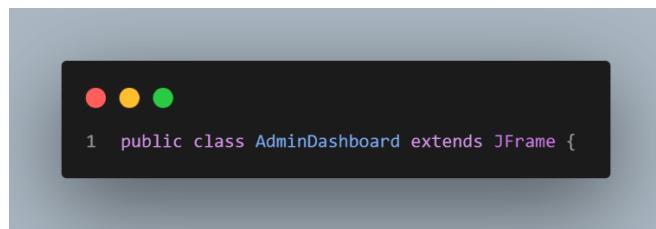
- Encapsulation



```
1 private final JPanel mainPanel;
```

Variabel mainPanel dibungkus di dalam class AdminDashboard dan diberi modifier private, artinya hanya bisa diakses oleh class itu sendiri.

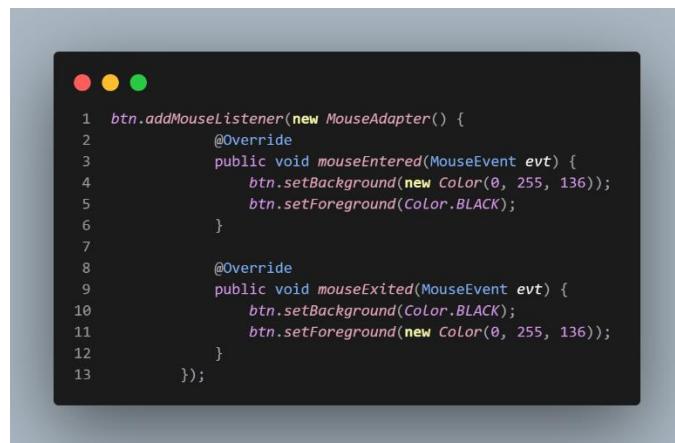
- Inheritance



```
1 public class AdminDashboard extends JFrame {
```

AdminDashboard adalah subclass dari JFrame, artinya mewarisi semua method dan atribut milik JFrame. Tanpa membuat ulang method seperti setSize(), setLayout(), add(), bisa langsung menggunakan dalam AdminDashboard.

- Polymorphism



```
1 btn.addMouseListener(new MouseAdapter() {
2     @Override
3     public void mouseEntered(MouseEvent evt) {
4         btn.setBackground(new Color(0, 255, 136));
5         btn.setForeground(Color.BLACK);
6     }
7     @Override
8     public void mouseExited(MouseEvent evt) {
9         btn.setBackground(Color.BLACK);
10        btn.setForeground(new Color(0, 255, 136));
11    }
12 });
13});
```

Method mouseEntered() dan mouseExited() dioverride dari class MouseAdapter. Ini memungkinkan untuk memodifikasi perilaku default saat kursor masuk atau keluar dari tombol contoh runtime polymorphism.

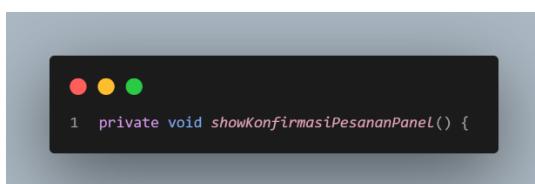
- Abstraction



Method `showCompanyProfile()` menyembunyikan detail tentang layout, pengaturan warna, dan teks yang ditampilkan. Pemanggil method tidak perlu tahu isi detailnya cukup tahu bahwa fungsi ini akan "menampilkan profil perusahaan".



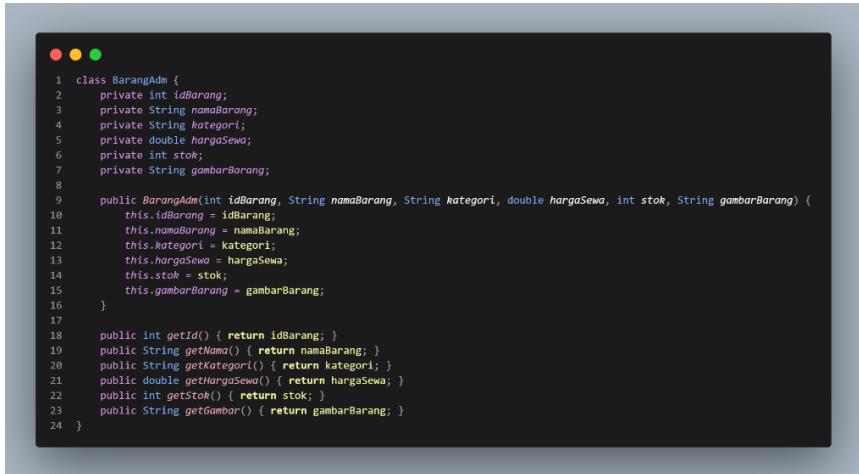
Method `showBarangAdmin()` menyembunyikan detail implementasi tampilan halaman kelola data barang. Yang memanggil method ini tidak perlu tahu bahwa `mainPanel` diatur ulang dan ditambahkan panel `BarangAdmin` cukup tahu bahwa method ini akan menampilkan bagian kelola barang.



Method `showKonfirmasiPesananPanel()` menyembunyikan bagaimana cara kerja tampilan halaman konfirmasi pesanan. Pengguna method cukup tahu bahwa fungsi ini akan menampilkan halaman konfirmasi, tanpa paham detil GUI-nya.

## B. Kelola Data Barang

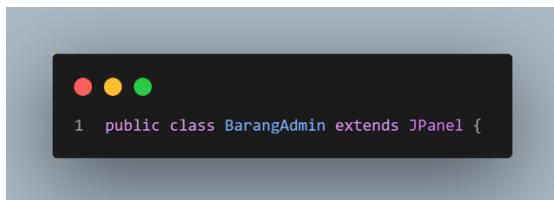
- Encapsulation



```
1  class BarangAdm {
2      private int idBarang;
3      private String namaBarang;
4      private String kategori;
5      private double hargaSewa;
6      private int stok;
7      private String gambarBarang;
8
9      public BarangAdm(int idBarang, String namaBarang, String kategori, double hargaSewa, int stok, String gambarBarang) {
10         this.idBarang = idBarang;
11         this.namaBarang = namaBarang;
12         this.kategori = kategori;
13         this.hargaSewa = hargaSewa;
14         this.stok = stok;
15         this.gambarBarang = gambarBarang;
16     }
17
18     public int getId() { return idBarang; }
19     public String getNama() { return namaBarang; }
20     public String getKategori() { return kategori; }
21     public double getHargaSewa() { return hargaSewa; }
22     public int getStok() { return stok; }
23     public String getGambar() { return gambarBarang; }
24 }
```

Class BarangAdm menyimpan data internal berupa atribut seperti idBarang, namaBarang, kategori, hargaSewa, stok, dan gambarBarang yang semuanya diberi modifier akses private yang tidak bisa diakses langsung dari luar kelas untuk menjaga integritas dan keamanan data. Sebagai gantinya, disediakan method getter untuk mengakses nilai-nilai properti tersebut. Dengan cara ini, jika suatu saat ingin menambahkan logika tambahan misalnya validasi data, perhitungan otomatis, format tampilan bisa dilakukan di dalam getter, tanpa mengubah struktur luar program. Pada konstruktor sebagai cara untuk menginisialisasi objek BarangAdm secara lengkap saat pertama kali dibuat. Semua atribut diisi nilai saat instansiasi objek, sehingga objek yang terbentuk selalu dalam keadaan valid.

- Inheritance



```
1  public class BarangAdmin extends JPanel {
```

BarangAdmin adalah subclass dari JPanel, artinya mewarisi semua method dan atribut milik JFrame. Tanpa membuat ulang method seperti setSize(), setLayout(), add(), bisa langsung menggunakannya dalam BarangAdmin.

- Polymorphism

```

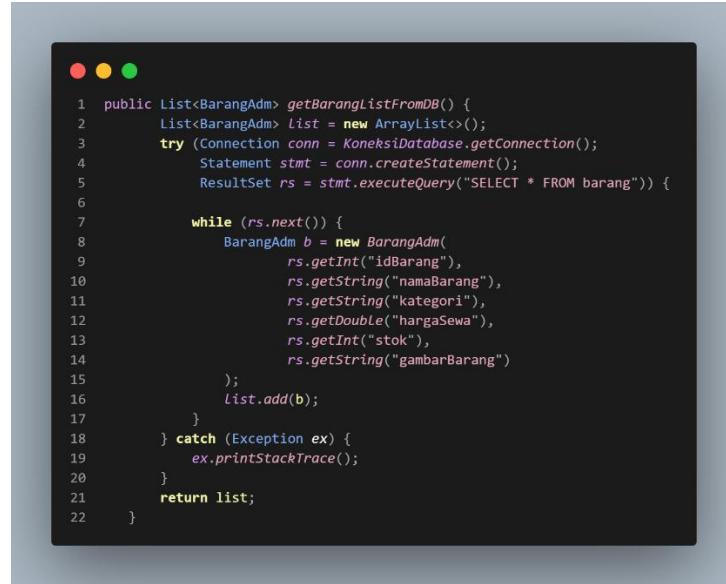
1  private void tampilkanFormTambah() {
2      tampilkanForm(null);
3  }
4
5  // Edit barang pakai objek BarangAdm (polimorfisme lewat overload method tampilkanForm)
6  public void editBarang(int id) {
7      try (Connection conn = KoneksiDatabase.getConnection()) {
8          PreparedStatement stmt = conn.prepareStatement("SELECT * FROM barang WHERE IdBarang=?");
9          stmt.setInt(1, id);
10         try (ResultSet rs = stmt.executeQuery()) {
11             if (rs.next()) {
12                 BarangAdm b = new BarangAdm(
13                     rs.getInt("IdBarang"),
14                     rs.getString("namabarang"),
15                     rs.getString("kategoris"),
16                     rs.getDouble("hargaSewa"),
17                     rs.getInt("stok"),
18                     rs.getString("gambarBarang")
19                 );
20                 tampilkanForm(b);
21             }
22         } catch (Exception e) {
23             e.printStackTrace();
24             JOptionPane.showMessageDialog(this, "Gagal mengambil data untuk edit.");
25         }
26     }
27 }
28
29 // Overloading method tampilkanForm untuk tambah (null) dan edit (BarangAdm)
30 private void tampilkanForm(BarangAdm barang) {
31     boolean isEdit = barang != null;
32
33     JDialog dialog = new JDialog((Frame) null, isEdit ? "Edit Barang" : "Tambah Barang", true);
34     dialog.setSize(600, 400);
35     dialog.setLocationRelativeTo(null);
36     dialog.setLayout(new BorderLayout());
37     dialog.getContentPane().setBackground(new Color(45, 45, 65));
38
39     JLabel lblTitle = new JLabel(isEdit ? "Edit Barang" : "Tambah Barang", SwingConstants.CENTER);
40     lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 22));
41     lblTitle.setForeground(new Color(0, 255, 136));
42     lblTitle.setBorder(BorderFactory.createEmptyBorder(15, 10, 15, 10));
43     dialog.add(lblTitle, BorderLayout.NORTH);
44
45     JTextField nama = new JTextField();
46     JTextField kategori = new JTextField();
47     JTextField harga = new JTextField();
48     JTextField stok = new JTextField();
49     JTextField gambar = new JTextField();
50
51     if (!isEdit) {
52         nama.setText(barang.getNama());
53         kategori.setText(barang.getKategori());
54         harga.setText(String.valueOf(barang.getHargaSewa()));
55         stok.setText(String.valueOf(barang.getStok()));
56         gambar.setText(barang.getGambar());
57     }
58
59     JTextField[] fields = {nama, kategori, harga, stok, gambar};
60     Dimension inputSize = new Dimension(350, 40);
61     for (JTextField tf : fields) {
62         tf.setPreferredSize(inputSize);
63         tf.setFont(new Font("Segoe UI", Font.PLAIN, 14));
64         tf.setBackground(Color.WHITE);
65         tf.setForeground(Color.BLACK);
66         tf.setCaretColor(Color.BLACK);
67     }
68
69     JButton browseBtn = new JButton("Browse");
70     browseBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
71     browseBtn.setBackground(new Color(0, 255, 136));
72     browseBtn.setForeground(Color.BLACK);
73     browseBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
74     browseBtn.setFocusable(false);
75     browseBtn.setPreferredWidth(new Dimension(60, 40));
76     browseBtn.addActionListener(e -> {
77         JFileChooser chooser = new JFileChooser("src/rental/");
78         int result = chooser.showOpenDialog(dialog);
79         if (result == JFileChooser.APPROVE_OPTION) {
80             gambar.setText(chooser.getSelectedFile().getName());
81         }
82     });
83
84     JPanel formPanel = new JPanel(new GridBagLayout());
85     formPanel.setBackground(new Color(45, 45, 65));
86     GridBagConstraints gbc = new GridBagConstraints();
87     gbc.insets = new Insets(10, 10, 10, 10);
88     gbc.fill = GridBagConstraints.HORIZONTAL;
89
90     String[] labels = {"Nama:", "Kategori:", "Harga Sewa:", "Stok:", "Gambar:"};
91     for (int i = 0; i < labels.length; i++) {
92         gbc.gridx = 0;
93         gbc.gridy = i;
94         gbc.weightx = 0.0;
95         JLabel lbl = new JLabel(labels[i]);
96         lbl.setForeground(Color.WHITE);
97         lbl.setFont(new Font("Segoe UI", Font.BOLD, 14));
98         formPanel.add(lbl, gbc);
99
100        gbc.gridx = 1;
101        gbc.weightx = 1.0;
102        formPanel.add(fields[i], gbc);
103
104        if (i == 4) {
105            gbc.gridx = 2;
106            gbc.weightx = 0.0;
107            formPanel.add(browseBtn, gbc);
108        }
109    }

```

Polimorfisme di kode ini terlihat dari dua method `tampilkanForm()` yang punya nama sama tapi parameter berbeda: satu dipanggil tanpa data (untuk tambah barang), satu dengan objek `BarangAdm` (untuk edit barang). Ini disebut method overloading atau

polimorfisme statis. Dengan cara ini, satu method bisa menangani dua fungsi berbeda menampilkan form tambah atau edit dengan logika yang disesuaikan berdasarkan ada atau tidaknya data. Polimorfisme membuat kode lebih fleksibel, rapi, dan mudah dipelihara karena fungsi yang mirip digabung dalam satu nama method.

- Abstraction



```
1 public List<BarangAdm> getBarangListFromDB() {
2     List<BarangAdm> list = new ArrayList<>();
3     try (Connection conn = KoneksiDatabase.getConnection();
4          Statement stmt = conn.createStatement();
5          ResultSet rs = stmt.executeQuery("SELECT * FROM barang")) {
6
7         while (rs.next()) {
8             BarangAdm b = new BarangAdm(
9                 rs.getInt("idBarang"),
10                rs.getString("namaBarang"),
11                rs.getString("kategori"),
12                rs.getDouble("hargaSewa"),
13                rs.getInt("stok"),
14                rs.getString("gambarBarang")
15            );
16            list.add(b);
17        }
18    } catch (Exception ex) {
19        ex.printStackTrace();
20    }
21    return list;
22 }
```

Metode `getBarangListFromDB()` menerapkan konsep abstraction dengan menyembunyikan detail teknis pengambilan data dari database kepada pengguna metode tersebut. Pengguna cukup memanggil metode ini untuk mendapatkan daftar objek `BarangAdm` tanpa harus tahu bagaimana koneksi ke database dibuat, bagaimana query SQL dijalankan, atau bagaimana data diproses dari `ResultSet`.

- Modularity



```
1 public void loadData() {
2     model.setRowCount(0);
3     for (BarangAdm b : getBarangListFromDB()) {
4         ImageIcon icon = new ImageIcon(new ImageIcon("src/rental/" + b.getGambar())
5                                         .getImage().getScaledInstance(60, 60, Image.SCALE_SMOOTH));
6         model.addRow(new Object[]{
7             b.getId(),
8             b.getNama(),
9             b.getKategori(),
10            "Rp. " + String.format("%.0f", b.getHargaSewa()).replace(".", ","),
11            b.getStok(),
12            icon,
13            ""
14        });
15    }
16 }
```

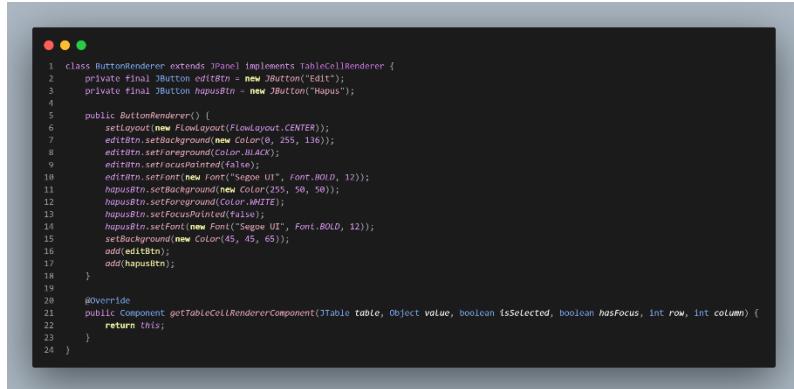
Modularitas di `loadData()` untuk mengisi tabel dengan data barang yang sudah diambil dari database lewat method terpisah (`getBarangListFromDB()`). Dengan memisahkan pengambilan data dan pemrosesan tampilan, kode jadi lebih terstruktur, mudah

dipelihara, dan perubahan di satu bagian tidak mempengaruhi bagian lain. Jadi, modularitas membantu membuat program lebih rapi dan fleksibel.



```
1 public void hapusBarang(int idbarang) {
2     int konfirmasi = JOptionPane.showConfirmDialog(this, "Apakah Anda yakin ingin menghapus barang ini?", "Konfirmasi Hapus", JOptionPane.YES_NO_OPTION);
3     if (konfirmasi == JOptionPane.YES_OPTION) {
4         try (Connection conn = KoneksiDatabase.getConnection()) {
5             Statement stat = conn.prepareStatement("DELETE FROM barang WHERE idbarang=?");
6             stat.setInt(1, idbarang);
7             stat.executeUpdate();
8             loadData();
9         } catch (Exception e) {
10            JOptionPane.showMessageDialog(this, "Gagal menghapus data: " + e.getMessage());
11        }
12    }
13 }
14 }
```

Metode hapusBarang menunjukkan penerapan modularitas dengan memisahkan tanggung jawab dalam fungsi yang jelas dan terstruktur. Pertama, metode ini meminta konfirmasi dari pengguna sebelum melakukan penghapusan data, sehingga menghindari kesalahan penghapusan yang tidak disengaja. Setelah konfirmasi diterima, kode menjalankan perintah SQL untuk menghapus data barang berdasarkan idBarang di database menggunakan mekanisme try-catch untuk menangani kemungkinan kesalahan dengan baik. Terakhir, metode memanggil fungsi loadData() untuk memperbarui tampilan data setelah penghapusan berhasil dilakukan.

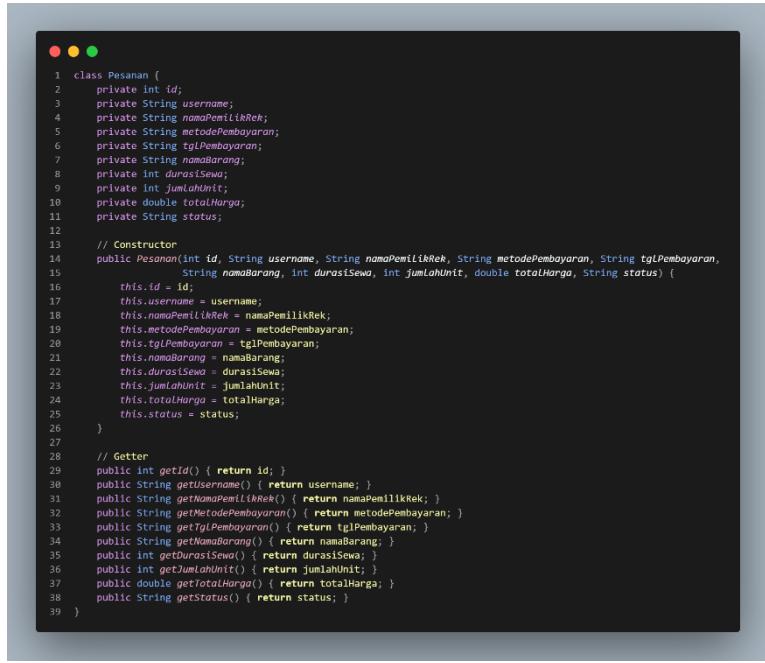


```
1 class ButtonRenderer extends JPanel implements TableCellRenderer {
2     private final JButton editBtn = new JButton("Edit");
3     private final JButton hapusBtn = new JButton("Hapus");
4     ...
5     public ButtonRenderer() {
6         setLayout(new FlowLayout(FlowLayout.CENTER));
7         editBtn.setBackground(new Color(0, 255, 136));
8         editBtn.setForeground(Color.BLACK);
9         editBtn.setFocusPainted(false);
10        editBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
11        hapusBtn.setBackground(new Color(255, 50, 50));
12        hapusBtn.setForeground(Color.WHITE);
13        hapusBtn.setFocusPainted(false);
14        hapusBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
15        setPreferredSize(new Dimension(45, 45));
16        add(editBtn);
17        add(hapusBtn);
18    }
19    ...
20    @Override
21    public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus, int row, int column) {
22        return this;
23    }
24 }
```

Kelas ButtonRenderer menunjukkan modularitas dengan memisahkan fungsi khusus untuk menampilkan tombol "Edit" dan "Hapus" dalam sebuah sel tabel secara terpisah dari logika utama aplikasi. Dengan mengimplementasikan interface TableCellRenderer, kelas ini bertanggung jawab hanya untuk rendering tampilan tombol di tabel, sehingga memudahkan pemeliharaan dan pengembangan kode.

### C. Konfirmasi Pesanan

- Encapsulation



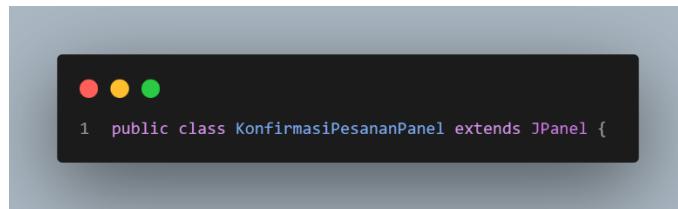
```

1  class Pesanan {
2      private int id;
3      private String username;
4      private String namaPemilikRek;
5      private String metodePembayaran;
6      private String tglPembayaran;
7      private String namaBarang;
8      private int durasiSewa;
9      private int jumlahUnit;
10     private double totalHarga;
11     private String status;
12
13     // Constructor
14     public Pesanan(int id, String username, String namaPemilikRek, String metodePembayaran, String tglPembayaran,
15                     String namaBarang, int durasiSewa, int jumlahUnit, double totalHarga, String status) {
16         this.id = id;
17         this.username = username;
18         this.namaPemilikRek = namaPemilikRek;
19         this.metodePembayaran = metodePembayaran;
20         this.tglPembayaran = tglPembayaran;
21         this.namaBarang = namaBarang;
22         this.durasiSewa = durasiSewa;
23         this.jumlahUnit = jumlahUnit;
24         this.totalHarga = totalHarga;
25         this.status = status;
26     }
27
28     // Getter
29     public int getId() { return id; }
30     public String getUsername() { return username; }
31     public String getNamaPemilikRek() { return namaPemilikRek; }
32     public String getMetodePembayaran() { return metodePembayaran; }
33     public String getTglPembayaran() { return tglPembayaran; }
34     public String getNamaBarang() { return namaBarang; }
35     public int getDurasiSewa() { return durasiSewa; }
36     public int getJumlahUnit() { return jumlahUnit; }
37     public double getTotalHarga() { return totalHarga; }
38     public String getStatus() { return status; }
39 }

```

Enkapsulasi pada class Pesanan ditunjukkan dengan menyembunyikan atribut-atribut data seperti id, username, namaPemilikRek, dan lain-lain dengan modifier private, sehingga atribut tersebut tidak dapat diakses langsung dari luar kelas. Akses dan manipulasi data hanya dapat dilakukan melalui metode getter yang disediakan. Dengan cara ini, data internal objek terlindungi dari perubahan yang tidak disengaja atau tidak valid oleh kode luar, sehingga menjaga integritas dan konsistensi objek. Enkapsulasi juga membuat kelas Pesanan menjadi modular dan lebih mudah dipelihara karena kontrol penuh terhadap bagaimana data diakses dan dimodifikasi.

- Inheritance



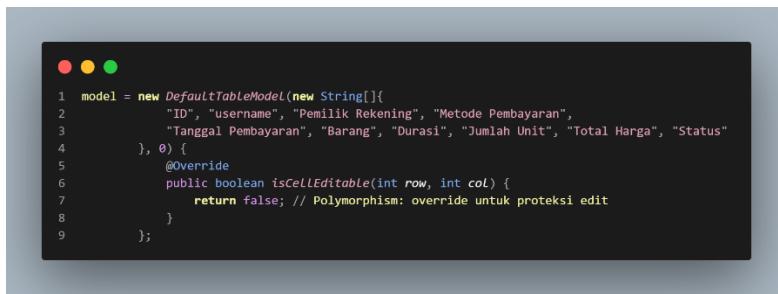
```

1  public class KonfirmasiPesananPanel extends JPanel {

```

KonfirmasiPesananPanel adalah subclass dari JPanel, artinya mewarisi semua method dan atribut milik JFrame. Tanpa membuat ulang method seperti setSize(), setLayout(), add(), bisa langsung menggunakan dalam KonfirmasiPesananPanel.

- Polymorphism



```
1 model = new DefaultTableModel(new String[]{
2     "ID", "username", "Pemilik Rekening", "Metode Pembayaran",
3     "Tanggal Pembayaran", "Barang", "Durasi", "Jumlah Unit", "Total Harga", "Status"
4 }, 0) {
5     @Override
6     public boolean isCellEditable(int row, int col) {
7         return false; // Polymorphism: override untuk proteksi edit
8     }
9 }
```

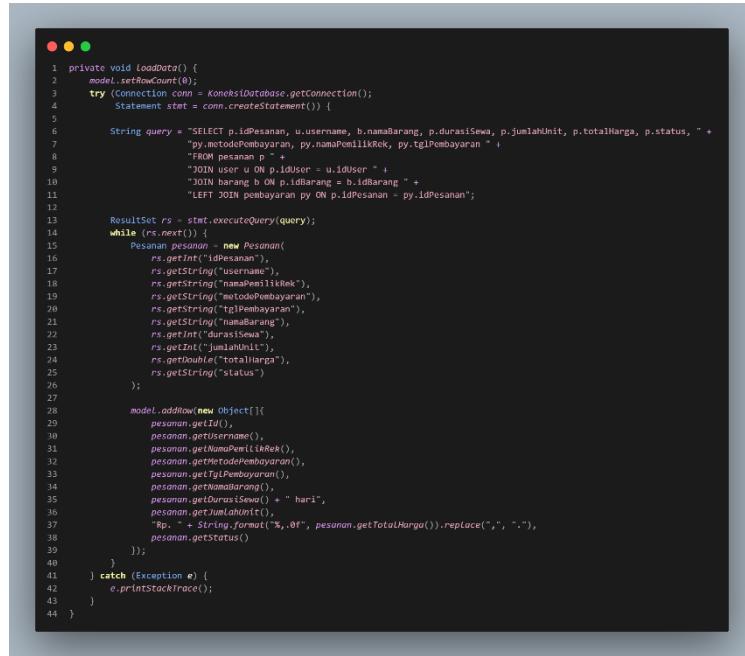
Pada kode ini, polimorfisme diterapkan dengan cara *override* metode `isCellEditable` dari kelas `DefaultTableModel` untuk mengubah perilaku defaultnya, yaitu membuat semua sel tabel menjadi tidak bisa diedit. Dengan ini, meskipun `DefaultTableModel` secara bawaan mungkin mengizinkan pengeditan sel, kelas turunan ini memberikan bentuk perilaku baru yang lebih sesuai dengan kebutuhan program melindungi data dari perubahan langsung oleh pengguna melalui antarmuka tabel.

- Modularity



```
1 private void setupTableUI() {
2     table.setRowHeight(70);
3     table.setFont(new Font("Segoe UI", Font.PLAIN, 14));
4     table.setBackground(new Color(45, 45, 65));
5     table.setForeground(Color.WHITE);
6     table.setSelectionBackground(new Color(0, 255, 136, 100));
7     table.setSelectionForeground(Color.WHITE);
8     table.setGridColor(new Color(60, 60, 80));
9     table.setShowGrid(true);
10
11    JTableHeader header = table.getTableHeader();
12    header.setFont(new Font("Segoe UI", Font.BOLD, 14));
13    header.setBackground(new Color(0, 255, 136));
14    header.setForeground(Color.BLACK);
15    header.setReorderingAllowed(false);
16 }
```

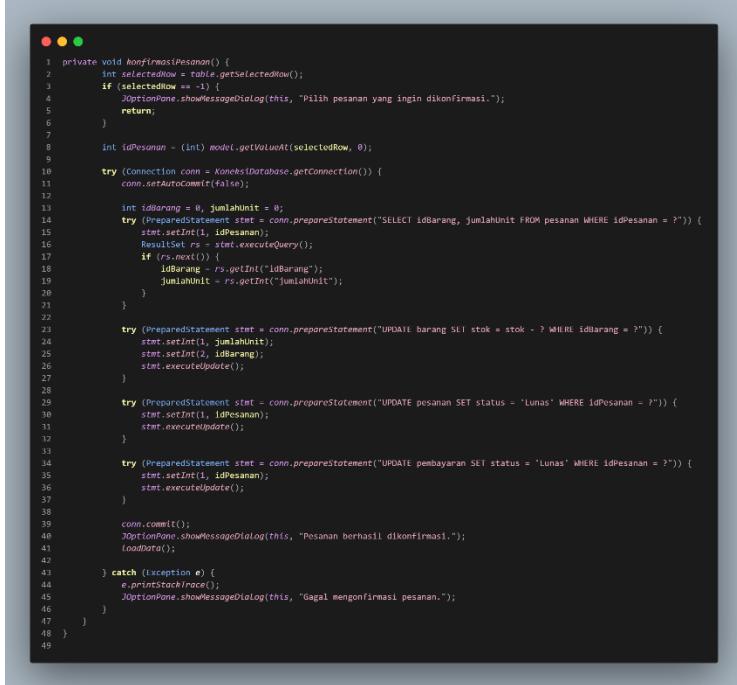
Pada metode `setupTableUI()`, modularitas diterapkan dengan memisahkan semua pengaturan tampilan dan gaya tabel ke dalam satu fungsi khusus. Hal ini membuat kode lebih terstruktur, mudah dibaca, dan mudah dirawat karena semua konfigurasi UI tabel berada di satu tempat terpisah dari logika bisnis atau pengambilan data.



```

1 private void loadData() {
2     model.setRowCount(0);
3     try (Connection conn = KoneksiDatabase.getConnection();
4          Statement stmt = conn.createStatement()) {
5
6         String query = "SELECT p.idPesanan, u.username, b.namaBarang, p.durasiSewa, p.jumlahUnit, p.totalHarga, p.status, " +
7             "py.metodePembayaran, py.namaCilikKek, py.tglPembayaran " +
8             "FROM pesanan p " +
9             "JOIN user u ON p.idUser = u.idUser " +
10            "JOIN barang b ON p.idBarang = b.idBarang " +
11            "LEFT JOIN pembayaran py ON p.idPesanan = py.idPesanan";
12
13        ResultSet rs = stmt.executeQuery(query);
14        while (rs.next()) {
15            Pesanan pesanan = new Pesanan();
16            pesanan.setIdPesanan(rs.getInt("idPesanan"));
17            pesanan.setUsername(rs.getString("username"));
18            pesanan.setIdBarang(rs.getInt("idBarang"));
19            pesanan.setMetodePembayaran(rs.getString("tglPembayaran"));
20            pesanan.setNamaBarang(rs.getString("namaBarang"));
21            pesanan.setDurasiSewa(rs.getInt("durasiSewa"));
22            pesanan.setJumlahUnit(rs.getInt("jumlahUnit"));
23            pesanan.setTotalHarga(rs.getDouble("totalHarga"));
24            pesanan.setStatus(rs.getString("status"));
25
26        };
27
28        model.addRow(new Object[]{
29            pesanan.getID(),
30            pesanan.getUsername(),
31            pesanan.getIdBarang(),
32            pesanan.getMetodePembayaran(),
33            pesanan.getNamaBarang(),
34            pesanan.getDurasiSewa(),
35            pesanan.getJumlahUnit(),
36            "Rp. " + String.format("%,.0f", pesanan.getTotalHarga()).replace(".", ","),
37            pesanan.getStatus()
38        });
39    } catch (Exception e) {
40        JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat memuat data.");
41        e.printStackTrace();
42    }
43 }
44 }
```

Modularitas pada metode loadData() terlihat dari bagaimana fungsi ini secara khusus menangani proses pengambilan data dari database dan memasukkannya ke dalam model tabel. Dengan memisahkan logika pengambilan dan pengolahan data dalam satu metode terpisah, kode menjadi lebih terorganisir dan mudah dipelihara.



```

1 private void konfirmasiPesanan() {
2     int selectedRow = table.getSelectedRow();
3     if (selectedRow == -1) {
4         JOptionPane.showMessageDialog(this, "Pilih pesanan yang ingin dikonfirmasi.");
5         return;
6     }
7     int idPesanan = (int) model.getValueAt(selectedRow, 0);
8     try (Connection conn = KoneksiDatabase.getConnection();
9          conn.setAutoCommit(false)) {
10
11         int idBarang = 0, jumlahUnit = 0;
12         try (PreparedStatement stmt = conn.prepareStatement("SELECT idBarang, jumlahUnit FROM pesanan WHERE idPesanan = ?")) {
13             stmt.setInt(1, idPesanan);
14             ResultSet rs = stmt.executeQuery();
15             if (rs.next()) {
16                 idBarang = rs.getInt("idBarang");
17                 jumlahUnit = rs.getInt("jumlahUnit");
18             }
19         }
20
21         try (PreparedStatement stmt = conn.prepareStatement("UPDATE barang SET stok = stok - ? WHERE idBarang = ?")) {
22             stmt.setInt(1, jumlahUnit);
23             stmt.setInt(2, idBarang);
24             stmt.executeUpdate();
25         }
26
27         try (PreparedStatement stmt = conn.prepareStatement("UPDATE pesanan SET status = 'Lunas' WHERE idPesanan = ?")) {
28             stmt.setInt(1, idPesanan);
29             stmt.executeUpdate();
30         }
31
32         try (PreparedStatement stmt = conn.prepareStatement("UPDATE pembayaran SET status = 'Lunas' WHERE idPesanan = ?")) {
33             stmt.setInt(1, idPesanan);
34             stmt.executeUpdate();
35         }
36
37         conn.commit();
38         JOptionPane.showMessageDialog(this, "Pesanan berhasil dikonfirmasi.");
39         loadData();
40
41     } catch (Exception e) {
42         JOptionPane.showMessageDialog(this, "Terjadi kesalahan saat mengkonfirmasi pesanan.");
43         e.printStackTrace();
44     }
45 }
46 }
```

Modularitas pada metode konfirmasiPesanan() terlihat dari pengelompokan langkah-langkah konfirmasi pesanan dalam satu fungsi terpisah yang jelas tugasnya: memvalidasi pilihan, memperbarui stok barang, dan mengubah status pesanan serta pembayaran di

database. Dengan mengemas semua proses ini dalam satu metode, kode menjadi lebih terstruktur, mudah dipahami, dan dapat diubah atau diperbaiki tanpa memengaruhi bagian lain dari program.

### 2.2.3 Customer Section

#### A. Dashboard Customer

- Enkapsulasi

```
public class DashboardFrame extends JFrame {  
    private JPanel mainPanel;  
    private int currentUserId;  
  
    public DashboardFrame(int idUser) {  
        this.currentUserId = idUser;
```

Atribut mainPanel dan currentUserId dideklarasikan sebagai private. Ini berarti bagian luar kelas DashboardFrame tidak dapat langsung mengakses atau mengubah nilai mainPanel atau currentUserId. Nilai currentUserId hanya dapat diinisialisasi melalui konstruktor (DashboardFrame(int idUser)) dan diakses secara terkontrol melalui metode public seperti showRiwayatPesananPanel(), di mana this.currentUserId digunakan sebagai argumen.

- Inheritance

```
public class DashboardFrame extends JFrame {
```

DashboardFrame mewarisi dari kelas JFrame (public class DashboardFrame extends JFrame). Ini berarti DashboardFrame secara otomatis mendapatkan semua karakteristik dan fungsionalitas dasar dari sebuah jendela (frame) Swing, seperti kemampuan untuk memiliki judul, ukuran, operasi penutupan, dan menampung komponen UI lainnya.

- Polimorfisme

```
btn.addMouseListener(new MouseAdapter() {  
    public void mouseEntered(MouseEvent evt) {  
        btn.setBackground(new Color(r:0, g:255, b:136));  
        btn.setForeground(Color.BLACK);  
    }  
  
    public void mouseExited(MouseEvent evt) {  
        btn.setBackground(Color.BLACK);  
        btn.setForeground(new Color(r:0, g:255, b:136));  
    }  
});
```

Ketika MouseListener ditambahkan ke tombol sidebar, sebuah *anonymous inner class* yang mewarisi dari MouseAdapter digunakan. MouseAdapter sendiri adalah kelas abstrak yang mengimplementasikan MouseListener dan menyediakan implementasi kosong untuk semua metodenya. Dengan mengoverride hanya mouseEntered() dan mouseExited().

## B. Lihat Daftar Barang

SewaListener

- Abstraksi



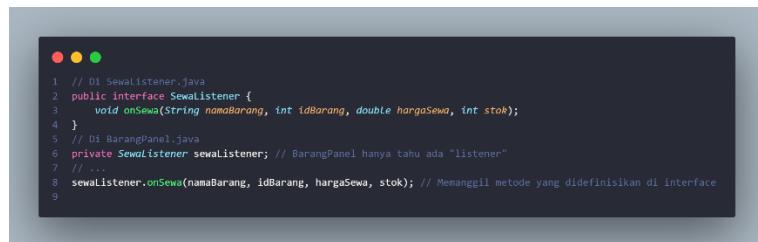
```
1 package rental;
2
3 public interface SewaListener {
4     void onSewa(String namaBarang, int idBarang, double hargaSewa, int stok);
5 }
```

Metode onSewa dideklarasikan tanpa implementasi, menunjukkan bahwa ia adalah metode abstrak yang harus diimplementasikan oleh kelas lain.

- Polimorfisme (sebagai Enabler)

Meskipun bukan contoh polimorfisme itu sendiri, *interface* ini adalah enabler utama untuk polimorfisme. Ia memungkinkan objek dari berbagai kelas yang berbeda untuk diperlakukan sebagai tipe SewaListener, dan metode onSewa() yang dipanggil akan dieksekusi sesuai implementasi spesifik dari objek tersebut.

- Modularitas dan Decoupling



```
1 // Di SewaListener.java
2 public interface SewaListener {
3     void onSewa(String namaBarang, int idBarang, double hargaSewa, int stok);
4 }
5 // Di BarangPanel.java
6 private SewaListener sewaListener; // BarangPanel hanya tahu ada "listener"
7 ...
8 sewaListener.onSewa(namaBarang, idBarang, hargaSewa, stok); // Memanggil metode yang didefinisikan di interface
9
```

Dengan menggunakan *interface*, SewaListener menciptakan pemisahan yang jelas antara komponen yang memicu peristiwa (misalnya, BarangPanel) dan komponen yang menanganinya (DashboardFrame). BarangPanel tidak perlu tahu detail tentang bagaimana peristiwa sewa akan diproses; ia hanya tahu harus memanggil onSewa().

BaseFormPanel

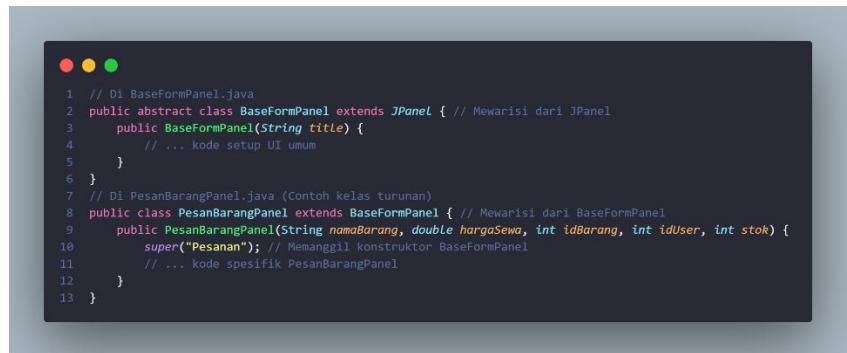
- Abstraksi



```
1 public abstract class BaseFormPanel extends JPanel {
```

Sebagai abstract class, BaseFormPanel menyediakan kerangka dasar untuk panel-panel formulir, mendefinisikan elemen UI umum (judul, tata letak, gaya latar belakang) tanpa mengimplementasikan logika formulir spesifik.

- Pewarisan (Inheritance)



```
1 // Di BaseFormPanel.java
2 public abstract class BaseFormPanel extends JPanel { // Mewarisi dari JPanel
3     public BaseFormPanel(String title) {
4         // ... kode setup UI umum
5     }
6 }
7 // Di PesanBarangPanel.java (Contoh kelas turunan)
8 public class PesanBarangPanel extends BaseFormPanel { // Mewarisi dari BaseFormPanel
9     public PesanBarangPanel(String namaBarang, double hargaSewa, int idBarang, int idUser, int stok) {
10         super("Pesanan"); // Memanggil konstruktor BaseFormPanel
11         // ... kode spesifik PesanBarangPanel
12     }
13 }
```

Berfungsi sebagai kelas induk untuk panel-panel formulir (seperti PesanBarangPanel dan PembayaranPanel). Subkelas mewarisi semua properti dan metode yang didefinisikan di sini, memastikan konsistensi visual dan mengurangi duplikasi kode untuk elemen UI dasar.

- 1) Daftar Barang Tersedia

- Enkapsulasi



```
1 private JPanel cardsPanel;
2 private SewaListener sewaListener; // Atribut private
3
4 private void loadData() { /* ... */ } // Metode private
```

Melindungi data internal (cardsPanel, sewaListener) dengan membuatnya private. Interaksi dengan atribut ini hanya melalui konstruktur atau metode terkontrol. Metode loadData() juga private, menyembunyikan detail pengambilan data dari luar.

- Pewarisan (Inheritance)



```
1 public class BarangPanel extends JPanel {
```

Mewarisi dari javax.swing.JPanel, mendapatkan kemampuan dasar sebagai komponen GUI Swing untuk menampung elemen lain.

- Polimorfisme



```
1 public BarangPanel(SewaListener sewaListener) { // Menerima objek bertipe interface SewaListener
2     this.sewaListener = sewaListener;
3     ...
4     btnSewa.addActionListener(e -> {
5         sewaListener.onSewa(namaBarang, idBarang, hargaSewa, stok); // Memanggil metode polimorfik
6     });
7 }
```

Menerapkan *interface* SewaListener yang memungkinkan BarangPanel berinteraksi dengan berbagai objek "pendengar" melalui satu kontrak. Panggilan sewaListener.onSewa() akan dieksekusi sesuai implementasi konkret dari objek SewaListener yang diberikan (misalnya, lambda di DashboardFrame).

- Modularitas

Berfokus pada tugas tunggal menampilkan daftar barang. Ia mendelegasikan pengambilan data ke KoneksiDatabase dan delegasikan penanganan aksi "Sewa" ke SewaListener, sehingga tidak perlu tahu detail implementasi dari tugas-tugas tersebut.

## 2) Pesanan

- Enkapsulasi



```
1 public class PesanBarangPanel extends BaseFormPanel {
2     // Field untuk menyimpan data barang dan user
3     private String namaBarang;
4     private int idBarang;
5     private intidUser;
6     private double hargaSewa;
7     private int stok;
```

Menyimpan detail pesanan (namaBarang, idBarang, idUser, hargaSewa, stok) sebagai private fields untuk melindungi integritas data internal. Data ini diakses dan diinisialisasi melalui konstruktur.

- Pewarisan (Inheritance)

```
● ● ●
1 public class PesanBarangPanel extends BaseFormPanel { // Mewarisi dari BaseFormPanel
2     public PesanBarangPanel(...) {
3         super("Pesanan"); // Memanggil konstruktor induk
4         // ...
5     }
6 }
```

Mewarisi dari BaseFormPanel, sehingga mendapatkan struktur UI dasar yang konsisten (judul, latar belakang, border) tanpa perlu menulis ulang kode tersebut.

- Polimorfisme

```
● ● ●
1 // Implementasi ChangeListener secara anonim untuk polimorfisme
2 Changelistener updateTotal = new Changelistener() {
3     public void stateChanged(ChangeEvent e) { /* ... */ }
4 };
5 // Interaksi polimorfik dengan DashboardFrame
6 DashboardFrame dashboard = (DashboardFrame) SwingUtilities.getWindowAncestor(this);
7 dashboard.showPembayaranPanel(namaBarang, hargaSewa, idBarang, idUser, stok, durasi, jumlah);
```

Menggunakan implementasi anonim dari ChangeListener untuk pembaruan harga dinamis. Selain itu, berinteraksi dengan DashboardFrame melalui *casting* setelah mendapatkan *ancestor window*, yang memungkinkan panggilan metode spesifik showPembayaranPanel() pada objek DashboardFrame.

- Modularitas

Berfokus pada pengumpulan detail pesanan (durasi, jumlah). Ia menerima data dari panel sebelumnya dan mendelegasikan langkah selanjutnya (navigasi ke pembayaran) ke DashboardFrame, menjaga tanggung jawabnya tetap spesifik.

### 3) Pembayaran

- Enkapsulasi

```
● ● ●
1 private String namaBarang;
2 private double hargaSewa;
3 private int idBarang, idUser, stok, durasi, jumlah;
```

Melindungi semua data pesanan (namaBarang, hargaSewa, idBarang, idUser, stok, durasi, jumlah) sebagai private fields, hanya dapat diinisialisasi melalui konstruktor dan digunakan secara internal.

- Pewarisan (Inheritance)

```

1 public class PembayaranPanel extends BaseFormPanel { // Mewarisi dari BaseFormPanel
2     public PembayaranPanel(...) {
3         super("Pembayaran"); // Memanggil konstruktor induk
4         // ...
5     }
6 }
```

Mewarisi dari BaseFormPanel, mendapatkan struktur UI dasar yang konsisten untuk formulir pembayaran.

- Polimorfisme

```

1 // Implementasi ActionListener secara anonim (lambda)
2 btnKonfirmasi.addActionListener(e -> { /* ... */ });
3 // Interaksi polimorfik dengan dashboardFrame
4 DashboardFrame dashboard = (DashboardFrame) SwingUtilities.getWindowAncestor(this);
5 dashboard.showRiwayatPesananPanel();
```

Menggunakan implementasi anonim dari ActionListener pada tombol konfirmasi. Interaksi dengan DashboardFrame untuk navigasi ke riwayat pesanan juga menunjukkan polimorfisme melalui *casting* tipe.

- Modularitas

Berfokus pada pemrosesan pembayaran dan interaksi database terkait. Ia menerima data dari PesanBarangPanel dan mendelegasikan navigasi ke DashboardFrame, serta interaksi database ke KoneksiDatabase.

## C. Riwayat Pesanan

- Encapsulation
  - RiwayatPesananPanel

```

private ArrayList<PesananRiwayat> getPesananFromDB(int idUser) {

private void showStrukDialog(PesananRiwayat p) {

private void addStrukRow(JPanel panel, GridBagConstraints gbc, String label, String value, Font font,
    Color labelColor, Color valueColor) {
```

Pada RiwayatPesananPanel, metode seperti getPesananFromDB(), showStrukDialog(), dan addStrukRow() dideklarasikan sebagai private. Ini menyembunyikan detail implementasi internal dari bagaimana data diambil, bagaimana dialog struk dibuat, atau bagaimana baris ditambahkan.

- PesananRiwayat

```
public class PesananRiwayat {  
    private int idPesanan;  
    private String namaBarang;  
    private String tanggalPesan;  
    private String status;  
    private double totalHarga;  
    private int jumlahUnit;  
    private int durasiSewa;  
    private String username;
```

Enkapsulasi pada kode ini di kelas PesananRiwayat, semua atribut seperti idPesanan, namaBarang, status, dan lainnya dideklarasikan sebagai private. Ini berarti tidak ada kode di luar kelas PesananRiwayat yang dapat langsung mengubah nilai atribut ini.

- Inheritance

```
public class RiwayatPesananPanel extends JPanel {
```

Inheritansi pada kode ini RiwayatPesananPanel mewarisi dari JPanel (extends JPanel). Ini berarti RiwayatPesananPanel secara otomatis "adalah sebuah" JPanel dan mendapatkan semua fitur dasar GUI seperti kemampuan untuk ditambahkan ke frame, memiliki layout, dan menampung komponen lain.

- Polymorphism

```
JTable table = new JTable(data, kolom) {  
    public boolean isCellEditable(int row, int column) {  
        return column == 7; // Hanya kolom aksi yang bisa diklik  
    }  
};  
table.getColumn(identifier:"Aksi").setCellRenderer(new ButtonRenderer());  
table.getColumn(identifier:"Aksi").setCellEditor(new ButtonEditor(new JCheckBox(), daftarPesanan));
```

Pada kode ini polimorfisme diterapkan Ketika JTable dibuat, metode isCellEditable() dioverride langsung di dalam definisi JTable anonim. Ini adalah contoh polimorfisme, di mana objek JTable ini berperilaku sedikit berbeda dari JTable standar, memungkinkan hanya kolom tertentu yang dapat diedit.

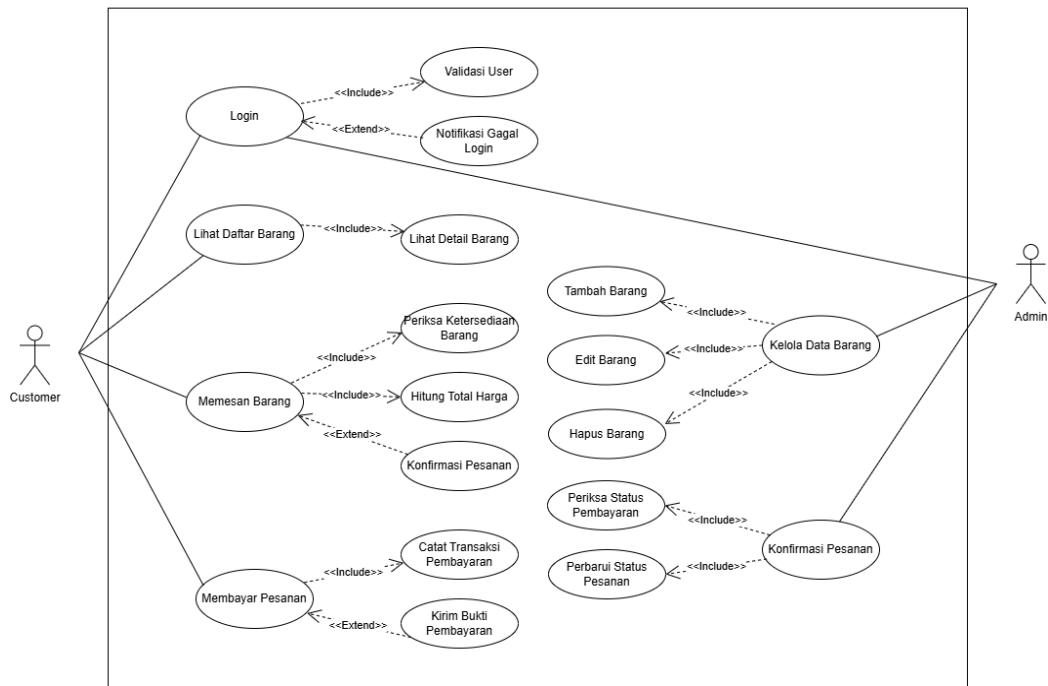
- Abstraksi

```
try (Connection conn = KoneksiDatabase.getConnection()) {
```

Ketika RiwayatPesananPanel memanggil KoneksiDatabase.getConnection(), panel ini hanya perlu tahu bahwa metode tersebut akan mengembalikan objek Connection yang dapat digunakan untuk berinteraksi dengan database.

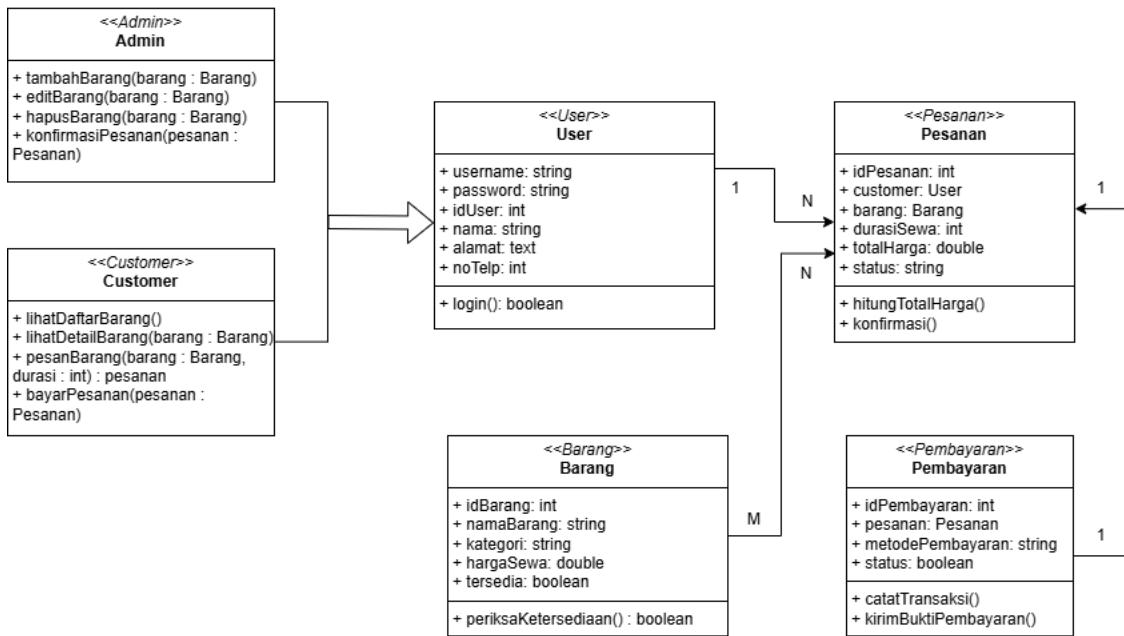
## 2.3 Perancangan Program dengan UML

### 2.3.1 Use Case Diagram



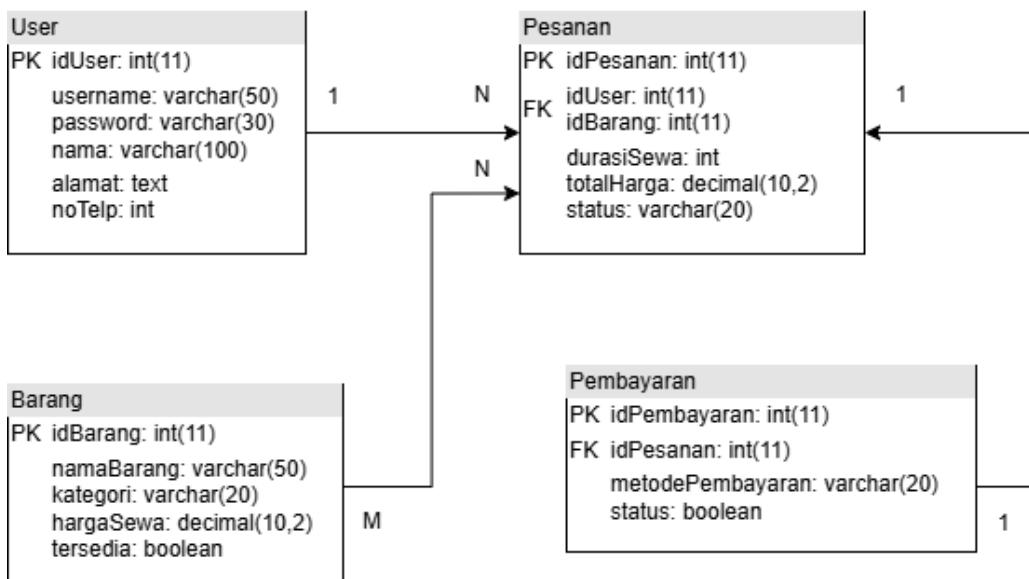
Use case diagram ini menggambarkan alur sistem penyewaan barang yang melibatkan proses login, pengelolaan barang, pemesanan, hingga pembayaran. Customer dapat melihat dan memesan barang setelah login yang divalidasi, termasuk melihat detail dan mengecek ketersediaan barang. Setelah pemesanan, sistem menghitung total harga dan memungkinkan konfirmasi pesanan. Proses pembayaran mencakup pencatatan transaksi dan pengiriman bukti pembayaran. Admin dapat mengelola data barang (tambah, edit, hapus), memeriksa status pembayaran, serta memperbarui dan mengonfirmasi status pesanan.

### 2.3.2 Class Diagram



Class diagram ini menunjukkan struktur utama sistem penyewaan barang, terdiri dari lima kelas: User, Barang, Pesanan, Pembayaran, dan dua role yaitu Admin dan Customer. User memiliki atribut identitas dan fungsi login, serta menjadi relasi dengan banyak Pesanan. Barang berisi informasi item sewaan dan ketersediaannya. Pesanan menghubungkan User dan Barang, mencakup durasi, total harga, serta fungsi konfirmasi dan perhitungan harga. Pembayaran terkait satu pesanan, dengan metode, status, dan fungsi pencatatan transaksi. Admin berperan dalam pengelolaan barang dan konfirmasi pesanan, sedangkan Customer melakukan pemesanan dan pembayaran.

### 2.3.3 Entity Relationship Diagram



Entity Relationship Diagram ini menggambarkan hubungan antar entitas dalam sistem penyewaan barang. Terdapat empat entitas utama: User, Barang, Pesanan, dan Pembayaran. Seorang User dapat membuat banyak Pesanan, dan setiap Pesanan dapat melibatkan satu Barang, namun satu Barang bisa dipesan oleh banyak User. Setiap Pesanan memiliki atribut durasi sewa, total harga, dan status. Entitas Pembayaran berelasi satu-satu dengan Pesanan, menyimpan metode pembayaran dan status pembayaran. Struktur ini mendukung pencatatan transaksi yang terintegrasi dari pengguna hingga pembayaran.

## 2.4 Implementasi Kode Program & Pengujian

### 2.4.1 Login and Register Section

- Koneksi Database

```
1 package rental;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6
7 public class KoneksiDatabase {
8     public static Connection getConnection() {
9         Connection conn = null;
10        String url = "jdbc:mysql://127.0.0.1:3306/dbrental1"; // Nama DB
11        String user = "root"; // Username MySQL
12        String password = ""; // Password MySQL
13
14        try {
15            // Load Driver (opsional untuk Java 8+, tapi tetap aman digunakan)
16            Class.forName(className:"com.mysql.cj.jdbc.Driver");
17
18            // Buat koneksi
19            conn = DriverManager.getConnection(url, user, password);
20            System.out.println(x:"Koneksi berhasil!");
21        } catch (ClassNotFoundException e) {
22            System.out.println(x:"Driver tidak ditemukan!");
23            e.printStackTrace();
24        } catch (SQLException e) {
25            System.out.println(x:"Koneksi gagal!");
26            e.printStackTrace();
27        }
28
29        return conn;
30    }
31
32    // Untuk testing koneksi
33    Run | Debug
34    public static void main(String[] args) {
35        getConnection();
36    }
}
```

Penjelasan

Kode program diatas merupakan kode program dalam koneksi java ke dalam database, dengan koneksi ini, maka data yang diinput dalam GUI java akan terbaca atau terinput dalam database yang telah dibangun. Try and catch digunakan untuk menangani kondisi koneksi, jika koneksi gagal akan menampilkan koneksi gagal, jika berhasil akan menampilkan pesan koneksi berhasil.

- Login

```
1 package rental;
2
3 import java.awt.*;
4 import java.sql.*;
5 import javax.swing.*;
6 import java.awt.event.MouseAdapter;
7 import java.awt.event.MouseEvent;
8
9 public class LoginFrame extends JFrame {
10     private JTextField emailField;
11     private JPasswordField passwordField;
12
13     public LoginFrame() {
14         setTitle("Login - Rental Properti Event");
15         setSize(width:700, height:400);
16         setLocationRelativeTo(c:null);
17         setDefaultCloseOperation(EXIT_ON_CLOSE);
18         setResizable(resizable:false);
19
20         JPanel contentPanel = new JPanel(new GridLayout(rows:1, cols:2));
21
22         JPanel leftPanel = new JPanel();
23         leftPanel.setLayout(mgr:null);
24         leftPanel.setBackground(Color.decode(nm:"#111111"));
25
26         JLabel titleLabel = new JLabel(text:"VenueVibe", SwingConstants.CENTER);
27         titleLabel.setFont(new Font(name:"SansSerif", Font.BOLD, size:24));
28         titleLabel.setForeground(Color.decode(nm:"#80FF00"));
29         titleLabel.setBounds(x:60, y:20, width:250, height:30);
30         leftPanel.add(titleLabel);
31
32         JLabel usernamelabel = new JLabel(text:"Username");
33         usernamelabel.setForeground(Color.GREEN);
34         usernamelabel.setBounds(x:60, y:80, width:100, height:20);
35         leftPanel.add(usernamelabel);
36
37         emailField = new JTextField();
38         emailField.setBounds(x:60, y:100, width:250, height:30);
39         leftPanel.add(emailField);
40
41         JLabel passLabel = new JLabel(text:"Password");
42         passLabel.setForeground(Color.GREEN);
43         passLabel.setBounds(x:60, y:140, width:100, height:20);
44         leftPanel.add(passLabel);
45
46         passwordField = new JPasswordField();
47         passwordField.setBounds(x:60, y:160, width:250, height:30);
48         leftPanel.add(passwordField);
49
50         JLabel forgotPass = new JLabel(text:"Forgot Password?");
51         forgotPass.setForeground(Color.LIGHT_GRAY);
52         forgotPass.setFont(new Font(name:"Arial", Font.PLAIN, size:11));
53         forgotPass.setBounds(x:60, y:195, width:150, height:20);
54         leftPanel.add(forgotPass);
```

```

56     forgotPass.setCursor(new Cursor(Cursor.HAND_CURSOR));
57     forgotPass.addMouseListener(new MouseAdapter() {
58         @Override
59         public void mouseClicked(MouseEvent e) {
60             new ForgotPasswordFrame();
61         }
62     });
63
64
65     JButton signUpBtn = new JButton(text:"Sign Up");
66     signUpBtn.setBounds(x:60, y:220, width:120, height:35);
67     signUpBtn.setBackground(Color.decode(nm:"#80FF00"));
68     signUpBtn.setForeground(Color.BLACK);
69     signUpBtn.setFocusPainted(b:false);
70     leftPanel.add(signUpBtn);
71
72     JButton loginBtn = new JButton(text:"Login");
73     loginBtn.setBounds(x:190, y:220, width:120, height:35);
74     loginBtn.setBackground(Color.LIGHT_GRAY);
75     loginBtn.setForeground(Color.BLACK);
76     loginBtn.setFocusPainted(b:false);
77     leftPanel.add(loginBtn);
78
79     loginBtn.addActionListener(e -> {
80         String username = emailField.getText();
81         String password = new String(passwordField.getPassword());
82
83         Authenticator auth = new Authenticator();
84         try {
85             User user = auth.login(username, password);
86             if (user != null) {
87                 JOptionPane.showMessageDialog(parentComponent:null, "Login Berhasil sebagai " + user.getRole());
88
89                 UserRoleHandler handler;
90                 if (user.getRole().equalsIgnoreCase(anotherString:"admin")) {
91                     handler = new AdminHandler();
92                 } else {
93                     handler = new CustomerHandler();
94                 }
95
96                 handler.openDashboard(user);
97                 dispose();
98             } else {
99                 JOptionPane.showMessageDialog(parentComponent:null, message:"Username atau Password salah!");
100            }
101        } catch (SQLException ex) {
102            ex.printStackTrace();
103            JOptionPane.showMessageDialog(parentComponent:null, message:"Koneksi ke database gagal!");
104        }
105    });
106
107
108    signUpBtn.addActionListener(e -> {
109        new RegisterFrame();
110        dispose();
111    });
112
113    JPanel rightPanel = new JPanel();
114    rightPanel.setBackground(Color.decode(nm:"#222222"));
115    rightPanel.setLayout(new GridBagLayout());
116
117    ImageIcon logoIcon = new ImageIcon(getClass().getResource(name:"/rental/logo.png"));
118    Image scaledImage = logoIcon.getImage().getScaledInstance(width:250, height:250, Image.SCALE_SMOOTH);
119    ImageIcon scaledIcon = new ImageIcon(scaledImage);
120    JLabel logoLabel = new JLabel(scaledIcon);
121    rightPanel.add(logoLabel);
122
123    contentPanel.add(leftPanel);
124    contentPanel.add(rightPanel);
125    add(contentPanel);
126
127    setVisible(b:true);
128}
129}
130

```

Penjelasan :

Kode program ini merupakan kode program login dalam sistem sewa barang rental properti. Kelas LoginFrame merupakan turunan dari JFrame dan menampilkan jendela login dengan dua panel:

- Panel kiri berisi form login (username, password), tombol *Login*, *Sign Up*, dan link *Forgot Password*.
- Panel kanan menampilkan logo aplikasi.

Saat pengguna menekan tombol Login, aplikasi akan:

1. Mengambil data input dari form.
2. Menggunakan kelas Authenticator untuk memverifikasi kredensial ke database.
3. Jika berhasil, aplikasi akan:
  - a. Menampilkan pesan login berhasil,
  - b. Mengecek peran pengguna (admin atau customer),
  - c. Mengarahkan pengguna ke dashboard yang sesuai (AdminHandler atau CustomerHandler).
4. Jika gagal, aplikasi akan menampilkan pesan kesalahan.

Tombol Sign Up akan membuka form pendaftaran (RegisterFrame), dan Forgot Password akan membuka form pemulihan sandi (ForgotPasswordFrame).

- Register

```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class RegisterFrame extends JFrame {
8     private JTextField namafield;
9     private JTextField usernamefield;
10    private JPasswordField passwordfield;
11    private JTextArea alamatArea;
12    private JTextField notelpfield;
13
14    public RegisterFrame() {
15        setTitle("Daftar Akun - Rental Properti");
16        setSize(800, height:450);
17        setDefaultCloseOperation(EXIT_ON_CLOSE);
18        setLocationRelativeTo(null);
19        setResizable(resizable:false);
20
21        // ===== Panel Utama - 2 Kolom
22        JPanel contentPanel = new JPanel(new GridLayout(rows:1, cols:2));
23        add(contentPanel);
24
25        // ===== Panel Kiri: Logo =====
26        JPanel leftPanel = new JPanel();
27        leftPanel.setBackground(Color.decode("#F1F1F1"));
28        leftPanel.setLayout(new GridBagLayout());
29
30        ImageIcon logoIcon = new ImageIcon(getClass().getResource(name:"/rental/logo.png"));
31        Image scaledImage = logoIcon.getImage().getScaledInstance(width:250, height:250, Image.SCALE_SMOOTH);
32        JLabel logoLabel = new JLabel(new ImageIcon(scaledImage));
33        leftPanel.add(logoLabel);
34
35        // ===== Panel Kanan: Form =====
36        JPanel rightPanel = new JPanel(layout:null);
37        rightPanel.setBackground(Color.decode("#EAEAEA"));
38
39        JLabel titleLabel = new JLabel(text:"Daftar Akun Baru");
40        titleLabel.setFont(new Font(name:"SansSerif", Font.BOLD, size:20));
41        titleLabel.setForeground(Color.WHITE);
42        titleLabel.setBounds(x:140, y:20, width:200, height:30);
43        rightPanel.add(titleLabel);
44
45        JLabel nameLabel = new JLabel(text:"Nama Lengkap:");
46        nameLabel.setBounds(x:50, y:70, width:100, height:20);
47        styleLabel(nameLabel, rightPanel);
48
49        namaField = new JTextField();
50        namaField.setBounds(x:160, y:70, width:200, height:25);
51        rightPanel.add(namaField);
52

```

```

53     JLabel usernameLabel = new JLabel(text:"Username:");
54     usernameLabel.setBounds(x:50, y:110, width:100, height:20);
55     styleLabel(usernameLabel, rightPanel);
56
57     usernameField = new JTextField();
58     usernameField.setBounds(x:160, y:110, width:200, height:25);
59     rightPanel.add(usernameField);
60
61     JLabel passwordLabel = new JLabel(text:"Password:");
62     passwordLabel.setBounds(x:50, y:150, width:100, height:20);
63     styleLabel(passwordLabel, rightPanel);
64
65     passwordField = new JPasswordField();
66     passwordField.setBounds(x:160, y:150, width:200, height:25);
67     rightPanel.add(passwordField);
68
69     JLabel alamatLabel = new JLabel(text:"Alamat:");
70     alamatLabel.setBounds(x:50, y:190, width:100, height:20);
71     styleLabel(alamatLabel, rightPanel);
72
73     alamatArea = new JTextArea();
74     alamatArea.setLineWrap(wrap:true);
75     alamatArea.setWrapStyleWord(word:true);
76     JScrollPane scrollPane = new JScrollPane(alamatArea);
77     scrollPane.setBounds(x:160, y:190, width:200, height:50);
78     rightPanel.add(scrollPane);
79
80     JLabel telpLabel = new JLabel(text:"No. Telepon:");
81     telpLabel.setBounds(x:50, y:250, width:100, height:20);
82     styleLabel(telpLabel, rightPanel);
83
84     noTelpField = new JTextField();
85     noTelpField.setBounds(x:160, y:250, width:200, height:25);
86     rightPanel.add(noTelpField);
87
88     JButton daftarBtn = new JButton(text:"Daftar");
89     daftarBtn.setBounds(x:160, y:300, width:90, height:30);
90     daftarBtn.setBackground(Color.decode("#46A003"));
91     daftarBtn.setForeground(Color.WHITE);
92     daftarBtn.setFocusPainted(false);
93     daftarBtn.addActionListener(e -> registerUser());
94     rightPanel.add(daftarBtn);
95
96     JButton loginBtn = new JButton(text:"Login");
97     loginBtn.setBounds(x:270, y:300, width:90, height:30);
98     loginBtn.setBackground(Color.LIGHT_GRAY);
99     loginBtn.setFocusPainted(false);
100    loginBtn.addActionListener(e -> {
101        new LoginFrame();
102        dispose();
103    });
104    rightPanel.add(loginBtn);
105
106    // Gabungkan ke frame
107    contentPanel.add(leftPanel);
108    contentPanel.add(rightPanel);
109    setVisible(b:true);
110 }
111
112 private void styleLabel(JLabel label, JPanel panel) {
113     label.setForeground(Color.decode("#80FF00"));
114     panel.add(label);
115 }
116
117 private void registerUser() {
118     String nama = namaField.getText().trim();
119     String username = usernameField.getText().trim();
120     String password = new String(passwordField.getPassword());
121     String alamat = alamatArea.getText().trim();
122     String noTelp = noTelpField.getText().trim();
123
124     if (nama.isEmpty() || username.isEmpty() || password.isEmpty() || alamat.isEmpty() || noTelp.isEmpty()) {
125         JOptionPane.showMessageDialog(this, message."Semua field harus diisi!");
126         return;
127     }
128
129     User newUser = new User(nama, username, password, alamat, noTelp);
130     RegisterHandler handler = new RegisterHandler();
131
132     try {
133         boolean success = handler.register(newUser);
134         if (success) {
135             JOptionPane.showMessageDialog(this, message."Pendaftaran berhasil! Silakan login.");
136             new LoginFrame();
137             dispose();
138         } else {
139             JOptionPane.showMessageDialog(this, message."Pendaftaran gagal.");
140         }
141     } catch (SQLIntegrityConstraintViolationException e) {
142         JOptionPane.showMessageDialog(this, message."Username sudah digunakan.");
143     } catch (SQLException e) {
144         e.printStackTrace();
145         JOptionPane.showMessageDialog(this, message."Kesalahan koneksi database.");
146     }
147 }
148
149 }

```

Penjelasan :

Kelas RegisterFrame adalah turunan dari JFrame yang menampilkan dua panel:

1. Panel kiri berisi logo aplikasi.
2. Panel kanan berisi form pendaftaran, yang mencakup field:
  - o Nama lengkap
  - o Username
  - o Password
  - o Alamat
  - o Nomor telepon

Pengguna dapat mengisi semua field dan menekan tombol "Daftar ". Ketika tombol ini ditekan:

1. Data input dari form dikumpulkan dan divalidasi agar tidak kosong.
2. Data dikemas menjadi objek User.
3. Objek RegisterHandler digunakan untuk menyimpan data user ke database melalui method register().
4. Jika berhasil, pengguna akan diarahkan ke halaman login.
5. Jika terjadi kesalahan seperti username sudah digunakan (SQLIntegrityConstraintViolationException), akan muncul pesan peringatan.

Tombol "Login" juga tersedia untuk kembali ke form login (LoginFrame).

- Forget Password

```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class ForgotPasswordFrame extends JFrame {
8     private JTextField usernameField;
9     private JPasswordField newPasswordField;
10
11     public ForgotPasswordFrame() {
12         setTitle("🔒 Reset Password");
13         setSize(400, height:250);
14         setLocationRelativeTo(null);
15         setResizable(resizable:false);
16         setDefaultCloseOperation(DISPOSE_ON_CLOSE);
17         setLayout(manager:null);
18         getContentPane().setBackground(Color.decode(nm:"#222222"));
19
20         JLabel usernameLabel = new JLabel(text:"Username:");
21         usernameLabel.setBounds(x:40, y:40, width:100, height:25);
22         usernameLabel.setForeground(Color.GREEN);
23         add(usernameLabel);
24
25         JTextField usernameField = new JTextField();
26         usernameField.setBounds(x:150, y:40, width:180, height:25);
27         add(usernameField);
28
29         JLabel passLabel = new JLabel(text:"Password Baru:");
30         passLabel.setBounds(x:40, y:80, width:100, height:25);
31         passLabel.setForeground(Color.GREEN);
32         add(passLabel);
33
34         JPasswordField newPasswordField = new JPasswordField();
35         newPasswordField.setBounds(x:150, y:80, width:180, height:25);
36         add(newPasswordField);
37
38         JButton resetBtn = new JButton(text:"Reset Password");
39         resetBtn.setBounds(x:130, y:130, width:140, height:30);
40         resetBtn.setBackground(Color.decode(nm:"#80FF00"));
41         resetBtn.setFocusPainted(b:false);
42         resetBtn.addActionListener(e -> setPassword());
43         add(resetBtn);
44
45         setVisible(b:true);
46     }
47
48     private void setPassword() {
49         String username = usernameField.getText().trim();
50         String newPassword = new String(newPasswordField.getPassword());
51
52         if (username.isEmpty() || newPassword.isEmpty()) {
53             JOptionPane.showMessageDialog(this, message:"Semua field wajib diisi!");
54             return;
55         }
56
57         try (Connection conn = KoneksiDatabase.getConnection();
58              PreparedStatement stmt = conn.prepareStatement(
59                  sql:"UPDATE user SET password = ? WHERE username = ?")) {
60
61             stmt.setString(parameterIndex:1, newPassword);
62             stmt.setString(parameterIndex:2, username);
63
64             int updated = stmt.executeUpdate();
65             if (updated > 0) {
66                 JOptionPane.showMessageDialog(this, message:"Password berhasil direset!");
67                 dispose();
68             } else {
69                 JOptionPane.showMessageDialog(this, message:"Username tidak ditemukan.");
70             }
71
72         } catch (SQLException ex) {
73             ex.printStackTrace();
74             JOptionPane.showMessageDialog(this, message:"Terjadi kesalahan saat mengakses database.");
75         }
76     }
77 }
78 
```

Penjelasan :

Ini merupakan kode program untuk reset password dengan nama Kelas

`ForgotPasswordFrame` yang merupakan tampilan antarmuka (GUI) berbasis `JFrame` yang berfungsi untuk:

1. Menginput username pengguna
2. Menginput password baru yang ingin digunakan

Tampilan ini terdiri dari:

1. Dua JLabel dan dua field input (JTextField untuk username dan JPasswordField untuk password baru)
2. Satu tombol "Reset Password" berwarna hijau terang

Saat tombol reset password ditekan:

1. Program akan memeriksa apakah semua field telah diisi.
2. Jika ya, maka akan dilakukan koneksi ke database melalui KoneksiDatabase.getConnection().
3. Menggunakan PreparedStatement, program menjalankan query UPDATE untuk mengubah password user yang sesuai dengan username.
4. Jika query berhasil dan username ditemukan, akan muncul notifikasi sukses dan jendela ditutup.
5. Jika username tidak ditemukan, akan ditampilkan pesan kesalahan.
6. Jika terjadi SQLException, akan ditangani dan ditampilkan sebagai pesan kesalahan.

- User.java

```

1  package rental;
2
3  public class User {
4      private int id;
5      private String username;
6      private String password;
7      private String role;
8      private String nama;
9      private String alamat;
10     private String noTelp;
11
12     // Constructor untuk login result
13     public User(int id, String username, String role) {
14         this.id = id;
15         this.username = username;
16         this.role = role;
17     }
18
19     // Constructor untuk register
20     public User(String nama, String username, String password, String alamat, String noTelp) {
21         this.nama = nama;
22         this.username = username;
23         this.password = password;
24         this.alamat = alamat;
25         this.noTelp = noTelp;
26         this.role = "user"; // default role
27     }
28
29     // Getter
30     public int getId() { return id; }
31     public String getUsername() { return username; }
32     public String getPassword() { return password; }
33     public String getRole() { return role; }
34     public String getName() { return nama; }
35     public String getAlamat() { return alamat; }
36     public String getNoTelp() { return noTelp; }
37 }
38

```

Penjelasan :

Kelas ini berfungsi sebagai model data yang menyimpan informasi user, seperti:

1. id (untuk identifikasi di database),
2. username,
3. password,
4. role (misalnya: *admin* atau *user*),
5. nama lengkap,
6. alamat, dan
7. noTelp (nomor telepon).

Constructor

1. Constructor pertama digunakan saat login, hanya memerlukan id, username, dan role.
2. Constructor kedua digunakan saat registrasi, menerima data lengkap seperti nama, username, password, alamat, dan noTelp, serta otomatis memberi role "user".

Disediakan method getter untuk mengakses nilai masing-masing atribut. Tidak ada setter, sehingga objek bersifat read-only setelah dibuat (immutable dari luar kelas).

- Authenticator.java

```
1 package rental;
2
3 import java.sql.*;
4
5 public class Authenticator {
6
7     public User login(String username, String password) throws SQLException {
8         try (Connection conn = KoneksiDatabase.getConnection();
9              PreparedStatement stmt = conn.prepareStatement(
10                  sql:"SELECT * FROM user WHERE username = ? AND password = ?")) {
11
12             stmt.setString(parameterIndex:1, username);
13             stmt.setString(parameterIndex:2, password);
14
15             try (ResultSet rs = stmt.executeQuery()) {
16                 if (rs.next()) {
17                     int id = rs.getInt(columnLabel:"idUser");
18                     String role = rs.getString(columnLabel:"role");
19                     return new User(id, username, role);
20                 }
21             }
22         }
23         return null;
24     }
25 }
26 }
```

Penjelasan :

Ini merupakan kode program bagian backend yang mana akan memverifikasi username dan password sebelum user masuk kedalam menu dashboard. Kode program ini tidak memiliki GUI, hanya berperan sebagai logical saja untuk mengatasi backend

Pengujian Login and Register Section – BlackBox – Testing

no	Skenario	Test Case	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1	Uji login	Input form username	Form username berhasil diinput dengan kata	Form username berhasil diinput	valid
2	Uji login	Input form password	Form password bisa diinput	Form password berhasil diinput	valid
3	Uji login	Klik button login dengan user dan password benar	User berhasil login dan diarahkan ke dashboard	User berhasil login dan diarahkan ke dashboard	valid
4	Uji login	Isi form username, password dengan salah data. Dan klik button login	Login gagal karena username dan password salah	Login gagal karena username dan password salah	valid
5	Uji lupa password	Isi form username	Username berhasil diisi	Username berhasil diisi	valid
6	Uji lupa password	Isi form password baru	Berhasil menginputkan password baru	Berhasil input password yang baru	Valid

7	Uji lupa password	Klik button reset password	Berhasil membuat password baru	Berhasil membuat password baru	Valid
8	Uji register	Isi form nama lengkap	Berhasil mengisi nama	Form berhasil diisi dengan nama	valid
9	Uji register	Isi form username	Berhasil mengisi username	Form berhasil diisi dengan username yang dibuat	valid
10	Uji register	Isi form password	Berhasil mengisi password	Form berhasil diisi dengan password yang baru dibuat	valid
11	Uji register	Isi form alamat	Berhasil mengisi alamat	Form berhasil diisi dengan alamat yang baru dibuat	valid
12	Uji register	Isi form no telepon	Berhasil mengisi no telepon	Form berhasil diisi dengan no telepon dalam bentuk integer	valid
18	Uji register	Klik button daftar	Data yang diinput berhasil didaftarkan	Data berhasil didaftar dan disimpan dalam database	valid

#### 2.4.2 Admin Section

- Admin Dashboard  
Kode Program



## Output



### Penjelasan

Kode program ini merupakan halaman dashboard admin. Saat program dijalankan, akan tampil jendela berjudul "*Admin Dashboard - VenueVibe*" dengan tampilan yang terdiri dari:

1. Header di atas berisi judul dan tombol "Logout".
2. Sidebar di sisi kiri dengan tiga menu:
  - a. Dashboard
  - b. Kelola Data Barang
  - c. Konfirmasi Pesanan
3. Main Panel di bagian tengah untuk menampilkan konten berdasarkan menu yang dipilih.

Secara default, saat aplikasi dibuka, menu "Dashboard" akan ditampilkan terlebih dahulu berisi profil perusahaan VenueVibe. Jika pengguna mengklik:

- Kelola Data Barang, maka BarangAdmin panel akan ditampilkan.
  - Konfirmasi Pesanan, maka KonfirmasiPesananPanel akan ditampilkan.
- 
- Kelola Data Barang

### Kode Program



## Output

ID	Nama	Kategori	Harga Sewa	Stok	Gambar	Aksi
1	Kursi Lipat	Furniture	Rp. 10.000	9		<button>Edit</button> <button>Hapus</button>
2	Lampu Panggung	Elektronik	Rp. 30.000	6		<button>Edit</button> <button>Hapus</button>
3	Panggung Portable	Peralatan	Rp. 500.000	5		<button>Edit</button> <button>Hapus</button>
4	Sound System	Elektronik	Rp. 750.000	10		<button>Edit</button> <button>Hapus</button>
5	Proyektor	Elektronik	Rp. 25.000	8		<button>Edit</button> <button>Hapus</button>
6	Mikrofon	Peralatan	Rp. 100.000	10		<button>Edit</button> <button>Hapus</button>

[Tambah Barang Baru](#)

## Penjelasan

Kode program ini merupakan untuk manajemen data barang pada sistem penyewaan barang. Outputnya adalah menampilkan tabel data barang dari database, lengkap dengan fitur:

- Tampilkan semua barang dari tabel barang di database.
- Tampilkan gambar barang dalam bentuk thumbnail.
- Tombol "Tambah Barang Baru" untuk membuka form input barang.
- Kolom "Aksi" berisi tombol Edit dan Hapus untuk setiap baris barang.

Saat aplikasi dijalankan, user akan melihat tabel dengan data barang seperti ID, nama, kategori, harga sewa, stok, dan gambar. Admin dapat:

- Menambahkan barang baru melalui form input.
- Mengedit data barang yang sudah ada (dengan mengisi form yang datanya otomatis dimuat).
- Menghapus barang dengan konfirmasi dialog.
- Semua perubahan disimpan ke database dan langsung direfleksikan ke tampilan.

- Konfirmasi Pesanan

## Kode Program

```

1 package com.mkyong.core;
2
3 import java.util.*;
4
5 import org.junit.Test;
6 import org.junit.runner.RunWith;
7 import org.junit.runners.Parameterized;
8 import org.junit.runners.Parameterized.Parameters;
9
10 import static org.junit.Assert.*;
11
12 /**
13  * Representing Data Reader abstractions
14  */
15 class DataReader {
16
17     protected List<Map<String, Object>> rows;
18
19     protected String schema;
20
21     protected String title;
22
23     protected String author;
24
25     protected String description;
26
27     protected String type;
28
29     protected String encoding;
30
31     protected String method;
32
33 }
34
35 /**
36  * DataReader interface
37  */
38 interface DataReader {
39
40     void setRows(List<Map<String, Object>> rows);
41
42     void setTitle(String title);
43
43     void setAuthor(String author);
44
45     void setDescription(String description);
46
47     void setType(String type);
48
49     void setEncoding(String encoding);
50
51     void setMethod(String method);
52
53 }
54
55 /**
56  * Reader
57  */
58 class Reader implements DataReader {
59
60     private List<Map<String, Object>> rows;
61
62     private String schema;
63
64     private String title;
65
66     private String author;
67
68     private String description;
69
70     private String type;
71
72     private String encoding;
73
74     private String method;
75
76
77     public void setRows(List<Map<String, Object>> rows) {
78         this.rows = rows;
79     }
80
81     public void setTitle(String title) {
82         this.title = title;
83     }
84
85     public void setAuthor(String author) {
86         this.author = author;
87     }
88
89     public void setDescription(String description) {
90         this.description = description;
91     }
92
93     public void setType(String type) {
94         this.type = type;
95     }
96
97     public void setEncoding(String encoding) {
98         this.encoding = encoding;
99     }
100    public void setMethod(String method) {
101        this.method = method;
102    }
103
104 }
105
106 /**
107  * Writer
108  */
109 class Writer implements DataReader {
110
111     private List<Map<String, Object>> rows;
112
113     private String schema;
114
115     private String title;
116
117     private String author;
118
119     private String description;
120
121     private String type;
122
123     private String encoding;
124
125     private String method;
126
127
128     public void setRows(List<Map<String, Object>> rows) {
129         this.rows = rows;
130     }
131
132     public void setTitle(String title) {
133         this.title = title;
134     }
135
136     public void setAuthor(String author) {
137         this.author = author;
138     }
139
140     public void setDescription(String description) {
141         this.description = description;
142     }
143
144     public void setType(String type) {
145         this.type = type;
146     }
147
148     public void setEncoding(String encoding) {
149         this.encoding = encoding;
150     }
151
152     public void setMethod(String method) {
153         this.method = method;
154     }
155
156 }
157
158 /**
159  * Data Reader Implementation
160  */
161 class DataReaderImplementation extends DataReader {
162
163     private List<Map<String, Object>> rows;
164
165     private String schema;
166
167     private String title;
168
169     private String author;
170
171     private String description;
172
173     private String type;
174
175     private String encoding;
176
177     private String method;
178
179
180     public void setRows(List<Map<String, Object>> rows) {
181         this.rows = rows;
182     }
183
184     public void setTitle(String title) {
185         this.title = title;
186     }
187
188     public void setAuthor(String author) {
189         this.author = author;
190     }
191
192     public void setDescription(String description) {
193         this.description = description;
194     }
195
196     public void setType(String type) {
197         this.type = type;
198     }
199
200     public void setEncoding(String encoding) {
201         this.encoding = encoding;
202     }
203
204     public void setMethod(String method) {
205         this.method = method;
206     }
207
208 }
209
210 /**
211  * Data Reader Implementation
212  */
213 class DataReaderImplementation2 extends DataReader {
214
215     private List<Map<String, Object>> rows;
216
217     private String schema;
218
219     private String title;
220
221     private String author;
222
223     private String description;
224
225     private String type;
226
227     private String encoding;
228
229     private String method;
230
231
232     public void setRows(List<Map<String, Object>> rows) {
233         this.rows = rows;
234     }
235
236     public void setTitle(String title) {
237         this.title = title;
238     }
239
240     public void setAuthor(String author) {
241         this.author = author;
242     }
243
244     public void setDescription(String description) {
245         this.description = description;
246     }
247
248     public void setType(String type) {
249         this.type = type;
250     }
251
252     public void setEncoding(String encoding) {
253         this.encoding = encoding;
254     }
255
256     public void setMethod(String method) {
257         this.method = method;
258     }
259
260 }
261
262 /**
263  * Data Reader Implementation
264  */
265 class DataReaderImplementation3 extends DataReader {
266
267     private List<Map<String, Object>> rows;
268
269     private String schema;
270
271     private String title;
272
273     private String author;
274
275     private String description;
276
277     private String type;
278
279     private String encoding;
280
281     private String method;
282
283
284     public void setRows(List<Map<String, Object>> rows) {
285         this.rows = rows;
286     }
287
288     public void setTitle(String title) {
289         this.title = title;
290     }
291
292     public void setAuthor(String author) {
293         this.author = author;
294     }
295
296     public void setDescription(String description) {
297         this.description = description;
298     }
299
300     public void setType(String type) {
301         this.type = type;
302     }
303
304     public void setEncoding(String encoding) {
305         this.encoding = encoding;
306     }
307
308     public void setMethod(String method) {
309         this.method = method;
310     }
311
312 }
313
314 /**
315  * Data Reader Implementation
316  */
317 class DataReaderImplementation4 extends DataReader {
318
319     private List<Map<String, Object>> rows;
320
321     private String schema;
322
323     private String title;
324
325     private String author;
326
327     private String description;
328
329     private String type;
330
331     private String encoding;
332
333     private String method;
334
335
336     public void setRows(List<Map<String, Object>> rows) {
337         this.rows = rows;
338     }
339
340     public void setTitle(String title) {
341         this.title = title;
342     }
343
344     public void setAuthor(String author) {
345         this.author = author;
346     }
347
348     public void setDescription(String description) {
349         this.description = description;
350     }
351
352     public void setType(String type) {
353         this.type = type;
354     }
355
356     public void setEncoding(String encoding) {
357         this.encoding = encoding;
358     }
359
360     public void setMethod(String method) {
361         this.method = method;
362     }
363
364 }
365
366 /**
367  * Data Reader Implementation
368  */
369 class DataReaderImplementation5 extends DataReader {
370
371     private List<Map<String, Object>> rows;
372
373     private String schema;
374
375     private String title;
376
377     private String author;
378
379     private String description;
380
381     private String type;
382
383     private String encoding;
384
385     private String method;
386
387
388     public void setRows(List<Map<String, Object>> rows) {
389         this.rows = rows;
390     }
391
392     public void setTitle(String title) {
393         this.title = title;
394     }
395
396     public void setAuthor(String author) {
397         this.author = author;
398     }
399
400     public void setDescription(String description) {
401         this.description = description;
402     }
403
404     public void setType(String type) {
405         this.type = type;
406     }
407
408     public void setEncoding(String encoding) {
409         this.encoding = encoding;
410     }
411
412     public void setMethod(String method) {
413         this.method = method;
414     }
415
416 }
417
418 /**
419  * Data Reader Implementation
420  */
421 class DataReaderImplementation6 extends DataReader {
422
423     private List<Map<String, Object>> rows;
424
425     private String schema;
426
427     private String title;
428
429     private String author;
430
431     private String description;
432
433     private String type;
434
435     private String encoding;
436
437     private String method;
438
439
440     public void setRows(List<Map<String, Object>> rows) {
441         this.rows = rows;
442     }
443
444     public void setTitle(String title) {
445         this.title = title;
446     }
447
448     public void setAuthor(String author) {
449         this.author = author;
450     }
451
452     public void setDescription(String description) {
453         this.description = description;
454     }
455
456     public void setType(String type) {
457         this.type = type;
458     }
459
460     public void setEncoding(String encoding) {
461         this.encoding = encoding;
462     }
463
464     public void setMethod(String method) {
465         this.method = method;
466     }
467
468 }
469
470 /**
471  * Data Reader Implementation
472  */
473 class DataReaderImplementation7 extends DataReader {
474
475     private List<Map<String, Object>> rows;
476
477     private String schema;
478
479     private String title;
480
481     private String author;
482
483     private String description;
484
485     private String type;
486
487     private String encoding;
488
489     private String method;
490
491
492     public void setRows(List<Map<String, Object>> rows) {
493         this.rows = rows;
494     }
495
496     public void setTitle(String title) {
497         this.title = title;
498     }
499
500     public void setAuthor(String author) {
501         this.author = author;
502     }
503
504     public void setDescription(String description) {
505         this.description = description;
506     }
507
508     public void setType(String type) {
509         this.type = type;
510     }
511
512     public void setEncoding(String encoding) {
513         this.encoding = encoding;
514     }
515
516     public void setMethod(String method) {
517         this.method = method;
518     }
519
520 }
521
522 /**
523  * Data Reader Implementation
524  */
525 class DataReaderImplementation8 extends DataReader {
526
527     private List<Map<String, Object>> rows;
528
529     private String schema;
530
531     private String title;
532
533     private String author;
534
535     private String description;
536
537     private String type;
538
539     private String encoding;
540
541     private String method;
542
543
544     public void setRows(List<Map<String, Object>> rows) {
545         this.rows = rows;
546     }
547
548     public void setTitle(String title) {
549         this.title = title;
550     }
551
552     public void setAuthor(String author) {
553         this.author = author;
554     }
555
556     public void setDescription(String description) {
557         this.description = description;
558     }
559
560     public void setType(String type) {
561         this.type = type;
562     }
563
564     public void setEncoding(String encoding) {
565         this.encoding = encoding;
566     }
567
568     public void setMethod(String method) {
569         this.method = method;
570     }
571
572 }
573
574 /**
575  * Data Reader Implementation
576  */
577 class DataReaderImplementation9 extends DataReader {
578
579     private List<Map<String, Object>> rows;
580
581     private String schema;
582
583     private String title;
584
585     private String author;
586
587     private String description;
588
589     private String type;
590
591     private String encoding;
592
593     private String method;
594
595
596     public void setRows(List<Map<String, Object>> rows) {
597         this.rows = rows;
598     }
599
600     public void setTitle(String title) {
601         this.title = title;
602     }
603
604     public void setAuthor(String author) {
605         this.author = author;
606     }
607
608     public void setDescription(String description) {
609         this.description = description;
610     }
611
612     public void setType(String type) {
613         this.type = type;
614     }
615
616     public void setEncoding(String encoding) {
617         this.encoding = encoding;
618     }
619
620     public void setMethod(String method) {
621         this.method = method;
622     }
623
624 }
625
626 /**
627  * Data Reader Implementation
628  */
629 class DataReaderImplementation10 extends DataReader {
630
631     private List<Map<String, Object>> rows;
632
633     private String schema;
634
635     private String title;
636
637     private String author;
638
639     private String description;
640
641     private String type;
642
643     private String encoding;
644
645     private String method;
646
647
648     public void setRows(List<Map<String, Object>> rows) {
649         this.rows = rows;
650     }
651
652     public void setTitle(String title) {
653         this.title = title;
654     }
655
656     public void setAuthor(String author) {
657         this.author = author;
658     }
659
660     public void setDescription(String description) {
661         this.description = description;
662     }
663
664     public void setType(String type) {
665         this.type = type;
666     }
667
668     public void setEncoding(String encoding) {
669         this.encoding = encoding;
670     }
671
672     public void setMethod(String method) {
673         this.method = method;
674     }
675
676 }
677
678 /**
679  * Data Reader Implementation
680  */
681 class DataReaderImplementation11 extends DataReader {
682
683     private List<Map<String, Object>> rows;
684
685     private String schema;
686
687     private String title;
688
689     private String author;
690
691     private String description;
692
693     private String type;
694
695     private String encoding;
696
697     private String method;
698
699
700     public void setRows(List<Map<String, Object>> rows) {
701         this.rows = rows;
702     }
703
704     public void setTitle(String title) {
705         this.title = title;
706     }
707
708     public void setAuthor(String author) {
709         this.author = author;
710     }
711
712     public void setDescription(String description) {
713         this.description = description;
714     }
715
716     public void setType(String type) {
717         this.type = type;
718     }
719
720     public void setEncoding(String encoding) {
721         this.encoding = encoding;
722     }
723
724     public void setMethod(String method) {
725         this.method = method;
726     }
727
728 }
729
730 /**
731  * Data Reader Implementation
732  */
733 class DataReaderImplementation12 extends DataReader {
734
735     private List<Map<String, Object>> rows;
736
737     private String schema;
738
739     private String title;
740
741     private String author;
742
743     private String description;
744
745     private String type;
746
747     private String encoding;
748
749     private String method;
750
751
752     public void setRows(List<Map<String, Object>> rows) {
753         this.rows = rows;
754     }
755
756     public void setTitle(String title) {
757         this.title = title;
758     }
759
760     public void setAuthor(String author) {
761         this.author = author;
762     }
763
764     public void setDescription(String description) {
765         this.description = description;
766     }
767
768     public void setType(String type) {
769         this.type = type;
770     }
771
772     public void setEncoding(String encoding) {
773         this.encoding = encoding;
774     }
775
776     public void setMethod(String method) {
777         this.method = method;
778     }
779
780 }
781
782 /**
783  * Data Reader Implementation
784  */
785 class DataReaderImplementation13 extends DataReader {
786
787     private List<Map<String, Object>> rows;
788
789     private String schema;
790
791     private String title;
792
793     private String author;
794
795     private String description;
796
797     private String type;
798
799     private String encoding;
800
801     private String method;
802
803
804     public void setRows(List<Map<String, Object>> rows) {
805         this.rows = rows;
806     }
807
808     public void setTitle(String title) {
809         this.title = title;
810     }
811
812     public void setAuthor(String author) {
813         this.author = author;
814     }
815
816     public void setDescription(String description) {
817         this.description = description;
818     }
819
820     public void setType(String type) {
821         this.type = type;
822     }
823
824     public void setEncoding(String encoding) {
825         this.encoding = encoding;
826     }
827
828     public void setMethod(String method) {
829         this.method = method;
830     }
831
832 }
833
834 /**
835  * Data Reader Implementation
836  */
837 class DataReaderImplementation14 extends DataReader {
838
839     private List<Map<String, Object>> rows;
840
841     private String schema;
842
843     private String title;
844
845     private String author;
846
847     private String description;
848
849     private String type;
850
851     private String encoding;
852
853     private String method;
854
855
856     public void setRows(List<Map<String, Object>> rows) {
857         this.rows = rows;
858     }
859
860     public void setTitle(String title) {
861         this.title = title;
862     }
863
864     public void setAuthor(String author) {
865         this.author = author;
866     }
867
868     public void setDescription(String description) {
869         this.description = description;
870     }
871
872     public void setType(String type) {
873         this.type = type;
874     }
875
876     public void setEncoding(String encoding) {
877         this.encoding = encoding;
878     }
879
880     public void setMethod(String method) {
881         this.method = method;
882     }
883
884 }
885
886 /**
887  * Data Reader Implementation
888  */
889 class DataReaderImplementation15 extends DataReader {
890
891     private List<Map<String, Object>> rows;
892
893     private String schema;
894
895     private String title;
896
897     private String author;
898
899     private String description;
900
901     private String type;
902
903     private String encoding;
904
905     private String method;
906
907
908     public void setRows(List<Map<String, Object>> rows) {
909         this.rows = rows;
910     }
911
912     public void setTitle(String title) {
913         this.title = title;
914     }
915
916     public void setAuthor(String author) {
917         this.author = author;
918     }
919
920     public void setDescription(String description) {
921         this.description = description;
922     }
923
924     public void setType(String type) {
925         this.type = type;
926     }
927
928     public void setEncoding(String encoding) {
929         this.encoding = encoding;
930     }
931
932     public void setMethod(String method) {
933         this.method = method;
934     }
935
936 }
937
938 /**
939  * Data Reader Implementation
940  */
941 class DataReaderImplementation16 extends DataReader {
942
943     private List<Map<String, Object>> rows;
944
945     private String schema;
946
947     private String title;
948
949     private String author;
950
951     private String description;
952
953     private String type;
954
955     private String encoding;
956
957     private String method;
958
959
960     public void setRows(List<Map<String, Object>> rows) {
961         this.rows = rows;
962     }
963
964     public void setTitle(String title) {
965         this.title = title;
966     }
967
968     public void setAuthor(String author) {
969         this.author = author;
970     }
971
972     public void setDescription(String description) {
973         this.description = description;
974     }
975
976     public void setType(String type) {
977         this.type = type;
978     }
979
980     public void setEncoding(String encoding) {
981         this.encoding = encoding;
982     }
983
984     public void setMethod(String method) {
985         this.method = method;
986     }
987
988 }
989
990 /**
991  * Data Reader Implementation
992  */
993 class DataReaderImplementation17 extends DataReader {
994
995     private List<Map<String, Object>> rows;
996
997     private String schema;
998
999     private String title;
1000
1001     private String author;
1002
1003     private String description;
1004
1005     private String type;
1006
1007     private String encoding;
1008
1009     private String method;
1010
1011
1012     public void setRows(List<Map<String, Object>> rows) {
1013         this.rows = rows;
1014     }
1015
1016     public void setTitle(String title) {
1017         this.title = title;
1018     }
1019
1020     public void setAuthor(String author) {
1021         this.author = author;
1022     }
1023
1024     public void setDescription(String description) {
1025         this.description = description;
1026     }
1027
1028     public void setType(String type) {
1029         this.type = type;
1030     }
1031
1032     public void setEncoding(String encoding) {
1033         this.encoding = encoding;
1034     }
1035
1036     public void setMethod(String method) {
1037         this.method = method;
1038     }
1039
1040 }
1041
1042 /**
1043  * Data Reader Implementation
1044  */
1045 class DataReaderImplementation18 extends DataReader {
1046
1047     private List<Map<String, Object>> rows;
1048
1049     private String schema;
1050
1051     private String title;
1052
1053     private String author;
1054
1055     private String description;
1056
1057     private String type;
1058
1059     private String encoding;
1060
1061     private String method;
1062
1063
1064     public void setRows(List<Map<String, Object>> rows) {
1065         this.rows = rows;
1066     }
1067
1068     public void setTitle(String title) {
1069         this.title = title;
1070     }
1071
1072     public void setAuthor(String author) {
1073         this.author = author;
1074     }
1075
1076     public void setDescription(String description) {
1077         this.description = description;
1078     }
1079
1080     public void setType(String type) {
1081         this.type = type;
1082     }
1083
1084     public void setEncoding(String encoding) {
1085         this.encoding = encoding;
1086     }
1087
1088     public void setMethod(String method) {
1089         this.method = method;
1090     }
1091
1092 }
1093
1094 /**
1095  * Data Reader Implementation
1096  */
1097 class DataReaderImplementation19 extends DataReader {
1098
1099     private List<Map<String, Object>> rows;
1100
1101     private String schema;
1102
1103     private String title;
1104
1105     private String author;
1106
1107     private String description;
1108
1109     private String type;
1110
1111     private String encoding;
1112
1113     private String method;
1114
1115
1116     public void setRows(List<Map<String, Object>> rows) {
1117         this.rows = rows;
1118     }
1119
1120     public void setTitle(String title) {
1121         this.title = title;
1122     }
1123
1124     public void setAuthor(String author) {
1125         this.author = author;
1126     }
1127
1128     public void setDescription(String description) {
1129         this.description = description;
1130     }
1131
1132     public void setType(String type) {
1133         this.type = type;
1134     }
1135
1136     public void setEncoding(String encoding) {
1137         this.encoding = encoding;
1138     }
1139
1140     public void setMethod(String method) {
1141         this.method = method;
1142     }
1143
1144 }
1145
1146 /**
1147  * Data Reader Implementation
1148  */
1149 class DataReaderImplementation20 extends DataReader {
1150
1151     private List<Map<String, Object>> rows;
1152
1153     private String schema;
1154
1155     private String title;
1156
1157     private String author;
1158
1159     private String description;
1160
1161     private String type;
1162
1163     private String encoding;
1164
1165     private String method;
1166
1167
1168     public void setRows(List<Map<String, Object>> rows) {
1169         this.rows = rows;
1170     }
1171
1172     public void setTitle(String title) {
1173         this.title = title;
1174     }
1175
1176     public void setAuthor(String author) {
1177         this.author = author;
1178     }
1179
1180     public void setDescription(String description) {
1181         this.description = description;
1182     }
1183
1184     public void setType(String type) {
1185         this.type = type;
1186     }
1187
1188     public void setEncoding(String encoding) {
1189         this.encoding = encoding;
1190     }
1191
1192     public void setMethod(String method) {
1193         this.method = method;
1194     }
1195
1196 }
1197
1198 /**
1199  * Data Reader Implementation
1200  */
1201 class DataReaderImplementation21 extends DataReader {
1202
1203     private List<Map<String, Object>> rows;
1204
1205     private String schema;
1206
1207     private String title;
1208
1209     private String author;
1210
1211     private String description;
1212
1213     private String type;
1214
1215     private String encoding;
1216
1217     private String method;
1218
1219
1220     public void setRows(List<Map<String, Object>> rows) {
1221         this.rows = rows;
1222     }
1223
1224     public void setTitle(String title) {
1225         this.title = title;
1226     }
1227
1228     public void setAuthor(String author) {
1229         this.author = author;
1230     }
1231
1232     public void setDescription(String description) {
1233         this.description = description;
1234     }
1235
1236     public void setType(String type) {
1237         this.type = type;
1238     }
1239
1240     public void setEncoding(String encoding) {
1241         this.encoding = encoding;
1242     }
1243
1244     public void setMethod(String method) {
1245         this.method = method;
1246     }
1247
1248 }
1249
1250 /**
1251  * Data Reader Implementation
1252  */
1253 class DataReaderImplementation22 extends DataReader {
1254
1255     private List<Map<String, Object>> rows;
1256
1257     private String schema;
1258
1259     private String title;
1260
1261     private String author;
1262
1263     private String description;
1264
1265     private String type;
1266
1267     private String encoding;
1268
1269     private String method;
1270
1271
1272     public void setRows(List<Map<String, Object>> rows) {
1273         this.rows = rows;
1274     }
1275
1276     public void setTitle(String title) {
1277         this.title = title;
1278     }
1279
1280     public void setAuthor(String author) {
1281         this.author = author;
1282     }
1283
1284     public void setDescription(String description) {
1285         this.description = description;
1286     }
1287
1288     public void setType(String type) {
1289         this.type = type;
1290     }
1291
1292     public void setEncoding(String encoding) {
1293         this.encoding = encoding;
1294     }
1295
1296     public void setMethod(String method) {
1297         this.method = method;
1298     }
1299
1300 }
1301
1302 /**
1303  * Data Reader Implementation
1304  */
1305 class DataReaderImplementation23 extends DataReader {
1306
1307     private List<Map<String, Object>> rows;
1308
1309     private String schema;
1310
1311     private String title;
1312
1313     private String author;
1314
1315     private String description;
1316
1317     private String type;
1318
1319     private String encoding;
1320
1321     private String method;
1322
1323
1324     public void setRows(List<Map<String, Object>> rows) {
1325         this.rows = rows;
1326     }
1327
1328     public void setTitle(String title) {
1329         this.title = title;
1330     }
1331
1332     public void setAuthor(String author) {
1333         this.author = author;
1334     }
1335
1336     public void setDescription(String description) {
1337         this.description = description;
1338     }
1339
1340     public void setType(String type) {
1341         this.type = type;
1342     }
1343
1344     public void setEncoding(String encoding) {
1345         this.encoding = encoding;
1346     }
1347
1348     public void setMethod(String method) {
1349         this.method = method;
1350     }
1351
1352 }
1353
1354 /**
1355  * Data Reader Implementation
1356  */
1357 class DataReaderImplementation24 extends DataReader {
1358
1359     private List<Map<String, Object>> rows;
1360
1361     private String schema;
1362
1363     private String title;
1364
1365     private String author;
1366
1367     private String description;
1368
1369     private String type;
1370
1371     private String encoding;
1372
1373     private String method;
1374
1375
1376     public void setRows(List<Map<String, Object>> rows) {
1377         this.rows = rows;
1378     }
1379
1380     public void setTitle(String title) {
1381         this.title = title;
1382     }
1383
1384     public void setAuthor(String author) {
1385         this.author = author;
1386     }
1387
1388     public void setDescription(String description) {
1389         this.description = description;
1390     }
1391
1392     public void setType(String type) {
1393         this.type = type;
1394     }
1395
1396     public void setEncoding(String encoding) {
1397         this.encoding = encoding;
1398     }
1399
1400     public void setMethod(String method) {
1401         this.method = method;
1402     }
1403
1404 }
1405
1406 /**
1407  * Data Reader Implementation
1408  */
1409 class DataReaderImplementation25 extends DataReader {
1410
1411     private List<Map<String, Object>> rows;
1412
1413     private String schema;
1414
1415     private String title;
1416
1417     private String author;
1418
1419     private String description;
1420
1421     private String type;
1422
1423     private String encoding;
1424
1425     private String method;
1426
1427
1428     public void setRows(List<Map<String, Object>> rows) {
1429         this.rows = rows;
1430     }
1431
1432     public void setTitle(String title) {
1433         this.title = title;
1434     }
1435
1436     public void setAuthor(String author) {
1437         this.author = author;
1438     }
1439
1440     public void setDescription(String description) {
1441         this.description = description;
1442     }
1443
1444     public void setType(String type) {
1445         this.type = type;
1446     }
1447
1448     public void setEncoding(String encoding) {
1449         this.encoding = encoding;
1450     }
1451
1452     public void setMethod(String method) {
1453         this.method = method;
1454     }
1455
1456 }
1457
1458 /**
1459  * Data Reader Implementation
1460  */
1461 class DataReaderImplementation26 extends DataReader {
1462
1463     private List<Map<String, Object>> rows;
1464
1465     private String schema;
1466
1467     private String title;
1468
1469     private String author;
1470
1471     private String description;
1472
1473     private String type;
1474
1475     private String encoding;
1476
1477     private String method;
1478
1479
1480     public void setRows(List<Map<String, Object>> rows) {
1481         this.rows = rows;
1482     }
1483
1484     public void setTitle(String title) {
1485         this.title = title;
1486     }
1487
1488     public void setAuthor(String author) {
1489         this.author = author;
1490     }
1491
1492     public void setDescription(String description) {
1493         this.description = description;
1494     }
1495
1496     public void setType(String type) {
1497         this.type = type;
1498     }
1499
1500     public void setEncoding(String encoding) {
1501         this.encoding = encoding;
1502     }
1503
1504     public void setMethod(String method) {
1505         this.method = method;
1506     }
1507
1508 }
1509
1510 /**
1511  * Data Reader Implementation
1512  */
1513 class DataReaderImplementation27 extends DataReader {
1514
1515     private List<Map<String, Object>> rows;
1516
1517     private String schema;
1518
1519     private String title;
1520
1521     private String author;
1522
1523     private String description;
1524
1525     private String type;
1526
1527     private String encoding;
1528
1529     private String method;
1530
1531
1532     public void setRows(List<Map<String, Object>> rows) {
1533         this.rows = rows;
1534     }
1535
1536     public void setTitle(String title) {
1537         this.title = title;
1538     }
1539
1540     public void setAuthor(String author) {
1541         this.author = author;
1542     }
1543
1544     public void setDescription(String description) {
1545         this.description = description;
1546     }
1547
1548     public void setType(String type) {
1549         this.type = type;
1550     }
1551
1552     public void setEncoding(String encoding) {
1553         this.encoding = encoding;
1554     }
1555
1556     public void setMethod(String method) {
1557         this.method = method;
1558     }
1559
1560 }
1561
1562 /**
1563  * Data Reader Implementation
1564  */
1565 class DataReaderImplementation28 extends DataReader {
1566
1567     private List<Map<String, Object>> rows;
1568
1569     private String schema;
1570
1571     private String title;
1572
1573     private String author;
1574
1575     private String description;
1576
1577     private String type;
1578
1579     private String encoding;
1580
1581     private String method;
1582
1583
1584     public void setRows(List<Map<String, Object>> rows) {
1585         this.rows = rows;
1586     }
1587
1588     public void setTitle(String title) {
1589         this.title = title;
1590     }
1591
1592     public void setAuthor(String author) {
1593         this.author = author;
1594     }
1595
1596     public void setDescription(String description) {
1597         this.description = description;
1598     }
1599
1600     public void setType(String type) {
1601         this.type = type;
1602     }
1603
1604     public void setEncoding(String encoding) {
1605         this.encoding = encoding;
1606     }
1607
1608     public void setMethod(String method) {
1609         this.method = method;
1610     }
1611
1612 }
1613
1614 /**
1615  * Data Reader Implementation
1616  */
1617 class DataReaderImplementation29 extends DataReader {
1618
1619     private List<Map<String, Object>> rows;
1620
1621     private String schema;
1622
1623     private String title;
1624
1625     private String author;
1626
1627     private String description;
1628
1629     private String type;
1630
1631     private String encoding;
1632
1633     private String method;
1634
1635
1636     public void setRows(List<Map<String, Object>> rows) {
1637         this.rows = rows;
1638     }
1639
1640     public void setTitle(String title) {
1641         this.title = title;
1642     }
1643
1644     public void setAuthor(String author) {
1645         this.author = author;
1646     }
1647
164
```

## Output

Konfirmasi Pesanan										
	ID	username	Pemilik Rekening	Metode Pembayaran	Tanggal Pembelian	Barang	Durasi	Jumlah Unit	Total Harga	Status
66	erlindwi	Erlin Haryanti	E-Wallet	2025-06-07	Kursi Lipat	1 hari	5	Rp. 50.000	Lunas	
67	icha	Faricha	E-Wallet	2025-06-07	Panggung Portable	1 hari	1	Rp. 500.000	Lunas	
68	icha	user	E-Wallet	2025-06-07	Lampu Panggung	1 hari	2	Rp. 60.000	Lunas	
69	test	test	E-Wallet	2025-06-07	Kursi Lipat	2 hari	1	Rp. 20.000	Lunas	
70	icha	icha	E-Wallet	2025-06-08	Proyektor	1 hari	1	Rp. 25.000	Lunas	

### Penjelasan

Kode program ini merupakan untuk menampilkan data pesanan sewa barang dari customer. Data diambil dari database dan ditampilkan secara tidak bisa diedit. Saat pengguna menekan tombol "Konfirmasi Pesanan", data pesanan yang dipilih akan:

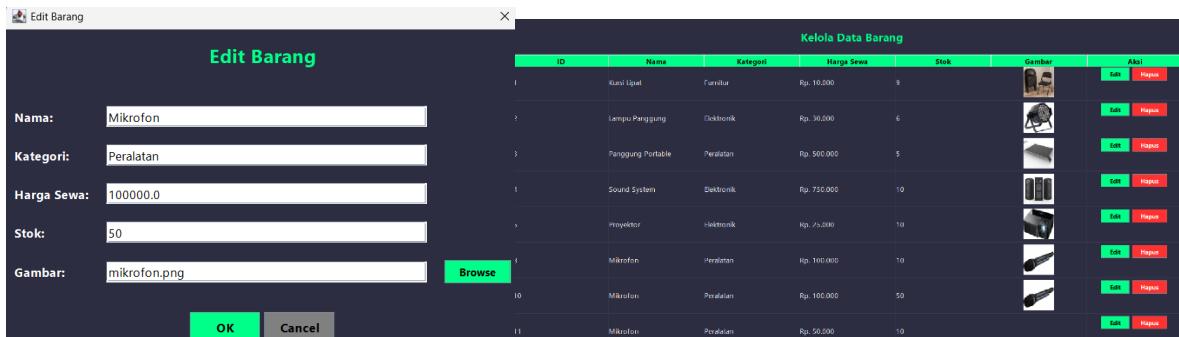
1. Dikonfirmasi pembayarannya (status diubah menjadi Lunas),
2. Mengurangi stok barang sesuai jumlah unit yang dipesan,
3. Mengupdate status pesanan dan pembayaran di tabel pesanan dan pembayaran.

### Pengujian Admin Section – Black Box Testing

No.	Fitur	Test Case ID	Input Data	Hasil yang Diharapkan
1	Dashboard Admin	DA-001	Login sebagai role admin	Dashboard admin menampilkan profil perusahaan VenueVibe dan menyediakan menu untuk mengelola data barang serta konfirmasi pesanan.

				
2		TB-001	Nama: Mikrofon Kategori: Peralatan Harga Sewa: 100.00 Stok: 10 Gambar: mikrofon.png	Item mikrofon berhasil ditambahkan, dan muncul dalam daftar barang pada dashboard admin.
3	Tambah Barang Baru	TB-002	Nama: Mikrofon Kategori: Peralatan Harga Sewa: 50.000 Stok: 10 Gambar: -	Item mikrofon berhasil ditambahkan, dan muncul dalam daftar barang pada dashboard admin, namun gambar nya tidak tampil.
4		TB-002	Nama: - Kategori: - Harga Sewa: -	Menampilkan pesan error: Gagal menyimpan data karena data kosong.

			Stok: - Gambar: -	
5	Edit Barang	EB-001	<p>Nama: Mikrofon Kategori: Peralatan Harga Sewa: 100.000 Stok: 50 Gambar: mikrofon.png</p> <p>Item mikrofon berhasil diubah, stok berubah menjadi 50.</p>	

6		EB-002	Nama: Mikrofon Kategori: Peralatan Harga Sewa: 100.000 Stok: - Gambar: mikrorfong.png	Menampilkan pesan error: Gagal menyimpan data karena data tidak boleh kosong.																																																															
			 <p><b>Edit Barang</b></p> <p>Name: Mikrofon Category: Peralatan Rent Price: 100000.0 Stock: 50 Image: mikrofon.png</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Category</th> <th>Rent Price</th> <th>Stock</th> <th>Image</th> <th>Action</th> </tr> </thead> <tbody> <tr><td>1</td><td>Kursi Lipat</td><td>Furniture</td><td>Rp. 10.000</td><td>9</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>2</td><td>Lampa Penggantung</td><td>Elektronik</td><td>Rp. 30.000</td><td>6</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>3</td><td>Penggantung Portable</td><td>Peralatan</td><td>Rp. 500.000</td><td>5</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>4</td><td>Sound System</td><td>Elektronik</td><td>Rp. 750.000</td><td>10</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>5</td><td>Projector</td><td>Elektronik</td><td>Rp. 250.000</td><td>10</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>6</td><td>Microphone</td><td>Peralatan</td><td>Rp. 100.000</td><td>10</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>7</td><td>Mikrofon</td><td>Peralatan</td><td>Rp. 100.000</td><td>50</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> <tr><td>8</td><td>Mikrofon</td><td>Peralatan</td><td>Rp. 50.000</td><td>10</td><td></td><td><input type="button" value="Edit"/> <input type="button" value="Delete"/></td></tr> </tbody> </table> <p><input type="button" value="OK"/> <input type="button" value="Cancel"/></p>	ID	Name	Category	Rent Price	Stock	Image	Action	1	Kursi Lipat	Furniture	Rp. 10.000	9		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	2	Lampa Penggantung	Elektronik	Rp. 30.000	6		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	3	Penggantung Portable	Peralatan	Rp. 500.000	5		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	4	Sound System	Elektronik	Rp. 750.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	5	Projector	Elektronik	Rp. 250.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	6	Microphone	Peralatan	Rp. 100.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	7	Mikrofon	Peralatan	Rp. 100.000	50		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	8	Mikrofon	Peralatan	Rp. 50.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>	
ID	Name	Category	Rent Price	Stock	Image	Action																																																													
1	Kursi Lipat	Furniture	Rp. 10.000	9		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
2	Lampa Penggantung	Elektronik	Rp. 30.000	6		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
3	Penggantung Portable	Peralatan	Rp. 500.000	5		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
4	Sound System	Elektronik	Rp. 750.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
5	Projector	Elektronik	Rp. 250.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
6	Microphone	Peralatan	Rp. 100.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
7	Mikrofon	Peralatan	Rp. 100.000	50		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
8	Mikrofon	Peralatan	Rp. 50.000	10		<input type="button" value="Edit"/> <input type="button" value="Delete"/>																																																													
7	Hapus Barang	HB-001	Klik button konfirmasi, dan jika pilihhapus maka data barang akan terhapus.	Menampilkan pesan konfirmasi, dan jika pilihhapus maka data barang akan terhapus.																																																															

Kelola Data Barang							
ID	Name	Kategori	Harga Satuan	Stock	Gambar	Aksi	
1	Kursi Lipat	Kantor	Rp. 10.000	5		<a href="#">Edit</a> <a href="#">Hapus</a>	
2	Lampu Penyekat	Elektronik	Rp. 20.000	6		<a href="#">Edit</a> <a href="#">Hapus</a>	
3	Penggong Portable	Perabotan	Rp. 500.000	5		<a href="#">Edit</a> <a href="#">Hapus</a>	
4	Sound System	Elektronik	Rp. 700.000	10		<a href="#">Edit</a> <a href="#">Hapus</a>	
5	Proyektor	Elektronik	Rp. 100.000	10		<a href="#">Edit</a> <a href="#">Hapus</a>	
6	Mesin Cuci	Perabotan	Rp. 100.000	10		<a href="#">Edit</a> <a href="#">Hapus</a>	
7	Mesin Cuci	Perabotan	Rp. 100.000	10		<a href="#">Edit</a> <a href="#">Hapus</a>	
8	Mesin Cuci	Perabotan	Rp. 100.000	10		<a href="#">Edit</a> <a href="#">Hapus</a>	

Konfirmasi Pesanan									
ID	username	Penulis Rekening	Metode Pembayaran	Tanggal Pembelian	Berang	Durasi	Jumlah Unit	Total Harga	Status
66	efendi	E-Wallet	E-Wallet	2025-06-07	Kursi Lipat	1 hari	5	Rp. 50.000	Lunas
67	ida	Felix	E-Wallet	2025-06-07	Penggong Portable	1 hari	1	Rp. 500.000	Lunas
68	ida	user	E-Wallet	2025-06-07	Lampu Penyekat	1 hari	2	Rp. 40.000	Lunas
69	test	test	E-Wallet	2025-06-07	Kursi Lipat	2 hari	1	Rp. 20.000	Lunas
70	ida	ida	E-Wallet	2025-06-08	Proyektor	1 hari	1	Rp. 10.000	Menunggu

Konfirmasi Pesanan									
ID	username	Penulis Rekening	Metode Pembayaran	Tanggal Pembelian	Berang	Durasi	Jumlah Unit	Total Harga	Status
66	efendi	E-Wallet	E-Wallet	2025-06-07	Kursi Lipat	1 hari	5	Rp. 50.000	Lunas
67	ida	Felix	E-Wallet	2025-06-07	Penggong Portable	1 hari	1	Rp. 500.000	Lunas
68	test	user	E-Wallet	2025-06-07	Lampu Penyekat	1 hari	2	Rp. 40.000	Lunas
69	test	test	E-Wallet	2025-06-07	Kursi Lipat	2 hari	1	Rp. 20.000	Lunas
70	ida	ida	E-Wallet	2025-06-08	Proyektor	1 hari	1	Rp. 10.000	Menunggu

#### 2.4.3 Customer Section

##### A. Dashboard Customer

Kode Program :

```

src > rental > J DashboardFrame.java > DashboardFrame > showRiwayatPesananPanel()
1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import javax.swing.text.*;
7
8 public class DashboardFrame extends JFrame {
9     private JPanel mainPanel;
10    private int currentUserid;
11
12    public DashboardFrame(int idUser) {
13        this.currentUserid = idUser;
14
15        setTitle(title="Dashboard - VenueVibe");
16        setSize(width:1000, height:600);
17        setLocationRelativeTo(null);
18        setDefaultCloseOperation(EXIT_ON_CLOSE);
19        setLayout(new BorderLayout());
20
21        // *** HEADER ***
22        JPanel header = new JPanel(new BorderLayout());
23        header.setBackground(new Color(r:20, g:20, b:30));
24        header.setPreferredSize(new Dimension(getWidth(), height:60));
25
26        JLabel title = new JLabel(text:"VenueVibe Dashboard", SwingConstants.LEFT);
27        title.setFont(new Font(name:"Segoe UI", Font.BOLD, size:22));
28        title.setForeground(new Color(r:0, g:255, b:136));
29        title.setBorder(BorderFactory.createEmptyBorder(top:10, left:10, bottom:10, right:10));
30
31        JButton logoutBtn = new JButton(text:"Logout");
32        logoutBtn.setBackground(new Color(r:20, g:20, b:30));
33        logoutBtn.setForeground(Color.WHITE);
34        logoutBtn.setFocusPainted(false);
35        logoutBtn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
36        logoutBtn.setBorder(BorderFactory.createEmptyBorder(top:8, left:8, bottom:8, right:20));
37        logoutBtn.addActionListener(e -> {
38            // Misal frame login belum dibuat, cukup keluar program
39            System.exit(status:0);
40        });
41
42        header.add(title, BorderLayout.WEST);
43        header.add(logoutBtn, BorderLayout.EAST);
44        add(header, BorderLayout.NORTH);
45
46        // *** SIDEBAR ***
47        JPanel sidebar = new JPanel();
48        sidebar.setBackground(new Color(r:30, g:30, b:30));
49        sidebar.setLayout(new BoxLayout(sidebar, BoxLayout.Y_AXIS));
50        sidebar.setPreferredSize(new Dimension(width:200, getHeight()));
51        sidebar.setBorder(BorderFactory.createEmptyBorder(top:20, left:10, bottom:20, right:10)); // Padding dalam
52
53        String[] menuItems = {
54            "Dashboard", "Lihat Daftar Barang", "Riwayat Pesanan"
55        };
56        for (String item : menuItems) {
57            JButton btn = createSidebarButton(item);
58            switch (item) {
59                case "Dashboard" -> btn.addActionListener(e -> showCompanyProfile());
60                case "Lihat Daftar Barang" -> btn.addActionListener(e -> showBarangPanel());
61                case "Riwayat Pesanan" -> btn.addActionListener(e -> showRiwayatPesananPanel());
62            }
63            sidebar.add(Box.createVerticalStrut(height:15));
64            sidebar.add(btn);
65        }
66
67        sidebar.add(Box.createVerticalGlue());
68        add(sidebar, BorderLayout.WEST);
69
70        // *** MAIN PANEL ***
71        mainPanel = new JPanel(new BorderLayout());
72        mainPanel.setBackground(new Color(r:30, g:30, b:47));
73        add(mainPanel, BorderLayout.CENTER);
74
75        showCompanyProfile(); // Load default view
76        setVisible(b:true);
77    }
78
79    private JButton createSidebarButton(String text) {
80        JButton btn = new JButton(text);
81        btn.setAlignment(Component.CENTER_ALIGNMENT);
82        btn.setMaximumSize(new Dimension(width:100, height:45));
83        btn.setFont(new Font(name:"Segoe UI", Font.BOLD, size:14));
84        btn.setBackground(Color.BLACK);
85        btn.setForeground(new Color(r:0, g:255, b:136));
86        btn.setFocusPainted(false);
87        btn.setCursor(Cursor.getPredefinedCursor(Cursor.HAND_CURSOR));
88        btn.setBorder(BorderFactory.createEmptyBorder(top:10, left:20, bottom:10, right:20));
89    }

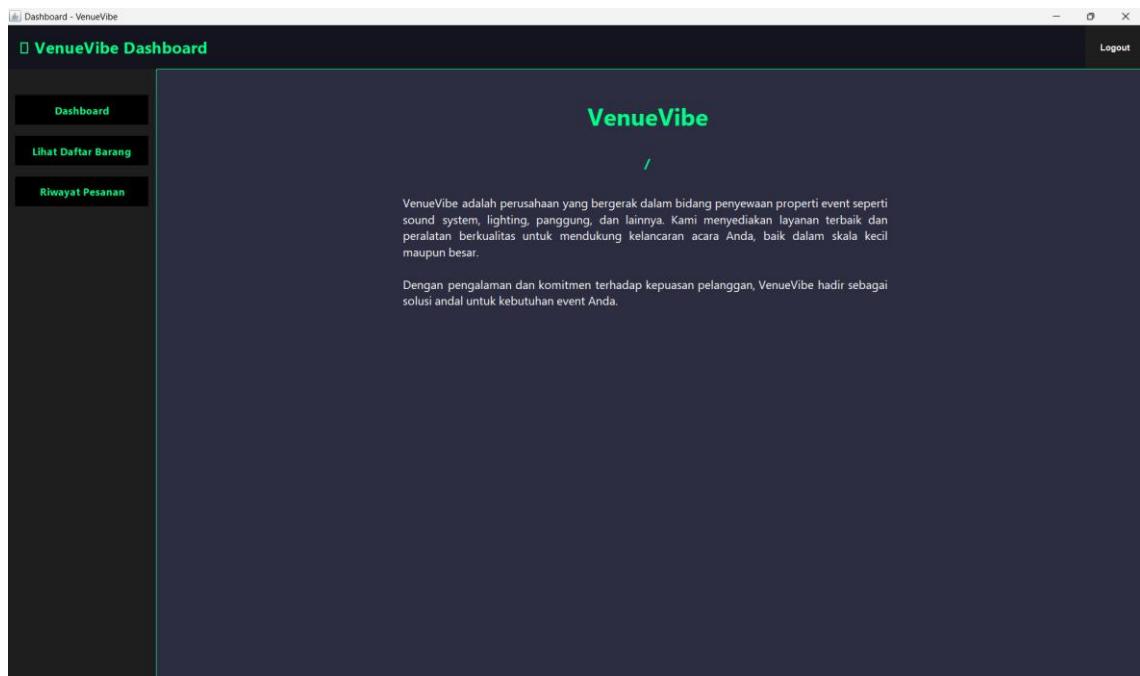
```

```

90     btn.addMouseListener(new MouseAdapter() {
91         public void mouseEntered(MouseEvent evt) {
92             btn.setBackground(new Color(r10, g1255, b136));
93             btn.setForeground(Color.BLACK);
94         }
95
96         public void mouseExited(MouseEvent evt) {
97             btn.setBackground(Color.BLACK);
98             btn.setForeground(new Color(r10, g1255, b136));
99         }
100    });
101
102    return btn;
103 }
104
105 private void showCompanyProfile() {
106     mainPanel.removeAll();
107
108     JPanel centerPanel = new JPanel();
109     centerPanel.setBackground(new Color(r145, g145, b165));
110     centerPanel.setLayout(new BoxLayout(centerPanel, BoxLayout.Y_AXIS));
111     centerPanel.setBorder(BorderFactory.createCompoundBorder(
112         BorderFactory.createLineBorder(new Color(r10, g1255, b136), thickness1, rounded:true),
113         BorderFactory.createEmptyBorder(top10, left10, bottom10, right10)));
114
115     JLabel titleLabel = new JLabel(text:"Venuvibe");
116     titleLabel.setFont(new Font(name:"Segoe UI", Font.BOLD, size12));
117     titleLabel.setForeground(new Color(r10, g1255, b136));
118     titleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
119     titleLabel.setBorder(BorderFactory.createEmptyBorder(top10, left10, bottom10, right10));
120
121     // Ganti logo dengan label toks sedianya (hindari error resource)
122     JLabel logoLabel = new JLabel(text:"");
123     logoLabel.setFont(new Font(name:"Segoe UI", Font.BOLD, size18));
124     logoLabel.setForeground(new Color(r10, g1255, b136));
125     logoLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
126     logoLabel.setBorder(BorderFactory.createEmptyBorder(top10, left10, bottom10, right10));
127
128     JTextPane descPane = new JTextPane();
129     descPane.setText(
130         "Venuvibe adalah perusahaan yang bergerak dalam bidang penyewaan properti event seperti sound system, lighting, panggung, dan lainnya. "
131         + "Kami menyediakan layanan terbaik dan peralatan berkualitas untuk mendukung kelancaran acara Anda, baik dalam skala kecil maupun besar.\n\n"
132         + "Dengan pengalaman dan komitmen terhadap kepuasan pelanggan, Venuvibe hadir sebagai solusi andal untuk kebutuhan event Anda.");
133     descPane.setFont(new Font(name:"Segoe UI", Font.PLAIN, size16));
134     descPane.setForeground(Color.WHITE);
135     descPane.setBackground(new Color(r145, g145, b165));
136     descPane.setEditable(false);
137     descPane.setMaximumSize(new Dimension(width700, height100));
138     descPane.setOpaque(true);
139     descPane.setBorder(BorderFactory.createEmptyBorder(top10, left10, bottom10, right10));
140
141     StyledDocument doc = descPane.getStyledDocument();
142     SimpleAttributeSet justify = new SimpleAttributeSet();
143     StyleConstants.setAlignment(justify, StyleConstants.ALIGN_JUSTIFIED);
144     doc.setParagraphAttributes(offset10, doc.getLength(), justify, replace:false);
145
146     centerPanel.add(titleLabel);
147     centerPanel.add(logoLabel);
148     centerPanel.add(descPane);
149
150     mainPanel.add(centerPanel, BorderLayout.CENTER);
151
152     mainPanel.revalidate();
153     mainPanel.repaint();
154 }
155
156
157 private void showMessage(String message) {
158     mainPanel.removeAll();
159     JPanel panel = new JPanel();
160     JLabel label = new JLabel(message);
161     label.setFont(new Font(name:"Segoe UI", Font.PLAIN, size18));
162     label.setForeground(Color.WHITE);
163     label.setAlignmentX(Component.CENTER_ALIGNMENT);
164
165     JPanel panel1 = new JPanel();
166     panel1.setBackground(new Color(r145, g145, b165));
167     panel1.setLayout(new BoxLayout(panel1, BoxLayout.Y_AXIS));
168     panel1.setBorder(BorderFactory.createEmptyBorder(top50, left50, bottom50, right50));
169
170     mainPanel.add(panel1, BorderLayout.CENTER);
171
172     mainPanel.revalidate();
173     mainPanel.repaint();
174 }
175
176 private void showBarangPanel() {
177     mainPanel.removeAll();
178     mainPanel.setLayout(new BorderLayout());
179     mainPanel.add(new BarangPanel((namaBarang, idBarang, hargaSewa, stok) -> {
180         | showPesanBarangPanel(namaBarang, hargaSewa, idBarang, currentUserId, stok);
181     }), BorderLayout.CENTER);
182     mainPanel.revalidate();
183     mainPanel.repaint();
184 }
185
186 private void showPesanBarangPanel(String namaBarang, double hargaSewa, int idBarang, int idUser, int stok) {
187     mainPanel.removeAll();
188     mainPanel.setLayout(new BorderLayout());
189     mainPanel.add(new PesanBarangPanel(namaBarang, hargaSewa, idBarang, idUser, stok), BorderLayout.CENTER);
190     mainPanel.revalidate();
191     mainPanel.repaint();
192 }
193
194 public void showPembayaranPanel(String namaBarang, double hargaSewa, int idBarang, int idUser, int stok, int durasi,
195     | int jumlah) {
196     mainPanel.removeAll();
197     mainPanel.setLayout(new BorderLayout());
198     mainPanel.add(new PembayaranPanel(namaBarang, hargaSewa, idBarang, idUser, stok, durasi, jumlah),
199     | | BorderLayout.CENTER);
200     mainPanel.revalidate();
201     mainPanel.repaint();
202 }
203
204 public void showRiwayatPesananPanel() {
205     mainPanel.removeAll();
206     mainPanel.setLayout(new BorderLayout());
207     mainPanel.add(new RiwayatPesananPanel(this.currentUserId), BorderLayout.CENTER);
208     mainPanel.revalidate();
209     mainPanel.repaint();
210 }
211
212 Run [Debug]
213 public static void main(String[] args) {
214     | SwingUtilities.invokeLater(() -> new DashboardFrame(idUser));
215 }
216

```

Output :



Penjelasan :

Kode program ini merupakan halaman dashboard admin. Saat program dijalankan, akan tampil jendela berjudul "*VenueVibe Dashboard*" dengan tampilan yang terdiri dari:

1. Header di atas berisi judul dan tombol "Logout".
2. Sidebar di sisi kiri dengan tiga menu:
  - a. Dashboard
  - b. Kelola Data Barang
  - c. Riwayat Pesanan
3. Main Panel di bagian tengah untuk menampilkan konten berdasarkan menu yang dipilih.

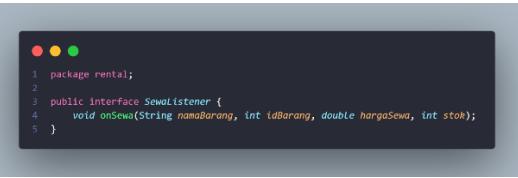
### Breakdown & Fungsionalitas

1. Konstruktor (DashboardFrame(int idUser)):
  - o Inisialisasi Jendela: Mengatur judul, ukuran, dan posisi jendela utama aplikasi.
  - o Manajemen User: Menerima dan menyimpan idUser untuk keperluan personalisasi data di panel-panel fitur.
  - o Layout Utama: Mengatur tata letak dasar jendela menjadi tiga area fungsional: header, sidebar, dan panel utama.

2. Header (Atas):
  - Identitas Aplikasi: Menampilkan judul "VenueVibe Dashboard".
  - Logout: Menyediakan tombol "Logout" untuk keluar dari aplikasi.
3. Sidebar (Kiri):
  - Navigasi Menu: Berisi tombol-tombol menu (Dashboard, Lihat Daftar Barang, Riwayat Pesanan) yang berfungsi sebagai navigasi utama.
  - Gaya Interaktif: Tombol-tombol memiliki efek *hover* yang menarik saat kursor diarahkan padanya.
4. Main Panel (Tengah):
  - Area Konten Dinamis: Bertindak sebagai wadah untuk menampilkan konten dari menu yang dipilih.
  - Tampilan Default: Saat aplikasi pertama kali dibuka, panel ini akan menampilkan profil atau deskripsi singkat tentang perusahaan VenueVibe.
5. Metode Navigasi Panel (showCompanyProfile(), showBarangPanel(), showRiwayatPesananPanel(), dll.):
  - Pembersihan & Penggantian Konten: Setiap metode ini bertanggung jawab untuk membersihkan mainPanel dari konten sebelumnya dan memuat panel baru (misalnya, BarangPanel atau RiwayatPesananPanel) sesuai dengan menu yang dipilih.
  - Penerusan Data User: Khusus untuk showRiwayatPesananPanel(), idUser yang tersimpan akan diteruskan untuk memuat riwayat pesanan yang spesifik.
  - Alur Pemesanan: Metode seperti showPesanBarangPanel() dan showPembayaranPanel() mengelola transisi ke layar pemesanan dan pembayaran, meneruskan detail barang dan user yang relevan.
6. main Method:
  - Titik Mulai Aplikasi: Sebagai metode utama, ia memulai aplikasi dengan membuat instance DashboardFrame di *Event Dispatch Thread (EDT)* untuk memastikan responsivitas UI.

## B. Lihat Daftar Barang

Kode Program:



```
1 package rental;
2
3 public interface SewaListener {
4     void onSewa(String namaBarang, int idBarang, double hargaSewa, int stok);
5 }
```

Penjelasan:

Kode program ini mendefinisikan sebuah *interface* bernama SewaListener yang berada dalam package rental. SewaListener berperan sebagai **kontrak** atau **blueprint** komunikasi antara komponen antarmuka pengguna yang menampilkan daftar barang dan komponen lain yang bertanggung jawab untuk merespons atau memproses aksi penyewaan. Ini adalah mekanisme *callback* yang memungkinkan *decoupling* (pemisahan ketergantungan) antara komponen-komponen tersebut.

### Breakdown Fungsionalitas:

- Definisi Kontrak (public interface SewaListener):

Interface ini mendefinisikan aturan bahwa setiap kelas yang mengimplementasikannya harus menyediakan implementasi metode yang dideklarasikan.

- Metode Abstrak (void onSewa(...)):

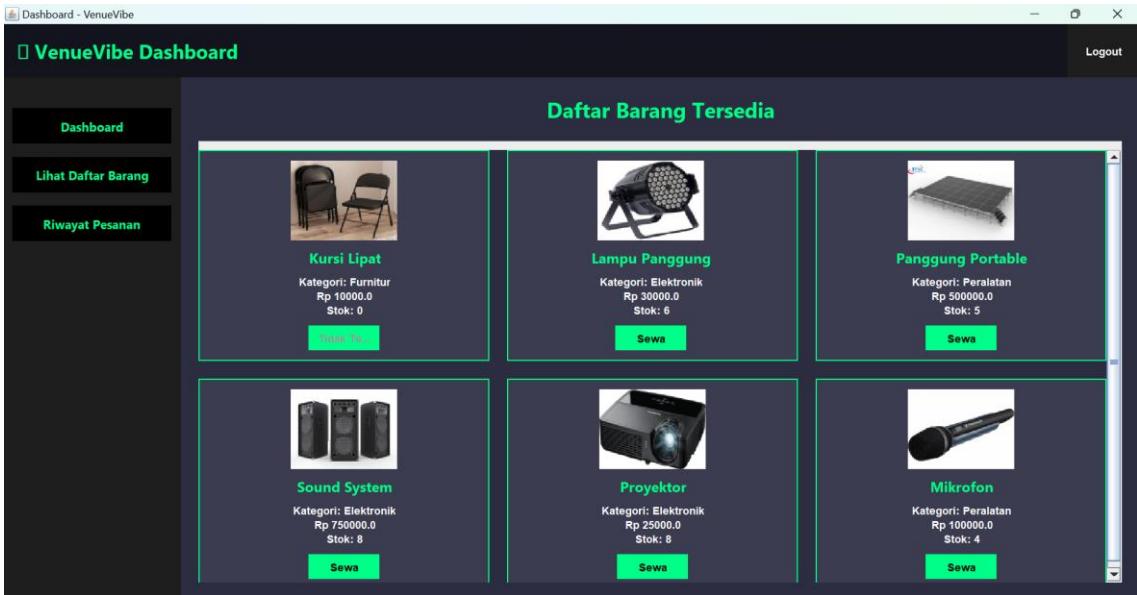
- SewaListener mendeklarasikan satu metode abstrak: onSewa. Metode ini dirancang untuk dipanggil ketika sebuah peristiwa "Sewa" terjadi (misalnya, saat tombol "Sewa" diklik).
- Parameter yang disertakan (namaBarang, idBarang, hargaSewa, stok) adalah informasi kunci mengenai barang yang akan disewa, yang diperlukan oleh "pendengar" (listener) untuk memproses permintaan sewa secara lengkap.

```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class BarangPanel extends JPanel {
8     private JPanel cardPanel;
9     private SewaListener sewaListener;
10
11     // konstruktor barangpanel, memerlukan listener untuk akksi sewa
12     public BarangPanel(SewaListener sewaListener) {
13         this.sewaListener = sewaListener;
14         setLayout(new BorderLayout());
15         setBackground(new Color(45, 45, 65));
16         setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));
17     }
18
19     // --- Judul Panel ---
20     JLabel title = new JLabel("Daftar Barang Tersedia");
21     title.setFont(new Font("Segoe UI", Font.BOLD, 24));
22     title.setForeground(Color.GRAY);
23     title.setHorizontalAlignment(Component.CENTER);
24     title.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
25     add(title, BorderLayout.NORTH);
26
27     // --- Panel Utama ---
28     JPanel cardPanel = new JPanel(new GridLayout(0, 3, 20, 20)); // 3 kolom
29     cardPanel.setBackground(new Color(45, 45, 65));
30     JScrollPane scrollPane = new JScrollPane(cardPanel);
31     scrollPane.setBorder(BorderFactory.createEmptyBorder(10, 0, 0, 0));
32     add(scrollPane, BorderLayout.CENTER);
33
34     // Must load barang dari database
35     loadData();
36
37     // Method untuk mengambil dan menampilkan data barang dari database
38     private void loadData() {
39         cardPanel.removeAll();
40
41         try (Connection conn = KoneksiDatabase.getConnection()) {
42             Statement stmt = conn.createStatement();
43             ResultSet rs = stmt.executeQuery("SELECT * FROM barang");
44
45             while (rs.next()) {
46                 // Buat panel baru untuk setiap barang
47                 JPanel card = new JPanel();
48                 card.setLayout(new BoxLayout(card, BoxLayout.Y_AXIS));
49                 card.setBackground(new Color(60, 60, 80));
50                 card.setBorder(BorderFactory.createCompoundBorder(
51                     BorderFactory.createInsetBorder(new Color(255, 136), 2),
52                     BorderFactory.createEmptyBorder(0, 10, 10, 10)));
53
54                 // Ambil data barang dari database
55                 int stok = rs.getInt("stok");
56                 String namabarang = rs.getString("namabarang");
57                 double hargaSewa = rs.getDouble("hargaSewa");
58
59                 // Buat label gambar barang
60                 String gambarPath = rs.getString("gambarBarang");
61                 if (gambarPath != null & !gambarPath.isEmpty()) {
62                     // Cari gambar di folder rental
63                     ImageIcon icon = null;
64                     java.net.URL imgURL = getClass().getResource("/rental/" + gambarPath);
65                     if (imgURL != null) {
66                         icon = new ImageIcon(imgURL);
67                         Image img = icon.getImage().getScaledInstance(120, 90, Image.SCALE_SMOOTH);
68                         JLabel lbGambar = new JLabel(new ImageIcon(img));
69                         lbGambar.setAlignmentX(Component.CENTER_ALIGNMENT);
70                         card.add(lbGambar);
71                     } else {
72                         // Jika gambar tidak ditemukan
73                         JLabel lbGambar = new JLabel("Tidak ada gambar");
74                         lbGambar.setForeground(Color.RED);
75                         lbGambar.setAlignmentX(Component.CENTER_ALIGNMENT);
76                         card.add(lbGambar);
77                     }
78                 }
79                 card.add(Box.createVerticalStrut(8));
80             }
81
82             // Tampilkan label barang
83             JLabel lbNama = new JLabel(rs.getString("namabarang"));
84             lbNama.setFont(new Font("Segoe UI", Font.BOLD, 16));
85             lbNama.setForeground(new Color(0, 255, 255));
86             lbNama.setAlignmentX(Component.CENTER_ALIGNMENT);
87             card.add(lbNama);
88
89             card.add(Box.createVerticalStrut(8));
90
91             // Tampilkan kategori barang
92             JLabel lbKategori = new JLabel("Kategori: " + rs.getString("kategori"));
93             lbKategori.setForeground(Color.WHITE);
94             lbKategori.setAlignmentX(Component.CENTER_ALIGNMENT);
95             card.add(lbKategori);
96
97             card.add(Box.createVerticalStrut(8));
98
99             // Tampilkan harga sewa
100            JLabel lbHarga = new JLabel("Rp " + rs.getDouble("hargaSewa"));
101            lbHarga.setForeground(Color.WHITE);
102            lbHarga.setAlignmentX(Component.CENTER_ALIGNMENT);
103            card.add(lbHarga);
104
105            // Tampilkan stok barang
106            JLabel lbStok = new JLabel("Stok: " + stok);
107            lbStok.setForeground(Color.WHITE);
108            lbStok.setAlignmentX(Component.CENTER_ALIGNMENT);
109            card.add(lbStok);
110
111            // Tambahkan tombol Sewa
112            JButton btnSewa = new JButton("Sewa");
113            btnSewa.setAlignmentX(Component.CENTER_ALIGNMENT);
114            btnSewa.setMargin(new Dimension(0, 20));
115            btnSewa.setRolloverEnabled(true);
116            btnSewa.setCursor(Cursor.getSystemCursor());
117            btnSewa.addActionListener(e -> {
118                if (stok > 0) {
119                    btnSewa.setEnabled(false);
120                    btnSewa.setText("Tidak Tersedia");
121
122                    // Set tombol sewa tidak aktif
123                    btnSewa.addActionListener(e2 -> {
124                        if (sewaListener != null) {
125                            sewaListener.onSewa(namabarang, idBarang, hargaSewa, stok);
126                        }
127                    });
128
129                    card.add(Box.createVerticalStrut(10));
130                    card.add(btnSewa);
131
132                    // Set tombol sewa aktif
133                    card.add(Box.createVerticalStrut(10));
134                    card.add(btnSewa);
135
136                    // Pindah ke panel tambahan
137                    cardPanel.add(card);
138
139                }
140            });
141            card.add(btnSewa);
142
143            // Pindah ke panel tambahan
144            cardPanel.add(card);
145
146        }
147
148        // Setelah semua kartu barang ditambahkan, update tampilan panel:
149        cardPanel.revalidate(); // Memerlukan layout panel dipersaral setelah perubahan isi
150        cardPanel.repaint(); // Memerlukan panel untuk menggaris bawahi agar perubahan terlihat
151
152    } catch (SQLException ex) {
153        // Jika terjadi error saat mengambil data dari database, tampilkan pesan error
154        JOptionPane.showMessageDialog(this, "Gagal memuat data: " + ex.getMessage(), "Error",
155            JOptionPane.ERROR_MESSAGE);
156    }
157}
158}

```

Output:



Penjelasan:

Kode program ini mendefinisikan kelas `BarangPanel`, yang merupakan salah satu komponen antarmuka pengguna (UI) dalam aplikasi "*VenueVibe*". Panel ini memiliki tanggung jawab utama untuk menampilkan daftar barang-barang yang tersedia untuk disewa kepada pengguna. Tampilan barang disajikan dalam format *cards* yang terstruktur, di mana setiap kartu menampilkan detail seperti gambar barang, nama, kategori, harga sewa, dan jumlah stok yang tersedia.

`BarangPanel` dirancang untuk mengambil data barang dari database. Saat panel ini diinisialisasi, ia akan secara otomatis memuat informasi barang dan menampilkannya dalam tata letak grid yang dapat digulir jika jumlah barang terlalu banyak. Pengguna dapat melihat detail setiap barang, dan jika stok barang tersedia, mereka dapat mengklik tombol "Sewa" pada kartu barang untuk melanjutkan proses pemesanan. Interaksi ini kemudian didelegasikan ke objek `SewaListener` untuk penanganan lebih lanjut.

#### **Breakdown Fungsionalitas:**

- Struktur Panel Utama:
  - `BarangPanel` adalah sebuah `JPanel` yang menggunakan `BorderLayout` sebagai manajer tata letak utama.
  - Header (`JLabel` "Daftar Barang Tersedia") ditempatkan di bagian atas (`BorderLayout.NORTH`).

- Area utama untuk kartu-kartu barang (cardsPanel) ditempatkan di bagian tengah (BorderLayout.CENTER) dan diatur dalam JScrollPane untuk memungkinkan pengguliran.
- Panel Kartu Barang (cardsPanel):
  - cardsPanel adalah JPanel yang menggunakan GridLayout(0, 3, 20, 20) untuk menampilkan barang-barang dalam 3 kolom dengan spasi antar kartu.
  - Warna latar belakang dan *padding* panel diatur untuk konsistensi visual.
- Pengambilan dan Tampilan Data (loadData() method):
  - Metode loadData() adalah inti fungsionalitas ini. Ia membersihkan cardsPanel terlebih dahulu, lalu mengambil data dari tabel barang di database melalui KoneksiDatabase.getConnection().
  - Untuk setiap baris data yang diambil, sebuah JPanel baru (card) dibuat. card ini berisi JLabel untuk gambar (memuat gambar dari *classpath rental/*), nama, kategori, harga sewa, dan stok.
  - Tombol "Sewa" ( JButton ) ditambahkan ke setiap kartu. Tombol ini dinonaktifkan dan teksnya diubah menjadi "Tidak Tersedia" jika stok barang adalah 0 atau kurang.
- Interaksi Tombol "Sewa":
 

Sebuah ActionListener ditambahkan ke btnSewa. Ketika tombol diklik, listener ini memanggil metode onSewa() pada objek sewaListener yang telah diberikan kepada BarangPanel. Detail barang yang relevan diteruskan sebagai parameter. Ini adalah mekanisme *callback* untuk memberitahu komponen lain tentang aksi sewa.
- Penanganan Kesalahan Database:
 

Blok try-catch digunakan untuk menangani SQLException yang mungkin terjadi selama pengambilan data dari database, menampilkan pesan JOptionPane kepada pengguna jika terjadi kesalahan.

## C. Pesanan

Kode Program:



```
1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5
6 public abstract class BaseFormPanel extends JPanel {
7     public BaseFormPanel(String title) {
8         setLayout(new BorderLayout());
9         setBackground(new Color(45, 45, 65));
10        setBorder(BorderFactory.createEmptyBorder(30, 120, 30, 120));
11        JLabel lblTitle = new JLabel(title);
12        lblTitle.setFont(new Font("Segoe UI", Font.BOLD, 22));
13        lblTitle.setForeground(new Color(8, 255, 136));
14        lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
15        add(lblTitle, BorderLayout.NORTH);
16    }
17 }
```

Penjelasan:

Kode program ini mendefinisikan `BaseFormPanel`, yaitu kelas abstrak yang berfungsi sebagai dasar untuk panel antarmuka pengguna dengan struktur formulir yang seragam. Kelas ini dirancang untuk menyediakan tata letak dan gaya dasar, seperti warna latar belakang, padding, dan label judul yang terpusat, guna menciptakan konsistensi visual di seluruh aplikasi. Sebagai kelas abstrak, `BaseFormPanel` tidak dapat diinisialisasi langsung menjadi objek, melainkan harus diwarisi oleh kelas turunan, seperti `PesanBarangPanel` dan `PembayaranPanel`. Kelas turunan ini akan melengkapi detail formulir atau konten spesifik, sambil tetap mewarisi semua properti dan perilaku dasar yang telah ditetapkan oleh `BaseFormPanel`. Pendekatan ini mendorong konsistensi visual di seluruh aplikasi dan mengurangi duplikasi kode.

#### Breakdown Fungsionalitas:

- Konstruktor `public BaseFormPanel(String title):`
  - Konstruktor ini menerima sebuah parameter `String title` yang akan digunakan sebagai teks untuk label judul panel.
  - Ketika kelas turunan memanggil konstruktor ini (menggunakan `super(title);`), ia akan menginisialisasi properti dasar panel yang sama untuk semua formulir.
- Pengaturan Tata Letak dan Gaya Umum:
  - `setLayout(new BorderLayout());`: Mengatur `BorderLayout` sebagai manajer tata letak utama panel, memungkinkan komponen ditempatkan di area NORTH, SOUTH, EAST, WEST, atau CENTER.
  - `setBackground(new Color(45, 45, 65));`: Menentukan warna latar belakang panel, menciptakan skema warna yang konsisten dan gelap.

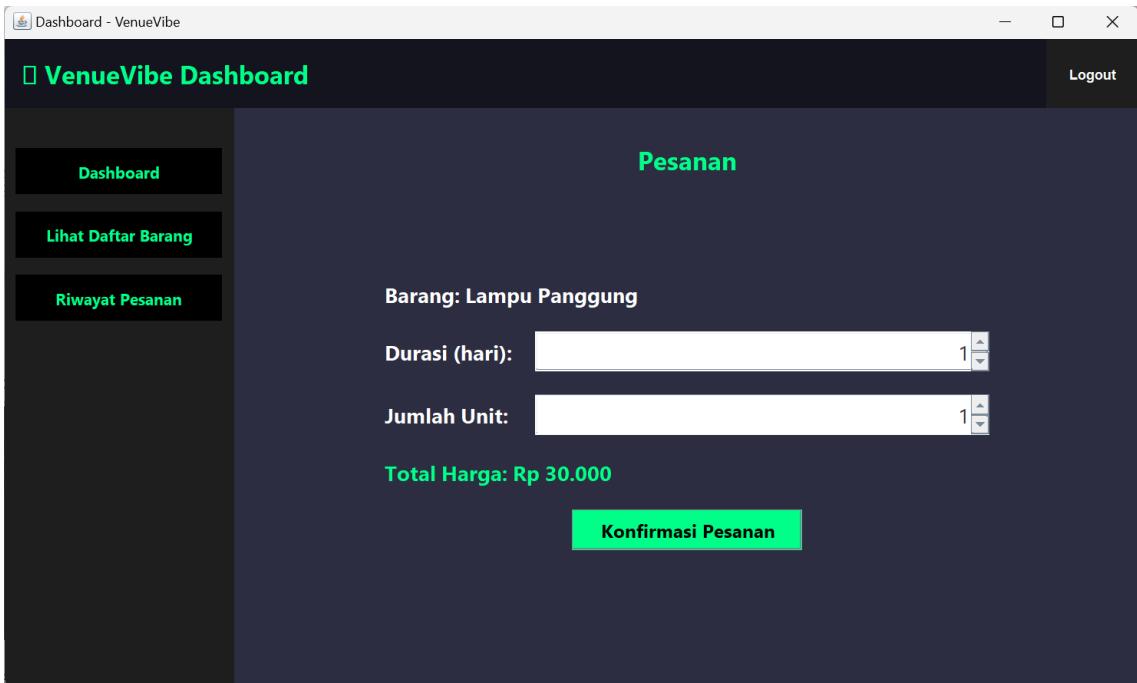
- setBorder(BorderFactory.createEmptyBorder(30, 120, 30, 120));:  
Menambahkan *padding* di sekitar konten panel, memberikan ruang visual yang nyaman dan terstruktur.
- Penambahan Label Judul Otomatis:
  - JLabel lblTitle = new JLabel(title);: Membuat label judul baru menggunakan teks yang diberikan di konstruktor.
  - Baris-baris berikutnya (setFont, setForeground, setHorizontalAlignment) mengatur gaya font, warna, dan perataan teks judul agar terlihat menonjol dan konsisten.
  - add(lblTitle, BorderLayout.NORTH);: Menambahkan label judul ini ke bagian atas panel, memastikan setiap panel turunan secara otomatis memiliki judul di posisi yang sama.

```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.awt.event.ActionEvent;
6 import java.awt.event.ActionListener;
7
8 public class PesananBarangPanel extends BaseFormPanel {
9
10    private String namaBarang;
11    private int idBarang;
12    private int idUser;
13    private double hargaSewa;
14    private int stok;
15
16    // Konstruktor menerima data barang dan user
17    public PesananBarangPanel(String namaBarang, double hargaSewa, int idBarang, int idUser, int stok) {
18        super("Pesanan");
19        this.namaBarang = namaBarang;
20        this.idBarang = idBarang;
21        this.idUser = idUser;
22        this.hargaSewa = hargaSewa;
23        this.stok = stok;
24
25        // Form panel untuk menampung komponen
26        JPanel formPanel = new JPanel(new GridBagLayout());
27        formPanel.setBackground(new Color(45, 45, 65));
28        GridBagConstraints gbc = new GridBagConstraints();
29        gbc.insets = new Insets(10, 10, 10, 10);
30        gbc.anchor = GridBagConstraints.NORTH;
31        gbc.fill = GridBagConstraints.HORIZONTAL;
32
33        Font labelFont = new Font("Segoe UI", Font.BOLD, 18);
34        Font fieldFont = new Font("Segoe UI", Font.PLAIN, 18);
35        Dimension fieldsize = new Dimension(300, 35);
36
37        // Label nama barang
38        JLabel lblNamaBarang = new JLabel("Barang: " + namaBarang);
39        lblNamaBarang.setForeground(Color.WHITE);
40        lblNamaBarang.setFont(labelFont);
41
42        // Label dan spinner untuk durasi sewa
43        JLabel lblDurasi = new JLabel("Durasi (hari*)");
44        lblDurasi.setForeground(Color.WHITE);
45        lblDurasi.setFont(labelFont);
46        Spinner spurasi = new Spinner(new SpinnerNumberModel(1, 1, 7, 1));
47        spurasi.setFont(fieldFont);
48        ((Spinner.DefaultEditor) spurasi.getEditor()).getTextField().setFont(fieldFont);
49        spurasi.setPreferredWidth(fieldsize);
50
51        // Label dan spinner untuk jumlah unit
52        JLabel lblJumlah = new JLabel("Jumlah Unit:");
53        lblJumlah.setForeground(Color.WHITE);
54        lblJumlah.setFont(labelFont);
55        Spinner spjumlah = new Spinner(new SpinnerNumberModel(1, 1, stok, 1));
56        spjumlah.setFont(fieldFont);
57        ((Spinner.DefaultEditor) spjumlah.getEditor()).getTextField().setFont(fieldFont);
58        spjumlah.setPreferredWidth(fieldsize);
59
60        // Label untuk menampilkan total harga
61        JLabel lblTotal = new JLabel("Total Harga: Rp 0");
62        lblTotal.setForeground(new Color(0, 255, 166));
63        lblTotal.setFont(labelFont);
64
65        // Tombol konfirmasi pesanan
66        JButton btnKonfirmasi = new JButton("Konfirmasi Pesanan");
67        btnKonfirmasi.setBackground(new Color(0, 255, 136));
68        btnKonfirmasi.setForeground(Color.BLACK);
69        btnKonfirmasi.setFont(new Font("Segoe UI", Font.BOLD, 16));
70        btnKonfirmasi.setPreferredSize(new Dimension(200, 35));
71
72        // Tambahkan komponen ke formPanel dengan GridBagLayout
73        gbc.gridx = 0;
74        gbc.gridy = 0;
75        gbc.gridwidth = 2;
76        formPanel.add(lblNamaBarang, gbc);
77
78        gbc.gridx++;
79        gbc.gridwidth = 1;
80        gbc.weightx = 0;
81        formPanel.add(lblDurasi, gbc);
82        gbc.gridx = 1;
83        gbc.weightx = 1;
84        formPanel.add(spurasi, gbc);
85
86        gbc.gridx = 0;
87        gbc.gridy++;
88        gbc.gridwidth = 2;
89        formPanel.add(lblJumlah, gbc);
90        gbc.gridx = 1;
91        gbc.weightx = 1;
92        formPanel.add(spjumlah, gbc);
93
94        gbc.gridx = 0;
95        gbc.gridy++;
96        gbc.gridwidth = 2;
97        gbc.anchor = GridBagConstraints.CENTER;
98        gbc.fill = GridBagConstraints.NONE;
99        formPanel.add(lblTotal, gbc);
100
101        // Tambahkan formPanel ke panel utama
102        add(formPanel, BorderLayout.CENTER);
103
104        // Listener untuk update total harga saat durasi/jumlah diubah
105        ChangeListener updateTotal = new ChangeListener() {
106            public void stateChanged(ChangeEvent e) {
107                int durasi = (int) spurasi.getValue();
108                int jumlah = (int) spjumlah.getValue();
109                double total = hargaSewa * durasi * jumlah;
110                lblTotal.setText("Total harga: Rp " + String.format("%,d", total));
111            }
112        };
113        spurasi.addActionListener(updateTotal);
114        spjumlah.addActionListener(updateTotal);
115        updateTotal.stateChanged(null); // Inisialisasi total harga
116
117        // Fungsi tombol konfirmasi validasi input dan navigasi ke pembayaran
118        btnKonfirmasi.addActionListener(e -> {
119            int durasi = (int) spurasi.getValue();
120            int jumlah = (int) spjumlah.getValue();
121            try {
122                // Validasi durasi dan jumlah
123                if (durasi < 1 || durasi > 7) {
124                    throw new IllegalArgumentException("Durasi harus antara 1 dan 7 hari.");
125                }
126                if (jumlah < 1 || jumlah > stok) {
127                    throw new IllegalArgumentException("Jumlah unit harus lebih besar dari 0 dan tidak lebih dari stok.");
128                }
129                // Kirim data ke PembayaranPanel melalui DashboardFrame
130                DashboardFrame dashboard = (DashboardFrame) SwingUtilities.getWindowAncestor(this);
131                dashboard.showPesananPanel(namaBarang, hargaSewa, idBarang, idUser, stok, durasi, jumlah);
132            } catch (IllegalArgumentException ex) {
133                JOptionPane.showMessageDialog(getContentPane(), ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
134            }
135        });
136    }
137}

```

Output:



Penjelasan:

Kode program ini mendefinisikan kelas `PesanBarangPanel`, sebuah komponen antarmuka pengguna (UI) yang dirancang untuk berfungsi sebagai halaman rincian dan konfirmasi pemesanan dalam aplikasi `VenueVibe`. Panel ini bertanggung jawab untuk menerima informasi barang yang telah dipilih dari panel sebelumnya (misalnya, `BarangPanel`), kemudian menampilkan formulir interaktif kepada pengguna. Di formulir ini, pengguna dapat menentukan durasi sewa yang diinginkan dan jumlah unit barang yang akan disewa. `PesanBarangPanel` secara otomatis menghitung dan menampilkan total harga sewa berdasarkan input pengguna.

Setelah pengguna mengonfirmasi pilihan mereka, panel ini akan melakukan validasi dasar terhadap input durasi dan jumlah unit. Jika input valid, ia akan meneruskan semua detail pesanan yang relevan ke panel pembayaran (`PembayaranPanel`) melalui `DashboardFrame` utama, yang bertindak sebagai pengelola navigasi antar halaman.

#### Breakdown Fungsionalitas:

- Pewarisan dari `BaseFormPanel`:

`PesanBarangPanel` mewarisi dari `BaseFormPanel` dengan memanggil `super("Pesanan")` di konstruktornya. Ini secara otomatis menyediakan struktur dasar panel, latar belakang, border, dan judul "Pesanan" yang konsisten.

- Penyimpanan Data Pesanan:  

Kelas ini memiliki *field* privat (namaBarang, idBarang, idUser, hargaSewa, stok) untuk menyimpan data barang dan ID pengguna yang diterima melalui konstruktor. Data ini penting untuk ditampilkan dan diteruskan ke tahap selanjutnya.
- Tata Letak Formulir (GridBagLayout):  

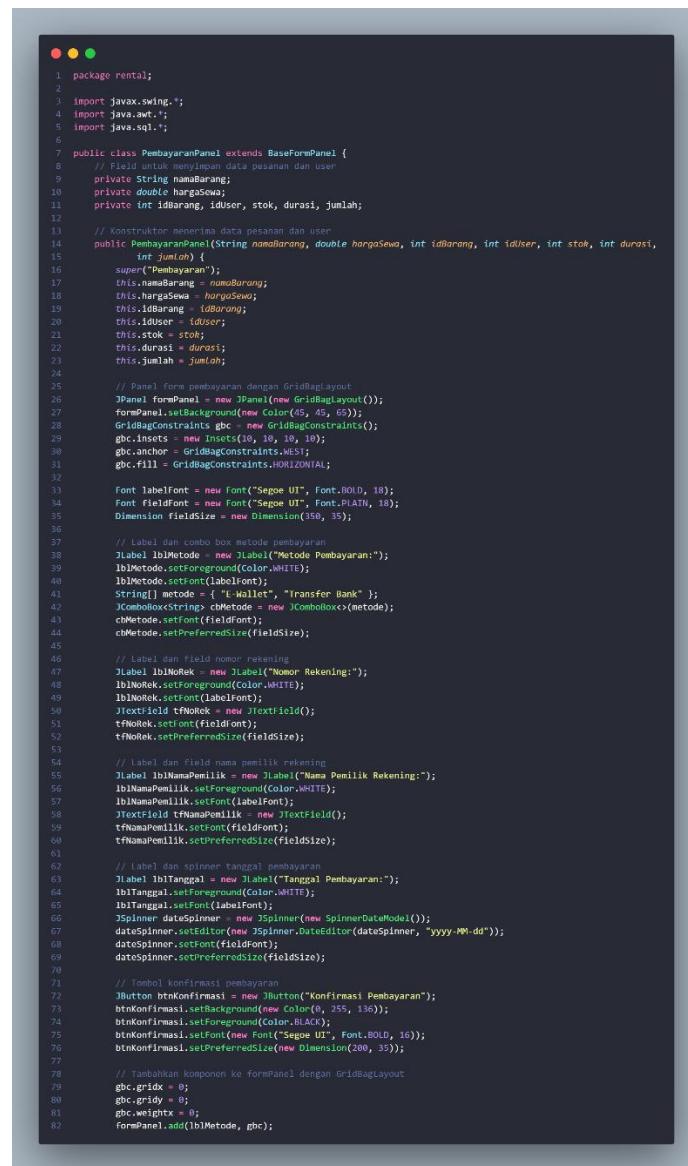
Sebuah JPanel (formPanel) dengan GridBagLayout digunakan untuk mengatur elemen-elemen formulir secara terstruktur. GridBagConstraints digunakan untuk mengontrol posisi, ukuran, dan perataan setiap komponen (label, spinner) di dalam grid.
- Komponen Input Interaktif:
  - JLabel lblNamaBarang: Menampilkan nama barang yang telah dipilih.
  - JSpinner spDurasi: Memungkinkan pengguna memilih durasi sewa dalam hari (dibatasi antara 1 hingga 7 hari).
  - JSpinner spJumlah: Memungkinkan pengguna memilih jumlah unit yang akan disewa (dibatasi antara 1 hingga stok barang yang tersedia).
- Penghitungan Total Harga Otomatis:  

Sebuah ChangeListener diimplementasikan dan ditambahkan ke spDurasi dan spJumlah. Setiap kali nilai pada salah satu spinner berubah, listener ini akan secara otomatis menghitung ulang total harga (hargaSewa \* durasi \* jumlah) dan memperbarui JLabel lblTotal secara *real-time*.
- Tombol Konfirmasi Pesanan (btnKonfirmasi):
  - Tombol ini adalah pemicu utama untuk melanjutkan proses.
  - Sebuah ActionListener ditambahkan untuk menangani klik tombol.
- Validasi Input:
  - Di dalam ActionListener tombol konfirmasi, dilakukan validasi dasar untuk memastikan durasi berada dalam rentang yang diizinkan dan jumlah unit tidak kurang dari 1 atau melebihi stok yang tersedia.
  - Jika validasi gagal, IllegalArgumentException akan dilemparkan dan JOptionPane akan menampilkan pesan kesalahan kepada pengguna.
- Navigasi ke Panel Pembayaran:

- Jika semua input valid, kode akan mendapatkan referensi ke DashboardFrame yang merupakan jendela utama aplikasi (SwingUtilities.getWindowAncestor(this)).
- Kemudian, metode showPembayaranPanel() pada DashboardFrame dipanggil, meneruskan semua detail pesanan yang diperlukan (nama barang, harga sewa, ID barang, ID pengguna, stok, durasi, dan jumlah unit). Ini menunjukkan bahwa DashboardFrame bertanggung jawab atas manajemen navigasi antar panel utama.

## D. Pembayaran

Kode Program:

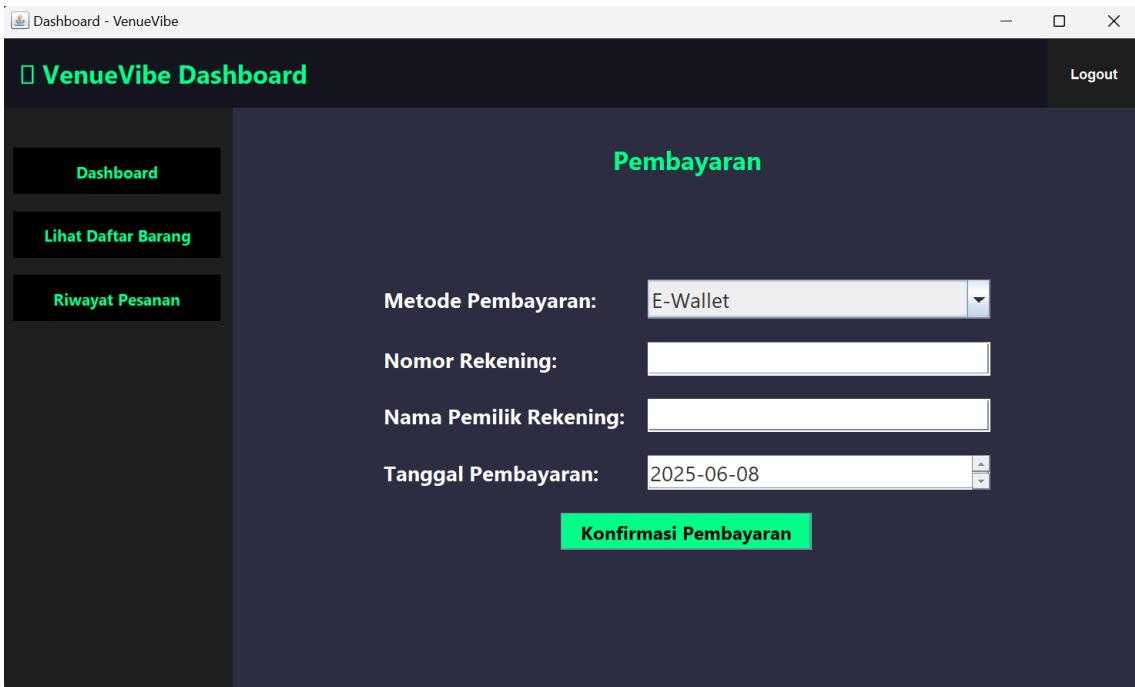


```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6
7 public class PembayaranPanel extends BaseFormPanel {
8     // Field untuk menyimpan data pesanan dan user
9     private String namaBarang;
10    private double hargaSewa;
11    private int idBarang, idUser, stok, durasi, jumlah;
12
13    // Konstruktor menerima data pesanan dan user
14    public PembayaranPanel(String namaBarang, double hargaSewa, int idBarang, int idUser, int stok, int durasi,
15                           int jumlah) {
16        super("Pembayaran");
17        this.namaBarang = namaBarang;
18        this.hargaSewa = hargaSewa;
19        this.idBarang = idBarang;
20        this.idUser = idUser;
21        this.stok = stok;
22        this.durasi = durasi;
23        this.jumlah = jumlah;
24
25        // Panel form pembayaran dengan GridBagLayout
26        JPanel formPanel = new JPanel(new GridBagLayout());
27        formPanel.setBackground(new Color(45, 45, 65));
28        GridBagConstraints gbc = new GridBagConstraints();
29        gbc.insets = new Insets(10, 10, 10, 10);
30        gbc.anchor = GridBagConstraints.WEST;
31        gbc.fill = GridBagConstraints.HORIZONTAL;
32
33        Font labelFont = new Font("Segoe UI", Font.BOLD, 18);
34        Font fieldFont = new Font("Segoe UI", Font.PLAIN, 18);
35        Dimension fieldSize = new Dimension(350, 35);
36
37        // Label dan combo box metode pembayaran
38        JLabel lblMetode = new JLabel("Metode Pembayaran:");
39        lblMetode.setForeground(Color.WHITE);
40        lblMetode.setFont(labelFont);
41        String[] metode = {"BANK", "Transfer Bank"};
42        JComboBox cbMetode = new JComboBox<>(metode);
43        cbMetode.setFont(fieldFont);
44        cbMetode.setPreferredSize(fieldSize);
45
46        // Label dan field nomor rekening
47        JLabel lblNrek = new JLabel("Nomor Rekening:");
48        lblNrek.setForeground(Color.WHITE);
49        lblNrek.setFont(labelFont);
50        JTextField tfNrek = new JTextField();
51        tfNrek.setFont(fieldFont);
52        tfNrek.setPreferredSize(fieldSize);
53
54        // Label dan field nama pemilik rekening
55        JLabel lblNamaPemilik = new JLabel("Nama Pemilik Rekening:");
56        lblNamaPemilik.setForeground(Color.WHITE);
57        lblNamaPemilik.setFont(labelFont);
58        JTextField tfNamaPemilik = new JTextField();
59        tfNamaPemilik.setFont(fieldFont);
60        tfNamaPemilik.setPreferredSize(fieldSize);
61
62        // Label dan spinner tanggal pembayaran
63        JLabel lblTanggal = new JLabel("Tanggal Pembayaran:");
64        lblTanggal.setForeground(Color.WHITE);
65        lblTanggal.setFont(labelFont);
66        JSpinner dateSpinner = new JSpinner(new SpinnerDateModel());
67        dateSpinner.setEditor(new JSpinner.DateEditor(dateSpinner, "yyyy-MM-dd"));
68        dateSpinner.setFont(fieldFont);
69        dateSpinner.setPreferredSize(fieldSize);
70
71        // Tombol konfirmasi pembayaran
72        JButton btnKonfirmasi = new JButton("Konfirmasi Pembayaran");
73        btnKonfirmasi.setBackground(new Color(0, 255, 136));
74        btnKonfirmasi.setForeground(Color.BLACK);
75        btnKonfirmasi.setFont(new Font("Segoe UI", Font.BOLD, 16));
76        btnKonfirmasi.setPreferredSize(new Dimension(200, 35));
77
78        // Tambahkan komponen ke formPanel dengan GridBagLayout
79        gbc.gridx = 0;
80        gbc.gridy = 0;
81        gbc.weightx = 0;
82        formPanel.add(lblMetode, gbc);

```

Output:



Penjelasan:

Kode program ini mendefinisikan kelas `PembayaranPanel`, yang merupakan salah satu panel antarmuka pengguna (UI) dalam aplikasi `VenueVibe`. Panel ini memiliki tanggung jawab untuk memfasilitasi proses pembayaran setelah pengguna mengonfirmasi detail pesanan di `PesanBarangPanel`. `PembayaranPanel` menerima semua detail pesanan dari panel sebelumnya melalui konstruktornya, kemudian menampilkan formulir di mana pengguna dapat memilih metode pembayaran, memasukkan informasi rekening (nomor dan nama pemilik), serta tanggal pembayaran. Fungsionalitas utama dari panel ini adalah **memproses transaksi penuh**, yang meliputi:

1. Mengumpulkan informasi pembayaran yang diperlukan dari pengguna.
2. Melakukan validasi awal terhadap input pengguna untuk memastikan semua *field* penting telah diisi.
3. Berinteraksi dengan database untuk menyimpan pesanan baru, mengurangi stok barang yang disewa, dan menyimpan detail pembayaran.
4. Memberikan umpan balik kepada pengguna mengenai status transaksi.
5. Mengarahkan pengguna ke halaman riwayat pesanan setelah transaksi berhasil diproses, memungkinkan mereka melacak status pesanan.

## **Breakdown Fungsionalitas:**

- Pewarisan dari BaseFormPanel:

PembayaranPanel mewarisi dari BaseFormPanel dengan memanggil super("Pembayaran") di konstruktornya, yang secara otomatis menyediakan struktur dasar panel, latar belakang, border, dan judul "Pembayaran" yang konsisten.

- Penyimpanan Data Pesanan:

Memiliki *field* privat (namaBarang, hargaSewa, idBarang, idUser, stok, durasi, jumlah) yang menyimpan detail lengkap pesanan yang diterima dari konstruktor. Data ini krusial untuk dicatat dalam database.

- Tata Letak Formulir (GridLayout):

Formulir pembayaran diatur menggunakan GridLayout dalam sebuah JPanel (formPanel), memastikan penempatan komponen yang fleksibel dan rapi dengan *padding* yang memadai.

- Komponen Input UI:

- JComboBox cbMetode: Memungkinkan pengguna memilih antara "E-Wallet" atau "Transfer Bank" sebagai metode pembayaran.
- JTextField tfNoRek: Untuk input nomor rekening pembayaran.
- JTextField tfNamaPemilik: Untuk input nama pemilik rekening.
- JSpinner dateSpinner: Memudahkan pengguna memilih tanggal pembayaran dengan format yang sudah ditentukan (yyyy-MM-dd).

- Tombol Konfirmasi Pembayaran (btnKonfirmasi):

- Tombol ini memicu serangkaian operasi ketika diklik.
- Sebuah ActionListener ditambahkan untuk menangani semua logika transaksi.

- Validasi Input Frontend:

Sebelum berinteraksi dengan database, kode melakukan validasi untuk memastikan bahwa semua *field* input wajib (nomor rekening, nama pemilik, metode, dan tanggal) tidak kosong. Jika ada yang kosong, JOptionPane akan menampilkan pesan error dan menghentikan proses.

- Logika Transaksi Database (Dalam ActionListener):

- Menggunakan try-with-resources untuk Connection dan PreparedStatement untuk memastikan sumber daya database ditutup secara otomatis.
  - INSERT Pesanan Baru: Sebuah *query* INSERT dijalankan untuk menyimpan detail pesanan (idUser, idBarang, durasiSewa, totalHarga, status ("Menunggu"), jumlahUnit) ke tabel pesanan. Statement.RETURN\_GENERATED\_KEYS digunakan untuk mendapatkan idPesanan yang baru dibuat, yang penting untuk langkah selanjutnya.
  - UPDATE Stok Barang: Stok barang di tabel barang dikurangi sebesar jumlah unit yang disewa melalui *query* UPDATE.
  - INSERT Data Pembayaran: Jika pesanan baru berhasil disimpan dan idPesananBaru didapatkan, sebuah *query* INSERT dijalankan untuk menyimpan detail pembayaran (idPesanan, metodePembayaran, nomorRek, namaPemilikRek, tglPembayaran, status ("Menunggu")) ke tabel pembayaran. Konversi java.util.Date ke java.sql.Date dilakukan untuk kompatibilitas database.
- Umpam Balik Pengguna dan Navigasi:
  - Setelah semua operasi database berhasil tanpa kesalahan, JOptionPane akan muncul untuk memberitahu pengguna bahwa pembayaran sedang dalam proses konfirmasi.
  - Selanjutnya, aplikasi akan secara otomatis berpindah ke RiwayatPesananPanel melalui panggilan dashboard.showRiwayatPesananPanel(), yang diakses dengan mendapatkan referensi ke DashboardFrame utama.
- Penanganan Kesalahan Database:
 

Blok catch (Exception ex) menangkap *exception* yang mungkin terjadi selama validasi atau operasi database, menampilkan pesan error yang relevan kepada pengguna.

## E. Riwayat Pesanan

Kode Program :

```

1 package rental;
2
3 import javax.swing.*;
4 import java.awt.*;
5 import java.sql.*;
6 import java.util.ArrayList;
7 import javax.swing.table.JTableHeader; // << Tambahkan baris ini
8
9 public class RiwayatPesananPanel extends JPanel {
10     public RiwayatPesananPanel(int iduser) {
11         // Set layout dan warna background panel utama
12         setLayout(new BorderLayout());
13         setBackground(new Color(rg45, gi45, bi65));
14
15         // Judul panel
16         JLabel label = new JLabel(text:"Riwayat Pesanan");
17         label.setFont(new Font(name:"Segoe UI", Font.BOLD, size22));
18         label.setForeground(new Color(rg45, gi55, bi138));
19         label.setHorizontalTextPosition(SwingConstants.CENTER);
20         label.setBorder(BorderFactory.createEmptyBorder(top10, left10, bottom10, right10));
21         add(label, BorderLayout.NORTH);
22
23         // Ambil data pesanan dari database untuk user ini
24         ArrayList<PesananRiwayat> daftarPesanan = getPesananFromDB(iduser);
25
26         // Definisikan nama kolom tabel
27         String[] kolom = {"ID Pesanan", "Nama Barang", "Status", "Total Harga", "Jumlah Unit",
28         | "Durasi Sewa", "Aksi" };
29
30         // Siapkan data untuk tabel
31         Object[][] data = new Object[daftarPesanan.size()][kolom.length];
32         for (int i=0; i<daftarPesanan.size(); i++) {
33             PesananRiwayat p = daftarPesanan.get(i);
34             data[i][0] = p.getIdpesan();
35             data[i][1] = p.getNamaBarang();
36             data[i][2] = p.getStatus();
37             data[i][3] = p.getTotalHarga();
38             data[i][4] = "%." + String.format(format:"%.0f", p.getTotalHarga());
39             data[i][5] = p.getJumlahUnit();
40             data[i][6] = p.getDurasiSewa() + " hari";
41             // Jika status lunas, tombol aktif, jika tidak, tombol nonaktif
42             data[i][7] = status.equals(ignoreCase("lunas")) ? "Cetak Struk" : "Tidak Tersedia";
43         }
44
45         // Buat tabel untuk menampilkan data pesanan
46         JTable table = new JTable(data, kolom);
47         public boolean isCellEditable(int row, int column) {
48             return column == 5; // hanya kolom aksi yang bisa di klik
49         }
50         table.getColumn("Aksi").setCellRenderer(new ButtonRenderer());
51         table.getColumn("Aksi").setCellEditor(new ButtonEditor(new JCheckBox(), daftarPesanan));
52
53         // Atur tampilan tabel agar lebih nyaman dilihat
54         table.setFillsViewportHeight(true);
55         table.setAutoWidth(true);
56         table.setFont(new Font(name:"Segoe UI", Font.PLAIN, size14));
57
58         // Styling header kolom agar senada dengan card dan font putih
59         TableHeader header = table.getTableHeader();
60         header.setFont(new Font(name:"Segoe UI", Font.BOLD, size18));
61         header.setBackground(new Color(rg45, gi45, bi65)); // warna card
62         header.setForeground(Color.WHITE); // font putih
63         header.setOpaque(false);
64         header.setReorderingAllowed(false); // optional
65
66         table.setBackground(new Color(rg45, gi45, bi65));
67         table.setForeground(Color.WHITE);
68         table.setSelectionBackground(new Color(rg45, gi55, bi138));
69         table.setSelectionForeground(Color.BLACK);
70
71         // Bungkus tabel dengan JScrollPane agar bisa discroll jika data banyak
72         JScrollPane scrollPane = new JScrollPane(table);
73         scrollPane.setViewportView(table);
74         scrollPane.setBackground(new Color(rg45, gi45, bi65));
75         scrollPane.setBorder(BorderFactory.createEmptyBorder(top10, left10, bottom10, right10));
76
77         // Tambahkan scrollpane (yang berisi tabel) ke panel utama
78         add(scrollPane, BorderLayout.CENTER);
79
80         System.out.println("iduser di RiwayatPesananPanel: " + iduser);
81     }
82
83     // Method untuk mengambil data seluruh dari database berdasarkan iduser
84     private ArrayList<PesananRiwayat> getPesananFromDB(int iduser) {
85         ArrayList<PesananRiwayat> pesanan = new ArrayList();
86         Connection conn = DriverManager.getConnection(url);
87         // Query join ke tabel barang dan pembayaran untuk mendapatkan info lengkap
88         String sql = "SELECT p.idpesan, b.namabarang, p.tglpembayaran, p.status, p.totalharga, p.jumlahunit, p.durasipesan, u.username "
89         | "FROM pesanan p "
90         | "JOIN barang b ON p.idbarang = b.idbarang "
91         | "JOIN user u ON p.iduser = u.iduser "
92         | "LEFT JOIN pembayaran pb ON p.idpesan = pb.idpesan "
93         | "AND pb.iduser = u.iduser "
94         | "WHERE p.iduser = ? ";
95         PreparedStatement ps = connection.prepareStatement(sql);
96         ps.setString(1, iduser);
97         ResultSet rs = ps.executeQuery();
98         while (rs.next()) {
99             PesananRiwayat r = new PesananRiwayat();
100             r.setIdpesan(rs.getInt("idpesan"));
101             r.setNamaBarang(rs.getString("namabarang"));
102             r.setTglpembayaran(rs.getDate("tglpembayaran"));
103             r.setStatus(rs.getString("status"));
104             r.setTotalharga(rs.getDouble("totalharga"));
105             r.setJumlahUnit(rs.getInt("jumlahunit"));
106             r.setDurasiSewa(rs.getInt("durasipesan"));
107             r.setUsername(rs.getString("username")); // cekan username
108         }
109     } catch (Exception e) {
110         JOptionPane.showMessageDialog(this, "Gagal mengambil data pesanan: " + e.getMessage());
111     }
112     return pesanan;
113 }
114
115 // Render untuk tombol di tabel
116 class ButtonRenderer extends JButton implements javax.swing.table.TableCellRenderer {
117     public ButtonRenderer() {
118         setOpaque(true);
119     }
120
121     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean hasFocus,
122         | int row, int column) {
123         if (value != null) {
124             // Jika tombol tidak aktif, tambahkan style
125             if ("Tidak Tersedia".equals(value)) {
126                 setEnabled(false);
127                 setForeground(Color.DARK_GRAY);
128                 setFont(new Font("Times New Roman", Font.PLAIN, 12));
129             } else {
130                 setEnabled(true);
131                 setForeground(new Color(rg45, gi55, bi138));
132                 setFont(new Font("Times New Roman", Font.PLAIN, 12));
133             }
134         }
135         return this;
136     }
137
138     // Editor untuk aksi tombol di tabel
139     class ButtonEditor extends DefaultCellEditor {
140         private JCheckBox checkBox;
141         private boolean clicked;
142         private PesananRiwayat pesanan;
143         private JTableHeader daftarPesanan;
144         private String label;
145
146         public ButtonEditor(JCheckBox checkBox, ArrayList<PesananRiwayat> daftarPesanan) {
147             super(checkBox);
148             this.checkBox = checkBox;
149             this.daftarPesanan = daftarPesanan;
150             this.pesanan = new Pesanan();
151             this.clicked = false;
152             this.pesanan.setAksi(label);
153             this.button.addActionListener(buttonActionListener);
154             this.button.addMouseListener(buttonMouseListener);
155             this.button.setOpaque(true);
156             this.button.setFocusable(false);
157             this.button.setRolloverEnabled(false);
158             this.button.setRolloverText("");
159             this.button.setRolloverImage(null);
160             this.button.setRolloverIcon(null);
161             this.button.setRolloverText(null);
162             this.button.setRolloverImage(null);
163             this.button.setRolloverIcon(null);
164             this.button.setRolloverText(null);
165             this.button.setRolloverImage(null);
166             this.button.setRolloverIcon(null);
167         }
168
169         public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row,
170             | int column) {
171             label = (String) value;
172             pesanan.setAksi(label);
173             pesanan.setDaftarPesanan(daftarPesanan);
174             pesanan.setRow(row);
175             pesanan.setLabel(label);
176             if ("Cetak Struk".equals(label)) {
177                 button.setRolloverImage(null);
178                 button.setRolloverIcon(null);
179                 button.setRolloverText(null);
180                 button.setRolloverImage(null);
181                 button.setRolloverIcon(null);
182                 button.setRolloverText(null);
183             } else {
184                 button.setRolloverImage(null);
185                 button.setRolloverIcon(null);
186                 button.setRolloverText(null);
187             }
188         }
189     }
190 }

```

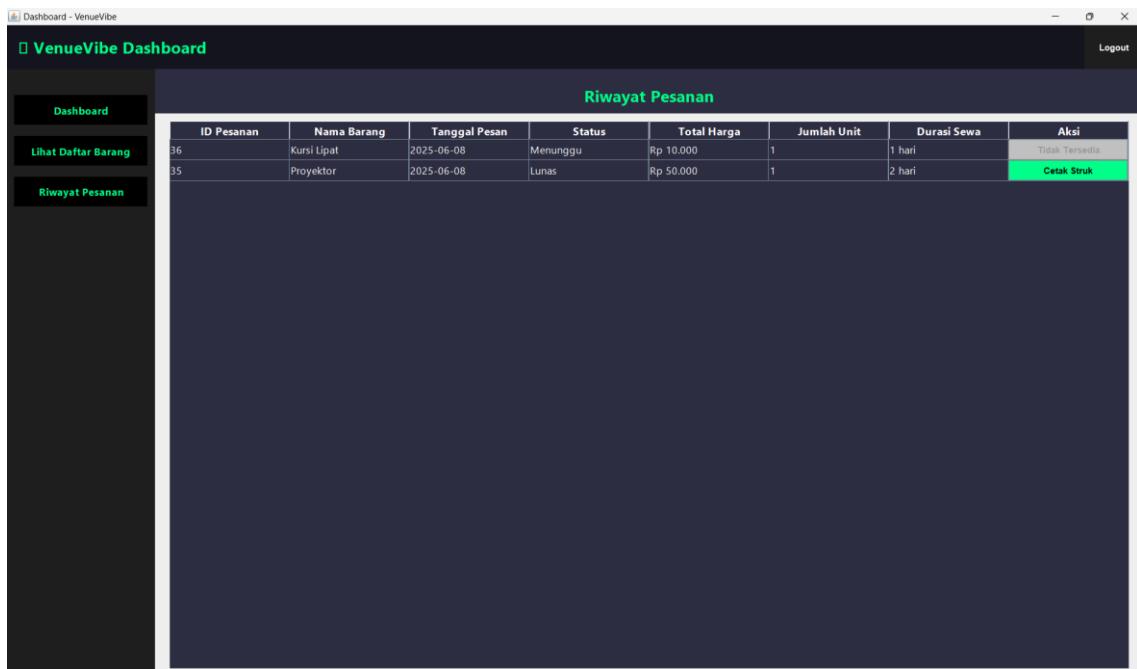
```

1    clicked = true;
2    return button;
3  }
4
5  public Object getColliderValue() {
6    if (clicked && "Cetak Struk".equals(label)) {
7      showstrukDialog(pesan);
8    }
9    clicked = false;
10   return label;
11 }
12
13 private void showstrukDialog(PesananRiyata p) {
14   JPanel panel = new JPanel();
15   panel.setBackground(new Color(245, 245, 255));
16   panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
17   panel.setBorder(BorderFactory.createCompoundBorder(
18     BorderFactory.createTitledBorder("New Color(#25, #255, #136), thickness:2, rounded:true),
19     BorderFactory.createCompoundBorder(
20       BorderFactory.createTitledBorder("top25, left25, bottom20, right15),
21       BorderFactory.createCompoundBorder(
22         BorderFactory.createTitledBorder("top25, left25, bottom20, right15),
23         BorderFactory.createTitledBorder("top25, left25, bottom20, right15));
24   );
25
26   JLabel title = new JLabel("Sous Barang VersusBar");
27   title.setFont(new Font("Sage UI", Font.BOLD, size18));
28   title.setForeground(new Color(255, 255, 255));
29   title.setAlignment(Component.CENTER_ALIGNMENT);
30   panel.add(title);
31   panel.add(Box.createVerticalStrut(height18));
32
33   // Panel isi struk dengan GridBagLayout agar label dimulai rata
34   JPanel isiPanel = new JPanel(new GridBagLayout());
35   isiPanel.setBackground(new Color(245, 245, 255));
36   isiPanel.setLayout(new GridBagLayout());
37   GridBagConstraints gbc = new GridBagConstraints();
38   gbc.insets = new Insets(5, 5, 5, 5);
39   gbc.anchor = GridBagConstraints.WEST;
40   gbc.gridx = 0;
41   gbc.gridy = 0;
42
43   Font isifont = new Font("Sage UI", Font.PLAIN, size15);
44   Color labelColor = new Color(255, 255, 255);
45   Color valueColor = Color.MCGEE;
46
47   // Helper untuk tambah baris
48   addStrukRow(isiPanel, gbc, label1, "Nama Pesanan", isifont, labelColor, valueColor);
49   addStrukRow(isiPanel, gbc, label2, "ID Pesanan", p.getIdPesanan(), isifont, labelColor, valueColor);
50   addStrukRow(isiPanel, gbc, label3, "Nama Barang", p.getNamaBarang(), isifont, labelColor, valueColor);
51   addStrukRow(isiPanel, gbc, label4, "Tanggal", p.getTanggalPesan(), isifont, labelColor, valueColor);
52   addStrukRow(isiPanel, gbc, label5, "Status", p.getStatus(), isifont, labelColor, valueColor);
53   addStrukRow(isiPanel, gbc, label6, "Jumlah", String.valueOf(p.getJumlahUnit()), isifont, labelColor, valueColor);
54   addStrukRow(isiPanel, gbc, label7, "Durasi", p.getDurasiSewa() + " hari", isifont, labelColor, valueColor);
55   addStrukRow(isiPanel, gbc, label8, "Total", "Rp " + String.format("%.2f", p.getTotalHarga()), isifont, labelColor,
56   valueColor);
57
58   panel.add(isiPanel);
59
60   // Styling tombol OK di JOptionPane (global)
61   UIManager.put("Key/Button.background", new Color(255, 255, 255));
62   UIManager.put("Key/Button.foreground", Color.BLACK);
63   UIManager.put("Key/Button.Font", new Font("Sage UI", Font.BOLD, size14));
64   UIManager.put("Key/OptionPane.background", new Color(245, 245, 255));
65   UIManager.put("Key/Panel.background", new Color(245, 245, 255));
66
67   JOptionPane.showMessageDialog(parentComponent(null, panel, title="Struk Pedenilan", JOptionPane.PLAIN_MESSAGE);
68
69   // Reset UIManager
70   UIManager.put("Key/Button.background", valueColor);
71   UIManager.put("Key/Button.foreground", valueColor);
72   UIManager.put("Key/Button.Font", valueColor);
73   UIManager.put("Key/OptionPane.background", valueColor);
74   UIManager.put("Key/Panel.background", valueColor);
75
76   // Helper method untuk menambah baris struk
77   private void addStrukRow(JPanel panel, GridBagConstraints gbc, String label, String value, Font font,
78   Color labelColor, Color valueColor) {
79     JLabel label1 = new JLabel(label + ":");
80     label1.setFont(font);
81     label1.setForeground(labelColor);
82     gbc.weightx = 0.4;
83     panel.add(label1, gbc);
84
85     JLabel lvalue = new JLabel(value);
86     lvalue.setFont(font);
87     lvalue.setForeground(valueColor);
88     gbc.weightx = 0.6;
89     panel.add(lvalue, gbc);
90
91     gbc.gridy++;
92   }
93
94   public boolean stopCellEditing() {
95     clicked = false;
96     return super.stopCellEditing();
97   }
98 }
99
100 }

1 package rental;
11
12 public class PesananRiyata {
13
14   private int idPesanan;
15   private String namaBarang;
16   private String tanggalPesan;
17   private String status;
18   private double totalHarga;
19   private int jumlahUnit;
20   private int durasiSewa;
21   private String username;
22
23   // Konstruktur untuk mengisi semua atribut saat objek dibuat
24   public PesananRiyata(int idPesanan, String namaBarang, String tanggalPesan, String status, double totalHarga, int jumlahUnit, int durasiSewa, String username) {
25     this.idPesanan = idPesanan;
26     this.namaBarang = namaBarang;
27     this.tanggalPesan = tanggalPesan;
28     this.status = status;
29     this.totalHarga = totalHarga;
30     this.jumlahUnit = jumlahUnit;
31     this.durasiSewa = durasiSewa;
32     this.username = username;
33   }
34
35   // Getter untuk mengambil nilai setiap atribut (digunakan saat menampilkan di tabel)
36   public int getIdPesanan() { return idPesanan; }
37   public String getNamaBarang() { return namaBarang; }
38   public String getTanggalPesan() { return tanggalPesan; }
39   public String getStatus() { return status; }
40   public double getTotalHarga() { return totalHarga; }
41   public int getJumlahUnit() { return jumlahUnit; }
42   public int getDurasiSewa() { return durasiSewa; }
43   public String getUsername() { return username; }
44 }

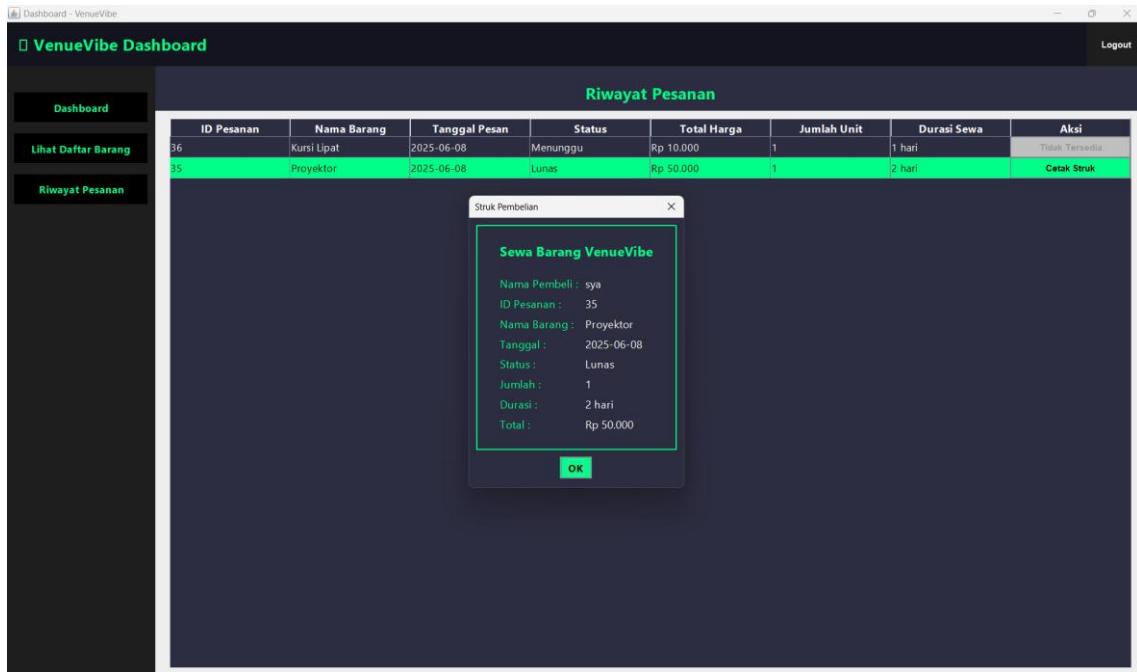
```

## Output :



The screenshot shows the VenueVibe Dashboard with the 'Riwayat Pesanan' (Order History) panel selected. The panel displays a table of order history:

ID Pesanan	Nama Barang	Tanggal Pesan	Status	Total Harga	Jumlah Unit	Durasi Sewa	Aksi
36	Kursi Lipat	2025-06-08	Menunggu	Rp 10.000	1	1 hari	Tidak Tersedia
35	Proyektor	2025-06-08	Lunas	Rp 50.000	1	2 hari	Cetak Struk

The screenshot shows the same dashboard, but a modal dialog titled 'Struk Pembelian' (Purchase Receipt) is open over the 'Riwayat Pesanan' panel. The dialog displays the details of the order with ID 35:

**Sewa Barang VenueVibe**

Nama Pembeli :	sya
ID Pesanan :	35
Nama Barang :	Proyektor
Tanggal :	2025-06-08
Status :	Lunas
Jumlah :	1
Durasi :	2 hari
Total :	Rp 50.000

At the bottom right of the dialog is an 'OK' button.

## Penjelasan :

Aplikasi rental ini memiliki modul RiwayatPesananPanel yang berfungsi sebagai halaman riwayat pesanan khusus untuk setiap pengguna yang masuk ke sistem. Tujuan utama dari panel ini adalah untuk menampilkan daftar komprehensif dari semua barang yang pernah disewa oleh pengguna tersebut. Di sisi lain, kelas PesananRiwayat bertindak sebagai model data atau cetak biru sederhana untuk merepresentasikan satu

entri riwayat pesanan. Interaksi antara pengguna dan data pesanan (misalnya, mencetak struk) juga dikelola sepenuhnya oleh RiwayatPesananPanel melalui komponen interaktifnya.

### **Breakdown & Fungsionalitas :**

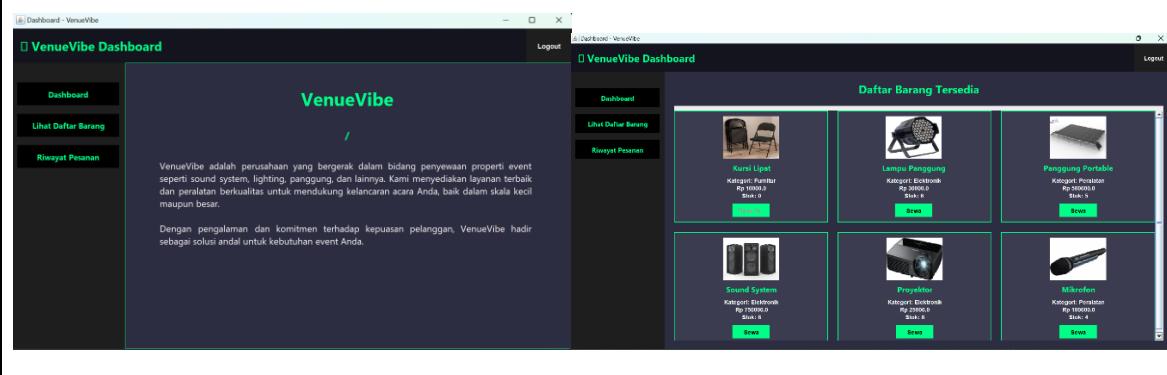
- Inisialisasi Halaman Riwayat Pesanan (RiwayatPesananPanel):
  - Pengaturan Tampilan Awal: Panel ini dimulai dengan konfigurasi dasar seperti tata letak (layout) dan warna latar belakang yang seragam (gelap) untuk menciptakan pengalaman pengguna yang konsisten. Sebuah judul besar "Riwayat Pesanan" diletakkan di bagian atas panel untuk kejelasan.
  - Pengambilan Data Pesanan: Saat panel dibuat, ia segera terhubung ke database untuk mengambil semua data pesanan yang relevan dengan ID pengguna yang sedang aktif. Data ini meliputi detail lengkap pesanan dari beberapa tabel terkait (pesanan, barang, user, pembayaran) untuk menyajikan informasi yang utuh.
  - Persiapan Data untuk Tabel: Data yang telah diambil dari database kemudian diolah dan disiapkan dalam format yang sesuai untuk ditampilkan di tabel (bentuk baris dan kolom). Selama proses ini, logika khusus diterapkan untuk kolom "Aksi": jika pesanan sudah berstatus "Lunas", teks "Cetak Struk" akan dipersiapkan, jika tidak, teks "Tidak Tersedia" akan digunakan.
- Penyajian Data dalam Tabel Interaktif:
  - Konstruksi Tabel (JTable): Data yang telah disiapkan ditampilkan dalam sebuah objek tabel (JTable). Tabel ini dikonfigurasi agar hanya kolom "Aksi" yang bisa diinteraksikan (diklik), sementara kolom lainnya bersifat hanya baca.
  - Visualisasi Tombol ("Aksi"): Untuk kolom "Aksi", sebuah komponen khusus (renderer) digunakan untuk menampilkan tombol atau teks secara visual. Jika status pesanan "Lunas", tombol "Cetak Struk" akan terlihat aktif dengan warna menonjol. Jika tidak, tombol akan terlihat nonaktif dengan warna abu-abu dan teks "Tidak Tersedia".
  - Penanganan Interaksi Tombol: Komponen khusus lainnya (editor) digunakan untuk mendeteksi dan menangani klik pada tombol "Cetak Struk". Ketika

tombol ini diklik, ia akan memicu fungsionalitas selanjutnya, yaitu menampilkan struk pembelian.

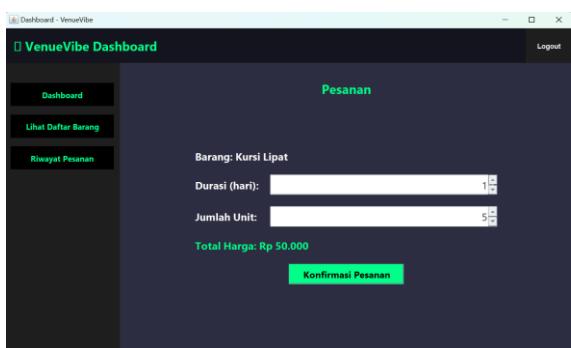
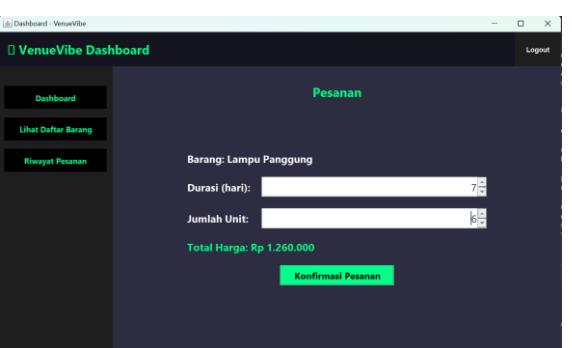
- Estetika dan Usabilitas Tabel: Tabel diberi gaya visual yang sesuai dengan tema aplikasi (warna latar belakang, font, warna teks, warna baris yang dipilih) dan juga diatur agar dapat digulir secara otomatis (JScrollPane) jika jumlah pesanan terlalu banyak, memastikan kenyamanan pengguna.
- Model Data Pesanan (PesananRiwayat):
  - Struktur Data: Kelas ini secara eksklusif berfokus pada penyimpanan data terkait satu pesanan. Ini memiliki atribut (variabel) untuk setiap bagian informasi pesanan (ID, nama barang, tanggal, status, harga, jumlah, durasi, username).
  - Inisialisasi Data: Sebuah konstruktor disediakan untuk memungkinkan objek PesananRiwayat dibuat dan diisi dengan semua data yang relevan dalam satu kali proses.
  - Akses Data Terkontrol: Metode getter (get...()) disediakan untuk setiap atribut. Ini adalah satu-satunya cara eksternal untuk membaca data dari objek PesananRiwayat, menjaga data internal tetap aman dan konsisten.
- Fungsionalitas "Cetak Struk":
  - Pemicu Dialog: Ketika pengguna mengklik tombol "Cetak Struk" pada baris pesanan yang berstatus "Lunas", sebuah fungsi internal akan dipanggil untuk memunculkan dialog struk.
  - Desain Dialog Struk: Dialog struk dibuat dengan panel terpisah yang memiliki tata letak dan gaya visual yang sama dengan panel utama (latar belakang gelap, aksen hijau).
  - Detail Struk: Dialog menampilkan ringkasan detail pesanan yang jelas dan mudah dibaca, seperti nama pembeli, ID pesanan, nama barang, tanggal transaksi, status ("Lunas"), jumlah unit, durasi sewa, dan total harga.
  - Styling Tombol Dialog: Bahkan tombol "OK" pada dialog struk diatur agar sesuai dengan tema aplikasi, dan pengaturan gaya ini direset setelah dialog ditutup untuk menghindari konflik.

## Pengujian Customer Section – Black Box Testing

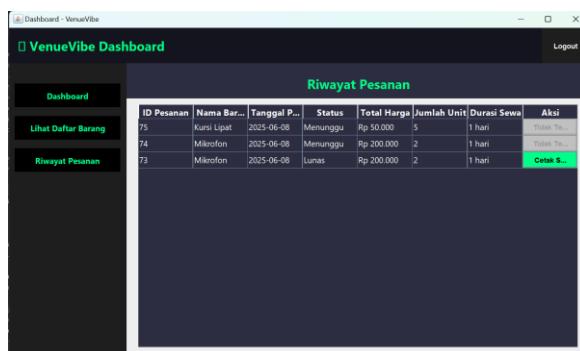
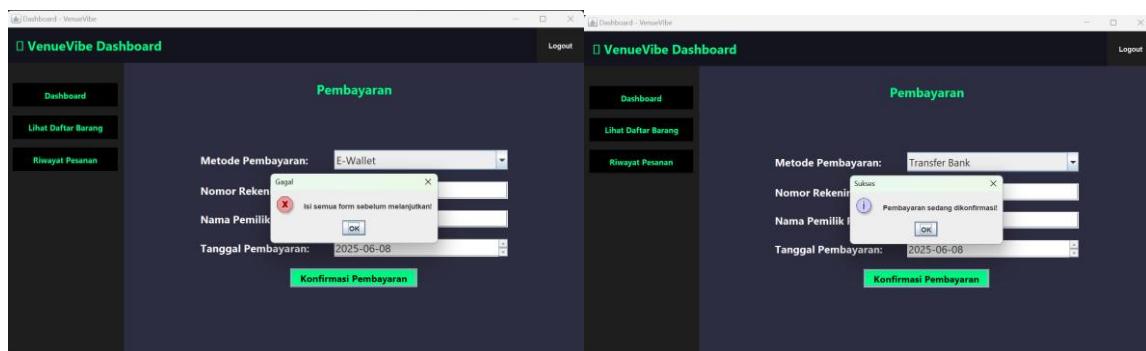
No.	Fitur	Test Case ID	Input Data	Hasil yang Diharapkan
1	Dashboard Customer	DC-001	Login sebagai role customer	Dashboard customer menampilkan sambutan atau informasi umum, dan menyediakan menu navigasi untuk "Lihat Daftar Barang" serta "Riwayat Pesanan".
2	Lihat Barang	LB-001	Klik menu "Lihat Daftar Barang"	Panel "Daftar Barang Tersedia" muncul, menampilkan daftar barang lengkap dengan gambar, nama, kategori, harga sewa, dan stok. Tombol "Sewa" terlihat aktif untuk barang dengan stok > 0 dan non-aktif untuk barang dengan stok = 0 (berubah menjadi "Tidak Tersedia").



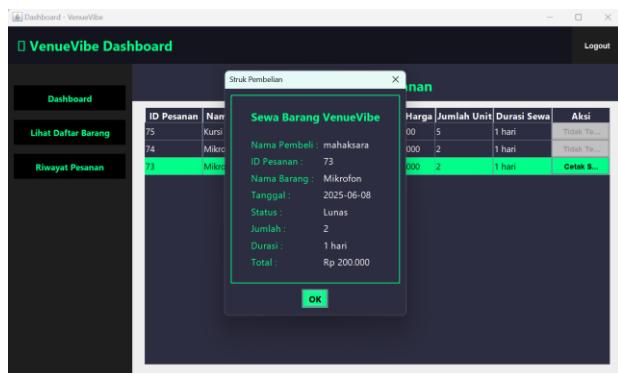
3	Pesanan	PS-001	Di "Daftar Barang Tersedia", klik tombol "Sewa" pada barang "Kursi Lipat" (stok > 0)	Panel "Pesanan" muncul dengan judul "Pesanan". Detail barang "Kursi Lipat" ditampilkan. Spinner "Durasi (hari)" dan "Jumlah Unit" memiliki nilai default 1. "Total Harga" otomatis terhitung dan ditampilkan sesuai harga sewa Lampu Panggung.
4	Pesanan	PS-002	Di panel "Pesanan", ubah "Jumlah Unit" menjadi 5	"Total Harga" otomatis diperbarui sesuai dengan hargaSewa * 3 * 2. JSpinner "Durasi (hari)" tidak dapat diubah melebihi 7 atau kurang dari 1 melalui klik panah. JSpinner "Jumlah Unit" tidak dapat diubah melebihi stok barang (misal: 6) atau kurang dari 1 melalui klik panah.

 				
Pembayaran	PB-001	Di panel "Pesanan", klik "Konfirmasi Pesanan" (setelah input valid)	Panel "Pembayaran" muncul dengan judul "Pembayaran". Form pembayaran (metode pembayaran, nomor rekening, nama pemilik rekening, tanggal pembayaran) ditampilkan.	
Pembayaran	PB-002	Di panel "Pembayaran", biarkan salah satu <i>field</i> kosong/tidak valid, lalu klik "Konfirmasi Pembayaran"	Muncul pesan error (JOptionPane) "Isi semua form sebelum melanjutkan!". Data pembayaran tidak tersimpan dan tidak ada navigasi.	
Pembayaran	PB-003	Di panel "Pembayaran", isi semua <i>field</i> dengan data valid (metode, no. rekening, nama pemilik, tanggal), lalu klik	Muncul pesan informasi (JOptionPane) "Pembayaran sedang dikonfirmasi!". Data pesanan dan pembayaran berhasil tersimpan ke database. Stok barang yang disewa berkurang di database. Aplikasi otomatis	

			"Konfirmasi Pembayaran"	berpindah ke panel "Riwayat Pesanan".
--	--	--	-------------------------	---------------------------------------



6	Riwayat Pesanan	RP-001	Di panel "Riwayat Pesanan" pada tabel ada kolom "Aksi", klik Cetak Struk pada pesanan yang berstatus "Lunas"	Muncul tampilan struk "Sewa Barang VenueVibe" yang berisi detail pesanan yang sudah lunas. Sementara pesanan yang statusnya menunggu, tidak bisa melakukan Cetak Struk atau Tidak Tersedia
---	-----------------	--------	--	--



## **BAB III**

### **KESIMPULAN**

Sistem rental properti event ini berhasil dikembangkan sebagai solusi modern untuk menggantikan sistem manual yang rentan kesalahan , menyediakan pengelolaan penyewaan yang otomatis dan terorganisir. Program ini mencakup fitur login, registrasi, dan reset password, dengan validasi yang memastikan keamanan data pengguna. Admin dapat mengelola data barang (menambah, mengedit, menghapus) dan mengonfirmasi pesanan pelanggan, termasuk memperbarui status pembayaran menjadi "Lunas". Di sisi pelanggan, mereka dapat melihat katalog barang, memesan barang dengan perhitungan harga otomatis, melakukan pembayaran, dan melacak riwayat pesanan mereka. Konsep Pemrograman Berorientasi Objek (PBO) seperti enkapsulasi, inheritansi, polimorfisme, abstraksi, dan modularitas diterapkan secara konsisten di seluruh kode, memastikan program mudah dikembangkan dan disesuaikan. Pengujian *black box* menunjukkan semua fitur berfungsi sesuai harapan, membuktikan efektivitas sistem ini dalam mendukung bisnis rental properti event.