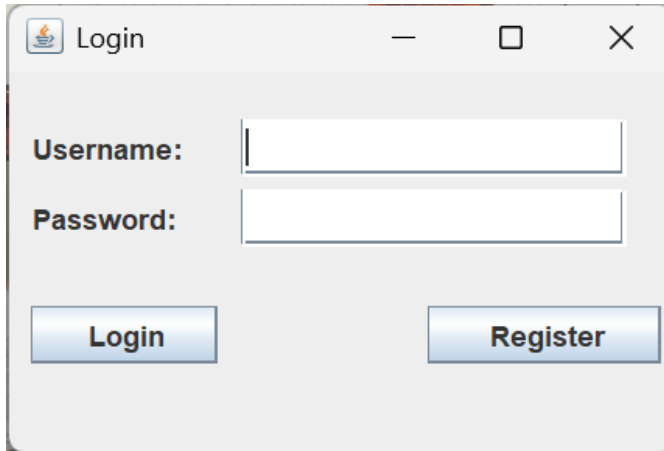


Vater E. T. Prawira
C11210039

Laporan Implementasi Desktop Client Sistem Pemesanan Tiket Konser Online

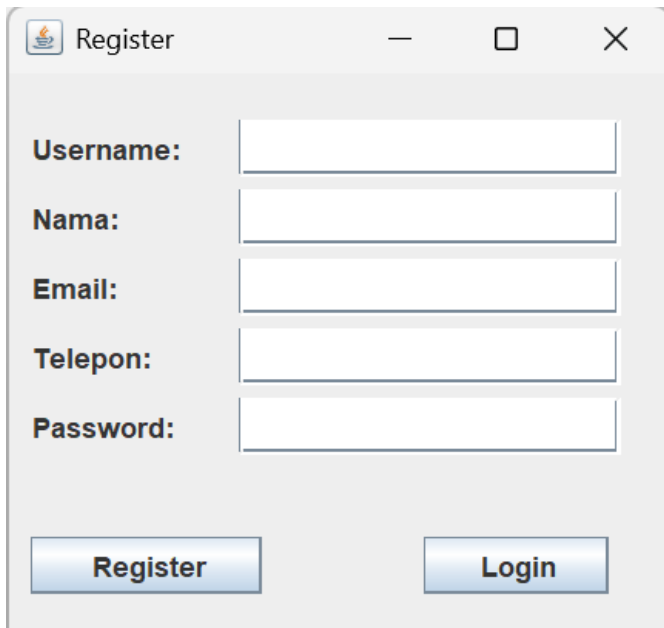
Github: <https://github.com/vaterprawira/Project-PAT>

Ketika aplikasi dijalankan maka user akan diminta untuk login dulu.

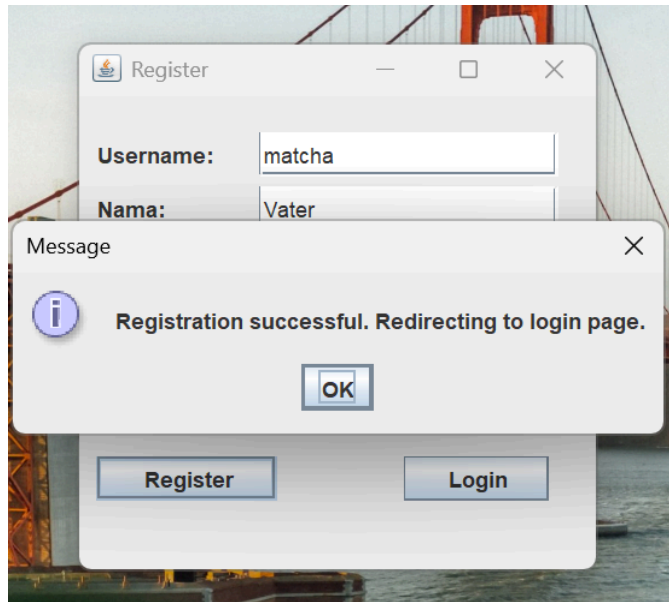


The screenshot shows a desktop application window titled "Login". It features a light gray background and a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The window contains two text input fields: "Username:" and "Password:". Below these fields are two buttons: "Login" and "Register". The "Login" button is positioned to the left of the "Register" button.

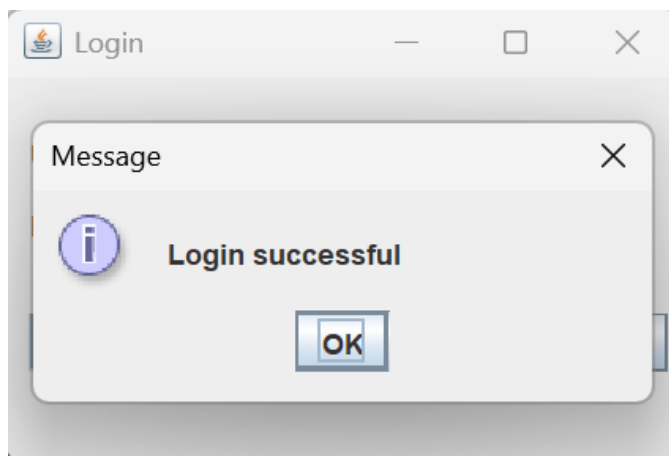
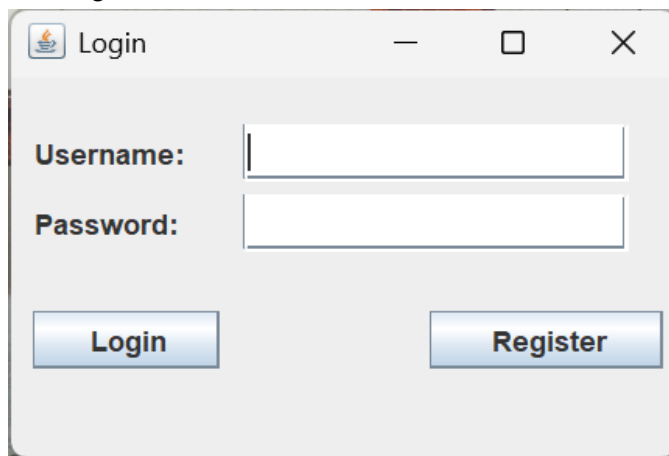
Jika user belum memiliki akun, dapat melakukan registrasi dahulu dengan mengklik tombol register, maka halaman registrasi akan muncul.



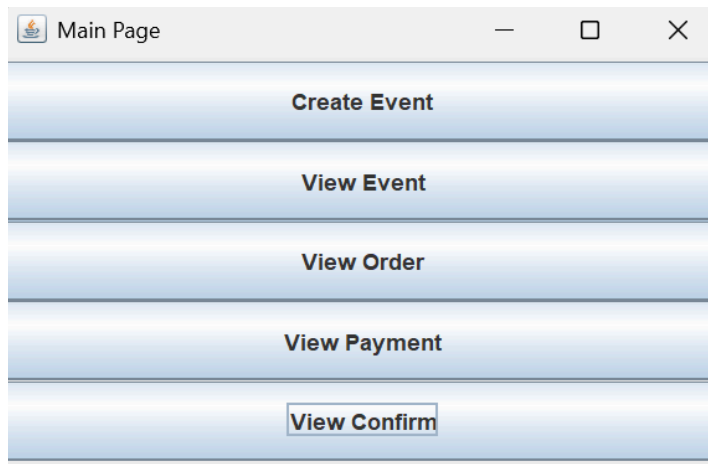
The screenshot shows a desktop application window titled "Register". It features a light gray background and a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The window contains five text input fields: "Username:", "Nama:", "Email:", "Telepon:", and "Password:". Below these fields are two buttons: "Register" and "Login". The "Register" button is positioned to the left of the "Login" button.



Jika registrasi berhasil maka user akan diarahkan ke halaman login.



Jika user berhasil melakukan login, maka halaman utama akan muncul yang dimana di halaman utama terdapat 5 tombol, yaitu create event, view event, view order, view payment, dan view confirm, jika ditekan akan membawa user ke masing-masing halaman.



Untuk tampilan halaman create event, seperti berikut:

The screenshot shows a window titled "Create Event" with a light gray background. It contains several form fields and buttons. The fields are labeled "Nama Konser:", "Artis:", "Deskripsi:", "Kategori Tiket:", "Lokasi:", "Tanggal:", and "Metode Pembayaran:". The "Kategori Tiket:" and "Metode Pembayaran:" fields contain tables with data. The "Create Event" button is at the bottom.

Kategori:	Cat 1	Harga:	200	Jumlah:	1000	Remove
Kategori:	Cat 2	Harga:	300	Jumlah:	1500	Remove

Add Kategori Tiket

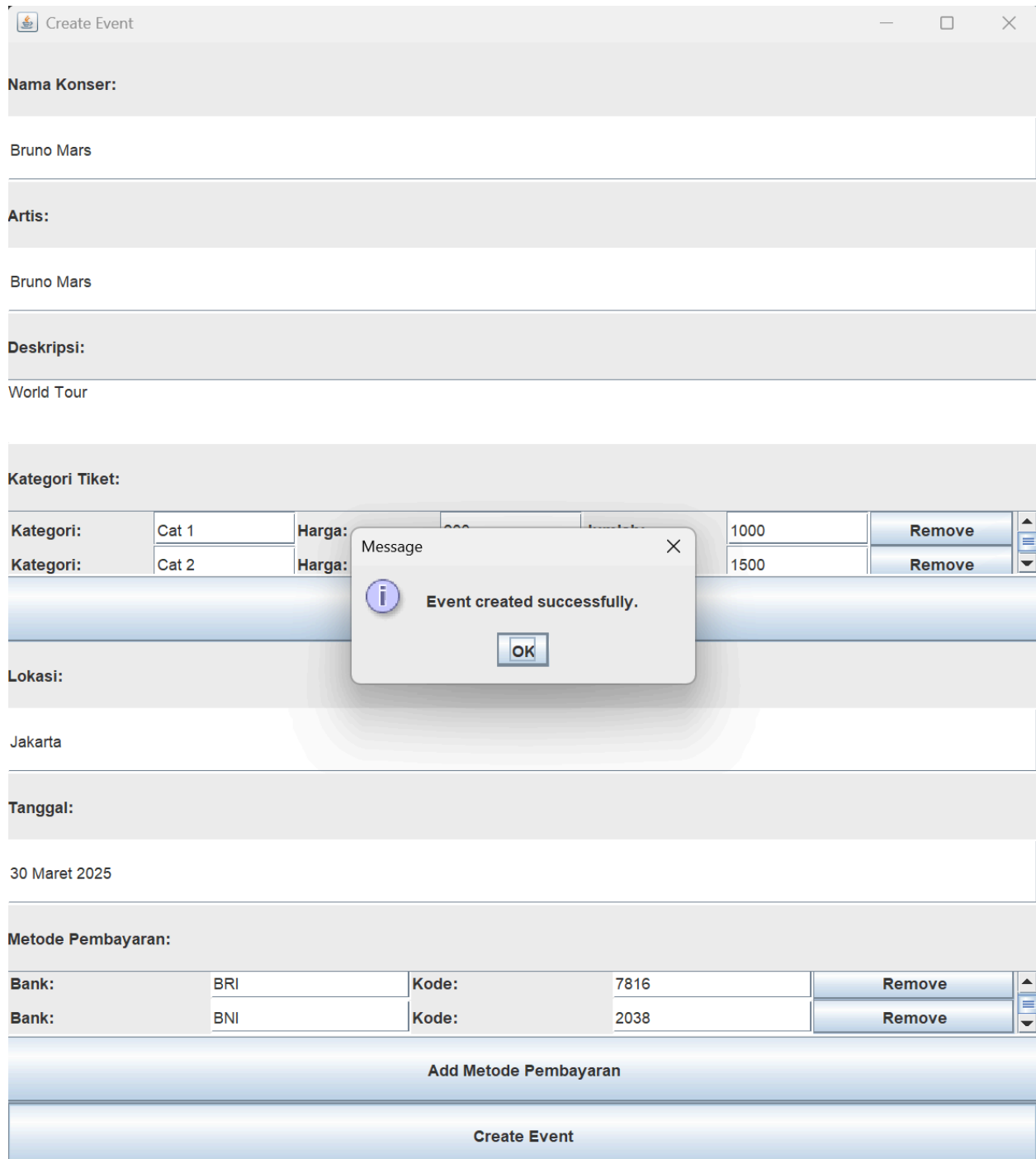
Bank:	BRI	Kode:	7816	Remove
Bank:	BNI	Kode:	2038	Remove

Add Metode Pembayaran

Create Event

Pada kolom pengisian kategori tiket dan metode pembayaran cukup berbeda dengan kolom pengisian yang lain, karena pada kolom pengisian kategori tiket dan metode pembayaran terdapat tombol untuk menambah kolom pengisian yang dimana pada kolom kategori tiket terdapat 3 kolom pengisian, yaitu kategori, harga, dan jumlah, sedangkan pada metode pembayaran terdapat 2 kolom pengisian, yaitu bank dan kode. Hal ini dilakukan untuk menyimpan data kategori tiket dan metode pembayaran dalam bentuk array.

Jika user berhasil menambahkan event ke database maka akan muncul notifikasi seperti, berikut:



The screenshot shows a 'Create Event' form with the following fields and data:

- Nama Konser:** Bruno Mars
- Artis:** Bruno Mars
- Deskripsi:** World Tour
- Kategori Tiket:**

Kategori:	Harga:	Jumlah:	
Cat 1	1000	1000	Remove
Cat 2	1500	1500	Remove
- Lokasi:** Jakarta
- Tanggal:** 30 Maret 2025
- Metode Pembayaran:**

Bank:	Kode:	
BRI	7816	Remove
BNI	2038	Remove

A modal message box is displayed in the center with the text: "Event created successfully." and an "OK" button.

At the bottom of the form, there are two buttons: "Add Metode Pembayaran" and "Create Event".

Kemudian untuk tampilan halaman view event seperti, berikut:

Detail Konser												
ID	Nama Konser	Artist	Jumlah Tiket	Lokasi	Tanggal	Kategori Tiket	Harga	Jumlah	Bank	Kode	Update	Delete
2	Nicole	NIKI	796	Jakarta	7 Desember 2024	VIP	1000	148	BCA	1890	Update	Delete
2	Nicole	NIKI	796	Jakarta	7 Desember 2024	Reguler	700	648	BNI	7826	Update	Delete
3	The Eras Tour	Taylor Swift	1000	Jakarta	7 Oktober 2024	VIP	1200	350	BCA	1890	Update	Delete
3	The Eras Tour	Taylor Swift	1000	Jakarta	7 Oktober 2024	Reguler	700	650	BNI	7826	Update	Delete
4	Bruno Mars	Bruno Mars	2500	Jakarta	30 Maret 2025	Cat 1	200	1000	BRI	7816	Update	Delete
4	Bruno Mars	Bruno Mars	2500	Jakarta	30 Maret 2025	Cat 2	300	1500	BNI	2038	Update	Delete

Di halaman tersebut terdapat tombol update dan delete, untuk tampilan menu update, seperti berikut:

Update Event

Nama Konser:

Artist:

Deskripsi:

Kategori Tiket:

Add Kategori

Kategori: VIP

Harga: 700

Jumlah: 700

Kategori: Reguler

Harga: 400

Jumlah: 1300

Metode Pembayaran:

Add Metode

Bank: BCA

Kode: 9183

Bank:

Kode:

Lokasi:

Tanggal:

Submit

Berikut tampilan menu update jika berhasil.

Update Event

Nama Konser:

Artist:

Deskripsi:

Kategori Tiket:

Kategori: VIP

Kategori: Reguler

Metode Pembayaran:

Add Metode

Bank: BCA

Kode: 9183

Bank:

Kode:

Lokasi:

Tanggal:

700

1300

Submit

Message

i

Event updated successfully!

OK

Untuk tampilan jika user melakukan delete pada halaman view event:

Detail Konser

ID	Nama Konser	Artist	Jumlah Tiket	Lokasi	Tanggal	Kategori Tiket	Harga	Jumlah	Bank	Kode	Update	Delete
2	Nicole	NIKI	796	Jakarta	7 Desember 2024	VIP	1000	148	BCA	1890	Update	Delete
2	Nicole	NIKI	796	Jakarta	7 Desember 2024	Reguler	700	648	BNI	7826	Update	Delete
3	The Eras Tour	Taylor Swift	1000	Jakarta	7 Oktober 2024	VIP	1000	1000	BCA	9890	Update	Delete
3	The Eras Tour	Taylor Swift	1000	Jakarta	7 Oktober 2024	Reguler	700	1000	BNI	6184	Update	Delete

Delete Event

?

Are you sure you want to delete this event?

Yes

No

Untuk tampilan halaman view order, seperti berikut:

Order			
Order ID	Event ID	Kategori Tiket	Total Harga
1	2	Kategori: VIP; Kategori: Reguler	3400.0

Untuk tampilan halaman view payment, seperti berikut:

Payment				
Payment ID	Order ID	Metode Pembayaran	Virtual Account	Status
1	1	BCA	1890367191017891	Menunggu pembayaran

Untuk tampilan halaman view confirmr, seperti berikut:

View Confirm		
Order ID	Status	Kode Booking
1	Pembayaran berhasil	J8NO72M

Penjelasan source code

UserApp.java

Metode `placeLoginComponents` menempatkan komponen-komponen GUI untuk form login, seperti label dan field untuk username dan password, serta tombol untuk login dan registrasi. Tombol login memiliki action listener yang menangani proses login. Ketika tombol login ditekan, aplikasi mengambil nilai username dan password dari field yang sesuai, kemudian mengirim permintaan POST ke API backend dengan menggunakan `HttpURLConnection`. Jika respons dari server adalah kode 200, berarti login berhasil dan sebuah pesan dialog "Login successful" ditampilkan. Jika tidak, pesan kesalahan ditampilkan.

Selain itu, ada juga tombol register yang ketika ditekan akan menutup jendela login dan menampilkan jendela registrasi dengan memanggil metode `showRegisterFrame`. Metode `showRegisterFrame` mirip dengan `showLoginFrame`, tetapi untuk form registrasi. Metode ini menambahkan komponen GUI untuk username, nama, email, telepon, dan password, serta dua tombol: satu untuk registrasi dan satu lagi untuk kembali ke halaman login. Ketika tombol registrasi ditekan, action listener-nya mengambil nilai dari field yang sesuai dan mengirim permintaan POST ke API backend dengan data registrasi. Jika registrasi berhasil, pengguna akan diarahkan kembali ke halaman login dengan menampilkan pesan dialog "Registration successful. Redirecting to login page". Jika tidak, pesan kesalahan akan ditampilkan.

```
import javax.swing.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;

public class UserApp {

    public static void main(String[] args) {
        showLoginFrame();
    }

    private static void showLoginFrame() {
        JFrame frame = new JFrame("Login");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 200);

        JPanel panel = new JPanel();
        frame.add(panel);
        placeLoginComponents(panel);

        frame.setVisible(true);
    }

    private static void showRegisterFrame() {
        JFrame frame = new JFrame("Register");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);

        JPanel panel = new JPanel();
        frame.add(panel);
        placeRegisterComponents(panel);

        frame.setVisible(true);
    }
}
```



```
private static void placeLoginComponents(JPanel panel) {
    panel.setLayout(null);

    JLabel userLabel = new JLabel("Username:");
    userLabel.setBounds(10, 20, 80, 25);
    panel.add(userLabel);

    JTextField userText = new JTextField(20);
    userText.setBounds(100, 20, 165, 25);
    panel.add(userText);

    JLabel passwordLabel = new JLabel("Password:");
    passwordLabel.setBounds(10, 50, 80, 25);
    panel.add(passwordLabel);

    JPasswordField passwordText = new JPasswordField(20);
    passwordText.setBounds(100, 50, 165, 25);
    panel.add(passwordText);

    JButton loginButton = new JButton("Login");
    loginButton.setBounds(10, 100, 80, 25);
    panel.add(loginButton);

    JButton registerButton = new JButton("Register");
    registerButton.setBounds(180, 100, 100, 25);
    panel.add(registerButton);

    loginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String username = userText.getText();
            String password = new String(passwordText.getPassword());

            try {
                String url = "http://localhost:8000/api/pat/login";
                HttpURLConnection conn = (HttpURLConnection) new
URL(url).openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("User-Agent",
"Chrome/51.0.2704.103");
```

```

        conn.setRequestProperty("Content-Type",
"application/json");

        String newMenu = "{\"username\": \"" + username + "\",
\"pass\": \"" + password + "\"}";

        conn.setDoOutput(true);
        OutputStream os = conn.getOutputStream();
        os.write(newMenu.getBytes());
        os.flush();
        os.close();

        int responseCode = conn.getResponseCode();

        if (responseCode == 200) {
            JOptionPane.showMessageDialog(panel, "Login
successful");

            JFrame frame = (JFrame)
SwingUtilities.getWindowAncestor(panel);
            frame.dispose();
            new Main();
        } else {
            JOptionPane.showMessageDialog(panel, "Invalid
username or password");
        }

        BufferedReader in = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
        String inputLine;
        StringBuffer response = new StringBuffer();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        }
        in.close();
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(panel, "Error connecting
to server: " + ex.getMessage());
    }

```

```

    }
    });

    registerButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JFrame frame = (JFrame)
SwingUtilities.getWindowAncestor(panel);
            frame.dispose();
            showRegisterFrame();
        }
    });
}

private static void placeRegisterComponents(JPanel panel) {
    panel.setLayout(null);

    JLabel userLabel = new JLabel("Username:");
    userLabel.setBounds(10, 20, 80, 25);
    panel.add(userLabel);

    JTextField userText = new JTextField(20);
    userText.setBounds(100, 20, 165, 25);
    panel.add(userText);

    JLabel nameLabel = new JLabel("Nama:");
    nameLabel.setBounds(10, 50, 80, 25);
    panel.add(nameLabel);

    JTextField nameText = new JTextField(20);
    nameText.setBounds(100, 50, 165, 25);
    panel.add(nameText);

    JLabel emailLabel = new JLabel("Email:");
    emailLabel.setBounds(10, 80, 80, 25);
    panel.add(emailLabel);

    JTextField emailText = new JTextField(20);
    emailText.setBounds(100, 80, 165, 25);
    panel.add(emailText);
}

```

```

JLabel phoneLabel = new JLabel("Telepon:");
phoneLabel.setBounds(10, 110, 80, 25);
panel.add(phoneLabel);

JTextField phoneText = new JTextField(20);
phoneText.setBounds(100, 110, 165, 25);
panel.add(phoneText);

JLabel passwordLabel = new JLabel("Password:");
passwordLabel.setBounds(10, 140, 80, 25);
panel.add(passwordLabel);

JPasswordField passwordText = new JPasswordField(20);
passwordText.setBounds(100, 140, 165, 25);
panel.add(passwordText);

JButton registerButton = new JButton("Register");
registerButton.setBounds(10, 200, 100, 25);
panel.add(registerButton);

JButton loginButton = new JButton("Login");
loginButton.setBounds(180, 200, 80, 25);
panel.add(loginButton);

registerButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String username = userText.getText();
        String password = new String(passwordText.getPassword());
        String nama = nameText.getText();
        String email = emailText.getText();
        String telepon = phoneText.getText();

        if (username.isEmpty() || password.isEmpty() ||
nama.isEmpty() || email.isEmpty() || telepon.isEmpty()) {
            JOptionPane.showMessageDialog(panel, "Please fill in
all fields.");
            return;
        }

        try {

```

```

URL url = new
URL("http://localhost:8000/api/pat/register/");
    HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("POST");
            conn.setRequestProperty("User-Agent",
"Chrome/51.0.2704.103");
            conn.setRequestProperty("Content-Type",
"application/json");

        String jsonString = String.format("{ \"username\":
\"%s\", \"pass\": \"%s\", \"nama\": \"%s\", \"email\": \"%s\",
\"telepon\": \"%s\" }",
            username, password, nama, email, telepon);

        conn.setDoOutput(true);
        OutputStream os = conn.getOutputStream();

os.write(jsonString.getBytes(StandardCharsets.UTF_8));
        os.flush();
        os.close();

        int responseCode = conn.getResponseCode();

        if (responseCode == 200) {
            JOptionPane.showMessageDialog(panel, "Registration
successful. Redirecting to login page.");
            JFrame frame = (JFrame)
SwingUtilities.getWindowAncestor(panel);
            frame.dispose();
            showLoginFrame();
        } else {
            JOptionPane.showMessageDialog(panel, "Registration
failed. Please try again.");
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(panel, "Error connecting
to server: " + ex.getMessage());
    }
}

```

```

        }
    });

    loginButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JFrame frame = (JFrame)
SwingUtilities.getWindowAncestor(panel);
            frame.dispose();
            showLoginFrame();
        }
    });
}
}
}

```

Main.java

Kode ini berfungsi untuk membuat sebuah jendela utama dengan beberapa tombol untuk menavigasi ke berbagai bagian aplikasi. Pertama, kelas `Main` meng-extend `JFrame`, yang berarti bahwa kelas ini adalah sebuah jendela aplikasi. Di dalam `Main`, beberapa pengaturan dasar jendela dilakukan, seperti judul jendela, ukuran, dan operasi penutupan.

Sebuah panel utama (`JPanel`) dibuat dan diatur menggunakan layout `GridLayout` dengan 6 baris dan 1 kolom. Lima tombol ditambahkan ke panel ini: "Create Event", "View Event", "View Order", "View Payment", dan "View Confirm". Tombol-tombol ini memiliki action listener yang akan mengeksekusi berbagai tindakan ketika ditekan. Misalnya, tombol "Create Event" akan membuat instance dari kelas `CreateEvent`, tombol "View Event" akan memanggil metode `main` dari kelas `ViewEvent`, tombol "View Order" akan memanggil metode `main` dari kelas `ViewOrder`, tombol "View Payment" akan memanggil metode `main` dari kelas `ViewPayment`, dan tombol "View Confirm" akan memanggil metode `main` dari kelas `ViewConfirm`.

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Main extends JFrame {

    public Main() {
        setTitle("Main Page");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

```

```
JPanel mainPanel = new JPanel();  
mainPanel.setLayout(new GridLayout(6, 1)); // Tambahkan satu baris  
lagi untuk tombol baru
```

```
JButton createEventButton = new JButton("Create Event");  
JButton viewEventButton = new JButton("View Event");  
JButton viewOrderButton = new JButton("View Order");  
JButton viewPaymentButton = new JButton("View Payment");  
// JButton createConfirmButton = new JButton("Create Confirm");  
JButton viewConfirmButton = new JButton("View Confirm"); //  
Tambahkan tombol View Confirm
```

```
createEventButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        new CreateEvent();  
    }  
});
```

```
viewEventButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        ViewEvent.main(new String[]{});  
    }  
});
```

```
viewOrderButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        ViewOrder.main(new String[]{});  
    }  
});
```

```
viewPaymentButton.addActionListener(new ActionListener() {  
    @Override  
    public void actionPerformed(ActionEvent e) {  
        ViewPayment.main(new String[]{});  
    }  
});
```

```

        // createConfirmButton.addActionListener(new ActionListener() {
        //     @Override
        //     public void actionPerformed(ActionEvent e) {
        //         CreateConfirm.main(new String[]{});
        //     }
        // });

        viewConfirmButton.addActionListener(new ActionListener() { //
Tambahkan listener untuk tombol View Confirm
        @Override
        public void actionPerformed(ActionEvent e) {
            ViewConfirm.main(new String[]{});
        }
    });

    mainPanel.add(createEventButton);
    mainPanel.add(viewEventButton);
    mainPanel.add(viewOrderButton);
    mainPanel.add(viewPaymentButton);
    // mainPanel.add(createConfirmButton);
    mainPanel.add(viewConfirmButton); // Tambahkan tombol ke panel

    add(mainPanel);
    setVisible(true);
}

public static void main(String[] args) {
    new Main();
}
}

```


CreateEvent.java

Kode ini mendefinisikan kelas `CreateEvent`, sebuah jendela aplikasi yang digunakan untuk membuat event baru dengan beberapa detail seperti nama konser, artis, deskripsi, kategori tiket, lokasi, tanggal, dan metode pembayaran. Metode `placeCreateEventComponents` digunakan untuk menambahkan berbagai komponen ke panel. Komponen yang ditambahkan meliputi label dan text field untuk nama konser, artis, deskripsi (dengan text area yang mendukung line wrap dan scroll), kategori tiket (dengan kemampuan menambah dan menghapus kategori), lokasi, dan tanggal. Selain itu, metode pembayaran juga ditambahkan dengan kemampuan menambah dan menghapus bank dan kode.

Tombol "Create Event" ditambahkan di bagian bawah panel. Ketika tombol ini ditekan, data dari semua text field dan text area dikumpulkan. Kategori tiket dan metode pembayaran dikonversi menjadi string JSON. Jika ada field yang kosong, akan muncul pesan peringatan. Jika semua field terisi, program akan mencoba mengirim data JSON ini ke server melalui HTTP POST request menggunakan `URLConnection`. Jika respons dari server adalah HTTP OK, akan muncul pesan bahwa event berhasil dibuat, jika tidak, akan muncul pesan kesalahan.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class CreateEvent extends JFrame {
    public CreateEvent() {
        setTitle("Create Event");
        setSize(800, 900);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JPanel createEventPanel = new JPanel(new GridLayout(0, 1));
        placeCreateEventComponents(createEventPanel);
        add(createEventPanel);

        setVisible(true);
    }
}
```

```

private void placeCreateEventComponents(JPanel panel) {
    panel.setLayout(new GridLayout(0, 1));

    panel.add(new JLabel("Nama Konser:"));
    JTextField namaKonserText = new JTextField(20);
    panel.add(namaKonserText);

    panel.add(new JLabel("Artis:"));
    JTextField artisText = new JTextField(20);
    panel.add(artisText);

    panel.add(new JLabel("Deskripsi:"));
    JTextArea deskripsiText = new JTextArea(2, 20);
    deskripsiText.setLineWrap(true);
    deskripsiText.setWrapStyleWord(true);
    panel.add(new JScrollPane(deskripsiText));

    // Kategori Tiket
    panel.add(new JLabel("Kategori Tiket:"));
    List<JPanel> kategoriPanels = new ArrayList<>();
    JPanel kategoriContainer = new JPanel();
    kategoriContainer.setLayout(new BoxLayout(kategoriContainer,
BoxLayout.Y_AXIS));
    panel.add(kategoriContainer);
    panel.add(new JScrollPane(kategoriContainer));
    JButton addKategoriButton = new JButton("Add Kategori Tiket");
    panel.add(addKategoriButton);

    addKategoriButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            JPanel kategoriPanel = new JPanel(new GridLayout(1, 5));
            kategoriPanel.add(new JLabel("Kategori:"));
            JTextField kategoriTiketText = new JTextField(10);
            kategoriPanel.add(kategoriTiketText);
            kategoriPanel.add(new JLabel("Harga:"));
            JTextField hargaText = new JTextField(10);
            kategoriPanel.add(hargaText);
            kategoriPanel.add(new JLabel("Jumlah:"));
            JTextField jumlahTiketText = new JTextField(10);
            kategoriPanel.add(jumlahTiketText);
        }
    });
}

```

```

        JButton removeButton = new JButton("Remove");
        kategoriPanel.add(removeButton);

        removeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                kategoriContainer.remove(kategoriPanel);
                kategoriPanels.remove(kategoriPanel);
                kategoriContainer.revalidate();
                kategoriContainer.repaint();
            }
        });

        kategoriPanels.add(kategoriPanel);
        kategoriContainer.add(kategoriPanel);
        kategoriContainer.revalidate();
        kategoriContainer.repaint();
    }
});

// Lokasi
panel.add(new JLabel("Lokasi:"));
JTextField lokasiText = new JTextField(20);
panel.add(lokasiText);

// Tanggal
panel.add(new JLabel("Tanggal:"));
JTextField tanggalText = new JTextField(20);
panel.add(tanggalText);

// Metode Pembayaran
panel.add(new JLabel("Metode Pembayaran:"));
List<JPanel> pembayaranPanels = new ArrayList<>();
JPanel pembayaranContainer = new JPanel();
pembayaranContainer.setLayout(new BoxLayout(pembayaranContainer,
BoxLayout.Y_AXIS));
panel.add(pembayaranContainer);
panel.add(new JScrollPane(pembayaranContainer));
JButton addPembayaranButton = new JButton("Add Metode
Pembayaran");

```

```

panel.add(addPembayaranButton);

addPembayaranButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JPanel pembayaranPanel = new JPanel(new GridLayout(1, 4));
        pembayaranPanel.add(new JLabel("Bank:"));
        JTextField bankText = new JTextField(10);
        pembayaranPanel.add(bankText);
        pembayaranPanel.add(new JLabel("Kode:"));
        JTextField kodeText = new JTextField(10);
        pembayaranPanel.add(kodeText);

        JButton removeButton = new JButton("Remove");
        pembayaranPanel.add(removeButton);

        removeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                pembayaranContainer.remove(pembayaranPanel);
                pembayaranPanels.remove(pembayaranPanel);
                pembayaranContainer.revalidate();
                pembayaranContainer.repaint();
            }
        });

        pembayaranPanels.add(pembayaranPanel);
        pembayaranContainer.add(pembayaranPanel);
        pembayaranContainer.revalidate();
        pembayaranContainer.repaint();
    }
});

// Submit Button
JButton submitButton = new JButton("Create Event");
panel.add(submitButton);

submitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String namaKonser = namaKonserText.getText();
        String artis = artisText.getText();
        String deskripsi = deskripsiText.getText();
    }
});

```

```

        // Convert kategoriPanels to JSON
        List<String> kategoriList = new ArrayList<>();
        int totalJumlahTiket = 0;
        for (JPanel kategoriPanel : kategoriPanels) {
            Component[] components =
kategoriPanel.getComponents();
            String kategori = ((JTextField)
components[1]).getText();
            String harga = ((JTextField) components[3]).getText();
            String jumlah = ((JTextField)
components[5]).getText();
            totalJumlahTiket += Integer.parseInt(jumlah);
            kategoriList.add(String.format("{\"kategori\":\"%s\",
\"harga\":\"%s\", \"jumlah\":\"%s\"}", kategori, harga, jumlah));
        }
        String kategori = "[" + String.join(", ", kategoriList) +
"]";

        String lokasi = lokasiText.getText();
        String tanggal = tanggalText.getText();

        // Convert pembayaranPanels to JSON
        List<String> pembayaranList = new ArrayList<>();
        for (JPanel pembayaranPanel : pembayaranPanels) {
            Component[] components =
pembayaranPanel.getComponents();
            String bank = ((JTextField) components[1]).getText();
            String kode = ((JTextField) components[3]).getText();
            pembayaranList.add(String.format("{\"bank\":\"%s\",
\"kode\":\"%s\"}", bank, kode));
        }
        String pembayaran = "[" + String.join(", ",
pembayaranList) + "]";

        if (namaKonser.isEmpty() || deskripsi.isEmpty() ||
kategori.isEmpty() || lokasi.isEmpty() || tanggal.isEmpty() ||
pembayaran.isEmpty()) {
            JOptionPane.showMessageDialog(panel, "Please fill in
all fields.");

```

```

        return;
    }

    try {
        URL url = new
URL("http://localhost:8000/api/pat/event");
        HttpURLConnection conn = (HttpURLConnection)
url.openConnection();
        conn.setRequestMethod("POST");
        conn.setRequestProperty("Content-Type",
"application/json");

        String jsonString = String.format(
            "{\"nama_konser\": \"%s\", \"artis\": \"%s\",
\"deskripsi\": \"%s\", \"kategori_tiket\": %s, \" +
            \"jumlah_tiket\": \"%d\", \"lokasi\":
\"%s\", \"tanggal\": \"%s\", \"metode_pembayaran\": %s}\",
            namaKonser, artis, deskripsi, kategori,
totalJumlahTiket, lokasi, tanggal, pembayaran);

        conn.setDoOutput(true);
        OutputStream os = conn.getOutputStream();

os.write(jsonString.getBytes(StandardCharsets.UTF_8));
        os.flush();
        os.close();

        int responseCode = conn.getResponseCode();
        if (responseCode == HttpURLConnection.HTTP_OK) {
            JOptionPane.showMessageDialog(panel, "Event
created successfully.");
        } else {
            JOptionPane.showMessageDialog(panel, "Failed to
create event. Please try again.");
        }
    } catch (Exception ex) {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(panel, "Error: " +
ex.getMessage());
    }
}

```

```

        }

    });

}

public static void main(String[] args) {
    new CreateEvent();
}
}

```

ViewEvent.java

Kelas `ViewEvent` berfungsi untuk menampilkan, memperbarui, dan menghapus detail konser. Dengan menggunakan tabel, data konser seperti ID, nama konser, artis, jumlah tiket, lokasi, tanggal, kategori tiket, harga, jumlah, bank, dan kode ditampilkan. Tombol "Update" dan "Delete" memungkinkan pengguna memperbarui atau menghapus entri konser. Fungsi `getJSONData` mengambil data JSON dari API, dan data ditampilkan dalam tabel. Listener tombol menangani logika untuk memperbarui dan menghapus data, serta memastikan tabel diperbarui setelah operasi. GUI ini membantu memanipulasi data konser dengan mudah melalui antarmuka pengguna grafis.

```

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableCellRenderer;
import javax.swing.table.TableCellEditor;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;

public class ViewEvent {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ViewEvent::createAndShowGUI);
    }
}

```

```

private static void createAndShowGUI() {
    JFrame frame = new JFrame("Detail Konser");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 600);

    JPanel mainPanel = new JPanel(new BorderLayout());

    JPanel viewEventPanel = new JPanel(new BorderLayout());
    placeViewEventComponents(viewEventPanel);

    mainPanel.add(viewEventPanel, BorderLayout.CENTER);

    frame.add(mainPanel);
    frame.setVisible(true);
}
//.....
}

```

ViewOrder.java

Kelas `ViewOrder` berfungsi untuk menampilkan detail pesanan dalam tabel. Program dimulai dengan metode `main`, yang memanggil `createAndShowGUI` untuk mengatur antarmuka pengguna. Metode `placeViewOrderComponents` bertanggung jawab untuk mengambil data JSON dari API menggunakan URL "http://localhost:8000/api/pat/order". Jika data JSON yang diterima memiliki status 200 dan berisi respons yang diharapkan, maka detail pesanan diekstrak dan ditampilkan dalam tabel. Tabel ini menampilkan kolom seperti "Order ID", "Event ID", "Kategori Tiket", dan "Total Harga". Data kategori tiket diproses lebih lanjut dari JSON string menjadi array JSON untuk ditampilkan dengan format yang mudah dibaca.

```

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;

public class ViewOrder {

```



```

public static void main(String[] args) {
    SwingUtilities.invokeLater(ViewOrder::createAndShowGUI);
}

private static void createAndShowGUI() {
    JFrame frame = new JFrame("Order");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800, 600);

    JPanel mainPanel = new JPanel(new BorderLayout());

    JPanel viewOrderPanel = new JPanel(new BorderLayout());
    placeViewOrderComponents(viewOrderPanel);

    mainPanel.add(viewOrderPanel, BorderLayout.CENTER);

    frame.add(mainPanel);
    frame.setVisible(true);
}

private static void placeViewOrderComponents(JPanel panel) {
    panel.removeAll();

    String url = "http://localhost:8000/api/pat/order";
    try {
        String jsonData = getJSONData(url);
        System.out.println("Response JSON: " + jsonData);
        JSONObject jsonObject = new JSONObject(jsonData);

        if (jsonObject.has("status") && jsonObject.getInt("status") ==
200 &&
        jsonObject.has("response")) {
            JSONArray orders = jsonObject.getJSONArray("response");

            String[] columnNames = {"Order ID", "Event ID", "Kategori
Tiket", "Total Harga"};

            DefaultTableModel model = new
DefaultTableModel(columnNames, 0);

```

```

        for (int i = 0; i < orders.length(); i++) {
            JSONObject order = orders.getJSONObject(i);

            int orderId = order.getInt("id");
            int eventId = order.getInt("eventId");

            String kategoriTiketStr =
order.getString("kategori_tiket");
            double totalHarga = order.getDouble("total_harga");

            // Parse kategori_tiket from JSON string
            JSONArray kategoriTiketArray = new
JSONArray(kategoriTiketStr);

            StringBuilder kategoriTiketBuilder = new
StringBuilder();

            for (int j = 0; j < kategoriTiketArray.length(); j++)
{
                JSONObject kategori =
kategoriTiketArray.getJSONObject(j);
                kategoriTiketBuilder.append("Kategori:
").append(kategori.getString("kategori"));
                if (j < kategoriTiketArray.length() - 1) {
                    kategoriTiketBuilder.append("; ");
                }
            }

            String kategoriTiket =
kategoriTiketBuilder.toString();

            model.addRow(new Object[]{orderId, eventId,
kategoriTiket, totalHarga});
        }

        JTable table = new JTable(model);
        JScrollPane scrollPane = new JScrollPane(table);

        panel.add(scrollPane, BorderLayout.CENTER);

    } else {
        JOptionPane.showMessageDialog(panel, "Error: Data format
is not as expected.");
    }
}

```

```

        } catch (Exception e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(panel, "Error: Failed to fetch
order details.");
        }

        panel.revalidate();
        panel.repaint();
    }

    private static String getJSONData(String urlString) throws Exception {
        URL url = new URL(urlString);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("Accept", "application/json");

        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("Failed : HTTP error code : " +
conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));

        StringBuilder sb = new StringBuilder();
        String output;
        while ((output = br.readLine()) != null) {
            sb.append(output);
        }
        conn.disconnect();
        return sb.toString();
    }
}

```

ViewPayment.java

Kelas `ViewPayment` berfungsi untuk menampilkan detail pembayaran dalam tabel. Program dimulai dengan metode `main`, yang memanggil `createAndShowGUI` untuk mengatur antarmuka pengguna. Metode `placeViewPaymentComponents` bertanggung jawab untuk mengambil data JSON dari API menggunakan URL "http://localhost:8000/api/pat/payment". Jika data JSON yang diterima memiliki status 200 dan berisi respons yang diharapkan, maka detail pembayaran diekstrak dan ditampilkan dalam tabel. Tabel ini menampilkan kolom seperti "Payment ID", "Order ID", "Metode Pembayaran", "Virtual Account", dan "Status".

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;

public class ViewPayment {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ViewPayment::createAndShowGUI);
    }

    private static void createAndShowGUI() {
        JFrame frame = new JFrame("Payment");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        JPanel mainPanel = new JPanel(new BorderLayout());

        JPanel viewPaymentPanel = new JPanel(new BorderLayout());
        placeViewPaymentComponents(viewPaymentPanel);

        mainPanel.add(viewPaymentPanel, BorderLayout.CENTER);

        frame.add(mainPanel);
        frame.setVisible(true);
    }
}
```

```

private static void placeViewPaymentComponents(JPanel panel) {
    panel.removeAll();

    String url = "http://localhost:8000/api/pat/payment";
    try {
        String jsonData = getJSONData(url);
        System.out.println("Response JSON: " + jsonData); // Cetak
JSON untuk debugging
        JSONObject jsonObject = new JSONObject(jsonData);

        // Pastikan kunci yang diharapkan ada dalam JSON sebelum
mengaksesnya
        if (jsonObject.has("status") && jsonObject.getInt("status") ==
200 &&
            jsonObject.has("response")) {
            JSONArray payments = jsonObject.getJSONArray("response");

            String[] columnNames = {"Payment ID", "Order ID", "Metode
Pembayaran", "Virtual Account", "Status"};
            DefaultTableModel model = new
DefaultTableModel(columnNames, 0);

            for (int i = 0; i < payments.length(); i++) {
                JSONObject payment = payments.getJSONObject(i);

                int paymentId = payment.getInt("id");
                int orderId = payment.getInt("orderId");
                String metodePembayaran =
payment.getString("metode_pembayaran");
                String virtualAccount =
payment.getString("virtual_account");
                String status = payment.getString("status");

                model.addRow(new Object[]{paymentId, orderId,
metodePembayaran, virtualAccount, status});
            }

            JTable table = new JTable(model);
            JScrollPane scrollPane = new JScrollPane(table);

```

```

        panel.add(scrollPane, BorderLayout.CENTER);

        } else {
            JOptionPane.showMessageDialog(panel, "Error: Data format
is not as expected.");
        }

    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(panel, "Error: Failed to fetch
payment details.");
    }

    panel.revalidate();
    panel.repaint();
}

private static String getJSONData(String urlString) throws Exception {
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Accept", "application/json");

    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " +
conn.getResponseCode());
    }

    BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));
    StringBuilder sb = new StringBuilder();
    String output;
    while ((output = br.readLine()) != null) {
        sb.append(output);
    }
    conn.disconnect();
    return sb.toString();
}
}

```

ViewConfirm.java

Kelas `ViewConfirm` berfungsi untuk menampilkan konfirmasi pesanan dalam tabel. Program dimulai dengan metode `main`, yang memanggil `createAndShowGUI` untuk mengatur antarmuka pengguna. Metode `placeViewConfirmComponents` bertanggung jawab untuk mengambil data JSON dari API menggunakan URL "http://localhost:8000/api/pat/confirm". Setelah menerima data JSON, program mencetak respons JSON tersebut untuk debugging. Kemudian, program memeriksa apakah data JSON mengandung kunci "status" dengan nilai 200 dan memiliki kunci "response". Jika ya, data pesanan konfirmasi diekstrak dan ditampilkan dalam tabel. Tabel ini menampilkan kolom seperti "Order ID", "Status", dan "Kode Booking".

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONArray;
import org.json.JSONObject;

public class ViewConfirm {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(ViewConfirm::createAndShowGUI);
    }

    private static void createAndShowGUI() {
        JFrame frame = new JFrame("View Confirm");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(800, 600);

        JPanel mainPanel = new JPanel(new BorderLayout());

        JPanel viewConfirmPanel = new JPanel(new BorderLayout());
        placeViewConfirmComponents(viewConfirmPanel);

        mainPanel.add(viewConfirmPanel, BorderLayout.CENTER);

        frame.add(mainPanel);
        frame.setVisible(true);
    }
}
```

```

private static void placeViewConfirmComponents(JPanel panel) {
    panel.removeAll();

    String url = "http://localhost:8000/api/pat/confirm";
    try {
        String jsonData = getJSONData(url);
        System.out.println("Response JSON: " + jsonData); // Cetak
JSON untuk debugging
        JSONObject jsonObject = new JSONObject(jsonData);

        // Pastikan kunci yang diharapkan ada dalam JSON sebelum
mengaksesnya
        if (jsonObject.has("status") && jsonObject.getInt("status") ==
200 &&
            jsonObject.has("response")) {
            JSONArray confirms = jsonObject.getJSONArray("response");

            String[] columnNames = {"Order ID", "Status", "Kode
Booking"};

            DefaultTableModel model = new
DefaultTableModel(columnNames, 0);

            for (int i = 0; i < confirms.length(); i++) {
                JSONObject confirm = confirms.getJSONObject(i);

                int orderId = confirm.getInt("orderId");
                String status = confirm.getString("status");
                String kodeBooking =
confirm.getString("kode_booking");

                model.addRow(new Object[]{orderId, status,
kodeBooking});
            }

            JTable table = new JTable(model);
            JScrollPane scrollPane = new JScrollPane(table);

            panel.add(scrollPane, BorderLayout.CENTER);

```



```

        } else {
            JOptionPane.showMessageDialog(panel, "Error: Data format
is not as expected.");
        }

    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(panel, "Error: Failed to fetch
confirm details.");
    }

    panel.revalidate();
    panel.repaint();
}

private static String getJSONData(String urlString) throws Exception {
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    conn.setRequestProperty("Accept", "application/json");

    if (conn.getResponseCode() != 200) {
        throw new RuntimeException("Failed : HTTP error code : " +
conn.getResponseCode());
    }

    BufferedReader br = new BufferedReader(new
InputStreamReader((conn.getInputStream())));
    StringBuilder sb = new StringBuilder();
    String output;
    while ((output = br.readLine()) != null) {
        sb.append(output);
    }
    conn.disconnect();
    return sb.toString();
}
}

```