

**LAPORAN PRAKTIKUM PEMOGRAMAN BERORIENTASI OBJEK  
(PBO)**



OLEH :

FAYI AMATULLAH AZHARA  
(2311537001)

DOSEN PENGUMPUL : Nurfiah, S.ST, M.Kom.

**FAKULTAS TEKNOLOGI INFORMASI  
DEPARTEMEN INFORMATIKA  
UNIVERSITAS ANDALAS**

## Membuat Model Order

```
package model;

public class Order {
    String id_order, nama, tanggal, tanggal_kembali, status, pembayaran, status_bayar;
    int total_harga, id_costumer;
    public String getNama() {
        return nama;
    }
    public void setNama(String nama) {
        this.nama = nama;
    }
    public String getId_order() {
        return id_order;
    }
    public void setId_order(String id_order) {
        this.id_order = id_order;
    }
    public String getTanggal() {
        return tanggal;
    }
    public void setTanggal(String tanggal) {
        this.tanggal = tanggal;
    }
    public String getTanggal_kembali() {
        return tanggal_kembali;
    }
    public void setTanggal_kembali(String tanggal_kembali) {
        this.tanggal_kembali = tanggal_kembali;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public String getPembayaran() {
        return pembayaran;
    }
    public void setPembayaran(String pembayaran) {
        this.pembayaran = pembayaran;
    }
    public String getStatus_bayar() {
        return status_bayar;
    }
    public void setStatus_bayar(String status_bayar) {
        this.status_bayar = status_bayar;
    }
    public int getTotal_harga() {
        return total_harga;
    }
    public void setTotal_harga(int total_harga) {
        this.total_harga = total_harga;
    }
    public int getId_costumer() {
        return id_costumer;
    }
    public void setId_costumer(int id_costumer) {
        this.id_costumer = id_costumer;
    }
}
```

Dalam model ini, terdapat sembilan entitas, yaitu id\_order, nama, tanggal, tanggal\_kembali, status, pembayaran, status\_pembayaran, total\_harga, dan id\_costumer. Setiap entitas dibuatkan metode setter dan getter untuk mempermudah manipulasi data.

## Membuat Tabel Order

```
package table;

import java.util.List;
import javax.swing.table.AbstractTableModel;
import model.Costumer;
import model.Order;

public class TableOrder extends AbstractTableModel {
    List<Order> ls;
    private String[] columnNames = {"ID", "Nama", "Tanggal", "Tanggal Pengembalian", "Status",
        "Total Harga", "Pembayaran", "Status Pembayaran"};
    public TableOrder(List<Order> ls) {
        this.ls = ls;
    }

    @Override
    public int getRowCount() {
        return ls.size();
    }

    @Override
    public int getColumnCount() {
        return 8;
    }

    @Override
    public String getColumnName(int column) {
        return columnNames[column];
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        switch (columnIndex) {
            case 0:
                return ls.get(rowIndex).getId_order();
            case 1:
                return ls.get(rowIndex).getNama();
            case 2:
                return ls.get(rowIndex).getTanggal();
            case 3:
                return ls.get(rowIndex).getTanggal_kembali();
            case 4:
                return ls.get(rowIndex).getStatus();
            case 5:
                return ls.get(rowIndex).getTotal_harga();
            case 6:
                return ls.get(rowIndex).getPembayaran();
            case 7:
                return ls.get(rowIndex).getStatus_bayar();
            default:
                return null;
        }
    }

    public Order getCostumerAt(int rowIndex) {
        return ls.get(rowIndex);
    }
}
```

Kelas ini dilengkapi dengan berbagai metode, antara lain:

Konstruktor TableOrder(List<Order> ls): Menginisialisasi tabel dengan daftar pesanan yang diberikan.

getRowCount(): Mengembalikan jumlah baris sesuai jumlah data dalam daftar pesanan.

getColumnCount(): Mengembalikan jumlah kolom tabel, yaitu delapan, sesuai elemen dalam array nama kolom.

getColumnName(int column): Mengembalikan nama kolom berdasarkan indeks yang diberikan.

getValueAt(int rowIndex, int columnIndex): Memberikan nilai sesuai posisi baris dan kolom tertentu.

getCostumerAt(int rowIndex): Mengembalikan objek pesanan tertentu untuk memudahkan pengolahan lebih lanjut.

Membuat OrderDAO dan OrderRepo

```
package DAO;

import java.util.List;

import model.Costumer;
import model.Order;

public interface OrderDAO {
    public List<Order> show();
    public void delete (String id);
    void save(Order order);
}
```

```
public class OrderRepo implements OrderDAO{
    private Connection connection;
    final String select = "SELECT id_order, cust.id AS id_costumer, cust.nama AS nama, tanggal, "
        + "tanggal_kembali, status, total_harga, pembayaran, status_bayar"
        + "FROM tabel_order AS tabel JOIN costumer AS cust ON (tabel.id_customer = cust.id);";
    final String delete = "DELETE FROM tabel_order WHERE id_order = ?;";

    public OrderRepo() {
        connection = Database.getConnection();
    }
}
```

Query JOIN: Digunakan untuk menggabungkan data dari tabel order dan customer, menghasilkan informasi nama pelanggan di tabel pesanan.

```
public List<Order> show(){
    List<Order> ls = null;
    try {
        ls = new ArrayList<Order>();
        Statement st = connection.createStatement();
        ResultSet rs = st.executeQuery(select);
        while(rs.next()) {
            Order order = new Order();
            order.setId_order(rs.getString("id_order"));
            order.setId_costumer(rs.getInt("id_costumer"));
            order.setNama(rs.getString("nama"));
            order.setTanggal(rs.getString("tanggal"));
            order.setTanggal_kembali(rs.getString("tanggal_kembali"));
            order.setStatus(rs.getString("status"));
            order.setTotal_harga(rs.getInt("total_harga"));
            order.setPembayaran(rs.getString("pembayaran"));
            order.setStatus_bayar(rs.getString("status_bayar"));
            ls.add(order);
        }
    }catch(SQLException e) {
        Logger.getLogger(CostumerDAO.class.getName()).log(Level.SEVERE, null, e);
    }
    return ls;
}
```

Metode show(): Mengambil semua data pesanan dari database, mengubahnya menjadi daftar objek Order. Data ini kemudian diproses dan ditambahkan ke daftar menggunakan loop.

```
public void delete(String id) {
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(delete);
        st.setString(1, id);
        st.executeUpdate();
    }catch(SQLException e) {
        e.printStackTrace();
    }finally {
        try {
            st.close();
        }catch(SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Metode delete(String id): Menghapus pesanan dari database berdasarkan id\_order tertentu, dengan menjaga keamanan dari SQL Injection menggunakan PreparedStatement.

```

@Override
public void save(Order order) {
    String save = "INSERT INTO tabel_order (id_order, id_customer, tanggal, tanggal_kembali, status, "
        + "total_harga, pembayaran, status_bayar) "
        + "VALUES (?, (SELECT id FROM costumer WHERE nama = ? LIMIT 1), ?, ?, ?, ?, ?, ?) "
        + "ON DUPLICATE KEY UPDATE "
        + "tanggal = VALUES(tanggal), "
        + "tanggal_kembali = VALUES(tanggal_kembali), "
        + "status = VALUES(status), "
        + "total_harga = VALUES(total_harga), "
        + "pembayaran = VALUES(pembayaran), "
        + "status_bayar = VALUES(status_bayar);";
    PreparedStatement st = null;
    try {
        st = connection.prepareStatement(save);
        st.setString(1, order.getId_order());
        st.setString(2, order.getNama()); // Assuming you have a method to get the customer's name
        st.setString(3, order.getTanggal());
        st.setString(4, order.getTanggal_kembali());
        st.setString(5, order.getStatus());
        st.setInt(6, order.getTotal_harga()); // Assuming total_harga is an int
        st.setString(7, order.getPembayaran());
        st.setString(8, order.getStatus_bayar());
        st.executeUpdate();
    } catch(SQLException e) {
        e.printStackTrace();
    } finally {
        try {
            st.close();
        } catch(SQLException e) {
            e.printStackTrace();
        }
    }
}

}

```

Metode save(Order order): Menyimpan data pesanan baru ke dalam database menggunakan perintah INSERT. Data diambil dari atribut objek Order yang dikirimkan.

## Update Frame Order Detail

The screenshot displays a Java Swing application window with a light gray border. On the left side, there is a vertical panel containing several input fields and dropdown menus:

- Order ID:** A text input field.
- Pelanggan:** A text input field.
- Tanggal:** A date input field with a calendar icon.
- Tanggal Pengambilan:** A date input field with a calendar icon.
- Status:** A dropdown menu currently set to "Progress".
- Total:** A text input field showing "Rp. 0".
- Pembayaran:** A dropdown menu currently set to "Tunai".
- Status Pembayaran:** A dropdown menu currently set to "Lunas".

Below these fields are two buttons: "Simpan" and "Batal".

On the right side, there are two main sections:

- Layanan:** A table showing service details. The columns are ID, Jenis, Status, and Harga. The data is as follows:

ID	Jenis	Status	Harga
1	Cuci	Aktif	4000.0
2	Setrika	Aktif	3000.0
3	Cuci+Setrika	Aktif	5000.0
- Harga/Kg:** A section with three input fields: "Jumlah" (Quantity), "Total" (Total Price), and a dropdown menu. Below this are four buttons: "Simpan", "Ubah", "Hapus", and "Batal".

At the bottom of the right panel, there is a table with the following structure and data:

id_order_detail	id_order	id_layanan	jumlah	total

Implementasi antarmuka berbasis Java Swing ini dirancang untuk mengelola detail pesanan, dengan komponen GUI seperti JPanel, JTable, dan lainnya. Fungsi utama mencakup pemutuan data layanan serta detail pesanan, perhitungan harga otomatis, dan kemampuan CRUD. Fitur seperti JDateChooser mempermudah pengguna dalam memilih tanggal.

Kode:

```
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    OrderDetailFrame frame = new OrderDetailFrame();
                    frame.setVisible(true);
                    frame.loadTable();
                    frame.loadTableOrderDetail();

                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public void reset() {
        txtHargaKg.setText("");
        txtJumlah.setText("");
        txtTotal.setText("");
    }
}
```

```
public void resetOrder() {
    txtOrderID.setText("");
}

public void setTanggal(Date date) {
    tanggal.setDate(date);
}

public void setTanggal_kembali(Date date) {
    tanggal_kembali.setDate(date);
}

public void setStatus(String status) {
    boxStatus.setSelectedItem(status);
}

public void setLblTotalHargaShow(String total) {
    lblTotalHargaShow.setText("Rp. " + total);
}

public void setBoxPembayaran(String pembayaran) {
    boxPembayaran.setSelectedItem(pembayaran);
}

public void setBoxPembayaran_1(String pembayaran) {
    boxPembayaran_1.setSelectedItem(pembayaran);
}

public static void setId_order(String orderId) {
    id_order = orderId;
}

public void loadTable() {
    ls = srv.show();
    TableService ts = new TableService(ls);
    tableService.setModel(ts);
    tableService.getTableHeader().setVisible(true);
}
```

```
public void loadTableOrderDetail() {
    ls_ord = ord.showById(id_order);
    TableOrderDetail tod = new TableOrderDetail(ls_ord);
    tableOrderDetail.setModel(tod);
    tableOrderDetail.getTableHeader().setVisible(true);
    updateTotalHarga();
}

public void updateTotalHarga() {
    int totalHarga = ord.total(id_order);
    total_harga = totalHarga;
}

public void setCostumer(Costumer costumer) {
    txtCostumer.setText(costumer.getNama());
}

public void setOrderID(String id_order) {
    txtOrderID.setText(id_order);
}

public JTextField getTxtOrderId() {
    return txtOrderID;
}

public void setTxtCostumer(String cust) {
    txtCostumer.setText(cust);
}

public String setTanggal(String tanggal) {
    return tgl = tanggal;
}

public void resetAll() {
    id_ord = null;
    id_order = null;
    total_harga = 0;
    tgl = null;
    tgl_kbl = null;
    setTxtCostumer("");
    setTanggal("");
    setTanggal_kembali(null);
    setLblTotalHargaShow("");
}
```

```
txtJumlah = new JTextField();
txtJumlah.setBounds(0, 80, 170, 19);
panelCRUD.add(txtJumlah);
txtJumlah.addKeyListener(new KeyAdapter() {
    @Override
    public void keyReleased(KeyEvent e) {
        String jumlahText = txtJumlah.getText();
        String hargaKgText = txtHargaKg.getText();
        if (!jumlahText.isEmpty() && !hargaKgText.isEmpty()) {
            try {
                double jumlah = Double.parseDouble(jumlahText);
                double hargaKg = Double.parseDouble(hargaKgText);
                double totalharga = jumlah * hargaKg;
                System.out.println(totalharga);
                txtTotal.setText(String.valueOf(totalharga));
            } catch (NumberFormatException ex) {
                txtTotal.setText("0");
            }
        } else {
            txtTotal.setText("0");
        }
    }
});
txtJumlah.setColumns(10);
```

```
lblTotalHargaShow = new JLabel("Rp. " + ord.total(txtOrderID.getText()));
lblTotalHargaShow.setBounds(0, 299, 170, 25);
panelAkumulasi.add(lblTotalHargaShow);
lblTotalHargaShow.setFont(new Font("Tahoma", Font.PLAIN, 16));
lblTotalHargaShow.setText("Rp. " + total_harga);

tanggal = new JDateChooser();
tanggal.setBounds(0, 137, 170, 19);
panelAkumulasi.add(tanggal);

tanggal_kembali = new JDateChooser();
tanggal_kembali.setBounds(0, 193, 170, 19);
panelAkumulasi.add(tanggal_kembali);
```

```
tableService = new JTable();
tableService.getTableHeader().setVisible(true);
tableService.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int selectedRow = tableService.getSelectedRow();

        if (selectedRow != -1) {
            String idLayanan = tableService.getValueAt(selectedRow, 0).toString();
            txtHargaKg.setText(tableService.getValueAt(selectedRow, 3).toString());
            lblId_layanan.setText(idLayanan);
            if (ord.exists(id_order, idLayanan)) {
                JOptionPane.showMessageDialog(null, "Service sudah ada di order detail.");
            } else {
            }
        }
    }
});
scrollPane.setViewportView(tableService);
```

```

 JButton btnSimpan = new JButton("Simpan");
 btnSimpan.setBounds(0, 113, 85, 34);
 panelCRUD.add(btnSimpan);
 btnSimpan.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
 if(id_order != null) {
 if (id_order != null) {
 String idLayanan = lblId_layanan.getText();
 if (ord.exists(id_order, idLayanan)) {
 JOptionPane.showMessageDialog(null, "Layanan ini sudah ada dalam detail order.", "Error", JOptionPane.ERROR_MESSAGE);
 } else {
 String jumlahText = txtJumlah.getText();
 String totalText = txtTotal.getText();

 if (jumlahText.isEmpty()) {
 JOptionPane.showMessageDialog(null, "Jumlah tidak boleh kosong.", "Error", JOptionPane.ERROR_MESSAGE);
 return;
 }

 if (totalText.isEmpty()) {
 JOptionPane.showMessageDialog(null, "Total tidak boleh kosong.", "Error", JOptionPane.ERROR_MESSAGE);
 return;
 }
 try {
 double jumlah = Double.parseDouble(jumlahText);
 double total = Double.parseDouble(totalText);
 OrderDetail orderdetail = new OrderDetail();
 orderdetail.setId_order(id_order);
 orderdetail.setid_layanan(idLayanan);
 orderdetail.setJumlah(jumlah);
 orderdetail.setTotal(total);
 ord.save(orderdetail);
 reset();
 loadTableOrderDetail();
 lblTotalHargaShow.setText("Rp. " + total_harga);
 } catch (NumberFormatException ex) {
 JOptionPane.showMessageDialog(null, "Jumlah dan total harus berupa angka.", "Error", JOptionPane.ERROR_MESSAGE);
 }
 }
 } else {
 JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan disimpan");
 }
 }
 });

```

```

 JButton btnUbah = new JButton("Ubah");
 btnUbah.setBounds(97, 113, 85, 34);
 panelCRUD.add(btnUbah);
 btnUbah.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
 if(id_ord != null) {
 OrderDetail orderdetail = new OrderDetail();
 orderdetail.setId_order(id_order);
 orderdetail.setId_layanan(lblId_layanan.getText());
 orderdetail.setJumlah(Double.parseDouble(txtJumlah.getText()));
 orderdetail.setTotal(Double.parseDouble(txtTotal.getText()));
 orderdetail.setId_order_detail(id_ord);
 ord.update(orderdetail);
 reset();
 loadTableOrderDetail();
 lblTotalHargaShow.setText("Rp. " + total_harga);
 }
 else {
 JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan diedit");
 }
 }
 });
 btnUbah.setFont(new Font("Tahoma", Font.PLAIN, 16));

```

```

 JButton btnHapus = new JButton("Hapus");
 btnHapus.setBounds(193, 113, 85, 34);
 panelCRUD.add(btnHapus);
 btnHapus.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
 if(id_ord != null) {
 ord.delete(id_ord);
 reset();
 loadTableOrderDetail();
 lblTotalHargaShow.setText("Rp. " + total_harga);

 } else {
 JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
 }
 }
 });
 btnHapus.setFont(new Font("Tahoma", Font.PLAIN, 16));

```

```

 JButton btnBatal = new JButton("Batal");
 btnBatal.setBounds(288, 113, 85, 34);
 panelCRUD.add(btnBatal);
 btnBatal.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
 reset();
 }
 });
 btnBatal.setFont(new Font("Tahoma", Font.PLAIN, 16));

```

```
tableOrderDetail = new JTable();
tableOrderDetail.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        id_ord = tableOrderDetail.getValueAt(tableOrderDetail.getSelectedRow(), 0).toString();
        int selectedRow = tableOrderDetail.getSelectedRow();
        if (selectedRow != -1) {
            String idLayanan = tableOrderDetail.getValueAt(selectedRow, 2).toString();
            for (Service service : ls) {
                if (service.getId().equals(idLayanan)) {
                    txtHargaKg.setText(String.valueOf(service.getHarga()));
                    break;
                }
            }
        }
    }
});
scrollPane_1.setViewportView(tableOrderDetail);
```

```
JButton btnSimpanOrder = new JButton("Simpan");
btnSimpanOrder.setBounds(0, 445, 85, 34);
panelAkumulasi.add(btnSimpanOrder);
btnSimpanOrder.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id_order != null) {
            Order order = new Order();
            order.setId_order(id_order);
            order.setNama(txtCostumer.getText());
            order.setTanggal(tgl);
            order.setTanggal_kembali(tgl_kbl);
            order.setStatus(boxStatus.getSelectedItem().toString());
            order.setTotal_harga(total_harga);
            order.setPembayaran(boxPembayaran.getSelectedItem().toString());
            order.setStatus_bayar(boxPembayaran_1.getSelectedItem().toString());
            opo.save(order);
            OrderFrame.loadTable();
            OrderDetailFrame.this.dispose();
            resetAll();

        }else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan disimpan");
        }
    }
});
```

```
btnSimpanOrder.setFont(new Font("Tahoma", Font.PLAIN, 16));
```

```
JButton btnBatalOrder = new JButton("Batal");
btnBatalOrder.setBounds(85, 445, 85, 34);
panelAkumulasi.add(btnBatalOrder);
btnBatalOrder.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
});
btnBatalOrder.setFont(new Font("Tahoma", Font.PLAIN, 16));
```

```
public void setTanggal(Date date) {
    tanggal.setDate(date);
}
public void setTanggal_kembali(Date date) {
    tanggal_kembali.setDate(date);
}

public void setStatus(String status) {
    boxStatus.setSelectedItem(status);
}
public void setLblTotalHargaShow(String total) {
    lblTotalHargaShow.setText("Rp. " + total);
}
public void setBoxPembayaran(String pembayaran) {
    boxPembayaran.setSelectedItem(pembayaran);
}
public void setBoxPembayaran_1(String pembayaran) {
    boxPembayaran_1.setSelectedItem(pembayaran);
}

public static void setId_order(String orderId) {
    id_order = orderId;
}
```

Kelas OrderDetailFrame merupakan implementasi antarmuka pengguna berbasis Java Swing yang dirancang untuk mengelola detail pesanan dalam aplikasi manajemen. Dalam konstruktor kelas ini, berbagai komponen GUI seperti JPanel, JTextField, JComboBox, JLabel, dan JTable diinisialisasi dan diatur untuk menciptakan tampilan yang terstruktur. Kelas ini menggunakan beberapa repositori, termasuk OrderRepo, ServiceRepo, dan OrderDetailRepo, untuk berinteraksi dengan data pesanan, layanan, dan detail pesanan. Metode utama seperti loadTable() dan loadTableOrderDetail() bertugas untuk memuat dan menampilkan data layanan serta detail pesanan berdasarkan id\_order. Selain itu, terdapat mekanisme event handling yang menangani interaksi pengguna, seperti memilih layanan dari tabel dan menghitung total harga secara otomatis saat jumlah diubah. Kelas ini juga menyediakan fungsi CRUD (Create, Read, Update, Delete) melalui tombol untuk menyimpan, mengubah, dan menghapus detail pesanan, serta membatalkan pengisian. Penggunaan JDateChooser memungkinkan pengguna untuk memilih tanggal dengan mudah, dan terdapat dialog untuk memilih pelanggan yang akan diisi di txtCostumer. Secara keseluruhan, OrderDetailFrame menyajikan antarmuka yang komprehensif untuk pengelolaan detail pesanan, memungkinkan pengguna untuk melakukan berbagai operasi dengan efisien.

Membuat Frame Order

The screenshot shows a macOS application window titled "Data Orderan". The window has a standard OS X look with red, yellow, and green close buttons at the top left. At the top right are three buttons: "Buat Orderan" (Create Order), "Hapus" (Delete), and "Edit/Detail". The main content area is a table with the following data:

ID	Nama	Tanggal	Tanggal Penge...	Status	Total Harga	Pembayaran	Status Pembay...
TRX-0001	charloette	2024-12-03	2024-12-05	Progress	23000	Non-Tunai	Lunas
TRX-0002	Henri	2024-12-05	2024-12-08	Diambil	21500	Tunai	Lunas
TRX-0003	william	2024-12-02	2024-12-05	Selesai	55000	Tunai	Lunas
TRX-0004	louis	2024-12-06	2024-12-09	Selesai	40000	Tunai	Lunas

Kode:

Inisiasi variable

```

private static final long serialVersionUID = 1L;
private JPanel contentPane;
private static JTable tableOrder;
private static OrderDetailFrame orderDetailFrame;

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                OrderFrame frame = new OrderFrame();
                frame.setVisible(true);
                frame.loadTable();

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

static OrderRepo ord = new OrderRepo();
static List<Order> ls;
public String id;
public String nama;
public String tanggal;
public String tanggal_kembali;
public String status;
public String total_harga;
public String pembayaran;
public String status_bayar;

```

Tombol “Buat Orderan”

```

JButton btnBuat_Orderan = new JButton("Buat Orderan");
btnBuat_Orderan.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String newOrderId = generateOrderId();
        orderDetailFrame.setOrderId(newOrderId);
        OrderDetailFrame.setId_order(newOrderId);
        orderDetailFrame.setVisible(true);
        orderDetailFrame.getTxtOrderId().setText(newOrderId);
        orderDetailFrame.loadTable();
        orderDetailFrame.loadTableOrderDetail();

    }
});
btnBuat_Orderan.setFont(new Font("Tahoma", Font.PLAIN, 11));
btnBuat_Orderan.setBounds(26, 60, 96, 25);
panel.add(btnBuat_Orderan);

```

Tombol hapus

```

JButton btnHapus = new JButton("Hapus");
btnHapus.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            ord.delete(id);
            id = null;
            loadTable();
        }
        else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan dihapus");
        }
    }
});
btnHapus.setFont(new Font("Tahoma", Font.PLAIN, 11));
btnHapus.setBounds(576, 60, 85, 25);
panel.add(btnHapus);

```

### Tombol Edit

```

JButton btnEdit = new JButton("Edit/Detail\r\n");
btnEdit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(id != null) {
            orderDetailFrame.setOrderID(id);
            OrderDetailFrame.setId_order(id);
            orderDetailFrame.setTxtCostumer(nama);
            SimpleDateFormat sdf_tanggal = new SimpleDateFormat("yyyy-MM-dd");
            try {
                Date parsedDate_tanggal = sdf_tanggal.parse(tanggal);
                orderDetailFrame.setTanggal(parsedDate_tanggal);
            } catch (ParseException ex) {
                ex.printStackTrace();
            }
            SimpleDateFormat sdf_tanggal_kembali = new SimpleDateFormat("yyyy-MM-dd");
            try {
                Date parsedDate = sdf_tanggal_kembali.parse(tanggal_kembali);
                orderDetailFrame.setTanggal_kembali(parsedDate);
            } catch (ParseException ex) {
                ex.printStackTrace();
            }
            orderDetailFrame.setStatus(status);
            orderDetailFrame.setLblTotalHargaShow(total_harga);
            orderDetailFrame.setBoxPembayaran(pembayaran);
            orderDetailFrame.setBoxPembayaran_1(status_bayar);

            orderDetailFrame.setVisible(true);
            orderDetailFrame.loadTable();
            orderDetailFrame.loadTableOrderDetail();

        }else {
            JOptionPane.showMessageDialog(null, "Silahkan pilih data yang akan diedit");
        }
    }
});
btnEdit.setFont(new Font("Tahoma", Font.PLAIN, 11));
btnEdit.setBounds(671, 60, 85, 25);
panel.add(btnEdit);

```

Method Tabel Order untuk setiap row yang di klik

```

tableOrder = new JTable();
tableOrder.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        int selectedRow = tableOrder.getSelectedRow();
        if (selectedRow != -1) {
            id = tableOrder.getValueAt(selectedRow, 0).toString();
        }

        nama = tableOrder.getValueAt(tableOrder.getSelectedRow(), 1).toString();
        tanggal = tableOrder.getValueAt(tableOrder.getSelectedRow(), 2).toString();
        tanggal_kembali = tableOrder.getValueAt(tableOrder.getSelectedRow(), 3).toString();
        status = tableOrder.getValueAt(tableOrder.getSelectedRow(), 4).toString();
        total_harga = tableOrder.getValueAt(tableOrder.getSelectedRow(), 5).toString();
        pembayaran = tableOrder.getValueAt(tableOrder.getSelectedRow(), 6).toString();
        status_bayar = tableOrder.getValueAt(tableOrder.getSelectedRow(), 7).toString();
    }
});
scrollPane.setViewportView(tableOrder);

```

Method Load Table

```

public static void loadTable() {
    ls = ord.show();
    TableOrder to = new TableOrder(ls);
    tableOrder.setModel(to);
    tableOrder.getTableHeader().setVisible(true);
}
public void reset() {

```

Method Generate Custom ID untuk id\_order

```

public String generateOrderID() {
    List<Order> existingOrders = ord.show();
    int maxId = 0;
    for (Order order : existingOrders) {
        String orderId = order.getId_order();
        if (orderId.startsWith("TRX-")) {
            int idNumber = Integer.parseInt(orderId.substring(4));
            if (idNumber > maxId) {
                maxId = idNumber;
            }
        }
    }
    maxId++;

    return String.format("TRX-%04d", maxId);
}

```

Antarmuka ini membantu mengelola dan menampilkan data pesanan dengan berbagai fitur seperti:

LoadTable(): Menampilkan daftar pesanan dalam tabel.

GenerateOrderID(): Membuat ID pesanan baru.

Interaksi Tabel: Pengguna dapat memilih baris untuk mengedit atau menghapus data, dengan tombol khusus untuk masing-masing fungsi.

