Laporan Praktikum

Pemrograman Berorientasi Objek



Disusun Oleh :

**Ahda Rindang Al-Amin (2311531003)**

Dosen Pengampu : Nurfiah, S.ST, M.Kom.

Departemen Informatika

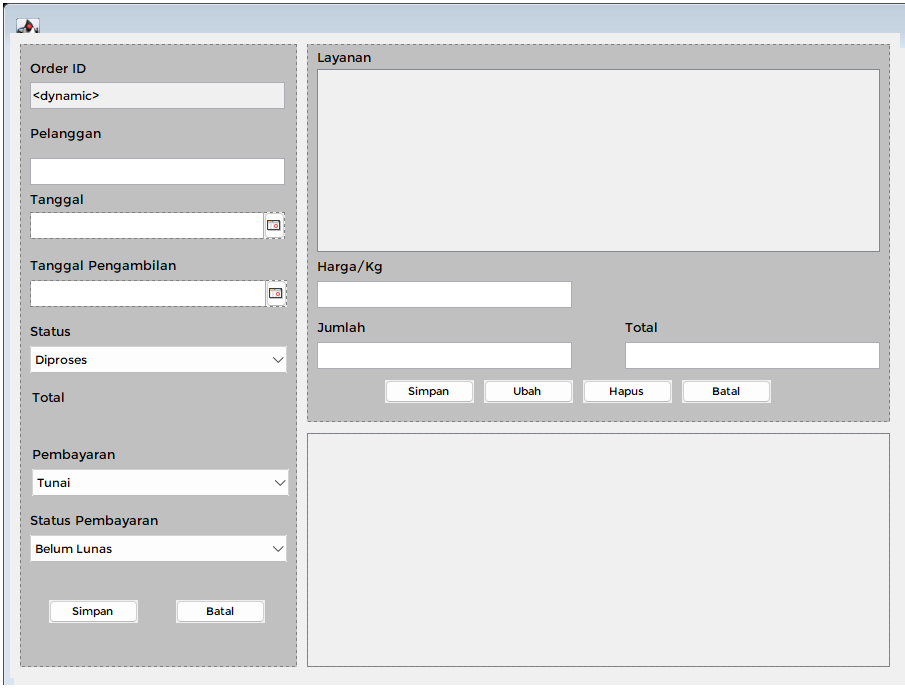
Fakultas Teknologi Informasi

Universitas Andalas

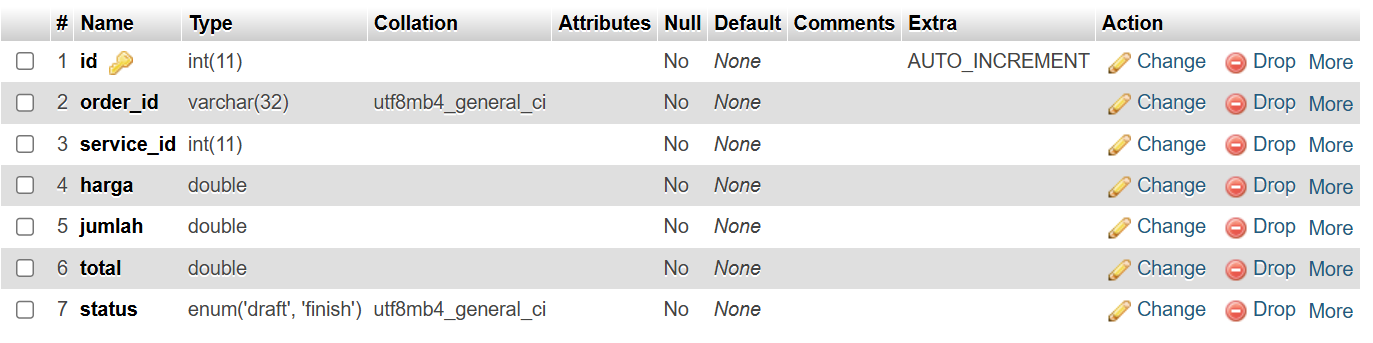
Tahun 2024

**Membuat Fungsi Order Laundry**

1. **Tujuan Praktikum**
2. Mampu membuat UI untuk melakukan order laundry
3. Mampu membuat fungsi pemesanan multi layanan laundry baik menambahkan, mengubah dan menghapusnya
4. Mampu membuat fungsi pemesanan laundry
5. **Metode Praktikum**
6. Buat Desain UI OrderDetailFrame di package ui



1. Buat tabel order\_detail di database



1. Menambahkan daftar layanan ke tabel layanan pada desian UI

* Tambahkan kode program berikut pada OrderDetailFrame

ServiceRepo sr = new ServiceRepo();

List<Service> ls\_service;

public String id\_service;

public static String *id\_pelanggan*="";

* Buat method untuk menampilkan data layanan

public void loadTableService() {

ls\_service = sr.show();

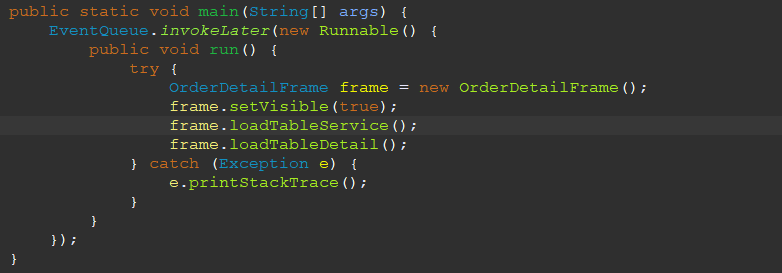
TableService tu = new TableService(ls\_service);

tableLayanan.setModel(tu);

tableLayanan.getTableHeader().setVisible(true);

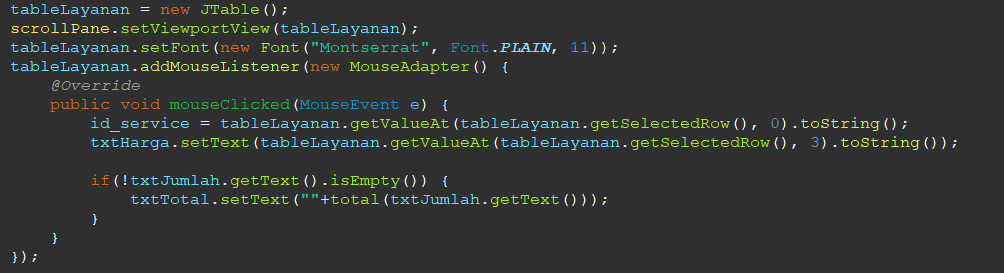
}

* Selanjutnya panggil method loadTableService di dalam method main



1. Menampilkan harga ketika layanan dipilih

* Add event handler mouseClicked pada tableService, kemudian tambhakan kode berikut



1. Buat method total untuk ketika txtHarga terisi, dan operator memasukkan jumlah, maka form total akan terisi secara otomatis.

* Method total di OrderDetailFrame

public double total(String jumlah) {

double result = 0;

if(jumlah.isEmpty()) {

result = 0;

} else {

result = Double.*parseDouble*(jumlah) \* Double.*parseDouble*(txtHarga.getText());

}

return result;

}

* Add event handler keyRelease di txtJumlah, lalu isi dengan kode berikut

txtJumlah = new JTextField();

txtJumlah.addKeyListener(new KeyAdapter() {

*@Override*

public void keyReleased(KeyEvent e) {

String value\_jumlah = txtJumlah.getText().toString();

txtTotal.setText(""+total(value\_jumlah));

}

});

1. Method reset

public void reset() {

txtHarga.setText("");

txtJumlah.setText("");

txtTotal.setText("");

id\_service=null;

id\_order\_detail=null;

}

1. CRUD OrderDetail

* Buat interface OrderDetailDAO pada package dao

public interface OrderDetailDAO {

void save(OrderDetail od);

public List<OrderDetail> show(String order\_id);

public void delete(String id);

public void update(OrderDetail od);

public String total(String order\_id);

}

* Tambahkan class OrderDetail pada package model

public class OrderDetail {

String id, order\_id, service\_id, harga, jumlah, total;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getOrder\_id() {

return order\_id;

}

public void setOrder\_id(String order\_id) {

this.order\_id = order\_id;

}

public String getService\_id() {

return service\_id;

}

public void setService\_id(String service\_id) {

this.service\_id = service\_id;

}

public String getHarga() {

return harga;

}

public void setHarga(String harga) {

this.harga = harga;

}

public String getJumlah() {

return jumlah;

}

public void setJumlah(String jumlah) {

this.jumlah = jumlah;

}

public String getTotal() {

return total;

}

public void setTotal(String total) {

this.total = total;

}

}

* Tambahkan OrderDetailRepo yang mengimplementasi OrderDetailDAO

public class OrderDetailRepo implements OrderDetailDAO{

private Connection connection;

final String insert = "INSERT INTO order\_detail (order\_id, service\_id, harga, jumlah, total) VALUES (?,?,?,?,?);";

final String delete = "DELETE FROM order\_detail WHERE id=?;";

final String update = "UPDATE order\_detail SET order\_id=?, service\_id=?, harga=?, jumlah=?, total=? WHERE id=?;";

public OrderDetailRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(OrderDetail cs) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, cs.getOrder\_id());

st.setString(2, cs.getService\_id());

st.setString(3, cs.getHarga());

st.setString(4, cs.getJumlah());

st.setString(5, cs.getTotal());

st.executeUpdate();

} catch(SQLException e){

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<OrderDetail> show(String order\_id) {

List<OrderDetail> ls = null;

try {

ls = new ArrayList<OrderDetail>();

Statement st = connection.createStatement();

final String select = "SELECT \* FROM order\_detail;";

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

OrderDetail cs = new OrderDetail();

cs.setId(rs.getString("id"));

cs.setOrder\_id(rs.getString("order\_id"));

cs.setService\_id(rs.getString("service\_id"));

cs.setHarga(rs.getString("harga"));

cs.setJumlah(rs.getString("jumlah"));

cs.setTotal(rs.getString("total"));

ls.add(cs);

}

} catch (SQLException e) {

Logger.*getLogger*(OrderDetailDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@Override*

public void delete(String id) {

// **TODO** Auto-generated method stub

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();

}

}

}

*@Override*

public void update(OrderDetail cs) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(update);

st.setString(1, cs.getOrder\_id());

st.setString(2, cs.getService\_id());

st.setString(3, cs.getHarga());

st.setString(4, cs.getJumlah());

st.setString(5, cs.getTotal());

st.setString(6, cs.getId());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch(SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public String total(String order\_id) {

// **TODO** Auto-generated method stub

String query\_total = "SELECT sum(total) as total FROM order\_detail WHERE order\_id="+"'"+order\_id+"'"+";";

Statement st;

ResultSet rs;

String result="";

try {

st = connection.createStatement();

rs = st.executeQuery(query\_total);

if(rs.next()) {

result = ""+rs.getDouble(1);

} else {

result="0";

}

} catch (SQLException e) {

e.printStackTrace();

}

return result;

}

}

* Tambahkan kode berikut ke OrderDetailFrame

OrderDetailRepo repo\_od = new OrderDetailRepo();

List<OrderDetail> ls\_od;

public String id\_order\_detail;

* Tambahkan kode berikut pada event handler btnSimpanDetail

if(id\_order\_detail == null) {

OrderDetail od = new OrderDetail();

od.setOrder\_id(txtOrderID.getText());

od.setService\_id(id\_service);

od.setHarga(txtHarga.getText());

od.setJumlah(txtJumlah.getText());

od.setTotal(txtTotal.getText());

repo\_od.save(od);

JOptionPane.*showMessageDialog*(null, "berhasil disimpan");

loadTableDetail();

reset();

lblTotal.setText(""+repo\_od.total(txtOrderID.getText()));

}

* Tambahkan kode berikut pada event handler btnUbahDetail

public void actionPerformed(ActionEvent e) {

if(id\_order\_detail != null) {

OrderDetail od = new OrderDetail();

od.setOrder\_id(txtOrderID.getText());

od.setService\_id(id\_service);

od.setHarga(txtHarga.getText());

od.setJumlah(txtJumlah.getText());

od.setTotal(txtTotal.getText());

od.setId(id\_order\_detail);

repo\_od.update(od);

loadTableDetail();

reset();

lblTotal.setText(""+repo\_od.total(txtOrderID.getText()));

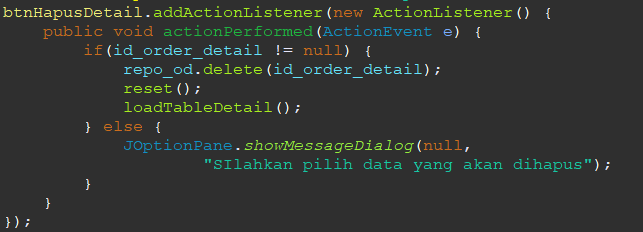
} else {

JOptionPane.*showMessageDialog*(null, "silahkan pilih order terlebih dahulu");

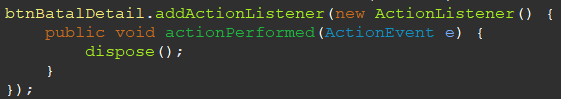
}

}

* Tambahkan kode berikut pada event handler btnHapusDetail



* Tambahkan kode berikut pada event handler btnBatalDetail



* Buat class TableOrderDetail di package tabel

public class TableOrderDetail extends AbstractTableModel {

List<OrderDetail> ls;

private String[] columnNames = {"ID", "Order ID", "Service ID", "Harga", "Jumlah", "Total"};

public TableOrderDetail(List<OrderDetail> ls) {

this.ls = ls;

}

*@Override*

public int getRowCount() {

// **TODO** Auto-generated method stub

return ls.size();

}

*@Override*

public int getColumnCount() {

// **TODO** Auto-generated method stub

return 6;

}

public String getColumnName(int column) {

return columnNames[column];

}

*@Override*

public Object getValueAt(int rowIndex, int columnIndex) {

// **TODO** Auto-generated method stub

switch (columnIndex) {

case 0:

return ls.get(rowIndex).getId();

case 1:

return ls.get(rowIndex).getOrder\_id();

case 2:

return ls.get(rowIndex).getService\_id();

case 3:

return ls.get(rowIndex).getHarga();

case 4:

return ls.get(rowIndex).getJumlah();

case 5:

return ls.get(rowIndex).getTotal();

default:

return null;

}

}

}

* Tambahkan method loadTableDetail() untuk menampilkan tabel Order Detail lalu panggil di method main

public void loadTableDetail() {

ls\_od = repo\_od.show(id\_order\_detail);

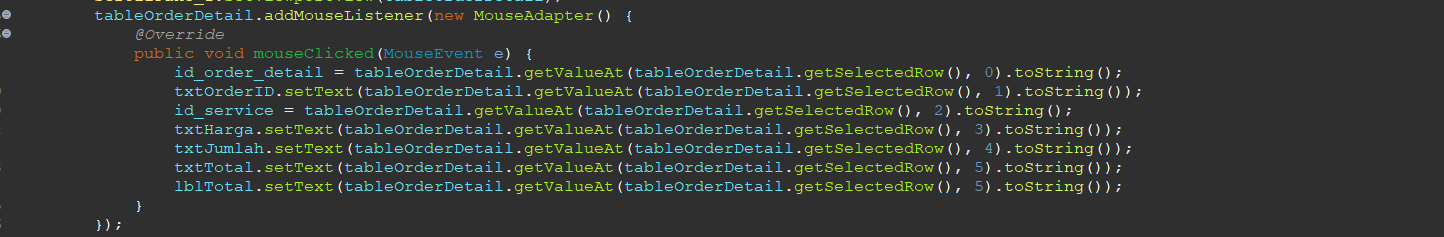
TableOrderDetail tu = new TableOrderDetail(ls\_od);

tableOrderDetail.setModel(tu);

tableOrderDetail.getTableHeader().setVisible(true);

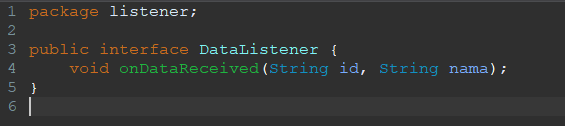
}

* Add event handler mouseClicked pada tabel Order Detail lalu isi dengan kode berikut



1. Buat JDialog Pelanggan untuk menampilkan nama-nama pelanggan yang ada pada aplikasi. Ketika operator klik form pelanggan maka akan ditampilkan JDialog daftar pelanggan, selanjutnya operator memilih pelanggan maka secara otomatis form pelanggan akan terisi oleh data pelanggan tersebut.

* Buat package listener, lalu buat interface DataListener



* Tambahkan JDialog DialogPelanggan pada package ui, isi dengan kode program berikut

public class DialogPelanggan extends JDialog {

private DataListener listener;

CustomerRepo usr = new CustomerRepo();

List<Customer> ls;

public String id;

private JTable tablePelanggan;

public DialogPelanggan(DataListener listener) {

this.listener = listener;

setModal(true);

setModalityType(*ModalityType*.***APPLICATION\_MODAL***);

setSize(450, 249);

setLocationRelativeTo(null);

setTitle("Data Pelanggan");

getContentPane().setLayout(null);

JScrollPane scrollPane = new JScrollPane();

scrollPane.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

}

});

scrollPane.setBounds(10, 10, 416, 132);

getContentPane().add(scrollPane);

tablePelanggan = new JTable();

tablePelanggan.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

listener.onDataReceived(tablePelanggan.getValueAt(tablePelanggan.getSelectedRow(),0).toString(), tablePelanggan.getValueAt(tablePelanggan.getSelectedRow(),1).toString());

dispose();

}

});

scrollPane.setViewportView(tablePelanggan);

loadTable();

}

public void loadTable() {

ls = usr.show();

TableCustomer tu = new TableCustomer(ls);

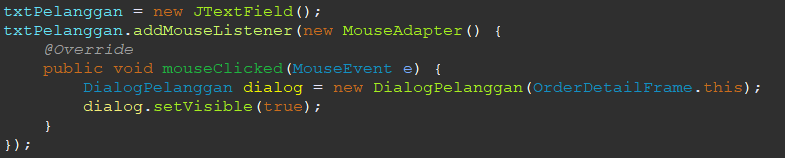
tablePelanggan.setModel(tu);

tablePelanggan.getTableHeader().setVisible(true);

}

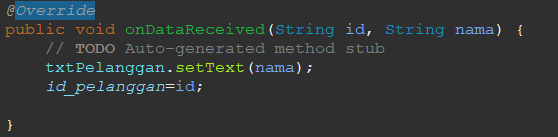
}

* Add event handler MouseListener pada txtPelanggan, lalu isi dengan kode berikut



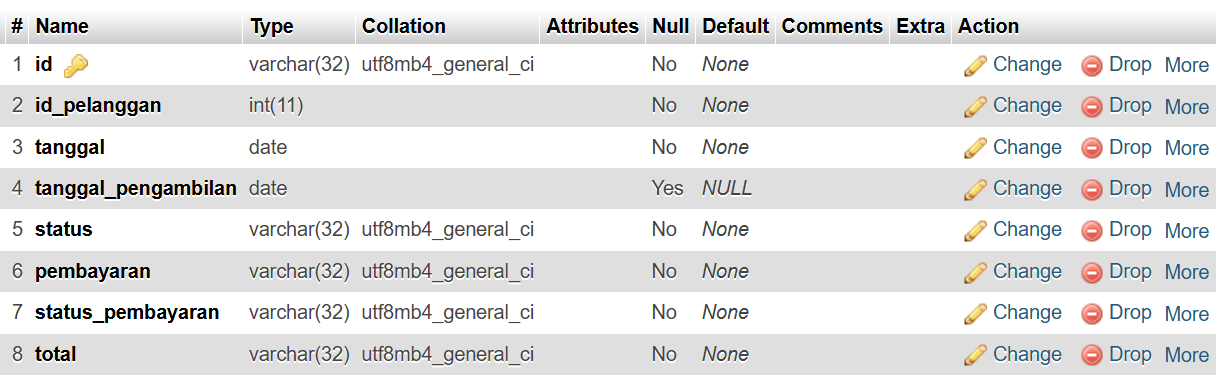
* Implements DataListener pada OrderDetailFrame, lalu implementasi method onDataReceived





1. Menyimpan data order

* Buat tabel orders pada database



* Tambahkan model Order

public class Order {

String id, id\_pelanggan, tanggal, tanggal\_pengambilan, status, pembayaran, status\_pembayaran, total;

public String getId() {

return id;

}

public void setId(String id) {

this.id = id;

}

public String getId\_pelanggan() {

return id\_pelanggan;

}

public void setId\_pelanggan(String id\_pelanggan) {

this.id\_pelanggan = id\_pelanggan;

}

public String getTanggal() {

return tanggal;

}

public void setTanggal(String tanggal) {

this.tanggal = tanggal;

}

public String getTanggal\_pengambilan() {

return tanggal\_pengambilan;

}

public void setTanggal\_pengambilan(String tanggal\_pengambilan) {

this.tanggal\_pengambilan = tanggal\_pengambilan;

}

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\_pembayaran() {

return status\_pembayaran;

}

public void setStatus\_pembayaran(String status\_pembayaran) {

this.status\_pembayaran = status\_pembayaran;

}

public String getTotal() {

return total;

}

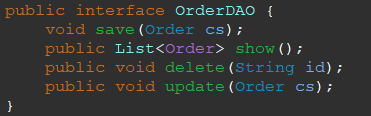
public void setTotal(String total) {

this.total = total;

}

}

* Tambahkan interface OrderDAO



* Tambahkan OrderRepo

public class OrderRepo implements OrderDAO{

private Connection connection;

final String insert = "INSERT INTO orders (id, id\_pelanggan, tanggal, tanggal\_pengambilan, status, pembayaran, status\_pembayaran, total) VALUES (?,?,?,?,?,?,?,?);";

final String select = "SELECT \* FROM orders;";

final String delete = "DELETE FROM orders WHERE id=?;";

final String delete\_detail = "DELETE FROM order\_detail WHERE order\_id=?;";

final String update = "UPDATE orders SET id\_pelanggan=?, tanggal=?, tanggal\_pengambilan=?, status=?, pembayaran=?, status\_pembayaran=?, total=? WHERE id=?;";

public OrderRepo() {

connection = Database.*koneksi*();

}

*@Override*

public void save(Order cs) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(insert);

st.setString(1, cs.getId());

st.setString(2, cs.getId\_pelanggan());

st.setString(3, cs.getTanggal());

st.setString(4, cs.getTanggal\_pengambilan());

st.setString(5, cs.getStatus());

st.setString(6, cs.getPembayaran());

st.setString(7, cs.getStatus\_pembayaran());

st.setString(8, cs.getTotal());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public List<Order> show() {

// **TODO** Auto-generated method stub

List<Order> ls = null;

try {

ls = new ArrayList<Order>();

Statement st = connection.createStatement();

ResultSet rs = st.executeQuery(select);

while(rs.next()) {

Order cs = new Order();

cs.setId(rs.getString("id"));

cs.setId\_pelanggan(rs.getString("id\_pelanggan"));

cs.setTanggal(rs.getString("tanggal"));

cs.setTanggal\_pengambilan(rs.getString("tanggal\_pengambilan"));

cs.setStatus(rs.getString("status"));

cs.setPembayaran(rs.getString("pembayaran"));

cs.setStatus\_pembayaran(rs.getString("status\_pembayaran"));

cs.setTotal(rs.getString("total"));

ls.add(cs);

}

} catch (SQLException e) {

Logger.*getLogger*(OrderDAO.class.getName()).log(Level.***SEVERE***, null, e);

}

return ls;

}

*@Override*

public void delete(String id) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

PreparedStatement st\_detail = null;

try {

st = connection.prepareStatement(delete);

st.setString(1, id);

st.executeUpdate();

st\_detail = connection.prepareStatement(delete\_detail);

st\_detail.setString(1, id);

st\_detail.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

st\_detail.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

*@Override*

public void update(Order cs) {

// **TODO** Auto-generated method stub

PreparedStatement st = null;

try {

st = connection.prepareStatement(update);

st.setString(1, cs.getId\_pelanggan());

st.setString(2, cs.getTanggal());

st.setString(3, cs.getTanggal\_pengambilan());

st.setString(4, cs.getStatus());

st.setString(5, cs.getPembayaran());

st.setString(6, cs.getStatus\_pembayaran());

st.setString(7, cs.getTotal());

st.setString(8, cs.getId());

st.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

st.close();

} catch (SQLException e) {

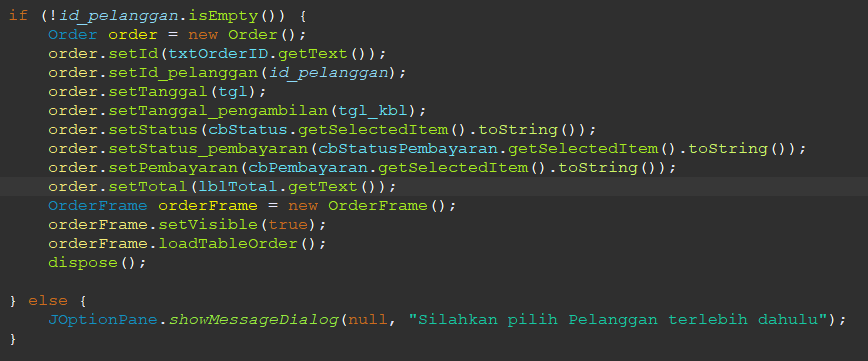
e.printStackTrace();

}

}

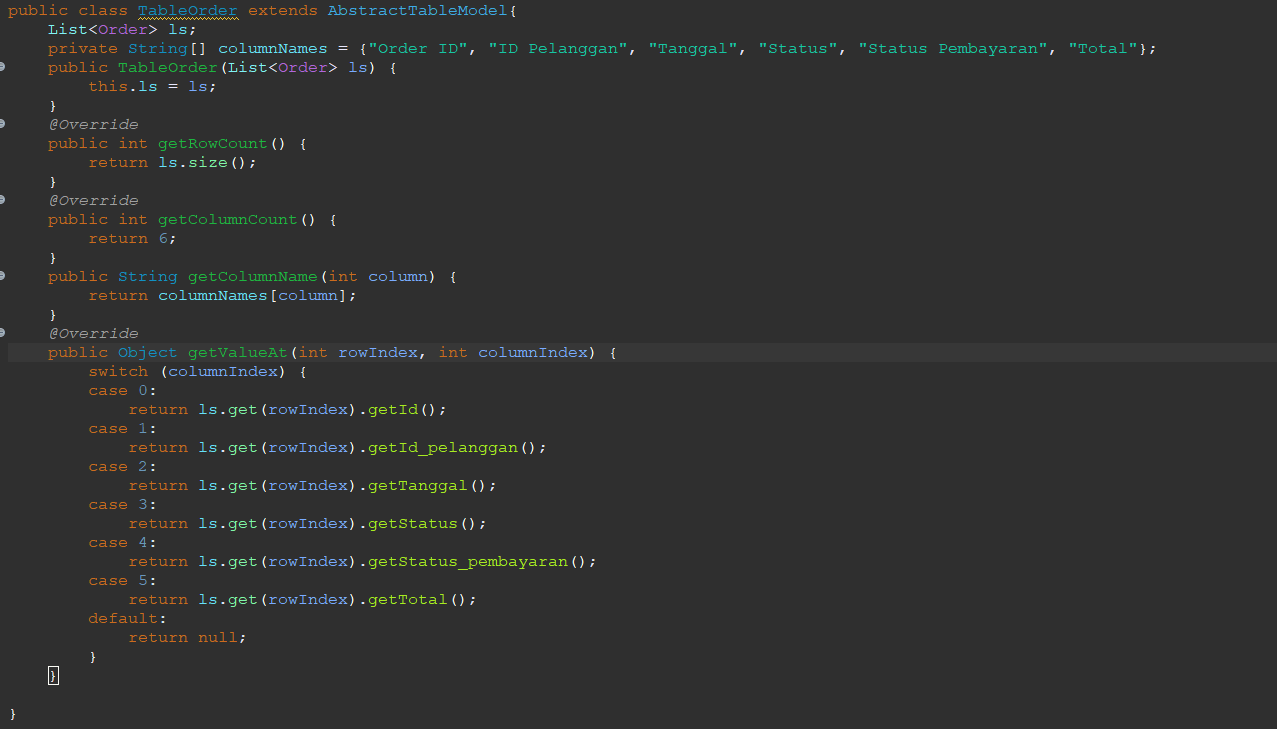
}

* Tambahkan kode berikut pada tombol SimpanOrder

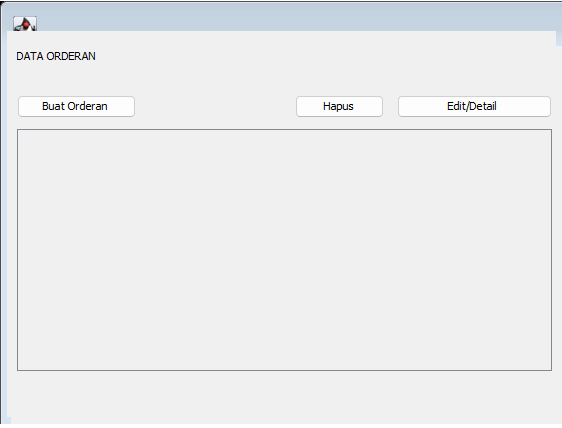


1. Menampilkan Data Order

* Buat class TableOrder



* Tambahkan ui OrderFrame



* Tambahkan kode berikut pada OrderFrame, lalu panggil method loadTableOrder() pada method main

OrderRepo repo\_od = new OrderRepo();

List<Order> ls\_od;

String order\_id="";

public void loadTableOrder() {

ls\_od = repo\_od.show();

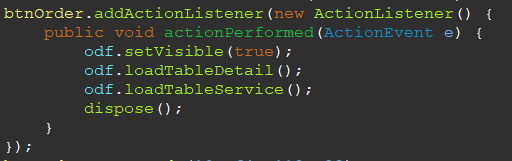
TableOrder tu = new TableOrder(ls\_od);

tableOrder.setModel(tu);

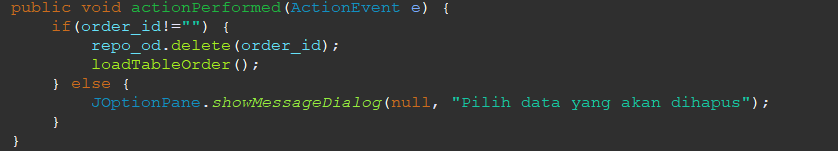
tableOrder.getTableHeader().setVisible(true);

}

* Tambahkan kode berikut pada tombol buat orderan



* Tambahkan kode berikut pada tombol hapus



**Latihan / Tugas**

* Order ID masih manual, untuk itu buatkanlah sebuah fungsi yang dapat generate Order ID secara otomatis dan unik, sehingga tidak ada duplikasi Order ID Ketika membuat pesananan/orderan.

1. Tambahkan method berikut di OrderRepo

public String generateOrderId() {

String newOrderId = "TRX-0001"; // Default untuk ID pertama

try {

String query = "SELECT id FROM orders ORDER BY id DESC LIMIT 1";

PreparedStatement stmt = connection.prepareStatement(query);

ResultSet rs = stmt.executeQuery();

if (rs.next()) {

String lastOrderId = rs.getString("id");

int lastNumber = Integer.*parseInt*(lastOrderId.split("-")[1]);

int newNumber = lastNumber + 1;

newOrderId = String.*format*("TRX-%04d", newNumber);

}

} catch (SQLException e) {

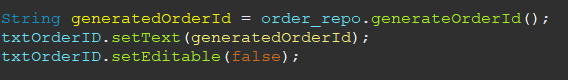
e.printStackTrace();

}

return newOrderId;

}

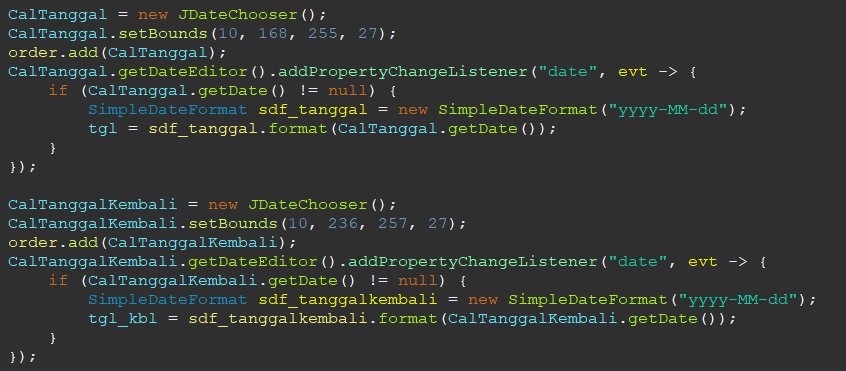
1. Tambahkan kode berikut setelah inisialisasi txtOrderID di OrderDetailFrame



1. Order ID akan digenerate secara otomatis TRX-0001, TRX-0002, dst

* Tanggal pesanan dan tanggal pengambilan silahkan ditambahkan menggunkan JCalender

1. Tambahkan external jar JCalendar di projek
2. Ganti kode tanggal yang sebelumnya JTextField menjadi seperti berikut

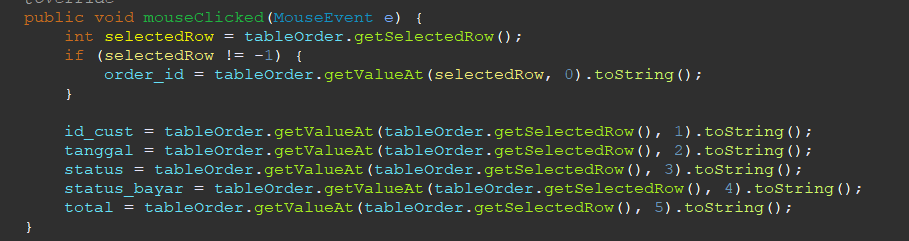


1. Perbaiki posisi JCalender pada desain
2. Lalu, yang sebelumnya get text dari textfield tanggal dan tanggalpengambilan diubah menjadi tgl dan tgl\_kbl

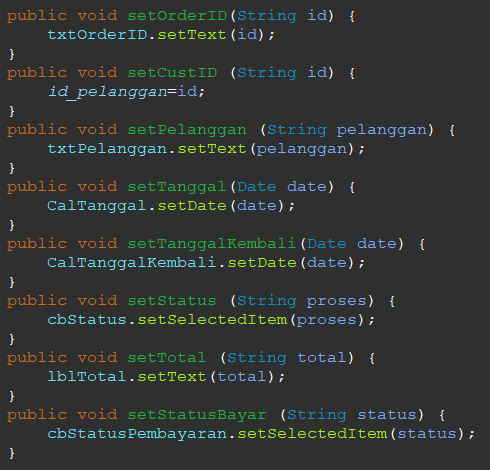


* Pada saat tombol Edit/Detail pada OrderFrame diklik maka akan memanggil OrderDetailFrame dan menampikan data-data orderan seseuai dengan data yang dipilih dan dapat dilakukan fungsi untuk edit orderan.

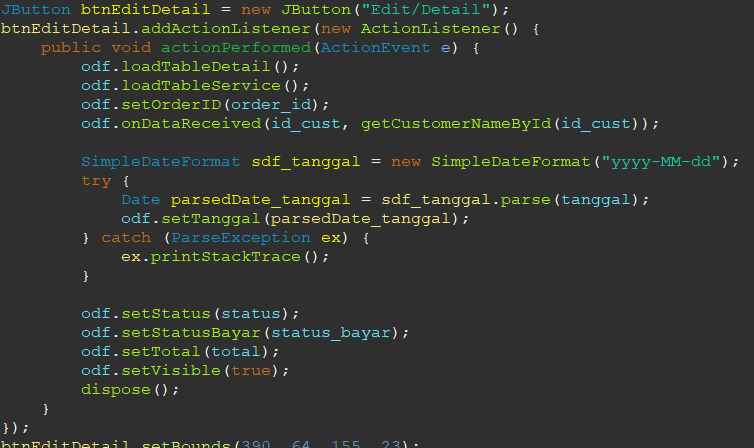
1. Add event handler mouseClicked pada table order, lalu isi dengan kode berikut



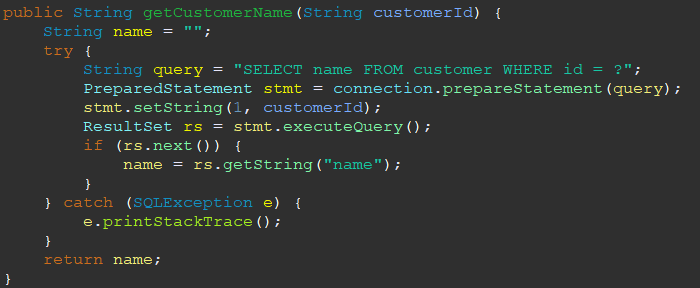
1. Lalu, tambahkan method Setter di OrderDetailFrame



1. Tambahkan kode berikut ke event handler tombol Edit/Detail



1. Tambahkan method berikut di OrderRepo



Method tersebut berfungsi untuk database dapat mengambil nama dari pelanggan dari hanya diberikan id.

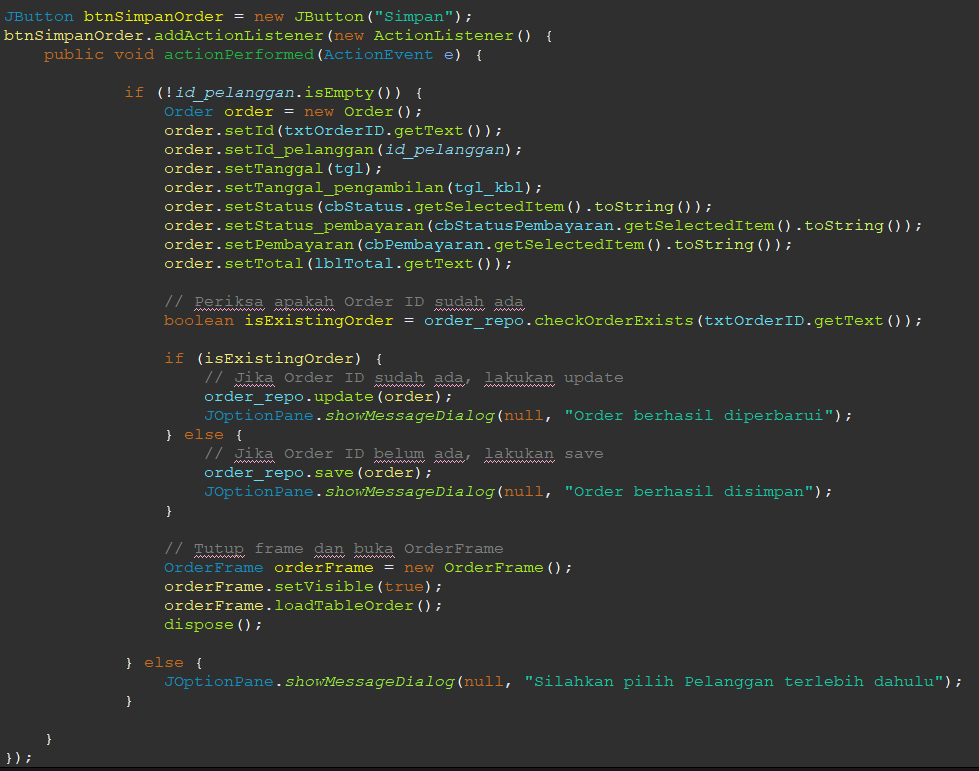
1. tambahkan method berikut di OrderFrame untuk memanggil method getCustomerName

private String getCustomerNameById(String customerId) {

return repo\_od.getCustomerName(customerId);

}

1. modifikasi tombol SimpanOrder pada OrderDetailFrame agar ketika id belum ada akan menjalankan perintah save ke database, dan jika id sudah ada akan update data yang ada di database



1. tambahkan method berikut di OrderRepo untuk mengecek ada tidaknya id order di database

