**LAPORAN RESMI**

**MODUL V**

**GUI DATABASE**

**PEMROGRAMAN VISUAL**



**NAMA : ANISYAFAAH**

**N.R.P : 22041100105**

**DOSEN : Ir. ACH. DAFID, S.T., M.T.**

**ASISTEN : NURI HIDAYATULOH**

**TGL PRAKTIKUM : 01 NOVEMBER 2023**

**Disetujui : 15 November 2023**

**Asisten**

**NURI HIDAYATULOH**

**21.04.411.00100**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

GUI (Graphical User Interface) untuk database dalam pengembangan aplikasi Java memiliki peran penting dalam mempermudah interaksi antara pengguna dan sistem database. GUI menyediakan antarmuka visual yang memungkinkan pengguna untuk berinteraksi dengan data secara intuitif melalui elemen-elemen seperti formulir, tabel, dan tombol. Penerapan GUI dalam pengelolaan database pada Java menjadi krusial karena memberikan pengguna kemampuan untuk memanipulasi data dengan lebih mudah tanpa harus menguasai bahasa SQL secara mendalam.

Penggunaan GUI database pada Java memberikan keuntungan signifikan dalam hal keterbacaan dan kejelasan, khususnya bagi mereka yang tidak memiliki latar belakang teknis yang kuat. Dengan menyediakan antarmuka grafis, pengembang dapat menghadirkan informasi secara visual, memfasilitasi pencarian, penyortiran, dan pembaruan data. Ini memberikan pengguna pengalaman yang lebih ramah, meningkatkan efisiensi operasional, dan mengurangi potensi kesalahan input yang mungkin terjadi dalam penggunaan perintah SQL manual.

Selain itu, dengan memanfaatkan fitur GUI, pengembang dapat dengan mudah menggabungkan fungsionalitas database dengan elemen-elemen lain dalam aplikasi, seperti grafik, laporan, atau fitur lainnya. Hal ini membuka peluang untuk menciptakan solusi perangkat lunak yang lebih komprehensif dan mudah diimplementasikan. Oleh karena itu, penggunaan GUI database pada Java bukan hanya sekadar aspek estetika, tetapi juga strategi desain yang penting dalam membangun aplikasi yang lebih efektif dan responsif.

## **Tujuan**

* Mampu membuat aplikasi berbasis GUI untuk memanipulasi basis data
* Mampu mem-populateisi untuk komponen swing GUI dengan isi dari database
* Mampu melakukan CRUD database via aplikasi swing java

# **BAB II DASAR TEORI**

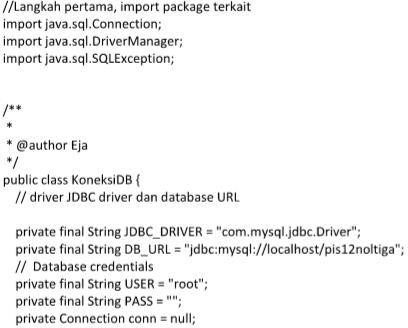
## **Insert Database**

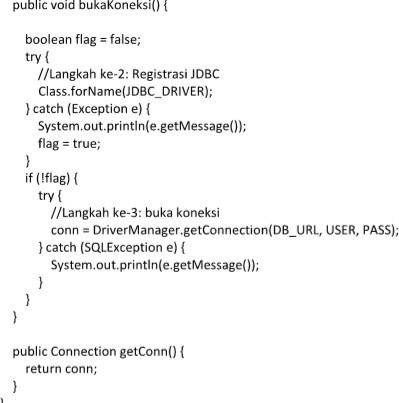
Form GUI dengan memanfaatkan modul pengaksesan database yang telah ada sebelumnya, Input ke GUI dengan menerima input user melalui text field, proses menjadi lebih gampang untuk dilakukan.

Kasus: Input ke table DesaNinja yang terdiri dari atribut id desa, nama, pemimpin. Urutannya:

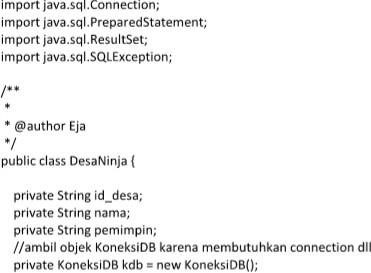
1. Buat 3 textfield di aplikasi gui dan 1 tombol “simpan”
2. Ambil input user dengan method getText. Simpan input tersebut ke beberapa variabel.
3. Buat 1 objek dari clas DesaNinja dengan parameter konstruktor adalah variabel yang telah dimiliki sebelumnya.
4. Panggil method yang merepresentasikan method save() atau method untuk menyimpan data tersebut.
5. Tampilkan pesan “Berhasil” jika input user berhasil dimasukkan ke database, tampilkan gagal jika gagal.

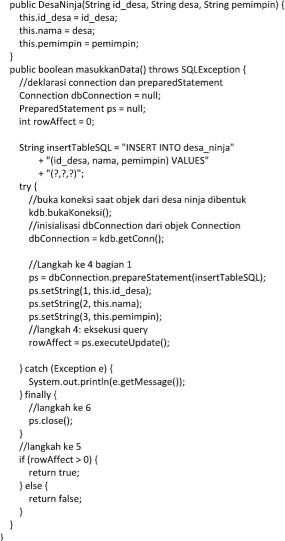
Secara detil: Langkah pertama, buat kelas “KoneksiDB”.



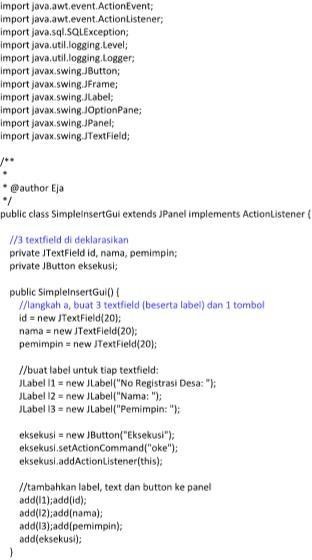


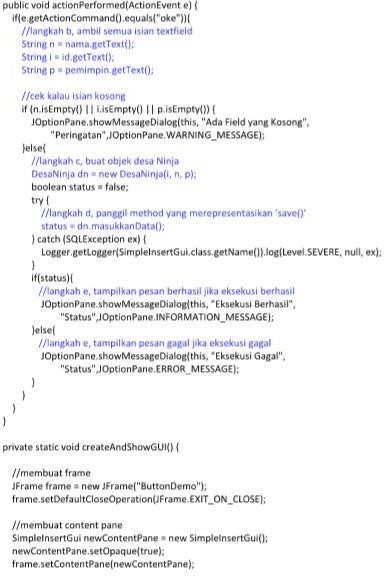
Buatlah Kelas DesaNinja (Fokus hanya ke method untuk menyimpan data saja).

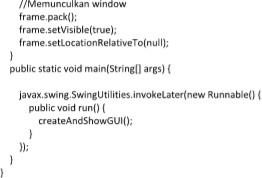




Kelas untuk swing java.







## **Select Database – JTable**

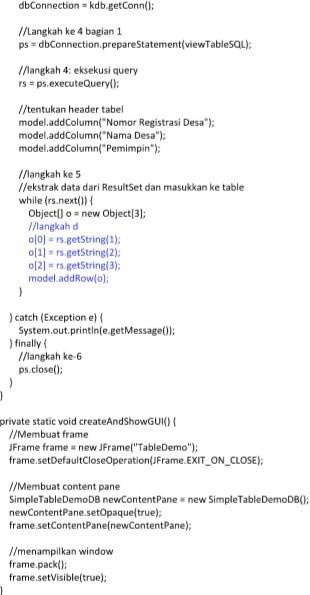
Mengisi tabel dengan data dari database mempunyai banyak cara. Salah satunya yang dituliskan pada modul ini, dengan tetap memanfaatkan kelas dari modul praktikum sebelumnya. Langkah-langkah:

* 1. Gunakan Kelas “KoneksiDB” untuk membuka koneksi dan lain-lain
  2. Deklarasikan JTable dengan modelnya pada kelas yang digunakan untuk penempatan swing.
  3. Ikuti langkah pengaksesan data, kecuali langkah ke-5 harus dimodifikasi. Tuliskan method pengaksesan data ini di kelas yang sama dengan kelas untuk menuliskan swing (boleh diletakkan berbeda, tapi yang dituliskan di modul ini adalah cara yang cukup sederhana).
  4. Langkah ke-5 dimodifikasi dengan tidak menampilkan hasil ResultSet tapi memasukkannya ke dalam model JTable.

Lengkapnya:







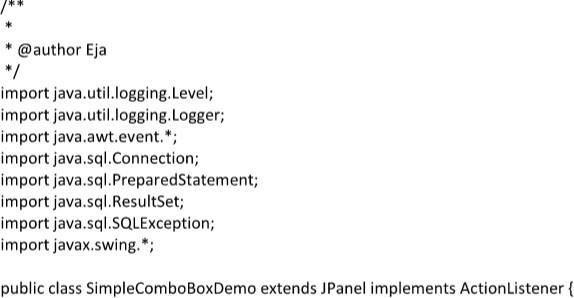


## **Select Database – JcomboBox**

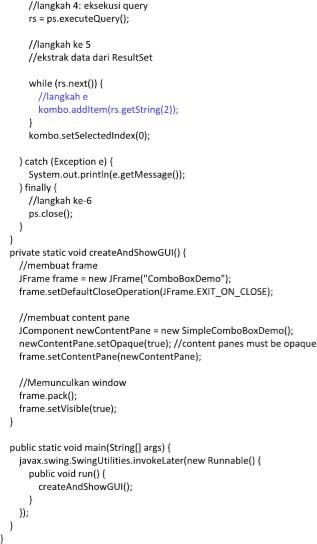
Mengisi combo box dengan data yang berasal dari database, sama seperti mengisi data dari database untuk diisikan ke table. Ada banyak cara yang bisa dilakukan. Urutan langkah pengerjaan yang dilakukan oleh modul praktikum ini adalah sebagai berikut:

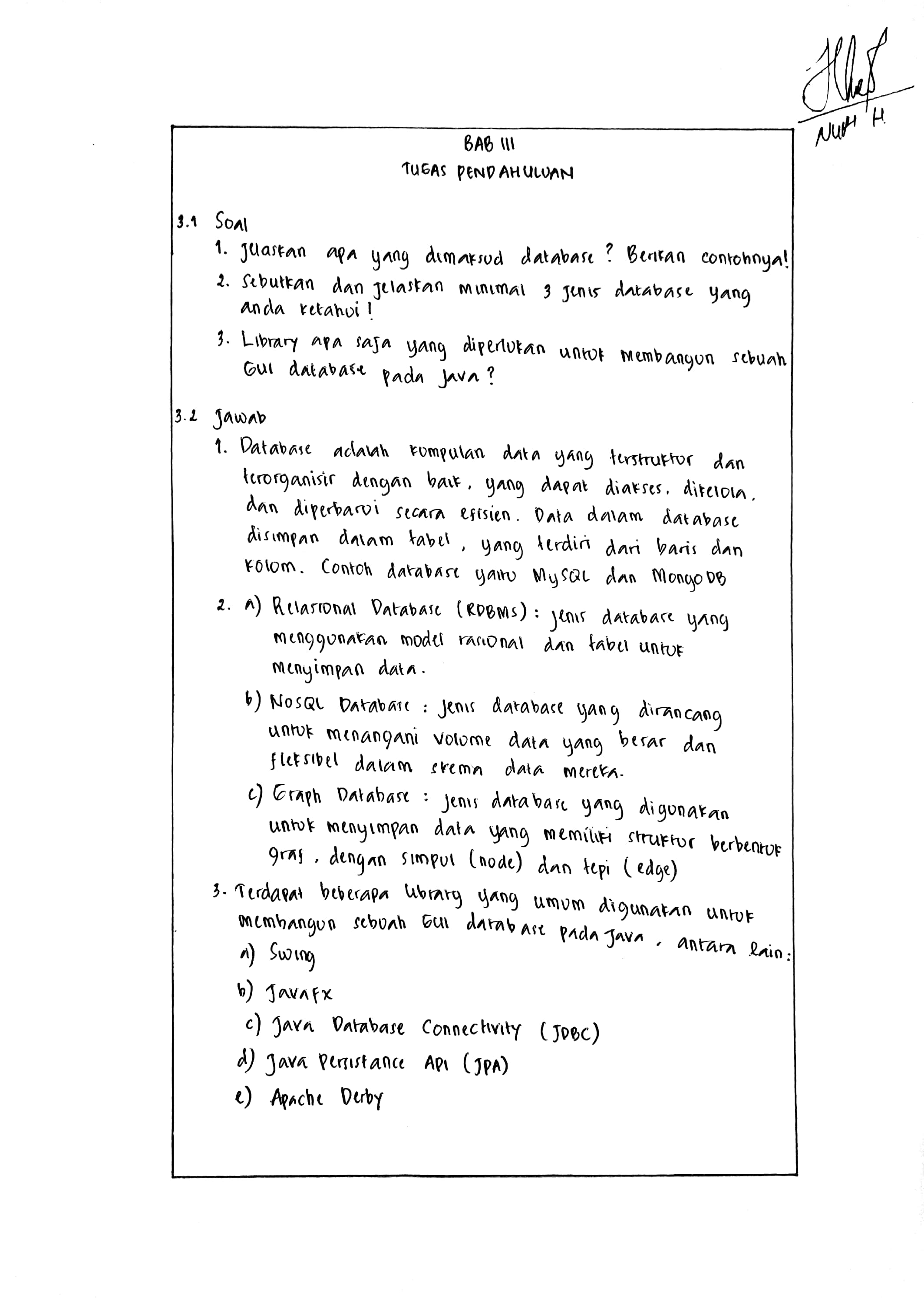
1. Gunakan Kelas “KoneksiDB” untuk membuka koneksi dan lain-lain
2. Deklarasikan JComboBox pada kelas yang digunakan untuk penempatan swing.
3. Ikuti langkah pengaksesan data, kecuali langkah ke-5 harus dimodifikasi. Tuliskan method pengaksesan data ini di kelas yang sama dengan kelas untuk menuliskan swing (alasan yang sama dengan sebelumnya).
4. Panggil method pengaksesan data tersebut pada konstruktor, sehingga saat ditampilkan pertama kali, combo box sudah menampilkan isi dari database.
5. Langkah ke-5 dimodifikasi dengan tidak menampilkan hasil ResultSet tapi memasukkannya ke dalam komponen JComboBox.

Lengkapnya:









# **BAB IV IMPLEMENTASI**

## **Source Code**

### **File Frame**

|  |
| --- |
| package modul5;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.sql.SQLException;  import java.sql.Statement;  import javax.swing.JOptionPane;  import javax.swing.table.DefaultTableModel;  public class frame extends javax.swing.JFrame {  private boolean isEditMode = false;    public frame() {  conn = new koneksii();  initComponents();  tampilData();  bersihkan();  tambahDropdown();  internal1.setVisible(false);  tabelHistory();  }    public void tambahDropdown() {  try {  Statement st = conn.getConnection().createStatement();  ResultSet rs = st.executeQuery("SELECT nama\_barang FROM barang");  while (rs.next()) {  String namaBarang = rs.getString("nama\_barang");  ComboBarang.addItem(namaBarang);  }  st.close();  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  }    public void tampilData(){  DefaultTableModel tbl = new DefaultTableModel();  tbl.addColumn("Nama Barang");  tbl.addColumn("Kode Barang");  tbl.addColumn("Harga Barang");    Tabel.setModel(tbl); // tbl\_mahasiswa disesuaikan dengan variable jtable  try {  Statement st = conn.getConnection().createStatement();  ResultSet rs = st.executeQuery("SELECT \* FROM barang");  while (rs.next()) {  tbl.addRow( new Object[] {  rs.getString("nama\_barang"),  rs.getString("kode\_barang"),  rs.getString("harga\_barang")  });  Tabel.setModel(tbl);  //Dropdown.addElement(rs.getString("nama\_barang"));  }  } catch (Exception e) {  }  }    private void bersihkan(){  TextHarga.setText("");  TextKode.setText("");  TextBarang.setText("");  }    public void tabelHistory(){  DefaultTableModel tbl = new DefaultTableModel();  tbl.addColumn("Pembeli");  tbl.addColumn("Barang");  tbl.addColumn("Kode");  tbl.addColumn("Harga");  tbl.addColumn("Jumlah");  tbl.addColumn("Harga Total");  tbl.addColumn("Uang");  tbl.addColumn("Kembalian");    tblRiwayat.setModel(tbl);  try{  Connection c = conn.getConnection();  Statement s = c.createStatement();  String sql = "SELECT \* FROM datapembeli";  ResultSet r = s.executeQuery(sql);    while (r.next()){  tbl.addRow(new Object[]{  r.getString("nama\_pembeli"),  r.getString("nama\_barang"),  r.getString("kode\_barang"),  r.getString("harga\_barang"),  r.getString("jumlah\_beli"),  r.getString("harga\_total"),  r.getString("uang\_pembeli"),  r.getString("kembalian"),  });  tblRiwayat.setModel(tbl);  }  }catch(SQLException e){  e.printStackTrace();  }  }  @SuppressWarnings("unchecked")  //Generated Code  private void ComboBarangActionPerformed(java.awt.event.ActionEvent evt) {  String selectedBarang = ComboBarang.getSelectedItem().toString();  try {  // Create a statement and execute a query to get the price from the database  Statement st = conn.getConnection().createStatement();  String query = "SELECT kode\_barang, harga\_barang FROM barang WHERE nama\_barang = '" + selectedBarang + "'";  ResultSet rs = st.executeQuery(query);  if (rs.next()) {  // mengset harga Harga\_Barang text field  Text\_Barang.setText(rs.getString("kode\_barang"));  Text\_Harga.setText(rs.getString("harga\_barang"));  } else {  // Handle the case where the item was not found in the database  Text\_Harga.setText("Item not found");  }  rs.close();  st.close();  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  }  private void HitungTotal4ActionPerformed(java.awt.event.ActionEvent evt) {  int jumlahPembelian = Integer.parseInt(TextJmlBeli4.getText());  // Mendapatkan nilai harga barang dari input atau dari database jika Anda ingin mengambilnya dari database  int hargaBarang = Integer.parseInt(Text\_Harga.getText());  // Menghitung total harga  int totalHarga = jumlahPembelian \* hargaBarang;  // Menampilkan total harga pada input total harga  TextTotalHarga4.setText(String.valueOf(totalHarga));  }  private void btnNextActionPerformed(java.awt.event.ActionEvent evt) {  internal1.setVisible(true);  }  private void ResetActionPerformed(java.awt.event.ActionEvent evt) {  TextBarang.setText("");  TextKode.setText("");  TextHarga.setText("");  Text\_Nama.setText("");  TextJmlBeli4.setText("");  TextTotalHarga4.setText("");  TextTotalDibayar1.setText("");  TextTotalKembali1.setText("");    isEditMode = false;  Simpan.setText("Simpan");    // Menghapus semua data dari tabel dalam database    }  private void HapusActionPerformed(java.awt.event.ActionEvent evt) {  int selectedRow = Tabel.getSelectedRow();  if (selectedRow == -1) {  JOptionPane.showMessageDialog(this, "Pilih baris yang ingin dihapus.");  } else {  DefaultTableModel model = (DefaultTableModel) Tabel.getModel();  // Mendapatkan kode barang dari baris yang dipilih  String kodeBarang = model.getValueAt(selectedRow, 1).toString();  // Melakukan penghapusan data dari database  try {  Statement st = conn.getConnection().createStatement();  st.executeUpdate("DELETE FROM barang WHERE kode\_barang = '" + kodeBarang + "'");  st.close();  JOptionPane.showMessageDialog(this, "Data berhasil dihapus.");  // Menghapus baris dari tabel  model.removeRow(selectedRow);  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  }  isEditMode = false;  Simpan.setText("Simpan");    TextBarang.setText("");  TextKode.setText("");  TextHarga.setText("");  }  private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {  if (isEditMode) {  // Anda berada dalam mode pengeditan, lakukan perintah pengeditan di sini  // Misalnya, Anda bisa menggunakan kode SQL UPDATE untuk mengubah data yang ada di database.  try {  Statement st = conn.getConnection().createStatement();  st.executeUpdate("UPDATE barang SET nama\_barang='" + TextBarang.getText() + "', kode\_barang='"  + TextKode.getText() + "', harga\_barang='" + TextHarga.getText() + "' WHERE kode\_barang='"  + TextKode.getText() + "'");  st.close();  JOptionPane.showMessageDialog(this, "Data berhasil diubah");  tampilData();  bersihkan();  tambahDropdown();  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  // Set kembali ke mode penambahan data baru  isEditMode = false;  Simpan.setText("Simpan");  } else {  // Anda berada dalam mode penambahan data baru, lakukan perintah penambahan di sini  try {  Statement st = conn.getConnection().createStatement();  st.executeUpdate("INSERT INTO barang VALUES('" + TextBarang.getText() + "', '"  + TextKode.getText() + "', '" + TextHarga.getText() + "')");  st.close();  JOptionPane.showMessageDialog(this, "Data berhasil disimpan");  tampilData();  bersihkan();  tambahDropdown();  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  }  }  private void TabelMouseClicked(java.awt.event.MouseEvent evt) {  DefaultTableModel model = (DefaultTableModel) Tabel.getModel();  int selectedRow = Tabel.getSelectedRow();  // Cek apakah ada baris yang dipilih  if (selectedRow >= 0) {  TextBarang.setText(model.getValueAt(selectedRow, 0).toString());  TextKode.setText(model.getValueAt(selectedRow, 1).toString());  TextHarga.setText(model.getValueAt(selectedRow, 2).toString());  isEditMode = true;  Simpan.setText("Edit"); // Ubah teks tombol menjadi "Edit"  }else {  // Anda berada dalam mode penambahan data baru, lakukan perintah penambahan di sini  try {  Statement st = conn.getConnection().createStatement();  st.executeUpdate("INSERT INTO barang VALUES('" + TextBarang.getText() + "', '"  + TextKode.getText() + "', '" + TextHarga.getText() + "')");  st.close();  JOptionPane.showMessageDialog(this, "Data berhasil disimpan");  tampilData();  bersihkan();  tambahDropdown();  } catch (Exception e) {  e.printStackTrace();  JOptionPane.showMessageDialog(this, "Terjadi kesalahan: " + e.getMessage());  }  }  }  private void HitungKembalian1ActionPerformed(java.awt.event.ActionEvent evt) {  int totalBayar = Integer.parseInt(TextTotalDibayar1.getText());  int totalHarga = Integer.parseInt(TextTotalHarga4.getText());  int kembalian = totalBayar - totalHarga;  if (kembalian < 0){  JOptionPane.showMessageDialog(this, "Uang yang anda bayarkan kurang");  }else {  TextTotalKembali1.setText(String.valueOf(kembalian));  }  }  private void btnSimpannActionPerformed(java.awt.event.ActionEvent evt) {  // TODO add your handling code here:  String barang = ComboBarang.getSelectedItem().toString();  String kode = Text\_Barang.getText();  String harga = Text\_Harga.getText();  String nama = Text\_Nama.getText();  String jumlah = TextJmlBeli4.getText();  String total = TextTotalHarga4.getText();  String bayar = TextTotalDibayar1.getText();  String kembalian = TextTotalKembali1.getText();    if(barang.isEmpty()||kode.isEmpty()||harga.isEmpty()||nama.isEmpty()||jumlah.isEmpty()||total.isEmpty()||bayar.isEmpty()||kembalian.isEmpty()){  JOptionPane.showMessageDialog(null, "Tolong Isi Terlebih Dahulu!");  }else{  try{  Connection c = conn.getConnection();  String sql = "INSERT INTO datapembeli VALUES (?,?,?,?,?,?,?,?)";  PreparedStatement p = c.prepareStatement(sql);  p.setString(1, nama);  p.setString(2, barang);  p.setString(3, kode);  p.setString(4, harga);  p.setString(5, jumlah);  p.setString(6, total);  p.setString(7, bayar);  p.setString(8, kembalian);  p.executeUpdate();  p.close();  tabelHistory();  JOptionPane.showMessageDialog(null, "Data Telah Ditambahkan");  }catch(SQLException e){  e.printStackTrace();  }  }  }  private void TabelMouseReleased(java.awt.event.MouseEvent evt) {  // TODO add your handling code here:  Simpan.setText("Simpan");    }  koneksi conn; |

### **File Koneksi**

|  |
| --- |
| package modul5;  import java.sql.Connection;  import java.sql.DriverManager;  public class koneksii {  private static final String JDBC\_DRIVER = "com.mysql.cj.jdbc.Driver";  private static final String DB\_URL = "jdbc:mysql://localhost/pemvis5";  private static final String USER = "root";  private static final String PASS = "";    private Connection conn = null;    public koneksii(){  try {  Class.forName(JDBC\_DRIVER);  conn = DriverManager.getConnection(DB\_URL, USER, PASS);  System.out.println("Koneksi Succes");  } catch (Exception e) {  System.out.println("Koneksi Gagal");  }  }    public Connection getConnection(){  return conn;  }    } |

## **Hasil**

****

****

****

Program di atas adalah program untuk membuat data penjualan pada sebuah kasir. Program di atas menggunakan tabbed pane di mana terdapat tiga tab yang digunakan untuk menambah data barang dan menghitung total pembelian. Pada tab pertama terdapat beberapa komponen GUI yang terdiri dari label, textfield, tabel, dan button. Textfield digunakan untuk menginputkan nama barang, kode barang, dan harga barang yang kemudian akan tersimpan pada komponen tabel, sedangkan button digunakan untuk menghapus, mengupdate, dan menyimpan pada tabel. Tab kedua berisi komponen GUI yaitu label, button, combo box, internal frame, dan textfield. Combo box digunakan untuk menampilkan daftar barang yang terdapat pada tabel sebelumnya. Internal frame digunakan untuk menampilkan perhitungan total harga dan total kembalian jika data pembeli telah terisi. Selanjutnya pada tab ketiga berisi daftar riwayat pembelian. Tab ini berisi komponen label dan tabel saja. Tabel digunakan untuk menyimpan daftar pembelian jika tombol simpan pada tab sebelumnya diklik. Hal ini tentunya menggunakan database yang dihubungkan langsung dengan file pada Java.

# **BAB V PENUTUP**

## **Analisa**

Pengimplementasian GUI database pada Java memberikan dampak positif besar pada pengembangan aplikasi. GUI memudahkan pengguna dalam berinteraksi dengan database tanpa perlu pengetahuan mendalam tentang SQL. Elemen visual seperti formulir, tombol, dan tabel membuat pengelolaan data menjadi lebih intuitif, mempercepat proses pengembangan dengan menghilangkan kebutuhan untuk menulis perintah SQL secara manual. Hal ini mengurangi hambatan bagi pengguna awam dan mempercepat proses pengembangan aplikasi dengan mengeliminasi kebutuhan untuk memahami dan menulis perintah SQL secara manual.

Selain itu, GUI database pada Java membuka peluang untuk meningkatkan kinerja dan keamanan aplikasi. Integrasi fitur pengamanan dan validasi input dengan antarmuka grafis mengurangi risiko kesalahan pengguna, memastikan integritas data, dan memungkinkan pembaruan otomatis serta notifikasi visual untuk informasi real-time. Secara keseluruhan, GUI memberikan kontrol yang lebih baik atas operasi database, memungkinkan pengembang menyusun solusi yang lebih efisien dan responsif.

## **Kesimpulan**

1. GUI Database adalah antarmuka visual yang memungkinkan pengguna berinteraksi dengan database menggunakan elemen-elemen grafis seperti formulir, tombol, dan tabel.
2. GUI database menyediakan cara yang intuitif dan ramah pengguna untuk memasukkan, mengedit, dan mengambil informasi dari database, meningkatkan aksesibilitas aplikasi dan mengurangi hambatan bagi pengguna yang tidak terampil dalam koding SQL.
3. Program yang sudah diimplementasikan di atas menggunakan komponen GUI dengan mengkoneksikan program dengan database.