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

1.1 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.

1.2 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

2.1 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:

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

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

```
//Langkah pertama, import package terkait
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


/**
 *
 * @author Eja
 */
public class KoneksiDB {
  // driver JDBC driver dan database URL

  private final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
  private final String DB_URL = "jdbc:mysql://localhost/pis12noltiga";
  //  Database credentials
  private final String USER = "root";
  private final String PASS = "";
  private Connection conn = null;
```

```
  public void bukaKoneksi() {

    boolean flag = false;
    try {
      //Langkah ke-2: Registrasi JDBC
      Class.forName(JDBC_DRIVER);
    } catch (Exception e) {
      System.out.println(e.getMessage());
      flag = true;
    }
    if (!flag) {
      try {
        //Langkah ke-3: buka koneksi
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
      } catch (SQLException e) {
        System.out.println(e.getMessage());
      }
    }
  }

  public Connection getConn() {
    return conn;
  }
}
```

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

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author Eja
 */
public class DesaNinja {

  private String id_desa;
  private String nama;
  private String pemimpin;
  //ambil objek KoneksiDB karena membutuhkan connection dll
  private KoneksiDB kdb = new KoneksiDB();
```

```
    public DesaNinja(String id_desa, String desa, String pemimpin) {
        this.id_desa = id_desa;
        this.nama = desa;
        this.pemimpin = pemimpin;
    }
    public boolean masukkanData() throws SQLException {
        //deklarasi connection dan preparedStatement
        Connection dbConnection = null;
        PreparedStatement ps = null;
        int rowAffect = 0;

        String insertTableSQL = "INSERT INTO desa_ninja"
                + "(id_desa, nama, pemimpin) VALUES"
                + "(?,?,?)";
        try {
            //buka koneksi saat objek dari desa ninja dibentuk
            kdb.bukaKoneksi();
            //inisialisasi dbConnection dari objek Connection
            dbConnection = kdb.getConn();

            //Langkah ke 4 bagian 1
            ps = dbConnection.prepareStatement(insertTableSQL);
            ps.setString(1, this.id_desa);
            ps.setString(2, this.nama);
            ps.setString(3, this.pemimpin);
            //langkah 4: eksekusi query
            rowAffect = ps.executeUpdate();

        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            //langkah ke 6
            ps.close();
        }
        //langkah ke 5
        if (rowAffect > 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

Kelas untuk swing java.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 * @author Eja
 */
public class SimpleInsertGui extends JPanel implements ActionListener {

   //3 textfield di deklarasikan
   private JTextField id, nama, pemimpin;
   private JButton eksekusi;

   public SimpleInsertGui() {
      //langkah a, buat 3 textfield (beserta label) dan 1 tombol
      id = new JTextField(20);
      nama = new JTextField(20);
      pemimpin = new JTextField(20);

      //buat label untuk tiap textfield:
      JLabel l1 = new JLabel("No Registrasi Desa: ");
      JLabel l2 = new JLabel("Nama: ");
      JLabel l3 = new JLabel("Pemimpin: ");

      eksekusi = new JButton("Eksekusi");
      eksekusi.setActionCommand("oke");
      eksekusi.addActionListener(this);

      //tambahkan label, text dan button ke panel
      add(l1);add(id);
      add(l2);add(nama);
      add(l3);add(pemimpin);
      add(eksekusi);
   }
```

```
public void actionPerformed(ActionEvent e) {
   if(e.getActionCommand().equals("oke")){
      //langkah b, ambil semua isian textfield
      String n = nama.getText();
      String i = id.getText();
      String p = pemimpin.getText();

      //cek kalau isian kosong
      if (n.isEmpty() || i.isEmpty() || p.isEmpty()) {
         JOptionPane.showMessageDialog(this, "Ada Field yang Kosong",
               "Peringatan",JOptionPane.WARNING_MESSAGE);
      }else{
         //langkah c, buat objek desa Ninja
         DesaNinja dn = new DesaNinja(i, n, p);
         boolean status = false;
         try {
            //langkah d, panggil method yang merepresentasikan 'save()'
            status = dn.masukkanData();
         } catch (SQLException ex) {
            Logger.getLogger(SimpleInsertGui.class.getName()).log(Level.SEVERE, null, ex);
         }
         if(status){
           //langkah e, tampilkan pesan berhasil jika eksekusi berhasil
            JOptionPane.showMessageDialog(this, "Eksekusi Berhasil",
                  "Status",JOptionPane.INFORMATION_MESSAGE);
         }else{
           //langkah e, tampilkan pesan gagal jika eksekusi gagal
            JOptionPane.showMessageDialog(this, "Eksekusi Gagal",
                  "Status",JOptionPane.ERROR_MESSAGE);
         }
      }
   }
}

private static void createAndShowGUI() {

   //membuat frame
   JFrame frame = new JFrame("ButtonDemo");
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   //membuat content pane
   SimpleInsertGui newContentPane = new SimpleInsertGui();
   newContentPane.setOpaque(true);
   frame.setContentPane(newContentPane);
```

```
        //Memunculkan window
        frame.pack();
        frame.setVisible(true);
        frame.setLocationRelativeTo(null);
    }
    public static void main(String[] args) {

        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

2.2 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:

a) Gunakan Kelas "KoneksiDB" untuk membuka koneksi dan lain-lain

b) Deklarasikan JTable dengan modelnya pada kelas yang digunakan untuk penempatan swing.

c) 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).

d) Langkah ke-5 dimodifikasi dengan tidak menampilkan hasil ResultSet tapi memasukkannya ke dalam model JTable.

Lengkapnya:

```
/**
 *
 * @author Eja
 */

import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import java.awt.Dimension;
```

```
import java.awt.GridLayout;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.table.DefaultTableModel;

public class SimpleTableDemoDB extends JPanel {
  //langkah b
  private DefaultTableModel model;
  private JTable table;

  //langkah a
  private KoneksiDB kdb = new KoneksiDB();

  public SimpleTableDemoDB() {
    super(new GridLayout(1,0));

    model = new DefaultTableModel();
    table = new JTable(model);
    table.setPreferredScrollableViewportSize(new Dimension(500, 70));
    table.setFillsViewportHeight(true);

    //Membuat scroll pane pada table
    JScrollPane scrollPane = new JScrollPane(table);
    //menambah scroll pane dan table di panel
    add(scrollPane);
    try {
      isiDataTabel();
    } catch (SQLException ex) {
      Logger.getLogger(SimpleTableDemoDB.class.getName()).log(Level.SEVERE, null, ex);
    }
  }

  //langkah c
  private void isiDataTabel() throws SQLException {
    Connection dbConnection = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    String viewTableSQL = "SELECT * FROM desa_ninja";

    try {
      //buka koneksi saat objek dari desa ninja dibentuk
      kdb.bukaKoneksi();
      //inisialisasi dbConnection dari objek Connection
```

```
            dbConnection = kdb.getConn();

            //Langkah ke 4 bagian 1
            ps = dbConnection.prepareStatement(viewTableSQL);

            //langkah 4: eksekusi query
            rs = ps.executeQuery();

            //tentukan header tabel
            model.addColumn("Nomor Registrasi Desa");
            model.addColumn("Nama Desa");
            model.addColumn("Pemimpin");

            //langkah ke 5
            //ekstrak data dari ResultSet dan masukkan ke table
            while (rs.next()) {
               Object[] o = new Object[3];
               //langkah d
               o[0] = rs.getString(1);
               o[1] = rs.getString(2);
               o[2] = rs.getString(3);
               model.addRow(o);
            }

         } catch (Exception e) {
            System.out.println(e.getMessage());
         } finally {
            //langkah ke-6
            ps.close();
         }
      }

      private static void createAndShowGUI() {
         //Membuat frame
         JFrame frame = new JFrame("TableDemo");
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

         //Membuat content pane
         SimpleTableDemoDB newContentPane = new SimpleTableDemoDB();
         newContentPane.setOpaque(true);
         frame.setContentPane(newContentPane);

         //menampilkan window
         frame.pack();
         frame.setVisible(true);
      }
```

```
    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

2.3 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:

a) Gunakan Kelas "KoneksiDB" untuk membuka koneksi dan lain-lain

b) Deklarasikan JComboBox pada kelas yang digunakan untuk penempatan swing.

c) 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).

d) Panggil method pengaksesan data tersebut pada konstruktor, sehingga saat ditampilkan pertama kali, combo box sudah menampilkan isi dari database.

e) Langkah ke-5 dimodifikasi dengan tidak menampilkan hasil ResultSet tapi memasukkannya ke dalam komponen JComboBox.

Lengkapnya:

```
/**
 *
 * @author Eja
 */
import java.util.logging.Level;
import java.util.logging.Logger;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.swing.*;

public class SimpleComboBoxDemo extends JPanel implements ActionListener {
```

```
//langkah a
private KoneksiDB kdb = new KoneksiDB();
//langkah b
private JComboBox kombo;

public SimpleComboBoxDemo() {
   super();
   kombo = new JComboBox();

   try {
      //langkah d
      this.isiDataKombo();
   } catch (SQLException ex) {
      Logger.getLogger(SimpleComboBoxDemo.class.getName()).log(Level.SEVERE, null, ex);
   }
   kombo.addActionListener(this);

   add(kombo);
}

/** Listener combo box */
public void actionPerformed(ActionEvent e) {
   JComboBox cb = (JComboBox) e.getSource();
   int pil = cb.getSelectedIndex();
   if (pil == 0) {
      System.out.println("Tidak ada yang terpilih");
   } else {
      System.out.println(cb.getSelectedItem());
   }
}
//langkah c
private void isiDataKombo() throws SQLException {
   Connection dbConnection = null;
   PreparedStatement ps = null;
   ResultSet rs = null;

   String viewTableSQL = "SELECT * FROM desa_ninja";
   try {
      //buka koneksi saat objek dari desa ninja dibentuk
      kdb.bukaKoneksi();
      //inisialisasi dbConnection dari objek Connection
      dbConnection = kdb.getConn();

      //Langkah ke 4 bagian 1
      ps = dbConnection.prepareStatement(viewTableSQL);
```

```
            //langkah 4: eksekusi query
            rs = ps.executeQuery();

            //langkah ke 5
            //ekstrak data dari ResultSet

            while (rs.next()) {
              //langkah e
              kombo.addItem(rs.getString(2));
            }
            kombo.setSelectedIndex(0);

        } catch (Exception e) {
            System.out.println(e.getMessage());
        } finally {
            //langkah ke-6
            ps.close();
        }
    }
    private static void createAndShowGUI() {
        //membuat frame
        JFrame frame = new JFrame("ComboBoxDemo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        //membuat content pane
        JComponent newContentPane = new SimpleComboBoxDemo();
        newContentPane.setOpaque(true); //content panes must be opaque
        frame.setContentPane(newContentPane);

        //Memunculkan window
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                createAndShowGUI();
            }
        });
    }
}
```

NUH H.

BAB III
TUGAS PENDAHULUAN

3.1 Soal

1. Jelaskan apa yang dimaksud database ? Berikan contohnya!
2. Sebutkan dan jelaskan minimal 3 jenis database yang anda ketahui!
3. Library apa saja yang diperlukan untuk membangun sebuah GUI database pada Java?

3.2 Jawab

1. Database adalah kumpulan data yang terstruktur dan terorganisir dengan baik, yang dapat diakses, dikelola, dan diperbarui secara efisien. Data dalam database disimpan dalam tabel, yang terdiri dari baris dan kolom. Contoh database yaitu MySQL dan MongoDB
2. a) Relasional Database (RDBMS) : jenis database yang menggunakan model rasional dan tabel untuk menyimpan data.
   b) NoSQL Database : jenis database yang dirancang untuk menangani volume data yang besar dan fleksibel dalam skema data mereka.
   c) Graph Database : jenis database yang digunakan untuk menyimpan data yang memiliki struktur berbentuk graf, dengan simpul (node) dan tepi (edge)
3. Terdapat beberapa library yang umum digunakan untuk membangun sebuah GUI database pada Java, antara lain:
   a) Swing
   b) JavaFx
   c) Java Database Connectivity (JDBC)
   d) Java Persistance API (JPA)
   e) Apache Derby

BAB IV

IMPLEMENTASI

4.1 Source Code

4.1.1 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;
```

4.1.2 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;
    }

}
```

4.2 Hasil


PROGRAM KASIR SEDERHANA
Input Barang
Pembelian
Riwayat
Input Data Barang
Nama Barang : Kotak Pensil
Kode Barang : 4
Harga Barang : 15000
Simpan
Hapus
Nama Barang | Kode Barang | Harga Barang
Buku Tulis | 1 | 5000
Pulpen | 2 | 4000
Pensil | 3 | 3000
Reset


PROGRAM KASIR SEDERHANA
Input Barang
Pembelian
Riwayat
Data Pembelian
Data Pembeli
Nama Pembeli : Tasya
Pilih Barang : Buku Tulis
Kode Barang : 1
Harga Barang : 5000
Next
Total Harga
Jumlah Pembelian : 2
Total Harga : 10000
Hitung
Total Bayar
Total Dibayar : 20000
Total Kembalian : 10000
Hitung
Simpan

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

5.1 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.

5.2 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.