

LAPORAN PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK



NAMA : ENSTIKA ELINDASARI

NIM : 24104410006

PERIODE : SEMESTER GENAP 2024/2025

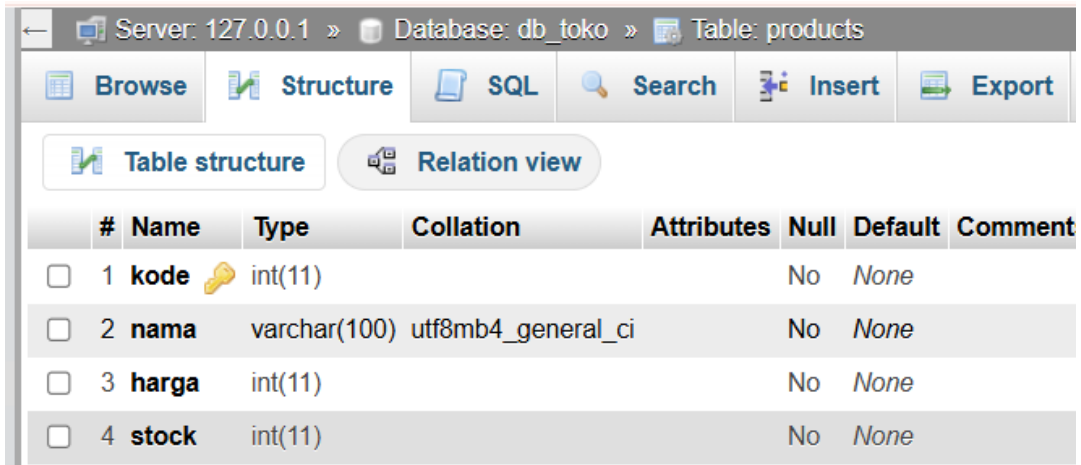
PROGRAM STUDY TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVRSITAS ISLAM BALITAR

2025

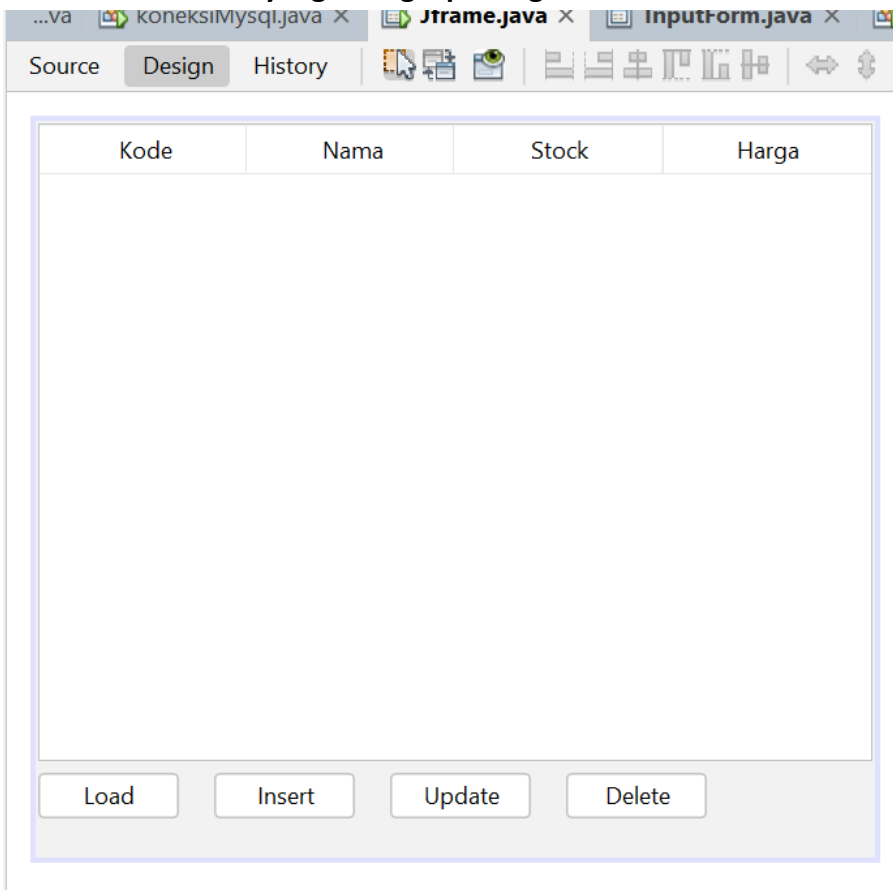
1. Pembuatan Database dengan nama db_toko dan table product



The screenshot shows the MySQL Workbench interface. The top bar indicates the connection to 'Server: 127.0.0.1', the selected database is 'Database: db_toko', and the selected table is 'Table: products'. Below the top bar are tabs for 'Browse', 'Structure', 'SQL', 'Search', 'Insert', and 'Export'. The 'Structure' tab is active, showing the 'Table structure' view. The table structure is displayed in a table with columns: #, Name, Type, Collation, Attributes, Null, Default, and Comment. The table 'products' has four columns: 'kode' (int(11), primary key), 'nama' (varchar(100), utf8mb4_general_ci), 'harga' (int(11)), and 'stock' (int(11)).

#	Name	Type	Collation	Attributes	Null	Default	Comment
1	kode	int(11)			No	None	
2	nama	varchar(100)	utf8mb4_general_ci		No	None	
3	harga	int(11)			No	None	
4	stock	int(11)			No	None	

2. Pembuatan JFrame yang dilengkapi dengan 4 JButton dan 1 JTable

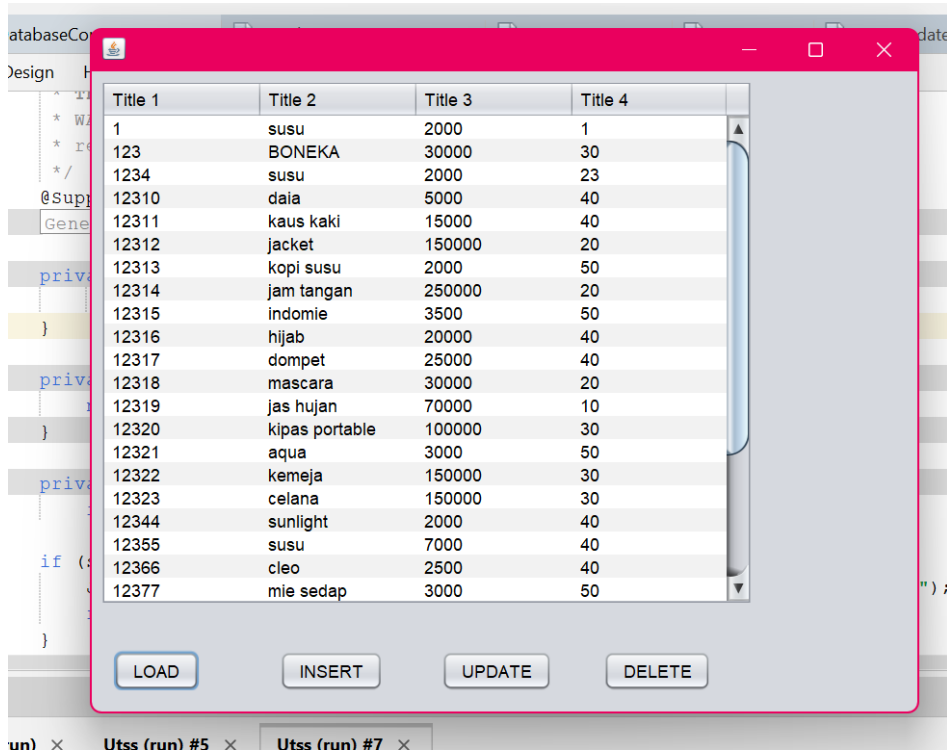


The screenshot shows a Java Swing JFrame window. The window has a title bar with three tabs: 'koneksiMysql.java', 'JFrame.java', and 'InputForm.java'. The 'JFrame.java' tab is active. The window contains a JTable with four columns: 'Kode', 'Nama', 'Stock', and 'Harga'. Below the table are four buttons: 'Load', 'Insert', 'Update', and 'Delete'.

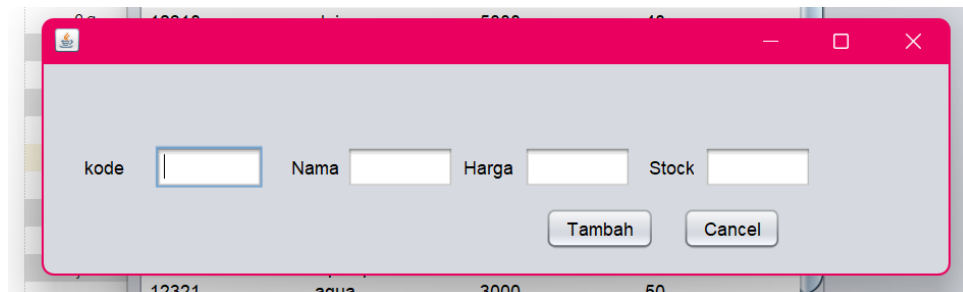
Kode	Nama	Stock	Harga
------	------	-------	-------

Load Insert Update Delete

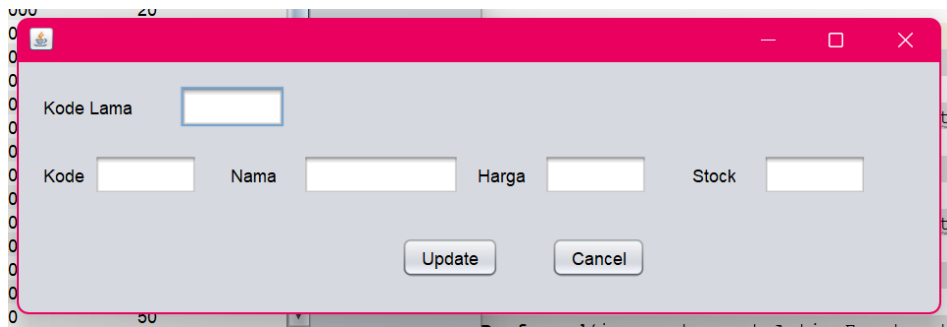
- Load: seleksi semua data di tabel produk



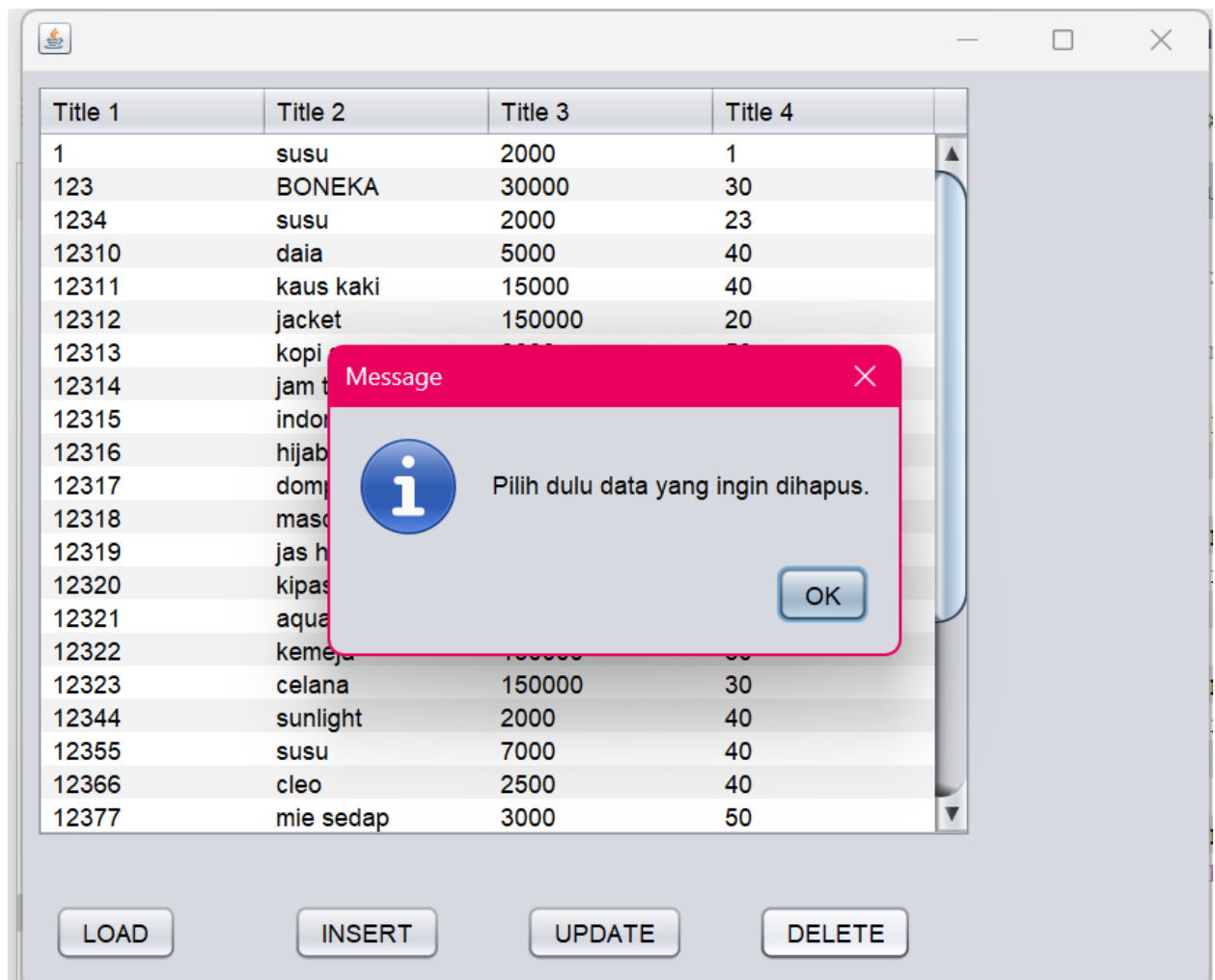
- Insert: memasukkan satu produk baru



- Update: menyeleksi salah satu isi tabel dan melakukan update



- Delete: menyeleksi salah satu isi tabel dan melakukan delete



3. Pada project anda harus menerapkan enkapsulasi, Inheritance, try-catch SQLException, GUI Swing

a. Enkapsulasi

```
17      @Override
18      public void simpanKeDatabase() {
19          try {
20              Connection conn = DatabaseConnection.getConnection();
21              PreparedStatement stmt = conn.prepareStatement(
22                  "INSERT INTO product (kode, nama, harga, stok) VALUES (?, ?, ?, ?)"
23              );
24              stmt.setInt(1, getKode());
25              stmt.setString(2, getNama());
26              stmt.setInt(3, getHarga());
27              stmt.setInt(4, getStok());
28              stmt.executeUpdate();
29              System.out.println("Produk berhasil disimpan ke database.");
30          } catch (SQLException e) {
31              System.err.println("Gagal menyimpan produk: " + e.getMessage());
32          }
33      }
34
35      Object getStok() {
36          throw new UnsupportedOperationException("Not supported yet."); // Generated from nk
37      }
38
39      Object getStok() {
40          throw new UnsupportedOperationException("Not supported yet."); // Generated from nk
41      }
42  }
```

b. Inheritance

```
public class Product extends AbstrackProduk {

    public Product(int kode, String nama, int harga, int stok) {
        super(kode, nama, harga, stok);
    }

    @Override
    public void simpanKeDatabase() {
        try {
            Connection conn = DatabaseConnection.getConnection();
            PreparedStatement stmt = conn.prepareStatement(
                "INSERT INTO product (kode, nama, harga, stok) VALUES (?, ?, ?, ?)"
            );
            stmt.setInt(1, getKode());
            stmt.setString(2, getNama());
            stmt.setInt(3, getHarga());
            stmt.setInt(4, getStok());
            stmt.executeUpdate();
            System.out.println("Produk berhasil disimpan ke database.");
        } catch (SQLException e) {
            System.err.println("Gagal menyimpan produk: " + e.getMessage());
        }
    }
}
```

c. try-catch SQLException

```

7  @Override
8  public void simpanKeDatabase() {
9      try {
10         Connection conn = DatabaseConnection.getConnection();
11         PreparedStatement stmt = conn.prepareStatement(
12             "INSERT INTO product (kode, nama, harga, stock) VALUES (?, ?, ?, ?)"
13         );
14         stmt.setInt(1, getKode());
15         stmt.setString(2, getNama());
16         stmt.setInt(3, getHarga());
17         stmt.setInt(4, getStok());
18         stmt.executeUpdate();
19         System.out.println("Produk berhasil disimpan ke database.");
20     } catch (SQLException e) {
21         System.err.println("Gagal menyimpan produk: " + e.getMessage());
22     }
23 }

```

d. GUI Swing

4. Kerjakan codingnya dan tulis laporannya disertai penjelasan coding dan hasil screenshot outputnya.

a. **Class Kasir2**

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

```

import java.sql.SQLException;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import java.util.Vector;
import javax.swing.table.DefaultTableModel;
import java.sql.Statement;
import java.sql.ResultSet;
/**
 *
 * @author ASUS
 */
public class Kasir2 extends javax.swing.JFrame {

    /**
     * Creates new form Kasir2
     */
    public Kasir2() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jScrollPane1 = new javax.swing.JScrollPane();
        jTable1 = new javax.swing.JTable();
        LOAD = new javax.swing.JButton();
        INSERT = new javax.swing.JButton();
        UPDATE = new javax.swing.JButton();
        DELETE = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

```

```

jTable1.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null},
        {null, null, null, null}
    },
    new String [] {
        "Title 1", "Title 2", "Title 3", "Title 4"
    }
));
jScrollPane1.setViewportView(jTable1);

LOAD.setText("LOAD");
LOAD.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        LOADActionPerformed(evt);
    }
});

INSERT.setText("INSERT");
INSERT.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        INSERTActionPerformed(evt);
    }
});

UPDATE.setText("UPDATE");
UPDATE.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        UPDATEActionPerformed(evt);
    }
});

DELETE.setText("DELETE");
DELETE.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        DELETEActionPerformed(evt);
    }
});

```



```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 31,  
Short.MAX_VALUE)
```

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BAS  
ELINE)
```

```
    .addComponent(LOAD)  
    .addComponent(INSERT)  
    .addComponent(UPDATE)  
    .addComponent(DELETE))  
    .addGap(15, 15, 15))
```

```
);
```

```
    pack();
```

```
}// </editor-fold>
```

```
private void INSERTActionPerformed(java.awt.event.ActionEvent evt) {  
    new FormInput().setVisible(true);  
}
```

```
private void UPDATEActionPerformed(java.awt.event.ActionEvent evt) {  
    new FormUpdate().setVisible(true);  
}
```

```
private void DELETEActionPerformed(java.awt.event.ActionEvent evt) {  
    int selectedRow = jTable1.getSelectedRow();
```

```
    if (selectedRow == -1) {  
        JOptionPane.showMessageDialog(this, "Pilih dulu data yang ingin  
dihapus.");  
        return;  
    }
```

```
    int confirm = JOptionPane.showConfirmDialog(this, "Yakin ingin menghapus  
data ini?", "Konfirmasi Hapus", JOptionPane.YES_NO_OPTION);
```

```
    if (confirm == JOptionPane.YES_OPTION) {  
        try {
```

```

        int id = Integer.parseInt(jTable1.getValueAt(selectedRow, 0).toString()); //
        Ambil ID dari kolom pertama (index 0)
        Connection conn = DatabaseConnection.getConnection();
        String sql = "DELETE FROM product WHERE kode = ?";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setInt(1, id);
        stmt.executeUpdate();

        // Hapus dari JTable juga
        DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
        model.removeRow(selectedRow);

        JOptionPane.showMessageDialog(this, "Data berhasil dihapus!");
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(this, "Gagal hapus data: " +
        e.getMessage());
    }
}

private void LOADActionPerformed(java.awt.event.ActionEvent evt) {

try (Connection conn = DatabaseConnection.DatabaseConnection();
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM products")) {

    DefaultTableModel model = (DefaultTableModel) jTable1.getModel();
    model.setRowCount(0);
    while (rs.next()) {
        Vector<Object> row = new Vector<>();
        row.add(rs.getInt("kode"));
        row.add(rs.getString("nama"));
        row.add(rs.getInt("harga"));
        row.add(rs.getInt("stock"));
        model.addRow(row);
    }
}

```

```

    } catch (SQLException ex) {
        JOptionPane.showMessageDialog(this, "Error saat memuat data: " +
ex.getMessage(), "Database Error", JOptionPane.ERROR_MESSAGE);
        ex.printStackTrace();
    }
}
/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code
(optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default
look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Kasir2.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Kasir2.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Kasir2.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);

```

```

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Kasir2.class.getName()).log(java.util.logging.L
evel.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new Kasir2().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton DELETE;
private javax.swing.JButton INSERT;
private javax.swing.JButton LOAD;
private javax.swing.JButton UPDATE;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTable jTable1;
// End of variables declaration
}

```

Penjelasan :

➤ **Struktur Utama:**

- **Class Kasir2:** Merupakan turunan dari JFrame (window utama aplikasi).
- **Komponen GUI:**
 - Tabel (jTable1) untuk menampilkan data produk.
 - Tombol: LOAD, INSERT, UPDATE, DELETE.

➤ **Fungsi Tombol**

- **LOAD:** Mengambil seluruh data dari tabel products di database, lalu menampilkannya ke dalam jTable1. Data yang diambil berupa kolom kode, nama, harga, dan stock. Jika terjadi error saat pengambilan data, akan muncul pesan error.
- **INSERT:** Membuka form input baru (FormInput) untuk menambah data produk ke database.

- **UPDATE:** Membuka form update (FormUpdate) untuk mengubah data produk yang sudah ada.
- **DELETE:** Menghapus baris yang dipilih di tabel.

Proses:

- Cek apakah ada baris yang dipilih.
- Konfirmasi penghapusan.
- Menghapus data dari database berdasarkan kode (diasumsikan sebagai primary key).
- Menghapus baris dari tabel tampilan.
- Menampilkan pesan sukses/gagal.

➤ **Koneksi Database**

- Menggunakan class DatabaseConnection (tidak terlihat di kode ini, diasumsikan menyediakan method getConnection()).
- Query SQL yang digunakan:
 - SELECT * FROM products untuk load data.
 - DELETE FROM product WHERE kode = ? untuk hapus data berdasarkan kode produk.

➤ **Fitur Lain**

- Menggunakan DefaultTableModel untuk mengelola data pada JTable.
- Penanganan error menggunakan JOptionPane untuk menampilkan pesan ke pengguna.

➤ **Kesimpulan**

Kode ini adalah aplikasi kasir sederhana berbasis Java Swing yang memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada data produk yang tersimpan di database. Antarmuka yang disediakan memudahkan untuk melihat, menambah, mengubah, dan menghapus data produk secara interaktif melalui tabel dan tombol-tombol yang tersedia.

b. Kelas Product

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

public class Product extends AbstrackProduk {

    public Product(int kode, String nama, int harga, int stok) {
        super(kode, nama, harga, stok);
    }
}
```

```

@Override
public void simpanKeDatabase() {
    try {
        Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO product (kode, nama, harga, stock) VALUES (?, ?, ?, ?)"
        );
        stmt.setInt(1, getKode());
        stmt.setString(2, getNama());
        stmt.setInt(3, getHarga());
        stmt.setInt(4, getStok());
        stmt.executeUpdate();
        System.out.println("Produk berhasil disimpan ke database.");
    } catch (SQLException e) {
        System.err.println("Gagal menyimpan produk: " + e.getMessage());
    }
}

```

```

Object gettock() {
    throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}

```

```

Object getStock() {
    throw new UnsupportedOperationException("Not supported yet."); //
Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}
}

```

Penjelasan :

1. Inheritance

- Product merupakan subclass dari AbstrackProduk (kemungkinan typo, seharusnya AbstractProduk).
- Dengan kata lain, Product mewarisi atribut dan method dari class AbstrackProduk.

2. Constructor

```
public class Product extends AbstrackProduk {

    public Product(int kode, String nama, int harga, int stok) {
        super(kode, nama, harga, stok);
    }
}
```

- Constructor menerima parameter kode, nama, harga, dan stok.
- Kemudian, constructor superclass (AbstrackProduk) dipanggil untuk menginisialisasi atribut-atribut tersebut.

3. Method Override

```
public void simpanKeDatabase() {
    try {
        Connection conn = DatabaseConnection.getConnection();
        PreparedStatement stmt = conn.prepareStatement(
            "INSERT INTO product (kode, nama, harga, stock) VALUES (?, ?, ?, ?)"
        );
        stmt.setInt(1, getKode());
        stmt.setString(2, getNama());
        stmt.setInt(3, getHarga());
        stmt.setInt(4, getStok());
        stmt.executeUpdate();
        System.out.println("Produk berhasil disimpan ke database.");
    } catch (SQLException e) {
        System.err.println("Gagal menyimpan produk: " + e.getMessage());
    }
}
```

- Method ini digunakan untuk **menyimpan data produk ke database**.
- Menggunakan koneksi database dari DatabaseConnection.getConnection().
- Menggunakan PreparedStatement untuk mengeksekusi query INSERT ke tabel product.
- Nilai-nilai yang diinsert diambil dari getter (getKode(), getNama(), dll) yang diasumsikan ada di superclass.
- Jika berhasil, menampilkan pesan sukses di console; jika gagal, menampilkan pesan error.

4. Method gettock() dan getStock()

```
Object gettock() {
    throw new UnsupportedOperationException("Not supported yet."); /
}

Object getStock() {
    throw new UnsupportedOperationException("Not supported yet."); /
}
```


- Kedua method ini **belum diimplementasikan** dan akan melempar exception jika dipanggil.
- Kemungkinan method ini dibuat oleh IDE secara otomatis, namun belum diisi fungsinya.

c. FormUpdate

```
import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import javax.swing.JOptionPane;

/**
 *
 * @author ASUS
 */
public class FormUpdate extends javax.swing.JFrame {

    /**
     * Creates new form FormUpdate
     */
    public FormUpdate() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
}
```

```

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    TfKodeBaru = new javax.swing.JTextField();
    tfNama = new javax.swing.JTextField();
    jLabel3 = new javax.swing.JLabel();
    tfHarga = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    tfStock = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jLabel5 = new javax.swing.JLabel();
    TfKodeLama = new javax.swing.JTextField();
    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    jLabel1.setText("Kode");
    jLabel2.setText("Nama");
    TfKodeBaru.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            TfKodeBaruActionPerformed(evt);
        }
    });
    jLabel3.setText("Harga");
    jLabel4.setText("Stock");

```

```

jButton1.setText("Update");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
jButton2.setText("Cancel");
jButton2.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton2ActionPerformed(evt);
    }
});
jLabel5.setText("Kode Lama");
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .add(layout.createSequentialGroup()
                    .add(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .add(layout.createSequentialGroup()
                            .addGap(260, 260, 260)
                            .addComponent(jButton1)
                            .addGap(35, 35, 35)
                            .addComponent(jButton2))
                        .add(layout.createSequentialGroup()
                            .addGap(17, 17, 17)

```

```

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel5)
        .addGap(32, 32, 32)
        .addComponent(TfKodeLama, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel1)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(TfKodeBaru, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(22, 22, 22)
        .addComponent(jLabel2)
        .addGap(18, 18, 18)
        .addComponent(tfNama, javax.swing.GroupLayout.PREFERRED_SIZE, 107,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(tfHarga, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(30, 30, 30)
        .addComponent(jLabel4)
        .addGap(18, 18, 18)

```

```

        .addComponent(tfStock, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))))
        .addContainerGap(50, Short.MAX_VALUE)
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(16, 16, 16)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel5)
                .addComponent(TfKodeLama, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel1)
                .addComponent(jLabel2)
                .addComponent(TfKodeBaru, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(tfNama, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3)
                .addComponent(tfHarga, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel4)
                .addComponent(tfStock, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 28,
Short.MAX_VALUE)

```

```

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jButton1)
            .addComponent(jButton2))
        .addGap(24, 24, 24))
    );

    pack();
} // </editor-fold>

```

```

private void TfKodeBaruActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
}

```

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    this.dispose();
}

```

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int kodeBaru = Integer.parseInt(TfKodeBaru.getText());
    int kodeLama = Integer.parseInt(TfKodeLama.getText());
    String nama = tfNama.getText();
    int harga = Integer.parseInt(tfHarga.getText());
    int stok = Integer.parseInt(tfStock.getText());
}

```

```

try {

```

```

    Connection conn = DatabaseConnection.getConnection(); // pastikan class koneksi sudah
sesuai

    String sql = "UPDATE products SET kode = ?, nama = ?, harga = ?, stock = ? WHERE kode = ?";

    PreparedStatement stmt = conn.prepareStatement(sql);

    stmt.setInt(1, kodeBaru);

    stmt.setString(2, nama);

    stmt.setInt(3, harga);

    stmt.setInt(4, stok);

    stmt.setInt(5, kodeLama);

    int rowsUpdated = stmt.executeUpdate();

    if (rowsUpdated > 0)
    {
        JOptionPane.showMessageDialog(this, "Data berhasil diupdate.");
        this.dispose();
    } else {
        JOptionPane.showMessageDialog(this, "Data gagal diupdate. Kode lama tidak ditemukan.");
    }

    stmt.close();

    conn.close();

} catch (NumberFormatException e) {
    JOptionPane.showMessageDialog(this, "Harga dan Stock harus berupa angka.");
} catch (SQLException e) {
    JOptionPane.showMessageDialog(this, "Kesalahan database: " + e.getMessage());
}
}

```

```

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
    * For details see
    http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(FormUpdate.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(FormUpdate.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

    } catch (IllegalAccessException ex) {

```



```
java.util.logging.Logger.getLogger(FormUpdate.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
java.util.logging.Logger.getLogger(FormUpdate.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
```

```
    }
```

```
//</editor-fold>
```

```
/* Create and display the form */
```

```
java.awt.EventQueue.invokeLater(new Runnable() {
```

```
    public void run() {
```

```
        new FormUpdate().setVisible(true);
```

```
    }
```

```
});
```

```
}
```

```
// Variables declaration - do not modify
```

```
private javax.swing.JTextField TfKodeBaru;
```

```
private javax.swing.JTextField TfKodeLama;
```

```
private javax.swing.JButton jButton1;
```

```
private javax.swing.JButton jButton2;
```

```
private javax.swing.JLabel jLabel1;
```

```
private javax.swing.JLabel jLabel2;
```

```
private javax.swing.JLabel jLabel3;
```

```
private javax.swing.JLabel jLabel4;
```

```

private javax.swing.JLabel jLabel5;
private javax.swing.JTextField tfHarga;
private javax.swing.JTextField tfNama;
private javax.swing.JTextField tfStock;

// End of variables declaration
}

```

Penjelasan:

➤ **Komponen Utama Form**

- **Label dan TextField:**

- TfKodeLama: untuk memasukkan kode produk lama (yang akan dicari di database).
- TfKodeBaru: untuk memasukkan kode produk baru (jika ingin mengubah kode produk).
- tfNama: untuk memasukkan nama produk.
- tfHarga: untuk memasukkan harga produk.
- tfStock: untuk memasukkan stok produk.

- **Button:**

- **Update (jButton1):** untuk menyimpan perubahan ke database.
- **Cancel (jButton2):** untuk menutup form tanpa menyimpan perubahan.

➤ **Proses Update Data**

- Ketika tombol **Update** ditekan:

1. Data dari seluruh field diambil dan dikonversi ke tipe data yang sesuai.
2. Dibuat koneksi ke database melalui `DatabaseConnection.getConnection()`.
3. Disiapkan query SQL:

```
String sql = "UPDATE products SET kode = ?, nama = ?, harga = ?, stock = ? WHERE kode = ?";
```

4. Nilai field di-set ke parameter query secara berurutan.

5. Query dieksekusi dengan executeUpdate().
6. Jika ada baris yang terupdate, tampilkan pesan sukses dan form ditutup; jika tidak, tampilkan pesan gagal (kode lama tidak ditemukan).
7. Penanganan error:
 - Jika input harga/stok bukan angka, tampilkan pesan error.
 - Jika terjadi error database, tampilkan pesan error.

➤ **Proses Cancel**

- Jika tombol Cancel ditekan, form langsung ditutup tanpa melakukan perubahan.

➤ **Inisialisasi dan Tampilan**

- Komponen GUI diinisialisasi di method initComponents().
- Form dapat dijalankan langsung melalui method main().

➤ **Kesimpulan**

Class FormUpdate adalah form GUI untuk mengedit data produk di database. Pengguna dapat memasukkan kode lama, kode baru, nama, harga, dan stok produk, lalu menekan tombol Update untuk menyimpan perubahan ke database. Jika update berhasil, akan muncul notifikasi sukses; jika gagal, notifikasi error akan tampil. Form ini juga menyediakan tombol Cancel untuk keluar tanpa perubahan.

d. FormInsert

```
import UTS.DatabaseConnection;
```

```
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;
```

```
import javax.swing.JOptionPane;
```

```
/*
```

```
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
```

```
* Click nbfs://nbhost/SystemFileSystem/Templates/GuiForms/JFrame.java to edit this template
```

```
*/
```

```
/**
```

```

*
* @author ASUS
*/

public class FormInput extends javax.swing.JFrame {

    /**
     * Creates new form FormInput
     */
    public FormInput() {
        initComponents();
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
    }

```

```

txtKode = new javax.swing.JTextField();
txtNama = new javax.swing.JTextField();
txtHarga = new javax.swing.JTextField();
txtStock = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
jButton2 = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel1.setText("kode");
jLabel2.setText("Nama");
jLabel3.setText("Harga");
jLabel4.setText("Stock");
txtKode.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtKodeActionPerformed(evt);
    }
});
txtNama.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        txtNamaActionPerformed(evt);
    }
});
jButton1.setText("Tambah");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
}

```

```

});

jButton2.setText("Cancel");

jButton2.addActionListener(new java.awt.event.ActionListener() {

    public void actionPerformed(java.awt.event.ActionEvent evt) {

        jButton2ActionPerformed(evt);

    }

});

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel1Layout.createSequentialGroup()

            .addGap(27, 27, 27)

            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 40,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(txtKode, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addGap(18, 18, 18)

            .addComponent(jLabel2)

            .addGap(5, 5, 5)

            .addComponent(txtNama, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

            .addComponent(jLabel3)

            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

        .addComponent(txtHarga, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(11, 11, 11)

        .addComponent(jLabel4)

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

        .addComponent(txtStock, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)

        .addContainerGap(90, Short.MAX_VALUE))

        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup())

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButton1)

        .addGap(18, 18, 18)

        .addComponent(jButton2)

        .addGap(110, 110, 110))

    );

jPanel1Layout.setVerticalGroup(

    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(jPanel1Layout.createSequentialGroup()

            .addGap(53, 53, 53)

            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

                .addComponent(jLabel1)

                .addComponent(jLabel2)

                .addComponent(jLabel3)

                .addComponent(jLabel4)

```

```

        .addComponent(txtKode, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(txtNama, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(txtHarga, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

        .addComponent(txtStock, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

        .addComponent(jButton2)

        .addComponent(jButton1))

.addContainerGap(16, Short.MAX_VALUE))

);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);

layout.setHorizontalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

);

layout.setVerticalGroup(

    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

);

```



```

        pack();
    } // </editor-fold>

    private void txtKodeActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        try (Connection conn = DatabaseConnection.DatabaseConnection()) {
            String sql = "INSERT INTO products (kode, nama, harga, stock) VALUES (?, ?, ?, ?)";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                int kode = Integer.parseInt(txtKode.getText());
                String nama = txtNama.getText();
                int harga = Integer.parseInt(txtHarga.getText());
                int stock = Integer.parseInt(txtStock.getText());
                pstmt.setInt(1, kode);
                pstmt.setString(2, nama);
                pstmt.setInt(3, harga);
                pstmt.setInt(4, stock);
                int rowsAffected = pstmt.executeUpdate();
                if (rowsAffected > 0) {
                    JOptionPane.showMessageDialog(this, "Data berhasil ditambahkan!", "Sukses",
JOptionPane.INFORMATION_MESSAGE);
                    this.dispose();
                } else {
                    JOptionPane.showMessageDialog(this, "Gagal menambahkan data.", "Error",
JOptionPane.ERROR_MESSAGE);
                }
            }
        }
    }
}

```

```

    } catch (SQLException | NumberFormatException ex) {

        JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);

        ex.printStackTrace();

    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

    this.dispose();

}

private void txtNamaActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

}

/**
 * @param args the command line arguments
 */

public static void main(String args[]) {

    /* Set the Nimbus look and feel */

    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */

    try {

        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {

            if ("Nimbus".equals(info.getName())) {

                javax.swing.UIManager.setLookAndFeel(info.getClassName());

```

```

        break;
    }
}
} catch (ClassNotFoundException ex) {

```

```

java.util.logging.Logger.getLogger(FormInput.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);

```

```

    } catch (InstantiationException ex) {

```

```

java.util.logging.Logger.getLogger(FormInput.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);

```

```

    } catch (IllegalAccessException ex) {

```

```

java.util.logging.Logger.getLogger(FormInput.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);

```

```

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

```

```

java.util.logging.Logger.getLogger(FormInput.class.getName()).log(java.util.logging.Level.SEVERE
, null, ex);

```

```

    }

```

```

//</editor-fold>

```

```

/* Create and display the form */

```

```

java.awt.EventQueue.invokeLater(new Runnable() {

```

```

    public void run() {

```

```

        new FormInput().setVisible(true);

```

```

    }

```

```

});

```

```

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField txtHarga;
private javax.swing.JTextField txtKode;
private javax.swing.JTextField txtNama;
private javax.swing.JTextField txtStock;

// End of variables declaration
}

```

Penjelasan:

1. Tujuan Kode

FormInput adalah **form GUI berbasis Java Swing** yang digunakan untuk **menambah data produk baru ke dalam database**. Form ini merupakan bagian dari aplikasi CRUD (Create, Read, Update, Delete) untuk manajemen produk.

2. Komponen Utama

- **Label dan TextField**
 - **txtKode**: untuk input kode produk (angka/integer).
 - **txtNama**: untuk input nama produk (teks).
 - **txtHarga**: untuk input harga produk (angka/integer).
 - **txtStock**: untuk input stok produk (angka/integer).
- **Button**

- **Tambah (jButton1)**: untuk menambah data produk ke database.
- **Cancel (jButton2)**: untuk menutup form tanpa menyimpan data.

3. Alur Kerja Tombol Tambah

Saat tombol **Tambah** ditekan:

1. Mengambil Data Input

- Mengambil nilai dari masing-masing text field, lalu mengonversi ke tipe data yang sesuai (kode, harga, dan stok menjadi integer).

2. Koneksi ke Database

- Membuka koneksi ke database menggunakan `DatabaseConnection.DatabaseConnection()`.

3. Menyiapkan dan Menjalankan Query SQL

- Menyiapkan query SQL:

```
String sql = "INSERT INTO products (kode, nama, harga, stock) VALUES (?, ?, ?, ?)";
```

- Mengisi parameter query dengan data input.
- Menjalankan query dengan `executeUpdate()`.

4. Feedback ke Pengguna

- Jika data berhasil ditambahkan (`rowsAffected > 0`), tampilkan pesan sukses dan form ditutup.
- Jika gagal, tampilkan pesan error.

5. Penanganan Error

- Jika terjadi kesalahan input (misal: input bukan angka) atau error database, tampilkan pesan error dengan detail.

4. Alur Kerja Tombol Cancel

- Jika tombol **Cancel** ditekan, form langsung ditutup tanpa menyimpan data.

5. Inisialisasi dan Tampilan

- Semua komponen GUI diinisialisasi dalam `initComponents()`.

- Form dapat dijalankan langsung melalui method main().

Kesimpulan

FormInput adalah form input data produk berbasis Java Swing yang berfungsi untuk menambah data produk ke dalam database. Form ini menyediakan field untuk kode, nama, harga, dan stok produk, serta tombol untuk menambah data atau membatalkan proses input.

e. DatabaseConnection

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

    private static final String URL = "jdbc:mysql://localhost:3306/db_toko";

    private static final String USER = "root"; // Ganti dengan username MySQL Anda

    private static final String PASSWORD = ""; // Ganti dengan password MySQL Anda

    private static Connection connection = null;

    // Metode untuk mendapatkan koneksi

    public static Connection DatabaseConnection() throws SQLException {

        if (connection == null || connection.isClosed()) {

            try {

                // Load driver

                Class.forName("com.mysql.cj.jdbc.Driver");

                // Buat koneksi

                connection = DriverManager.getConnection(URL, USER, PASSWORD);

                System.out.println("Koneksi ke database berhasil!");

            } catch (ClassNotFoundException e) {

                System.err.println("JDBC Driver tidak ditemukan. Pastikan file JAR sudah ditambahkan ke classpath.");
            }
        }
    }
}
```

```

        e.printStackTrace();
    }
}
return connection;
}
// Metode untuk menutup koneksi
public static void closeConnection() {
    if (connection != null) {
        try {
            connection.close();
            System.out.println("Koneksi database ditutup.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
static Connection getConnection() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver"); // Pastikan driver sudah ada
        return DriverManager.getConnection("jdbc:mysql://localhost:3306/db_toko", "username",
"password");
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}
}

```

```
}
```

Penjelasan:

➤ **Import Statements:**

- `import java.sql.Connection;`
- `import java.sql.DriverManager;`
- `import java.sql.SQLException;`
 - Kelas-kelas ini adalah bagian dari Java Database Connectivity (JDBC) API.
 - `Connection`: Merepresentasikan koneksi aktif ke database.
 - `DriverManager`: Digunakan untuk mengelola kumpulan driver JDBC dan membuat koneksi ke database.
 - `SQLException`: Kelas pengecualian (exception) yang dilempar ketika terjadi kesalahan dalam operasi database.

➤ **Deklarasi Konstanta dan Variabel Statis:**

- `private static final String URL = "jdbc:mysql://localhost:3306/db_toko";`
 - `URL`: Konstanta string yang menyimpan URL koneksi database. Ini adalah alamat database MySQL Anda.
 - `jdbc:mysql://`: Protokol untuk JDBC ke MySQL.
 - `localhost`: Host tempat server MySQL berjalan (biasanya komputer Anda sendiri).
 - `3306`: Port default untuk MySQL.
 - `/db_toko`: Nama database yang akan dihubungkan.
- `private static final String USER = "root";`
 - `USER`: Konstanta string untuk username database MySQL Anda. Umumnya "root" untuk instalasi default, tetapi harus diganti jika Anda memiliki username lain.
- `private static final String PASSWORD = "";`

- PASSWORD: Konstanta string untuk password database MySQL Anda. Jika Anda tidak memiliki password untuk user "root", biarkan kosong. **Penting: Dalam aplikasi nyata, jangan menyimpan password secara hardcode seperti ini. Gunakan metode yang lebih aman.**
- private static Connection connection = null;
 - connection: Variabel statis yang akan menyimpan objek Connection ke database. Dideklarasikan sebagai static agar hanya ada satu instance koneksi yang digunakan di seluruh aplikasi (strategi *singleton* untuk koneksi database sederhana). Diinisialisasi dengan null karena belum ada koneksi saat aplikasi dimulai.

➤ **Metode DatabaseConnection() (Metode Utama untuk Mendapatkan Koneksi)**

- public static Connection DatabaseConnection() throws SQLException { ... }
 - Ini adalah metode statis dan publik yang bertanggung jawab untuk membuat dan mengembalikan objek Connection.
 - throws SQLException: Menunjukkan bahwa metode ini dapat melempar SQLException jika terjadi masalah saat berinteraksi dengan database. Pemanggil metode ini harus menangani pengecualian ini.
 - if (connection == null || connection.isClosed()) { ... }
 - Ini adalah bagian penting untuk *connection management*. Ini memeriksa dua hal:
 1. connection == null: Apakah objek connection belum pernah diinisialisasi (koneksi pertama kali).
 2. connection.isClosed(): Apakah koneksi yang ada saat ini sudah ditutup.
 - Jika salah satu kondisi di atas benar (koneksi belum ada atau sudah ditutup), maka akan dibuat koneksi baru. Ini mencegah pembuatan koneksi berulang kali jika sudah ada yang aktif, dan juga mengatasi kasus koneksi yang mati/ditutup.
 - **Blok try-catch (ClassNotFoundException):**
 - Class.forName("com.mysql.cj.jdbc.Driver");

- Baris ini memuat driver JDBC untuk MySQL ke dalam memori. Ini adalah langkah pertama yang harus dilakukan sebelum mencoba terhubung ke database.
- ClassNotFoundException akan terjadi jika file JAR driver MySQL Connector/J tidak ditemukan di classpath proyek Anda. Pesan error yang dicetak akan membantu mengidentifikasi masalah ini.
- `connection = DriverManager.getConnection(URL, USER, PASSWORD);`
 - Ini adalah baris yang sebenarnya membuat koneksi ke database menggunakan DriverManager dan detail URL, username, dan password yang telah ditentukan.
- `System.out.println("Koneksi ke database berhasil!");`
 - Pesan ini dicetak ke konsol jika koneksi berhasil dibuat. Berguna untuk debugging.
- `return connection;`
 - Mengembalikan objek Connection yang telah dibuat atau yang sudah ada sebelumnya.

➤ **Metode `closeConnection()` (Metode untuk Menutup Koneksi)**

- `public static void closeConnection() { ... }`
 - Metode statis dan publik ini bertanggung jawab untuk menutup koneksi database yang aktif.
 - `if (connection != null) { ... }`
 - Memastikan bahwa ada objek Connection yang valid sebelum mencoba menutupnya, untuk menghindari NullPointerException.
 - `connection.close();`
 - Menutup koneksi database. Ini sangat penting untuk membebaskan sumber daya database.
 - `System.out.println("Koneksi database ditutup.");`
 - Pesan konfirmasi bahwa koneksi telah ditutup.
 - `catch (SQLException e) { ... }`

- Menangani potensi SQLException jika terjadi masalah saat menutup koneksi.

➤ **Metode getConnection() (Metode Duplikat/Berpotensi Masalah)**

- static Connection getConnection() { ... }
 - Metode ini memiliki nama yang lebih umum (getConnection()) dan mengembalikan Connection.
 - **MASALAH SERIUS:**
 - return
DriverManager.getConnection("jdbc:mysql://localhost:3306/db_toko",
"username", "password");

➤ **Kesimpulan:**

Kelas DatabaseConnection ini bertujuan untuk menyediakan cara terpusat dalam mengelola koneksi database di aplikasi Java Anda. Ide dasarnya bagus untuk enkapsulasi koneksi. Namun, ada **duplikasi metode koneksi (DatabaseConnection() dan getConnection())** dan satu metode memiliki **implementasi yang jauh lebih baik (DatabaseConnection())** daripada yang lain. Metode getConnection() yang terakhir justru berpotensi menimbulkan masalah karena akan selalu membuat koneksi baru dan menggunakan *hardcoded* username/password yang salah.