

**LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK**



NAMA : ENSTIKA ELINDASARI

NIM : 24104410006

PERIODE : SEMESTER GENAP 2024/2025

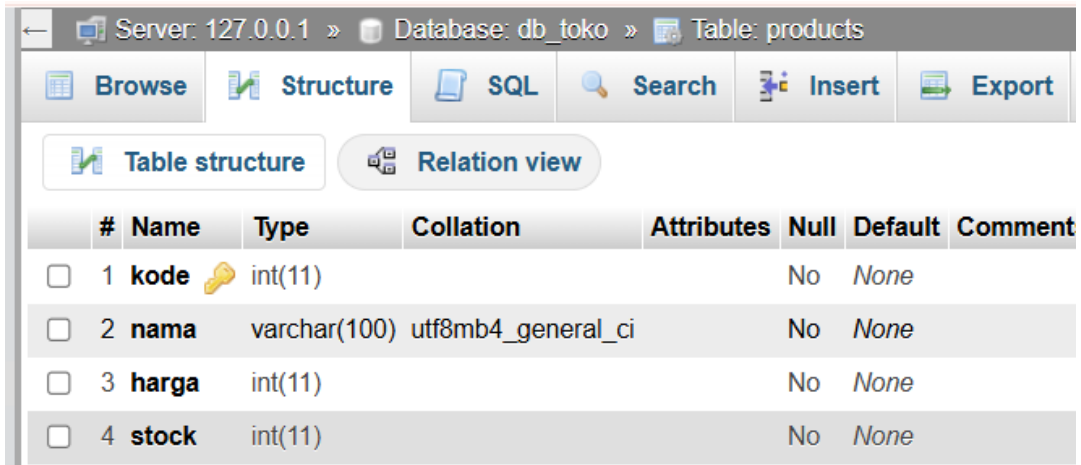
**PROGRAM STUDY TEKNIK INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**

**UNIVERSITAS ISLAM BALITAR**

**2025**

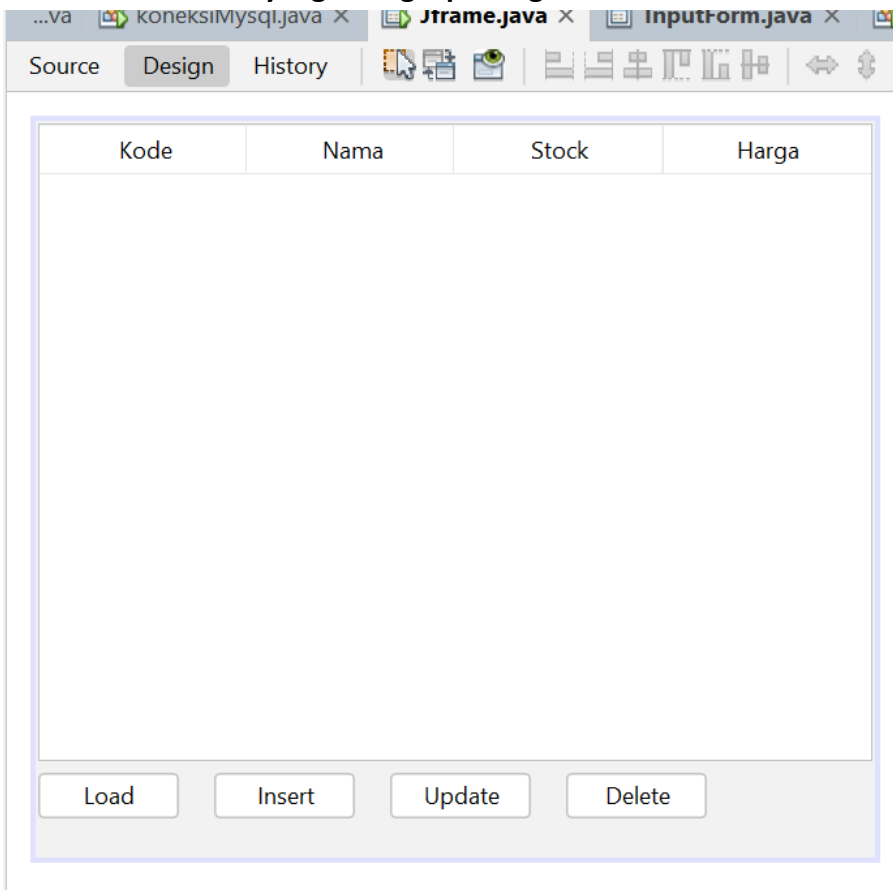
## 1. Pembuatan Database dengan nama db\_toko dan table product



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

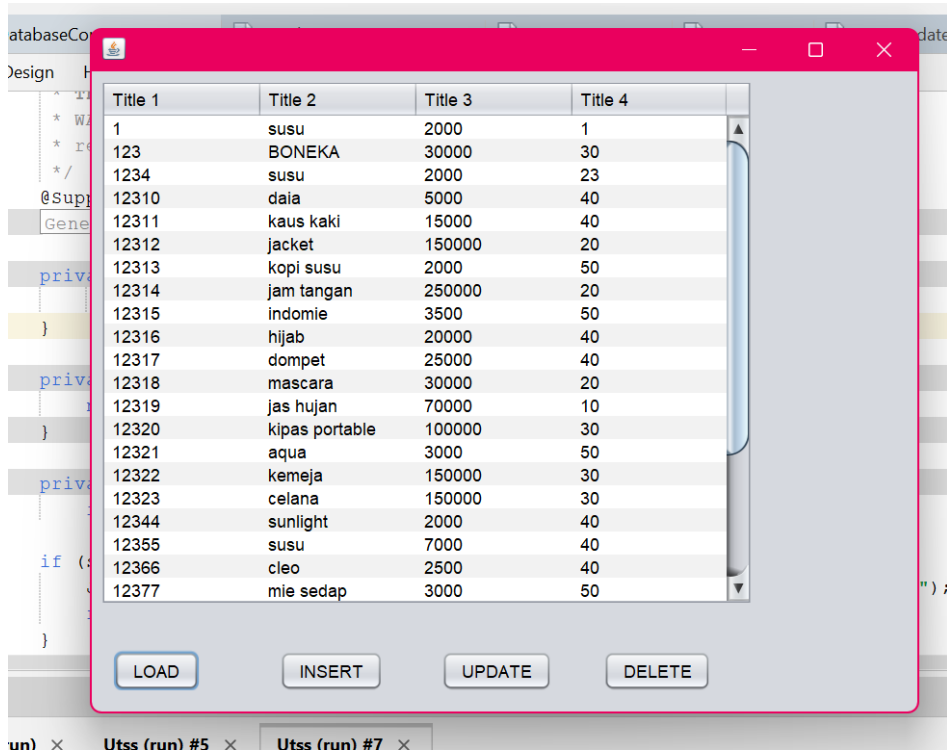


The screenshot shows a Java Swing JFrame window. The window has a title bar with three tabs: 'koneksiMysql.java', 'JFrame.java', and 'InputForm.java'. The 'JFrame.java' tab is active. The window contains a JTable with four columns: 'Kode', 'Nama', 'Stock', and 'Harga'. Below the table are four buttons: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

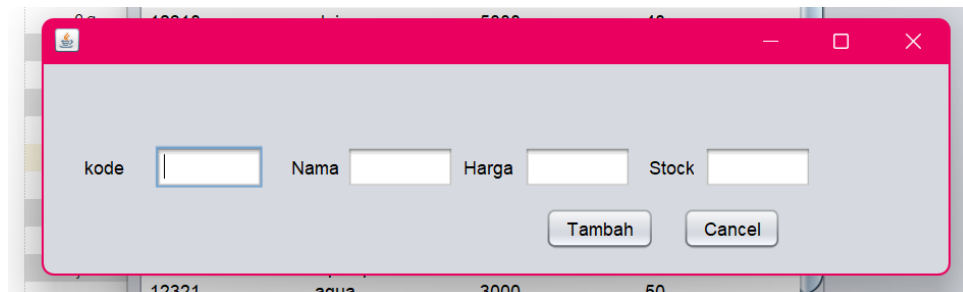
Kode	Nama	Stock	Harga
------	------	-------	-------

Load Insert Update Delete

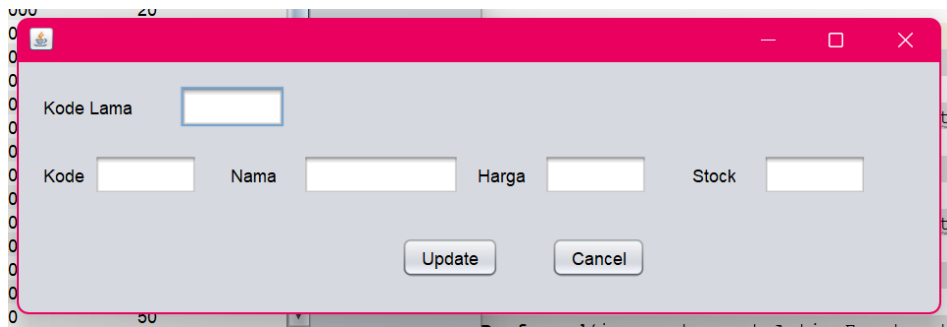
- Load: seleksi semua data di tabel produk



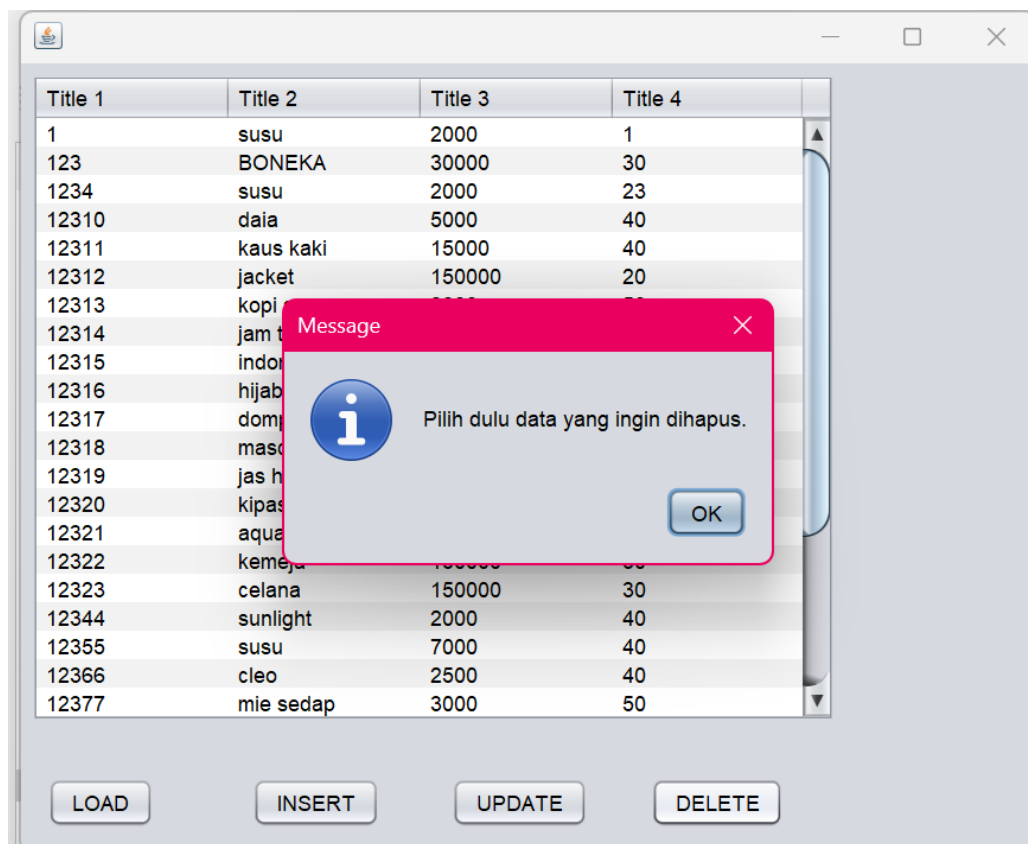
- Insert: memasukkan satu produk baru



- 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

```
private void LoadActionPerformed(java.awt.event.ActionEvent evt) {  
    try {  
        Connection conn = DatabaseConnection.DatabaseConnection(); Statement stmt = conn.createStatement(); Res  
        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",  
            ex.printStackTrace());  
    }  
}
```

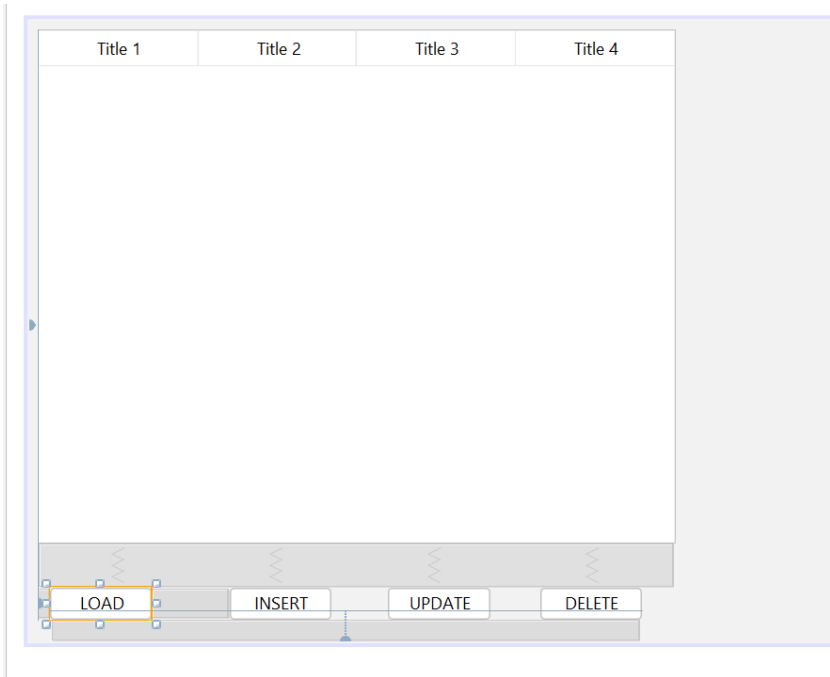
#### b. Inheritance

```
20 //  
21 public class Kasir extends javax.swing.JFrame {  
22  
23     /**  
24      * Creates new form Kasir  
25      */  
26     public Kasir() {  
27         initComponents();  
28     }  
}
```

#### c. try-catch SQLException

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

#### d. GUI Swing



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

##### a. Class Kasir

```
package projectpbo;

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;

public class Kasir extends javax.swing.JFrame {
```

```

/**
 * Creates new form Kasir
 */
public Kasir() {
    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 [] {
        "Kode", "Nama", "Harga", "Stock"
    }
));
jScrollPane1.setViewportViewView(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);
    }
});

```





```

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 280,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(36, 36, 36)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(Load)
                .addComponent(Insert)
                .addComponent(Update)
                .addComponent>Delete))
            .addGap(0, 10, Short.MAX_VALUE))
        );
pack();
} // </editor-fold>

```

```

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();
}
}

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.DatabaseConnection();
            String sql = "DELETE FROM products 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());
    }
}

}

/**
 * @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(Kasir.class.getName()).log(java.util.logging.Level.SEVERE, null,
ex);

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

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

    } catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Kasir.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 Kasir().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:**

### ➤ **Komponen GUI Utama:**

#### **1. jTable1 (javax.swing.JTable):**

- Komponen ini menampilkan data produk dalam format tabel. Saat aplikasi dijalankan, tabel ini diinisialisasi dengan kolom "Kode", "Nama", "Harga", dan "Stock". Data aktual akan dimuat dari database.

#### **2. Load (javax.swing.JButton):**

- Tombol ini memicu operasi "Read". Ketika diklik, ia akan mengambil seluruh data produk dari database dan menampilkannya di jTable1.

#### **3. Insert (javax.swing.JButton):**

- Tombol ini memicu operasi "Create". Ketika diklik, ia akan membuka sebuah jendela baru (FormInput) yang memungkinkan pengguna untuk memasukkan detail produk baru yang akan disimpan ke database.

#### **4. Update (javax.swing.JButton):**

- Tombol ini memicu operasi "Update". Ketika diklik, ia akan membuka sebuah jendela baru (FormUpdate) yang dimaksudkan untuk mengedit data produk yang sudah ada. **Catatan:** Dalam implementasi yang diberikan, data dari baris yang dipilih di jTable1 belum otomatis diteruskan ke FormUpdate. FormUpdate akan muncul dalam keadaan kosong.

#### **5. Delete (javax.swing.JButton):**

- Tombol ini memicu operasi "Delete". Ketika diklik, ia akan menghapus baris data produk yang sedang dipilih di jTable1 dari database.

➤ **Alur Kerja Metode-Metode Kunci:**

**1. Konstruktor Kasir():**

- Ketika objek Kasir dibuat (yaitu, saat aplikasi dimulai), konstruktor ini memanggil `initComponents()`.

**2. `initComponents()`:**

- Metode ini (dihasilkan secara otomatis oleh GUI builder) bertanggung jawab untuk:
  - Menginisialisasi semua komponen GUI (tabel, tombol).
  - Menentukan properti visual komponen (teks tombol, model tabel awal).
  - Menambahkan komponen ke dalam frame.
  - Mengaitkan *event listener* (misalnya, `ActionListener` untuk tombol) yang akan memanggil metode spesifik ketika suatu tindakan dilakukan oleh pengguna (misalnya, tombol diklik).
  - Mengatur tata letak komponen di dalam jendela.

**3. `LoadActionPerformed(java.awt.event.ActionEvent evt)`:**

- **Tujuan:** Mengambil data produk dari database dan menampilkannya di `jTable1`.
- **Proses:**
  - Menggunakan blok `try-with-resources` untuk memastikan koneksi database (`Connection`), pernyataan SQL (`Statement`), dan hasil query (`ResultSet`) ditutup secara otomatis setelah digunakan.
  - Terhubung ke database melalui `DatabaseConnection.DatabaseConnection()` (ada potensi kesalahan penulisan metode di sini, seharusnya `DatabaseConnection.getConnection()`).
  - Mengeksekusi query SQL `SELECT * FROM products` untuk mendapatkan semua data.
  - Mereset baris yang ada di `jTable1` (`model.setRowCount(0)`).
  - Mengulang setiap baris data dari `ResultSet`. Untuk setiap baris, data (kode, nama, harga, stok) diambil dan ditambahkan ke `DefaultTableModel` yang terhubung dengan `jTable1`.

- Menangani SQLException dengan menampilkan JOptionPane dan mencetak *stack trace* jika terjadi kesalahan database.
  - **Catatan Penting:** Tipe data kode diambil sebagai int (`rs.getInt("kode")`) di sini, padahal pada database yang umum digunakan, kode seringkali VARCHAR (string). Ini bisa menyebabkan SQLException jika format kode tidak bisa dikonversi ke integer.
4. **InsertActionPerformed(java.awt.event.ActionEvent evt):**
- **Tujuan:** Memulai proses penambahan data produk baru.
  - **Proses:** Membuat dan menampilkan instance baru dari kelas FormInput (diasumsikan sebagai jendela terpisah untuk input data).
5. **UpdateActionPerformed(java.awt.event.ActionEvent evt):**
- **Tujuan:** Memulai proses pembaruan data produk yang sudah ada.
  - **Proses:** Membuat dan menampilkan instance baru dari kelas FormUpdate (diasumsikan sebagai jendela terpisah untuk input update data). **Perlu perbaikan untuk meneruskan data yang dipilih dari tabel ke FormUpdate.**
6. **DeleteActionPerformed(java.awt.event.ActionEvent evt):**
- **Tujuan:** Menghapus data produk dari database dan dari jTable1.
  - **Proses:**
    - Mengecek apakah ada baris yang dipilih di jTable1. Jika tidak, menampilkan peringatan.
    - Menampilkan dialog konfirmasi kepada pengguna sebelum melanjutkan penghapusan.
    - Jika pengguna mengkonfirmasi:
      - Mengambil nilai "kode" dari baris yang dipilih di jTable1. **Catatan:** Sama seperti LoadActionPerformed, ini mengonversi kode ke int (`Integer.parseInt(...)`) yang bisa menjadi masalah jika kode di database adalah string (VARCHAR).
      - Membangun dan mengeksekusi PreparedStatement SQL DELETE FROM products WHERE kode = ?.



- Menghapus baris yang sesuai dari DefaultTableModel (model.removeRow(selectedRow)) setelah berhasil dihapus dari database.
- Menampilkan pesan sukses atau error melalui JOptionPane.
- Menangani SQLException jika terjadi kesalahan database.

## 7. **main(String args[]):**

- Metode ini adalah titik masuk eksekusi program.
- Mengatur *Look and Feel* GUI (opsional, untuk tampilan visual yang lebih baik).
- Menjadwalkan pembuatan dan tampilan Kasir frame di *Event Dispatch Thread (EDT)* untuk memastikan *thread safety* GUI.

### **b. Form Input**

```
/*
 * 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
 */
package projectpbo;

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 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() {

        jLabel2 = new javax.swing.JLabel();
        tfKode = new javax.swing.JTextField();
        jLabel1 = new javax.swing.JLabel();
        Nama = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        Harga = new javax.swing.JTextField();
    }
}

```

```

jLabel4 = new javax.swing.JLabel();
Stock = new javax.swing.JTextField();
INSERT = new javax.swing.JButton();
CANCEL = new javax.swing.JButton();
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
jLabel2.setText("Kode");
tfKode.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        tfKodeActionPerformed(evt);
    }
});
jLabel1.setText("Nama");
Nama.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
       >NamaActionPerformed(evt);
    }
});
jLabel3.setText("Harga");
Harga.setText("jTextField1");
jLabel4.setText("Stock");
Stock.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        StockActionPerformed(evt);
    }
});
INSERT.setText("INSERT");

```

```

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

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

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(22, 22, 22)
            .addComponent(jLabel2)
            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(tfKode, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(49, 49, 49)
            .addComponent(jLabel1)
            .addGap(18, 18, 18)
            .addComponent>Nama, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(55, 55, 55)
            .addComponent(jLabel3)
            .addGap(18, 18, 18)
            .addComponent(Harga, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
                javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(68, 68, 68)

```

```

        .addComponent(jLabel4)

        .addGap(18, 18, 18)

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

        .addContainerGap(49, Short.MAX_VALUE))

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

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

        .addComponent(INSERT)

        .addGap(18, 18, 18)

        .addComponent(CANCEL)

        .addGap(69, 69, 69))
);

layout.setVerticalGroup(

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

        .addGroup(layout.createSequentialGroup())

            .addGap(63, 63, 63)

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

                .addComponent(jLabel2)

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

                .addComponent(jLabel1)

                .addComponent>Nama, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

                .addComponent(jLabel3)

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

```

```

        .addComponent(jLabel4)

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

        .addGap(34, 34, 34)

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

            .addComponent(INSERT)

            .addComponent(CANCEL))

        .addContainerGap(31, Short.MAX_VALUE))

    );

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

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

    // TODO add your handling code here:

}

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

    // TODO add your handling code here:

}

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

    // TODO add your handling code here:

}

private void INSERTActionPerformed(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(tfKode.getText());

            String nama = Nama.getText();

```

```

int harga = Integer.parseInt(Harga.getText());
int stock = Integer.parseInt(Stock.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();
}
}

/**
 * @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 CANCEL;

private javax.swing.JTextField Harga;

private javax.swing.JButton INSERT;

private javax.swing.JTextField Nama;

private javax.swing.JTextField Stock;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JTextField tfKode;

// End of variables declaration

}

```

PENJELASAN:

➤ **Komponen GUI Utama di Formulir Ini:**

1. **Label (jLabel2, jLabel1, jLabel3, jLabel4):** Ini adalah teks statis yang berfungsi sebagai judul atau penanda untuk setiap kolom input.
  - "Kode"

- "Nama"
  - "Harga"
  - "Stock"
2. **Kolom Input Teks (tfKode, Nama, Harga, Stock):** Ini adalah kotak-kotak di mana pengguna akan mengetikkan informasi produk baru.
- tfKode: Untuk memasukkan kode produk (misalnya, P001).
  - Nama: Untuk memasukkan nama produk (misalnya, Pensil).
  - Harga: Untuk memasukkan harga produk (misalnya, 2500).
  - Stock: Untuk memasukkan jumlah stok produk (misalnya, 100).
  - **Catatan:** Pada `Harga.setText("jTextField1");` di `initComponents()`, ada *placeholder* teks "jTextField1". Ini seharusnya dihapus agar kolom harga kosong saat form dibuka.
3. **Tombol (INSERT, CANCEL):**
- **"INSERT":** Tombol ini untuk **menyimpan data** yang sudah diisi di form ke database.
  - **"CANCEL":** Tombol ini untuk **membatalkan** pengisian form dan menutup jendela ini tanpa menyimpan data. (Perhatikan: kode untuk tombol CANCEL belum diimplementasikan, saat ini hanya ada deklarasi tombolnya saja).
- **Alur Kerja Utama:**
1. **Ketika FormInput Dibuka (FormInput() constructor dan initComponents()):**
- Sama seperti Kasir, konstruktor memanggil `initComponents()`.
  - `initComponents()` secara otomatis membuat label, kotak input, dan tombol, lalu mengatur posisinya di jendela. Ini juga mengaitkan aksi ke tombol "INSERT".
2. **Saat Pengguna Mengisi Data:**
- Pengguna mengetikkan Kode, Nama, Harga, dan Stock di kotak input masing-masing.
3. **Saat Anda Klik Tombol "INSERT" (INSERTActionPerformed):**
- Ini adalah bagian paling penting yang menangani penyimpanan data.

- **Blok try-with-resources untuk Koneksi Database:**

- `try (Connection conn = DatabaseConnection.DatabaseConnection()) { ... }`
  - Mencoba mendapatkan koneksi ke database melalui kelas `DatabaseConnection` yang sudah kita bahas sebelumnya. Ini akan memastikan koneksi ditutup secara otomatis.
  - **Potensi typo:** Perhatikan lagi `DatabaseConnection.DatabaseConnection()`. Seharusnya memanggil `DatabaseConnection.getConnection()` jika itu nama metode yang benar untuk mendapatkan koneksi di kelas `DatabaseConnection` Anda.

- **Query SQL:**

- `String sql = "INSERT INTO products (kode, nama, harga, stock) VALUES (?, ?, ?, ?)";`
  - Ini adalah perintah SQL untuk memasukkan data baru. `?` adalah *placeholder* untuk nilai yang akan dimasukkan, ini melindungi dari serangan SQL Injection.

- **Mengambil Data dari Input Form:**

- `int kode = Integer.parseInt(tfKode.getText());`
- `String nama = Nama.getText();`
- `int harga = Integer.parseInt(Harga.getText());`
- `int stock = Integer.parseInt(Stock.getText());`
  - Program mengambil teks dari setiap kotak input.
  - Untuk "Harga" dan "Stock", teks diubah menjadi angka (int) menggunakan `Integer.parseInt()`.
  - **Potensi Masalah:** Untuk `tfKode`, Anda juga mengonversinya ke int. Jika kode produk Anda sebenarnya adalah gabungan huruf dan angka (misalnya "P001"), maka `Integer.parseInt()` akan gagal dan program akan error. Seharusnya `String kode = tfKode.getText();` dan nanti di `pstmt.setString(1, kode);` jika kode di database adalah VARCHAR.

- **Mengatur Nilai ke SQL Query:**

- pstmt.setInt(1, kode); (atau setString jika kode berupa string)
- pstmt.setString(2, nama);
- pstmt.setInt(3, harga);
- pstmt.setInt(4, stock);
  - Nilai yang diambil dari form kemudian diatur ke *placeholder* ? dalam query SQL. Angka di awal (1, 2, 3, 4) menunjukkan posisi ? dalam query.
- **Mengeksekusi Query:**
  - int rowsAffected = pstmt.executeUpdate();
    - Perintah SQL INSERT dijalankan. rowsAffected akan menunjukkan berapa banyak baris data yang berhasil ditambahkan (idealnya 1).
- **Memberi Tahu Pengguna Hasilnya:**
  - Jika rowsAffected > 0, berarti data berhasil ditambahkan, dan program menampilkan pesan "Data berhasil ditambahkan!" lalu menutup jendela form (this.dispose();).
  - Jika rowsAffected 0 atau kurang, berarti ada masalah, dan program menampilkan pesan "Gagal menambahkan data."
- **Penanganan Error (catch block):**
  - catch (SQLException | NumberFormatException ex) { ... }
    - Program menangani dua jenis error yang mungkin terjadi:
      - SQLException: Masalah terkait database (misalnya, koneksi putus, query salah, data duplikat).
      - NumberFormatException: Masalah jika pengguna memasukkan teks non-angka di kolom "Harga" atau "Stock" yang seharusnya angka.
    - Jika error terjadi, program menampilkan pesan error kepada pengguna dan mencetak detail error ke konsol untuk debugging.

#### 4. main(String args[]):

- Sama seperti di kelas Kasir, ini adalah titik awal program. Ketika FormInput dijalankan langsung sebagai aplikasi (bukan dari Kasir), ia akan menampilkan jendela ini.

### c. Form Update

```
package projectpbo;

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;

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() {

    jLabel4 = new javax.swing.JLabel();

    Stock = new javax.swing.JTextField();

    INSERT = new javax.swing.JButton();

    CANCEL = new javax.swing.JButton();

    jLabel2 = new javax.swing.JLabel();

    tfKodeBaru = new javax.swing.JTextField();

    jLabel1 = new javax.swing.JLabel();

    tfNama = new javax.swing.JTextField();

    jLabel3 = new javax.swing.JLabel();

    Harga = new javax.swing.JTextField();

    jLabel5 = new javax.swing.JLabel();

    tfKode = new javax.swing.JTextField();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jLabel4.setText("Stock");

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

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

            StockActionPerformed(evt);

        }

    });

    INSERT.setText("INSERT");

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

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

```

```

        INSERTActionPerformed(evt);
    }
});
CANCEL.setText("CANCEL");
jLabel2.setText("Kode Baru");
tfKodeBaru.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        tfKodeBaruActionPerformed(evt);
    }
});
jLabel1.setText("Nama");
tfNama.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        tfNamaActionPerformed(evt);
    }
});
jLabel3.setText("Harga");
jLabel5.setText("Kode Lama");
tfKode.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        tfKodeActionPerformed(evt);
    }
});
javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

```

```

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup())
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(INSET)
                .addGap(18, 18, 18)
                .addComponent(CANCEL)
                .addGap(69, 69, 69))
        .addGroup(layout.createSequentialGroup()
            .addGap(19, 19, 19)
            .addGroup(layout.createSequentialGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel2)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                    .addComponent(tfKodeBaru, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGroup(layout.createSequentialGroup()
                    .addComponent(jLabel5)
                    .addGap(18, 18, 18)
                    .addComponent(tfKode, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addGap(40, 40, 40)
            .addComponent(jLabel1)
            .addGap(18, 18, 18)
            .addComponent(tfNama, javax.swing.GroupLayout.PREFERRED_SIZE, 71,
javax.swing.GroupLayout.PREFERRED_SIZE))

```



```

        .addGap(55, 55, 55)

        .addComponent(jLabel3)

        .addGap(18, 18, 18)

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

        .addGap(68, 68, 68)

        .addComponent(jLabel4)

        .addGap(18, 18, 18)

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

        .addContainerGap(14, Short.MAX_VALUE))
    );

    layout.setVerticalGroup(

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

        .addGroup(layout.createSequentialGroup())

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

            .addGroup(layout.createSequentialGroup())

                .addGap(63, 63, 63)

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

                .addComponent(jLabel1)

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

                .addComponent(jLabel3)

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

                .addComponent(jLabel4)

```

```

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

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

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

        .addContainerGap()

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

            .addComponent(jLabel5)

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

            .addGap(26, 26, 26)))

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

            .addComponent(jLabel2)

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

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

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

                .addComponent(INsert)

                .addComponent(CANCEL))

            .addContainerGap(24, Short.MAX_VALUE))

    );

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

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

    // TODO add your handling code here:

}

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

```

```

        // TODO add your handling code here:
    }

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

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

    private void INSERTActionPerformed(java.awt.event.ActionEvent evt) {
        int kodeLama = Integer.parseInt(tfKode.getText());
        int kodeBaru = Integer.parseInt(tfKodeBaru.getText());
        String nama = tfNama.getText();
        int harga = Integer.parseInt(Harga.getText());
        int stok = Integer.parseInt(Stock.getText());
        try {
            Connection conn = DatabaseConnection.DatabaseConnection(); // 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.JButton CANCEL;

```

```

private javax.swing.JTextField Harga;
private javax.swing.JButton INSERT;
private javax.swing.JTextField Stock;
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 tfKode;
private javax.swing.JTextField tfKodeBaru;
private javax.swing.JTextField tfNama;

// End of variables declaration
}

```

#### **PENJELASAN:**

##### **Perbedaan Utama dengan FormInput:**

- **Tujuan:** Jika FormInput untuk menambah data baru, FormUpdate ini untuk mengubah data yang sudah ada.
- **Input "Kode Lama":** Ada kolom tambahan untuk "Kode Lama" (tfKode) yang seharusnya digunakan untuk mengidentifikasi produk mana yang akan diubah.

##### **Komponen GUI Utama di Formulir Ini:**

1. **Label (jLabel4, jLabel2, jLabel1, jLabel3, jLabel5):** Teks statis sebagai penanda untuk setiap kolom input.
  - jLabel5: "Kode Lama" (untuk mengidentifikasi data yang akan di-update)
  - jLabel2: "Kode Baru" (jika kode produk ingin diubah)
  - jLabel1: "Nama"
  - jLabel3: "Harga"
  - jLabel4: "Stock"

2. **Kolom Input Teks (Stock, tfKodeBaru, tfNama, Harga, tfKode):** Ini adalah kotak-kotak di mana pengguna akan melihat data produk yang sudah ada dan bisa mengeditnya.
  - tfKode: Untuk memasukkan **kode produk lama** (produk yang ingin diubah).
  - tfKodeBaru: Untuk memasukkan **kode produk baru** (jika kode produk juga ingin diubah).
  - tfNama: Untuk memasukkan nama produk baru.
  - Harga: Untuk memasukkan harga produk baru.
  - Stock: Untuk memasukkan jumlah stok produk baru.
3. **Tombol (INSERT, CANCEL):**
  - **"INSERT"**: (Sebenarnya lebih tepat dinamakan "UPDATE" atau "SIMPAN PERUBAHAN" di sini) Tombol ini untuk **menyimpan perubahan** data yang sudah diedit di form ke database.
  - **"CANCEL"**: Tombol ini untuk **membatalkan** proses pengeditan dan menutup jendela ini tanpa menyimpan perubahan. (Perhatikan: kode untuk tombol CANCEL belum diimplementasikan, saat ini hanya ada deklarasi tombolnya saja).

#### **Alur Kerja Utama:**

1. **Ketika FormUpdate Dibuka (FormUpdate() constructor dan initComponents()):**
  - Konstruktor memanggil initComponents() yang membuat semua elemen GUI (label, kotak input, tombol) dan mengatur tata letaknya.
  - ActionListener diatur untuk tombol "INSERT".
  - **Penting:** Saat ini, ketika FormUpdate dibuka dari Kasir, kolom inputnya akan kosong. Seharusnya, **data dari baris yang dipilih di Kasir perlu dikirimkan ke FormUpdate agar kolom input terisi otomatis** dengan data yang ingin di-update. Ini membutuhkan perubahan pada cara FormUpdate dipanggil dari Kasir, atau penambahan metode khusus di FormUpdate untuk menerima dan mengisi data.
2. **Saat Pengguna Mengisi/Mengubah Data:**
  - Pengguna memasukkan kodeLama (identifikasi produk yang akan diubah).
  - Pengguna mengedit nilai di kolom tfKodeBaru, tfNama, Harga, dan Stock.
3. **Saat Anda Klik Tombol "INSERT" (INSERTActionPerformed):**

- Ini adalah metode yang menangani logika pembaruan data.
- **Mengambil Data dari Input Form:**
  - `int kodeLama = Integer.parseInt(tfKode.getText());`
  - `int kodeBaru = Integer.parseInt(tfKodeBaru.getText());`
  - `String nama = tfNama.getText();`
  - `int harga = Integer.parseInt(Harga.getText());`
  - `int stok = Integer.parseInt(Stock.getText());`
    - Nilai dari semua kotak input diambil.
    - **Kritikal:** Semua kode, harga, dan stok di-*parse* ke int. Jika kode di database Anda adalah VARCHAR (teks seperti "P001"), maka `Integer.parseInt()` akan menyebabkan `NumberFormatException`. Ini adalah masalah yang sama seperti di `FormInput` dan `Kasir`.
- **Koneksi Database:**
  - `Connection conn = DatabaseConnection.DatabaseConnection();`
    - Mendapatkan koneksi ke database. Lagi-lagi, pastikan nama metodenya sudah konsisten (`DatabaseConnection.getConnection()`).
- **Query SQL UPDATE:**
  - `String sql = "UPDATE products SET kode = ?, nama = ?, harga = ?, stock = ? WHERE kode = ?";`
    - Ini adalah perintah SQL untuk memperbarui data.
    - `SET kode = ?, nama = ?, harga = ?, stock = ?`: Menentukan kolom-kolom yang akan diubah dan nilai barunya.
    - `WHERE kode = ?`: Ini adalah kondisi kunci. Perintah ini hanya akan mengubah baris di mana kode sama dengan `kodeLama` yang Anda masukkan.
- **Mengatur Nilai ke SQL Query (PreparedStatement):**
  - `stmt.setInt(1, kodeBaru);`
  - `stmt.setString(2, nama);`



- `stmt.setInt(3, harga);`
- `stmt.setInt(4, stok);`
- `stmt.setInt(5, kodeLama);`
  - Nilai yang diambil dari form dimasukkan ke *placeholder* ? sesuai urutannya.
  - `kodeBaru` akan menjadi nilai baru untuk kolom kode.
  - `kodeLama` digunakan untuk WHERE clause, yaitu mengidentifikasi baris mana yang akan diupdate.
- **Mengeksekusi Query:**
  - `int rowsUpdated = stmt.executeUpdate();`
    - Perintah UPDATE dijalankan. `rowsUpdated` akan menunjukkan berapa banyak baris yang terpengaruh (idealnya 1).
- **Memberi Tahu Pengguna Hasilnya:**
  - Jika `rowsUpdated > 0`, berarti data berhasil diupdate, dan program menampilkan pesan sukses lalu menutup jendela form (`this.dispose();`).
  - Jika `rowsUpdated 0`, berarti `kodeLama` yang dimasukkan tidak ditemukan di database, dan program menampilkan pesan "Data gagal diupdate. Kode lama tidak ditemukan."
- **Menutup Sumber Daya:**
  - `stmt.close();` dan `conn.close();`
    - Ini sangat penting untuk menutup `PreparedStatement` dan `Connection` untuk membebaskan sumber daya database. Namun, jika `DatabaseConnection` Anda menggunakan pola Singleton, `conn.close()` di sini mungkin tidak diinginkan karena akan menutup koneksi yang mungkin masih dibutuhkan oleh bagian lain aplikasi. Lebih baik biarkan `DatabaseConnection` yang mengelola penutupan koneksi secara global jika itu polanya.
- **Penanganan Error (catch block):**
  - `catch (NumberFormatException e):` Menangani jika ada input teks di kolom kode, harga, atau stock yang seharusnya angka.

- catch (SQLException e): Menangani masalah terkait database.
- Menampilkan pesan error kepada pengguna melalui JOptionPane.

#### 4. **main(String args[]):**

- Metode ini adalah titik masuk eksekusi jika FormUpdate dijalankan secara langsung.

#### **d. Database Connection**

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package projectpbo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

/**
 *
 * @author ASUS
 */
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 {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Koneksi ke database berhasil!");
        } catch (ClassNotFoundException e) {
            System.err.println("JDBC Driver tidak ditemukan.");
            throw new SQLException("Driver tidak ditemukan", e);
        }
    }
    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() throws SQLException, ClassNotFoundException {
    Class.forName("com.mysql.cj.jdbc.Driver");

```

```

return DriverManager.getConnection("jdbc:mysql://localhost:3306/db_toko", "root", "");
}
}

```

## PENJELASAN:

### 1. package projectpbo;

- Menunjukkan bahwa kelas ini merupakan bagian dari *package* projectpbo.

### 2. Import Statements:

- java.sql.Connection, java.sql.DriverManager, java.sql.SQLException: Ini adalah kelas-kelas inti dari JDBC API yang diperlukan untuk bekerja dengan database di Java.

### 3. Konstanta dan Variabel Statis:

- private static final String URL = "jdbc:mysql://localhost:3306/db\_toko";
  - URL koneksi ke database MySQL Anda. db\_toko adalah nama databasenya.
- private static final String USER = "root";
  - Username untuk login ke database.
- private static final String PASSWORD = "";
  - Password untuk login ke database. (Kosong berarti tidak ada password, umum untuk instalasi MySQL default lokal).
- private static Connection connection = null;
  - Variabel statis ini akan menyimpan objek koneksi database yang aktif. Dideklarasikan static agar seluruh bagian aplikasi berbagi satu koneksi yang sama (pola desain *singleton* sederhana untuk koneksi database).

### 4. Metode DatabaseConnection() (Metode Utama untuk Mendapatkan Koneksi)

- public static Connection DatabaseConnection() throws SQLException { ... }
  - Ini adalah metode *public* dan *static* yang digunakan oleh bagian lain aplikasi untuk mendapatkan objek Connection.
  - if (connection == null || connection.isClosed()) { ... }
    - Ini adalah logika untuk memastikan bahwa hanya satu koneksi yang aktif pada satu waktu, dan akan membuat koneksi baru hanya jika belum ada

atau koneksi yang ada sudah terputus/ditutup. Ini penting untuk efisiensi dan stabilitas.

- **Blok try-catch:**

- `Class.forName("com.mysql.cj.jdbc.Driver");`
  - Baris ini memuat driver JDBC MySQL ke dalam memori. Ini adalah langkah penting agar Java tahu cara berkomunikasi dengan MySQL.
- `connection = DriverManager.getConnection(URL, USER, PASSWORD);`
  - Ini adalah baris yang benar-benar membuat koneksi ke database menggunakan URL, username, dan password yang telah ditentukan.
- `System.out.println("Koneksi ke database berhasil!");`
  - Pesan ini dicetak ke konsol jika koneksi berhasil, berguna untuk debugging.
- `catch (ClassNotFoundException e) { ... }`
  - Jika driver JDBC tidak ditemukan (misalnya, file JAR mysql-connector-java.jar belum ditambahkan ke *classpath* proyek), pengecualian ini akan tertangkap. Program akan mencetak pesan kesalahan dan kemudian *melemparkan* `SQLException` baru (yang membungkus `ClassNotFoundException` asli) ke pemanggil. Ini adalah penanganan yang lebih baik karena memaksa pemanggil untuk menyadari bahwa masalah koneksi bisa terjadi.

## 5. Metode `closeConnection()` (Metode untuk Menutup Koneksi)

- `public static void closeConnection() { ... }`
  - Metode *static* ini digunakan untuk menutup koneksi database yang sedang aktif.
  - `if (connection != null) { ... }`
    - Memastikan ada koneksi sebelum mencoba menutupnya untuk menghindari *NullPointerException*.
  - `connection.close();`
    - Menutup koneksi ke database, membebaskan sumber daya.

- `System.out.println("Koneksi database ditutup.");`
  - Pesan konfirmasi ke konsol.
- `catch (SQLException e) { ... }`
  - Menangani potensi kesalahan saat menutup koneksi.

## 6. Metode `getConnection()` (Metode Duplikat yang Berbeda Implementasi)

- `static Connection getConnection() throws SQLException, ClassNotFoundException { ... }`
  - **Perhatian Penting:** Ini adalah metode *static* lain yang juga bertujuan untuk mendapatkan koneksi.
  - `Class.forName("com.mysql.cj.jdbc.Driver");`
    - Sama-sama memuat driver.
  - `return DriverManager.getConnection("jdbc:mysql://localhost:3306/db_toko", "root", "");`
    - **Perbedaan Krusial:** Metode ini **selalu membuat koneksi baru setiap kali dipanggil**. Ia tidak memeriksa apakah connection yang statis itu sudah ada atau sudah ditutup, tidak menggunakan konstanta USER dan PASSWORD yang didefinisikan di atas, dan tidak ada logika untuk memastikan hanya satu koneksi yang digunakan.
    - **Masalah:** Menggunakan metode ini secara berulang di aplikasi dapat menyebabkan:
      - **Overhead:** Membuka dan menutup koneksi adalah operasi yang mahal.
      - **Batas Koneksi:** Server database memiliki batas jumlah koneksi yang bisa dibuka. Jika terlalu banyak koneksi dibuka dan tidak ditutup dengan benar, aplikasi bisa crash atau server database menjadi tidak responsif.
      - **Inkonsistensi:** Ada dua cara berbeda untuk mendapatkan koneksi di kelas ini, yang dapat membingungkan dan menyebabkan bug.
- `getConnection()` agar lebih standar dan jelas.

