DATABASE

17.1. JDBC

JDBC adalah Application Programming Interface (API) yang dirancang untuk mengakses database universal berdasarkan SQL. JDBC terdiri atsa JDBC 1.0 API yang memberika fingsi-fungsi dasar untuk akses data. JDBC 2.0 API memberikan tambahan ke fungsi-fungsi dasar dengan kelebihan-kelebihan lain yang lebih mutakhir.

JDBC adalah suatu nama trade mark, bukan sebuah singkatan. Tapi JDBC sering dikira singkatan dari Java Database Connectivity. JDBC API terdiri dari sejumlah class dan interface yang ditulis dalam bahasa Java yang menyediakan API standar sebagai alat bantu bagi pembuat program dan memberikan kemungkinan untuk menulis aplikasi database dengan menggunakan semua Java API.

JDBC API memudahkan untuk mengirim statement SQL ke sistem database relasional dan mendukung bermacam-macam bahasa SQL. Keunggulan JDBC API adalah sebuah aplikasi dapat mengakses sembarang sumber data dan dapat berjalan pada sembarang platform yang mempunyai Java Virtual Machine (JVM). Sehingga kita tidak perlu menulis satu program untuk mengakses database Sybase, Oracle atau Access dan lain-lain. Kita cukup menulis satu program yang menggunakan JDBC API, dan program dapat mengirimkan statement SQL atau statement lain ke sumber data tertentu. Dengan aplikasi yang ditulis dalam bahasa Java seseorang tidak perlu khawatir untuk menulis aplikasi yang berbeda-beda agar dapat berjalan pada platform yang berbeda-beda.

Teknologi JDBC mampu untuk melakukan tiga hal berikut:

a. Membangun sebuah koneksi ke sumber data (data source).
b. Mengirim statement ke sumber data.
c. Memproses hasil dari statement tersebut

Pada pemrograman Java dengan menggunakan JDBC, ada beberapa langkah yang secara umum harus dilakukan sehingga aplikasi tersebut dapat berinteraksi dengan database server.

Langkah-langkah untuk berinteraksi dengan database server dengan menggunakan JDBC adalah sebagai berikut :

a. Mengimpor package java.sql
b. Memanggil Driver JDBC
c. Membangun Koneksi
d. Membuat Statement
e. Melakukan Query
f. Memproses Hasil
g. Menutup Koneksi
h. Penanganan Error

17.2. Mengimpor Package java.sql

Pertama-tama yang harus dilakukan sebelum kita membuat program JDBC adalah mengimpor package java.sql terbih dahulu, karena di dalam package java.sql tersebut terdapat kelas-kelas yang akan digunakan dalam proses-proses berintekasi dengan database server misalnya kelas DriverManager, Connection, dan ResultSet.

Hal ini sangat penting dilakukan karena bagi pemula seringkali lupa untuk mengimpor package yang kelas-kelas yang akan digunakan terdapat di dalamnya, sehingga mengakibatkan kegagalan dalam mengkompile program Java.

Adapun pernyataan untuk mengimpor package java.sql adalah sebagai berikut :

```
import java.sql.*;
```

Pernyataan tersebut ditulis dibagian atas sebelum kita menulis kelas.

17.3. Memanggil Driver JDBC

Langkah pertama untuk melakukan koneksi dengan database server adalah dengan memanggil JDBC Driver dari database server yang kita gunakan. Driver adalah library yang digunakan untuk berkomunikasi dengan database server. Driver dari setiap database server berbeda-beda, sehingga kita harus menyesuaikan Driver JDBC sesuai dengan database server yang akan digunakan.

Sebelum kita memanggil driver JDBC, kita harus menginstal driver tersebut. Untuk menginstall connector JDBC dapat dilakukan dengan beberapa cara, antara lain :

a. Dengan meng-copy file jar ke folder ext dari java, misalnya berada di C:\Program Files\Java\jre7\lib\ext.
b. Meng-extract file jar kedalam suatu folder kemudian mensetting CLASSPATH ke folder tersebut.

File jar untuk MySQL : mysql-connector-java-5.1.14-bin.jar
File jar untuk SQL Server : sqljdbc4.jar

Berikut ini adalah listing program untuk memanggil driver JDBC.

```
Class.forName(namaDriver);
```

atau

```
Class.forName(namaDriver).newInstance();
```

Kedua cara di atas memiliki fungsi yang sama yaitu melakukan registrasi class driver dan melakukan intansiasi. Apabila driver yang dimaksud tidak ditemukan, maka program akan menghasilkan exception berupa ClassNotFoundException. Untuk menghasilkan exception apabila driver tidak ditemukan, maka diperlukan penambahan try-catch. Adapun cara menambahkan try-catch untuk penanganan error apabila driver tidak ditemukan adalah sebagai berikut :

```
try{
    Class.forName(namaDriver);
} catch (Exception ex){
    //Pesan kesalahan
}
```

Contoh listing memanggil driver untuk database server menggunakan MySQL adalah :

```
try{
    Class.forName("com.mysql.jdbc.Driver");
} catch (Exception ex){
    JOptionPane.showMessageDialog(null,"JDBC Driver tidak ditemukan atau
rusak\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
}
```

Dari contoh listing di atas dapat dijelaskan bahwa apabila driver yang dipanggil tidak ditemukan, maka program akan menampilkan pesan melalui kotak dialog JOptionPane dengan isi pesan adalah "`JDBC Driver tidak ditemukan atau rusak`" diikuti pesan kesalahan dari exception. Penanganan error sangat penting dilakukan karena dapat membantu kita dalam mengetahui kesalahan-kesalahan yang terjadi dalam menjalankan program sehingga kita dapat mengatasi kesalahan-kesalahan tersebut.

Berikut ini adalah daftar nama-nama driver dari beberapa database server yang sering digunakan.

Tabel 17.1. Nama Driver dari beberapa Database Server

Database Server	Nama Driver
JDBC-ODBC	sun.jdbc.odbc.JdbcOdbcDriver
MySQL	com.mysql.jdbc.Driver
PostgreSQL	org.postgresql.Driver
Microsoft SQLServer	com.microsoft.jdbc.sqlserver.SQLServerDriver
Oracle	oracle.jdbc.driver.OracleDriver
IBM DB2	COM.ibm.db2.jdbc.app.DB2Driver

17.4. Membuat Koneksi

Setelah melakukan pemanggilan terhadap driver JDBC, langkah selanjutnya adalah membangun koneksi dengan menggunakan interface Connection. Object Connection yang dibuat untuk membangun koneksi dengan database server tidak dengan cara membuat object baru dari interface Connection melainkan dari class DriverManager dengan menggunakan methode getConnection().

```
Connection cn = DriverManager.getConnection(<argumen>)
```

Untuk menangani error yang mungkin terjadi pada proses melakukan koneksi dengan database maka ditambahkan try-catch. Exception yang akan dihasilkan pada proses ini adalah berupa SQLException. Adapun cara penulisan listingnya adalah sebagai berikut :

```
try {
    //... koneksi database
} catch (Exception ex){
    //... penanganan error koneksi
}
```

Ada beberapa macam argumen yang berbeda dari methode getConnection() yang dipanggil dari DriverManager, yaitu :

a. `getConnection(String url)`

Pada metode ini kita hanya memerlukan argumen URL, sedangkan untuk data user dan password sudah diikutkan secara langsung pada URL sehingga tidak perlu lagi secara terpisah mendefinisikan data user dan password.

Adapun penulisan nilai URL dari methode getConnection(String url) adalah sebagai berikut :

```
 jdbc:<DBServer>://[Host][:Port]/<namaDB>?<user=User>&<password=P
asword>
```

Misalkan kita menggunakan database server berupa MySQL dengan spesifikasi menggunakan host adalah localhost dan port default (3306), nama database adalah DbTokoABC, nama user adalah root, dan password adalah root. Maka penulisan URL adalah sebagai berikut :

```
 "jdbc:mysql://localhost:3306/DbTokoABC?user=root&password=root"
```

Berikut ini contoh penggunaan methode ini didalam program :

```
 try {
    Connection cn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/DbTokoABC
?user=root&password=root");
 } catch (Exception ex) {
    JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

b. `getConnection(String url, Properties info)`

Pada methode ini memerlukan URL dan sebuah object Properties. Sebelum menggunakan methode ini, kita harus melakukan import package berupa java.util.*, ini dikarenakan object Properties terdapat pada package tersebut. Object Properties berisikan spesifikasi dari setiap parameter database misalnya user name, password, autocommit, dan sebagainya.

Berikut ini contoh penggunaan methode ini didalam program :

```
 try {
    String url = "jdbc:mysql://localhost:3306/DbTokoABC";
    Properties prop = new java.util.Properties(); // tidak
mengimpor kelas
    prop.put("user","root");
    prop.put("password","root");

    Connection cn = DriverManager.getConnection(url, prop);
 } catch (Exception ex) {
    JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

c. `getConnection(String url, String user, String password)`

Pada methode ini memerlukan argumen berupa URL, user name, dan password. Metode ini secara langsung mendefinisikan nilai URL, user name dan password. Berikut ini contoh penggunaan methode ini didalam program :

```
 try {
     String url = "jdbc:mysql://localhost:3306/DbTokoABC";
     String user = "root"
     String password = "root"

     Connection cn = DriverManager.getConnection(url, user,
password);
 } catch (Exception ex) {
     JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

Berikut ini adalah daftar penulisan URL dari beberapa database server yang sering digunakan.

Tabel 17.2. Penulisan URL pada beberapa Database

Database Server	Nama URL	Contoh penggunaan
JDBC-ODBC	jdbc:odbc:<NamaDatabase>	jdbc:odbc: DbTokoABC
MySQL	jdbc:mysql://<nmHost>:<port>/<nmDB>	jdbc:mysql://localhost:3306/DbTokoABC
PostgreSQL	jdbc:postgresql://<nmHost>:<port>/<nmDB>	jdbc:postgresql://localhost:5432/DbTokoABC
Microsoft SQLServer	jdbc:microsoft:sqlserver://<nmHost>:<port>; DatabaseName=<namaDatabase>	jdbc:microsoft:sqlserver://localhost:1433; DatabaseName=DbTokoABC
Oracle	jdbc:oracle:thin:@<nmHost>:<port>:<nmDB>	jdbc:oracle:thin:@localhost:1521: DbTokoABC
IBM DB2	jdbc:db2:<NamaDatabase>	jdbc:db2: DbTokoABC

17.5. Membuat Obyek Statemen

JDBC API menyediakan interface yang berfungsi untuk melakukan proses pengiriman statement SQL yang terdapat pada package java.sql. Di dalam JDBC API disediakan tiga buah interface untuk fungsi tersebut yaitu :

a. Statement

Interface ini dibuat oleh methode Connection.createStatement(). Object Statement digunakan untuk pengiriman statement SQL tanpa parameter.

```
Statement sta = Connection.createStatement();
```

b. PreparedStatement

Interface ini dibuat oleh methode Connection.prepareStatement(). Object PreparedStatement digunakan untuk pengiriman statement SQL dengan atau tanpa parameter. Dengan object ini, kita dapat menampung satu atau lebih parameter sebagai argumen input (perameter IN). Interface ini memiliki

performa lebih baik dibandingkan dengan interface Statement karena dapat menjalankan beberapa proses dalam sekali pengiriman perintah SQL.

```
PreparedStatement preSta = Connection.prepareStatement();
```

c. CallableStatement
Interface ini dibuat oleh methode Connection.prepareCall(). Object CallableStatement digunakan untuk menjalankan store procedure SQL.

```
CallableStatement stat = Connection.prepareCall();
```

17.6. Melakukan Perintah SQL

Setelah kita memiliki object statement, kita dapat menggunakannya untuk melakukan pengiriman perintah SQL dan mengeksekusinya. Metode eksekusi yang digunakan untuk perintah SQL terbagi menjadi dua bagian yaitu untuk perintah SELECT methode eksekusi yang digunakan adalah executeQery() dengan nilai kembaliannya adalah ResultSet, dan untuk perintah INSERT, UPDATE, DELETE methode eksekusi yang digunakan adalah executeUpdate().

Berikut ini adalah contoh melakukan eksekusi perintah SQL dan mengambil hasilnya (ResultSet) dengan menggunakan perintah SELECT.

```
 cn = DriverManager.getConnection(StringConnection);
 String SQLStatemen = "Select * from TbBarang where KodeBarang='"+
TxtKodeBarang.getText()+"'";
 Statement sta = cn.createStatement();
 ResultSet rset = sta.executeQuery(SQLStatemen);

 rset.first();
 if (rset.getRow()>0){
   TxtNamaBarang.setText(rset.getString("NamaBarang"));
   TxtSatuanBarang.setText(rset.getString("SatuanBarang"));
   TxtHargaBarang.setText(rset.getString("HargaBarang"));
   TxtStockBarang.setText(rset.getString("StockBarang"));
   sta.close();
   rset.close();
 } else {
   sta.close();
   rset.close();
   ClearFormBarang();
   JOptionPane.showMessageDialog(null,"Kode barang tidak ada");
 }
```

Untuk menyimpan data dapat menggunakan pernyataan INSERT, berikut ini contoh penggunaan perintah INSERT :

```
 SQLStatemen = "insert into TbBarang values ('"+TxtKodeBarang.getText()+
"','"+TxtNamaBarang.getText()+"','"+TxtSatuanBarang.getText()+"','"+
TxtHargaBarang.getText()+"','"+TxtStockBarang.getText()+"')";

 sta = cn.createStatement();
 int simpan = sta.executeUpdate(SQLStatemen);
```

```
 if (simpan==1){
   TxtKodeBarang.setText("");
   ClearFormBarang();
   JOptionPane.showMessageDialog(null,"Sudah tersimpan");
 } else {
   JOptionPane.showMessageDialog(null,"Gagal menyimpan data
barang","Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

Sedangkan untuk melakukan update data yang sudah tersimpan, kita dapat menggunakan pernyataan UPDATE seperti berikut ini :

```
 SQLStatemen = "update TbBarang set NamaBarang='"+
TxtNamaBarang.getText()+
     "', SatuanBarang='"+TxtSatuanBarang.getText()+
     "',HargaBarang='"+TxtHargaBarang.getText()+
     "', StockBarang='"+TxtStockBarang.getText()+
     "' where KodeBarang='"+TxtKodeBarang.getText()+"'";
 sta = cn.createStatement();
 int simpan = sta.executeUpdate(SQLStatemen);

 if (simpan==1){
   TxtKodeBarang.setText("");
   ClearFormBarang();
   JOptionPane.showMessageDialog(null,"Sudah tersimpan");
 } else {
   JOptionPane.showMessageDialog(null,"Gagal menyimpan data
barang","Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

Berikut ini adalah contoh melakukan eksekusi perintah SQL dengan menggunakan perintah DELETE.

```
 SQLStatemen = "delete from TbBarang where KodeBarang='"+
TxtKodeBarang.getText()+"'";
 sta = cn.createStatement();
 int simpan = sta.executeUpdate(SQLStatemen);

 if (simpan==1){
   TxtKodeBarang.setText("");
   ClearFormBarang();
   JOptionPane.showMessageDialog(null,"Sudah dihapus");
 } else {
   JOptionPane.showMessageDialog(null,"Gagal menghapus data barang",
"Kesalahan",JOptionPane.ERROR_MESSAGE);
 }
```

17.7. Menutup Koneksi

Penutupan terhadap koneksi database perlu dilakukan agar sumber daya yang digunakan oleh object Connection dapat digunakan lagi oleh proses atau program yang lain.

Sebelum kita menutup koneksi database, kita perlu melepas object Statement dengan kode sebagai berikut :

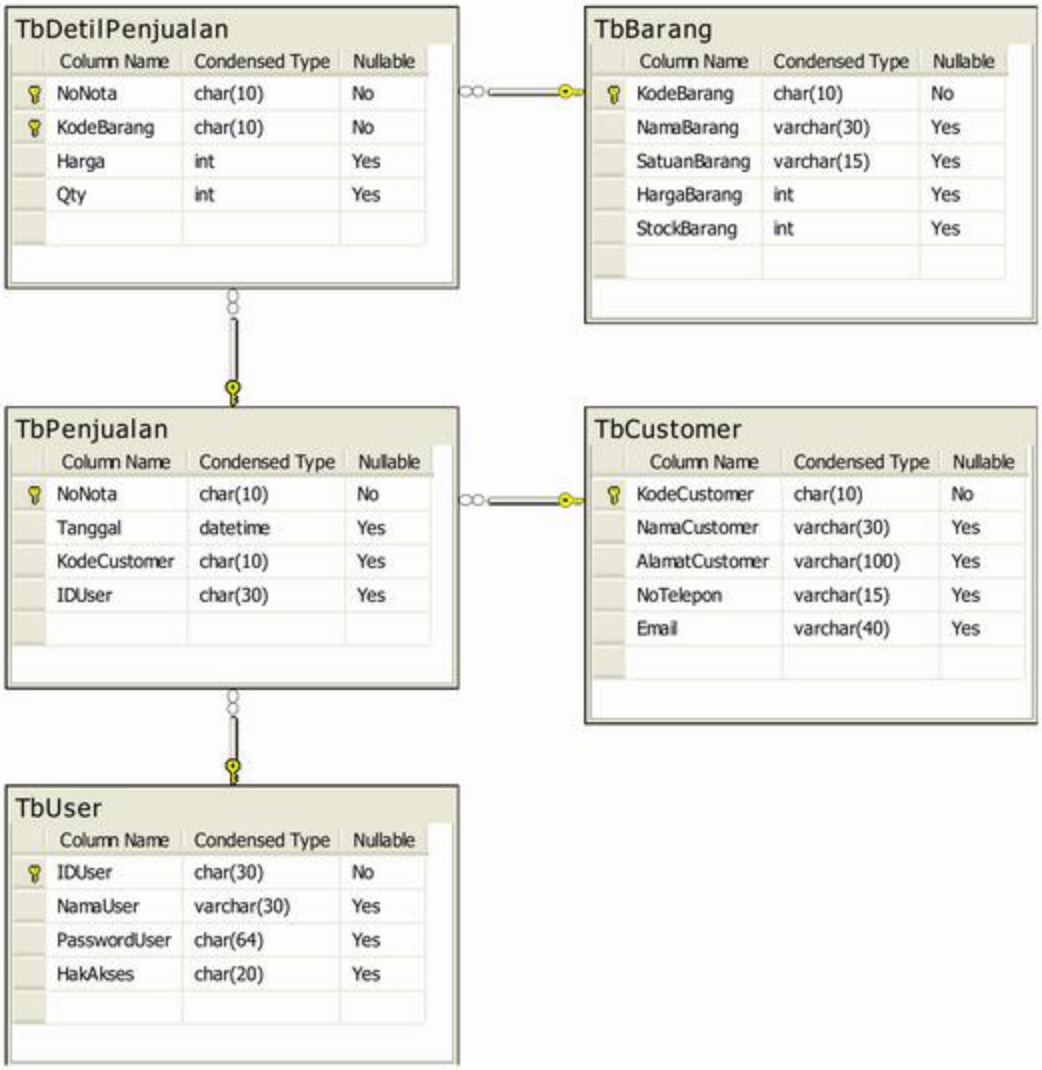
```
 statement.close();
```

Untuk menutup koneksi dengan database server dapat kita lakukan dengan kode sebagai berikut :

```
connection.close();
```

17.8. Contoh Aplikasi

Pertama kita buat database dengan nama DbTokoABC dengan struktur table sebagai berikut :



Gambar 17.1. Struktur tabel dalam database DbTokoABC

Berikut ini adalah Listing program contoh aplikasi (Listing lebih lengkap ada di materi pertemuan 18) :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableColumn;
```

```
 import java.sql.*;

 public class AplikasiToko extends JFrame{
   /*Untuk koneksi ke MySQL*/
   final static String StringDriver="com.mysql.jdbc.Driver";
   final static String
StringConnection="jdbc:mysql://localhost:3306/DbTokoABC?user=root&passwor
d=";

   /*Untuk koneksi ke SQLServer*/
   //final static String
StringDriver="com.microsoft.sqlserver.jdbc.SQLServerDriver";
   //final static String
StringConnection="jdbc:sqlserver://localhost:1433;databaseName=DbTokoABC;
integratedSecurity=true";

   private JDesktopPane frmMDI;
   private JInternalFrame frmBarang;
   private JInternalFrame frmCustomer;
   private JPanel pnlBarang;
   private JPanel pnlCustomer;
   private JPanel pnlUserAccount;

   private JMenuBar MenuBar = new JMenuBar();

   private JMenu MenuMaster = new JMenu("Master Data");
   private JMenuItem MenuBarang = new JMenuItem("Barang"),
           MenuCustomer = new JMenuItem("Customer");

   private JMenu MenuTransaksi = new JMenu("Transaksi");
   private JMenuItem MenuPenjualan = new JMenuItem("Penjualan"),
           MenuPembelian = new JMenuItem("Pembelian");

   private JMenuItem MenuExit = new JMenuItem("Exit");

   /*Komponen untuk Form Barang*/
   private static JLabel LblKodeBarang = new JLabel("Kode Barang");
   private static JTextField TxtKodeBarang = new JTextField();
   private static JLabel LblNamaBarang = new JLabel("Nama Barang");
   private static JTextField TxtNamaBarang = new JTextField();
   private static JLabel LblSatuanBarang = new JLabel("Satuan");
   private static JTextField TxtSatuanBarang = new JTextField();
   private static JLabel LblHargaBarang = new JLabel("Harga Barang");
   private static JTextField TxtHargaBarang = new JTextField();
   private static JLabel LblStockBarang = new JLabel("Stock Barang");
   private static JTextField TxtStockBarang = new JTextField();

   private static JButton TblBarangDelete = new JButton("Delete");
   private static JButton TblBarangSave = new JButton("Save");
   private static JButton TblBarangCancel = new JButton("Cancel");

   /*Komponen untuk Form Customer*/
   private static JLabel LblKodeCustomer = new JLabel("Kode Customer");
   private static JTextField TxtKodeCustomer = new JTextField();
   private static JLabel LblNamaCustomer = new JLabel("Nama Customer");
   private static JTextField TxtNamaCustomer = new JTextField();
   private static JLabel LblAlamatCustomer = new JLabel("Alamat Customer"
);
   private static JTextField TxtAlamatCustomer = new JTextField();
   private static JLabel LblNoTeleponCustomer = new JLabel("No. Telepon"
);
```

```
   private static JTextField TxtNoTeleponCustomer = new JTextField();
   private static JLabel LblEmailCustomer = new JLabel("Email");
   private static JTextField TxtEmailCustomer = new JTextField();

   private static JButton TblCustomerDelete = new JButton("Delete");
   private static JButton TblCustomerSave = new JButton("Save");
   private static JButton TblCustomerCancel = new JButton("Cancel");

   Dimension dimensi = Toolkit.getDefaultToolkit().getScreenSize();

   AplikasiToko(){
     super("Aplikasi Toko");
     setSize((int)(0.7*dimensi.width),(int)(0.7*dimensi.height));
     setLocation(dimensi.width/2-getWidth()/2,dimensi.height/2-
getHeight()/2);
     setDefaultCloseOperation(EXIT_ON_CLOSE);
     frmMDI = new JDesktopPane ();
     frmMDI.setLayout(null);
     //insets = frmMDI.getInsets();
     this.add(frmMDI);

     /*Menambahkan menu MDI*/
     MenuMaster.add(MenuBarang);
     MenuMaster.add(MenuCustomer);
     MenuBar.add(MenuMaster);

     MenuTransaksi.add(MenuPenjualan);
     MenuTransaksi.add(MenuPembelian);
     MenuBar.add(MenuTransaksi);
     MenuBar.add(MenuExit);

     /*Mendeteksi event pada menu*/
     MenuBarang.addActionListener(new MenuHandler());
     MenuCustomer.addActionListener(new MenuHandler());
     MenuExit.addActionListener(new MenuHandler());

     /*Mendeteksi event pada Button di Form Barang*/
     TblBarangDelete.addActionListener(new TombolBarangHandler());
     TblBarangSave.addActionListener(new TombolBarangHandler());
     TblBarangCancel.addActionListener(new TombolBarangHandler());

     /*Mendeteksi event pada Button di Form Customer*/
     TblCustomerDelete.addActionListener(new TombolCustomerHandler());
     TblCustomerSave.addActionListener(new TombolCustomerHandler());
     TblCustomerCancel.addActionListener(new TombolCustomerHandler());

     /*Mendeteksi event pada TxtKodeBarang di Form Barang*/
     TxtKodeBarang.addKeyListener(new KeyAdapter() {
       public void keyPressed(KeyEvent e) {
         int keyCode = e.getKeyCode();
         if (keyCode==KeyEvent.VK_ENTER) {
           /*Mulai mencari data barang*/
           Boolean JDBC_Err=false;
           Connection cn = null;

           try {
             cn = DriverManager.getConnection(StringConnection);
           } catch (Exception ex) {
             JDBC_Err=true;
             JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
```

```
            }

            if (!JDBC_Err){
              try {
                cn = DriverManager.getConnection(StringConnection);
                String SQLStatemen = "Select * from TbBarang where
KodeBarang='"+TxtKodeBarang.getText()+"'";
                Statement sta = cn.createStatement();
                ResultSet rset = sta.executeQuery(SQLStatemen);

                rset.first();
                if (rset.getRow()>0){
                  TxtNamaBarang.setText(rset.getString("NamaBarang"));
                  TxtSatuanBarang.setText(rset.getString("SatuanBarang"));
                  TxtHargaBarang.setText(rset.getString("HargaBarang"));
                  TxtStockBarang.setText(rset.getString("StockBarang"));
                  sta.close();
                  rset.close();
                } else {
                  sta.close();
                  rset.close();
                  ClearFormBarang();
                  JOptionPane.showMessageDialog(null,"Kode barang tidak
ada");
                }
              } catch (Exception ex){
                JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbBarang\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
              }
            }
            /*selesai mencari data barang*/
          }
        }
      });

      /*Mendeteksi event pada TxtKodeCustomer di Form Customer*/
      TxtKodeCustomer.addKeyListener(new KeyAdapter() {
        public void keyPressed(KeyEvent e) {
          int keyCode = e.getKeyCode();
          if (keyCode==KeyEvent.VK_ENTER) {
            /*Mulai mencari data Customer*/
            Boolean JDBC_Err=false;
            Connection cn = null;

            try {
              cn = DriverManager.getConnection(StringConnection);
             } catch (Exception ex) {
              JDBC_Err=true;
              JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
            }

            if (!JDBC_Err){
              try {
                cn = DriverManager.getConnection(StringConnection);
                String SQLStatemen = "Select * from TbCustomer where
KodeCustomer='"+TxtKodeCustomer.getText()+"'";
                Statement sta = cn.createStatement();
                ResultSet rset = sta.executeQuery(SQLStatemen);

                rset.first();
```

```
                if (rset.getRow()>0){
                  TxtNamaCustomer.setText(rset.getString("NamaCustomer"));

TxtAlamatCustomer.setText(rset.getString("AlamatCustomer"));

TxtNoTeleponCustomer.setText(rset.getString("NoTelepon"));
                  TxtEmailCustomer.setText(rset.getString("Email"));
                  sta.close();
                  rset.close();
                } else {
                  sta.close();
                  rset.close();
                  ClearFormCustomer();
                  JOptionPane.showMessageDialog(null,"Kode customer tidak
ada");
                }
              } catch (Exception ex){
                JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbCustomer\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
              }
            }
            /*selesai mencari data customer*/
          }
        }
      });

      setContentPane(frmMDI);

      frmBarang = new JInternalFrame();
      frmBarang.setTitle("Master Data Barang");
      frmCustomer = new JInternalFrame("Master Data Customer");

      /*Pengaturan tampilan Form Barang*/
      pnlBarang = new JPanel ();
      LblKodeBarang.setBounds(30,20,160,20);
      pnlBarang.add(LblKodeBarang);
      TxtKodeBarang.setBounds(120,20,100,20);
      pnlBarang.add(TxtKodeBarang);
      LblNamaBarang.setBounds(30,45,160,20);
      pnlBarang.add(LblNamaBarang);
      TxtNamaBarang.setBounds(120,45,200,20);
      pnlBarang.add(TxtNamaBarang);
      LblSatuanBarang.setBounds(30,70,160,20);
      pnlBarang.add(LblSatuanBarang);
      TxtSatuanBarang.setBounds(120,70,100,20);
      pnlBarang.add(TxtSatuanBarang);
      LblHargaBarang.setBounds(30,95,160,20);
      pnlBarang.add(LblHargaBarang);
      TxtHargaBarang.setBounds(120,95,100,20);
      pnlBarang.add(TxtHargaBarang);
      LblStockBarang.setBounds(30,120,160,20);
      pnlBarang.add(LblStockBarang);
      TxtStockBarang.setBounds(120,120,100,20);
      pnlBarang.add(TxtStockBarang);

      /*Menampilkan tombol di form Barang*/
      TblBarangDelete.setBounds(50,180,80,30);
      pnlBarang.add(TblBarangDelete);
      TblBarangSave.setBounds(140,180,80,30);
      pnlBarang.add(TblBarangSave);
      TblBarangCancel.setBounds(230,180,80,30);
```

```
    pnlBarang.add(TblBarangCancel);
    pnlBarang.setLayout(null);
    frmBarang.add(pnlBarang);

    /*Pengaturan tampilan Form Customer*/
    pnlCustomer = new JPanel ();
    LblKodeCustomer.setBounds(30,20,160,20);
    pnlCustomer.add(LblKodeCustomer);
    TxtKodeCustomer.setBounds(135,20,100,20);
    pnlCustomer.add(TxtKodeCustomer);
    LblNamaCustomer.setBounds(30,45,160,20);
    pnlCustomer.add(LblNamaCustomer);
    TxtNamaCustomer.setBounds(135,45,200,20);
    pnlCustomer.add(TxtNamaCustomer);
    LblAlamatCustomer.setBounds(30,70,160,20);
    pnlCustomer.add(LblAlamatCustomer);
    TxtAlamatCustomer.setBounds(135,70,320,20);
    pnlCustomer.add(TxtAlamatCustomer);
    LblNoTeleponCustomer.setBounds(30,95,160,20);
    pnlCustomer.add(LblNoTeleponCustomer);
    TxtNoTeleponCustomer.setBounds(135,95,200,20);
    pnlCustomer.add(TxtNoTeleponCustomer);
    LblEmailCustomer.setBounds(30,120,160,20);
    pnlCustomer.add(LblEmailCustomer);
    TxtEmailCustomer.setBounds(135,120,160,20);
    pnlCustomer.add(TxtEmailCustomer);

    /*Menampilkan tombol di form Customer*/
    TblCustomerDelete.setBounds(80,180,80,30);
    pnlCustomer.add(TblCustomerDelete);
    TblCustomerSave.setBounds(170,180,80,30);
    pnlCustomer.add(TblCustomerSave);
    TblCustomerCancel.setBounds(260,180,80,30);
    pnlCustomer.add(TblCustomerCancel);
    pnlCustomer.setLayout(null);
    frmCustomer.add(pnlCustomer);

    frmMDI.add(frmBarang);
    frmMDI.add(frmCustomer);

    frmBarang.setBounds(10,10,367,270);
    frmCustomer.setBounds(30,30,500,270);

    setJMenuBar(MenuBar);
    setVisible(true);
  }

  private class MenuHandler implements ActionListener {
    public void actionPerformed(ActionEvent e){
      JMenuItem M = (JMenuItem)e.getSource();

      if (M.getText().equals("Barang")) {
        frmBarang.setVisible(true);
      } else if (M.getText().equals("Customer")) {
         frmCustomer.setVisible(true);
      } else if (M.getText().equals("Exit")) {
        dispose();
      }
    }
  }
```

```
   private void ClearFormBarang(){
     TxtNamaBarang.setText("");
     TxtSatuanBarang.setText("");
     TxtHargaBarang.setText("");
     TxtStockBarang.setText("");
   }

   private void ClearFormCustomer(){
     TxtNamaCustomer.setText("");
     TxtAlamatCustomer.setText("");
     TxtNoTeleponCustomer.setText("");
     TxtEmailCustomer.setText("");
   }

   private class TombolBarangHandler implements ActionListener {
     public void actionPerformed(ActionEvent e){
       JButton TblPilih = (JButton)e.getSource();
       Boolean JDBC_Err = false;

       if (TblPilih.getText().equals("Delete")) {
          /*Mulai menghapus data barang*/
          Connection cn = null;

          try {
            cn = DriverManager.getConnection(StringConnection);
          } catch (Exception ex) {
            JDBC_Err=true;
            JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
          }

          if (!JDBC_Err){
            try {
              cn = DriverManager.getConnection(StringConnection);
              String SQLStatemen = "Select * from TbBarang where
KodeBarang='"+TxtKodeBarang.getText()+"'";
              Statement sta = cn.createStatement();
              ResultSet rset = sta.executeQuery(SQLStatemen);

              rset.first();
              if (rset.getRow()>0){
                sta.close();
                rset.close();

                SQLStatemen = "delete from TbBarang where KodeBarang='"+
TxtKodeBarang.getText()+"'";
                sta = cn.createStatement();
                int simpan = sta.executeUpdate(SQLStatemen);

                if (simpan==1){
                  TxtKodeBarang.setText("");
                  ClearFormBarang();
                  JOptionPane.showMessageDialog(null,"Sudah dihapus");
                } else {
                  JOptionPane.showMessageDialog(null,"Gagal menghapus
data barang","Kesalahan",JOptionPane.ERROR_MESSAGE);
                }
              } else {
                sta.close();
                rset.close();
```

```
                JOptionPane.showMessageDialog(null,"Kode barang tidak
ada");
              }
            } catch (Exception ex){
              JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbBarang\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
            }
          }
          /*Selesai menghapus data barang*/
       } else if (TblPilih.getText().equals("Save")) {
          /*Mulai menyimpan data barang*/
          Connection cn = null;

          try {
            cn = DriverManager.getConnection(StringConnection);
          } catch (Exception ex) {
            JDBC_Err=true;
            JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
          }

          if (!JDBC_Err){
            try {
              cn = DriverManager.getConnection(StringConnection);
              String SQLStatemen = "Select * from TbBarang where
KodeBarang='"+TxtKodeBarang.getText()+"'";
              Statement sta = cn.createStatement();
              ResultSet rset = sta.executeQuery(SQLStatemen);

              rset.first();
              if (rset.getRow()>0){
                sta.close();
                rset.close();
                Object[] arrOpsi = {"Ya","Tidak"};
                int pilih=JOptionPane.showOptionDialog(null,"Kode Barang
sudah ada\nApakah data
diupdate?","Konfirmasi",JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_ME
SSAGE,null,arrOpsi,arrOpsi[0]);
                if (pilih==0){
                  SQLStatemen = "update TbBarang set NamaBarang='"+
TxtNamaBarang.getText()+
                      "', SatuanBarang='"+TxtSatuanBarang.getText()+
                      "',HargaBarang='"+TxtHargaBarang.getText()+
                      "', StockBarang='"+TxtStockBarang.getText()+
                      "' where KodeBarang='"+TxtKodeBarang.getText()+"'";
                  sta = cn.createStatement();
                  int simpan = sta.executeUpdate(SQLStatemen);

                  if (simpan==1){
                    TxtKodeBarang.setText("");
                    ClearFormBarang();
                    JOptionPane.showMessageDialog(null,"Sudah
tersimpan");
                  } else {
                    JOptionPane.showMessageDialog(null,"Gagal menyimpan
data barang","Kesalahan",JOptionPane.ERROR_MESSAGE);
                  }
                }
              } else {
                sta.close();
                rset.close();
```

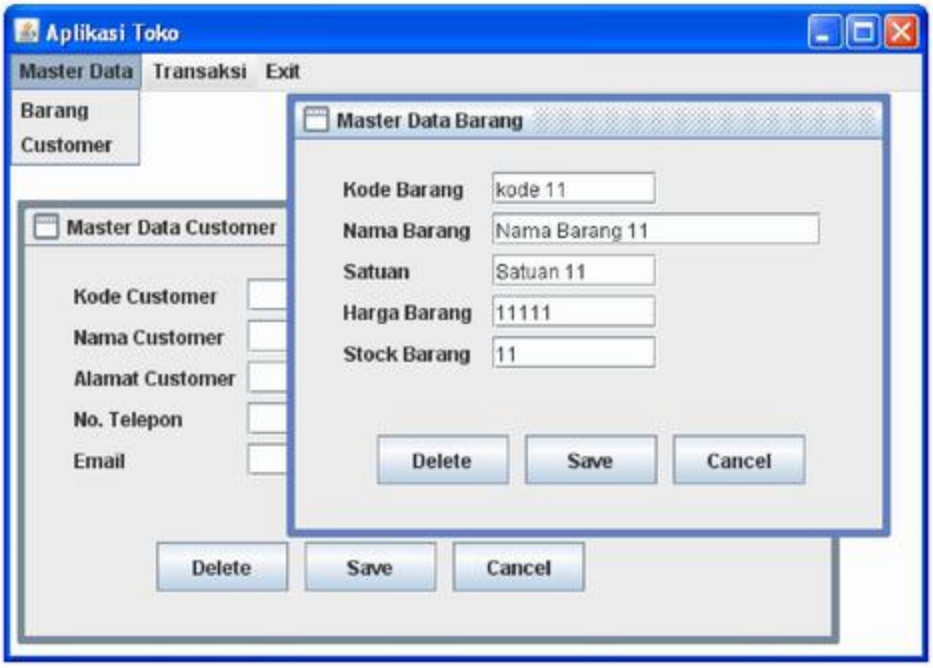
```
                SQLStatemen = "insert into TbBarang values
('"+TxtKodeBarang.getText()+"','"+TxtNamaBarang.getText()+"','"+TxtSatuan
Barang.getText()+"','"+TxtHargaBarang.getText()+"','"+TxtStockBarang.getT
ext()+"')";
                sta = cn.createStatement();
                int simpan = sta.executeUpdate(SQLStatemen);

                if (simpan==1){
                  TxtKodeBarang.setText("");
                  ClearFormBarang();
                  JOptionPane.showMessageDialog(null,"Sudah tersimpan");
                } else {
                  JOptionPane.showMessageDialog(null,"Gagal menyimpan
data barang","Kesalahan",JOptionPane.ERROR_MESSAGE);
                }
              }
            } catch (Exception ex){
              JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbBarang\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
            }
          }
          /*selesai menyimpan data barang*/
        } else if (TblPilih.getText().equals("Cancel")) {
          frmBarang.setVisible(false);
        }
      }
    }

    private class TombolCustomerHandler implements ActionListener {
      public void actionPerformed(ActionEvent e){
        JButton TblPilih = (JButton)e.getSource();
        Boolean JDBC_Err = false;

        if (TblPilih.getText().equals("Delete")) {
          /*Mulai menghapus data Customer*/
          Connection cn = null;

          try {
             cn = DriverManager.getConnection(StringConnection);
          } catch (Exception ex) {
             JDBC_Err=true;
             JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
          }

          if (!JDBC_Err){
            try {
              cn = DriverManager.getConnection(StringConnection);
              String SQLStatemen = "Select * from TbCustomer where
KodeCustomer='"+TxtKodeCustomer.getText()+"'";
              Statement sta = cn.createStatement();
              ResultSet rset = sta.executeQuery(SQLStatemen);

              rset.first();
              if (rset.getRow()>0){
                sta.close();
                rset.close();

                SQLStatemen = "delete from TbCustomer where
KodeCustomer='"+TxtKodeCustomer.getText()+"'";
```

```
                sta = cn.createStatement();
                int simpan = sta.executeUpdate(SQLStatemen);

                if (simpan==1){
                  TxtKodeCustomer.setText("");
                  ClearFormCustomer();
                  JOptionPane.showMessageDialog(null,"Sudah dihapus");
                } else {
                  JOptionPane.showMessageDialog(null,"Gagal menghapus data
customer","Kesalahan",JOptionPane.ERROR_MESSAGE);
                }
              } else {
                sta.close();
                rset.close();

                JOptionPane.showMessageDialog(null,"Kode customer tidak
ada");
              }
            } catch (Exception ex){
              JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbCustomer\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
            }
          }
          /*Selesai menghapus data Customer*/
        } else if (TblPilih.getText().equals("Save")) {
          /*Mulai menyimpan data Customer*/
          Connection cn = null;

          try {
             cn = DriverManager.getConnection(StringConnection);
          } catch (Exception ex) {
             JDBC_Err=true;
             JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
          }

          if (!JDBC_Err){
            try {
              cn = DriverManager.getConnection(StringConnection);
              String SQLStatemen = "Select * from TbCustomer where
KodeCustomer='"+TxtKodeCustomer.getText()+"'";
              Statement sta = cn.createStatement();
              ResultSet rset = sta.executeQuery(SQLStatemen);

              rset.first();
              if (rset.getRow()>0){
                sta.close();
                rset.close();
                Object[] arrOpsi = {"Ya","Tidak"};
                int pilih=JOptionPane.showOptionDialog(null,"Kode customer
sudah ada\nApakah data diupdate?","Konfirmasi",
JOptionPane.YES_NO_OPTION,JOptionPane.QUESTION_MESSAGE,null,arrOpsi,arrOp
si[0]);
                if (pilih==0){
                  SQLStatemen = "update TbCustomer set NamaCustomer='"+
TxtNamaCustomer.getText()+
                       "', AlamatCustomer='"+TxtAlamatCustomer.getText()+
                       "', NoTelepon='"+TxtNoTeleponCustomer.getText()+
                       "', Email='"+TxtEmailCustomer.getText()+
                       "' where KodeCustomer='"+TxtKodeCustomer.getText()+
"'";
```

```
                sta = cn.createStatement();
                int simpan = sta.executeUpdate(SQLStatemen);

                if (simpan==1){
                  TxtKodeCustomer.setText("");
                  ClearFormCustomer();
                  JOptionPane.showMessageDialog(null,"Sudah tersimpan");
                } else {
                  JOptionPane.showMessageDialog(null,"Gagal menyimpan
data customer","Kesalahan",JOptionPane.ERROR_MESSAGE);
                }
              }
            } else {
              sta.close();
              rset.close();

              SQLStatemen = "insert into TbCustomer values ('"+
TxtKodeCustomer.getText()+"','"+TxtNamaCustomer.getText()+"','"+
TxtAlamatCustomer.getText()+"','"+TxtNoTeleponCustomer.getText()+
"','"+TxtEmailCustomer.getText()+"')";
              sta = cn.createStatement();
              int simpan = sta.executeUpdate(SQLStatemen);

              if (simpan==1){
                TxtKodeCustomer.setText("");
                ClearFormCustomer();
                JOptionPane.showMessageDialog(null,"Sudah tersimpan");
              } else {
                JOptionPane.showMessageDialog(null,"Gagal menyimpan data
customer","Kesalahan",JOptionPane.ERROR_MESSAGE);
              }
            }
          } catch (Exception ex){
            JOptionPane.showMessageDialog(null,"Tidak dapat membuka
tabel TbCustomer\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
          }
        }
        /*selesai menyimpan data Customer*/
      } else if (TblPilih.getText().equals("Cancel")) {
        frmCustomer.setVisible(false);
      }
    }
  }

  public static void main(String args[]){
    AplikasiToko frameku = new AplikasiToko();
    Boolean JDBC_Err = false;

    try{
      Class.forName(StringDriver);
    } catch (Exception ex){
      JDBC_Err=true;
      JOptionPane.showMessageDialog(null,"JDBC Driver tidak ditemukan
atau rusak\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
    }

    if (!JDBC_Err){
      try {
        Connection cn = DriverManager.getConnection(StringConnection);
      } catch (Exception ex) {
```

```
            JOptionPane.showMessageDialog(null,"Koneksi ke database
DbTokoABC gagal\n"+ex,"Kesalahan",JOptionPane.ERROR_MESSAGE);
        }
      }
    }
  }
```

Tampilan dari program diatas adalah :



Gambar 17.2. Tampilan aplikasi database