

LAPORAN APLIKASI TOKO KUE

Mata Kuliah: Pemrograman Berorientasi Objek

Dosen Pengampu: Dionisia Bhisetya Rarasati, S.Kom., M.T.I



Bridging Education To The Real World

Disusun oleh:

31200040 – Phangestin Jen

Kelas 3PSI51

Prodi Sistem Informasi

Fakultas Teknologi dan Desain

UNIVERSITAS BUNDA MULIA

2021

Java Class Koneksi

```
13 public class Koneksi {
14
15     static Connection koneksi;
16     public static Statement stm;
17
18     public static Connection GetConnection() throws SQLException {
19         try {
20             Class.forName("com.mysql.jdbc.Driver");
21             koneksi = (Connection) DriverManager.getConnection("jdbc:mysql://localhost/dbtoko", "root", "");
22             stm = koneksi.createStatement();
23             System.out.println("koneksi berhasil");
24         } catch (Exception e) {
25             JOptionPane.showMessageDialog(null, "Koneksi Gagal!" + e.getMessage());
26         }
27         return koneksi;
28     }
29
30     static PreparedStatement prepareStatement(String sql) {
31         throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools |
32     }
33 }
34
```

Merupakan tampilan dari isi Java Class Koneksi yang digunakan untuk dapat terhubung ke database MySQL. Pada baris 15 sintaks Connection yang terdapat pada import library ditampilkan dengan variabel koneksi, begitupun dengan baris 16. Pada baris 20 merupakan penggunaan nama library database yang digunakan yaitu com.mysql.jdbc.Driver yang kemudian pada baris 21 variabel koneksi dihubungkan ke alamat localhost database dbtoko dengan username root dan password "". Apabila koneksi ke database berhasil, maka akan muncul ke sistem dengan output "koneksi berhasil", dan pada baris ke 27 mengembalikan nilai dari variabel koneksi.

JFrameForm Login :

```
12 public class Login extends javax.swing.JFrame {
13
14     Connection con;
15     Statement stat;
16     ResultSet rs;
17     String sql;
18
19     private void kosong() {
20         txt_user.setText("");
21         txt_pass.setText("");
22     }
23
24     public Login() throws SQLException {
25         initComponents();
26         Koneksi DB = new Koneksi();
27         DB.GetConnection();
28         con = DB.koneksi;
29         stat = DB.stm;
30     }
31 }
```

Line 14 hingga 17 merupakan pendeklarasian yang dipakai untuk dapat terhubung kedalam Java Class Koneksi. Pada line 19 terdapat suatu method dengan sifatnya private yang artinya hanya dapat diakses di kelas itu sendiri. Kegunaan dari private void Kosong() pada JFrameForm Login ialah untuk mengatur ulang tampilan text field menjadi kosong ketika membuka aplikasi ataupun ketika ingin menginputkan ulang data inputan sehingga user tidak perlu melakukan penghapusan manual ketika ingin melakukan penginputan.

Selanjutnya pada baris 24 merupakan method yang digunakan agar JFrameForm tersebut dapat dikoneksikan ke database sehingga apabila user menginputkan username dan password yang sesuai dengan data, maka akan langsung dapat mengakses ke database.

```
153 private void loginActionPerformed(java.awt.event.ActionEvent evt) {  
154     try {  
155         sql = "SELECT * FROM tblogin WHERE username='"+txt_user.getText()+"' AND password='"+txt_pass.getText()+"'";  
156         rs = stat.executeQuery(sql);  
157         if (rs.next()) {  
158             if (txt_user.getText().equals(rs.getString("username")) && txt_pass.getText().equals(rs.getString("password"))) {  
159                 JOptionPane.showMessageDialog(null, "Anda berhasil Login");  
160                 new MenuUtama().setVisible(true);  
161                 this.setVisible(false);  
162             }  
163         }  
164     }  
165     sql = "SELECT * FROM tbsignup WHERE username='"+txt_user.getText()+"' AND password='"+txt_pass.getText()+"'";  
166     rs = stat.executeQuery(sql);  
167     if (rs.next()) {  
168         if (txt_user.getText().equals(rs.getString("username")) && txt_pass.getText().equals(rs.getString("password"))) {  
169             JOptionPane.showMessageDialog(null, "Anda berhasil Login");  
170             new MenuUtama().setVisible(true);  
171             this.setVisible(false);  
172         }  
173     }  
174     else {  
175         JOptionPane.showMessageDialog(null, "Username atau Password salah");  
176         kosong();  
177     }  
178 } catch (Exception e) {  
180     JOptionPane.showMessageDialog(this, "Gagal login");  
181 }  
182 }
```

Masih dalam JFrameForm untuk Login. Method private loginActionPerformed tersebut merupakan method yang secara otomatis muncul ketika kita melakukan desain pada Form dengan menggunakan Button. Pada baris 155 hingga seterusnya merupakan sintaks yang digunakan pada MySQL yang apabila user menginputkan username dan password sesuai dengan data yang terdaftar dalam database MySQL, maka akan muncul text dialog “Anda Berhasil Login”. Kemudian apabila terdapat kata atau huruf yang salah pada username maupun password, maka akan muncul text box “Username atau Password salah” pada line 175. Penggunaan try selalu diikuti dengan penggunaan catch, dimana pada baris 179 terdapat catch yang artinya ketika tidak berhasil terkoneksi ke database, maka akan muncul text dialog Gagal Login.

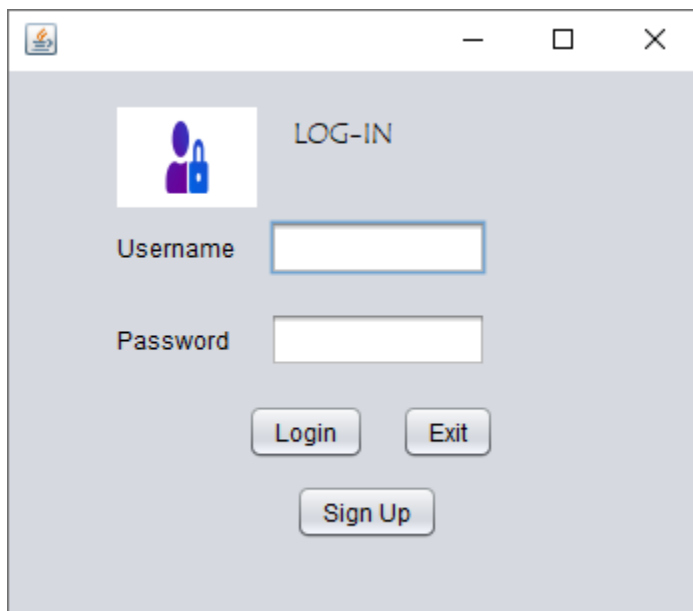
```
149 private void exitActionPerformed(java.awt.event.ActionEvent evt) {  
150     dispose();  
151 }  
152
```

Pada method button tersebut, ketika user menekan button exit, maka program akan melakukan dispose(); yang kegunaannya adalah untuk melakukan tindakan keluar atau menutup program.

```
184 private void signupActionPerformed(java.awt.event.ActionEvent evt) {  
185     try {  
186         new MenuSignUp().setVisible(true);  
187     } catch (SQLException ex) {  
188         Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);  
189     }  
190     dispose();  
191 }
```

Pada method tersebut merupakan aksi yang akan dilakukan program ketika user menekan tombol signup, maka program akan mengarahkan ke Form MenuSignUp yang nantinya akan di tampilkan menggunakan setVisible(true). Kemudian akan dilakukan dispose atau keluar dari program login secara otomatis dan mengganti tampilan ke signup.

Tampilan menu login :



JFrameForm MenuSignUp

```
10 public class MenuSignUp extends javax.swing.JFrame {
11
12     String Username;
13     String Password;
14     String rePassword;
15     int panjangUser = 55;
16
17     Connection con;
18     Statement stat;
19     ResultSet rs;
20
```

Merupakan pendeklarasian variable yang nantinya variable-variabel pada baris 12 hingga 15 digunakan sebagai variable penampung dari inputan user pada text field. Kemudian variable dari baris 17 hingga 19 merupakan variable yang dipakai untuk dapat mengakses ke data table yang terdapat pada database.

```
21 public MenuSignUp() throws SQLException {
22     initComponents();
23     Koneksi DB = new Koneksi(); //Menginisialisasi Java class Koneksi sebagai DB
24     DB.getConnection(); //Melakukan pemanggilan method getConnection(); yang terdapat pada kelas Koneksi
25     con = DB.koneksi; //menggunakan variabel con sebagai koneksi ke database
26     stat = DB.stm;
27 }
28
29 public void bersih() {
30     usertext.setText("");
31     passtext.setText("");
32     repasstext.setText("");
33 }
```

Terdapat method MenuSignUp yang merupakan penghubung agar terkoneksi ke database. Pada baris 23 merupakan penginisialisasian java class Koneksi dengan variable DB. Kemudian dilakukan pemanggilan method getConnection() yang terdapat dalam kelas Koneksi yang selanjutnya menggunakan variable con sebagai penampung DB.koneksi ke database.

Juga terdapat method `bersih()` yang berguna ketika user membuka ulang program atau telah selesai melakukan penginputan data, maka dilakukan pemanggilan method ini agar text field kembali menjadi kosong seperti semula tanpa user harus menghapusnya manual.

```
148 private void exitActionPerformed(java.awt.event.ActionEvent evt) {  
149     try {  
150         new Login().setVisible(true);  
151     } catch (SQLException ex) {  
152         Logger.getLogger(MenuSignUp.class.getName()).log(Level.SEVERE, null, ex);  
153     }  
154     dispose();  
155 }
```

Merupakan sintaks dari button exit yang digunakan pada tampilan yang nantinya ketika user menekan button exit, maka akan kembali ke menu login yang kemudian menu sign up yang sebelumnya akan dilakukan `dispose()`; atau keluar sehingga berganti menjadi tampilan login.

```
157 private void saveActionPerformed(java.awt.event.ActionEvent evt) {  
158     Password = passtext.getText();  
159     rePassword = repasstext.getText();  
160     panjangUser=usertext.getText().length();  
161  
162     try {  
163         if (rePassword.equals(Password) && panjangUser>=2) {  
164             stat.executeUpdate("Insert into tbsignup values(" +  
165                 "'" + usertext.getText() + "'," +  
166                 "'" + String.valueOf(passtext.getPassword()) + "'," +  
167                 "'" + String.valueOf(repasstext.getPassword()) + "'" +  
168                 ");");  
169             {  
170                 JOptionPane.showMessageDialog(null, "Save berhasil");  
171             }  
172             bersih();  
173         } else {  
174             JOptionPane.showMessageDialog(null, "Akses ditolak");  
175             bersih();  
176         }  
177     } catch (Exception e) {  
178         JOptionPane.showMessageDialog(this, "Error");  
179     }  
180 }
```

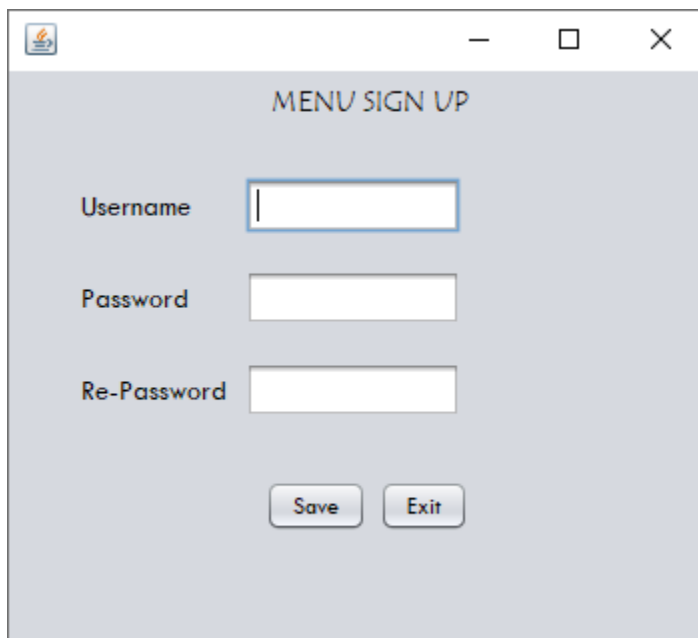
Merupakan codingan dari button Save yang terdapat pada tampilan signup. Baris 158 hingga 160 merupakan variable penampung yang nantinya akan diisi dengan inputan user yang berada pada text field. Setelahnya, terdapat percabangan yang mengharuskan user menginputkan username, password, repassword dengan minimal 2 huruf kemudian pada baris 164 merupakan perintah eksekusi ke database berupa update yang disertai dengan sintaks kueri pada MySQL yaitu menyisipkan ke tabel tbsignup yang terdapat pada database dengan inputan user. Jika data pada database berhasil diinputkan, maka akan muncul GUI “Save Berhasil” yang kemudian seluruh text field kembali menjadi kosong dengan perintah pemanggilan `bersih()`, namun ketika seluruh inputan textfield lebih kecil dari 2 huruf, maka akan muncul GUI “Akses ditolak”. Selanjutnya

pada baris 177 terdapat perintah catch yang akan tampil GUI “Error” apabila terjadi kesalahan pada penginputan data field ke MySQL.

```
182 public static void main(String args[]) {  
183     /* Set the Nimbus look and feel */  
184     Look and feel setting code (optional)  
205  
206     /* Create and display the form */  
207     java.awt.EventQueue.invokeLater(new Runnable() {  
208         public void run() {  
209             try {  
210                 new MenuSignUp().setVisible(true);  
211             } catch (SQLException ex) {  
212                 Logger.getLogger(MenuSignUp.class.getName()).log(Level.SEVERE, null, ex);  
213             }  
214         }  
215     });  
216 }
```

Merupakan gerbang utama dalam menjalankan JFrameForm MenuSignUp yang pada method run() akan dilakukan pemanggilan MenuSignUp agar tampil ke layar dengan menggunakan setVisible.

Tampilan MenuSignUp:



The screenshot shows a Java Swing window titled "MENU SIGN UP". The window has a light gray background and a standard title bar with minimize, maximize, and close buttons. Inside the window, there are three text input fields arranged vertically. The first field is labeled "Username", the second "Password", and the third "Re-Password". Below these fields are two buttons: "Save" and "Exit". The "Username" field is currently selected, indicated by a blue border and a cursor.

JFrameForm MenuUtama

```
14 public class MenuUtama extends javax.swing.JFrame {
15
16
17     public MenuUtama() {
18         initComponents();
19     }
20
21     @SuppressWarnings("unchecked")
22     Generated Code
107
108     private void jMenuItem1ActionPerformed(java.awt.event.ActionEvent evt) {
109         DataKaryawan karyawan = new DataKaryawan();
110         jDesktopPane2.add(karyawan);
111         karyawan.setVisible(true);
112     }
113
114     private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
115         Transaksi transaksi = new Transaksi();
116         jDesktopPane2.add(transaksi);
117         transaksi.setVisible(true);
118     }
```

Pada JFrameForm MenuUtama di baris 17 merupakan fungsi untuk pemanggilan agar terkoneksi dengan javaclass koneksi yang menghubungkan ke database MySQL. Pada MenuUtama terdapat button menuitem1 yang merupakan isi dari menu bar yang ketika di klik akan diarahkan ke JInternalFrameForm DataKaryawan untuk menginputkan data karyawan. Kemudian penggunaan jDesktopPane2 ialah untuk menampilkan desain dari karyawan.

Selanjutnya pada menuitem2 yang terdapat pada menu bar, ketika di klik akan langsung diarahkan ke JInternalFrameForm Transaksi yang digunakan untuk melakukan transaksi pembayaran atau pemesanan kue.


```

120 private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
121     DataTransaksi dt = null;
122     try {
123         dt = new DataTransaksi();
124     } catch (SQLException ex) {
125         Logger.getLogger(MenuUtama.class.getName()).log(Level.SEVERE, null, ex);
126     }
127     jDesktopPane2.add(dt);
128     dt.setVisible(true);
129 }
130
131 public static void main(String args[]) {
132     /* Set the Nimbus look and feel */
133     Look and feel setting code (optional)
134
135     /* Create and display the form */
136     java.awt.EventQueue.invokeLater(new Runnable() {
137         public void run() {
138             try {
139                 new Login().setVisible(true);
140             } catch (SQLException ex) {
141                 Logger.getLogger(MenuUtama.class.getName()).log(Level.SEVERE, null, ex);
142             }
143         }
144     });
145 }
146

```

Kemudian pada MenuUtama terdapat menuItem3 pada menu bar yang berguna untuk menampilkan laporan data transaksi yang telah terjadi. Ketika di klik akan diarahkan pada JInternalFrameForm DataTransaksi yang ditampilkan dengan variable dt sesuai pada baris 123.

Tampilan MenuUtama :



JInternalFrameForm Transaksi

```
13 public class Transaksi extends javax.swing.JInternalFrame {
14
15     private void kosong() {
16         nama1.setText("");
17         nama2.setText("");
18         nama3.setText("");
19         nama4.setText("");
20         harga1.setText("");
21         harga2.setText("");
22         harga3.setText("");
23         harga4.setText("");
24         banyak1.setText("");
25         banyak2.setText("");
26         banyak3.setText("");
27         banyak4.setText("");
28         total1.setText("");
29         total2.setText("");
30         total3.setText("");
31         total4.setText("");
32         totalbiaya.setText("");
33         bayar.setText("");
34         kembali.setText("");
35     }
36
37     public Transaksi() {
38         initComponents();
39     }
40 }
```

JInternalFrameForm Transaksi merupakan tampilan yang akan muncul ketika user melakukan klik pada menu item transaksi di dalam menu bar. Pada baris 15 terdapat method bernama `kosong()` yang berguna untuk ketika user selesai melakukan penginputan transaksi maka variable-variabel yang merupakan textfield akan di set kembali kosong seperti awal.

Baris 37 merupakan method untuk penghubung ke kelas Koneksi yang digunakan agar dapat terkoneksi ke database.

```

392 private void ProsesHitungActionPerformed(java.awt.event.ActionEvent evt) {
393     int a1 = Integer.parseInt(harga1.getText());
394     int b1 = Integer.parseInt(banyak1.getText());
395     int t1 = a1*b1;
396     total1.setText(""+t1);
397     int a2 = Integer.parseInt(harga2.getText());
398     int b2 = Integer.parseInt(banyak2.getText());
399     int t2 = a2*b2;
400     total2.setText(""+t2);
401     int a3 = Integer.parseInt(harga3.getText());
402     int b3 = Integer.parseInt(banyak3.getText());
403     int t3 = a3*b3;
404     total3.setText(""+t3);
405     int a4 = Integer.parseInt(harga4.getText());
406     int b4 = Integer.parseInt(banyak4.getText());
407     int t4 = a4*b4;
408     total4.setText(""+t4);
409     int c1 = Integer.parseInt(total1.getText());
410     int c2 = Integer.parseInt(total2.getText());
411     int c3 = Integer.parseInt(total3.getText());
412     int c4 = Integer.parseInt(total4.getText());
413     int hasil = c1+c2+c3+c4;
414     totalbiaya.setText(""+hasil);
415 }
416
417 private void NewActionPerformed(java.awt.event.ActionEvent evt) {
418     kosong();
419 }

```

Terdapat button bernama Proses Hitung pada desain Transaksi yang ketika user menekan button tersebut, maka akan dilakukan proses perhitungan. Variabel-variabel text field yang sudah terisi akan diubah menjadi integer menggunakan sintaks `Integer.parseInt` karena dalam transaksi, user haruslah menginputkan angka yang terkait dengan jumlah pemesanan. Baris 393, `a1` merupakan variable penampung dari penginputan text field pada variable `harga1` dan begitupun variable integer `b1`. Ketika `a1` dan `b1` sudah terisi, maka akan dilakukan perhitungan dengan rumus $a1 * b1$ yang hasilnya akan ditampung pada variable `t1` sesuai dengan baris 395. Selanjutnya, ketika sudah diproses perhitungannya maka variable text field `total1` akan di set isinya dengan memunculkan total perhitungannya. Begitupun seterusnya hingga baris 412. Kemudian pada baris 413 dari total-total yang sudah dilakukan perhitungannya akan di jumlahkan seluruhnya dengan hasilnya ditampung dalam variable `hasil` yang kemudian pada baris berikutnya, text field dengan variable `totalbiaya` akan di set untuk memunculkan hasil dari perhitungan dengna memanggil variable `hasil`.

Pada baris 417 terdapat button New yang ketika user menekan button tersebut maka akan kembali seperti semula dimana semua transaksi yang dilakukan akan kembali kosong seperti awal.

```

421 private void hitungActionPerformed(java.awt.event.ActionEvent evt) {
422     int a1 = Integer.parseInt(totalbiaya.getText());
423     int a2 = Integer.parseInt(bayar.getText());
424     int hasil = a2 - a1;
425     kembali.setText(""+hasil);
426
427     if(a2 < a1) {
428         JOptionPane.showMessageDialog(null, "Uang yang anda inputkan tidak mencukupi!");
429         kosong();
430     }
431 }
432
433 private void paket1ActionPerformed(java.awt.event.ActionEvent evt) {
434     nama1.setText("Bika Ambon");
435     harga1.setText("15000");
436     nama2.setText("Klepon");
437     harga2.setText("2500");
438     nama3.setText("Kue ku");
439     harga3.setText("6000");
440     nama4.setText("Kue Lumpur");
441     harga4.setText("8000");
442 }

```

Terdapat button hitung pada Transaksi yang berguna untuk menghitung total harga seluruh transaksi yang dilakukan dan total pembayaran yang dilakukan oleh customer. Pada baris 422 terdapat variable a1 sebagai penampung dari hasil konversi text field pada totalbiaya. Dan pada baris 423 terdapat variable a2 sebagai penampung dari hasil konversi text field pada bayar yang diinputkan user. Kemudian nantinya variable totalbiaya akan diselisihkan dengan bayar yang berarti harga dari seluruh pemesanan akan dikurangi dengan jumlah uang pembayaran yang dilakukan customer. Kemudian terdapat variable textfield kembali yang akan menunjukkan informasi jumlah uang kembali apabila uang pembayaran lebih besar. Pada baris 427 terdapat percabangan yang menunjukkan apabila user menginputkan nominal pembayaran lebih kecil daripada total pembayaran yang harus dibayar, maka akan muncul GUI “Uang yang anda inputkan tidak mencukupi!”

Pada menu Transaksi, took kue ini memiliki 5 paket kue lokal dengan total 20 menu yang di setiap paket memiliki 4 menu. Pada baris 433 merupakan button untuk paket 1 yang ketika user menekan button tersebut maka text field nama dan harga akan secara otomatis terisi dengan isi dari paket tersebut. Begitupun paket kue 2 hingga paket kue 5.

```

488 private void pesanActionPerformed(java.awt.event.ActionEvent evt) {
489     VerifikasiCustomer verifikasi = new VerifikasiCustomer();
490     verifikasi.setVisible(true);
491 }
492
493 private void backActionPerformed(java.awt.event.ActionEvent evt) {
494     new MenuUtama().setVisible(true);
495     this.setVisible(false);
496 }

```

Dalam Transaksi, terdapat button pesan yang ketika user telah menginputkan seluruh total pembayaran, maka harus menekan button pesan yang nantinya akan diarahkan kepada JInternalFrameForm VerifikasiCustomer().

Kemudian dalam Transaksi juga terdapat button back yang ketika user menekan button tersebut akan kembali ke MenuUtama() .

Tampilan Menu Transaksi :

ProgramToko

Data Transaksi Report

Back

PAKET KUE

PAKET KUE 1 PAKET KUE 2 PAKET KUE 3 PAKET KUE 4 PAKET KUE 5

NAMA	HARGA	BANYAKNYA	TOTAL
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

TOTAL HARGA

BAYAR

KEMBALI

PROSES TRANSAKSI

HITUNG

New PESAN

User hanya perlu menginputkan banyaknya dan bayar saja, untuk nama, harga, total, total_harga, dan kembali secara otomatis akan muncul ketika user menekan tombol button. Pilih paket > input banyaknya > klik button proses transaksi > input pembayaran > klik button hitung > kemudian klik button pesan.

JFrameForm VerifikasiCustomer

```
13 public class VerifikasiCustomer extends javax.swing.JFrame {
14
15     private void kosong() {
16         txtkode.setText (null);
17         txtnama.setText (null);
18         txtalamat.setText (null);
19         txttanggal.setText (null);
20         txtno.setText (null);
21         txttotal.setText (null);
22     }
```

Pada JFrameForm VerifikasiCustomer terdapat method kosong() yang digunakan agar tampilan dari text field yang sudah diberikan nama variable-variabel seperti pada baris ke 16 hingga 21 ketika di tampilkan ke layar maupun setelah melakukan penginputan, textfield tersebut akan kembali kosong seperti semula.

```
24 public void datatable() {
25     DefaultTableModel tbl = new DefaultTableModel();
26     tbl.addColumn("Kode");
27     tbl.addColumn("Nama");
28     tbl.addColumn("Tanggal Pembelian");
29     tbl.addColumn("Pilihan Paket");
30     tbl.addColumn("Alamat");
31     tbl.addColumn("No. Telepon");
32     tbl.addColumn("Pembayaran");
33     table.setModel(tbl);
34     try {
35         Statement statement = (Statement)Koneksi.GetConnection().createStatement();
36         ResultSet res = statement.executeQuery("SELECT * from tbcustomer");
37         while(res.next()) {
38             tbl.addRow(new Object[] {
39                 res.getString("Kode"),
40                 res.getString("Nama"),
41                 res.getString("TanggalP"),
42                 res.getString("PilihanPaket"),
43                 res.getString("Alamat"),
44                 res.getString("No.Telepon"),
45                 res.getString("TotalPembayaran")
46             });
47             table.setModel(tbl);
48         }
49     } catch (Exception e) {
50         JOptionPane.showMessageDialog(rootPane, "Salah");
51     }
52 }
```

Pada VerifikasiCustomer() terdapat tabel yang berguna untuk menampilkan data transaksi yang terjadi. Method yang digunakan untuk table tersebut ialah menggunakan datatable() seperti pada baris ke-24. Pada baris ke-25, sintaks DefaultTableModel(); yang merupakan sintaks default dari import library yang digunakan untuk melakukan perubahan pada isi tabel yang kemudian di tampung menggunakan variabel tbl. Pada baris 26 hingga 32 merupakan proses pemberian nama kolom pada tabel yang kemudian pada baris ke 33, tabel yang sudah di desain yang diberi variable table di set sebagai model dari tbl sehingga pada desain dengan variable name table

berisi kolom yang sudah kita berikan namanya. Selanjutnya pada baris 35 adalah sintaks untuk menghubungkan ke koneksi database yang kemudian jika berhasil, maka akan melakukan `SELECT * FROM tbcustomer` yang akan mengakses `tbcustomer` yang terdapat pada database. Kemudian pada baris 39 hingga 45 dilakukan penghubung antar kolom yang telah kita buat pada baris 26 sampai 32 yang nanti isinya akan dihubungkan kedalam database sesuai dengan kolomnya. Ketika terdapat kesalahan saat melakukan koneksi ke database, maka akan muncul GUI “Salah” seperti pada baris 50.

```

272 private void submitActionPerformed(java.awt.event.ActionEvent evt) {
273     String kode = txtkode.getText();
274     String nama = txtnama.getText();
275     String tgl = txttanggal.getText();
276     String alamat = txtalamat.getText();
277     String telepon = txtno.getText();
278     String pilihan = null;
279     String total = txttotal.getText();
280     if(paket1.isSelected()) {
281         pilihan = "Paket 1";
282     }
283     else if (paket2.isSelected()) {
284         pilihan = "Paket 2";
285     }
286     else if (paket3.isSelected()) {
287         pilihan = "Paket 3";
288     }
289     else if (paket4.isSelected()) {
290         pilihan = "Paket 4";
291     }
292     else if (paket5.isSelected()) {
293         pilihan = "Paket 5";
294     }
295
296     try {
297         Statement statement = (Statement) Koneksi.GetConnection().createStatement();
298         statement.executeUpdate("Insert into tbcustomer VALUES ('"+kode + "','" + nama + "','" + tgl + "','" + pilihan + "','"
299             + alamat + "','" + telepon + "','" + total + "')");
300         statement.close();

```

Pada VerifikasiCustomer terdapat tombol submit yang ketika user telah menginputkan data yang terdapat pada text field yang nantinya akan ditampung kedalam variable pada baris 273 hingga 279. Untuk variable paket1 hingga paket5 karena pada desain ialah menggunakan radio button yang sudah dilakukan pengelompokkan, maka terdapat percabangan yang user hanya dapat memilih salah satu dari 5 radio button tersebut. Dimana ketika user telah memilih paket sesuai dengan data transaksi, maka variable pilihan akan di set dengan data salah satu dari paket1 hingga paket5.


```

296     try {
297         Statement statement = (Statement) Koneksi.GetConnection().createStatement();
298         statement.executeUpdate("Insert into tbcustomer VALUES ('"+kode + "','" + nama + "','" + tgl + "','" + pilihan + "','"
299             + alamat + "','" + telepon + "','" + total + "')");
300         statement.close();
301         JOptionPane.showMessageDialog(null, "Terima kasih sudah memesan, pesanan Anda akan segera kami proses!");
302     } catch (Exception e) {
303         JOptionPane.showMessageDialog(null, "Maaf terjadi kesalahan dalam verifikasi, silahkan ulangi kembali. ");
304     }
305     datatable();
306     kosong();
307 }
308
309 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
310     new MenuUtama().setVisible(true);
311     this.setVisible(false);
312 }
313
314
315 public static void main(String args[]) {
316     /* Set the Nimbus look and feel */
317     Look and feel setting code (optional)
318
319     /* Create and display the form */
320     java.awt.EventQueue.invokeLater(new Runnable() {
321         public void run() {
322             new VerifikasiCustomer().setVisible(true);
323         }
324     });
325 }


```

Pada baris 297 terdapat sintaks yang digunakan agar terhubung ke koneksi database yang kemudian pada baris 298 dilakukan update dengan sintaks kueri MySQL berupa penambahan data field yang diisi dengan variable kode, nama, tgl, pilihan, alamat, telepon, dan total, dimana variable-variabel tersebut merupakan inputan yang telah dilakukan dalam verifikasi customer. Setelah menekan tombol submit, maka statement atau verifikasi telah selesai dilakukan, dan akan muncul GUI “Terima kasih sudah memesan, pesanan akan segera di proses.” Seperti pada baris 301. Ketika terjadi kesalahan, maka terdapat GUI yang menampilkan pesan kesalahan. Setelah selesai menekan submit, maka tabel yang terdapat dalam VerifikasiCustomer akan tampil data-data dari customer yang telah melakukan transaksi dengan pemanggilan method datatable() pada baris 305. Kemudian pada baris 306 memanggil variable kosong() yang berarti ketika user sudah selesai melakukan penginputan maka text field akan kembali kosong seperti semula.

Pada VerifikasiCustomer juga terdapat button kembali yang ketika user menekan tombol tersebut akan diarahkan kepada MenuUtama() sesuai dengan baris 310.

Pada baris 315 merupakan gerbang utama ketika user telah selesai melakukan transaksi, maka secara otomatis akan memanggil VerifikasiCustomer.

Tampilan Verifikasi Customer :



SILAHKAN ISI DATA UNTUK VERIFIKASI CUSTOMER

Kode Customer

Nama

Tanggal (dd/mm...

Pilihan Paket

☐ Paket 1 ☐ Paket 2 ☐ Paket 3 ☒ Paket 4 ☐ Paket 5

Alamat

No.Telepon

Nominal Pembayaran

SUBMIT

Kode	Nama	Tanggal P...	Pilihan Paket	Alamat	No. Telepon	Pembayaran
k-01	gegege	27 desem...	Paket 3	qweqwe	01341341	45000
k-02	qweqwe	31/12/2012	Paket 2	erqweqweq	1231231	95000
k-03	Phangestin...	29/12/2021	Paket 4	Jl.Semana...	0812649844	120000

Ketika user telah menginputkan data verifikasi, maka akan muncul namanya pada tabel.

JInternalFrameForm DataTransaksi

```
19 public DataTransaksi() throws SQLException {
20     initComponents();
21     Koneksi DB = new Koneksi();
22     DB.GetConnection();
23     con = DB.koneksi;
24     stat = DB.stm;
25
26     Object[] row= {"Kode", "Nama", "Tanggal Pembelian", "Pilihan Paket", "Alamat", "No. Telp", "Total Bayar"};
27     tabMode = new DefaultTableModel (null, row);
28     table.setModel(tabMode);
29
30     MenampilkanData();
31 }
```

Terdapat method yang merupakan penghubung ke koneksi database. Kemudian pada baris 26 menggunakan variable penampung row sebagai object dari pemberian nama baris kolom. Kemudian pada baris 27 menggunakan variable tabMode untuk penampung dari penggunaan defaultTableModel yang ada pada import library dengan disertakan pemanggilan object row sehingga pada baris 28, variable yang merupakan tabel pada desain akan di set dengan nama kolom yang sudah di buat objectnya.

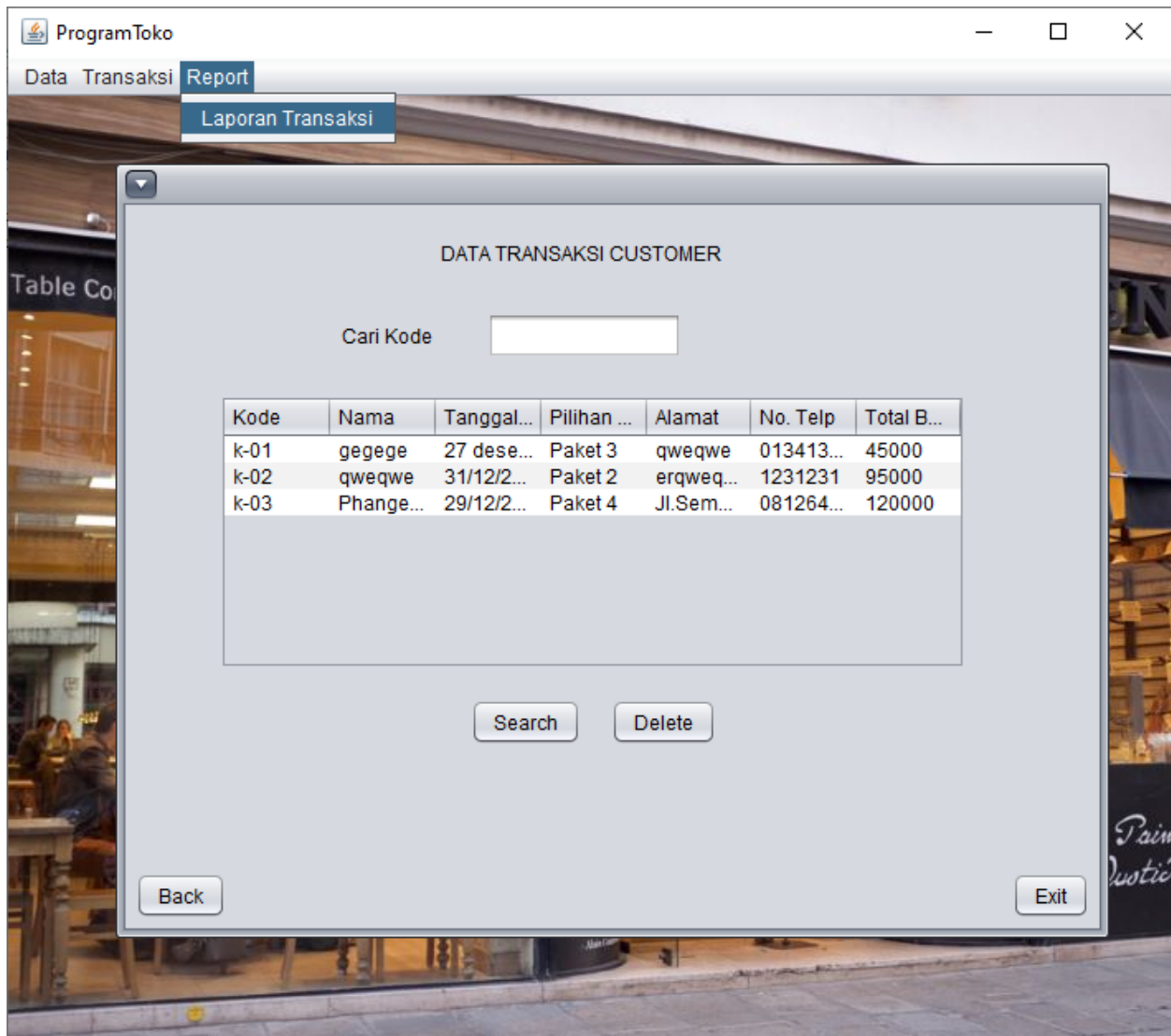
```
63 private void MenampilkanData() {
64     try {
65
66         sql = "SELECT * FROM tbcustomer";
67         rs = stat.executeQuery(sql);
68
69         while (rs.next()) {
70             tabMode.addRow(new Object[] {rs.getString(1), rs.getString(2), rs.getString(3), rs.getString(4),
71             rs.getString(5), rs.getString(6), rs.getString(7)});
72         }
73     }
74
75     catch (Exception e) {
76         JOptionPane.showMessageDialog(this, "ERROR");
77     }
78 }
79
80 @SuppressWarnings("unchecked")
81 // Generated Code
82
214
215 private void exitActionPerformed(java.awt.event.ActionEvent evt) {
216     System.exit(0);
217 }
218
219 private void backActionPerformed(java.awt.event.ActionEvent evt) {
220     new MenuUtama().setVisible(true);
221     dispose();
222 }
```

Terdapat method MenampilkanData() yang berguna ketika user ingin mengetahui berapa banyak transaksi yang terjadi, maka user dapat melihatnya dengan memilih menubar report maka akan langsung tampil data transaksi yang terjadi. Pada baris 66, merupakan sintaks untuk terhubung ke database pada variable sql yang kemudian melakukan SELECT * FROM tbcustomer yang berarti akan menampilkan data yang terdapat pada tbcustomer yang terdapat dalam database yang kemudian pada baris 67 akan dilakukan eksekusi untuk menampilkannya ke layar. Kemudian pada baris 70 dilakukan penambahan baris kolom dengan memanggil object yang sebelumnya sudah dibuat dengan urutan angkanya seperti rs.getString(1) sampai rs.getString(7) merupakan urutan nama baris kolom yang sudah di set sebelumnya. Ketika koneksi ke database gagal, maka akan muncul GUI "ERROR".

Pada DataTransaksi, terdapat button exit yang ketika user menekan button tersebut maka akan keluar dari program.

Terdapat juga button back yang berguna ketika user menekan button tersebut akan kembali ke halaman MenuUtama dengan dilakukannya pemanggilan method MenuUtama() pada program.

Tampilan Data Transaksi :



ProgramToko

Data Transaksi Report

Laporan Transaksi

DATA TRANSAKSI CUSTOMER

Cari Kode

Kode	Nama	Tanggal...	Pilihan ...	Alamat	No. Telp	Total B...
k-01	gegege	27 dese...	Paket 3	qweqwe	013413...	45000
k-02	qweqwe	31/12/2...	Paket 2	erqweq...	1231231	95000
k-03	Phange...	29/12/2...	Paket 4	Jl.Sem...	081264...	120000

Search Delete

Back Exit

JInternalFrameForm DataKaryawan

```
13 public class DataKaryawan extends javax.swing.JInternalFrame {
14
15 private void kosong() {
16     txtkode.setText(null);
17     txtnama.setText(null);
18     txttelepon.setText(null);
19     txtalamat.setText(null);
20 }
```

Terdapat method kosong() yang berguna untuk mengosongkan text field ketika program berjalan ataupun ketika user telah selesai melakukan penginputan data.

```
22 public void datatable() {
23     DefaultTableModel tbl = new DefaultTableModel();
24     tbl.addColumn("KodeKaryawan");
25     tbl.addColumn("Nama");
26     tbl.addColumn("No.Telepon");
27     tbl.addColumn("JenisKelamin");
28     tbl.addColumn("Alamat");
29     table.setModel(tbl);
30     try {
31         Statement statement = (Statement)Koneksi.GetConnection().createStatement();
32         ResultSet res = statement.executeQuery("SELECT * from tbkaryawan");
33         while(res.next()){
34             tbl.addRow(new Object[] {
35                 res.getString("KodeKaryawan"),
36                 res.getString("Nama"),
37                 res.getString("handphone"),
38                 res.getString("JenisKelamin"),
39                 res.getString("Alamat"),
40             });
41             table.setModel(tbl);
42         }
43     } catch (Exception e) {
44         JOptionPane.showMessageDialog(rootPane, "Salah");
45     }
46 }
```

Merupakan kodingan untuk melakukan pengeditan pada isi tabel yang terdapat pada desain dengan baris 24 hingga 29 merupakan penambahan nama kolom. Selanjutnya pada baris 31 yang merupakan sintaks untuk terkoneksi ke database, maka melakukan eksekusi kueri yang menampilkan data dari tbkaryawan pada database dengan nama kolom yang sesuai dengan pemberian nama pada tabel sebelumnya. Apabila koneksi ke database gagal, maka akan muncul GUI “Salah”.

```

289 private void bsaveActionPerformed(java.awt.event.ActionEvent evt) {
290     String kode = txtkode.getText();
291     String nama = txtnama.getText();
292     String alamat = txtalamat.getText();
293     String telepon = txttelepon.getText();
294     String jeniskelamin = null;
295     if(jRadioButton1.isSelected()) {
296         jeniskelamin = "Laki-Laki";
297     }
298     else if (jRadioButton2.isSelected()) {
299         jeniskelamin = "Perempuan";
300     }
301
302     try {
303         Statement statement = (Statement) Koneksi.GetConnection().createStatement();
304         statement.executeUpdate("Insert into tbkaryawan VALUES ('"+kode + "','" + nama+ "','" + telepon+ "','"
305             + jeniskelamin + "','" + alamat+ "')");
306         statement.close();
307         JOptionPane.showMessageDialog(null, "Data berhasil disimpan !");
308     } catch (Exception e) {
309         JOptionPane.showMessageDialog(null, "Data tidak berhasil disimpan! ");
310     }
311     datatable();
312     kosong();
313 }

```

Pada DataKaryawan terdapat proses CRUD (Create,Read,Update, dan Delete) dalam tabel. Button yang pertama ialah Save yang berkaitan dengan Create. Baris 290 hingga 294 merupakan pendeklarasian variable sebagai penampung dari isi text field yang diinputkan oleh user. Terdapat proses percabangan dari baris 295 hingga 300 yang merupakan radio button untuk pilihan jenis kelamin, ketika user memilih radiobutton1 maka variable penampung jeniskelamin akan otomatis terisi “Laki-laki”, dan jika user memilih radiobutton2 maka variable jeniskelamin akan otomatis terisi “Perempuan”. Pada baris 303 merupakan sintaks untuk terhubung ke database yang jika berhasil, maka akan melakukan update pada tbkaryawan yang terdapat pada database dengan sintaks kueri Insert yang berisi data-data variable penampung yang sudah diisi oleh user. Jika data berhasil disimpan ke database, maka akan muncul GUI “Data berhasil disimpan”, namun jika sebaliknya, maka akan muncul GUI “Data tidak berhasil disimpan”. Kemudian dilakukan pemanggilan method datatable() yang berfungsi untuk menampilkan data-data yang sudah diinputkan sebelumnya, dan kemudian pada baris 312 memanggil method kosong() untuk mengembalikan isi dari text field kosong seperti semula.

```

270 private void bupdateActionPerformed(java.awt.event.ActionEvent evt) {
271     int ok = JOptionPane.showConfirmDialog(null, "Apakah Yakin Untuk Update Record ini???", "Confirmation", JOptionPane.YES_NO_OPTION);
272     /JOptionPane.showMessageDialog(null, "Silahkan masukkan data yang akan di update beserta pengisian kembali seluruh datanya.");
273     if(ok==0) {
274         try{
275             String sql = "UPDATE tbkaryawan SET KodeKaryawan = '" + txtkode.getText() + "', handphone = '"
276                 + txttelepon.getText() + "', Nama = '" + txtnama.getText() + "', Alamat = '" + txtalamat.getText()
277                 + "' WHERE KodeKaryawan = '" + txtkode.getText() + "'";
278             java.sql.Connection conn=(Connection)Koneksi.GetConnection();
279             java.sql.PreparedStatement pst=conn.prepareStatement(sql);
280             pst.execute();
281             JOptionPane.showMessageDialog(null, "Update Data Sukses");
282         }
283         catch (Exception e) {
284             JOptionPane.showMessageDialog(null, "Update Data Gagal");
285         }
286     }
287     datatable();
288     kosong();
289 }

```

Button selanjutnya pada DataKaryawan adalah update, ketika user telah melakukan search terhadap data yang ingin di update, maka user menginputkan ulang semua datanya. Pada baris 271 akan muncul GUI yes / no option, ketika user menekan tombol “yes” maka akan masuk ke percabangan di baris 273 yang tahap pertama ialah melakukan pembaruan pada data yang ada di dalam database dengan sintaks kueri di baris 275 yang selanjutnya dikoneksikan ke database. Jika data sudah di perbaharui dan telah terkoneksi ke database, maka akan muncul GUI “Update data sukses”, namun jika gagal maka akan muncul GUI “update data gagal”. Setelahnya pada baris 287 dilakukan pemanggilan datatable() untuk menampilkan data yang terbaru dan isi dari textfield yang sebelumnya sudah dipakai untuk melakukan update pada data akan kembali kosong pada baris 288.

```

315 private void bdeleteActionPerformed(java.awt.event.ActionEvent evt) {
316     int ok = JOptionPane.showConfirmDialog(null, "Apakah Yakin Untuk Menghapus Record ini???",
317         "Confirmation", JOptionPane.YES_NO_OPTION);
318     String kode = txtkode.getText();
319     if(ok==0) {
320         try {
321             Statement statement = (Statement) Koneksi.GetConnection().createStatement();
322             statement.executeUpdate("DELETE from tbkaryawan WHERE KodeKaryawan =(' + kode + ')");
323             JOptionPane.showMessageDialog(null, "Data berhasil terhapus !");
324             txtkode.setText("");
325             txtnama.setText("");
326             txttelepon.setText("");
327             String jeniskelamin = null;
328             txtalamat.setText("");
329             txtkode.requestFocus();
330         }
331         catch (Exception e) {
332             JOptionPane.showMessageDialog(null, "Data gagal di hapus!");
333         }
334         datatable();
335         kosong();
336     }
337 }

```

Selanjutnya terdapat button delete yang ketika user menekan button tersebut akan menghapus salah satu data yang sudah dilakukan pencarian. Pada baris 316, merupakan sintaks untuk menampilkan GUI dengan YES/NO opsi “Apakah anda yakin untuk menghapus record ini?” Jika iya, maka masuk ke percabangan baris 319 yang dilakukan koneksi ke database terlebih dahulu kemudian melakukan penghapusan data yang dipilih dari tbkaryawan yang ada pada database dengan kondisi jika KodeKaryawan yang akan dihapus sesuai dengan pencarian kode karyawan sebelumnya. Jika data berhasil terhapus dari database, maka akan muncul GUI “Data berhasil terhapus” yang kemudian text field akan menjadi kosong kembali dengan sintaks pada baris 324 hingga 329. Namun jika data gagal dihapus pada database, maka akan muncul GUI “data gagal dihapus”. Selanjutnya pada baris 334 akan tampil data terbaru dengan datatable().

```

339 private void bexitActionPerformed(java.awt.event.ActionEvent evt) {
340     dispose();
341 }
342
343 private void bsearchActionPerformed(java.awt.event.ActionEvent evt) {
344     JOptionPane.showMessageDialog(null, "Berikut adalah hasil pencarian kode karyawan yang ingin cari.");
345     try {
346         Statement statement = (Statement) Koneksi.GetConnection().createStatement();
347         ResultSet res = statement.executeQuery("SELECT * FROM tbkaryawan WHERE " + "KodeKaryawan='" + txtkode.getText() + "'");
348         DefaultTableModel tbl = new DefaultTableModel();
349         tbl.addColumn("Kode Karyawan");
350         tbl.addColumn("Nama");
351         tbl.addColumn("No. Telepon");
352         tbl.addColumn("Jenis Kelamin");
353         tbl.addColumn("Alamat");
354
355         table.setModel(tbl);
356         while (res.next()) {
357             tbl.addRow(new Object[] {
358                 res.getString("KodeKaryawan"),
359                 res.getString("Nama"),
360                 res.getString("handphone"),
361                 res.getString("JenisKelamin"),
362                 res.getString("Alamat"),
363             });
364             table.setModel(tbl);
365         }
366     } catch (Exception e) {
367         JOptionPane.showMessageDialog(rootPane, "Salah2");
368     }
369 }

```

Kemudian proses CRUD yang terakhir adalah melakukan pencarian. Ketika user menekan button search, maka akan muncul GUI yang akan menunjukkan hasil pencarian. Pada baris 346 merupakan sintaks untuk penghubung ke database agar bisa mengakses data yang terdapat dalam MySQL. Kemudian pada baris 347 merupakan sintaks yang digunakan untuk kueri MySQL yang berarti memilih tbkaryawan dalam database kemudian akan ditampilkan ke layar dengan tabel yang sudah diberi nama kolomnya dengan baris 349 hingga 353. Kemudian pada baris 355, variable tabel yang terdapat pada desain akan di set dengan default tabel model yang diambil dari import library yang kemudian pada baris selanjutnya akan dilakukan pengulangan dengan menggunakan while untuk menampilkan dari data paling atas hingga data paling bawah. Jika koneksi ke database terdapat masalah, maka akan muncul GUI “Salah”.

Tampilan Data Karyawan :

ProgramToko

Data Transaksi Report

DATA KARYAWAN

Kode Karyawan

Nama

No. Telepon

Jenis Kelamin ☐ Laki-Laki ☐ Perempuan

Alamat

Update Save Search Delete Exit

KodeKarya...	Nama	No.Telepon	JenisKela...	Alamat
123	jen1	3123124	Laki-Laki	rarara
1234	jen3	11111	Perempuan	4314432

Setiap data karyawan yang ingin dilakukan delete dan update, maka harus dilakukan search kode karyawannya terlebih dahulu. Untuk update, memerlukan penginputan keseluruhan data kembali.

Tampilan Database untuk tabel tbcustomer :

The screenshot shows the phpMyAdmin interface for the 'dbtoko' database. The 'Table structure' tab is selected for the 'tbcustomer' table. The table has 7 columns: Kode, Nama, TanggalP, PilihanPaket, Alamat, No.Telepon, and TotalPembayaran. All columns are of type 'varchar' with varying lengths and use the 'utf8mb4_general_ci' collation. The 'TotalPembayaran' column is the last column in the table.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Kode	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
2	Nama	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
3	TanggalP	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
4	PilihanPaket	varchar(10)	utf8mb4_general_ci		No	None			Change Drop More
5	Alamat	varchar(55)	utf8mb4_general_ci		No	None			Change Drop More
6	No.Telepon	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
7	TotalPembayaran	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More

Below the table structure, there are options to 'Check all', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There are also buttons for 'Add to central columns' and 'Remove from central columns'. At the bottom, there is a section for 'Indexes' which shows 'No index defined!'.

Tampilan database untuk tabel tbkaryawan :

The screenshot shows the phpMyAdmin interface for the 'dbtoko' database. The 'Table structure' tab is selected for the 'tbkaryawan' table. The table has 5 columns: KodeKaryawan, Nama, handphone, JenisKelamin, and Alamat. All columns are of type 'varchar' with varying lengths and use the 'utf8mb4_general_ci' collation. The 'Alamat' column is the last column in the table.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	KodeKaryawan	varchar(8)	utf8mb4_general_ci		No	None			Change Drop More
2	Nama	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
3	handphone	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
4	JenisKelamin	varchar(20)	utf8mb4_general_ci		No	None			Change Drop More
5	Alamat	varchar(50)	utf8mb4_general_ci		No	None			Change Drop More

Below the table structure, there are options to 'Check all', 'With selected', 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There are also buttons for 'Add to central columns' and 'Remove from central columns'. At the bottom, there is a section for 'Indexes' which shows 'No index defined!'. Below this, there is a form to 'Create an index on' with a dropdown for '1' and a 'Go' button.

Tampilan untuk database tblogin :

The screenshot shows the phpMyAdmin interface for the 'dbtoko' database, specifically the 'tblogin' table. The left sidebar displays a tree view of databases, with 'dbtoko' selected. The main panel shows the 'Table structure' tab for 'tblogin'. The table has two columns: 'username' (varchar(15), utf8mb4_general_ci) and 'password' (varchar(30), utf8mb4_general_ci). Below the table structure, there are sections for 'Indexes' and 'Partitions', both indicating 'No index defined!' and 'No partitioning defined!' respectively. The 'Console' tab at the bottom is empty.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	username	varchar(15)	utf8mb4_general_ci		No	None			Change Drop More
2	password	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More

Tampilan untuk database tbsignup :

The screenshot shows the phpMyAdmin interface for the 'dbtoko' database, specifically the 'tbsignup' table. The left sidebar displays a tree view of databases, with 'dbtoko' selected. The main panel shows the 'Table structure' tab for 'tbsignup'. The table has three columns: 'username' (varchar(15), utf8mb4_general_ci), 'password' (varchar(30), utf8mb4_general_ci), and 'repassword' (varchar(30), utf8mb4_general_ci). Below the table structure, there are sections for 'Indexes' and 'Partitions', both indicating 'No index defined!' and 'No partitioning defined!' respectively. The 'Console' tab at the bottom is empty.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	username	varchar(15)	utf8mb4_general_ci		No	None			Change Drop More
2	password	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More
3	repassword	varchar(30)	utf8mb4_general_ci		No	None			Change Drop More