

MANUAL BOOK

PROJECT LAB REKAYASA PERANGKAT LUNAK 2



DISUSUN OLEH :

Nama : Mohammad Hikmal Ceisar

NPM : 50421844

Kelas : 4IA26

2024

DESKRIPSI APLIKASI

Sistem Pengelolaan Cargo BYOZ

Aplikasi ini dirancang untuk membantu perusahaan pengiriman barang dalam mengelola data pengiriman cargo secara efektif dan efisien. Sistem ini memungkinkan pengguna untuk:

- Mencatat dan menyimpan informasi terkait pengiriman barang.
- Mengelola data pengiriman menggunakan antarmuka berbasis GUI.
- Berintegrasi dengan database menggunakan teknologi Hibernate untuk mempermudah manipulasi data.

Fitur Utama:

1. Pencatatan Pengiriman

Pengguna dapat mencatat informasi pengiriman, termasuk data barang, tujuan, tanggal pengiriman, status pengiriman, dan sebagainya.

2. Tampilan Data dalam Tabel

Data pengiriman ditampilkan dalam format tabel menggunakan DefaultTableModel, yang mempermudah pengguna untuk membaca dan mengelola informasi.

3. Integrasi Database

Menggunakan Hibernate untuk menghubungkan aplikasi dengan database, sehingga pengelolaan data menjadi lebih efisien. Class CargoEntity menunjukkan representasi objek yang digunakan untuk operasi database.

4. Manajemen Transaksi

Aplikasi dilengkapi dengan fitur transaksi database untuk memastikan data yang disimpan konsisten dan terintegrasi dengan baik.

5. Pengolahan Data SQL

Aplikasi memungkinkan pengguna untuk berinteraksi langsung dengan database menggunakan perintah SQL bawaan yang dikelola melalui objek Connection, Statement, dan ResultSet.

6. Antarmuka Pengguna

Dibangun dengan Java Swing, aplikasi ini menyediakan GUI yang user-friendly untuk pengguna awam.

Komponen Teknis

1. Teknologi yang Digunakan:

- **Java Swing** untuk pengembangan antarmuka pengguna.
- **Hibernate** untuk ORM (Object Relational Mapping) dengan database.
- **JDBC** untuk konektivitas SQL langsung.
- **JTable** untuk menampilkan data dalam format tabel.

2. Kebutuhan Sistem:

- **Perangkat Keras:**
 - Laptop/PC dengan RAM minimal 4 GB.
- **Perangkat Lunak:**
 - JDK versi terbaru.
 - Hibernate Framework.
 - Database Management System (MySQL atau sejenisnya).

3. Struktur Kelas:

- `Cargo` sebagai class utama yang berisi logika aplikasi.
- `CargoEntity` sebagai representasi entitas database.
- Integrasi dengan Hibernate untuk pengelolaan database.

Manfaat Aplikasi

- Mempermudah perusahaan dalam mengelola data pengiriman barang.
- Mengurangi kesalahan pencatatan data pengiriman.
- Meningkatkan efisiensi dalam pencarian dan pengelolaan informasi.

TUJUAN APLIKASI

- **Mengotomatiskan Pengelolaan Data Pengiriman Barang**

Mengurangi pekerjaan manual dalam pencatatan dan pengelolaan informasi pengiriman barang dengan menyediakan sistem berbasis komputer yang terintegrasi.

- **Menyediakan Antarmuka Pengguna yang Mudah Digunakan**

Menggunakan Java Swing untuk menciptakan tampilan yang user-friendly, sehingga pengguna awam dapat dengan mudah mengakses, menginput, dan membaca data terkait pengiriman barang.

- **Mempermudah Akses dan Pengelolaan Data**

Data pengiriman barang disimpan dalam database yang dapat diakses secara efisien melalui antarmuka aplikasi. Informasi ditampilkan dalam tabel untuk mempermudah pencarian dan pengelolaan data.

- **Mengintegrasikan Sistem dengan Database Secara Efisien**

Menggunakan Hibernate sebagai framework ORM (Object-Relational Mapping) untuk mengelola data dalam database, sehingga manipulasi data menjadi lebih sederhana dan cepat.

- **Meningkatkan Akurasi dan Konsistensi Data**

Dengan dukungan fitur transaksi dari Hibernate dan JDBC, aplikasi memastikan bahwa data pengiriman barang tersimpan dengan akurat dan konsisten di dalam database.

- **Menyediakan Fitur Monitoring Pengiriman**

Menyimpan dan menampilkan informasi pengiriman, seperti ID pengiriman, nama barang, tujuan, tanggal pengiriman, dan status pengiriman, untuk memudahkan monitoring.

- **Mendukung Pengambilan Keputusan**

Informasi yang tersimpan di dalam sistem dapat digunakan sebagai referensi untuk analisis, evaluasi, dan pengambilan keputusan operasional terkait pengiriman barang.

- **Meminimalkan Kesalahan dan Kehilangan Data**

Dengan sistem berbasis komputer, risiko kesalahan manusia (human error) dan kehilangan data akibat pencatatan manual dapat diminimalkan.

BATASAN MASALAH

Batasan Masalah Sistem Pengelolaan Cargo BYOZ

1. Cakupan Data Pengiriman

Aplikasi hanya menangani data yang terkait dengan pengiriman barang, seperti:

- ID pengiriman.
- Nama barang.
- Tujuan pengiriman.
- Tanggal pengiriman.
- Status pengiriman (Proses, Selesai, atau Batal).

2. Fokus pada Pengelolaan Database Lokal

Aplikasi dirancang untuk menggunakan database lokal (misalnya, MySQL) dengan koneksi menggunakan Hibernate atau JDBC. Tidak ada fitur untuk integrasi dengan database berbasis cloud.

3. Lingkup Pengguna Terbatas

Aplikasi dirancang untuk digunakan oleh internal perusahaan pengiriman saja, bukan untuk pelanggan atau pihak eksternal.

4. Tidak Mendukung Multi-User

Aplikasi ini tidak memiliki fitur autentikasi atau manajemen pengguna, sehingga hanya dapat digunakan oleh satu pengguna dalam satu sesi.

5. Fitur Transaksi Sederhana

Proses transaksi database terbatas pada pencatatan, pengubahan, dan penghapusan data pengiriman tanpa mendukung operasi bisnis yang lebih kompleks, seperti pelacakan pengiriman secara real-time.

6. Tidak Ada Integrasi dengan Sistem Lain

Aplikasi ini berdiri sendiri dan tidak terintegrasi dengan sistem lain, seperti aplikasi logistik, sistem pembayaran, atau API pihak ketiga.

7. Tampilan Antarmuka Sederhana

Antarmuka aplikasi berbasis Java Swing dirancang hanya untuk fungsi pengelolaan data, tanpa fitur desain modern atau responsif untuk perangkat lain seperti ponsel atau tablet.

8. Tidak Mendukung Penyimpanan Arsip Otomatis

Aplikasi tidak menyediakan fitur otomatis untuk arsip data pengiriman lama, sehingga data lama harus dikelola secara manual oleh pengguna.

9. Keterbatasan Format Laporan

Sistem belum mendukung fitur cetak laporan atau ekspor data ke format lain seperti PDF atau Excel.

10. Pengolahan Data Terbatas pada SQL Sederhana

Pengolahan data pada aplikasi menggunakan query SQL sederhana tanpa mendukung analisis data atau visualisasi informasi.

KEBUTUHAN SPESIFIK

No	Fitur	Kebutuhan Fungsional	Kebutuhan Antarmuka	Kebutuhan untuk Kerja
1	Pendaftaran Cargo	Sistem harus dapat merekam data barang, termasuk berat, dimensi, dan deskripsi barang.	Form input dengan field untuk nama barang, berat, dimensi, dan deskripsi.	Koneksi ke database untuk menyimpan data barang.
2	Tracking Cargo	Sistem memungkinkan pengguna melacak posisi barang.	Halaman untuk memasukkan kode tracking dan menampilkan lokasi terkini.	Integrasi dengan API penyedia layanan pelacakan lokasi.
3	Pengelolaan Stok Gudang	Sistem dapat memperbarui status barang di gudang (masuk, keluar, pending).	Dashboard dengan informasi real-time terkait stok barang di gudang.	Koneksi ke sistem manajemen gudang (WMS) untuk sinkronisasi data.
4	Notifikasi Pelanggan	Sistem harus mengirim notifikasi status barang ke pelanggan.	Panel untuk mengatur preferensi notifikasi (email/SMS).	Koneksi ke penyedia layanan SMS dan email gateway.
5	Laporan Keuangan	Sistem menghasilkan laporan pendapatan dan biaya operasional terkait pengelolaan cargo.	Antarmuka grafik atau tabel untuk melihat laporan keuangan.	Akses ke data transaksi dan modul analisis laporan.

KEBUTUHAN PERANGKAT

KEBUTUHAN PERANGKAT KERAS (HARDWARE)

No	Komponen	Spesifikasi Minimal
1	Prosesor	Intel Core i3 generasi ke-8 atau setara
2	RAM	4 GB (disarankan 8 GB untuk performa lebih baik)
3	Penyimpanan	250 GB HDD (disarankan SSD 128 GB atau lebih)
4	Resolusi Layar	1366 x 768 piksel
5	Perangkat Tambahan	Keyboard, Mouse, dan Printer (untuk cetak laporan)
6	Jaringan	Ethernet atau Wi-Fi untuk koneksi internet

KEBUTUHAN PERANGKAT LUNAK (SOFTWARE)

No	Komponen	Spesifikasi/Versi
1	Sistem Operasi	Windows 10/11, Linux (Ubuntu 20.04+), atau macOS 10.15+
2	Java Development Kit (JDK)	JDK 17 atau lebih tinggi
3	Integrated Development Environment (IDE)	IntelliJ IDEA, Eclipse, atau NetBeans
4	Database	MySQL 8.0, PostgreSQL, atau SQLite
5	Library/Framework Tambahan	JavaFX (untuk GUI), JDBC (untuk database)
6	API Integrasi	JavaMail API, JasperReports
7	Web Browser	Google Chrome atau Mozilla Firefox (untuk testing jika berbasis web)
8	Virtualisasi (Opsional)	Docker (jika ingin mengemas sistem dalam container)

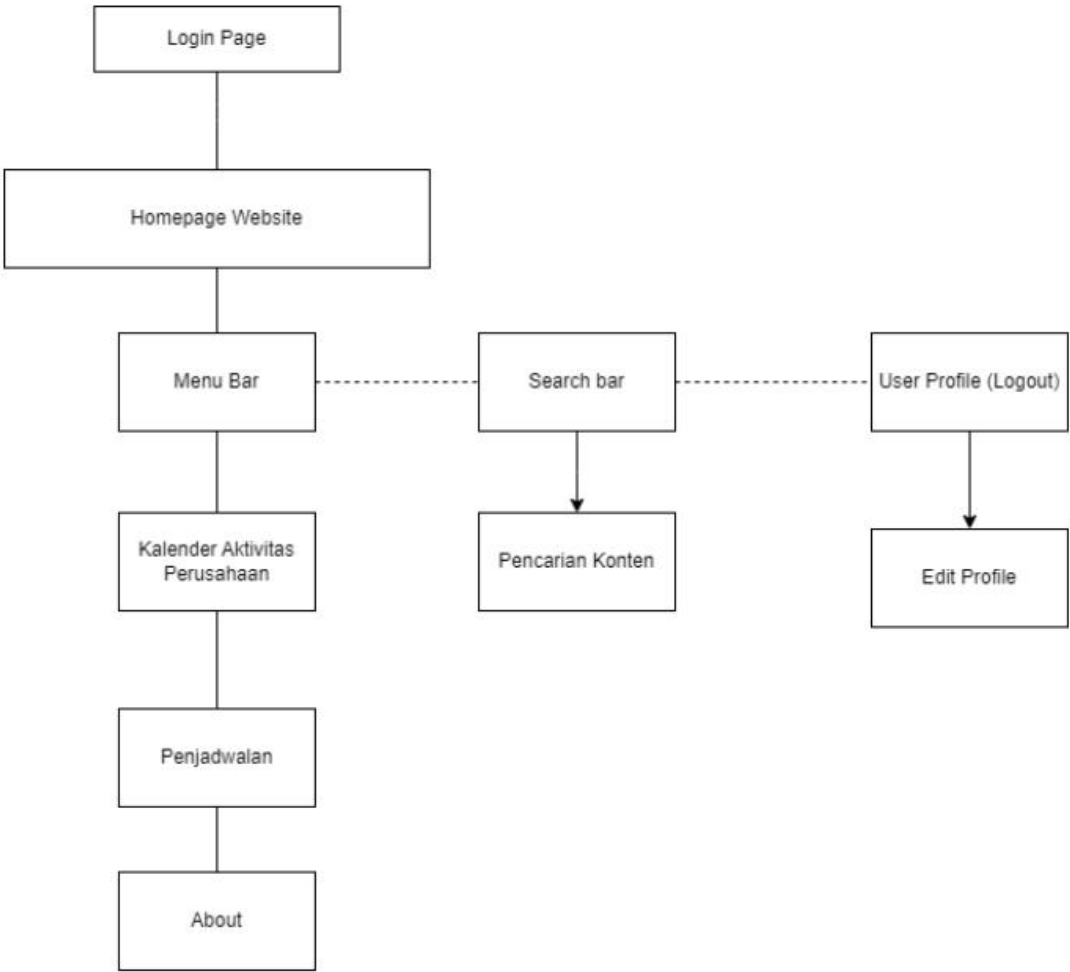
KEBUTUHAN JARINGAN

No	Komponen	Spesifikasi
1	Koneksi Internet	Minimal 5 Mbps
2	Server Lokal	Minimal prosesor Intel Xeon, RAM 16 GB
3	Protokol Keamanan	SSL/TLS untuk keamanan data

KEBUTUHAN PERANGKAT PENDUKUNG (OPSIONAL)

No	Perangkat	Deskripsi
1	Barcode Scanner	Untuk mempermudah pendaftaran barang dengan kode unik
2	Label Printer	Untuk mencetak label kargo
3	Mobile Device	Aplikasi pelacakan yang mendukung perangkat Android/iOS

STRUKTUR NAVIGASI

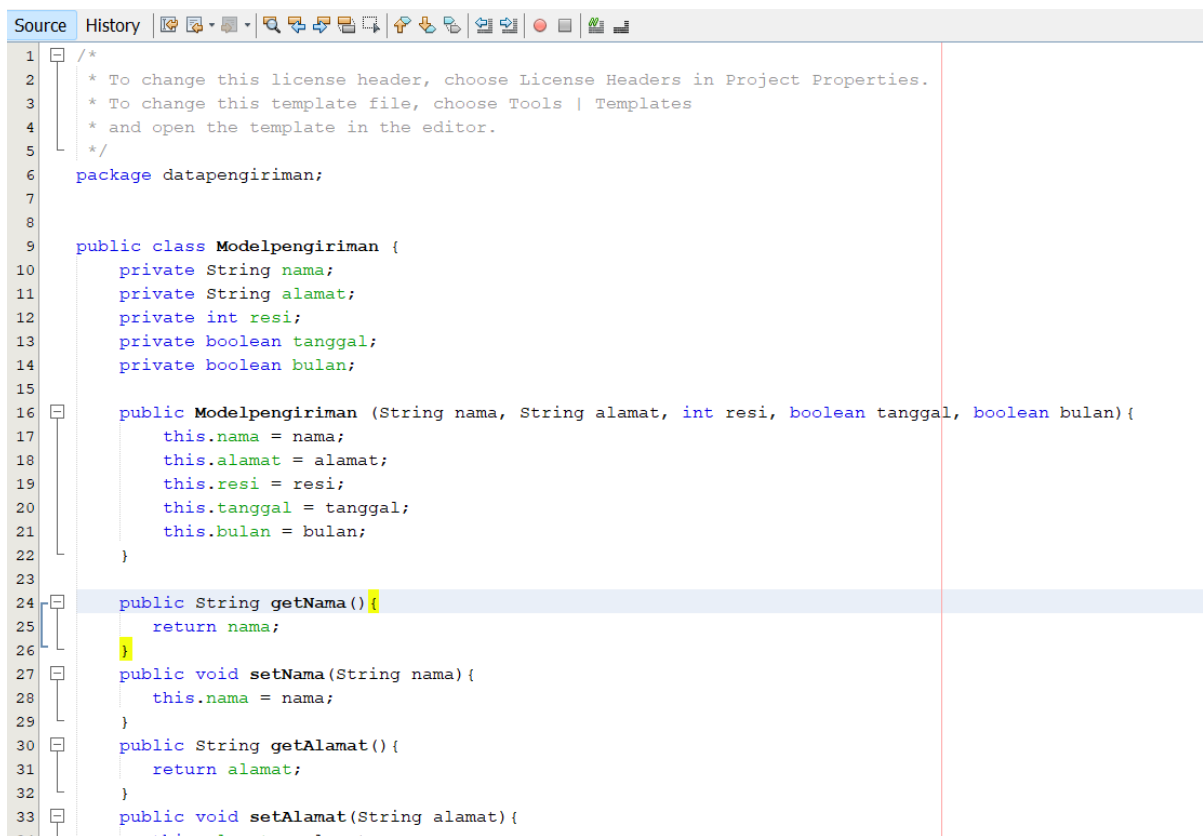
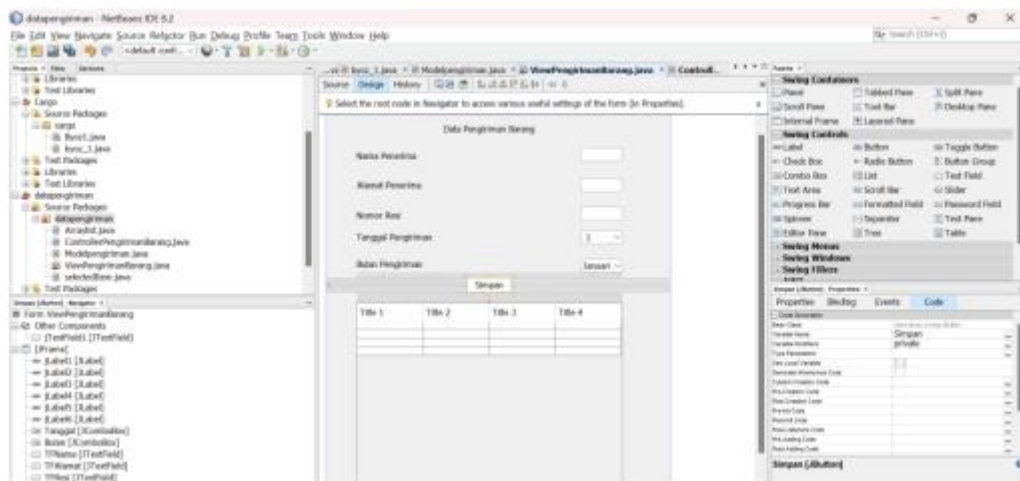
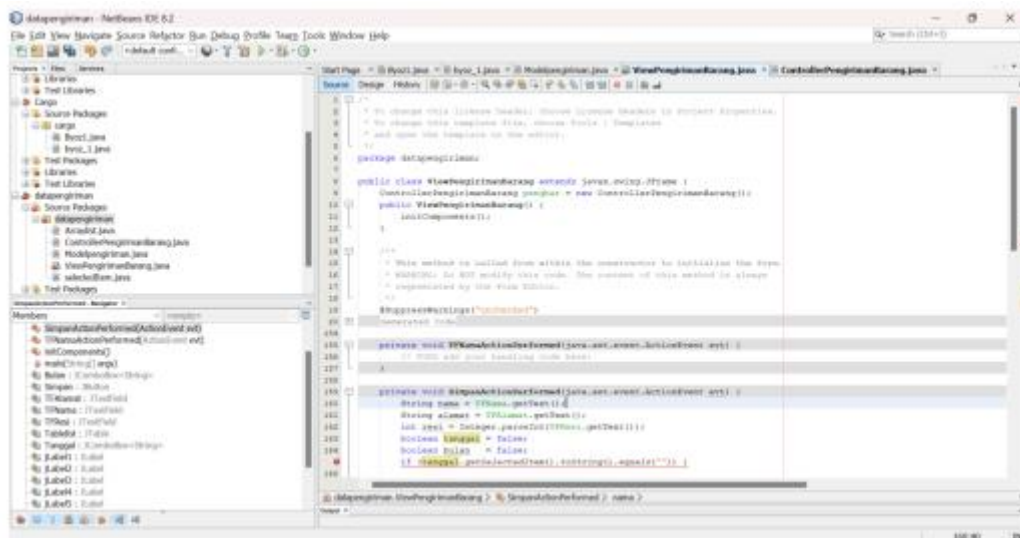


LISTING

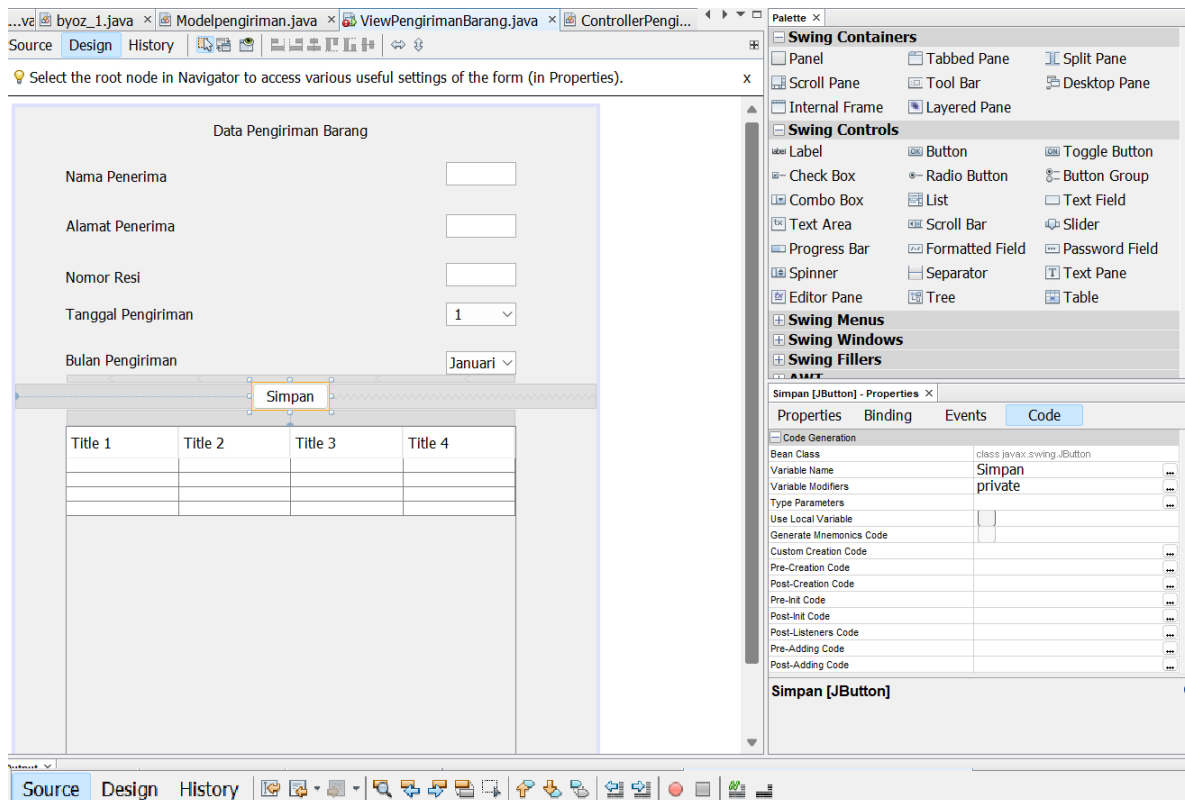
Pertemuan 2

```
...4 lines
5 package byoz_1;
6
7 /**
8  *
9  * @author MSI
10 */
11 import java.util.Scanner;
12 public class byoz_1 {
13     int melayani;
14     int pengiriman;
15     double BarangAntarPulau;
16     public byoz_1(int pengiriman) {
17         this.pengiriman=pengiriman;
18         System.out.println("Paket perhari: "+pengiriman);
19     }
20     public byoz_1 (int BarangAntarPulau, int melayani){
21         this.BarangAntarPulau=BarangAntarPulau;
22         this.melayani=melayani;
23         int byr=BarangAntarPulau*melayani;
24         System.out.println("pengiriman perhari: "+byr);
25     }
26     public static void pilihan(){
27         Scanner input = new Scanner(System.in);
28         System.out.println("1.Laut 1 bulan 2x ");
29         System.out.println("2.Udara 1 bulan 4x ");
30         System.out.println("Silahkan pilih Pengiriman: ");
31
32         int pilih = input.nextInt();
33         switch (pilih){
34             case 1 -> byoz_1.laut();
35             case 2 -> byoz_1.udara();
36         }
37     }
38     public static void lautbyoz(){
39         Scanner input = new Scanner(System.in);
40         System.out.println("-----");
41         System.out.println("Pilih Jenis Pengiriman");
42         System.out.println("-----");
43         System.out.println("1. Pengiriman Melalui Udara");
44         System.out.println("Masukkan Pilihan: ");
45         int pilih1 = input.nextInt();
46         switch(pilih1){
47             case 1 -> {
48                 byoz udaral =new byoz(200000);
49                 udaral.tampiludara();
50             }
51         }
52     }
53 }
54
55 }
56
57 }
```

Pertemuan 3



```
24 public String getNama() {
25     return nama;
26 }
27 public void setNama(String nama) {
28     this.nama = nama;
29 }
30 public String getAlamat() {
31     return alamat;
32 }
33 public void setAlamat(String alamat) {
34     this.alamat = alamat;
35 }
36 public int getResi() {
37     return resi;
38 }
39 public void setResi(int resi) {
40     this.resi = resi;
41 }
42 public boolean isTanggal() {
43     return tanggal;
44 }
45 public void setTanggal(boolean tanggal) {
46     this.tanggal = tanggal;
47 }
48 public boolean isBulan() {
49     return bulan;
50 }
51 public void setBulan(boolean bulan) {
52     this.bulan = bulan;
53 }
54 }
55
```



```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package datapengiriman;
7
8  public class ViewPengirimanBarang extends javax.swing.JFrame {
9      ControllerPengirimanBarang pengbar = new ControllerPengirimanBarang();
10     public ViewPengirimanBarang() {
11         initComponents();
12     }
13
14     /**
15      * This method is called from within the constructor to initialize the form.
16      * WARNING: Do NOT modify this code. The content of this method is always
17      * regenerated by the Form Editor.
18      */
19     @SuppressWarnings("unchecked")
20     Generated Code
154
155     private void TFNamaActionPerformed(java.awt.event.ActionEvent evt) {
156         // TODO add your handling code here:
157     }
158
159     private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {
160         String nama = TFNama.getText();
161         String alamat = TFAlamat.getText();
162         int resi = Integer.parseInt(TFResi.getText());
163         boolean tanggal = false;
164         boolean bulan = false;
165         if (tanggal.getSelectedItem().toString().equals("")) {

```

```

167     }
168 }
169
170 /**
171  * @param args the command line arguments
172  */
173 public static void main(String args[]) {
174     /* Set the Nimbus look and feel */
175     Look and feel setting code (optional)
176
177     /* Create and display the form */
178     java.awt.EventQueue.invokeLater(new Runnable() {
179         public void run() {
180             new ViewPengirimanBarang().setVisible(true);
181         }
182     });
183 }
184
185 // Variables declaration - do not modify
186 private javax.swing.JComboBox<String> Bulan;
187 private javax.swing.JButton Simpan;
188 private javax.swing.JTextField TFAlamat;
189 private javax.swing.JTextField TFNama;
190 private javax.swing.JTextField TFResi;
191 private javax.swing.JTable Tablelist;
192 private javax.swing.JComboBox<String> Tanggal;
193 private javax.swing.JLabel jLabel1;
194 private javax.swing.JLabel jLabel2;
195 private javax.swing.JLabel jLabel3;
196 private javax.swing.JLabel jLabel4;
197 private javax.swing.JLabel jLabel5;
198 private javax.swing.JLabel jLabel6;
199 private javax.swing.JScrollPane jScrollPane1;
200
201 private javax.swing.JTextField jTextField1;
202 // End of variables declaration
203 }
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223

```


Pertemuan 4



The screenshot displays an IDE with a Java source file and its project structure. The source file contains the following code:

```
1
2 import java.awt.HeadlessException;
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import javax.swing.JOptionPane;
7
8 /*
9  * To change this license header, choose License Headers in Project Properties.
10  * To change this template file, choose Tools | Templates
11  * and open the template in the editor.
12  */
13
14 /**
15  *
16  * @author MSI
17  */
18 public class Cargo extends javax.swing.JFrame {
19     private Statment St;
20     private Connection Con;
21     private Resultset Rs;
22     private final String sql="";
23
24     /**
25      * Creates new form Cargo
26      */
27     public Cargo() {
28         initComponents();
29         KoneksiDatabase();
30     }
31
32     private void KoneksiDatabase() {
33         try {
34             //Menentukan Driver Database
35             Class.forName("com.mysql.jdbc.Driver");
36
37             private void KoneksiDatabase() {
38                 try {
39                     //Menentukan Driver Database
40                     Class.forName("com.mysql.jdbc.Driver");
41                     // untuk mengkoneksikan DB ke driver
42                     Con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_cargo", "root", "");
43                     // pesan koneksi berhasil
44                     JOptionPane.showMessageDialog(null, "Koneksi Berhasil");
45                 } catch (HeadlessException | ClassNotFoundException | SQLException e) {
46                     // pesan Koneksi gagal
47                     System.out.println("Koneksi Gagal" + e.getMessage());
48                 }
49             }
50
51     private void KoneksiDatabase() {
52         try {
53             //Menentukan Driver Database
54             Class.forName("com.mysql.jdbc.Driver");
55             // untuk mengkoneksikan DB ke driver
56             Con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_cargo", "root", "");
57             // pesan koneksi berhasil
58             JOptionPane.showMessageDialog(null, "Koneksi Berhasil");
59         } catch (HeadlessException | ClassNotFoundException | SQLException e) {
60             // pesan Koneksi gagal
61             System.out.println("Koneksi Gagal" + e.getMessage());
62         }
63     }
64 }
```

The project structure on the right shows the following components:

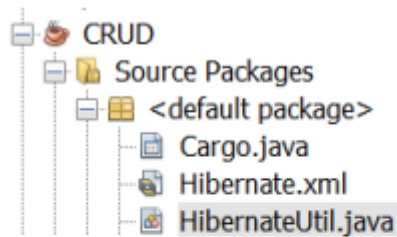
- CRUD
 - Source Packages
 - <default package>
 - Cargo.java
 - Resultset.java
 - Statment.java
 - Test Packages
 - Libraries
 - MySQL JDBC Driver - mysql-connector-java-5.1.23-bin.jar
 - JDK 1.8 (Default)
 - Test Libraries

Pertemuan 5

```
272 private void EditJBAActionPerformed(java.awt.event.ActionEvent evt) {  
273     try {  
274         String nama = txtNama.getText(); // Ambil nilai dari input teks nama  
275         String alamat = txtAlamat.getText(); // Ambil nilai dari input teks alamat  
276         String resi = txtResi.getText(); // Ambil nilai dari input teks resi  
277         String tanggal = txtTanggal.getText(); // Ambil nilai dari input teks tanggal  
278         String bulan = txtBulan.getText(); // Ambil nilai dari input teks bulan  
279  
280         // SQL query untuk update data  
281         String sql = "UPDATE tabel_pengiriman SET nama = ?, alamat = ?, resi = ?, tanggal = ?, bulan = ? WHERE id = ?";  
282         PreparedStatement pst = Con.prepareStatement(sql);  
283         pst.setString(1, nama);  
284         pst.setString(2, alamat);  
285         pst.setString(3, resi);  
286         pst.setString(4, tanggal);  
287         pst.setString(5, bulan);  
288  
289         int rowsUpdated = pst.executeUpdate();  
290         if (rowsUpdated > 0) {  
291             JOptionPane.showMessageDialog(null, "Data berhasil diupdate!");  
292         } else {  
293             JOptionPane.showMessageDialog(null, "Data tidak ditemukan!");  
294         }  
295     } catch (SQLException e) {  
296         JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());  
297     }  
298 }
```

```
301 private void DelJBAActionPerformed(java.awt.event.ActionEvent evt) {  
302     try {  
303         String nama = txtnama.getText(); // Ambil ID dari input teks  
304  
305         // SQL query untuk delete data  
306         String sql = "DELETE FROM tabel_pengiriman WHERE id = ?";  
307         PreparedStatement pst = Con.prepareStatement(sql);  
308         pst.setString(1, nama);  
309  
310         int rowsDeleted = pst.executeUpdate();  
311         if (rowsDeleted > 0) {  
312             JOptionPane.showMessageDialog(null, "Data berhasil dihapus!");  
313         } else {  
314             JOptionPane.showMessageDialog(null, "Data tidak ditemukan!");  
315         }  
316     } catch (SQLException e) {  
317         JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());  
318     }  
319 }  
320 }
```

Pertemuan 6



```
231
232 private void EditJActionPerformed(java.awt.event.ActionEvent evt) {
233     String nama = TFNama.getText();
234     String alamat = TFAlamat.getText();
235     String resi = TFResi.getText();
236     String tanggal = Tanggal.getSelectedText().toString();
237     String bulan = Bulan.getSelectedText().toString();
238     int id = Integer.parseInt(TFId.getText());
239
240     Session session = HibernateUtil.getSessionFactory().openSession();
241     Transaction tx = session.beginTransaction();
242
243     try {
244         CargoEntity cargo = (CargoEntity) session.get(CargoEntity.class, id);
245         if (cargo != null) {
246             cargo.setNama(nama);
247             cargo.setAlamat(alamat);
248             cargo.setResi(resi);
249             cargo.setTanggal(tanggal);
250             cargo.setBulan(bulan);
251
252             session.update(cargo); // Update data
253             tx.commit();
254             JOptionPane.showMessageDialog(this, "Data berhasil diupdate!");
255         } else {
256             JOptionPane.showMessageDialog(this, "Data tidak ditemukan!");
257         }
258     } catch (HeadlessException e) {
259         tx.rollback();
260         JOptionPane.showMessageDialog(this, "Gagal mengupdate data: " + e.getMessage());
261     } finally {
262         session.close();
263     }
264 }
```

LOGIKA

Pertemuan 2

```
13 public class Byoz1 {
```

Pada line ke- 13 adalah Deskripsi Kelas yang kesimpan dengan nama file Byoz1 :

Kelas ini digunakan untuk mengatur informasi pengiriman barang. Terdapat beberapa atribut dan konstruktor yang memungkinkan untuk memproses pengiriman paket, baik melalui kapal laut atau udara.

```
int melayani;  
int pengiriman;  
double BarangAntarPulau;
```

Selanjutnya, pada line ke- 14,15, dan 16 adalah Konstruktor yang berguna untuk, sebagai berikut :

- **Konstruktor 1** menerima satu parameter pengiriman dan menampilkan jumlah paket yang dikirimkan per hari.
- **Konstruktor 2** menerima dua parameter BarangAntarPulau dan melayani. Biaya pengiriman dihitung berdasarkan jumlah barang yang dikirim dan jumlah layanan, kemudian hasilnya dicetak sebagai output.

```
30 public static void pilihan() {  
31     Scanner input = new Scanner( source: System.in);  
32     System.out.println( x: "1. Laut 1 bulan 2x");  
33     System.out.println( x: "2. Udara 1 bulan 4x");  
34     System.out.println( x: "Silahkan pilih Pengiriman: ");  
35     int pilih = input.nextInt();  
36     switch (pilih) {  
37         case 1 -> lautbyoz();  
38         case 2 -> udarabyoz();  
39     }  
40 }  
41  
42 public static void lautbyoz() [...15 lines]  
57  
58 public static void udarabyoz() [...17 lines]  
75
```

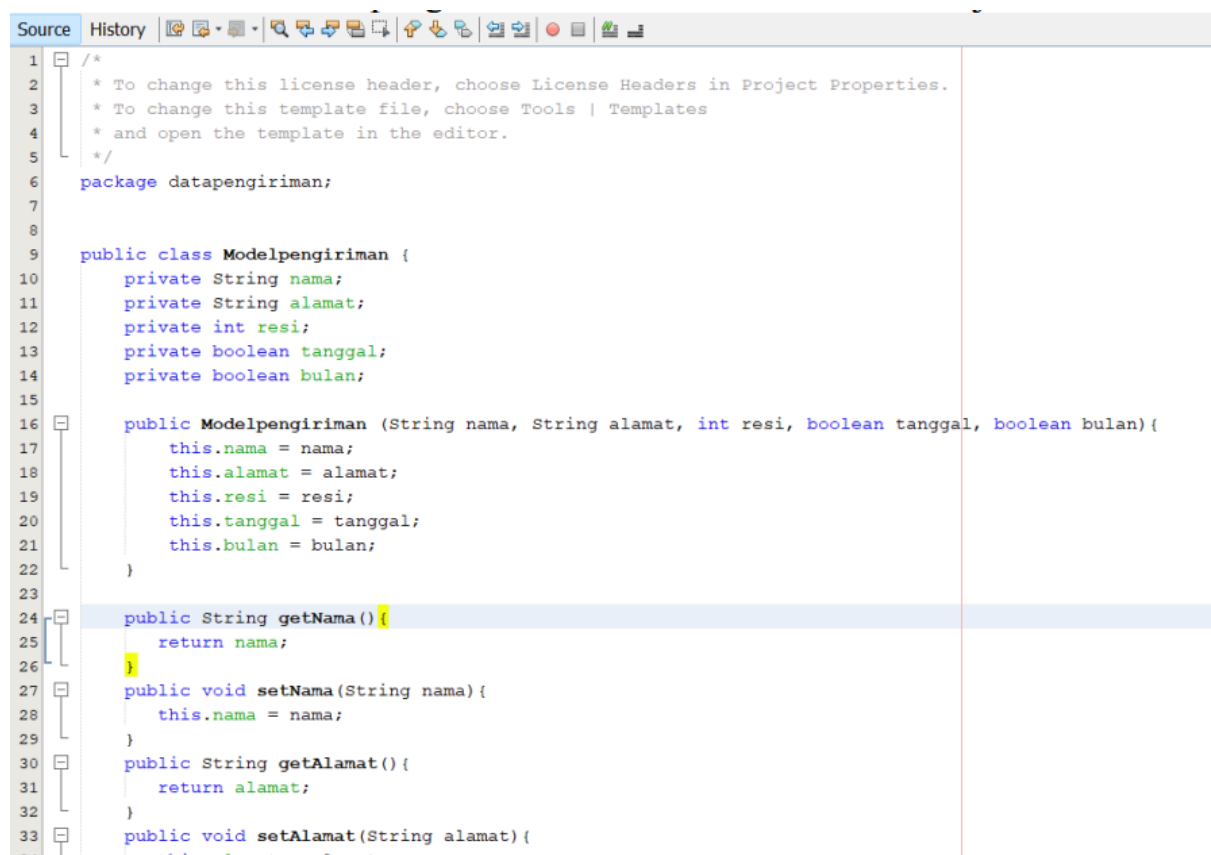
Selanjutnya, pada line ke- 30 sampai 71 adalah pilihan pengiriman yang berfungsi untuk, sebagai berikut :

- **Metode pilihan()** menampilkan dua opsi pengiriman, yaitu melalui laut (dengan frekuensi 2 kali per bulan) dan udara (4 kali per bulan).

- Berdasarkan pilihan pengguna, program akan memanggil metode `lautbyoz()` atau `udarabyoz()`, yang menampilkan informasi lebih detail tentang jenis pengiriman yang dipilih.

Pertemuan 3

Saya akan menjelaskan maksud dari `Modelpengiriman.java`, sebagaimana contoh code dari `Modelpengiriman` ada dibawah ini dan akan dijelaskan :



```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package datapengiriman;
7
8
9  public class Modelpengiriman {
10     private String nama;
11     private String alamat;
12     private int resi;
13     private boolean tanggal;
14     private boolean bulan;
15
16     public Modelpengiriman (String nama, String alamat, int resi, boolean tanggal, boolean bulan){
17         this.nama = nama;
18         this.alamat = alamat;
19         this.resi = resi;
20         this.tanggal = tanggal;
21         this.bulan = bulan;
22     }
23
24     public String getNama(){
25         return nama;
26     }
27     public void setNama(String nama){
28         this.nama = nama;
29     }
30     public String getAlamat(){
31         return alamat;
32     }
33     public void setAlamat(String alamat){
34         this.alamat = alamat;
35     }
36 }
```

Getter dan Setter

Kelas ini menyediakan metode getter dan setter untuk setiap atribut, yang memungkinkan kelas atau metode lain mengakses dan memodifikasi atribut secara individual.

1. **getNama():** Mengembalikan nilai dari nama.
2. **setNama(String nama):** Menetapkan nilai dari nama.
3. **getAlamat():** Mengembalikan nilai dari alamat.
4. **setAlamat(String alamat):** Menetapkan nilai dari alamat.
5. **getResi():** Mengembalikan nilai dari resi.
6. **setResi(int resi):** Menetapkan nilai dari resi.
7. **isTanggal():** Mengembalikan nilai dari tanggal (digunakan sebagai flag boolean).
8. **setTanggal(boolean tanggal):** Menetapkan nilai dari tanggal.

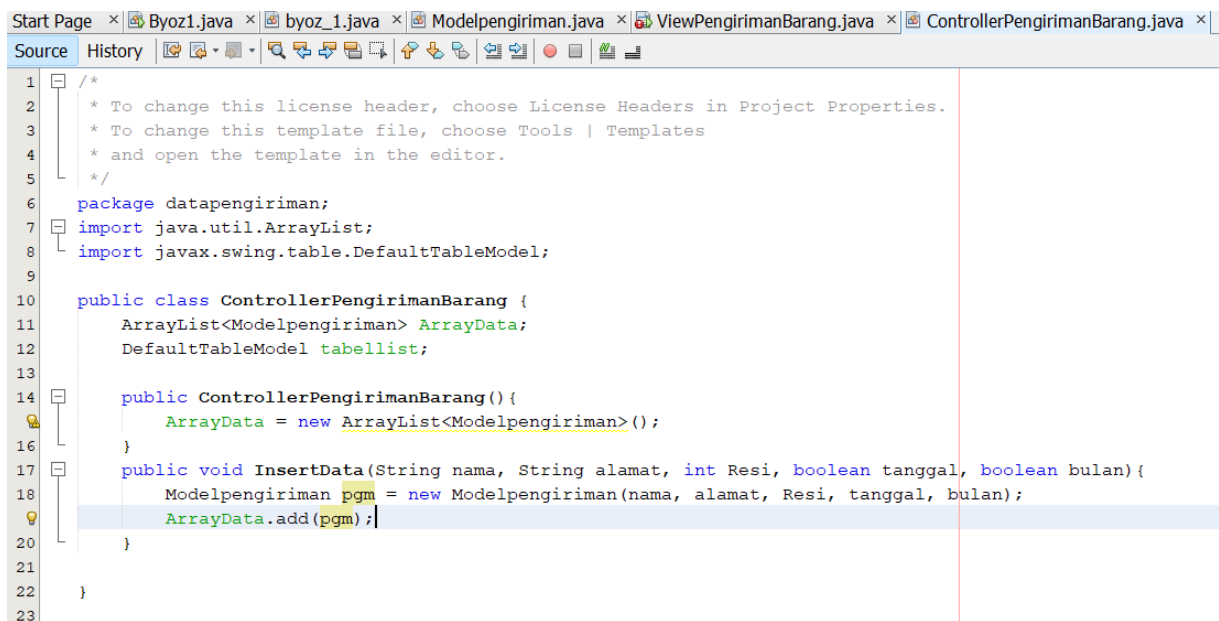
9. **isBulan()**: Mengembalikan nilai dari bulan (digunakan sebagai flag boolean).

10. **setBulan(boolean bulan)**: Menetapkan nilai dari bulan

- Kelas ini mengenkapsulasi data pengiriman dengan menyimpan nama penerima, alamat, nomor pelacakan, dan flag terkait tanggal.
- Dengan menggunakan metode getter dan setter, Modelpengiriman memastikan bahwa setiap atribut dapat diakses dan dimodifikasi secara terkendali. Ini adalah pendekatan standar untuk enkapsulasi dalam pemrograman berorientasi objek.
- Nilai boolean tanggal dan bulan dapat merepresentasikan apakah pengiriman dijadwalkan untuk hari (tanggal) atau bulan tertentu (bulan), meskipun penggunaan tepatnya akan bergantung pada bagaimana program lainnya menafsirkan flag ini.

Secara keseluruhan, **Modelpengiriman** berfungsi sebagai wadah data untuk informasi pengiriman, menyediakan cara untuk menyimpan, mengakses, dan memodifikasi detail utama yang terkait dengan pengiriman.

Selanjutnya, saya akan melogikakan **ControllerPengirimanBarang.java**, sebagai berikut:



```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package datapengiriman;
7  import java.util.ArrayList;
8  import javax.swing.table.DefaultTableModel;
9
10 public class ControllerPengirimanBarang {
11     ArrayList<Modelpengiriman> ArrayData;
12     DefaultTableModel tabellist;
13
14     public ControllerPengirimanBarang(){
15         ArrayData = new ArrayList<Modelpengiriman>();
16     }
17     public void InsertData(String nama, String alamat, int Resi, boolean tanggal, boolean bulan){
18         Modelpengiriman pgm = new Modelpengiriman(nama, alamat, Resi, tanggal, bulan);
19         ArrayData.add(pgm);
20     }
21 }
22
23
```

Logika Keseluruhan

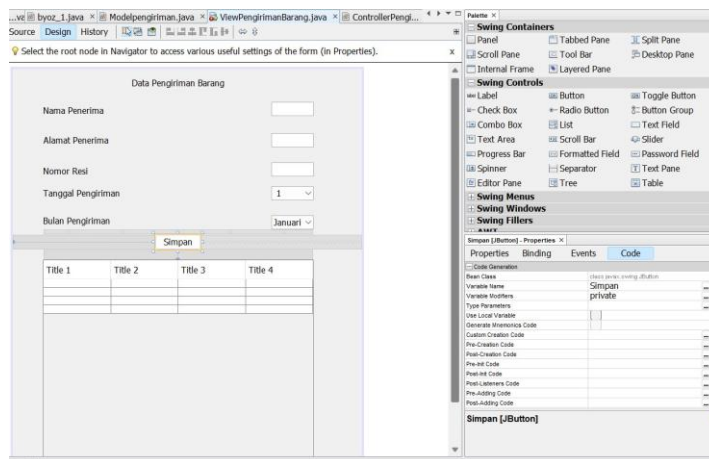
- Kelas **ControllerPengirimanBarang** bertindak sebagai pengelola data pengiriman, menyimpan setiap data yang dimasukkan dalam sebuah daftar (array list) berisi objek **Modelpengiriman**.
- Melalui metode **InsertData**, pengguna dapat menambahkan data pengiriman baru ke dalam **ArrayData**.

- Kelas ini mendukung pengelolaan data secara terstruktur dan memungkinkan data ditampilkan dalam bentuk tabel (dengan tabellist), meskipun fungsionalitas tersebut belum diterapkan dalam kode yang ada.

Rangkuman Logika

- Kelas ini menyimpan daftar data pengiriman dalam ArrayList bernama ArrayData.
- Metode InsertData membuat objek Modelpengiriman dengan data yang diberikan, lalumenambahkannya ke dalam ArrayData.
- Meskipun tabellist dideklarasikan untuk tampilan tabel, saat ini atribut tersebut belum digunakan dalam kode ini.

Dan Terakhir saya akan menjelaskan dari code data dan design ViewPengirimanBarang, Sebagai berikut :



```

1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6   package datapengiriman;
7
8   public class ViewPengirimanBarang extends javax.swing.JFrame {
9       ControllerPengirimanBarang pengbar = new ControllerPengirimanBarang();
10      public ViewPengirimanBarang() {
11          initComponents();
12      }
13
14      /**
15       * This method is called from within the constructor to initialize the form.
16       * WARNING: DO NOT modify this code. The content of this method is always
17       * regenerated by the Form Editor.
18       */
19      @SuppressWarnings("unchecked")
20      Generated Code
21
22      private void TFnamaActionPerformed(java.awt.event.ActionEvent evt) {
23          // TODO add your handling code here:
24      }
25
26      private void SimpanActionPerformed(java.awt.event.ActionEvent evt) {
27          String nama = TFnama.getText();
28          String alamat = TFalamat.getText();
29          int resi = Integer.parseInt(TFresi.getText());
30          boolean tanggal = false;
31          boolean bulan = false;
32          if (tanggal.getSelectedItems().toString().equals("")) {

```

```

167 |     }
168 | }
169 |
170 | /**
171 |  * @param args the command line arguments
172 |  */
173 | public static void main(String args[]) {
174 |     /* Set the Nimbus look and feel */
175 |     Look and feel setting code (optional)
176 |
177 |     /* Create and display the form */
178 |     java.awt.EventQueue.invokeLater(new Runnable() {
179 |         public void run() {
180 |             new ViewPengirimanBarang().setVisible(true);
181 |         }
182 |     });
183 | }
184 |
185 | // Variables declaration - do not modify
186 | private javax.swing.JComboBox<String> Bulan;
187 | private javax.swing.JButton Simpan;
188 | private javax.swing.JTextField TFAlamat;
189 | private javax.swing.JTextField TFNama;
190 | private javax.swing.JTextField TFResi;
191 | private javax.swing.JTable Tablelist;
192 | private javax.swing.JComboBox<String> Tanggal;
193 | private javax.swing.JLabel jLabel1;
194 | private javax.swing.JLabel jLabel2;
195 | private javax.swing.JLabel jLabel3;
196 | private javax.swing.JLabel jLabel4;
197 | private javax.swing.JLabel jLabel5;
198 | private javax.swing.JLabel jLabel6;
199 | private javax.swing.JScrollPane jScrollPane1;
200 |
201 | private javax.swing.JTextField jTextField1;
202 | // End of variables declaration
203 | }
204 |
205 |
206 |
207 |
208 |
209 |
210 |
211 |
212 |
213 |
214 |
215 |
216 |
217 |
218 |
219 |
220 |
221 |
222 |
223 |

```

Pembuatan Frame:

- ViewPengirimanBarang adalah kelas utama yang memperluas javax.swing.JFrame dan berfungsi sebagai jendela utama aplikasi.
- Di dalam kelas ini, initComponents() dipanggil untuk menginisialisasi komponen GUI, seperti label, tombol, bidang teks, dan tabel.

Komponen GUI:

- Label jLabel1 hingga jLabel6 digunakan untuk menampilkan teks deskripsi seperti "Nama Penerima", "Alamat Penerima", "Nomor Resi", "Tanggal Pengiriman", dan "Bulan Pengiriman".
- TFNama, TFAlamat, dan TFResi adalah bidang teks untuk memasukkan nama penerima, alamat penerima, dan nomor resi.
- Tanggal dan Bulan adalah combo box untuk memilih tanggal dan bulan pengiriman.
- Simpan adalah tombol untuk menyimpan data yang dimasukkan.
- Tablelist adalah tabel untuk menampilkan data yang disimpan.

Event Listener pada Tombol Simpan:

- Ketika tombol Simpan diklik, metode `SimpanActionPerformed` dipanggil.
- Dalam metode ini, data dari `TFNama`, `TFAalamat`, `TFResi`, `Tanggal`, dan `Bulan` diambil sebagai `String` dan disimpan ke dalam variabel `nama`, `alamat`, `resi`, `tanggal`, dan `bulan`.
- Dilakukan pengecekan apakah semua field telah terisi (nomor resi, tanggal, dan alamat tidak boleh kosong).
- Jika semua field telah diisi, data akan ditambahkan ke tabel (`Tablelist`) menggunakan `DefaultTableModel`.
- Setelah data disimpan, field input dikosongkan untuk siap menerima data baru.
- Jika ada field yang kosong, pesan error ditampilkan menggunakan `JOptionPane.showMessageDialog`.

Main Method:

- Metode `main` membuat instance `ViewPengirimanBarang` dan membuatnya terlihat di layar menggunakan `setVisible(true)`.

Secara keseluruhan, aplikasi ini bertugas menampilkan form untuk input data pengiriman barang dan memungkinkan pengguna menyimpan data tersebut ke tabel tampilan.

Pertemuan 4

Langkah pertama saya membuat database terlebih dahulu sebelum membuat program aplikasi cargo/pengiriman barang, contoh struktur sebagai berikut :

Server: 127.0.0.1 » Database: db_cargo » Tabel: tb_cargo

Jelajahi Struktur SQL Cari Tambahkan Ekspor Impor Hak Akses Operasi Pelacakan Trigger

Struktur tabel Tampilan hubungan

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id_penerima	int(15)			Tidak	Tidak ada		AUTO_INCREMENT	Ubah Hapus Lainnya
2	nama_penerima	varchar(30)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
3	alamat_penerima	varchar(40)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
4	nomor_resi	varchar(50)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya
5	tanggal	tinyint(4)			Tidak	Tidak ada			Ubah Hapus Lainnya
6	bulan	varchar(20)	utf8mb4_general_ci		Tidak	Tidak ada			Ubah Hapus Lainnya

Pilih Semua Dengan pilihan: Jelajahi Ubah Hapus Utama Unik Indeks Spasial Teks penuh Add to central

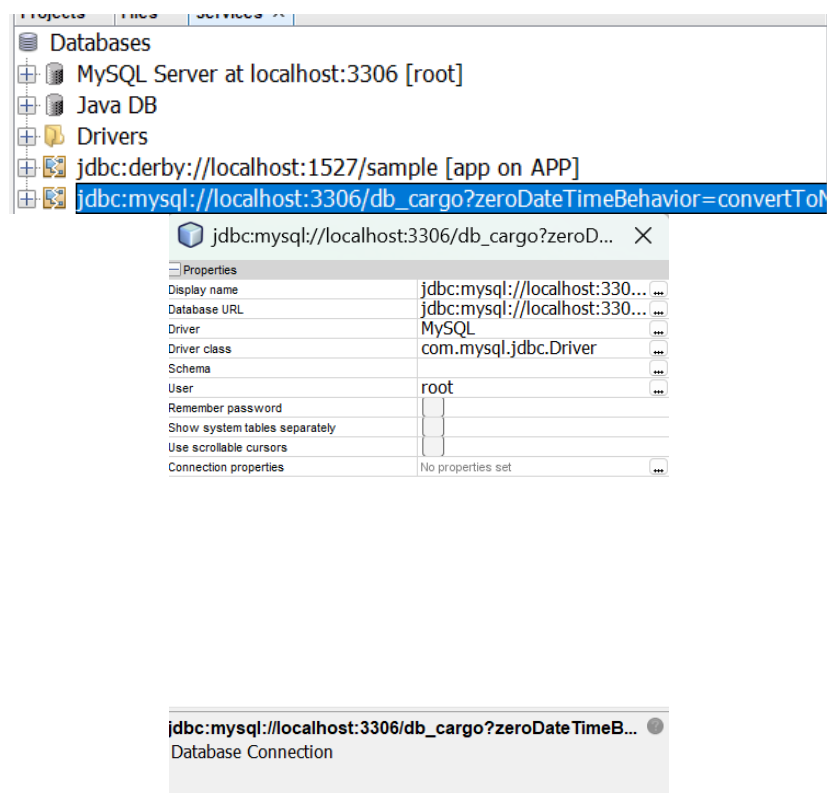
Cetak Usulkan struktur tabel Lacak tabel Move columns Normalisasi

Tambahkan 1 kolom setelah bulan Kirim

Indeks

Tindakan	Nama kunci	Jenis	Unik	Dipadatkan	Kolom	Kardinalitas	Penyortiran	Tak Ternilai	Komentar
Ubah Rename Hapus	PRIMARY	BTREE	Ya	Tidak	id_penerima	0	A	Tidak	

Kemudian, contoh sudah tersambungnya netbeans(java) ke mysql, contohnya sebagai berikut :



Selanjutnya, saya akan melogikakan code dari program cargo.java, sebagai berikut :

1. Import Statements

```
import java.awt.HeadlessException;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import javax.swing.JOptionPane;  
import javax.swing.table.DefaultTableModel;
```

Mengimpor kelas-kelas yang diperlukan:

- `HeadlessException`: untuk menangani error terkait antarmuka grafis di sistem tanpa tampilan.
- `Connection`, `DriverManager`, `SQLException`: untuk mengelola koneksi ke database MySQL dan menangani error yang mungkin terjadi saat koneksi.
- `JOptionPane`: untuk menampilkan pesan ke pengguna.
- `DefaultTableModel`: untuk mengelola data dalam tabel pada GUI.

2. Deklarasi Kelas Cargo

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

Cargo adalah kelas yang merupakan turunan dari `JFrame`, yang berarti kelas ini merupakan jendela GUI pada aplikasi desktop Java Swing.

3. Deklarasi Variabel

```
private Statement St;  
private Connection Con;  
private ResultSet Rs;  
private final String sql = "";
```

- `St`: Objek `Statement` untuk menjalankan perintah SQL pada database.
- `Con`: Objek `Connection` yang mewakili koneksi ke database.
- `Rs`: Objek `ResultSet` untuk menyimpan hasil dari query yang dieksekusi pada database.
- `sql`: String kosong yang nantinya bisa digunakan untuk menyimpan perintah SQL.

4. Konstruktor Cargo()

```
public Cargo() {
    initComponents();
    KoneksiDatabase();
}
```

- Konstruktor ini memanggil metode `initComponents()` untuk menginisialisasi elemen-elemen GUI.
- Memanggil `KoneksiDatabase()` untuk mencoba menghubungkan aplikasi ke database.

5. Metode `KoneksiDatabase()`

```
private void KoneksiDatabase(){
try {
    // Menentukan Driver Database

    Class.forName("com.mysql.jdbc.Driver");
    // untuk mengkoneksikan DB ke driver

    Con = DriverManager.getConnection("jdbc:mysql://localhost:3306/db_cargo",
    "root", "");

    // pesan koneksi berhasil

    JOptionPane.showMessageDialog(null, "Koneksi Berhasil");

} catch (HeadlessException | ClassNotFoundException | SQLException e) {

    // pesan Koneksi gagal

    System.out.println("Koneksi Gagal" + e.getMessage());
}
}
```

- Metode ini bertujuan untuk menghubungkan aplikasi dengan database MySQL (`db_cargo`) yang terletak di `localhost` dengan port `3306`.
- Dalam `try` block:
- `Class.forName("com.mysql.jdbc.Driver")`: Memuat driver MySQL JDBC untuk menghubungkan Java dengan MySQL.
- `DriverManager.getConnection(...)`: Mencoba menghubungkan ke database `db_cargo` menggunakan username `root` dan password kosong (`""`).

- Jika koneksi berhasil, menampilkan dialog pesan "Koneksi Berhasil" kepada pengguna.
- `catch` block menangani berbagai kemungkinan error:
- `HeadlessException`: Error terkait antarmuka.
- `ClassNotFoundException`: Error jika driver MySQL JDBC tidak ditemukan.
- `SQLException`: Error jika terjadi masalah saat koneksi dengan database.
- Jika error terjadi, mencetak pesan error ke konsol dengan tambahan `e.getMessage()` yang menjelaskan alasan kegagalan koneksi.

Kesimpulan

Kode ini membangun antarmuka awal dari aplikasi Java GUI yang berhubungan dengan sistem kargo atau pengiriman barang, di mana kelas Cargo berfungsi sebagai jendela utama. Pada inisialisasi, aplikasi mencoba terhubung ke database MySQL (db_cargo) dan akan menampilkan pesan apakah koneksi berhasil atau gagal.

Pertemuan 5

Berikut penjelasan logika dari kode tersebut:

1. Mengambil nilai dari input teks:

Nilai-nilai yang dimasukkan pengguna diambil dari **text field** (seperti `txtNama`, `txtAlamat`, dll.) dan disimpan ke dalam variabel string seperti `nama`, `alamat`, `resi`, `tanggal`, dan `bulan`.

```
String nama = txtNama.getText();
String alamat = txtAlamat.getText();
String resi = txtResi.getText();
String tanggal = txtTanggal.getText();
String bulan = txtBulan.getText();
```

2. Menyiapkan query SQL untuk mengupdate data:

Query menggunakan **PreparedStatement** dengan placeholder (?) untuk mencegah SQL injection. Placeholder akan diisi dengan data dari variabel yang sebelumnya diambil.

```
String sql = "UPDATE tabel_pengiriman SET nama = ?, alamat = ?, resi = ?, tanggal = ?, bulan = ? WHERE id = ?";
```

```
PreparedStatement pst = Con.prepareStatement(sql);
```

3. Mengatur parameter query:

Metode `setString` digunakan untuk mengganti placeholder ? dalam query dengan nilai

aktual dari variabel nama, alamat, resi, tanggal, dan bulan.

```
pst.setString(1, nama);  
pst.setString(2, alamat);  
pst.setString(3, resi);  
pst.setString(4, tanggal);  
pst.setString(5, bulan);
```

4. **Menjalankan query:**

Query dieksekusi menggunakan executeUpdate(), yang mengembalikan jumlah baris yang terpengaruh dalam tabel database.

```
int rowsUpdated = pst.executeUpdate();
```

5. **Memberikan umpan balik kepada pengguna:**

- Jika rowsUpdated > 0, artinya data berhasil diupdate, dan pesan sukses ditampilkan.
- Jika tidak ada baris yang diupdate, maka data dengan id yang diberikan tidak ditemukan.

```
if (rowsUpdated > 0) {  
    JOptionPane.showMessageDialog(null, "Data berhasil diupdate!");  
} else {  
    JOptionPane.showMessageDialog(null, "Data tidak ditemukan!");  
}
```

6. **Penanganan kesalahan:**

Jika terjadi kesalahan dalam proses query (misalnya, koneksi gagal atau query salah), akan ditangkap oleh blok catch dan pesan kesalahan ditampilkan menggunakan JOptionPane.

```
} catch (SQLException e) {  
    JOptionPane.showMessageDialog(null, "Error: " + e.getMessage());  
}
```

Pertemuan 6

```
231
232 private void EditJActionPerformed(java.awt.event.ActionEvent evt) {
233     String nama = TFNama.getText();
234     String alamat = TFAlamat.getText();
235     String resi = TFResi.getText();
236     String tanggal = Tanggal.getSelectedItem().toString();
237     String bulan = Bulan.getSelectedItem().toString();
238     int id = Integer.parseInt(TFId.getText());
239
240     Session session = HibernateUtil.getSessionFactory().openSession();
241     Transaction tx = session.beginTransaction();
242
243     try {
244         CargoEntity cargo = (CargoEntity) session.get(CargoEntity.class, id);
245         if (cargo != null) {
246             cargo.setNama(nama);
247             cargo.setAlamat(alamat);
248             cargo.setResi(resi);
249             cargo.setTanggal(tanggal);
250             cargo.setBulan(bulan);
251
252             session.update(cargo); // Update data
253             tx.commit();
254             JOptionPane.showMessageDialog(this, "Data berhasil diupdate!");
255         } else {
256             JOptionPane.showMessageDialog(this, "Data tidak ditemukan!");
257         }
258     } catch (HeadlessException e) {
259         tx.rollback();
260         JOptionPane.showMessageDialog(this, "Gagal mengupdate data: " + e.getMessage());
261     } finally {
262         session.close();
263     }
264
265 }
```

sebuah event handler untuk tombol **Edit** (EditJButton) pada aplikasi pengelolaan data kargo. Berikut adalah langkah-langkah kerja program:

1. Mengambil Input dari Form

Data diambil dari beberapa komponen GUI (TextField, ComboBox) seperti berikut:

- **TFNama**: Mengambil nama kargo.
- **TFAlamat**: Mengambil alamat tujuan.
- **TFResi**: Mengambil nomor resi kargo.
- **Tanggal** dan **Bulan**: Mengambil tanggal dan bulan pengiriman dari komponen dropdown.
- **TFId**: Mengambil ID kargo yang ingin diubah, kemudian dikonversi ke tipe integer menggunakan `Integer.parseInt()`.

2. Membuka Koneksi ke Database

Menggunakan Hibernate, koneksi ke database dibuka dengan:

- **HibernateUtil.getSessionFactory().openSession()**: Membuka sesi Hibernate untuk komunikasi dengan database.
- **Transaction tx**: Memulai transaksi untuk memastikan data diubah secara konsisten.

3. Mencari Data Berdasarkan ID

- Program menggunakan `session.get(CargoEntity.class, id)` untuk mencari entitas kargo berdasarkan **ID**.
- Jika data dengan ID tersebut ditemukan:
 - Nilai atribut entitas kargo diperbarui dengan data baru dari form.
- Jika data tidak ditemukan:
 - Menampilkan dialog pesan "Data tidak ditemukan!".

4. Memperbarui Data ke Database

- **session.update(cargo)**: Memperbarui data entitas di database.

- **tx.commit()**: Menyimpan perubahan ke database.
 - Jika berhasil, dialog pesan "Data berhasil diupdate!" ditampilkan.
 - 5. **Menangani Kesalahan**
 - Jika terjadi error selama proses (misalnya kesalahan koneksi atau database), transaksi akan dibatalkan dengan **tx.rollback()**.
 - Dialog pesan kesalahan ditampilkan, berisi rincian dari error tersebut (e.getMessage()).
 - 6. **Menutup Sesi**
 - **session.close()**: Menutup sesi Hibernate setelah operasi selesai, baik berhasil maupun gagal, untuk menghindari kebocoran koneksi.
-

Penjelasan Singkat Komponen Utama

- **Try-Catch-Finally**: Digunakan untuk menangani error dan memastikan sesi ditutup dalam semua kondisi.
- **Dialog Pesan**: Menggunakan JOptionPane.showMessageDialog untuk memberikan umpan balik kepada pengguna, baik saat berhasil maupun gagal.
- **Hibernate**: Framework untuk mempermudah operasi CRUD pada database berbasis objek.

OUTPUT

Pertemuan 2

Output 1: Pengiriman Laut

```
1. Laut 1 bulan 2x
2. Udara 1 bulan 4x
Silahkan pilih Pengiriman:
1
=====
Pilih Jenis Pengiriman
=====
1. Pengiriman Melalui kapal laut
Masukkan Pilihan:
1
Paket perhari: 200000
Pengiriman melalui kapal laut dipilih
```

Output 2: Pengiriman Udara

```
1. Laut 1 bulan 2x
2. Udara 1 bulan 4x
Silahkan pilih Pengiriman:
2
=====
Pilih Jenis Pengiriman
=====
1. Pengiriman Melalui Udara
Masukkan Pilihan:
1
Paket perhari: 200000
Pengiriman melalui udara dipilih
```

Pertemuan 3

Data Pengiriman Barang

Nama Penerima

Alamat Penerima

Nomor Resi

Tanggal Pengiriman

1

▼

Bulan Pengiriman

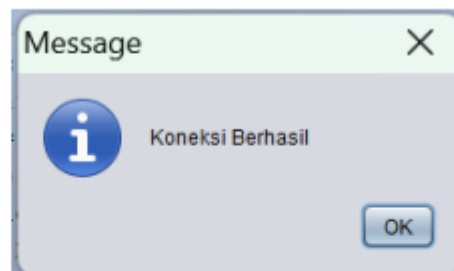
Januari

▼

Simpan

Title 1	Title 2	Title 3	Title 4
Hikmal	J1	50421844	7 Januari
try	j3	2222	2 April

Pertemuan 4



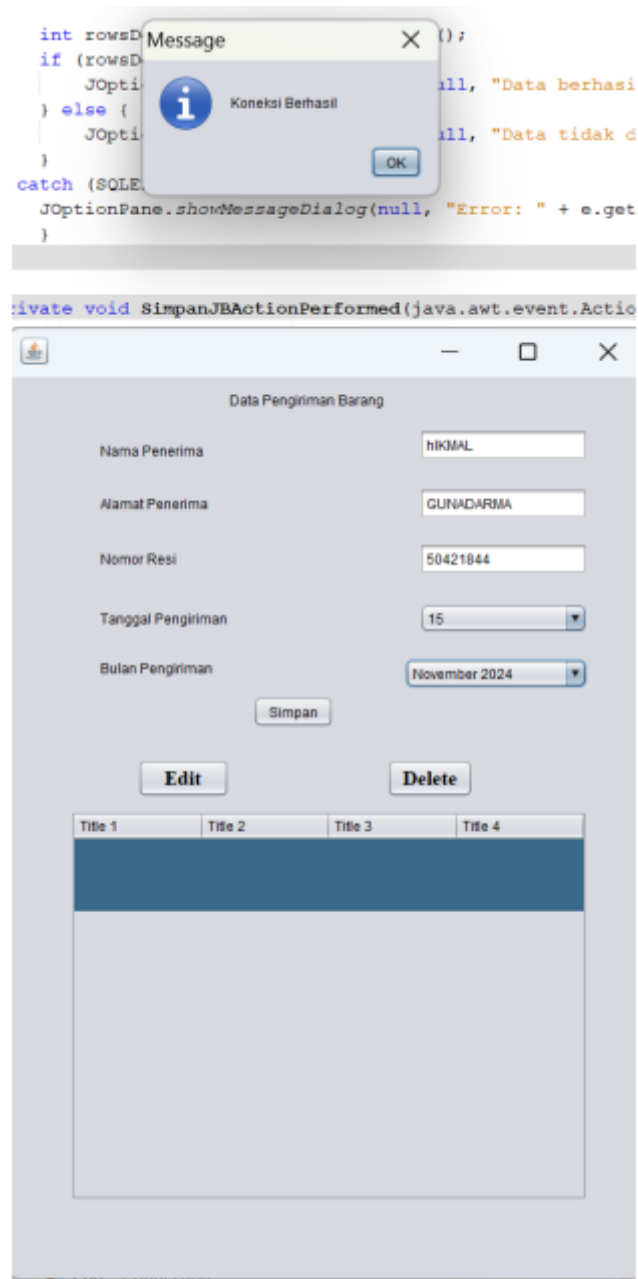
Application window titled "Data Pengiriman Barang" with a standard Windows title bar (minimize, maximize, close buttons).

Fields for data entry:


- Nama Penerima:
- Alamat Penerima:
- Nomor Resi:
- Tanggal Pengiriman:
- Bulan Pengiriman:

Title 1	Title 2	Title 3	Title 4

Pertemuan 5



Pertemuan 6

—□×

Data Pengiriman Barang

Nama Penerima

Alamat Penerima

Nomor Resi

Tanggal Pengiriman

22

▼

Bulan Pengiriman

November 2024

▼

Simpan

Edit

Delete

Title 1	Title 2	Title 3	Title 4
Mohammad Hikm...	Gunadarma J1	50421844	22 November 2024
Muhammad Try	Gunadarma J3	12345678	22 November 2024
Marcel Crishtian	Gunadarma J6	50426455	22 November 2024
Lukman Khafi	Gunadarma J2	50423755	22 November 2024