

# **PRAKTIKUM REKAYASA PERANGKAT LUNAK 2**



## **MANUAL BOOK**

### **Perancangan Aplikasi E-Groceries Menggunakan Netbeans**

**Nama Praktikan : Prasetya Harkrisnowo**  
**Kelas : 4IA18**  
**Fakultas : Teknologi Industri**  
**Jurusan : Teknik Informatika**  
**PJ : Marcell**

**Ditulis Guna Melengkapi Sebagian Syarat Praktikum Rekayasa Perangkat Lunak  
Jenjang S1  
Universitas Gunadarma**

**2023**

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Dalam era digital ini, kebiasaan belanja masyarakat telah berubah secara signifikan. Aplikasi e-groceries atau belanja online untuk kebutuhan harian seperti makanan dan barang rumah tangga menjadi tren yang semakin populer. Latar belakang tersebut didukung oleh perkembangan teknologi informasi dan keinginan masyarakat untuk memperoleh kepraktisan serta kenyamanan dalam berbelanja. Oleh karena itu, kehadiran aplikasi e-groceries menjadi solusi efektif untuk memenuhi kebutuhan konsumen modern.

### **1.2 Tujuan**

Tujuan utama dari aplikasi GrocerEase adalah meningkatkan kemudahan berbelanja dengan menyederhanakan proses belanja online, menghemat waktu pengguna dengan menghilangkan kebutuhan untuk pergi ke toko fisik, membantu dalam manajemen belanjaan harian, memberikan pilihan pengiriman yang fleksibel, meningkatkan transparansi produk dengan ulasan pengguna, dan memungkinkan integrasi dengan berbagai toko fisik dan online, sehingga menghadirkan pengalaman berbelanja yang lebih nyaman, efisien, dan hemat waktu bagi pengguna.

### **1.3 Batasan Masalah**

Adapun batasan masalah atau ruang lingkup dalam pengembangan aplikasi Grocerease adalah sebagai berikut:

1. Platform: Aplikasi Grocerease akan tersedia untuk pengguna perangkat berbasis Android.
2. Wilayah Layanan: Aplikasi ini akan beroperasi di wilayah perkotaan besar. Akses ke aplikasi mungkin terbatas pada wilayah-wilayah ini, dan layanan pengiriman akan disesuaikan dengan wilayah-wilayah tersebut.
3. Bahasa: Aplikasi GrocerEase tidak mendukung bahasa selain bahasa Indonesia.
4. Detail Produk: Aplikasi GrocerEase menampilkan daftar produk yang tersedia ditoko online.
5. Fitur Pemesanan: Aplikasi GrocerEase tidak memiliki fitur pembayaran, pengiriman, atau pengembalian barang, tetapi menggunakan fitur keranjang untuk pemesanan produk.
6. Produk dan Kategori: Grocerease akan fokus pada produk-produk kebutuhan sehari-hari seperti makanan, minuman, barang-barang rumah tangga, dan produk konsumen umum lainnya.

## **BAB II**

### **PEMBAHASAN**

#### **2.1 NetBeans**

Netbeans adalah salah satu IDE (Integrated Development Environment) yang digunakan untuk membuat aplikasi berbasis Java. Netbeans menyediakan berbagai fitur yang memudahkan pengembang dalam menulis, menguji, dan menjalankan kode Java. Beberapa fitur tersebut antara lain:

- Editor kode yang mendukung sintaksis highlight, code completion, refactoring, dan debugging.
- Project management yang memungkinkan pengembang untuk mengatur struktur dan dependensi proyek Java.
- GUI builder yang memungkinkan pengembang untuk mendesain antarmuka grafis dengan mudah menggunakan drag and drop.
- Platform modular yang memungkinkan pengembang untuk menambahkan plugin dan modul sesuai kebutuhan.
- Integrasi dengan berbagai alat dan layanan lain seperti Git, Maven, JUnit, MySQL, dan lain-lain.

#### **2.2 Spring**

Spring adalah salah satu framework Java yang populer dan banyak digunakan oleh para developer. Spring memiliki beberapa fitur yang membuatnya menarik, seperti dependency injection, aspect-oriented programming, dan web MVC. Spring juga mendukung integrasi dengan berbagai teknologi lain, seperti Hibernate, JPA, JDBC, RESTful API, dan lain-lain. Spring memudahkan developer untuk membuat aplikasi yang modular, fleksibel, dan berkualitas tinggi.

## **2.3 Hibernate**

Hibernate adalah alat ORM (Object Relational Mapping) yang memetakan objek ke data yang disimpan dalam database. Alat ORM mempermudah penciptaan data, manipulasi data, dan akses data<sup>1</sup>. Alat ORM secara internal menggunakan API JDBC untuk berinteraksi dengan database.

Hibernate juga mengimplementasikan spesifikasi JPA (Java Persistence API) untuk persistensi data. JPA adalah spesifikasi Java yang memberikan fungsionalitas tertentu dan standar untuk alat ORM. Paket `javax.persistence` berisi kelas dan antarmuka JPA.

### **2.3.1.1 Fungsi Hibernate**

Fungsi utama Hibernate adalah memetakan objek-objek dalam aplikasi Java ke tabel-tabel dalam database, serta menyediakan mekanisme untuk menyimpan, mengambil, dan mengelola data antara aplikasi dan database secara transparan.

### **2.3.1.2 Manfaat Hibernate**

Berikut adalah beberapa manfaat menggunakan Hibernate dalam pengembangan aplikasi Java:

1. Efisiensi Waktu: Hibernate mempermudah pengolahan data, seperti operasi insert, update, delete, dan select. Ini menghemat waktu karena Anda tidak perlu menulis query SQL secara manual.
2. Independen Database: Hibernate menggunakan HQL (Hibernate Query Language) yang independen dari database. Ini berarti Anda dapat beralih ke database lain dengan sedikit konfigurasi dan tanpa perlu mengubah kode aplikasi.
3. Pemetaan Otomatis: Hibernate dapat memetakan model domain berorientasi objek ke dalam database relasional. Ini memudahkan Anda dalam mengelola data dan struktur database.
4. Pemeliharaan Skema Database: Hibernate dapat memelihara skema database hanya dengan menggunakan file XML.
5. Dukungan JPA: Hibernate mendukung Java Persistence API (JPA), yang memberikan standar untuk alat ORM. Ini memastikan bahwa kode Anda tetap konsisten dan mudah dipahami.

6. Gratis dan Open Source: Hibernate adalah kerangka kerja open source dan gratis. Ini berarti Anda dapat menggunakannya dan memodifikasinya sesuai kebutuhan Anda.

### **2.3.2 Kerugian dan Masalah Hibernate**

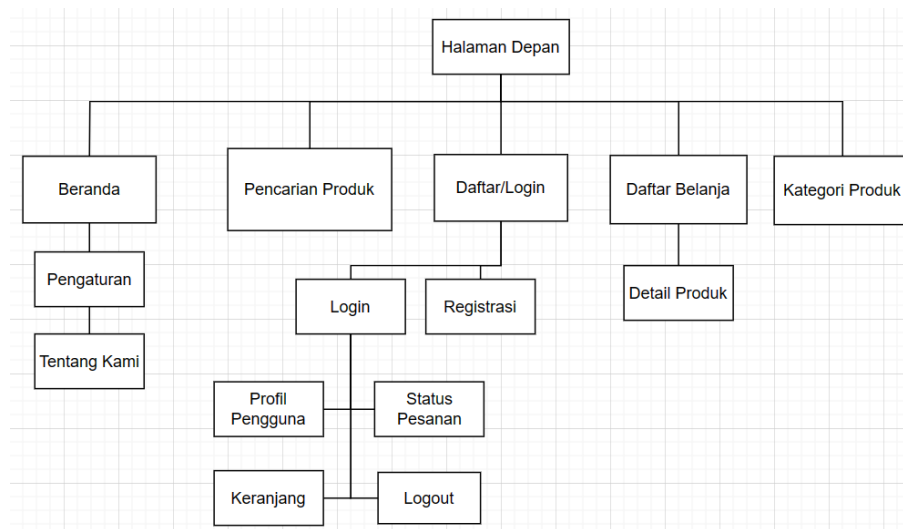
Berikut adalah beberapa kerugian dan masalah yang mungkin dihadapi saat menggunakan Hibernate:

1. Kurva Belajar: Hibernate memiliki banyak fitur dan konsep yang perlu dipahami, sehingga membutuhkan waktu untuk mempelajarinya.
2. Performa: Hibernate cenderung lebih lambat dibandingkan dengan JDBC murni karena Hibernate menghasilkan banyak pernyataan SQL dalam runtime. Selain itu, terkadang penyetelan performa dan debugging bisa menjadi sulit.
3. Kekurangan Kontrol: Dalam beberapa kasus, Hibernate dapat menyebabkan kehilangan kontrol terhadap beberapa aspek operasi database.
4. Kompleksitas: Hibernate dapat menambah kompleksitas ke aplikasi, terutama jika aplikasi tersebut sudah cukup kompleks.
5. Masalah dengan Pengolahan Batch: Hibernate mungkin tidak cocok untuk pengolahan batch, dalam hal ini disarankan untuk menggunakan JDBC murni.
6. Masalah Memori: Set data yang besar masih bisa menyebabkan masalah memori.
7. Keterbatasan dalam Akses Jarak Jauh dan Distribusi: Hibernate memiliki sedikit atau tidak ada kemampuan untuk akses jarak jauh dan distribusi.
8. Keterbatasan dalam Klustering: Hibernate memiliki keterbatasan dalam kemampuannya untuk melakukan klustering

## **2.4 ORM(Object Relational Mapping)**

Object Relational Mapping (ORM) adalah teknik yang merubah suatu tabel menjadi sebuah objek yang nantinya mudah untuk digunakan. Objek yang dibuat memiliki properti yang sama dengan field-field yang ada pada tabel tersebut. ORM memungkinkan kita melakukan query dan memanipulasi data di database menggunakan pemrograman berorientasi objek.

## 2.5 Struktur Navigasi



Aplikasi E-groceries memiliki struktur navigasi yang terorganisir. Pengguna dapat memulai dari halaman depan untuk mencari produk atau masuk ke akun melalui daftar/login. Di daftar belanja, mereka dapat mengelola produk yang dipilih. Kategori produk memudahkan pencarian. Submenu pengaturan menyediakan informasi "Tentang Kami". Fitur login memberikan akses ke profil, status pesanan, dan keranjang belanja. Detail produk menyajikan informasi rinci. Dengan struktur ini, aplikasi memastikan pengalaman belanja yang intuitif dan efisien.

## 2.6 MySQL

MySQL adalah salah satu sistem manajemen database yang menggunakan bahasa SQL (Structured Query Language) untuk mengelola data yang disimpan dalam bentuk tabel relasional. MySQL memiliki banyak fungsi dan fitur yang dapat membantu dalam pengembangan aplikasi berbasis data, seperti:

- Mendukung integrasi dengan bahasa pemrograman lain, seperti PHP, Java, Python, dan lainnya.
- Tidak membutuhkan RAM besar, sehingga dapat berjalan di berbagai platform.
- Mendukung multi user, yaitu dapat melayani banyak pengguna secara bersamaan.
- Bersifat open source, yaitu dapat digunakan secara gratis dan dimodifikasi sesuai kebutuhan.
- Memiliki struktur tabel yang fleksibel, yaitu dapat menambah, mengubah, atau menghapus kolom tanpa mengganggu data yang ada,

## 2.7 JDBC

JDBC adalah singkatan dari Java Database Connectivity, yaitu sebuah API Java yang berfungsi untuk menghubungkan dan mengeksekusi query dengan database relasional. JDBC API menggunakan driver JDBC yang ditulis dalam bahasa Java untuk berkomunikasi dengan berbagai jenis database, seperti MySQL, Oracle, SQL Server, dan lainnya.

## 2.8 Kebutuhan Spesifik Dan Kebutuhan Perangkat

### 2.8.1 Kebutuhan Spesifik

No	Fitur	Kebutuhan Fungsional	Kebutuhan Antarmuka	Kebutuhan Unjuk Kerja
1	Registrasi Pengguna	Pengguna dapat mendaftar dengan mengisi informasi pribadi dan alamat. Data pendaftaran akan disimpan dalam basis data.	<ul style="list-style-type: none"><li>Formulir pendaftaran dengan kolom untuk nama, alamat, email, kata sandi, dan nomor telepon.</li><li>- Tombol "Daftar".</li></ul>	<ul style="list-style-type: none"><li>Dapat dimasukkan kedalam akun user/admin/</li><li>Dapat digunakan secara bersamaan oleh banyak user</li></ul>
2	Pencarian Produk	Pengguna dapat mencari produk berdasarkan nama, merek, atau kategori. Hasil pencarian akan ditampilkan dalam daftar.	<ul style="list-style-type: none"><li>Kotak pencarian.</li><li>- Tombol "Cari".</li></ul>	<ul style="list-style-type: none"><li>Hasil pencarian harus ditampilkan dalam waktu kurang dari 1 detik.</li></ul>



3	Daftar Belanja Pribadi	Pengguna dapat membuat, mengedit, dan menghapus daftar belanja pribadi mereka. Daftar belanja akan tersimpan dalam profil pengguna.	Halaman daftar belanja pribadi dengan opsi tambah, edit, dan hapus produk.	<ul style="list-style-type: none"> <li>Pengguna dapat mengelola daftar belanja mereka tanpa keterlambatan berarti.</li> </ul>
4	Ulasan dan Peringkat Produk	Pengguna dapat memberikan ulasan dan peringkat produk. Ulasan akan ditampilkan untuk membantu pengguna lain dalam memilih produk.	<ul style="list-style-type: none"> <li>Halaman produk dengan opsi untuk memberikan ulasan dan peringkat.</li> </ul>	Ulasan dan peringkat produk harus terlihat oleh pengguna lain dalam 2 jam setelah diunggah.
5	Integrasi Toko Mitra	Aplikasi akan terintegrasi dengan sistem inventaris toko mitra. Ketersediaan produk harus diperbarui secara real-time.	Integrasi dengan sistem inventaris toko mitra.	Ketersediaan produk harus diperbarui secara real-time setiap kali ada perubahan stok.

### 2.8.2 Kebutuhan Perangkat

#### 1. Perangkat PC.

Spesifikasi Minimum PC:

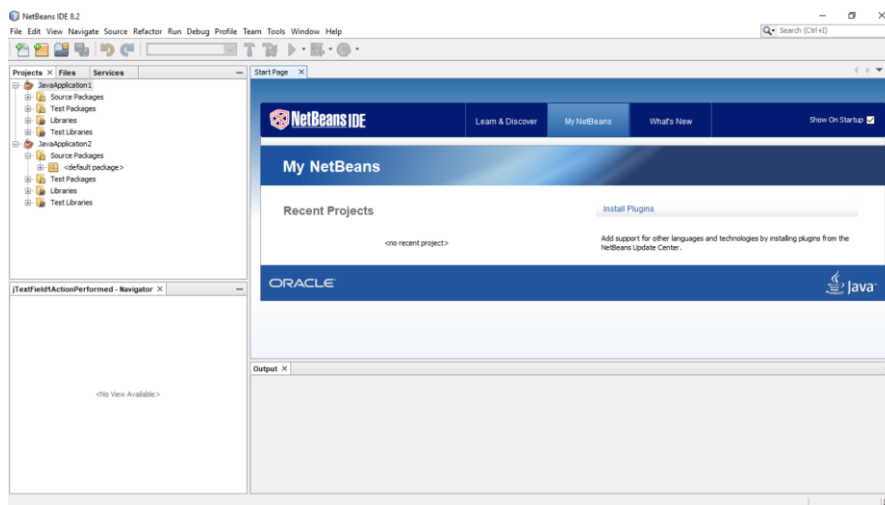
- Prosesor (CPU): Intel Core i3 atau AMD Ryzen 3.
- RAM: 4 GB (Direkomendasikan: 8 GB atau lebih untuk kinerja yang lebih baik).
- Penyimpanan: SSD dengan ruang kosong minimal 4 GB (untuk instalasi Android Studio dan proyek).
- Sistem Operasi: Windows 7/8/10 (64-bit), macOS, atau distribusi Linux yang kompatibel.
- Kartu Grafis: Prosesor grafis yang mendukung OpenGL 2.0 atau yang lebih tinggi.
- Resolusi Layar: 1280x800 piksel atau lebih.
- Koneksi Internet: Dibutuhkan untuk mengunduh paket-paket dan dependensi.

## BAB III

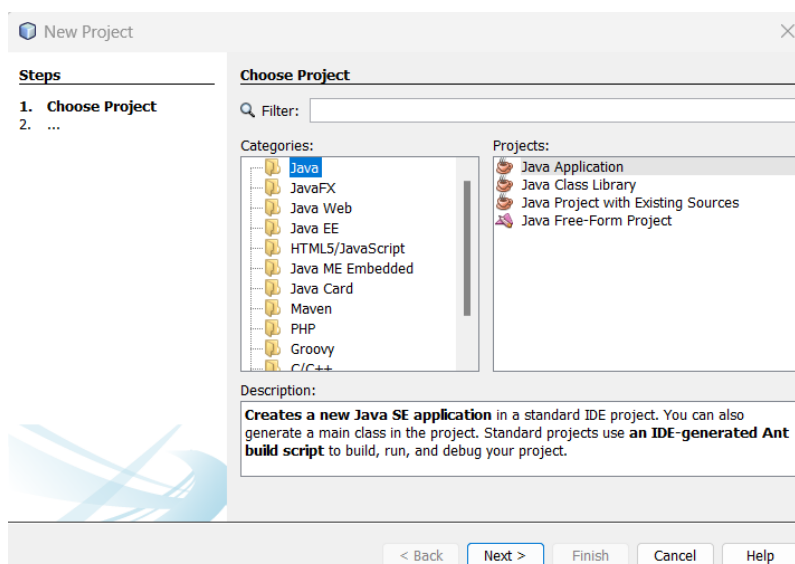
### ANALISA DAN PERANCANGAN

#### 3.1 Membuat OOP Pada Java

Pada Sesi kali ini, akan dijelaskan cara implementasi konsep Object-Oriented Programming (OOP) pada bahasa pemrograman Java menggunakan lingkungan pengembangan NetBeans. Proyek yang akan kita buat yaitu DataBarang2. Berikut merupakan tampilan awal dari Netbeans 8.2.

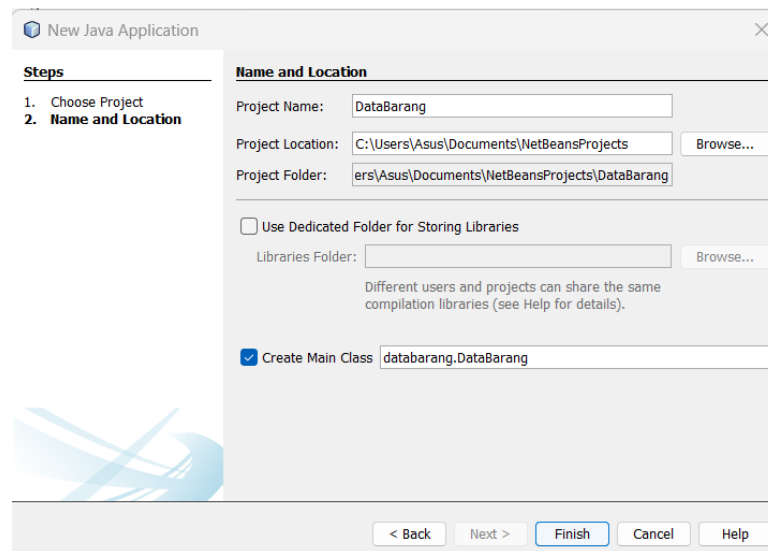


Navigasikan ke file -> New Project. maka akan muncul tampilan dibawah ini

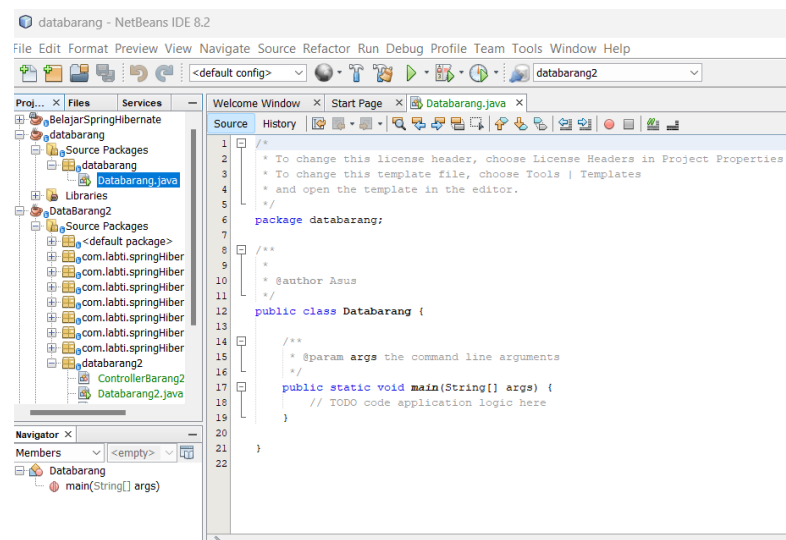


Pada Categories, pilihlah '**java**' dan pada Projects pilih '**Java Application**'.

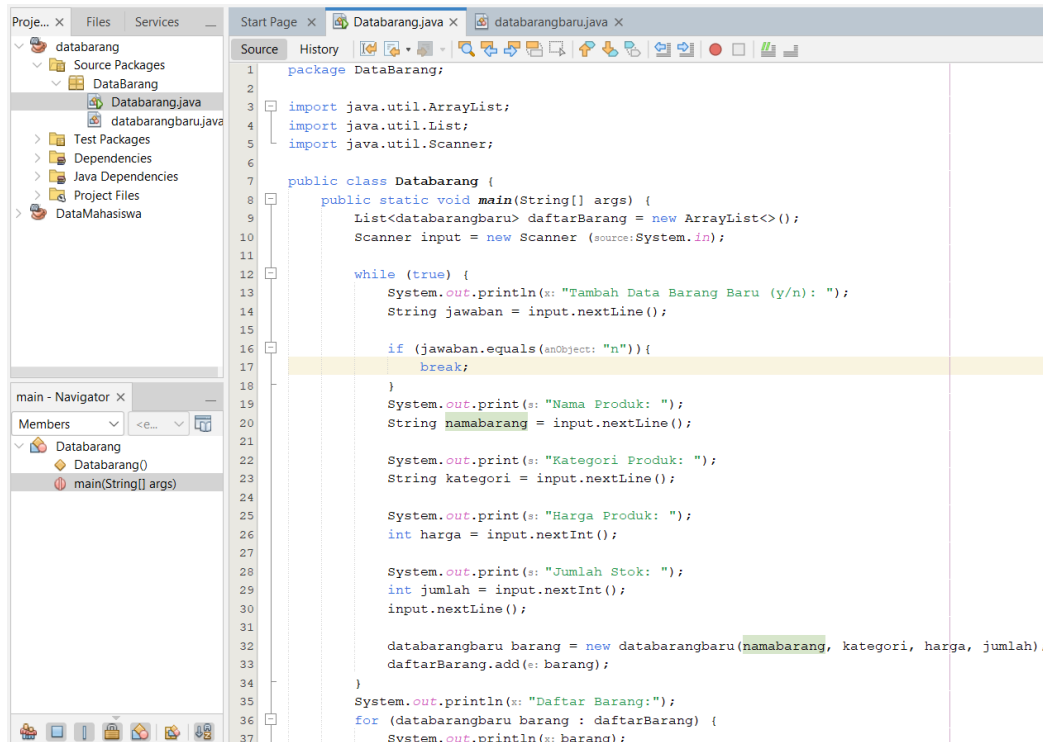
Setelah memilih Java Application pada Projects, kemudian membuat Project Name dengan nama **“DataBarang2”** dan jika sudah tekan **“Finish”**.



Setelah mengklik finish tampilan yang akan muncul sebagai berikut:



Setelah kita membuat java class, selanjutnya membuat kodingan pada Databarang2.java sebagai berikut:



Berikut adalah penjelasan Logika dari file Databarang.java:

### 3.1.1 Databarang.java

```
1 package DataBarang;
```

Ini adalah deklarasi paket (package) yang digunakan untuk mengorganisasi kelas-kelas Java dalam sebuah folder atau direktori.

```

3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Scanner;

```

Program mengimpor kelas-kelas yang diperlukan dari paket java.util, seperti ArrayList (untuk menyimpan daftar barang), List (interface untuk daftar barang), dan Scanner (untuk input dari pengguna).

```
7 public class Databarang {
```

Kelas Databarang merupakan kelas utama yang berisi metode main untuk menjalankan program.

```

9      List<databarangbaru> daftarBarang = new ArrayList<>();
10     Scanner input = new Scanner (source: System.in);

```

- daftarBarang adalah objek List yang akan menyimpan daftar barang (objek databarangbaru).
- input adalah objek Scanner untuk menerima masukan dari pengguna.

```

12     while (true) {
13         System.out.println(x: "Tambah Data Barang Baru (y/n): ");
14         String jawaban = input.nextLine();
15
16         if (jawaban.equals(anObject: "n")){
17             break;
18         }

```

- Program meminta pengguna untuk menambahkan data barang baru dengan pertanyaan "Tambah Data Barang Baru (y/n)". Jika pengguna memasukkan "n", pengulangan berhenti.
- Program akan meminta pengguna untuk memasukkan nama produk, kategori produk, harga produk, dan jumlah stok produk.

```

19     System.out.print(s: "Nama Produk: ");
20     String namabarang = input.nextLine();
21
22     System.out.print(s: "Kategori Produk: ");
23     String kategori = input.nextLine();
24
25     System.out.print(s: "Harga Produk: ");
26     int harga = input.nextInt();
27
28     System.out.print(s: "Jumlah Stok: ");
29     int jumlah = input.nextInt();
30     input.nextLine();

```

Meminta pengguna untuk memasukkan detail barang (nama, kategori, harga, jumlah).

```

32     databarangbaru barang = new databarangbaru(namabarang, kategori, harga, jumlah);
33     daftarBarang.add(e: barang);

```

- Data barang baru kemudian disimpan dalam objek databarangbaru dan ditambahkan ke dalam daftarBarang.

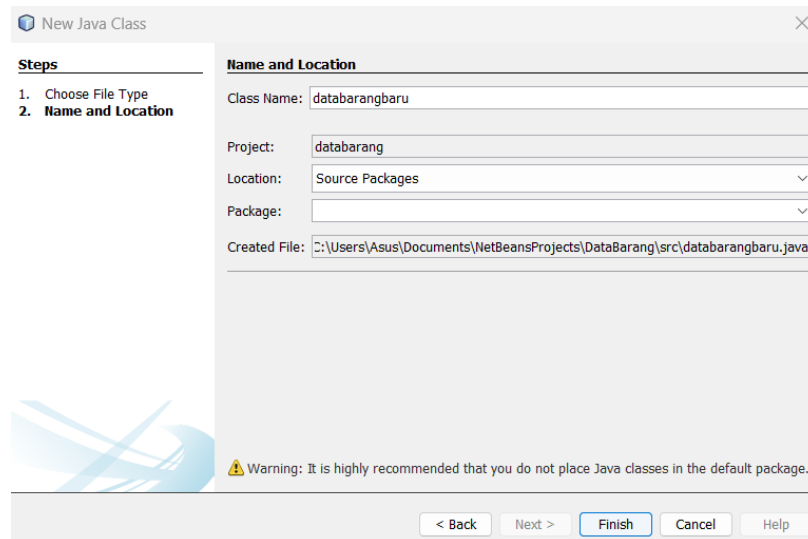
```

35     System.out.println(x: "Daftar Barang:");
36     for (databarangbaru barang : daftarBarang) {
37         System.out.println(x: barang);
38     }
39 }
40 }

```

Program mencetak semua barang yang telah dimasukkan ke dalam daftarBarang.

Setelah membuat databarang.java, selanjutnya membuat databarangbaru.java. Lakukan seperti pada cara pertama, klik kanan pada bagian proyek -> New -> Java Class. Kemudian ganti nama Class Name menjadi **“databarangbaru”**. Setelah itu klik Finish.



Kemudian membuatkan kodingan sebagai berikut :

```

Welcome Window x Start Page x Databarang.java x ControllerBarang2.java x Databarang2.java x databarangbaru2.java x
Source History
1 package databarang2;
2
3 public class databarangbaru2 {
4     private final String namabarang;
5     private final String kategori;
6     private final int harga;
7     private final int jumlah;
8
9     public databarangbaru2(String namabarang, String kategori, int harga, int jumlah){
10         this.namabarang = namabarang;
11         this.kategori = kategori;
12         this.harga = harga;
13         this.jumlah = jumlah;
14     }
15
16     public String getNamaBarang() {
17         return namabarang;
18     }
19
20     public String getKategori(){
21         return kategori;
22     }
23
24     public int getHarga(){
25         return harga;
26     }
27
28     public int getJumlah(){
29         return jumlah;
30     }
31
32     @Override
33     public String toString(){
34         return "Nama Barang: " + namabarang + "Kategori Barang: " + kategori + ", Harga : " + harga + ", Jumlah Stok: " + jumlah;
35     }
36 }

```

Berikut adalah logika dari file databarangbaru2.java:

### 3.1.2 Databarangbaru.java

```
1 package DataBarang;
```

Kelas ini bertindak sebagai representasi dari objek barang dengan atribut yang mendefinisikan detail barang seperti nama, kategori, harga, dan jumlah stok.

```
4 private final String namabarang;  
5 private final String kategori;  
6 private final int harga;  
7 private final int jumlah;
```

- Kelas ini memiliki atribut-atribut yang bersifat privat (hanya dapat diakses di dalam kelas itu sendiri), yang merepresentasikan nama barang (namabarang), kategori (kategori), harga (harga), dan jumlah stok (jumlah).
- Atribut-atribut ini diinisialisasi saat objek dibuat dan bersifat final, yang berarti nilainya tidak dapat diubah setelah objek dibuat.

```
9 public databarangbaru(String namabarang, String kategori, int harga, int jumlah){  
10     this.namabarang = namabarang;  
11     this.kategori = kategori;  
12     this.harga = harga;  
13     this.jumlah = jumlah;  
14 }
```

Konstruktor kelas ini digunakan untuk membuat objek databarangbaru. Saat objek dibuat, nilai dari atribut-atribut (nama barang, kategori, harga, jumlah) ditentukan.

```
16 public String getNamaBarang() {  
17     return namabarang;  
18 }  
19  
20 public String getKategori(){  
21     return kategori;  
22 }  
23  
24 public int getHarga(){  
25     return harga;  
26 }  
27  
28 public int getJumlah(){  
29     return jumlah;  
30 }
```



Metode-metode get digunakan untuk mengakses nilai atribut-atribut ini dari luar kelas. Ini merupakan prinsip enkapsulasi dalam pemrograman, di mana atribut-atribut bersifat privat dan diakses melalui metode-metode publik.

```
32     @Override
33     public String toString(){
34         return "Nama Barang: " + namabarang + "Kategori Barang: " + kategori + ", Harga : " + harga + ", Jumlah Stok: " + jumlah;
35     }
36 }
```

- Metode toString dioverride untuk menghasilkan representasi dalam bentuk string dari objek databarangbaru.
- Ketika objek ini dicetak menggunakan System.out.println, metode toString akan dipanggil dan mencetak detail barang (nama, kategori, harga, jumlah) dalam format tertentu.

### 3.1.3 Tampilan Output OOP Pada Java

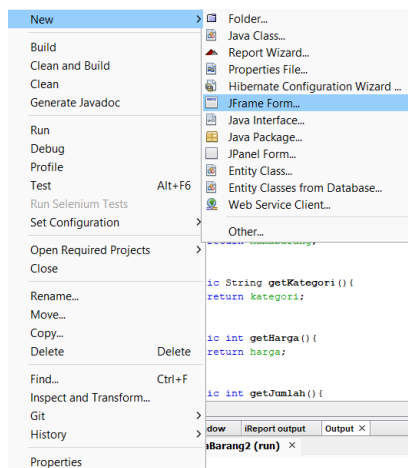
Ini adalah Tampilan Ouput pada saat file databarangbaru2.java dijalankan:

```
Output - Run (databarang)
--- exec:3.1.0:exec (default-cli) @ databarang ---
Tambah Data Barang Baru (y/n):
y
Nama Produk: Biskuit Rasa Coklat
Kategori Produk: Makanan
Harga Produk: 15000
Jumlah Stok: 50
Tambah Data Barang Baru (y/n):
y
Nama Produk: Shampoo Anti Ketombe
Kategori Produk: Kesehatan Dan Kecantikan
Harga Produk: 25000
Jumlah Stok: 30
Tambah Data Barang Baru (y/n):
y
Nama Produk: Baju T-Shirt Polos
Kategori Produk: Fashion
Harga Produk: 35000
Jumlah Stok: 40
Tambah Data Barang Baru (y/n):
n
Daftar Barang:
Nama Barang: Biskuit Rasa CoklatKategori Barang: Makanan, Harga : 15000, Jumlah Stok: 50
Nama Barang: Shampoo Anti KetombeKategori Barang: Kesehatan Dan Kecantikan, Harga : 25000, Jumlah Stok: 30
Nama Barang: Baju T-Shirt PolosKategori Barang: Fashion, Harga : 35000, Jumlah Stok: 40
-----
BUILD SUCCESS
-----
```

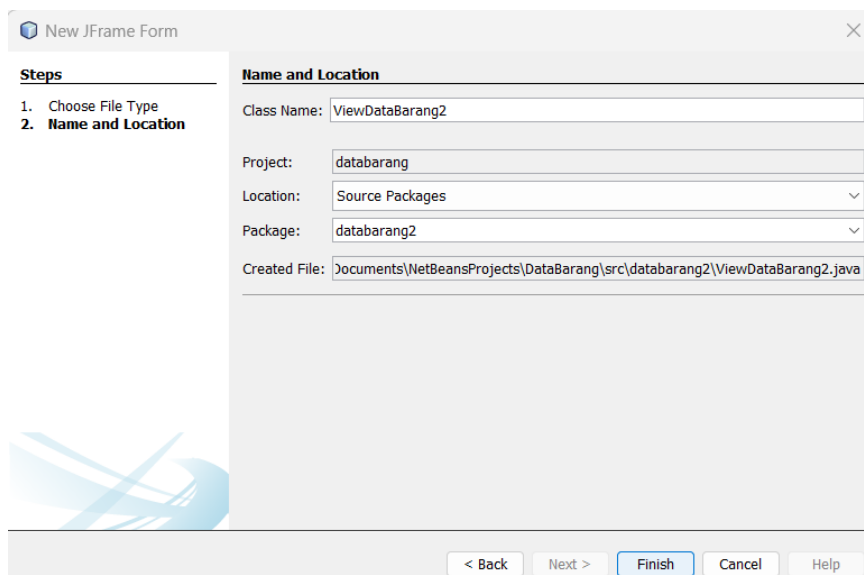
### 3.2 Membuat MVC Netbeans

Pada tahap kali ini, saya akan menjelaskan mengenai sebuah program dengan menggunakan pendekatan Model-View-Controller (MVC) dalam lingkungan pengembangan perangkat lunak menggunakan platform NetBeans. Program yang akan saya bahas adalah aplikasi manajemen data barang, yang dirancang untuk mencatat informasi seperti nama barang, kategori, harga, dan jumlah stok.

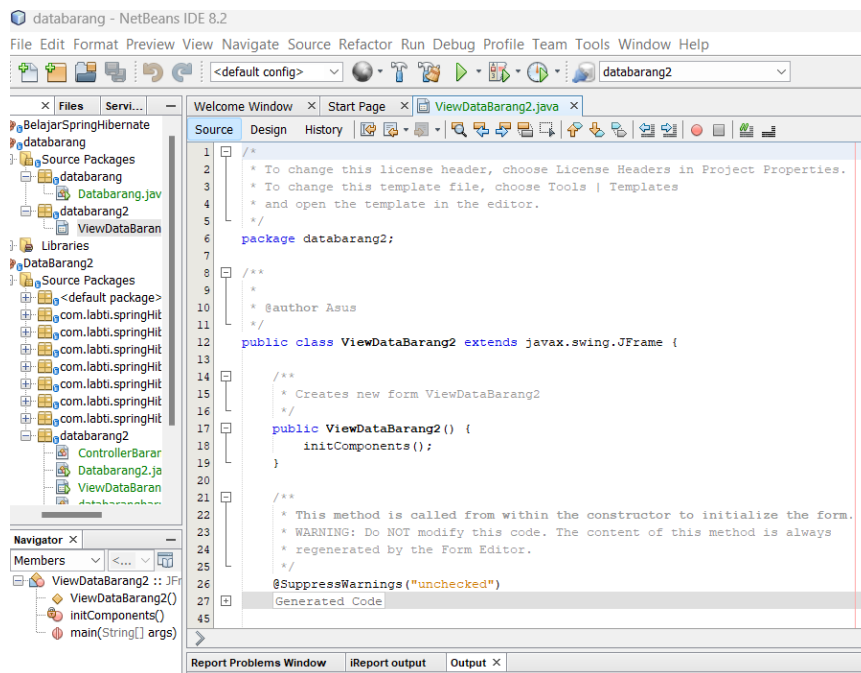
Untuk membuat sebuah model View baru silahkan klik kanan pada projek yang sudah dibuat, kemudian pilih JFrameForm.



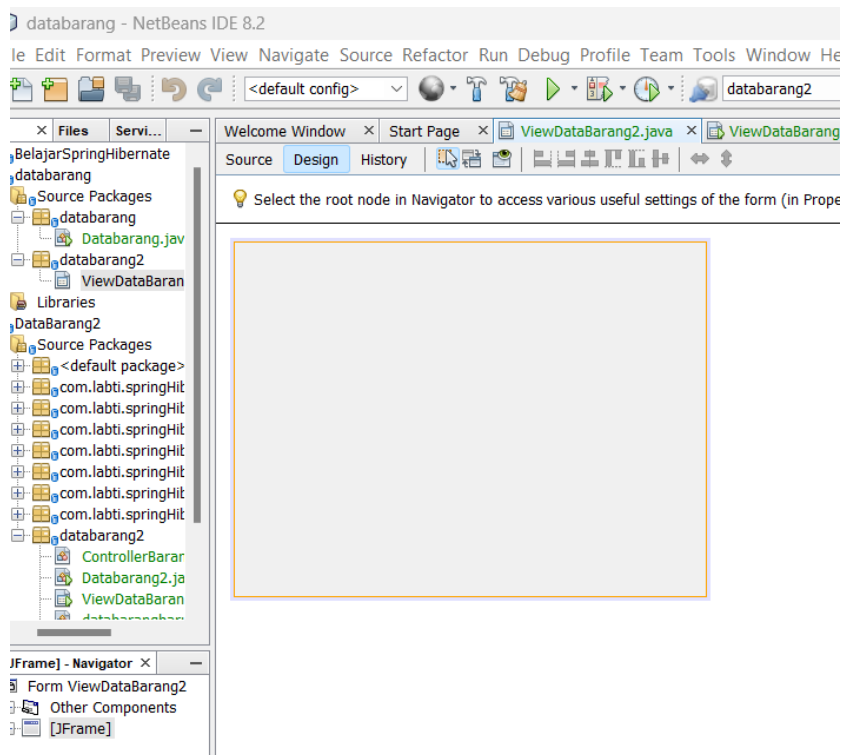
Kemudian buat nama Class Name baru dengan nama “**ViewDataBarang2**” didalam project databarang2



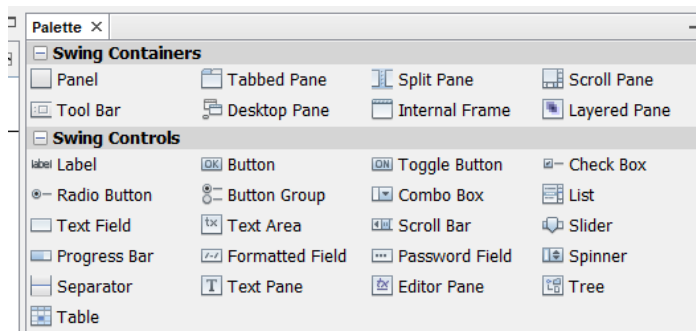
Selanjutnya klik “Finish”, maka file yang dibuat akan muncul dengan sendirinya.



Setelah itu, terdapat 3 tab yang menunjukkan bagian Source, Design, dan History. Kemudian pilih pada bagian design untuk memulai merancang sebuah tampilan interface pada MVC JFrame.



Di bagian kanan terdapat pallette yang berfungsi sebagai tools rancangan untuk memberikan fungsi dari tampilan yang akan dibuat:



Pada bagian ini, pilih label untuk memberikan text pada inputan dan judul. Kemudian menambahkan button sebagai fungsi saat kita melakukan sebuah aksi tertentu, Dan text Area untuk menambahkan sebuah deskripsi dari informasi barang tersebut. Yang terakhir yaitu tabel untuk menampilkan data pada baris dan kolom tertentu. Jika sudah di desain maka tampilan yang keluar seperti dibawah ini:

Berikan variabel pada Jtextfield disebelah bagian label dan menambahkan button dengan ketentuan:

Id Produk (JTextField)	id
Kategori Produk (JTextField)	kategori
Harga Produk (JTextField)	harga
Jumlah Produk (JTextField)	jumlah
Create (Button)	buat

Klik 2x pada button Create untuk kembali ke file ViewDataBarang2.java. kemudian pada bagian source kita menambahkan sebuah fungsi pada button tersebut dengan nama variabel “buatActionPerformed”. lakukan kodingan seperti dibawah ini:

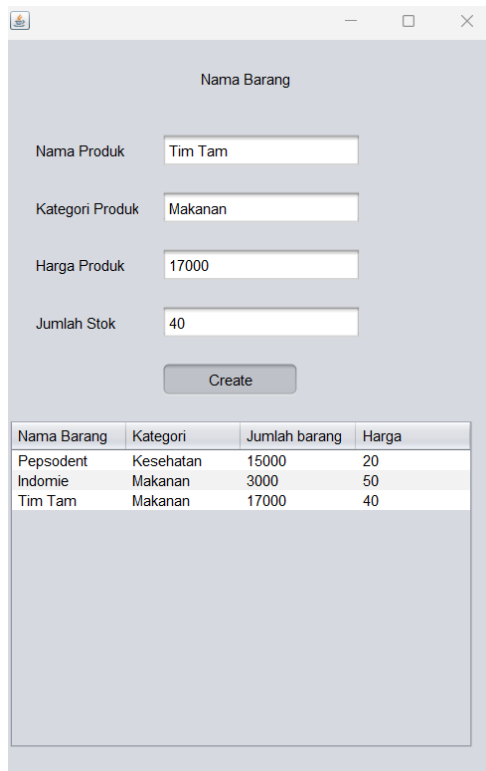
```
Start Page x ControllerBarang2.java x ViewDataBarang2.java x
Source Design History
1 package databarang2;
2
3 public class ViewDataBarang2 extends javax.swing.JFrame {
4     ControllerBarang2 mhs = new ControllerBarang2();
5
6
7     public ViewDataBarang2() {
8         initComponents();
9     }
10
11     @SuppressWarnings("unchecked")
12     Generated Code
120
121     private void buatActionPerformed(java.awt.event.ActionEvent evt) {
122         String namabarangku = namabarang.getText();
123         String kategoriku = kategori.getText();
124         int hargaku = Integer.parseInt(s: harga.getText());
125         int jumlahku = Integer.parseInt(s: jumlah.getText());
126
127         mhs.InsertData(namabarang: namabarangku, kategori: kategoriku, harga: hargaku, jumlah: jumlahku)
128         tablelist.setModel(dataModel:mhs.showData());
129     }
130
131     /**
132     * @param args the command line arguments
133     */
134     public static void main(String args[]) {
135         /* Set the Nimbus look and feel */
136         Look and feel setting code (optional)
137
138         /* Create and display the form */
139         java.awt.EventQueue.invokeLater(new Runnable() {
140             public void run() {
141                 new ViewDataBarang2().setVisible(true);
142             }
143         });
144     }
145 }
```

Berikut adalah Penjelasan dari logika program yang sudah dibuat:

- ViewDataBarang2 adalah JFrame Java Swing yang berfungsi sebagai antarmuka pengguna untuk melihat dan memasukkan data tentang produk.
- Kelas ini memiliki variabel instance yang mewakili berbagai komponen Swing seperti label, kolom teks, tombol toggle, dan tabel.
- Konstruktor ViewDataBarang2() menginisialisasi komponen antarmuka pengguna dengan memanggil metode initComponents().
- Metode initComponents() adalah kode yang dibuat secara otomatis yang mengatur tata letak dan properti dari komponen antarmuka pengguna.
- Metode buatActionPerformed adalah penanganan acara untuk tombol "Create" (buat). Metode ini mengambil data dari kolom teks (namabarang, kategori, harga, jumlah), mengonversi data numerik ke dalam bentuk integer, kemudian memanggil metode InsertData dari kelas ControllerBarang2 untuk memasukkan data ke dalam model data yang mendasarinya. Terakhir, metode ini memperbarui tabel dengan data yang baru saja dimasukkan dengan memanggil showData() pada pengontrol.
- Metode main mengatur tampilan antarmuka pengguna dan menciptakan instance dari ViewDataBarang2, sehingga frame dapat terlihat.

### 3.2.1 Tampilan Output Pada MVC Netbeans

Berikut adalah output dari program Program Pengisian data barang saat dieksekusi :



Nama Barang	Kategori	Jumlah barang	Harga
Pepsodent	Kesehatan	15000	20
Indomie	Makanan	3000	50
Tim Tam	Makanan	17000	40

### 3.3 Membuat Fungsi Button Update Dan Delete

Pada bagian ke-3 ini, saya akan menjelaskan mengenai Object Relational Mapping (ORM) pada program Java NetBeans untuk pengisian data barang. Selain atribut yang telah diberikan sebelumnya, yaitu "nama produk," "kategori produk," "harga produk," dan "jumlah produk," kali ini saya juga menambahkan fungsi delete dan update untuk memperkaya fungsionalitas program.

Kode program yang disediakan adalah implementasi sederhana dari aplikasi pengelolaan data barang menggunakan Java NetBeans. Aplikasi ini menggunakan konsep ORM untuk menyimpan dan mengambil data dari database.

Untuk membuat button fungsi update dan delete, buatlah design dibawah ini terlebih dahulu:

**Nama Barang**

Id Produk

Kategori Produk

Harga Produk

Jumlah Stok

Nama Produk	Kategori Pro...	Harga Produk	Jumlah Stok

Setelah menambahkan button update dan delete tersebut, kemudian tekan tombol button update. setelah diarahkan ke source, balik ke tab design, kemudian klik tombol delete 2x sehingga method tersebut akan men-generate dengan sendirinya. Kemudian kita buat fungsi button tersebut.

```

160
161 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
162     int selectedRow = tablelist.getSelectedRow();
163
164     if (selectedRow == -1) {
165         JOptionPane.showMessageDialog(parentComponent: this, message: "Pilih baris yang ingin dihapus.",
166                                     title: "Peringatan", messageType: JOptionPane.WARNING_MESSAGE);
167     } else {
168         DefaultTableModel model = (DefaultTableModel) tablelist.getModel();
169         model.removeRow(row: selectedRow);
170     }
171 }
172
173 private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
174     int selectedRow = tablelist.getSelectedRow();
175
176     // Check if a row is selected
177     if (selectedRow == -1) {
178         JOptionPane.showMessageDialog(parentComponent: this, message: "Select a row to edit.",
179                                     title: "Warning", messageType: JOptionPane.WARNING_MESSAGE);
180     } else {
181         // Get the model and data from the selected row
182         DefaultTableModel model = (DefaultTableModel) tablelist.getModel();
183         String namabarangu = namabarang.getText();
184         String kategoriku = kategori.getText();
185         int hargaku = Integer.parseInt(s: harga.getText());
186         int jumlahku = Integer.parseInt(s: jumlah.getText());
187
188         // Update the values in the table model
189         model.setValueAt(sValue: namabarangu, row: selectedRow, column: 0);
190         model.setValueAt(sValue: kategoriku, row: selectedRow, column: 1);
191         model.setValueAt(sValue: hargaku, row: selectedRow, column: 2);
192         model.setValueAt(sValue: jumlahku, row: selectedRow, column: 3);
193     }
194 }
195
  
```

Berikut adalah ringkasan fungsi utama dalam kode program:

```
162 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    int selectedRow = tablelist.getSelectedRow();
```

Blok program pertama (**jButton2ActionPerformed**) adalah untuk menghapus baris yang dipilih dari tabel. Pertama, program mendapatkan indeks baris yang dipilih dengan menggunakan `tablelist.getSelectedRow()`.

```
164     if (selectedRow == -1){  
165         JOptionPane.showMessageDialog(parentComponent: this, message: "Pilih baris yang ingin dihapus.",  
166             title: "Peringatan", messageType: JOptionPane.WARNING_MESSAGE);
```

Kemudian, program memeriksa apakah ada baris yang dipilih atau tidak dengan menggunakan `if (selectedRow == -1)`. Jika tidak ada baris yang dipilih, program akan menampilkan pesan peringatan dengan menggunakan `JOptionPane.showMessageDialog`.

```
167     } else {  
168         DefaultTableModel model = (DefaultTableModel) tablelist.getModel();  
169         model.removeRow(row: selectedRow);  
170     }  
171 }
```

Jika ada baris yang dipilih, program akan mendapatkan model tabel dengan menggunakan `(DefaultTableModel) tablelist.getModel()` dan menghapus baris tersebut dengan menggunakan `model.removeRow(selectedRow)`.

```
162 private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    int selectedRow = tablelist.getSelectedRow();
```

Blok program kedua (**jButton1ActionPerformed**) adalah untuk mengedit baris yang dipilih dari tabel. Pertama, program mendapatkan indeks baris yang dipilih dengan menggunakan `tablelist.getSelectedRow()`.

```
164     if (selectedRow == -1){  
165         JOptionPane.showMessageDialog(parentComponent: this, message: "Pilih baris yang ingin dihapus.",  
166             title: "Perinqatan", messageType: JOptionPane.WARNING_MESSAGE);
```

Kemudian, program memeriksa apakah ada baris yang dipilih atau tidak dengan menggunakan `if (selectedRow == -1)`. Jika tidak ada baris yang dipilih, program akan menampilkan pesan peringatan dengan menggunakan `JOptionPane.showMessageDialog`.



```

180 } else {
181     // Get the model and data from the selected row
182     DefaultTableModel model = (DefaultTableModel) tablelist.getModel();
183     String namabarangku = namabarang.getText();
184     String kategoriku = kategori.getText();
185     int hargaku = Integer.parseInt(s: harga.getText());
186     int jumlahku = Integer.parseInt(s: jumlah.getText());

```

Jika ada baris yang dipilih, program akan mendapatkan model tabel dengan menggunakan `(DefaultTableModel) tablelist.getModel()` dan data dari baris tersebut. Selanjutnya, program akan mendapatkan data baru dari input pengguna dengan menggunakan `namabarang.getText()`, `kategoriku.getText()`, `Integer.parseInt(harga.getText())`, dan `Integer.parseInt(jumlah.getText())`.

```

189     model.setValueAt(aValue:namabarangku, row:selectedRow, column:0);
190     model.setValueAt(aValue:kategoriku, row:selectedRow, column:1);
191     model.setValueAt(aValue:hargaku, row:selectedRow, column:2);
192     model.setValueAt(aValue:jumlahku, row:selectedRow, column:3);
193 }
194 }

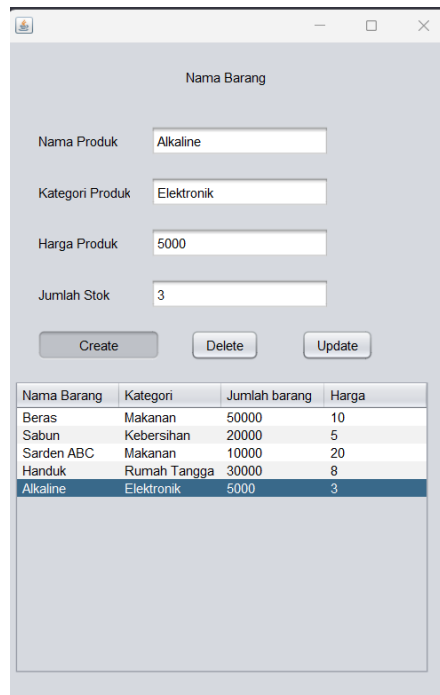
```

Terakhir, program akan memperbarui nilai-nilai di model tabel dengan menggunakan `model.setValueAt` untuk setiap kolom.

### 3.3.1 Tampilan Output Pada Fungsi Button Update Dan Delete

Berikut adalah output dari program diatas:

Record data barang setelah di create 5 data:



Nama Barang

Nama Produk: Alkaline

Kategori Produk: Elektronik

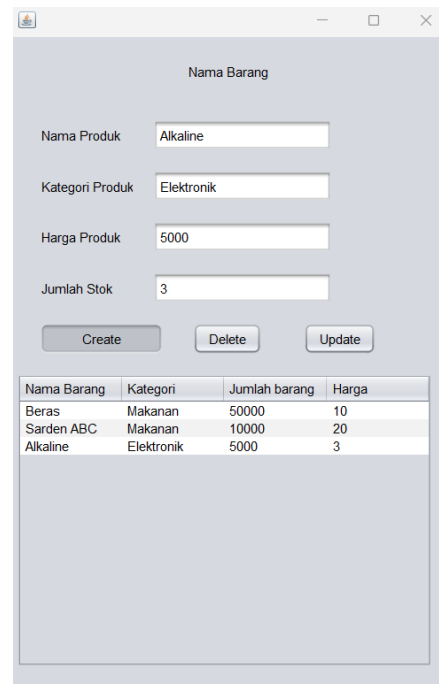
Harga Produk: 5000

Jumlah Stok: 3

Create Delete Update

Nama Barang	Kategori	Jumlah barang	Harga
Beras	Makanan	50000	10
Sabun	Kebersihan	20000	5
Sarden ABC	Makanan	10000	20
Handuk	Rumah Tangga	30000	8
Alkaline	Elektronik	5000	3

Data Handuk dan sabun Ketika Dihapus:



Nama Barang

Nama Produk: Alkaline

Kategori Produk: Elektronik

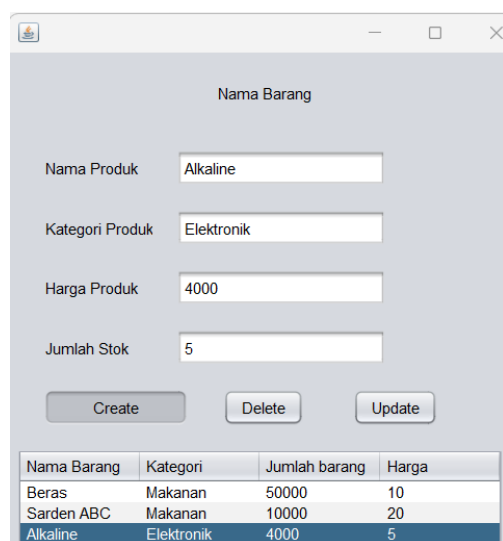
Harga Produk: 5000

Jumlah Stok: 3

Create Delete Update

Nama Barang	Kategori	Jumlah barang	Harga
Beras	Makanan	50000	10
Sarden ABC	Makanan	10000	20
Alkaline	Elektronik	5000	3

Data Produk Alkaline Ketika diubah Harga produk dan jumlah stok :



Nama Barang

Nama Produk: Alkaline

Kategori Produk: Elektronik

Harga Produk: 4000

Jumlah Stok: 5

Create Delete Update

Nama Barang	Kategori	Jumlah barang	Harga
Beras	Makanan	50000	10
Sarden ABC	Makanan	10000	20
Alkaline	Elektronik	4000	5

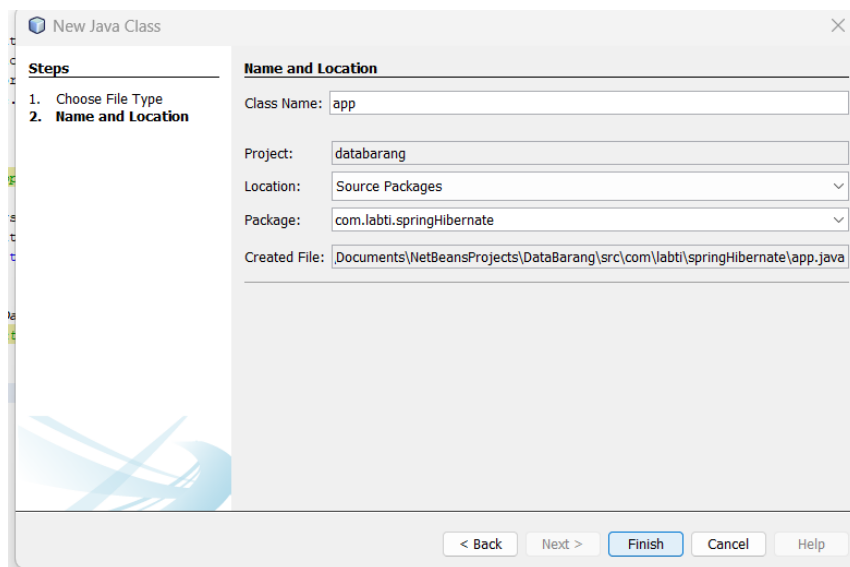
### 3.4 Mengimplementasikan Spring dan Hibernate Pada Java Netbeans

Pada praktikum kali ini, saya akan menjelaskan tentang pembuatan program Java menggunakan NetBeans dengan menggunakan kerangka kerja Spring dan Hibernate. Program ini bertujuan untuk mengelola data barang, dan kita akan melihat bagaimana Spring dan Hibernate dapat digunakan bersama-sama untuk menyederhanakan pengembangan aplikasi Java.

Berikut adalah langkah-langkah dalam membuat projek spring dan hibernate:

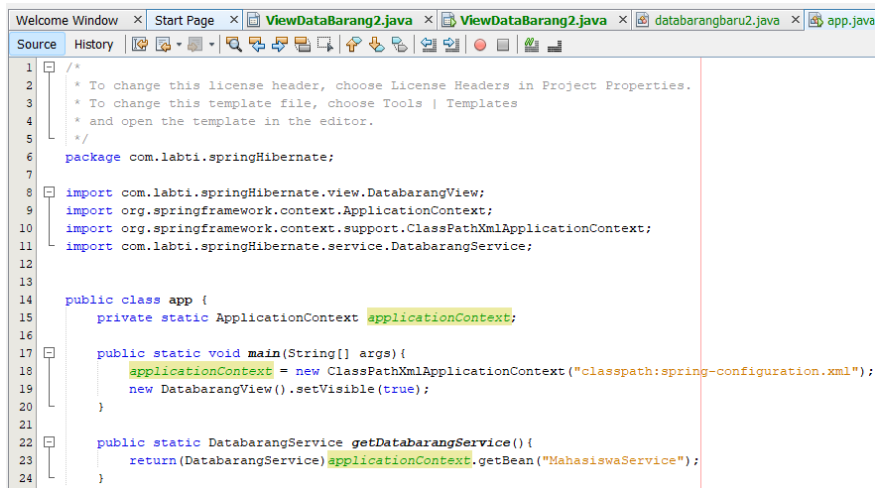
#### 3.4.1 Membuat app.java

Untuk membuat projek Spring dan Hibernate, pertama-tama kita akan membuat Java Class terlebih dahulu.



Jika sudah maka Klik Finish.

Pada Class App.java kodingkan kodingan seperti dibawah ini:



```

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 com.labti.springHibernate;
7
8  import com.labti.springHibernate.view.DatabarangView;
9  import org.springframework.context.ApplicationContext;
10 import org.springframework.context.support.ClassPathXmlApplicationContext;
11 import com.labti.springHibernate.service.DatabarangService;
12
13
14 public class app {
15     private static ApplicationContext applicationContext;
16
17     public static void main(String[] args) {
18         applicationContext = new ClassPathXmlApplicationContext("classpath:spring-configuration.xml");
19         new DatabarangView().setVisible(true);
20     }
21
22     public static DatabarangService getDatabarangService() {
23         return (DatabarangService) applicationContext.getBean("MahasiswaService");
24     }
25 }

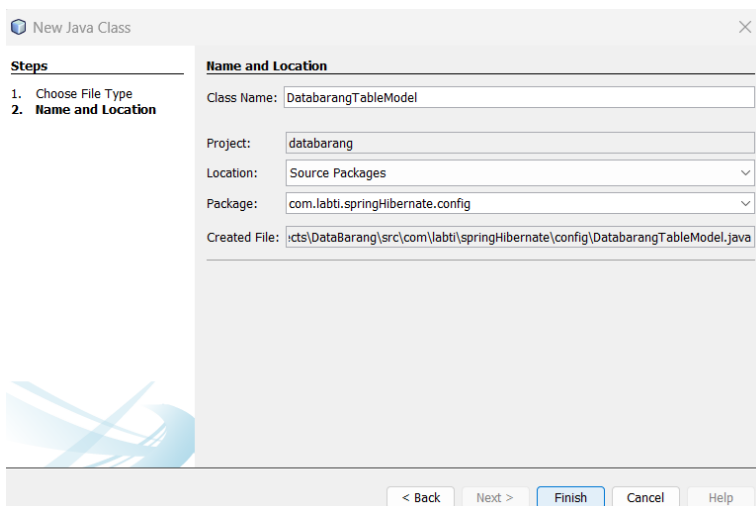
```

Berikut adalah penjelasan dari logika Listing diatas:

Kode di atas adalah bagian dari aplikasi Java menggunakan Spring dan Hibernate untuk manajemen data barang. Kode tersebut memulai dengan komentar lisensi dan pengaturan proyek, diikuti deklarasi paket dan impor kelas yang diperlukan. Kelas utama, **app**, memiliki metode **main** yang inialisasi konteks Spring dari file konfigurasi XML dan menampilkan objek **DatabarangView**. Terdapat juga metode **getDatabarangService** untuk mendapatkan bean **DatabarangService**, meskipun terdapat potensi kesalahan penamaan bean ("MahasiswaService" seharusnya "DatabarangService"). Secara keseluruhan, kode ini membentuk kerangka dasar aplikasi manajemen data barang dengan Spring dan Hibernate.

### 3.4.2 Membuat DatabarangTableModel.java

Selanjutnya membuat java class DatabarangTableModel.java. klik kanan projek kemudian new -> klik java application berada didalam package com.labti.springHibernate.model.config.



Setelah membuat class `DatabarangTableModel.java`, selanjutnya membuat kodingan dibawah:

```
1 package com.labti.springHibernate.config;
2
3 import com.labti.springHibernate.model.Databarang;
4 import java.util.ArrayList;
5 import java.util.List;
6 import javax.swing.table.AbstractTableModel;
7
8 public class DatabarangTableModel extends AbstractTableModel {
9
10     private List<Databarang> databarangs = new ArrayList<>();
11     private final String HEADER[] = {"ID", "Nama", "HARGA", "DESKRIPSI"};
12
13     public DatabarangTableModel(List<Databarang> mahasiswa) {
14         this.databarangs = mahasiswa;
15     }
16
17     @Override
18     public int getRowCount() {
19         return databarangs.size();
20     }
21
22     @Override
23     public int getColumnCount() {
24         return HEADER.length;
25     }
26
27     @Override
28     public String getColumnName(int columnIndex) {
29         return HEADER[columnIndex];
30     }
31
32     @Override
33     public Object getValueAt(int rowIndex, int columnIndex) {
34         Databarang databarang = databarangs.get(rowIndex);
35         switch (columnIndex) {
36             case 0:
37                 return databarang.getId();
38             case 1:
39                 return databarang.getNama();
40             case 2:
41                 return databarang.getHarga();
42             case 3:
43                 return databarang.getDeskripsi();
44         }
45     }
46 }
47
48
```

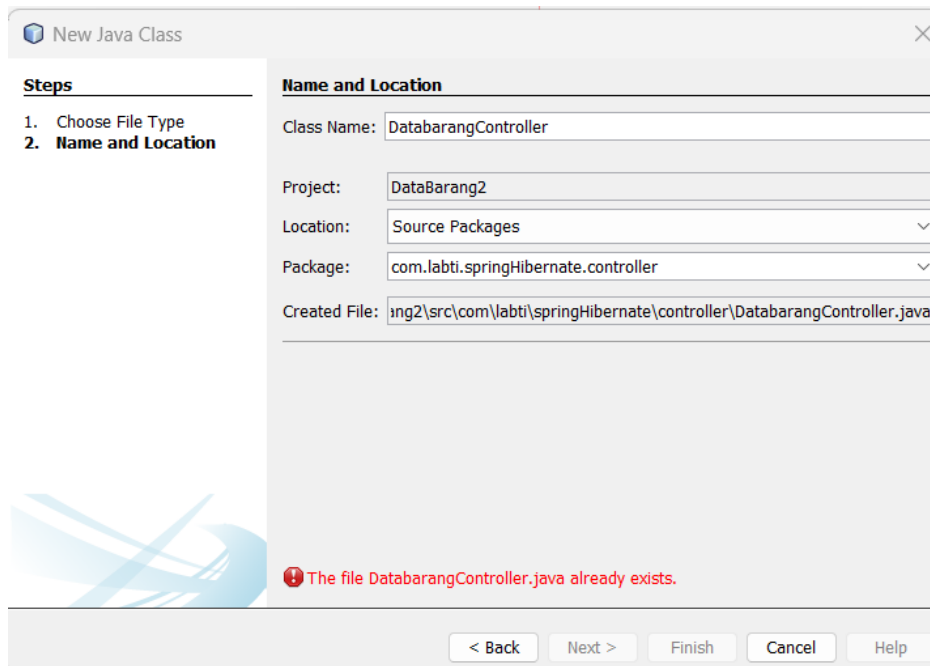
```
31
32
33 @Override
34 public Object getValueAt(int rowIndex, int columnIndex) {
35     Databarang databarang = databarangs.get(rowIndex);
36     switch (columnIndex) {
37         case 0:
38             return databarang.getId();
39         case 1:
40             return databarang.getNama();
41         case 2:
42             return databarang.getHarga();
43         case 3:
44             return databarang.getDeskripsi();
45         default:
46             return null;
47     }
48 }
```

Berikut adalah penjelasan dari logika Listing diatas:

Kode di atas adalah implementasi kelas **DatabarangTableModel** untuk model data `JTable` dalam aplikasi Java Swing. Kelas ini menerima daftar objek **Databarang** melalui konstruktor dan mengimplementasikan metode-metode abstrak dari kelas **AbstractTableModel**. Dengan menggunakan kelas ini, integrasi data objek **Databarang** ke dalam antarmuka pengguna Swing, terutama `Jtable`, dapat dilakukan dengan lebih efisien dan mudah.

### 3.4.3 Membuat DatabarangController.java

Sekarang kita akan membuat sebuah DatabarangController.java:



Berikut adalah kodingan dari class java diatas:

```
Source History
1 package com.labti.springHibernate.controller;
2
3 import com.labti.springHibernate.app;
4 import com.labti.springHibernate.config.DatabarangTableModel;
5 import com.labti.springHibernate.model.Databarang;
6 import com.labti.springHibernate.view.DatabarangView;
7 import java.util.List;
8 import javax.swing.JOptionPane;
9
10 public class DatabarangController {
11     private final DatabarangView databarangView;
12     private DatabarangTableModel databarangTableModel;
13     private List<Databarang> databarangs;
14
15     public DatabarangController (DatabarangView databarangView) {
16         this.databarangView = databarangView;
17     }
18
19     public void tampilData() {
20         databarangs = app.getDatabarangService().getMahasiswas();
21         databarangTableModel = new DatabarangTableModel(databarangs);
22         this.databarangView.getTabel().setModel(databarangTableModel);
23     }
24
25     public void show() {
26         int index = this.databarangView.getTabel().getSelectedRow();
27         this.databarangView.getNpm().setText(String.valueOf(
28             this.databarangView.getTabel().getValueAt(index, 0)));
29         this.databarangView.getNama().setText(String.valueOf(
30             this.databarangView.getTabel().getValueAt(index, 1)));
31         this.databarangView.getHarga().setText(String.valueOf(
32             this.databarangView.getTabel().getValueAt(index, 2)));
33         this.databarangView.getDeskripsi().setText(String.valueOf(
34             this.databarangView.getTabel().getValueAt(index, 3)));
35     }
36
37     public void clear() {
38         this.databarangView.getNpm().setText("");
39         this.databarangView.getNama().setText("");
40         this.databarangView.getHarga().setText("");
41         this.databarangView.getDeskripsi().setText("");
42     }
43 }
```

```

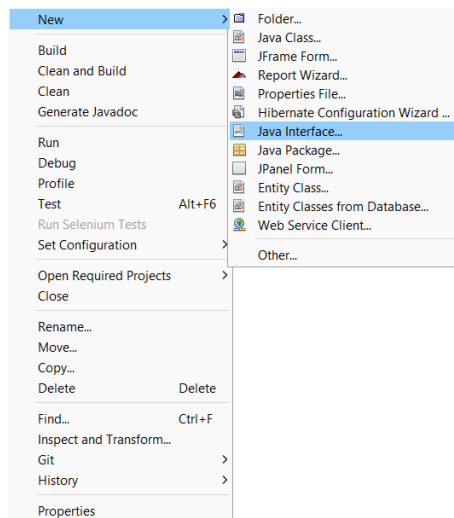
44 public void saveDatabarang() {
45     Databarang databarang = new Databarang();
46     databarang.setId(this.databarangView.getNpm().getText());
47     databarang.setName(this.databarangView.getName().getText());
48     databarang.setHarga(this.databarangView.getHarga().getText());
49     databarang.setDeskripsi(this.databarangView.getDeskripsi().getText());
50     app.getDatabarangService().save(databarang);
51     JOptionPane.showMessageDialog(null, "Data Berhasil di simpan", "info",
52         JOptionPane.INFORMATION_MESSAGE);
53     clear();
54     tampilData();
55 }
56
57 public void updateDatabarang() {
58     Databarang databarang = new Databarang();
59     databarang.setId(this.databarangView.getNpm().getText());
60     databarang.setName(this.databarangView.getName().getText());
61     databarang.setHarga(this.databarangView.getHarga().getText());
62     databarang.setDeskripsi(this.databarangView.getDeskripsi().getText());
63     app.getDatabarangService().update(databarang);
64     JOptionPane.showMessageDialog(null, "Data berhasil di Edit", "info",
65         JOptionPane.INFORMATION_MESSAGE);
66     clear();
67     tampilData();
68 }
69
70 public void deleteDatabarang() {
71     if(this.databarangView.getNpm().getText() == null){
72         JOptionPane.showMessageDialog(null, "mahasiswa belum dipilih", "error", JOptionPane.ERROR_MESSAGE);
73     }else{
74         Databarang databarang = new Databarang();
75         databarang.setId(this.databarangView.getNpm().getText());
76         int pilih = JOptionPane.showConfirmDialog(null, "Apakah data ingin dihapus ?", "Warning", JOptionPane.YES_NO_OPTION,
77             JOptionPane.WARNING_MESSAGE);
78         if (pilih == JOptionPane.YES_OPTION){
79             app.getDatabarangService().delete(databarang);
80             JOptionPane.showMessageDialog(null, "Data Berhasil di Hapus", "info", JOptionPane.INFORMATION_MESSAGE);
81             clear();
82             tampilData();
83         }
84     }
85 }

```

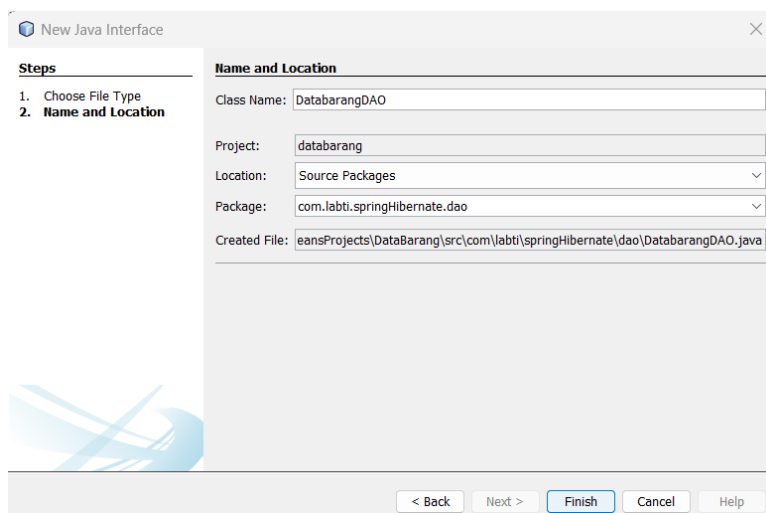
Kode di atas adalah implementasi kelas DatabarangController yang mengendalikan logika bisnis aplikasi manajemen data barang. Metode utamanya, seperti tampilData, show, clear, saveDatabarang, updateDatabarang, dan deleteDatabarang, menangani tampilan, interaksi pengguna, dan operasi CRUD terhadap objek Databarang menggunakan layanan DatabarangService. Dengan pendekatan ini, kelas ini mengikuti pola desain MVC, memudahkan struktur dan pemeliharaan aplikasi Java Swing dan Hibernate.

### 3.4.4 Membuat DatabarangDAO.java

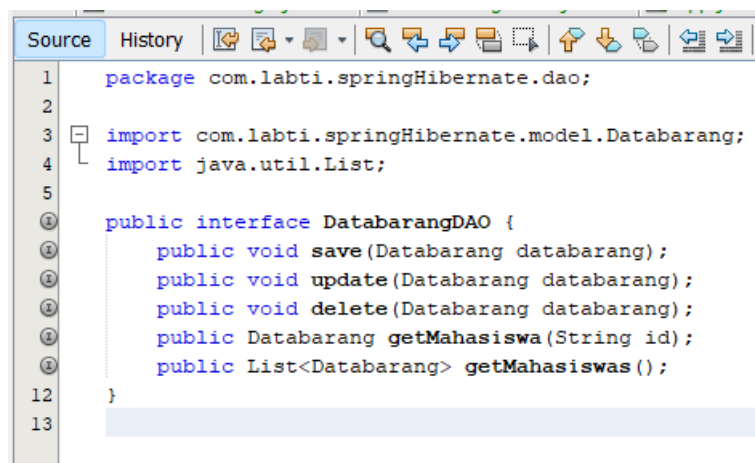
Untuk membuat DatabarangDAO.java, hal yang harus diperhatikan yaitu membuat Class java dalam bentuk interface, berikut adalah cara menambahkan file java class interface :



Klik kanan projek, kemudian New -> Java Interface. Setelah itu ganti nama menjadi DatabarangDAO.java.



Kemudian lakukan pengkodean program seperti dibawah ini:





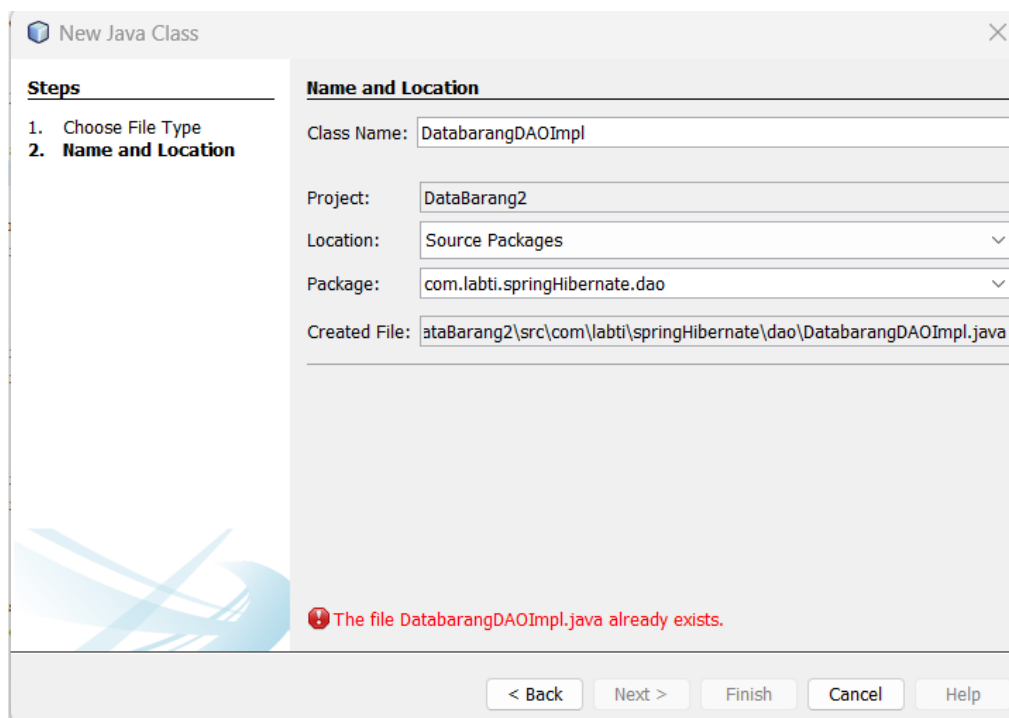
Berikut adalah logika dari program diatas:

Antarmuka **DatabarangDAO** menyediakan kontrak standar untuk operasi dasar pada objek **Databarang**, termasuk **save**, **update**, **delete**, serta metode untuk mengambil objek atau daftar objek. Dengan menyediakan abstraksi semacam ini, antarmuka memungkinkan fleksibilitas dalam implementasi sumber data atau penyimpanan data. Kesederhanaan kontrak ini mendukung modularitas dan kemudahan pengembangan dalam konteks aplikasi manajemen data barang menggunakan Spring dan Hibernate.

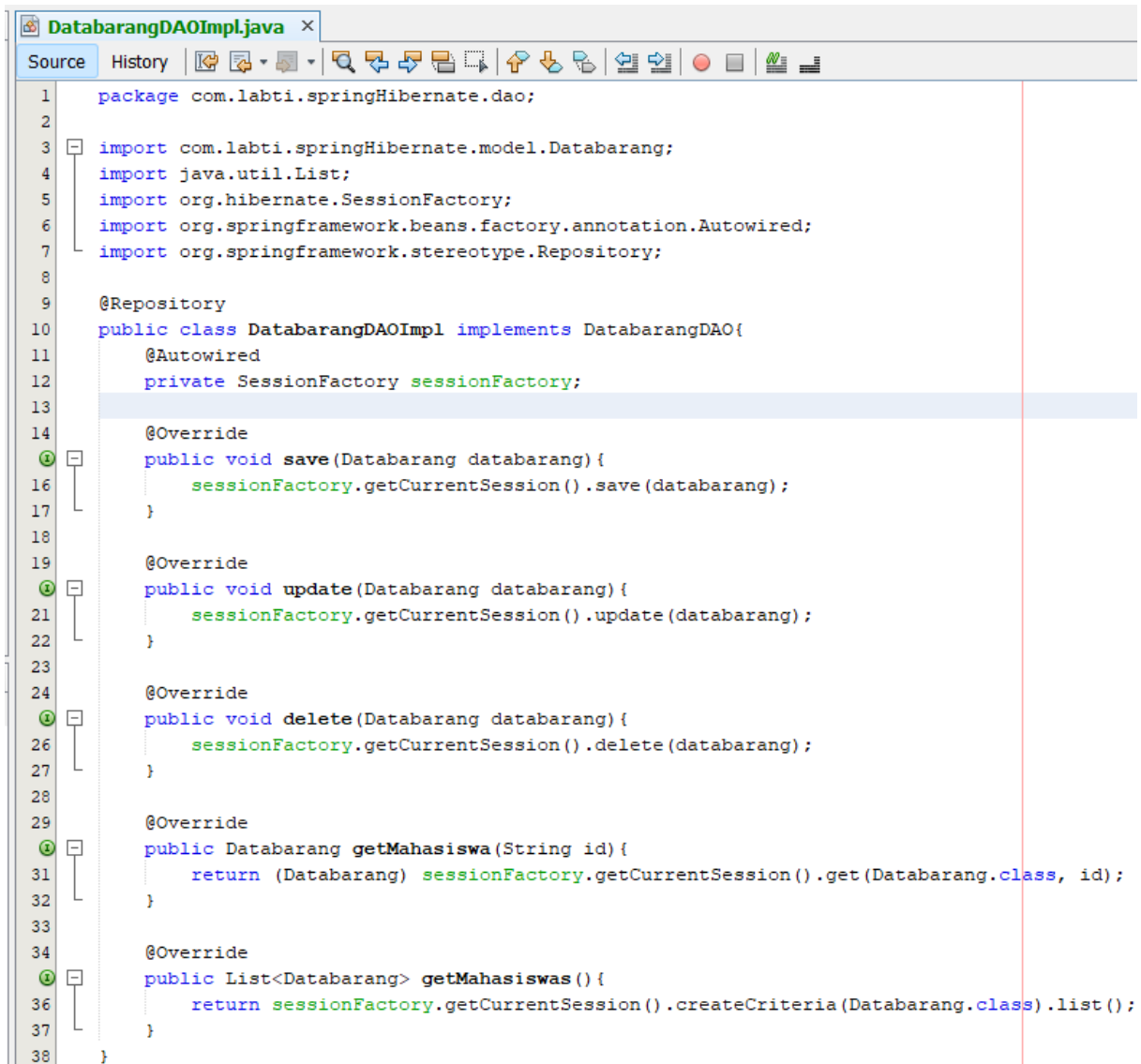
### 3.4.5 Membuat DatabarangDAOImpl

Selanjutnya membuat DatabarangDAOImpl:

Kita buat classnya terlebih dahulu dengan nama DatabarangDAOImpl di package com.labti.springHibernate.dao



Kemudian mendeklarasikan logika kodingan dibawah ini:

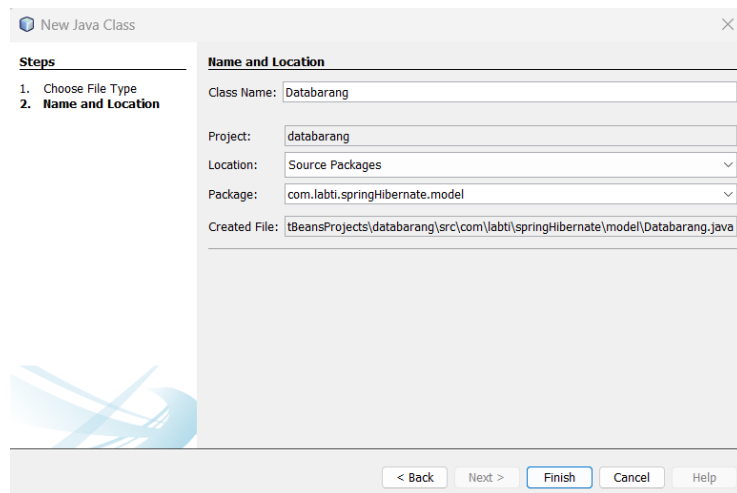


```
1 package com.labti.springHibernate.dao;
2
3 import com.labti.springHibernate.model.Databarang;
4 import java.util.List;
5 import org.hibernate.SessionFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Repository;
8
9 @Repository
10 public class DatabarangDAOImpl implements DatabarangDAO{
11     @Autowired
12     private SessionFactory sessionFactory;
13
14     @Override
15     public void save(Databarang databarang){
16         sessionFactory.getCurrentSession().save(databarang);
17     }
18
19     @Override
20     public void update(Databarang databarang){
21         sessionFactory.getCurrentSession().update(databarang);
22     }
23
24     @Override
25     public void delete(Databarang databarang){
26         sessionFactory.getCurrentSession().delete(databarang);
27     }
28
29     @Override
30     public Databarang getMahasiswa(String id){
31         return (Databarang) sessionFactory.getCurrentSession().get(Databarang.class, id);
32     }
33
34     @Override
35     public List<Databarang> getMahasiswas(){
36         return sessionFactory.getCurrentSession().createCriteria(Databarang.class).list();
37     }
38 }
```

Kode **DatabarangDAOImpl** adalah implementasi DAO untuk entitas **Databarang** dengan Hibernate dan Spring. Dengan anotasi **@Repository** dan **@Autowired**, kelas ini berperan sebagai repositori Spring yang menggunakan **SessionFactory**. Metode **save**, **update**, **delete**, **getMahasiswa**, dan **getMahasiswas** menyederhanakan operasi CRUD dan pengambilan data dari database. Sebagai implementasi konkret dari antarmuka **DatabarangDAO**, kelas ini mendukung efisiensi operasi database dalam aplikasi manajemen data barang.

### 3.4.6 Membuat Databarang.java

Selanjutnya kita akan membuat sebuah Java Class baru dengan nama **Databarang.java**, yang berada didalam package **com.labti.springHibernate.model**.



Berikut adalah kodingan dari Class Databarang.java:

```
1 package com.labti.springHibernate.model;
2
3 import java.io.Serializable;
4 import javax.persistence.Column;
5 import javax.persistence.Entity;
6 import javax.persistence.Id;
7 import javax.persistence.Table;
8
9 @Table(name = "tb_mahasiswa")
10 @Entity
11 public class Databarang {
12     @Id
13     @Column(name = "id", length = 8)
14     private String id;
15
16     @Column(name = "nama", length = 50)
17     private String nama;
18
19     @Column(name = "harga", length = 10)
20     private String harga;
21
22     @Column(name = "deskripsi", length = 150)
23     private String deskripsi;
24
25     public String getId(){
26         return id;
27     }
28     public void setId(String id){
29         this.id = id;
30     }
31
32     public String getNama(){
33         return nama;
34     }
35
36     public void setNama(String nama){
37         this.nama = nama;
38     }
39
40     public String getHarga(){
41         return harga;
42     }
43
44     public void setHarga(String harga){
```

```

45         this.harga = harga;
46     }
47
48     public String getDeskripsi(){
49         return deskripsi;
50     }
51     public void setDeskripsi(String deskripsi){
52         this.deskripsi = deskripsi;
53     }
54 }

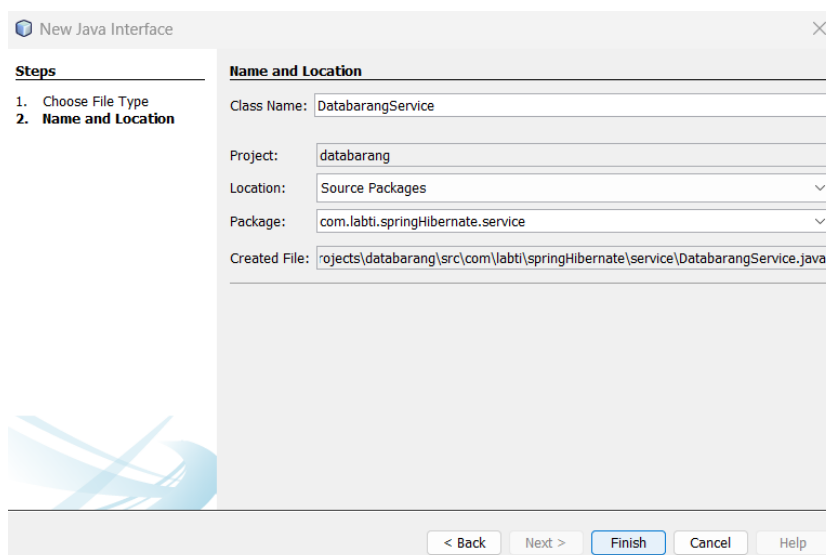
```

Berikut adalah penjelasan dari logika program diatas.

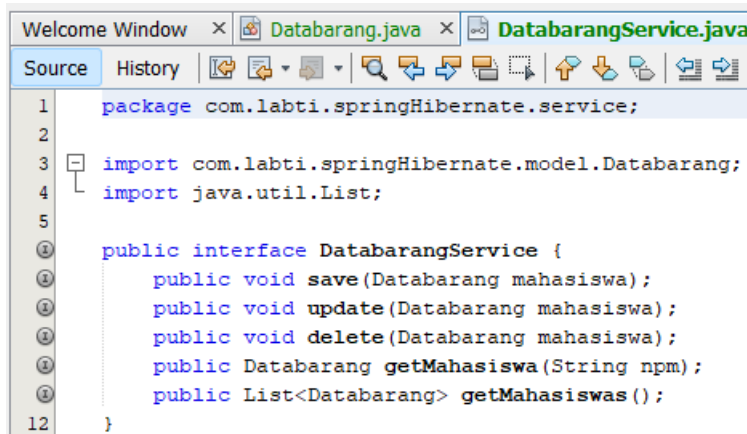
Kode di atas adalah definisi kelas **Databarang** sebagai model entitas untuk tabel "tb\_mahasiswa". Dengan anotasi JPA, atribut-atribut seperti **id**, **nama**, **harga**, dan **deskripsi** dikonfigurasi untuk melakukan pemetaan ke kolom-kolom yang sesuai dalam database. Metode getter dan setter yang sesuai memungkinkan akses dan pengaturan nilai atribut. Dengan struktur ini, kelas ini siap digunakan untuk berinteraksi dengan database dalam aplikasi manajemen data barang menggunakan Hibernate.

### 3.4.7 Membuat DatabarangService.java

Setelahnya kita akan membuat sebuah interface DatabarangService.java dengan klik kanan projek kemudian new java inteface. File java class ini akan memuat di package com.labti.springHibernate.service.



Membuatkan kodingan seperti yang pada dibawah ini:

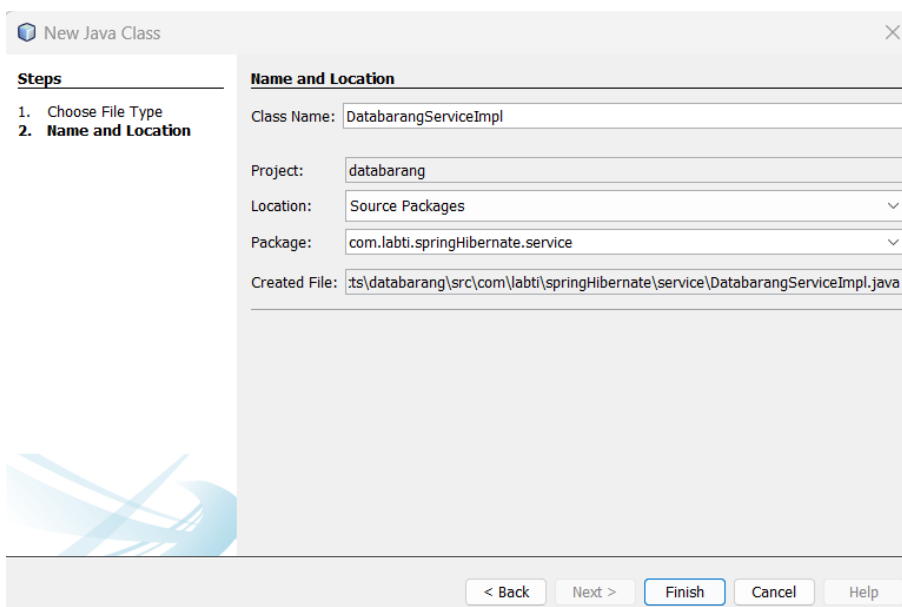


```
1 package com.labti.springHibernate.service;
2
3 import com.labti.springHibernate.model.Databarang;
4 import java.util.List;
5
6 public interface DatabarangService {
7     public void save(Databarang mahasiswa);
8     public void update(Databarang mahasiswa);
9     public void delete(Databarang mahasiswa);
10    public Databarang getMahasiswa(String npm);
11    public List<Databarang> getMahasiswas();
12 }
```

Antarmuka **DatabarangService** menentukan kontrak untuk layanan terhadap objek **Databarang**. Dengan metode seperti **save**, **update**, **delete**, **getMahasiswa**, dan **getMahasiswas**, antarmuka ini mencakup operasi dasar CRUD untuk penyimpanan, pembaruan, penghapusan, serta pengambilan data objek **Databarang** dari database. Dengan memberikan kontrak ini, antarmuka mendukung fleksibilitas dan abstraksi dalam implementasi layanan spesifik yang diperlukan dalam konteks aplikasi manajemen data barang.

### 3.4.8 Membuat DatabarangServiceImpl.java

Pada tahap ini kita akan membuat sebuah Java Class **DatabarangServiceImpl.java**.



Selanjutnya kita akan mengkodekan sebuah Class yang sudah dibuat tadi.

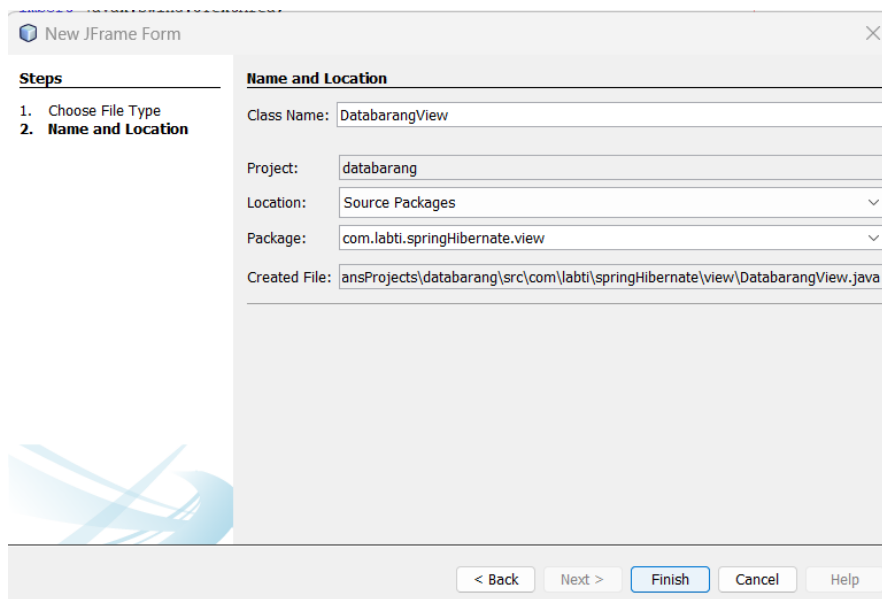
```
1 package com.labti.springHibernate.service;
2
3 import com.labti.springHibernate.model.Databarang;
4 import java.util.List;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7 import org.springframework.transaction.annotation.Transactional;
8 import com.labti.springHibernate.dao.DatabarangDAO;
9
10 @Service("MahasiswaService")
11 @Transactional(readOnly = true)
12
13 public class DatabarangServiceImpl implements DatabarangService {
14     @Autowired
15     private DatabarangDAO mahasiswaDao;
16
17     @Transactional
18     @Override
19     public void save(Databarang mahasiswa) {
20         mahasiswaDao.save(mahasiswa);
21     }
22
23     @Transactional
24     @Override
25     public void update(Databarang mahasiswa) {
26         mahasiswaDao.update(mahasiswa);
27     }
28
29     @Transactional
30     @Override
31     public void delete(Databarang mahasiswa) {
32         mahasiswaDao.delete(mahasiswa);
33     }
34
35     @Override
36     public Databarang getMahasiswa(String npm) {
37         return mahasiswaDao.getMahasiswa(npm);
38     }
39
40     @Override
41     public List<Databarang> getMahasiswas() {
42         return mahasiswaDao.getMahasiswas();
43     }
44 }
```

Kode di atas merupakan implementasi kelas **DatabarangServiceImpl** untuk layanan manajemen data barang. Dengan anotasi seperti **@Service**, **@Transactional**, dan **@Autowired**, kelas ini diidentifikasi sebagai komponen layanan Spring yang menggunakan antarmuka **DatabarangDAO** untuk operasi CRUD dan pengambilan data dari database. Dengan metode seperti **save**, **update**, **delete**, **getMahasiswa**, dan **getMahasiswas**, kelas ini menyederhanakan interaksi antara lapisan bisnis dan akses data dalam aplikasi manajemen data barang.

### 3.4.9 Membuat DatabarangView.java

Setelah kita membuat semua Program Class java yang sudah dibutuhkan, selanjutnya yaitu membuat JFrame **DatabarangView.java** dan dengan tampilan seperti dibawah ini:

Membuat sebuah JFrame untuk DatabarangView:



Sesudah membuat DatabarangView, menambahkan Listing yang sudah dibuat pada gambar dibawah :

```
Welcome Window x Databarang.java x DatabarangService.java x DatabarangServiceImpl.java x DatabarangView.java x
Source Design History
1 package com.labti.springHibernate.view;
2
3 import com.labti.springHibernate.controller.DatabarangController;
4 import javax.swing.JOptionPane;
5 import javax.swing.JTable;
6 import javax.swing.JTextArea;
7 import javax.swing.JTextField;
8 import net.sf.jasperreports.engine.JasperCompileManager;
9 import net.sf.jasperreports.engine.JasperFillManager;
10 import net.sf.jasperreports.engine.JasperPrint;
11 import net.sf.jasperreports.engine.JasperReport;
12 import net.sf.jasperreports.view.JasperViewer;
13
14 public class DatabarangView extends javax.swing.JFrame {
15     private final DatabarangController databarangController = new DatabarangController(this);
16
17     public DatabarangView() {
18         initComponents();
19         databarangController.tampilData();
20     }
21
22     @SuppressWarnings("unchecked")
23 }
```

```

186 private void hapusActionPerformed(java.awt.event.ActionEvent evt) {
187     databarangController.deleteDatabarang();
188 }
189
190 private void updateActionPerformed(java.awt.event.ActionEvent evt) {
191     databarangController.updateDatabarang();
192 }
193
194 private void tabelMouseClicked(java.awt.event.MouseEvent evt) {
195     databarangController.show();
196 }
197
198 private void idActionPerformed(java.awt.event.ActionEvent evt) {
199
200 }

211 public JTextArea getDeskripsi() {
212     return deskripsi;
213 }
214
215 public void setDeskripsi(JTextArea deskripsi) {
216     this.deskripsi = deskripsi;
217 }
218
219 public JTextField getHarga() {
220     return harga;
221 }
222
223 public void setHarga(JTextField harga) {
224     this.harga = harga;
225 }
226
227 public JTextField getNama() {
228     return nama;
229 }
230
231 public void setNama(JTextField Nama) {
232     this.nama = Nama;
233 }
234
235 public JTextField getNpm() {
236     return id;
237 }
238
239 public void setNpm(JTextField id) {
240     this.id = id;
241 }
242
243 public JTable getTabel() {
244     return tabel;
245 }
246
247 public void setTabel(JTable tabel) {
248     this.tabel = tabel;
249 }

250 // Variables declaration - do not modify
251 private javax.swing.JTextArea deskripsi;
252 private javax.swing.JButton hapus;
253 private javax.swing.JTextField harga;
254 private javax.swing.JTextField id;
255 private javax.swing.JLabel jLabel1;
256 private javax.swing.JLabel jLabel2;
257 private javax.swing.JLabel jLabel3;
258 private javax.swing.JLabel jLabel4;
259 private javax.swing.JLabel jLabel5;
260 private javax.swing.JScrollPane jScrollPane1;
261 private javax.swing.JScrollPane jScrollPane2;
262 private javax.swing.JTextField nama;
263 private javax.swing.JButton simpan;
264 private javax.swing.JTable tabel;
265 private javax.swing.JButton update;
266 // End of variables declaration
267 }

```

Berikut adalah penjelasan pada source di DatabarangView.java.

Kode di atas adalah implementasi antarmuka pengguna (UI) menggunakan JFrame untuk aplikasi manajemen data barang. Dalam kelas **DatabarangView**, elemen-elemen GUI seperti JTextField, JTextArea, JButton, dan JTable diinisialisasi melalui metode  **initComponents**. Metode ini dihasilkan oleh pembuat formulir GUI pada NetBeans.



Beberapa metode, seperti **simpanActionPerformed**, **hapusActionPerformed**, **updateActionPerformed**, dan **tabelMouseClicked**, memanggil metode dari objek **DatabarangController**, mencerminkan pola desain arsitektur MVC (Model-View-Controller). Kelas ini juga menangani tampilan laporan menggunakan JasperReports setelah operasi penyimpanan berhasil.

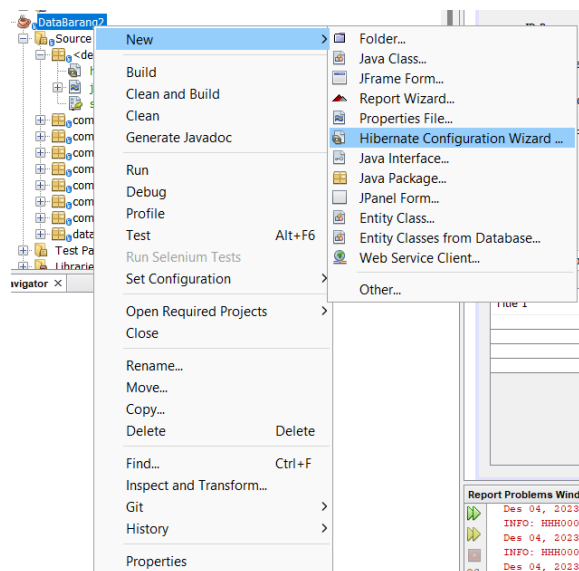
Metode getter dan setter digunakan untuk mendapatkan dan mengatur nilai elemen GUI, memungkinkan akses dan manipulasi nilai-nilai tersebut dari kelas lain. Keseluruhan struktur kelas ini menciptakan antarmuka pengguna responsif untuk berinteraksi dengan data barang dalam aplikasi menggunakan Java Swing.

### 3.4.10 Membuat Konfigurasi Hibernate, Spring dan JDBC

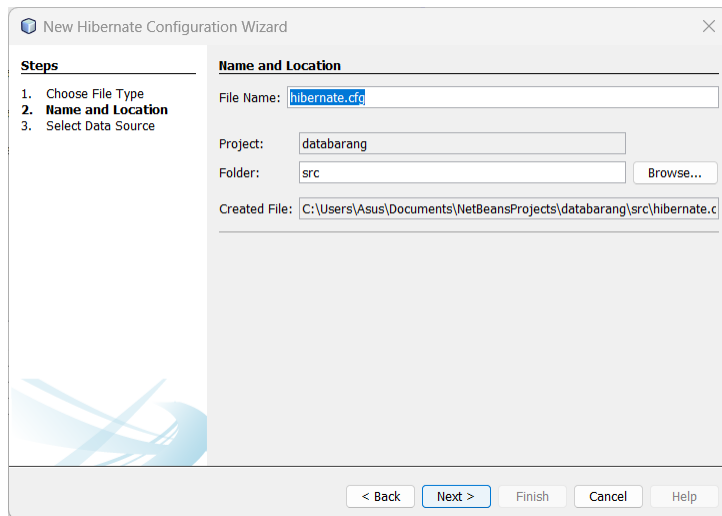
#### 3.4.10.1 Membuat Konfigurasi Hibernate

Setelah Membuat semua Java Class yang dibutuhkan, selanjutnya membuat sebuah Konfigurasi dari spring dan Hibernate.

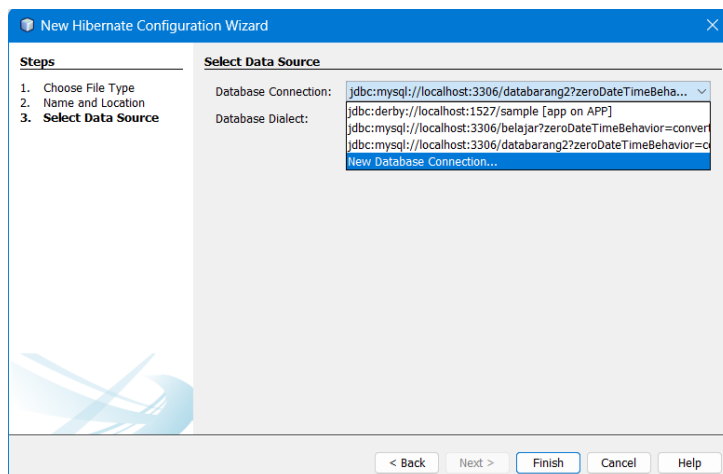
Berikut adalah Langkah-langkah untuk menambahkan Hibernate pada proyek yang sudah dibuat. Yang pertama dilakukan adalah klik kanan pada proyek yang sudah dibuat, kemudian klik New -> Hibernate Configuration Wizard.



Pada menu Ini, klik Next untuk melanjutkan.

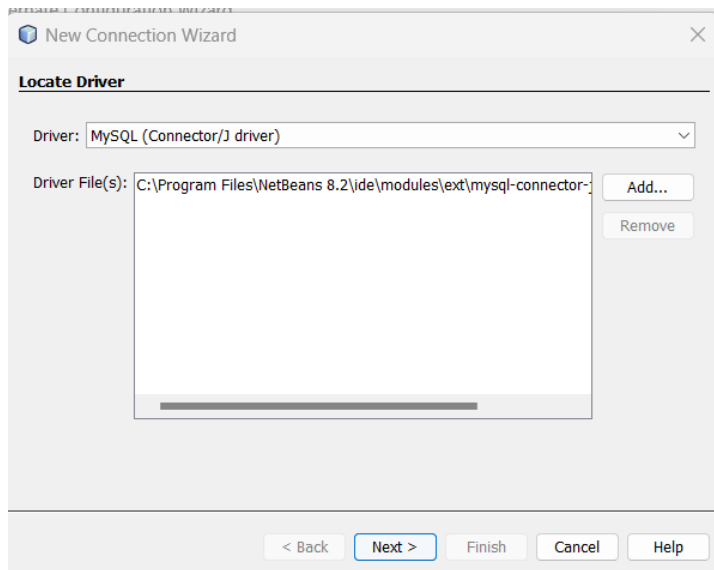


Setelah itu, Pada 'Database Connection' Tekan bagian panah ke bawah untuk mengganti database Connection yang ingin ditambahkan



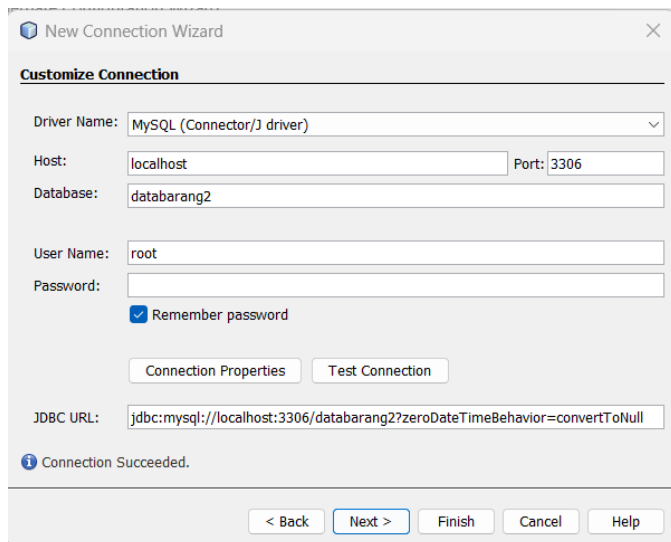
Pada bagian Locate Driver, pilih Driver "MySQL (Connector/J Driver), pastikan sudah menginstall File jar mysql-connector-jdbc. Jika belum silahkan download di Link <https://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.23>. Pada bagian File tersebut klik .jar untuk mendownload file tersebut.



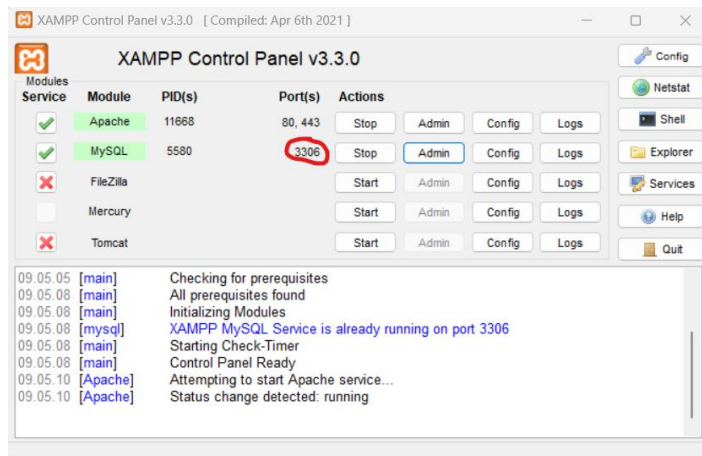


Jika sudah menambahkan File tersebut Silahkan Klik Next.

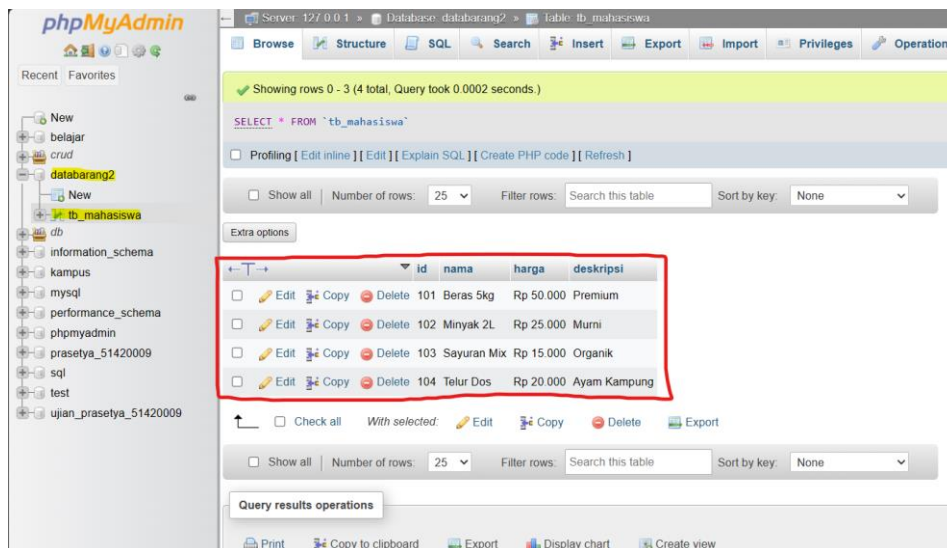
Pada bagian Customize Connection. Lakukan Setting seperti dibawah ini. Ganti Database dengan nama “databarang2” dan centang Remember password. Setelah itu lakukan Test Connection. Jika berhasil maka dapat terkoneksi ke Database databarang2. jika gagal maka periksa apakah di MySQL tersebut sudah membuat Database terlebih dahulu atau belum. dan pastikan juga Portnya sama dan berjalan di Xampp.



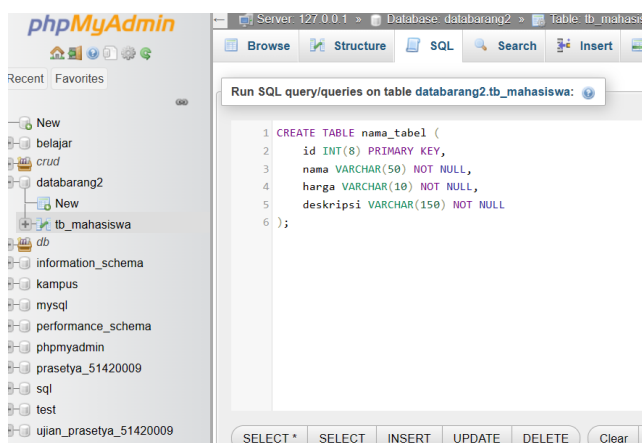
Ini adalah bagian Port yang Berjalan pada Xampp. Untuk mengecek database MySQL silahkan Klik pada bagian Admin.



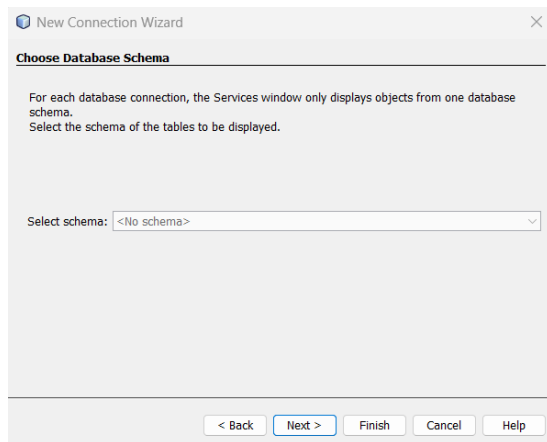
Buatlah Database dengan nama databarang2,



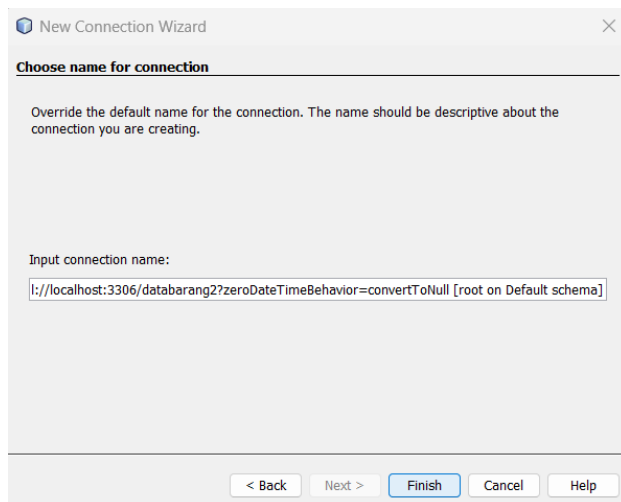
Setelah membuat database, selanjutnya membuat tabel dengan nama tb\_mahasiswa. Tambahkan Query pada tab SQL dengan ketentuan sebagai berikut.



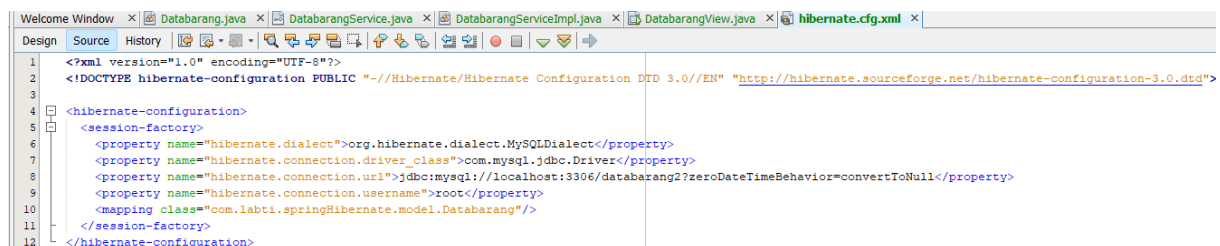
Setelah Test Connection berhasil, Klik Next. Kemudian pada Choose Database Schema klik Next lagi.



Selanjutnya klik Finish untuk menyelesaikan setting Configuration.



Lakukan Kodingan Listing pada Hibernate.cfg seperti yang tertera pada gambar dibawah ini.



File XML di atas adalah konfigurasi Hibernate yang mengatur koneksi ke database MySQL dan pemetaan objek-relasional. Elemen **<hibernate-configuration>** menandakan awal konfigurasi Hibernate, dan **<session-factory>** mendefinisikan konfigurasi sesi Hibernate, termasuk properti seperti dialek Hibernate, kelas driver JDBC, URL koneksi, serta kredensial untuk database.

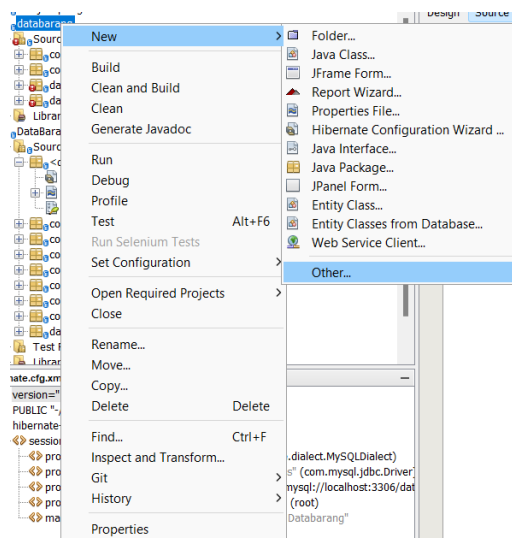
Selain itu, elemen `<mapping class="com.labti.springHibernate.model.Databarang"/>` menyatakan pemetaan objek Java kelas **Databarang** ke struktur tabel di database.

File ini memberikan informasi penting yang diperlukan untuk menghubungkan aplikasi Spring Hibernate ke database MySQL dan untuk memetakan objek **Databarang**.

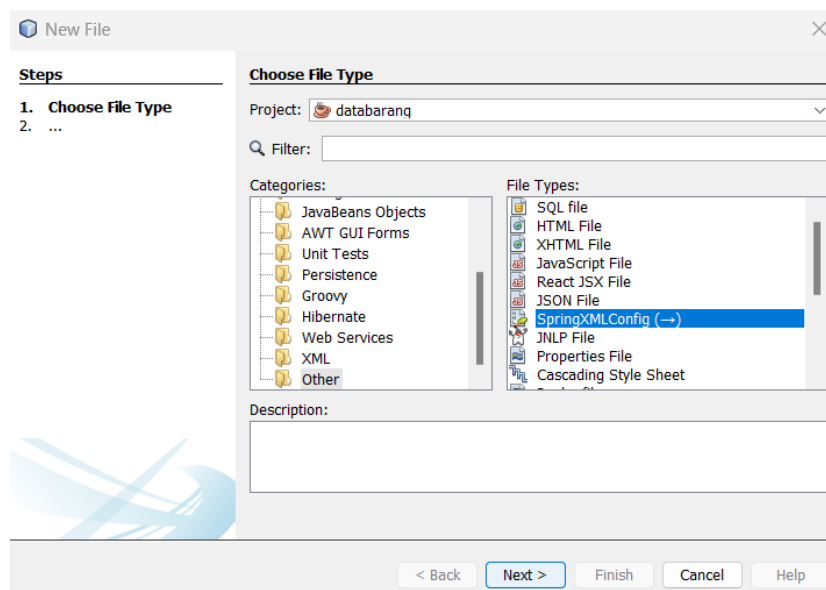
### 3.4.10.2 Membuat Konfigurasi Spring

Selanjutnya Saya akan membuat sebuah Spring framework untuk proyek data barang.

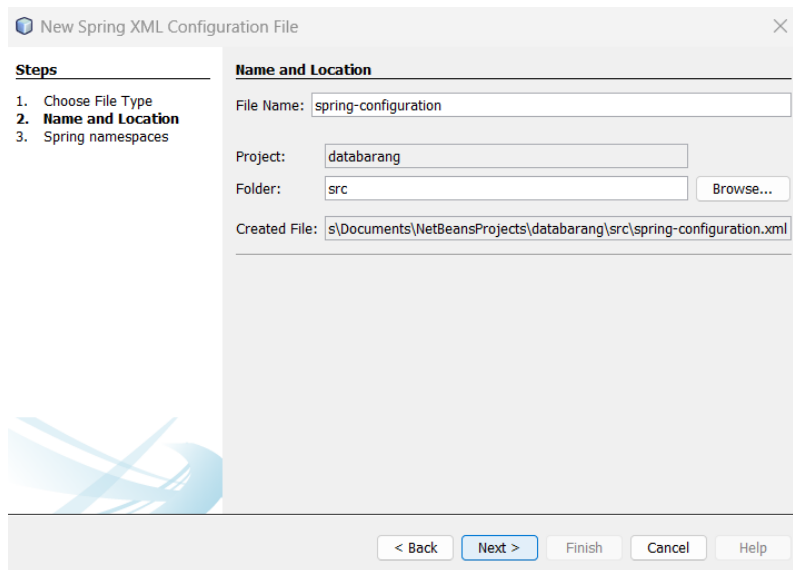
Pada proyek databarang, klik kanan kemudian New-> Other.



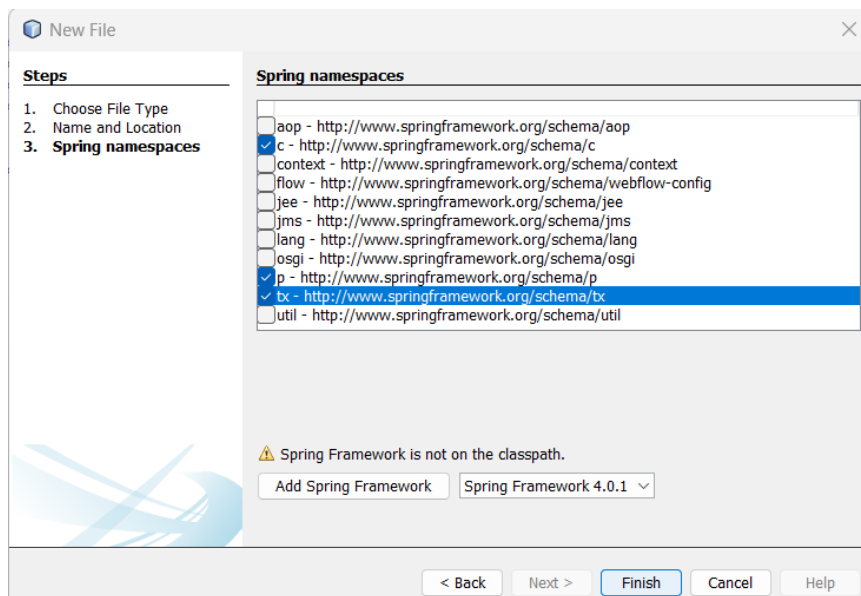
Saat muncul tampilan box 'Choose File Type', Dibawah 'Categories' pilih other -> SpringXMLConfig pada File Types.



Ubah Nama File untuk Spring XML Configuration menjadi “spring-configuration, kemudian klik Next >.

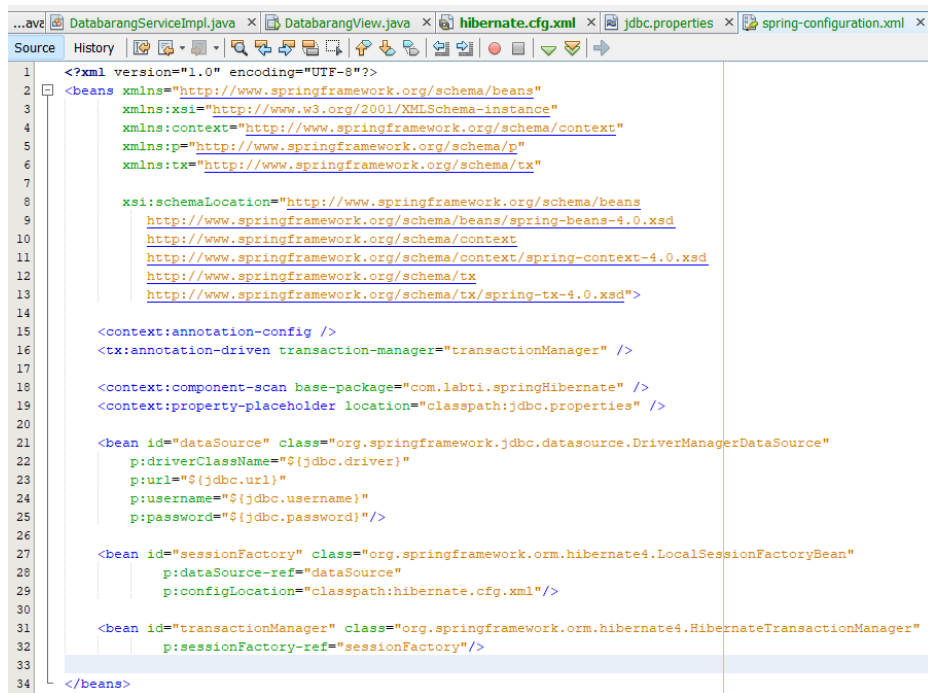


Pada bagian Spring namespaces, centang “c”, “p”, “tx”, kemudian Klik juga Add Spring Framework dengan versi Spring Framework 4.0.1. Jika sudah tekan ‘Finish’.



Setelah selesai menambahkan File spring-configuration, selanjutnya menambahkan kodingan seperti yang tertera pada Listing dibawah ini.





```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:p="http://www.springframework.org/schema/p"
6       xmlns:tx="http://www.springframework.org/schema/tx"
7
8       xsi:schemaLocation="http://www.springframework.org/schema/beans
9                           http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
10                          http://www.springframework.org/schema/context
11                          http://www.springframework.org/schema/context/spring-context-4.0.xsd
12                          http://www.springframework.org/schema/tx
13                          http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">
14
15     <context:annotation-config />
16     <tx:annotation-driven transaction-manager="transactionManager" />
17
18     <context:component-scan base-package="com.labti.springHibernate" />
19     <context:property-placeholder location="classpath:jdbc.properties" />
20
21     <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource"
22           p:driverClassName="${jdbc.driver}"
23           p:url="${jdbc.url}"
24           p:username="${jdbc.username}"
25           p:password="${jdbc.password}"/>
26
27     <bean id="sessionFactory" class="org.springframework.orm.hibernate4.LocalSessionFactoryBean"
28           p:dataSource-ref="dataSource"
29           p:configLocation="classpath:hibernate.cfg.xml"/>
30
31     <bean id="transactionManager" class="org.springframework.orm.hibernate4.HibernateTransactionManager"
32           p:sessionFactory-ref="sessionFactory"/>
33
34 </beans>
```

File XML di atas adalah konfigurasi Spring untuk integrasi dengan Hibernate dalam aplikasi Spring Hibernate. Anotasi dan konvensi konfigurasi digunakan untuk memudahkan pengembangan, termasuk aktivasi pemrosesan anotasi dan transaksi.

Elemen **<context:component-scan base-package="com.labti.springHibernate" />** menentukan paket yang akan dipindai oleh Spring untuk menemukan komponen-komponen yang dikelola.

**<context:property-placeholder location="classpath:jdbc.properties" />** digunakan untuk mengambil konfigurasi koneksi database dari file **jdbc.properties**.

Bean **dataSource**, **sessionFactory**, dan **transactionManager** digunakan untuk mengonfigurasi koneksi database, sesi Hibernate, dan manajer transaksi Hibernate dengan efisien.

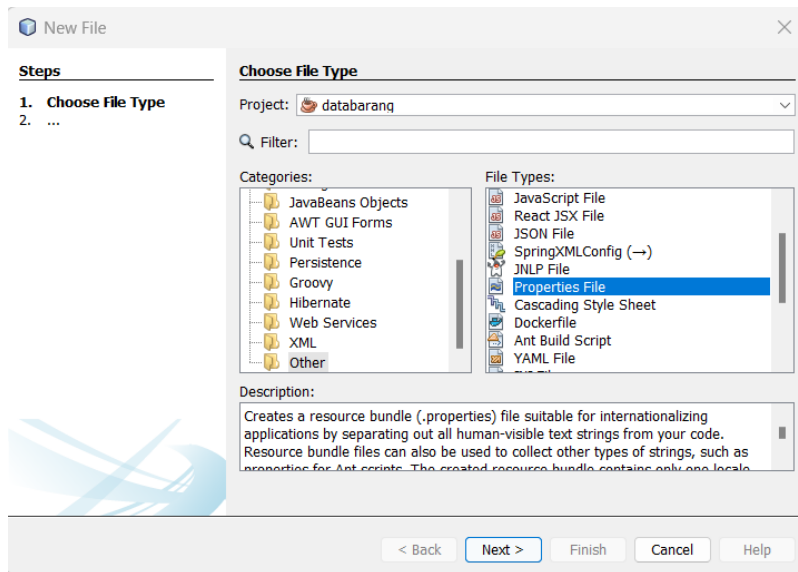
Integrasi antara Spring dan Hibernate mempermudah manajemen komponen, transaksi, dan koneksi database dalam aplikasi Spring Hibernate.

### 3.4.10.3 Membuat Konfigurasi JDBC

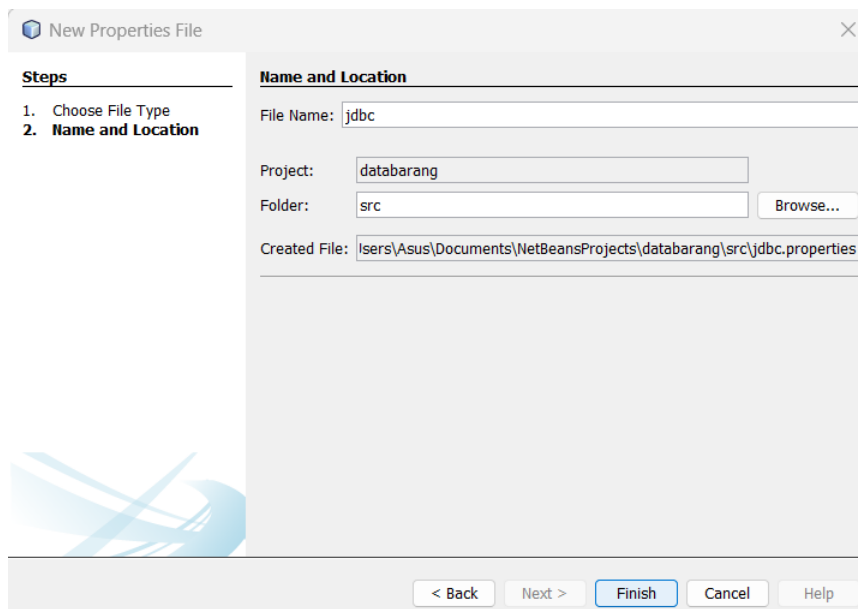
Setelah kita membuat konfigurasi Spring, Hibernate, Selanjutnya membuat konfigurasi JDBC.

Berikut adalah Langkah-langkah dalam pembuatan JDBC.

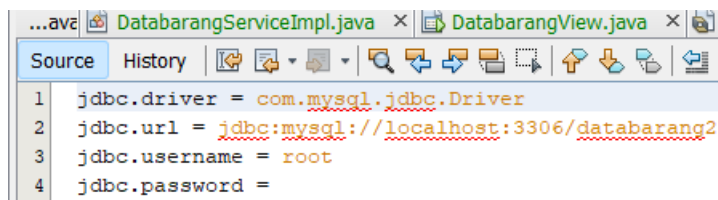
Yang dilakukan pertama yaitu klik kanan pada bagian proyek yang sudah dibuat, kemudian klik kanan kemudian Pilih New -> Other. Kemudian akan muncul Tampilan kotak dialog sebagai berikut. pada Categories pilih Other -> Properties File. Setelah itu klik Next.



Pada 'Name and Location' silahkan ganti File Name dengan 'jdbc' kemudian klik Finish.



Buatlah program JDBC seperti pada yang tertera dibawah ini.



```

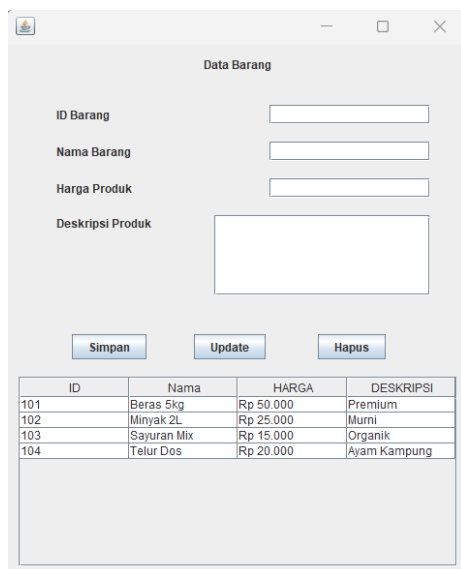
1 jdbc.driver = com.mysql.jdbc.Driver
2 jdbc.url = jdbc:mysql://localhost:3306/databarang2
3 jdbc.username = root
4 jdbc.password =
  
```

File konfigurasi **jdbc.properties** di atas berisi informasi koneksi database untuk aplikasi Spring Hibernate. Property **jdbc.driver** menetapkan kelas driver JDBC sebagai **com.mysql.jdbc.Driver**. Selanjutnya, **jdbc.url** menyediakan URL koneksi database MySQL pada **localhost:3306** dengan nama database **databarang2**. Property **jdbc.username** dan **jdbc.password** menyimpan kredensial untuk akses ke database, dengan username **root** dan password kosong. Konfigurasi ini memungkinkan pengelolaan koneksi database tanpa perlu mengubah kode sumber aplikasi.

### 3.4.11 Tampilan Output Implementasi Spring, Hibernate, JDBC

Berikut ini adalah tampilan setelah menjalankan app.java.

Tampilan Data pada JFrame

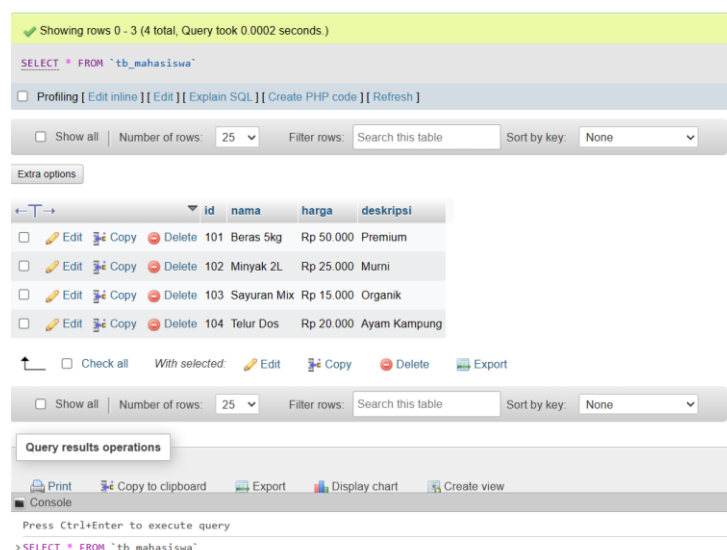


The JFrame titled "Data Barang" contains the following elements:

- Input fields for "ID Barang", "Nama Barang", and "Harga Produk".
- A text area for "Deskripsi Produk".
- Buttons for "Simpan", "Update", and "Hapus".
- A table displaying the following data:
 

ID	Nama	HARGA	DESKRIPSI
101	Beras 5kg	Rp 50.000	Premium
102	Minyak 2L	Rp 25.000	Murni
103	Sayuran Mix	Rp 15.000	Organik
104	Telur Dos	Rp 20.000	Ayam Kampung

Tampilan Data pada MySQL:



The MySQL query client shows the following information:

- Status: Showing rows 0 - 3 (4 total). Query took 0.0002 seconds.
- Query: `SELECT * FROM `tb_mahasiswa``
- Options: Show all, Number of rows: 25, Filter rows: Search this table, Sort by key: None.
- Query results table:
 

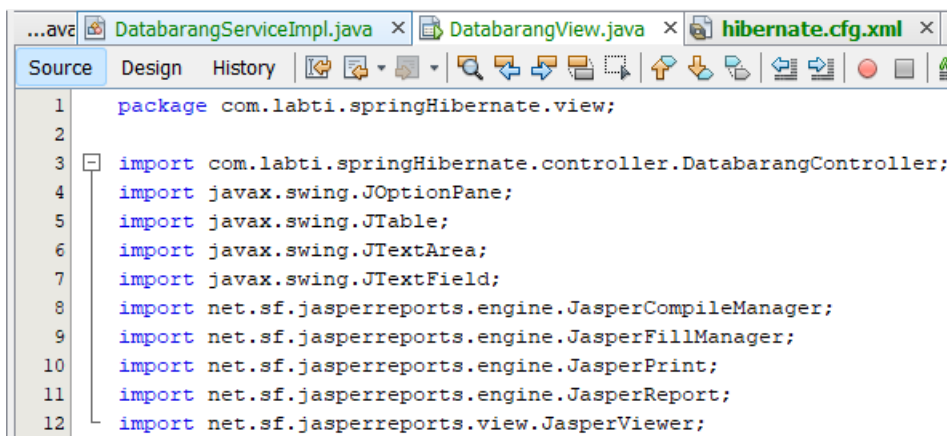
	id	nama	harga	deskripsi
<input type="checkbox"/>	101	Beras 5kg	Rp 50.000	Premium
<input type="checkbox"/>	102	Minyak 2L	Rp 25.000	Murni
<input type="checkbox"/>	103	Sayuran Mix	Rp 15.000	Organik
<input type="checkbox"/>	104	Telur Dos	Rp 20.000	Ayam Kampung
- Query results operations: Print, Copy to clipboard, Export, Display chart, Create view.
- Console: Press Ctrl+Enter to execute query. Command: `> SELECT * FROM `tb_mahasiswa``

### 3.5 Membuat JasperReports

Pada bagian ini, saya akan mengeksplorasi implementasi NetBeans menggunakan Aspek Berorientasi Objek (AOP) dan JasperReports dalam pengembangan aplikasi Java. Aspek Berorientasi Objek memungkinkan saya untuk memisahkan tanggung jawab-tanggung jawab tertentu dalam aplikasi saya, sementara JasperReports memberikan kemampuan untuk membuat laporan dengan tata letak yang dapat disesuaikan.

#### 3.5.1 Mendeklarasikan JasperCompile Dan JasperFillManager

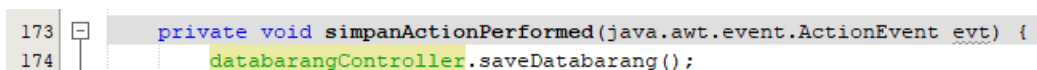
Pada bagian DatabarangView.java, Mendeklarasikan Import dengan ketentuan sebagai berikut.



```
1 package com.labti.springHibernate.view;
2
3 import com.labti.springHibernate.controller.DatabarangController;
4 import javax.swing.JOptionPane;
5 import javax.swing.JTable;
6 import javax.swing.JTextArea;
7 import javax.swing.JTextField;
8 import net.sf.jasperreports.engine.JasperCompileManager;
9 import net.sf.jasperreports.engine.JasperFillManager;
10 import net.sf.jasperreports.engine.JasperPrint;
11 import net.sf.jasperreports.engine.JasperReport;
12 import net.sf.jasperreports.view.JasperViewer;
```

Kode di atas adalah bagian dari sebuah kelas Java yang merupakan bagian dari aplikasi Java yang menggunakan kerangka kerja Spring dan Hibernate untuk mengelola data barang dan mendeklarasikan sebuah import paket.

Kemudian ke tab Design -> Klik Tombol Simpan, maka muncul tab berikut ini.



```
173 private void simpanActionPerformed(java.awt.event.ActionEvent evt) {
174     databarangController.saveDatabarang();
```

Pada bagian metode **saveDatabarang()** dari objek **databarangController**. Ini menunjukkan bahwa ada suatu kontroler (**databarangController**) yang mengelola logika penyimpanan data barang.

Setelah itu melakukan kodingan pada jasperReport seperti dibawah ini.

```

173 private void simpanActionPerformed(java.awt.event.ActionEvent evt) {
174     databarangController.saveDatabarang();
175     try{
176         JasperReport jasperReport = JasperCompileManager.compileReport(getClass().
177             getResourceAsStream("reportDatabarang.jrxml"));
178         JasperPrint jp = JasperFillManager.fillReport(jasperReport, null, Koneksi.
179             getConnection());
180         JasperViewer.viewReport(jp, false);
181     }catch(Exception e){
182         JOptionPane.showMessageDialog(rootPane, e);
183     }
184 }

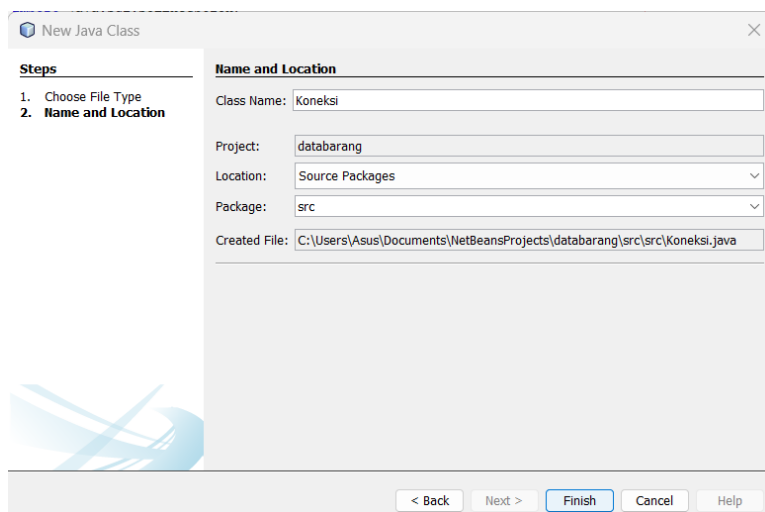
```

Berikut adalah Penjelasan dari logika kodingan diatas.

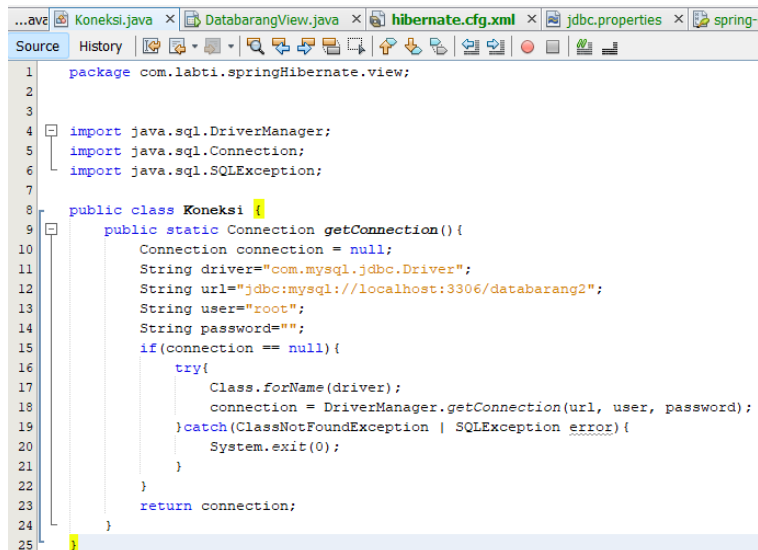
Metode **simpanActionPerformed** dalam antarmuka pengguna Swing memanggil metode **saveDatabarang** dari objek **databarangController** untuk menyimpan data barang. Selanjutnya, dalam blok percobaan, laporan JasperReports di-generate dan ditampilkan menggunakan **JasperCompileManager** dan **JasperFillManager**. Kesalahan selama proses ini ditangkap dan ditampilkan dalam kotak dialog pesan. Logika ini memungkinkan fungsionalitas penyimpanan data dan generasi laporan dalam satu aksi pengguna.

### 3.5.2 Membuat Class Koneksi.java

Pada bagian ini akan diulas mengenai pembuatan Class baru dengan nama Koneksi.java. Berikut adalah Contoh dari pembuatan Class Koneksi.java. Klik Finish untuk selesai.



Pada Koneksi.java, membuat sebuah kodingan perintah sebagai berikut.



```

1 package com.labti.springHibernate.view;
2
3
4 import java.sql.DriverManager;
5 import java.sql.Connection;
6 import java.sql.SQLException;
7
8 public class Koneksi {
9     public static Connection getConnection(){
10         Connection connection = null;
11         String driver="com.mysql.jdbc.Driver";
12         String url="jdbc:mysql://localhost:3306/databarang2";
13         String user="root";
14         String password="";
15         if(connection == null){
16             try{
17                 Class.forName(driver);
18                 connection = DriverManager.getConnection(url, user, password);
19             }catch(ClassNotFoundException | SQLException error){
20                 System.exit(0);
21             }
22         }
23         return connection;
24     }
25 }

```

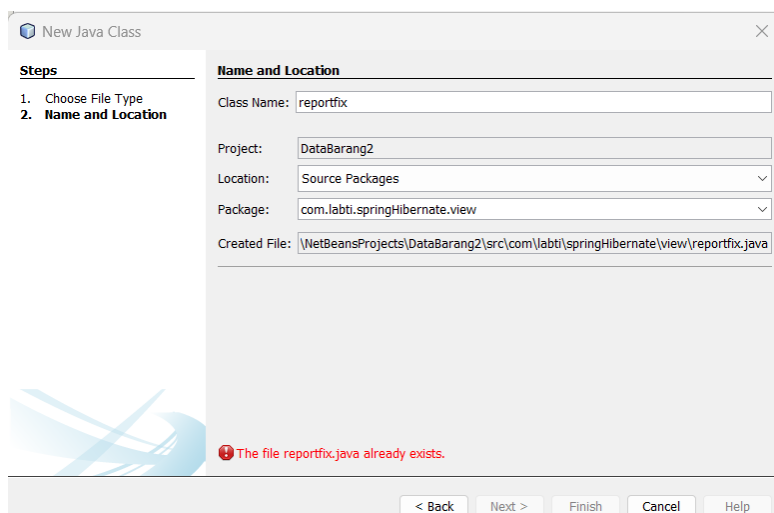
Kelas **Koneksi** menyediakan metode statis **getConnection** untuk mengambil objek **Connection** yang merepresentasikan koneksi ke database MySQL. Variabel-variabel seperti **driver**, **url**, **user**, dan **password** diinisialisasi dengan nilai-nilai yang sesuai.

Dalam blok **try**, kelas driver JDBC dimuat menggunakan **Class.forName**, dan koneksi ke database diinisiasi dengan **DriverManager.getConnection**. Jika ada kesalahan selama proses ini, program keluar.

Logika ini memastikan bahwa hanya satu koneksi yang dibuat dan digunakan selama aplikasi berjalan, sehingga mengoptimalkan pengelolaan sumber daya.

### 3.5.3 Membuat Class ReportFix.java

Untuk membuat Class java ReportFix.java, klik kanan project. Setelah itu navigasikan ke New -> Java Class.



New Java Class

**Steps**

1. Choose File Type
2. Name and Location

**Name and Location**

Class Name: reportfix

Project: DataBarang2

Location: Source Packages

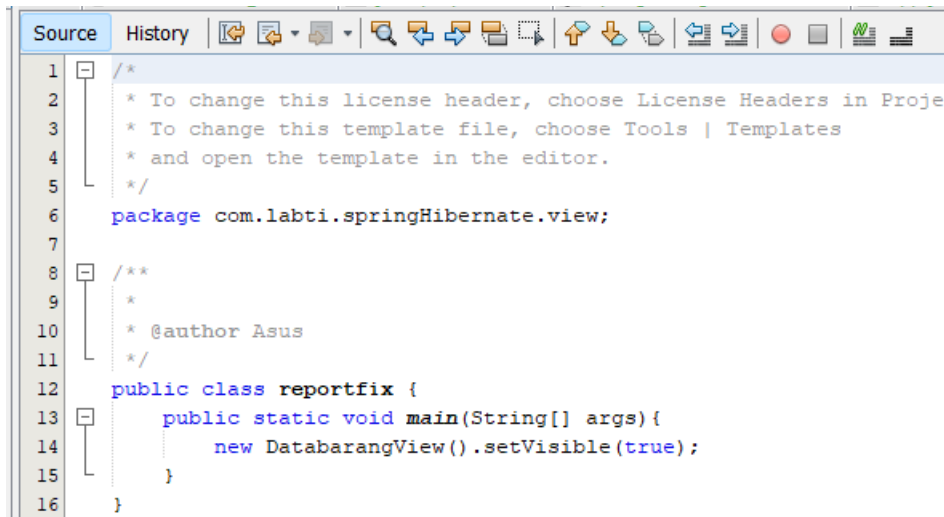
Package: com.labti.springHibernate.view

Created File: |NetBeansProjects|DataBarang2|src|com|labti|springHibernate|view|reportfix.java

The file reportfix.java already exists.

< Back Next > Finish Cancel Help

Setelah Membuat sebuah Class ReportFix.java, Selanjutnya membuat kodingan seperti dibawah ini.



```
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 com.labti.springHibernate.view;
7
8   /**
9    *
10   * @author Asus
11   */
12   public class reportfix {
13       public static void main(String[] args){
14           new DatabarangView().setVisible(true);
15       }
16   }
```

Kelas **reportfix** merupakan kelas dengan metode **main** yang digunakan untuk memulai aplikasi. Di dalam metode **main**, sebuah objek dari kelas **DatabarangView** dibuat dan ditampilkan menggunakan metode **setVisible(true)**. Ini memicu tampilan antarmuka pengguna (UI) dari aplikasi untuk muncul di layar. Logika ini menunjukkan bahwa aplikasi dimulai dengan menampilkan tampilan data barang ke pengguna secara langsung.

Jadi, program ini secara praktis membuat dan menampilkan antarmuka pengguna untuk manajemen data barang.

Tampilan pada Frame Input data barang:

— □ ✕

### Data Barang

**ID Barang**

**Nama Barang**

**Harga Produk**

**Deskripsi Produk**

Simpan

Update

Hapus

ID	Nama	HARGA	DESKRIPSI
101	Beras 5kg	Rp 50.000	Premium
102	Minyak 2L	Rp 25.000	Murni
103	Sayuran Mix	Rp 15.000	Organik
104	Telur Dos	Rp 20.000	Ayam Kampung
105	Susu 800g	Rp 50.000	Formula

Tampilan pada reportDatabarang.jrxml:



### 3.6 Langkah-langkah Pembuatan Akun Git Dan Publish Project

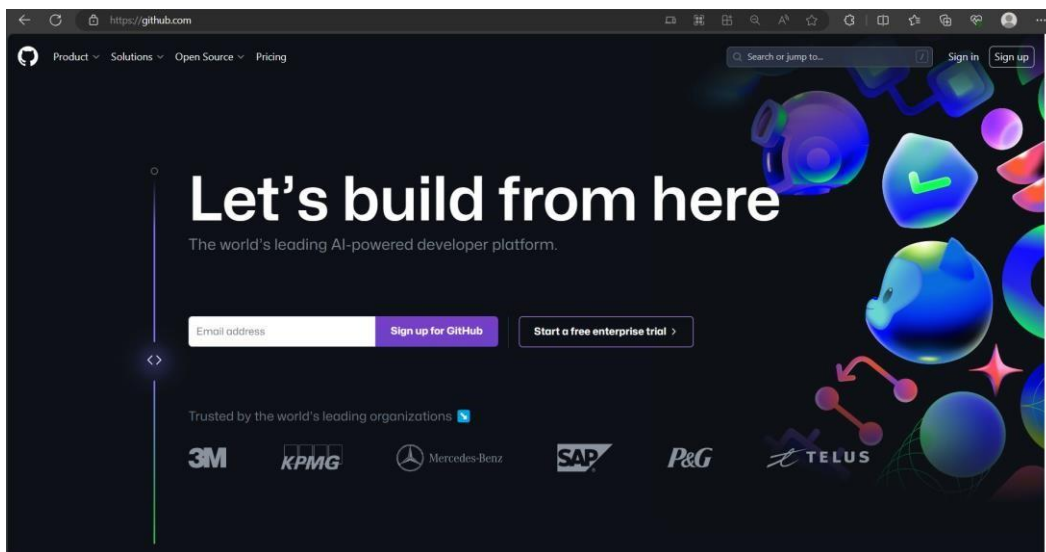
Pada praktikum ke-7 kali ini, saya akan membahas langkah-langkah tutorial pembuatan akun GitHub dan proses memasukkan file ke dalam GitHub menggunakan perintah-perintah yang ada di aplikasi Git. GitHub adalah platform kolaborasi pengembangan perangkat lunak yang memungkinkan para pengembang bekerja sama dalam pengembangan proyek secara efisien. Git, sebagai sistem kontrol versi, membantu dalam manajemen perubahan kode sumber.

Berikut adalah tutorial singkat tentang cara membuat akun GitHub dan memasukkan file ke dalam GitHub menggunakan perintah-perintah yang ada di aplikasi Github.

#### 3.6.1 Langkah-langkah Tutorial Git

##### 1. Membuat Akun GitHub

- Buka situs web GitHub di <https://github.com>.




- Klik tombol "Sign up" di pojok kanan atas.
- Isi formulir pendaftaran dengan username, alamat email, dan password yang diinginkan.

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ k9230prasetya@gmail.com

Create a password\*

→ .....  Continue

Password is strong

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

- Setelah mengisi formulir, klik "Create account".

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ k9230prasetya@gmail.com

Create a password\*

✓ .....

Enter a username\*

✓ Pras67brb

Email preferences

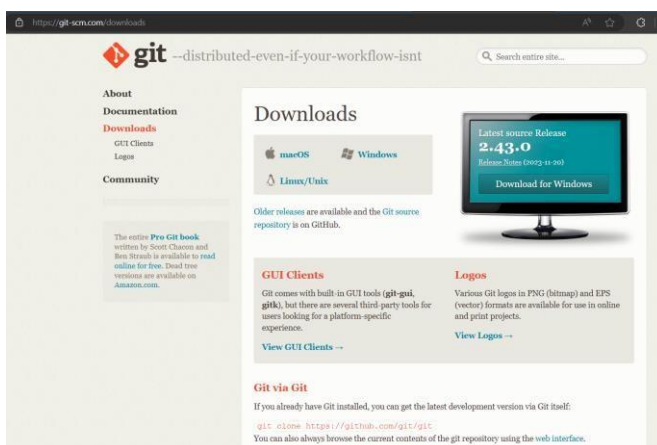
☒ Receive occasional product updates and announcements.

Continue

- Ikuti petunjuk selanjutnya untuk menyelesaikan proses pendaftaran.

## 2. Menginstal Git

- Unduh dan instal Git dari <https://git-scm.com/downloads>.



- Setelah instalasi selesai, buka Git Bash atau terminal.

### 3. Mengatur Git

- Atur username dan email Git Anda dengan perintah: `git config --global user.name "Your Name"`

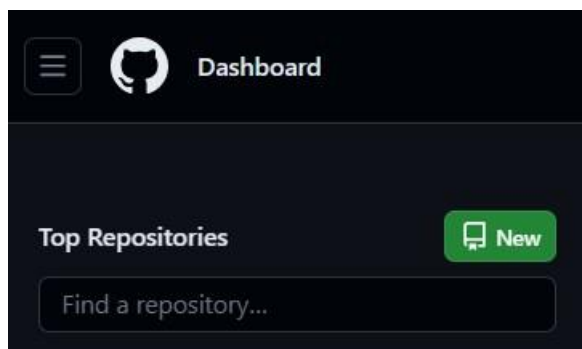
`git config --global user.email "youremail@example.com"`

```
Asus@LAPTOP-000FQ7EE MINGW64 ~
$ git config --global user.name Prasetya58

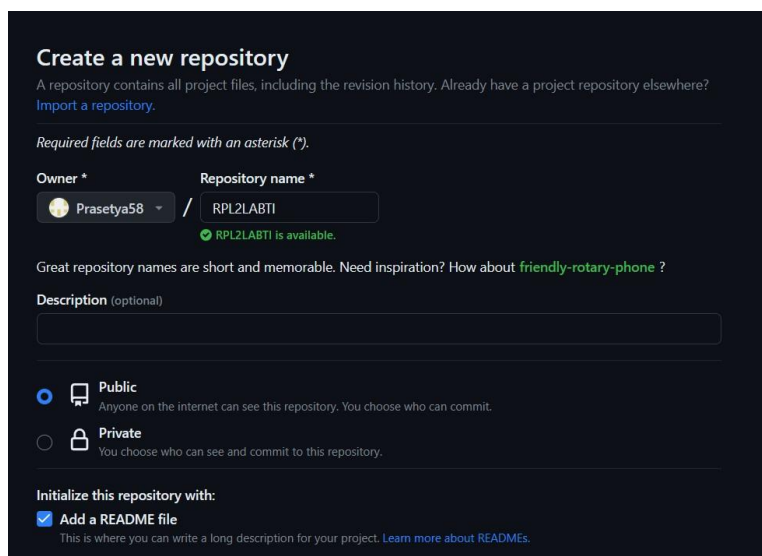
Asus@LAPTOP-000FQ7EE MINGW64 ~
$ git config --global user.email k9230prasetya.h@gmail.com
```

### 4. Membuat Repositori Baru

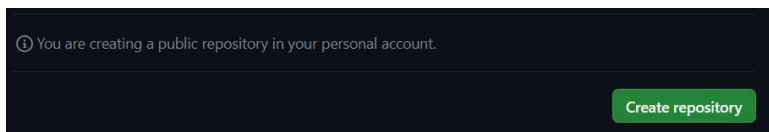
- Di GitHub, klik tombol "New repository" di halaman utama atau di dashboard Anda.



- Beri nama repositori dan deskripsi (opsional), pilih apakah repositori ini akan bersifat publik atau pribadi, dan apakah Anda ingin menambahkan file README.

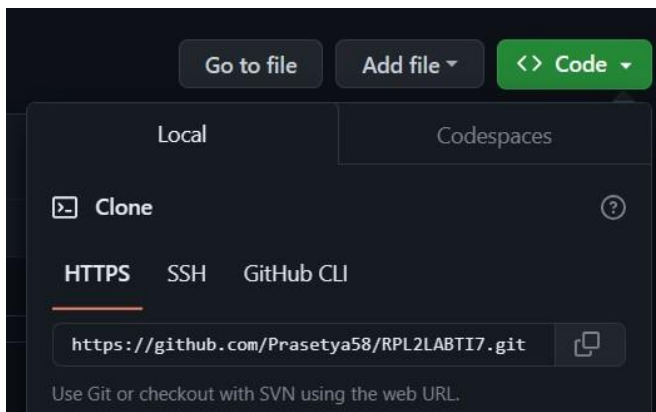
The image shows the 'Create a new repository' form on GitHub. The form has a title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, it says 'Required fields are marked with an asterisk (\*)'. The form has two main sections: 'Owner' and 'Repository name'. The 'Owner' section has a dropdown menu showing 'Prasetya58'. The 'Repository name' section has a text input field containing 'RPL2LABTI' and a green checkmark indicating 'RPL2LABTI is available.'. Below these fields, there's a hint: 'Great repository names are short and memorable. Need inspiration? How about [friendly-rotary-phone](#) ?'. The 'Description' section has a label 'Description (optional)' and a text input field. At the bottom, there are two radio buttons for 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone on the internet can see this repository. You choose who can commit.'. The 'Private' option has a description: 'You choose who can see and commit to this repository.'. At the very bottom, there's a section 'Initialize this repository with:' with a checked checkbox for 'Add a README file' and a link 'Learn more about READMEs.'.

- Klik "Create repository".

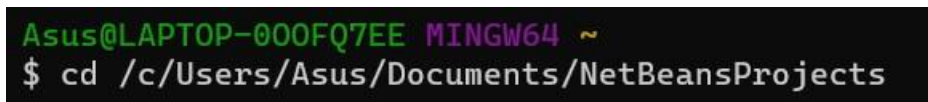


## 5. Mengkloning Repositori ke direktori komputer lokal.

- Navigasikan ke Tab `<>` **Code**, kemudian klik tombol **Code** `<>` berwarna hijau yang ada pada repository tersebut.
- Salin URL repository GitHub Anda.

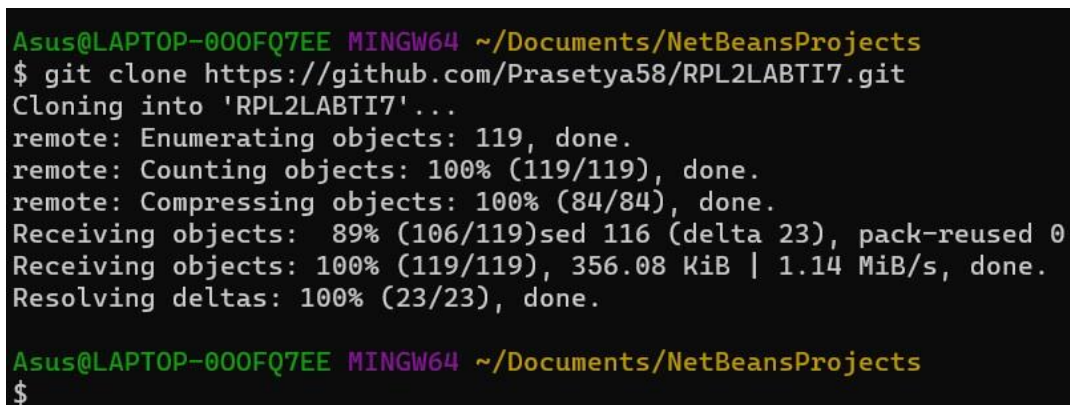


- Ubah direktori yang digunakan dengan directory proyek yang ingin digunakan



- Buka Git Bash atau terminal, navigasikan ke direktori tempat Anda ingin menyimpan repository, lalu jalankan perintah:

*'git clone "URL Repositori"'*



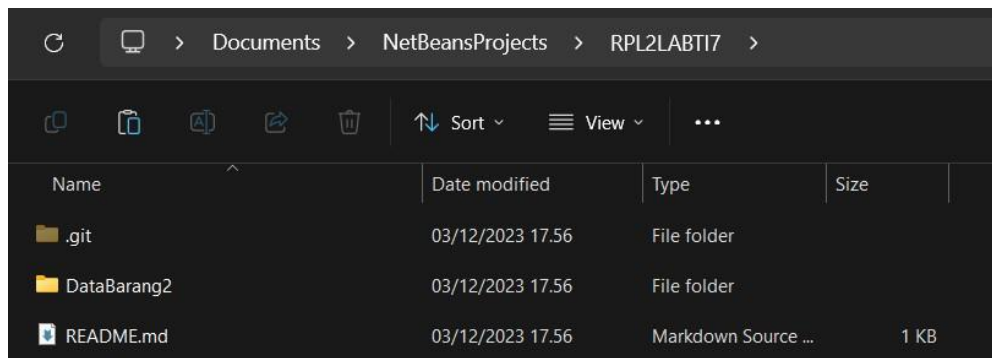
## 6. Menambahkan File ke Repositori

- Pindah ke direktori repositori di terminal Anda (cd nama-repositori).

```
Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects
$ cd C:/Users/Asus/Documents/NetBeansProjects/RPL2LABTI7

Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects/RPL2LABTI7 (main)
$
```

- Pindahkan folder proyek Netbeans ke folder repositori clone baru yang sudah dibuat.



- Tambahkan file ke Git dengan perintah:

*'git add .'*

```
Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects/RPL2LABTI7 (main)
$ git add .
```

- Buat commit dengan pesan yang menjelaskan perubahan:

*'git commit -m "Pesan Commit"'*

```
Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects/RPL2LABTI7 (main)
$ git commit -m "1st Push"
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

## 7. Mendorong Perubahan ke GitHub

- Dorong commit ke GitHub dengan perintah:

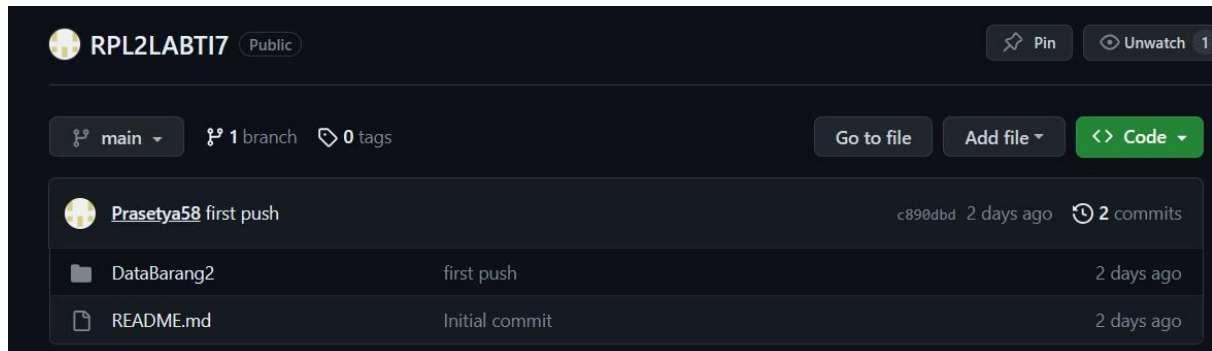
*'git push origin main'*

- Masukkan username dan password GitHub Anda jika diminta.

```
Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects/RPL2LABTI7 (main)
$ git push origin main
Everything up-to-date

Asus@LAPTOP-000FQ7EE MINGW64 ~/Documents/NetBeansProjects/RPL2LABTI7 (main)
$
```

### 3.6.2 Hasil Penempatan Folder Proyek Pada Repository “RPL2LABTI7”



Dengan mengikuti langkah-langkah di atas, Anda telah berhasil membuat akun GitHub, menginstal Git, membuat repositori baru, mengkloning repositori ke komputer lokal, menambahkan dan melakukan commit pada file, serta mendorong perubahan ke GitHub. Semua langkah tersebut memungkinkan kolaborasi yang efisien dalam pengembangan proyek perangkat lunak.

## **BAB IV**

### **PENUTUP**

#### **4.1 Kesimpulan**

Dengan adanya praktikum ini, telah berhasil dirancang sebuah aplikasi E-Groceries menggunakan Netbeans. Aplikasi ini memiliki beberapa fitur yang memudahkan pengguna dalam melakukan pembelian barang kebutuhan sehari-hari secara online. Kesimpulan yang dapat diambil dari pengembangan aplikasi ini adalah:

- Aplikasi E-Groceries mampu memberikan kemudahan akses bagi pengguna dalam melakukan belanja kebutuhan sehari-hari tanpa harus datang langsung ke toko.
- Integrasi dengan Netbeans sebagai platform pengembangan aplikasi membantu mempercepat proses pengembangan dan memastikan aplikasi dapat berjalan dengan stabil.
- Fitur-fitur yang disematkan dalam aplikasi, seperti keranjang belanja, riwayat transaksi, dan pencarian produk, memberikan pengalaman berbelanja yang lebih baik bagi pengguna.
- Dengan adanya aplikasi ini, diharapkan dapat meningkatkan efisiensi waktu dan tenaga pengguna dalam memenuhi kebutuhan sehari-hari.

#### **4.2 Saran**

Beberapa saran yang dapat diterapkan antara lain:

- Lebih teliti pada saat implementasi fitur keamanan untuk melindungi data pengguna dan transaksi agar tidak mudah disalahgunakan.
- Melakukan pengujian secara menyeluruh untuk memastikan bahwa semua fitur dalam aplikasi berjalan dengan baik dan tidak ada bug yang dapat mengganggu pengalaman pengguna.
- Menerima masukan dan feedback dari pengguna untuk terus melakukan pembaruan dan peningkatan sesuai dengan kebutuhan dan harapan pengguna.
- Menyusun strategi pemasaran yang efektif untuk meningkatkan jumlah pengguna aplikasi dan memperluas cakupan pasar.