

# **LAPORAN AKHIR PRAKTIKUM**

Mata Praktikum : RPL 2  
Kelas : 4IA27  
Praktikum ke - : 3 & 4  
Tanggal : 26 Oktober 2024  
Materi : Pom.xml & Model Tabel Mahasiswa  
NPM : 51421307  
Nama : Riezky Arif Fadhilah  
Ketua Asisten :  
Jumlah Lembar : 11



**LABORATORIUM TEKNIK INFORMATIKA**  
**UNIVERSITAS GUNADARMA**

## Laporan Akhir 3

### Act3\_51421307

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  */
4
5  package me.iky.act_51421307;
6
7  /**
8   *
9   * @author Riezky
10  */
11  public class Act_51421307 {
12
13      public static void main(String[] args) {
14          System.out.println("Hello World!");
15      }
16  }
17
```

- Membuat file baru dengan nama ACT3\_50421445, yang di dalamnya hanya berisikan perintah mencetak hello world saja

### Pom.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  <modelVersion>4.0.0</modelVersion>
4  <groupId>me.iky</groupId>
5  <artifactId>act_51421307</artifactId>
6  <version>1.0-SNAPSHOT</version>
7  <packaging>jar</packaging>
8  <properties>
9      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10     <maven.compiler.source>17</maven.compiler.source>
11     <maven.compiler.target>17</maven.compiler.target>
12     <exec.mainClass>me.iky.act_51421307.Act_51421307</exec.mainClass>
13 </properties>
14 <dependencies>
15     <dependency>
16         <groupId>mysql</groupId>
17         <artifactId>mysql-connector-java</artifactId>
18         <version>8.0.33</version>
19     </dependency>
20 </dependencies>
21 </project>
```

- **Model Version:**  
Bagian `<modelVersion>4.0.0</modelVersion>` adalah versi standar yang digunakan Maven buat ngatur proyek. Ini versi dasarnya, yaitu 4.0.0.
- **Group ID:**  
`<groupId>e.iky</groupId>` ini semacam "nama belakang" proyek, biasanya nunjukin nama organisasi atau tim yang bikin proyek ini. Di sini pakai e.iky.
- **Artifact ID:**  
`<artifactId>act_51421307</artifactId>` adalah "nama unik" untuk proyek ini di dalam grupnya. Jadi, ini kayak nama pendeknya, yaitu act\_51421307.
- **Version:**  
`<version>1.0-SNAPSHOT</version>` ini versi proyeknya. Kata SNAPSHOT artinya proyek ini masih tahap pengembangan, jadi belum versi final.
- **Packaging:**  
`<packaging>jar</packaging>` artinya output atau hasil akhirnya nanti berbentuk file JAR, yang biasa dipakai untuk aplikasi berbasis Java.

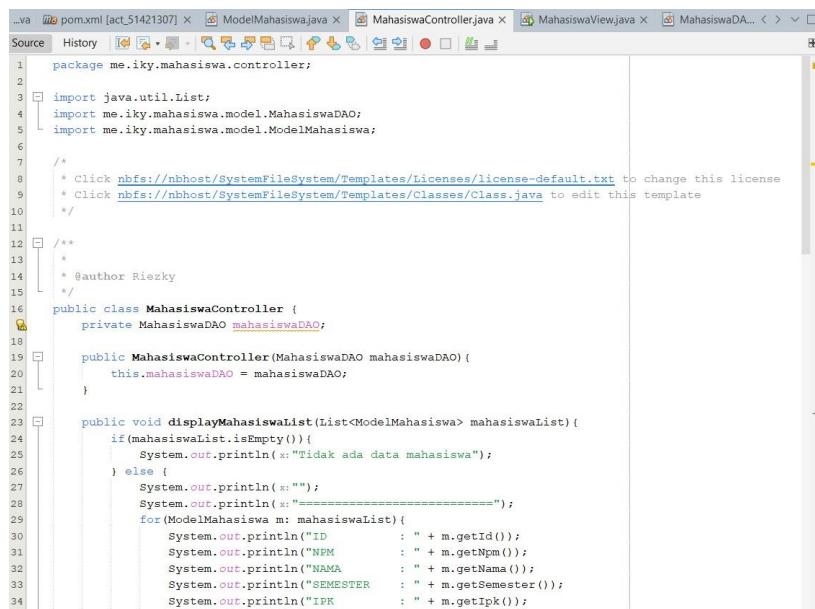
## Build Encoding:

`<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>` ini nentuin kalau proyek ini pakai format karakter UTF-8, biar nanti teksnya nggak error pas dijalankan di berbagai sistem.

## Versi Java:

- `<maven.compiler.source>17</maven.compiler.source>` dan `<maven.compiler.target>17</maven.compiler.target>` kegunannya untuk memberitahu kalau proyek ini pakai Java versi 17 buat kompilasi atau ngejalanin program.
- **Main Class:**  
`<exec.mainClass>e.iky.act_51421307.Act_51421307</exec.mainClass>` ini nunjukin kelas utama yang akan dijalankan saat program di-start, di sini kelas utamanya adalah `Act_51421307`.

## Mahasiswa.Controller



```
1 package me.iky.mahasiswa.controller;
2
3 import java.util.List;
4 import me.iky.mahasiswa.model.MahasiswaDAO;
5 import me.iky.mahasiswa.model.ModelMahasiswa;
6
7
8 /*
9  * Click nbfs://nbhost/SystemFileSystem/Templates/licenses/license-default.txt to change this license
10  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
11  */
12
13 /**
14  *
15  * @author Riezky
16  */
17 public class MahasiswaController {
18     private MahasiswaDAO mahasiswaDAO;
19
20     public MahasiswaController(MahasiswaDAO mahasiswaDAO) {
21         this.mahasiswaDAO = mahasiswaDAO;
22     }
23
24     public void displayMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
25         if (mahasiswaList.isEmpty()) {
26             System.out.println("Tidak ada data mahasiswa");
27         } else {
28             System.out.println("");
29             System.out.println("=====");
30             for (ModelMahasiswa m : mahasiswaList) {
31                 System.out.println("ID : " + m.getId());
32                 System.out.println("NPM : " + m.getNpm());
33                 System.out.println("NAMA : " + m.getNama());
34                 System.out.println("SEMESTER : " + m.getSemester());
35                 System.out.println("IEK : " + m.getIpk());
36             }
37         }
38     }
39 }
```

## Package dan Import:

- `package e.iky.mahasiswa.controller;` mendefinisikan bahwa kelas ini berada dalam paket `e.iky.mahasiswa.controller`.
- Ada beberapa import, seperti `java.util.List`, `e.iky.mahasiswa.model.MahasiswaDAO`, dan `e.iky.mahasiswa.model.ModelMahasiswa`, yang menyediakan akses ke tipe data `List` dan kelas-kelas lain dalam proyek ini, yaitu `MahasiswaDAO` dan `ModelMahasiswa`.

## Deklarasi Kelas MahasiswaController:

- Kelas `MahasiswaController` memiliki anotasi `@author Riezky`, yang mungkin menunjukkan siapa penulis atau pengembang kelas ini.
- Kelas ini menggunakan pola MVC (Model-View-Controller), di mana `MahasiswaController` bertugas mengelola data dari model (`MahasiswaDAO` dan `ModelMahasiswa`) dan menampilkannya sesuai kebutuhan.

### Atribut mahasiswaDAO:

- private MahasiswaDAO mahasiswaDAO; adalah variabel privat yang mengacu pada objek MahasiswaDAO. Objek ini akan digunakan untuk mengambil atau memanipulasi data mahasiswa.

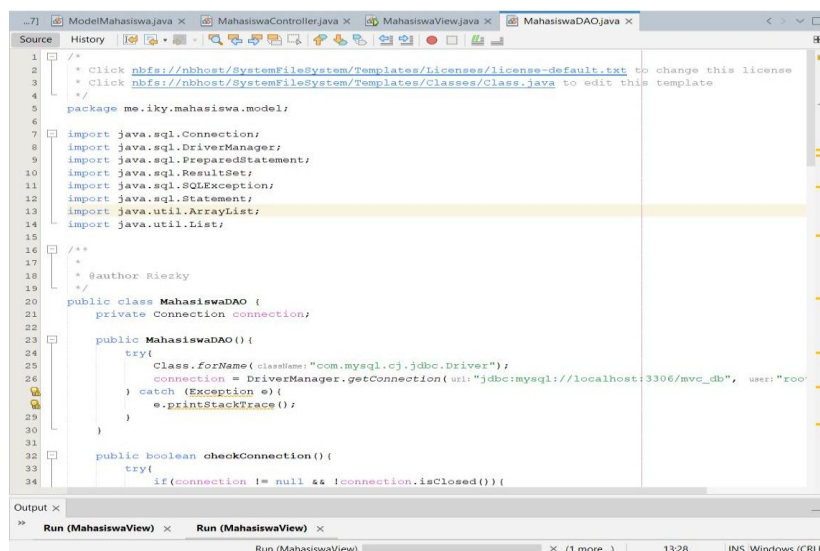
### Konstruktor MahasiswaController:

- public MahasiswaController(MahasiswaDAO mahasiswaDAO) {...} adalah konstruktor yang menerima parameter MahasiswaDAO mahasiswaDAO dan menginisialisasi atribut mahasiswaDAO dengan objek yang diberikan sebagai argumen.

### Method displayMahasiswaList:

- Method ini bertanggung jawab untuk menampilkan daftar mahasiswa yang diberikan sebagai parameter List<ModelMahasiswa> mahasiswaList.
- Pertama, dicek apakah mahasiswaList kosong (mahasiswaList.isEmpty()). Jika kosong, maka akan mencetak pesan "tidak ada data mahasiswa".
- Jika mahasiswaList tidak kosong, method ini akan melakukan iterasi melalui setiap objek ModelMahasiswa di dalam daftar dan mencetak detailnya, seperti ID, NPM, NAMA, SEMESTER, dan IPK.
- Tampilan ini menggunakan System.out.println untuk mencetak data ke konsol atau terminal.

## MahasiswaDAO



```
1  package me.iky.mahasiswa.model;
2
3  /*
4   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
5   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
6   */
7
8  import java.sql.Connection;
9  import java.sql.DriverManager;
10 import java.sql.PreparedStatement;
11 import java.sql.ResultSet;
12 import java.sql.SQLException;
13 import java.util.ArrayList;
14 import java.util.List;
15
16 /**
17  *
18  * @author Riezky
19  */
20 public class MahasiswaDAO {
21     private Connection connection;
22
23     public MahasiswaDAO() {
24         try {
25             Class.forName("com.mysql.cj.jdbc.Driver");
26             connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mvc_db", "root", "");
27         } catch (Exception e) {
28             e.printStackTrace();
29         }
30     }
31
32     public boolean checkConnection() {
33         try {
34             if (connection != null && !connection.isClosed()) {
```

### Model (MahasiswaDAO.java, ModelMahasiswa.java):

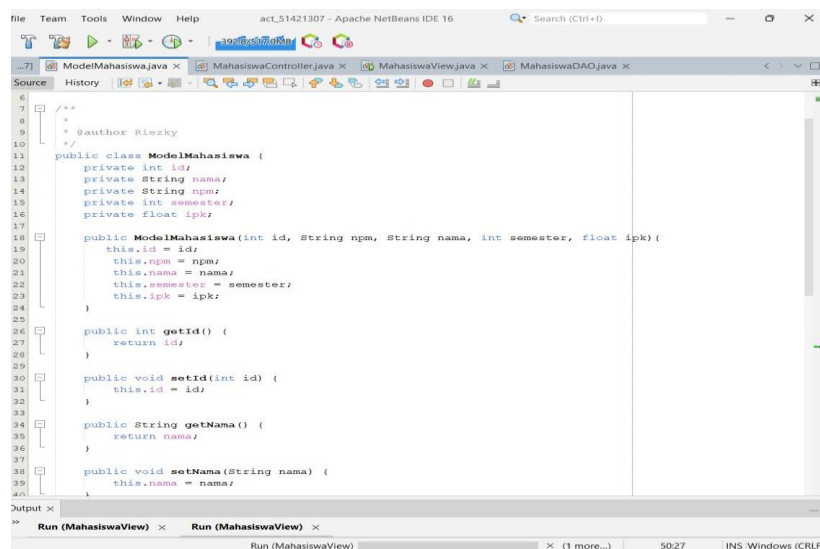
- Berisi kelas MahasiswaDAO yang bertugas untuk mengelola data mahasiswa. Pada file ini, ada kode untuk koneksi ke database MySQL, menggunakan driver com.mysql.cj.jdbc.Driver.
- Fungsi MahasiswaDAO mencoba membuat koneksi dengan database yang ada di localhost pada port 3306, dengan nama database mvc\_db dan menggunakan username "root" serta password kosong.

- Ada juga metode `checkConnection()` yang digunakan untuk memastikan apakah koneksi ke database berhasil atau tidak.

### View (MahasiswaView.java):

- File ini kemungkinan berisi tampilan antarmuka pengguna (UI) untuk menampilkan data mahasiswa, meskipun di gambar ini tidak terlihat isi detail dari file tersebut.
- Controller (MahasiswaController.java):
- Paket ini mengatur logika atau kendali alur data antara model (data mahasiswa) dan view (tampilan).

## Model Mahasiswa



### Atribut (Properties):

- `private int id;` : Menyimpan ID mahasiswa.
- `private String nama;` : Menyimpan nama mahasiswa.
- `private String npm;` : Menyimpan Nomor Pokok Mahasiswa (NPM) mahasiswa.
- `private int semester;` : Menyimpan semester mahasiswa.
- `private float ipk;` : Menyimpan IPK (Indeks Prestasi Kumulatif) mahasiswa.

### Constructor:

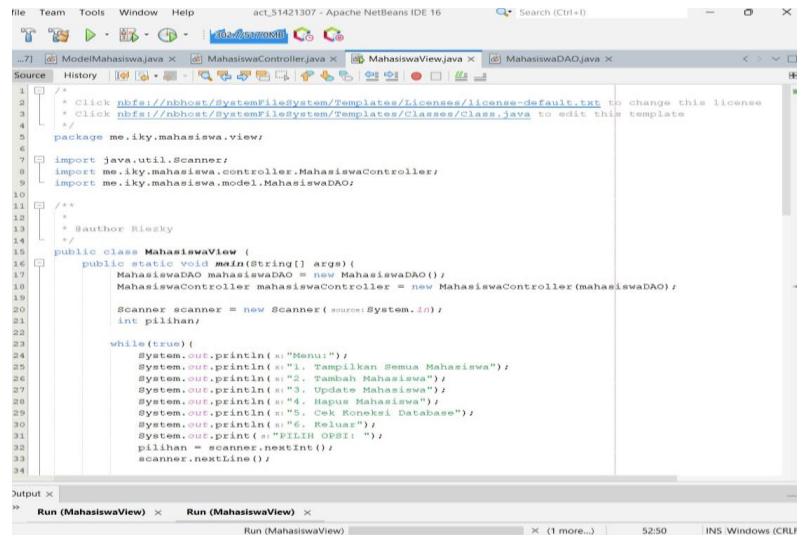
- Selanjutnya yaitu disini ada `public ModelMahasiswa` terdiri dari (`int id`, `String npm`, `String nama`, `int semester`, `float ipk`): Merupakan constructor yang digunakan untuk membuat objek `ModelMahasiswa` dengan menginisialisasi atribut-atributnya. Parameter di dalam constructor ini akan mengisi nilai-nilai awal dari atribut seperti `id`, `npm`, `nama`, `semester`, dan `ipk`.

### Getter dan Setter:

Setiap atribut memiliki metode getter dan setter:

- Getter (getId, getName, dll.): Mengambil nilai dari atribut.
- Setter (setId, setName, dll.): Mengubah nilai atribut.

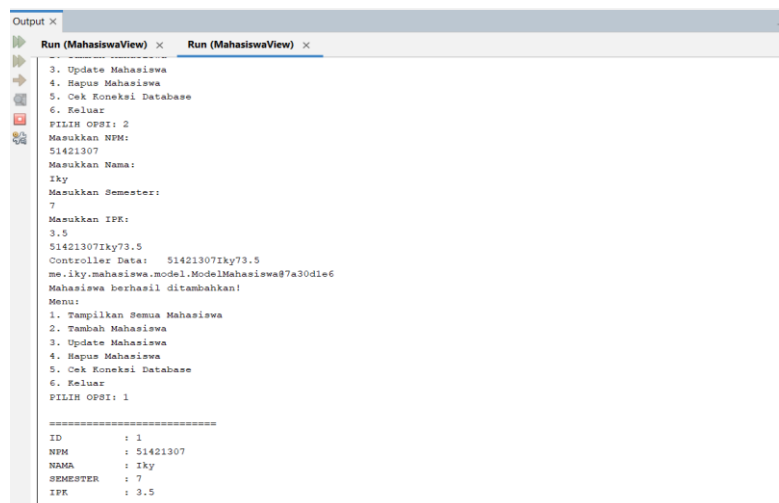
## MahasiswaView



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4  */
5  package me.iky.mahasiswa.view;
6
7  import java.util.Scanner;
8  import me.iky.mahasiswa.controller.MahasiswaController;
9  import me.iky.mahasiswa.model.MahasiswaDAO;
10
11  /**
12   *
13   * @author Riecky
14   */
15  public class MahasiswaView {
16      public static void main(String[] args) {
17          MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
18          MahasiswaController mahasiswaController = new MahasiswaController(mahasiswaDAO);
19
20          Scanner scanner = new Scanner(System.in);
21          int pilihan;
22
23          while (true) {
24              System.out.println("Menu:");
25              System.out.println("1. Tampilkan Semua Mahasiswa");
26              System.out.println("2. Tambah Mahasiswa");
27              System.out.println("3. Update Mahasiswa");
28              System.out.println("4. Hapus Mahasiswa");
29              System.out.println("5. Cek Koneksi Database");
30              System.out.println("6. Keluar");
31              System.out.print("PILIH OPSI: ");
32              pilihan = scanner.nextInt();
33              scanner.nextLine();
34          }
35      }
36  }
```

Selanjutnya untuk codingan diatas yaitu File MahasiswaView.java di dalam proyek Java yang sedang dibuka di NetBeans. File ini adalah bagian dari lapisan View dalam arsitektur Model-View-Controller (MVC), yang digunakan untuk menampilkan antarmuka pengguna dan menerima input dari pengguna.

## Output

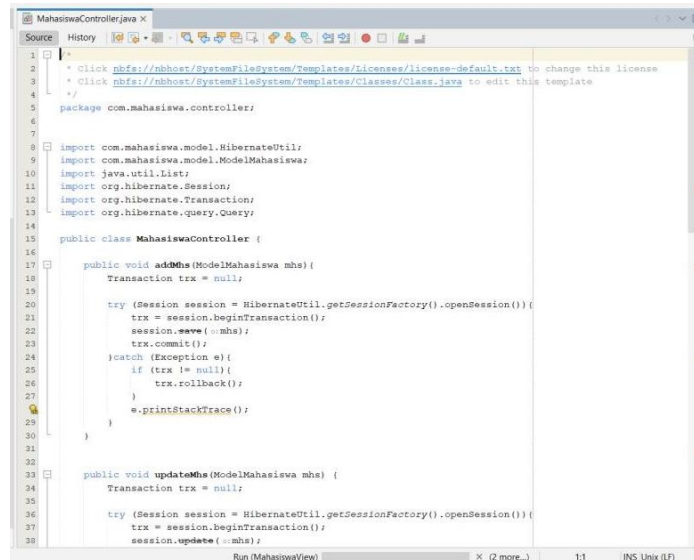


```
Run (MahasiswaView) x Run (MahasiswaView) x
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI: 2
Masukkan NPM:
51421307
Masukkan Nama:
Iky
Masukkan Semester:
7
Masukkan IPK:
3.5
51421307Iky73.5
Controller Data: 51421307Iky73.5
me.iky.mahasiswa.model.ModelMahasiswa$7a30d1e6
Mahasiswa berhasil ditambahkan!
Menu:
1. Tampilkan Semua Mahasiswa
2. Tambah Mahasiswa
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI: 1
=====
ID          : 1
NPM         : 51421307
NAMA        : Iky
SEMESTER    : 7
IPK         : 3.5
=====
```

Jadi selanjutnya untuk outputnya itu seperti diatas disini outputnya itu bisa menambahkan NPM, NAMA, SEMESTER, dan IPK dengan menginput sesuatu kita bisa melihat tampilan dari output tersebut.

## Laporan Akhir 4

### com.mahasiswa.controller



```
1  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
3  */
4
5  package com.mahasiswa.controller;
6
7
8  import com.mahasiswa.model.HibernateUtil;
9  import com.mahasiswa.model.ModelMahasiswa;
10 import java.util.List;
11 import org.hibernate.Session;
12 import org.hibernate.Transaction;
13 import org.hibernate.query.Query;
14
15 public class MahasiswaController {
16
17     public void addMhs(ModelMahasiswa mhs) {
18         Transaction trx = null;
19
20         try (Session session = HibernateUtil.getSessionFactory().openSession()) {
21             trx = session.beginTransaction();
22             session.save(mhs);
23             trx.commit();
24         } catch (Exception e) {
25             if (trx != null) {
26                 trx.rollback();
27             }
28             e.printStackTrace();
29         }
30     }
31
32     public void updateMhs(ModelMahasiswa mhs) {
33         Transaction trx = null;
34
35         try (Session session = HibernateUtil.getSessionFactory().openSession()) {
36             trx = session.beginTransaction();
37             session.update(mhs);
38             trx.commit();
39         } catch (Exception e) {
40             if (trx != null) {
41                 trx.rollback();
42             }
43             e.printStackTrace();
44         }
45     }
46
47     public void deleteMhs(int id) {
48         Transaction trx = null;
49
50         try (Session session = HibernateUtil.getSessionFactory().openSession()) {
51             trx = session.beginTransaction();
52             ModelMahasiswa mhs = session.get(ModelMahasiswa.class, id);
53             if (mhs != null) {
54                 session.delete(mhs);
55                 System.out.println("Berhasil hapus");
56             }
57             trx.commit();
58         } catch (Exception e) {
59             if (trx != null) {
60                 trx.rollback();
61             }
62             e.printStackTrace();
63         }
64     }
65
66 }
67
```

- Selanjutnya untuk line 8 sampai 13 kita melakukan import.
- Untuk line 17-30 kita melakukan Metode addmhs, yang fungsinya untuk menambahkan data mahasiswa baru ke dalam database.
- Trx adalah objek Transaction untuk memastikan bahwa perubahan pada database dijalankan dalam satu transaksi.
- Session membuka koneksi ke database.
- session.save(mhs); digunakan untuk menyimpan objek mhs (data mahasiswa baru) ke dalam database.
- Jika tidak ada kesalahan, trx.commit(); digunakan untuk menyimpan perubahan secara permanen. Jika terjadi kesalahan, transaksi akan di-rollback.



```
33     public void updateMhs(ModelMahasiswa mhs) {
34         Transaction trx = null;
35
36         try (Session session = HibernateUtil.getSessionFactory().openSession()) {
37             trx = session.beginTransaction();
38             session.update(mhs);
39             trx.commit();
40         } catch (Exception e) {
41             if (trx != null) {
42                 trx.rollback();
43             }
44             e.printStackTrace();
45         }
46     }
47
48     public void deleteMhs(int id) {
49         Transaction trx = null;
50
51         try (Session session = HibernateUtil.getSessionFactory().openSession()) {
52             trx = session.beginTransaction();
53             ModelMahasiswa mhs = session.get(ModelMahasiswa.class, id);
54             if (mhs != null) {
55                 session.delete(mhs);
56                 System.out.println("Berhasil hapus");
57             }
58             trx.commit();
59         } catch (Exception e) {
60             if (trx != null) {
61                 trx.rollback();
62             }
63             e.printStackTrace();
64         }
65     }
66
67 }
```



Untuk line 33 sampai dengan 47 ini merupakan Metode updateMhs (Mengupdate Data Mahasiswa):

- updateMhs(ModelMahasiswa mhs): Metode ini digunakan untuk memperbarui informasi mahasiswa di database.
- Di dalamnya, metode ini membuka sesi (koneksi ke database) dan memulai transaksi.
- session.update(mhs); melakukan proses pembaruan data berdasarkan objek mhs yang dikirim sebagai parameter.
- trx.commit(); mengonfirmasi perubahan ke database.
- Jika terjadi kesalahan (Exception), program akan melakukan trx.rollback(); untuk membatalkan perubahan, sehingga data di database tidak berubah.

Dan selanjutnya untuk line 49 sampai 67 ini yaitu Metode deleteMhs (Menghapus Data Mahasiswa):

- deleteMhs(int id): Metode ini digunakan untuk menghapus data mahasiswa berdasarkan ID.
- Mirip dengan updateMhs, metode ini juga membuka sesi dan memulai transaksi.
- session.get(ModelMahasiswa.class, id); mencari data mahasiswa berdasarkan ID yang diberikan. Jika ditemukan, data tersebut akan dihapus dengan session.delete(mhs);.
- trx.commit(); mengonfirmasi penghapusan ke database, dan mencetak pesan "Berhasil hapus" di konsol.
- Jika terjadi kesalahan, trx.rollback(); akan membatalkan penghapusan.

```
public List<ModelMahasiswa> getAllMahasiswa() {
    Transaction trx = null;
    List<ModelMahasiswa> listMhs = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()) {
        trx = session.beginTransaction();
        // Using HQL (Hibernate Query Language) to fetch all records
        Query<ModelMahasiswa> query = session.createQuery("from ModelMahasiswa", type: ModelMahasiswa.class);
        listMhs = query.list(); // Fetch all results

        trx.commit(); // Commit transaction
    } catch (Exception e) {
        if (trx != null) {
            trx.rollback(); // Rollback transaction in case of error
        }
        e.printStackTrace();
    }

    // Return the fetched list
    return listMhs;
}
```

Untuk selanjutnya disini terdapat Fungsi getAllMahasiswa() ini merupakan bagian dari sistem yang bertugas mengambil data mahasiswa dari database secara efisien, menangani error, dan memastikan bahwa hanya transaksi yang valid yang diterima ke dalam database.



## HibernateUtil.java

```
Source History
1 package com.mahasiswa.model;
2
3 import org.hibernate.Session;
4 import org.hibernate.SessionFactory;
5 import org.hibernate.cfg.Configuration;
6
7 public class HibernateUtil {
8     private static SessionFactory sessionFactory;
9
10    static {
11        try {
12            // Create the SessionFactory from hibernate.cfg.xml
13            sessionFactory = new Configuration().configure().buildSessionFactory();
14        } catch (Throwable ex) {
15            // Make sure you log the exception, as it might be swallowed
16            System.err.println("Initial SessionFactory creation failed." + ex);
17            throw new ExceptionInInitializerError(ex);
18        }
19    }
20
21    public static SessionFactory getSessionFactory() {
22        return sessionFactory;
23    }
24
25    public static void testConnection() {
26        try (Session session = sessionFactory.openSession()) {
27            System.out.println("Connection to the database was successful!");
28        } catch (Exception e) {
29            System.err.println("Failed to connect to the database.");
30            e.printStackTrace();
31        }
32    }
33 }
```

Selanjutnya untuk Kode ini bertujuan untuk mengatur koneksi ke database menggunakan Hibernate, sehingga kelas-kelas lain dalam proyek bisa dengan mudah membuka dan menutup sesi koneksi tanpa perlu membuat ulang konfigurasi. Nah pada Kelas HibernateUtil ini adalah kelas utilitas yang menyediakan konfigurasi dan manajemen sesi database menggunakan Hibernate. Ini memastikan bahwa SessionFactory hanya dibuat sekali untuk efisiensi, dan menyediakan metode testConnection() untuk memastikan koneksi ke database berfungsi dengan baik.

## ModelMahasiswa.java

```
10 public class ModelMahasiswa {
11
12     @Id
13     @GeneratedValue(strategy = GenerationType.IDENTITY)
14     @Column(name = "id")
15     private int id;
16
17     @Column(name = "npm", nullable = false, length = 8)
18     private String npm;
19
20     @Column(name = "nama", nullable = false, length = 50)
21     private String nama;
22
23     @Column(name = "semester")
24     private int semester;
25
26     @Column(name = "ipk")
27     private float ipk;
28
29     public ModelMahasiswa() {
30
31     }
32
33     public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
34         this.id = id;
35         this.npm = npm;
36         this.nama = nama;
37         this.semester = semester;
38         this.ipk = ipk;
39     }
40
41     public int getId() {
42         return id;
43     }
44
45     public void setId(int id) {
46         this.id = id;
47     }
48 }
```

Dan untuk gambar diatas ini yaitu Kelas ModelMahasiswa ini adalah representasi dari entitas mahasiswa dalam database. Kelas ini menggunakan anotasi JPA (Java Persistence API) untuk menentukan pengaturan kolom-kolom pada tabel database yang menyimpan data mahasiswa. Jadi pada Kelas ModelMahasiswa ini mewakili data mahasiswa dan berfungsi sebagai model data yang dapat dipetakan ke tabel database. Dengan getter dan setter, kelas ini memungkinkan pengambilan dan perubahan data mahasiswa dengan aman.

## ModelTabelMahasiswa.java

```
public class ModelTabelMahasiswa extends AbstractTableModel{
    private List<ModelMahasiswa> mahasiswaList;
    private String[] columnNames = {"ID", "NPM", "Nama", "Semester", "IPK"};

    public ModelTabelMahasiswa(List<ModelMahasiswa> mahasiswaList) {
        this.mahasiswaList = mahasiswaList;
    }

    @Override
    public int getRowCount() {
        return mahasiswaList.size(); // Jumlah baris sesuai dengan jumlah data mahasiswa
    }

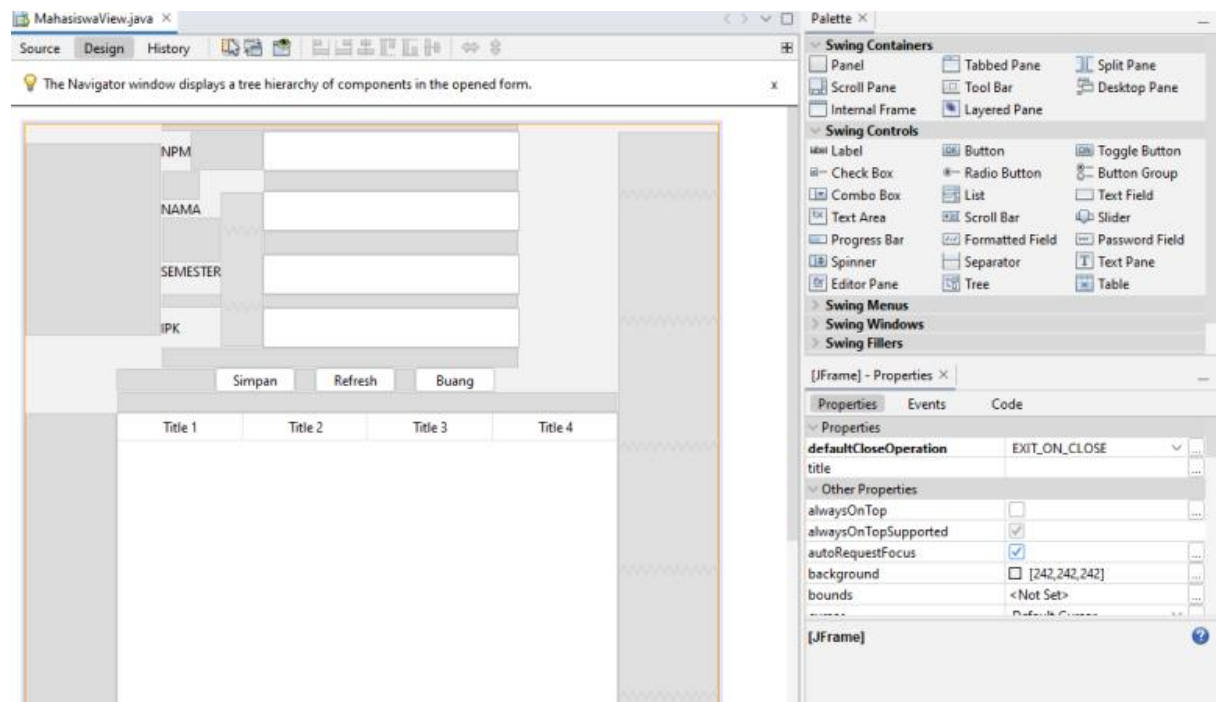
    @Override
    public int getColumnCount() {
        return columnNames.length; // Jumlah kolom sesuai dengan jumlah elemen dalam columnNames
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        ModelMahasiswa mahasiswa = mahasiswaList.get(rowIndex);
        switch (columnIndex) {
            case 0:
                return mahasiswa.getId();
            case 1:
                return mahasiswa.getNpm();
            case 2:
                return mahasiswa.getNama();
            case 3:
                return mahasiswa.getSemester();
            case 4:
                return mahasiswa.getIpk();
            default:
                return null;
        }
    }
}
```

Jadi pada codingan diatas menunjukkan sebuah kode Java yang membuat model tabel untuk menampilkan data mahasiswa dalam aplikasi GUI (biasanya dengan *JTable*). Kelas yang dibuat ini bernama ModelTableMahasiswa dan memperpanjang (extends) AbstractTableModel, yang merupakan kelas bawaan Java untuk mengelola data yang ingin ditampilkan dalam tabel.

Jadi pada gambar diatas merupakan kelas ModelTableMahasiswa berfungsi sebagai model data untuk tabel yang menampilkan informasi mahasiswa dalam aplikasi GUI. Kelas ini mengatur jumlah baris berdasarkan jumlah data mahasiswa yang ada dan jumlah kolom berdasarkan atribut mahasiswa (ID, NPM, Nama, Semester, IPK). Metode getValueAt mengatur data yang akan ditampilkan di setiap sel berdasarkan posisi baris dan kolomnya. Intinya, kelas ini mempermudah pengaturan dan penyajian data mahasiswa dalam bentuk tabel pada aplikasi Java.

## MahasiswaView.java



Merupakan tampilan JFrame Design Form yang sudah dibuat.

## OUTPUT

The screenshot shows the output of the application. The form displays the input values: NPM (51421307), NAMA (Riezky), SEMESTER (7), and IPK (37). The buttons 'Simpan', 'Refresh', and 'Buang' are visible. Below the buttons is a table with columns ID, NPM, Nama, Semester, and IPK, containing one row of data: ID 3, NPM 51421307, Nama Riezky, Semester 7, and IPK 37.0.

ID	NPM	Nama	Semester	IPK
3	51421307	Riezky	7	37.0

Dan menghasilkan sebuah output seperti gambar yang diatas ini.