

LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 3

Tanggal : 29 Oktober 2024

Materi : Konsep Model – View – Controller (MVC),
Pembuatan Program dengan Konsep MVC, Instalasi & Penerapan, Dan
Akses Database dengan JDBC

NPM : 50421130

Nama : Alvi Haikal Farwiza

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

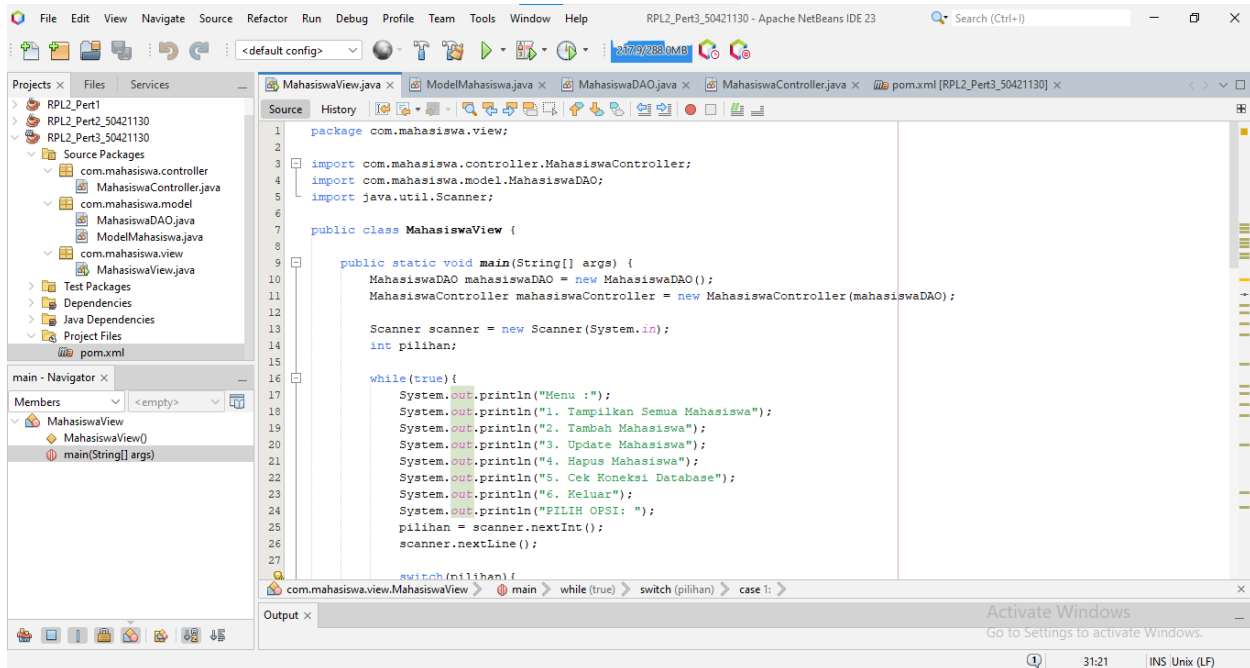
Jumlah Lembar : 13 Lembar

**LABORATORIUM TEKNIK INFORMATIKA
UNIVERSITAS GUNADARMA
2024**

Jelaskan satu per satu codingan kalian dari hasil screenshot activity!

1. File Code Mahasiswa.java

File ini adalah file dari View dalam arsitektur MVC yang bertanggung jawab untuk antarmuka pengguna



Fungsi main adalah fungsi utama yang menjalankan program. Inisialisasi MahasiswaDAO, MahasiswaController, dan Scanner untuk input dari user. Menyediakan menu pilihan untuk user dan mengarahkan ke fungsi yang sesuai di MahasiswaController.

```
6
7 public class MahasiswaView {
8
9     public static void main(String[] args) {
10         MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
11         MahasiswaController mahasiswaController = new MahasiswaController(mahasiswaDAO);
12     }
```

Tampilkan Semua Mahasiswa (case 1): Memanggil displayAllMahasiswa() di MahasiswaController untuk menampilkan seluruh data mahasiswa.

```
switch(pilihan) {
    case 1:
        mahasiswaController.displayAllMahasiswa();
        break;
```

Tambah Mahasiswa (case 2): Meminta pengguna memasukkan data mahasiswa (NPM, Nama, Semester, IPK), lalu memanggil addMahasiswa() di MahasiswaController.

```
case 2:
    System.out.println("Masukan NPM: ");
    String npm = scanner.next();
    System.out.println("Masukan Nama: ");
    String nama = scanner.next();
    System.out.println("Masukan Semester: ");
    int semester = scanner.nextInt();
    System.out.println("Masukan IPK: ");
    float ipk = scanner.nextFloat();

    mahasiswaController.addMahasiswa(npm, nama, semester, ipk);
    break;
```

Update Mahasiswa (case 3): Meminta pengguna memasukkan ID mahasiswa, NPM, Nama, Semester, dan IPK baru, lalu memanggil updateMahasiswa() di MahasiswaController.

```
case 3:
    System.out.println("Masukan ID Mahasiswa: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    System.out.println("Masukan NPM: ");
    String npmBaru = scanner.next();
    System.out.println("Masukan Nama: ");
    String namaBaru = scanner.next();
    System.out.println("Masukan Semester: ");
    int semesterBaru = scanner.nextInt();
    System.out.println("Masukan IPK: ");
    float ipkBaru = scanner.nextFloat();

    mahasiswaController.updateMahasiswa(id, npmBaru, namaBaru, semesterBaru, ipkBaru);
    break;
```

Hapus Mahasiswa (case 4): Meminta pengguna memasukkan ID mahasiswa yang ingin dihapus, lalu memanggil deleteMahasiswa() di MahasiswaController.

```
case 4:
    System.out.println("Masukan ID Mahasiswa yg Ingin Dihapus: ");
    int idHapus = scanner.nextInt();
    mahasiswaController.deleteMahasiswa(idHapus);
    break;
```

Cek Koneksi Database (case 5): Memanggil checkDatabaseConnection() di MahasiswaController untuk memeriksa status koneksi ke database.

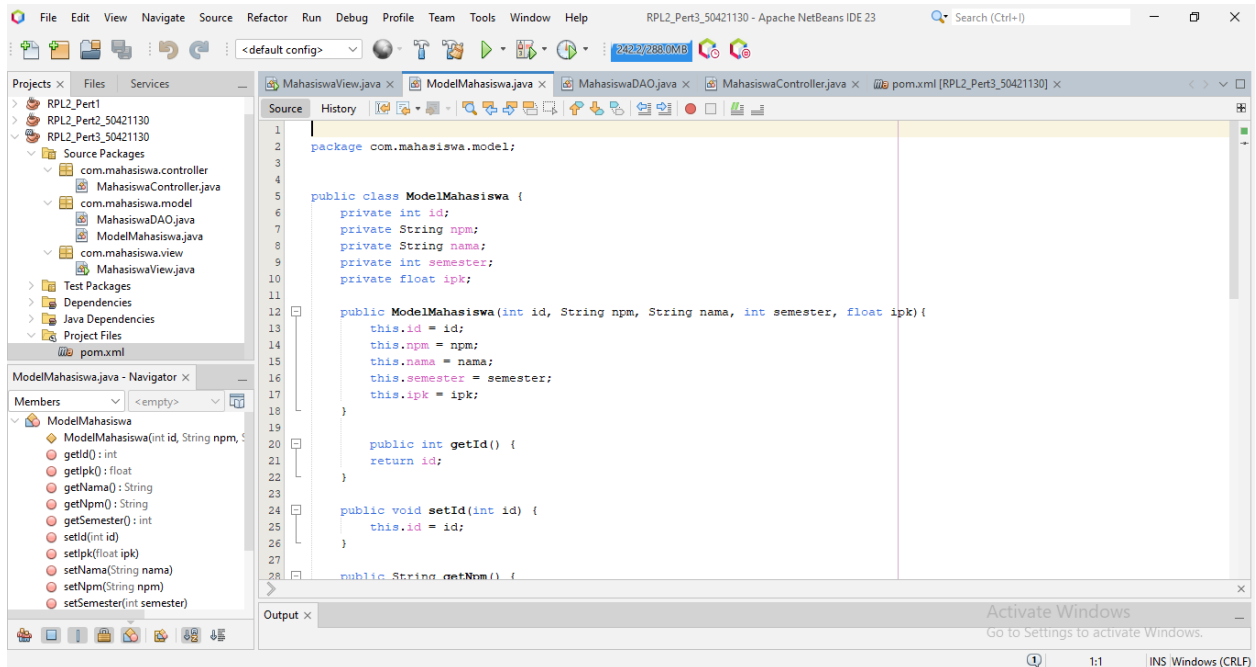
```
case 5:
    mahasiswaController.checkDatabaseConnection();
    break;
```

Keluar (case 6): Menutup koneksi database menggunakan `closeConnection()` di `MahasiswaController`, lalu mengakhiri program.

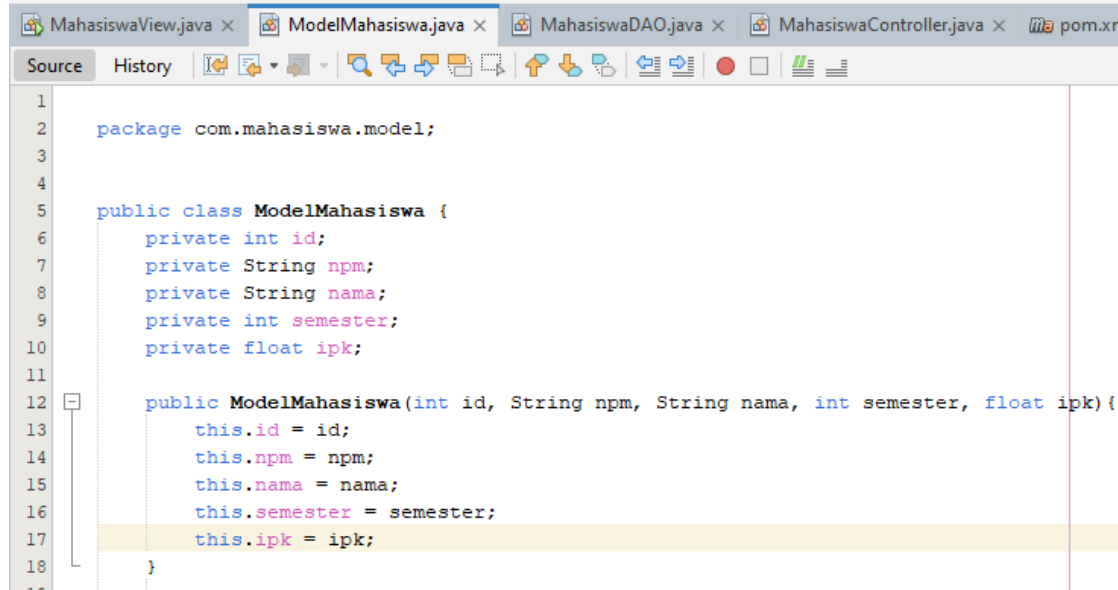
```
case 6:
    mahasiswaController.closeConnection();
    System.out.println("Program selesai");
    return;
default:
    System.out.println("Input Tidak Valid");
```

2. File Code ModelMahasiswa.java (Model)

Model Mahasiswa adalah model mendefinisikan struktur data mahasiswa, termasuk atribut yang ada di database (id, npm, nama, semester, ipk).



Constructor ModelMahasiswa(int id, String npm, String nama, int semester, float ipk):
Menginisialisasi objek ModelMahasiswa dengan ID, NPM, Nama, Semester, dan IPK.



```
1 package com.mahasiswa.model;
2
3
4
5 public class ModelMahasiswa {
6     private int id;
7     private String npm;
8     private String nama;
9     private int semester;
10    private float ipk;
11
12    public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk){
13        this.id = id;
14        this.npm = npm;
15        this.nama = nama;
16        this.semester = semester;
17        this.ipk = ipk;
18    }
19 }
```

Getter dan Setter: Digunakan untuk mengakses dan memodifikasi nilai dari atribut:

- **getId() dan setId(int id):** Mengakses atau mengubah ID mahasiswa.

```
] public int getId() {
-     return id;
- }

] public void setId(int id) {
-     this.id = id;
- }
```

- **getNpm() dan setNpm(String npm):** Mengakses atau mengubah NPM mahasiswa.

```
public String getNpm() {
    return npm;
}

public void setNpm(String npm) {
    this.npm = npm;
}
```

- **getNama() dan setNama(String nama):** Mengakses atau mengubah nama mahasiswa.

```

public String getNama() {
    return nama;
}

public void setNama(String nama) {
    this.nama = nama;
}

```

- **getSemester() dan setSemester(int semester):** Mengakses atau mengubah semester mahasiswa.

```

public int getSemester() {
    return semester;
}

public void setSemester(int semester) {
    this.semester = semester;
}

```

- **getIpk() dan setIpk(float ipk):** Mengakses atau mengubah IPK mahasiswa.

```

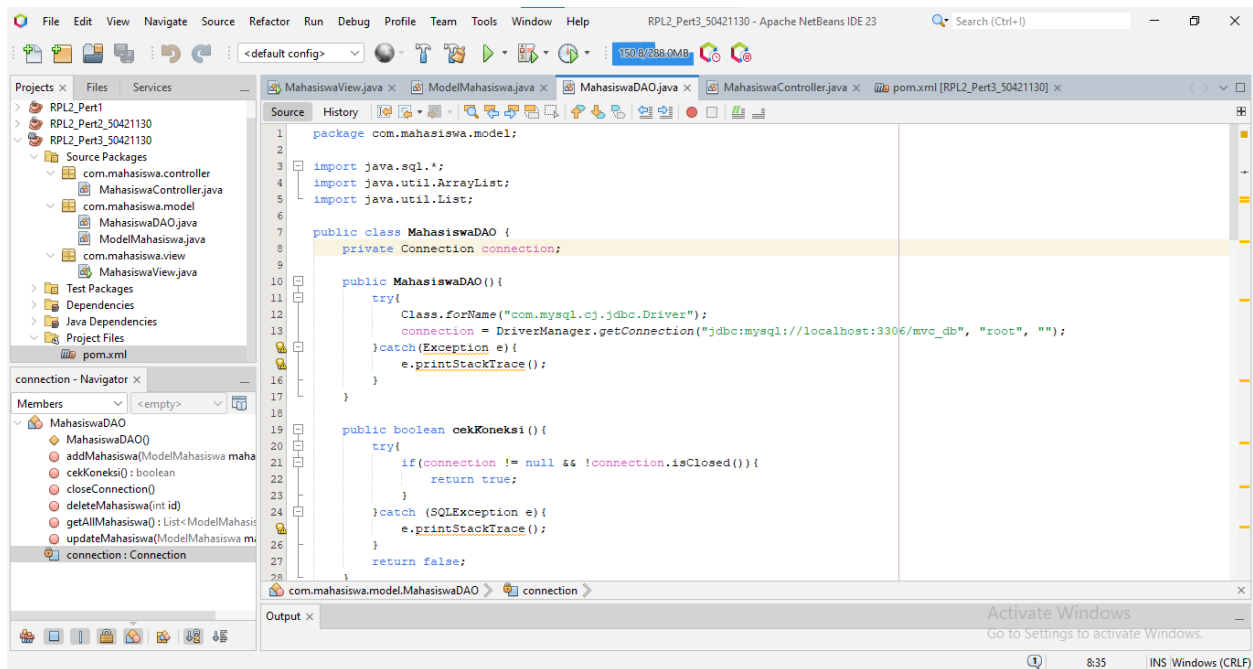
    public float getIpk() {
        return ipk;
    }

    public void setIpk(float ipk) {
        this.ipk = ipk;
    }
}

```

3. File Code MahasiswaDAO.java :

File MahasiswaDAO adalah file yang bertanggung jawab untuk berinteraksi dengan database.



Constructor MahasiswaDAO(): Menginisialisasi koneksi ke database menggunakan JDBC Driver dan URL yang disediakan. Jika koneksi gagal, akan mencetak stack trace dari exception.

```
public class MahasiswaDAO {  
    private Connection connection;  
  
    public MahasiswaDAO() {  
        try {  
            Class.forName("com.mysql.cj.jdbc.Driver");  
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mvc_db", "root", "");  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

cekKoneksi(): Memeriksa apakah koneksi database berhasil dilakukan atau tidak. Mengembalikan nilai true jika terkoneksi, dan false jika tidak.

```
public boolean cekKoneksi() {  
    try {  
        if (connection != null && !connection.isClosed()) {  
            return true;  
        }  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return false;  
}
```

addMahasiswa(ModelMahasiswa mahasiswa): Menambahkan data mahasiswa baru ke dalam tabel mahasiswa di database dengan menggunakan pernyataan SQL INSERT INTO. Mengambil NPM, Nama, Semester, dan IPK dari objek ModelMahasiswa.

```
public void addMahasiswa(ModelMahasiswa mahasiswa){
    String sql = "INSERT INTO mahasiswa (npm, nama, semester, ipk) VALUES (?, ?, ?, ?)";
    try{
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setString(1, mahasiswa.getNpm());
        pstmt.setString(2, mahasiswa.getNama());
        pstmt.setInt(3, mahasiswa.getSemester());
        pstmt.setFloat(4, mahasiswa.getIpk());
        pstmt.executeUpdate();
    } catch (SQLException e){
        e.printStackTrace();
    }
}
```

getAllMahasiswa(): Mengambil seluruh data mahasiswa dari tabel mahasiswa di database. Menyimpan setiap baris hasil query sebagai objek ModelMahasiswa dan mengembalikannya dalam bentuk List<ModelMahasiswa>.

```
public List<ModelMahasiswa> getAllMahasiswa(){
    List<ModelMahasiswa> mahasiswaList = new ArrayList<>();
    String sql = "SELECT * FROM mahasiswa";
    try{
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        while(rs.next()){
            mahasiswaList.add(new ModelMahasiswa(
                rs.getInt("id"),
                rs.getString("npm"),
                rs.getString("nama"),
                rs.getInt("semester"),
                rs.getFloat("ipk")
            ));
        }
    } catch (SQLException e){
        e.printStackTrace();
    }
    return mahasiswaList;
}
```

updateMahasiswa(ModelMahasiswa mahasiswa): Memperbarui data mahasiswa berdasarkan ID dengan nilai baru dari atribut NPM, Nama, Semester, dan IPK yang diambil dari ModelMahasiswa. Menggunakan SQL UPDATE.


```

public void updateMahasiswa(ModelMahasiswa mahasiswa){
    String sql = "UPDATE mahasiswa SET npm = ?, nama = ?, semester = ?, ipk = ? WHERE id = ?";
    try{
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setString(1, mahasiswa.getNpm());
        pstmt.setString(2, mahasiswa.getNama());
        pstmt.setInt(3, mahasiswa.getSemester());
        pstmt.setFloat(4, mahasiswa.getIpk());
        pstmt.setInt(5, mahasiswa.getId());
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

deleteMahasiswa(int id): Menghapus data mahasiswa dari database berdasarkan ID mahasiswa dengan pernyataan SQL DELETE FROM.

```

public void deleteMahasiswa(int id){
    String sql = "DELETE FROM mahasiswa WHERE id = ?";
    try{
        PreparedStatement pstmt = connection.prepareStatement(sql);
        pstmt.setInt(1, id);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

closeConnection(): Menutup koneksi ke database jika koneksi tidak kosong, untuk memastikan koneksi tidak tetap terbuka saat aplikasi berhenti.

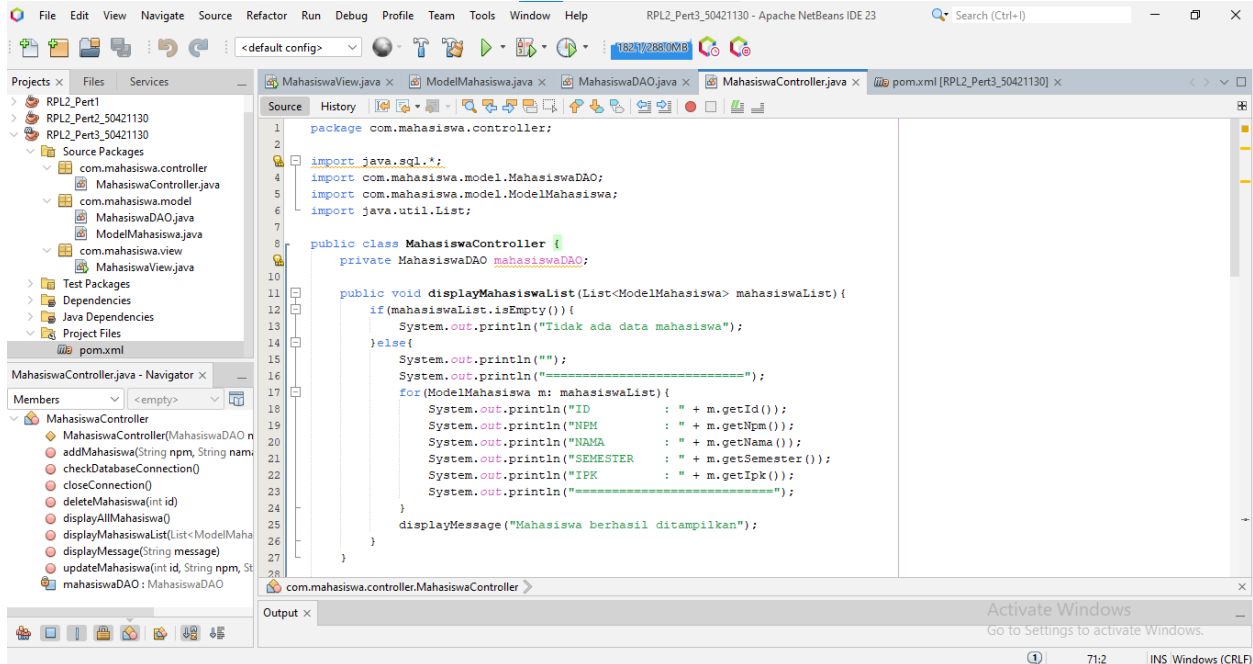
```

public void closeConnection(){
    try{
        if(connection != null){
            connection.close();
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

4. File Code MahasiswaController.java

MahasiswaController adalah Controller yang bertindak sebagai penghubung antara MahasiswaView dan MahasiswaDAO, yang menerima input dari MahasiswaView, memprosesnya, dan memanggil metode yang relevan di MahasiswaDAO.



Constructor MahasiswaController(MahasiswaDAO mahasiswaDAO): Menginisialisasi MahasiswaController dengan MahasiswaDAO.

```
public class MahasiswaController {
```

displayMahasiswaList(List<ModelMahasiswa> mahasiswaList): Menampilkan daftar mahasiswa. Jika mahasiswaList kosong, akan mencetak "Tidak ada data mahasiswa". Jika tidak kosong, mencetak detail setiap mahasiswa di daftar, seperti ID, NPM, Nama, Semester, dan IPK.

```
public void displayMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
    if (mahasiswaList.isEmpty()) {
        System.out.println("Tidak ada data mahasiswa");
    } else {
        System.out.println("");
        System.out.println("=====");
        for (ModelMahasiswa m: mahasiswaList) {
            System.out.println("ID       : " + m.getId());
            System.out.println("NPM      : " + m.getNpm());
            System.out.println("NAMA     : " + m.getNama());
            System.out.println("SEMESTER : " + m.getSemester());
            System.out.println("IPK      : " + m.getIpk());
            System.out.println("=====");
        }
        displayMessage("Mahasiswa berhasil ditampilkan");
    }
}
```

displayMessage(String message): Menampilkan pesan di konsol, berguna untuk notifikasi kepada pengguna.

```
]      public void displayMessage(String message) {  
-      System.out.println(message);  
      }
```

Constructor MahasiswaController(MahasiswaDAO mahasiswaDAO): Menginisialisasi MahasiswaController dengan MahasiswaDAO.

```
[      public MahasiswaController(MahasiswaDAO mahasiswaDAO) {  
      this.mahasiswaDAO = mahasiswaDAO;  
      }
```

checkDatabaseConnection(): Memanggil cekKoneksi() dari MahasiswaDAO untuk memeriksa apakah koneksi database berhasil atau gagal, dan menampilkan pesan berdasarkan hasil.

```
[      public void checkDatabaseConnection() {  
-      boolean isConnected = mahasiswaDAO.cekKoneksi();  
-      if (isConnected) {  
-          displayMessage("Koneksi ke db berhasil");  
-      } else {  
-          displayMessage("Koneksi DB gagal");  
-      }  
      }
```

displayAllMahasiswa(): Mengambil seluruh data mahasiswa dari MahasiswaDAO dan menampilkannya melalui displayMahasiswaList().

```
[      public void displayAllMahasiswa() {  
      List<ModelMahasiswa> mahasiswaList = mahasiswaDAO.getAllMahasiswa();  
      displayMahasiswaList(mahasiswaList);  
      }
```

addMahasiswa(String npm, String nama, int semester, float ipk): Membuat objek ModelMahasiswa baru dengan data mahasiswa yang diberikan, lalu menambahkannya ke database melalui addMahasiswa() di MahasiswaDAO.

```
      public void addMahasiswa(String npm, String nama, int semester, float ipk) {  
      ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(0, npm, nama, semester, ipk);  
      mahasiswaDAO.addMahasiswa(mahasiswaBaru);  
      displayMessage("Mahasiswa berhasil ditambahkan");  
      }
```

updateMahasiswa(int id, String npm, String nama, int semester, float ipk): Membuat objek ModelMahasiswa dengan data yang diperbarui berdasarkan ID mahasiswa, lalu memanggil updateMahasiswa() di MahasiswaDAO untuk memperbarui data di database.

```
public void updateMahasiswa(int id, String npm, String nama, int semester, float ipk){
    ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(id, npm, nama, semester, ipk);
    mahasiswaDAO.updateMahasiswa(mahasiswaBaru);
    displayMessage("Mahasiswa berhasil diperbaharui");
}
```

deleteMahasiswa(int id): Menghapus data mahasiswa dari database berdasarkan ID yang diberikan, dengan memanggil deleteMahasiswa() di MahasiswaDAO.

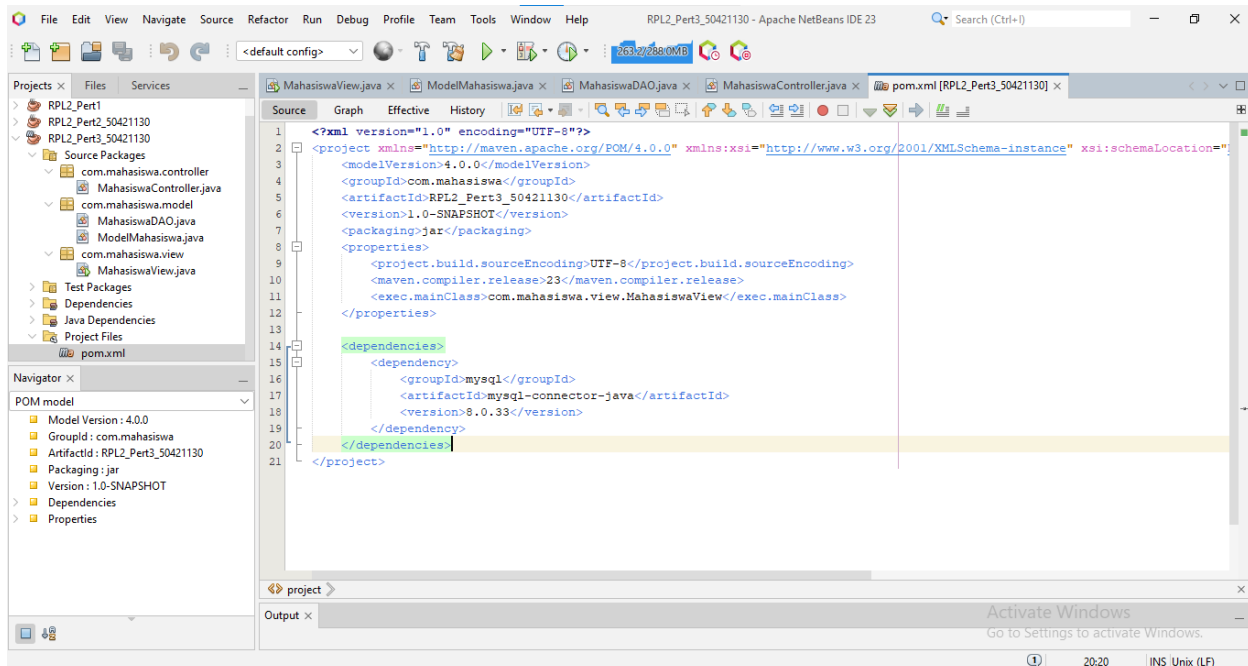
```
public void deleteMahasiswa(int id){
    mahasiswaDAO.deleteMahasiswa(id);
    displayMessage("Mahasiswa Berhasil Dihapus!");
}
```

closeConnection(): Menutup koneksi database dengan memanggil closeConnection() di MahasiswaDAO.

```
public void closeConnection(){
    mahasiswaDAO.closeConnection();
}
```

5. Code File Pom.xml

File pom.xml adalah konfigurasi untuk proyek Maven yang berisi informasi tentang proyek, versi dependensi, dan pengaturan kompilasi.



mysql-connector-java: Dependensi untuk konektor JDBC MySQL dengan versi 8.0.33, yang memungkinkan aplikasi terhubung ke database MySQL.

A screenshot of an IDE's code editor showing an XML snippet. On the left, a vertical scrollbar and line numbers are visible, with lines 3 through 10 shown. The code is as follows:

```
3  
4 <dependencies>  
5   <dependency>  
6     <groupId>mysql</groupId>  
7     <artifactId>mysql-connector-java</artifactId>  
8     <version>8.0.33</version>  
9   </dependency>  
10 </dependencies>
```

The opening tag `<dependencies>` is highlighted in green. The closing tag `</dependencies>` is also highlighted in green. The entire code block is set against a light yellow background.