

ACTIVITY PERTEMUAN 3

Nama : Aditya Rachmansyah Hamid

Kelas : 4IA28

NPM : 50421048

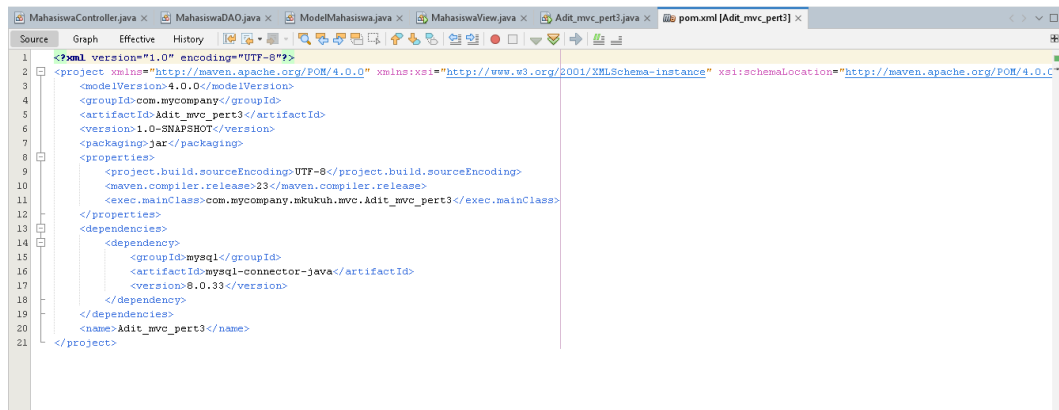
Mata Praktikum : Rekayasa Perangkat Lunak 2

Soal:

Mengikuti instruksi dari PJ

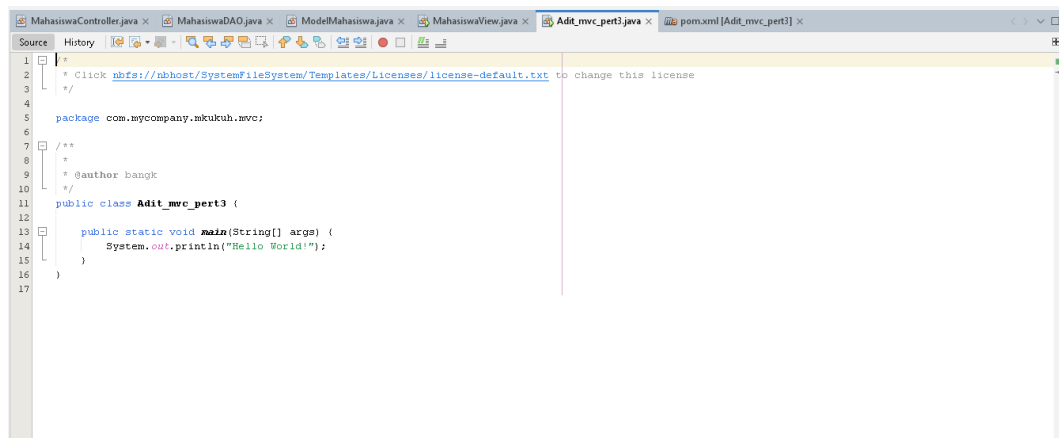
Jawab:

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" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
3 <modelVersion>4.0.0</modelVersion>
4 <groupId>com.mycompany</groupId>
5 <artifactId>Adit_mvc_pert3</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.release>23</maven.compiler.release>
11 <exec.mainClass>com.mycompany.mkukuh.mvc.Adit_mvc_pert3</exec.mainClass>
12 </properties>
13 <dependencies>
14 <dependency>
15 <groupId>mysql</groupId>
16 <artifactId>mysql-connector-java</artifactId>
17 <version>8.0.33</version>
18 </dependency>
19 </dependencies>
20 <name>Adit_mvc_pert3</name>
21 </project>
```

Main class: Adit_mvc_pert3.java



```
1
2 * Click https://nhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 */
4
5 package com.mycompany.mkukuh.mvc;
6
7 /**
8  *
9  * @author bangk
10 */
11 public class Adit_mvc_pert3 {
12
13     public static void main(String[] args) {
14         System.out.println("Hello World!");
15     }
16 }
17
```

MahasiswaView.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package com.mahasiswa.view;

import com.mahasiswa.controller.MahasiswaController;
import com.mahasiswa.model.MahasiswaDAO;
import java.util.Scanner;

/**
 *
 * @author Aditrhamid
 */

public class MahasiswaView {

    public static void main(String[] args){

        MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();

        MahasiswaController mahasiswaController = new
MahasiswaController(mahasiswaDAO);

        Scanner scanner = new Scanner(System.in);

        int pilihan;

        while(true){

            System.out.println("Menu:");

            System.out.println("1. Tampilkan Semua Mahasiswa");

            System.out.println("2. Tambah Mahasiswa");

            System.out.println("3. Update Mahasiswa");
```

```
System.out.println("4. Hapus Mahasiswa");
System.out.println("5. Cek Koneksi Database");
System.out.println("6. Keluar");
System.out.print("PILIH OPSI: ");
pilihan = scanner.nextInt();
scanner.nextLine();

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

    case 2:
        // tambah mhs
        System.out.println("Masukkan NPM: ");
        String npm = scanner.next();
        System.out.println("Masukkan Nama: ");
        String nama = scanner.next();
        System.out.println("Masukkan Semester: ");
        int semester = scanner.nextInt();
        System.out.println("Masukkan IPK: ");
        float ipk = scanner.nextFloat();
        System.out.println(npm + nama + semester + ipk);
        mahasiswaController.addMahasiswa(npm, nama, semester, ipk);
        break;

    case 3:
        System.out.print("Masukkan ID mahasiswa: ");
        int id = scanner.nextInt();
        scanner.nextLine();
```

```

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

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

        break;
    case 4:
        System.out.print("Masukkan ID Mahasiswa: ");
        int idHapus = scanner.nextInt();
        mahasiswaController.deleteMahasiswa(idHapus);
    case 5:
        mahasiswaController.checkDatabaseConnection();
        break;
    case 6:
        // Keluar
        mahasiswaController.closeConnection();
        System.out.println("Program selesai.");
        return;
    default:
        System.out.println("Input Tidak valid");
    }
}
}
}
}

```

ModelMahasiswa.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package com.mahasiswa.model;

/**
 *
 * @author Aditrhamid
 */
public class ModelMahasiswa {
    private int id;
    private String npm;
    private String nama;
    private int semester;
    private float ipk;

    public ModelMahasiswa(int id, String npm, String nama, int semester, float
ipk){
        this.id = id;
        this.npm = npm;
        this.nama = nama;
        this.semester = semester;
        this.ipk = ipk;
    }

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

```
}

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

public String getNpm() {
    return npm;
}

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

public String getNama() {
    return nama;
}

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

public int getSemester() {
    return semester;
}

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

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

MahasiswaDAO.java:

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt  
to change this license  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit  
this template  
 */  
package com.mahasiswa.model;  
  
import java.sql.*;  
import java.util.ArrayList;  
import java.util.List;  
/**  
 *  
 * @author Aditrhamid  
 */  
public class MahasiswaDAO {  
    private Connection connection;  
  
    public MahasiswaDAO () {
```

```

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            connection
            DriverManager.getConnection("jdbc:mysql://localhost:3306/kukuh_mvc",
            "root", "");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

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

    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();
    }
}

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;
}

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());
        pstmt.executeUpdate();
    } catch(SQLException e){
        e.printStackTrace();
    }
}

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();
    }
}

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

```

MahasiswaController.java:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package com.mahasiswa.controller;

import com.mahasiswa.model.MahasiswaDAO;
import com.mahasiswa.model.ModelMahasiswa;
import java.util.List;
/**
 *
 * @author Aditrhamid
 */
public class MahasiswaController {

    private final MahasiswaDAO mahasiswaDAO;

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

    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("=====");
    }
}
}

```

```

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

```

```

public void checkDatabaseConnection(){
    boolean isConnected = mahasiswaDAO.checkConnection();
    if (isConnected){
        displayMessage("Koneksi ke db berhasil");
    } else{
        displayMessage("Koneksi DB Gagal");
    }
}

```

// READ ALL (Menampilkan semua mahasiswa)

```

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

```

```
        displayMahasiswaList(mahasiswaList);
    }

    public void addMahasiswa(String npm, String nama, int semester, float ipk){
        ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(0, npm, nama,
semester, ipk);

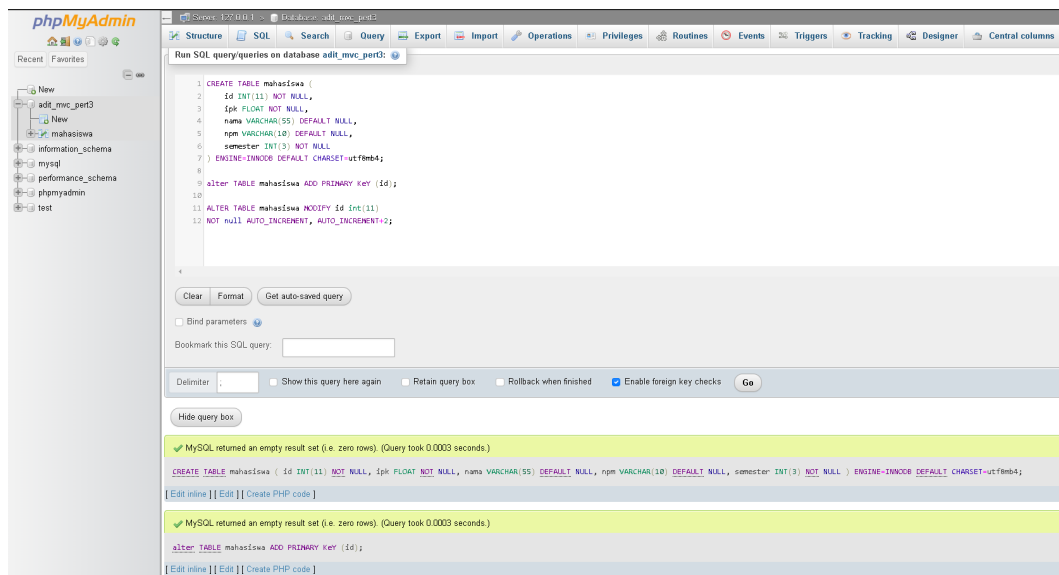
        System.out.println("Controller Data: " + npm + nama + semester + ipk);
        System.out.println(mahasiswaBaru);
        mahasiswaDAO.addMahasiswa(mahasiswaBaru);
        displayMessage("Mahasiswa berhasil ditambahkan!");
    }

    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 diperbarui!");
    }

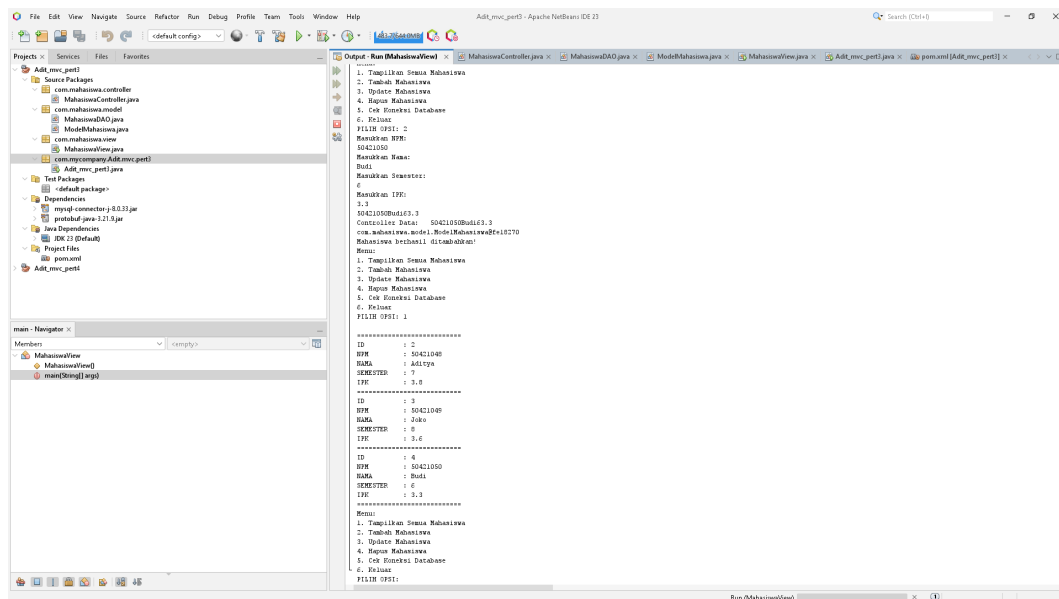
    public void deleteMahasiswa(int id){
        mahasiswaDAO.deleteMahasiswa(id);
        displayMessage("Mahasiswa Berhasil Dihapus!");
    }

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

MySQL Query:



Output MahasiswaView.java:



Penjelasan singkat dari setiap file di atas:

1. MahasiswaController.java

MahasiswaController adalah kelas kontrol yang berfungsi untuk mengelola alur data antara tampilan dan model. Kelas ini menyediakan berbagai metode, seperti:

1. **displayAllMahasiswa():** Menampilkan daftar seluruh mahasiswa.
2. **addMahasiswa():** Menambah data mahasiswa baru.
3. **updateMahasiswa():** Memperbarui data mahasiswa yang sudah ada.

4. **deleteMahasiswa()**: Menghapus data mahasiswa.
5. **checkDatabaseConnection()**: Mengecek koneksi ke database.
6. **closeConnection()**: Menutup koneksi database. Controller ini juga menggunakan MahasiswaDAO untuk akses database dan ModelMahasiswa sebagai representasi data mahasiswa.

2. MahasiswaDAO.java

MahasiswaDAO adalah kelas Data Access Object (DAO) yang mengelola semua interaksi dengan database. Metode utama di kelas ini meliputi:

1. **checkConnection()**: Mengecek apakah koneksi database berhasil.
2. **addMahasiswa()**: Menambahkan mahasiswa ke database.
3. **getAllMahasiswa()**: Mengambil semua data mahasiswa dari database.
4. **updateMahasiswa()**: Memperbarui data mahasiswa.
5. **deleteMahasiswa()**: Menghapus mahasiswa berdasarkan ID.
6. **closeConnection()**: Menutup koneksi dengan database.

3. ModelMahasiswa.java

ModelMahasiswa adalah kelas model yang merepresentasikan entitas mahasiswa. Kelas ini memiliki atribut seperti id, npm, nama, semester, dan ipk, serta getter dan setter untuk setiap atribut. Kelas ini berfungsi sebagai wadah data untuk objek mahasiswa yang akan disimpan atau diambil dari database.

4. MahasiswaView.java

MahasiswaView adalah tampilan utama aplikasi yang menyediakan antarmuka konsol untuk interaksi dengan pengguna. Pengguna dapat:

1. Menampilkan daftar mahasiswa,
2. Menambah, memperbarui, dan menghapus data mahasiswa,
3. Mengecek koneksi database,
4. Keluar dari program. Tampilan ini berkomunikasi dengan MahasiswaController untuk melakukan operasi pada data mahasiswa.

5. Adit_mvc_pert3.java

Adit_mvc_pert3 adalah kelas utama dari aplikasi, berfungsi sebagai entry point atau titik awal program.

Kombinasi dari file ini membentuk aplikasi berbasis **Model-View-Controller (MVC)** yang terstruktur, dengan ModelMahasiswa sebagai model data, MahasiswaController sebagai pengontrol data dan alur logika, MahasiswaDAO untuk manajemen database, dan MahasiswaView untuk tampilan interaksi pengguna.

ACTIVITY PERTEMUAN 4

Nama : Aditya Rachmansyah Hamid

Kelas : 4IA28

NPM : 50421048

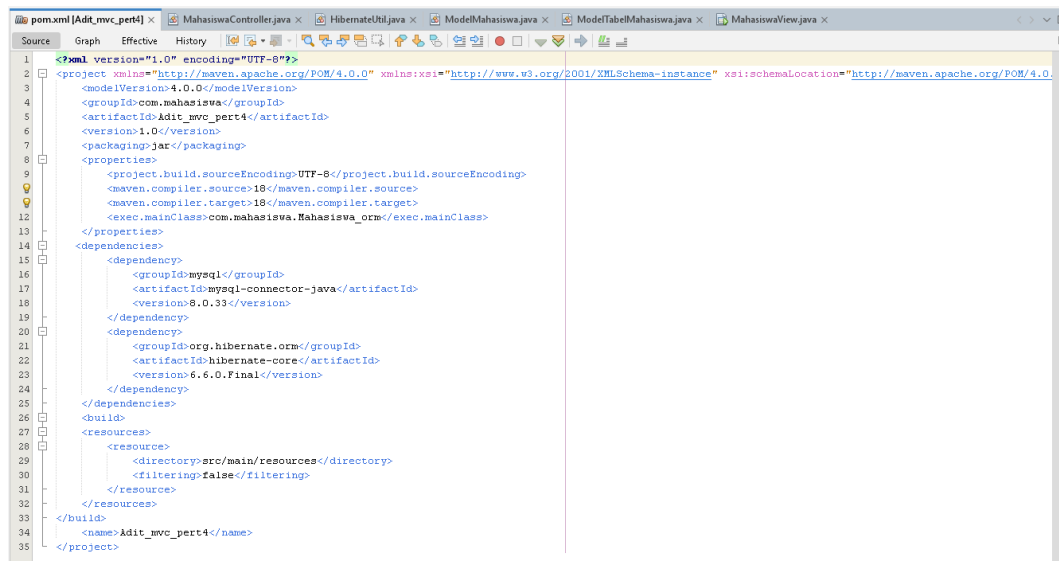
Mata Praktikum : Rekayasa Perangkat Lunak 2

Soal:

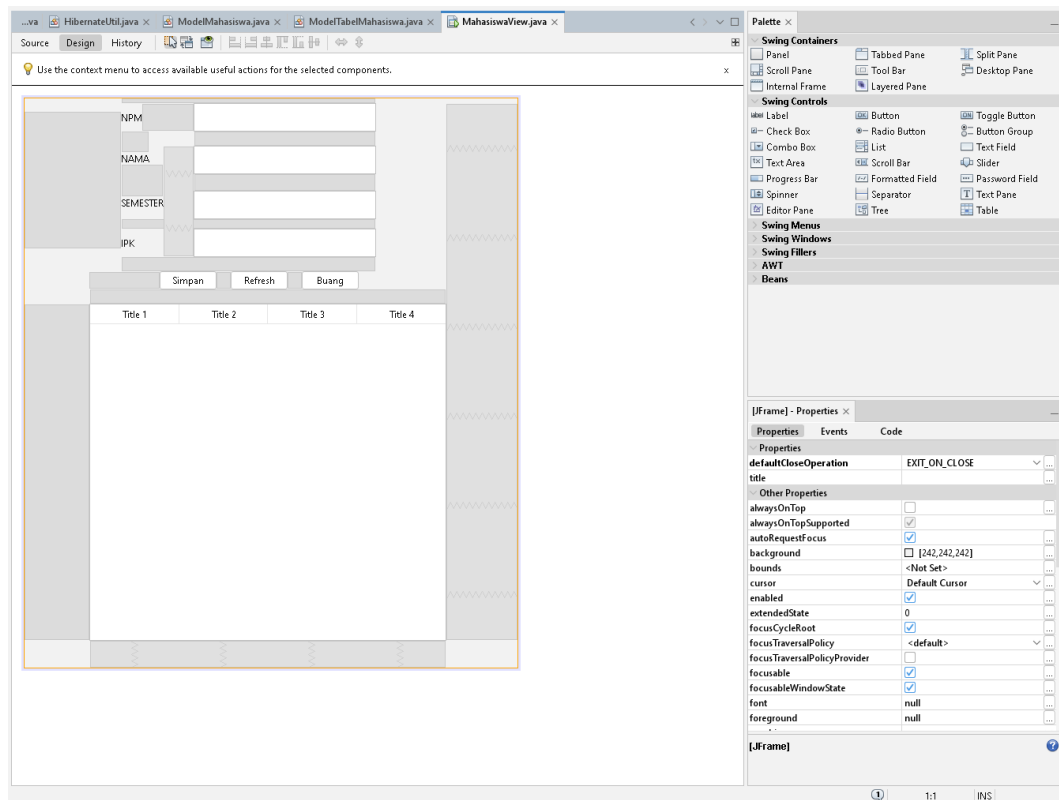
Mengikuti instruksi dari PJ

Jawab:

pom.xml:



MahasiswaView.java



ModelMahasiswa.java:

```
/*  
  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt  
to change this license  
  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit  
this template  
  
 */  
  
package com.mahasiswa.model;  
  
import jakarta.persistence.*;  
  
/**  
 *  
 * @author Aditrhamid  
 */
```

```
@Entity
@Table(name = "mahasiswa")
public class ModelMahasiswa {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;

    @Column(name = "npm", nullable = false, length = 8)
    private String npm;

    @Column(name = "nama", nullable = false, length = 50)
    private String nama;

    @Column(name = "semester")
    private int semester;

    @Column(name = "ipk")
    private float ipk;

    public ModelMahasiswa(){

    }

    public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk){
        this.id = id;
        this.npm = npm;
        this.nama = nama;
```

```
        this.semester = semester;

        this.ipk = ipk;
    }

    public int getId() {
        return id;
    }

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

    public String getNpm() {
        return npm;
    }

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

    public String getNama() {
        return nama;
    }

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

    public int getSemester() {
        return semester;
    }
}
```

```
}

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

public float getIpk() {
    return ipk;
}

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

}
```

HibernateUtil.java:

```
package com.mahasiswa.model;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static SessionFactory sessionFactory;

    static {
        try {
```

```

        // Create the SessionFactory from hibernate.cfg.xml
        sessionFactory = new Configuration().configure().buildSessionFactory();
    } catch (Throwable ex) {
        // Make sure you log the exception, as it might be swallowed
        System.err.println("Initial SessionFactory creation failed." + ex);
        throw new ExceptionInInitializerError(ex);
    }
}

public static SessionFactory getSessionFactory() {
    return sessionFactory;
}

public static void testConnection() {
    try (Session session = sessionFactory.openSession()) {
        System.out.println("Connection to the database was successful!");
    } catch (Exception e) {
        System.err.println("Failed to connect to the database.");
        e.printStackTrace();
    }
}
}

```

MahasiswaController.java:

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */
package com.mahasiswa.controller;

```

```
import com.mahasiswa.model.HibernateUtil;
import com.mahasiswa.model.ModelMahasiswa;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

public class MahasiswaController {

    public void addMhs(ModelMahasiswa mhs){
        Transaction trx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            session.save(mhs);
            trx.commit();
        } catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
    }

    public void updateMhs(ModelMahasiswa mhs) {
        Transaction trx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()){
```

```

        trx = session.beginTransaction();

        session.update(mhs);

        trx.commit();
    } catch (Exception e){
        if (trx != null){
            trx.rollback();
        }
        e.printStackTrace();
    }
}

public void deleteMhs(int id) {
    Transaction trx = null;

    try (Session session = HibernateUtil.getSessionFactory().openSession()){
        trx = session.beginTransaction();

        ModelMahasiswa mhs = session.get(ModelMahasiswa.class, id);

        if(mhs != null){
            session.delete(mhs);

            System.out.println("Berhasil hapus");
        }

        trx.commit();
    } catch (Exception e){
        if (trx != null){
            trx.rollback();
        }
        e.printStackTrace();
    }
}

```

```

    }

    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", 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;
    }
}

```

ModelTabelMahasiswa.java

```

/*

* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license

```


* Click <nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java> to edit this template

```
*/  
  
package com.mahasiswa.model;  
  
import javax.swing.table.AbstractTableModel;  
  
import java.util.List;  
  
/**  
 *  
 * @author Aditrhamid  
 */  
  
public class ModelTabelMahasiswa extends AbstractTableModel {  
    private List<ModelMahasiswa> mahasiswaList;  
  
    private final 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;  
    }  
}
```

@Override

```
public String getColumnName(int column) {  
    return columnNames[column]; // Mengatur nama kolom  
}
```

@Override

```
public boolean isCellEditable(int rowIndex, int columnIndex) {  
    return false; // Semua sel tidak dapat diedit  
}
```

```
// Method untuk menambahkan atau memodifikasi data, jika dibutuhkan
public void setMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
    this.mahasiswaList = mahasiswaList;
    fireTableDataChanged(); // Memberitahu JTable bahwa data telah berubah
}
}
```