LAPORAN AKHIR PRAKTIKUM

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 4

Tanggal : 5 November 2024

Materi : Konsep Dasar Object Relation Mapping(ORM) Dan Framework Hibernate

NPM : 51421055

Nama : Muhammad Ridho

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

Jumlah Lembar : 19 Lembar

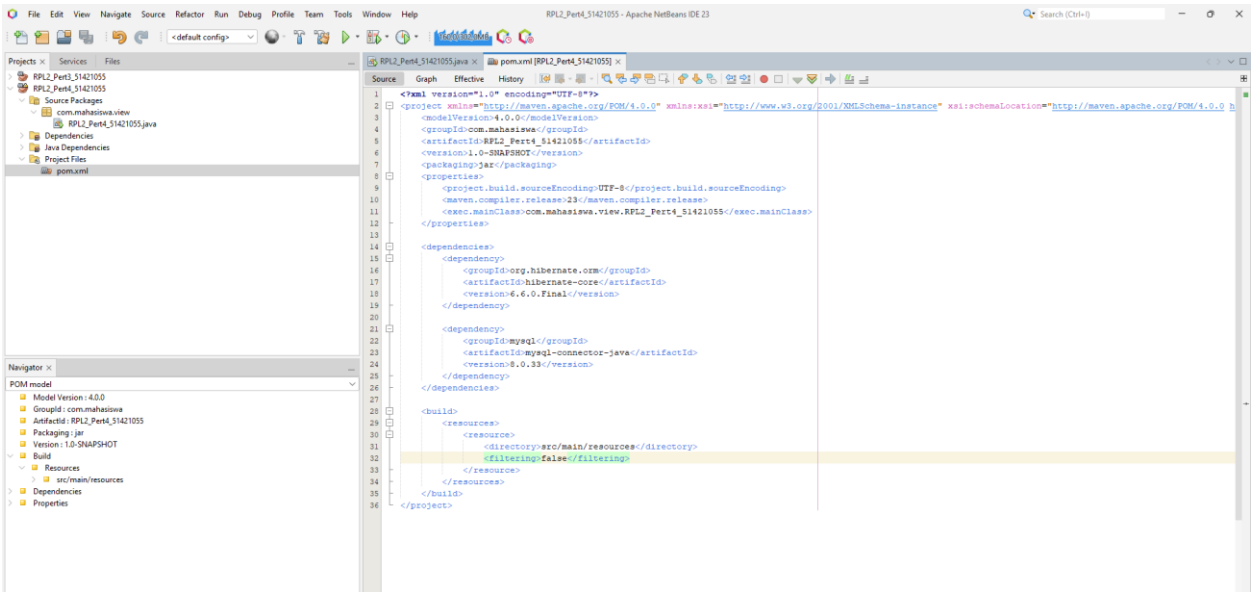
LABORATORIUM TEKNIK INFORMATIKA
UNIVERSITAS GUNADARMA
2024

Soal:

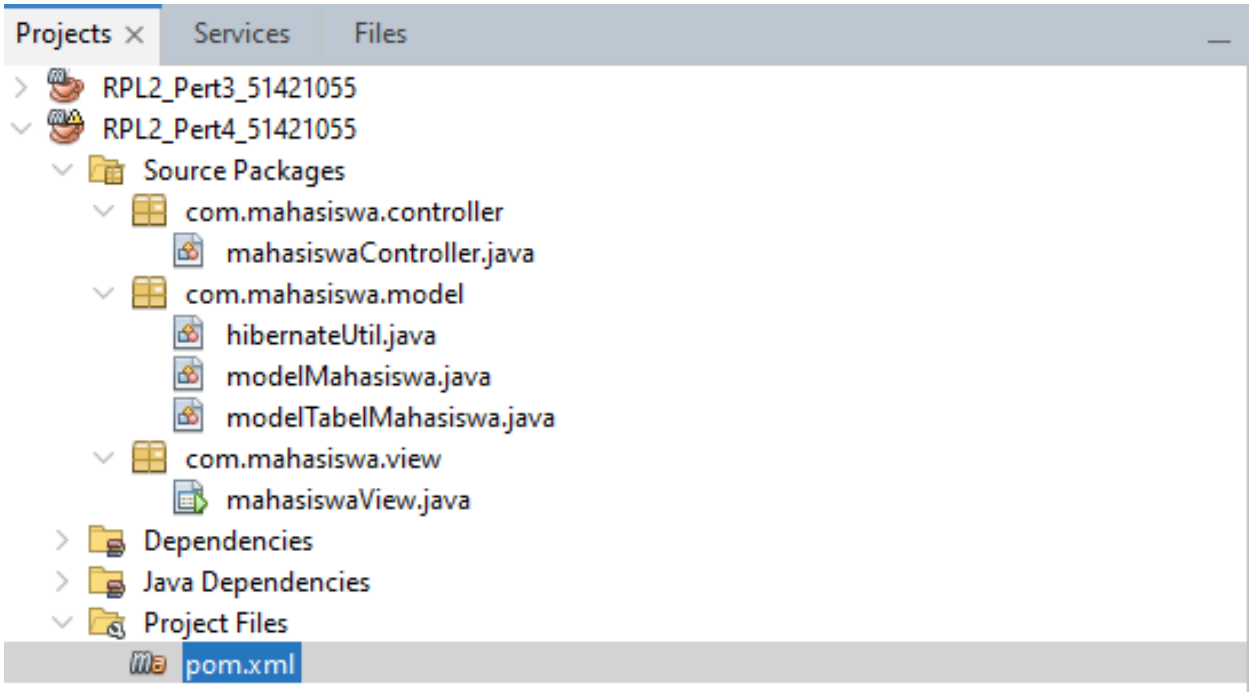
1. Jelaskan satu per satu Codingan kalian dari hasil screenshoot activity.

Jawaban:

Pom.xml

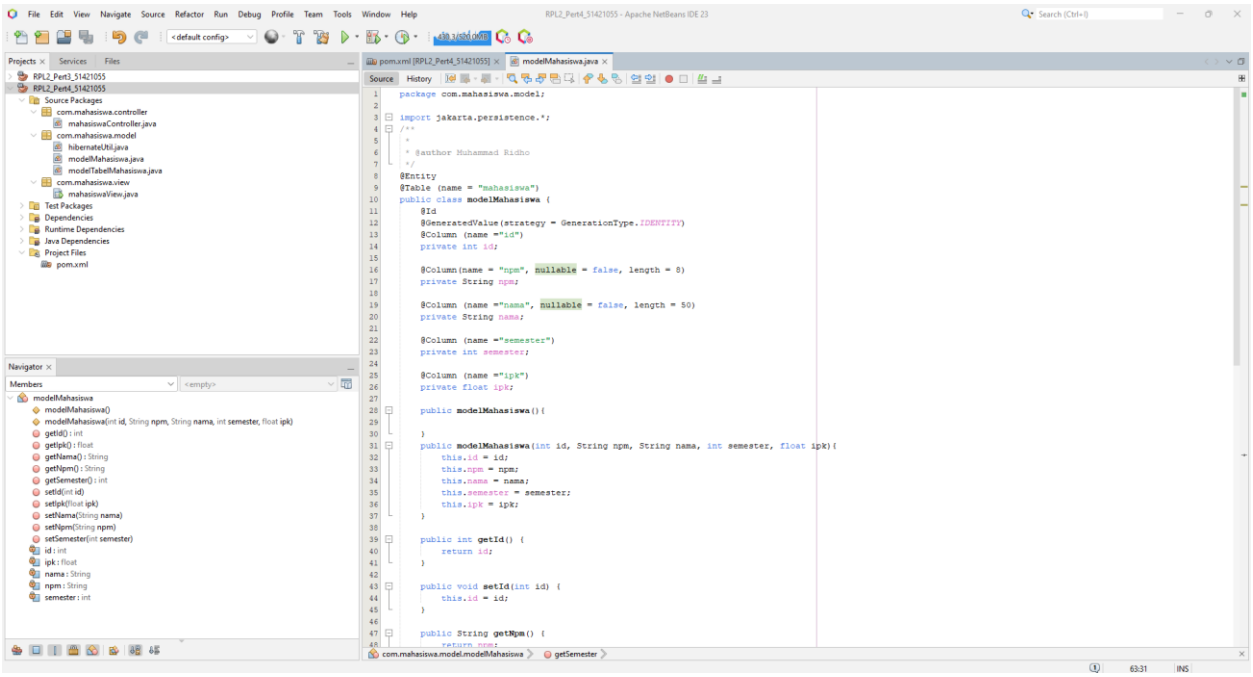


Setelah Dibuat projectnya kita buat dependencies pada pom.xml.



Setelah menambah dependencies pada **pom.xml** kita membuat java package dan java sesuai keperluan dan tidak pula pada package com.mahasiswa.view kita jadikan jframe agar terlihat interfacenya.

modalMahasiswa.java



Sourcecode:

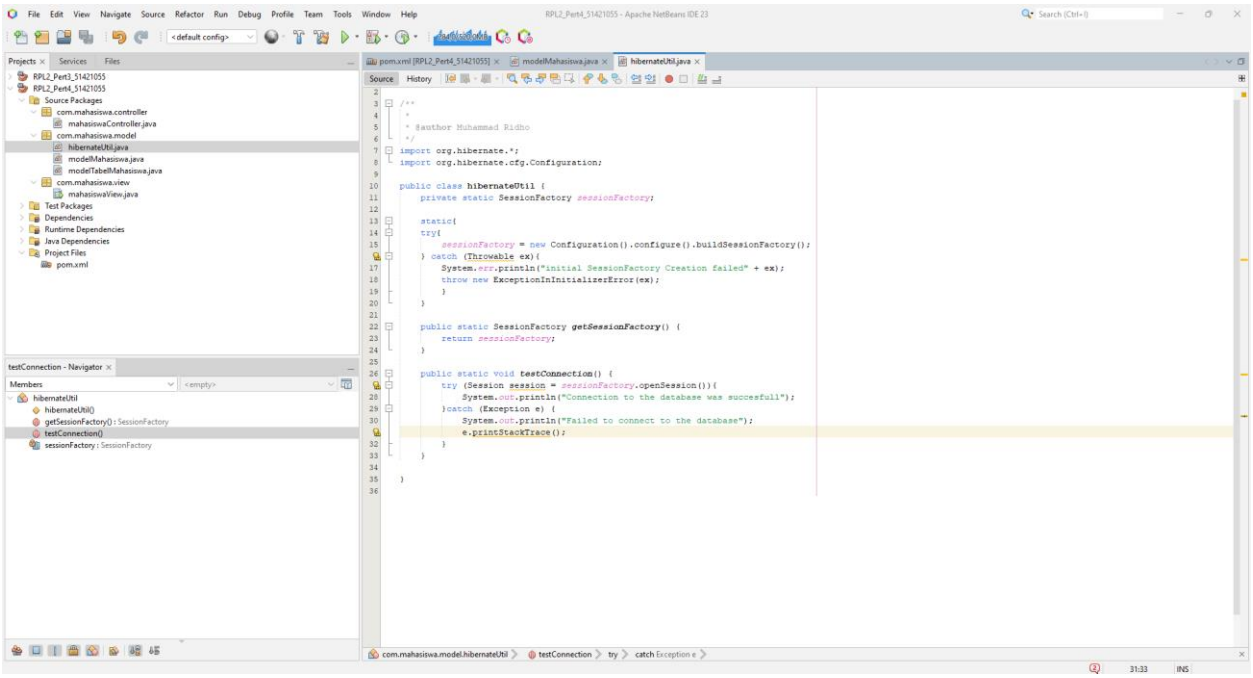
```
package com.mahasiswa.model;

import jakarta.persistence.*;
/**
 *
 * @author Muhammad Ridho
 */
@Entity
@Table (name = "mahasiswa")
public  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;
    }
}
```

Penjelasan: Codingan di atas adalah definisi class **modelMahasiswa** yang digunakan dalam konteks pemrograman Java dengan JPA (Jakarta Persistence API). Class ini bertindak sebagai model untuk entitas mahasiswa yang akan dipetakan ke dalam tabel database. Class ini memiliki beberapa atribut: id, npm, nama, semester, dan ipk, yang masing-masing dipetakan ke kolom yang sesuai dalam tabel mahasiswa. Anotasi @Entity menandakan bahwa class ini adalah entitas JPA, dan @Table menentukan nama tabel di database. Atribut id menggunakan anotasi @Id dan @GeneratedValue untuk penentuan nilai ID yang otomatis di-generate oleh database. Setiap atribut dilengkapi dengan getter dan setter untuk mengakses dan mengubah nilainya. Class ini juga memiliki dua konstruktor: konstruktor default dan konstruktor dengan parameter untuk inisialisasi nilai-nilai atribut.

hibernateUtil.java



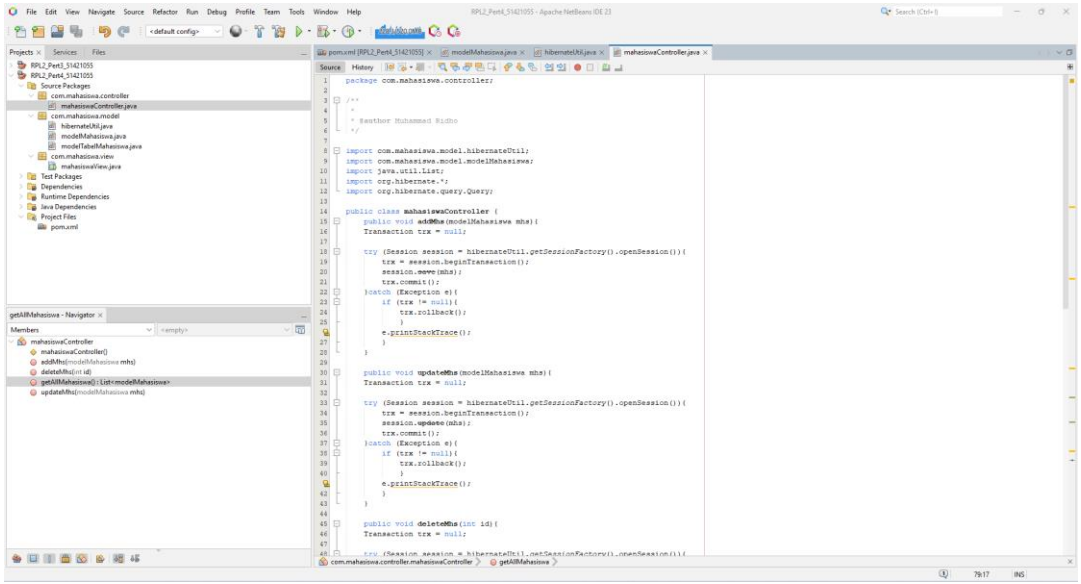
Sourcecode:

```
package com.mahasiswa.model;


/**
 *
 * @author Muhammad Ridho
 */
import org.hibernate.*;
import org.hibernate.cfg.Configuration;
public  hibernateUtil {
    private static SessionFactory sessionFactory;
    static{
    try{
        sessionFactory = new Configuration().configure().buildSessionFactory();
    } catch (Throwable ex){
        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 succesfull");
        }catch (Exception e) {
            System.out.println("Failed to connect to the database");
            e.printStackTrace();
        }
    }

}
```

Penjelasan: Codingan di atas mendefinisikan class **hibernateUtil**, yang berfungsi sebagai utilitas untuk mengelola koneksi database dengan menggunakan Hibernate, sebuah ORM (Object-Relational Mapping) framework di Java. Class ini memiliki atribut sessionFactory, yang merupakan objek SessionFactory dari Hibernate. Objek ini diinisialisasi dalam blok static yang akan dieksekusi saat class pertama kali dimuat. Di dalam blok ini, sessionFactory dibuat melalui konfigurasi Hibernate yang secara otomatis memuat pengaturan dari file konfigurasi hibernate.cfg.xml. Jika pembuatan sessionFactory gagal, akan muncul pesan error, dan ExceptionInInitializerError dilempar untuk menunjukkan kegagalan inisialisasi. Class ini juga menyediakan method getSessionFactory() untuk mengakses sessionFactory. Selain itu, terdapat method testConnection(), yang membuka sesi Hibernate dan menampilkan pesan keberhasilan jika koneksi berhasil atau menampilkan pesan error jika koneksi gagal.

mahasiswaController.java



Sourcecode:

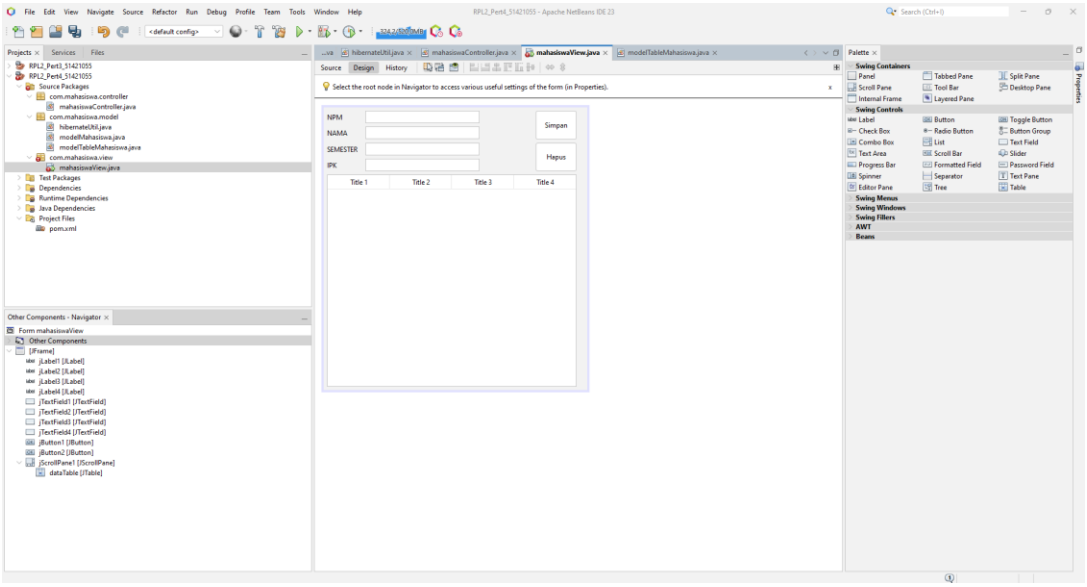
```
package com.mahasiswa.controller;

/**
 *
 * @author Muhammad Ridho
 */
import com.mahasiswa.model.hibernateUtil;
import com.mahasiswa.model.modelMahasiswa;
import java.util.List;
import org.hibernate.*;
import org.hibernate.query.Query;
public  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., id);
    if (mhs != null){
        session.delete(mhs);
        System.out.println("Berhasil dihapus");
    }
    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();
      Query<modelMahasiswa> query = session.createQuery("from modelMahasiswa",
modelMahasiswa.);
      listMhs = query.list();
      trx.commit();
   }catch (Exception e){
      if (trx != null){
         trx.rollback();
         }
      e.printStackTrace();
      }
   return listMhs;
   }
}
```

Penjelasan: Codingan diatas terdapat Class **mahasiswaController** pada kode di atas mengelola operasi CRUD (Create, Read, Update, Delete) untuk entitas modelMahasiswa menggunakan Hibernate. Class ini memiliki beberapa method utama, yaitu addMhs(modelMahasiswa mhs), updateMhs(modelMahasiswa mhs), deleteMhs(int id), dan getAllMahasiswa(). Method addMhs berfungsi untuk menyimpan data mahasiswa baru ke database dengan memulai transaksi, menyimpan objek mhs, dan melakukan komit. Jika terjadi error, transaksi akan dibatalkan. Method updateMhs digunakan untuk memperbarui data mahasiswa yang sudah ada dengan cara yang serupa, yaitu memulai transaksi, memperbarui objek mahasiswa, dan mengkomit transaksi, atau melakukan rollback jika terjadi kesalahan. Method deleteMhs menghapus data mahasiswa berdasarkan ID, di mana data mahasiswa pertama diambil berdasarkan ID, lalu dihapus jika ditemukan, dan diakhiri dengan komit. Jika terjadi kesalahan, transaksi akan di-rollback. Terakhir, method getAllMahasiswa() digunakan untuk mengambil semua data mahasiswa dari database menggunakan query HQL "from modelMahasiswa" dan mengembalikan hasilnya dalam bentuk List<modelMahasiswa>. Method ini juga mengelola transaksi dengan komit atau rollback jika ada error.

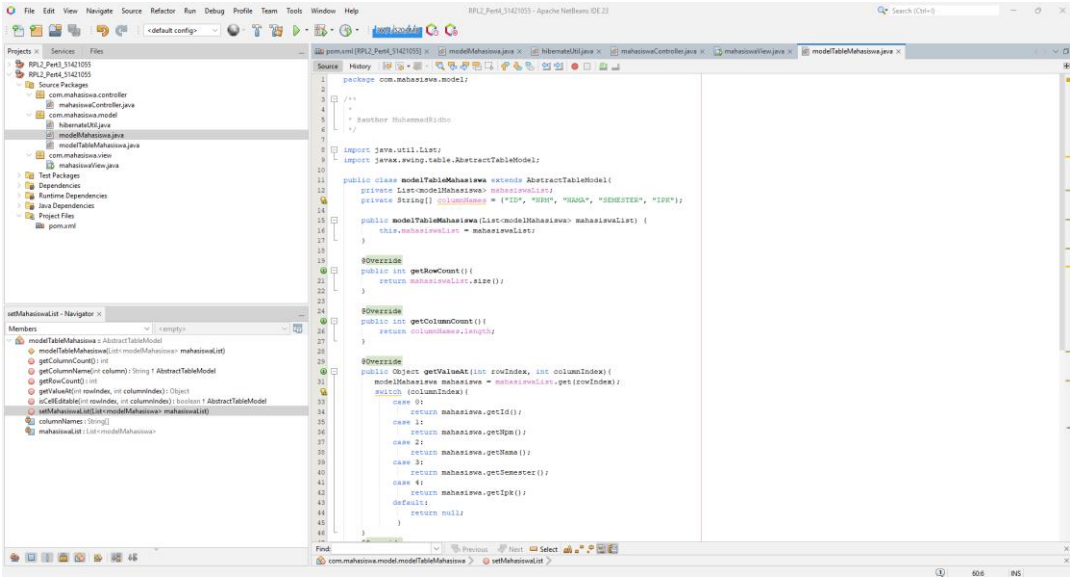
mahasiswaView.java



Pembuatan form pada mahasiswa.view.java menggunakan jframe.

Penjelasan: Disini kita membuat jframe untuk pembuatan form yang berisi data NPM, NAMA, SEMESTER, DAN IPK. Disini ada label (untuk informasi form), textfield(semacam input), table(berisi data-data dalam bentuk tabel), dan 2 button yang berfungsi untuk menyimpan data dan menghapusnya.

modelTableMahasiswa.java



Sourcecode:

package com.mahasiswa.model;

*/***

***

```
 * @author MuhammadRidho
 */
import java.util.List;
import javax.swing.table.AbstractTableModel;
public  modelTableMahasiswa extends AbstractTableModel{
   private List<modelMahasiswa> mahasiswaList;
   private String[] columnNames = {"ID", "NPM", "NAMA", "SEMESTER", "IPK"};
   public modelTableMahasiswa(List<modelMahasiswa> mahasiswaList) {
      this.mahasiswaList = mahasiswaList;
   }
   @Override
   public int getRowCount(){
      return mahasiswaList.size();
   }
   @Override
   public int getColumnCount(){
      return columnNames.length;
   }
   @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:
```

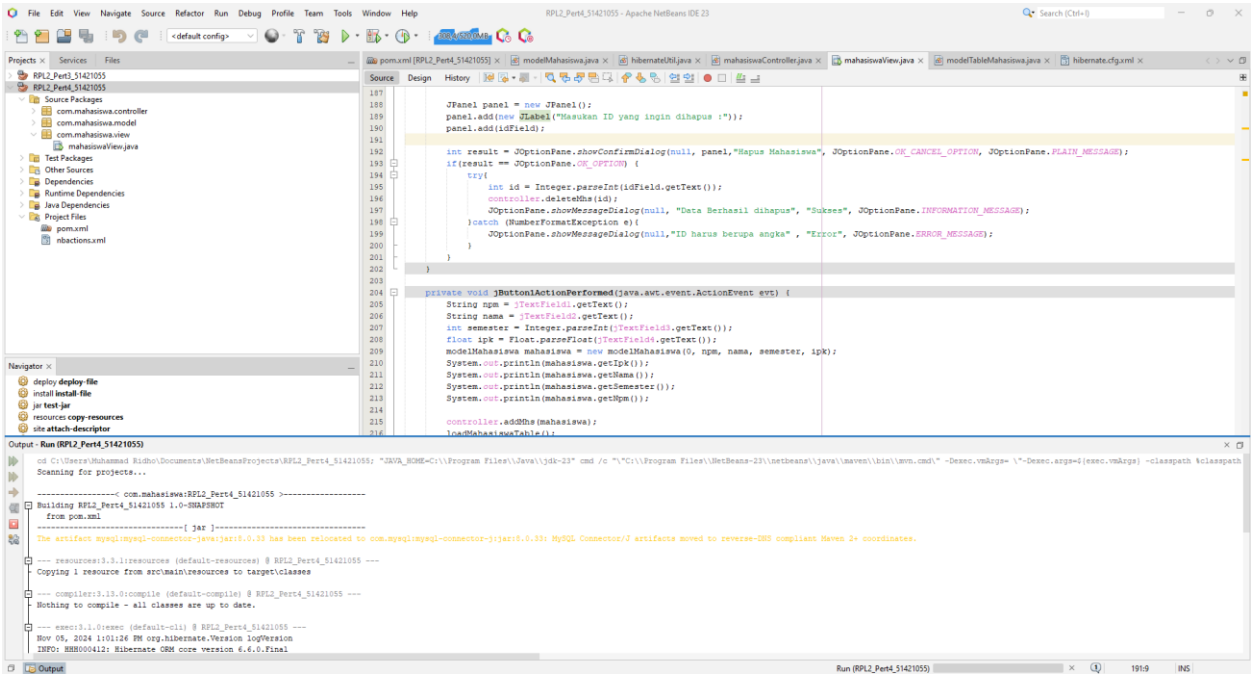
```
            return mahasiswa.getIpk();
        default:
            return null;
        }
    }
    @Override
    public String getColumnName(int column){
        return columnNames[column];
    }
    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex){
        return false;
    }
    public void setMahasiswaList(List<modelMahasiswa> mahasiswaList){
        this.mahasiswaList = mahasiswaList;
        fireTableDataChanged();
    }
}
```

Penjelasan: Class **modelTableMahasiswa** pada kode di atas adalah class model tabel yang digunakan untuk menampilkan data modelMahasiswa dalam bentuk tabel pada komponen GUI berbasis Swing. Class ini memperluas AbstractTableModel, yang menyediakan kerangka kerja dasar untuk mengelola data tabel. Class ini memiliki atribut mahasiswaList, yang menyimpan daftar objek modelMahasiswa, dan columnNames, yang menyimpan nama kolom tabel: "ID", "NPM", "NAMA", "SEMESTER", dan "IPK".

Metode getRowCount() mengembalikan jumlah baris sesuai dengan jumlah objek dalam mahasiswaList, dan getColumnCount() mengembalikan jumlah kolom berdasarkan panjang array columnNames. Metode getValueAt(int rowIndex, int columnIndex) digunakan untuk mengambil data di sel tertentu, di mana data ditentukan berdasarkan indeks baris dan kolom. Metode ini menggunakan struktur switch untuk memilih atribut modelMahasiswa yang akan ditampilkan sesuai kolom yang diinginkan.

Selain itu, metode getColumnName(int column) mengembalikan nama kolom sesuai dengan indeks kolom, dan isCellEditable(int rowIndex, int columnIndex) menentukan bahwa semua sel tidak dapat diedit (mengembalikan false). Ada juga metode setMahasiswaList(List<modelMahasiswa> mahasiswaList), yang memperbarui daftar mahasiswa dan memanggil fireTableDataChanged() untuk memberitahu tabel bahwa data telah berubah, sehingga tampilan tabel akan diperbarui.

mahasiswaView.java



Pembuatan source code mahasiswaView untuk mengaktifkan fungsi button dan formnya:

```
package com.mahasiswa.view;

/**
 *
 * @author Muhammad Ridho
 */
import com.mahasiswa.model.modelTableMahasiswa;
import com.mahasiswa.model.modelMahasiswa;
import com.mahasiswa.controller.mahasiswaController;
import com.mahasiswa.model.hibernateUtil;
import java.util.List;
import javax.swing.*;


public mahasiswaView extends javax.swing.JFrame {
    private mahasiswaController controller;
    public void loadMahasiswaTable(){
        List<modelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
        modelTableMahasiswa tableModel = new modelTableMahasiswa(listMahasiswa);
```

```
    dataTable.setModel(tableModel);
  }
  /**
   * Creates new form mahasiswaView
   */
  public mahasiswaView() {
     initComponents();
     controller = new mahasiswaController();
     hibernateUtil.testConnection();
     loadMahasiswaTable();
  }
  public void clearTextField(){
     jTextField1.setText("");
     jTextField2.setText("");
     jTextField4.setText("");
     jTextField4.setText("");
  }
  /**
   * This method is called from within the constructor to initialize the form.
   * WARNING: Do NOT modify this code. The content of this method is always
   * regenerated by the Form Editor.
   */
  @SuppressWarnings("unchecked")
   JTextField idField = new JTextField(5);
     JPanel panel = new JPanel();
     panel.add(new JLabel("Masukan ID yang ingin dihapus :"));
     panel.add(idField);

     int result = JOptionPane.showConfirmDialog(null, panel,"Hapus Mahasiswa",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
```

```
    if(result == JOptionPane.OK_OPTION) {
        try{
            int id = Integer.parseInt(idField.getText());
            controller.deleteMhs(id);
            JOptionPane.showMessageDialog(null, "Data Berhasil dihapus", "Sukses",
JOptionPane.INFORMATION_MESSAGE);
        }catch (NumberFormatException e){
            JOptionPane.showMessageDialog(null,"ID harus berupa angka" , "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }
    String npm = jTextField1.getText();
    String nama = jTextField2.getText();
    int semester = Integer.parseInt(jTextField3.getText());
    float ipk = Float.parseFloat(jTextField4.getText());
    modelMahasiswa mahasiswa = new modelMahasiswa(0, npm, nama, semester, ipk);
    System.out.println(mahasiswa.getIpk());
    System.out.println(mahasiswa.getNama());
    System.out.println(mahasiswa.getSemester());
    System.out.println(mahasiswa.getNpm());

    controller.addMhs(mahasiswa);
    loadMahasiswaTable();
    clearTextField();
```

Penjelasan: Class **mahasiswaView** di atas merupakan bagian dari tampilan (view) pada aplikasi berbasis Swing untuk menampilkan dan mengelola data mahasiswa. Class ini memperluas JFrame untuk membuat antarmuka pengguna dengan beberapa elemen GUI, termasuk tabel dan kotak dialog.

Class ini memiliki metode loadMahasiswaTable() yang mengambil data mahasiswa dari database melalui mahasiswaController dan kemudian mengatur model tabel (modelTableMahasiswa) ke dalam komponen tabel dataTable. Ini memungkinkan data mahasiswa ditampilkan dalam tabel di GUI.

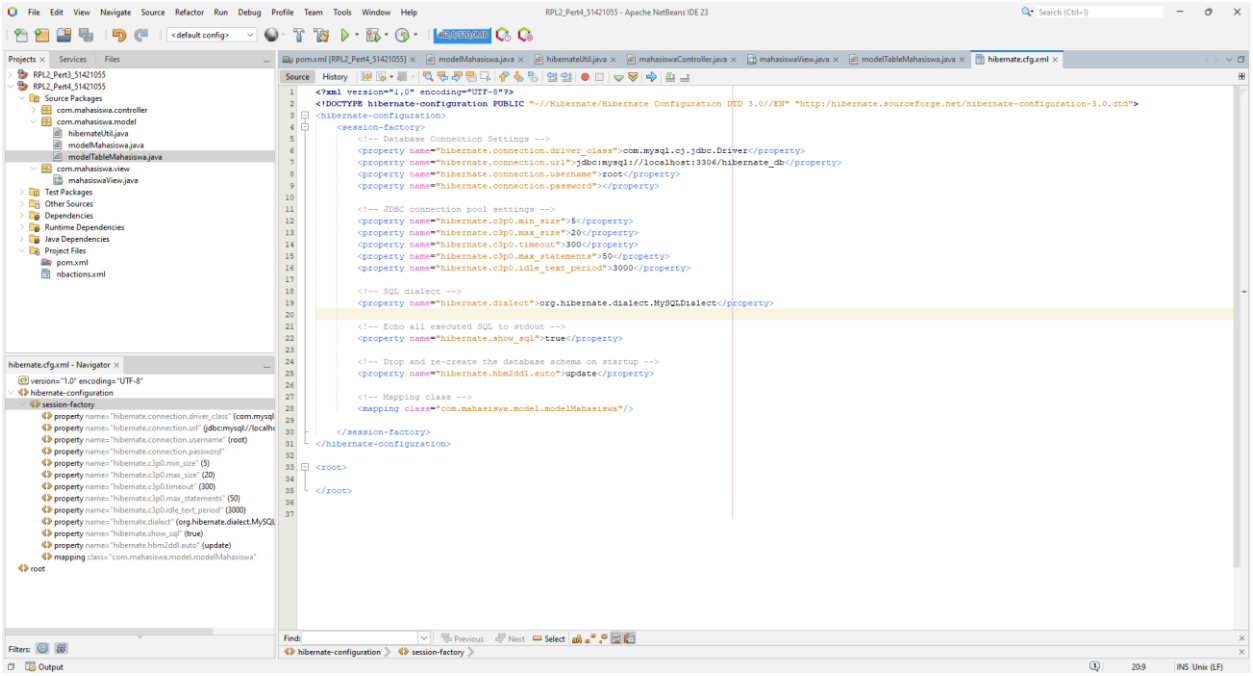
Konstruktor mahasiswaView() menginisialisasi komponen tampilan dengan memanggil initComponents() (untuk mengatur elemen-elemen GUI), menginisialisasi mahasiswaController, menguji koneksi database melalui hibernateUtil.testConnection(), dan memuat data mahasiswa ke dalam tabel melalui loadMahasiswaTable().

Metode clearTextField() mengosongkan kolom input teks yang digunakan untuk memasukkan data mahasiswa, seperti jTextField1, jTextField2, jTextField3, dan jTextField4.

Bagian dari kode juga mengatur dialog JOptionPane yang memungkinkan pengguna memasukkan ID mahasiswa yang ingin dihapus. Setelah ID dimasukkan, program akan memvalidasi ID tersebut sebagai angka dan kemudian memanggil deleteMhs(id) dari mahasiswaController untuk menghapus data mahasiswa yang sesuai. Jika ID tidak valid, pesan error akan muncul.

Selain itu, kode membaca data dari kolom input (NPM, Nama, Semester, dan IPK), membuat objek modelMahasiswa baru, dan menggunakan metode addMhs() untuk menambahkannya ke database. Setelah itu, tabel diperbarui melalui loadMahasiswaTable(), dan kolom input dibersihkan dengan clearTextField().

Hibernate.cfg.xml



Sourcecode:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http:/hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<!-- Database Connection Settings -->

```
        <property name="hibernate.connection.driver_">com.mysql.cj.jdbc.Driver</property>
        <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/hibernate_db</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>
        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_text_period">3000</property>
        <!-- SQL dialect -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>
        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>
        <!-- Mapping  -->
        <mapping ="com.mahasiswa.model.modelMahasiswa"/>
    </session-factory>
</hibernate-configuration>
<root>
</root>
```

Penjelasan: Konfigurasi XML di atas adalah file **hibernate.cfg.xml** untuk pengaturan Hibernate. Dalam konfigurasi ini, bagian <session-factory> menentukan pengaturan koneksi database. hibernate.connection.driver_ mengatur driver MySQL (com.mysql.cj.jdbc.Driver), dengan URL database jdbc:mysql://localhost:3306/hibernate_db, dan kredensial (username root, tanpa password). Pengaturan pool koneksi diatur dengan hibernate.c3p0.min_size (minimal 5 koneksi) dan hibernate.c3p0.max_size (maksimal 20 koneksi), dengan waktu timeout, max_statements, dan idle_test_period untuk mengelola idle connections. Properti hibernate.dialect menentukan SQL dialect yang digunakan untuk MySQL, dan hibernate.show_sql diatur true agar SQL yang dieksekusi ditampilkan di konsol. Properti hibernate.hbm2ddl.auto diset update untuk memperbarui skema database saat aplikasi dimulai, dan bagian <mapping> mencantumkan class entitas modelMahasiswa untuk pemetaan ke tabel database.

Output Source Code