# LAPORAN AKHIR PRAKTIKUM

| | |
|---|---|
| Mata Praktikum | : Rekayasa Perangkat Lunak 2 |
| Kelas | : 4IA07 |
| Praktikum ke- | : 6 |
| Tanggal | : 22 November 2025 |
| Materi | : |
| NPM | : 51422084 |
| Nama | : Muhammad Izzuddin Almansyur |
| Ketua Asisten | : Haikal Abizar |
| Jumlah Lembar | : 9 |


## LABORATORIUM INFORMATIKA

## UNIVERSITAS GUNADARMA

## 2025

1. Jelaskan kode dan langkah-langkah program yang telah dibuat!

**pom.xml**

```
    </properties>
    <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.0</version>
    <relativePath/>
</parent>

<dependencies>
    <!-- Hibernate + Spring Data JPA -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <!-- MySQL Connector -->
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.33</version>
    </dependency>

    <!-- Spring Boot Web dependency (for MVC if needed) -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- Testing dependencies -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
</project>
```

File pom.xml adalah konfigurasi Maven yang digunakan untuk mengatur dependensi dan pengaturan build pada project. Di dalam file ini ditetapkan informasi dasar project seperti groupId, artifactId, versi aplikasi, serta packaging yang menggunakan format JAR. Bagian <parent> menunjukkan bahwa project ini menggunakan Spring Boot dengan versi 3.4.0 sebagai dasar sehingga seluruh konfigurasi default Spring Boot akan otomatis ikut terpasang. Pada bagian <dependencies>, ditambahkan library penting yaitu spring-boot-starter-data-jpa untuk ORM dan akses database menggunakan Hibernate, mysql-connector-java agar aplikasi dapat terhubung dengan database MySQL, spring-boot-starter-web jika suatu saat ingin menambahkan fungsi berbasis REST atau MVC, serta spring-boot-starter-test untuk kebutuhan testing. Di dalam <properties>, terdapat pengaturan encoding UTF-8, versi compiler Java pada release 24, dan penentuan main class yang akan dijalankan ketika aplikasi dieksekusi. Bagian <build> memuat spring-boot-maven-plugin yang memungkinkan aplikasi dijalankan langsung melalui command Maven dan melakukan packaging menjadi JAR executable.

**ModelMahasiswa.java**

```java
import jakarta.persistence.*;

/**
 *
 * @author Izza
 */
@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;

    // constructor kosong untuk ORM
    public ModelMahasiswa() {

    }

     //constructor ada isinya untuk dipanggil di aplikasi
    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;
    }
```

ModelMahasiswa adalah kelas entity yang mewakili tabel mahasiswa pada database. Kelas ini diberi anotasi @Entity agar dikenali oleh Hibernate sebagai objek yang akan dipetakan ke tabel, sedangkan anotasi @Table(name="mahasiswa") memastikan nama tabel sesuai dengan yang ada di database. Field id berfungsi sebagai primary key dan diberi anotasi @Id serta @GeneratedValue(strategy = GenerationType.IDENTITY) sehingga nilainya akan otomatis di-generate oleh database, sesuai tipe auto-increment MySQL. Setiap atribut seperti npm, nama, semester, dan ipk dipetakan ke kolom database menggunakan anotasi @Column, lengkap dengan pengaturan panjang karakter dan nullable. Kelas ini memiliki constructor kosong yang dibutuhkan oleh JPA saat proses ORM berjalan, serta constructor berisi parameter untuk memudahkan instansiasi data dari aplikasi. Method getter dan setter digunakan agar setiap properti dapat diakses dan dimodifikasi melalui encapsulation. Pada bagian akhir terdapat method toString, meskipun terdapat kesalahan teks label, namun tujuannya untuk menampilkan representasi string dari objek mahasiswa ketika dicetak.

**ModelTableMahasiswa.java**

```java
import java.util.List;
import javax.swing.table.AbstractTableModel;

/**
 *
 * @author Izza
 */
public class 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]; // 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
    }
}
```

ModelTableMahasiswa adalah kelas yang memperluas AbstractTableModel untuk menghubungkan data mahasiswa dengan komponen JTable pada aplikasi Swing. Kelas ini menyimpan list objek ModelMahasiswa yang akan ditampilkan dalam bentuk tabel, serta array nama kolom agar JTable mengetahui label setiap kolom. Method getRowCount mengembalikan jumlah baris sesuai banyaknya data dalam list, sedangkan getColumnCount menentukan jumlah kolom berdasarkan panjang array columnNames. Method getValueAt dipanggil JTable setiap kali membutuhkan nilai dalam sel tertentu dengan mengambil data mahasiswa berdasarkan indeks baris dan mengembalikan nilai sesuai kolom yang diminta melalui switch. Nama kolom ditentukan melalui override getColumnName sehingga header tabel dapat muncul sesuai dengan field mahasiswa. Kelas ini juga mendefinisikan seluruh sel tidak dapat diedit dengan mengembalikan false pada isCellEditable. Terdapat method tambahan setMahasiswaList untuk memperbarui isi tabel ketika data berubah, kemudian memanggil fireTableDataChanged agar JTable otomatis melakukan refresh tampilan.

**MahasiswaRepository.java**

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Interface.java to edit this template
 */
package com.mycompany.rpl2_pert6_izzuddin.repository;

import com.mycompany.rpl2_pert6_izzuddin.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;

/**
 *
 * @author Izza
 */
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Integer> {

}
```

MahasiswaRepository merupakan interface yang mewarisi JpaRepository sehingga tidak memerlukan implementasi manual untuk operasi database. Dengan memberikan tipe ModelMahasiswa sebagai entity dan Integer sebagai tipe primary key, repository ini secara otomatis memiliki berbagai method CRUD seperti save, findById, findAll, dan deleteById tanpa perlu ditulis ulang. Spring Data JPA akan membuatkan implementasinya secara otomatis pada saat runtime. Karena berbentuk interface, kelas ini cukup kosong tanpa anotasi tambahan, namun tetap berfungsi penuh sebagai penghubung antara service dan database.

**MahasiswaService.java**

```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.mycompany.rpl2_pert6_izzuddin.service;

import com.mycompany.rpl2_pert6_izzuddin.model.ModelMahasiswa;
import com.mycompany.rpl2_pert6_izzuddin.repository.MahasiswaRepository;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

/**
 *
 * @author Izza
 */
@Service
public class MahasiswaService {
    @Autowired
    private MahasiswaRepository repository;

    public void addMhs(ModelMahasiswa mhs) {
        repository.save(mhs);
    }

    public ModelMahasiswa getMhs(int id) {
        return repository.findById(id).orElse(null);
    }

    public void updateMhs(ModelMahasiswa mhs) {
        repository.save(mhs);
    }

    public void deleteMhs(int id) {
        repository.deleteById(id);
    }

    public List<ModelMahasiswa> getAllMahasiswa() {
        return repository.findAll();
    }

}
```

MahasiswaService adalah kelas service beranotasi @Service yang berfungsi sebagai lapisan logika bisnis antara repository dan layer aplikasi lain seperti controller atau UI. Repository di-inject menggunakan anotasi @Autowired sehingga Spring akan

menyediakan instance MahasiswaRepository secara otomatis. Method addMhs menyimpan data mahasiswa baru ke database melalui save, sementara getMhs mengambil satu data berdasarkan ID menggunakan findById. Method updateMhs menggunakan save kembali karena pada JPA menyimpan dan mengupdate menggunakan fungsi yang sama. deleteMhs menghapus data dari database berdasarkan ID, sedangkan getAllMahasiswa mengembalikan seluruh data dalam bentuk list. Kelas ini memastikan agar UI tidak berinteraksi langsung dengan database, sehingga arsitektur tetap terstruktur dan mudah dikelola.

**MahasiswaView**

```
7  import com.mycompany.rpl2_pert6_izzuddin.controller.MahasiswaController;
8  import com.mycompany.rpl2_pert6_izzuddin.model.ModelMahasiswa;
9  import com.mycompany.rpl2_pert6_izzuddin.model.ModelTableMahasiswa;
10 import java.util.List;
11 import javax.swing.JLabel;
12 import javax.swing.JOptionPane;
13 import javax.swing.JPanel;
14 import javax.swing.JTextField;
15
16 /**
17  *
18  * @author Izza
19  */
20 public class MahasiswaView extends javax.swing.JFrame {
21
22     private static final java.util.logging.Logger logger = java.util.logging.Logger.getLogger(MahasiswaView.class.getName());
23
24     private MahasiswaController controller;
25
26     /**
27      * Creates new form MahasiswaView
28      */
29     public MahasiswaView(MahasiswaController controller) {
30         this.controller = controller;
31         initComponents();
32         loadMahasiswaTable();
33     }
34
35     private MahasiswaView() {
36         throw new UnsupportedOperationException("Not Supported yet.");
37     }
38
39     public void loadMahasiswaTable() {
40         List <ModelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
41
42         ModelTableMahasiswa tableModel = new ModelTableMahasiswa(listMahasiswa);
43
44         dataTable.setModel(tableModel);
45     }
46
```

MahasiswaView adalah class JFrame yang berfungsi sebagai antarmuka pengguna berbasis Swing. Kelas ini menerima MahasiswaController pada constructor sehingga UI dapat memanggil logika data melalui controller, bukan langsung ke service. Pada saat form dibuat, method initComponents dijalankan untuk membentuk seluruh komponen GUI seperti field input, tombol, tabel, dan label. Setelah tampilan terbentuk, loadMahasiswaTable dipanggil untuk mengambil seluruh data mahasiswa dari controller, kemudian menetapkan ModelTableMahasiswa sebagai model JTable agar data muncul di layar. Tombol Simpan membaca input dari field npm, nama, semester, dan ipk, lalu membuat objek ModelMahasiswa baru, mengirimkannya ke controller untuk disimpan, dan memperbarui tabel. Tombol Refresh hanya memanggil

loadMahasiswaTable untuk memperbarui tampilan, sedangkan tombol Hapus menampilkan dialog input ID menggunakan JOptionPane, kemudian menghapus data jika ID valid. Method main menyiapkan tampilan dengan tema Nimbus dan menampilkan form secara langsung menggunakan EventQueue agar sesuai standar Swing thread safety.

**MahasiswaController.java**

```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.mycompany.rpl2_pert6_izzuddin.controller;

import com.mycompany.rpl2_pert6_izzuddin.model.ModelMahasiswa;
import com.mycompany.rpl2_pert6_izzuddin.service.MahasiswaService;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;

/**
 *
 * @author Izza
 */
@Controller
public class MahasiswaController {
    @Autowired
    private MahasiswaService mahasiswaService;

    public String addMahasiswa(@RequestBody ModelMahasiswa mhs) {
        mahasiswaService.addMhs(mhs);
        return "Mahasiswa added successfully";
    }

    public ModelMahasiswa getMahasiswa(@PathVariable int id) {
        return mahasiswaService.getMhs(id);
    }

    public String updateMahasiswa(@RequestBody ModelMahasiswa mhs){
        mahasiswaService.updateMhs(mhs);
        return "Mahasiswa updated successfully";
    }

    public String deleteMahasiswa(@PathVariable int id) {
        mahasiswaService.deleteMhs(id);
        return "Mahasiswa deleted successfuly";
    }

    public List<ModelMahasiswa> getAllMahasiswa() {
        return mahasiswaService.getAllMahasiswa();
    }
}
```

Kelas *MahasiswaController* bertugas menjadi pengatur alur data antara view dan service. Controller menerima permintaan dari tampilan seperti menambah, mengambil, atau menghapus mahasiswa lalu meneruskannya ke MahasiswaService. Dengan struktur ini, tampilan tidak berhubungan langsung dengan database atau repository sehingga pola MVC dapat diterapkan dengan rapi. Controller memastikan setiap aksi dari pengguna diproses di layer yang tepat tanpa mencampurkan logika UI dengan logika data.

**application.propertis**

```
# Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
# Click nbfs://nbhost/SystemFileSystem/Templates/Other/properties.properties to edit this template


# Konfigurasi MySQL Hibernate
spring.datasource.url=jdbc:mysql://localhost:3306/rpl2_pert5_db?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
server.port=9090
# Hibernate settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

File application.properties menyimpan konfigurasi untuk koneksi database pada aplikasi Spring Boot. Di dalamnya biasanya terdapat pengaturan seperti URL database MySQL, username, password, serta properti Hibernate seperti ddl-auto untuk mengatur apakah tabel akan dibuat otomatis. File ini memungkinkan aplikasi terhubung ke database tanpa harus menulis konfigurasi secara manual di dalam kode Java.

**MahasiswaApp.java**

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

package com.mycompany.rpl2_pert6_izzuddin;

import com.mycompany.rpl2_pert6_izzuddin.controller.MahasiswaController;
import com.mycompany.rpl2_pert6_izzuddin.service.MahasiswaService;
import com.mycompany.rpl2_pert6_izzuddin.view.MahasiswaView;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.ApplicationArguments;
import org.springframework.boot.ApplicationRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;

/**
 *
 * @author Izza
 */
@SpringBootApplication
public class MahasiswaApp implements ApplicationRunner{
    @Autowired
    private MahasiswaService mahasiswaService;

    public static void main(String[] args) {
        System.setProperty("java.awt.headless", "false"); // Disable headless mode

        // Start the Spring application and get the application context
        ApplicationContext context = SpringApplication.run(MahasiswaApp.class, args);

        // Instantiate the view and inject the controller manually
        MahasiswaController controller = context.getBean(MahasiswaController.class);
        MahasiswaView mahasiswaView = new MahasiswaView(controller);
        mahasiswaView.setVisible(true);
    }

    @Override
    public void run(ApplicationArguments args) throws Exception {
        // Implement this method if you need to execute logic after Spring application starts
        // Otherwise, you can leave it as is.
    }

}
```

MahasiswaApp.java adalah kelas utama aplikasi yang diberi anotasi @SpringBootApplication sehingga Spring Boot dapat melakukan scanning otomatis terhadap controller, service, dan repository yang ada dalam proyek. Method main menjalankan aplikasi menggunakan SpringApplication.run, yang menyalakan konteks Spring dan menginisialisasi dependency. Class ini mengimplementasikan

CommandLineRunner sehingga method run akan dijalankan segera setelah Spring selesai melakukan proses inisialisasi. Di dalam method run, EventQueue.invokeLater dipanggil agar MahasiswaView ditampilkan pada thread UI Swing yang tepat. Controller di-inject menggunakan @Autowired sehingga MahasiswaView dapat memanggil logika pengelolaan data tanpa membuat koneksi database secara manual.

**Langkah – Langkah Pembuatan:**

Langkah pembuatan aplikasi ini di NetBeans dimulai dengan membuat proyek baru berbasis Maven, lalu memilih template Java dengan konfigurasi otomatis. Setelah proyek terbentuk, file pom.xml diperbarui dengan menambahkan parent Spring Boot dan dependensi yang diperlukan. Selanjutnya struktur paket dibuat terpisah yaitu model, repository, service, controller, dan view agar aplikasi lebih terorganisir. Entity ModelMahasiswa dibuat terlebih dahulu agar Hibernate dapat memetakan tabel, kemudian repository didefinisikan untuk mengelola akses database, service dibuat agar logika bisnis terpisah dari tampilan, dan terakhir tampilan Swing dibangun menggunakan GUI Builder NetBeans. Setelah seluruh bagian terhubung, aplikasi dapat dijalankan langsung dari NetBeans dan proses penyimpanan, penampilan, serta penghapusan data mahasiswa dapat dilakukan melalui antarmuka yang telah dibuat.