ACTIVITY PERTEMUAN 5

NAMA : **Johan**

NPM : **50421686**

KELAS : **4IA28**

MATERI : **Spring Boot**

MATA PRAKTIKUM : **Rekayasa Perangkat Lunak 2**

Source Code:

Application.properties

```
# Konfigurasi MySQL Hibernate
spring.datasource.url=jdbc:mysql://localhost:3306/rpl_pert5?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true

```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLS
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>RPL_Johan_Spring</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>com.mycompany.rpl_johan_spring.RPL_Johan_Spring</exec.mainClass>
    </properties>
    <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.3</version>
    <relativePath/>
</parent>

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

```
28        <!-- MySQL Connector -->
29        <dependency>
30            <groupId>mysql</groupId>
31            <artifactId>mysql-connector-java</artifactId>
32            <version>8.0.33</version>
33        </dependency>
34
35        <!-- Spring Boot Web dependency (for MVC if needed) -->
36        <dependency>
37            <groupId>org.springframework.boot</groupId>
38            <artifactId>spring-boot-starter-web</artifactId>
39        </dependency>
40
41        <!-- Testing dependencies -->
42        <dependency>
43            <groupId>org.springframework.boot</groupId>
44            <artifactId>spring-boot-starter-test</artifactId>
45            <scope>test</scope>
46        </dependency>
47    </dependencies>
48
49    <build>
50        <plugins>
51            <plugin>
52                <groupId>org.springframework.boot</groupId>
53                <artifactId>spring-boot-maven-plugin</artifactId>
54            </plugin>
55        </plugins>
56    </build>
57    </project>
```

Com.mahasiswa – Pertemuan5SpringBootApplication.java

```
5     package com.mahasiswa;
6
7     import com.mahasiswa.controller.MahasiswaController;
8     import org.springframework.beans.factory.annotation.Autowired;
9     import org.springframework.boot.CommandLineRunner;
10    import org.springframework.boot.SpringApplication;
11    import org.springframework.boot.autoconfigure.SpringBootApplication;
12
13    /**
14     *
15     * @author Johan
16     */
17    @SpringBootApplication
18    public class Pertemuan5SpringBootApplication implements CommandLineRunner{
19
20        @Autowired
21        private MahasiswaController mhsController;
22        public static void main(String[] args) {
23            SpringApplication.run(Pertemuan5SpringBootApplication.class, args);
24        }
25
26        @Override
27        public void run(String... args) throws Exception {
28            mhsController.tampilkanMenu();
29        }
30
31    }
```

Com.mahasiswa.controller – MahasiswaController.java

```java
package com.mahasiswa.controller;

import com.mahasiswa.model.ModelMahasiswa;
import com.mahasiswa.repository.MahasiswaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

import java.util.List;
import java.util.Scanner;

/**
 *
 * @author Johan
 */
@Controller
public class MahasiswaController {
    @Autowired
    private MahasiswaRepository mahasiswaRepository;

    public void tampilkanMenu() {
        Scanner scanner = new Scanner(System.in);
        int opsi;

        do {
            System.out.println("\nMenu:");
            System.out.println("1. Tampilkan semua mahasiswa");
            System.out.println("2. Tambah mahasiswa baru");
            System.out.println("3. Cek koneksi database");
            System.out.println("4. Keluar");
            System.out.print("Pilih opsi: ");
            opsi = scanner.nextInt();
            scanner.nextLine(); // menangkap newline
            switch (opsi) {
                case 1:
                    tampilkanSemuaMahasiswa();
                    break;
                case 2:
                    tambahMahasiswa(scanner);
                    break;
                case 3:
                    cekKoneksi();
                    break;
                case 4:
                    System.out.println("Keluar dari program.");
                    break;
                default:
                    System.out.println("Opsi tidak valid, coba lagi.");
            }

        } while (opsi != 4);
    }

    private void tampilkanSemuaMahasiswa() {
        List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
        if (mahasiswaList.isEmpty()) {
            System.out.println("Tidak ada data mahasiswa.");
        } else {
            mahasiswaList.forEach(mahasiswa -> System.out.println(mahasiswa));
        }
    }
```

Com.mahasiswa.model – ModelMahasiswa.java

```java
package com.mahasiswa.model;

import jakarta.persistence.*;

/**
 *
 * @author Johan
 */
@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;
```

```java
    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;
    }
```

```java
    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;
    }

    @Override
    public String toString() {
        return "Mahasiswa{" +
            "id=" + id +
            ", npm='" + npm + '\'' +
            ", nama='" + nama + '\'' +
            ", semester=" + semester +
            ", ipk=" + ipk +
            '}';
    }

}
```

Com.mahasiswa.repository – MahasiswaRepository.java

```java
package com.mahasiswa.repository;

import com.mahasiswa.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 *
 * @author Johan
 */
@Repository
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {

}
```

Output:

1. Cek koneksi database

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 3
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Koneksi ke database berhasil.
```

2. Menambah data ke-1 mahasiswa baru

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 2
Masukkan NPM : 50421686
Masukkan Nama : Johan
Masukkan Semester : 4
Masukkan IPK : 4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa berhasil ditambahkan.
```

3. Menambah data ke-2 mahasiswa baru

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 2
Masukkan NPM : 50421687
Masukkan Nama : Karina
Masukkan Semester : 6
Masukkan IPK : 3.9
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa berhasil ditambahkan.
```

4. Menambah data ke-3 mahasiswa baru

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 2
Masukkan NPM : 50421688
Masukkan Nama : NingNing
Masukkan Semester : 5
Masukkan IPK : 3.97
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa berhasil ditambahkan.
```

5. Menampilkan data semua mahasiswa baru yang telah ditambahkan ke database

```
Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih opsi: 1
Hibernate: select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0
Mahasiswa{id=1, npm='50421686', nama='Johan', semester=4, ipk=4.0}
Mahasiswa{id=2, npm='50421687', nama='Karina', semester=6, ipk=3.9}
Mahasiswa{id=3, npm='50421688', nama='NingNing', semester=5, ipk=3.97}
```

ACTIVITY PERTEMUAN 6

NAMA : **Johan**

NPM : **50421686**

KELAS : **4IA28**

MATERI : **Spring AOP**

MATA PRAKTIKUM : **Rekayasa Perangkat Lunak 2**

Source Code:

Application.properties

```
# Konfigurasi MySQL Hibernate
spring.datasource.url=jdbc:mysql://localhost:3306/rpl_pert6?useSSL=false&serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# Hibernate settings
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>RPL_Johan_SpringAOP</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <exec.mainClass>com.mycompany.rpl_johan_springaop.RPL_Johan_SpringAOP</exec.mainClass>
    </properties>
    <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.3</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>
```

```java
        <!-- 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>
```

Com.mahasiswa – MahasiswaApp.java

```java
package com.mahasiswa;

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

/**
 *
 * @author Johan
 */

@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);
    }
```

```java
    @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.
    }
}
```

Com.mahasiswa.controller – MahasiswaController.java

```java
package com.mahasiswa.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.mahasiswa.model.ModelMahasiswa;
import com.mahasiswa.service.MahasiswaService;

import java.util.List;
import org.springframework.stereotype.Controller;


@Controller
public class MahasiswaController {

    @Autowired
    private MahasiswaService mahasiswaService;

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

    // Get Mahasiswa by ID
    public ModelMahasiswa getMahasiswa(@PathVariable int id) {
        return mahasiswaService.getMhs(id);
    }

    // Update Mahasiswa
    public String updateMahasiswa(@RequestBody ModelMahasiswa mhs) {
        mahasiswaService.updateMhs(mhs);
        return "Mahasiswa updated successfully";
    }
    // Delete Mahasiswa by ID
    public String deleteMahasiswa(@PathVariable int id) {
        mahasiswaService.deleteMhs(id);
        return "Mahasiswa deleted successfully";
    }

    // Get all Mahasiswa
    public List<ModelMahasiswa> getAllMahasiswa() {
        return mahasiswaService.getAllMahasiswa();
    }
}
```

Com.mahasiswa.model – ModelMahasiswa.java

```java
package com.mahasiswa.model;

import jakarta.persistence.*;

/**
 *
 * @author Johan
 */

@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(){

    }
```

```java
    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;
    }
```

```java
    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;
    }
}
```

Com.mahasiswa.model – ModelTableMahasiswa.java

```java
package com.mahasiswa.model;

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

/**
 *
 * @author Johan
 */
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;
        }
    }
```

```java
    @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
    }
}
```

Com.mahasiswa.repository – MahasiswaRepository.java

```java
package com.mahasiswa.repository;

import com.mahasiswa.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Integer> {
    public Object findById(int id);

    public void deleteById(int id);

}
```

Com.mahasiswa.service – MahasiswaService.java

```java
package com.mahasiswa.service;

import com.mahasiswa.model.ModelMahasiswa;
import com.mahasiswa.repository.MahasiswaRepository;
import jakarta.transaction.Transactional;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

@Service
public class MahasiswaService {

    @Autowired
    private MahasiswaRepository repository;

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

    public ModelMahasiswa getMhs(int id) {
        ModelMahasiswa mahasiswa = (ModelMahasiswa) repository.findById(id);
        return mahasiswa != null ? mahasiswa : null;
    }
```

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

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

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

Com.mahasiswa.view – MahasiswaView.java

```java
package com.mahasiswa.view;

import com.mahasiswa.controller.MahasiswaController;
import com.mahasiswa.model.ModelMahasiswa;
import com.mahasiswa.model.ModelTabelMahasiswa;
import java.util.List;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

/**
 *
 * @author Johan
 */
public class MahasiswaView extends javax.swing.JFrame {
    private MahasiswaController controller;

    /**
     * Creates new form MahasiswaView
     */
    public MahasiswaView(MahasiswaController controller) {
        this.controller = controller;
        initComponents();
        loadMahasiswaTable();
    }
    private MahasiswaView() {
        throw new UnsupportedOperationException("Not supported yet."); // Gen
    }

    public void loadMahasiswaTable() {
    // Ambil data dari controller
    List<ModelMahasiswa> listMahasiswa = controller.getAllMahasiswa();

    // Buat model tabel kustom dengan data mahasiswa
    ModelTabelMahasiswa tableModel = new ModelTabelMahasiswa(listMahasiswa);

    // Set model pada JTable
    dataTable.setModel(tableModel);
}
```

```java
    private void simpanButtonActionPerformed(java.awt.event.ActionEvent evt) {
        String npm = getNpmField().getText();
        String nama = getNamaField().getText();
        int semester = Integer.parseInt(getSemesterField().getText());
        float ipk = Float.parseFloat(getIpkField().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.addMahasiswa(mahasiswa);
        loadMahasiswaTable();
    }

    private void buangButtonActionPerformed(java.awt.event.ActionEvent evt) {
        JTextField idField = new JTextField(5);

    // Membuat panel untuk menampung JTextField
    JPanel panel = new JPanel();
    panel.add(new JLabel("Masukkan ID yang ingin dihapus:"));
    panel.add(idField);

    // Menampilkan dialog box dengan JTextField, tombol OK, dan Cancel
    int result = JOptionPane.showConfirmDialog(null, panel,
        "Hapus Mahasiswa", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
```

```java
    // Jika tombol OK ditekan
    if (result == JOptionPane.OK_OPTION) {
        try {
            // Mengambil input ID dan memanggil metode deleteMhs
            int id = Integer.parseInt(idField.getText());
            controller.deleteMahasiswa(id);
            JOptionPane.showMessageDialog(null, "Data berhasil dihapus.", "Sukses", JOptionPane.INFORMATION_MESSAGE);
        } catch (NumberFormatException e) {
            // Menangani error jika ID yang dimasukkan bukan angka
            JOptionPane.showMessageDialog(null, "ID harus berupa angka.", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    loadMahasiswaTable();
    }
```

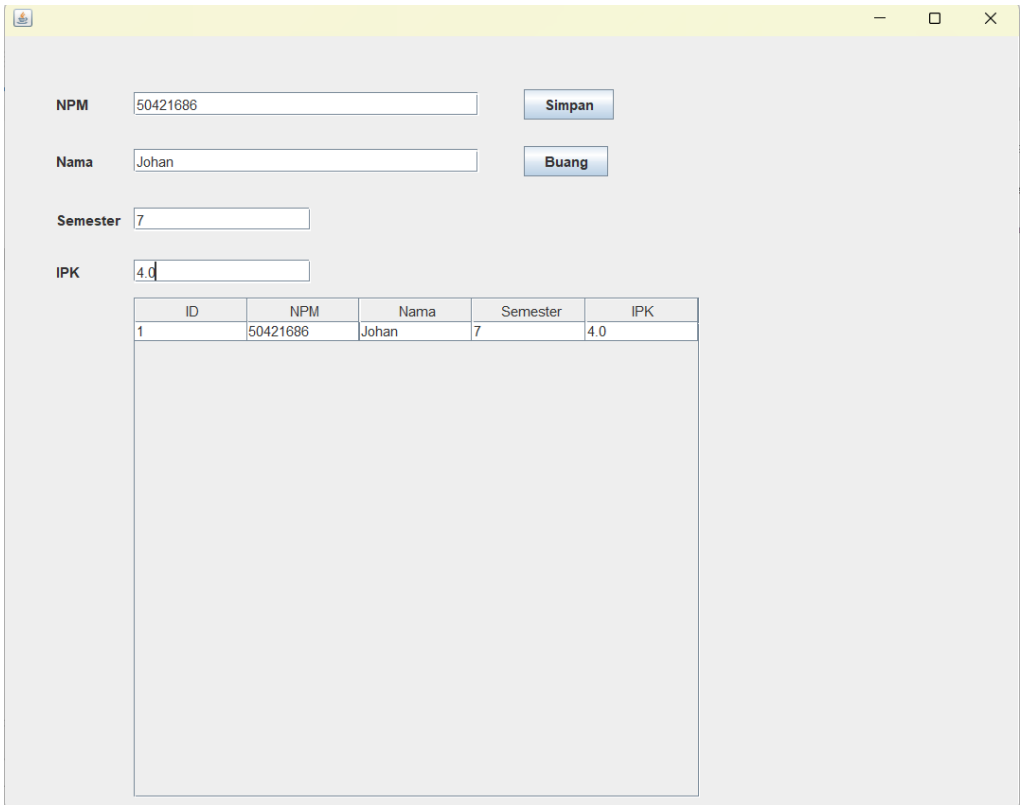
```java
    loadMahasiswaTable();
    }

    public JTextField getIpkField() {
        return ipkField;
    }

    public void setIpkField(JTextField ipkField) {
        this.ipkField = ipkField;
    }

    public JTextField getNamaField() {
        return namaField;
    }

    public void setNamaField(JTextField namaField) {
        this.namaField = namaField;
    }

    public JTextField getNpmField() {
        return npmField;
    }

    public void setNpmField(JTextField npmField) {
        this.npmField = npmField;
    }
```

```
227     public JTextField getSemesterField() {
228         return semesterField;
229     }
230
231     public void setSemesterField(JTextField semesterField) {
232         this.semesterField = semesterField;
233     }
```

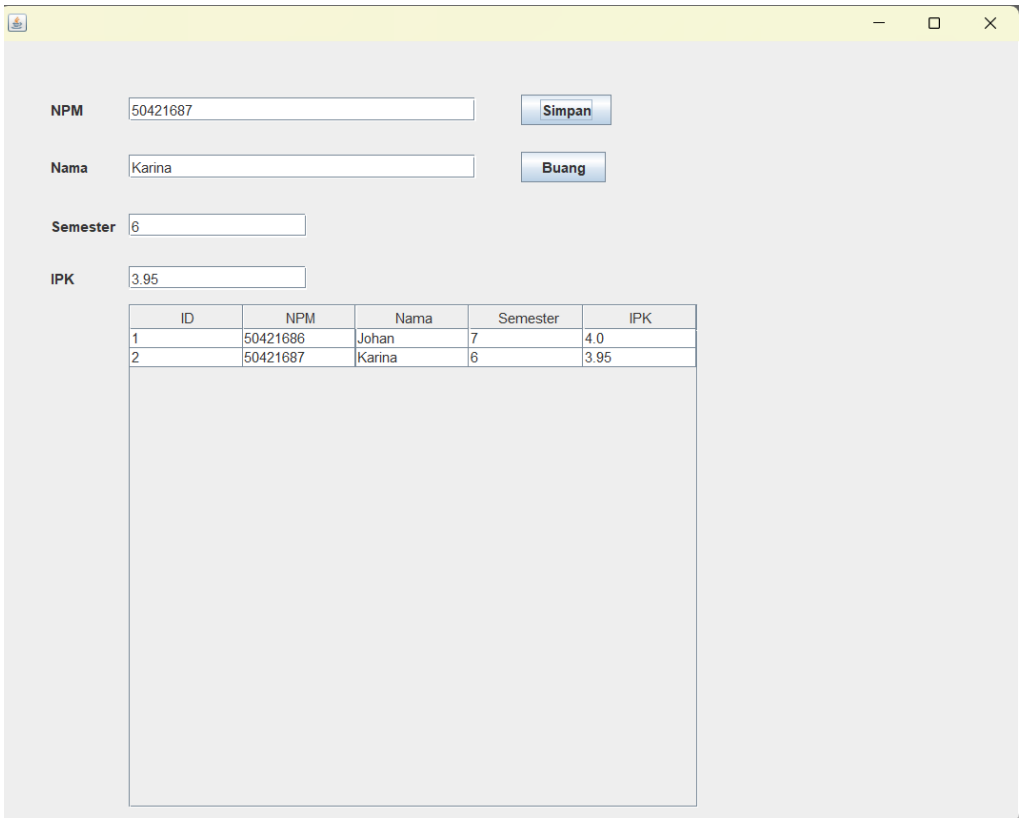
```
238     public static void main(String args[]) {
239         /* Set the Nimbus look and feel */
240         Look and feel setting code (optional)
261
262         /* Create and display the form */
            java.awt.EventQueue.invokeLater(new Runnable() {
                public void run() {
265                 new MahasiswaView().setVisible(true);
266             }
267         });
268     }
269
270     // Variables declaration - do not modify
271     private javax.swing.JButton buangButton;
272     private javax.swing.JTable dataTable;
273     private javax.swing.JTextField ipkField;
274     private javax.swing.JLabel jLabel1;
275     private javax.swing.JLabel jLabel2;
276     private javax.swing.JLabel jLabel3;
277     private javax.swing.JLabel jLabel4;
278     private javax.swing.JScrollPane jScrollPane1;
279     private javax.swing.JTextField namaField;
280     private javax.swing.JTextField npmField;
281     private javax.swing.JTextField semesterField;
282     private javax.swing.JButton simpanButton;
283     // End of variables declaration
284 }
```

Output:

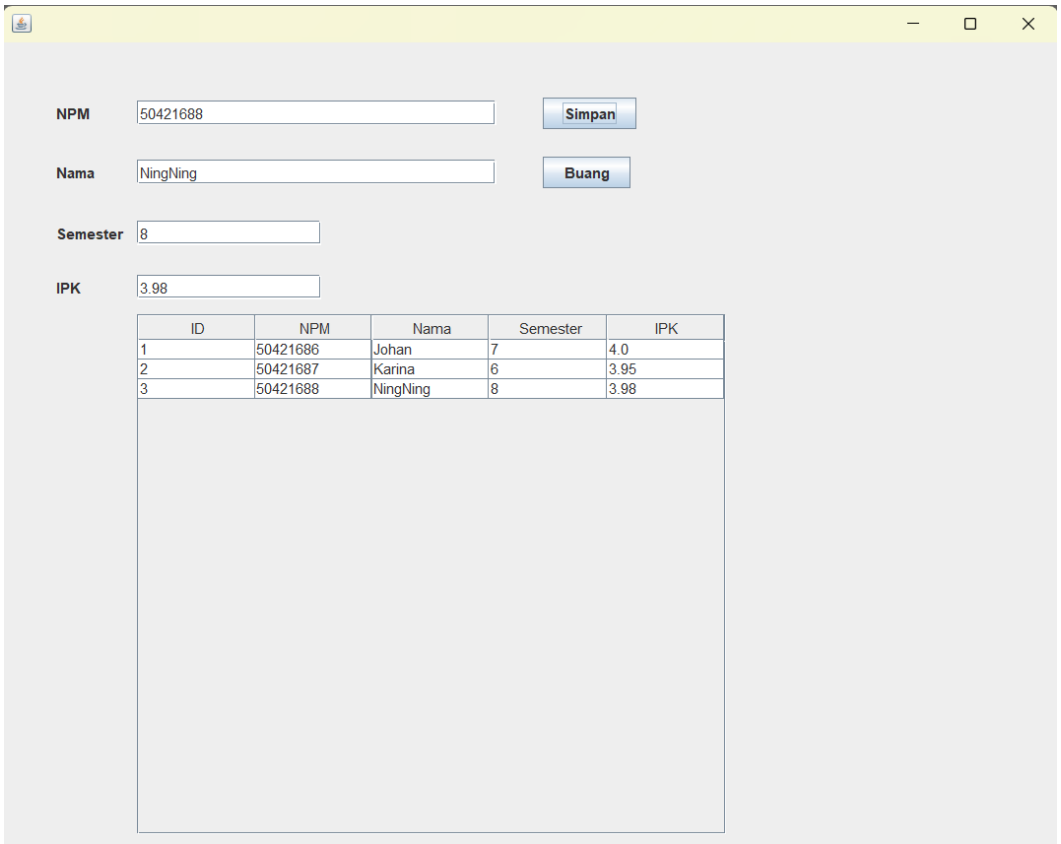
1. Menambahkan data ke-1



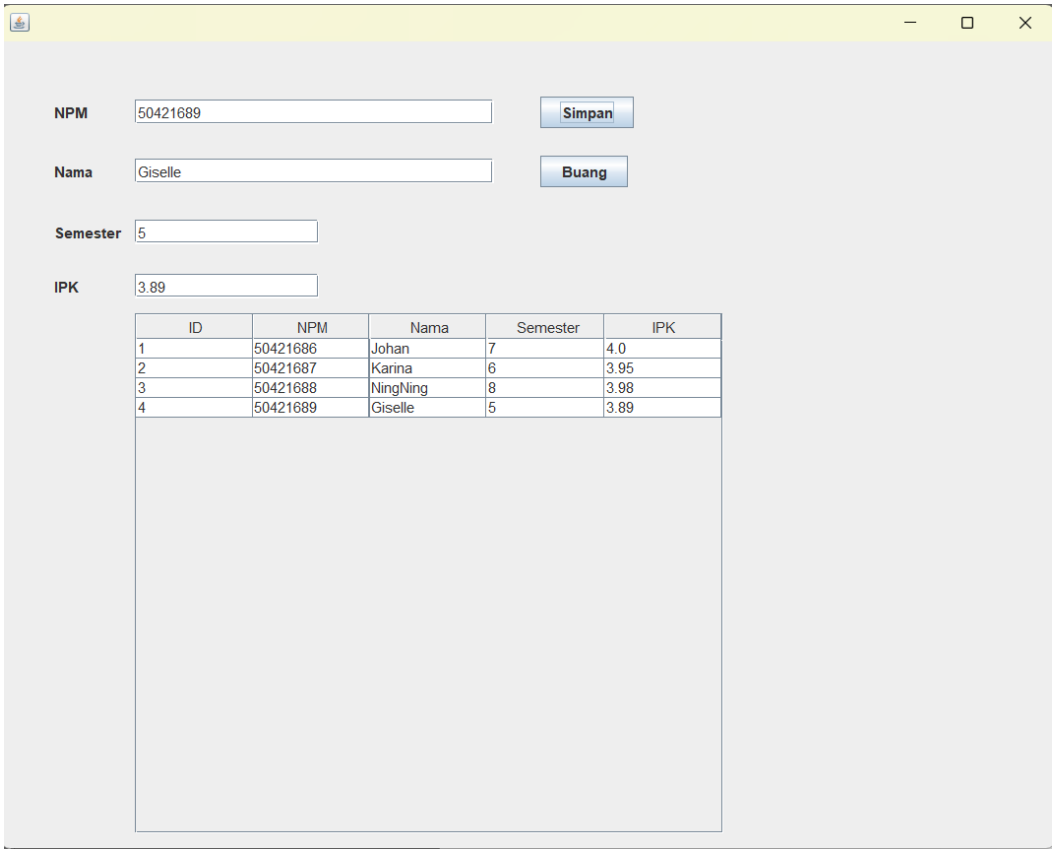
2. Menambahkan data ke-2



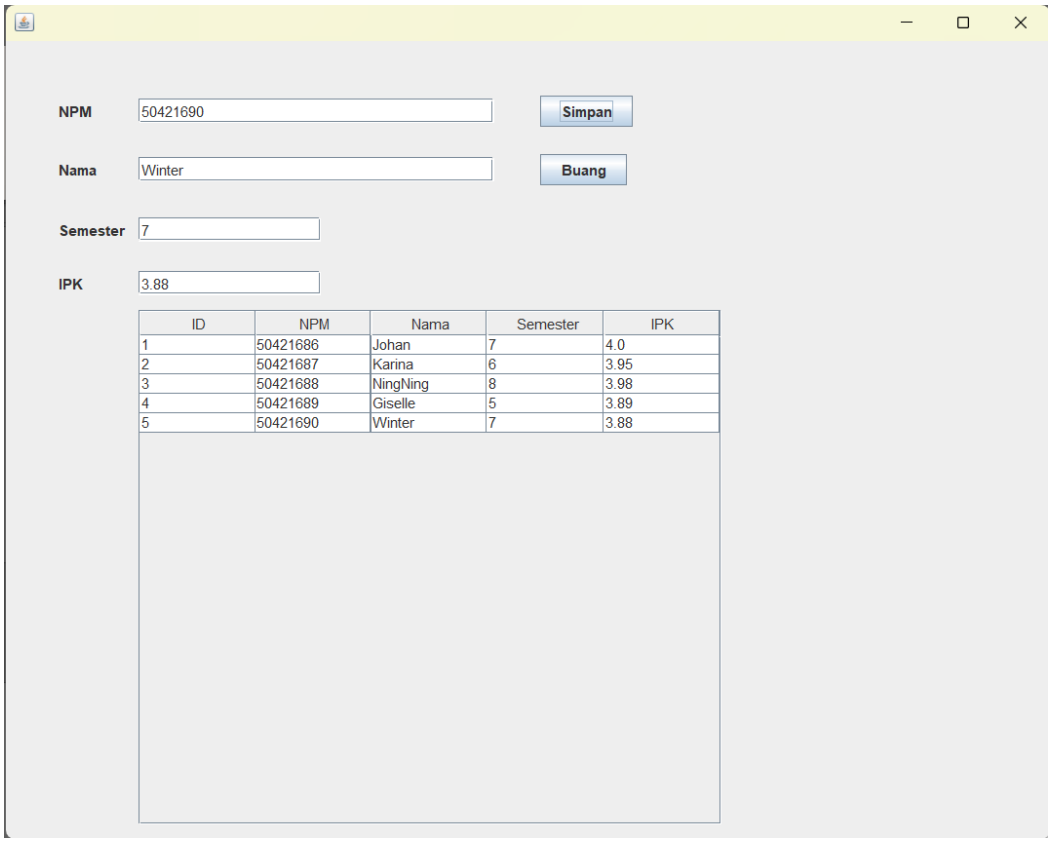
3. Menambahkan data ke-3



4. Menambahkan data ke-4

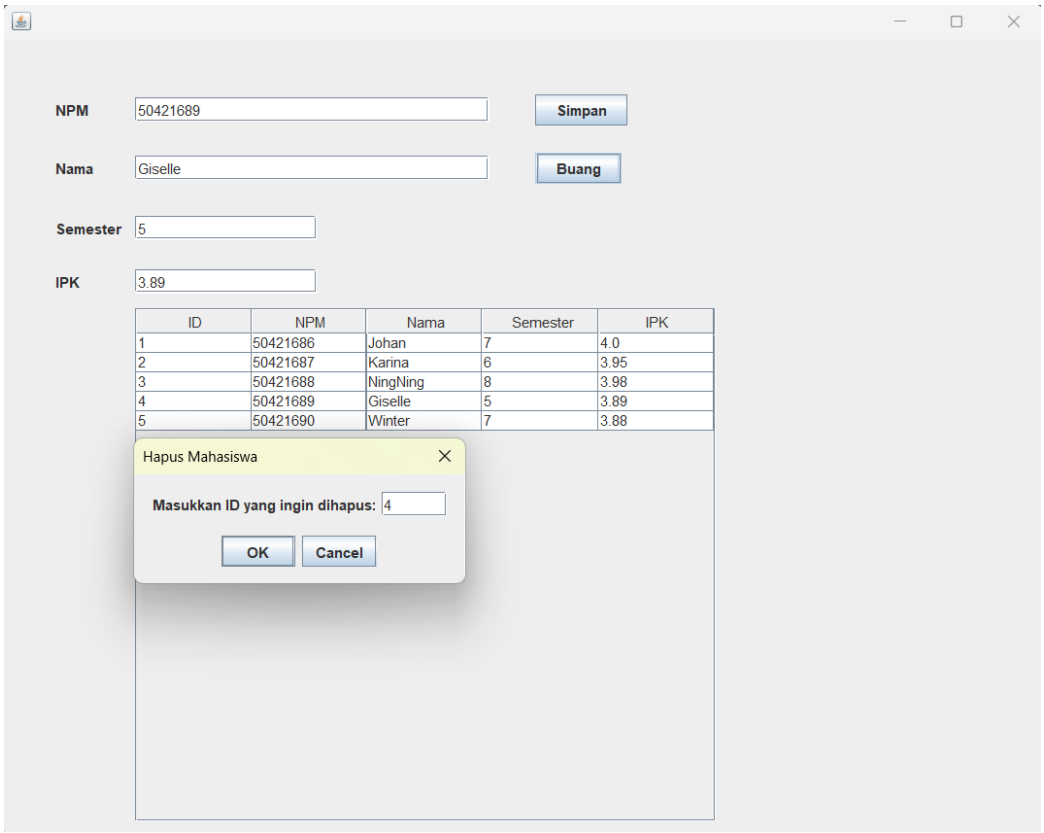


5. Menambahkan data ke-5



6. Menghapus 1 data (Data dengan ID 4 yang dihapus)

   a. Konfirmasi ID untuk data yang ingin dihapus



b. Tampilan tabel setelah data dengan ID no.4 dihapus