

ACTIVITY PERTEMUAN 5

NAMA : Johan

NPM : 50421686

KELAS : 4IA28

MATERI : Spring Boot

MATA PRAKTIKUM : Rekayasa Perangkat Lunak 2

Source Code:

Application.properties

```
5 # Konfigurasi MySQL Hibernate
6 spring.datasource.url=jdbc:mysql://localhost:3306/rpl_pert5?useSSL=false&serverTimezone=UTC
7 spring.datasource.username=root
8 spring.datasource.password=
9 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
10
11 # Hibernate settings
12 spring.jpa.hibernate.ddl-auto=update
13 spring.jpa.show-sql=true
14
```

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"
3     <modelVersion>4.0.0</modelVersion>
4     <groupId>com.mycompany</groupId>
5     <artifactId>RPL_Johan_Spring</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.source>21</maven.compiler.source>
11         <maven.compiler.target>21</maven.compiler.target>
12         <exec.mainClass>com.mycompany.rpl_johan_spring.RPL_Johan_Spring</exec.mainClass>
13     </properties>
14     <parent>
15         <groupId>org.springframework.boot</groupId>
16         <artifactId>spring-boot-starter-parent</artifactId>
17         <version>3.3.3</version>
18         <relativePath/>
19     </parent>
20
21     <dependencies>
22         <!-- Hibernate + Spring Data JPA -->
23         <dependency>
24             <groupId>org.springframework.boot</groupId>
25             <artifactId>spring-boot-starter-data-jpa</artifactId>
26         </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

```
5 package com.mahasiswa.controller;
6
7 import com.mahasiswa.model.ModelMahasiswa;
8 import com.mahasiswa.repository.MahasiswaRepository;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.stereotype.Controller;
11
12 import java.util.List;
13 import java.util.Scanner;
14
15 /**
16  *
17  * @author Johan
18  */
19 @Controller
20 public class MahasiswaController {
21     @Autowired
22     private MahasiswaRepository mahasiswaRepository;
23
24     public void tampilkanMenu() {
25         Scanner scanner = new Scanner(System.in);
26         int opsi;
27
28         do {
29             System.out.println("\nMenu:");
30             System.out.println("1. Tampilkan semua mahasiswa");
31             System.out.println("2. Tambah mahasiswa baru");
32             System.out.println("3. Cek koneksi database");
33             System.out.println("4. Keluar");
34             System.out.print("Pilih opsi: ");
35             opsi = scanner.nextInt();
36             scanner.nextLine(); // menangkap newline
37             switch (opsi) {
38                 case 1:
39                     tampilkanSemuaMahasiswa();
40                     break;
41                 case 2:
42                     tambahMahasiswa(scanner);
43                     break;
44                 case 3:
45                     cekKoneksi();
46                     break;
47                 case 4:
48                     System.out.println("Keluar dari program.");
49                     break;
50                 default:
51                     System.out.println("Opsi tidak valid, coba lagi.");
52             }
53         } while (opsi != 4);
54     }
55
56     private void tampilkanSemuaMahasiswa() {
57         List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
58         if (mahasiswaList.isEmpty()) {
59             System.out.println("Tidak ada data mahasiswa.");
60         } else {
61             mahasiswaList.forEach(mahasiswa -> System.out.println(mahasiswa));
62         }
63     }
64 }
65
```

Com.mahasiswa.model – ModelMahasiswa.java

```
5 package com.mahasiswa.model;
6
7 import jakarta.persistence.*;
8
9 /**
10  *
11  * @author Johan
12  */
13 @Entity
14 @Table(name = "mahasiswa")
15
16 public class ModelMahasiswa {
17
18     @Id
19     @GeneratedValue(strategy = GenerationType.IDENTITY)
20     @Column(name = "id")
21     private int id;
22
23     @Column(name = "npm", nullable = false, length = 8)
24     private String npm;
25
26     @Column(name = "nama", nullable = false, length = 50)
27     private String nama;
28
29     @Column(name = "semester")
30     private int semester;
31
32     @Column(name = "ipk")
33     private float ipk;
34
35     public ModelMahasiswa() {
36     }
37
38     public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
39         this.id = id;
40         this.npm = npm;
41         this.nama = nama;
42         this.semester = semester;
43         this.ipk = ipk;
44     }
45
46     public int getId() {
47         return id;
48     }
49
50     public void setId(int id) {
51         this.id = id;
52     }
53
54     public String getNpm() {
55         return npm;
56     }
57
58     public void setNpm(String npm) {
59         this.npm = npm;
60     }
61
62     public String getNama() {
63         return nama;
64     }
65
66     public void setNama(String nama) {
67         this.nama = nama;
68     }
69
70 }
```

```

72     public int getSemester() {
73         return semester;
74     }
75
76     public void setSemester(int semester) {
77         this.semester = semester;
78     }
79
80     public float getIpk() {
81         return ipk;
82     }
83
84     public void setIpk(float ipk) {
85         this.ipk = ipk;
86     }
87
88     @Override
89     public String toString() {
90         return "Mahasiswa{" +
91             "id=" + id +
92             ", npm='" + npm + '\'' +
93             ", nama='" + nama + '\'' +
94             ", semester=" + semester +
95             ", ipk=" + ipk +
96             '}';
97     }
98
99 }

```

Com.mahasiswa.repository – MahasiswaRepository.java

```

5     package com.mahasiswa.repository;
6
7     import com.mahasiswa.model.ModelMahasiswa;
8     import org.springframework.data.jpa.repository.JpaRepository;
9     import org.springframework.stereotype.Repository;
10
11     /**
12      *
13      * @author Johan
14      */
15     @Repository
16     public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {
17
18     }

```

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

```
6 # Konfigurasi MySQL Hibernate
7 spring.datasource.url=jdbc:mysql://localhost:3306/rpl_pert6?useSSL=false&serverTimezone=UTC
8 spring.datasource.username=root
9 spring.datasource.password=
10 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
11
12 # Hibernate settings
13 spring.jpa.hibernate.ddl-auto=update
14 spring.jpa.show-sql=true
```

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
3 <modelVersion>4.0.0</modelVersion>
4 <groupId>com.mycompany</groupId>
5 <artifactId>RPL_Johan_SpringAOP</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.source>21</maven.compiler.source>
11 <maven.compiler.target>21</maven.compiler.target>
12 <exec.mainClass>com.mycompany.rpl_johan_springaop.RPL_Johan_SpringAOP</exec.mainClass>
13 </properties>
14 <parent>
15 <groupId>org.springframework.boot</groupId>
16 <artifactId>spring-boot-starter-parent</artifactId>
17 <version>3.3.3</version>
18 <relativePath/>
19 </parent>
20
21 <dependencies>
22 <!-- Hibernate + Spring Data JPA -->
23 <dependency>
24 <groupId>org.springframework.boot</groupId>
25 <artifactId>spring-boot-starter-data-jpa</artifactId>
26 </dependency>
27
28 <!-- MySQL Connector -->
29 <dependency>
30 <groupId>mysql</groupId>
31 <artifactId>mysql-connector-java</artifactId>
32 <version>8.0.33</version>
33 </dependency>
```



```

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 – MahasiswaApp.java

```

5  package com.mahasiswa;
6
7  import com.mahasiswa.controller.MahasiswaController;
8  import com.mahasiswa.service.MahasiswaService;
9  import com.mahasiswa.view.MahasiswaView;
10 import org.springframework.boot.ApplicationArguments;
11 import org.springframework.boot.ApplicationRunner;
12 import org.springframework.boot.SpringApplication;
13 import org.springframework.boot.autoconfigure.SpringBootApplication;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.context.ApplicationContext;
16
17 /**
18  *
19  * @author Johan
20  */
21
22 @SpringBootApplication
23 public class MahasiswaApp implements ApplicationRunner {
24
25     @Autowired
26     private MahasiswaService mahasiswaService;
27
28     public static void main(String[] args) {
29         System.setProperty("java.awt.headless", "false"); // Disable headless mode
30
31         // Start the Spring application and get the application context
32         ApplicationContext context = SpringApplication.run(MahasiswaApp.class, args);
33
34         // Instantiate the view and inject the controller manually
35         MahasiswaController controller = context.getBean(MahasiswaController.class);
36         MahasiswaView mahasiswaView = new MahasiswaView(controller);
37         mahasiswaView.setVisible(true);
38     }

```

```

40      @Override
41      public void run(ApplicationArguments args) throws Exception {
42          // Implement this method if you need to execute logic after Spring application starts
43          // Otherwise, you can leave it as is.
44      }
45  }

```

Com.mahasiswa.controller – MahasiswaController.java

```

1  package com.mahasiswa.controller;
2
3  import org.springframework.beans.factory.annotation.Autowired;
4  import org.springframework.web.bind.annotation.*;
5  import com.mahasiswa.model.ModelMahasiswa;
6  import com.mahasiswa.service.MahasiswaService;
7
8  import java.util.List;
9  import org.springframework.stereotype.Controller;
10
11
12  @Controller
13  public class MahasiswaController {
14
15      @Autowired
16      private MahasiswaService mahasiswaService;
17
18      // Add new Mahasiswa
19      public String addMahasiswa(@RequestBody ModelMahasiswa mhs) {
20          mahasiswaService.addMhs(mhs);
21          return "Mahasiswa added successfully";
22      }
23
24      // Get Mahasiswa by ID
25      public ModelMahasiswa getMahasiswa(@PathVariable int id) {
26          return mahasiswaService.getMhs(id);
27      }
28
29      // Update Mahasiswa
30      public String updateMahasiswa(@RequestBody ModelMahasiswa mhs) {
31          mahasiswaService.updateMhs(mhs);
32          return "Mahasiswa updated successfully";
33      }
34
35      // Delete Mahasiswa by ID
36      public String deleteMahasiswa(@PathVariable int id) {
37          mahasiswaService.deleteMhs(id);
38          return "Mahasiswa deleted successfully";
39      }
40
41      // Get all Mahasiswa
42      public List<ModelMahasiswa> getAllMahasiswa() {
43          return mahasiswaService.getAllMahasiswa();
44      }
45  }

```

Com.mahasiswa.model – ModelMahasiswa.java

```
5 package com.mahasiswa.model;
6
7 import jakarta.persistence.*;
8
9 /**
10  *
11  * @author Johan
12  */
13
14 @Entity
15 @Table(name = "mahasiswa")
16
17 public class ModelMahasiswa {
18
19     @Id
20     @GeneratedValue(strategy = GenerationType.IDENTITY)
21     @Column(name = "id")
22     private int id;
23
24     @Column(name = "npm", nullable = false, length = 8)
25     private String npm;
26
27     @Column(name = "nama", nullable = false, length = 50)
28     private String nama;
29
30     @Column(name = "semester")
31     private int semester;
32
33     @Column(name = "ipk")
34     private float ipk;
35
36     public ModelMahasiswa() {
37
38     }
39
40     public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
41         this.id = id;
42         this.npm = npm;
43         this.nama = nama;
44         this.semester = semester;
45         this.ipk = ipk;
46     }
47
48     public int getId() {
49         return id;
50     }
51
52     public void setId(int id) {
53         this.id = id;
54     }
55
56     public String getNpm() {
57         return npm;
58     }
59
60     public void setNpm(String npm) {
61         this.npm = npm;
62     }
63
64     public String getNama() {
65         return nama;
66     }
```

```

68     public void setName(String nama) {
69         this.nama = nama;
70     }
71
72     public int getSemester() {
73         return semester;
74     }
75
76     public void setSemester(int semester) {
77         this.semester = semester;
78     }
79
80     public float getIpk() {
81         return ipk;
82     }
83
84     public void setIpk(float ipk) {
85         this.ipk = ipk;
86     }
87 }

```

Com.mahasiswa.model – ModelTableMahasiswa.java

```

5     package com.mahasiswa.model;
6
7     import javax.swing.table.AbstractTableModel;
8     import java.util.List;
9
10    /**
11     *
12     * @author Johan
13     */
14    public class ModelTabelMahasiswa extends AbstractTableModel{
15        private List<ModelMahasiswa> mahasiswaList;
16        private String[] columnNames = {"ID", "NPM", "Nama", "Semester", "IPK"};
17
18        public ModelTabelMahasiswa(List<ModelMahasiswa> mahasiswaList) {
19            this.mahasiswaList = mahasiswaList;
20        }
21
22        @Override
23        public int getRowCount() {
24            return mahasiswaList.size(); // Jumlah baris sesuai dengan jumlah data mahasiswa
25        }
26
27        @Override
28        public int getColumnCount() {
29            return columnNames.length; // Jumlah kolom sesuai dengan jumlah elemen dalam columnNames
30        }
31
32        @Override
33        public Object getValueAt(int rowIndex, int columnIndex) {
34            ModelMahasiswa mahasiswa = mahasiswaList.get(rowIndex);
35            switch (columnIndex) {
36                case 0:
37                    return mahasiswa.getId();
38                case 1:
39                    return mahasiswa.getNpm();
40                case 2:
41                    return mahasiswa.getNama();
42                case 3:
43                    return mahasiswa.getSemester();
44                case 4:
45                    return mahasiswa.getIpk();
46                default:
47                    return null;
48            }
49        }

```

```

51      @Override
52      public String getColumnName(int column) {
53          return columnNames[column]; // Mengatur nama kolom
54      }
55
56      @Override
57      public boolean isCellEditable(int rowIndex, int columnIndex) {
58          return false; // Semua sel tidak dapat diedit
59      }
60
61      // Method untuk menambahkan atau memodifikasi data, jika dibutuhkan
62      public void setMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
63          this.mahasiswaList = mahasiswaList;
64          fireTableDataChanged(); // Memberitahu JTable bahwa data telah berubah
65      }
66  }

```

Com.mahasiswa.repository – MahasiswaRepository.java

```

1  package com.mahasiswa.repository;
2
3  import com.mahasiswa.model.ModelMahasiswa;
4  import org.springframework.data.jpa.repository.JpaRepository;
5  import org.springframework.stereotype.Repository;
6
7  @Repository
8  public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Integer> {
9      public Object findById(int id);
10
11      public void deleteById(int id);
12
13  }

```

Com.mahasiswa.service – MahasiswaService.java

```

1  package com.mahasiswa.service;
2
3  import com.mahasiswa.model.ModelMahasiswa;
4  import com.mahasiswa.repository.MahasiswaRepository;
5  import jakarta.transaction.Transactional;
6  import java.util.List;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.stereotype.Service;
9
10 @Service
11 public class MahasiswaService {
12
13     @Autowired
14     private MahasiswaRepository repository;
15
16     public void addMhs(ModelMahasiswa mhs) {
17         repository.save(mhs);
18     }
19
20     public ModelMahasiswa getMhs(int id) {
21         ModelMahasiswa mahasiswa = (ModelMahasiswa) repository.findById(id);
22         return mahasiswa != null ? mahasiswa : null;
23     }

```

```

25     public void updateMhs (ModelMahasiswa mhs) {
26         repository.save(mhs);
27     }
28
29     @Transactional
30     public void deleteMhs(int id) {
31         repository.deleteById(id);
32     }
33
34     public List<ModelMahasiswa> getAllMahasiswa() {
35         return repository.findAll();
36     }
37 }

```

Com.mahasiswa.view – MahasiswaView.java

```

5     package com.mahasiswa.view;
6
7     import com.mahasiswa.controller.MahasiswaController;
8     import com.mahasiswa.model.ModelMahasiswa;
9     import com.mahasiswa.model.ModelTabelMahasiswa;
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 Johan
19     */
20    public class MahasiswaView extends javax.swing.JFrame {
21        private MahasiswaController controller;
22
23        /**
24         * Creates new form MahasiswaView
25         */
26        public MahasiswaView(MahasiswaController controller) {
27            this.controller = controller;
28            initComponents();
29            loadMahasiswaTable();
30        }
31
32        private MahasiswaView() {
33            throw new UnsupportedOperationException("Not supported yet."); // Gen
34        }
35
36        public void loadMahasiswaTable() {
37            // Ambil data dari controller
38            List<ModelMahasiswa> listMahasiswa = controller.getAllMahasiswa();
39
40            // Buat model tabel kustom dengan data mahasiswa
41            ModelTabelMahasiswa tableModel = new ModelTabelMahasiswa(listMahasiswa);
42
43            // Set model pada JTable
44            dataTable.setModel(tableModel);
45        }

```

```

162 private void simpanButtonActionPerformed(java.awt.event.ActionEvent evt) {
163     String npm = getNpmField().getText();
164     String nama = getNamaField().getText();
165     int semester = Integer.parseInt(getSemesterField().getText());
166     float ipk = Float.parseFloat(getIpkField().getText());
167     ModelMahasiswa mahasiswa = new ModelMahasiswa(0, npm, nama, semester, ipk);
168     System.out.println(mahasiswa.getIpk());
169     System.out.println(mahasiswa.getNama());
170     System.out.println(mahasiswa.getSemester());
171     System.out.println(mahasiswa.getNpm());
172     controller.addMahasiswa(mahasiswa);
173     loadMahasiswaTable();
174 }
175
176 private void buangButtonActionPerformed(java.awt.event.ActionEvent evt) {
177     JTextField idField = new JTextField(5);
178
179     // Membuat panel untuk menampung JTextField
180     JPanel panel = new JPanel();
181     panel.add(new JLabel("Masukkan ID yang ingin dihapus:"));
182     panel.add(idField);
183
184     // Menampilkan dialog box dengan JTextField, tombol OK, dan Cancel
185     int result = JOptionPane.showConfirmDialog(null, panel,
186         "Hapus Mahasiswa", JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
187
188     // Jika tombol OK ditekan
189     if (result == JOptionPane.OK_OPTION) {
190         try {
191             // Mengambil input ID dan memanggil metode deleteMhs
192             int id = Integer.parseInt(idField.getText());
193             controller.deleteMahasiswa(id);
194             JOptionPane.showMessageDialog(null, "Data berhasil dihapus.", "Sukses", JOptionPane.INFORMATION_MESSAGE);
195         } catch (NumberFormatException e) {
196             // Menangani error jika ID yang dimasukkan bukan angka
197             JOptionPane.showMessageDialog(null, "ID harus berupa angka.", "Error", JOptionPane.ERROR_MESSAGE);
198         }
199     }
200     loadMahasiswaTable();
201 }
202
203 public JTextField getIpkField() {
204     return ipkField;
205 }
206
207 public void setIpkField(JTextField ipkField) {
208     this.ipkField = ipkField;
209 }
210
211 public JTextField getNamaField() {
212     return namaField;
213 }
214
215 public void setNamaField(JTextField namaField) {
216     this.namaField = namaField;
217 }
218
219 public JTextField getNpmField() {
220     return npmField;
221 }
222
223 public void setNpmField(JTextField npmField) {
224     this.npmField = npmField;
225 }

```

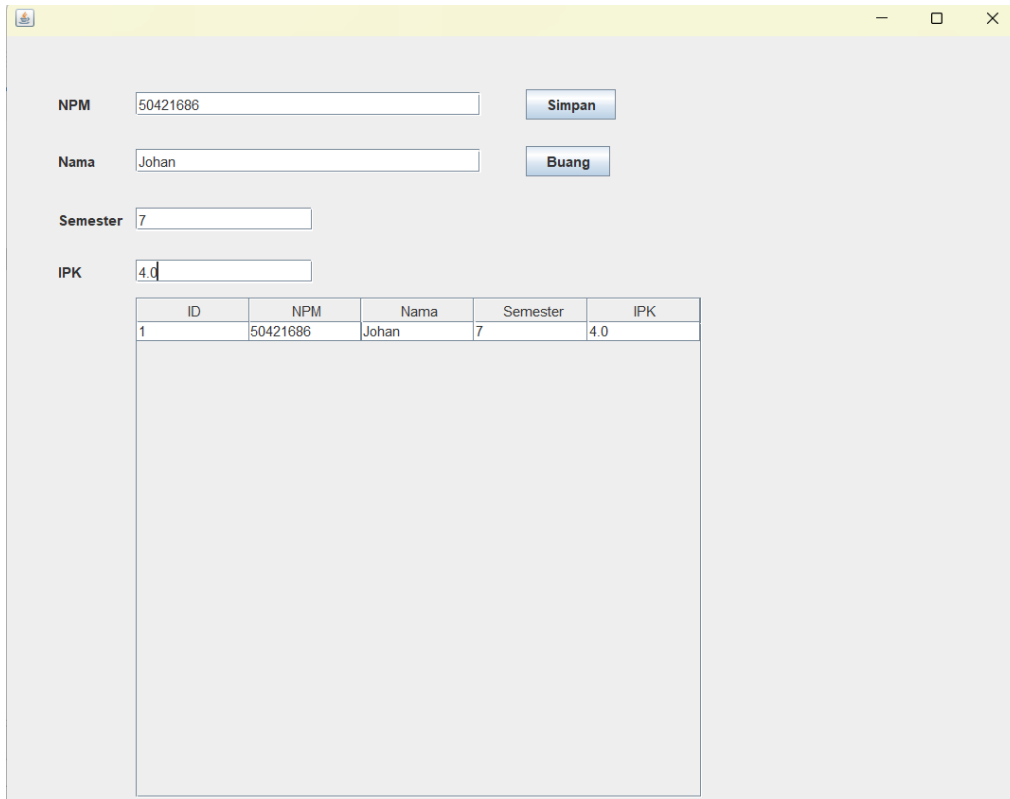
```

227     public JTextField getSemesterField() {
228         return semesterField;
229     }
230
231     public void setSemesterField(JTextField semesterField) {
232         this.semesterField = semesterField;
233     }
234
235     public static void main(String args[]) {
236         /* Set the Nimbus look and feel */
237         Look and feel setting code (optional)
238
239         /* Create and display the form */
240         java.awt.EventQueue.invokeLater(new Runnable() {
241             public void run() {
242                 new MahasiswaView().setVisible(true);
243             }
244         });
245     }
246
247     // Variables declaration - do not modify
248     private javax.swing.JButton buangButton;
249     private javax.swing.JTable dataTable;
250     private javax.swing.JTextField ipkField;
251     private javax.swing.JLabel jLabel1;
252     private javax.swing.JLabel jLabel2;
253     private javax.swing.JLabel jLabel3;
254     private javax.swing.JLabel jLabel4;
255     private javax.swing.JScrollPane jScrollPane1;
256     private javax.swing.JTextField namaField;
257     private javax.swing.JTextField npmField;
258     private javax.swing.JTextField semesterField;
259     private javax.swing.JButton simpanButton;
260     // End of variables declaration
261 }

```


Output:

1. Menambahkan data ke-1



NPM: 50421686

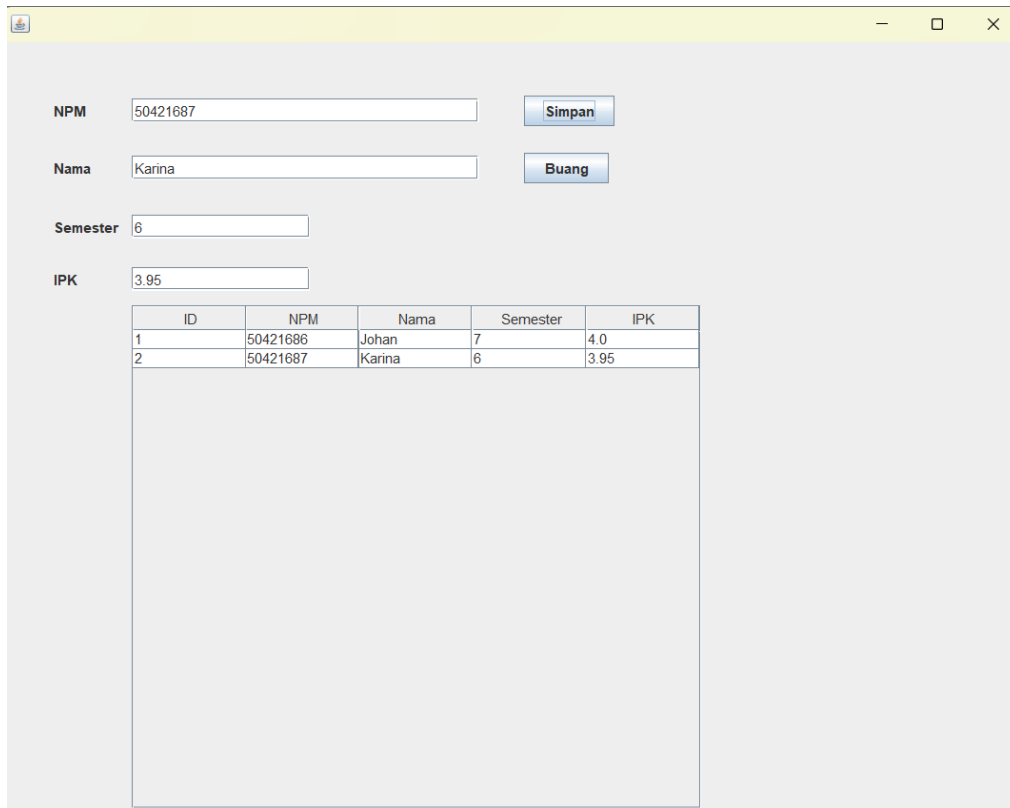
Nama: Johan

Semester: 7

IPK: 4.0

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0

2. Menambahkan data ke-2



NPM: 50421687

Nama: Karina

Semester: 6

IPK: 3.95

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95

3. Menambahkan data ke-3

NPM

50421688

Simpan

Nama

NingNing

Buang

Semester

8

IPK

3.98

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95
3	50421688	NingNing	8	3.98

4. Menambahkan data ke-4

NPM

50421689

Simpan

Nama

Giselle

Buang

Semester

5

IPK

3.89

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95
3	50421688	NingNing	8	3.98
4	50421689	Giselle	5	3.89

5. Menambahkan data ke-5

The screenshot shows a web application window with a light gray background. At the top, there are four input fields and two buttons. The first row has 'NPM' with the value '50421690' and a 'Simpan' button. The second row has 'Nama' with the value 'Winter' and a 'Buang' button. The third row has 'Semester' with the value '7'. The fourth row has 'IPK' with the value '3.88'. Below these fields is a table with 5 columns: ID, NPM, Nama, Semester, and IPK. The table contains 5 rows of data. Below the table is a large empty rectangular area.

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95
3	50421688	NingNing	8	3.98
4	50421689	Giselle	5	3.89
5	50421690	Winter	7	3.88

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

a. Konfirmasi ID untuk data yang ingin dihapus

The screenshot shows the same application window as before, but with different data. The 'NPM' field now contains '50421689', 'Nama' contains 'Giselle', 'Semester' contains '5', and 'IPK' contains '3.89'. The 'Simpan' and 'Buang' buttons are still present. The table below shows the same 5 rows of data as before. A modal dialog box titled 'Hapus Mahasiswa' is open in the foreground. It has a close button (X) in the top right corner. Inside the dialog, there is a label 'Masukkan ID yang ingin dihapus:' followed by an input field containing the number '4'. At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

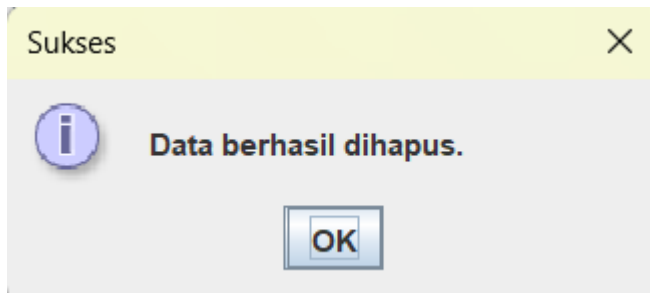
ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95
3	50421688	NingNing	8	3.98
4	50421689	Giselle	5	3.89
5	50421690	Winter	7	3.88

Hapus Mahasiswa

Masukkan ID yang ingin dihapus: 4

OK Cancel

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



The main application window has a yellow title bar with standard window controls. It contains a form with the following fields and buttons:

- NPM:** A text input field containing "50421689" and a "Simpan" button.
- Nama:** A text input field containing "Giselle" and a "Buang" button.
- Semester:** A text input field containing "5".
- IPK:** A text input field containing "3.89".

Below the form is a table with 5 columns: ID, NPM, Nama, Semester, and IPK. The table contains 5 rows of data:

ID	NPM	Nama	Semester	IPK
1	50421686	Johan	7	4.0
2	50421687	Karina	6	3.95
3	50421688	NingNing	8	3.98
5	50421690	Winter	7	3.88