

## ACTIVITY PERTEMUAN 5

NAMA : Fizri Rosdiansyah

NPM : 50421535

KELAS : 4IA06

MATERI : Framework Spring, Pembuatan Project Spring, dan Hibernate

MATA PRAKTIKUM : Rekayasa Perangkat Lunak 2

---

### Soal

1. Dependency injection (DI) adalah konsep penting dalam Spring Framework. Jelaskan apa itu dependency injection dan mengapa DI sangat penting dalam pengembangan aplikasi Spring Boot.

2. Screenshot Code dan Output

### Jawab

1. **Dependency Injection (DI)** adalah teknik desain yang memungkinkan sebuah objek menerima (atau "*di-inject*") dependensi yang dibutuhkan dari luar alih-alih membuatnya sendiri. Dalam konteks Spring Framework, DI adalah bagian dari konsep **Inversion of Control (IoC)**, yang berarti bahwa kontrol penciptaan dan pengelolaan objek berada di tangan container Spring, bukan di kelas itu sendiri.

Penting Karena :

- Meningkatkan Modularitas dan Fleksibilitas:
- Mendukung Pengujian Unit (Unit Testing):
- Pola Desain yang Lebih Bersih dan Lebih Rapi:
- Efisiensi dalam Manajemen Dependensi:
- Konsistensi Konfigurasi dan Pengelolaan Komponen:

2.

## A. Pertemuan5\_50421535

The image displays two screenshots of an IDE (likely IntelliJ IDEA) showing the development and execution of a Spring Boot application.

**Top Screenshot:** The source code of `Pertemuan5_50421535.java` is shown. The code includes package declarations, imports for `me.fizri.controller.MahasiswaController`, `org.springframework.beans.factory.annotation.Autowired`, `org.springframework.boot.CommandLineRunner`, `org.springframework.boot.SpringApplication`, and `org.springframework.boot.autoconfigure.SpringBootApplication`. The class `Pertemuan5_50421535` implements `CommandLineRunner` and contains an `@Autowired` field `MahasiswaController mhsController`.

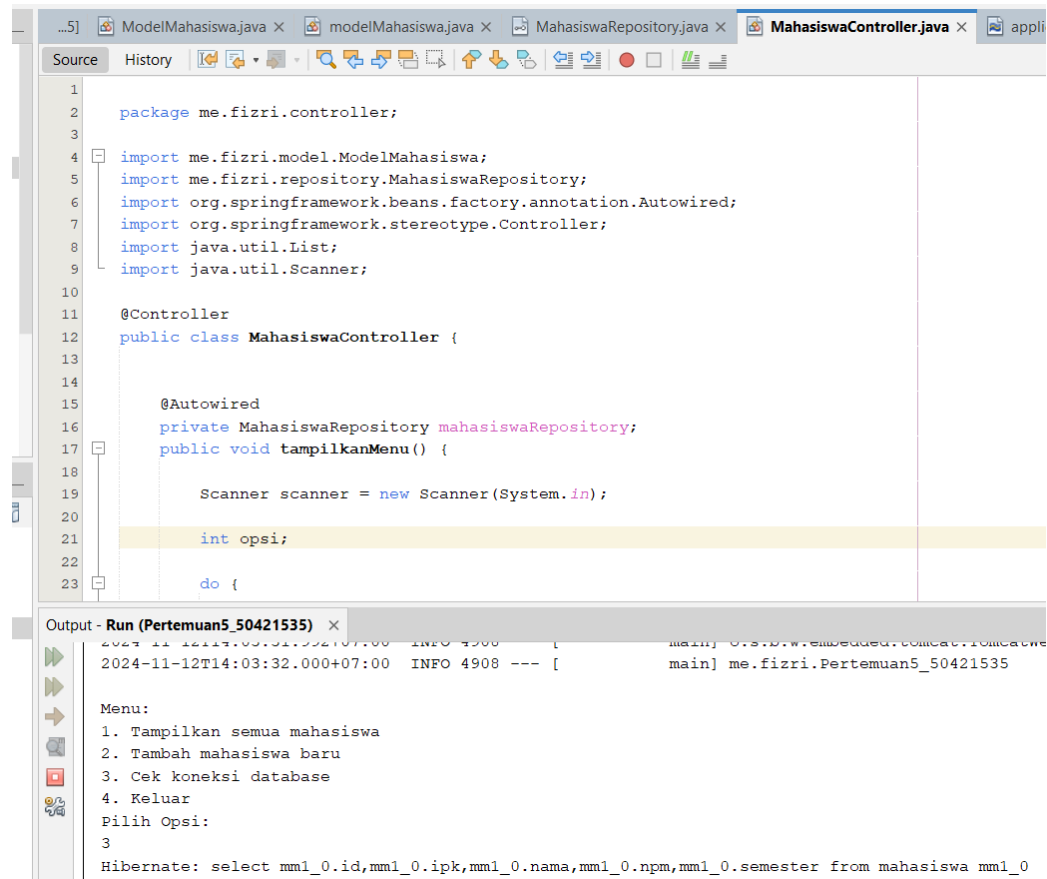
**Bottom Screenshot:** The source code of `Pertemuan5_50421535.java` is shown, including the `main` method and the `run` method. The `main` method calls `SpringApplication.run(Pertemuan5_50421535.class, args)`. The `run` method calls `mhsController.tampilkanMenu()`.

**Output - Run (Pertemuan5\_50421535):**

```
Masukkan ipk:
4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?,?,?,?)
Mahasiswa Berhasil ditambahkan.

Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
1
Hibernate: select mml_0.id,mml_0.ipk,mml_0.nama,mml_0.npm,mml_0.semester from mahasiswa mml_0
Mahasiswa(id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0')
```

## B.MahasiswaController

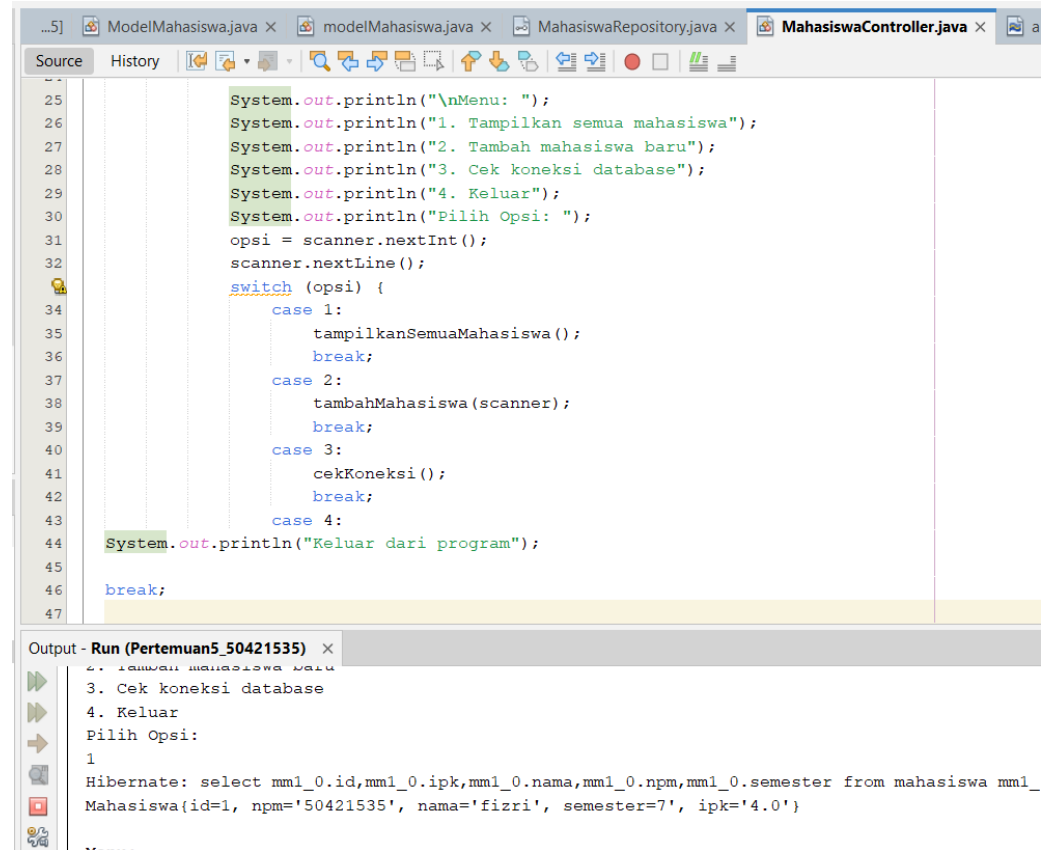


The screenshot shows the source code of the `MahasiswaController` class in the `me.fizri.controller` package. It imports `ModelMahasiswa`, `MahasiswaRepository`, `Autowired`, `Controller`, `List`, and `Scanner`. The class is annotated with `@Controller` and contains an `@Autowired` `MahasiswaRepository` instance. The `tampilkanMenu()` method initializes a `Scanner` and defines a menu with four options: 1. Tampilkan semua mahasiswa, 2. Tambah mahasiswa baru, 3. Cek koneksi database, and 4. Keluar. It prompts the user to 'Pilih Opsi:' and prints a Hibernate SQL query.

```
1 package me.fizri.controller;
2
3
4 import me.fizri.model.ModelMahasiswa;
5 import me.fizri.repository.MahasiswaRepository;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Controller;
8 import java.util.List;
9 import java.util.Scanner;
10
11 @Controller
12 public class MahasiswaController {
13
14
15     @Autowired
16     private MahasiswaRepository mahasiswaRepository;
17     public void tampilkanMenu() {
18
19         Scanner scanner = new Scanner(System.in);
20
21         int opsi;
22
23         do {
24
25             System.out.println("\nMenu: ");
26             System.out.println("1. Tampilkan semua mahasiswa");
27             System.out.println("2. Tambah mahasiswa baru");
28             System.out.println("3. Cek koneksi database");
29             System.out.println("4. Keluar");
30             System.out.println("Pilih Opsi: ");
31             opsi = scanner.nextInt();
32             scanner.nextLine();
33             switch (opsi) {
34                 case 1:
35                     tampilkanSemuaMahasiswa();
36                     break;
37                 case 2:
38                     tambahMahasiswa(scanner);
39                     break;
40                 case 3:
41                     cekKoneksi();
42                     break;
43                 case 4:
44                     System.out.println("Keluar dari program");
45                     break;
46             }
47         } while (opsi != 4);
48     }
49 }
```

Output - Run (Pertemuan5\_50421535) x

```
2024-11-12T14:03:32.000+07:00 INFO 4908 [main] O.S.D.W.Embedded:comcat:Tomcatw
2024-11-12T14:03:32.000+07:00 INFO 4908 --- [main] me.fizri.Pertemuan5_50421535
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
```



The screenshot shows the continuation of the `MahasiswaController` class. It implements the `tampilkanSemuaMahasiswa()` method by printing a menu and a list of students. The students listed are: 1. Tampilkan semua mahasiswa, 2. Tambah mahasiswa baru, 3. Cek koneksi database, 4. Keluar. It prompts the user to 'Pilih Opsi:' and prints a Hibernate SQL query.

```
25 System.out.println("\nMenu: ");
26 System.out.println("1. Tampilkan semua mahasiswa");
27 System.out.println("2. Tambah mahasiswa baru");
28 System.out.println("3. Cek koneksi database");
29 System.out.println("4. Keluar");
30 System.out.println("Pilih Opsi: ");
31 opsi = scanner.nextInt();
32 scanner.nextLine();
33 switch (opsi) {
34     case 1:
35         tampilkanSemuaMahasiswa();
36         break;
37     case 2:
38         tambahMahasiswa(scanner);
39         break;
40     case 3:
41         cekKoneksi();
42         break;
43     case 4:
44         System.out.println("Keluar dari program");
45         break;
46 }
47 }
```

Output - Run (Pertemuan5\_50421535) x

```
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_
Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...[5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x applic
Source History
48 default:
49 System.out.println("Opsi tidak valid, coba ");
50 }
51 } while (opsi != 4);
52 }
53
54 private void tampilkanSemuaMahasiswa () {
55
56 List<ModelMahasiswa> mahasiswaList = mahasiswaRepository.findAll();
57
58 if (mahasiswaList.isEmpty()) {
59
60 System.out.println("Tidak ada data mahasiswa.");
61
62 } else {
63
64 mahasiswaList.forEach (mahasiswa -> System.out.println (mahasiswa));
65
66 }
67
68 }
69
70
Output - Run (Pertemuan5_50421535) x
2. tampilkan mahasiswa data
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='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...[5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x application.properties x
Source History
69
70 private void tambahMahasiswa (Scanner scanner) {
71
72 System.out.println("Masukkan NPM: ");
73 String npm = scanner.nextLine();
74 System.out.println("Masukkan Nama: ");
75 String nama = scanner.nextLine();
76 System.out.println("Masukkan semester: ");
77 int semester = scanner.nextInt();
78 System.out.println("Masukkan ipk: ");
79 float ipk = scanner.nextFloat();
80 ModelMahasiswa mahasiswa = new ModelMahasiswa (0, npm, nama, semester, ipk);
81 mahasiswaRepository.save(mahasiswa);
82 System.out.println("Mahasiswa Berhasil ditambahkan.");
83 }
84
85 private void cekKoneksi () {
86
87 try {
88
89 mahasiswaRepository.findAll();
90
91 System.out.println("Koneksi ke database berhasil");
92
93 } catch (Exception e) {
94 System.out.println("Gagal koneksi ke database");
95 }
96
97 }
98
99
Output - Run (Pertemuan5_50421535) x
2. tampilkan mahasiswa data
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='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```

89     mahasiswaRepository.findAll();
90
91     System.out.println("Koneksi ke database berhasil");
92
93 } catch (Exception e) {
94
95     System.out.println("Gagal terhubung ke database.");
96
97 }
98
99 }
100 }

```

Output - Run (Pertemuan5\_50421535) X

```

2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}

```

## C.ModelMahasiswa

```

...5] ModelMahasiswa.java X modelMahasiswa.java X MahasiswaRepository.java X MahasiswaController.java X app
Source History
1 package me.fizri.model;
2
3 import jakarta.persistence.Column;
4 import jakarta.persistence.Entity;
5 import jakarta.persistence.GeneratedValue;
6 import jakarta.persistence.GenerationType;
7 import jakarta.persistence.Id;
8 import jakarta.persistence.Table;
9
10 @Entity
11 @Table(name = "mahasiswa")
12 public class ModelMahasiswa {
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     @Column(name = "id")
16     private int id;
17
18     @Column(name = "npm", nullable = false, length = 8)
19     private String npm;
20
21     @Column(name = "nama", nullable = false, length = 50)
22     private String nama;
23

```

Output - Run (Pertemuan5\_50421535) X

```

2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}

```

```
...[5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x appli
Source History
21 @Column(name = "nama", nullable = false, length = 50)
22 private String nama;
23
24 @Column(name = "semester")
25 private int semester;
26
27 @Column(name = "ipk")
28 private float ipk;
29
30 public ModelMahasiswa() {
31
32 }
33
34 public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
35     this.id = id;
36     this.npm = npm;
37     this.nama = nama;
38     this.semester = semester;
39     this.ipk = ipk;
40 }
41
42 public int getId() {
43     return id;
44 }
```

Output - Run (Pertemuan5\_50421535) x

```
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='50421535', nama='fizri', semester=7, ipk='4.0'}
```

```
...[5] ModelMahasiswa.java x modelMahasiswa.java x MahasiswaRepository.java x MahasiswaController.java x appli
Source History
41
42 public int getId() {
43     return id;
44 }
45
46 public void setId(int id) {
47     this.id = id;
48 }
49
50 public String getNpm() {
51     return npm;
52 }
53
54 public void setNpm(String npm) {
55     this.npm = npm;
56 }
57
58 public String getNama() {
59     return nama;
60 }
61
62 public void setNama(String nama) {
63     this.nama = nama;
64 }
```

Output - Run (Pertemuan5\_50421535) x

```
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='50421535', nama='fizri', semester=7, ipk='4.0'}
```



## D.MahasiswaRepository.java

The screenshot shows an IDE window with the source code of `MahasiswaRepository.java`. The code defines a package `me.fizri.repository`, imports `me.fizri.model.ModelMahasiswa`, `org.springframework.data.jpa.repository.JpaRepository`, and `org.springframework.stereotype.Repository`. It then defines a `@Repository` annotated interface `MahasiswaRepository` that extends `JpaRepository<ModelMahasiswa, Long>`.

Below the source code, the 'Output' window shows the results of running the application. It displays a menu with options 1 through 4, where option 1 is selected. The output then shows a Hibernate SQL query: `select mm1_0.id,mm1_0.ipk,mm1_0.nama,mm1_0.npm,mm1_0.semester from mahasiswa mm1_0` and a corresponding Java object: `Mahasiswa{id=1, npm='50421535', nama='fizri', semester=7, ipk='4.0'}`.

```
1 package me.fizri.repository;
2 import me.fizri.model.ModelMahasiswa;
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7
8 public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa, Long> {
9
10 }
11
12
```

Output - Run (Pertemuan5\_50421535) x

2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}

## E.pom.xml

The screenshot shows an IDE window with the `pom.xml` file. The XML content defines the project's dependencies, including `spring-boot-starter-parent` (version 3.3.3), `spring-boot-starter-data-jpa`, `mysql` (version 8.0.33), and `spring-boot-starter-web`.

Below the XML, the 'Output' window shows the same results as the previous screenshot, indicating that the application was run successfully and the database query was executed.

```
15
16 <groupId>org.springframework.boot</groupId>
17 <artifactId>spring-boot-starter-parent</artifactId>
18 <version>3.3.3</version>
19 <relativePath/>
20 </parent>
21 <dependencies>
22
23 <dependency>
24 <groupId>org.springframework.boot</groupId>
25 <artifactId>spring-boot-starter-data-jpa</artifactId>
26 </dependency>
27
28 <dependency>
29 <groupId>mysql</groupId>
30 <artifactId>mysql-connector-java</artifactId>
31 <version>8.0.33</version>
32 </dependency>
33
34 <dependency>
35 <groupId>org.springframework.boot</groupId>
36 <artifactId>spring-boot-starter-web</artifactId>
37 </dependency>

```

Output - Run (Pertemuan5\_50421535) x

2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}



The screenshot shows an IDE with the following tabs: `...va`, `pom.xml [Pertemuan5_50421535]`, `ModelMahasiswa.java`, `modelMahasiswa.java`, `MahasiswaRepository.java`, and `MahasiswaController.java`. The `pom.xml` file is open in the Source view, showing the following XML structure:

```
33
34 <dependency>
35 <groupId>org.springframework.boot</groupId>
36 <artifactId>spring-boot-starter-web</artifactId>
37 </dependency>
38
39
40 <dependency>
41 <groupId>org.springframework.boot</groupId>
42 <artifactId>spring-boot-starter-test</artifactId>
43 <scope>test</scope>
44 </dependency>
45 </dependencies>
46 <build>
47
48 <plugins>
49 <plugin>
50 <groupId>org.springframework.boot</groupId>
51 <artifactId>spring-boot-maven-plugin</artifactId>
52 </plugin>
53 </plugins>
54 </build>
55
```

The Output view shows the following text:

```
Output - Run (Pertemuan5_50421535)
2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}
```

## F.application.properties

The screenshot shows an IDE with the following tabs: `...5]`, `ModelMahasiswa.java`, `modelMahasiswa.java`, `MahasiswaRepository.java`, `MahasiswaController.java`, and `application.properties`. The `application.properties` file is open in the Source view, showing the following properties:

```
1 spring.datasource.url=jdbc:mysql://localhost:3306/spring_50421535?useSSL=false&serverTimezone=UTC
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5
6 spring.jpa.hibernate.ddl-auto=update
7 spring.jpa.show-sql=true
```

The Output view shows the following text:

```
Output - Run (Pertemuan5_50421535)
2. Tampilkan 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='50421535', nama='fizri', semester=7, ipk='4.0'}
```

## G.Output

```
Output - Run (Pertemuan5_50421535) x
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

Menu:
1. Tampilkan semua mahasiswa
2. Tambah mahasiswa baru
3. Cek koneksi database
4. Keluar
Pilih Opsi:
2
Masukkan NPM:
50421535
Masukkan Nama:
fizri
Masukkan semester:
7
Masukkan ipk:
4
Hibernate: insert into mahasiswa (ipk,nama,npm,semester) values (?, ?, ?, ?)
Mahasiswa Berhasil ditambahkan.

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='50421535', nama='fizri', semester=7, ipk='4.0'}
```