

# PUPUT DEBBYLA WALZAHRA

51421182\_4IA06

Rekayasa Perangkat Lunak 2

Konsep Framework Spring, Pembuatan Project Spring Dan Hibernate

---

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

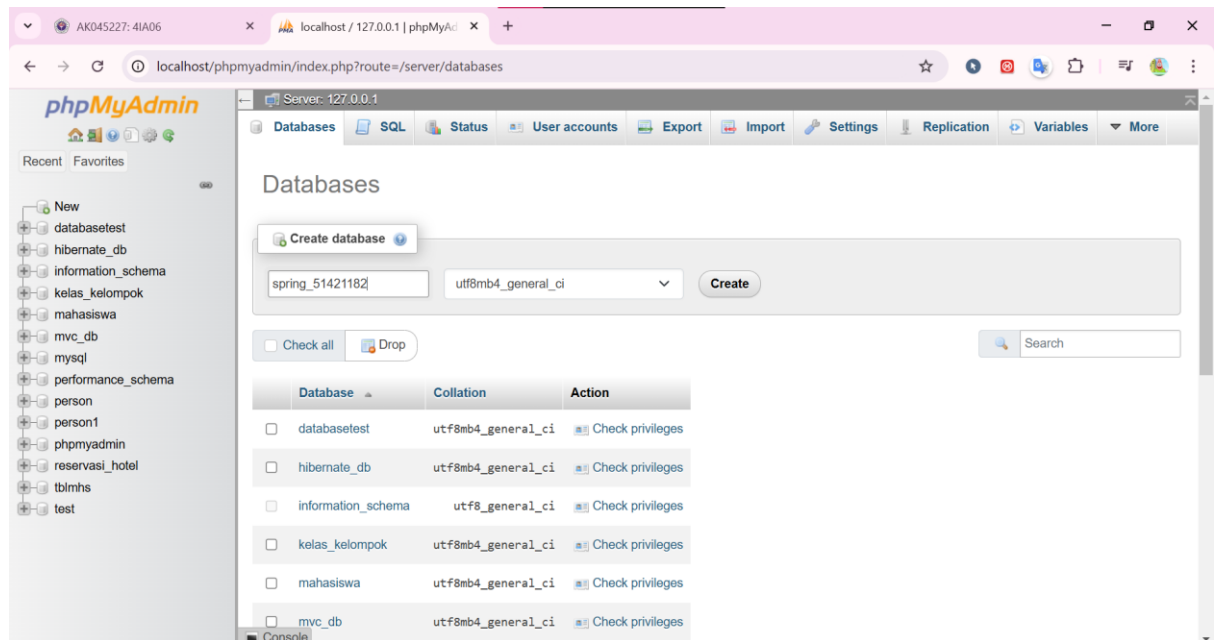
Jawab:

- Dependency Injection (DI) adalah sebuah teknik dalam pemrograman yang bertujuan untuk memisahkan pembuatan objek (instansiasi) dari penggunaan objek tersebut. Dalam konteks yang lebih sederhana, DI adalah cara untuk memberikan objek yang dibutuhkan oleh suatu kelas (atau komponen) tanpa harus membuat objek tersebut secara langsung di dalam kelas tersebut.
- DI memainkan peran yang sangat penting dalam pengembangan aplikasi berbasis Spring Boot karena beberapa alasan berikut:
  1. Meningkatkan Keterbacaan dan Pemeliharaan Kode:
    - Kode yang lebih bersih: Dengan DI, kode menjadi lebih mudah dibaca karena logika inti dari kelas tidak tercampur dengan detail implementasi pembuatan objek.
    - Pengujian yang lebih mudah: Kelas yang menggunakan DI dapat diuji dengan lebih mudah karena dependensi dapat diganti dengan mock object untuk tujuan pengujian.
  2. Meningkatkan Kekompakan (Loose Coupling):
    - Komponen yang independen: Komponen-komponen dalam aplikasi menjadi lebih independen satu sama lain. Perubahan pada satu komponen tidak akan berdampak langsung pada komponen lainnya.
    - Kemudahan dalam penggantian komponen: Komponen yang sudah ada dapat dengan mudah diganti dengan komponen lain yang memiliki interface yang sama.
  3. Mendukung Prinsip SOLID:
    - Dependency Inversion Principle: DI mendukung prinsip ini dengan membalikkan ketergantungan sehingga kelas-kelas tingkat tinggi tidak bergantung pada kelas-kelas tingkat rendah.
    - Open-Closed Principle: Kelas-kelas menjadi lebih terbuka untuk ekstensi namun tertutup untuk modifikasi.
  4. Memudahkan Manajemen Konfigurasi:
    - Konfigurasi terpusat: Konfigurasi dependensi dapat dilakukan secara terpusat, misalnya melalui file konfigurasi atau anotasi.
    - Fleksibilitas: Konfigurasi dapat dengan mudah diubah tanpa mengubah kode sumber.
  5. Mendukung Pemrograman Berorientasi Aspek:
    - AOP (Aspect Oriented Programming): DI dapat diintegrasikan dengan AOP untuk menambahkan fungsionalitas tambahan ke objek, seperti logging, transaksi, atau keamanan.

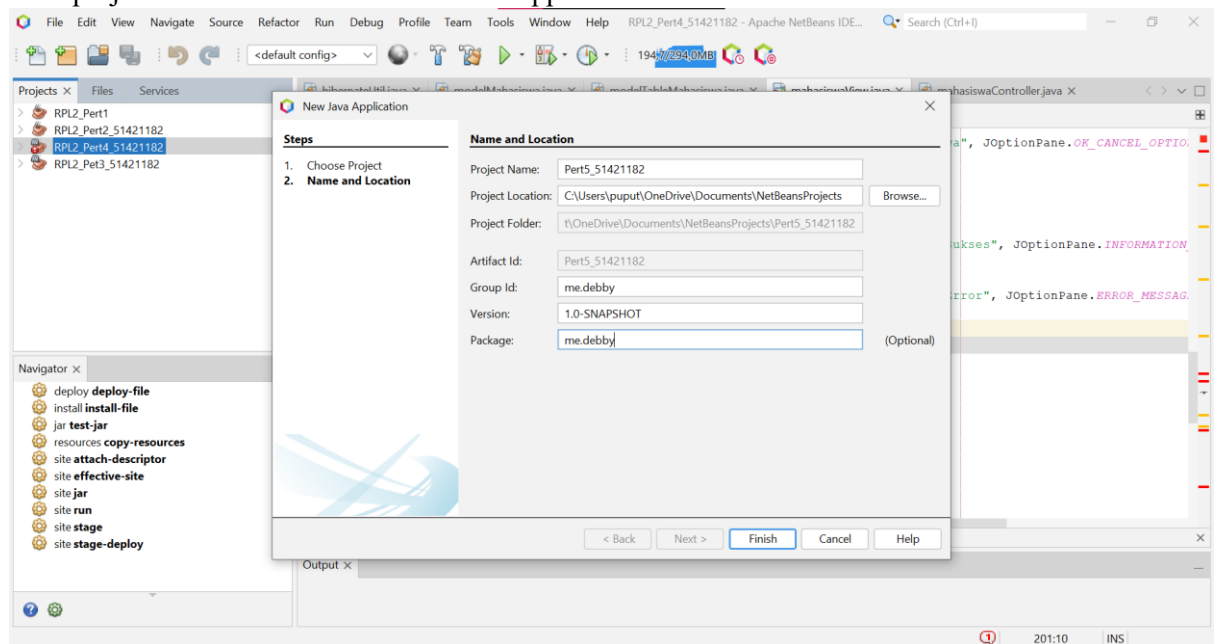
2) Screenshot kode dan output program!

Jawab:

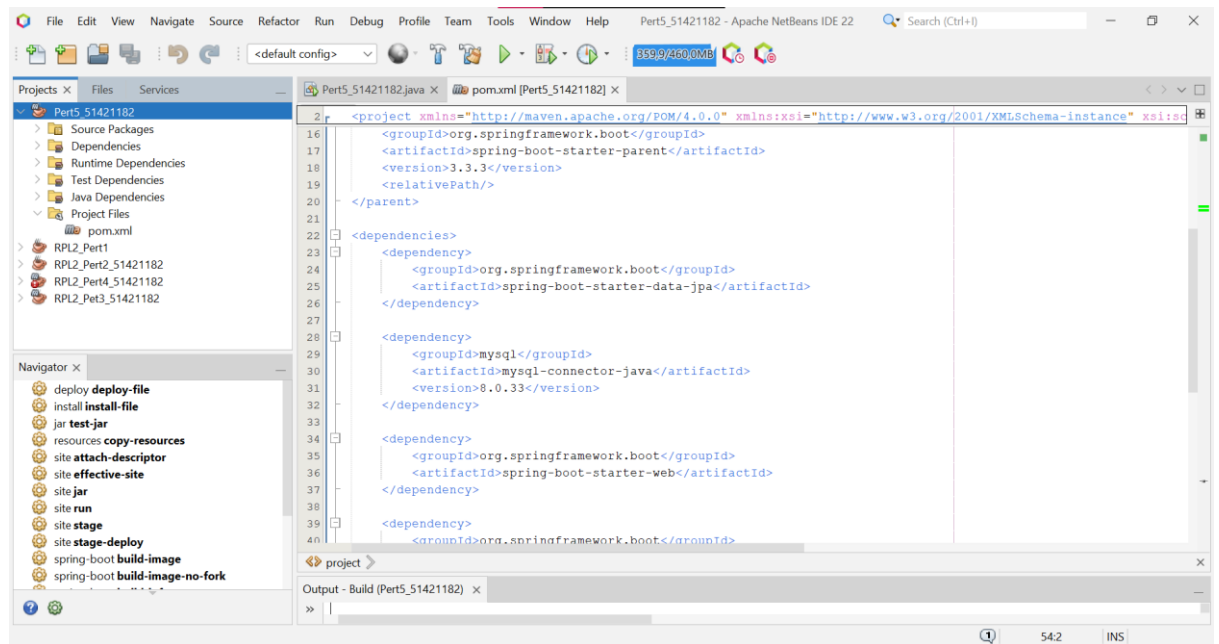
1. Buat database di phpmyadmin



## 2. Buat project baru Java with Maven > Java Application



## 3. XML pada pom.xml



Code:

```
<?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-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>me.debby</groupId>
  <artifactId>Pert5_51421182</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>22</maven.compiler.source>
    <maven.compiler.target>22</maven.compiler.target>
    <exec.mainClass>me.debby.Pert5_51421182</exec.mainClass>
  </properties>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.3</version>
    <relativePath/>
  </parent>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

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

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
  </dependencies>
```

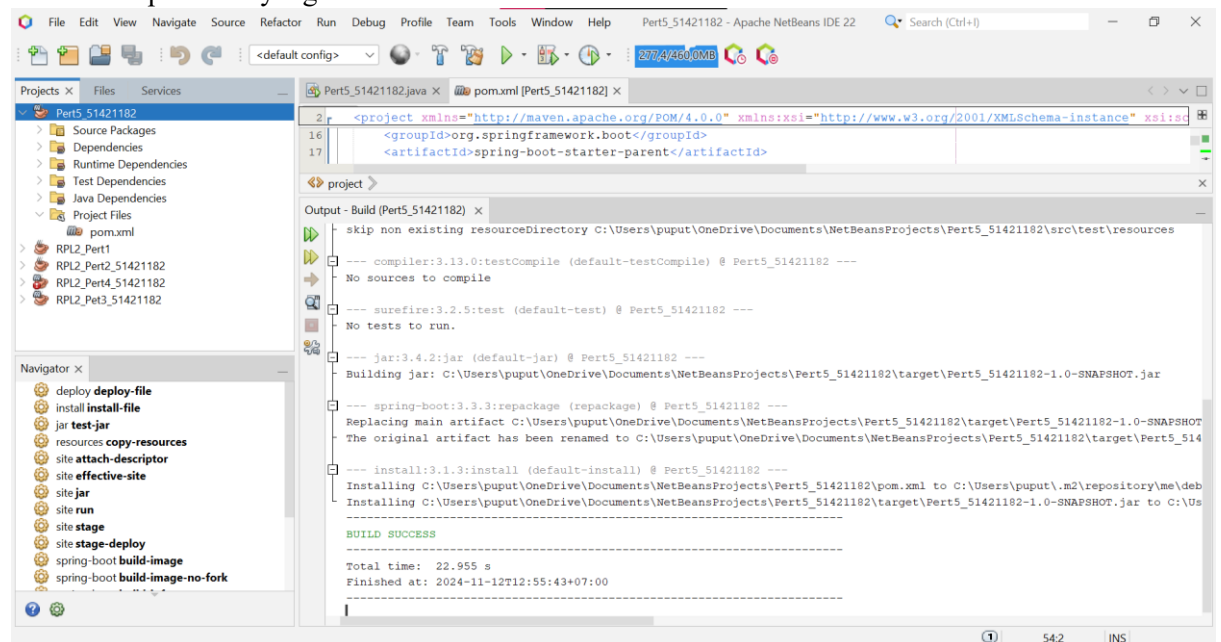
```

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

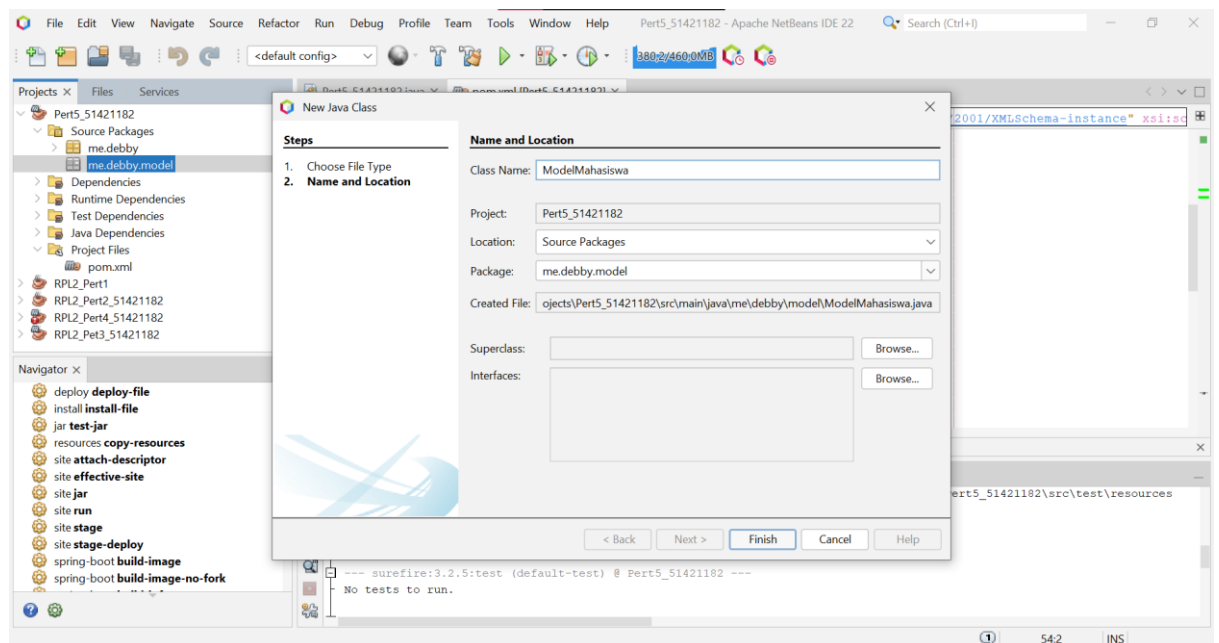
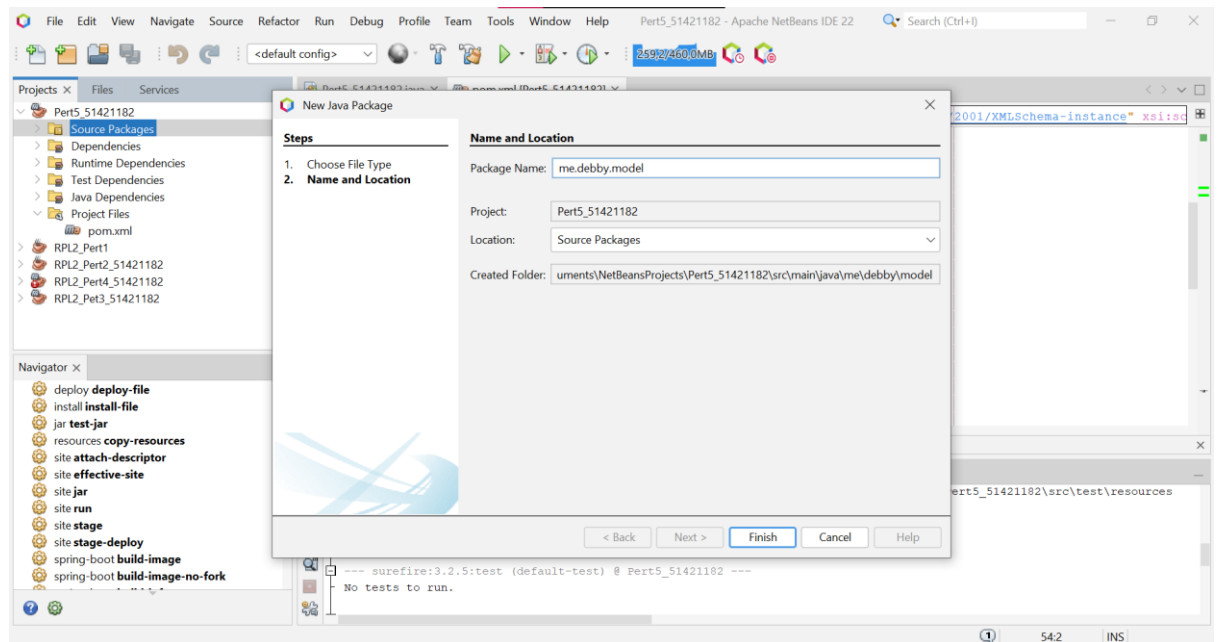
  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
</project>

```

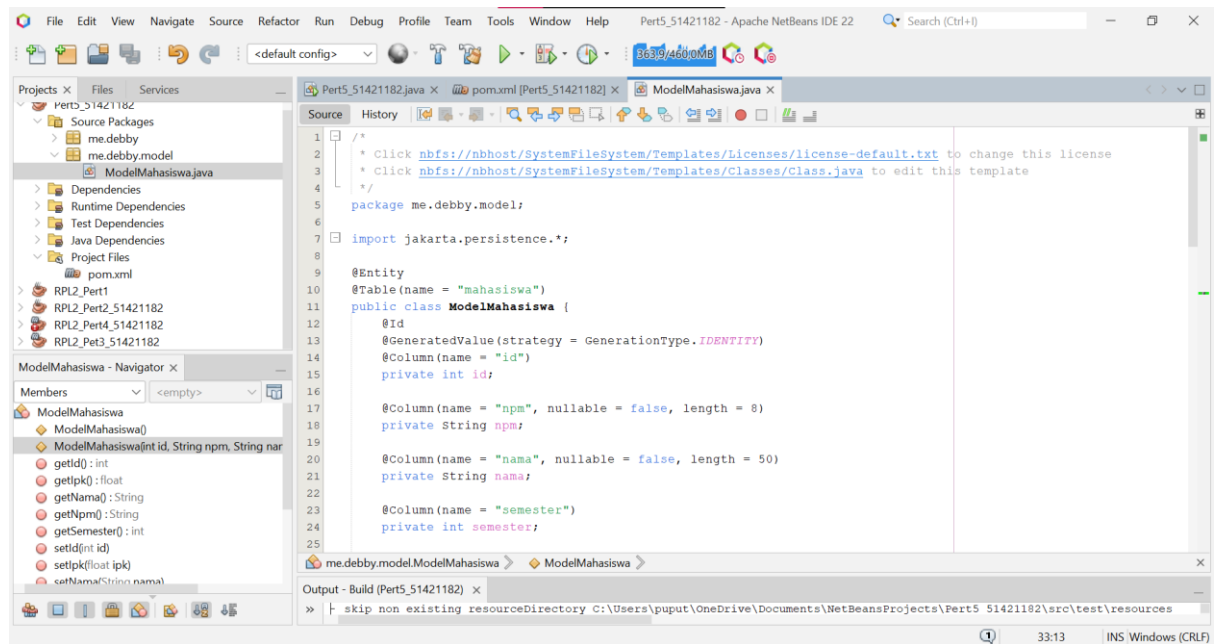
Lalu build pom.xml yang sudah dibuat



4. Buat Java Class didalam Java Package didalam Source Package



5. Masukkan code ModelMahasiswa.java



## Code:

```

/*
 * 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 me.debby.model;

import jakarta.persistence.*;

@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() {

    }

    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 getName() {
        return nama;
    }

    public void setName(String nama) {
        this.nama = nama;
    }

    public int getSemester() {
        return semester;
    }

    public void setSemester(int semester) {
        this.semester = semester;
    }

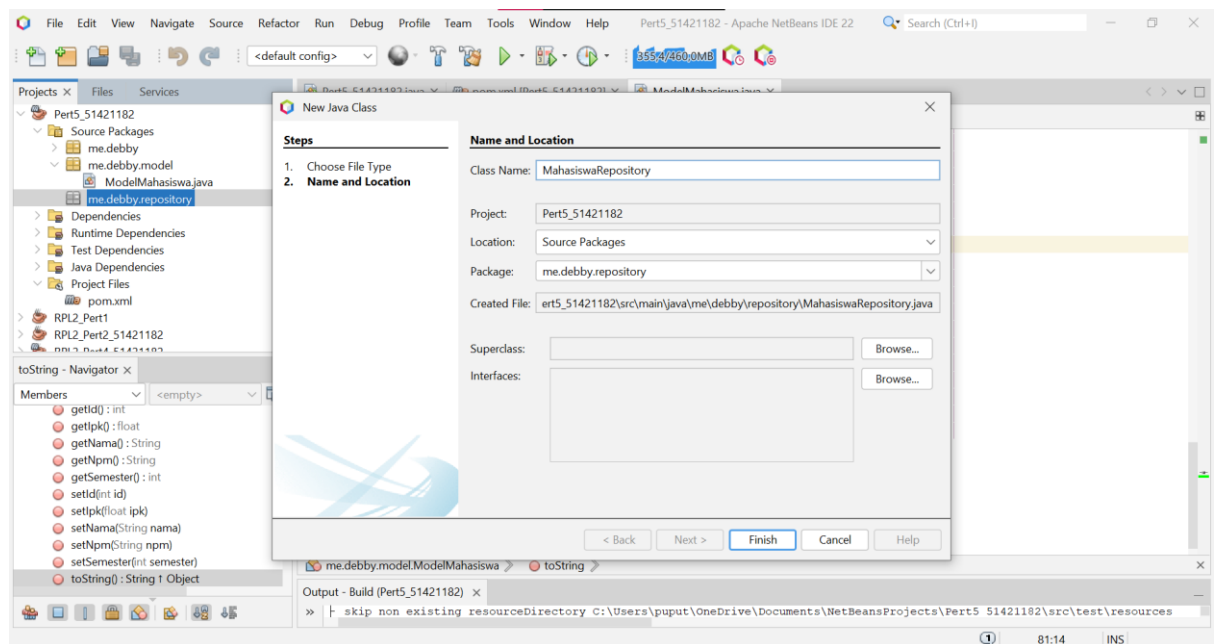
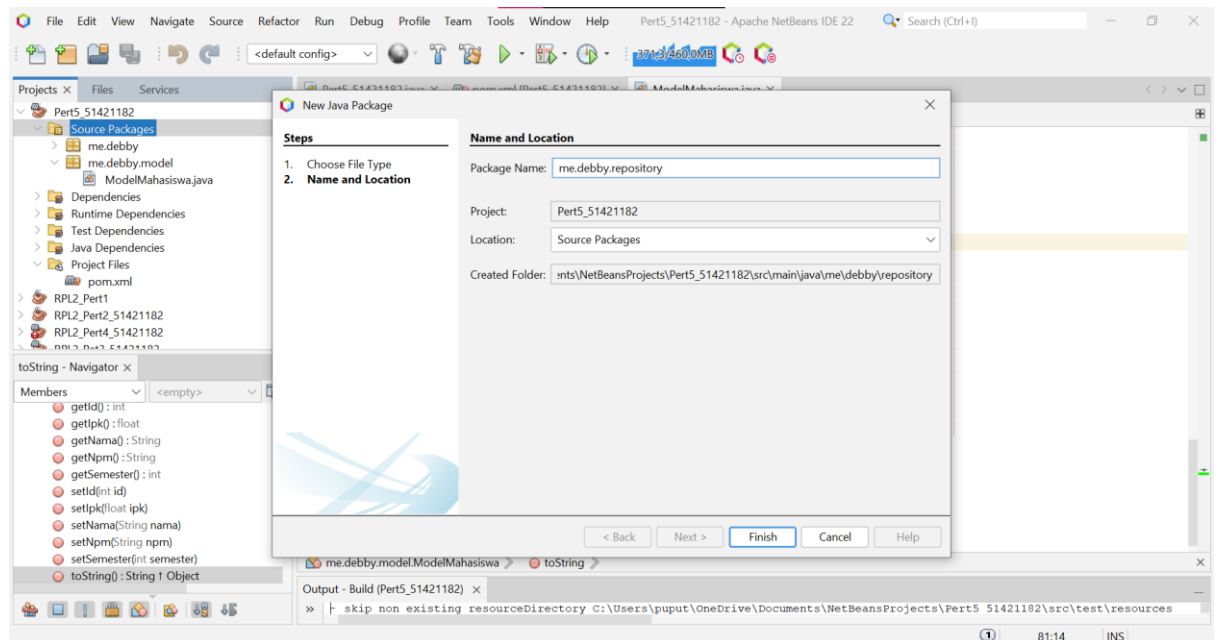
    public float getIpk() {
        return ipk;
    }

    public void setIpk(float ipk) {
        this.ipk = ipk;
    }

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

```

## 6. Buat Java Class didalam Java Package didalam Source Package



Code:

```

/*
 * 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 me.debby.repository;

import me.debby.model.ModelMahasiswa;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

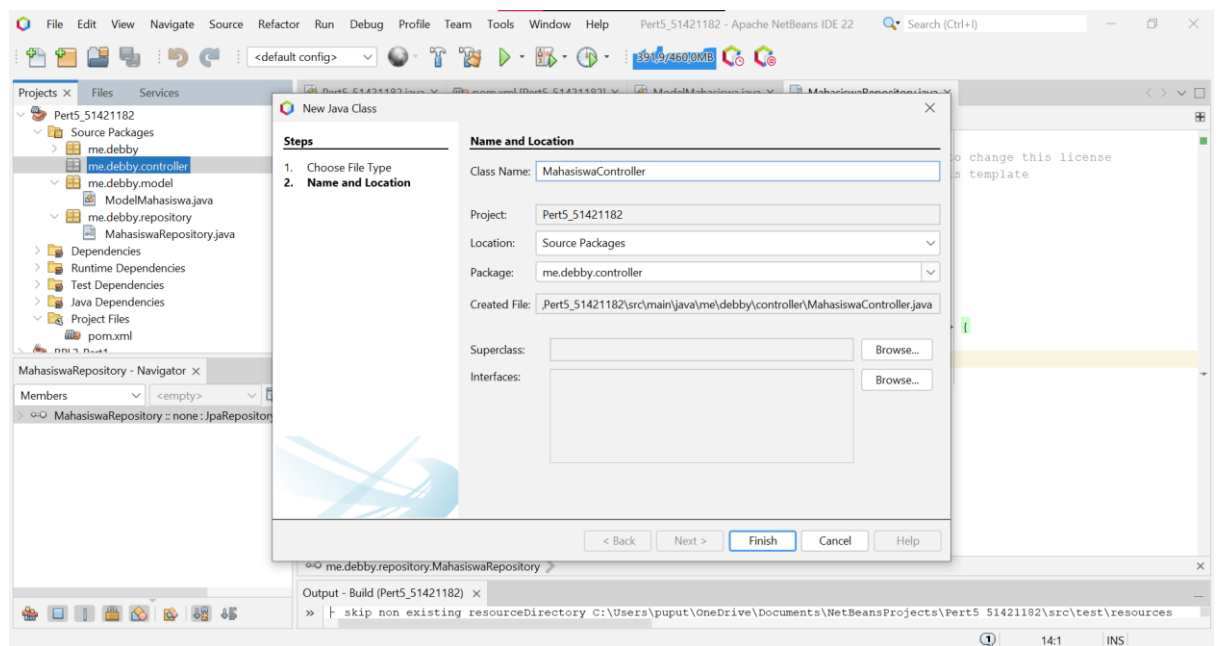
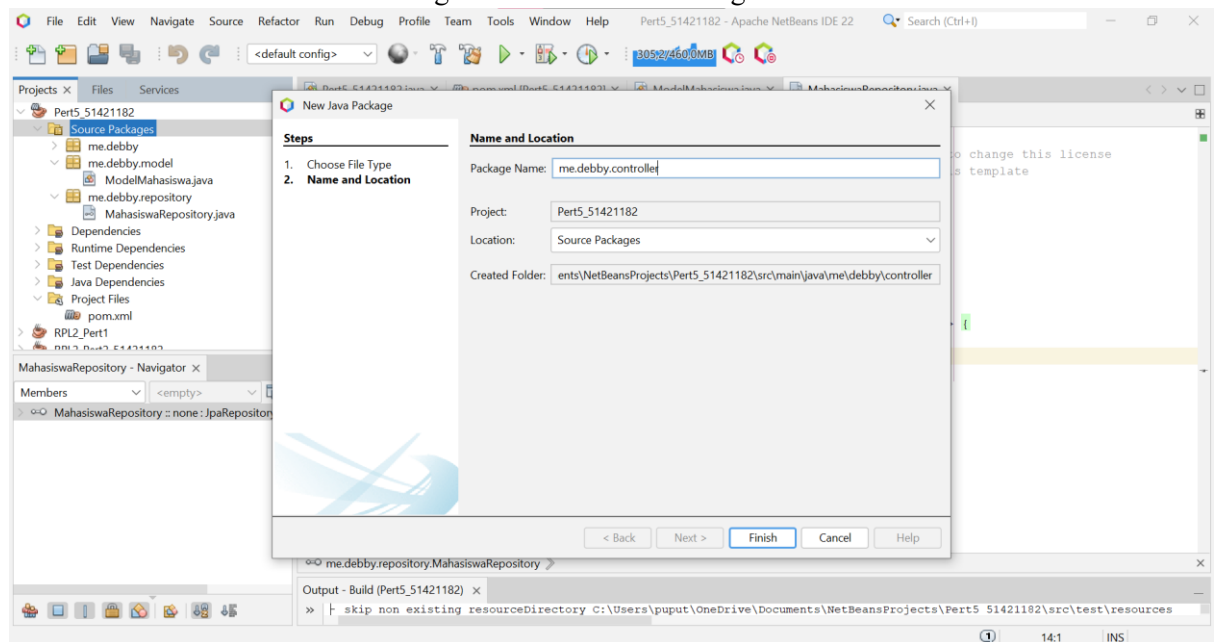
@Repository
public interface MahasiswaRepository extends JpaRepository<ModelMahasiswa,
Long> {

}

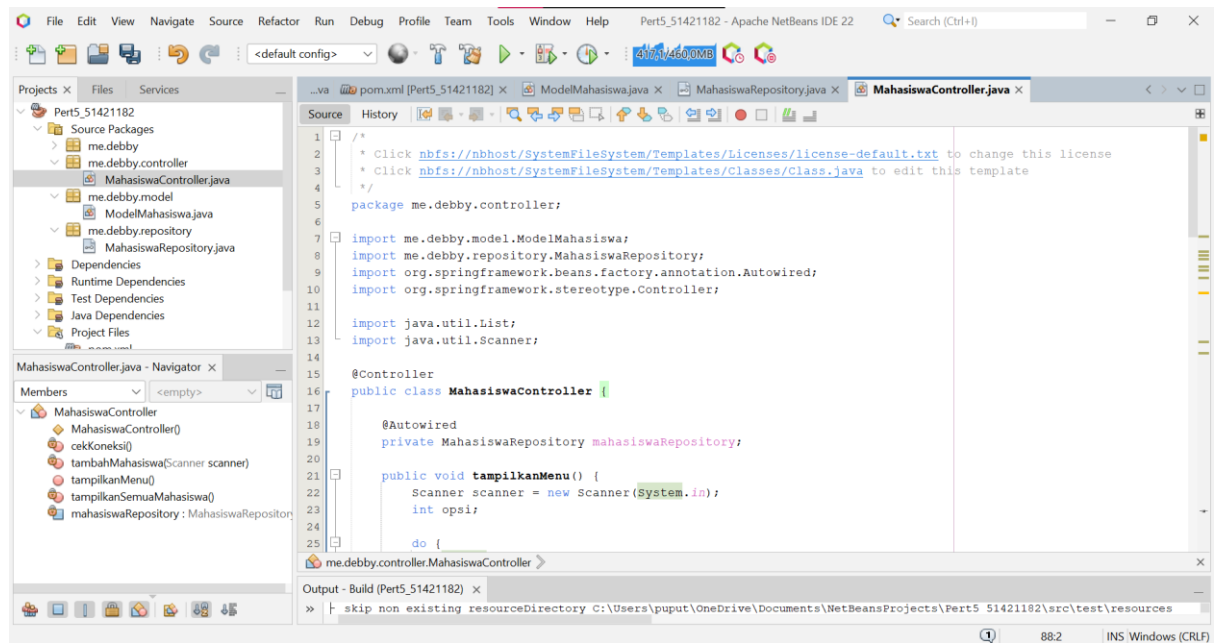
```



## 7. Buat Java Class didalam Java Package didalam Source Package



## 8. Masukkan code berikut pada MahasiswaController.java



## Code:

```

/*
 * 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 me.debby.controller;

import me.debby.model.ModelMahasiswa;
import me.debby.repository.MahasiswaRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;

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

@Controller
public class MahasiswaController {

    @Autowired
    private MahasiswaRepository mahasiswaRepository;

    public void tampilanMenu() {
        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();

            switch (opsi) {
                case 1:
                    tampilanSemuaMahasiswa();

```

```

        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));
    }
}

private void tambahMahasiswa(Scanner scanner) {
    System.out.print("Masukkan NPM: ");
    String npm = scanner.nextLine();
    System.out.print("Masukkan Nama: ");
    String nama = scanner.nextLine();
    System.out.print("Masukkan semester: ");
    int semester = scanner.nextInt();
    System.out.print("Masukkan IPK: ");
    float ipk = scanner.nextFloat();

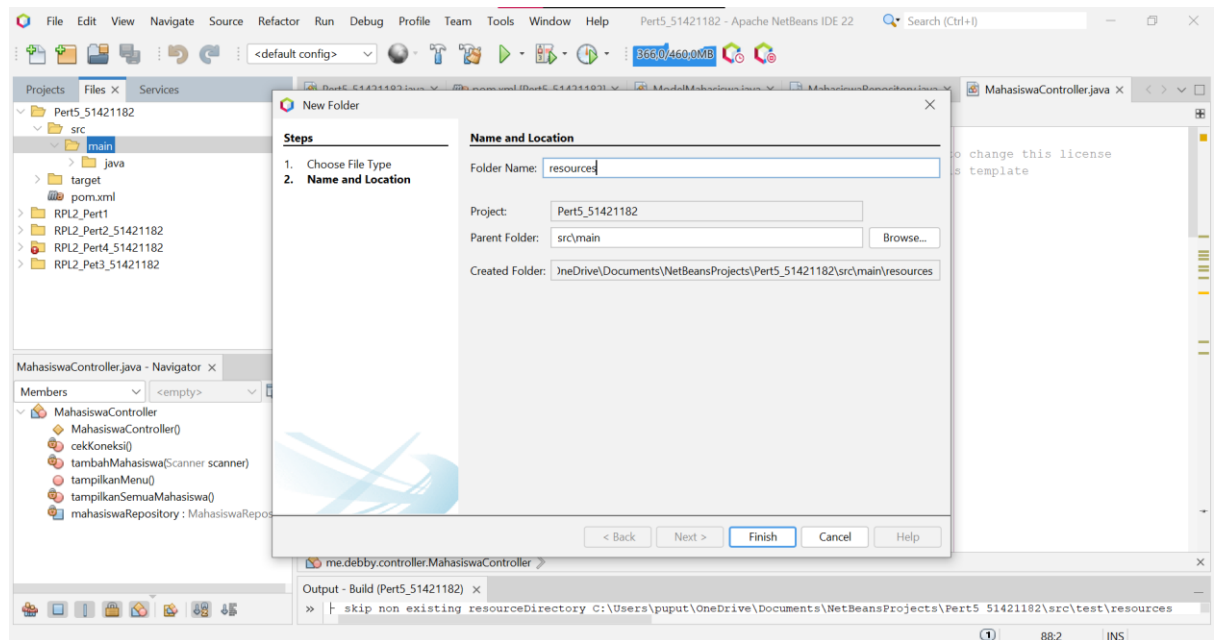
    ModelMahasiswa mahasiswa = new ModelMahasiswa(0, npm, nama,
semester, ipk);
    mahasiswaRepository.save(mahasiswa);

    System.out.println("Mahasiswa Berhasil ditambahkan.");
}

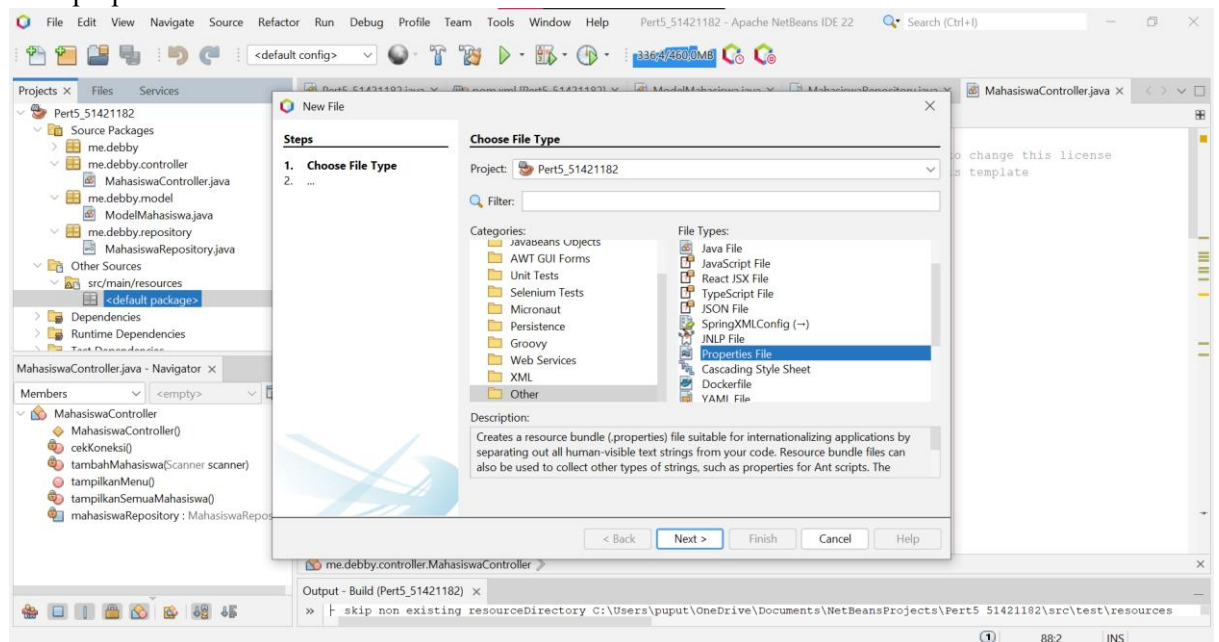
private void cekKoneksi() {
    try {
        mahasiswaRepository.findAll();
        System.out.println("Koneksi ke database berhasil");
    } catch (Exception e) {
        System.out.println("Gagal terhubung ke database");
    }
}
}

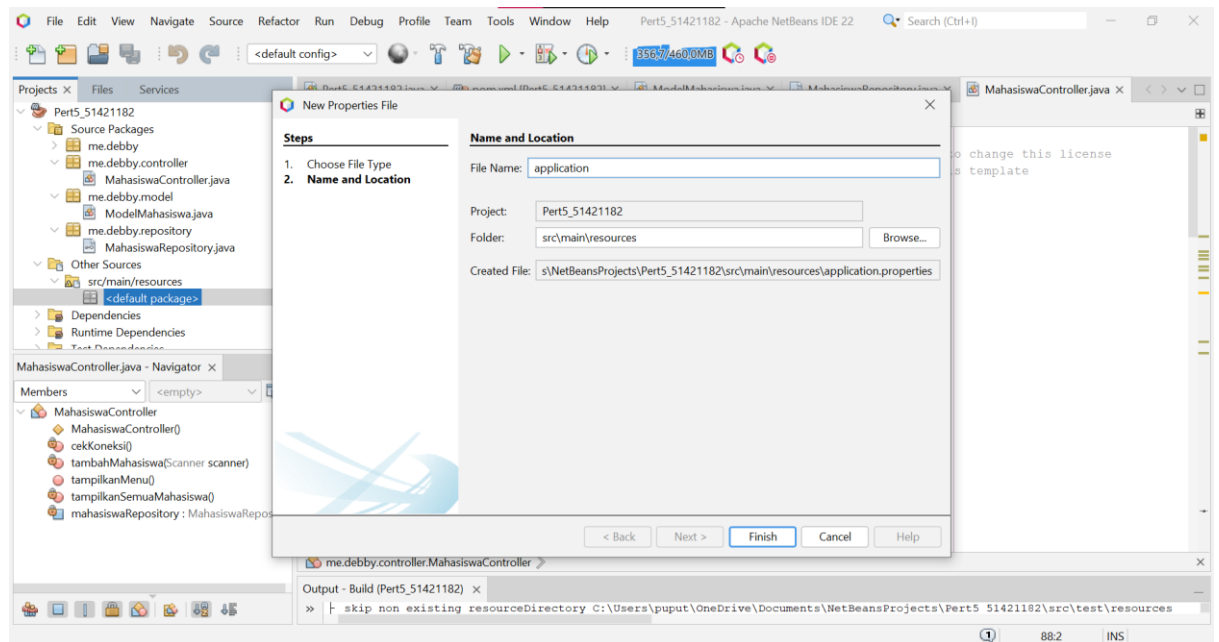
```

## 9. Buat folder pada project Pert4\_51421182

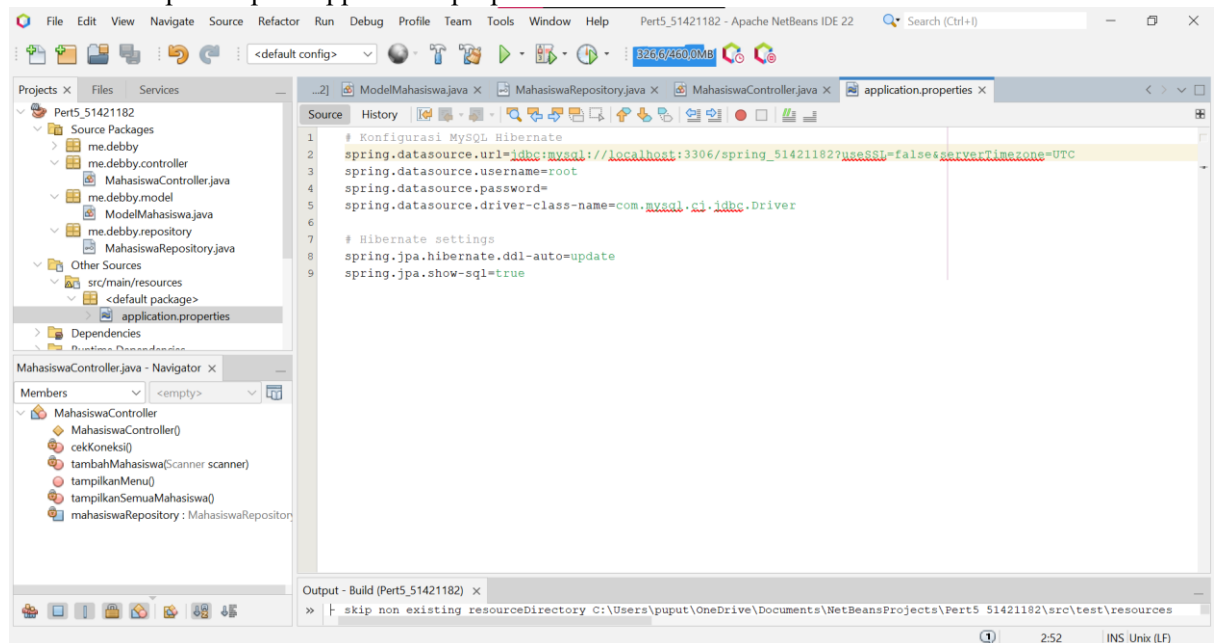


## 10. Buat properties file baru





## 11. Masukkan Properties pada application.properties

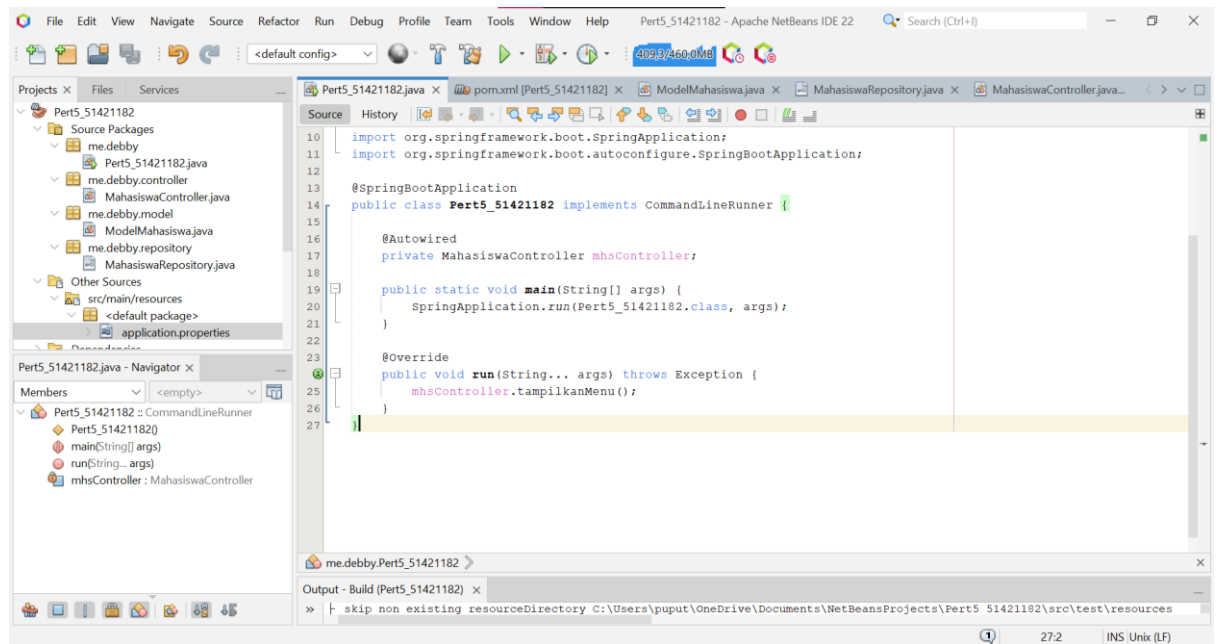


### Properties:

```
# Konfigurasi MySQL Hibernate
spring.datasource.url=jdbc:mysql://localhost:3306/spring_51421182?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
```

## 12. Masukkan code pada Pert5\_51421182.java



Code:

```

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

package me.debby;

import me.debby.controller.MahasiswaController;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Pert5_51421182 implements CommandLineRunner {

    @Autowired
    private MahasiswaController mhsController;

    public static void main(String[] args) {
        SpringApplication.run(Pert5_51421182.class, args);
    }

    @Override
    public void run(String... args) throws Exception {
        mhsController.tampilkanMenu();
    }

}

```

### 13. Output

