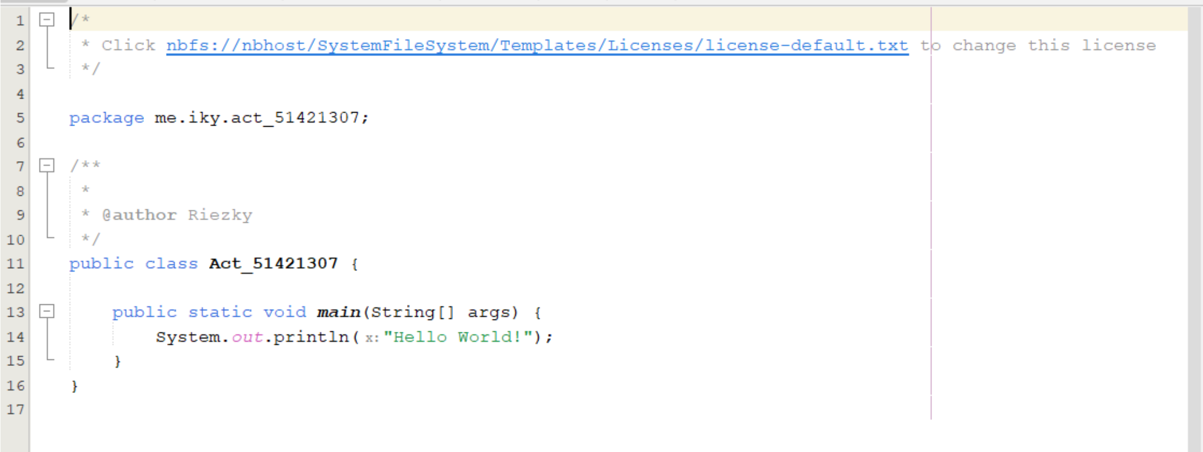
Nama : Riezky Arif Fadhilah

NPM : 51421307

Kelas : 4IA27

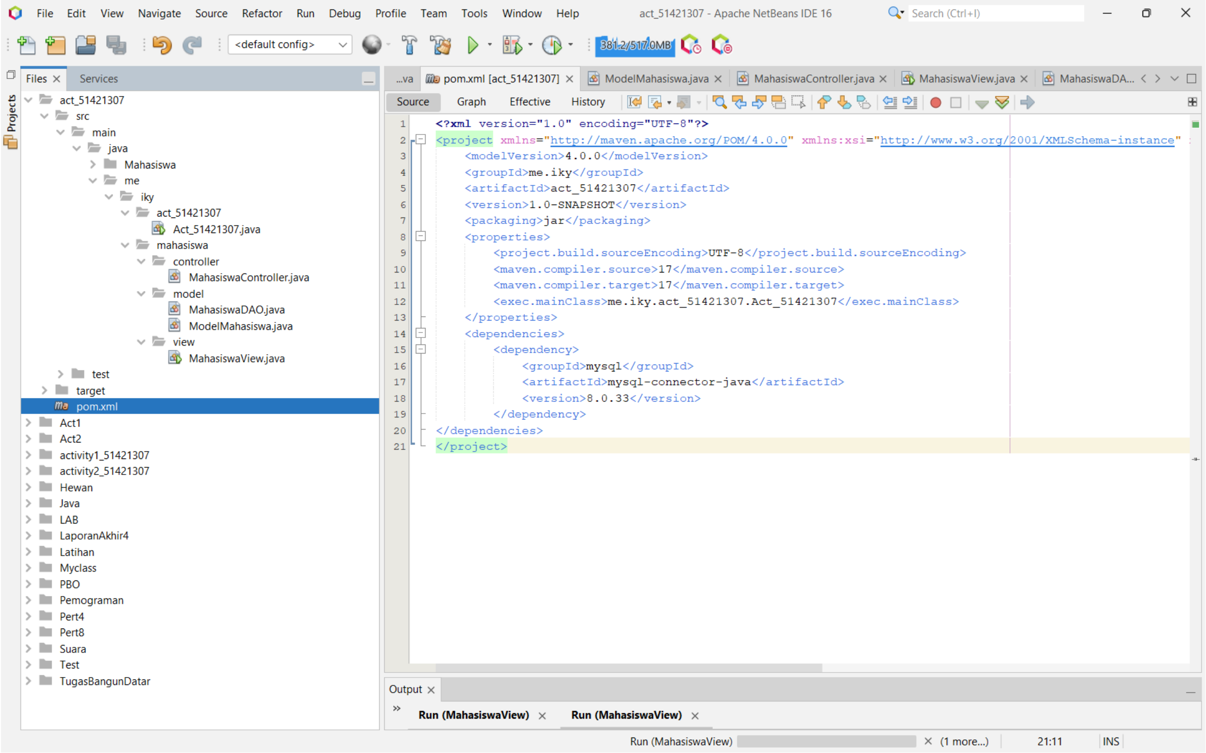
Activity 3

Act3_51421307



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

package me.iky.act_51421307;

/**
 *
 * @author Riezky
 */
public class Act_51421307 {

    public static void main(String[] args) {
        System.out.println( x: "Hello World!");
    }
}
```

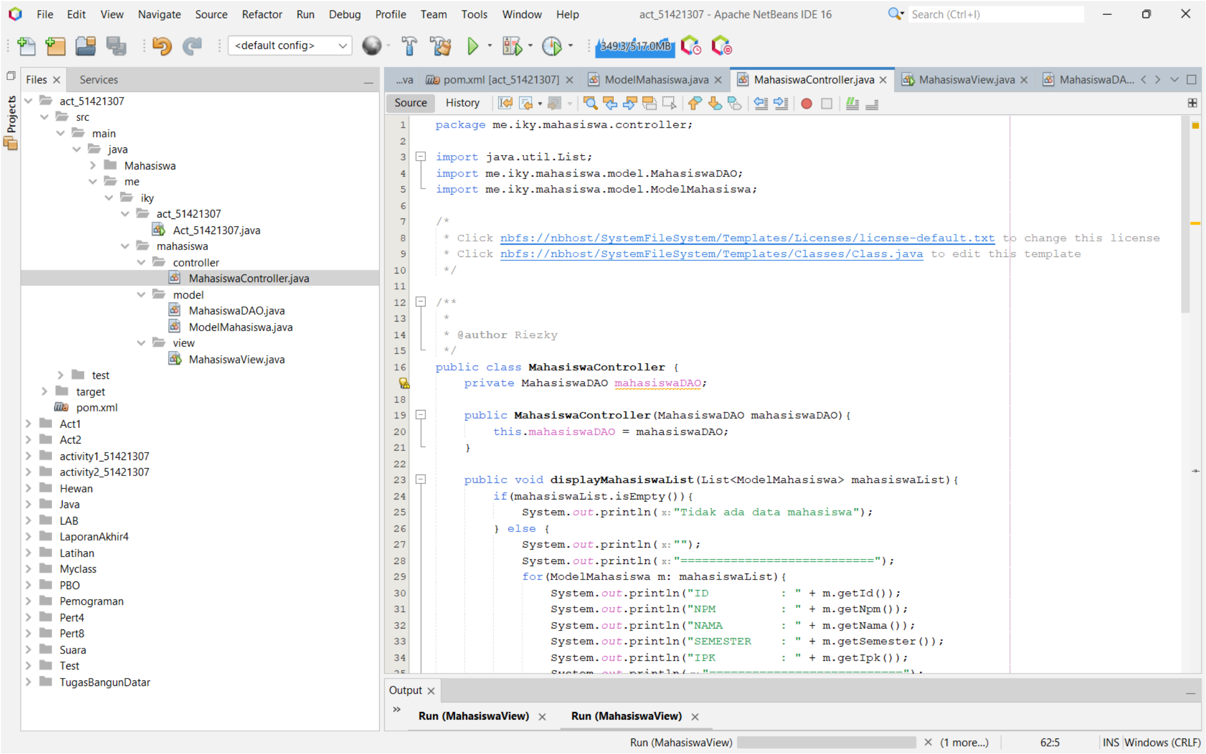
Pom.xml



```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-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>me.iky</groupId>
    <artifactId>act_51421307</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <exec.mainClass>me.iky.act_51421307.Act_51421307</exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
</dependencies>
</project>
```

Controller

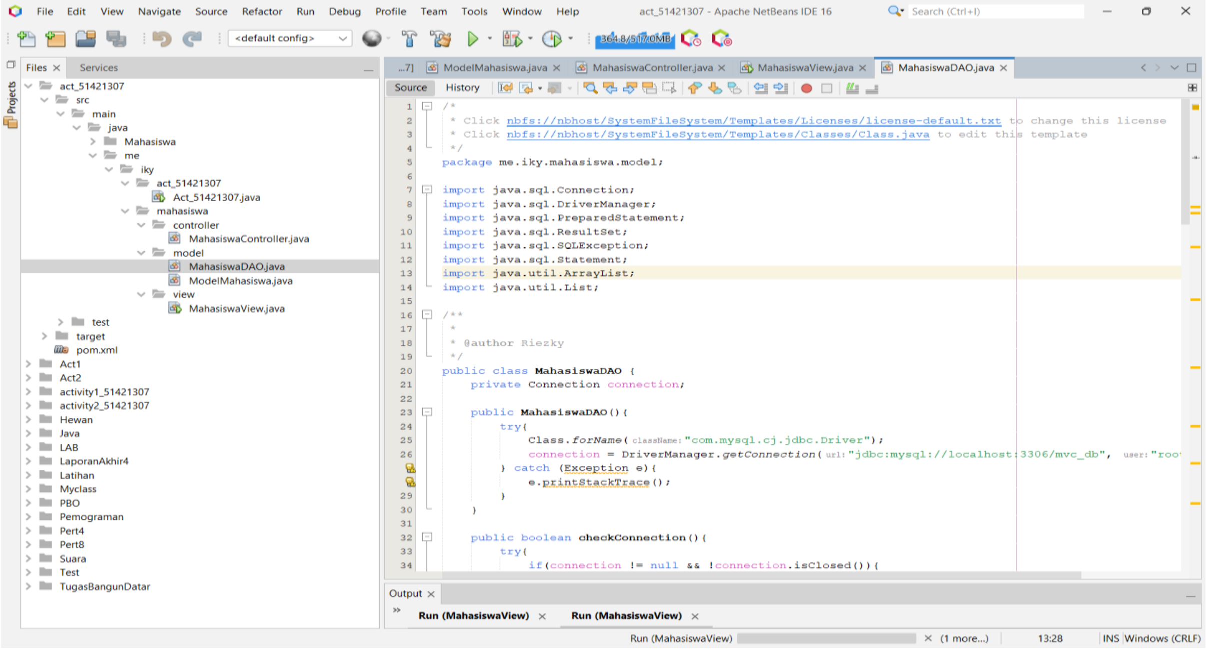
Mahasiswa.Controller



```java
package me.iky.mahasiswa.controller;

import java.util.List;
import me.iky.mahasiswa.model.MahasiswaDAO;
import me.iky.mahasiswa.model.ModelMahasiswa;

/*
 * 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
 */

/**
 *
 * @author Riezky
 */
public class MahasiswaController {
    private MahasiswaDAO mahasiswaDAO;

    public MahasiswaController(MahasiswaDAO mahasiswaDAO){
        this.mahasiswaDAO = mahasiswaDAO;
    }

    public void displayMahasiswaList(List<ModelMahasiswa> mahasiswaList){
        if(mahasiswaList.isEmpty()){
            System.out.println("Tidak ada data mahasiswa");
        } else {
            System.out.println("");
            System.out.println("==========================");
            for(ModelMahasiswa m: mahasiswaList){
                System.out.println("ID          : " + m.getId());
                System.out.println("NPM         : " + m.getNpm());
                System.out.println("NAMA        : " + m.getNama());
                System.out.println("SEMESTER    : " + m.getSemester());
                System.out.println("IPK         : " + m.getIpk());
```

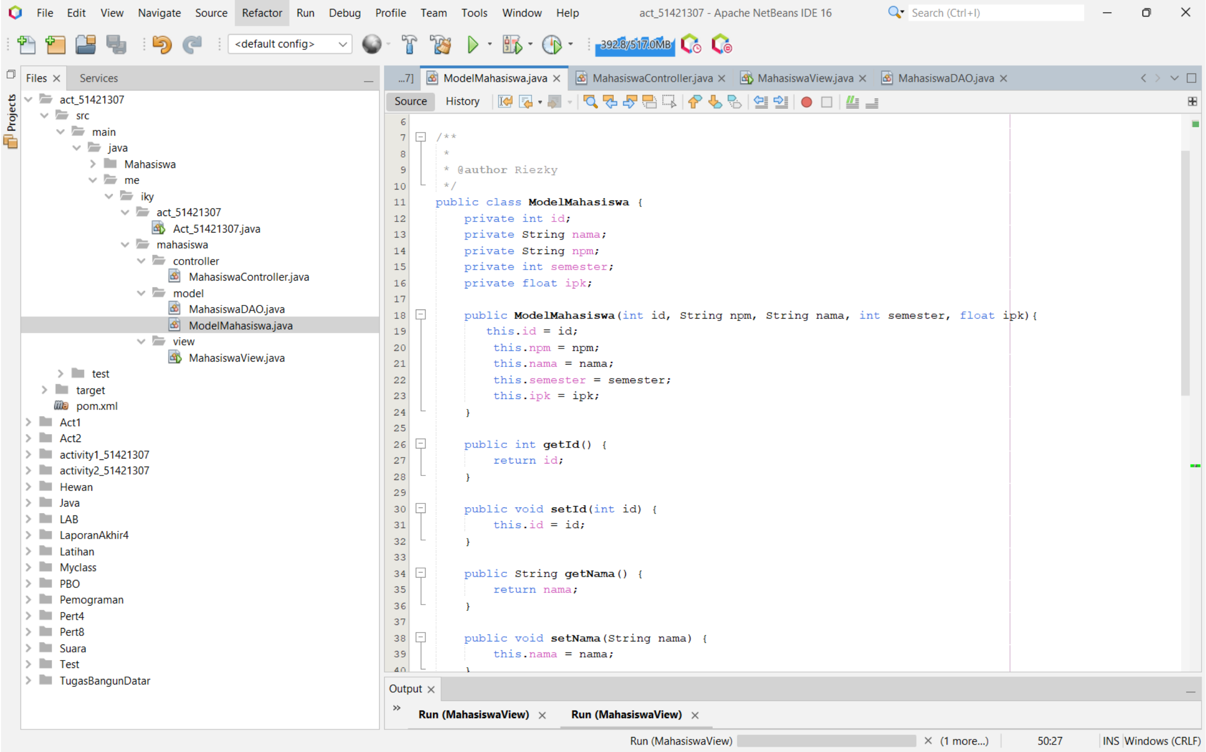
Model

MahasiswaDAO



```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 me.iky.mahasiswa.model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author Riezky
 */
public class MahasiswaDAO {
    private Connection connection;

    public MahasiswaDAO(){
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mvc_db", "roo
        } catch (Exception e){
            e.printStackTrace();
        }
    }

    public boolean checkConnection(){
        try{
            if(connection != null && !connection.isClosed()){
```

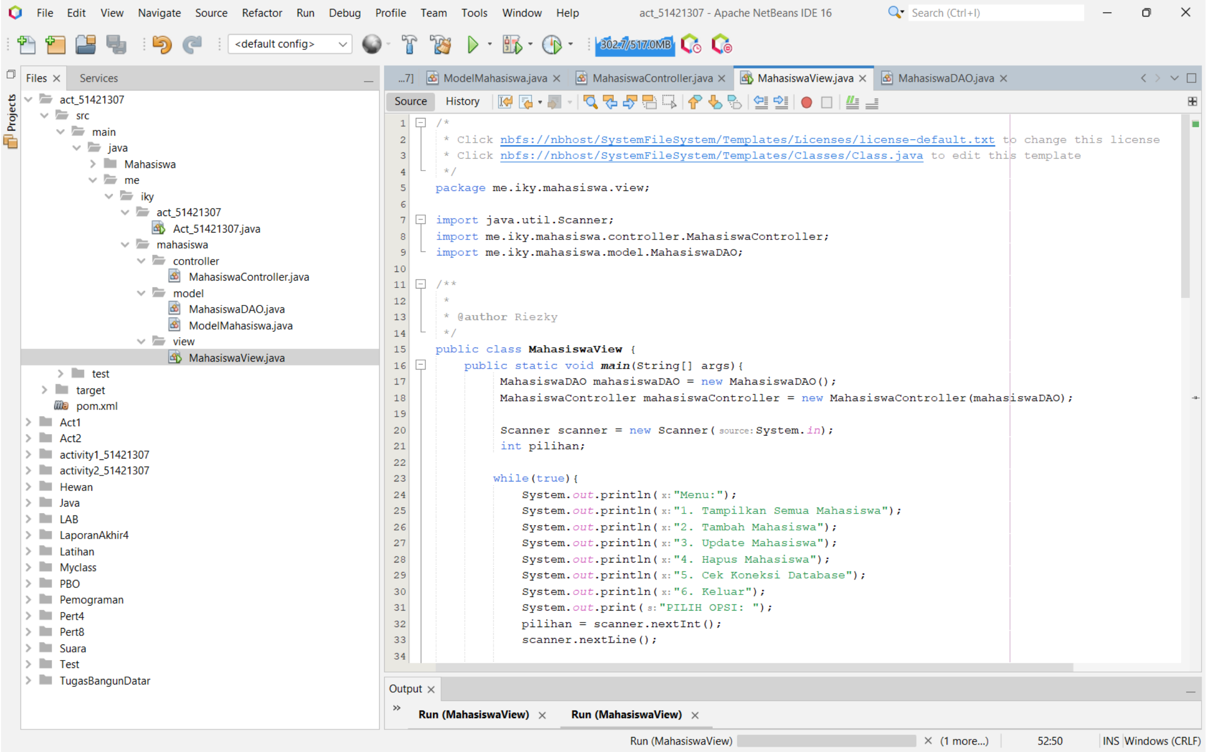
Model Mahasiswa



```
/**
 *
 * @author Riezky
 */
public class ModelMahasiswa {
    private int id;
    private String nama;
    private String npm;
    private int semester;
    private float ipk;

    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 getNama() {
        return nama;
    }

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

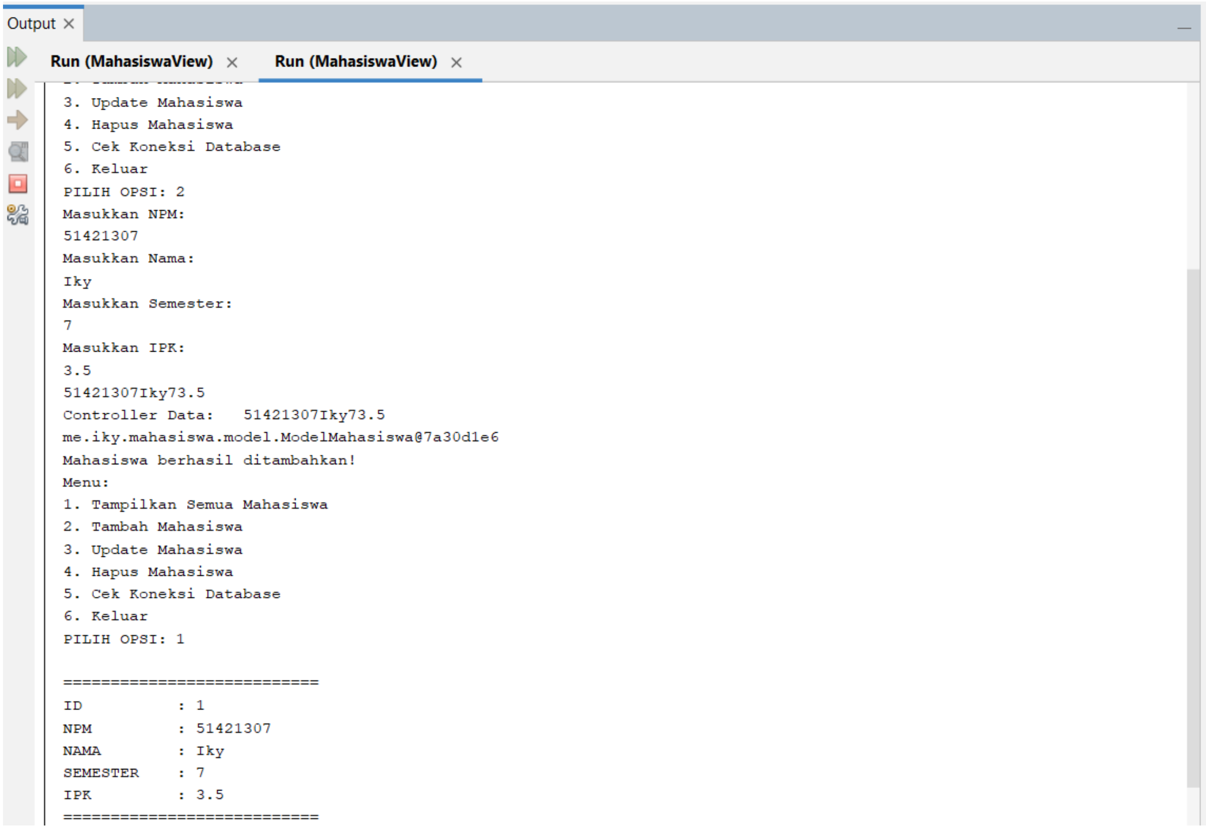
View

MahasiswaView



```
/*
 * 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.iky.mahasiswa.view;

import java.util.Scanner;
import me.iky.mahasiswa.controller.MahasiswaController;
import me.iky.mahasiswa.model.MahasiswaDAO;

/**
 *
 * @author Riezky
 */
public class MahasiswaView {
    public static void main(String[] args){
          MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
          MahasiswaController mahasiswaController = new MahasiswaController(mahasiswaDAO);

          Scanner scanner = new Scanner( source: System.in);
          int pilihan;

        while(true){
            System.out.println( x: "Menu:");
            System.out.println( x: "1. Tampilkan Semua Mahasiswa");
            System.out.println( x: "2. Tambah Mahasiswa");
            System.out.println( x: "3. Update Mahasiswa");
            System.out.println( x: "4. Hapus Mahasiswa");
            System.out.println( x: "5. Cek Koneksi Database");
            System.out.println( x: "6. Keluar");
            System.out.print( s: "PILIH OPSI: ");
            pilihan = scanner.nextInt();
            scanner.nextLine();
```

Output

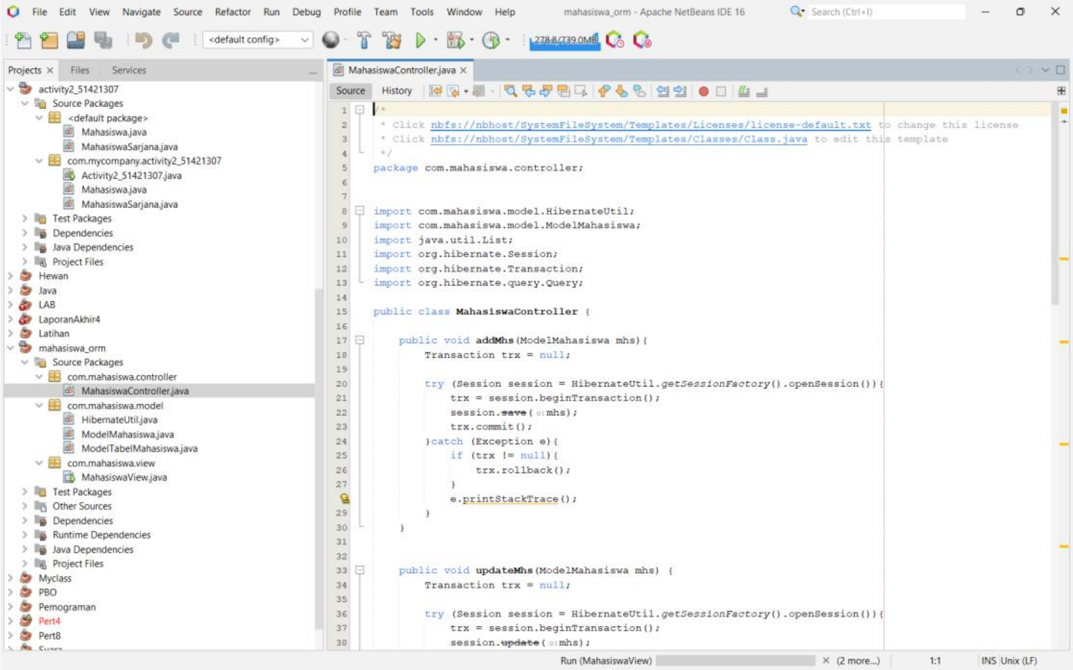


```
2. Tambah Mahasiswa
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI: 2
Masukkan NPM:
51421307
Masukkan Nama:
Iky
Masukkan Semester:
7
Masukkan IPK:
3.5
51421307Iky73.5
Controller Data:   51421307Iky73.5
me.iky.mahasiswa.model.ModelMahasiswa@7a30d1e6
Mahasiswa berhasil ditambahkan!
Menu:
1. Tampilkan Semua Mahasiswa
2. Tambah Mahasiswa
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI: 1

===========================
ID           : 1
NPM          : 51421307
NAMA         : Iky
SEMESTER     : 7
IPK          : 3.5
===========================
```

Activity 4

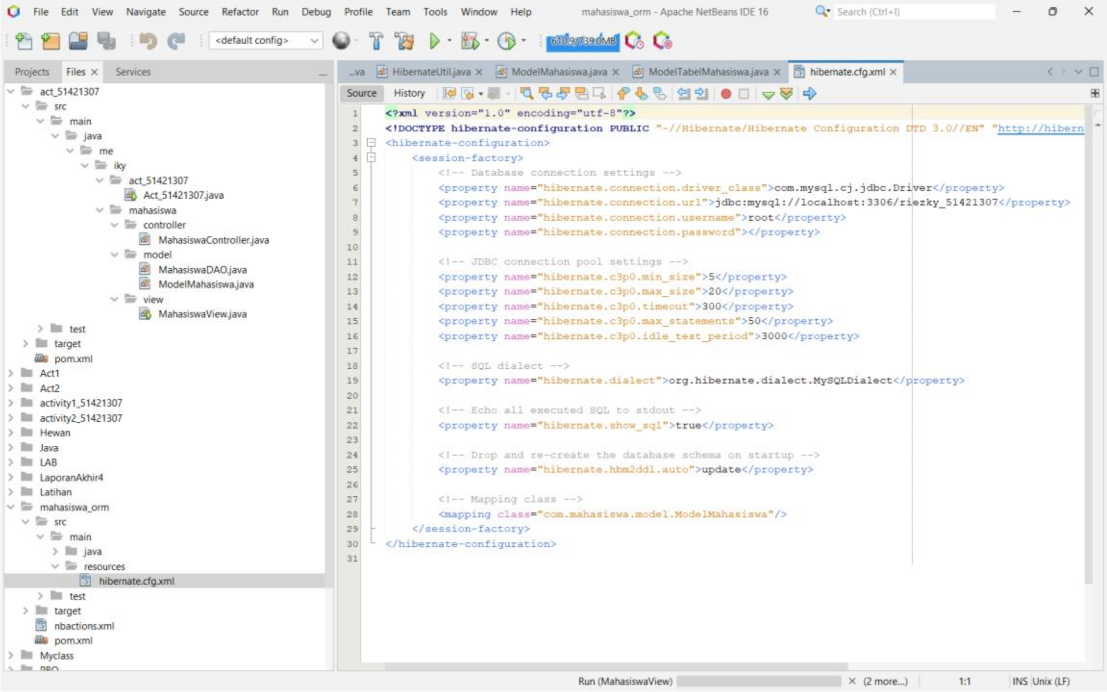
com.mahasiswa.controller

mahasiswa.controller


mahasiswa_orm - Apache NetBeans IDE 16
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
MahasiswaController.java
Source History
Run (MahasiswaView)
(2 more...)
1:1
INS Unix (LF)

```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.mahasiswa.controller;


import com.mahasiswa.model.HibernateUtil;
import com.mahasiswa.model.ModelMahasiswa;
import java.util.List;
import org.hibernate.Session;
import org.hibernate.Transaction;
import org.hibernate.query.Query;

public class MahasiswaController {

    public void addMhs(ModelMahasiswa mhs){
        Transaction trx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            session.save(mhs);
            trx.commit();
        }catch (Exception e){
            if (trx != null){
                trx.rollback();
            }
            e.printStackTrace();
        }
    }


    public void updateMhs(ModelMahasiswa mhs) {
        Transaction trx = null;

        try (Session session = HibernateUtil.getSessionFactory().openSession()){
            trx = session.beginTransaction();
            session.update(mhs);
```

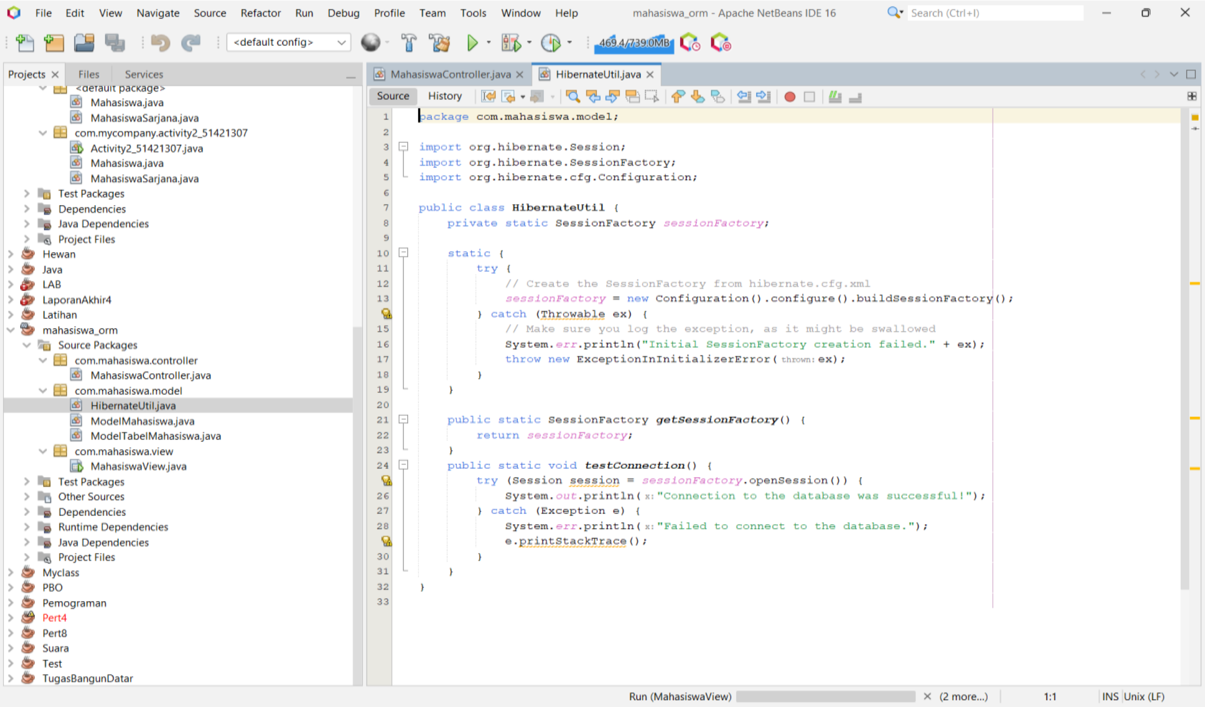
hibernate.cfg


mahasiswa_orm - Apache NetBeans IDE 16
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
HibernateUtil.java ModelMahasiswa.java ModelTabelMahasiswa.java hibernate.cfg.xml
Source History
Run (MahasiswaView)
(2 more...)
1:1
INS Unix (LF)

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibern
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="hibernate.connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/riezky_51421307</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>

        <!-- JDBC connection pool settings -->
        <property name="hibernate.c3p0.min_size">5</property>
        <property name="hibernate.c3p0.max_size">20</property>
        <property name="hibernate.c3p0.timeout">300</property>
        <property name="hibernate.c3p0.max_statements">50</property>
        <property name="hibernate.c3p0.idle_test_period">3000</property>

        <!-- SQL dialect -->
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

        <!-- Echo all executed SQL to stdout -->
        <property name="hibernate.show_sql">true</property>

        <!-- Drop and re-create the database schema on startup -->
        <property name="hibernate.hbm2ddl.auto">update</property>

        <!-- Mapping class -->
        <mapping class="com.mahasiswa.model.ModelMahasiswa"/>
    </session-factory>
</hibernate-configuration>
```

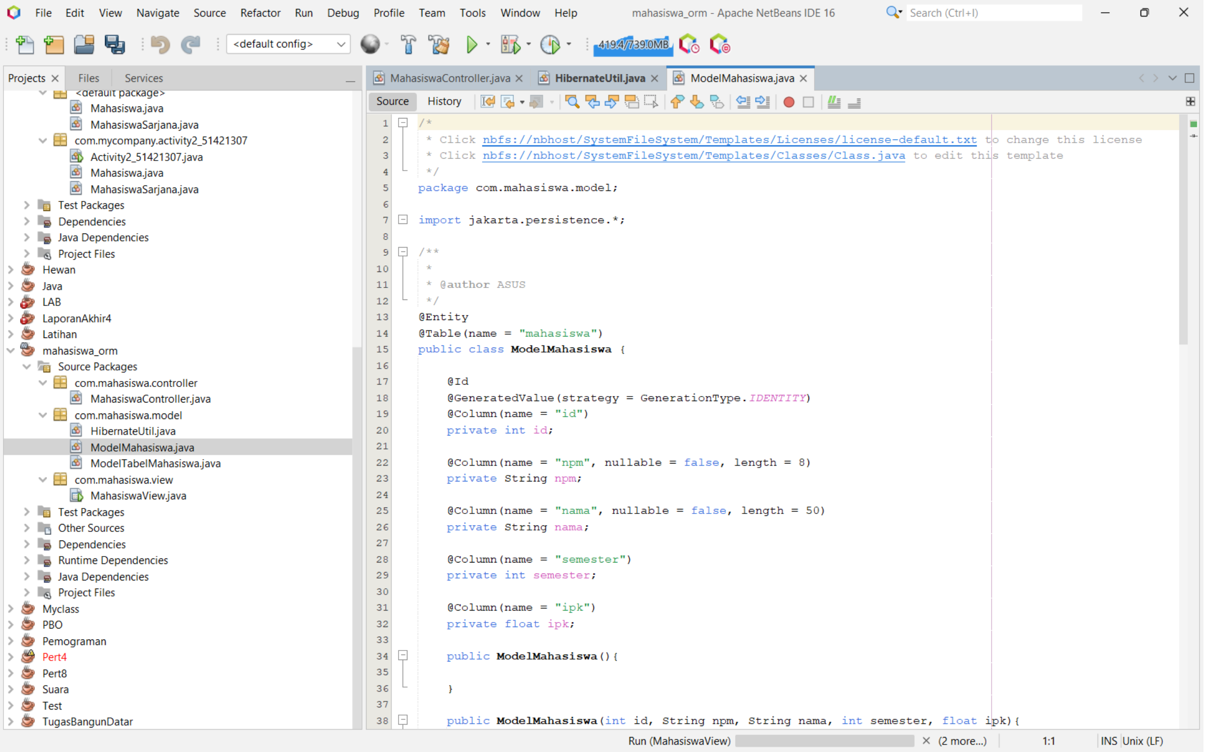
com.mahasiswa.model

HibernateUtil.java



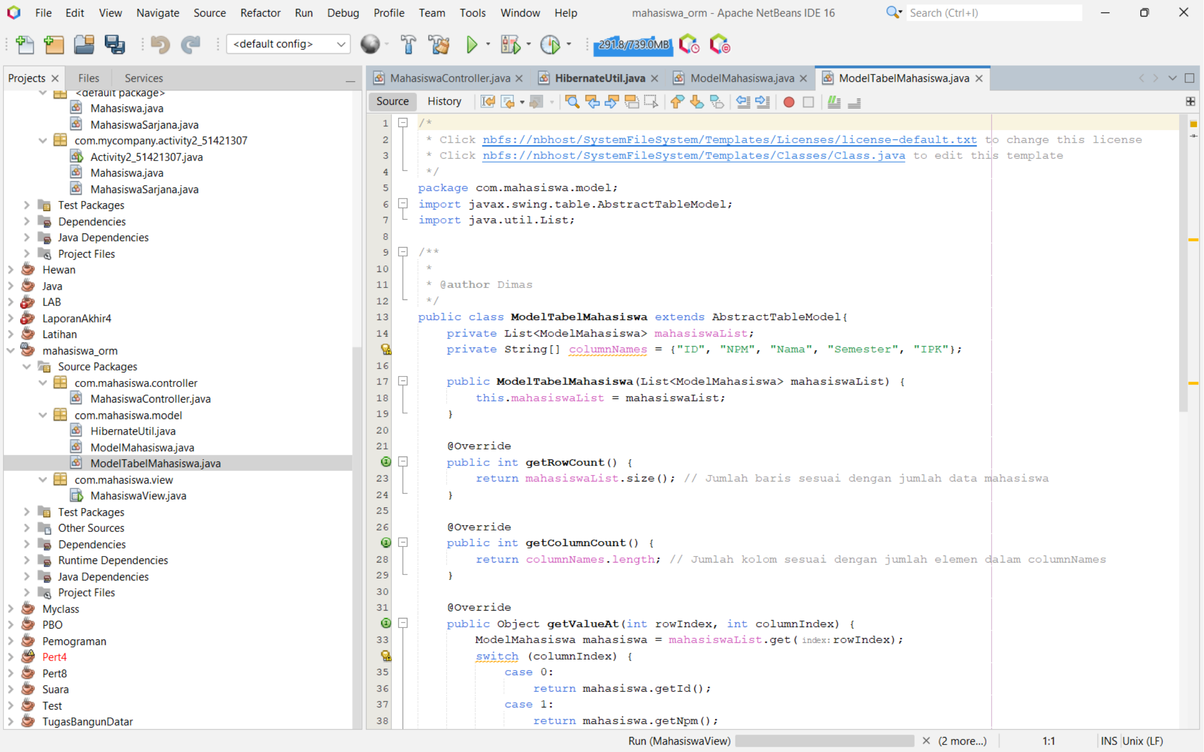
```java
package com.mahasiswa.model;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {
    private static SessionFactory sessionFactory;

    static {
        try {
            // Create the SessionFactory from hibernate.cfg.xml
            sessionFactory = new Configuration().configure().buildSessionFactory();
        } catch (Throwable ex) {
            // Make sure you log the exception, as it might be swallowed
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError( thrown: ex);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
    public static void testConnection() {
        try (Session session = sessionFactory.openSession()) {
            System.out.println( x: "Connection to the database was successful!");
        } catch (Exception e) {
            System.err.println( x: "Failed to connect to the database.");
            e.printStackTrace();
        }
    }
}
```

ModelMahasiswa.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.mahasiswa.model;

import jakarta.persistence.*;

/**
 *
 * @author ASUS
 */
@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){
```

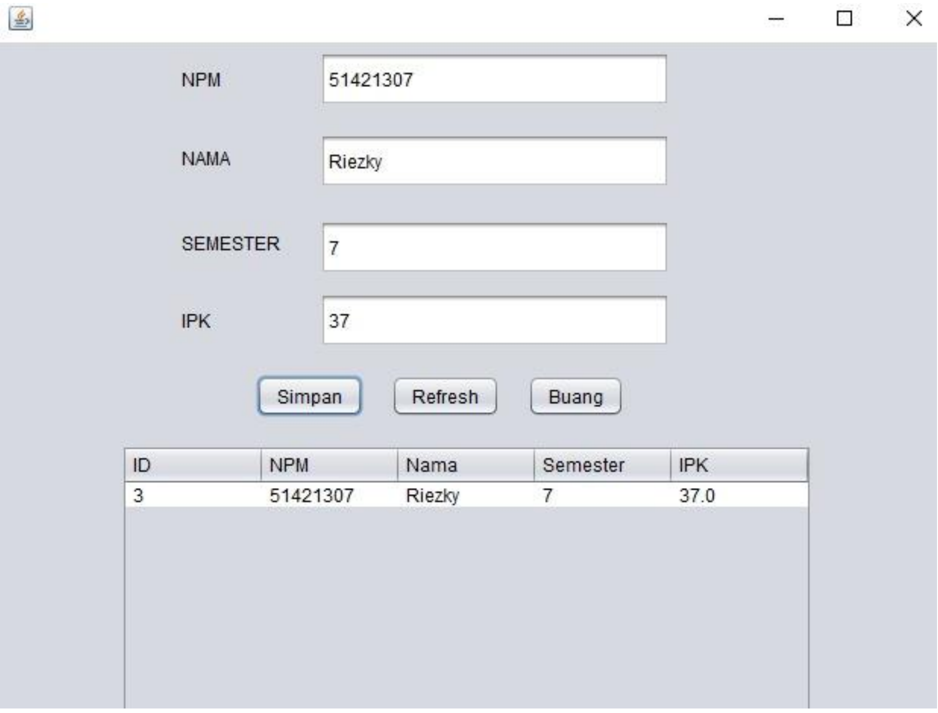
ModelTabelMahasiswa.java



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

/**
 *
 * @author Dimas
 */
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( index: rowIndex);
        switch (columnIndex) {
            case 0:
                return mahasiswa.getId();
            case 1:
                return mahasiswa.getNpm();
```

com.mahasiswa.view

Output



NPM 51421307

NAMA Riezky

SEMESTER 7

IPK 37

ID	NPM	Nama	Semester	IPK
3	51421307	Riezky	7	37.0

1. **Pengertian mvc ?**

Jawab :

MVC (Model-View-Controller) adalah pola desain (design pattern) yang membagi aplikasi menjadi tiga komponen utama, yaitu Model, View, dan Controller. Pola ini digunakan untuk memisahkan logika aplikasi dari antarmuka pengguna sehingga memudahkan pengembangan, pemeliharaan, dan pembaruan aplikasi.

2. **Pengertian orm dan hibernate ?**

Jawab :

- **ORM (Object-Relational Mapping)** adalah teknik yang digunakan untuk menghubungkan objek dalam bahasa pemrograman dengan tabel pada database relasional. ORM memungkinkan pengembang untuk berinteraksi dengan database menggunakan objek, sehingga memudahkan manipulasi data tanpa harus menulis perintah SQL secara langsung.

- Hibernate adalah framework ORM untuk bahasa Java yang mempermudah pengembang dalam melakukan operasi database dengan mengkonversi tabel database menjadi objek Java. Hibernate menyediakan fitur untuk mengelola data secara otomatis, seperti koneksi database, pemetaan objek ke tabel, dan lain-lain.

3. **Perbedaan hibernate dan mvc konvensional biasa?**

Jawab:

- **Hibernate** berfokus pada pemetaan objek ke database dan pengelolaan data secara otomatis menggunakan ORM. Ini memudahkan interaksi dengan database tanpa harus menulis kode SQL secara langsung.
- **MVC Konvensional** adalah pola desain untuk memisahkan logika bisnis, tampilan, dan alur data aplikasi, namun tidak mencakup pengelolaan data ke database. Dalam MVC konvensional, pengelolaan data masih harus dilakukan dengan menulis kode SQL atau menggunakan metode manual.