**LAPORAN AKHIR PRAKTIKUM**

Mata Praktikum : Rekayasa Perangkat Lunak 2

Kelas : 4IA06

Praktikum ke- : 3

Tanggal : 29 Oktober 2024

Materi : Konsep Model-View-Controller (MVC), Pembuatan Program dengan Konsep MVC, Instalasi & Penerapan, dan Akses Database dengan JDBC

NPM : 51421517

Nama : William Devin Septianus Pranggono

Ketua Asisten :

Paraf Asisten :

Nama Asisten : Gilbert Jefferson Faozato Mendrofa

Jumlah Lembar : 25 lembar

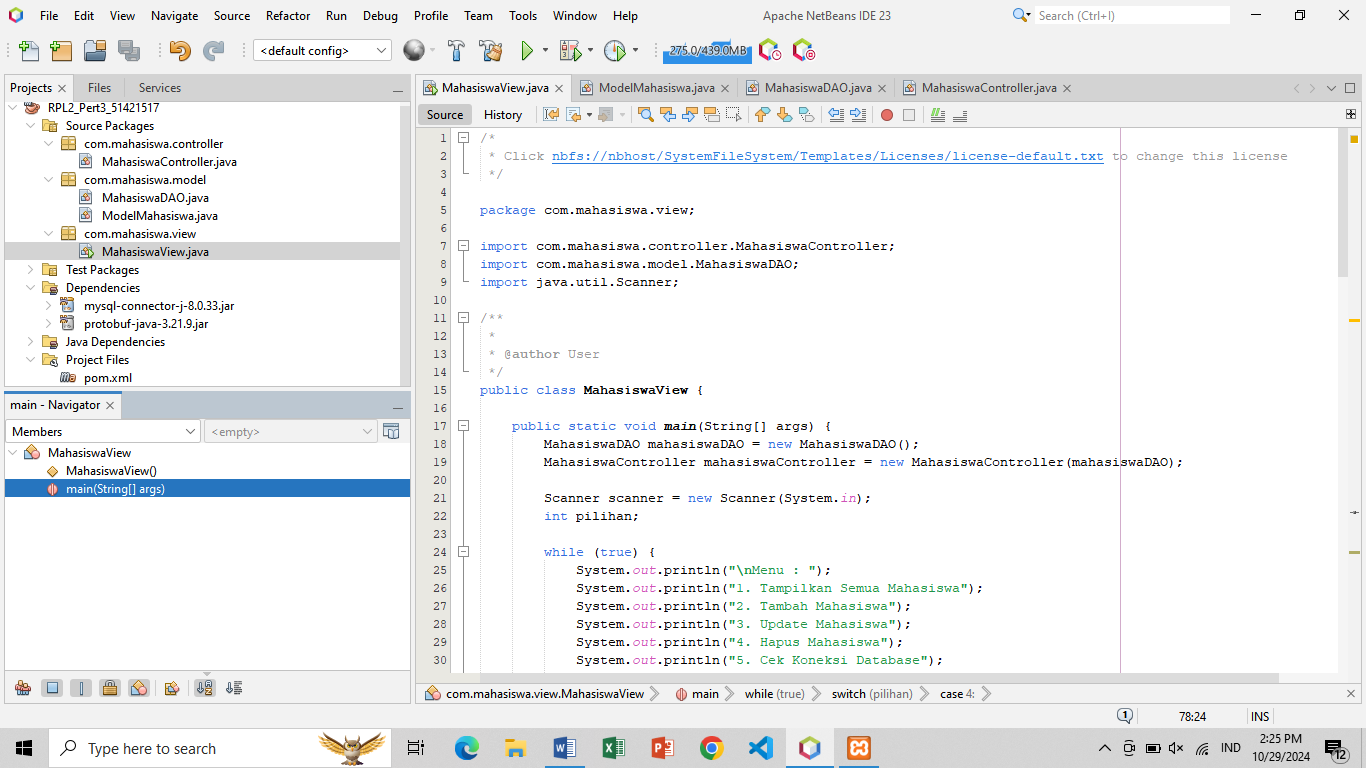
**LABORATORIUM TEKNIK INFORMATIKA**

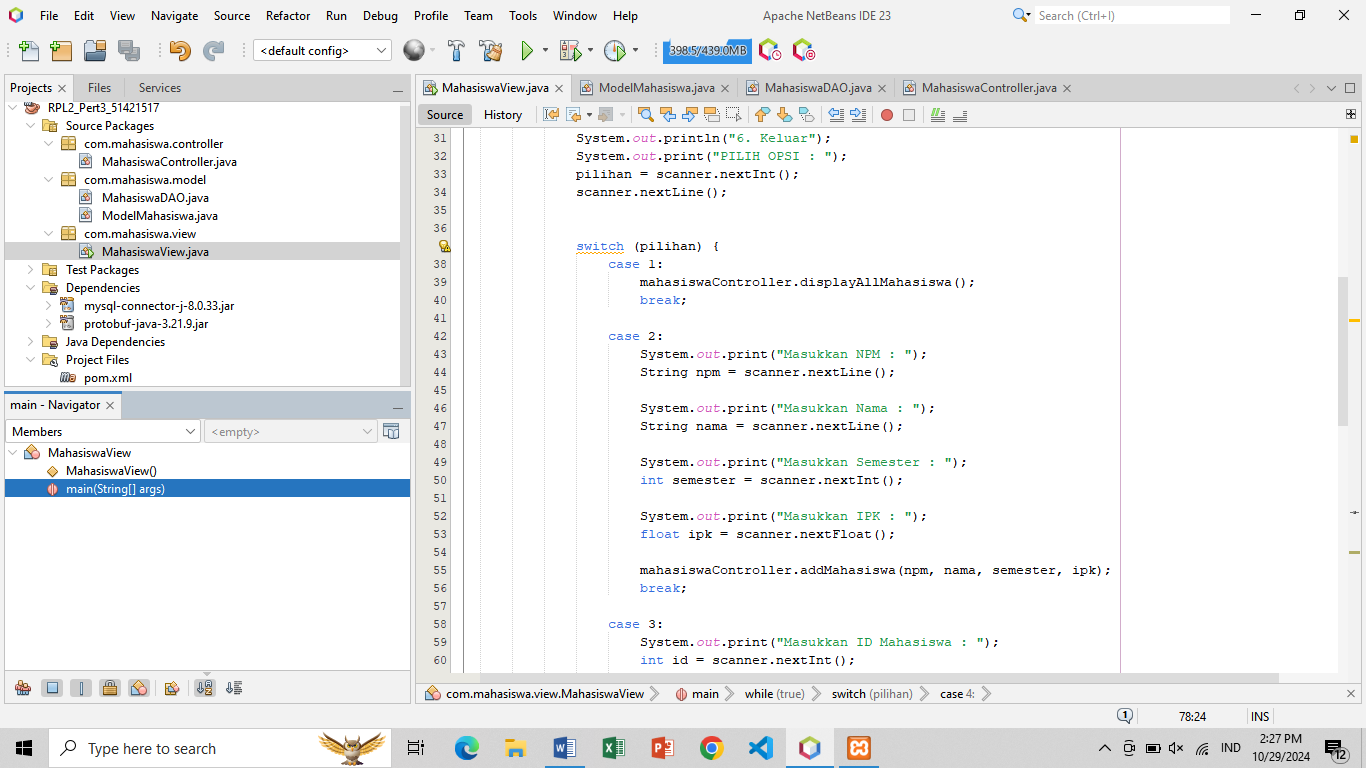
**UNIVERSITAS GUNADARMA**

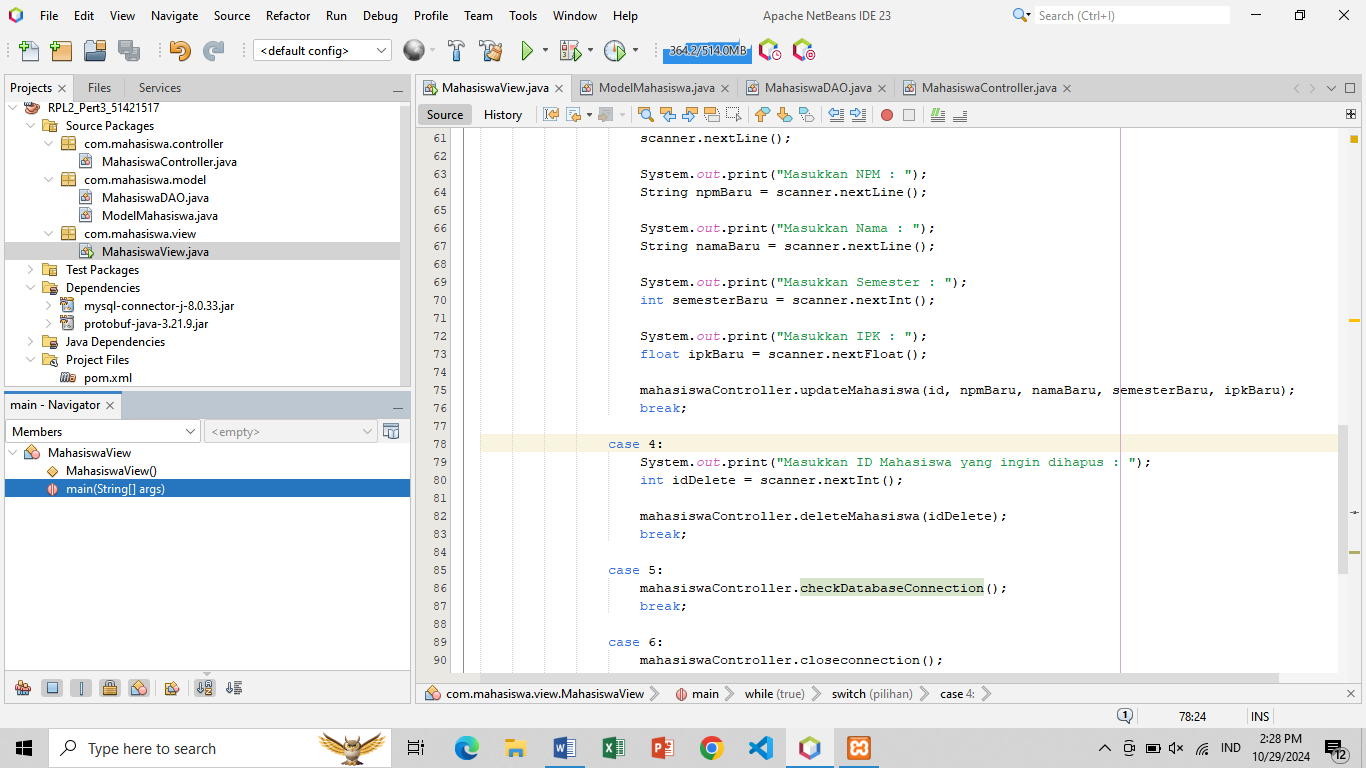
**2024**

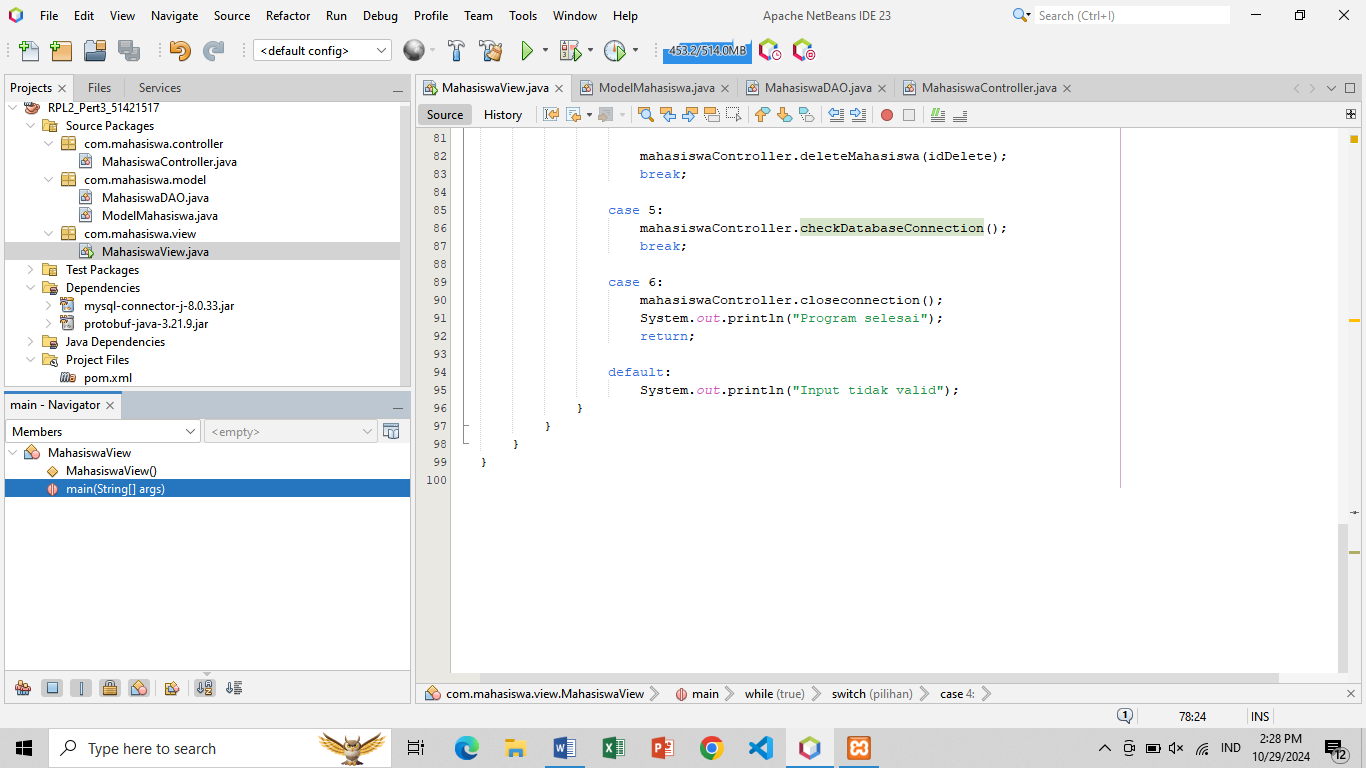
**LISTING PROGRAM**

1. Mahasiswa.java

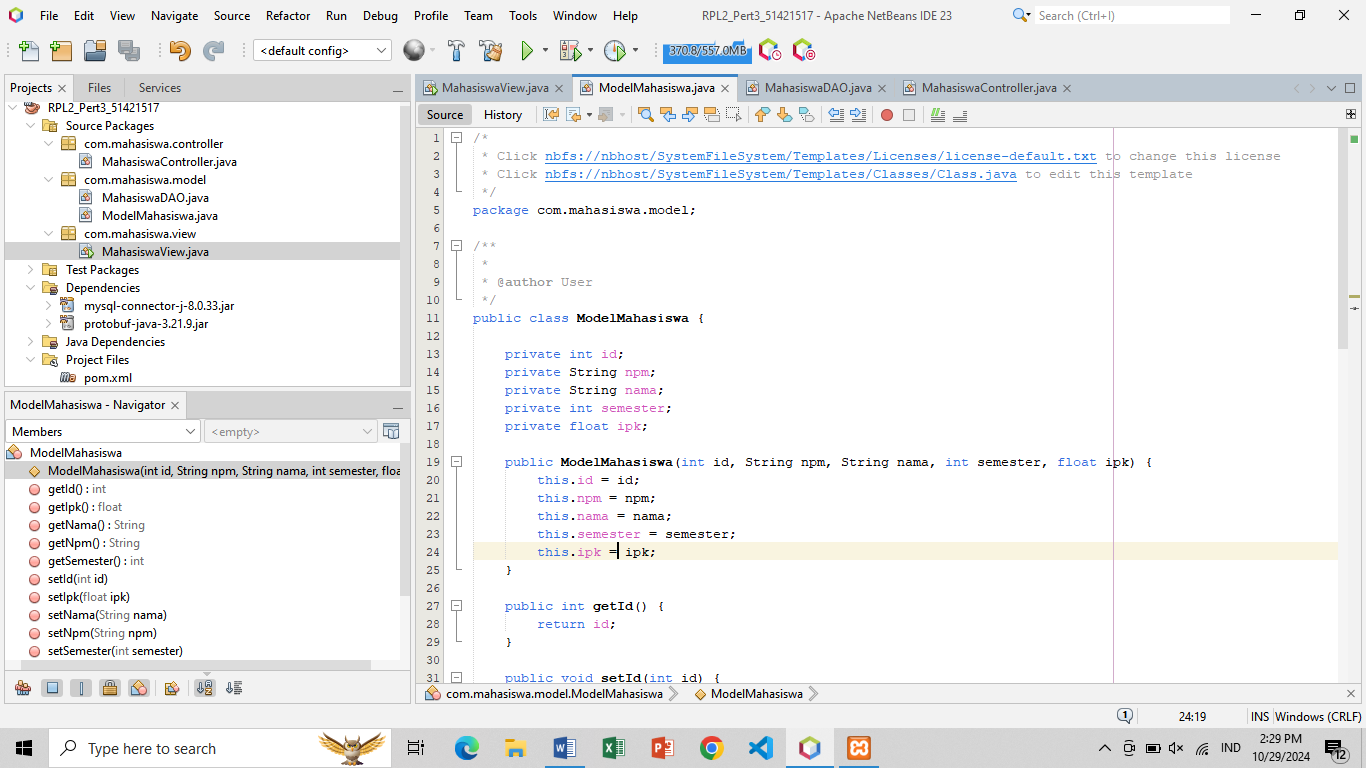


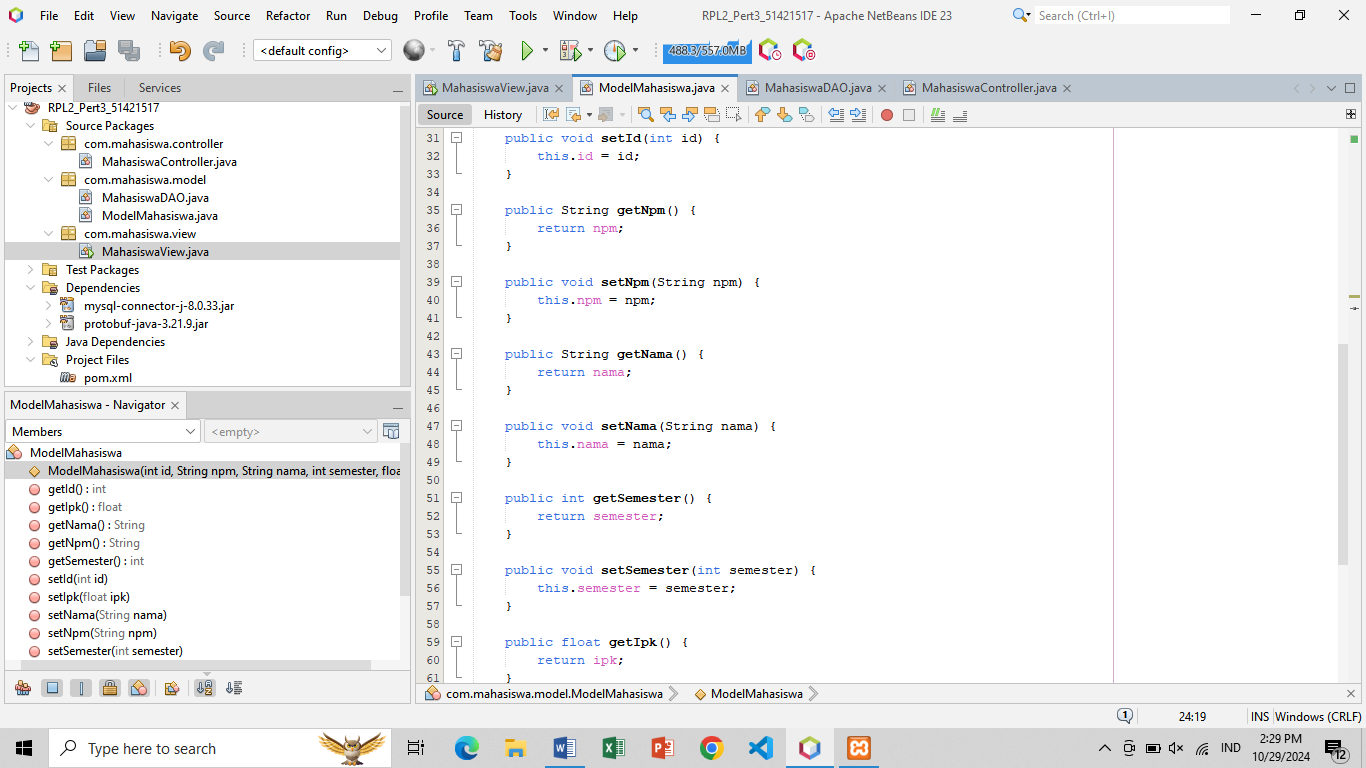


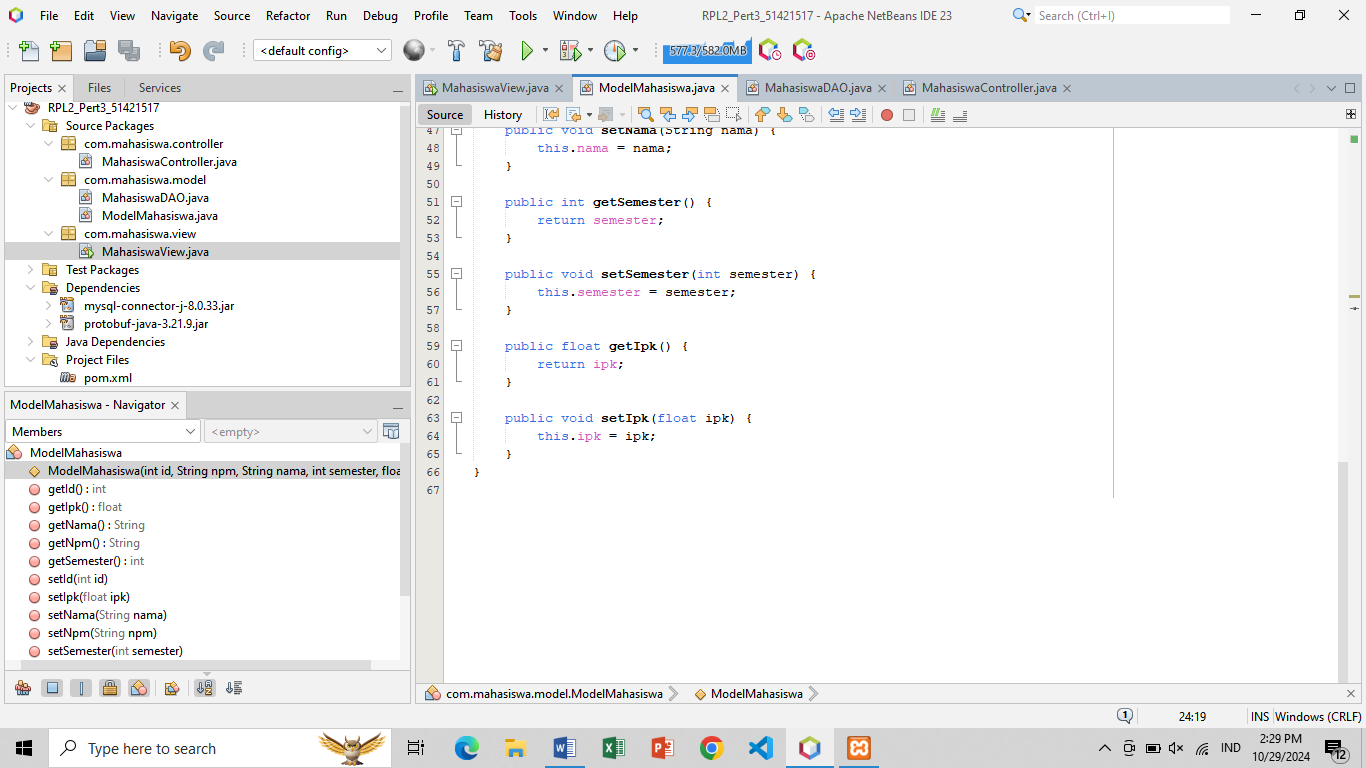




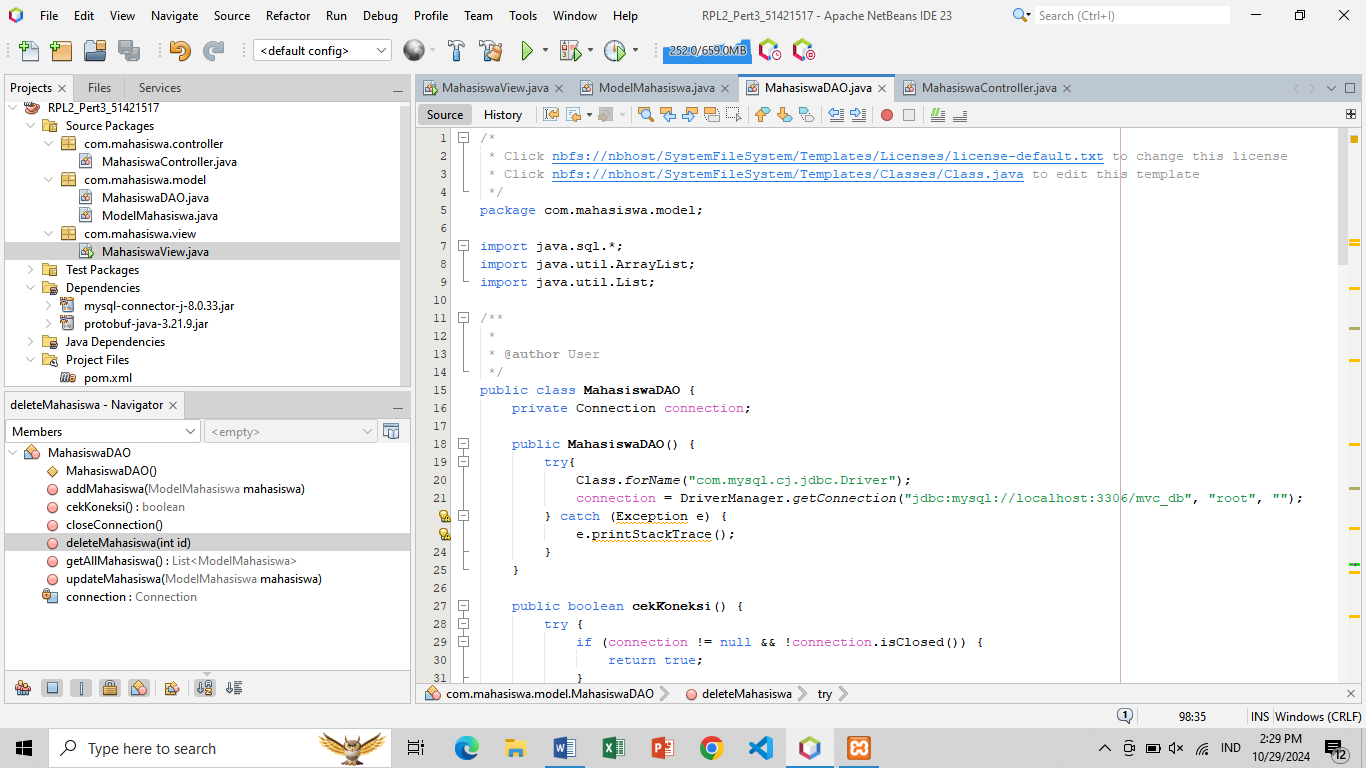
1. ModelMahasiswa.java

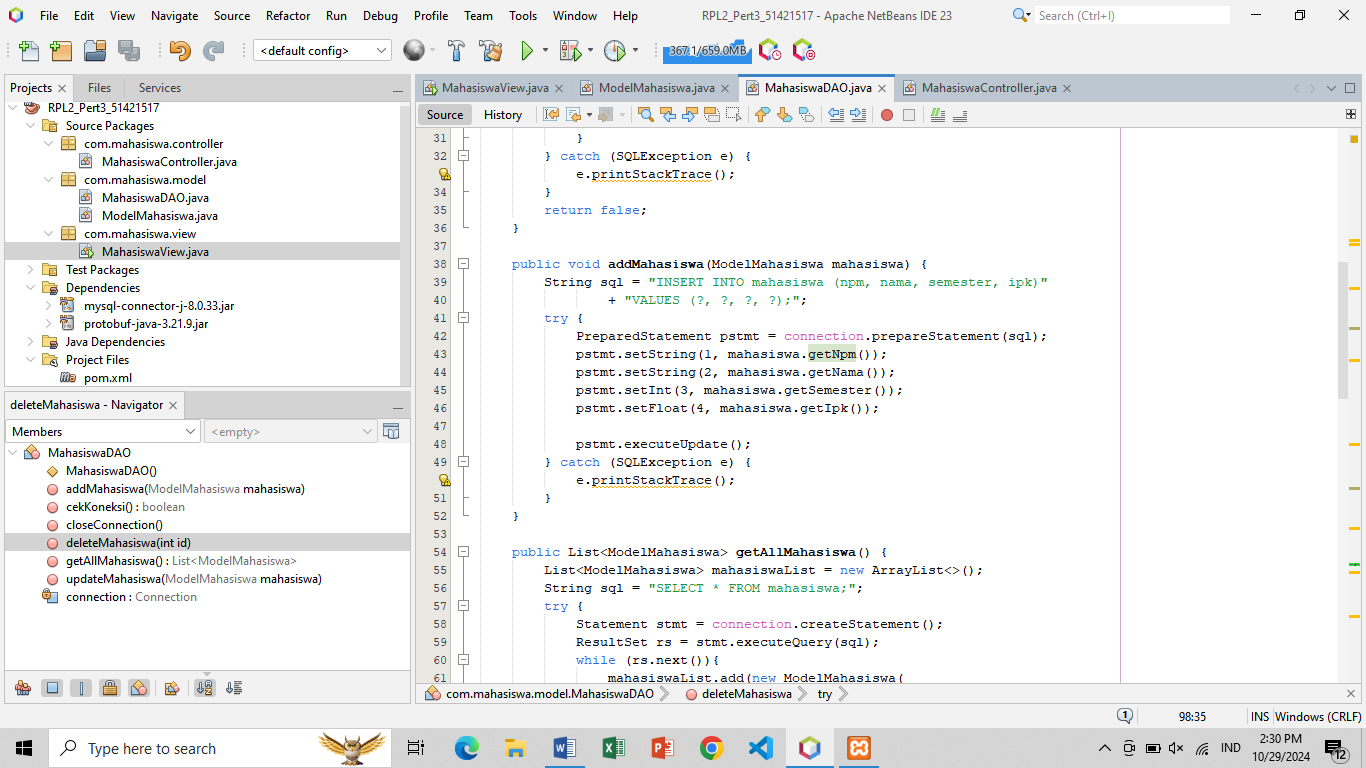


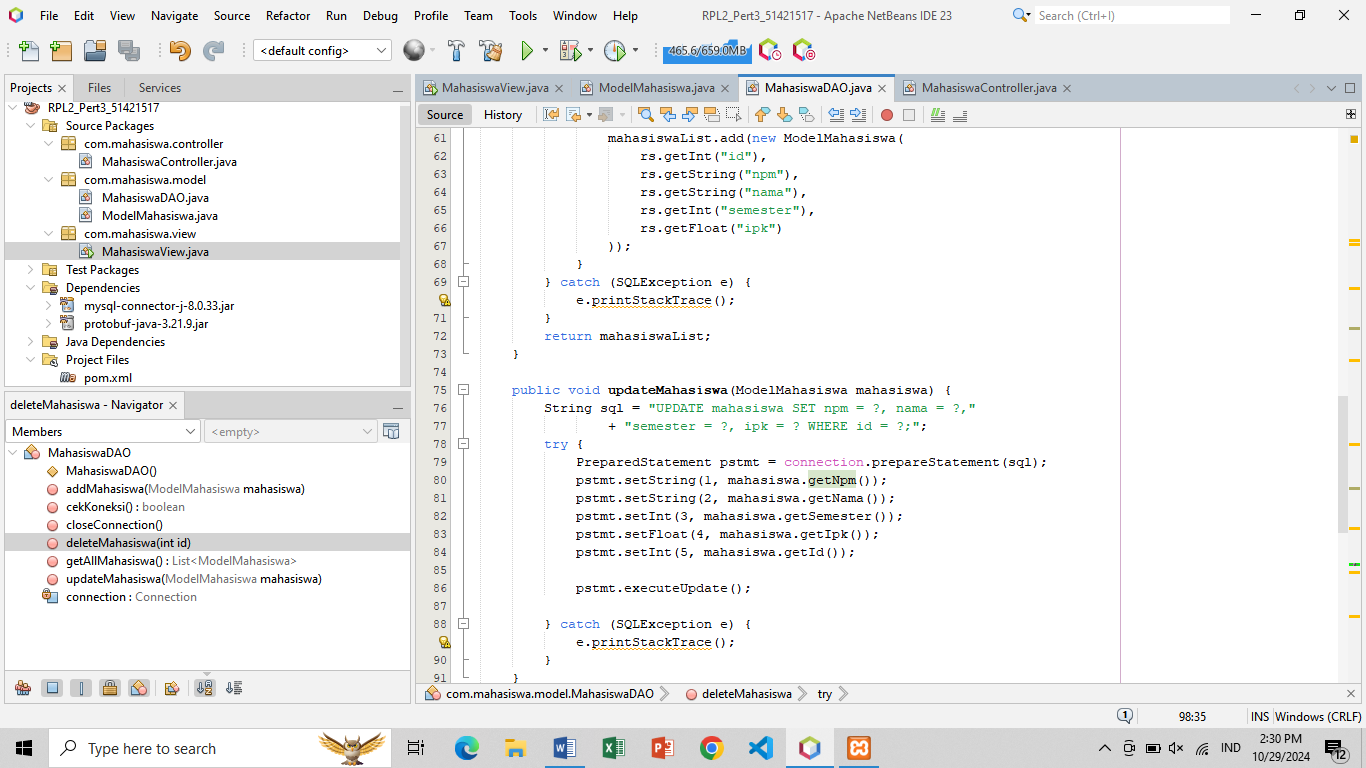


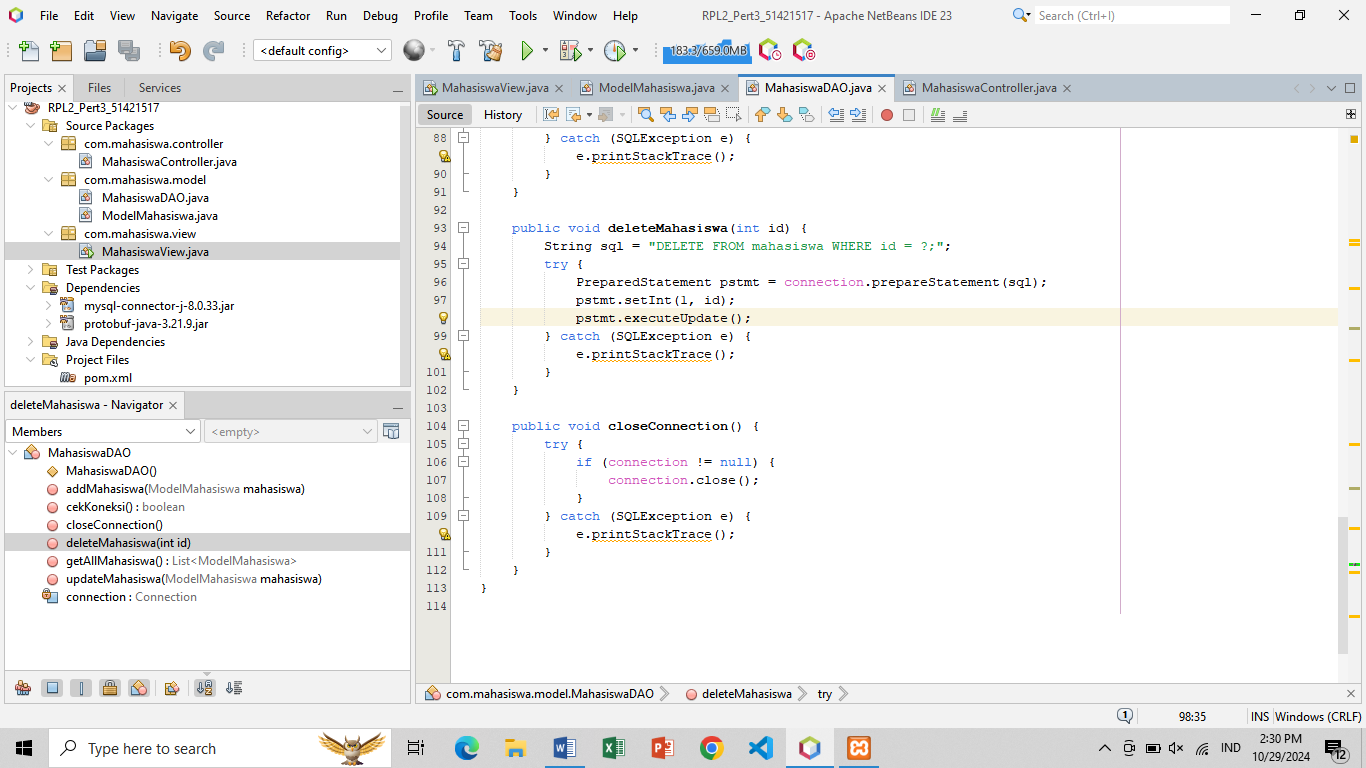


1. MahasiswaDAO.java

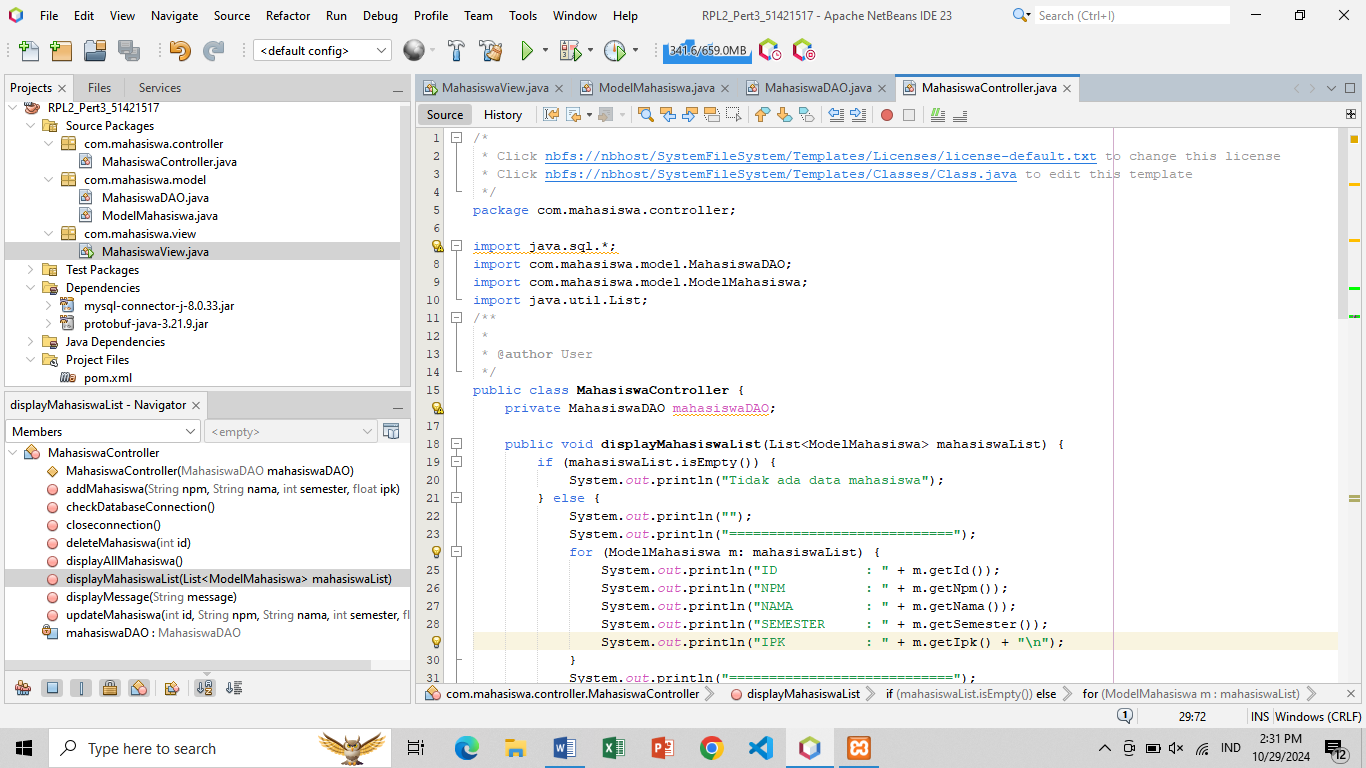


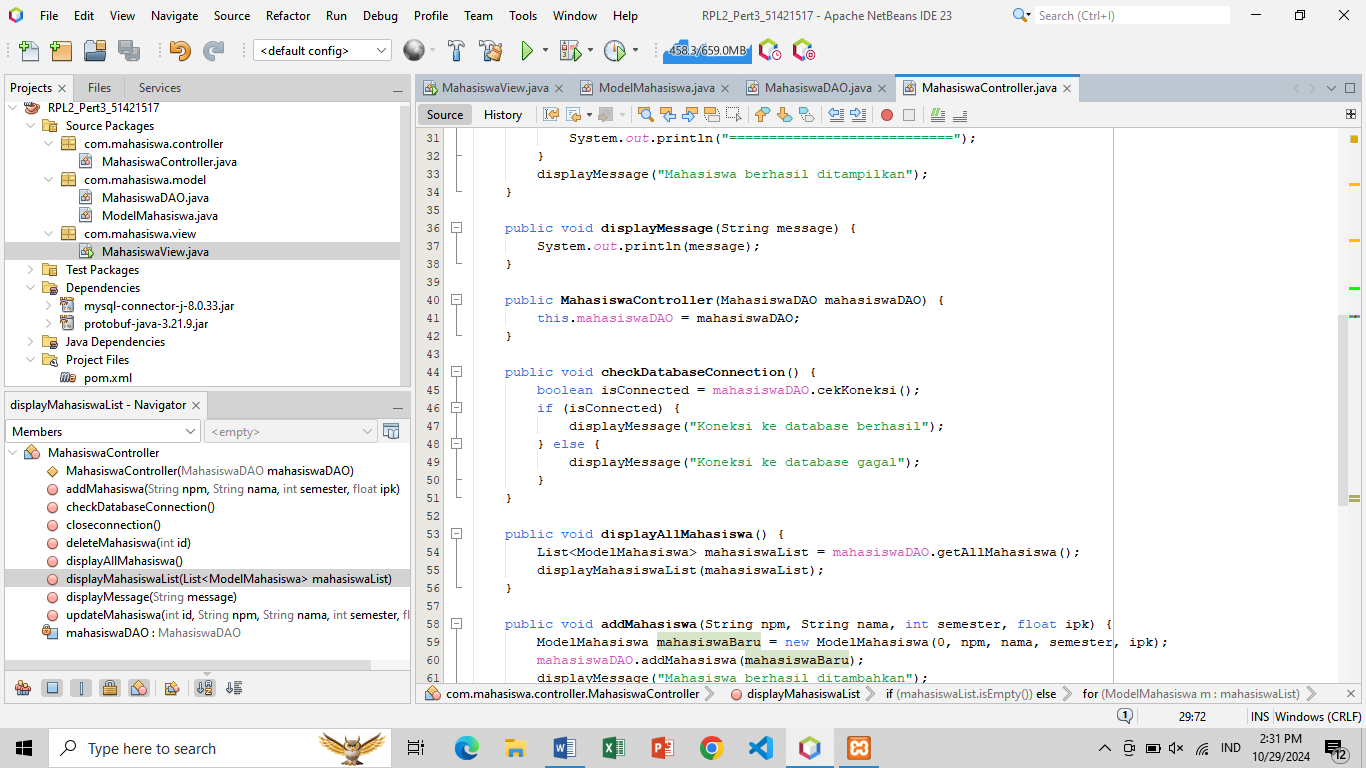


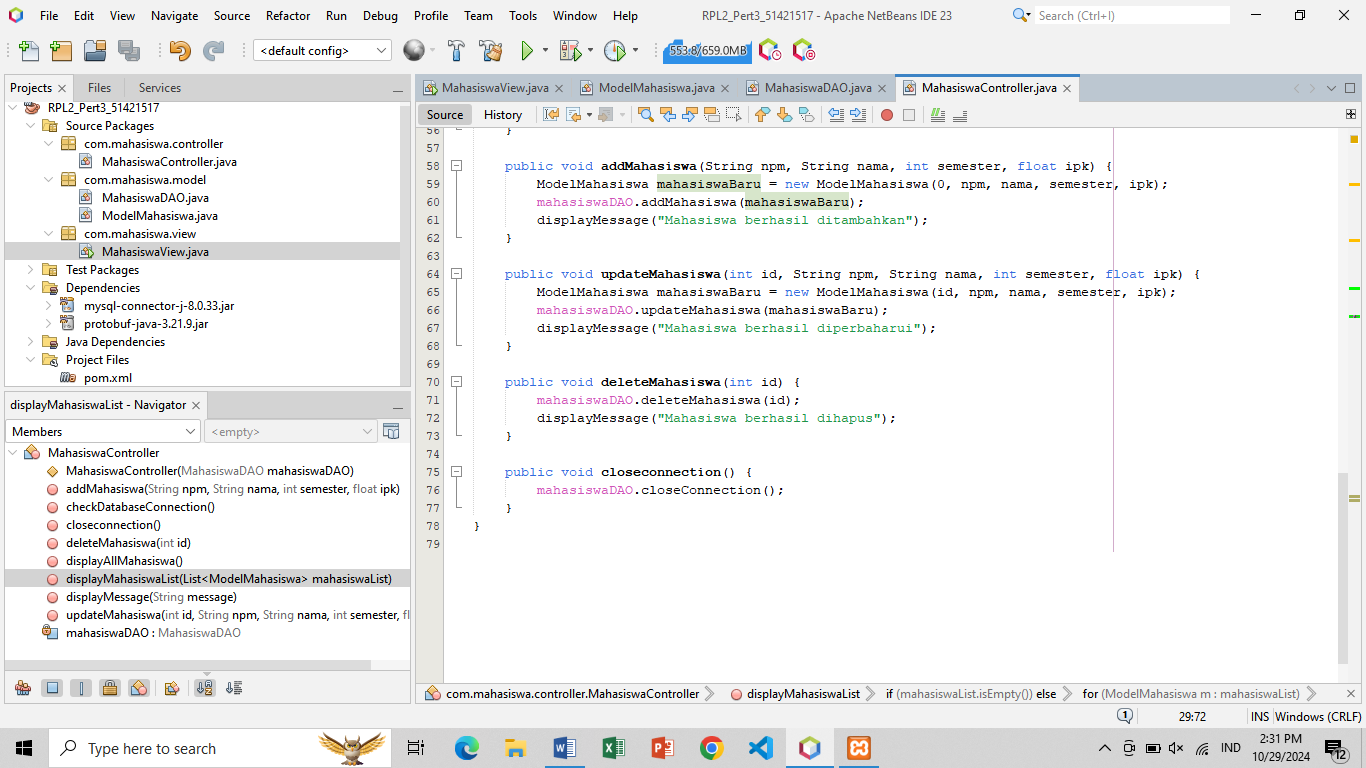




1. MahasiswaController.java







**LOGIKA PROGRAM**

1. ModelMahasiswa.java

Langkah pertama adalah mendefinisikan atribut dari *class* ModelMahasiswa. Atribut memiliki *identifier private*, sehingga memerlukan *method getter* dan *setter*.

public class **ModelMahasiswa** {

private int id;

private String npm;

private String nama;

private int semester;

private float ipk;

Kemudian, di bawah ini adalah kode untuk *constructor* ketika objek dari *class* diinstansiasi.

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;

}

Terakhir, berikut adalah *method getter* dan *setter* untuk memodifikasi dan mengakses atribut dari *class*.

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;

}

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;

}

}

1. MahasiswaDAO.java

MahasiswaDAO berperan sebagai *Data Access Object* yang menyediakan akses terhadap *database* dan sebagai perantara antara *controller* dan *database*. *Class* MahasiswaDAO memiliki atribut *connection*, yaitu objek *Connection* dari *package* java.sql.\*.

public class **MahasiswaDAO** {

private Connection connection;

Kemudian, *constructor* untuk *class* MahasiswaDAO didefinisikan tanpa menerima parameter apa pun. Ketika *class* MahasiswaDAO diinstansiasi, maka akan memuat Driver JDBC yang memungkinkan Java untuk terhubung ke *database*. Kemudian, atribut *connection* adalah objek *Connection* yang akan terhubung dengan *database* sesuai URL yang diberikan.

public **MahasiswaDAO**() {

try{

Class.forName("com.mysql.cj.jdbc.Driver");

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mvc\_db", "root", "");

} catch (Exception e) {

e.printStackTrace();

}

}

*Method* cekKoneksi akan mengecek apakah *connection* memiliki nilai (berhasil terhubung ke *database*) dan apakah *connection* tidak tertutup, dan akan mengembalikan nilai *true* jika kedua syarat tersebut terpenuhi, dan *false* untuk sebaliknya.

public boolean **cekKoneksi**() {

try {

if (connection != null && !connection.isClosed()) {

return true;

}

} catch (SQLException e) {

e.printStackTrace();

}

return false;

}

*Method* addMahasiswa menerima objek ModelMahasiswa dan akan menambahkan data ke tabel mahasiswa dengan atribut yang dimiliki objek ModelMahasiswa.

public void **addMahasiswa**(ModelMahasiswa mahasiswa) {

String sql = "INSERT INTO mahasiswa (npm, nama, semester, ipk)"

+ "VALUES (?, ?, ?, ?);";

try {

PreparedStatement pstmt = connection.prepareStatement(sql);

pstmt.setString(1, mahasiswa.getNpm());

pstmt.setString(2, mahasiswa.getNama());

pstmt.setInt(3, mahasiswa.getSemester());

pstmt.setFloat(4, mahasiswa.getIpk());

pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

Kemudian, *method* getAllMahasiswa akan mengambil seluruh data dari tabel mahasiswa pada *database* dan akan mengembalikan List bertipe data ModelMahasiswa.

public List<ModelMahasiswa> **getAllMahasiswa**() {

List<ModelMahasiswa> mahasiswaList = new ArrayList<>();

String sql = "SELECT \* FROM mahasiswa;";

try {

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()){

mahasiswaList.add(new ModelMahasiswa(

rs.getInt("id"),

rs.getString("npm"),

rs.getString("nama"),

rs.getInt("semester"),

rs.getFloat("ipk")

));

}

} catch (SQLException e) {

e.printStackTrace();

}

return mahasiswaList;

}

*Method* updateMahasiswa akan memperbarui *record* yang telah ada dengan data dari parameter mahasiswa bertipe data ModelMahasiswa.

public void **updateMahasiswa**(ModelMahasiswa mahasiswa) {

String sql = "UPDATE mahasiswa SET npm = ?, nama = ?,"

+ "semester = ?, ipk = ? WHERE id = ?;";

try {

PreparedStatement pstmt = connection.prepareStatement(sql);

pstmt.setString(1, mahasiswa.getNpm());

pstmt.setString(2, mahasiswa.getNama());

pstmt.setInt(3, mahasiswa.getSemester());

pstmt.setFloat(4, mahasiswa.getIpk());

pstmt.setInt(5, mahasiswa.getId());

pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

*Method* deleteMahasiswa akan menghapus *record* dari tabel mahasiswa berdasarkan parameter *id* yang diberikan.

public void **deleteMahasiswa**(int id) {

String sql = "DELETE FROM mahasiswa WHERE id = ?;";

try {

PreparedStatement pstmt = connection.prepareStatement(sql);

pstmt.setInt(1, id);

pstmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

*Method* closeConnection akan menutup koneksi dengan *database* apabila variabel *connection* tidak bernilai *null*.

public void **closeConnection**() {

try {

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

1. MahasiswaController.java

*Class* MahasiswaController berperan sebagai *Controller*, yang merupakan perantara *Model* dengan *View*. Atribut yang dimiliki adalah objek MahasiswaDAO.

public class **MahasiswaController** {

private MahasiswaDAO mahasiswaDAO;

*Method* displayMahasiswaList berfungsi untuk menampilkan seluruh data dari tabel mahasiswa. Data tersebut merupakan parameter mahasiswaList bertipe data List ModelMahasiswa.

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() + "\n");

}

System.out.println("============================");

}

displayMessage("Mahasiswa berhasil ditampilkan");

}

*Method* displayMessage berfungsi untuk menampilkan String message yang merupakan parameter dari *method* ke layar CLI.

public void **displayMessage**(String message) {

System.out.println(message);

}

*Constructor* MahasiswaController menginisiasi atribut mahasiswaDAO dengan parameter bertipe data MahasiswaDAO.

public **MahasiswaController**(MahasiswaDAO mahasiswaDAO) {

this.mahasiswaDAO = mahasiswaDAO;

}

*Method* checkDatabaseConnection berfungsi memeriksa status koneksi *database* dengan memanggil *method* cekKoneksi milik objek mahasiswaDAO, dan akan menampilkan pesan status ke layar CLI.

public void **checkDatabaseConnection**() {

boolean isConnected = mahasiswaDAO.cekKoneksi();

if (isConnected) {

displayMessage("Koneksi ke database berhasil");

} else {

displayMessage("Koneksi ke database gagal");

}

}

*Method* displayAllMahasiswa berfungsi untuk menampilkan seluruh data mahasiswa dengan memanggil *method* getAllMahasiswa milik objek mahasiswaDAO untuk mendapatkan data, dan displayMahasiswaList untuk menampilkan data.

public void **displayAllMahasiswa**() {

List<ModelMahasiswa> mahasiswaList = mahasiswaDAO.getAllMahasiswa();

displayMahasiswaList(mahasiswaList);

}

*Method* addMahasiswa berfungsi untuk menambahkan *record* ke tabel mahasiswa menggunakan data yang diberikan dari parameter *method*. Penambahan *record* dilakukan dengan memanggil *method* addMahasiswa milik objek mahasiswaDAO.

public void **addMahasiswa**(String npm, String nama, int semester, float ipk) {

ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(0, npm, nama, semester, ipk);

mahasiswaDAO.addMahasiswa(mahasiswaBaru);

displayMessage("Mahasiswa berhasil ditambahkan");

}

*Method* updateMahasiswa berfungsi memperbarui data dengan nilai yang diberikan pada parameter. Pembaruan data dilakukan dengan memanggil *method* updateMahasiswa milik objek mahasiswaDAO.

public void **updateMahasiswa**(int id, String npm, String nama, int semester, float ipk) {

ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(id, npm, nama, semester, ipk);

mahasiswaDAO.updateMahasiswa(mahasiswaBaru);

displayMessage("Mahasiswa berhasil diperbaharui");

}

*Method* deleteMahasiswa berfungsi menghapus *record* dari tabel mahasiswa berdasarkan *id* yang diberikan pada parameter.

public void **deleteMahasiswa**(int id) {

mahasiswaDAO.deleteMahasiswa(id);

displayMessage("Mahasiswa berhasil dihapus");

}

*Method* closeConnection berfungsi menutup koneksi ke *database* dengan memanggil *method* closeConnection milik objeck mahasiswaDAO.

public void **closeconnection**() {

mahasiswaDAO.closeConnection();

}

}

1. MahasiswaView.java

*Class* ini adalah *class* yang memuat *method main* sebagai *method* utama, dan sekaligus berperan sebagai *View*, yang akan menampilkan *prompt* dan data ke layar CLI.

public class **MahasiswaView** {

public static void **main**(String[] args) {

MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();

MahasiswaController mahasiswaController = new MahasiswaController(mahasiswaDAO);

Scanner scanner = new Scanner(System.in);

int pilihan;

while (true) {

System.out.println("\nMenu : ");

System.out.println("1. Tampilkan Semua Mahasiswa");

System.out.println("2. Tambah Mahasiswa");

System.out.println("3. Update Mahasiswa");

System.out.println("4. Hapus Mahasiswa");

System.out.println("5. Cek Koneksi Database");

System.out.println("6. Keluar");

System.out.print("PILIH OPSI : ");

pilihan = scanner.nextInt();

scanner.nextLine();

switch (pilihan) {

case 1:

mahasiswaController.displayAllMahasiswa();

break;

case 2:

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();

mahasiswaController.addMahasiswa(npm, nama, semester, ipk);

break;

case 3:

System.out.print("Masukkan ID Mahasiswa : ");

int id = scanner.nextInt();

scanner.nextLine();

System.out.print("Masukkan NPM : ");

String npmBaru = scanner.nextLine();

System.out.print("Masukkan Nama : ");

String namaBaru = scanner.nextLine();

System.out.print("Masukkan Semester : ");

int semesterBaru = scanner.nextInt();

System.out.print("Masukkan IPK : ");

float ipkBaru = scanner.nextFloat();

mahasiswaController.updateMahasiswa(id, npmBaru, namaBaru, semesterBaru, ipkBaru);

break;

case 4:

System.out.print("Masukkan ID Mahasiswa yang ingin dihapus : ");

int idDelete = scanner.nextInt();

mahasiswaController.deleteMahasiswa(idDelete);

break;

case 5:

mahasiswaController.checkDatabaseConnection();

break;

case 6:

mahasiswaController.closeconnection();

System.out.println("Program selesai");

return;

default:

System.out.println("Input tidak valid");

}

}

}

}

**OUTPUT PROGRAM**

