LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Rekayasa Perangkat Lunak 2
Kelas	: 4IA07
Praktikum ke-	: 3
Tanggal	: 1 November 2025
Materi	: MVC (Model – View – Controller)
NPM	: 51422084
Nama	: Muhammad Izzuddin Almansyur
Ketua Asisten	: Haikal Abizar
Jumlah Lembar	: 8

LABORATORIUM INFORMATIKA

UNIVERSITAS GUNADARMA

2025

1. Apa yang kalian ketahui tentang MVC?

   MVC adalah singkatan dari **Model-View-Controller**, sebuah pola desain yang digunakan dalam pengembangan aplikasi, terutama aplikasi web dan desktop. Tujuannya adalah untuk memisahkan logika aplikasi agar lebih mudah dikelola, diuji, dan dikembangkan.

   - **Model**: Bagian yang menangani data dan logika bisnis. Ia berinteraksi langsung dengan database dan bertanggung jawab atas pengambilan, penyimpanan, dan manipulasi data.
   - **View**: Bagian yang bertanggung jawab untuk menampilkan data kepada pengguna. Ini adalah antarmuka pengguna (UI) yang menerima data dari Model dan menampilkannya dengan cara yang ramah pengguna.
   - **Controller**: Penghubung antara Model dan View. Ia menerima input dari pengguna, memprosesnya (misalnya validasi), dan menentukan bagaimana Model dan View harus bereaksi.

2. Jelaskan kode dan langkah-langkah program yang telah dibuat!

   Langkah – langkah pembuatan program pada netbeans:

   1) Buka **NetBeans → File → New Project → Maven → Java Application.**
   2) Beri nama, lalu klik **Finish**
   3) Buka file pom.xml, dan tambahkan dependency
   4) Simpan, lalu Maven akan otomatis download library JDBC.
   5) Di dalam src/main/java/com/mycompany/rpl2_pert3/:
      Klik kanan → New → Java Package dan beri nama

   - Model
   - Controller
   - View

   6) Buat class java di masing – masing package:

   - Model = ModelMahasiswa, MahasiswaDAO
   - Controller = MahasiswaController
   - View = MahasiswaView

   7) Buat database baru dengan xampp beserta tabel tabelnya

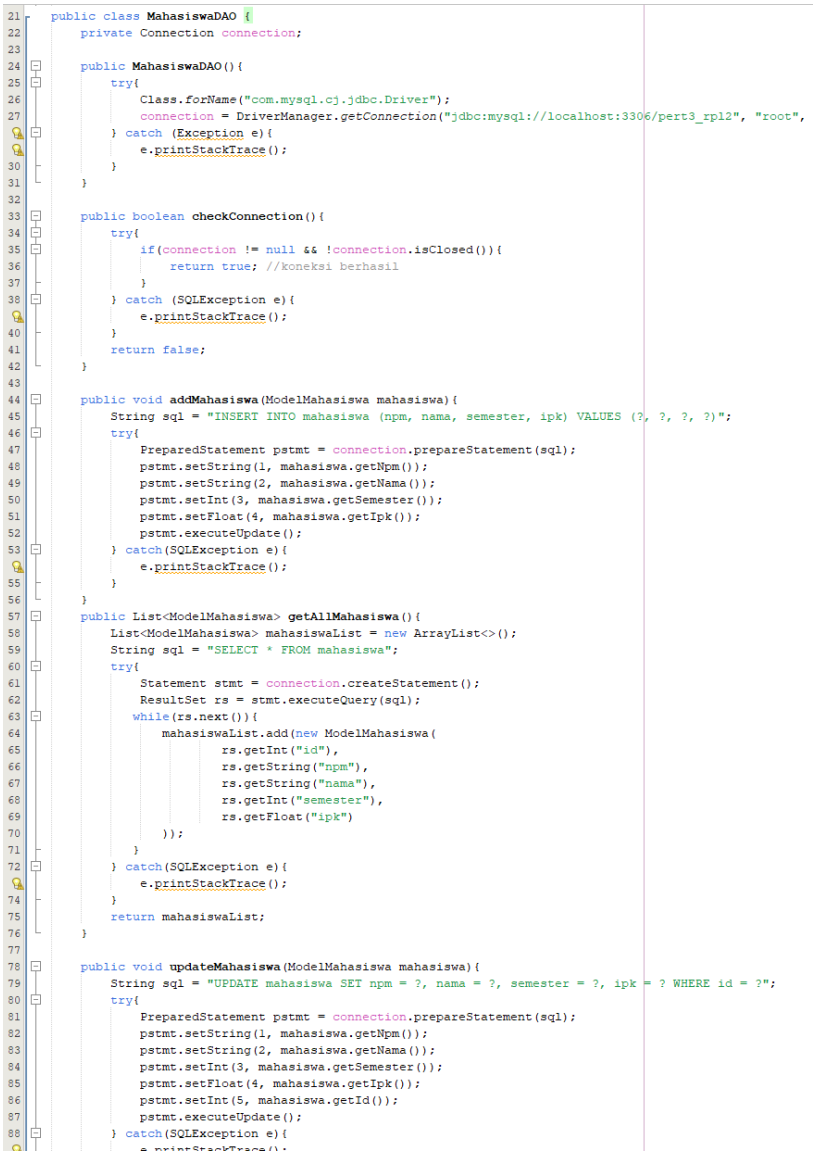
Penjelasan Kode:

ModelMahasiswa.java

```java
public class ModelMahasiswa {

    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;
    }
    private int id;
    private String npm;
    private String nama;
    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;
    }

}
```

Kelas ModelMahasiswa berfungsi sebagai **model data (data entity)** yang merepresentasikan satu objek mahasiswa. Kelas ini berisi lima atribut utama, yaitu id, npm, nama, semester, dan ipk. Masing-masing atribut memiliki **getter dan setter** agar data dapat diakses dan diubah secara aman dari luar kelas. Konstruktor disediakan untuk memudahkan pembuatan objek mahasiswa baru dengan langsung mengisi semua atribut. Kelas ini tidak berhubungan langsung dengan database, melainkan hanya sebagai **wadah (struktur)** yang digunakan oleh kelas lain seperti MahasiswaDAO dan MahasiswaController untuk menyimpan dan memanipulasi data mahasiswa. Dengan adanya model ini, data mahasiswa dapat dikelola secara terorganisir sesuai dengan konsep *Object-Oriented Programming (OOP)*.

MahasiswaDAO.java



```
public class MahasiswaDAO {
    private Connection connection;

    public MahasiswaDAO(){
        try{
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/pert3_rpl2", "root",
        } catch (Exception e){
            e.printStackTrace();
        }
    }

    public boolean checkConnection(){
        try{
            if(connection != null && !connection.isClosed()){
                return true; //koneksi berhasil
            }
        } catch (SQLException e){
            e.printStackTrace();
        }
        return false;
    }

    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();
        }
    }
    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;
    }

    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){
```

Kelas MahasiswaDAO bertanggung jawab sebagai **lapisan akses data (Data Access Object / DAO)** yang mengatur semua interaksi dengan database MySQL. Di dalam kelas ini, terdapat pengaturan koneksi database menggunakan DriverManager, serta berbagai metode seperti addMahasiswa(), getAllMahasiswa(), updateMahasiswa(), dan deleteMahasiswa() untuk melakukan operasi CRUD (*Create*, *Read*, *Update*, *Delete*). Metode checkConnection() digunakan untuk memastikan bahwa koneksi ke database berjalan dengan baik, sedangkan closeConnection() menutup koneksi agar sumber daya tidak terbuang. Setiap operasi database menggunakan **PreparedStatement** agar query lebih aman dan efisien. Dengan adanya kelas ini, logika pengelolaan data terpisah dari tampilan dan logika utama program, sehingga program lebih terstruktur dan mudah dipelihara.

MahasiswaController.java

```java
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());
                System.out.println("============================");
            }
        }
    }


    public void displayMessage(String message){
        System.out.println(message);
    }



    public void checkDatabaseConnection(){
        boolean isConnected = mahasiswaDAO.checkConnection();
        if (isConnected){
            displayMessage("Koneksi ke db berhasil");
        } else{
            displayMessage("Koneksi DB Gagal");
        }
    }

    // READ ALL (Menampilkan semua mahasiswa)
    public void displayAllMahasiswa(){
        List<ModelMahasiswa> mahasiswaList = mahasiswaDAO.getAllMahasiswa();
        displayMahasiswaList(mahasiswaList);
    }

    public void addMahasiswa(String npm, String nama, int semester, float ipk){
        ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(0, npm, nama, semester, ipk);
        System.out.println("Controller Data:   " + npm + nama + semester + ipk);
        System.out.println(mahasiswaBaru);
        mahasiswaDAO.addMahasiswa(mahasiswaBaru);
        displayMessage("Mahasiswa berhasil ditambahkan!");
    }

    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 diperbarui!");
    }

    public void deleteMahasiswa(int id){
        mahasiswaDAO.deleteMahasiswa(id);
        displayMessage("Mahasiswa Berhasil Dihapus!");
    }

    public void closeConnection() {
        mahasiswaDAO.closeConnection();
    }

}
```

Kelas MahasiswaController berperan sebagai penghubung antara tampilan (View) dan lapisan data (Model/DAO). Kelas ini berisi logika bisnis utama yang menentukan bagaimana data dari pengguna dikirim ke database dan bagaimana hasilnya ditampilkan kembali ke pengguna. Misalnya, metode addMahasiswa() menerima data dari pengguna, membuat objek ModelMahasiswa, lalu mengirimkannya ke MahasiswaDAO untuk disimpan ke database. Metode lain seperti updateMahasiswa(),

deleteMahasiswa(), dan displayAllMahasiswa() digunakan untuk memperbarui, menghapus, serta menampilkan data mahasiswa yang tersimpan. Selain itu, checkDatabaseConnection() digunakan untuk menguji apakah koneksi ke database berhasil dilakukan. Dengan adanya controller ini, program memiliki alur kerja yang rapi karena setiap lapisan menjalankan tugas spesifik tanpa saling bercampur.

MahasiswaView.java

```
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("Menu:");
            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:
                    // tambah mhs
                    System.out.println("Masukkan NPM: ");
                    String npm = scanner.next();
                    System.out.println("Masukkan Nama: ");
                    String nama = scanner.next();
                    System.out.println("Masukkan Semester: ");
                    int semester = scanner.nextInt();
                    System.out.println("Masukkan IPK: ");
                    float ipk = scanner.nextFloat();
                    System.out.println(npm + nama + semester + ipk);

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

                case 3:
                    System.out.print("Masukkan ID mahasiswa: ");
                    int id = scanner.nextInt();
                    scanner.nextLine();

                    System.out.println("Masukkan NPM: ");
                    String npmBaru = scanner.next();
                    System.out.println("Masukkan Nama: ");
                    String namaBaru = scanner.next();
                    System.out.println("Masukkan Semester: ");
                    int semesterBaru = scanner.nextInt();
                    System.out.println("Masukkan IPK: ");
                    float ipkBaru = scanner.nextFloat();

                    mahasiswaController.updateMahasiswa(id, npmBaru, namaBaru, semesterBaru, ipkBaru);
                    break;
                case 4:
                    System.out.print("Masukkan ID Mahasiswa: ");
                    int idHapus = scanner.nextInt();
                    mahasiswaController.deleteMahasiswa(idHapus);
                case 5:
                    mahasiswaController.checkDatabaseConnection();
                    break;
                case 6:
                    // Keluar
                    mahasiswaController.closeConnection();
                    System.out.println("Program selesai.");
                    return;
                default:
```

Kelas MahasiswaView berfungsi sebagai **antarmuka pengguna (User Interface)** berbasis konsol yang menampilkan menu dan menerima input dari pengguna. Kelas ini menampilkan menu pilihan seperti menampilkan seluruh data mahasiswa, menambah, memperbarui, menghapus data, mengecek koneksi database, dan keluar dari program. Input pengguna dibaca menggunakan objek Scanner, kemudian diteruskan ke

MahasiswaController untuk diproses. Dengan cara ini, MahasiswaView hanya bertugas mengatur interaksi pengguna, tanpa terlibat langsung dalam proses logika bisnis atau manipulasi database. Hal ini mencerminkan konsep *separation of concerns* dalam pola desain MVC, di mana tampilan, logika, dan data dikelola secara terpisah namun saling terhubung.

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" xsi:schemaLocatio
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany</groupId>
    <artifactId>rpl2_pert3</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.release>24</maven.compiler.release>
        <exec.mainClass>com.mycompany.rpl2_pert3.Rpl2_pert3</exec.mainClass>
    </properties>
    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.33</version>
        </dependency>
    </dependencies>
</project>
```

File pom.xml adalah **berkas konfigurasi Maven** yang mengatur dependensi dan pengaturan build proyek. Di dalamnya terdapat identitas proyek seperti groupId, artifactId, dan version. File ini juga menentukan versi Java yang digunakan serta dependensi eksternal yang dibutuhkan, yaitu mysql-connector-java versi 8.0.33 yang digunakan untuk menghubungkan Java dengan database MySQL melalui JDBC. Dengan adanya pom.xml, NetBeans atau Maven secara otomatis akan mengunduh library yang diperlukan sehingga programmer tidak perlu menambahkannya secara manual. Selain itu, file ini juga memastikan proyek bisa dijalankan dengan lingkungan build yang konsisten di berbagai komputer.

Kesimpulan

Secara keseluruhan, kelima kode tersebut membentuk sebuah aplikasi CRUD sederhana berbasis konsol dengan pola arsitektur **Model–View–Controller (MVC)**.

- **Model** (ModelMahasiswa dan MahasiswaDAO) bertanggung jawab terhadap struktur data dan pengelolaan database.
- **Controller** (MahasiswaController) berperan sebagai pengatur alur logika antara pengguna dan data.
- **View** (MahasiswaView) menjadi media interaksi antara pengguna dan sistem.

- **pom.xml** mendukung proyek dengan mengelola dependensi otomatis melalui Maven.

Dengan pembagian struktur seperti ini, aplikasi menjadi lebih mudah dikembangkan, diuji, dan dipelihara karena setiap komponen memiliki tanggung jawab yang jelas dan terpisah.