

ACTIVITY PERTEMUAN 2

NAMA : FX Edrianto Putra Wijaya

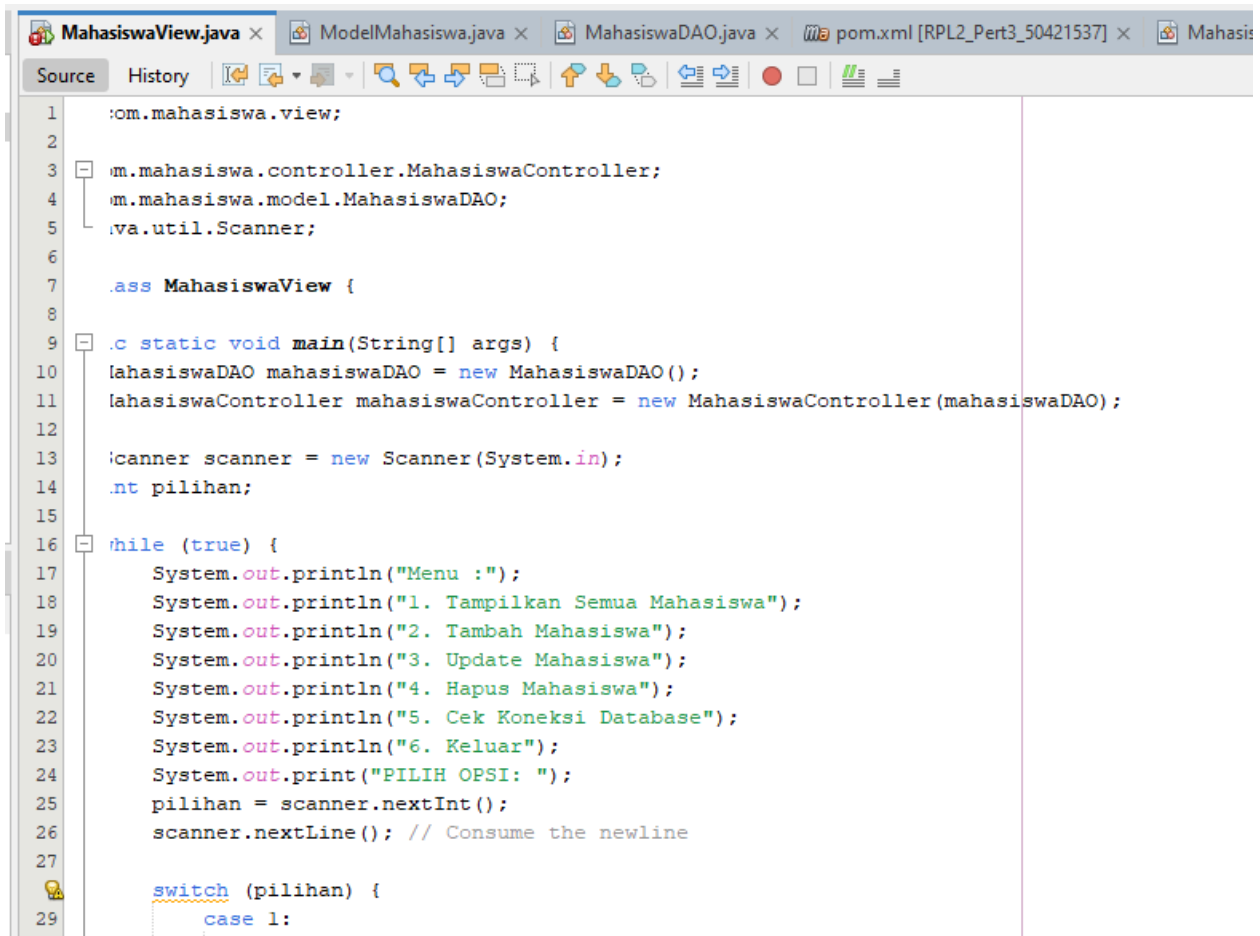
NPM : 50421537

KELAS : 4IA13

MATERI : Konsep MVC

MATA PRAKTIKUM : Rekayasa Perangkat Lunak 2

(Screenshoot langkah-langkah sesuai video pembelajaran dan jelaskan dengan ringkas)



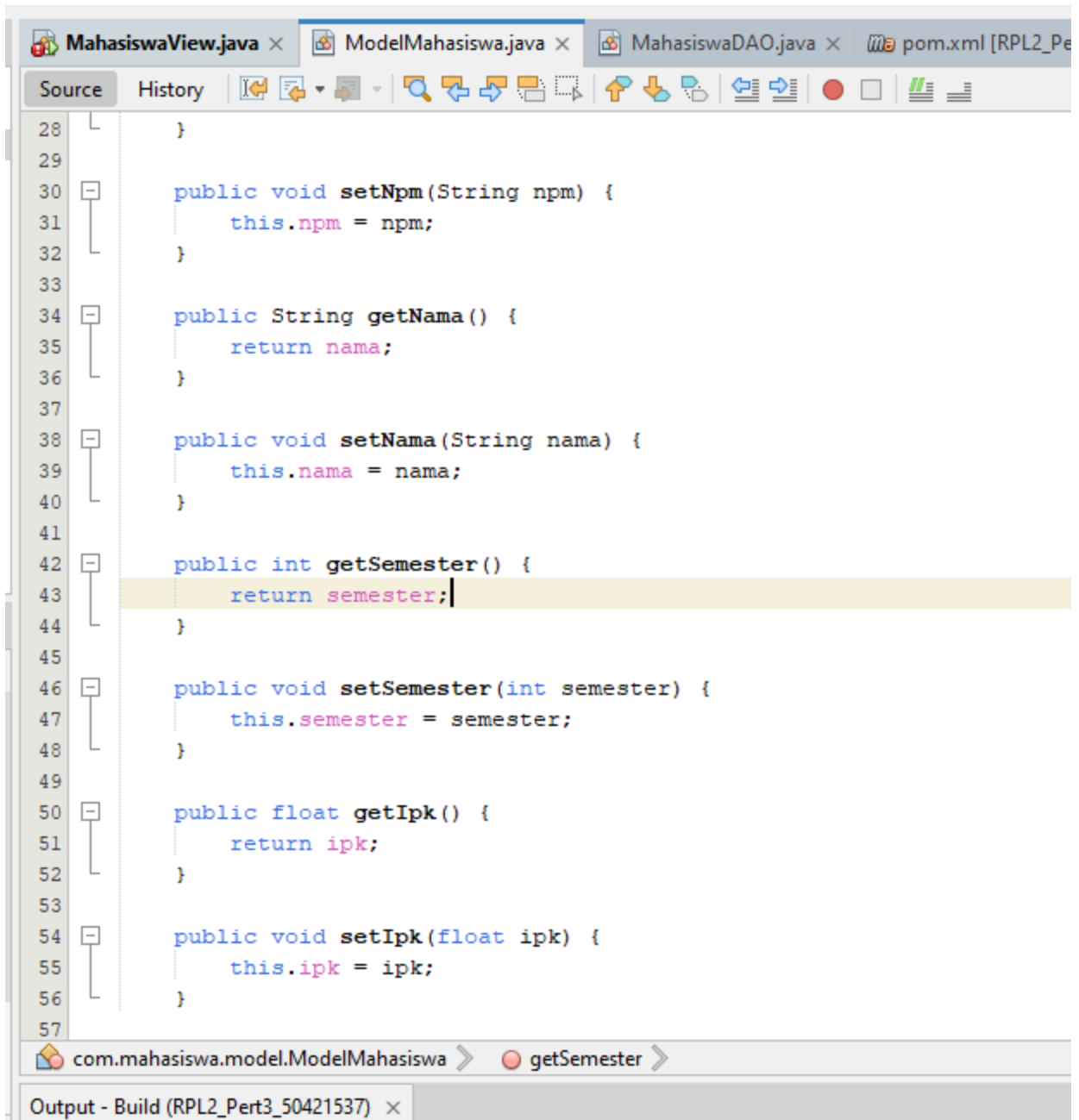
```
1  :om.mahasiswa.view;
2
3  import om.mahasiswa.controller.MahasiswaController;
4  import om.mahasiswa.model.MahasiswaDAO;
5  import java.util.Scanner;
6
7  class MahasiswaView {
8
9      static void main(String[] args) {
10         MahasiswaDAO mahasiswaDAO = new MahasiswaDAO();
11         MahasiswaController mahasiswaController = new MahasiswaController(mahasiswaDAO);
12
13         Scanner scanner = new Scanner(System.in);
14         int pilihan;
15
16         while (true) {
17             System.out.println("Menu :");
18             System.out.println("1. Tampilkan Semua Mahasiswa");
19             System.out.println("2. Tambah Mahasiswa");
20             System.out.println("3. Update Mahasiswa");
21             System.out.println("4. Hapus Mahasiswa");
22             System.out.println("5. Cek Koneksi Database");
23             System.out.println("6. Keluar");
24             System.out.print("PILIH OPSI: ");
25             pilihan = scanner.nextInt();
26             scanner.nextLine(); // Consume the newline
27
28             switch (pilihan) {
29                 case 1:
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java
Source History
31         break;
32     case 2:
33         System.out.print("Masukan NPM: ");
34         String npm = scanner.nextLine(); // Use nextLine for names with spaces
35         System.out.print("Masukan Nama: ");
36         String nama = scanner.nextLine(); // Use nextLine for names with spaces
37         System.out.print("Masukan Semester: ");
38         int semester = scanner.nextInt();
39         System.out.print("Masukan IPK: ");
40         float ipk = scanner.nextFloat();
41         mahasiswaController.addMahasiswa(npm, nama, semester, ipk);
42         break;
43     case 3:
44         System.out.print("Masukan ID Mahasiswa yang akan di-update: ");
45         int idUpdate = scanner.nextInt();
46         scanner.nextLine(); // Consume the newline
47         System.out.print("Masukan NPM: ");
48         String npmUpdate = scanner.nextLine(); // Use nextLine for names with spaces
49         System.out.print("Masukan Nama: ");
50         String namaUpdate = scanner.nextLine(); // Use nextLine for names with spaces
51         System.out.print("Masukan IPK: ");
52         float ipkUpdate = scanner.nextFloat();
53         mahasiswaController.updateMahasiswa(idUpdate, npmUpdate, namaUpdate, ipkUpdate);
54         break;
55     case 4:
56         System.out.print("Masukan ID Mahasiswa yang akan dihapus: ");
57         int idDelete = scanner.nextInt();
58         mahasiswaController.deleteMahasiswa(idDelete);
59         break;

com.mahasiswa.view.MahasiswaView > main > while (true) > switch (pilihan) > case 2: >
Output - Build (RPL2_Pert3_50421537) x
>> | ----- [ jar ] -----
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source History
46 scanner.nextLine(); // Consume the newline
47 System.out.print("Masukan NPM: ");
48 String npmUpdate = scanner.nextLine(); // Use nextLine for names with spaces
49 System.out.print("Masukan Nama: ");
50 String namaUpdate = scanner.nextLine(); // Use nextLine for names with spaces
51 System.out.print("Masukan IPK: ");
52 float ipkUpdate = scanner.nextFloat();
53 mahasiswaController.updateMahasiswa(idUpdate, npmUpdate, namaUpdate, ipkUpdate);
54 break;
55 case 4:
56 System.out.print("Masukan ID Mahasiswa yang akan dihapus: ");
57 int idDelete = scanner.nextInt();
58 mahasiswaController.deleteMahasiswa(idDelete);
59 break;
60 case 5:
61 mahasiswaController.checkDatabaseConnection();
62 break;
63 case 6:
64 System.out.println("Keluar dari program...");
65 mahasiswaController.closeConnection();
66 scanner.close();
67 return;
68 default:
69 System.out.println("Ops! tidak valid!");
70 }
71
72
73
74
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source History
1 package com.mahasiswa.model;
2
3 public class ModelMahasiswa {
4     private int id;
5     private String npm;
6     private String nama;
7     private int semester;
8     private float ipk;
9
10    public ModelMahasiswa(int id, String npm, String nama, int semester, float ipk) {
11        this.id = id;
12        this.npm = npm;
13        this.nama = nama;
14        this.semester = semester;
15        this.ipk = ipk;
16    }
17
18    public int getId() {
19        return id;
20    }
21
22    public void setId(int id) {
23        this.id = id;
24    }
25
26    public String getNpm() {
27        return npm;
28    }
29
30    public void setNpm(String npm) {
31
32    }
33
34    }
35
36    }
37
38    }
39
40    }
41
42    }
43
44    }
45
46    }
47
48    }
49
50    }
51
52    }
53
54    }
55
56    }
57
58    }
59
60    }
61
62    }
63
64    }
65
66    }
67
68    }
69
70    }
71
72    }
73
74    }
75
76    }
77
78    }
79
80    }
81
82    }
83
84    }
85
86    }
87
88    }
89
90    }
91
92    }
93
94    }
95
96    }
97
98    }
99
100   }
```

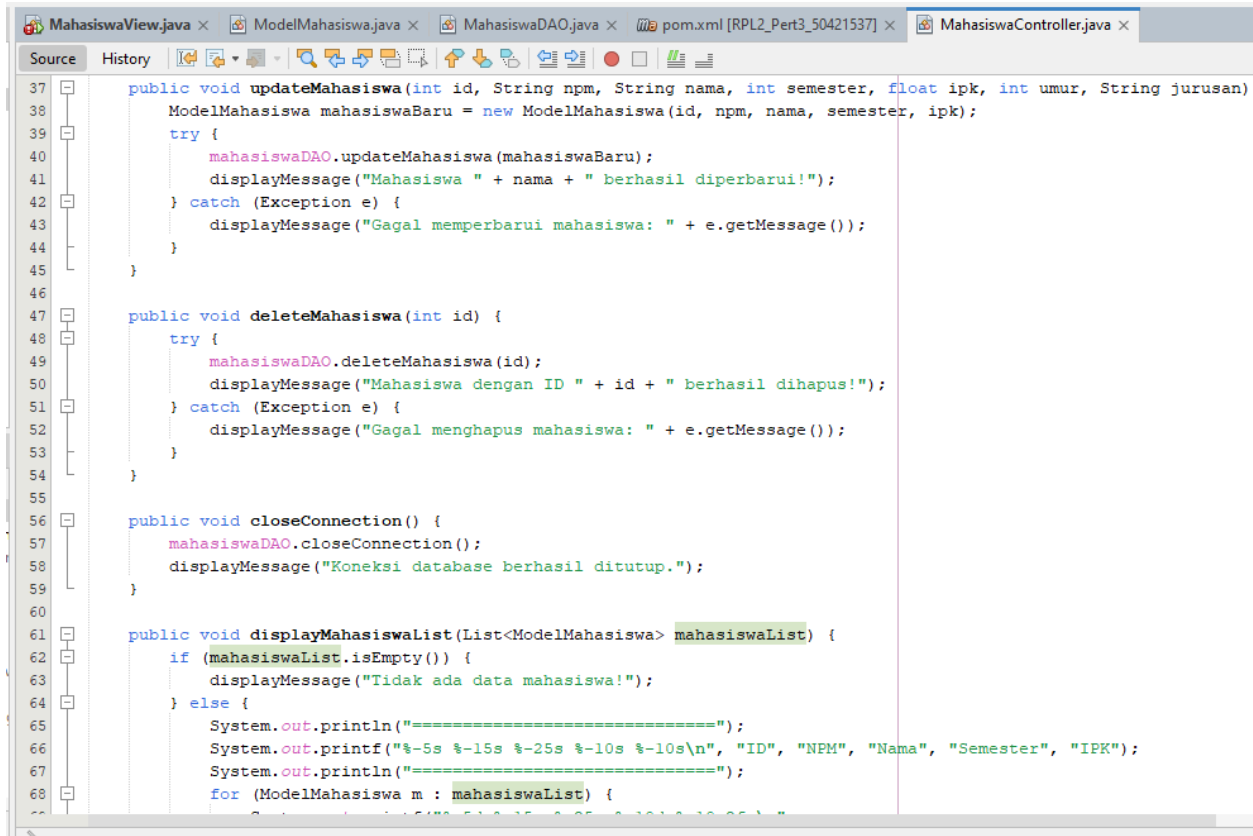


```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source History
1 package com.mahasiswa.model;
2
3 import java.sql.*;
4 import java.util.ArrayList;
5 import java.util.List;
6
7 public class MahasiswaDAO {
8     private Connection connection;
9
10    public MahasiswaDAO() {
11        try {
12            Class.forName("com.mysql.cj.jdbc.Driver");
13            connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/mvc_db", "root", "");
14        } catch (ClassNotFoundException | SQLException e) {
15            e.printStackTrace();
16        }
17    }
18
19    public boolean cekKoneksi() {
20        try {
21            return connection != null && !connection.isClosed();
22        } catch (SQLException e) {
23            e.printStackTrace();
24        }
25        return false;
26    }
27
28    public void addMahasiswa(ModelMahasiswa mahasiswa) {
29        String sql = "INSERT INTO mahasiswa (npm, nama, semester, ipk, umur, jurusan) VALUES (?, ?, ?, ?, ?, ?)";
30        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
31            com.mahasiswa.model.MahasiswaDAO > updateMahasiswa > try >
Output - Build (RPL2_Pert3_50421537) x
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source History
70 pstmt.setInt(7, mahasiswa.getId());
71 pstmt.executeUpdate();
72 System.out.println("Mahasiswa " + mahasiswa.getNama() + " berhasil diperbarui.");
73 } catch (SQLException e) {
74     e.printStackTrace();
75 }
76 }
77
78 public void deleteMahasiswa(int id) {
79     String sql = "DELETE FROM mahasiswa WHERE id = ?";
80     try (PreparedStatement pstmt = connection.prepareStatement(sql)) {
81         pstmt.setInt(1, id);
82         pstmt.executeUpdate();
83         System.out.println("Mahasiswa dengan ID " + id + " berhasil dihapus.");
84     } catch (SQLException e) {
85         e.printStackTrace();
86     }
87 }
88
89 public void closeConnection() {
90     try {
91         if (connection != null) {
92             connection.close();
93         }
94     } catch (SQLException e) {
95         e.printStackTrace();
96     }
97 }
98 }
99
com.mahasiswa.model.MahasiswaDAO > updateMahasiswa > try >
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source Graph Effective History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 <modelVersion>4.0.0</modelVersion>
4 <groupId>com.mahasiswa</groupId>
5 <artifactId>RPL2_Pert3_50421537</artifactId>
6 <version>1.0-SNAPSHOT</version>
7 <packaging>jar</packaging>
8 <properties>
9 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10 <maven.compiler.release>17</maven.compiler.release>
11 <exec.mainClass>com.mahasiswa.view.MahasiswaView</exec.mainClass>
12 </properties>
13
14 <dependencies>
15 <dependency>
16 <groupId>mysql</groupId>
17 <artifactId>mysql-connector-java</artifactId>
18 <version>8.0.33</version>
19 </dependency>
20 </dependencies>
21 </project>
```

```
MahasiswaView.java x ModelMahasiswa.java x MahasiswaDAO.java x pom.xml [RPL2_Pert3_50421537] x MahasiswaController.java x
Source History
1 package com.mahasiswa.controller;
2
3 import com.mahasiswa.model.MahasiswaDAO;
4 import com.mahasiswa.model.ModelMahasiswa;
5 import java.util.List;
6
7 public class MahasiswaController {
8     private final MahasiswaDAO mahasiswaDAO;
9
10     public MahasiswaController(MahasiswaDAO mahasiswaDAO) {
11         this.mahasiswaDAO = mahasiswaDAO;
12     }
13
14     public void checkDatabaseConnection() {
15         if (mahasiswaDAO.cekKoneksi()) {
16             displayMessage("Koneksi ke DB berhasil!");
17         } else {
18             displayMessage("Koneksi DB gagal!");
19         }
20     }
21
22     public void displayAllMahasiswa() {
23         List<ModelMahasiswa> mahasiswaList = mahasiswaDAO.getAllMahasiswa();
24         displayMahasiswaList(mahasiswaList);
25     }
26
27     public void addMahasiswa(String npm, String nama, int semester, float ipk, int umur, String jurusan) {
28         ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(0, npm, nama, semester, ipk);
29         try {
30             mahasiswaDAO.addMahasiswa(mahasiswaBaru);
31             displayMessage("Mahasiswa " + nama + " berhasil ditambahkan!");
32         } catch (Exception e) {
33             displayMessage("Gagal menambahkan mahasiswa: " + e.getMessage());
34         }
35     }
36 }
```



```
37 public void updateMahasiswa(int id, String npm, String nama, int semester, float ipk, int umur, String jurusan)
38 ModelMahasiswa mahasiswaBaru = new ModelMahasiswa(id, npm, nama, semester, ipk);
39 try {
40     mahasiswaDAO.updateMahasiswa(mahasiswaBaru);
41     displayMessage("Mahasiswa " + nama + " berhasil diperbarui!");
42 } catch (Exception e) {
43     displayMessage("Gagal memperbarui mahasiswa: " + e.getMessage());
44 }
45 }
46
47 public void deleteMahasiswa(int id) {
48     try {
49         mahasiswaDAO.deleteMahasiswa(id);
50         displayMessage("Mahasiswa dengan ID " + id + " berhasil dihapus!");
51     } catch (Exception e) {
52         displayMessage("Gagal menghapus mahasiswa: " + e.getMessage());
53     }
54 }
55
56 public void closeConnection() {
57     mahasiswaDAO.closeConnection();
58     displayMessage("Koneksi database berhasil ditutup.");
59 }
60
61 public void displayMahasiswaList(List<ModelMahasiswa> mahasiswaList) {
62     if (mahasiswaList.isEmpty()) {
63         displayMessage("Tidak ada data mahasiswa!");
64     } else {
65         System.out.println("=====");
66         System.out.printf("%-5s %-15s %-25s %-10s %-10s\n", "ID", "NPM", "Nama", "Semester", "IPK");
67         System.out.println("=====");
68         for (ModelMahasiswa m : mahasiswaList) {
69             System.out.printf("%-5s %-15s %-25s %-10s %-10s\n", m.getId(), m.getNpm(), m.getNama(), m.getSemester(), m.getIpK());
70         }
71     }
72 }
```

Jelaskan cara membuat mvc_db

Membuat aplikasi dengan struktur Model-View-Controller dan database (MVC_DB) menggunakan XAMPP melibatkan beberapa langkah. Berikut adalah cara dasar untuk melakukannya:

1. Siapkan XAMPP dan Database

- Instalasi XAMPP: Unduh dan instal XAMPP untuk menyediakan server lokal, MySQL, dan PHP.
- Buat Database:
 1. Buka phpMyAdmin dari dashboard XAMPP (<http://localhost/phpmyadmin>).
 2. Buat database baru, misalnya dengan nama mvc_db.
 3. Buat tabel yang akan digunakan dalam aplikasi (misalnya tabel users dengan kolom id, name, email, dll.).

2. Buat Struktur Direktori

- Buat folder proyek Anda di htdocs (misalnya htdocs/mvc_db).
- Di dalam folder proyek, buat tiga folder utama: models, views, dan controllers untuk mencerminkan struktur MVC.

3. Buat File index.php untuk Routing

File index.php akan menjadi pintu masuk utama aplikasi. Di dalam file ini, Anda akan menentukan kontroler mana yang akan dijalankan sesuai permintaan pengguna.

4. Buat routes.php untuk Routing

File routes.php akan memanggil controller dan method yang sesuai berdasarkan URL.

5. Membuat Model

Model adalah bagian aplikasi yang berinteraksi dengan database. Buat model di dalam folder models, misalnya User.php.

6. Membuat Controller

Controller menangani permintaan dari pengguna dan memilih data yang diperlukan dari model. Simpan di dalam folder controllers, misalnya UsersController.php.

7. Membuat View

View menampilkan data ke pengguna. Simpan di folder views, misalnya views/users/index.php.

8. Menambahkan Koneksi Database

Buat file Db.php di folder models untuk mengatur koneksi database.

9. Uji Jalankan Aplikasi

- Buka browser, dan akses http://localhost/mvc_db/index.php?controller=users&action=index.
- Jika semua langkah benar, Anda akan melihat daftar pengguna dari tabel users yang ditampilkan di halaman.

Dengan struktur MVC ini, Anda memiliki pemisahan yang jelas antara logika bisnis (Model), alur kontrol (Controller), dan tampilan (View).