LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Pengolahan Citra
Kelas	: 4IA01
Praktikum ke-	: 3
Tanggal	: 3 November 2025
Materi	: MVC
NPM	: 50422657
Nama	: Harry Mardika
Ketua Asisten	: Satya Bara Justitia
Paraf Asisten	: -
Nama Asisten	: 1.
	2.
	3.
Jumlah Lembar	: ... Lembar

LABORATORIUM INFORMATIKA

UNIVERSITAS GUNADARMA

2025

Soal

1. Apa yang kalian ketahui tentang MVC?
2. Jelaskan kode dan langkah – langkah program yang telah dibuat!

Penjelasan

---

1. MVC (Model-View-Controller) adalah sebuah pola arsitektur perangkat lunak yang bertujuan untuk memisahkan antara logika bisnis, tampilan antarmuka, dan pengendalian alur data. Pemisahan ini membuat pengembangan aplikasi menjadi lebih terstruktur, mudah dikelola, dan fleksibel untuk dikembangkan di kemudian hari. MVC terdiri dari tiga komponen utama, yaitu Model, View, dan Controller, yang masing-masing memiliki peran dan tanggung jawab berbeda namun saling berinteraksi untuk menjalankan fungsi aplikasi secara keseluruhan.

Komponen Model berperan dalam mengelola data dan logika bisnis aplikasi. Model bertanggung jawab untuk mengambil, menyimpan, dan memproses data, baik dari basis data maupun sumber lainnya. Model tidak memiliki hubungan langsung dengan tampilan, sehingga perubahan pada data tidak secara otomatis memengaruhi antarmuka pengguna. Selanjutnya, View merupakan bagian yang menampilkan informasi kepada pengguna dalam bentuk yang mudah dimengerti, seperti teks, grafik, tabel, atau halaman web. View hanya menampilkan data yang diberikan oleh Model melalui Controller, tanpa melakukan pemrosesan logika bisnis. Komponen terakhir adalah Controller, yang berfungsi sebagai penghubung antara Model dan View. Controller menerima input dari pengguna (misalnya klik tombol atau masukan data), memprosesnya sesuai dengan logika tertentu, kemudian menentukan bagaimana data dari Model akan ditampilkan kembali melalui View. Dengan kata lain, Controller mengatur aliran data dan interaksi pengguna terhadap aplikasi.

Dengan penerapan pola MVC, pengembang dapat bekerja secara paralel pada bagian yang berbeda desainer UI fokus pada View, pengembang logika fokus pada Model, dan pengembang interaksi fokus pada Controller. Pola ini juga mempermudah proses pemeliharaan dan pengujian, karena setiap komponen dapat diperbarui atau diuji secara independen tanpa memengaruhi bagian lain. Oleh karena itu, MVC banyak digunakan dalam pengembangan aplikasi web dan desktop modern, seperti Django (Python), Laravel (PHP), Ruby on Rails, dan ASP.NET MVC, karena kemampuannya menjaga modularitas dan efisiensi dalam siklus pengembangan perangkat lunak.

2.

- Overview

Program pada ACT merupakan aplikasi berbasis console Java yang dirancang untuk mengelola data mahasiswa menggunakan database MySQL. Aplikasi ini mengimplementasikan pola arsitektur Model-View-Controller (MVC) yang memisahkan logika bisnis, representasi data, dan antarmuka pengguna secara terstruktur. Program menyediakan fitur lengkap untuk operasi CRUD (Create, Read, Update, Delete) terhadap data mahasiswa yang disimpan dalam database relasional. Struktur aplikasi terdiri dari empat komponen utama: ModelMahasiswa sebagai entitas data, MahasiswaDAO sebagai lapisan akses data, MahasiswaController sebagai pengendali logika bisnis, dan MahasiswaView sebagai antarmuka pengguna. Setiap komponen memiliki tanggung jawab spesifik yang mendukung prinsip separation of concerns dalam pengembangan perangkat lunak.

- Arsitektur MVC
  - Model Layer - Representasi Data

    ModelMahasiswa.java berfungsi sebagai blueprint atau cetakan untuk objek mahasiswa yang akan digunakan dalam aplikasi. Kelas ini mengimplementasikan konsep encapsulation dengan menyembunyikan atribut data menggunakan modifier private dan menyediakan akses melalui method getter dan setter. Atribut yang didefinisikan meliputi id sebagai primary key, npm untuk nomor pokok mahasiswa, nama untuk identitas mahasiswa, semester untuk tingkat akademik, dan ipk untuk indeks prestasi kumulatif.

    Constructor pada kelas ini memungkinkan inisialisasi objek mahasiswa dengan semua parameter yang diperlukan, memastikan bahwa setiap objek yang dibuat memiliki data lengkap. Method getter memungkinkan pengambilan nilai atribut dari luar kelas, sementara method setter memungkinkan perubahan nilai atribut dengan validasi yang dapat ditambahkan di masa depan. Pendekatan ini memberikan fleksibilitas dalam pengelolaan data sambil menjaga integritas struktur objek. MahasiswaDAO.java berperan sebagai Data Access Object yang bertanggung jawab untuk semua operasi database. Kelas ini menggunakan koneksi JDBC untuk berkomunikasi dengan database MySQL dan menyediakan abstraksi untuk operasi-operasi database kompleks. Setiap method dalam kelas ini dirancang untuk menangani satu operasi spesifik terhadap tabel mahasiswa, memungkinkan pemisahan yang jelas antara logika bisnis dan logika akses data.

  - Controller Layer - Logika Bisnis

    MahasiswaController.java berperan sebagai mediator antara lapisan view dan model, mengelola alur eksekusi program dan mengkoordinasikan operasi antara komponen-komponen aplikasi. Controller ini menerima input dari layer view, memproses permintaan menggunakan services dari DAO, dan mengembalikan hasil kepada view untuk ditampilkan kepada pengguna. Setiap method dalam controller memiliki tanggung jawab spesifik untuk satu jenis operasi, seperti menampilkan data, menambah data, mengupdate data, atau menghapus data. Controller juga menangani formatting output dan pesan-pesan yang akan ditampilkan kepada pengguna, memastikan bahwa informasi disajikan dengan cara yang konsisten dan mudah dipahami. Implementasi ini memungkinkan perubahan pada tampilan atau logika bisnis tanpa mempengaruhi komponen lainnya, mendukung prinsip loose coupling dalam desain perangkat lunak.

  - View Layer - Antarmuka Pengguna

    MahasiswaView.java berfungsi sebagai antarmuka pengguna yang mengelola interaksi dengan user melalui console. Kelas ini mengimplementasikan menu-driven interface yang memungkinkan pengguna memilih operasi yang ingin dilakukan melalui sistem numbering yang intuitif. Input handling dalam kelas ini dirancang untuk menangani berbagai tipe data dengan robust error handling, termasuk penggunaan scanner.nextLine() untuk membersihkan buffer input dan mencegah InputMismatchException. Menu loop dalam aplikasi dirancang untuk berjalan terus-menerus hingga pengguna memilih opsi keluar, memberikan pengalaman pengguna yang seamless untuk melakukan multiple operations dalam satu session. Setiap opsi menu terhubung langsung dengan method yang sesuai di controller, memastikan bahwa alur data mengikuti pola MVC yang telah ditetapkan.

- Fungsionalitas
  - Create Operation - Menambah Data Mahasiswa

    Operasi penambahan data mahasiswa dimulai dari view yang menerima input pengguna berupa NPM, nama, semester, dan IPK. Data ini kemudian diteruskan ke controller yang melakukan validasi dasar sebelum membuat objek ModelMahasiswa baru. Controller memanggil method addMahasiswa() pada DAO yang menyiapkan prepared statement SQL untuk insert data ke database. Penggunaan prepared statement tidak hanya meningkatkan performance tetapi juga memberikan perlindungan terhadap SQL injection attacks. Proses ini melibatkan konversi data dari format input pengguna ke format yang sesuai dengan struktur database, termasuk handling untuk tipe data yang berbeda seperti string, integer, dan float. Error handling pada level DAO memastikan bahwa jika terjadi kesalahan database, aplikasi tidak crash dan pengguna mendapat feedback yang appropriat tentang status operasi.

  - Read Operation - Menampilkan Data Mahasiswa

    Operasi pembacaan data menggunakan method getAllMahasiswa() yang mengeksekusi query SELECT untuk mengambil semua record dari tabel mahasiswa. DAO mengembalikan List<ModelMahasiswa> yang berisi semua objek mahasiswa yang berhasil diambil dari database. Controller kemudian memformat data ini untuk ditampilkan melalui view dengan format yang user-friendly. Implementasi menggunakan ResultSet untuk iterasi melalui hasil query dan mapping setiap row ke objek ModelMahasiswa. Proses mapping ini memastikan bahwa tipe data database sesuai dengan tipe data Java object, termasuk handling untuk null values dan data validation. Output formatting di controller memastikan bahwa data ditampilkan dengan layout yang konsisten dan mudah dibaca.

  - Update Operation - Mengubah Data Mahasiswa

    Operasi update dimulai dengan input ID mahasiswa yang akan diupdate, diikuti dengan data baru untuk semua field. Controller menyusun objek ModelMahasiswa dengan ID yang sudah ada dan data baru yang diinput pengguna. DAO kemudian mengeksekusi UPDATE query menggunakan WHERE clause berdasarkan ID untuk memastikan hanya record yang tepat yang dimodifikasi. Proses update menggunakan prepared statement dengan parameter binding untuk setiap field yang akan diupdate. Hal ini memastikan bahwa semua data tersimpan dengan format yang benar dan memberikan performance yang optimal. Error handling pada operasi ini mencakup validasi bahwa record dengan ID tersebut benar-benar exists dalam database sebelum melakukan update.

  - Delete Operation - Menghapus Data Mahasiswa

    Operasi penghapusan menggunakan ID mahasiswa sebagai parameter untuk menentukan record mana yang akan dihapus. DAO mengeksekusi DELETE query dengan WHERE clause yang spesifik untuk menghindari accidental deletion of multiple records. Penggunaan prepared statement dengan parameter ID memastikan bahwa operasi penghapusan aman dan tepat sasaran. Controller menangani feedback kepada pengguna tentang status operasi penghapusan, termasuk konfirmasi bahwa data telah berhasil dihapus atau pesan error jika operasi gagal. Implementasi ini tidak menggunakan soft delete, sehingga data yang dihapus benar-benar dihilangkan dari database secara permanen.

- Manajemen Database dan Koneksi
  - Koneksi Database MySQL

    Aplikasi menggunakan MySQL Connector/J sebagai JDBC driver untuk establishing connection ke database MySQL. Koneksi diinisialisasi dalam constructor MahasiswaDAO menggunakan DriverManager.getConnection() dengan parameter URL database, username, dan password. URL koneksi menggunakan format jdbc:mysql://localhost:3306/pert3_50422657 yang menunjukkan koneksi ke local MySQL server pada port default dengan database specific untuk praktikum ini. Class loading untuk MySQL driver dilakukan menggunakan Class.forName("com.mysql.cj.jdbc.Driver") yang memastikan bahwa driver tersedia sebelum attempt connection dibuat. Error handling pada proses koneksi mencakup ClassNotFoundException untuk missing driver dan SQLException untuk connection issues, memberikan diagnostic information yang helpful untuk troubleshooting.

  - Connection Management dan Validation

    Method checkConnection() mengimplementasikan validation logic untuk memastikan bahwa koneksi database masih active dan usable. Validation ini tidak hanya memeriksa apakah connection object tidak null dan tidak closed, tetapi juga melakukan test query untuk memverifikasi bahwa koneksi benar-benar functional. Jika koneksi bermasalah, method ini mencoba untuk re-establish connection secara otomatis. Proper connection closing diimplementasikan dalam method closeConnection() untuk memastikan bahwa database resources dilepaskan dengan benar ketika aplikasi selesai digunakan. Hal ini penting untuk mencegah connection leaks yang dapat menyebabkan database server kehabisan available connections ketika aplikasi dijalankan berulang kali.

- Álur Eksekusi
  - Program Initialization

    Ketika aplikasi dimulai, main method dalam MahasiswaView melakukan inisialisasi komponen-komponen MVC secara berurutan. Pertama, MahasiswaDAO object dibuat yang secara otomatis menginisialisasi koneksi database. Kemudian, MahasiswaController dibuat dengan dependency injection DAO object. Scanner object untuk input handling juga diinisialisasi pada tahap ini. Menu loop kemudian dimulai dengan menampilkan opsi-opsi yang tersedia kepada pengguna. Loop ini berjalan indefinitely hingga pengguna memilih opsi exit, memungkinkan multiple operations dalam single session. State management dalam loop memastikan bahwa setiap iterasi dimulai dengan clean state dan ready untuk operation berikutnya.

  - Request-Response Flow

    Setiap user interaction mengikuti pattern yang konsisten: input dari user di-capture oleh view, di-validate dan di-process oleh controller, data operation dilakukan oleh DAO, dan result di-format dan ditampilkan kembali kepada user melalui view. Flow ini memastikan bahwa setiap layer bertanggung jawab hanya untuk domain-specific logic dan tidak melakukan cross-cutting concerns. Data transformation terjadi pada setiap level sesuai dengan kebutuhan: raw string input dari user dikonversi ke appropriate Java types, Java objects di-serialize ke SQL parameters untuk database operations, dan database results di-

mapped kembali ke Java objects untuk business logic processing. Setiap transformation dilengkapi dengan validation untuk memastikan data integrity sepanjang alur eksekusi.