**LAPORAN AKHIR PRAKTIKUM**

Mata Praktikum : RPL 2

Kelas : 4IA06

Praktikum ke- : 3

Tanggal : 29 October 2024

Materi : Konsep MVC

NPM : 50421535

Nama : Fizri Rosdiansyah

Ketua Asisten : Gilbert Jefferson Faozato Mendrofa

Paraf Asisten :

Nama Asisten :

Jumlah Lembar : 10 Lembar

LABORATORIUM TEKNIK INFORMATIKA UNIVERSITAS GUNADARMA

2024

Rekayasa Perangkat Lunak 2

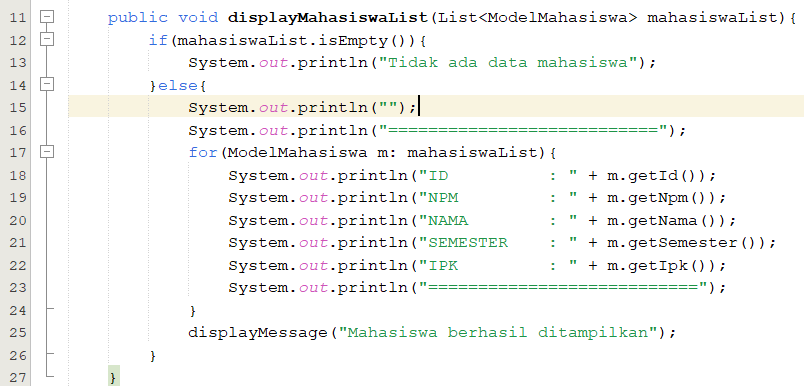
Soal

1. Jelaskan satu per satu Codingan kalian dari hasil screenshot Activity

Jawab

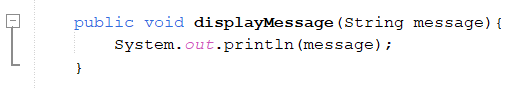
1.

A.Mahasiswa Controller

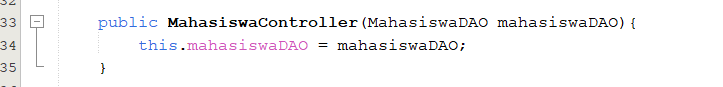


Metode ini menampilkan daftar mahasiswa:

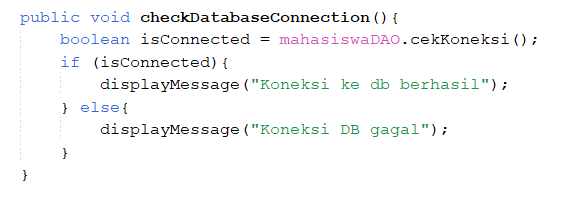
* Jika mahasiswaList kosong, ditampilkan pesan "Tidak ada data mahasiswa".
* Jika ada data, iterasi dilakukan untuk menampilkan detail tiap mahasiswa (ID, NPM, Nama, Semester, IPK).
* Menampilkan pesan "Mahasiswa berhasil ditampilkan" setelah iterasi selesai.



Metode ini hanya menampilkan pesan yang diteruskan sebagai parameter ke konsol.

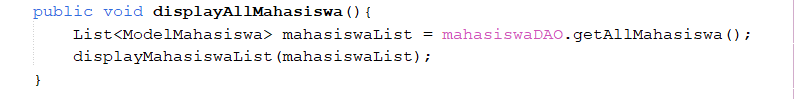


Konstruktor ini digunakan untuk menginisialisasi MahasiswaController dengan objek MahasiswaDAO, yang memungkinkan interaksi dengan database.



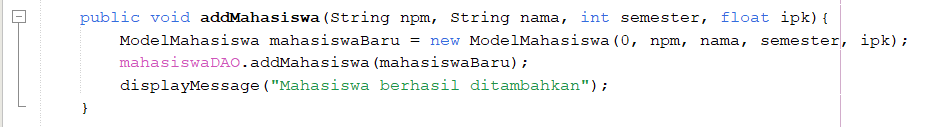
Metode ini mengecek koneksi ke database:

* Memanggil metode cekKoneksi() dari mahasiswaDAO untuk mendapatkan status koneksi.
* Menampilkan pesan "Koneksi ke db berhasil" jika terkoneksi, atau "Koneksi DB gagal" jika tidak.



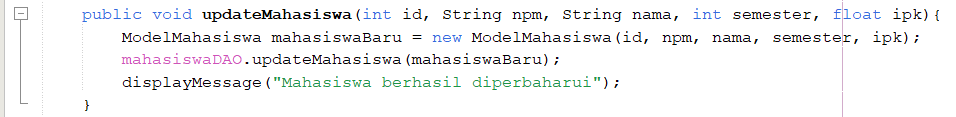
Metode ini menampilkan seluruh data mahasiswa:

* Mengambil semua data mahasiswa dengan getAllMahasiswa dari mahasiswaDAO.
* Menampilkan data tersebut menggunakan displayMahasiswaList.



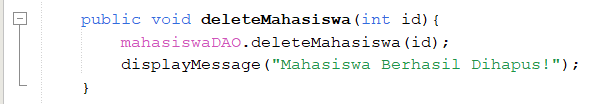
Metode ini menambahkan mahasiswa baru ke database:

* Membuat objek ModelMahasiswa baru dengan parameter NPM, Nama, Semester, dan IPK.
* Menyimpan data baru ini ke database menggunakan addMahasiswa dari mahasiswaDAO.
* Menampilkan pesan "Mahasiswa berhasil ditambahkan" di konsol.



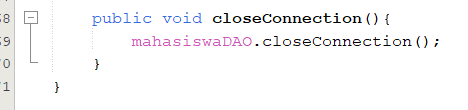
Metode ini memperbarui data mahasiswa yang ada berdasarkan ID:

* Membuat objek ModelMahasiswa dengan ID dan data baru.
* Memperbarui data mahasiswa di database menggunakan updateMahasiswa dari mahasiswaDAO.
* Menampilkan pesan "Mahasiswa berhasil diperbaharui" setelah pembaruan berhasil.



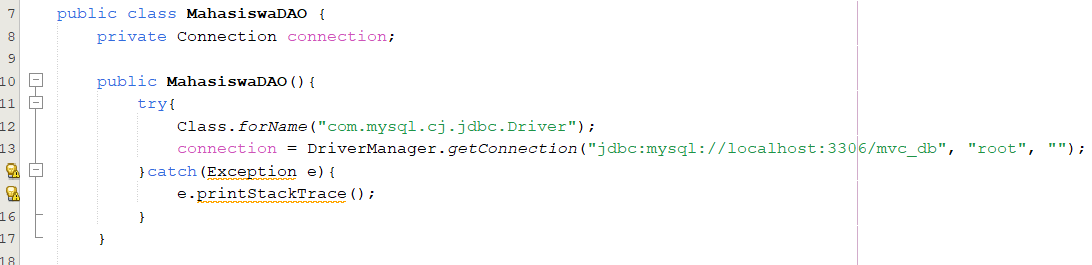
Metode ini menghapus data mahasiswa berdasarkan ID:

* Memanggil deleteMahasiswa dari mahasiswaDAO untuk menghapus data dari database.
* Menampilkan pesan "Mahasiswa Berhasil Dihapus!" setelah penghapusan.



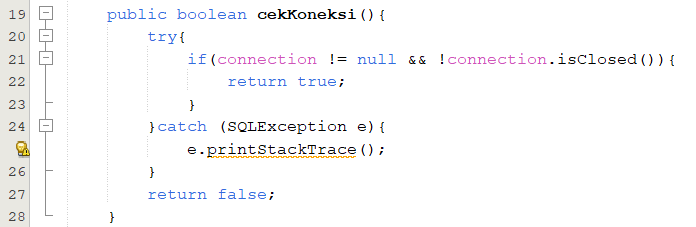
Metode ini menutup koneksi database dengan memanggil closeConnection pada mahasiswaDAO.

B.MahasiswaDAO



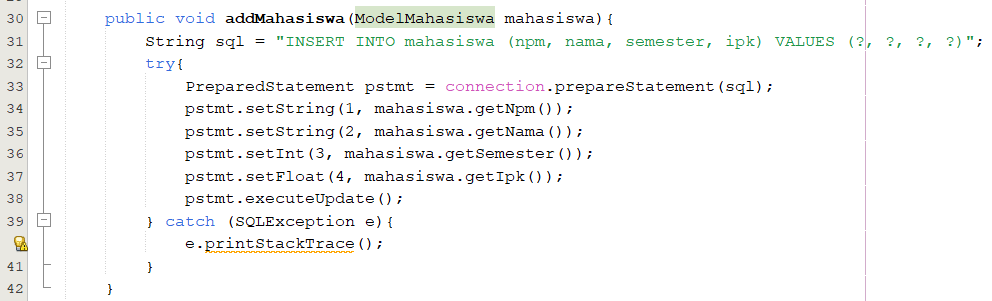
Konstruktor ini digunakan untuk menginisialisasi koneksi database:

* Class.forName("com.mysql.cj.jdbc.Driver") memuat driver MySQL JDBC.
* DriverManager.getConnection(...) digunakan untuk menghubungkan ke database mvc\_db di localhost dengan username root dan tanpa password
* Jika terjadi kesalahan, e.printStackTrace() mencetak detail error ke konsol.



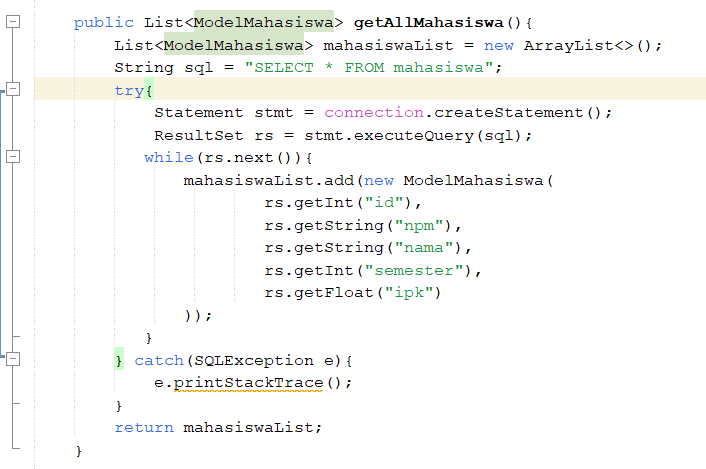
Metode ini mengecek apakah koneksi ke database berhasil:

* Mengembalikan true jika koneksi aktif dan belum ditutup.
* Jika ada kesalahan SQL, detailnya akan dicetak, dan metode mengembalikan false.



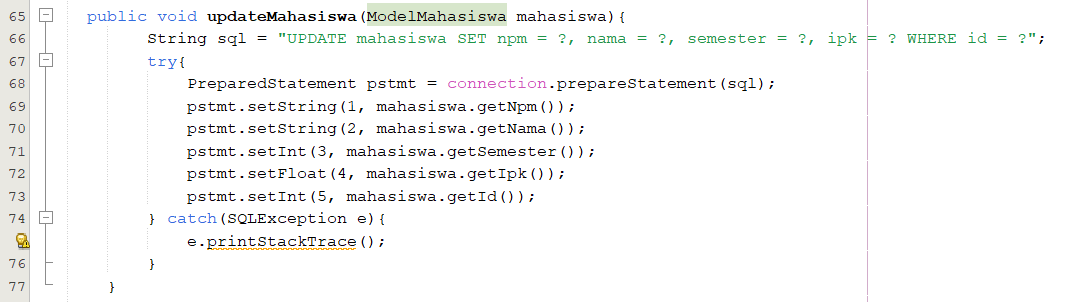
Metode ini menambahkan data mahasiswa baru ke tabel mahasiswa:

* Menggunakan PreparedStatement untuk mencegah SQL injection.
* Mengisi nilai npm, nama, semester, dan ipk dari objek ModelMahasiswa yang diterima sebagai parameter.
* executeUpdate() mengeksekusi query untuk memasukkan data ke database.



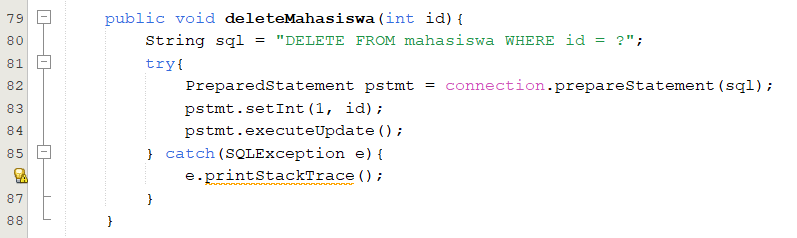
Metode ini mengambil semua data dari tabel mahasiswa dan mengembalikannya dalam bentuk List<ModelMahasiswa>:

* Menggunakan Statement dan ResultSet untuk mengeksekusi dan menyimpan hasil query SELECT \* FROM mahasiswa.
* Menggunakan while(rs.next()) untuk mengiterasi hasil query, membuat objek ModelMahasiswa untuk setiap baris data, dan menambahkannya ke mahasiswaList.
* Mengembalikan mahasiswaList yang berisi data semua mahasiswa.



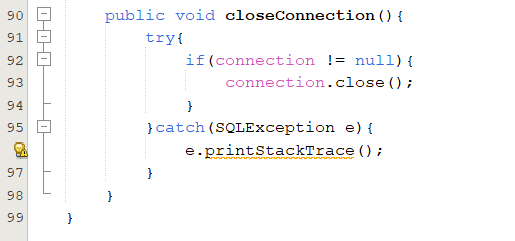
Metode ini memperbarui data mahasiswa berdasarkan ID yang diberikan:

* Menggunakan PreparedStatement dengan query UPDATE untuk memperbarui data mahasiswa.
* Mengisi nilai npm, nama, semester, ipk, dan id dari objek ModelMahasiswa yang diberikan sebagai parameter.
* executeUpdate() mengeksekusi query untuk memperbarui data di database.



Metode ini menghapus data mahasiswa berdasarkan ID:

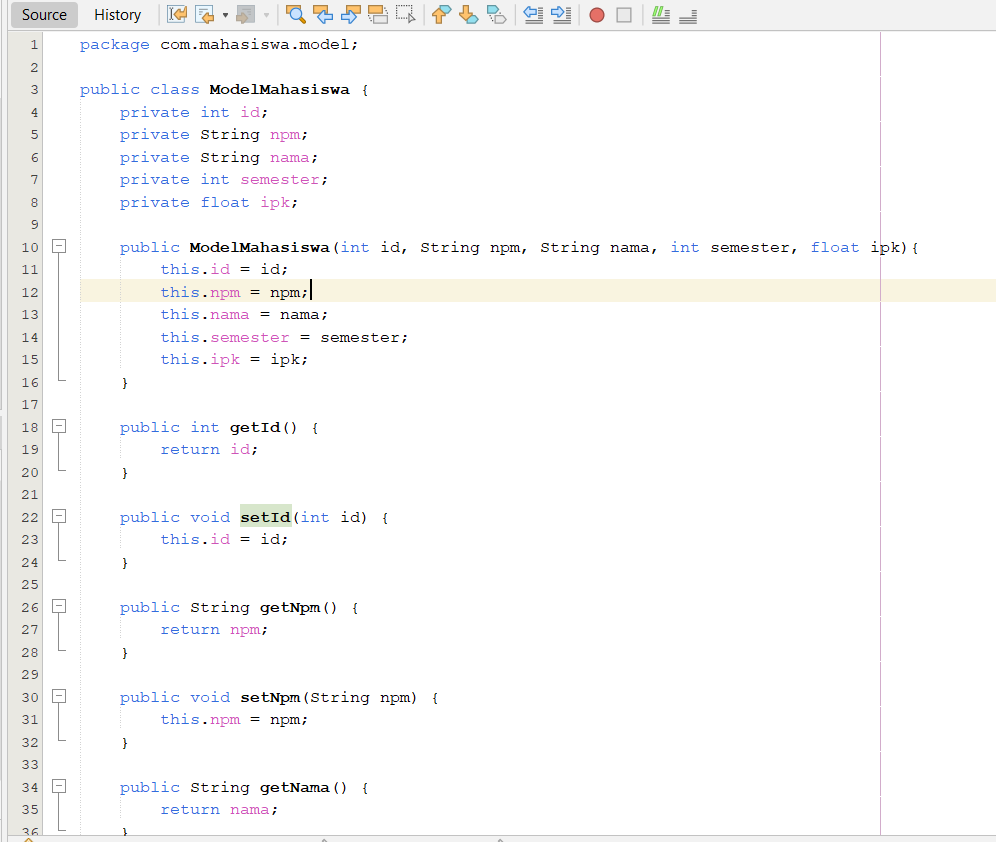
* Menggunakan PreparedStatement untuk mencegah SQL injection.
* Memasukkan nilai id sebagai parameter dalam query DELETE FROM mahasiswa WHERE id = ?.
* executeUpdate() mengeksekusi query untuk menghapus data dari database.



Metode ini menutup koneksi database jika masih terbuka:

* Mengecek apakah connection tidak null, lalu memanggil close() untuk menutup koneksi.
* Jika terjadi kesalahan saat menutup koneksi, detailnya dicetak ke konsol.

C.ModelMahasiswa



Kelas ModelMahasiswa adalah kelas model yang merepresentasikan entitas mahasiswa dalam sistem. Kelas ini memiliki beberapa atribut (atau field) yang mendeskripsikan data mahasiswa:

* id (int): Menyimpan ID unik untuk setiap mahasiswa.
* npm (String): Menyimpan Nomor Pokok Mahasiswa (NPM), yang umumnya adalah identifikasi unik mahasiswa di universitas.
* nama (String): Menyimpan nama mahasiswa.
* semester (int): Menyimpan semester aktif mahasiswa.
* ipk (float): Menyimpan IPK (Indeks Prestasi Kumulatif) mahasiswa.

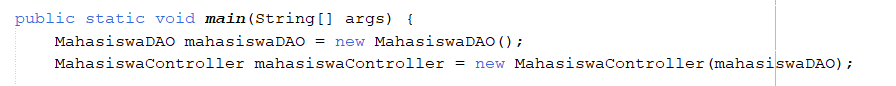
Field ini diatur sebagai private, sehingga hanya dapat diakses dan diubah melalui metode getter dan setter yang disediakan.

Konstruktor disini memungkinkan untuk membuat objek ModelMahasiswa baru dengan semua atribut yang diperlukan. Konstruktor ini menerima lima parameter:

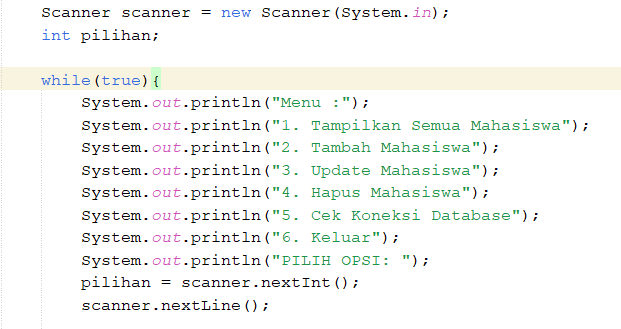
* id, npm, nama, semester, dan ipk.
* Masing-masing parameter ini kemudian diinisialisasi ke field yang sesuai di dalam kelas menggunakan this.

getter dan setter digunakan untuk mengakses dan memodifikasi nilai-nilai dari field id, npm, nama, semester, dan ipk dari luar kelas ini.

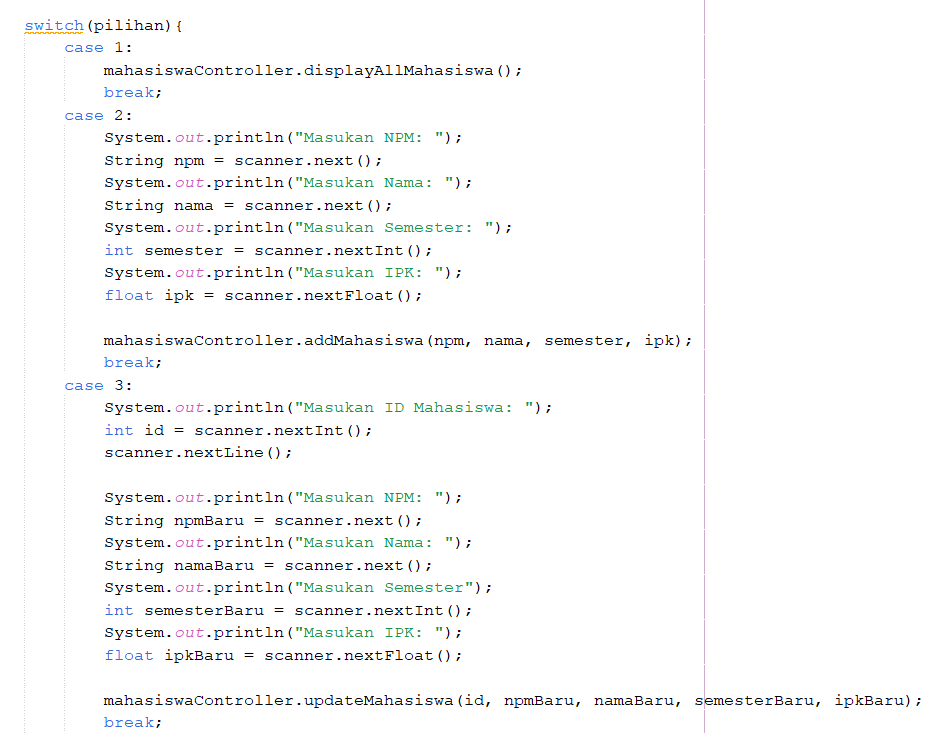
D.MahasiswaView



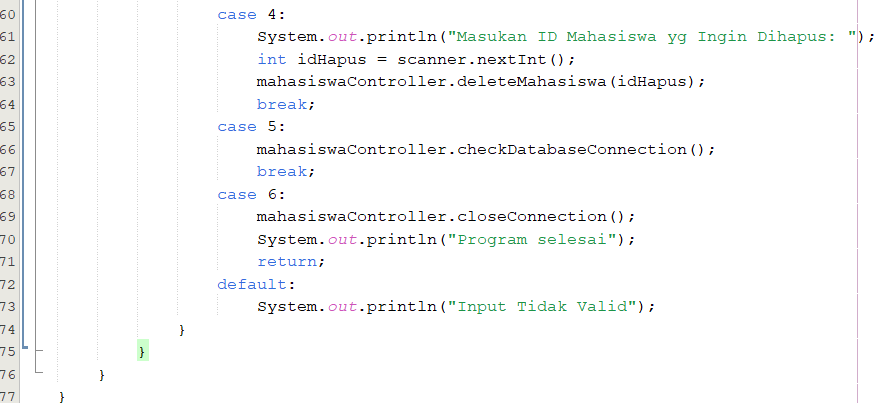
* Membuat objek MahasiswaDAO untuk mengelola interaksi dengan database.
* Membuat objek MahasiswaController, yang akan menerima mahasiswaDAO sebagai parameter agar dapat menggunakan akses database.



* **Scanner**: Digunakan untuk menerima input pengguna dari konsol.
* **Looping Menu**: Menampilkan menu berulang kali dengan pilihan fitur aplikasi.
* **Pilihan Input**: Meminta input nomor dari pengguna untuk memilih opsi pada menu.

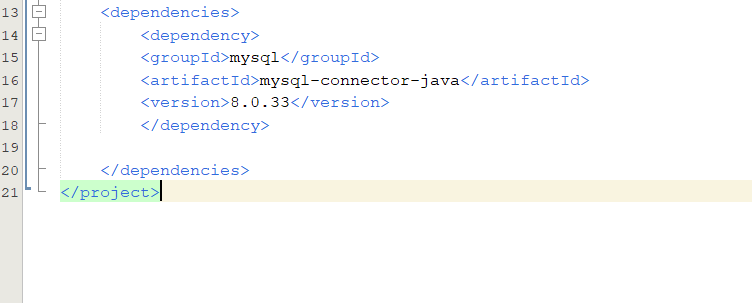


* **case 1**: Menampilkan semua data mahasiswa dengan memanggil displayAllMahasiswa() pada mahasiswaController.
* **case 2**: Menambahkan data mahasiswa baru. Program meminta npm, nama, semester, dan ipk dari pengguna, kemudian memanggil addMahasiswa() untuk menyimpan data baru.
* **case 3**: Memperbarui data mahasiswa berdasarkan id. Program meminta data baru dari pengguna (npm, nama, semester, ipk) dan memanggil updateMahasiswa() pada mahasiswaController.



* **case 4**: Menghapus data mahasiswa berdasarkan id. Program meminta id dari pengguna dan memanggil deleteMahasiswa() pada mahasiswaController.
* **case 5**: Memeriksa koneksi database dengan memanggil checkDatabaseConnection() pada mahasiswaController.
* **case 6**: Mengakhiri program dengan menutup koneksi ke database (closeConnection()) dan menampilkan pesan "Program selesai".
* **default**: Menampilkan pesan "Input Tidak Valid" jika pengguna memasukkan angka yang tidak ada dalam menu.

E.pom.xml



Kegunaan dari kode ini adalah untuk mengimpor MySQL Connector/J ke dalam proyek Java. Dengan adanya dependensi ini, netbeans dapat menggunakan API JDBC untuk menghubungkan aplikasi Java dengan database MySQL, melakukan operasi seperti query, pembaruan, dan pengelolaan data.