NAMA : THEODORE GABBELAMBOK SITUMORANG

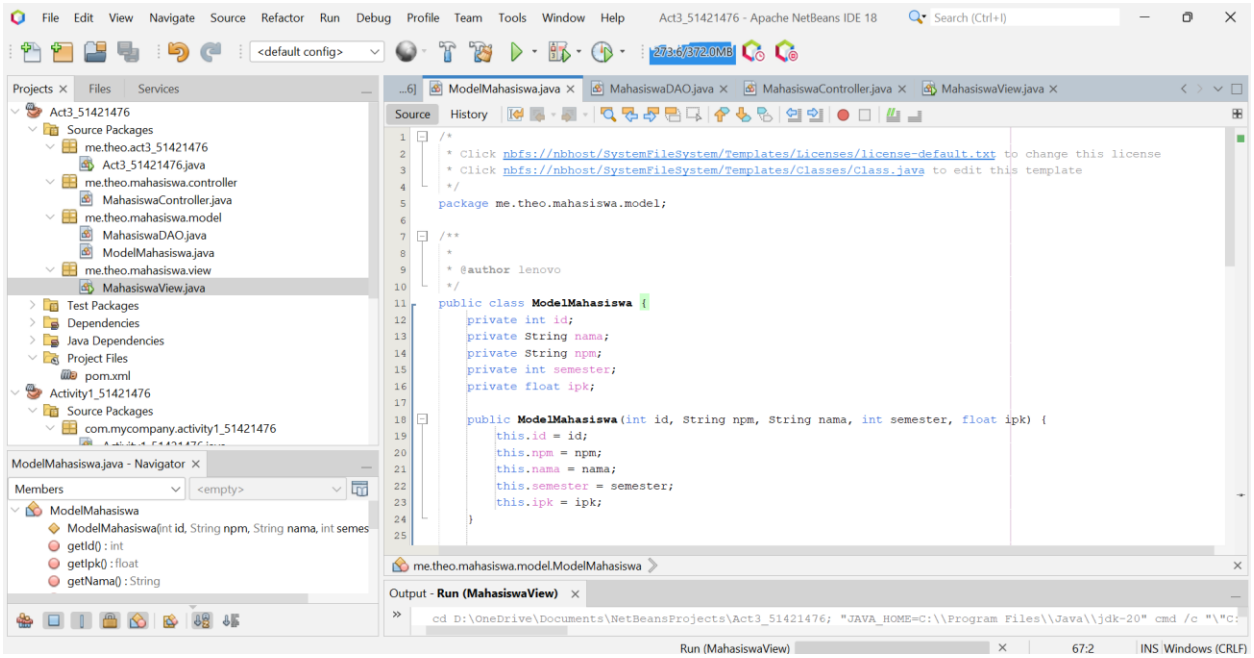
KELAS : 4IA27

NPM : 51421476

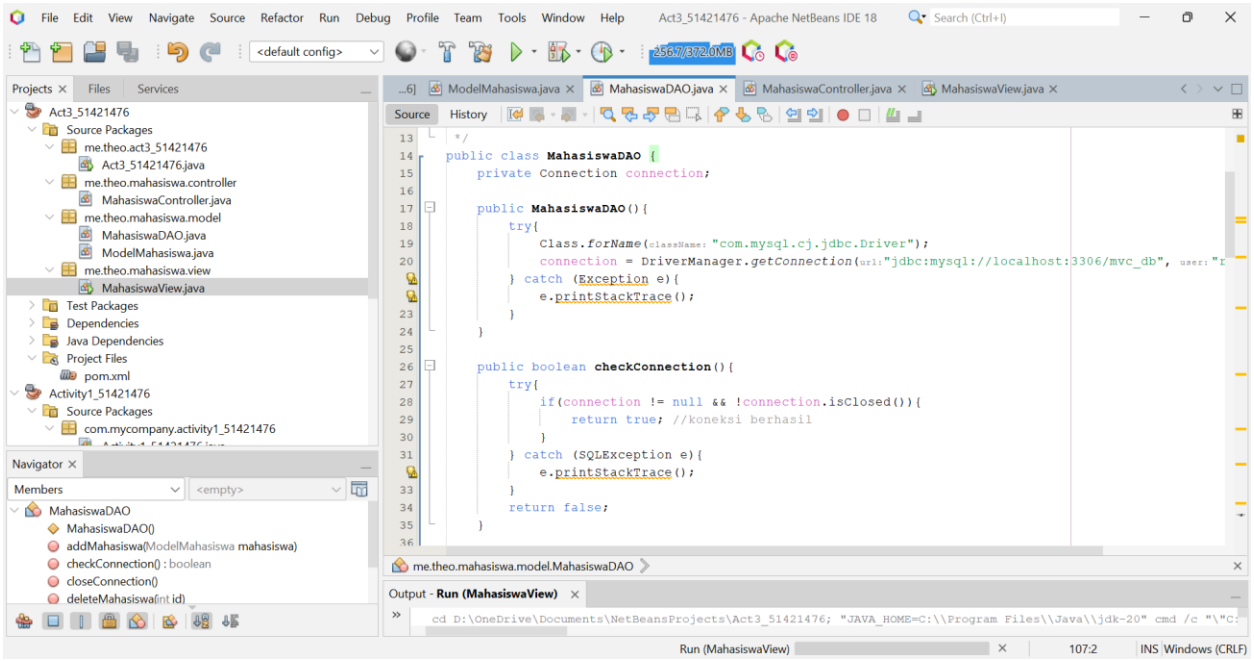
Pertemuan 3
Rekayasa Perangkat Lunak

INPUT

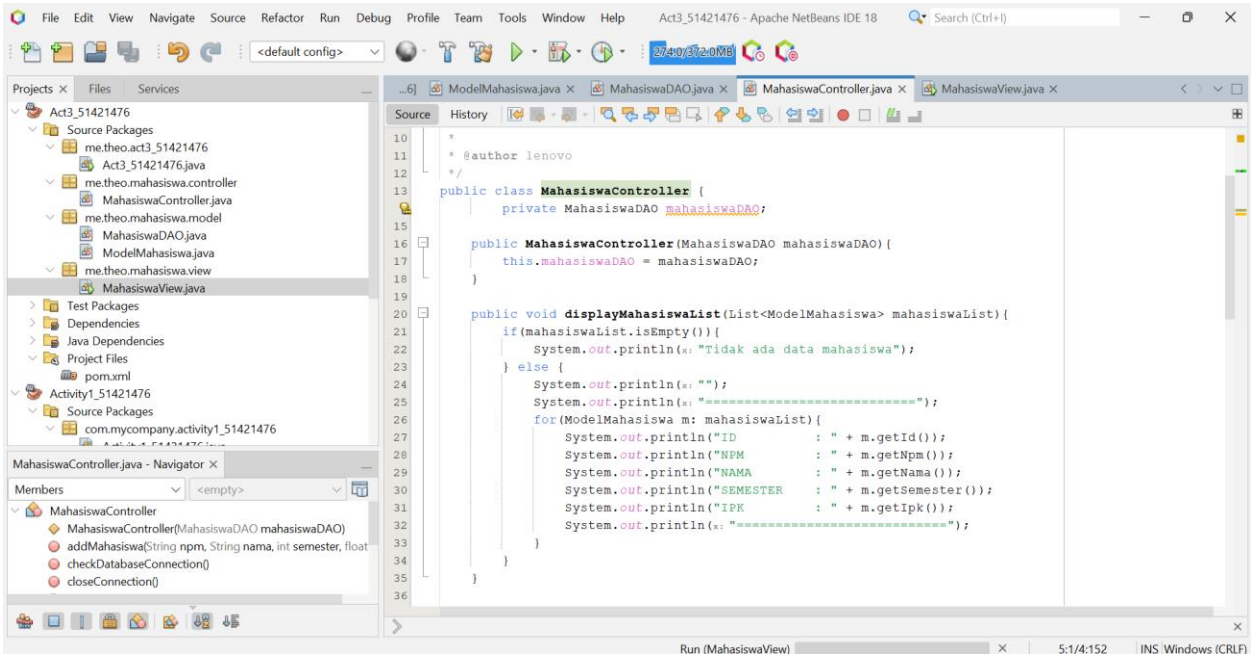
ModelMahasiswa.java



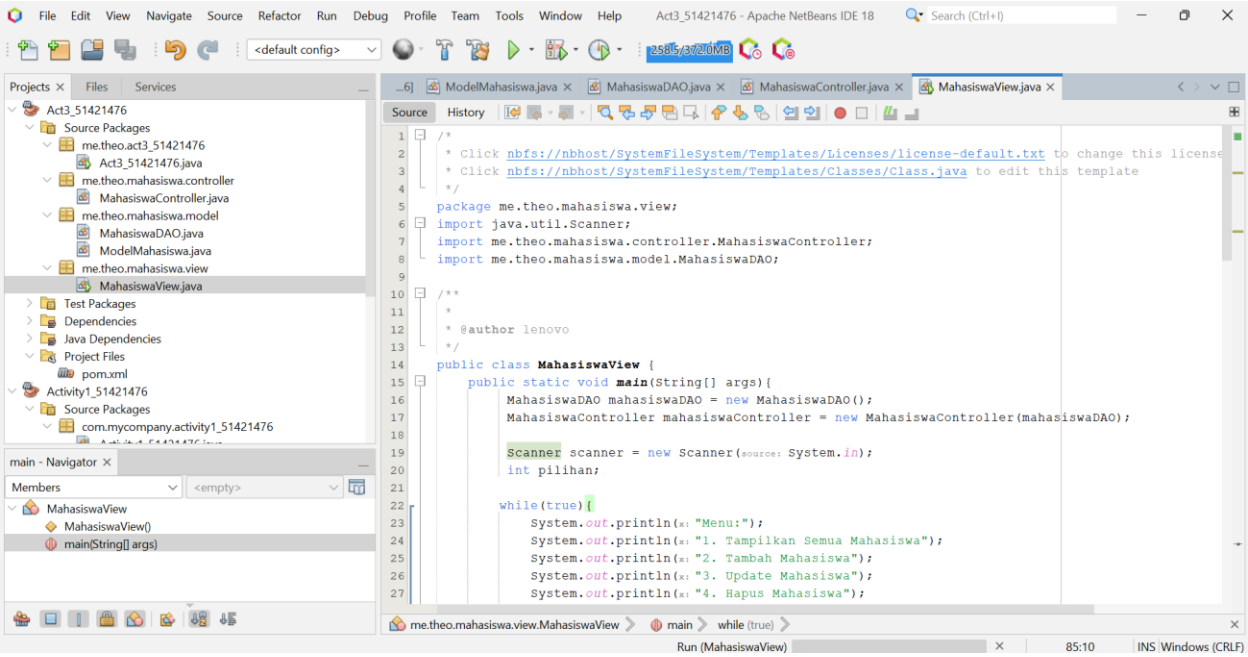
MahasiswaDAO.java



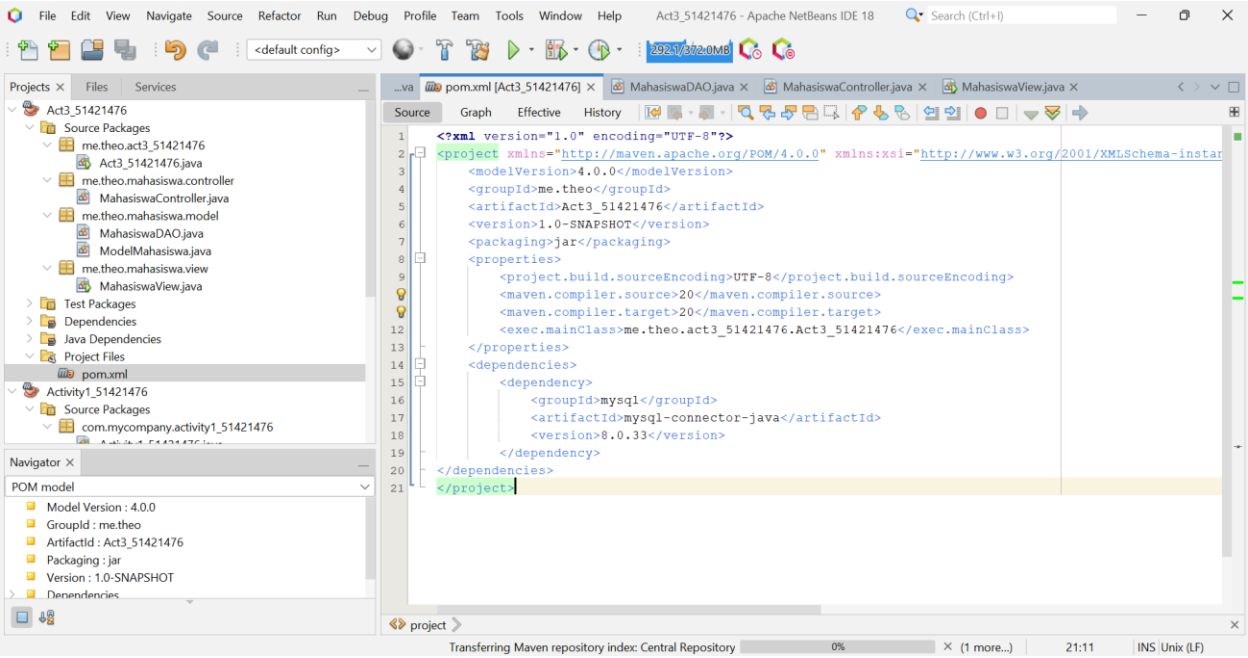
MahasiswaController.java



MahasiswaView.java

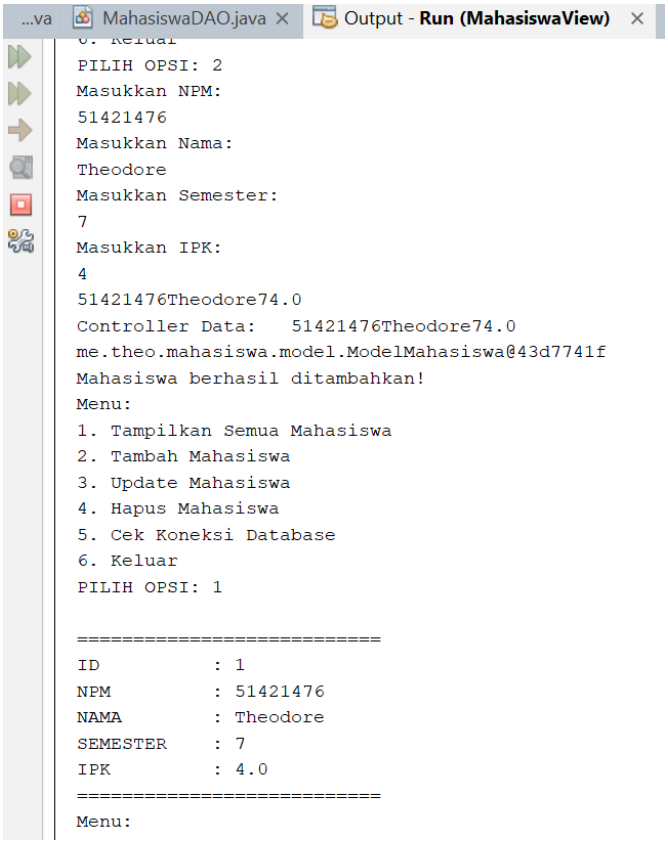


Pom.xml



OUTPUT


Output - Run (MahasiswaView)
cd D:\OneDrive\Documents\NetBeansProjects\Act3_51421476; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:
Scanning for projects...
----------------------< me.theo:Act3_51421476 >-----------------------
Building Act3_51421476 1.0-SNAPSHOT
from pom.xml
--------------------------------[jar]---------------------------------
The artifact mysql:mysql-connector-java:jar:8.0.33 has been relocated to com.mysql:mysql-connector-j:jar:8.0.33:
--- resources:3.3.0:resources (default-resources) @ Act3_51421476 ---
skip non existing resourceDirectory D:\OneDrive\Documents\NetBeansProjects\Act3_51421476\src\main\resources
--- compiler:3.10.1:compile (default-compile) @ Act3_51421476 ---
Changes detected - recompiling the module!
Compiling 5 source files to D:\OneDrive\Documents\NetBeansProjects\Act3_51421476\target\classes
--- exec:3.1.0:exec (default-cli) @ Act3_51421476 ---
Menu:
1. Tampilkan Semua Mahasiswa
2. Tambah Mahasiswa
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI:


...va
MahasiswaDAO.java
Output - Run (MahasiswaView)
6. Keluar
PILIH OPSI: 2
Masukkan NPM:
51421476
Masukkan Nama:
Theodore
Masukkan Semester:
7
Masukkan IPK:
4
51421476Theodore74.0
Controller Data: 51421476Theodore74.0
me.theo.mahasiswa.model.ModelMahasiswa@43d7741f
Mahasiswa berhasil ditambahkan!
Menu:
1. Tampilkan Semua Mahasiswa
2. Tambah Mahasiswa
3. Update Mahasiswa
4. Hapus Mahasiswa
5. Cek Koneksi Database
6. Keluar
PILIH OPSI: 1
==========
ID : 1
NPM : 51421476
NAMA : Theodore
SEMESTER : 7
IPK : 4.0
==========
Menu:

NAMA : **THEODORE GABBELAMBOK SITUMORANG**

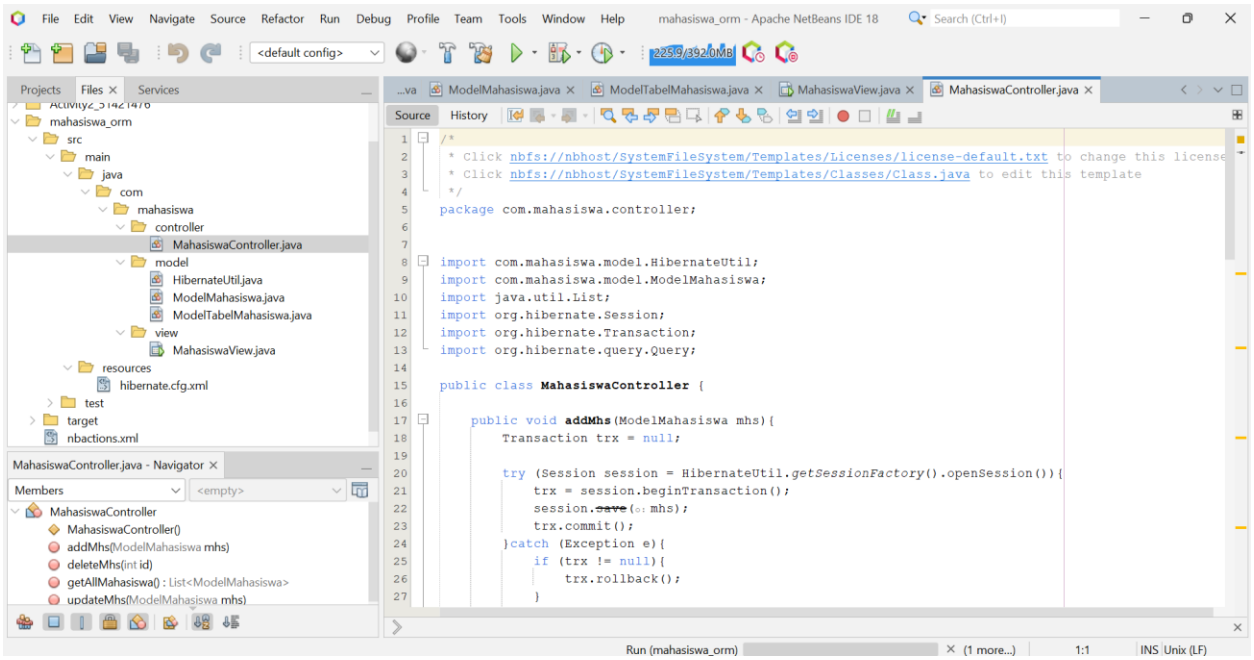
KELAS : **4IA27**

NPM : **51421476**

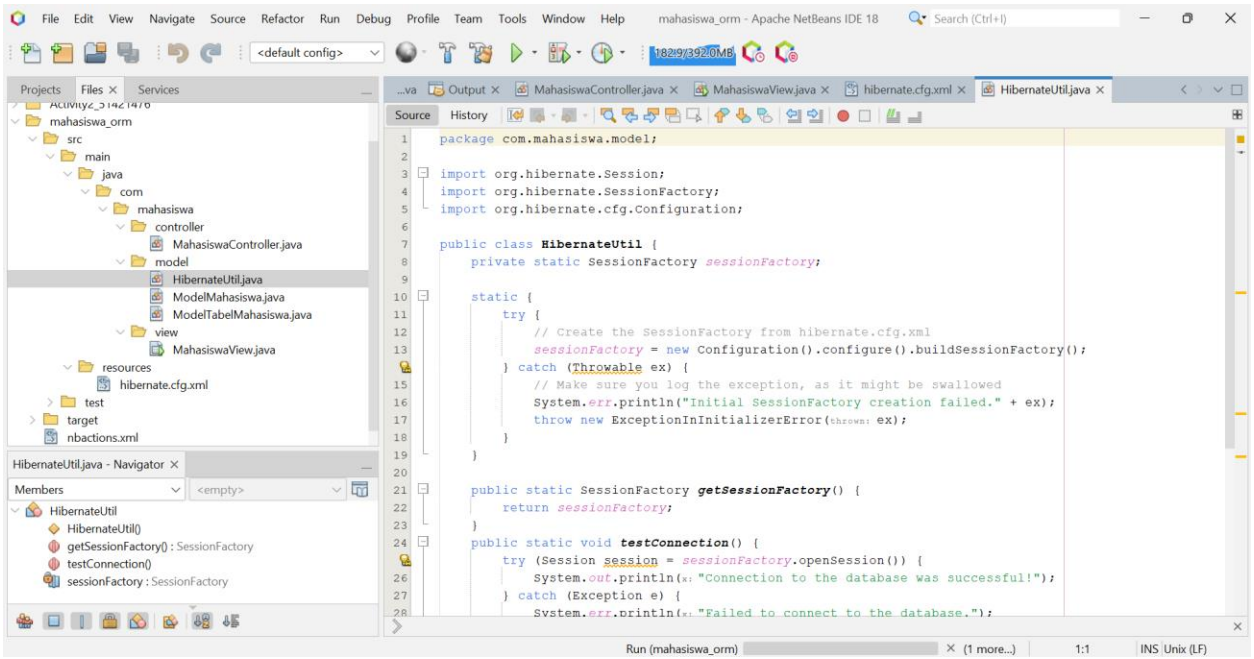
Pertemuan 4
Rekayasa Perangkat Lunak

INPUT

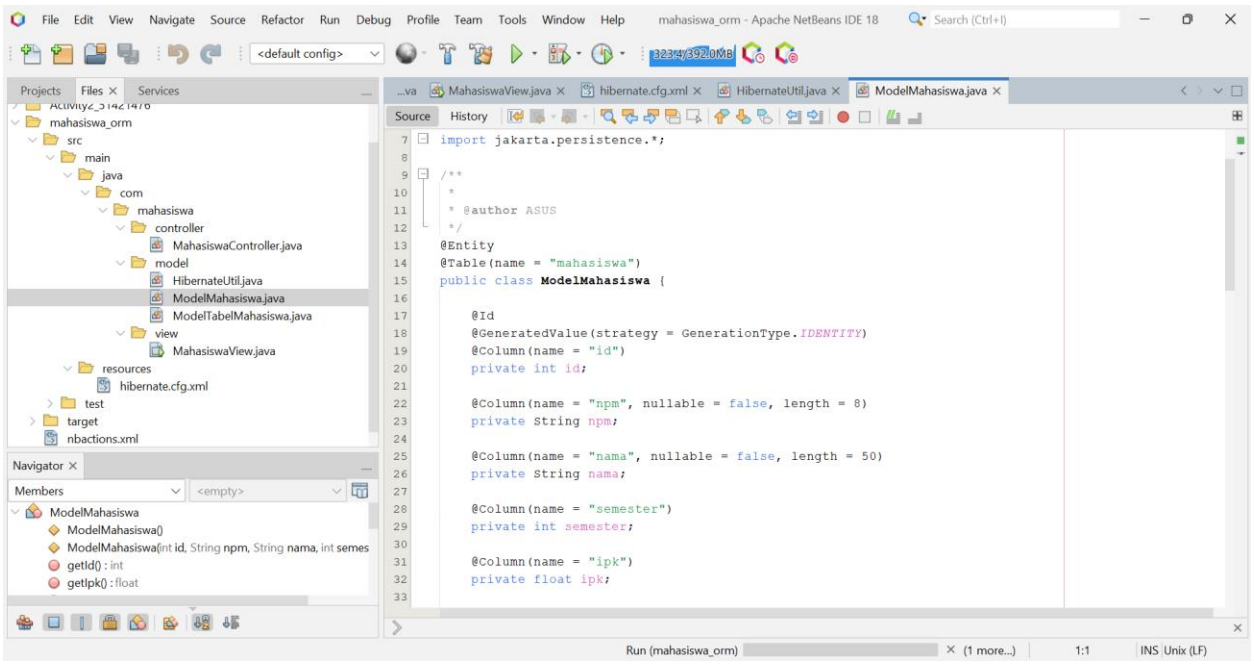
MahasiswaController.java



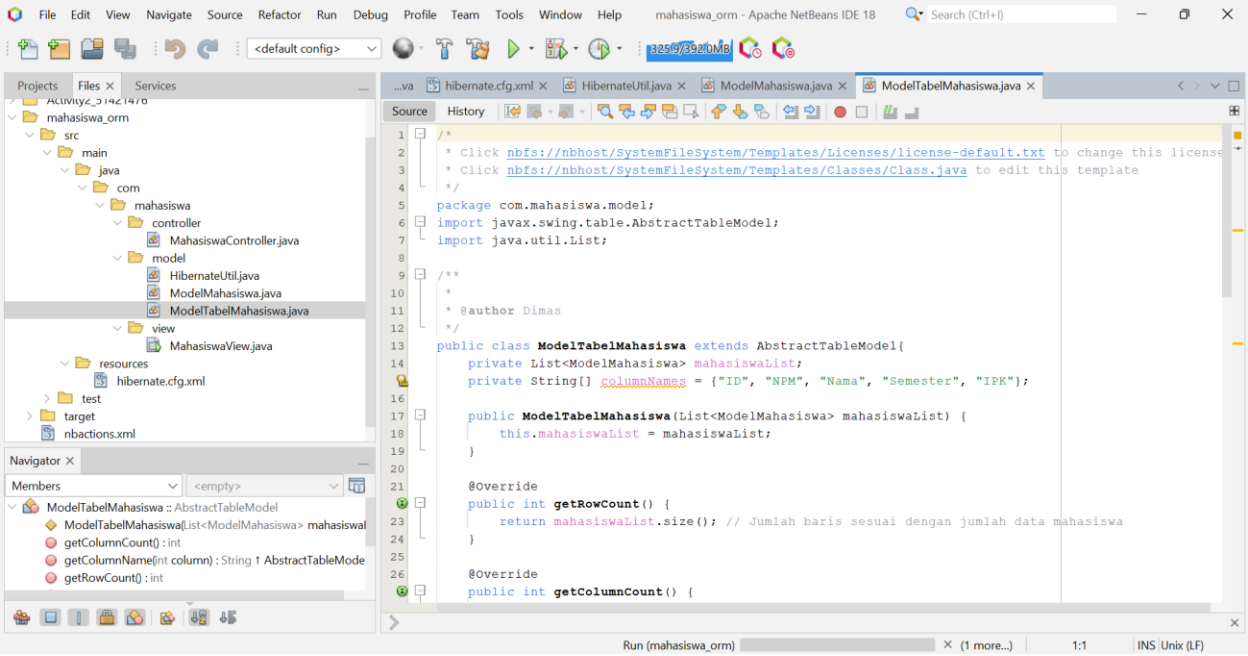
HibernateUtil.java



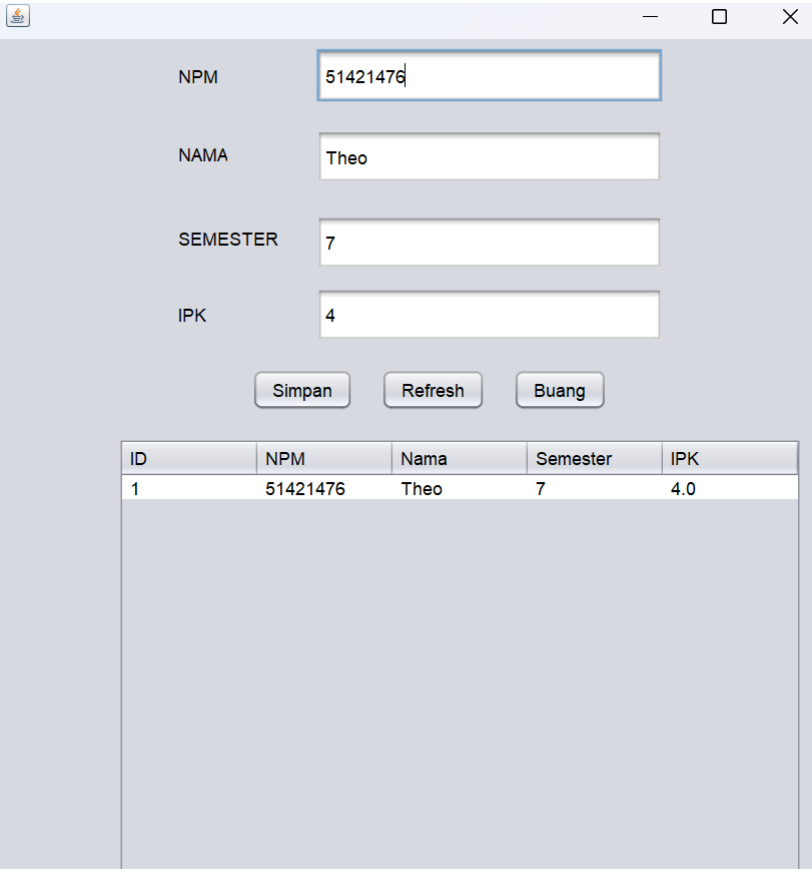
ModelMahasiswa.java

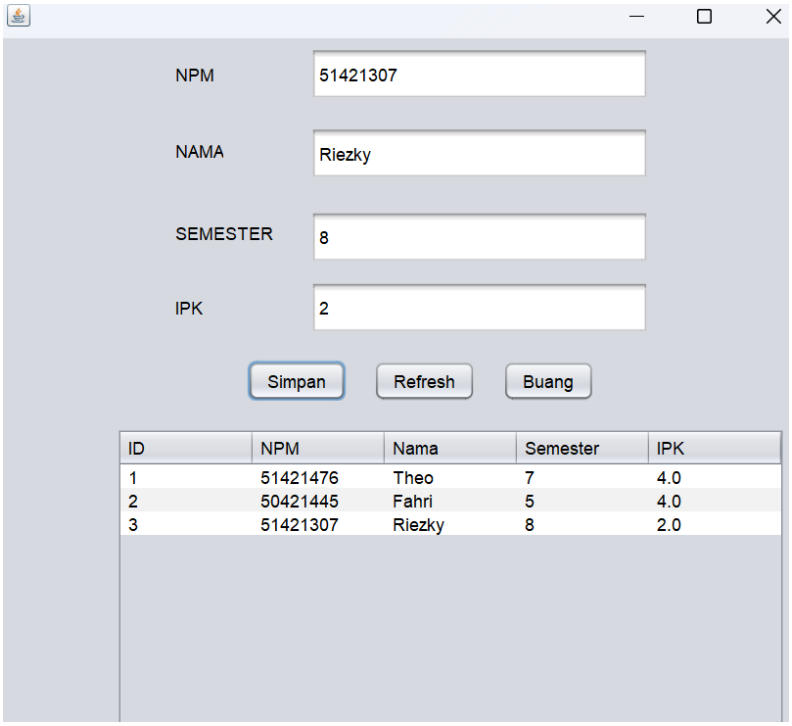


ModelTabelMahasiswa.java



OUTPUT





Pengertian MVC?

MVC (Model-View-Controller) adalah sebuah pola arsitektur perangkat lunak yang digunakan untuk memisahkan aplikasi menjadi tiga komponen utama:

1. **Model**: Komponen ini bertanggung jawab untuk mengelola data, logika bisnis, dan aturan yang berkaitan dengan data. Model berfungsi untuk berinteraksi dengan basis data dan memberikan data kepada komponen lain.
2. **View**: Komponen ini bertugas untuk menampilkan data kepada pengguna. View menerima data dari Model dan bertanggung jawab untuk mengatur bagaimana data tersebut ditampilkan, misalnya dalam bentuk halaman web atau antarmuka pengguna.
3. **Controller**: Komponen ini berfungsi sebagai perantara antara Model dan View. Controller menerima input dari pengguna, memprosesnya (misalnya dengan mengubah data dalam Model), dan memperbarui View agar mencerminkan perubahan tersebut.

Keuntungan Menggunakan MVC:

- **Pemeliharaan yang Lebih Mudah**: Dengan memisahkan komponen, pengembang dapat memelihara dan memperbarui setiap bagian secara terpisah tanpa memengaruhi yang lain.
- **Pengembangan yang Terorganisir**: MVC membantu tim pengembang untuk bekerja secara bersamaan dengan membagi tanggung jawab.
- **Fleksibilitas dan Reusabilitas**: Komponen dapat digunakan kembali dalam proyek lain, dan tampilan dapat dengan mudah diganti tanpa mengubah logika bisnis.

Pola ini banyak digunakan dalam pengembangan aplikasi web dan desktop, serta dalam kerangka kerja seperti Ruby on Rails, ASP.NET MVC, dan Laravel.

Pengertian orm dan hibernate?

ORM (Object-Relational Mapping) adalah sebuah teknik dalam pemrograman yang memungkinkan data dari sebuah basis data relasional dapat diakses dan dimanipulasi sebagai objek dalam bahasa pemrograman berorientasi objek (OOP). Dengan ORM, pengembang tidak perlu menulis kode SQL secara manual untuk berinteraksi dengan database, karena ORM menyediakan mekanisme untuk mengkonversi objek menjadi tabel database, dan sebaliknya.

Kelebihan ORM

1. **Mengurangi Kompleksitas**: Menghindari penulisan SQL secara manual.
2. **Pemeliharaan Kode yang Lebih Mudah**: Kode yang lebih konsisten dan rapi.
3. **Keamanan**: Mengurangi potensi terjadinya SQL Injection dengan parameterisasi query.

Hibernate

Hibernate adalah salah satu implementasi ORM paling populer, terutama di ekosistem bahasa pemrograman Java. Hibernate menyediakan cara bagi pengembang untuk menyimpan, memperbarui, dan mengambil objek Java dari database tanpa menulis SQL secara langsung.

Fitur Utama Hibernate:

- **Mapping Otomatis**: Menghubungkan (mapping) antara objek Java dengan tabel di database.
- **Lazy Loading**: Memungkinkan pengambilan data secara ditunda hingga benar-benar dibutuhkan.
- **Caching**: Menyediakan cache untuk meningkatkan performa.
- **Database-agnostic**: Mendukung berbagai jenis database dengan sedikit atau tanpa modifikasi pada kode.

Hibernate membuat proses pengembangan lebih cepat, mengurangi kesalahan, dan meningkatkan efisiensi dalam pengelolaan data berbasis objek di Java.

Perbedaan hibernate dan mvc konvensional biasa?

Hibernate dan MVC konvensional adalah dua konsep yang berbeda, dengan fokus dan fungsi yang tidak sama dalam pengembangan perangkat lunak:

1. Tujuan Utama:
    - Hibernate: Adalah framework ORM yang berfungsi untuk memetakan objek dalam bahasa pemrograman (seperti Java) ke dalam tabel-tabel di database relasional, sehingga mempermudah interaksi dengan database tanpa perlu menulis SQL secara manual.
    - MVC Konvensional: Adalah pola arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga komponen utama (Model, View, dan Controller) untuk menjaga struktur kode yang bersih dan terorganisir dalam menangani data, tampilan, dan kontrol interaksi pengguna.
2. Fokus dan Fungsi Utama:
    - Hibernate: Fokus pada interaksi dengan database. Hibernate menyediakan abstraksi untuk akses dan manipulasi data dalam database, mendukung fitur seperti lazy loading, caching, dan pengelolaan hubungan antar tabel.
    - MVC Konvensional: Fokus pada pengaturan arsitektur aplikasi secara keseluruhan. Dengan membagi aplikasi menjadi Model, View, dan Controller, MVC memastikan pengelolaan data, antarmuka, dan logika kontrol dilakukan secara terpisah.
3. Posisi dalam Aplikasi:
    - Hibernate: Biasa berada dalam lapisan Model di arsitektur MVC. Hibernate digunakan untuk menangani akses data di bagian Model, mengelola objek dan relasinya dalam basis data.
    - MVC Konvensional: Adalah struktur umum aplikasi. Semua bagian (Model, View, Controller) adalah bagian dari MVC, dengan Hibernate hanya sebagai salah satu alat yang mungkin digunakan di lapisan Model untuk mengelola data.
4. Jenis Framework:
    - Hibernate: Adalah framework ORM yang membantu akses data dalam database.
    - MVC Konvensional: Adalah pola arsitektur aplikasi. Tidak berfungsi sebagai ORM, tetapi digunakan untuk mendesain arsitektur aplikasi secara menyeluruh.

Contoh Penerapan

Dalam aplikasi berbasis Java yang mengikuti pola MVC, Hibernate biasanya akan berada di bagian Model untuk menangani pemetaan objek dengan tabel-tabel database, sedangkan View dan Controller akan diurus oleh bagian yang berbeda dalam framework MVC.