LAPORAN AKHIR PRAKTIKUM

Mata Praktikum	: Rekayasa Perangkat Lunak 2
Kelas	: 4IA11
Praktikum ke-	6
Tanggal	: 22 November 2024
Materi	: Implementasi Dependency Injection pada Java Spring
NPM	: 50421786
Nama	: Marsillo Dito Saputra
Ketua Asisten	:
Paraf Asisten	:
Nama Asisten	: Dimas Renaldy Sriwirawan
Jumlah Lembar	: Lembar

LABORATORIUM TEKNIK INFORMATIKA

UNIVERSITAS GUNADARMA

2024

LISTING PROGRAM

(Jawab Soal Untuk Laporan Akhir)

Soal Essay!

Jelaskan apa yang dimaksud dengan clean architecture dan implementasi nya pada

project yang sudah dibuat

Serta berikan penjelasan terhadap isi kodingan kalian

Jawab :

Clean Architecture dan Implementasinya pada Proyek

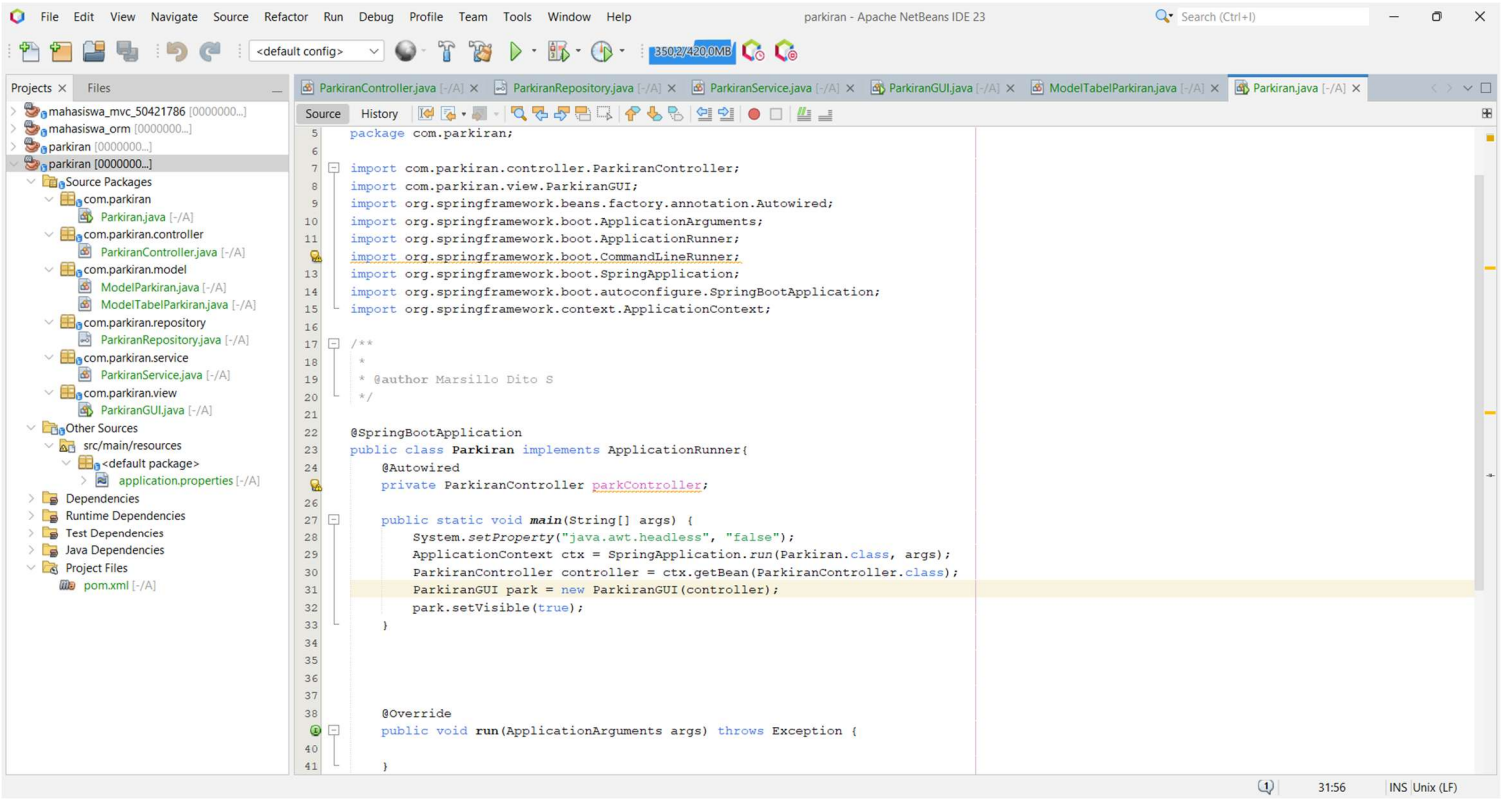
Clean Architecture adalah pendekatan desain perangkat lunak yang memisahkan kode ke dalam beberapa lapisan (layers) untuk memaksimalkan keterpisahan kepentingan (separation of concerns). Dengan arsitektur ini, kode menjadi lebih mudah dikelola, diuji, dan dimodifikasi tanpa memengaruhi seluruh sistem.

Prinsip utama clean architecture:

1. **Independency:** Setiap lapisan tidak bergantung pada detail implementasi lapisan lain.
2. **Dependency Rule:** Ketergantungan harus selalu mengarah ke dalam (ke lapisan yang lebih inti).
3. **Entities:** Mewakili inti domain atau bisnis logic.
4. **Use Cases**: Mengimplementasikan alur aplikasi tanpa bergantung pada teknologi eksternal.
5. **Interface Adapters:** Menghubungkan lapisan domain dengan framework atau teknologi eksternal.
6. **Frameworks and Drivers:** Lapisan paling luar yang bergantung pada teknologi.

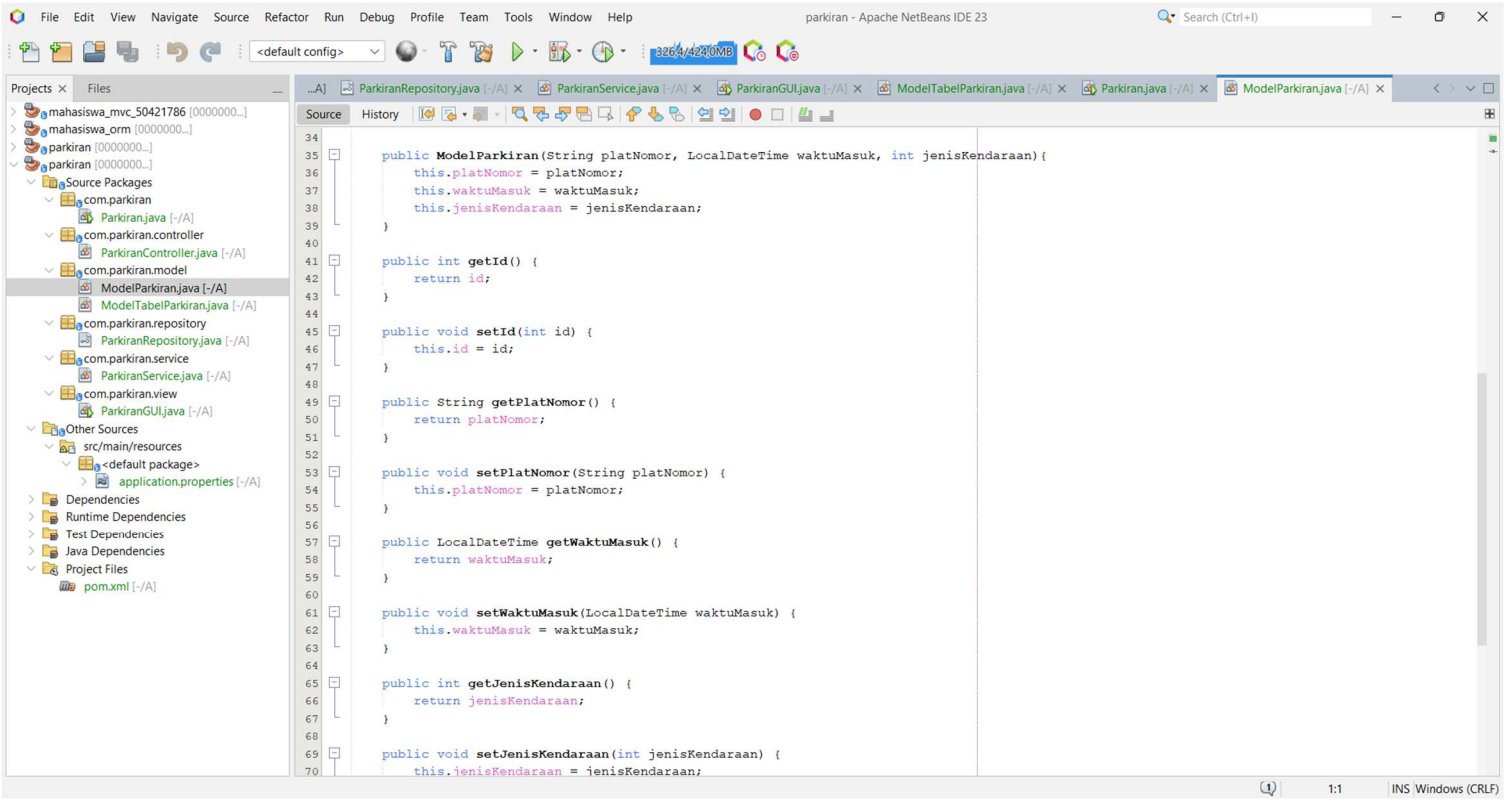
Implementasi pada Proyek Parkiran Pada proyek ini, konsep **clean architecture** diterapkan sebagai berikut:

- **Entities:** *ModelParkiran* berisi data inti tentang kendaraan di parkiran (*e.g., platNomor, waktuMasuk, jenisKendaraan*).
- **Use Cases:** Dijalankan oleh *ParkiranService* untuk menangani logika bisnis seperti:
  - Proses masuk parkir (*masukParkir*).
  - Perhitungan biaya parkir dan durasi parkir saat keluar (*keluarParkir*).
- **Interface Adapters:** *ParkiranController* sebagai perantara antara GUI (interface) dan *ParkiranService.*
- **Frameworks and Drivers:** Lapisan teknologi Spring Boot mengelola dependensi, pengontrol, dan antarmuka GUI melalui *ParkiranGUI.*



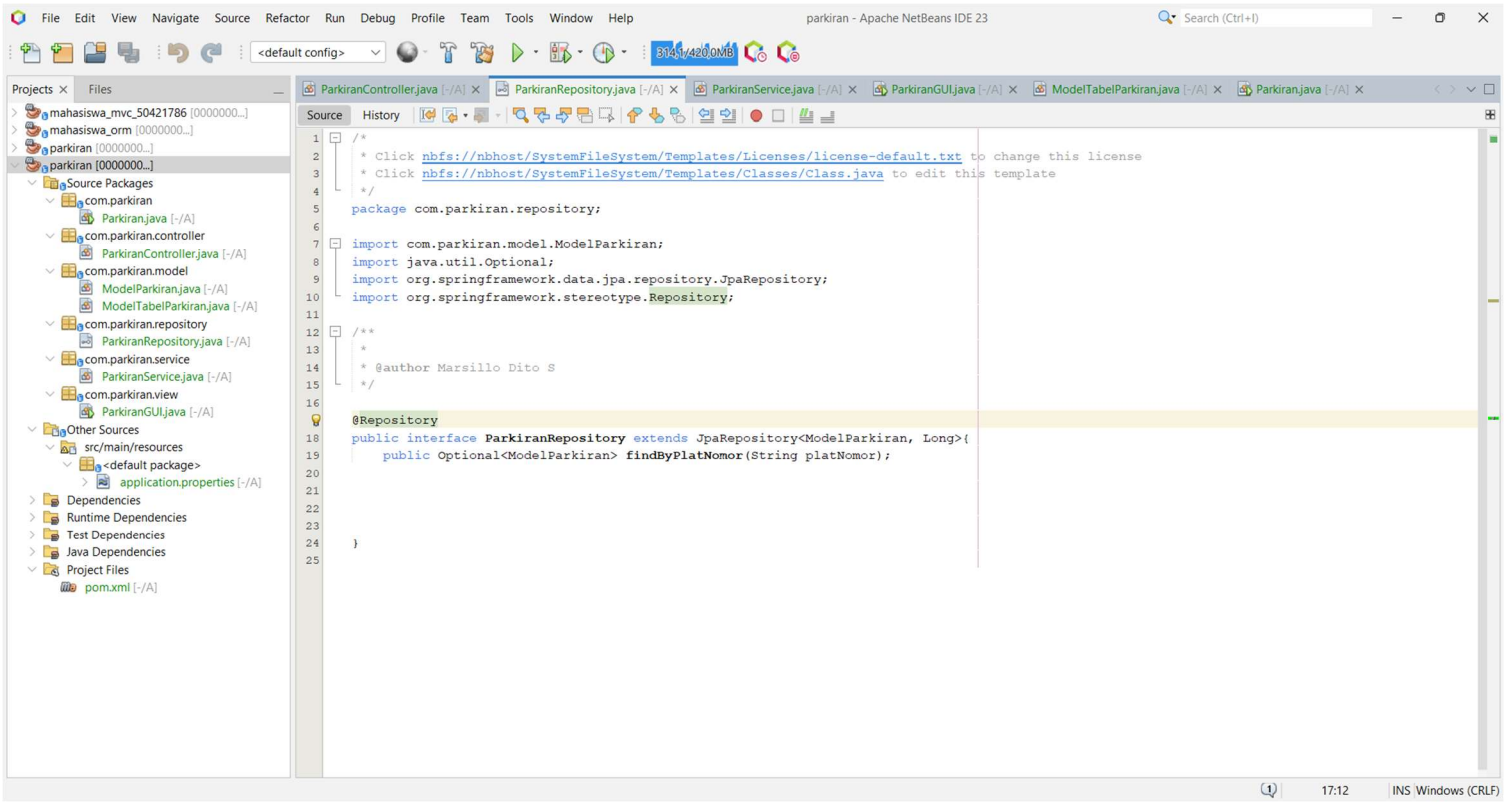
Parkiran.java

- Merupakan **entry point** aplikasi menggunakan **Spring Boot**.
- ParkiranController dan ParkiranGUI dihubungkan menggunakan dependency injection (DI).
- Konfigurasi System.setProperty("java.awt.headless", "false") memastikan GUI dapat berjalan dengan Spring Boot.



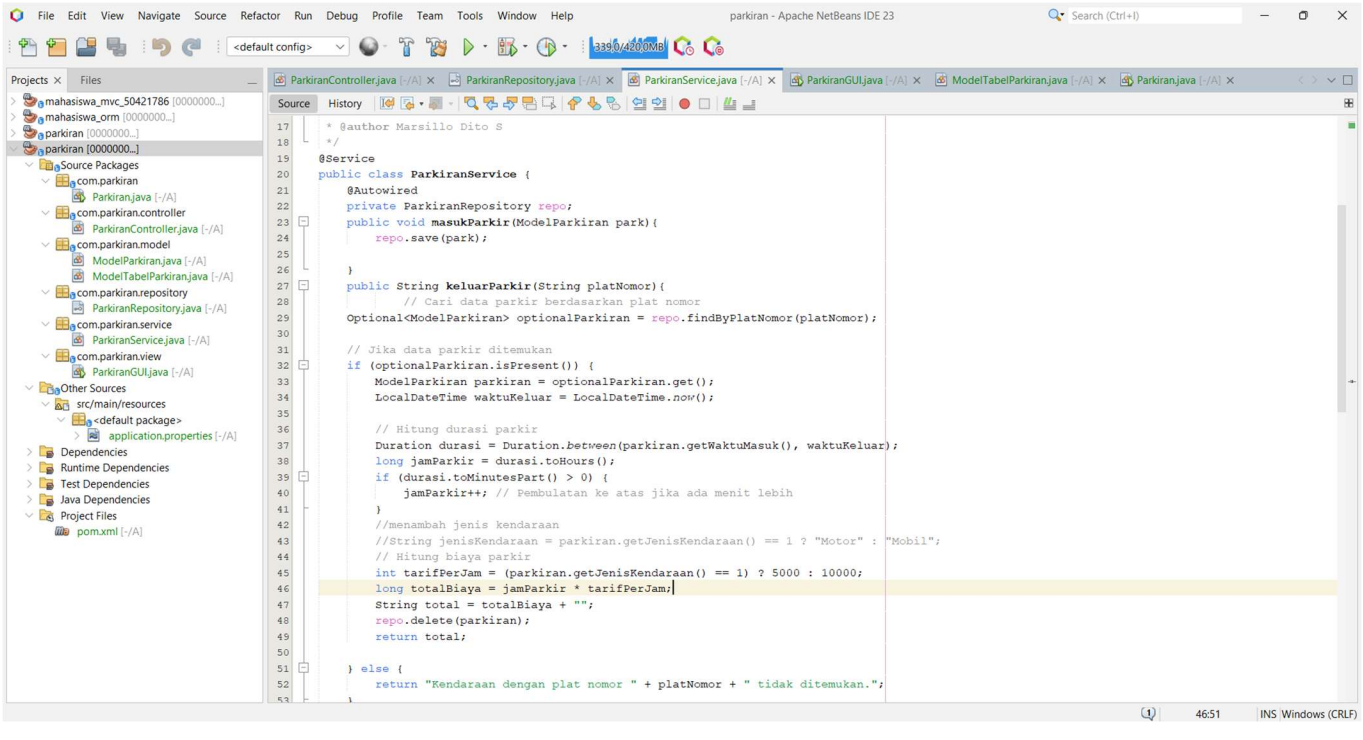
ModelParkiran.java (belum terlihat di file, diasumsikan ada)

- Entity class berisi atribut:
  - platNomor: Plat kendaraan.
  - waktuMasuk: Waktu kendaraan masuk.
  - jenisKendaraan: Jenis kendaraan (motor/mobil).



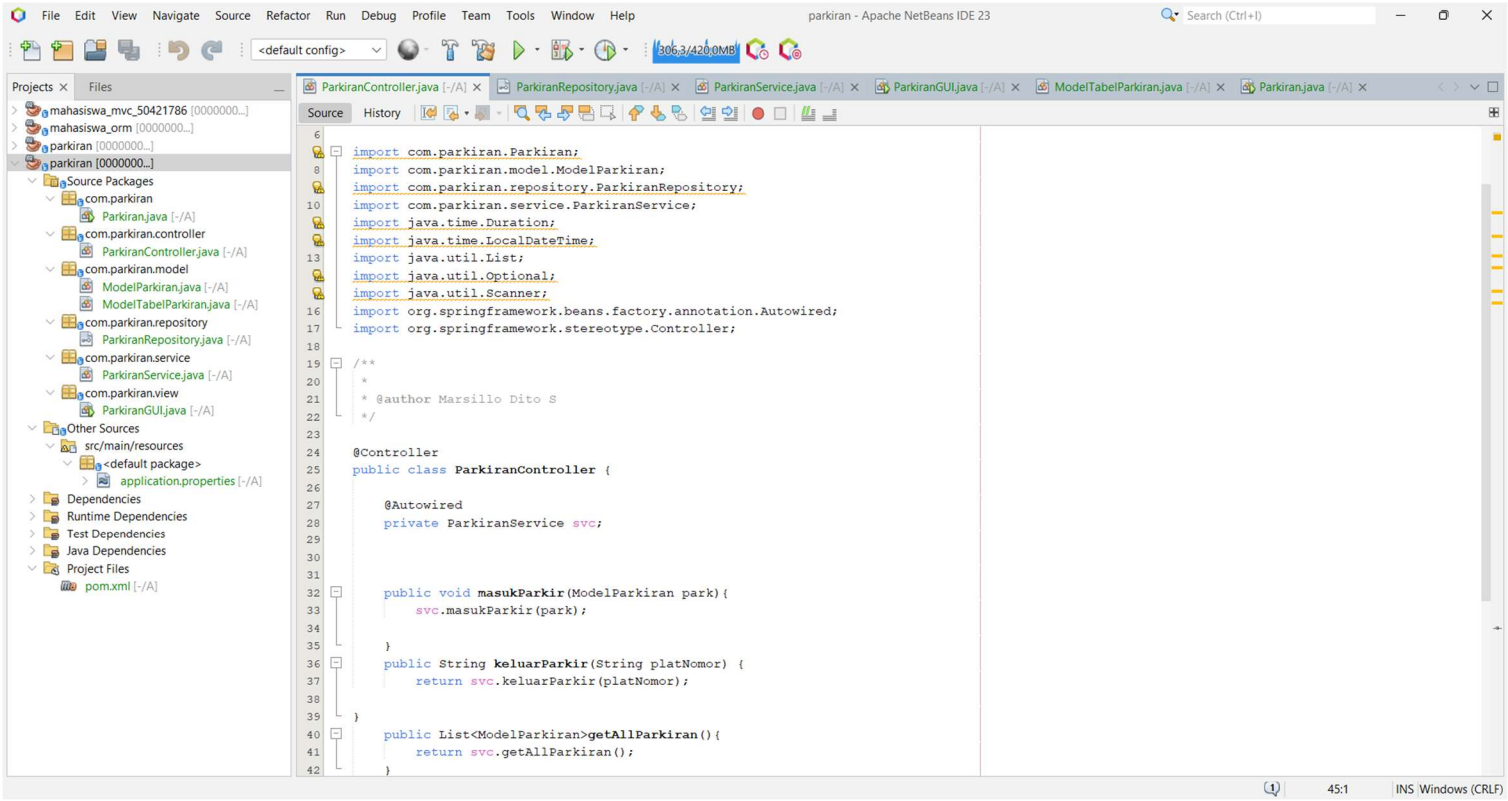
ParkiranRepository.java

- Lapisan **repository** menggunakan **JPA** untuk mengelola data parkiran di database.
- findByPlatNomor memungkinkan pencarian data parkir berdasarkan plat nomor.



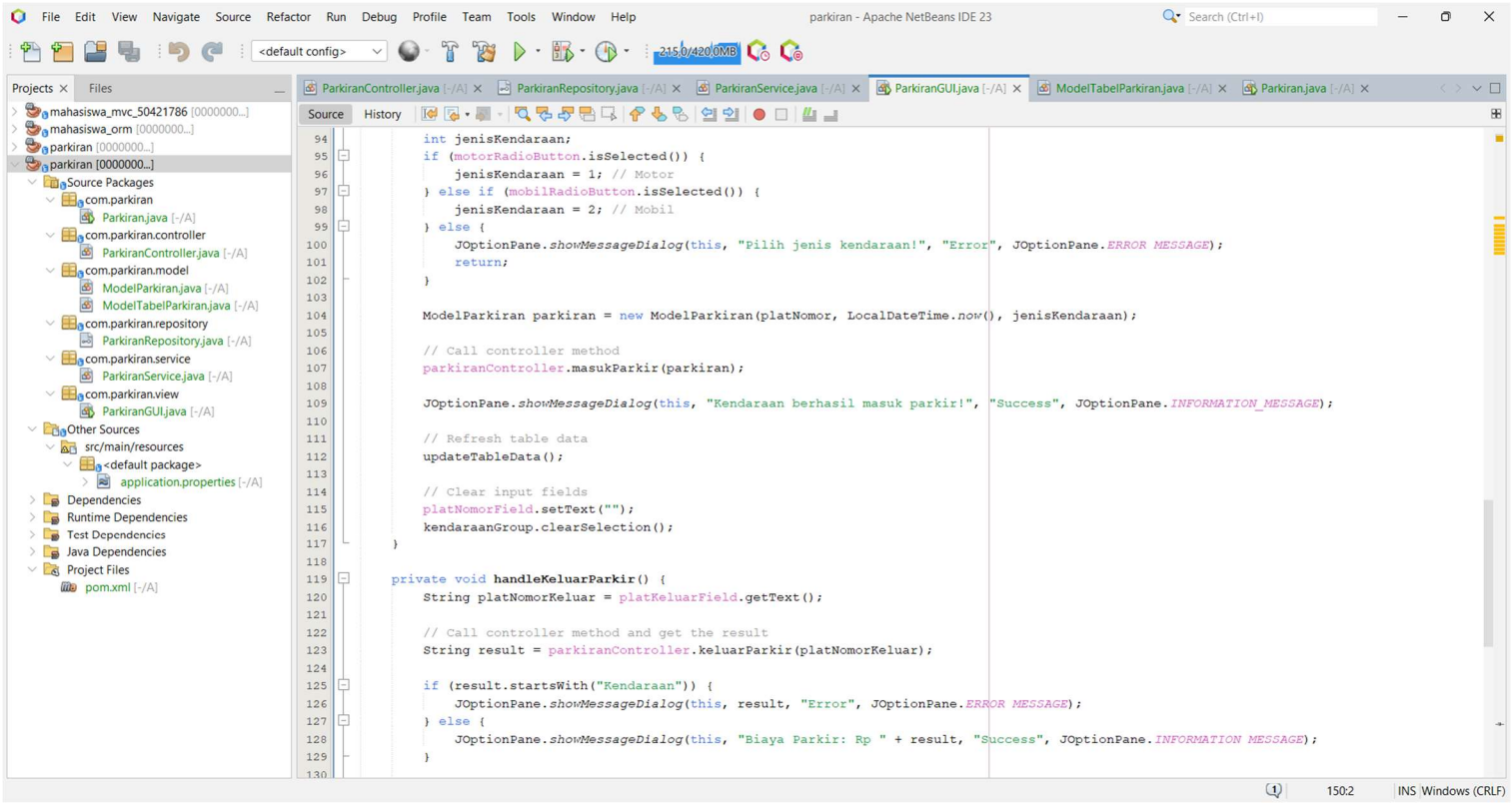
ParkiranService.java

- Berisi logika bisnis utama:
  - **masukParkir**: Menyimpan data kendaraan yang masuk ke database.
  - **keluarParkir**:
    - Mencari data berdasarkan platNomor.
    - Menghitung durasi parkir dan biaya berdasarkan jenis kendaraan.
    - Menghapus data kendaraan dari database setelah keluar.
  - **getAllParkiran**: Mengambil semua data parkiran dari database.



ParkiranController.java

- Bertindak sebagai **interface adapter** yang menerima input dari GUI dan mengarahkan ke ParkiranService.
- Fungsi utama:
  - masukParkir: Memanggil service untuk menyimpan data.
  - keluarParkir: Memanggil service untuk proses keluar.
  - getAllParkiran: Mengembalikan semua data parkiran ke GUI.

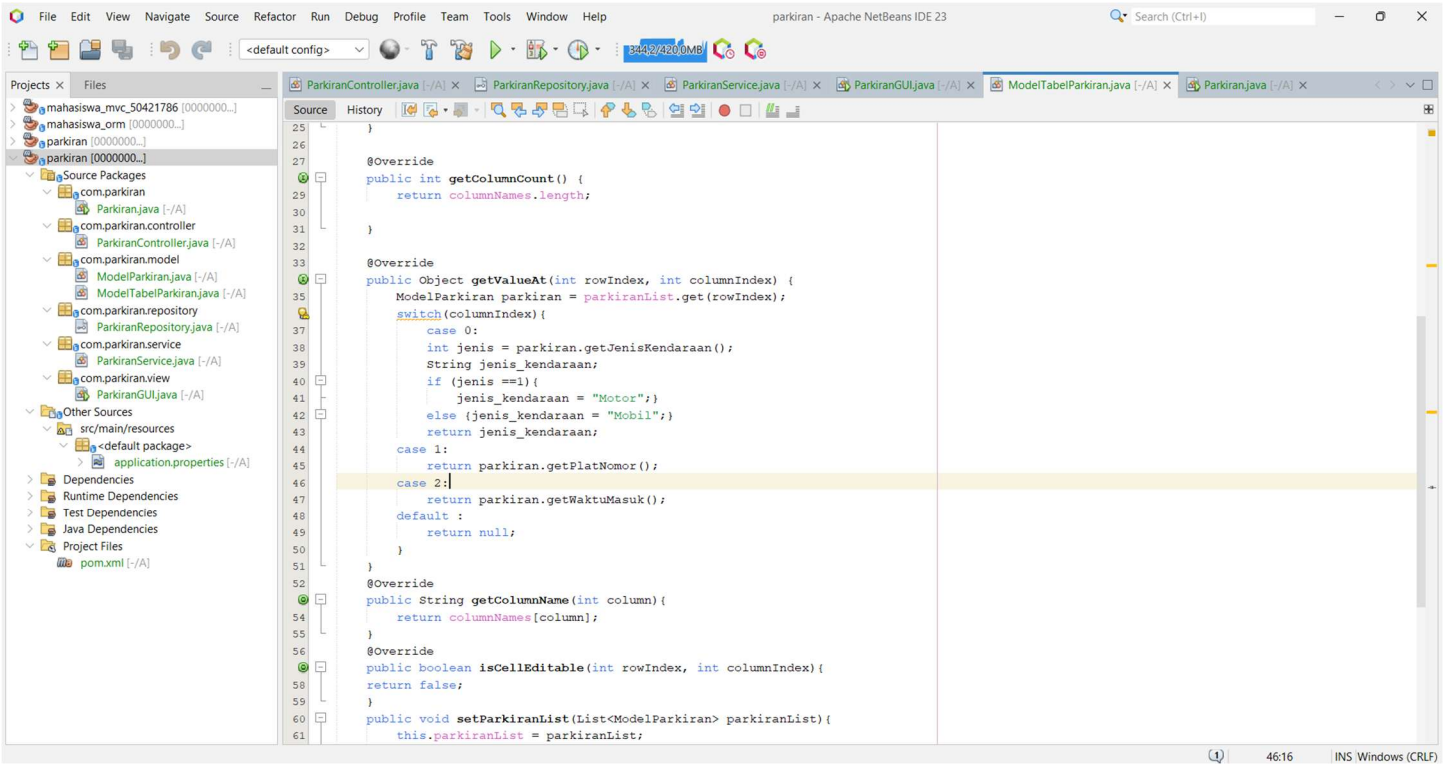


```
        int jenisKendaraan;
        if (motorRadioButton.isSelected()) {
            jenisKendaraan = 1; // Motor
        } else if (mobilRadioButton.isSelected()) {
            jenisKendaraan = 2; // Mobil
        } else {
            JOptionPane.showMessageDialog(this, "Pilih jenis kendaraan!", "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        ModelParkiran parkiran = new ModelParkiran(platNomor, LocalDateTime.now(), jenisKendaraan);

        // Call controller method
        parkiranController.masukParkir(parkiran);

        JOptionPane.showMessageDialog(this, "Kendaraan berhasil masuk parkir!", "Success", JOptionPane.INFORMATION_MESSAGE);

        // Refresh table data
        updateTableData();

        // Clear input fields
        platNomorField.setText("");
        kendaraanGroup.clearSelection();
    }

    private void handleKeluarParkir() {
        String platNomorKeluar = platKeluarField.getText();

        // Call controller method and get the result
        String result = parkiranController.keluarParkir(platNomorKeluar);

        if (result.startsWith("Kendaraan")) {
            JOptionPane.showMessageDialog(this, result, "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog(this, "Biaya Parkir: Rp " + result, "Success", JOptionPane.INFORMATION_MESSAGE);
        }
```

ParkiranGUI.java

- Lapisan GUI berbasis **Swing** untuk antarmuka pengguna.
- Fitur:

  Masuk Parkir: Mengambil input pengguna, seperti plat nomor dan jenis kendaraan, lalu mengirimkan ke controller.
  Keluar Parkir: Mengambil plat nomor kendaraan, meminta controller untuk menghitung biaya parkir, dan menampilkan hasil.
  Tabel Data: Menampilkan semua kendaraan yang sedang parkir.



```
    }

    @Override
    public int getColumnCount() {
        return columnNames.length;

    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        ModelParkiran parkiran = parkiranList.get(rowIndex);
        switch(columnIndex){
            case 0:
            int jenis = parkiran.getJenisKendaraan();
            String jenis_kendaraan;
            if (jenis ==1){
                jenis_kendaraan = "Motor";}
            else {jenis_kendaraan = "Mobil";}
            return jenis_kendaraan;
        case 1:
            return parkiran.getPlatNomor();
        case 2:
            return parkiran.getWaktuMasuk();
        default :
            return null;
        }
    }
    @Override
    public String getColumnName(int column){
        return columnNames[column];
    }
    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex){
    return false;
    }
    public void setParkiranList(List<ModelParkiran> parkiranList){
        this.parkiranList = parkiranList;
```

ModelTabelParkiran.java

- Model tabel custom untuk digunakan di **JTable**.
- Menyediakan data dari List<ModelParkiran> ke tabel GUI.
- Fungsi seperti setParkiranList memungkinkan pembaruan data secara dinamis.

Keunggulan Clean Architecture di Proyek Ini

- **Mudah Dipahami**: Setiap kelas memiliki tanggung jawab yang jelas sesuai lapisannya.
- **Testable**: Logika bisnis di ParkiranService dapat diuji terpisah tanpa melibatkan GUI atau database.
- **Fleksibel**: Perubahan pada lapisan GUI tidak memengaruhi logika bisnis.
- **Reusability**: Jika ingin mengganti GUI Swing dengan REST API, hanya lapisan antarmuka yang perlu diubah.