# **LAPORAN TUGAS BESAR PBO**



**Disusun Oleh :**

- Yakhe Kenneth Sugiharto (230712379)

- Beda Arya Wimala (230712345)

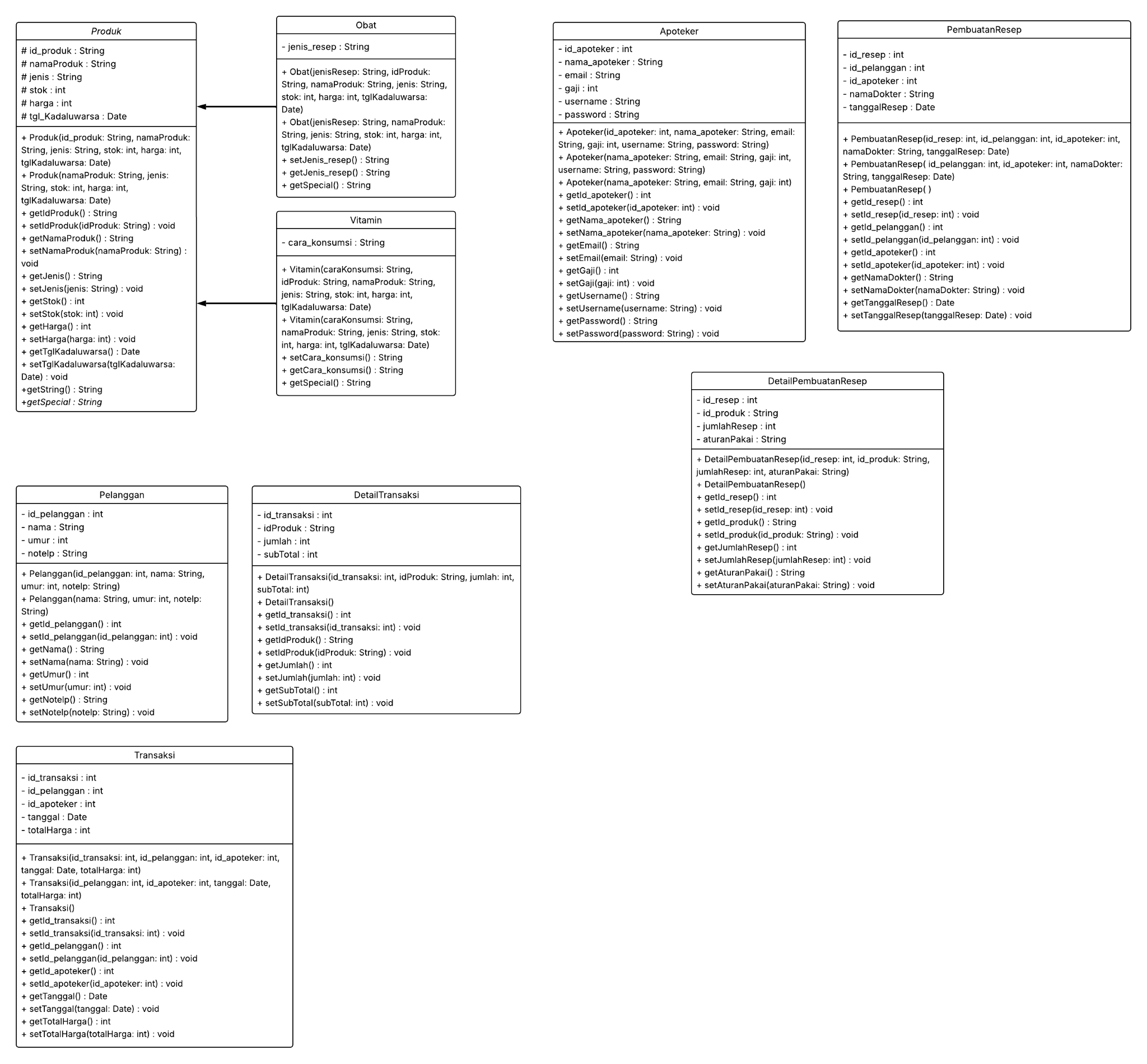
- Michael Lim (230712366)

- Dwi Miftah Estiawan (230712378)

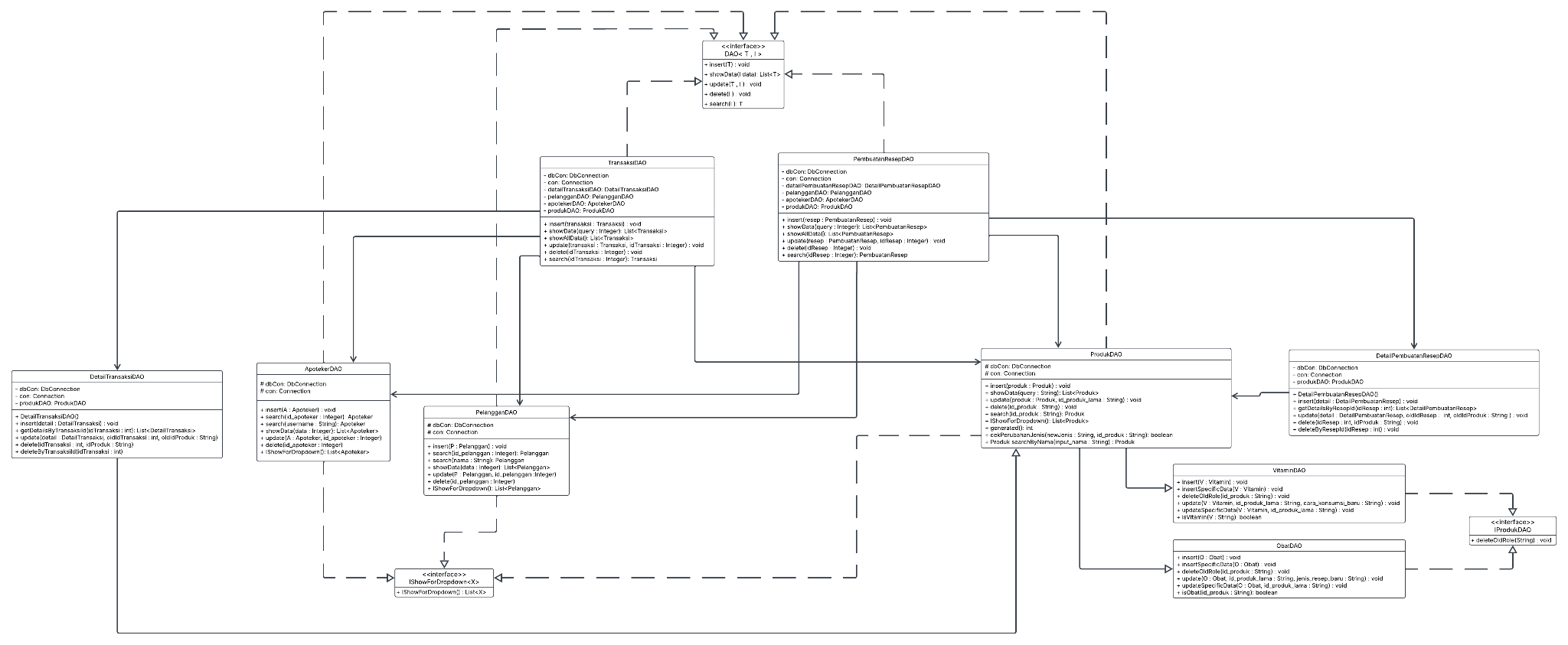
**PROGRAM STUDI INFORMATIKA FAKULTAS TEKNOLOGI INDSUTRI UNIVERSITAS ATMA JAYA YOGYAKARTA TAHUN AKADEMIK 2024/2025**

**CLASS DIAGRAM & Notasi UML**

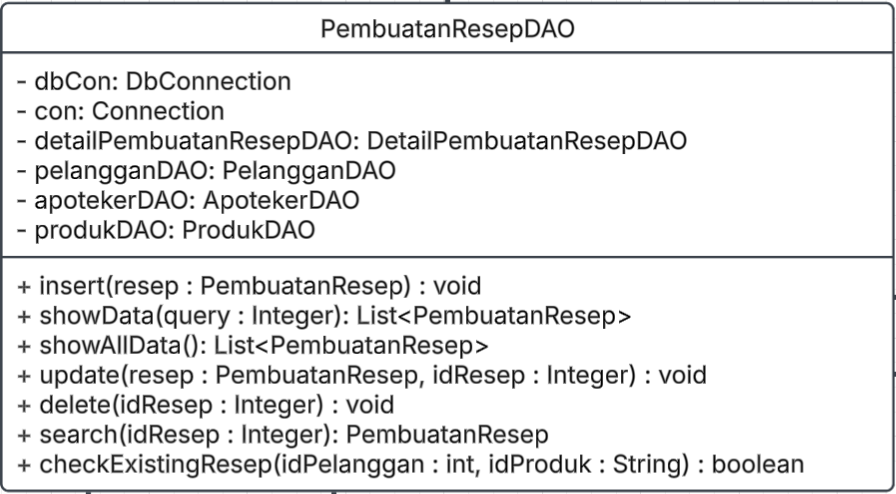
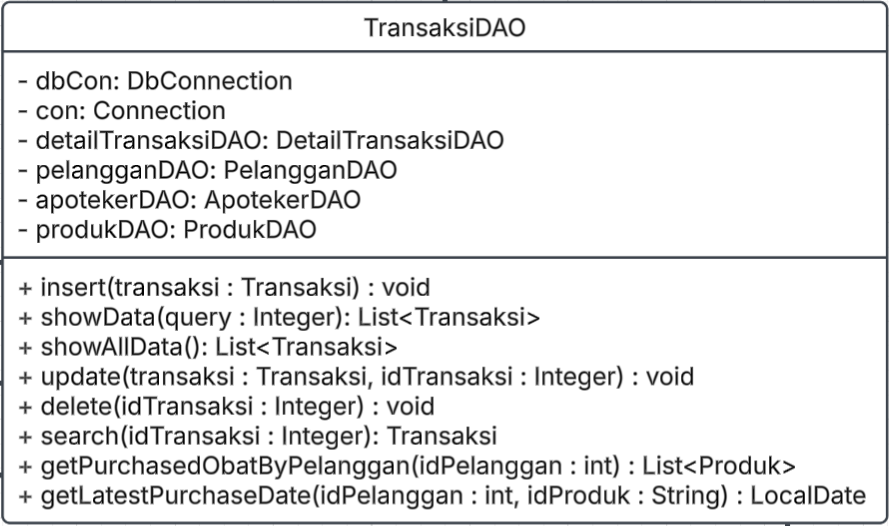
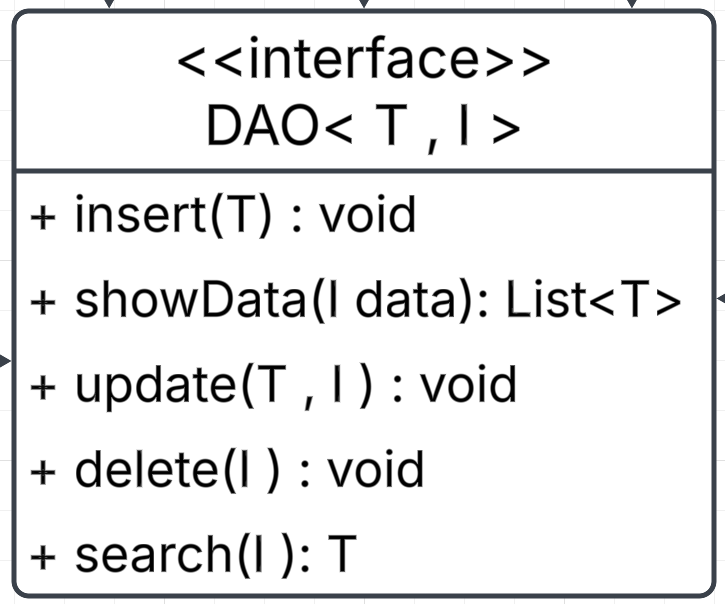
1. **Model**

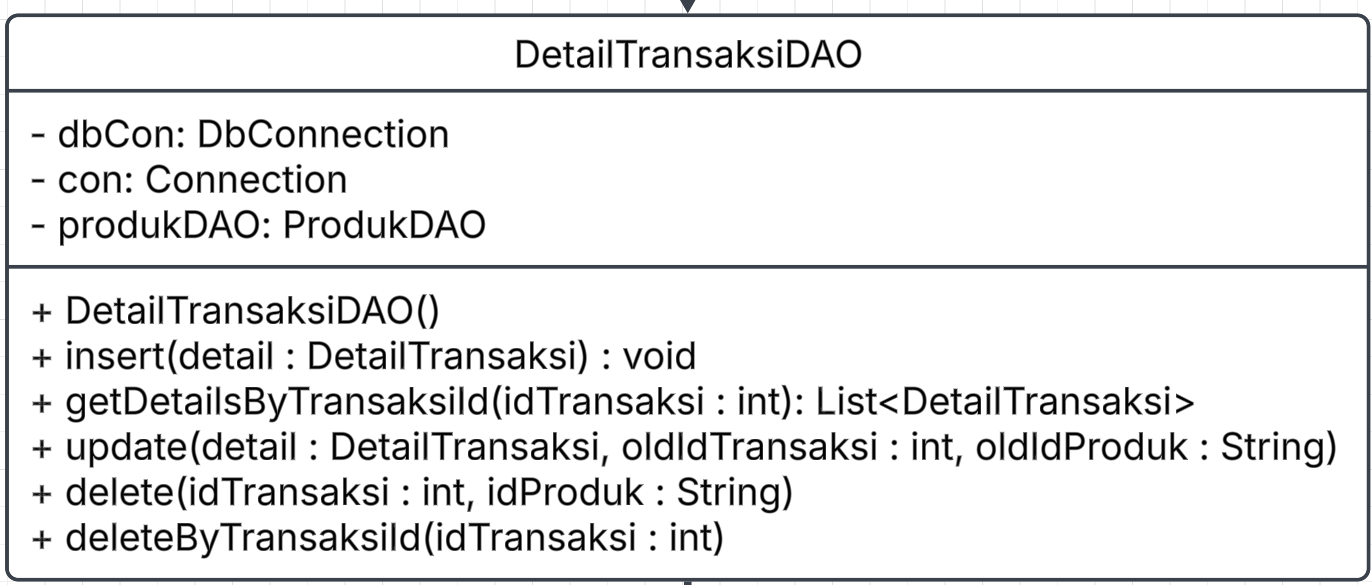


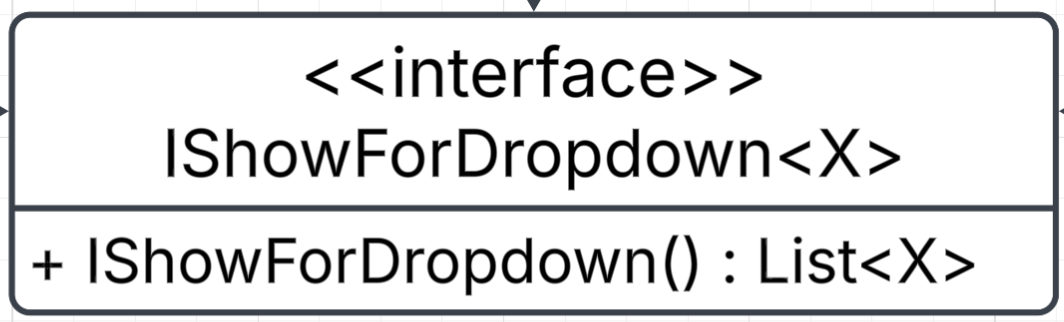
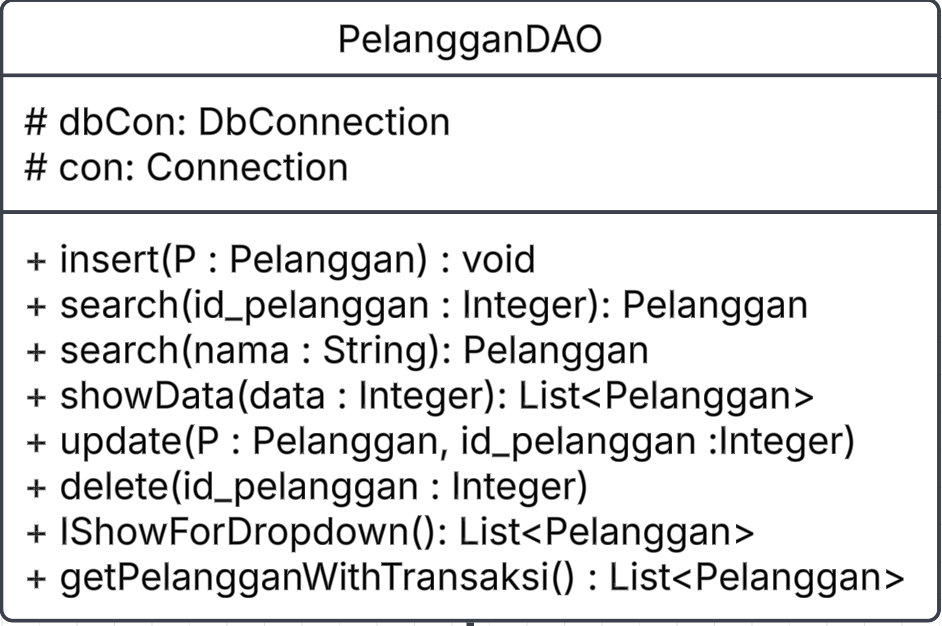
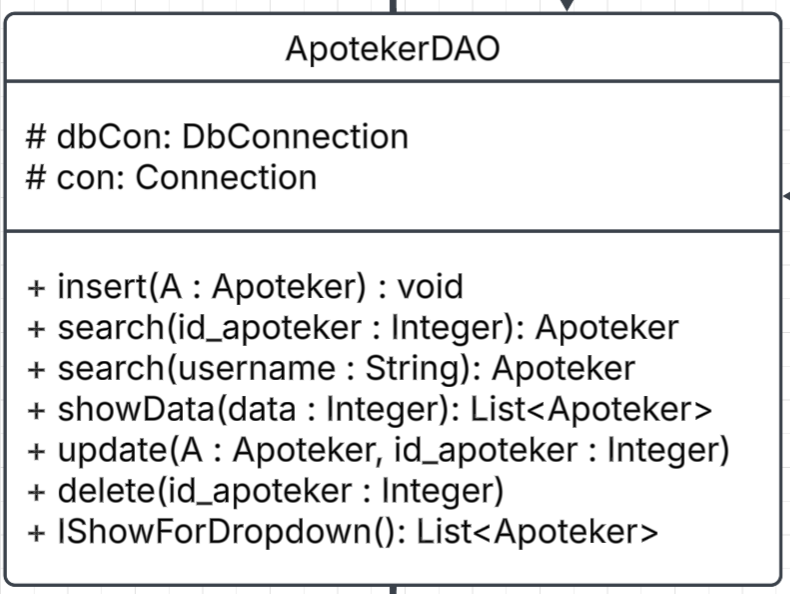
1. **DAO**

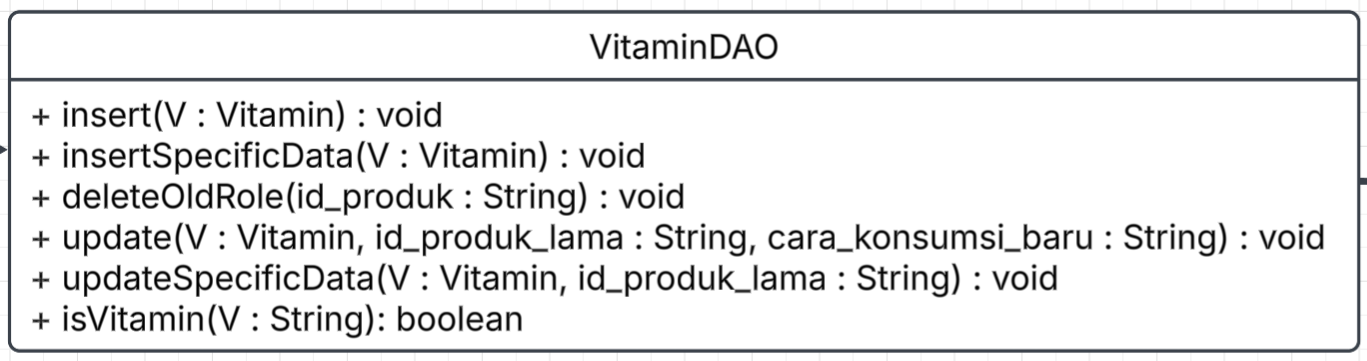
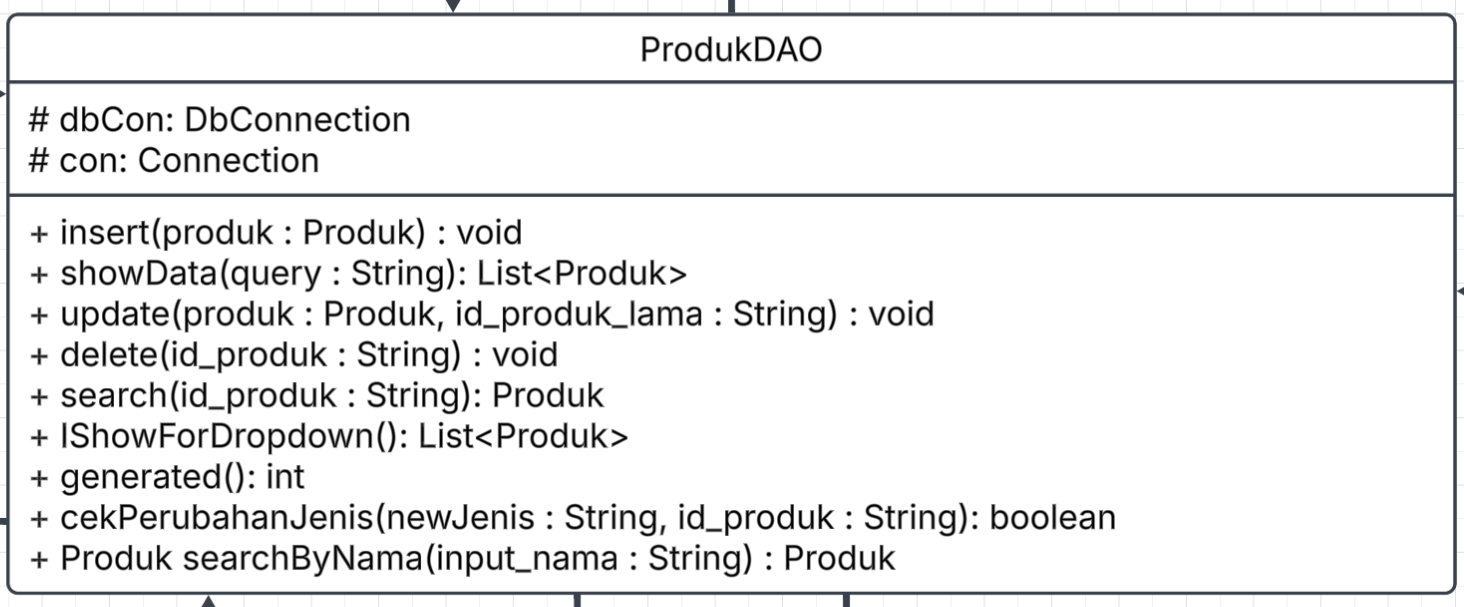


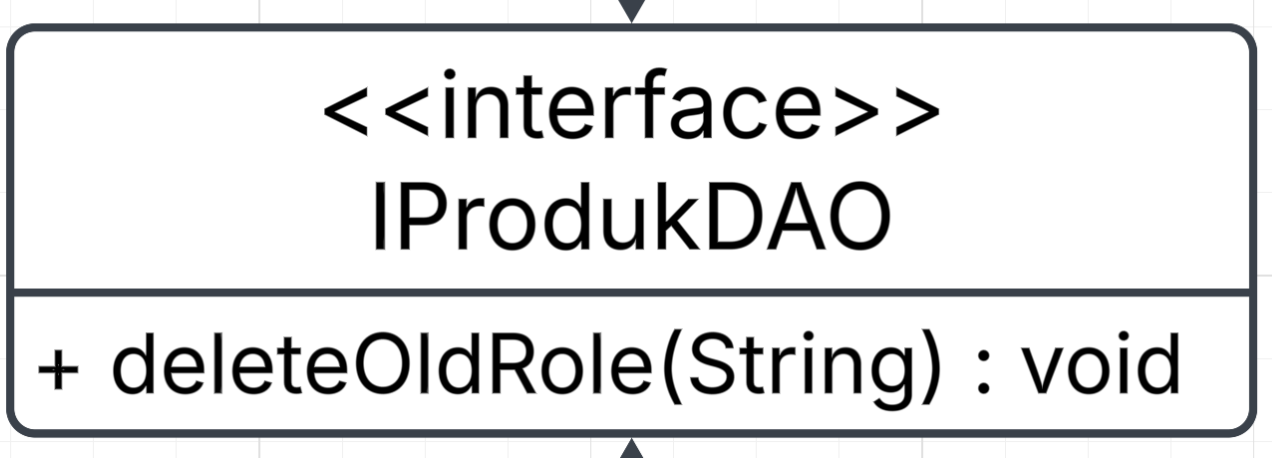
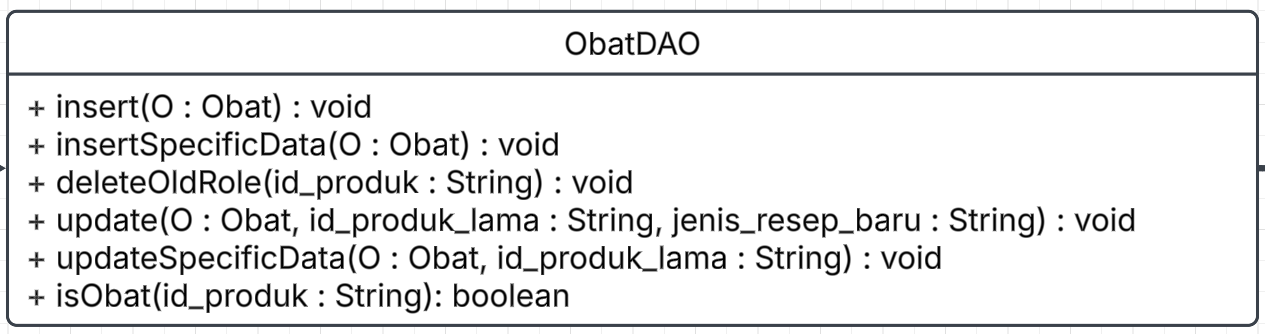
**Rincian tiap tabel :**

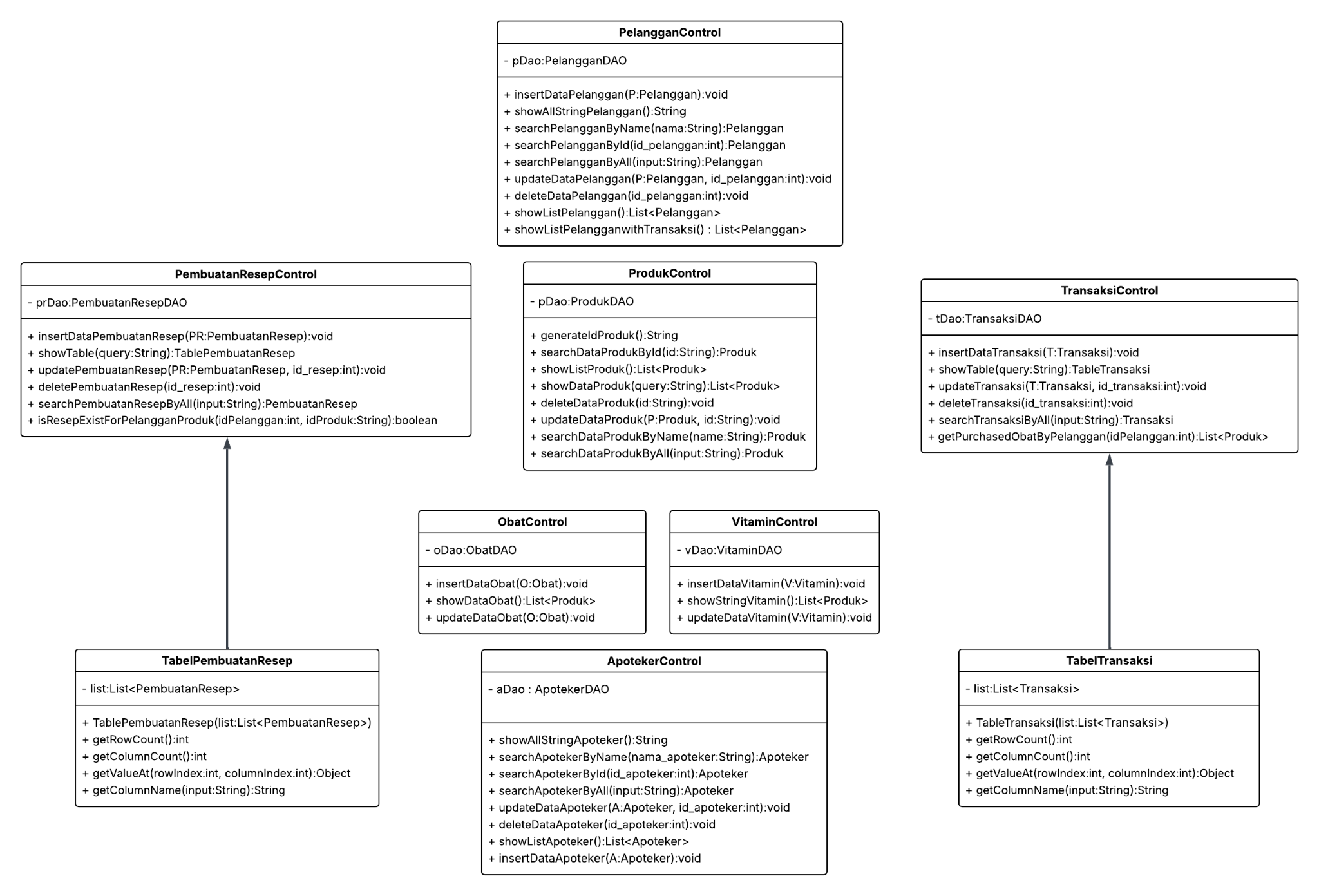




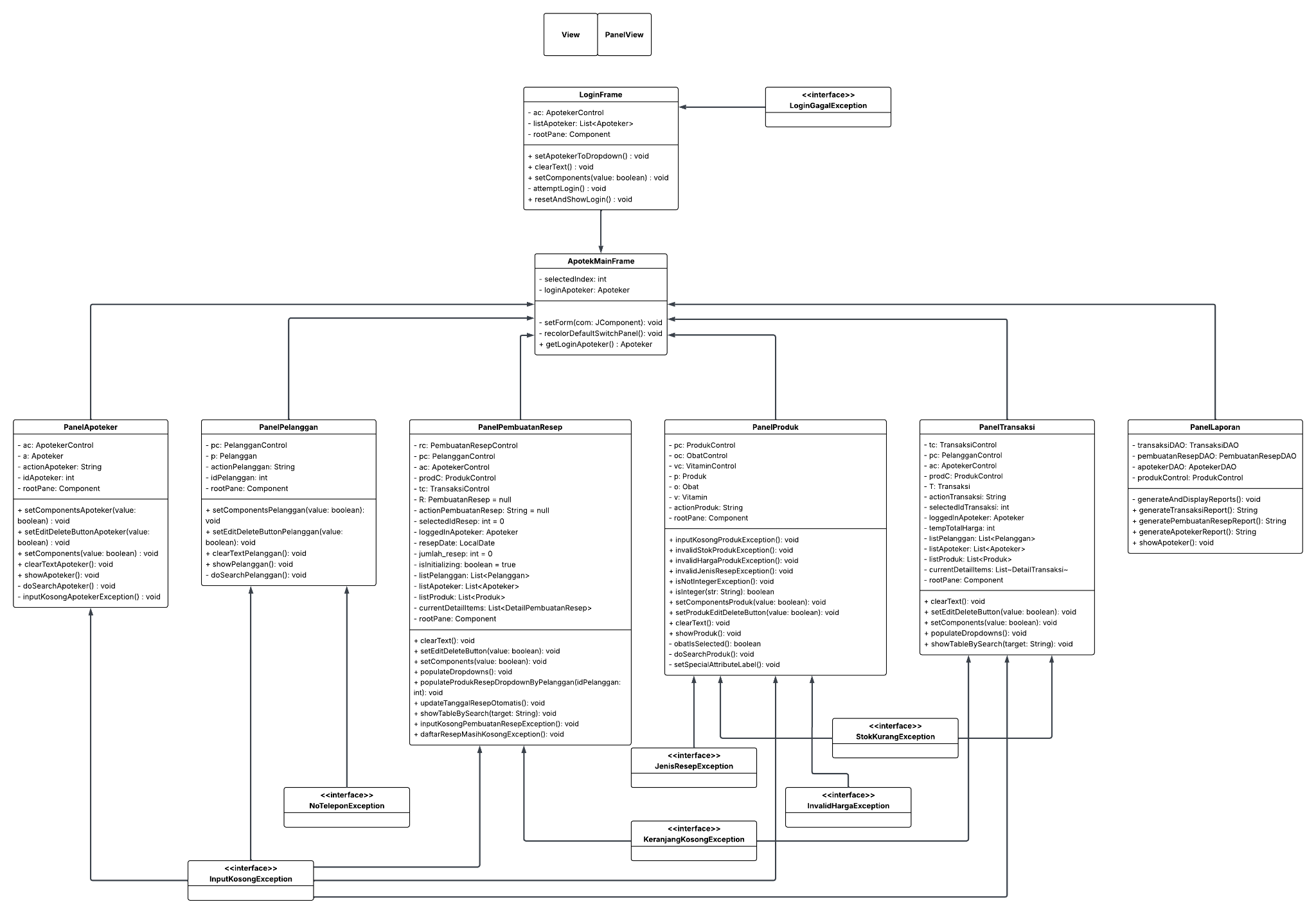




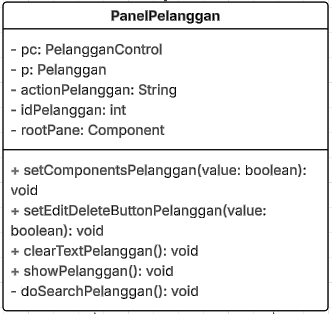
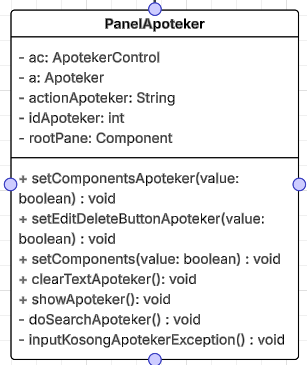


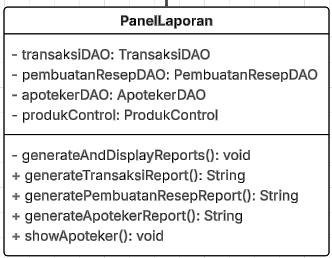
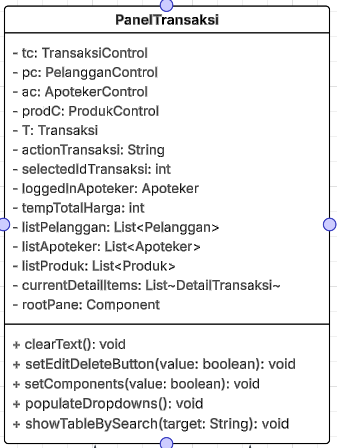
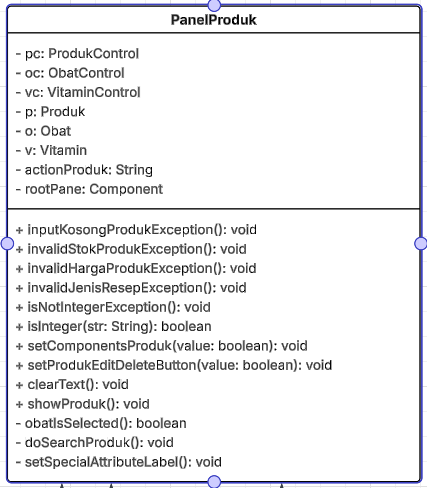
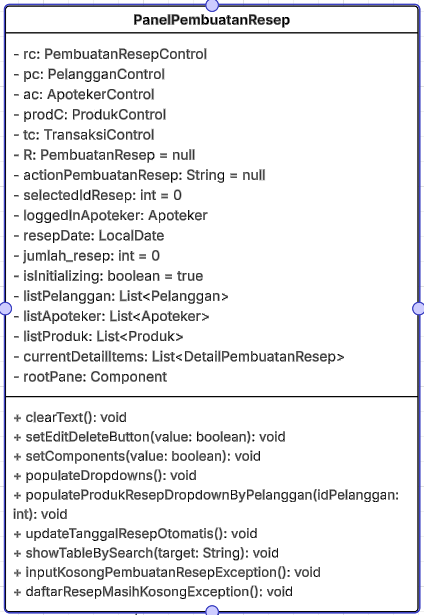
**Controller**

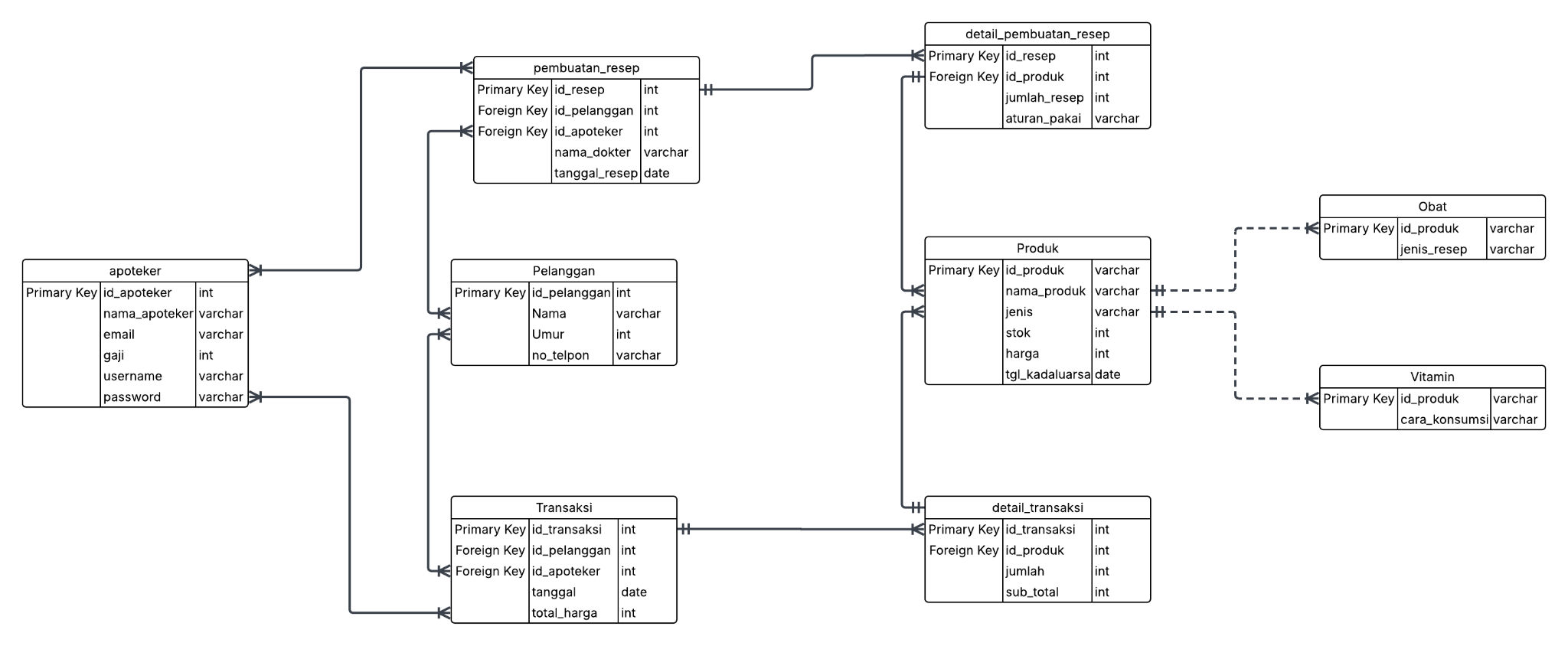
**View & PanelView**

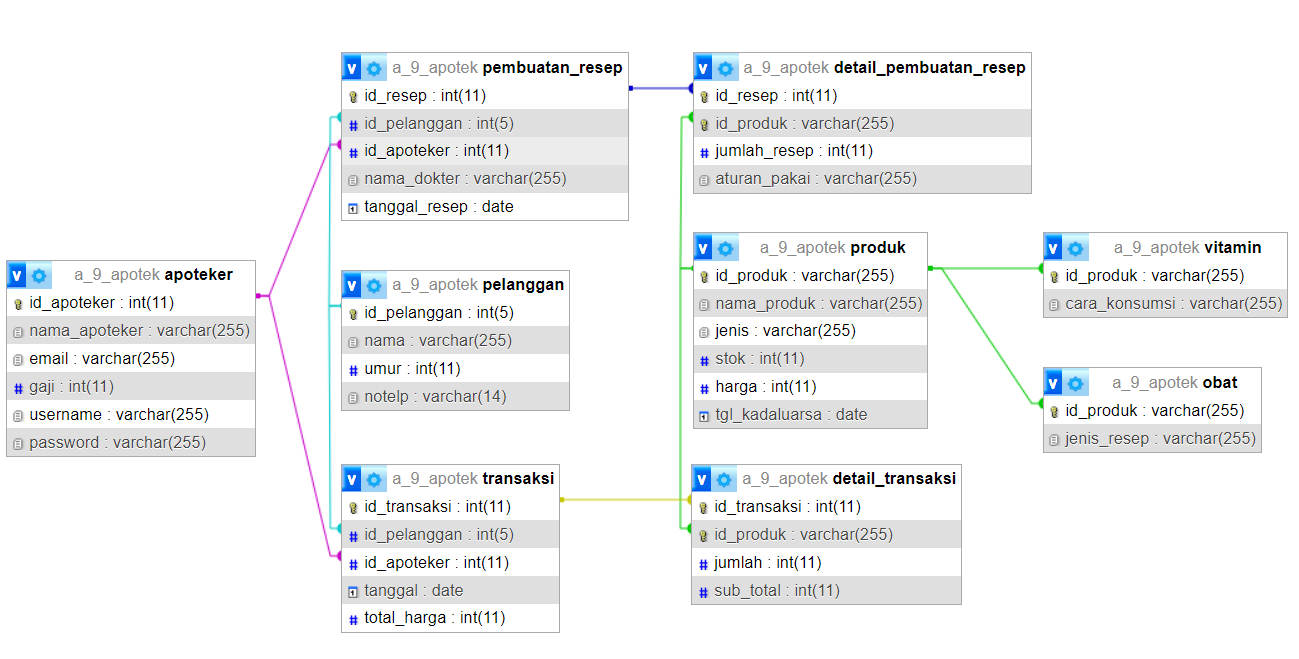
****

**Rincian Tiap Tabel :**

****

****

**ERD Rancangan Data**



**( Designer Database )**

**Penjelasan Singkat Alur Program :**

Program yang kami buat adalah program sistem manajemen untuk sebuah Apotek. Program kami mempunyai **9 Tabel**, dengan 7 tabel entitas kuat, dan 2 tabel entitas lemah

* **Apoteker**
* **Pelanggan**
* **Produk**
* **Obat**
* **Vitamin**
* **Transaksi**
* **PembuatanResep**
* **DetailTransaksi (Entitas lemah)**
* **DetailPembuatanResep (Entitas lemah)**

Kedua tabel entitas lemah tersebut kami gunakan untuk menangani kasus Many-To-Many, dimana setiap Produk bisa berada di banyak Transaksi atau PembuatanResep, dan tiap Transaksi atau PembuatanResep bisa melibatkan banyak Produk.

Program yang kami buat memiliki **3 data master**, yaitu **Apoteker, Pelanggan**, dan **Produk**, dimana ketiga entitas tersebut dapat dilakukan CRUD nya. Transaksi yang dapat dilakukan di sistem ini adalah pembelian produk, dan pembuatan resep. Setiap transaksi tersebut akan ditangani oleh seorang Apoteker.

Relasi terjadi antara entity Produk, Obat, dan Vitamin. Disini Produk berperan sebagai Superclass, dan Vitamin serta Obat berperan sebagai Subclass yang meng-extends dari Produk. Pembelian produk dapat dilakukan secara campur, sesuai dengan yang diinputkan oleh user.

Program kami juga memiliki sistem “**Report**”, dimana laporan ini akan menampilkan :

* **Transaksi pembelian produk** (Total pendapatan dari transaksi, jumlah transaksi per Bulan, Produk terlaris secara Kuantitas
* **Laporan pembuatan Resep** (Jumlah resep yang dibuat oleh tiap Dokter, serta Obat yang paling sering diresepkan)
* **Laporan kinerja Apoteker** (Apoteker yang paling banyak melayani, jumlah layanan per Apoteker)

Ketiga laporan tersebut akan ditampilkan dalam satu JTextArea besar.

Program kami tentunya juga menggunakan **Exception**, sebagai error handlingnya. Excepion - exception tersebut diantaranya :

* **InputKosongException** : menangani inputan yang kosong pada JtextField, ataupun inputan-inputan lainnya
* **InvalidHargaException** : error handling untuk inputan harga produk yang kurang dari 1000 rupiah
* **JenisResepException** : error handling untuk jenis resep yang hanya boleh “Wajib” dan “Bebas”
* **KeranjangKosongException** : digunakan agar tombol “Bayar” dan “BuatResep” tidak dapat diakses, apabila belum ada item yang dimasukkan ke Keranjang saat Transaksi, dan ke daftar pembuatan resep saat PembuatanResep
* **LoginGagalException** : error handling ketika inputan username dan password tidak sesuai (Username diambil dari nama apoteker yang di double, dan Password diambil dari nama apoteker ditambah angka 123 dibelakangnya. Misal nama\_apoteker = Michael, maka username = MichaelMichael, password = Michael123)
* **NoTelponException** : digunakan untuk menangani inputan nomor telepon yang tidak sesuai ketentuan (diawali 0 atau +62)
* **StokKurangException** : digunakan untuk menangani 2 hal, yaitu jika stok produk yang diinputkan < 1, dan jika pembelian produk melebihi sisa stok yang tersedia

**Penjelasan Umum Konsep PBO yang diterapkan dalam Code Program :**

1. Abstraksi

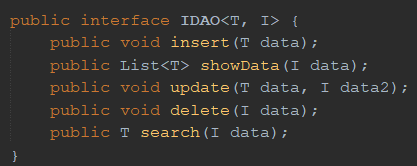
Abstraksi adalah proses menyederhanakan kompleksitas sistem dan menyembunyikan detail implementasi yang rumit. Abstraksi ini dapat diwujudkan melalui kelas abstrak dan interface. Penerapan abstraksi pada kode program dapat dilihat pada kelas Produk pada bagian model dimana kelas Produk adalah kelas abstrak yang mempunyai method abstrak getSpecial(). Abstraksi ini juga dapat ditemui pada ApotekerDAO dimana ApotekerDAO melakukan implementasi terhadap interface IDAO.

1. Kelas Produk

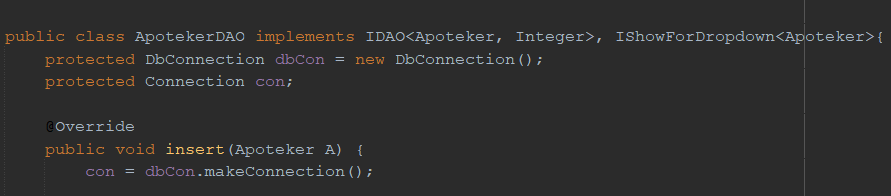




1. Interface IDAO

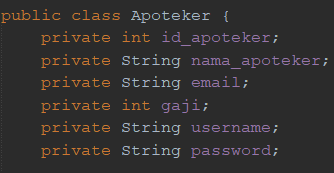
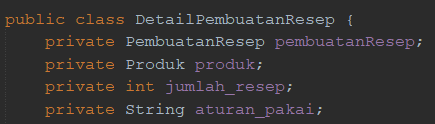


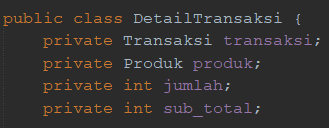
1. ApotekerDAO implentasi IDAO



2. Enkapsulasi

Enkapsulasi adalah suatu cara untuk mengamankan data atribut, method ke dalam satu kelas dan menyembunyikan dari akses yang tidak diizinkan. Atribut dideklarasikan sebagai private di dalam kelas untuk mencegah akses langsung dari luar kelas. Penerapan prinsip enkapsulapsi ini sudah diterapkan pada semua kelas yang ada di bagian model, dengan pengecualian kelas Produk dikarenakan semua atribut di kelas produk perlu diwariskan.

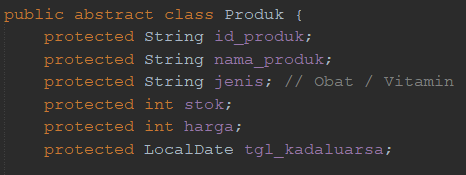


Semua atribut pada gambar tersebut memiliki visibilitas private yang berarti nilai atribut tersebut tidak dapat diakses dan diubah secara langsung dari luar. Nilai - nilai tersebut hanya dapat diubah dengan menggunakan method public yang ada di tiap kelas seperti getUmur() dan setUmur(int umur).

3. Pewarisan

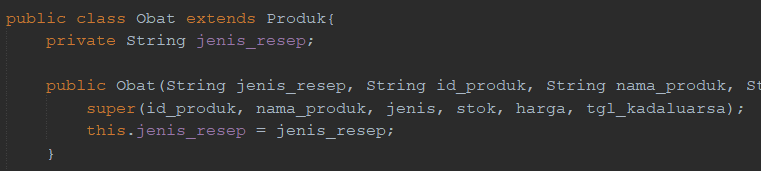
Pewarisan adalah sebuah relasi “is-a” yang terjadi antara kelas induk (superclass) dan kelas anak (subclass). Konsep ini dapat terjadi apabila kelas anak dapat mewarisi sifat kelas induknya. Penggunaan pewarisan dapat memperbolehkan kelas turunan untuk menggunakan atribut dan method dari kelas induknya. Konsep pewarisan ini menggunakan visibilitas protected agar dapat diakses oleh kelas turunannya. Penerapan pewarisan dalam kode program dapat dilihat pada kelas produk, obat, dan vitamin pada bagian model dimana obat adalah extend dari produk dan vitamin adalah extend dari produk.

1. Superclass :

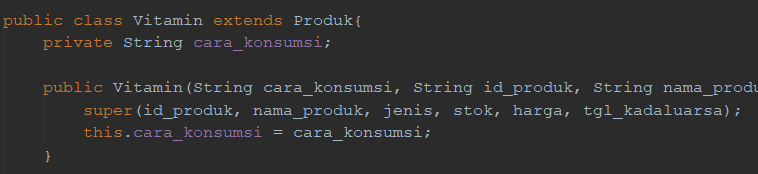


1. Subclass :

**Obat**



**Vitamin**



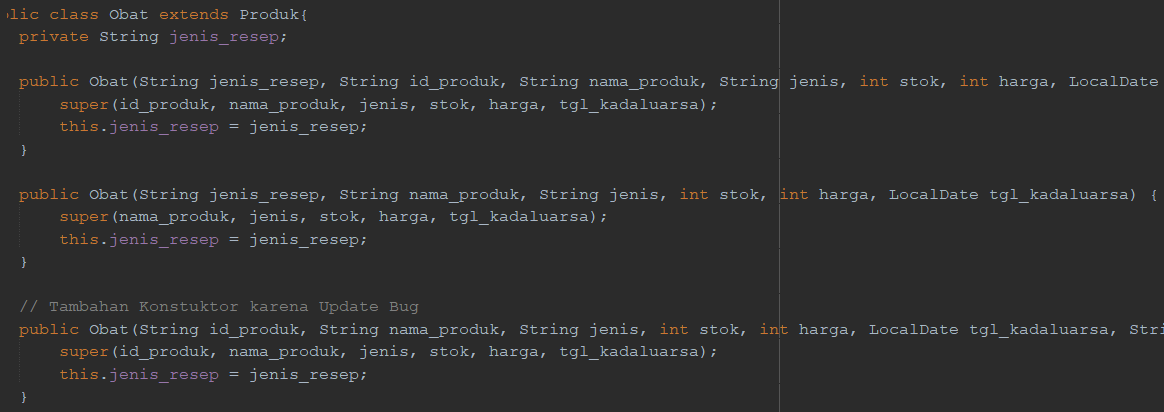
Kelas obat dan vitamin mewarisi semua atribut dari kelas induk Produk yang memiliki visibilitas protected. Pemanggilan super pada konstruktor adalah cara subclass untuk memanggil nilai atribut milik superclass.

4. Polimorfisme

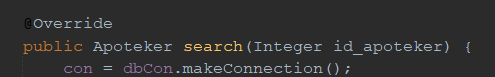
Polimorfisme adalah kemampuan beberapa objek untuk memberi cara respon yang berbeda - beda terhadap permintaan yang sama. Polimorfisme terbagi menjadi 2 jenis, yaitu polimorfisme statis dan dinamis.

Polimorfisme statis adalah proses dimana sebuah method dipanggil saat compiling. Polimorfisme statis dapat dibagi menjadi 2 jenis, yaitu overloading methods dan overloading constructor. Overloading methods adalah jenis polimorfisme yang mendeklarasikan method dengan nama yang sama tetapi memiliki isi parameter yang berbeda - beda. Overloading constructor adalah jenis polimorfisme statis yang mendeklarasikan lebih dari 1 constructor. Tiap constructor memiliki isi parameter yang berbeda - beda.

Dalam kode pemrograman, konsep polimorfisme statis ini sudah diterapkan pada hampir semua kelas model dimana kelas - kelas tersebut menggunakan overloading constructors.



Terlihat pada gambar tersebut, terdapat 3 constructor dimana urutan dan jumlahnya berbeda. Dikarenakan terdapat perbedaan urutan dan jumlah, hal ini sudah cukup untuk memenuhi syarat polimorfisme statis, overloading constructors. Untuk overloading method, terdapat pada kelas ApotekerDAO pada bagian DAO dimana terdapat 2 method search dengan isi parameter yang berbeda.

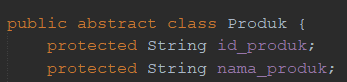




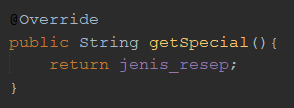
Jenis polimorfisme yang kedua adalah polimorfisme dinamis. Polimorfisme dinamis adalah proses dimana sebuah method dipanggil saat runtime, bukan saat compiling. Polimorfisme dinamis dibagi menjadi 2 jenis, yaitu overriding method pada kelas konkret dan overriding method pada kelas abstrak. Kedua jenis ini menggunakan konsep pewarisan untuk menerapkannya. Overriding method kelas konkret bertujuan untuk memperluas method yang sudah ada dari kelas induk, sementara overriding method kelas abstrak, subclass harus melakukan overriding terhadap semua method abstract yang ada pada superclass abstrak. Metode pada kelas induk kosong dan tidak menyediakan detail implementasinya.

Pada kode program, overriding methods kelas abstrak ini sudah diterapkan pada kelas produk, obat dan vitamin dimana produk adalah kelas abstrak dan memiliki method abstract yang telah di override oleh obat dan vitamin.

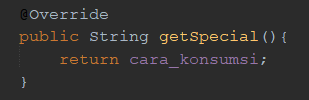
1. Kelas Produk:



1. Kelas Obat:



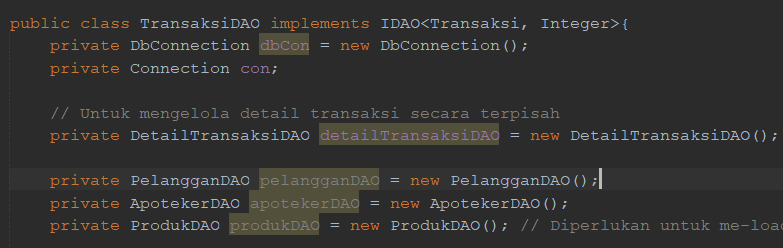
1. Kelas Vitamin:



5. Komunikasi Antar Objek

Dalam sistem yang kompleks, pasti memiliki sebuah relasi antara kelas satu dengan yang lain. Relasi memiliki beberapa jenis, yaitu pewarisan, agregasi, komposisi, asosiasi, dependensi. Agregasi adalah hubungan dimana objek tersusun dari objek lain dan memiliki sifat yang lemah karena objek penyusun dapat berdiri sendiri walaupun objek agregat dihapus. Komposisi adalah hubungan yang dimana objek tidak bisa berdiri sendiri tanpa objek penyusunnya dan memiliki sifat yang kuat. Asosiasi adalah hubungan antara objek dari kelas yang berbeda dan objek saling berhubungan satu sama lain. Dependensi adalah ketergantungan antar kelas untuk berfungsi dengan benar.

Penerapan relasi pada program dapat dilihat pada kelas TransaksiDAO pada bagian DAO. TransaksiDAO memiliki relasi komposisi dengan DetailTransaksiDAO, PelangganDAO, ApotekerDAO, ProdukDAO.



TransaksiDAO juga memiliki dependensi terhadap objek dari Transaksi.



Contoh relasi lainnya adalah pewarisan yang ada terjadi antara kelas produk, obat, vitamin pada model dan produkDAO, obatDAO, vitaminDAO pada DAO.



