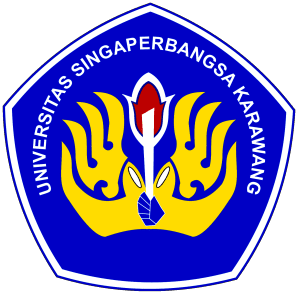
**LAPORAN**

**UJIAN AKHIR SEMESTER**

**PEMROGRAMAN BERORIENTASI OBJEK**

**“Sistem Manajemen Apotek”**

Dosen Pengampu : Taufik Ridwan, M.T.



KELOMPOK VIII

4A – SISTEM INFORMASI

|  |  |
| --- | --- |
| 1. Faisal Anwar | 2310631250052 |
| 1. Faisal Hidayat Sukma | 2310631250053 |
| 1. Rafid Faiz Putra | 2310631250026 |
| 1. Ulin Nuha | 2310631250036 |

**FAKULTAS ILMU KOMPUTER**

**UNIVERSITAS SINGAPERBANGSA KARAWANG**

**2025**

# DAFTAR ISI

[DAFTAR ISI 1](#_Toc199428703)

[**1.** **LATAR BELAKANG** 2](#_Toc199428704)

[**2.** **RUMUSAN MASALAH** 2](#_Toc199428705)

[**3.** **TUJUAN MASALAH** 2](#_Toc199428706)

[**4.** **FITUR – FITUR APLIKASI** 3](#_Toc199428707)

[**5.** **PENJELASAN KONSEP OOP DI PROGRAM** 5](#_Toc199428708)

[**6.** **UNIFIED MODELLING LANGUAGE (UML)** 9](#_Toc199428709)

[**7.** **IMPLEMENTASI KODE PROGRAM & PENGUJIAN** 10](#_Toc199428710)

[**8.** **KESIMPULAN** 11](#_Toc199428711)

## **LATAR BELAKANG**

Perkembangan teknologi informasi yang pesat telah mendorong berbagai sektor usaha untuk melakukan digitalisasi dalam sistem operasionalnya, tidak terkecuali di bidang kesehatan. Salah satu aspek penting dalam pelayanan kesehatan adalah pengelolaan apotek. Apotek tidak hanya berfungsi sebagai tempat penjualan obat, tetapi juga menjadi bagian penting dalam mendukung proses penyembuhan pasien melalui penyediaan obat yang tepat, cepat, dan akurat.

Namun, dalam praktiknya, banyak apotek masih mengelola data transaksi, stok obat, dan data pelanggan secara manual. Hal ini menimbulkan berbagai permasalahan, seperti keterlambatan pelayanan, kesalahan pencatatan, dan kesulitan dalam pelacakan data stok obat. Oleh karena itu, dibutuhkan sistem manajemen apotek yang dapat mengelola semua aktivitas tersebut secara terkomputerisasi.

Melalui mata kuliah Pemrograman Berorientasi Objek (PBO), mahasiswa dituntut untuk mampu merancang dan mengimplementasikan solusi perangkat lunak yang berbasis objek. Dengan memanfaatkan konsep-konsep OOP seperti enkapsulasi, pewarisan, dan polimorfisme, sistem manajemen apotek yang dikembangkan diharapkan dapat memberikan kemudahan dalam pengelolaan data dan meningkatkan efisiensi kerja apoteker.

## **RUMUSAN MASALAH**

1. Bagaimana merancang sistem manajemen apotek menggunakan pendekatan pemrograman berorientasi objek?
2. Bagaimana sistem ini dapat membantu pengelolaan data obat, transaksi penjualan, dan stok barang?
3. Bagaimana sistem dapat memastikan integritas dan akurasi data dalam setiap proses operasional apotek?

## **TUJUAN MASALAH**

1. Mengembangkan sistem manajemen apotek berbasis desktop yang terstruktur dan terorganisir menggunakan prinsip-prinsip OOP.
2. Mempermudah pengelolaan data obat, transaksi penjualan, dan stok barang melalui sistem terkomputerisasi.
3. Meningkatkan efisiensi dan akurasi dalam operasional apotek dengan mengurangi pekerjaan manual yang rawan kesalahan.

## **FITUR – FITUR APLIKASI**

Berikut ini adalah beberapa fitur yang ada pada program manajemen apotek yang ada pada sistem kami :

1. Select Table

Aplikasi menyediakan pilihan untuk memilih dan mengelola berbagai tabel utama yang berkaitan dengan kegiatan operasional apotek. Tabel-tabel yang dapat dipilih dan dikelola antara lain:

1. **detail\_transaksi\_penjualan**, berisi rincian setiap transaksi penjualan.
2. **kategori\_obat**, berisi data kategori atau golongan dari obat.
3. **obat**, menyimpan data obat secara detail (nama, stok, harga, dll).
4. **supplier** – memuat informasi tentang pemasok obat.
5. **transaksi\_penjualan**, mencatat informasi umum dari setiap transaksi penjualan.
6. Refresh Table

Setiap tampilan data pada aplikasi dilengkapi dengan fitur **Refresh** yang berfungsi untuk memperbarui isi tabel secara otomatis. Fitur ini memungkinkan pengguna melihat data terbaru setelah melakukan penambahan, pengubahan, atau penghapusan, tanpa harus menutup dan membuka ulang aplikasi. Dengan demikian, konsistensi tampilan data tetap terjaga dan pengguna dapat bekerja lebih efisien.

1. Add Record

Pengguna dapat menambahkan data baru ke dalam setiap tabel yang tersedia. Misalnya:

1. Menambah data obat baru ke dalam sistem.
2. Menambahkan transaksi penjualan yang baru.
3. Memasukkan data supplier atau kategori obat.

Fitur ini merupakan bagian penting dalam proses input data untuk kebutuhan operasional harian apotek.

1. Edit Record

Aplikasi menyediakan kemampuan untuk **mengedit/memperbarui** data yang sudah tersimpan dalam database. Dengan fitur ini, pengguna dapat memperbaiki data yang salah atau mengubah informasi tertentu, seperti harga obat, jumlah stok, atau detail transaksi.

1. Delete Redord

Fitur ini memungkinkan pengguna untuk **menghapus data** yang sudah tidak diperlukan atau salah input. Penghapusan dapat dilakukan pada data di semua tabel, seperti obat yang sudah tidak tersedia, supplier yang tidak lagi aktif, atau transaksi yang tidak valid.

1. Generate Receipt

Khusus pada tabel detail\_transaksi\_penjualan, aplikasi dilengkapi dengan fitur **Generate Receipt**, yang berfungsi untuk mencetak struk atau bukti transaksi penjualan. Fitur ini sangat penting dalam memberikan dokumentasi resmi kepada pelanggan sekaligus menyimpan riwayat transaksi bagi pihak apotek.

## **PENJELASAN KONSEP OOP DI PROGRAM**

1. Class dan Object

**public** **class** j\_apotek **extends** JFrame {

*// Class utama yang merepresentasikan aplikasi apotek*

}

* Class j\_apotek: Merupakan blueprint untuk membuat object aplikasi manajemen apotek
* Object creation: Dibuat melalui new j\_apotek() di method main()

1. Inheritance (Pewarisan)

*// Inheritance dari JFrame*

**public** **class** j\_apotek **extends** JFrame

*// Inner class inheritance*

**private** **class** ModernScrollBarUI **extends** javax.swing.plaf.basic.BasicScrollBarUI

* Class j\_apotek mewarisi semua method dan property dari JFrame
* Dapat menggunakan method seperti setTitle(), setSize(), setDefaultCloseOperation() tanpa menulis ulang
* ModernScrollBarUI mewarisi fungsionalitas scrollbar dan hanya override yang diperlukan

1. Enkapsulasi

// Private fields - data tersembunyi dari luar

private static final String DB\_URL = "jdbc:mysql://localhost:3306/apotek";

private static final String USER = "root";

private static final String PASSWORD = "";

private Connection connection;

private JTable dataTable;

private DefaultTableModel tableModel;

// Private methods - implementasi detail tersembunyi

private void connectToDatabase() { ... }

private void loadTableData(String tableName) { ... }

private JButton createModernButton(String text, Color backgroundColor) { ... }

* **Data hiding**: Field database credentials dan UI components di-private
* **Method hiding**: Helper methods seperti createModernButton() tidak perlu diakses dari luar
* **Controlled access**: Hanya method public yang bisa diakses dari luar class

1. Polymorphism
2. Method Overriding

// Override method dari BasicScrollBarUI

@Override

protected void configureScrollBarColors() {

this.thumbColor = BORDER\_COLOR;

this.trackColor = BACKGROUND\_COLOR;

}

@Override

protected JButton createDecreaseButton(int orientation) {

return createZeroButton();

}

// Override dari DefaultTableCellRenderer

@Override

public Component getTableCellRendererComponent(JTable table, Object value,

boolean isSelected, boolean hasFocus, int row, int column) {

// Custom implementation

}

1. Implementasi Interface

// ActionListener interface

refreshButton.addActionListener(e -> loadTableData((String) tableSelector.getSelectedItem()));

addButton.addActionListener(e -> addRecord((String) tableSelector.getSelectedItem()));

// MouseListener interface (anonymous class)

button.addMouseListener(new MouseAdapter() {

@Override

public void mouseEntered(MouseEvent e) {

button.setBackground(backgroundColor.darker());

}

@Override

public void mouseExited(MouseEvent e) {

button.setBackground(backgroundColor);

}

});

1. Abstraction

// Abstraksi dalam UI creation

private JPanel createHeaderPanel() { ... }

private JPanel createTablePanel() { ... }

private JPanel createButtonPanel() { ... }

// Abstraksi database operations

private void addRecord(String tableName) { ... }

private void editRecord(String tableName) { ... }

private void deleteRecord(String tableName) { ... }

* User tidak perlu tahu detail implementasi UI creation
* Database operations diabstraksi menjadi method yang mudah dipahami
* Complex styling logic disembunyikan dalam helper methods

1. Konstruktor dan Inisialization

public j\_apotek() {

// Constructor dengan initialization sequence

setupModernUIDefaults();

setTitle("Apotek Database Manager");

setSize(1000, 700);

// ... more initialization

initModernUI();

connectToDatabase();

loadTableData((String) tableSelector.getSelectedItem());

setVisible(true);

}

1. Static Members

// Static constants - shared across all instances

private static final Color PRIMARY\_COLOR = new Color(74, 144, 226);

private static final Color SECONDARY\_COLOR = new Color(108, 117, 125);

private static final String DB\_URL = "jdbc:mysql://localhost:3306/apotek";

// Static method - entry point

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new j\_apotek());

}

1. Compotition

// Class j\_apotek "has-a" relationship dengan components

private JComboBox<String> tableSelector;

private JTable dataTable;

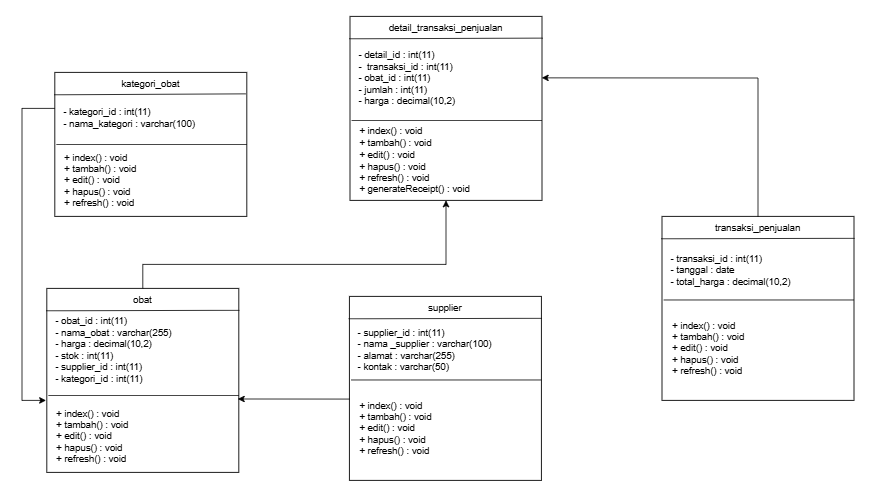
private JScrollPane scrollPane;

private DefaultTableModel tableModel;

private Connection connection;

## **UNIFIED MODELLING LANGUAGE (UML)**

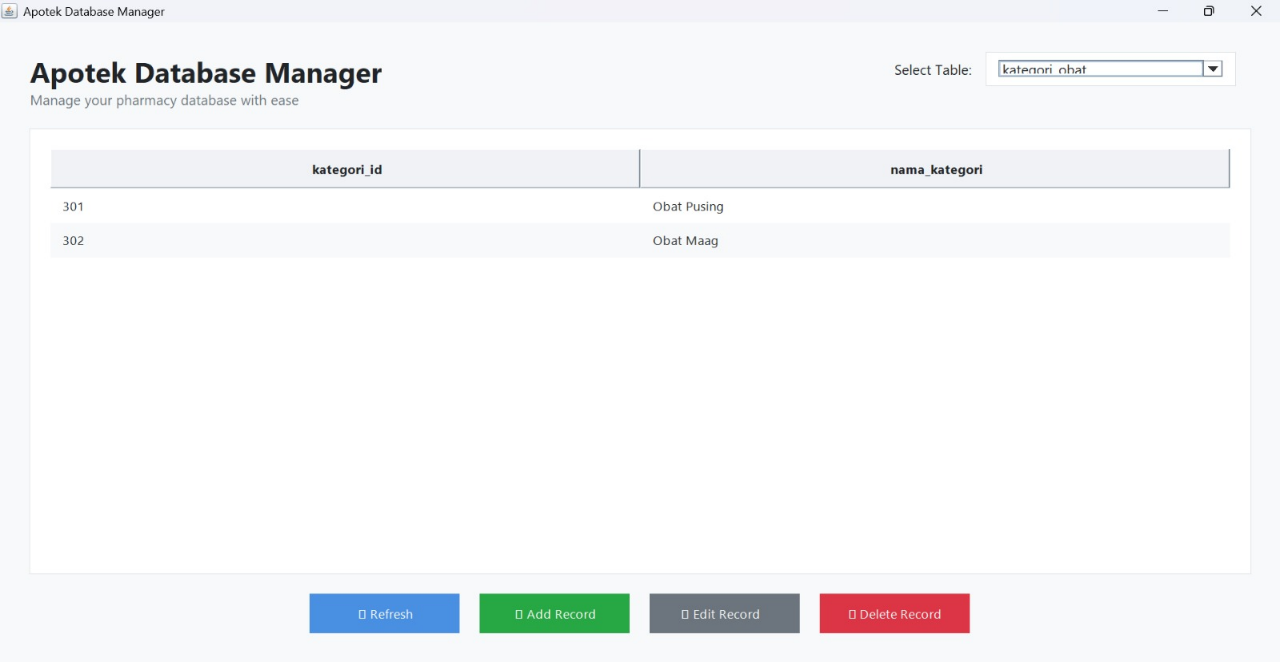
1. Class Diagram



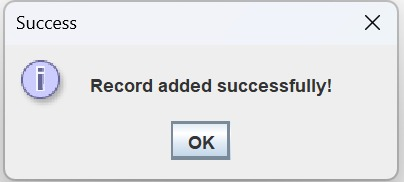
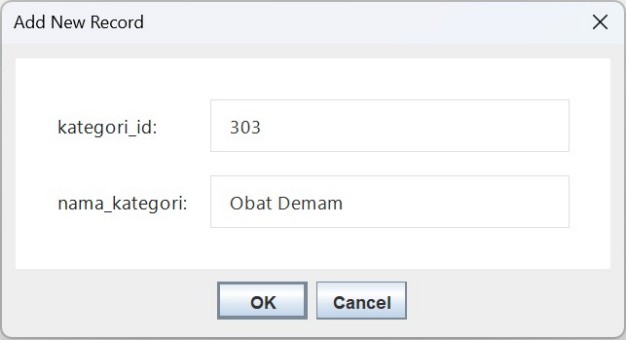
## **IMPLEMENTASI KODE PROGRAM & PENGUJIAN**

Berikut adalah tampilan dan gambaran bagaimana program kami berjalan

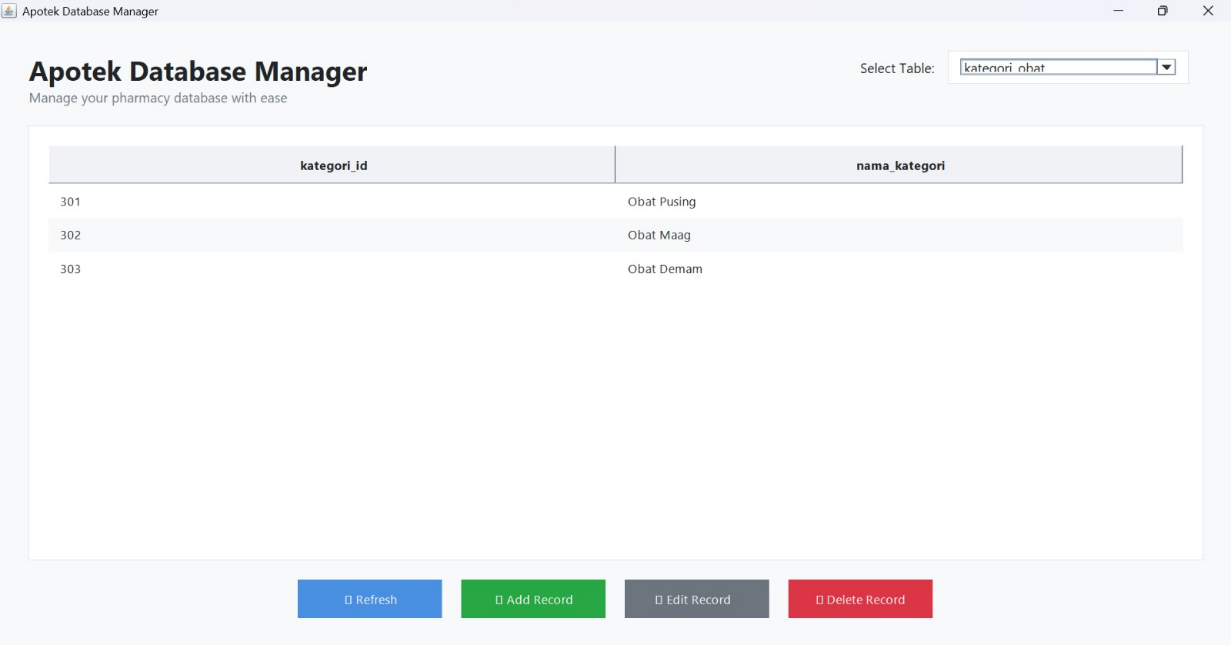
Halaman awal Ketika ingin menambah data di kategori obat



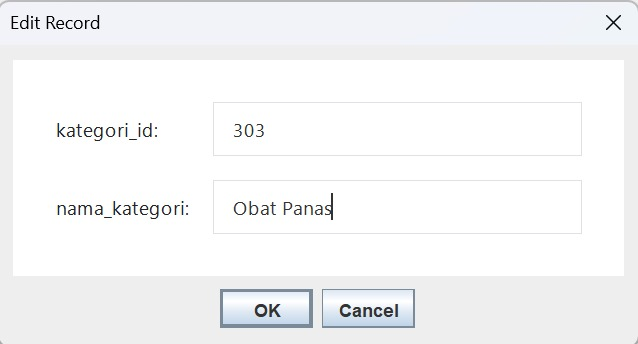
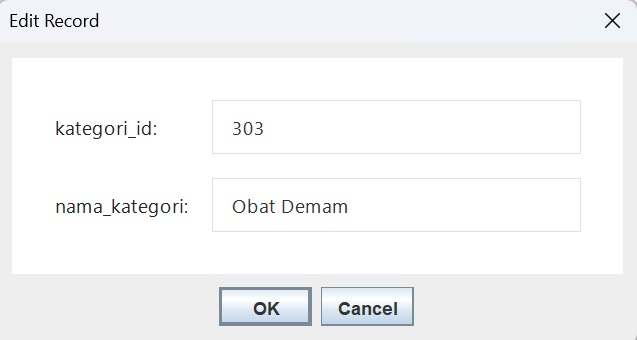
Menambahkan kategori obat dan jika berhsil di tambahkan



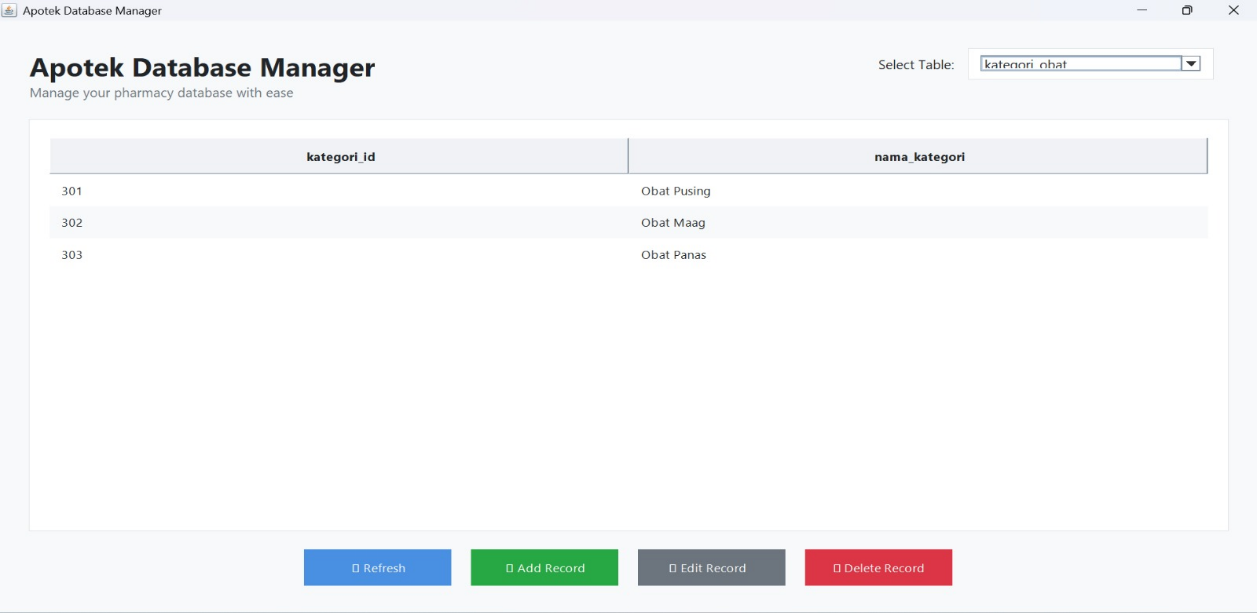
Halaman setelah data dimasukan dan ketika ingin mengedit data



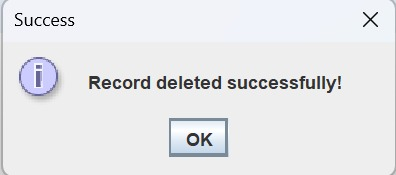
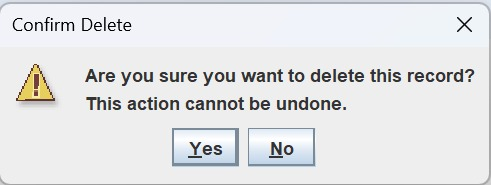
Sebagai contoh, mengubah obat demam ke obat panas dan berhasil di ubah



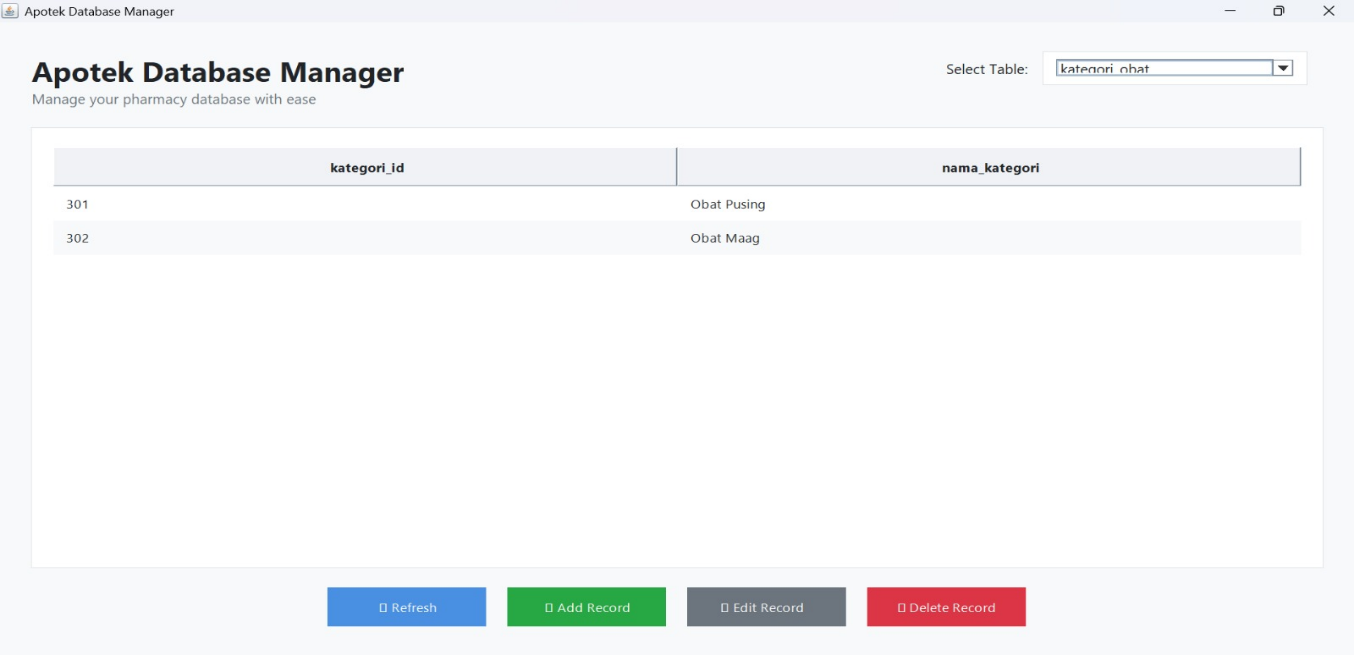
Halaman ketika sudah di ubah/update dan halaman awal ketika akan menghapus

s

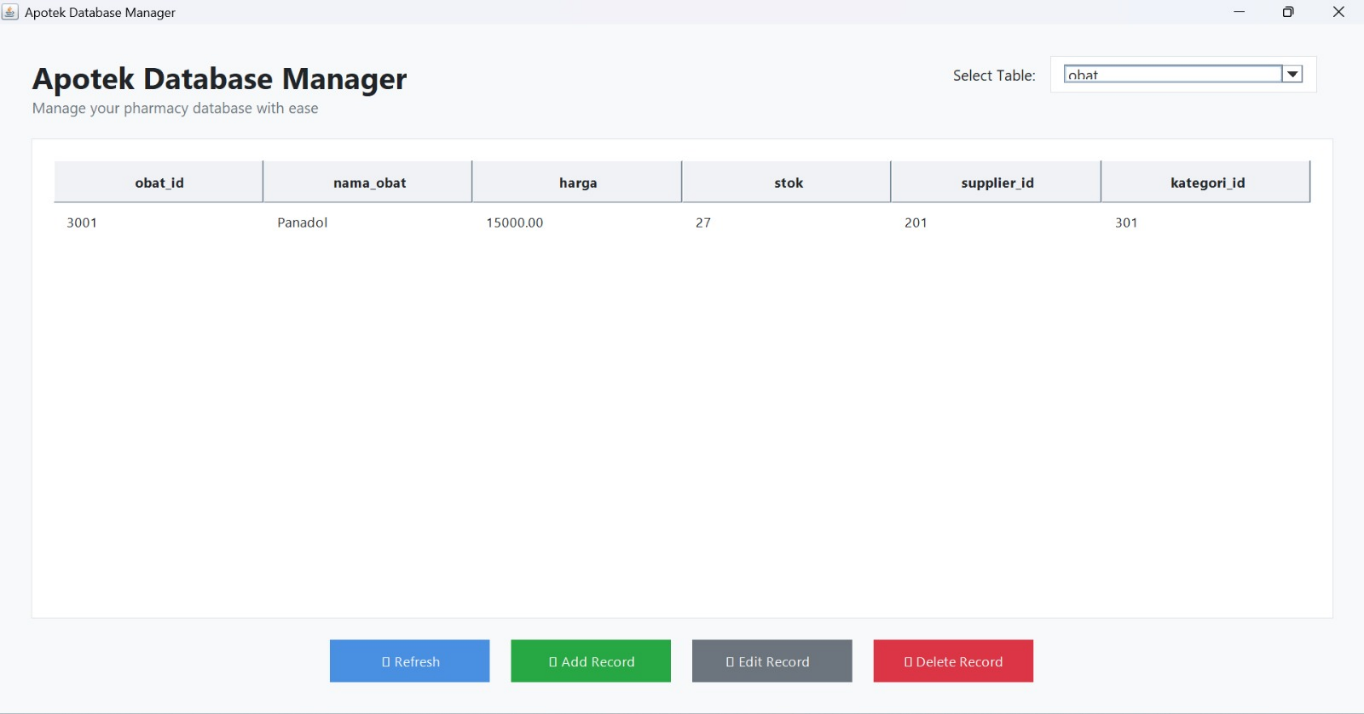
Pertanyaan ketika salah satu data akan di hapus dan jika sudah berhasil dihapus



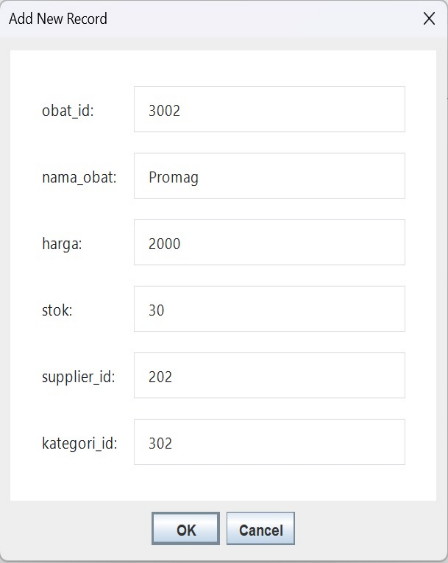
Halaman setelah salah satu kategori obat dihapus



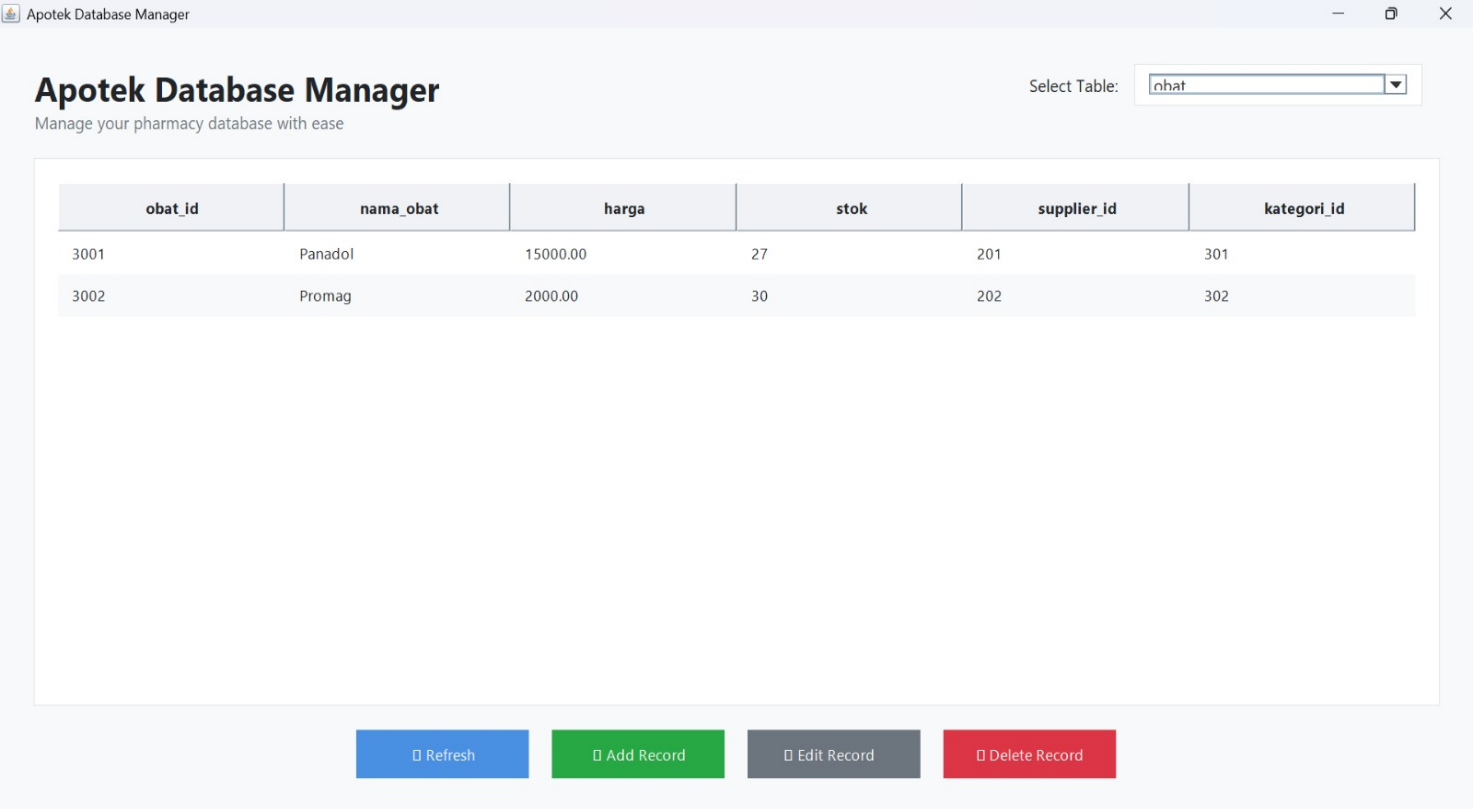
* Halaman awal ketika masuk ke menu obat



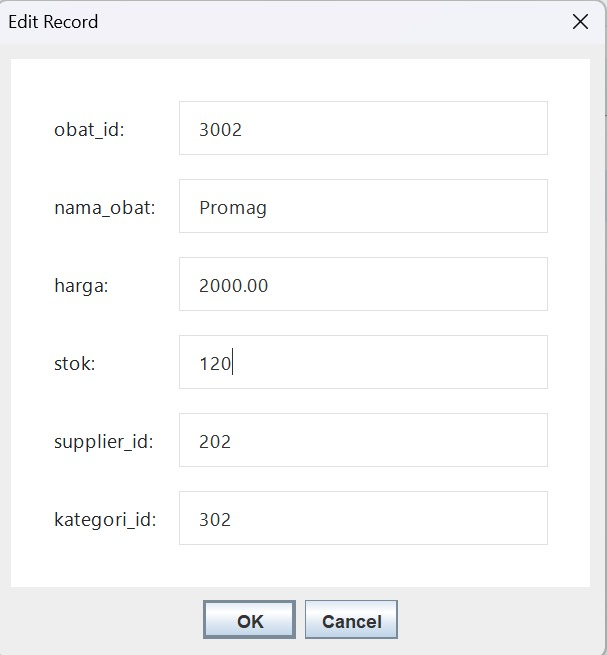
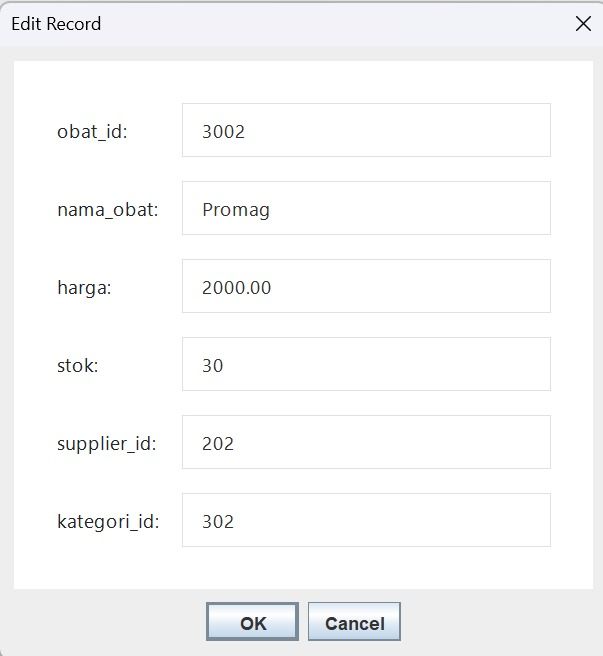
Ketika ingin menambahkan obat



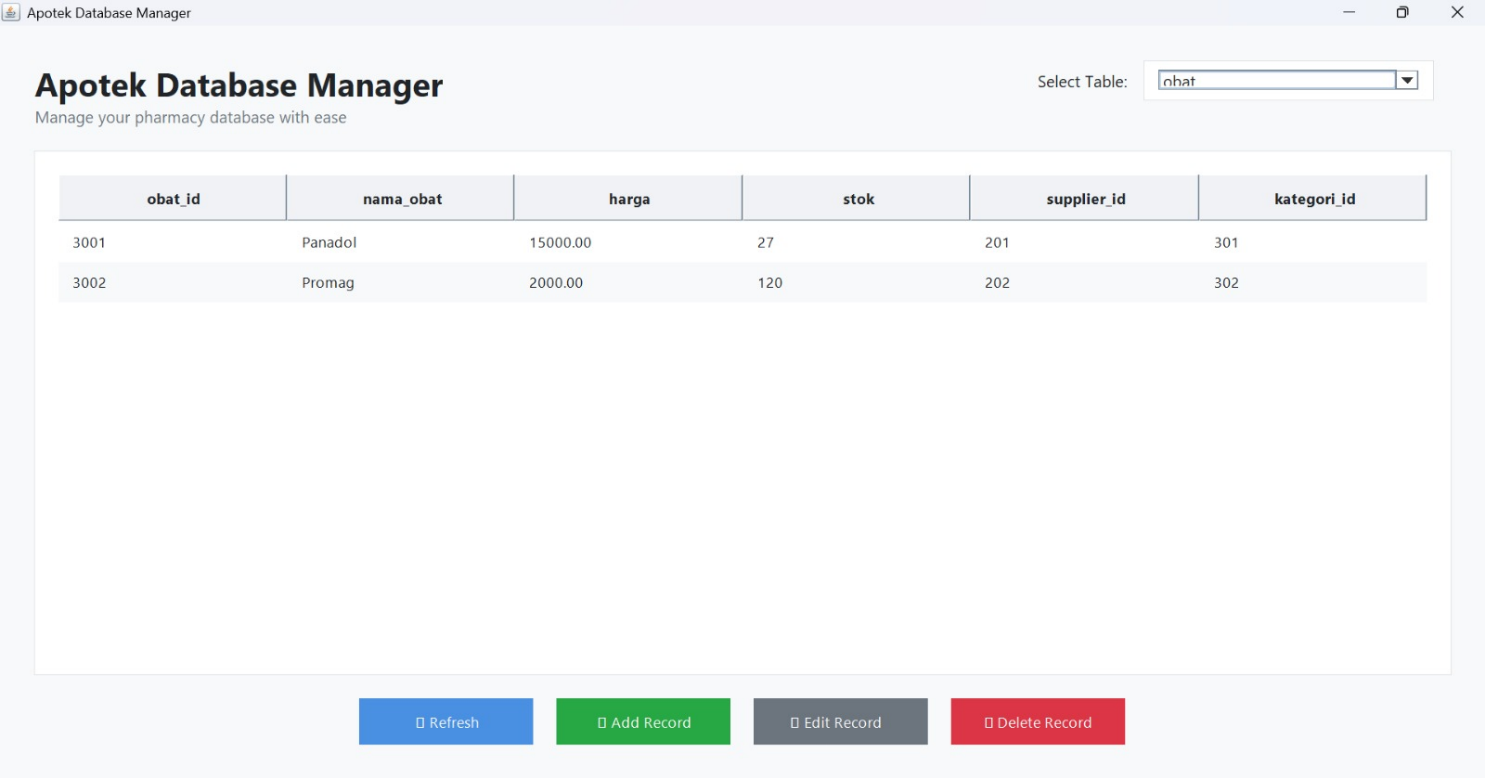
Halaman ketika obat berhasil ditambahkan dan akan di edit



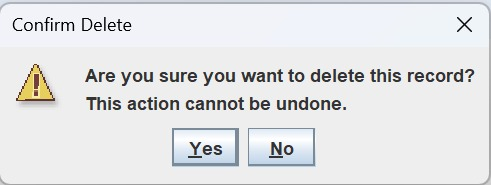
Sebagai contoh, edit stok dari 30 ke 120



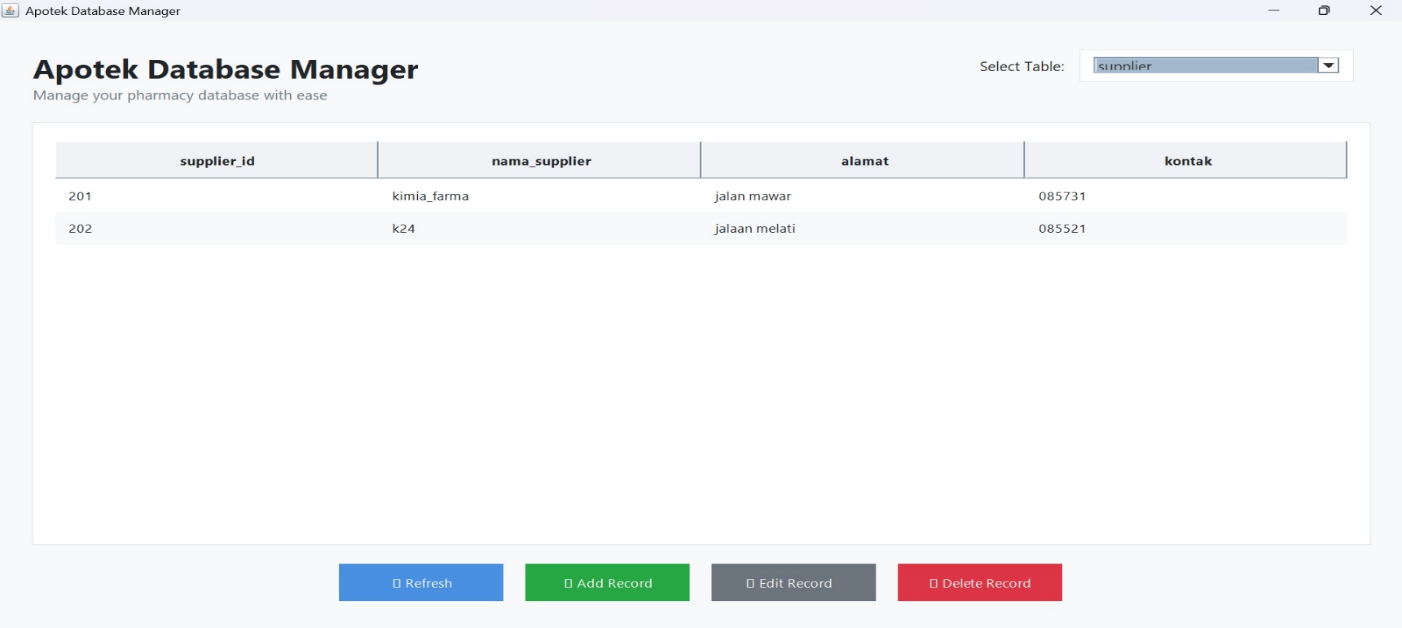
Halaman setelah diedit dan ketika ingin menghapus



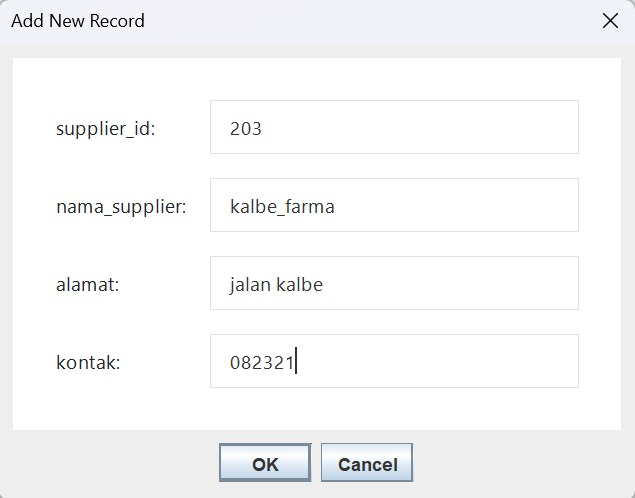
Konfirmasi ketika ingin menghapus



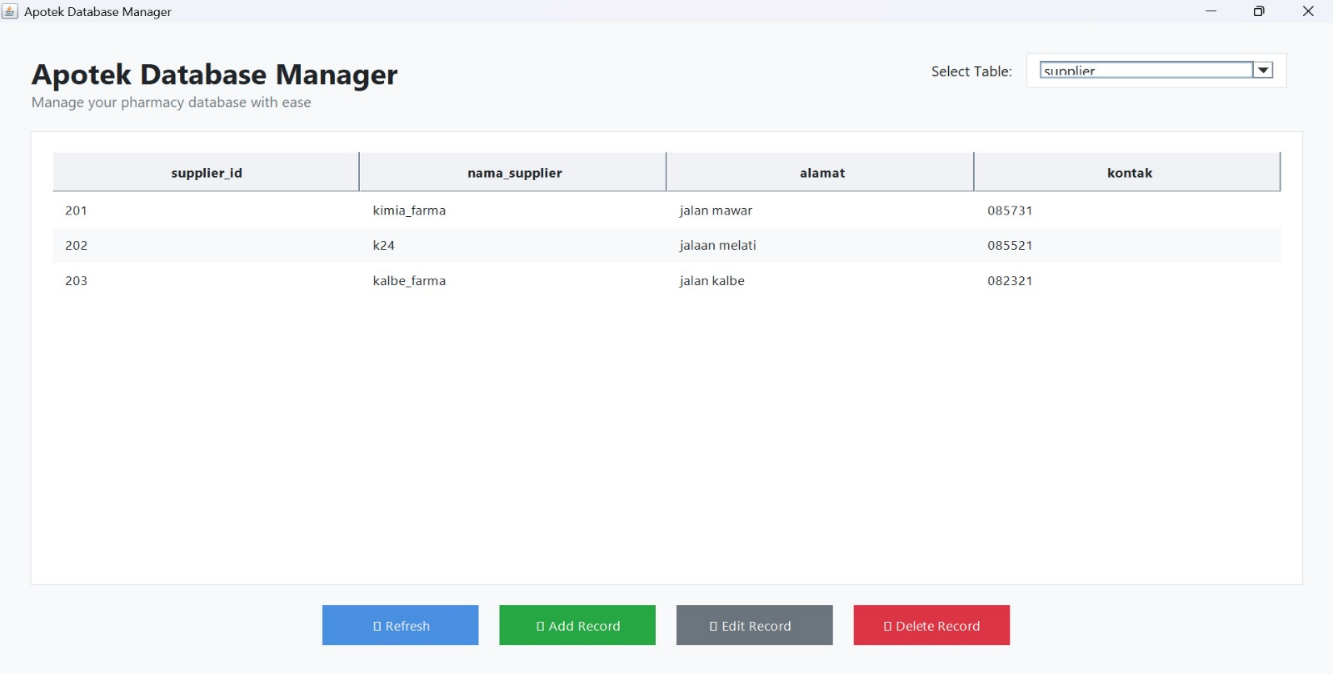
* Halaman awal tabel supplier dan ketika ingin menambahkan supplier



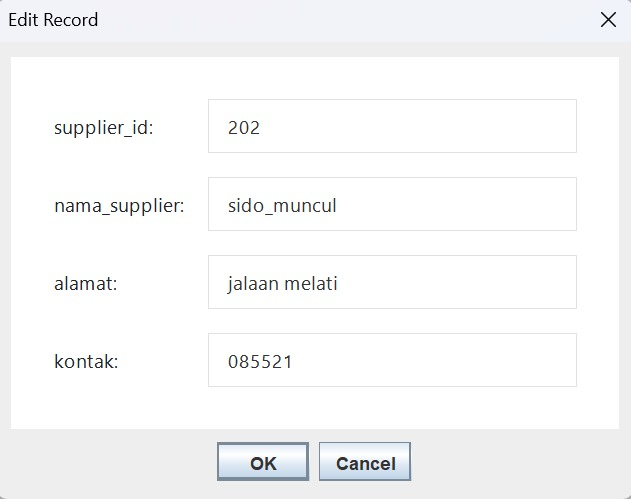
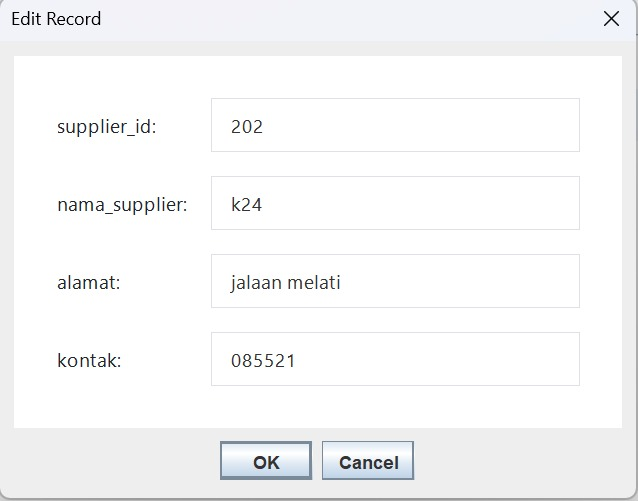
Menambahkan supplier



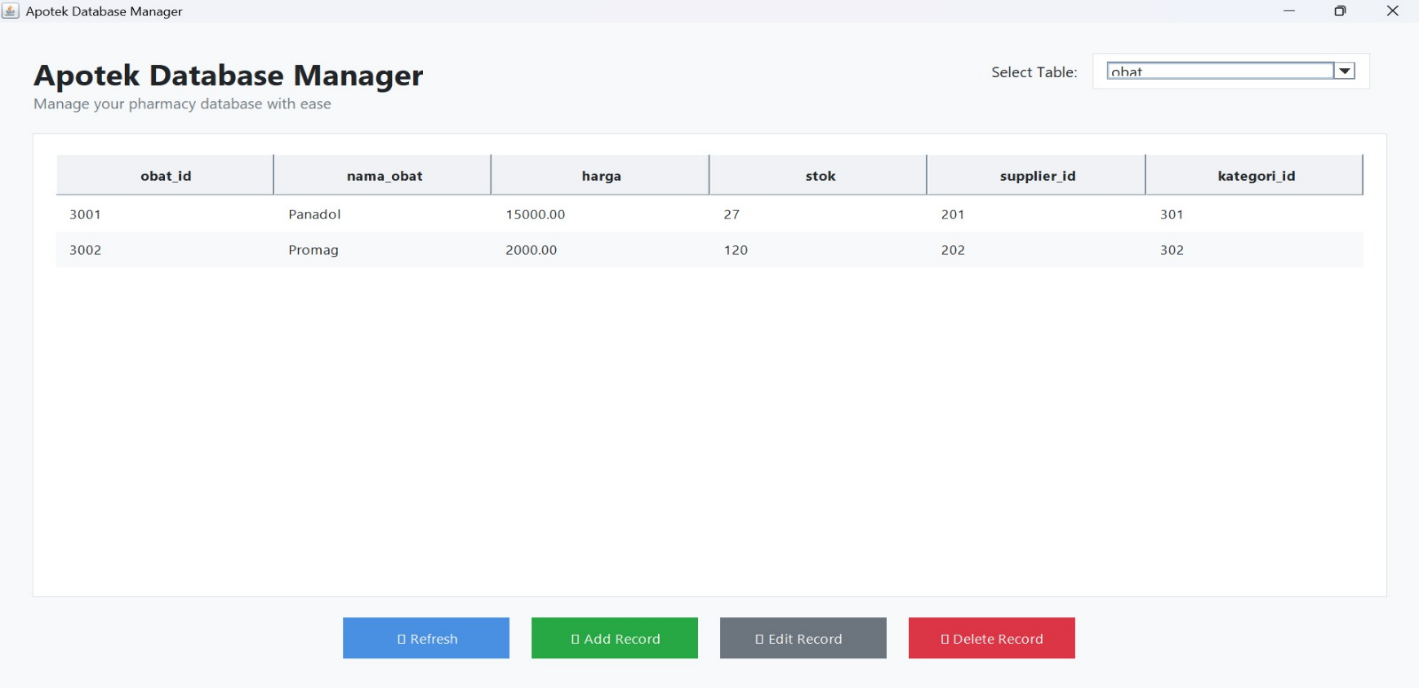
Halaman setelah ditambah dan akan mengedit data supplier



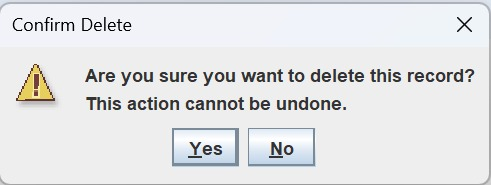
Sebagai contoh, mengedit nama supplier dari “k24” ke “sido\_muncul”d



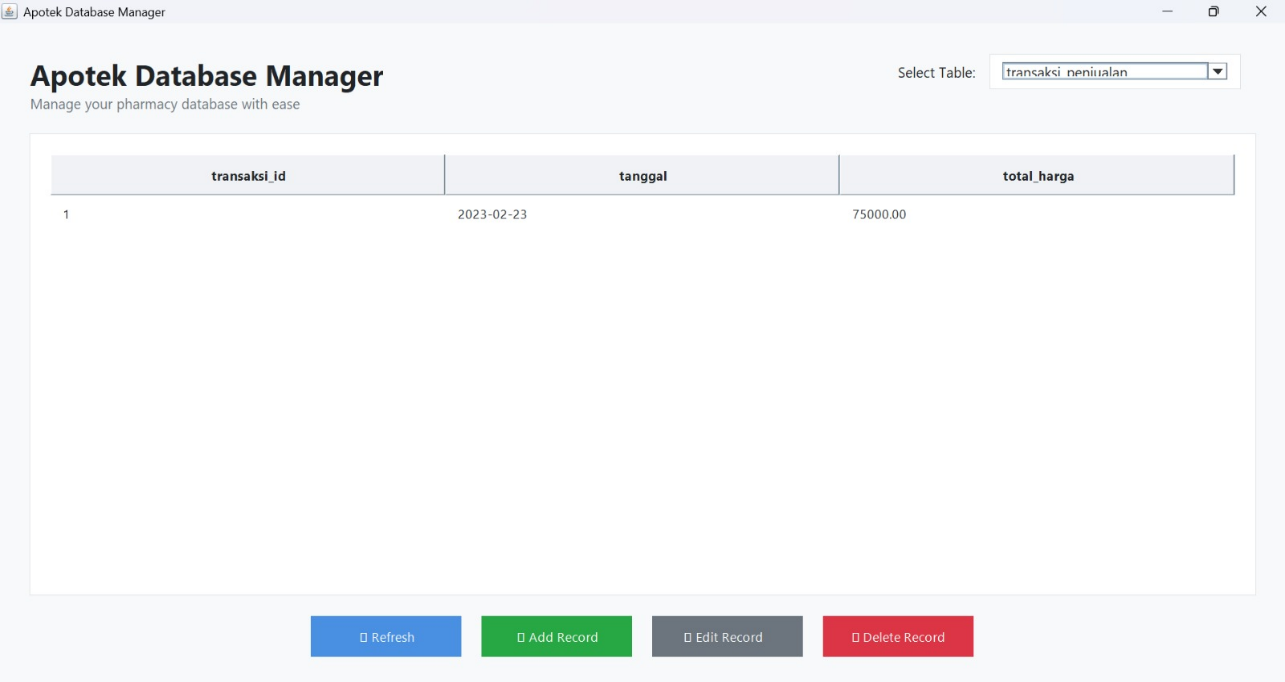
Halaman setelah di edit dan ingin menghapus



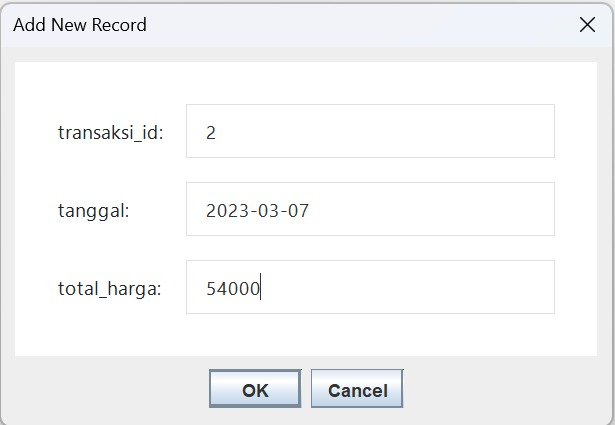
Konfirmasi ketika menghapus



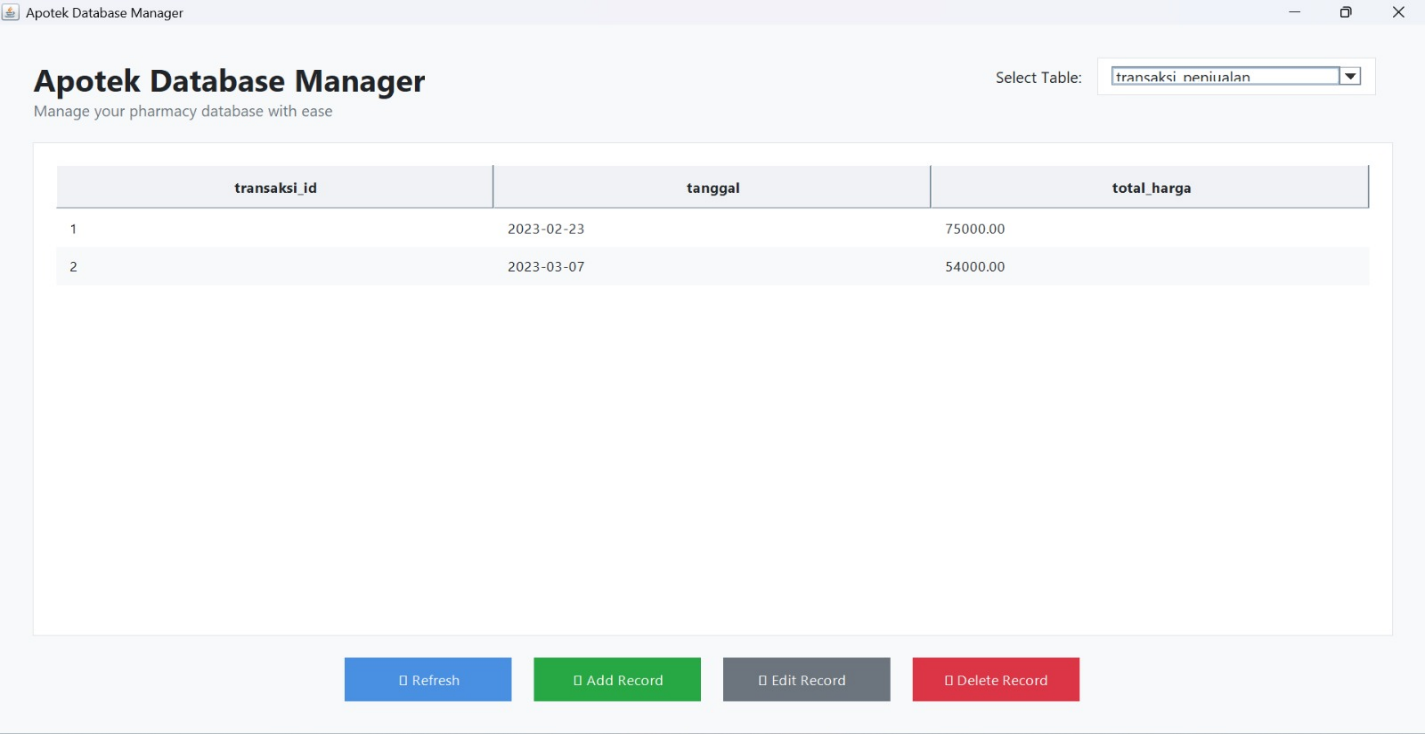
* Halaman awal tebel transasi penjualan dan ketika akan menambahkan datanya



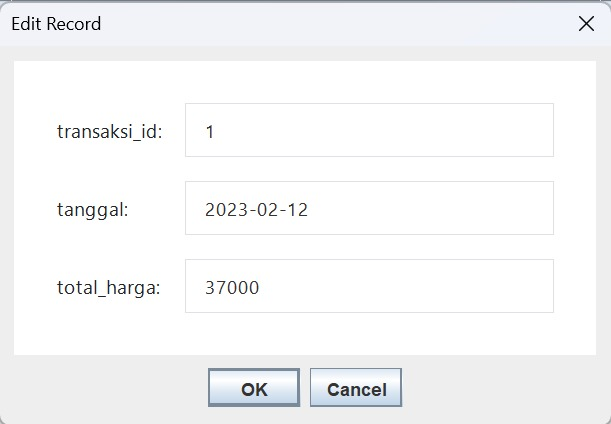
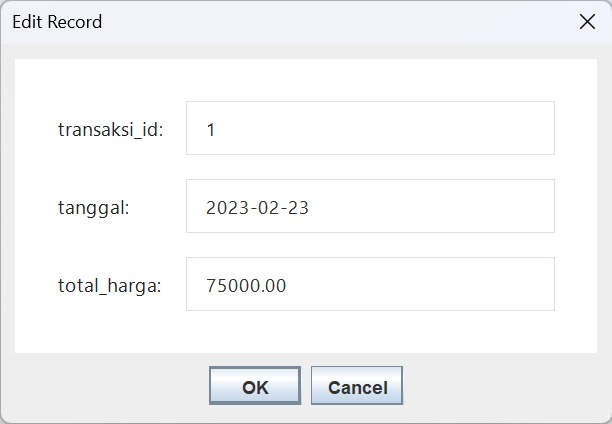
Menambahkan transaksi



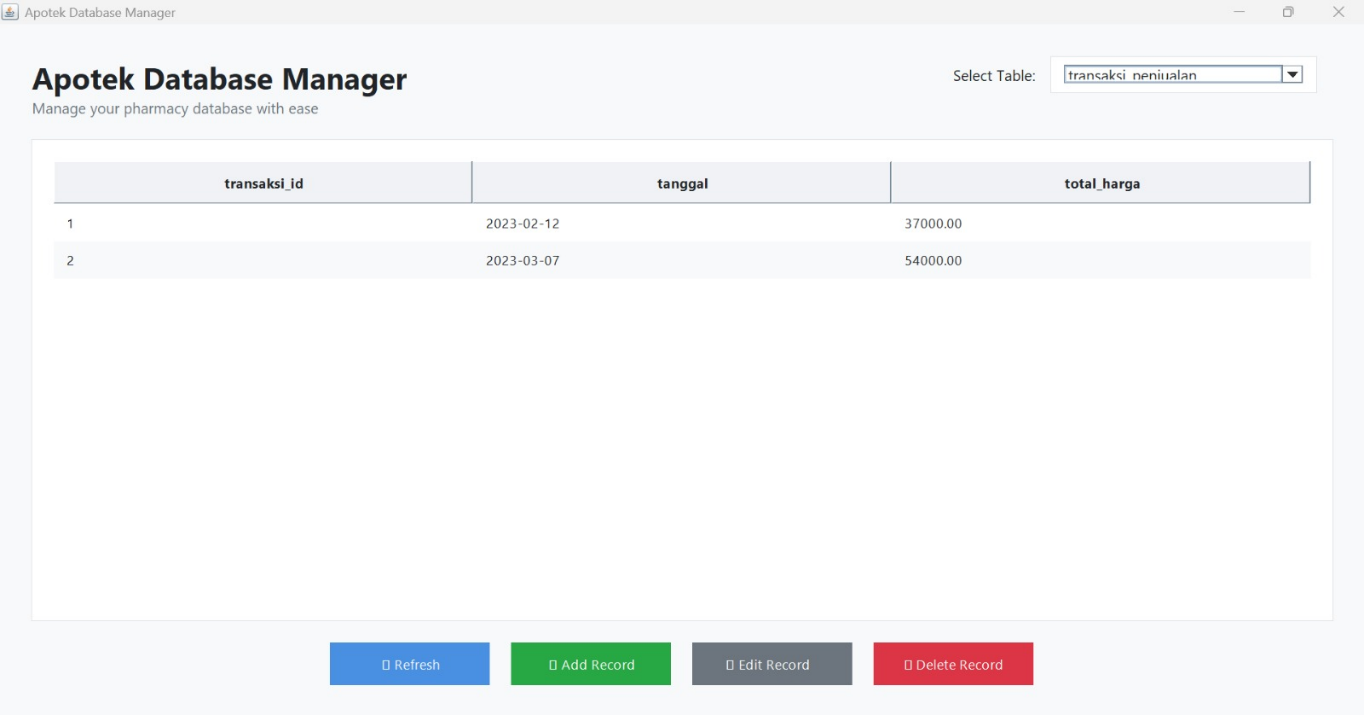
Halaman setelah menambahkan transaksi dan ketika ingin mengedit



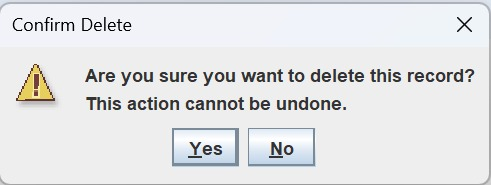
Sebagai contoh, mengubah tanggal dan total harga



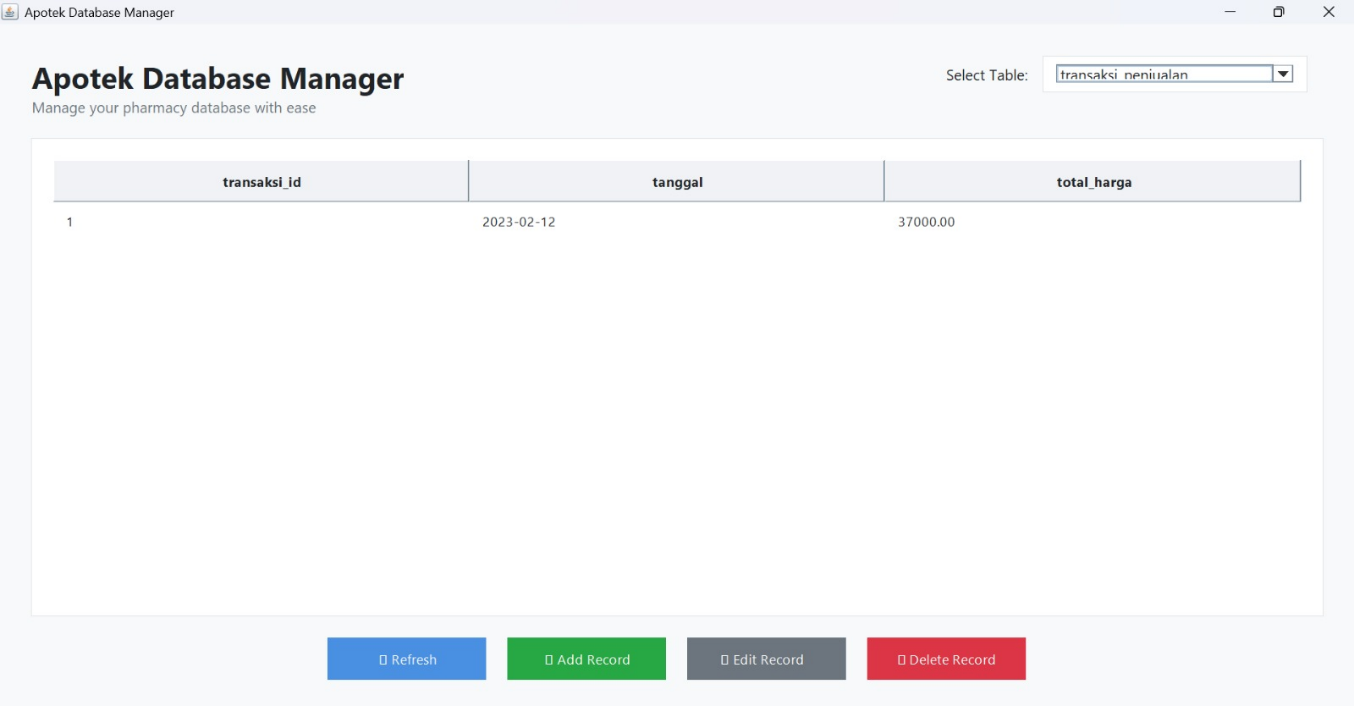
Halaman setelah diedit dan ingin menghapus



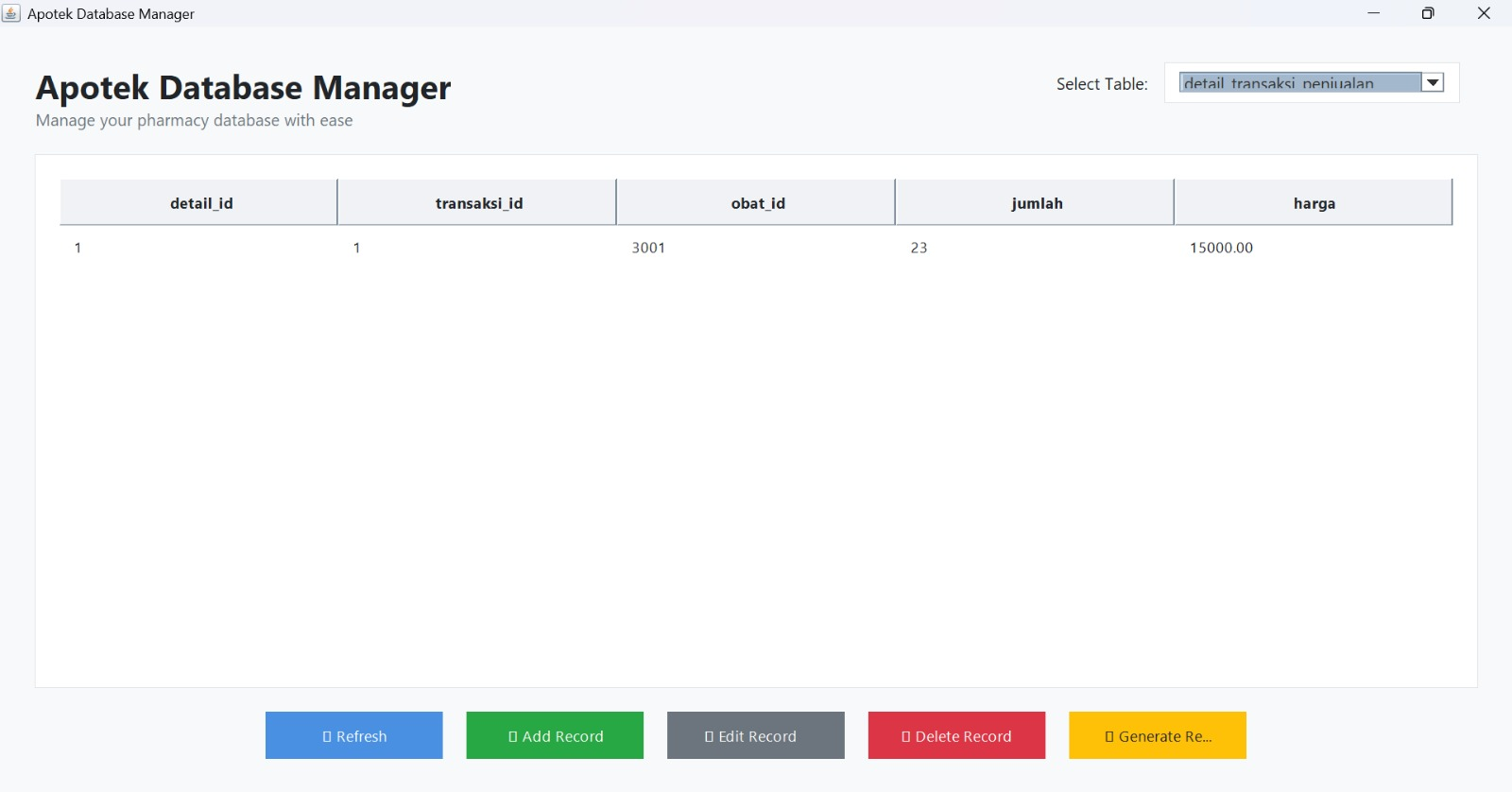
Konfirmasi ketika menghapus



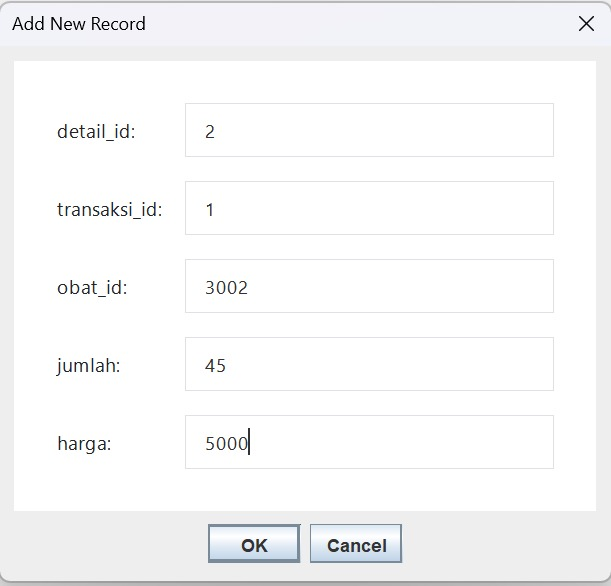
Halaman ketika selasai di hapus



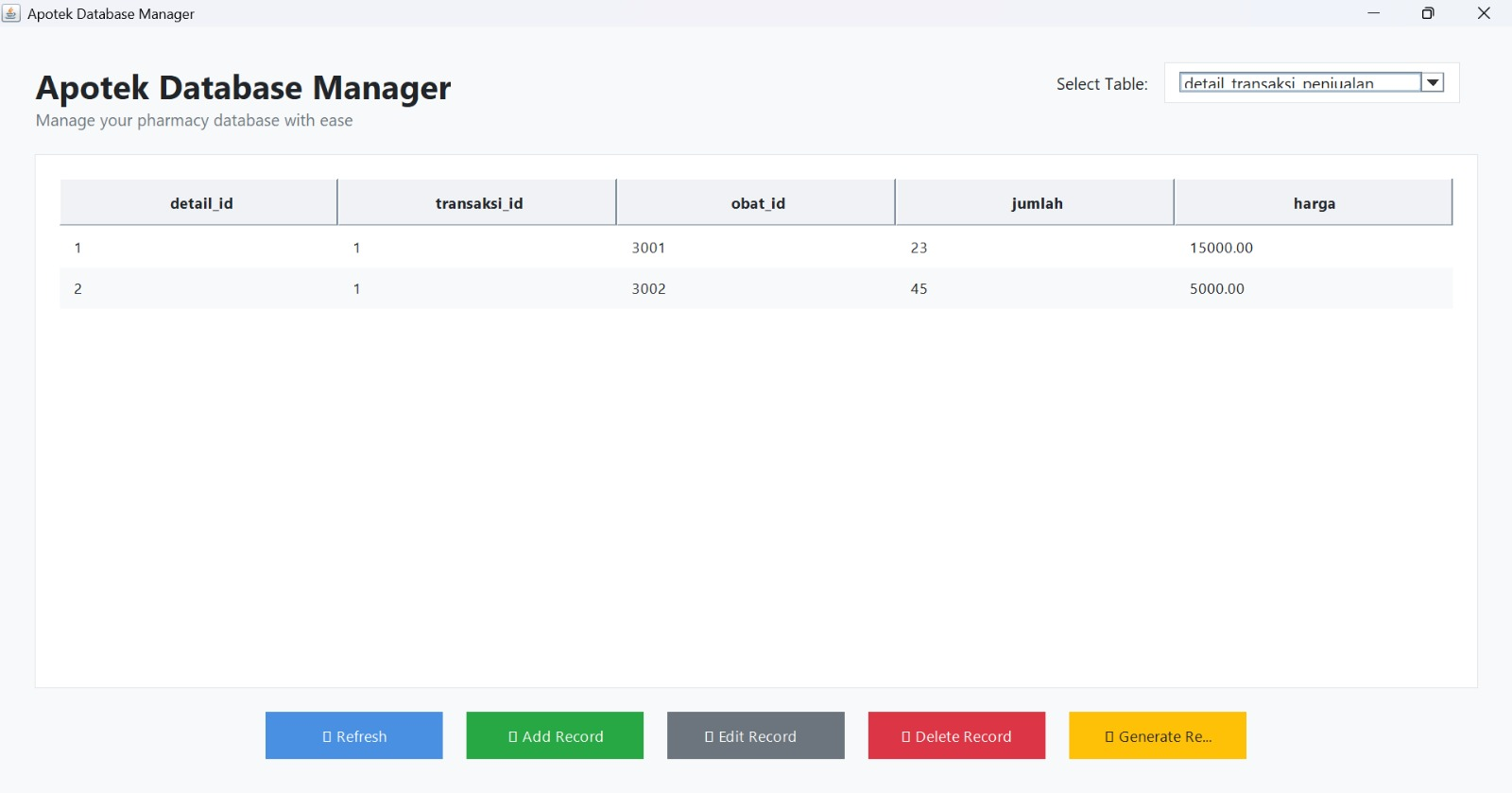
* Halaman awal tabel detail transaksi penjualan dan ingin menambahkan datanya



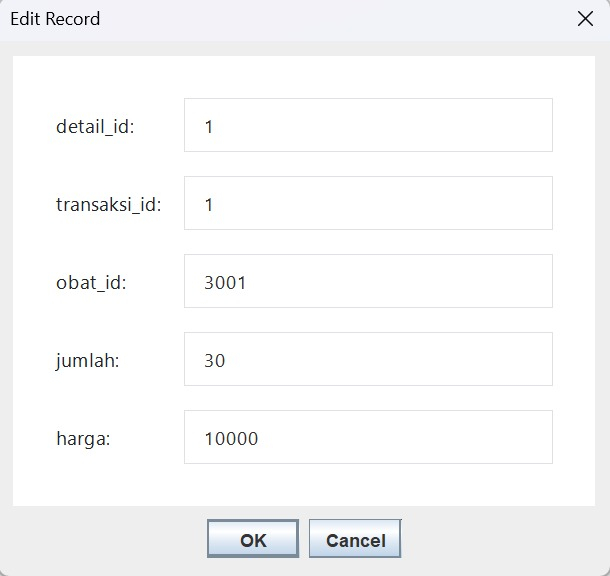
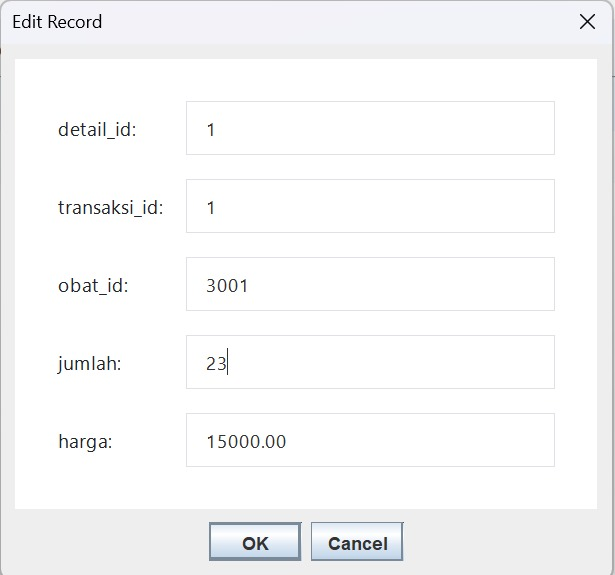
Menambahkan detail transaksi



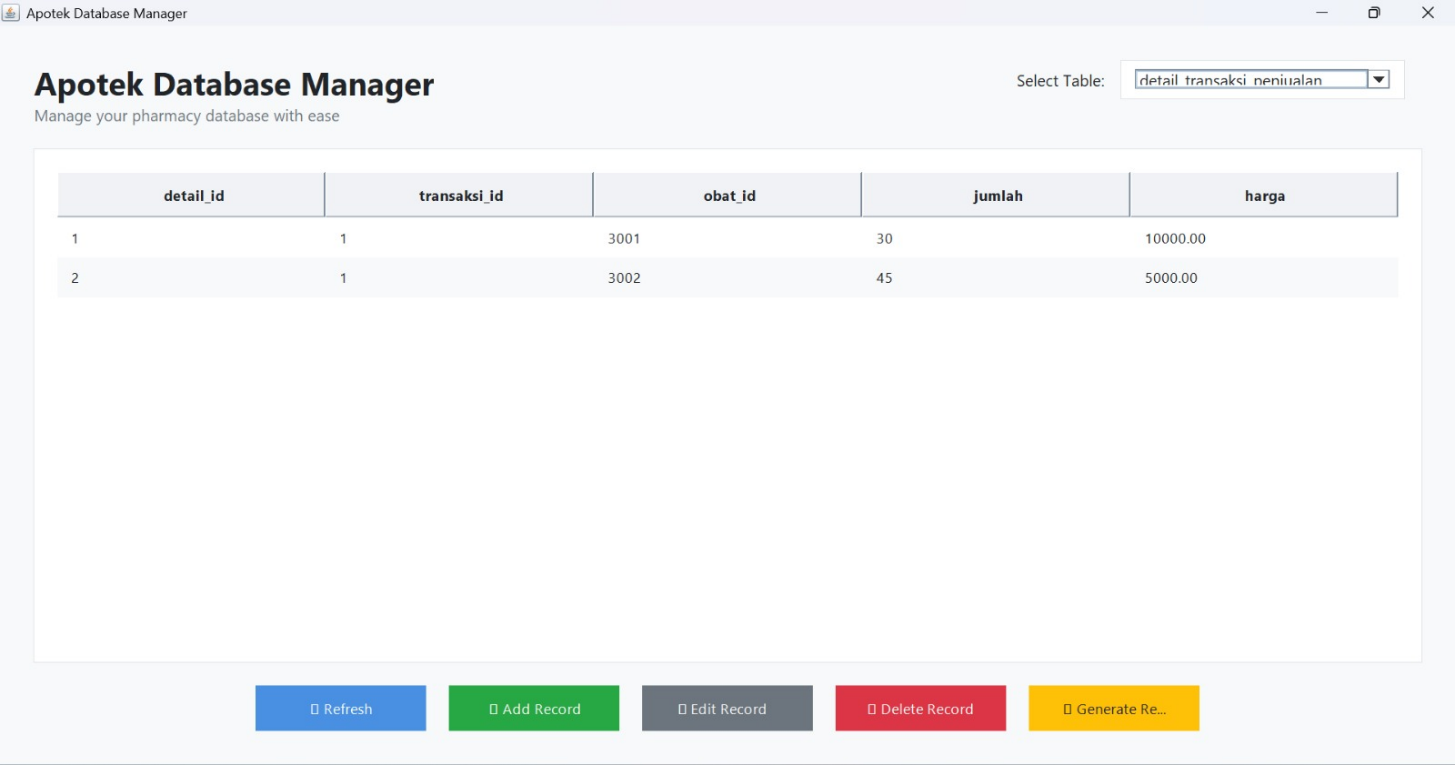
Halaman setlah detail transaksi ditambah dan ingin mengedit



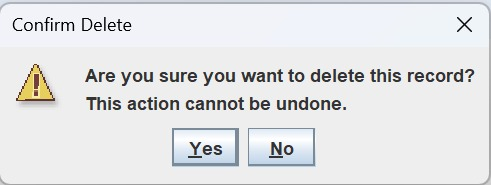
Sebagai contoh, mengubah jumlah dari 23 ke 30



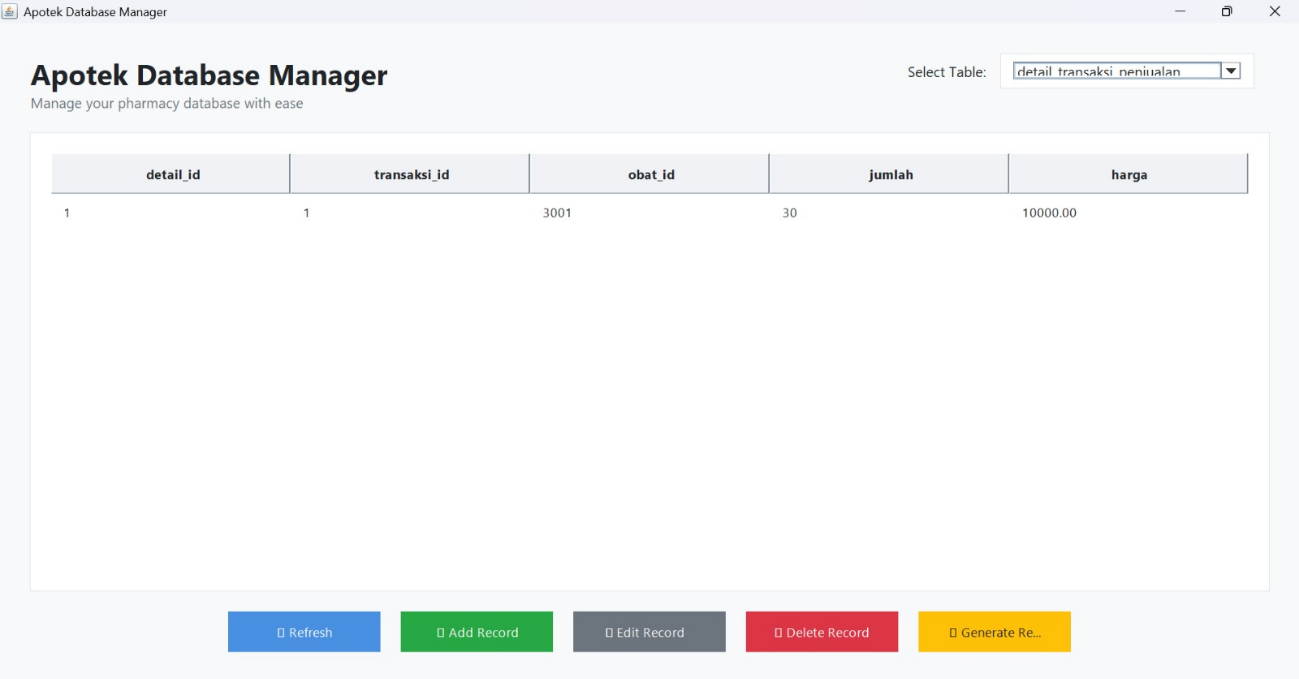
Halaman setelah di edit dan ingin menghapus



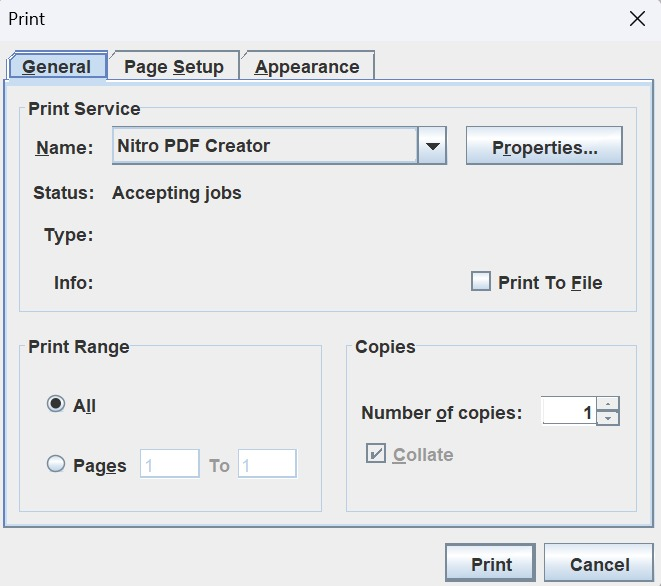
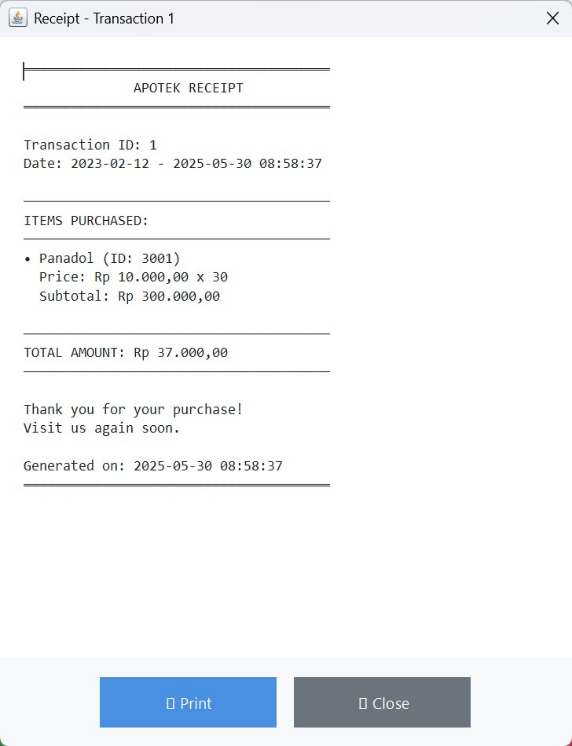
Konfirmasi ketika menghapus



Halaman setelah dihapus



* Struk dan print struk setelah melakukan pembayaran



## **KESIMPULAN**

Berdasarkan proses perancangan dan implementasi yang telah dilakukan, dapat disimpulkan bahwa aplikasi *Sistem Manajemen Apotek* ini berhasil dikembangkan dengan menerapkan prinsip-prinsip Pemrograman Berorientasi Objek (OOP). Aplikasi ini dirancang untuk membantu pengelolaan data pada apotek, mulai dari manajemen obat, kategori, supplier, hingga transaksi penjualan.

Fitur-fitur utama yang telah berhasil diimplementasikan antara lain pemilihan tabel data, penambahan, pengeditan, penghapusan, penyegaran (refresh) data, serta pembuatan struk transaksi penjualan. Dengan adanya fitur-fitur tersebut, proses operasional di apotek dapat dilakukan secara lebih terstruktur, cepat, dan efisien.

Selain itu, penggunaan pendekatan OOP memungkinkan sistem lebih modular, mudah dikembangkan, dan dikelola. Implementasi seperti ini menjadi bukti nyata penerapan teori yang telah dipelajari dalam mata kuliah PBO ke dalam solusi perangkat lunak yang berguna di dunia nyata.

Secara keseluruhan, tugas ini tidak hanya meningkatkan pemahaman teknis mahasiswa dalam pemrograman Java, tetapi juga melatih keterampilan analisis dan desain sistem yang dibutuhkan dalam pengembangan perangkat lunak skala kecil hingga menengah.