PRAKTEK PEMROGRAMAN BERORIENTASI OBJEK

UI GUI POS

Penyusun :

Rizky Satria Gunawan

(241511089)

2C-D3

PROGRAM STUDI D3 TEKNIK INFORMATIKA

JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA

POLITEKNIK NEGERI BANDUNG

Link Github : Mata-Kuliah-PBO/FolderTugas11 at main · RizkySatria123/Mata-Kuliah-PBO

Latar Belakang

Di era bisnis modern, efisiensi operasional menjadi kunci kesuksesan. Aplikasi Point of Sales (POS) atau sistem kasir adalah alat fundamental yang membantu bisnis mengelola transaksi penjualan secara cepat, akurat, dan terorganisir. Aplikasi ini tidak hanya mencatat penjualan tetapi juga dapat mengelola data produk dan inventaris.

Untuk membangun aplikasi desktop yang interaktif dan ramah pengguna, Java menyediakan *library* antarmuka grafis (GUI) yang kuat bernama Java Swing. Java Swing dipilih sebagai teknologi untuk praktikum ini karena menyediakan seperangkat komponen GUI yang kaya (seperti tabel, tombol, dan panel) serta fleksibilitas dalam mengelola tata letak. Proyek ini bertujuan membuat aplikasi POS sederhana untuk mensimulasikan proses transaksi ritel.

Tujuan Praktikum

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

- Menerapkan konsep dasar Pemrograman Berorientasi Objek (PBO) seperti **Class**, **Object**, dan **Encapsulation** dalam sebuah aplikasi nyata.
- Mengimplementasikan antarmuka grafis (GUI) yang fungsional menggunakan komponen-komponen inti **Java Swing**.
- Mengelola dan menangani interaksi pengguna melalui mekanisme **Event Handling** di Java.
- Membuat sebuah aplikasi fungsional yang dapat mensimulasikan alur kerja dasar dari proses transaksi penjualan, mulai dari pemilihan produk hingga pembuatan struk.

Ruang Lingkup

Aplikasi POS yang dibangun dalam praktikum ini memiliki fungsionalitas sebagai berikut:

1. Menampilkan daftar produk (ID, Nama, Harga) dalam sebuah tabel.
2. Memungkinkan pengguna memilih produk dari tabel, menentukan kuantitas, dan menambahkannya ke keranjang belanja.
3. Menampilkan isi keranjang belanja (termasuk subtotal) dalam tabel terpisah.
4. Menghitung total harga dari seluruh item di keranjang secara otomatis.

5. Memproses "Checkout", yang akan men-generate struk belanja dalam format teks dan mengosongkan keranjang.
6. Menyediakan opsi untuk "mencetak" struk (menggunakan dialog cetak sistem).

Aplikasi ini belum mencakup fitur-fitur kompleks seperti koneksi ke database, manajemen inventaris, atau autentikasi pengguna.

Dasar Teori

Pemrograman Berorientasi Objek (PBO)

PBO adalah paradigma pemrograman yang berfokus pada "objek" yang memiliki data (atribut) dan perilaku (metode).

- **Class dan Object:** Class adalah cetak biru (blueprint) untuk membuat objek. Dalam aplikasi ini, PointOfSales adalah class utama yang juga merupakan sebuah *object* JFrame. Objek-objek lain seperti JTable, JButton, dan DefaultTableModel dibuat (diinstansiasi) dari kelasnya masing-masing untuk membangun aplikasi.
- **Encapsulation:** Enkapsulasi adalah konsep membungkus data (variabel) dan metode yang beroperasi pada data tersebut ke dalam satu unit (kelas). Dalam kode ini, data-data penting seperti productTableModel dan cartTableModel dideklarasikan sebagai *private final field*. Akses dan modifikasi terhadap data ini diatur melalui metode-metode di dalam kelas PointOfSales, seperti addSelectedProductToCart() dan updateTotals().

Java Swing

Java Swing adalah *toolkit* GUI (Graphical User Interface) untuk Java yang merupakan bagian dari Java Foundation Classes (JFC). Swing menyediakan komponen yang ringan, *pluggable look-and-feel*, dan independen terhadap platform. Komponen kunci yang digunakan:

- **JFrame:** Bertindak sebagai jendela utama aplikasi. Kelas PointOfSales mewarisi (extends) JFrame untuk menjadi jendela itu sendiri.
- **JTable dan DefaultTableModel:** JTable digunakan untuk menampilkan data dalam bentuk grid dua dimensi. DefaultTableModel adalah model data yang paling umum digunakan untuk JTable, yang menyimpan data produk dan keranjang.

- **JSpinner:** Komponen input yang memungkinkan pengguna memilih angka (kuantitas) dari rentang yang telah ditentukan.
- **JButton:** Tombol standar yang digunakan untuk memicu aksi, seperti "Add to Cart", "Checkout", dan "Cetak".
- **JTextArea:** Area teks multi-baris yang digunakan untuk menampilkan struk belanja yang telah di-generate.
- **Layout Manager (BorderLayout, BoxLayout):** Digunakan untuk mengatur tata letak dan posisi komponen-komponen GUI di dalam jendela, seperti BorderLayout.CENTER, BorderLayout.SOUTH, dan BorderLayout.EAST.

Event Handling

Event handling adalah mekanisme untuk merespons interaksi dari pengguna (seperti klik tombol atau pemilihan item). Java menggunakan model *event-listener*.

- **ActionListener:** Digunakan untuk menangani aksi, biasanya klik tombol. Dalam kode ini, ActionListener (dibuat menggunakan *lambda expression*) ditambahkan ke JButton ("Add to Cart", "Checkout") dan JMenuItem ("Exit", "About").
- **ListSelectionListener:** Digunakan untuk mendeteksi perubahan pada seleksi di sebuah komponen. Dalam kode ini, ListSelectionListener dipasang pada productTable untuk mendeteksi produk mana yang sedang dipilih oleh pengguna, yang kemudian memicu metode updateSelectedProductLabel().

Implementasi dan Pembahasan

Analisis kode PointOfSales.java:

Struktur Kelas Utama

Kelas PointOfSales didefinisikan sebagai public class PointOfSales extends JFrame. Ini berarti PointOfSales adalah sebuah JFrame, yang menjadikannya jendela utama dari aplikasi. Semua komponen GUI dan data model (seperti productTable, cartTableModel, totalLabel) dideklarasikan sebagai *fields* (atribut) di dalam kelas ini, sehingga dapat diakses oleh semua metode di dalamnya.

Inisialisasi GUI

Di dalam *constructor* PointOfSales(), semua komponen GUI diinisialisasi dan dirakit:

1. Properti dasar JFrame diatur, seperti judul (super("POIN Off-Sales - Java Swing")), setDefaultCloseOperation, dan setLayout(new BorderLayout(10, 10)).
2. JMenuBar dibuat melalui createMenuBar() dan dipasang.
3. Model tabel (productTableModel dan cartTableModel) dibuat oleh metodenya masing-masing.
4. JTable (productTable dan cartTable) dibuat menggunakan model tersebut. Properti tabel diatur, seperti setSelectionMode dan setPreferredWidth untuk kolom.
5. Panel-panel utama (panel kontrol di SOUTH dan panel struk di EAST) dibuat oleh metode createControlPanel() dan createReceiptPanel(), lalu ditambahkan ke JFrame menggunakan BorderLayout.

Model Tabel (Data Model)

Aplikasi ini menggunakan DefaultTableModel untuk productTable dan cartTable.

- **createProductTableModel():** Metode ini mengembalikan DefaultTableModel yang diisi dengan data produk *hardcoded* (statis). Sebuah *inner class* anonim digunakan untuk meng-override dua metode:
    - isCellEditable(int row, int column): Selalu mengembalikan false agar pengguna tidak dapat mengedit data produk langsung di tabel.
    - getColumnClass(int columnIndex): Mengembalikan Double.class untuk kolom harga. Ini penting agar tabel dapat mengurutkan kolom harga sebagai angka, bukan teks.
- **createCartTableModel():** Mirip dengan model produk, metode ini mendefinisikan kolom untuk keranjang dan meng-override getColumnClass untuk tipe data Integer (Qty) dan Double (Harga, Subtotal).

Penanganan Aksi (Event Handling)

Logika inti aplikasi diatur oleh *event listener*:

- **Pemilihan Produk:** Sebuah ListSelectionListener ditambahkan ke productTable.getSelectionModel(). Setiap kali pengguna memilih baris yang berbeda di tabel produk, listener ini memanggil updateSelectedProductLabel(). Metode ini mengambil data dari baris yang dipilih dan memperbarui selectedProductLabel dengan teks seperti "Dipilih: P001 - Air Mineral (Rp3.000,00)".

- **Tambah ke Keranjang:** JButton "Add to Cart" memiliki ActionListener yang memanggil addSelectedProductToCart(). Metode ini:
    1. Memeriksa apakah ada produk yang dipilih.
    2. Mengambil data produk (ID, nama, harga) dari productTableModel.
    3. Mengambil kuantitas dari qtySpinner.
    4. Menghitung subtotal (price * quantity).
    5. Menambahkan baris baru ke cartTableModel dengan data tersebut.
    6. Memanggil updateTotals().
- **Update Total:** Metode updateTotals() dipanggil setiap kali item ditambahkan ke keranjang. Metode ini memanggil calculateTotal() (yang melakukan *looping* pada cartTableModel untuk menjumlahkan subtotal) dan kemudian memperbarui teks pada totalLabel dan pointsLabel.
- **Checkout:** JButton "Checkout" memanggil performCheckout(). Metode ini:
    1. Membuat struk belanja dalam bentuk String menggunakan StringBuilder.
    2. Struk ini mencakup *timestamp*, *header* tabel, daftar item dari cartTableModel, total, dan poin.
    3. Teks struk yang sudah jadi ditampilkan di receiptArea.
    4. Keranjang dikosongkan dengan cartTableModel.setRowCount(0) dan total di-update kembali ke nol.

Formatting Kustom

Untuk menampilkan harga dalam format mata uang Rupiah (misal, "Rp3.000,00"), metode setCurrencyRenderer(JTable table, int columnIndex) digunakan.

1. Metode ini membuat DefaultTableCellRenderer kustom (melalui *inner class* anonim).
2. Di dalam *renderer* ini, metode setValue di-override. Jika nilai yang diterima adalah Number, nilai tersebut diformat menggunakan formatCurrency(double amount).
3. Metode formatCurrency sendiri menggunakan NumberFormat.getCurrencyInstance(ID_LOCALE) untuk memastikan format mata uang yang benar (menggunakan "Rp" dan pemisah ribuan/desimal Indonesia).

4. *Renderer* ini kemudian diterapkan ke kolom harga di productTable dan cartTable.

5. Hasil (Contoh Tampilan)

Tampilan Utama

Aplikasi menampilkan satu jendela utama. Tata letak dibagi menjadi tiga bagian utama menggunakan BorderLayout:

1. **CENTER:** Sebuah JTabbedPane (panel tab) yang berisi dua tab:
    - Tab "Produk": Menampilkan JTable dengan daftar produk yang tersedia.
    - Tab "Keranjang": Menampilkan JTable yang awalnya kosong, berisi item yang ditambahkan pengguna.
2. **SOUTH:** Sebuah panel kontrol (createControlPanel) yang dibagi dua:
    - Sisi kiri ("Input"): Menampilkan label produk yang dipilih, JSpinner untuk kuantitas, dan tombol "Add to Cart".
    - Sisi kanan ("Ringkasan"): Menampilkan totalLabel, pointsLabel, dan tombol "Checkout" serta "Cetak".
3. **EAST:** Sebuah panel struk (createReceiptPanel) yang menampilkan JTextArea (area teks) besar yang awalnya kosong, siap untuk menampilkan struk setelah *checkout*.

Alur Penggunaan

Skenario penggunaan aplikasi berjalan sebagai berikut:

1. Pengguna menjalankan aplikasi. Daftar produk (Air Mineral, Kopi Sachet, dll.) terlihat di tab "Produk".
2. Pengguna mengklik "Air Mineral" pada tabel produk. Label di panel kontrol berubah menjadi "Dipilih: P001 - Air Mineral (Rp3.000,00)".
3. Pengguna mengubah nilai JSpinner Qty menjadi 3, lalu menekan tombol "Add to Cart".
4. Satu baris baru muncul di tab "Keranjang" berisi "P001", "Air Mineral", "3", "Rp3.000,00", "Rp9.000,00".

5. Secara bersamaan, label di panel kontrol ter-update menjadi "Total: Rp9.000,00" dan "Points: 0".
6. Pengguna kembali ke tab "Produk", memilih "Snack Keripik", Qty 2, dan menekan "Add to Cart".
7. Tab "Keranjang" kini memiliki dua item. Total ter-update menjadi "Total: Rp21.000,00" (9.000 + 12.000) dan "Points: 21".
8. Pengguna menekan tombol "Checkout".
9. Area teks struk di sebelah kanan (receiptArea) terisi dengan teks struk lengkap yang merinci kedua pembelian, total, dan poin. Tab "Keranjang" dan label Total/Poin kembali ke keadaan kosong/nol.

Output :

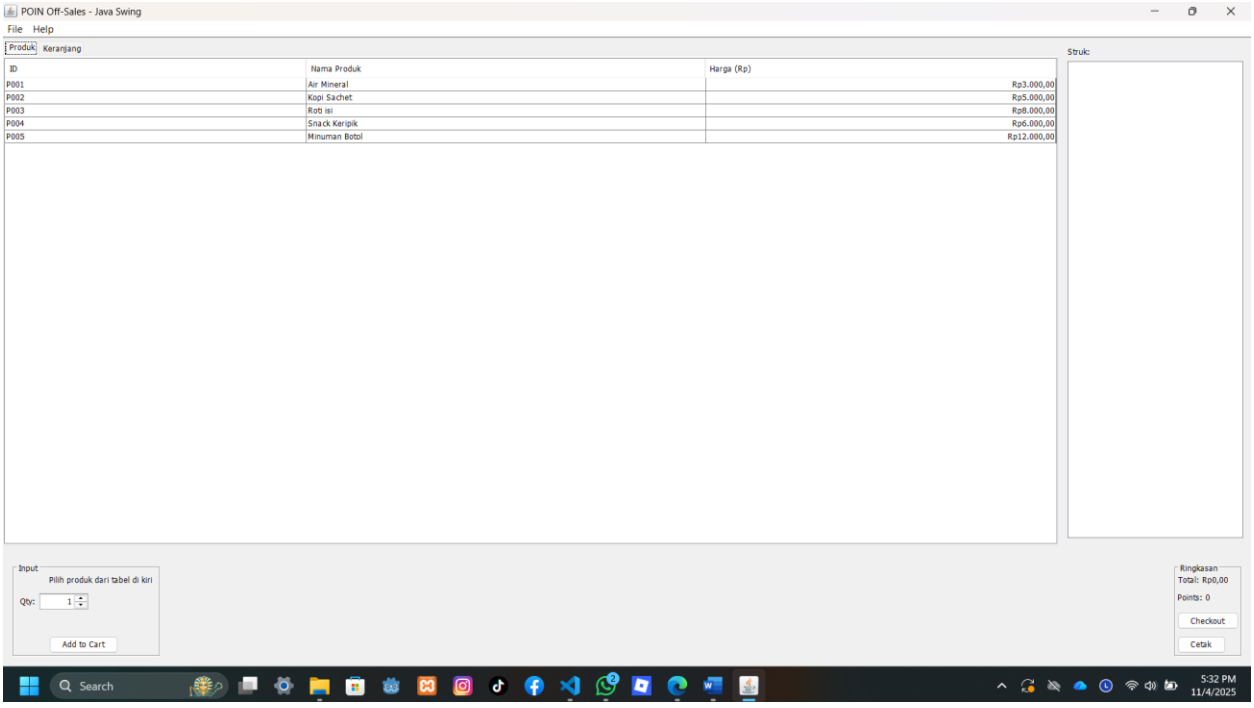

Figure 1 Menu awal

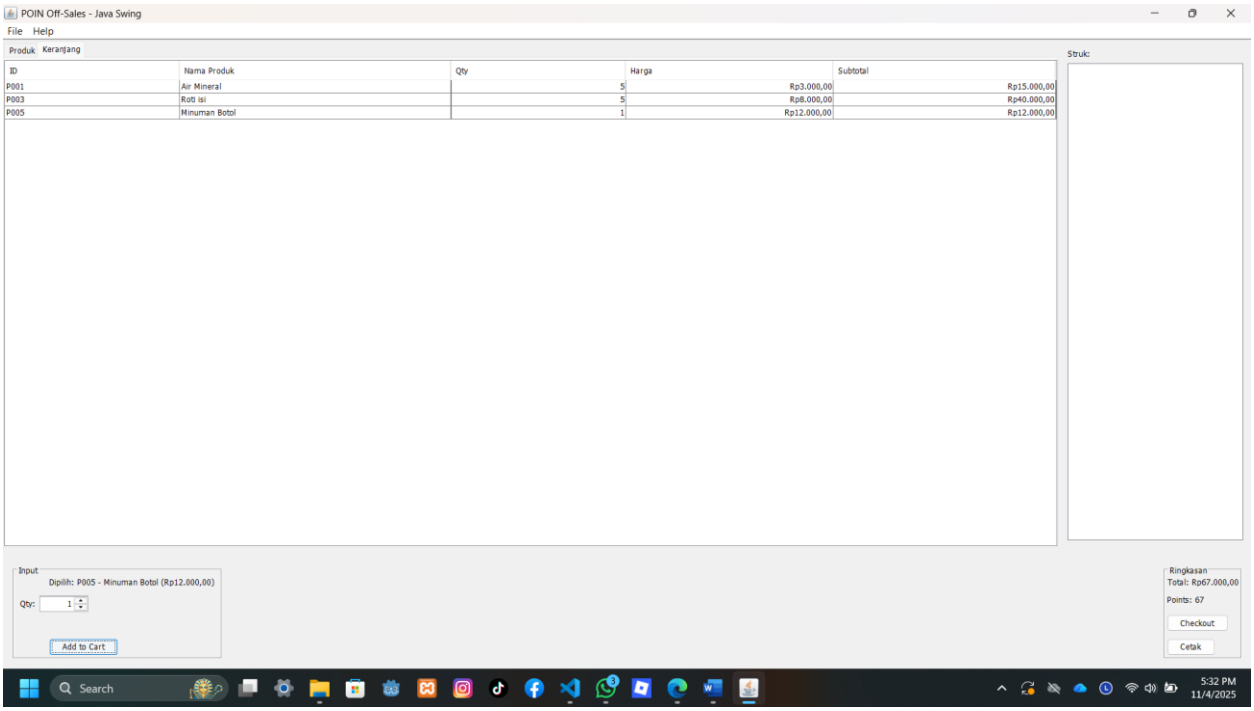

Figure 2 Keranjang belanjaan

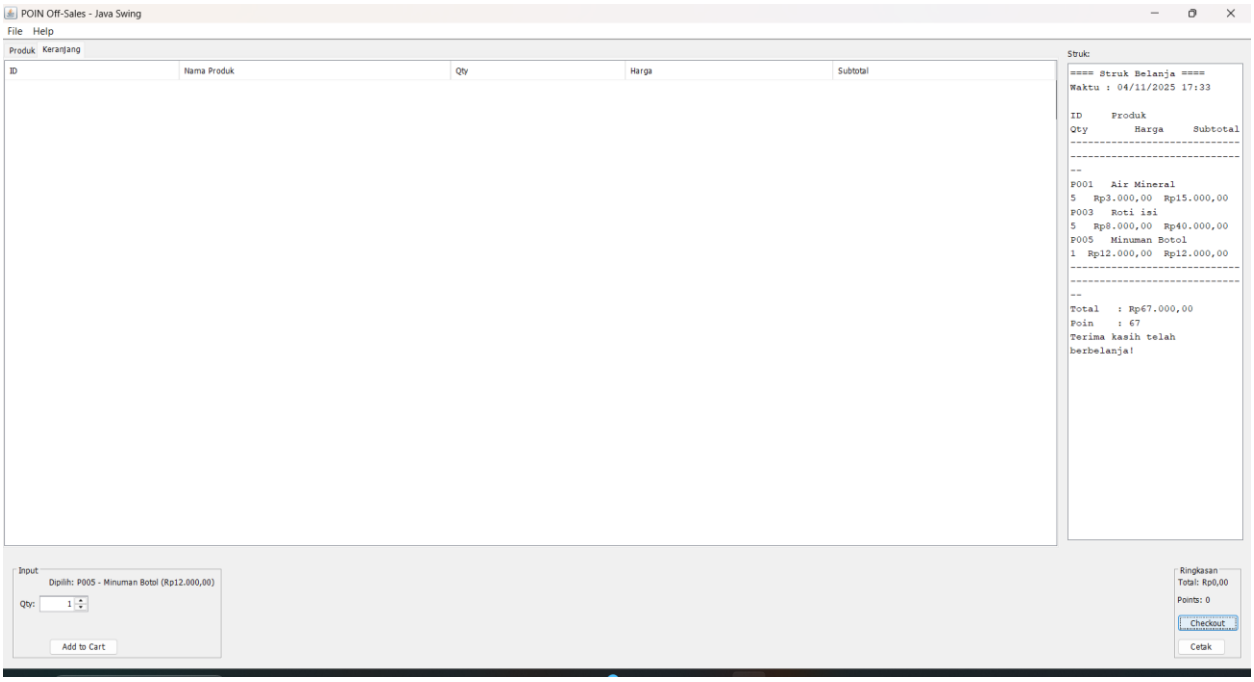

Figure 3 Setelah di checkout, akan muncul struk belanja

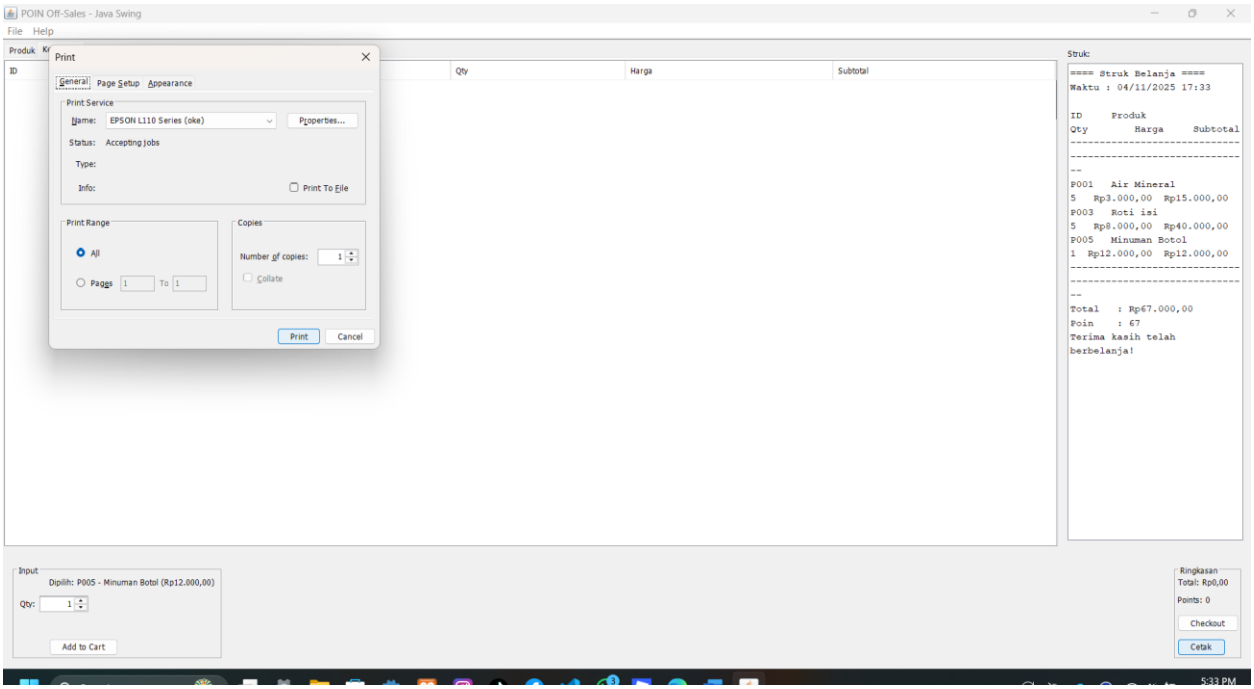

Figure 4 saat memilih cetak

6. Kesimpulan

Aplikasi Point of Sales (POS) sederhana telah berhasil dibuat menggunakan *library* Java Swing dan menerapkan konsep-konsep inti Pemrograman Berorientasi Objek. Konsep PBO seperti **Class** (PointOfSales sebagai JFrame), **Object** (instansiasi JTable, JButton), dan **Encapsulation** (penggunaan *private fields* untuk model tabel) telah diterapkan dengan baik.

Fungsionalitas utama aplikasi, seperti menampilkan produk, menambah item ke keranjang, menghitung total belanja secara dinamis, dan men-generate struk belanja, telah berjalan sesuai harapan. Penggunaan **Event Handling** (via ActionListener dan ListSelectionListener) terbukti efektif dalam membuat aplikasi yang interaktif dan responsif terhadap input pengguna.

(Opsional) **Pengembangan Lanjut:** Untuk pengembangan di masa depan, aplikasi ini dapat ditingkatkan dengan beberapa fitur, seperti:

- Mengganti data produk *hardcoded* dengan koneksi ke **Database** (misalnya MySQL atau PostgreSQL) untuk manajemen produk yang persisten.
- Menambahkan fungsionalitas **Manajemen Stok** (inventaris) yang berkurang setiap kali terjadi penjualan.

- Implementasi sistem **Login** dan autentikasi untuk kasir.