

LAPORAN AKHIR TUGAS BESAR PRAKTIKUM PEMOGRAMAN 1

Perdagangan / Pencatatan Transaksi Harian (Queue)



Dipersiapkan Oleh:

Kelas: D

Ketua: 233040029 - Tyezar Hasan Khusaeni

Anggota: 233040001 - Kresna Satria Dewantoro

233040017 - Aqillah Lean Andera

233040036 – Moch Sachrul Akbar Ramadhan

Nama Asisten Laboratorium:

Bhadrika Aryaputra Hermawan, Lisvindanu

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PASUNDAN BANDUNG
2024**

Daftar Isi

Daftar Isi.....	2
1. Pendahuluan.....	3
1.1. Deskripsi.....	3
1.2. Lingkup Pengerjaan.....	4
1.3. Dokumentasi Pengerjaan.....	5
1.4. Pembagian Tugas.....	6
2. Analisis dan Perancangan.....	7
2.1. Analisis Kebutuhan Sistem.....	7
2.1.1. Kebutuhan Fungsional.....	7
2.1.2. Kebutuhan Non Fungsional.....	8
2.2. Perancangan Struktur Data.....	9
2.3. Perancangan Algoritma.....	10
3. Implementasi Perangkat Lunak.....	13
3.1. Kakas dan Perangkat Lunak.....	13
3.1.1. Spesifikasi Kakas/Hardware.....	13
3.1.2. Daftar Perangkat Lunak.....	14
3.2. Struktur Proyek.....	15
3.3. Implementasi Kelas dan Struktur Data.....	16
3.4. Fitur-Fitur Aplikasi.....	21
Kesimpulan.....	26
Lampiran.....	28

1. Pendahuluan

1.1. Deskripsi

Latar Belakang:

Dalam operasional sehari-hari, restoran atau tempat makan seringkali mengalami antrean pemesanan dari pelanggan yang datang silih berganti. Tanpa pencatatan yang terstruktur, proses pelayanan dapat menjadi tidak efisien dan berisiko terjadi kesalahan dalam pemrosesan pesanan. Untuk itu, dibutuhkan suatu sistem yang dapat mencatat dan mengelola transaksi secara berurutan dan efisien.

Tema pencatatan transaksi dipilih karena merupakan salah satu studi kasus nyata yang paling umum dalam dunia perdagangan, terutama di sektor makanan dan minuman. Selain itu, proyek ini memungkinkan implementasi langsung dari konsep struktur data Queue, yaitu antrian yang bekerja dengan prinsip *First In, First Out (FIFO)*, yang sangat relevan untuk memproses transaksi pelanggan secara berurutan sesuai waktu kedatangan.

Tujuan:

Tujuan utama dari pengembangan aplikasi ini adalah untuk membangun sebuah sistem pencatatan transaksi yang mampu:

- Mencatat data transaksi dari pelanggan, termasuk beberapa pesanan sekaligus.
- Menyimpan dan menampilkan antrian transaksi secara real-time.
- Memproses transaksi secara berurutan sesuai prinsip FIFO.
- Menyediakan fitur untuk melihat total pendapatan
- melakukan pencarian transaksi berdasarkan nama pelanggan.

Manfaat:

Dengan adanya aplikasi ini, diharapkan proses pencatatan dan pengelolaan transaksi di restoran dapat berjalan lebih tertib, cepat, dan akurat. Selain itu, aplikasi ini juga dapat menjadi media implementasi struktur data antrian (*Queue*) secara nyata dalam dunia pemrograman.

Struktur Data Utama:

Struktur data yang digunakan dalam pengembangan sistem ini adalah Queue berbasis Linked List, yang memungkinkan penambahan dan penghapusan elemen secara efisien dari ujung antrian.

1.2. Lingkup Pengerjaan

Ruang Lingkup:

Aplikasi ini dikembangkan dengan fitur-fitur utama sebagai berikut:

- **Tambah Transaksi (Enqueue):**
Pengguna dapat memasukkan data transaksi berupa nama pelanggan dan daftar menu yang dipesan (bisa lebih dari satu menu), lengkap dengan jumlah dan harga satuan.
- **Lihat Antrian Transaksi:**
Menampilkan seluruh transaksi dalam antrian, beserta rincian menu, jumlah, harga satuan, dan total harga per transaksi.
- **Proses Transaksi (Dequeue):**
Mengeluarkan transaksi yang berada paling depan dalam antrian dan menambahkannya ke total pendapatan.
- **Lihat Total Pendapatan:**
Menampilkan akumulasi total pendapatan dari semua transaksi yang telah diproses.
- **Cari Transaksi Berdasarkan Nama Pelanggan:**
Melakukan pencarian data transaksi dalam antrian berdasarkan nama pelanggan.

Batasan:

- Sistem hanya menyimpan data selama aplikasi berjalan (belum ada fitur penyimpanan ke file atau database).
- Setiap transaksi hanya bisa diproses satu per satu dan tidak dapat dikembalikan setelah dikeluarkan dari antrian.
- Belum terdapat fitur untuk menghapus atau mengedit transaksi yang belum diproses.

Asumsi:

- Setiap pelanggan dapat melakukan pemesanan lebih dari satu menu dalam satu transaksi.
- Antrian transaksi selalu diproses secara berurutan (FIFO).
- Harga satuan dan jumlah pesanan dimasukkan secara benar oleh pengguna.
- Transaksi yang telah diproses dianggap telah selesai dan tidak akan diubah kembali.

1.3. Dokumentasi Pengerjaan

Proyek tugas besar ini disimpan di repository GitHub dengan akses publik. Repository ini berisi seluruh kode sumber aplikasi pencatatan transaksi restoran berbasis Java menggunakan struktur data Queue. Struktur folder telah disusun sesuai dengan package Java, serta seluruh fitur yang telah dijelaskan sebelumnya dapat ditemukan dan diuji langsung dari kode di repository ini.

Link Repository:

https://github.com/Tizarhasan/PP12025_D_Cetalogy

Panduan Akses Repository:

1. Buka link repository di atas melalui browser.
2. Klik tombol “Code” dan pilih opsi Download ZIP untuk mengunduh keseluruhan proyek, atau
3. Gunakan Git dengan perintah berikut untuk meng-clone repository:

```
git clone https://github.com/Tizarhasan/PP12025_D_Catalogy.git
```

4. Buka folder proyek di IDE Java.
5. Jalankan file MainApp.java yang berada di dalam package App untuk menjalankan aplikasi.

1.4. Pembagian Tugas

NPM	Nama	Tugas yang Dikerjakan
233040029	Tyezar Hasan Khusaeni	Membuat Main Program
233040001	Kresna Satria Dewantoro	Struktur Data Queue enqueue() , dequeue() , front() , isEmpty() , dan size()
233040017	Aqillah Lean Andera	Membuat kelas Transaksi
233040036	Moch Sachrul Akbar Ramadhan	Struktur Data Queue displayQueue() dan cariTransaksi()

2. Analisis dan Perancangan

2.1. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem bertujuan untuk mengidentifikasi secara rinci apa saja yang harus disediakan oleh sistem agar dapat berjalan sesuai dengan fungsinya. Kebutuhan dibagi menjadi dua bagian, yaitu kebutuhan fungsional dan non-fungsional.

2.1.1. Kebutuhan Fungsional

Aplikasi pencatatan transaksi restoran ini menggunakan struktur data Queue (antrian) dengan prinsip FIFO (First In, First Out). Berikut adalah kebutuhan fungsional sistem:

- **Menambah Transaksi (Enqueue)**
 - Pengguna dapat menambahkan data transaksi berupa:
 - Nama pelanggan
 - Daftar menu (lebih dari satu menu)
 - Jumlah pesanan untuk setiap menu
 - Harga satuan untuk setiap menu
 - Sistem akan menghitung dan menyimpan total harga transaksi.
- **Melihat Antrian Transaksi**
 - Menampilkan seluruh antrian transaksi dari yang pertama hingga terakhir masuk.
 - Menampilkan informasi transaksi secara detail: nama pelanggan, daftar menu, jumlah, harga satuan, dan total harga.
- **Memproses Transaksi (Dequeue)**
 - Menghapus transaksi paling awal dari antrian ketika transaksi telah selesai diproses.
 - Menambahkan nilai total harga transaksi yang diproses ke total pendapatan harian.
- **Melihat Total Pendapatan**
 - Menampilkan jumlah pendapatan dari seluruh transaksi yang telah diproses.

- **Mencari Transaksi Berdasarkan Nama Pelanggan**

- Mencari transaksi di dalam antrian berdasarkan nama pelanggan.
- Menampilkan informasi transaksi jika ditemukan.

2.1.2. Kebutuhan Non Fungsional

Berikut ini adalah kebutuhan non-fungsional yang mendukung kinerja dan kenyamanan pengguna dalam menjalankan aplikasi:

- **Antarmuka Console yang Mudah Digunakan**

- Aplikasi dirancang berbasis teks (console-based) dengan menu yang sederhana dan mudah dimengerti.

- **Validasi Input**

- Validasi input dilakukan untuk memastikan data yang dimasukkan sesuai dengan format yang diharapkan (misalnya input angka untuk jumlah dan harga).
- Input nama pelanggan dan menu tidak boleh kosong.

- **Kecepatan dan Responsif**

- Karena menggunakan struktur data Queue berbasis linked list sederhana, aplikasi dapat merespon dengan cepat dalam operasi enqueue dan dequeue tanpa penundaan.

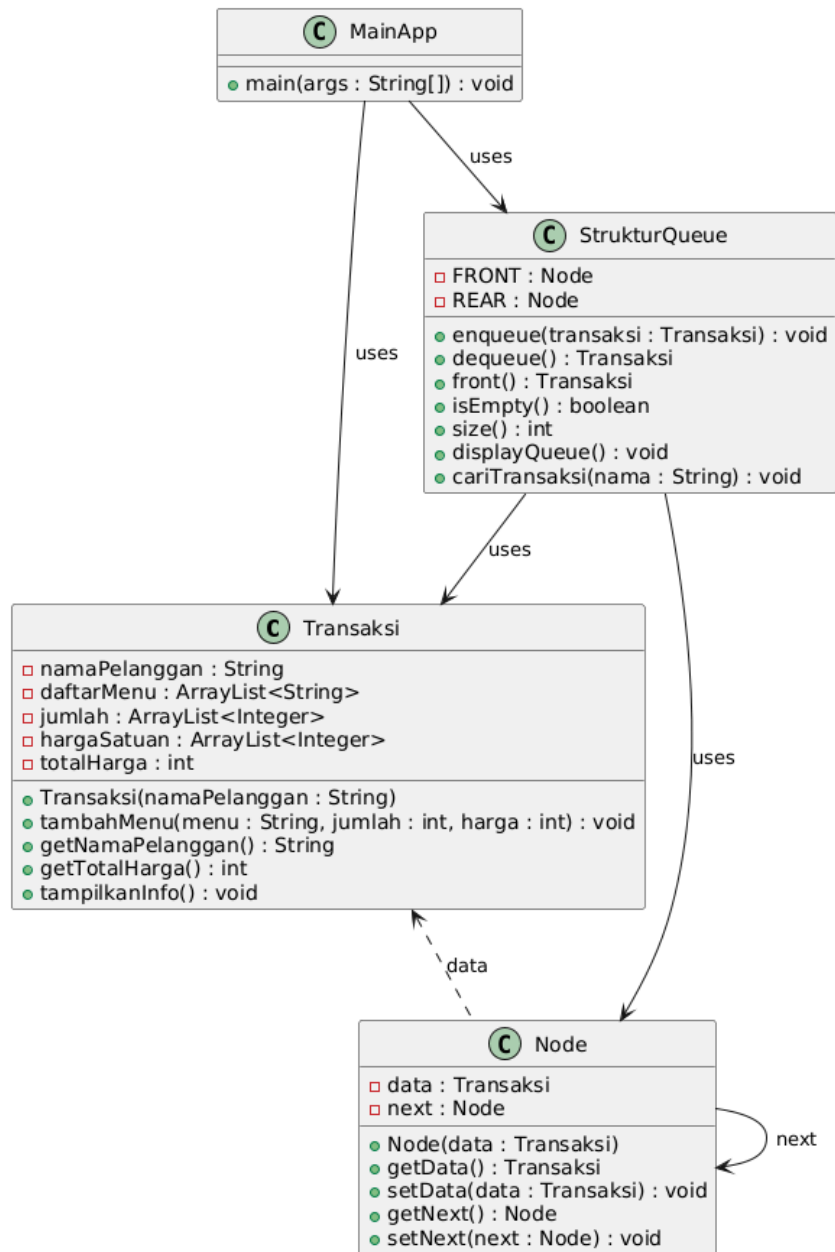
- **Error Handling**

- Sistem menangani beberapa kondisi kesalahan, seperti:
 - Menampilkan pesan jika transaksi kosong ketika ingin diproses.
 - Menampilkan pesan ketika nama pelanggan tidak ditemukan saat pencarian.

- **Efisiensi Memori**

- Data transaksi disimpan menggunakan linked list dinamis melalui objek Node, sehingga memori digunakan secara efisien sesuai jumlah data yang dimasukkan.

2.2. Perancangan Struktur Data

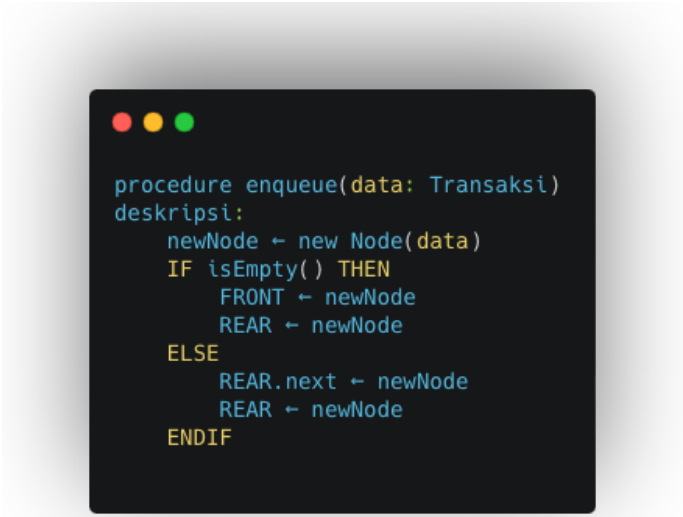


2.3. Perancangan Algoritma

Dalam aplikasi ini, kami mengimplementasikan beberapa algoritma utama yang berkaitan dengan operasi dasar struktur data Queue dan pencarian data transaksi berdasarkan nama pelanggan. Berikut penjelasan algoritma-algoritma utama dalam sistem:

1. Algoritma Enqueue (Menambahkan Transaksi ke Antrian)

Algoritma ini digunakan untuk menambahkan objek transaksi ke ujung antrian. Operasi ini sesuai prinsip FIFO (First In First Out).

A code editor window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains the following pseudocode for the enqueue procedure:

```
procedure enqueue(data: Transaksi)
deskripsi:
    newNode ← new Node(data)
    IF isEmpty() THEN
        FRONT ← newNode
        REAR ← newNode
    ELSE
        REAR.next ← newNode
        REAR ← newNode
    ENDIF
```

2. Algoritma Dequeue (Memproses dan Menghapus Transaksi Paling Depan)

Operasi ini mengambil dan menghapus data transaksi paling depan dari antrian.

```
procedure dequeue()
deskripsi:
  IF isEmpty() THEN
    return null
  ELSE
    hasil ← FRONT.data
    IF FRONT == REAR THEN
      FRONT ← null
      REAR ← null
    ELSE
      FRONT ← FRONT.next
    ENDIF
    return hasil
  ENDIF
```

3. Algoritma DisplayQueue (Menampilkan Semua Transaksi dalam Antrian)

Algoritma ini digunakan untuk menampilkan seluruh isi antrian secara berurutan dari yang paling awal masuk hingga terakhir.

```
procedure displayQueue()
deskripsi:
  IF isEmpty() THEN
    cetak "Antrian kosong."
  ELSE
    current ← FRONT
    nomor ← 1
    WHILE current ≠ null DO
      cetak "Transaksi #" + nomor
      tampilkan data current
      current ← current.next
      nomor ← nomor + 1
    ENDWHILE
  ENDIF
```

4. Algoritma Cari Transaksi Berdasarkan Nama Pelanggan

Fungsi ini melakukan pencarian linear untuk menemukan transaksi berdasarkan nama pelanggan.

```
procedure cariTransaksi(namaDicari: string)
deskripsi:
  current ← FRONT
  ditemukan ← false
  WHILE current ≠ null DO
    IF current.data.namaPelanggan = namaDicari THEN
      cetak "Transaksi ditemukan"
      tampilkan data current
      ditemukan ← true
      break
    ENDIF
    current ← current.next
  ENDWHILE

  IF NOT ditemukan THEN
    cetak "Transaksi tidak ditemukan"
  ENDIF
```

3. Implementasi Perangkat Lunak

3.1. Kakas dan Perangkat Lunak

3.1.1. Spesifikasi Kakas/Hardware

Perangkat keras yang digunakan dalam pengembangan aplikasi ini adalah sebagai berikut:

- MacBook Pro 2017
 - Processor : Intel Core i7
 - RAM : 16GB
 - Storage : HDD 1TB
 - OS : macOS Ventura

- MSI Thin GF63
 - Processor : Intel Core i7 12550H
 - RAM : 8GB
 - Storage : SSD 512GB
 - OS : Windows 10

- Asus Rog strix g15 g513rm
 - Processor : AMD Ryzen 7 6800H
 - RAM : 16GB
 - Storage : SSD 1TB
 - OS : Windows 11

- Asus Vivobook X415DAP
 - Processor : AMD Ryzen 3 3250U
 - RAM : 8GB
 - Storage : SSD 512GB
 - OS : Windows 11

3.1.2. Daftar Perangkat Lunak

Berikut adalah perangkat lunak dan tools yang digunakan dalam pengembangan aplikasi:

- IDE (Integrated Development Environment):
VSCode
- Java Version:
Java Development Kit (JDK) 23.0.2
- Version Control:

Git (untuk pengelolaan versi lokal)

GitHub (untuk penyimpanan dan kolaborasi proyek)
- Tools Lainnya :

Plantuml (untuk membuat diagram perancangan struktur data)

carbon.sh.now (untuk tampilan kode agar lebih nyaman dilihat)

3.2. Struktur Proyek

Struktur direktori dari proyek ini disusun dengan struktur sebagai berikut:

```
PP12025_D_Catalogy/
├── src/
│   ├── entity/          # Berisi kelas-kelas model data transaksi dan node queue
│   │   ├── Node.java
│   │   └── Transaksi.java
│   ├── services/        # Berisi kelas yang mengelola logika struktur data Queue
│   │   └── StrukturQueue.java
│   └── MainApp.java      # Kelas utama tempat program dijalankan
├── docs/                 # Berisi dokumentasi proyek, laporan, dan diagram
│   └── laporan_tugas_besar.pdf
└── README.md             # Panduan umum mengenai proyek
```

3.3. Implementasi Kelas dan Struktur Data

No.	Nama File	Package	Deskripsi Fungsi
1	MainApp	Service	<ul style="list-style-type: none"> - Menjalankan Program Utama - Menyediakan Menu Interaktif - Validasi Input Pengguna
2	Node	Entity	<ul style="list-style-type: none"> - Menyimpan objek Transaksi - Mengembalikan data Transaksi - Mengubah isi data Transaksi - Referensi ke node berikutnya dalam antrian - Menentukan siapa node berikutnya dalam antrian - Menunjuk ke node berikutnya dalam antrian (Linked List)
3	Transaksi	Entity	<ul style="list-style-type: none"> - Menyimpan nama pelanggan, menu yang dipesan, jumlah yang dipesan, harga satuan, total harga - Menambahkan item pesanan dan menghitung total harga - Mengambil nama pelanggan dari

No.	Nama File	Package	Deskripsi Fungsi
			<p>transaksi dan total harga dari seluruh pesanan</p> <ul style="list-style-type: none"> - Menampilkan semua detail transaksi (menu, jumlah, harga, total)
4	StrukturQueue	Service	<ul style="list-style-type: none"> - Menambahkan transaksi ke akhir antrian - Menghapus dan mengembalikan transaksi paling depan (yang paling lama) - Mengecek apakah antrian kosong - Menghitung jumlah transaksi dalam antrian - Menampilkan semua transaksi dalam antrian - Mencari dan menampilkan transaksi berdasarkan nama pelanggan

Penjelasan Struktur Data:

Struktur data utama yang digunakan dalam aplikasi pencatatan transaksi ini adalah Queue berbasis Linked List dan Array List. Struktur ini dipilih karena sesuai dengan kebutuhan sistem yang memproses transaksi secara berurutan (FIFO – *First In, First Out*), di mana transaksi yang lebih dahulu masuk akan lebih dahulu diproses.

Potongan kode penting

```
package services;

import entity.Node;
import entity.Transaksi;

public class StrukturQueue {
    private Node FRONT, REAR;

    public void enqueue(Transaksi transaksi) {
        Node newNode = new Node(transaksi);
        if (isEmpty()) {
            FRONT = newNode;
            REAR = newNode;
        } else {
            REAR.setNext(newNode);
            REAR = newNode;
        }
    }

    public Transaksi dequeue() {
        Transaksi hasil = null;
        if (FRONT != null) {
            hasil = FRONT.getData();
            if (FRONT == REAR) {
                FRONT = null;
                REAR = null;
            } else {
                FRONT = FRONT.getNext();
            }
        }
        return hasil;
    }

    public Transaksi front() {
        if (FRONT != null) {
            return FRONT.getData();
        } else {
            return null;
        }
    }
}
```

```

    public boolean isEmpty() {
        return FRONT == null;
    }

    public int size() {
        int count = 0;
        Node current = FRONT;
        while (current != null) {
            count++;
            current = current.getNext();
        }
        return count;
    }

    public void displayQueue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
        } else {
            Node current = FRONT;
            int nomor = 1;
            while (current != null) {
                System.out.println("Transaksi #" + nomor++);
                current.getData().tampilkanInfo();
                System.out.println("-----");
                current = current.getNext();
            }
            System.out.println("Total antrian: " + size());
        }
    }

    public void cariTransaksi(String namaDicari) {
        Node current = FRONT;
        boolean ditemukan = false;

        while (current != null) {
            Transaksi data = current.getData();
            if (data.getNamaPelanggan().equalsIgnoreCase(namaDicari)) {
                System.out.println("Transaksi ditemukan:");
                data.tampilkanInfo();
                ditemukan = true;
                break; // berhenti setelah ditemukan
            }
            current = current.getNext();
        }

        if (!ditemukan) {
            System.out.println("Transaksi atas nama \"" + namaDicari + "\" tidak ditemukan dalam antrian.");
        }
    }
}

```

```

package entity;

public class Node {
    private Transaksi data;
    private Node next;

    public Node(Transaksi data){
        this.data = data;
    }

    public Transaksi getData() {
        return data;
    }

    public void setData(Transaksi data) {
        this.data = data;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }
}

```

```

package entity;

import java.util.ArrayList;

public class Transaksi {
    private String namaPelanggan;
    private ArrayList<String> daftarMenu;
    private ArrayList<Integer> jumlah;
    private ArrayList<Integer> hargaSatuan;
    private int totalHarga;

    public Transaksi(String namaPelanggan) {
        this.namaPelanggan = namaPelanggan;
        this.daftarMenu = new ArrayList<>();
        this.jumlah = new ArrayList<>();
        this.hargaSatuan = new ArrayList<>();
        this.totalHarga = 0;
    }

    public void tambahMenu(String menu, int jumlahPesanan, int harga) {
        daftarMenu.add(menu);
        jumlah.add(jumlahPesanan);
        hargaSatuan.add(harga);
        totalHarga += jumlahPesanan * harga;
    }

    public String getNamaPelanggan() {
        return namaPelanggan;
    }

    public int getTotalHarga() {
        return totalHarga;
    }

    public void tampilkanInfo() {
        System.out.println("Nama Pelanggan: " + namaPelanggan);
        for (int i = 0; i < daftarMenu.size(); i++) {
            System.out.println("- " + daftarMenu.get(i) + " | Jumlah: " + jumlah.get(i) + " | Harga Satuan: " +
            hargaSatuan.get(i));
        }
        System.out.println("Total Harga: " + totalHarga);
    }
}

```

3.4. Fitur-Fitur Aplikasi

- Fitur Tambah Transaksi

Deskripsi: Menambahkan transaksi ke antrian (enqueue)

Cara Kerja:

- Program meminta pengguna memasukkan nama pelanggan
- Dibuat objek Transaksi baru berdasarkan nama tersebut
- Pengguna diminta memasukkan nama menu, jumlah pesanan, dan harga satuan
- Input divalidasi agar jumlah > 0 dan harga ≥ 0
- Program menyimpan data menu ke dalam objek Transaksi melalui method tambahMenu()
- Pertanyaan "Tambah menu lagi?(y/n)"
- Jika jawab "y", pengguna bisa menambahkan menu, jumlah, dan harga berikutnya
- Jika jawab "n", lanjut ke proses berikutnya
- Objek Transaksi dimasukkan ke dalam struktur StrukturQueue
- Transaksi akan menunggu untuk diproses secara FIFO (First In, First Out)
- Input: Nama pelanggan, nama menu, jumlah pesanan, harga satuan

Output: "Transaksi berhasil ditambahkan"

Screenshot:

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 1
Nama pelanggan: Kresna
Nama menu: Nasi Goreng
Jumlah pesanan: 1
Harga satuan: 15000
Tambah menu lagi? (y/n): n
Transaksi berhasil ditambahkan.
```

- Fitur Lihat Antrian Transaksi

Deskripsi: Melihat antrian transaksi yang sudah di tambahkan

Cara Kerja:

- Program memanggil method displayQueue() dari objek queue
- Method isEmpty() akan mengecek apakah node FRONT bernilai null
- Jika kosong, langsung tampilkan “Antrian Kosong”
- Jika antrian tidak kosong Iterasi dilakukan dari FRONT hingga REAR.
- Setiap Node berisi satu Transaksi, dan program akan memanggil tampilkanInfo() dari setiap transaksi
- Di akhir tampilan, sistem mencetak jumlah total antrian saat ini

Input: -

Output:

Jika antrian kosong: “Antrian Kosong”

Jika ada transaksi dalam antrian: “Total antrian: [Jumlah Antrian]”

Screenshot:

- Jika Antrian Kosong

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 2

== Daftar Antrian Transaksi ==
Antrian kosong.
```

- Jika Ada Transaksi Didalam Antrian

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 2

== Daftar Antrian Transaksi ==
Transaksi #1
Nama Pelanggan: Kresna
- Nasi Goreng | Jumlah: 2 | Harga Satuan: 10000
Total Harga: 20000
-----
Total antrian: 1
```

- Fitur Proses Transaksi

Deskripsi: Memproses Transaksi yang ada di antrian

Cara Kerja:

- Program akan menjalankan method dequeue() pada objek queue
- Di dalam kelas StrukturQueue transaksi yang berada di paling depan (FRONT) diambil
- Di dalam kelas StrukturQueue Jika hanya satu transaksi di antrian, maka FRONT dan REAR diset ke null
- Di dalam kelas StrukturQueue Jika lebih dari satu, FRONT digeser ke node berikutnya
- Jika dequeue() berhasil (antrian tidak kosong), data transaksi ditampilkan
- Total harga dari transaksi yang baru diproses ditambahkan ke variabel totalPendapatan.
- Jika dequeue() mengembalikan null, maka tampil pesan bahwa tidak ada transaksi untuk diproses

Input: -

Output:

- Jika tidak ada transaksi (antrian kosong):
“Tidak ada transaksi untuk diproses”
- Jika ada transaksi dalam antrian:
Nama Pelanggan: [Nama Pelanggan]
Menu | Jumlah | Harga Satuan
Total Harga: [Jumlah]

Screenshot:

- Jika tidak ada transaksi (antrian kosong)

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Tidak ada transaksi untuk diproses.
```

- Jika ada transaksi dalam antrian

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Transaksi diproses:
Nama Pelanggan: Kresna
- Nasi Ayam gulai | Jumlah: 2 | Harga Satuan: 20000
Total Harga: 40000

```

- Fitur Lihat Total Pendapatan

Deskripsi: Melihat total pembayaran yang sudah diproses

Cara Kerja:

- Program langsung menampilkan nilai dari variabel totalPendapatan
- Dikalkulasi secara otomatis setiap kali transaksi diproses (di-dequeue) lewat menu 3

Input: -

Output: “Total pendapatan hari ini: Rp [Jumlah]”

Screenshot:

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 4
Total pendapatan hari ini: Rp 66000

```

- Fitur Cari Transaksi Berdasarkan Nama Pelanggan

Deskripsi: Mencari transaksi yang sudah diproses berdasarkan nama pelanggan

Cara Kerja:

- Program meminta input nama pelanggan dari pengguna
- Nama yang diinput akan dikirim ke method queue.cariTransaksi(namaCari)
- Dalam method cariTransaksi() di kelas StrukturQueue, program melakukan Iterasi dari node FRONT hingga REAR pada queue kemudian Mengecek apakah

nama pelanggan cocok (equalsIgnoreCase). Jika cocok menampilkan detail transaksi menggunakan tampilkanInfo() lalu menampilkan pesan "Transaksi ditemukan" dan berhenti mencari (dengan break).

- Jika tidak ditemukan hingga akhir maka menampilkan pesan bahwa nama tidak ditemukan

Input: String Nama pelanggan

Output:

- Jika transaksi ditemukan
Transaksi ditemukan:
Nama pelanggan :
Nama menu | Jumlah | Harga satuan
Total Harga
- Jika transaksi tidak ditemukan
Transaksi atas nama "[Nama]" tidak ditemukan dalam antrian.

Screenshot:

- Jika Transaksi Ditemukan

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 5
Masukkan nama pelanggan yang ingin dicari: Kresna
Transaksi ditemukan:
Nama Pelanggan: Kresna
- Nasi Rendang | Jumlah: 2 | Harga Satuan: 29000
Total Harga: 58000
```

- Jika Transaksi Tidak Ditemukan

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 5
Masukkan nama pelanggan yang ingin dicari: Kresna
Transaksi atas nama "Kresna" tidak ditemukan dalam antrian.
```

Kesimpulan

kesimpulan dari pengerjaan tugas besar ini, meliputi:

Pencapaian:

- Fitur-fitur yang berhasil diimplementasikan
Fitur-fitur utama seperti tambah transaksi, lihat antrian, proses transaksi, cari transaksi, dan lihat total pendapatan berhasil diimplementasikan dan juga sudah dicoba.
- Struktur data yang berhasil diterapkan
Struktur data Queue berbasis Linked List berhasil diterapkan untuk mendukung proses FIFO (First In First Out).
- Pembelajaran yang diperoleh
Kami memperoleh pembelajaran penting mengenai implementasi struktur data dinamis dan manajemen antrian dalam konteks aplikasi nyata.

Tantangan yang Dihadapi:

- Kesulitan teknis yang ditemui
Ada beberapa kesulitan teknis yang dialami, seperti logika penambahan dan penghapusan elemen pada Queue (kesalahan syntax), serta validasi input dari pengguna. Kami juga sulit mengatur waktu untuk diskusi secara tatap muka.
- Cara mengatasi masalah tersebut
Karena kami sulit untuk mengatur waktu diskusi tatap muka kami berdiskusi melalui platform discord, debugging setiap ada update codingan. Setiap ada progress pengerjaan kami selalu terhubung melalui discord terlebih dahulu agar bisa berkomunikasi dengan jelas.

Evaluasi:

- Kelebihan dan kekurangan sistem
 - Kelebihan :
Sistem berjalan sesuai konsep FIFO, sederhana, dan cukup efisien untuk aplikasi console.

- Kekurangan :

Data tidak disimpan secara permanen, belum bisa mengedit atau menghapus transaksi yang belum diproses, serta belum menangani validasi input secara menyeluruh.

● Potensi pengembangan lebih lanjut

Penambahan penyimpanan data menggunakan file/database, tampilan antarmuka grafis, serta fitur tambahan seperti pengelolaan histori transaksi berdasarkan tanggal

Lampiran

A. Kode Sumber

Potongan kode penting dari setiap kelas

- Kelas Node

```
package entity;

public class Node {
    private Transaksi data;
    private Node next;

    public Node(Transaksi data){
        this.data = data;
    }

    public Transaksi getData() {
        return data;
    }

    public void setData(Transaksi data) {
        this.data = data;
    }

    public Node getNext() {
        return next;
    }

    public void setNext(Node next) {
        this.next = next;
    }
}
```

- Kelas StrukturQueue

```
package services;

import entity.Node;
import entity.Transaksi;

public class StrukturQueue {
    private Node FRONT, REAR;

    public void enqueue(Transaksi transaksi) {
        Node newNode = new Node(transaksi);
        if (isEmpty()) {
            FRONT = newNode;
            REAR = newNode;
        } else {
            REAR.setNext(newNode);
            REAR = newNode;
        }
    }

    public Transaksi dequeue() {
        Transaksi hasil = null;
        if (FRONT != null) {
            hasil = FRONT.getData();
            if (FRONT == REAR) {
                FRONT = null;
                REAR = null;
            } else {
                FRONT = FRONT.getNext();
            }
        }
        return hasil;
    }

    public Transaksi front() {
        if (FRONT != null) {
            return FRONT.getData();
        } else {
            return null;
        }
    }
}
```

```

public boolean isEmpty() {
    return FRONT == null;
}

public int size() {
    int count = 0;
    Node current = FRONT;
    while (current != null) {
        count++;
        current = current.getNext();
    }
    return count;
}

public void displayQueue() {
    if (isEmpty()) {
        System.out.println("Antrian kosong.");
    } else {
        Node current = FRONT;
        int nomor = 1;
        while (current != null) {
            System.out.println("Transaksi #" + nomor++);
            current.getData().tampilkanInfo();
            System.out.println("-----");
            current = current.getNext();
        }
        System.out.println("Total antrian: " + size());
    }
}

public void cariTransaksi(String namaDicari) {
    Node current = FRONT;
    boolean ditemukan = false;

    while (current != null) {
        Transaksi data = current.getData();
        if (data.getNamaPelanggan().equalsIgnoreCase(namaDicari)) {
            System.out.println("Transaksi ditemukan:");
            data.tampilkanInfo();
            ditemukan = true;
            break; // berhenti setelah ditemukan
        }
        current = current.getNext();
    }

    if (!ditemukan) {
        System.out.println("Transaksi atas nama \"" + namaDicari + "\" tidak ditemukan dalam antrian.");
    }
}
}

```

- Kelas Transaksi

```
package entity;

import java.util.ArrayList;

public class Transaksi {
    private String namaPelanggan;
    private ArrayList<String> daftarMenu;
    private ArrayList<Integer> jumlah;
    private ArrayList<Integer> hargaSatuan;
    private int totalHarga;

    public Transaksi(String namaPelanggan) {
        this.namaPelanggan = namaPelanggan;
        this.daftarMenu = new ArrayList<>();
        this.jumlah = new ArrayList<>();
        this.hargaSatuan = new ArrayList<>();
        this.totalHarga = 0;
    }

    public void tambahMenu(String menu, int jumlahPesanan, int harga) {
        daftarMenu.add(menu);
        jumlah.add(jumlahPesanan);
        hargaSatuan.add(harga);
        totalHarga += jumlahPesanan * harga;
    }

    public String getNamaPelanggan() {
        return namaPelanggan;
    }

    public int getTotalHarga() {
        return totalHarga;
    }

    public void tampilkanInfo() {
        System.out.println("Nama Pelanggan: " + namaPelanggan);
        for (int i = 0; i < daftarMenu.size(); i++) {
            System.out.println("- " + daftarMenu.get(i) + " | Jumlah: " + jumlah.get(i) + " | Harga Satuan: " +
            hargaSatuan.get(i));
        }
        System.out.println("Total Harga: " + totalHarga);
    }
}
```

- Kelas MainApp

```
import entity.Transaksi;
import services.StrukturQueue;

import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StrukturQueue queue = new StrukturQueue();

        int totalPendapatan = 0;
        boolean running = true;

        while (running) {
            System.out.println("\n=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===");
            System.out.println("1. Tambah Transaksi");
            System.out.println("2. Lihat Antrian Transaksi");
            System.out.println("3. Proses Transaksi");
            System.out.println("4. Lihat Total Pendapatan");
            System.out.println("5. Cari Transaksi Berdasarkan Nama Pelanggan");
            System.out.println("6. Keluar");

            int pilihan = 0;
            while (true) {
                System.out.print("Pilih menu: ");
                try {
                    pilihan = Integer.parseInt(scanner.nextLine());
                    break;
                } catch (NumberFormatException e) {
                    System.out.println("Input tidak valid. Harap masukkan
angka.");
                }
            }
        }
    }
}
```



```

switch (pilihan) {
    case 1:
        System.out.print("Nama pelanggan: ");
        String nama = scanner.nextLine();
        Transaksi transaksi = new Transaksi(nama);

        boolean tambahMenu = true;
        while (tambahMenu) {
            System.out.print("Nama menu: ");
            String menu = scanner.nextLine();

            int jumlah = 0;
            while (true) {
                System.out.print("Jumlah pesanan: ");
                try {
                    jumlah = Integer.parseInt(scanner.nextLine());
                    if (jumlah > 0) break;
                    else System.out.println("Jumlah harus lebih dari 0.");
                } catch (NumberFormatException e) {
                    System.out.println("Input tidak valid. Masukkan angka yang
benar.");
                }
            }

            int harga = 0;
            while (true) {
                System.out.print("Harga satuan: ");
                try {
                    harga = Integer.parseInt(scanner.nextLine());
                    if (harga >= 0) break;
                    else System.out.println("Harga tidak boleh negatif.");
                } catch (NumberFormatException e) {
                    System.out.println("Input tidak valid. Masukkan angka yang
benar.");
                }
            }

            transaksi.tambahMenu(menu, jumlah, harga);

            System.out.print("Tambah menu lagi? (y/n): ");
            String lagi = scanner.nextLine();
            if (!lagi.equalsIgnoreCase("y")) {
                tambahMenu = false;
            }
        }

        queue.enqueue(transaksi);
        System.out.println("Transaksi berhasil ditambahkan.");
        break;

    case 2:
        System.out.println("\n== Daftar Antrian Transaksi ==");
        queue.displayQueue();
        break;
}

```

```

case 3:
    Transaksi diproses = queue.dequeue();
    if (diproses != null) {
        System.out.println("Transaksi diproses:");
        diproses.tampilkanInfo();
        totalPendapatan += diproses.getTotalHarga();
    } else {
        System.out.println("Tidak ada transaksi untuk diproses.");
    }
    break;

case 4:
    System.out.println("Total pendapatan hari ini: Rp " +
totalPendapatan);    break;

case 5:
    System.out.print("Masukkan nama pelanggan yang ingin dicari: ");
    String namaCari = scanner.nextLine();
    queue.cariTransaksi(namaCari);
    break;

case 6:
    System.out.println("Program selesai. Terima kasih!");
    running = false;
    break;

default:
    System.out.println("Pilihan tidak valid.");
}
}

scanner.close();
}
}

```

Penjelasan singkat implementasi struktur data

Struktur data utama yang digunakan dalam aplikasi ini adalah Queue (antrian) berbasis Linked List dan Array List (supaya bisa memesan lebih dari satu menu). Setiap elemen dalam antrian direpresentasikan oleh Node, yang menyimpan data transaksi dan referensi ke node berikutnya.

Antrian memiliki dua pointer utama:

- front untuk menunjuk ke elemen pertama (yang akan diproses terlebih dahulu).
- rear untuk menunjuk ke elemen terakhir (tempat elemen baru ditambahkan).

Operasi enqueue akan menambahkan node baru di belakang antrian, sedangkan operasi dequeue akan menghapus node dari depan. Pendekatan ini sangat sesuai untuk mencatat dan memproses transaksi pelanggan secara berurutan (FIFO – *First In, First Out*), yang mencerminkan antrean nyata dalam pelayanan restoran.

B. Repository GitHub

Link Repository:

https://github.com/Tizarhasan/PP12025_D_Cetalogy

Panduan Akses Repository:

1. Buka link repository di atas melalui browser.
2. Klik tombol “Code” dan pilih opsi Download ZIP untuk mengunduh keseluruhan proyek, atau
3. Gunakan Git dengan perintah berikut untuk meng-clone repository:

```
git clone https://github.com/Tizarhasan/PP12025_D_Cetalogy.git
```

4. Buka folder proyek di IDE Java.
5. Jalankan file MainApp.java yang berada di dalam package App untuk menjalankan aplikasi.

C. Log Aktivitas GitHub


Activity


dev


All activity


All users


All time


Resolve merge conflict
 **Tizarhasan** pushed 2 commits • c270d48...f3fc488 • 19 minutes ago


Selesaikan merge dan hapus file .class dari repo
 **sachrulakbar21** pushed 3 commits • 65e81f0...c270d48 • 18 hours ago


Validasi input angka agar tidak crash jika input bukan integer
 **Tizarhasan** pushed 2 commits • d3913bc...65e81f0 • 19 hours ago


menambahkan fitur multi-menu
 **Tizarhasan** pushed 3 commits • 34a73f9...d3913bc • 19 hours ago


ubah kalimat
 **Tizarhasan** pushed 1 commit • 93ca985...34a73f9 • yesterday


Tambah menu cari nama
 **Tizarhasan** pushed 2 commits • 2cf2191...93ca985 • yesterday


Tambah Fitur Mencari Nama
 **kresnasatria** pushed 2 commits • d0e4557...2cf2191 • yesterday

create MainApp
 **Tizarhasan** pushed 2 commits • e1858dc...d0e4557 • yesterday

update struktur queue
 **sachrulakbar21** pushed 2 commits • cff2183...e1858dc • yesterday

Commit Kelas StrukturQueue
 **kresnasatria** pushed 4 commits • 8db8d55...cff2183 • yesterday

Kelas Main App
 **AleenBoy** pushed 1 commit • eadb54a...8db8d55 • yesterday

Commit Kelas Transaksi
 **AleenBoy** created this branch • eadb54a • yesterday

D. Screenshot Aplikasi

Tampilan menu utama

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu:
```

Output dari setiap fitur utama

- Output Fitur Tambah Transaksi

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 1
Nama pelanggan: Kresna
Nama menu: Nasi Gulai
Jumlah pesanan: 2
Harga satuan: 24000
Tambah menu lagi? (y/n): n
Transaksi berhasil ditambahkan.
```

- Output Fitur Lihat Antrian Transaksi

- Jika ada antrian

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 2

== Daftar Antrian Transaksi ==
Transaksi #1
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 2 | Harga Satuan: 24000
Total Harga: 48000
-----
Total antrian: 1
```

- Jika tidak ada antrian

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 2

== Daftar Antrian Transaksi ==
Antrian kosong.

```

- Output Proses Transaksi

- Jika di antrian ada transaksi

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Transaksi diproses:
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 2 | Harga Satuan: 24000
Total Harga: 48000

```

- Jika di antrian tidak ada transaksi

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Tidak ada transaksi untuk diproses.

```

- Output Lihat Total Pendapatan

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 4
Total pendapatan hari ini: Rp 24000

```

- Output Cari Transaksi Berdasarkan Nama
 - Jika ada transaksi di antrian

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 5
Masukkan nama pelanggan yang ingin dicari: Kresna
Transaksi ditemukan:
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 1 | Harga Satuan: 24000
Total Harga: 24000

```

- Jika tidak ada transaksi di antrian

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 5
Masukkan nama pelanggan yang ingin dicari: Kresna
Transaksi atas nama "Kresna" tidak ditemukan dalam antrian.

```


- Output Keluar

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 6
Program selesai. Terima kasih!

```

Contoh penggunaan aplikasi

- Tambah transaksi

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 1
Nama pelanggan: Kresna
Nama menu: Nasi Gulai
Jumlah pesanan: 2
Harga satuan: 24000
Tambah menu lagi? (y/n): n
Transaksi berhasil ditambahkan.

```

- Pilih menu nomor 1
- Pengguna menginput nama pelanggan, nama menu, jumlah pesanan, harga satuan
- lalu muncul output “Tambah menu lagi? (y/n)”
- jika pengguna ingin menambah menu lagi tekan “y”, jika tidak tekan “n”

- Cara Lihat Antrian Transaksi

- Pilih menu no 2
- jika ada transaksi di antrian maka akan tampil

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 2

== Daftar Antrian Transaksi ==
Transaksi #1
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 2 | Harga Satuan: 24000
Total Harga: 48000
-----
Total antrian: 1

```

- jika tidak ada transaksi di antrian akan keluar

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Tidak ada transaksi untuk diproses.

```

- Cara Proses Transaksi

- pilih menu no 3
- jika ada transaksi maka akan tampil

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Transaksi diproses:
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 2 | Harga Satuan: 24000
Total Harga: 48000

```

- jika tidak ada transaksi di antrian

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 3
Tidak ada transaksi untuk diproses.

```

- Cara Lihat Total Pendapatan
 - pilih menu no 4
 - akan tampil

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 4
Total pendapatan hari ini: Rp 24000

```

- Cara Cari Transaksi Berdasarkan Nama Pelanggan
 - pilih menu no 5
 - input nama pelanggan yang ingin dicari
 - jika di antrian ada transaksi dengan nama pelanggan yang dicari maka akan tampil

```

=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===
1. Tambah Transaksi
2. Lihat Antrian Transaksi
3. Proses Transaksi
4. Lihat Total Pendapatan
5. Cari Transaksi Berdasarkan Nama Pelanggan
6. Keluar
Pilih menu: 5
Masukkan nama pelanggan yang ingin dicari: Kresna
Transaksi ditemukan:
Nama Pelanggan: Kresna
- Nasi Gulai | Jumlah: 1 | Harga Satuan: 24000
Total Harga: 24000

```

- jika di antrian tidak ada transaksi dengan nama pelanggan yang dicari

```
=== APLIKASI PENCATATAN TRANSAKSI RESTORAN ===  
1. Tambah Transaksi  
2. Lihat Antrian Transaksi  
3. Proses Transaksi  
4. Lihat Total Pendapatan  
5. Cari Transaksi Berdasarkan Nama Pelanggan  
6. Keluar  
Pilih menu: 5  
Masukkan nama pelanggan yang ingin dicari: Kresna  
Transaksi atas nama "Kresna" tidak ditemukan dalam antrian.
```