

LAPORAN AKHIR TUGAS BESAR

PRAKTIKUM PEMOGRAMAN 1

Perdagangan/Daftar Produk Toko Kelontong(Linkedlist)



Dipersiapkan Oleh:

Kelas: A

Ketua: 233040054 - Emmir Fahrezi

Anggota: 233040056 - Hikmat Pandu Raharja

233040069 - Moh Faiz Khairan

233040072 – Andyka Khaerulana

233040062 – Fahri Rizqon Arsiansyah

Nama Asisten Laboratorium:

Dzikri Setiawan, Muhammad Daffa Musyaffa, Muhamad Marsa Nur Jaman

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS PASUNDAN BANDUNG
2025

Daftar Isi

Daftar Isi	2
1. Pendahuluan	3
1.1. Deskripsi.....	3
1.2. Lingkup Pengerjaan.....	3
1.3 Dokumentasi Pengerjaan	6
1.4 Pembagian Tugas.....	7
2. Pendahuluan	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.....	21
3.1 Kakas dan Perangkat Lunak	21
3.1.2. Daftar Perangkat Lunak	22
3.2 Struktur Proyek	23
3.3 Implementasi Kelas dan Struktur Data	23
3.4 Fitur Fitur Aplikasi.....	27
4. Kesimpulan	35
Lampiran	38

1. Pendahuluan

1.1. Deskripsi.

Pada saat era digital pada saat ini, banyak pelaku usaha toko kelontong masih mencatat stok produk secara manual di buku tulis. Cara ini terkadang sering sekali terdapat masalah, seperti ketidak efisienan, dan juga menyulitkan saat ingin mencari produk atau data barang. Maka dari itu, dibutuhkan sebuah solusi digital yang sederhana dan mudah digunakan. Tema yang kami pilih yaitu "Daftar Produk Toko Kelontong" dipilih karena sesuai dengan permasalahan nyata di lingkungan masyarakat dan mudah dipahami sebagai studi kasus pembelajaran pemrograman dasar.

1. Tujuan dari aplikasi ini dikembangkan yaitu dengan tujuan untuk:

- Membantu pengelolaan data produk secara efisien.
- Mempermudah proses pencatatan, pencarian, dan pembaruan produk.
- Mengaplikasikan konsep dasar pemrograman dan struktur data dalam proyek nyata.
- Melatih kemampuan tim dalam menerapkan algoritma dan membuat sistem terstruktur.

2. Manfaat dengan adanya aplikasi ini adalah:

- Pemilik toko kelontong dapat mencatat dan mengelola produk tanpa perlu pengetahuan teknologi yang kompleks.
- Mengurangi risiko kehilangan data akibat pencatatan manual.
- Dapat mencari atau mengupdate data produk dengan cepat dan akurat.

3. Struktur Data Utama.

Aplikasi ini mengimplementasikan struktur data linked list sebagai penyimpanan utama data produk secara fleksibel tanpa perlu menggeser data lain, dan sangat cocok untuk data dinamis seperti inventaris toko.

1.2. Lingkup Pengerjaan.

Proyek ini berfokus pada pengembangan aplikasi sederhana untuk daftar produk di toko kelontong menggunakan bahasa pemrograman dasar dan struktur data linked list.

1. Adapun fitur-fitur yang dikembangkan meliputi:

- **Tambah Produk Sembako (di HEAD)**

Fitur ini memungkinkan pengguna untuk menambahkan data produk baru pada posisi paling awal (HEAD) dalam struktur data linked list. Biasanya digunakan untuk produk kategori sembako yang dianggap prioritas utama.

- **Tambah Produk Makanan dan Minuman Ringan (di TAIL)**

Fitur ini menambahkan produk baru di bagian akhir (TAIL) dari linked list. Dikhususkan untuk kategori makanan dan minuman ringan, produk baru akan ditambahkan setelah semua produk yang sudah ada.

- **Tambah Produk Alat Mandi/Cuci dan lainnya (di tengah)**

Produk dari kategori ini akan ditempatkan di posisi tengah dalam daftar. Perhitungan posisi tengah dilakukan dengan mengambil nilai $(\text{jumlah data} / 2) + 1$, lalu produk disisipkan di sana.

- **Tampilkan Semua Produk**

Fitur ini menampilkan semua produk yang tersimpan di dalam linked list. Informasi yang ditampilkan meliputi nama, kategori, stok, harga, dan total nilai dari masing-masing produk.

- **Update Produk Berdasarkan Posisi**

Pengguna dapat memperbarui informasi produk tertentu berdasarkan posisinya dalam daftar. Posisi diminta dari pengguna, lalu data pada node tersebut diganti dengan data baru yang diinput.

- **Hapus Produk Berdasarkan Posisi**

Produk dapat dihapus dari daftar dengan menyebutkan posisinya. Jika posisi adalah 1, maka produk paling awal (HEAD) akan dihapus. Jika tidak, maka program akan menelusuri node hingga menemukan posisi tersebut untuk dihapus.

- **Cari Produk Berdasarkan Nama/Kategori**

Fitur pencarian ini memungkinkan pengguna untuk mencari produk berdasarkan nama atau kategori. Pencarian bersifat tidak sensitif terhadap huruf kapital, dan dapat menemukan lebih dari satu hasil jika cocok.

- **Simpan Semua Produk ke File**

Data produk yang ada dalam program akan disimpan kembali ke file eksternal produk.txt. Hal ini memungkinkan data untuk dipanggil kembali saat program dijalankan ulang.

- **Keluar Program**

Fitur ini digunakan untuk mengakhiri perulangan menu dan menutup program. Setelah fitur ini dipilih, Scanner juga ditutup untuk mengakhiri input dari pengguna.

2. Batasan.

Dalam pengembangan aplikasi ini, terdapat beberapa keterbatasan sistem, yaitu:

- Aplikasi Berbasis Teks, antarmuka berbasis teks saja, belum mendukung GUI (Graphical User Interface).
- Satu Jenis Struktur Data, hanya menggunakan struktur data linked list untuk menyimpan data produk, tanpa kombinasi dengan struktur data lain seperti array atau tree.
- Tidak Ada Multiuser atau Autentikasi, Aplikasi ini hanya dirancang untuk satu pengguna tanpa login atau hak akses berbeda.
- Validasi Data Terbatas, validasi input dari pengguna masih sederhana dan belum menangani semua kasus kesalahan secara menyeluruh.

3. Asumsi

Dalam perancangan dan pengembangan aplikasi ini, beberapa asumsi yang digunakan antara lain:

- Pengguna adalah Pemilik atau Pengelola Toko Kelontong
Pengguna memiliki pengetahuan dasar dalam menggunakan komputer, seperti mengetik dan memilih menu.
- Jumlah Produk Tidak Terlalu Banyak
Aplikasi diasumsikan digunakan untuk toko kecil hingga menengah, sehingga efisiensi linked list masih mencukupi.
- Penggunaan Satu Perangkat Saja
Aplikasi dijalankan di satu komputer/laptop, tidak mendukung jaringan atau cloud.
- Tidak Ada Kebutuhan Backup atau Restore Data
Karena proyek fokus pada pembelajaran dasar pemrograman, aspek penyimpanan permanen tidak menjadi prioritas utama

1.3 Dokumentasi Pengerjaan

Seluruh hasil pengembangan aplikasi ini telah dikelola melalui sebuah repository GitHub yang dapat diakses secara publik. Repository ini mencakup seluruh kode program aplikasi manajemen produk toko kelontong yang dibangun menggunakan bahasa Java dan struktur data Linked List. Struktur folder dalam proyek telah diatur berdasarkan package Java untuk memudahkan navigasi dan pengelolaan kode. Setiap fitur yang dijelaskan pada laporan ini dapat ditemukan dan dijalankan langsung melalui kode sumber yang tersedia di dalam repository.

Berikut merupakan link menuju repository:

https://github.com/emmirmahrezi/Cibibesor_A_Tubes_PP1

Panduan Akses Repository.

- Buka browser (Google Chrome, Mozilla Firefox, dll).
- Kunjungi link repository berikut:
https://github.com/emmirmahrezi/Cibibesor_A_Tubes_PP1
- Setelah halaman terbuka, user akan melihat tampilan berisi file dan folder proyek Java.
- Untuk mengunduh proyek ke computer user:
 1. Klik tombol hijau “Code”.
 2. Pilih Download ZIP.
 3. Ekstrak file ZIP-nya seperti biasa.
- Untuk meng-clone menggunakan Git (opsional, untuk pengguna Git):
 1. Buka Terminal atau Git Bash.
 2. Ketik perintah berikut:
 1. Bash
 2. Salin
 3. Edit
 4. git clone https://github.com/emmirmahrezi/Cibibesor_A_Tubes_PP1
[1.git](https://github.com/emmirmahrezi/Cibibesor_A_Tubes_PP1)
 5. Tekan Enter.
- Buka folder proyek di IDE seperti IntelliJ IDEA, NetBeans, atau VS Code.

- Masuk ke folder TokoKelontong/linkedlist/.
- Buka dan jalankan file MainApp.java untuk menjalankan programnya.

1.4 Pembagian Tugas

NPM	Nama	Tugas yang Dikerjakan
223040054	Emmir Fahrezi	<ol style="list-style-type: none"> 1. Membuat kelas Node beserta isinya. 2. Membuat kelas Produk beserta isinya. 3. Membuat kelas struktur. 4. Membuat atribut head. 5. Membuat method isEmpty. 6. Membuat method tambahHead. 7. Dan juga tambahTail.
233040056	Hikmat Pandu Raharja	<ol style="list-style-type: none"> 1. Membuat fitur update produk 2. Membuat klas MainApp
233040062	Fahri Rizqon Arsiansyah	<ol style="list-style-type: none"> 1. Membuat fitur tampilkan produk 2. Membuat fitur muat dari file
233040069	Mohamad Faiz Khairan	<ol style="list-style-type: none"> 1. Membuat method tambah produk 2. Membuat method simpanKeFile
233040072	Andyka Khaerulana	<ol style="list-style-type: none"> 1. Membuat fitur hapus produk 2. Membuat fitur cari produk

2. Pendahuluan

2.1. Analisis Kebutuhan Sistem

2.1.1 Kebutuhan Fungsional

1. TambahProduk

Aplikasi ini harus dapat memungkinkan pengguna untuk menambahkan data produk baru ke dalam sistem. Setiap produk memiliki atribut berupa nama produk, harga produk, dan jumlah stok.

Fitur ini sangat penting agar pengguna dapat terus memperbarui daftar produk yang dijual di toko kelontong.

2. Melihat Daftar Produk

Sistem harus mampu menampilkan seluruh data produk yang tersimpan secara rapi. Data yang ditampilkan meliputi nama produk, harga, dan stok yang tersedia. Hal ini memudahkan pengguna dalam memantau dan mengelola barang dagangan yang ada.

3. Menghapus Produk

Fitur ini memungkinkan pengguna untuk menghapus data produk yang sudah tidak dijual atau produk yang sudah habis stoknya. Dengan demikian, data yang ada tetap akurat dan up-to-date.

4. Traversal (Menelusuri Data)

Aplikasi harus mampu menampilkan semua isi linked list secara berurutan dari node pertama (head) hingga node terakhir (tail). Fitur ini digunakan untuk meninjau seluruh data yang sedang tersimpan dalam struktur list.

5. Mengedit Produk

Sistem sebaiknya dilengkapi dengan fitur untuk memperbarui data produk. Hal ini mempermudah pengguna ketika terjadi perubahan harga atau jumlah stok produk di toko kelontong.

6. Validasi Input

Aplikasi harus memiliki fitur validasi input yang memastikan data yang dimasukkan benar dan sesuai format yang diinginkan. Contohnya, nama produk tidak boleh kosong, dan harga serta stok harus berupa angka positif. Validasi input ini sangat penting untuk menjaga integritas data di dalam aplikasi.

Fitur-fitur di atas memastikan bahwa aplikasi dapat memanipulasi data secara dinamis dan efisien menggunakan prinsip dasar dari struktur Linked List.

2.1.2 Kebutuhan Non Fungsional

Agar aplikasi berjalan dengan optimal dan nyaman digunakan, terdapat beberapa aspek teknis yang mendukung kinerja aplikasi, yaitu:

1. Kecepatan Respon Aplikasi

Aplikasi harus memiliki waktu respon yang cepat dalam melakukan operasi seperti insert, delete, search, dan traversal pada linked list. Hal ini penting agar pengguna tidak mengalami jeda atau keterlambatan saat berinteraksi dengan sistem.

2. Interface yang Mudah Digunakan

Aplikasi dirancang dengan antarmuka berbasis console (console-based) yang sederhana dan mudah dipahami. Setiap menu dan perintah ditampilkan secara jelas untuk memandu pengguna dalam menjalankan fungsi-fungsi yang tersedia.

3. Validasi Input yang Memadai

Setiap data yang dimasukkan oleh pengguna harus divalidasi terlebih dahulu untuk memastikan bahwa input sesuai dengan format yang diharapkan (misalnya tidak kosong, bertipe angka jika diperlukan, dll). Validasi ini mencegah terjadinya kesalahan logika saat program dijalankan.

4. Error Handling yang Baik

Aplikasi harus mampu menangani berbagai kemungkinan kesalahan dengan memberikan pesan yang informatif, seperti jika pengguna mencoba menghapus data yang tidak ada, atau mencari data dalam list kosong. Dengan demikian, aplikasi tetap stabil dan tidak mengalami crash.

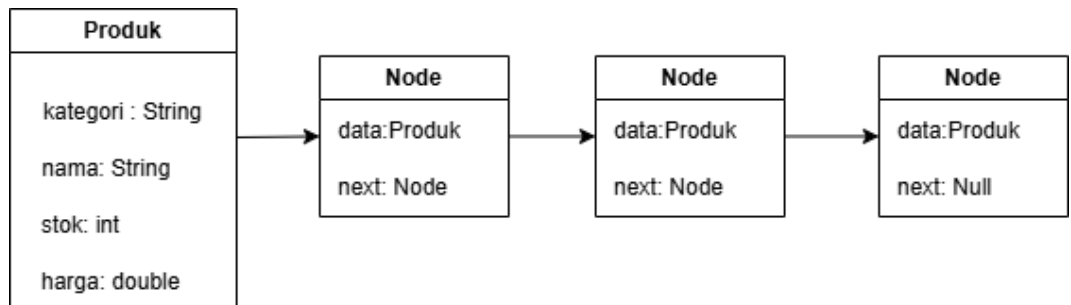
5. Efisiensi Penggunaan Memori

Struktur data linked list secara alami mendukung efisiensi memori karena data disimpan secara dinamis. Aplikasi harus memastikan bahwa node yang tidak digunakan (misalnya setelah delete) dihapus dari memori untuk mencegah memory leak dan menjaga performa sistem.

2.2. Perancangan Struktur Data

Kami menggunakan struktur data Linked List, di mana setiap produk disimpan sebagai sebuah Node yang berisi informasi:

- Kategori
- Nama Produk
- Stok
- Harga Satuan.
- Pointer ke node berikutnya



alasan mengapa memilih Linked List:

- Mudah ditambah/hapus: Tidak perlu menggeser elemen seperti pada array.
- Data dinamis: Cocok kalau jumlah data bisa bertambah/berkurang selama program berjalan.
- Traversal fleksibel: Bisa dengan mudah membaca seluruh isi daftar secara berurutan.

Cara kerja struktur data dalam konteks dalam aplikasi toko kelontong:

- Saat user menambah produk, kamu buat Node baru lalu sambungkan ke akhir list.
- Saat user menghapus produk, kamu cari node-nya lalu putus koneksi dari linked list.
- Saat user menampilkan semua produk, kamu lakukan traversal dari head ke tail, cetak data per node.
- Saat mencari produk, kamu cek satu per satu node, sampai ketemu nama yang sesuai.

2.3. Perancangan Algoritma.

Algoritma tambahMid :

```

procedure tambahMid(data: Produk)
deklarasi
    newNode: Node
    curNode, prevNode: Node
    index, size, mid: Integer

deskripsi
    newNode ← Node(data)
    size ← getSize()
    mid ← size / 2 + 1

    IF (isEmpty() OR mid ≤ 1) THEN
        tambahHead(data)
    ELSE
        curNode ← HEAD
        prevNode ← null
        index ← 1

        WHILE (curNode ≠ null AND
index < mid) DO
            prevNode ← curNode
            curNode ← curNode.next
            index ← index + 1
        ENDWHILE

        IF (prevNode ≠ null) THEN
            prevNode.next ← newNode
            newNode.next ← curNode
        ELSE
            tambahTail(data)
        ENDIF

        Tampilkan "Produk ditambahkan
di tengah."
    ENDIF

```

Penjelasan :

tambahMid(Produk data) berfungsi menambahkan produk di posisi tengah list.

- Hitung ukuran list dan tentukan posisi tengah.
- Kalau list kosong atau posisi tengah ≤ 1 , tambahkan di depan (tambahHead).
- Kalau tidak, cari posisi tengah, lalu sisipkan node baru di sana.
- Jika posisi tengah tidak ditemukan, tambahkan di akhir (tambahTail).
- Tampilkan pesan bahwa produk ditambahkan di tengah.

Algoritma tampilkanProduk:

```
procedure tampilkanProduk
deklarasi
  curNode : Node
  posisi  : integer
  p       : Produk

deskripsi
  IF (isEmpty()) THEN
    cetak("Daftar produk kosong.")
  ELSE
    curNode ← HEAD
    posisi ← 1

    WHILE (curNode ≠ null) DO
      p ← curNode.getData()
      cetak(posisi + ". " + p.getKategori() + " - " + p.getNama() +
            " | Stok: " + p.getStok() + " | Harga: Rp" + p.getHarga())

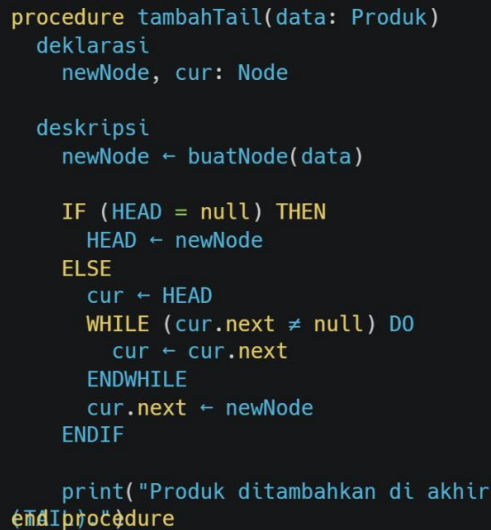
      curNode ← curNode.getNext()
      posisi ← posisi + 1
    ENDWHILE
  ENDIF
```

Penjelasan:

tampilkanProduk() berfungsi untuk menampilkan seluruh data produk yang ada di dalam list secara beruntun

- Cek terlebih dahulu apakah list kosong dengan fungsi isEmpty().
- Jika list kosong,, tampilkan pesan bahwa daftar produk kosong.
- Jika list tidak kosong, mulai dari node pertama (HEAD).
- Ambil data produk dari setiap node, lalu tampilkan informasi kategori, nama produk, stok, dan harga.
- Lanjutkan ke node berikutnya sampai seluruh data dalam list ditampilkan.
- Setiap produk ditampilkan dengan nomor urut berdasarkan posisi dalam list.

Algoritma tambahTail:



```
procedure tambahTail(data: Produk)
  deklarasi
    newNode, cur: Node

  deskripsi
    newNode ← buatNode(data)

    IF (HEAD = null) THEN
      HEAD ← newNode
    ELSE
      cur ← HEAD
      WHILE (cur.next ≠ null) DO
        cur ← cur.next
      ENDWHILE
      cur.next ← newNode
    ENDIF

    print("Produk ditambahkan di akhir")
endprocedure
```

Penjelasan:

Metode tambahTail adalah method yang digunakan untuk menambahkan produk ke bagianpaling akhir (TAIL) dari linked list. Cocok untuk kategori seperti sembako, bumbu dapur, atau kategori lain yang tidak perlu langsung tampil di awal.

Langkah Kerja:

1. Buat node baru
 - Node ini akan menyimpan data produk baru yang ingin ditambahkan ke daftar.
2. Cek apakah list kosong
 - Jika HEAD == null, berarti list masih kosong.
 - Maka, node baru langsung menjadi HEAD (karena dia satu-satunya node).
3. Jika list tidak kosong
 - Lakukan traversal mulai dari HEAD.

- Bergerak ke node berikutnya (`cur = cur.getNext()`) sampai mencapai node terakhir (yang `next`-nya `null`).
- 4. Tambahkan node baru di akhir
 - Setelah berada di node terakhir, arahkan pointer `next` dari node terakhir ke `newNode`.
- 5. Tampilkan pesan ke user
 - Program mencetak: Produk ditambahkan di akhir (TAIL). sebagai notifikasi.

Algoritma `tambahHead`:

```

procedure tambahHead(data: Produk)
  deklarasi
    newNode: Node

  deskripsi
    newNode ← buatNode(data)
    newNode.next ← HEAD
    HEAD ← newNode
    print("Produk ditambahkan di awal (HEAD).")
  end procedure

  ⚡ Penjelasan Singkat:
  buatNode(data) itu maksudnya instansiasi objek baru Node(data)

  newNode.next ← HEAD artinya node baru menunjuk ke node awal sebelumnya
  HEAD ← newNode artinya sekarang node baru jadi kepala (HEAD)

  print(...) buat kasih notifikasi di konsol

  Mau gue buatin juga versi untuk tambahTail, hapusHead, hapusMid, atau yang lainnya dalam format ini?
  Gampang bro tinggal bilang aja 🍷

```

Penjelasan:

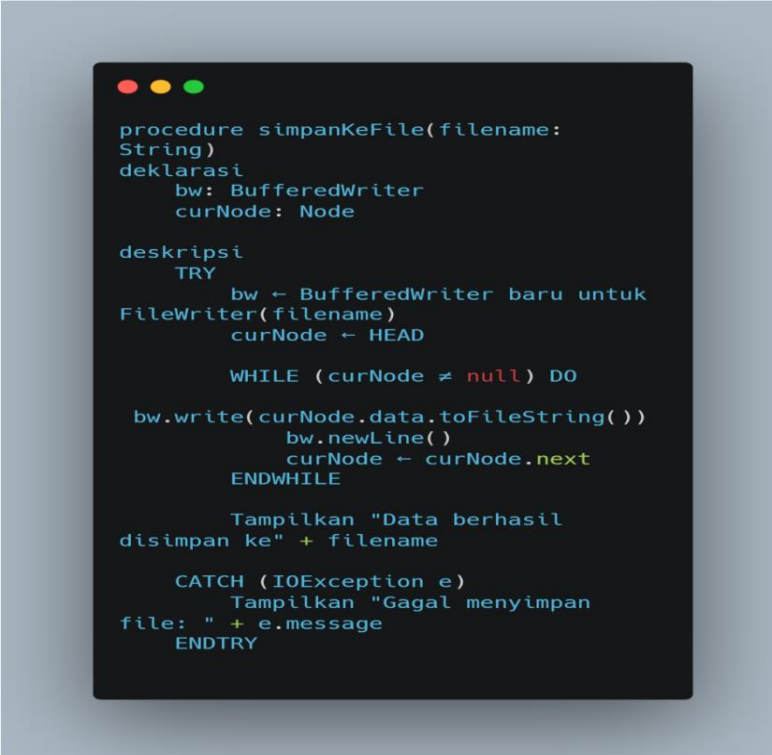
berfungsi untuk menambahkan produk di posisi tengah dari linked list. Cocok digunakan untuk kategori produk yang tidak terlalu sering dibeli maupun tidak terlalu jarang (seperti sabun, alat mandi, alat tulis, dll).

Langkah Kerja:

1. Hitung ukuran list terlebih dahulu.
Dilakukan dengan traversal dari HEAD hingga `null`, sambil hitung jumlah node (`size`).
2. Tentukan posisi tengah.
Misalnya: $\text{posisiTengah} = \text{size} / 2$.

3. Jika list kosong atau posisi tengah ≤ 1
Langsung tambahkan menggunakan tambahHead() karena tidak ada node lain atau hanya satu.
4. Jika list tidak kosong
Lakukan traversal hingga node sebelum posisi tengah.
Sisipkan node baru di antara dua node tersebut:
 - o nodeBaru.next \leftarrow nodeSaatIni
 - o nodeSebelumnya.next \leftarrow nodeBaru
5. Jika traversal gagal (posisi tidak ditemukan)
Tambahkan saja di akhir list pakai tambahTail() sebagai backup.
6. Tampilkan pesan: Produk ditambahkan di posisi tengah.

Algoritma simpanKeFile :



```

procedure simpanKeFile(filename:
String)
deklarasi
    bw: BufferedWriter
    curNode: Node

deskripsi
    TRY
        bw ← BufferedWriter baru untuk
        FileWriter(filename)
        curNode ← HEAD

        WHILE (curNode ≠ null) DO

            bw.write(curNode.data.toFileString())
            bw.newLine()
            curNode ← curNode.next
        ENDWHILE

        Tampilkan "Data berhasil
        disimpan ke" + filename

        CATCH (IOException e)
            Tampilkan "Gagal menyimpan
            file: " + e.message
        ENDTRY
  
```

Penjelasan :

simpanKeFile(String filename) berfungsi menyimpan data produk dari linked list ke file teks.

- Membuka file dengan nama filename untuk ditulis.
- Mulai dari HEAD, baca data setiap node.
- Tulis data tersebut ke file baris per baris menggunakan method `toFileString()`.
- Jika berhasil, tampilkan pesan sukses.
- Jika gagal (terjadi error saat menulis file), tampilkan pesan error.

Algoritma MainApp:

```

Procedure Main
  Buat objek input sebagai Scanner
  Buat objek list sebagai StrukturProduk
  Panggil list.muatDariFile("produk.txt") untuk membaca produk
  dari file

  Ulangi
    Tampilkan menu:
    1. Tambah Produk Sembako (di HEAD)
    2. Tambah Produk Makanan dan Minuman Ringan (di TAIL)
    3. Tambah Produk Alat Mandi/Cuci dan lainnya (di tengah)
    4. Tampilkan semua produk
    5. Update produk berdasarkan posisi
    6. Hapus produk berdasarkan posisi
    7. Cari produk berdasarkan nama/kategori
    8. Simpan semua produk ke file
    9. Keluar

    Baca pilihan dari user

    Pilih berdasarkan pilihan:
    Kasus 1:
      Panggil list.tambahHead(inputProduk(input))
    Kasus 2:
      Panggil list.tambahTail(inputProduk(input))
    Kasus 3:
      data ← inputProduk(input)
      mid ← (list.getSize() / 2) + 1
      Panggil list.tambahMid(data)
      Tampilkan "Produk ditambahkan di tengah"
    Kasus 4:
      Panggil list.tampilkanProduk()
    Kasus 5:
      Minta posisi dari user
      dataBaru ← inputProduk(input)
      Panggil list.updateProduk(posisi, dataBaru)
    Kasus 6:
      Minta posisi dari user
      Panggil list.hapusProduk(posisi)
    Kasus 7:
      Minta kata kunci dari user
      Panggil list.cariProduk(keyword)
    Kasus 8:
      Panggil list.simpanKeFile("produk.txt")
    Kasus 9:
      Tampilkan "Keluar program."
    Default:
      Tampilkan "Pilihan tidak valid."

    Sampai pilihan = 9


  Tutup Scanner input
End Procedure

```

Penjelasan:

Program ini adalah aplikasi manajemen produk toko kelontong berbasis menu yang memungkinkan pengguna untuk: Menambah produk (di awal, akhir, atau tengah list), Menampilkan semua produk, Memperbarui atau menghapus produk berdasarkan posisi, Mencari produk berdasarkan nama/kategori, Menyimpan data ke file, Mengelola produk dengan struktur data linked list

Algoritma UpdateProduk:



```
Procedure updateProduk(posisi: Int, dataBaru: Produk)
  curNode ← HEAD          // Mulai dari node pertama
  index ← 1                // Indeks awal dimulai dari 1

  // Telusuri node sampai posisi ditemukan atau mencapai akhir list
  While curNode ≠ null AND index < posisi do
    curNode ← curNode.next
    index ← index + 1
  End While

  // Jika node ditemukan pada posisi yang dimaksud
  If curNode ≠ null then
    curNode.data ← dataBaru
    Print "Produk di posisi", posisi, "berhasil diperbarui."
  Else
    Print "Posisi tidak ditemukan."
  End If
End Procedure
```

Penjelasan:

Method ini digunakan untuk memperbarui data produk yang berada pada posisi tertentu dalam struktur data Linked List. Cara Kerja: Mulai dari node pertama (HEAD). Telusuri node satu per satu sampai posisi yang diminta. Jika posisi ditemukan: Ganti data node tersebut dengan data baru. Jika tidak ditemukan. Tampilkan pesan bahwa posisi tidak ditemukan.

Algoritma Kelas Node:

```
class Node
  deklarasi
    data: Produk
    next: Node

  constructor Node(data: Produk)
    deskripsi
      this.data ← data
      this.next ← null
    endconstructor

  function getData() → Produk
    deskripsi
      return this.data
    endfunction

  procedure setData(data: Produk)
    deskripsi
      this.data ← data
    endprocedure

  function getNext() → Node
    deskripsi
      return this.next
    endfunction

  procedure setNext(next: Node)
    deskripsi
      this.next ← next
    endprocedure
```

Penjelasan:


Langkah Kerja

1. Menghitung ukuran list
 - Traversal dari head sampai null, sambil menghitung jumlah node.
 - Ini menentukan berapa banyak elemen yang sudah ada.
2. Menentukan posisi tengah
 - Posisi tengah adalah $\text{size} / 2$.
 - Contoh: jika $\text{size} = 4 \rightarrow \text{posisi tengah} = 2$ (disisipkan setelah node ke-2).
3. Kondisi list kosong atau hanya 1 elemen
 - Jika $\text{head} = \text{null}$ atau $\text{posisi tengah} \leq 1$, maka produk disisipkan di awal menggunakan `tambahHead()`.
4. Traversal menuju posisi tengah
 - Loop dari node pertama ke posisi sebelum titik tengah.
 - Simpan referensi `previous` dan `current` untuk menyisipkan node baru di antaranya.
5. Jika gagal menemukan posisi ($\text{current} = \text{null}$)
 - Sebagai backup, produk ditambahkan di akhir list dengan `tambahTail()`.

6. Tampilkan pesan keberhasilan

- Konfirmasi bahwa produk berhasil ditambahkan di tengah.

Algoritma hapusProduk :



```
PROCEDURE hapusProduk(posisi: integer)
  IF list kosong THEN
    PRINT "List kosong."
    RETURN
  ENDIF

  IF posisi = 1 THEN
    HEAD ← HEAD.next
    PRINT "Produk posisi 1 berhasil dihapus."
    RETURN
  ENDIF

  curNode ← HEAD
  prevNode ← NULL
  index ← 1

  WHILE curNode ≠ NULL AND index < posisi DO
    prevNode ← curNode
    curNode ← curNode.next
    index ← index + 1
  ENDWHILE

  IF curNode ≠ NULL THEN
    prevNode.next ← curNode.next
    PRINT "Produk posisi " + posisi + " berhasil dihapus."
  ELSE
    PRINT "Posisi tidak ditemukan."
  ENDIF
ENDPROCEDURE
```

Penjelasan :

Berfungsi untuk menghapus produk dari list berdasarkan posisi tertentu. Pertama, dicek apakah list kosong. Jika iya, tampil pesan bahwa list kosong. Jika posisi yang ingin dihapus adalah posisi pertama, maka HEAD langsung digeser ke node berikutnya. Jika posisi lebih dari satu, maka pointer akan menelusuri list sampai mencapai posisi yang dimaksud. Setelah ditemukan, node tersebut dihapus dengan cara memutus link dari node sebelumnya. Jika posisi tidak ditemukan, maka tampilkan pesan bahwa posisi tidak ditemukan. Prosedur ini menggunakan dua pointer yaitu curNode dan prevNode untuk melacak posisi saat ini dan node sebelumnya selama proses penelusuran. Variabel indeks juga digunakan untuk mencocokkan posisi saat ini dengan posisi yang ingin dihapus. Penghapusan dilakukan secara aman tanpa mengganggu struktur list secara keseluruhan.

Algoritma cariProduk :

A screenshot of a code editor with a dark background and light-colored text. The code is written in a pseudo-language for a linked list. It defines a procedure named 'hapusProduk' that takes a 'posisi' (position) as an integer parameter. The procedure first checks if the list is empty ('list kosong'). If so, it prints 'List kosong.' and returns. If the position is 1, it updates the 'HEAD' to 'HEAD.next', prints 'Produk posisi 1 berhasil dihapus.', and returns. For other positions, it initializes 'curNode' to 'HEAD', 'prevNode' to 'NULL', and 'index' to 1. It then enters a 'WHILE' loop that continues as long as 'curNode' is not NULL and 'index' is less than the 'posisi'. Inside the loop, it updates 'prevNode' to 'curNode' and 'curNode' to 'curNode.next', and increments 'index'. After the loop, it checks if 'curNode' is not NULL. If true, it updates 'prevNode.next' to 'curNode.next', prints 'Produk posisi ' + posisi + ' berhasil dihapus.', and returns. If false, it prints 'Posisi tidak ditemukan.' and returns. The procedure ends with 'ENDPROCEDURE'.

```
PROCEDURE hapusProduk(posisi: integer)
  IF list kosong THEN
    PRINT "List kosong."
    RETURN
  ENDIF

  IF posisi = 1 THEN
    HEAD ← HEAD.next
    PRINT "Produk posisi 1 berhasil dihapus."
    RETURN
  ENDIF

  curNode ← HEAD
  prevNode ← NULL
  index ← 1

  WHILE curNode ≠ NULL AND index < posisi DO
    prevNode ← curNode
    curNode ← curNode.next
    index ← index + 1
  ENDWHILE

  IF curNode ≠ NULL THEN
    prevNode.next ← curNode.next
    PRINT "Produk posisi " + posisi + " berhasil
    dihapus."
  ELSE
    PRINT "Posisi tidak ditemukan."
  ENDIF
ENDPROCEDURE
```

Penjelasan:

Digunakan untuk mencari produk berdasarkan nama atau kategori yang mengandung kata kunci tertentu. Jika list kosong, tampilkan pesan bahwa daftar kosong. Jika tidak, list akan ditelusuri satu per satu. Setiap produk dicek apakah kategori sama dengan keyword atau nama mengandung keyword (abaikan kapital). Jika cocok, tampilkan detail produk lengkap beserta posisi, stok, harga, dan total nilai produk. Jika tidak ditemukan satupun, maka tampilkan pesan bahwa produk tidak ditemukan.

Pencarian dilakukan secara linear dari node pertama hingga terakhir, dan proses ini memungkinkan pencarian fleksibel tanpa memperhatikan huruf kapital atau kata kunci yang tidak lengkap.

Algoritma muatDariFile :



```
procedure muatDariFile(namaBerkas)
deklarasi
    br      : BufferedReader
    baris   : string
    produk  : Produk

deskripsi
    coba
        buka berkas dengan nama 'namaBerkas' sebagai BufferedReader → br

        WHILE (baris ← br.readLine()) ≠ null DO
            produk ← Produk.dariTeks(baris)
            panggil tambahTail(produk)
        ENDWHILE

        cetak("Data berhasil dimuat dari " + namaBerkas)

    jika terjadi kesalahan saat membaca berkas (IOException) THEN
        cetak("Berkas tidak ditemukan. Memulai dari data kosong.")
    END
```

Penjelasan:

Prosedur ini digunakan untuk membaca data produk dari sebuah file teks (biasanya produk.txt) dan kemudian menambahkan setiap data produk ke dalam struktur data yang digunakan program, yaitu linked list.

3. Implementasi perangkat lunak.

3.1 Kakas dan Perangkat Lunak

3.1.1. Spesifikasi Kakas/Hardware

1. Laptop Fahri

- ASUS TUF F15 FX506HC
- Processor : i5-1140H RAM:
- RAM : 16GB
- VGA: RTX3050 4GB
- Memory: 512GB

2. Laptop Dyka

- ACER ASPIRE A514-55G
- Processor : Intel Core i5-1235U
- RAM: 8 GB
- VGA: NVIDIA GeForce MX550 2 GB

- Memory: 512GB

3. Laptop Faiz

- ASUS TUF F15 FX506LH
- Processor : i5-10300H
- RAM : 16GB
- VGA : GTX1650 4GB
- Memory : 512GB

4. Laptop Emmir

- ASUSTeK COMPUTER INC
- Processor : AMD Ryzen 3 7320U
- RAM : 8GB
- VGA : AMD RADEON (TM) Graphics
- Memory : 512GB

5. Laptop Hikmat

- Lenovo ideapad flex 5
- Processor: amd ryzen 5000 series 7 ram: 8GB
- VGA: radeon
- Memory: 512GB

3.1.2. Daftar Perangkat Lunak

- VS Code

Visual Studio Code adalah editor kode sumber yang dikembangkan oleh Microsoft. Perangkat lunak ini digunakan dalam pengerjaan program Java karena memiliki berbagai fitur yang sangat membantu proses pengembangan perangkat lunak

- MS WORD

Microsoft Word digunakan sebagai alat bantu untuk menyusun laporan praktikum atau dokumentasi proyek

IDE: VS Code

Java Version: JDK 23

Version Control: Git, GitHub

Tools lainnya: Carbon.now.sh

3.2 Struktur Proyek

Struktur direktori dari proyek ini disusun dengan struktur sebagai berikut:

Cibibesor_A_Tubes_PP1/

```
├── TUBESPP1/
│   ├── docs/ #berisi dokumentasi proyek
│   │   ├── Laporan_Tugas_Besar.pdf
│   │   └── src/
│   │       ├── entity/ # Kelas kelas model data produk dan juga node linkedlist
│   │       │   ├── Node/ #kelas node
│   │       │   ├── Produk/ #kelas produk
│   │       │   └── Service/ # kelas kelas untuk mengelola logika struktur data linkedlist
│   │       │       ├── StrukturProduk/ #fitur method
│   │       │       └── MainApp/ #kelas utama tempat program dijalankan
│   ├── Produk.txt/ # Dokume produk
│   └── README.md # Panduan mengenai proyek
```

3.3 Implementasi Kelas dan Struktur Data

No.	Nama File	Package	Deskripsi Fungsi
1	Produk	Entity	Model data produk
2	Node	Entity	Node linked list
3	StrukturProduk	Service	Operasi linked list (CRUD)
4	MainApp	Service	Menu utama aplikasi

Implementasi Struktur Data Utama

Program ini menggunakan Linked List sebagai struktur data utama untuk menyimpan dan mengelola data produk. Struktur data ini dikelola melalui dua class utama:

- Node merepresentasikan simpul (node) dalam linked list

- StrukturProduk mengatur seluruh operasi terhadap list, seperti tambah, hapus, cari, update, dan simpan/muat dari file.

Class Node

Setiap node menyimpan objek Produk dan referensi ke node berikutnya.

```
public class Node {
    private Produk data;
    private Node next;

    public Node(Produk data) {
        this.data = data;
        this.next = null;
    }

    public Produk getData() { return data; }
    public void setData(Produk data) { this.data = data; }
    public Node getNext() { return next; }
    public void setNext(Node next) { this.next = next; }
}
```

Class SrtukturProduk

Menambahkan Node ke Awal (Head)

```
public void tambahHead(Produk data) {
    Node newNode = new Node(data);
    newNode.setNext(HEAD);
    HEAD = newNode;
}
```

Menambahkan Node ke Akhir (Tail)

```
public void tambahTail(Produk data) {
    Node newNode = new Node(data);
    if (isEmpty()) {
        HEAD = newNode;
    } else {
        Node cur = HEAD;
        while (cur.getNext() != null) {
            cur = cur.getNext();
        }
        cur.setNext(newNode);
    }
}
```


Menambahkan Node di Tengah

<pre>public void tambahMid(Produk data) { int mid = getSize() / 2 + 1; // traversal untuk cari posisi tengah, lalu sisipkan }</pre>

Menampilkan Semua Produk

<pre>public void tampilkanProduk() { Node curNode = HEAD; while (curNode != null) { Produk p = curNode.getData(); System.out.println(p.getNama()); curNode = curNode.getNext(); } }</pre>

Mengupdate Produk pada Posisi Tertentu
--

<pre>public void updateProduk(int posisi, Produk dataBaru) { Node curNode = HEAD; int index = 1; while (curNode != null && index < posisi) { curNode = curNode.getNext(); index++; } if (curNode != null) { curNode.setData(dataBaru); } }</pre>

Menghapus Produk pada Posisi Tertentu

<pre>public void hapusProduk(int posisi) { if (posisi == 1) { HEAD = HEAD.getNext(); return; } // traversal cari posisi, lalu hapus</pre>

```
}
```

Mencari Produk Berdasarkan Nama/Kategori

```
public void cariProduk(String keyword) {  
    Node curNode = HEAD;  
    while (curNode != null) {  
        Produk p = curNode.getData();  
        if(p.getNama().contains(keyword) || p.getKategori  
().equalsIgnoreCase (keyword)) {  
            System.out.println(p.getNama());  
        }  
        curNode = curNode.getNext();  
    }  
}
```

Simpan dan Muat dari File

Menyimpan ke File

```
public void simpanKeFile(String filename) {  
    Node curNode = HEAD;  
    while (curNode != null) {  
        bw.write(curNode.getData().toFileString());  
        curNode = curNode.getNext();  
    }  
}
```

Memuat dari File

```
public void muatDariFile(String filename) {  
    String line;  
    while ((line = br.readLine()) != null) {  
        Produk p = Produk.fromFileString(line);  
        tambahTail(p);  
    }  
}
```

Kompleksitas Waktu

Operasi	Kompleksitas Waktu	Penjelasan Singkat
tambahHead()	$O(1)$	Langsung ubah HEAD
tambahTail()	$O(n)$	Harus traverse sampai node terakhir
tambahMid()	$O(n)$	Hitung size dan cari posisi tengah
hapusProduk(posisi)	$O(n)$	Cari node di posisi tertentu dan hapus
updateProduk(posisi)	$O(n)$	Cari dan ubah data pada node posisi tertentu
tampilkanProduk()	$O(n)$	Traverse dan tampilkan semua node
cariProduk(keyword)	$O(n)$	Cek satu per satu nama/kategori produk
simpanKeFile()	$O(n)$	Traverse dan tulis ke file
muatDariFile()	$O(n)$	Baca file dan tambah satu per satu ke list

Keterangan:

$O(1)$ – Konstanta (Cepat, Tidak Tergantung Jumlah Data)

- Artinya: waktu eksekusi **tetap**, tidak peduli seberapa besar data.

$O(n)$ – Linear (Semakin Banyak Data, Semakin Lama)

- Artinya: waktu eksekusi **bertambah sebanding** dengan jumlah data.

Harus **melintasi semua node** dari awal sampai akhir.

3.4 Fitur Aplikasi

1. Fitur: Tambah Produk Sembako

Deskripsi: Fitur ini digunakan untuk menambahkan produk sembako baru ke awal (HEAD) dari linkedlist.

Cara Kerja:

1. Pengguna memilih menu “Tambah Produk Sembako”.
2. Program meminta input berupa: kategori, nama produk, stok, dan juga harga.

3. Sistem membuat node baru dengan data tersebut.
4. Node baru diarahkan menunjuk ke HEAD lama.
5. HEAD diperbarui agar menunjuk ke node baru. Input: Kategori, nama produk, stok, harga satuan output: pesan “produk ditambahkan di awal (HEAD).”

Input:

Kategori: 'Sembako'

Nama Produk: Contoh 'Beras Senia'

Stok: Contoh 10

Harga Satuan: Contoh 50000

Output: Hasil yang ditampilkan

Validasi Produk ditambahkan di awal (HEAD).

Screenshot:

```
=== MENU TOKO KELONTONG ===
1. Tambah Produk Sembako
2. Tambah Produk Makanan dan Minuman Ringan
3. Tambah Produk Alat kebutuhan Mandi Mencuci dan lain-lain
4. Tampilkan Produk
5. Update Produk
6. Hapus Produk
7. Cari Produk
8. Simpan ke File
9. Keluar
Pilih menu: 1
```

```
Kategori: Makanan
Nama: Beras senia
Stok: 10
Harga: 50000
Produk ditambahkan di awal (HEAD).
```

```
1. Makanan - Beras senia | Stok: 10 | Harga: Rp50000.0
2. makanan - lays | Stok: 800 | Harga: Rp15000.0
3. makanan - pilus | Stok: 200 | Harga: Rp500.0
4. elektronik - kipas | Stok: 2323 | Harga: Rp90000.0
5. elektronik - lampu | Stok: 100 | Harga: Rp16000.0
6. minuman - ale ale | Stok: 800 | Harga: Rp1000.0
7. minuman - teh botol | Stok: 500 | Harga: Rp6000.0
```

• Fitur: Tambah Produk Makanan dan Minuman Ringan

Deskripsi: Fitur ini digunakan untuk menambahkan produk dari kategori 'Makanan', 'Minuman', atau 'Makanan Ringan Anak' ke posisi akhir(TAIL) dari linked lis

Cara Kerja:

1. Pengguna memilih menu “Tambah Produk (otomatis posisi berdasarkan kategori)”.

2. Program meminta input berupa: kategori, nama produk, stok, dan harga satuan.
3. Sistem mengenali bahwa kategori termasuk dalam 'Makanan', 'Minuman', atau 'Makanan Ringan Anak'.
4. Node baru dibuat dengan data tersebut.
5. Node ditambahkan di bagian akhir list (TAIL).
6. Pointer node terakhir sebelumnya diubah untuk menunjuk node baru.

Input:

Kategori: 'Makanan'

Nama Produk: Contoh 'Taro'

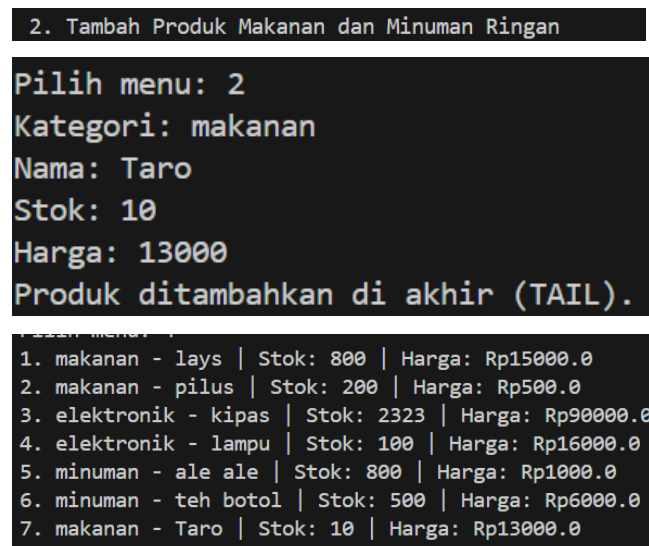
Stok: Contoh 10

Harga Satuan: Contoh 13000

Output:

Pesan "Produk ditambahkan di akhir (TAIL)."

Screenshot:



```

2. Tambah Produk Makanan dan Minuman Ringan

Pilih menu: 2
Kategori: makanan
Nama: Taro
Stok: 10
Harga: 13000
Produk ditambahkan di akhir (TAIL).

1. makanan - lays | Stok: 800 | Harga: Rp15000.0
2. makanan - pilus | Stok: 200 | Harga: Rp500.0
3. elektronik - kipas | Stok: 2323 | Harga: Rp90000.0
4. elektronik - lampu | Stok: 100 | Harga: Rp16000.0
5. minuman - ale ale | Stok: 800 | Harga: Rp1000.0
6. minuman - teh botol | Stok: 500 | Harga: Rp6000.0
7. makanan - Taro | Stok: 10 | Harga: Rp13000.0
  
```

• **Fitur: Tambah Produk Alat kebutuhan Mandi Mencuci dan lain-lain**

Deskripsi: Fitur ini digunakan untuk menambahkan produk dari kategori seperti 'Alat Kebersihan', 'Peralatan Mandi', 'Alat Tulis', dan kategori lainnya yang tidak termasuk ke dalam Head atau Tail. Produk ini

ditambahkan di posisi tengah (MID) dari linked list karena tidak bersifat sangat mendesak maupun sangat jarang digunakan.

Cara Kerja:

1. Pengguna memilih menu “Tambah Produk (otomatis posisi berdasarkan kategori)”.
2. Program meminta input berupa: kategori, nama produk, stok, dan harga satuan.
3. Sistem mengenali bahwa kategori tidak termasuk dalam Head atau Tail.
4. Node baru dibuat dengan data tersebut.
5. Node ditambahkan ke posisi ke-2 (tengah) dalam list.
6. Pointer node sebelumnya diarahkan ke node baru, dan node baru diarahkan ke node sesudahnya.

Input:

Kategori: Peralatan Mandi

Nama Produk: Contoh 'Sabun Lifebuoy'

Stok: Contoh 5

Harga Satuan: Contoh 8000

Output:

Pesan "Produk ditambahkan di posisi tengah."

Screenshot:

```
3. Tambah Produk Alat kebutuhan Mandi Mencuci dan lain-lain

Pilih menu: 3
Kategori: Peralatan mandi
Nama: Sabun lifebuoy
Stok: 5
Harga: 8000
Produk ditambahkan di tengah.

Pilih menu: 4
1. makanan - lays | Stok: 800 | Harga: Rp15000.0
2. makanan - pilus | Stok: 200 | Harga: Rp500.0
3. elektronik - kipas | Stok: 2323 | Harga: Rp90000.0
4. Peralatan mandi - Sabun lifebuoy | Stok: 5 | Harga: Rp8000.0
5. elektronik - lampu | Stok: 100 | Harga: Rp16000.0
6. minuman - ale ale | Stok: 800 | Harga: Rp1000.0
7. minuman - teh botol | Stok: 500 | Harga: Rp6000.0
```

• Fitur: Tampilkan semua Produk

Deskripsi:

Fitur ini digunakan untuk menampilkan seluruh daftar produk toko kelontong yang telah dimasukkan ke dalam struktur data linked list. Penampilan dilakukan dari node pertama (HEAD) hingga node terakhir, dengan mencetak informasi lengkap dari setiap produk.

Cara Kerja:

1. Pengguna memilih menu “Tampilkan Produk”.
2. Program memeriksa apakah linked list kosong.
3. Jika tidak kosong, program melakukan traversal dari node HEAD.
4. Setiap node ditampilkan berurutan, mencetak:
 - Nomor urut
 - Kategori
 - Nama Produk
 - Stok
 - Harga Satuan
 - Total (stok × harga)
5. Proses berhenti saat pointer next = null.

Input: ketik no 4

Output: menampilkan semua produk di txt

Screenshot:

```
Pilih menu: 4
1. makanan - lays | Stok: 800 | Harga: Rp15000.0
2. makanan - pilus | Stok: 200 | Harga: Rp500.0
3. elektronik - kipas | Stok: 2323 | Harga: Rp90000.0
4. Peralatan mandi - Sabun lifebuoy | Stok: 5 | Harga: Rp8000.0
5. elektronik - lampu | Stok: 100 | Harga: Rp16000.0
6. minuman - ale ale | Stok: 800 | Harga: Rp1000.0
7. minuman - teh botol | Stok: 500 | Harga: Rp6000.0
```

• Fitur: Update Produk

Deskripsi:

Fitur ini digunakan untuk mengganti atau memperbaiki informasi produk tertentu yang sudah ada dalam daftar produk (linked list), berdasarkan posisi produk di list

Cara Kerja:

1. Program meminta pengguna memasukkan posisi produk yang ingin diubah.
2. Program membaca data produk baru yang akan menggantikan data lama (kategori, nama, stok, dan harga).
3. Program menelusuri linked list dari awal hingga mencapai node pada posisi yang ditentukan. Jika posisi valid (ada node-nya), maka data produk pada node tersebut akan diperbarui.
4. Jika posisi tidak valid (melewati panjang list), maka ditampilkan pesan kesalahan.

Input: Posisi:

urutan node dalam daftar (6)

Data baru:

- Kategori: Contoh 'Minuman'
- Nama Produk: Contoh 'Teh Pucuk Harum'
- Stok: Contoh 12
- Harga Satuan: Contoh 4000

Output: Pesan "Produk di posisi X berhasil diperbarui." Saat ditampilkan kembali, data produk di posisi tersebut sudah berubah.

Screenshot:

```
Posisi produk yang ingin diupdate: 6
Kategori: Minuman
Nama: teh pucuk harum
Stok: 12
Harga: 4000
Produk di posisi 6 berhasil diperbarui.
```

• Fitur: Hapus Produk

- Deskripsi:
Fitur ini digunakan untuk menghapus produk dari daftar (linked list) berdasarkan posisi produk yang dimasukkan oleh pengguna. Setelah penghapusan, node pada posisi tersebut akan dihapus dari memori dan tidak akan tampil lagi dalam daftar produk..
- Cara Kerja:

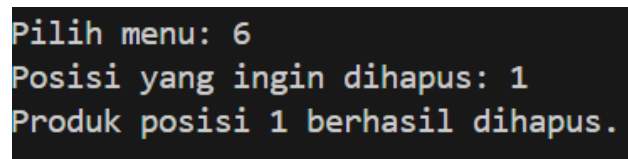
1. Pengguna memilih menu “Hapus Produk”.
2. Program meminta input posisi (misalnya: 1, 2, 3...).
3. Jika posisi = 1, HEAD dipindahkan ke node berikutnya.
4. Jika posisi > 1, program traversal cari node sebelum posisi itu.
5. Node sebelumnya disambungkan ke node sesudah node yang dihapus.
6. Node target diputus dan dihapus dari memori.
7. Program menampilkan validasi produk posisi 1 berhasil di hapus.

- Input: Posisi: angka urutan produk di daftar (contoh: 2)
- Output:

Pesan: Produk berhasil dihapus.

Daftar produk akan tampil tanpa produk yang dihapus.

Screenshot:



```
Pilih menu: 6
Posisi yang ingin dihapus: 1
Produk posisi 1 berhasil dihapus.
```

• Fitur: Cari Produk

- Deskripsi:
Fitur Cari Produk digunakan untuk mencari dan menampilkan produk dalam daftar (linked list) berdasarkan nama atau kategori yang mengandung kata kunci (keyword) yang diberikan oleh pengguna. Pencarian bersifat tidak sensitif huruf besar/kecil. Bertujuan untuk membantu pengguna menemukan produk tertentu dari daftar dengan cepat menggunakan nama atau kategori sebagai kata kunci pencarian.
- Cara Kerja:
 1. pengguna memilih menu “Cari Produk”.
 2. Program meminta pengguna memasukkan keyword pencarian (bebas: nama atau kategori).
 3. Program melakukan traversal dari node HEAD ke node terakhir.

4. Setiap node dicek: Apakah nama produk mengandung keyword Atau kategori produk sama persis dengan keyword, Jika ditemukan, data produk ditampilkan.
 5. Jika tidak ada yang cocok, tampilkan pesan “Produk tidak ditemukan.”
- Input: Keyword pencarian (contoh: sabun, makanan, beras)
 - Output:

Screenshot:

```
Pilih menu: 7
Masukkan kategori/nama produk: sabun
Posisi: 3 | [Peralatan mandi] Sabun lifebuoy | Stok: 5 | Harga Satuan: Rp8000.0 | Total:
Rp40000.0
```

• Fitur: Simpan Semua Produk Ke File

- Deskripsi:

Fitur Simpan ke File berfungsi untuk menyimpan seluruh data produk dari linked list ke dalam file teks. Setiap node dibaca dari awal (head) hingga akhir, lalu ditulis ke file dalam format tertentu menggunakan BufferedWriter. Tujuannya adalah agar data produk dapat disimpan secara permanen dan digunakan kembali saat program dijalankan kembali. Fitur ini juga menangani kesalahan penulisan file dan memberi notifikasi jika proses gagal atau berhasil.
- Cara Kerja:
 1. Pengguna memilih menu “Simpan ke File”.
 2. Program membuka atau membuat file baru bernama produk.txt.
 3. Program melakukan traversal seluruh linked list dari HEAD ke node terakhir.
 4. Setiap node diubah ke format teks (misal dipisahkan dengan ;) lalu ditulis ke file.
 5. Setelah semua data ditulis, file ditutup dan muncul pesan Data berhasil disimpan ke produk.txt.
- Input: Tidak ada (Proses otomatis saat menu dipilih)
- Output:

File produk.txt berisi data produk dalam format.

Screenshot:

```
Pilih menu: 8
Data berhasil disimpan ke produk.txt
```

• Fitur [Keluar]

- Deskripsi:

Fitur ini menangani keluar dari aplikasi dan validasi input menu. Saat pengguna memilih menu nomor 9, program akan menampilkan pesan “Keluar program.” dan menghentikan perulangan do-while, sehingga aplikasi ditutup secara normal. Jika pengguna memasukkan angka di luar rentang menu (1–9), akan ditampilkan pesan “Pilihan tidak valid.” untuk memberi tahu bahwa input tidak sesuai. Setelah perulangan berhenti, `input.close()` dipanggil untuk menutup scanner dan mengakhiri program dengan rapi.
- Cara Kerja:
 1. Pengguna memilih menu “Keluar”.
 2. Program menampilkan pesan konfirmasi keluar.
 3. Program menutup semua input/output (Scanner, file, dll).
 4. Aplikasi berhenti berjalan dan kembali ke command line atau desktop.
- Input: Pilihan menu angka untuk keluar (angka 9)
- Output:

Pesan: keluar program.

Aplikasi berhenti

Screenshot:

```
Pilih menu: 9
Keluar program.
```

4. Kesimpulan

Proyek tugas besar ini telah berhasil diselesaikan dengan mengembangkan aplikasi berbasis konsol untuk mengelola data produk pada toko kelontong menggunakan bahasa pemrograman Java. Dalam pengembangannya, aplikasi ini menerapkan struktur data Linked List sebagai inti dari pengelolaan data dinamis. Seluruh fitur dasar yang berkaitan

dengan penambahan, penghapusan, pencarian, serta penyimpanan data telah berhasil diimplementasikan dan diuji.

Pencapaian:

- Penambahan produk di awal (tambahHead), akhir (tambahTail), dan tengah (tambahMid).
- Menampilkan daftar produk dengan informasi lengkap.
- Memperbarui dan menghapus data produk berdasarkan posisi.
- Mencari produk berdasarkan nama atau kategori.
- Menyimpan dan memuat data dari file eksternal menggunakan BufferedWriter dan BufferedReader.
- Struktur data Linked List telah diterapkan secara penuh dalam pengelolaan data dinamis, mulai dari penambahan, penghapusan, pencarian, hingga penyimpanan ke file.
- Pembelajaran yang diperoleh oleh tim meliputi:
Penerapan konsep Object-Oriented Programming (OOP).
Pengelolaan struktur data dinamis (Linked List).
Penerapan algoritma dasar dalam operasi CRUD.
Implementasi proses input/output file di Java.

Tantangan yang Dihadapi:

- Kesulitan teknis yang ditemui :
 - Validasi input yang terbatas dan dapat menyebabkan program error saat pengguna memasukkan data yang tidak sesuai.
 - GitHub Collab yang terkadang error
- Cara mengatasi masalah tersebut
 - Melakukan pengujian secara berulang (testing manual) pada setiap fitur setelah implementasi untuk mendeteksi dan memperbaiki bug.
 - Komunikasi tim secara aktif, misalnya lewat grup chat, agar tidak mengedit file yang sama di waktu bersamaan.

Evaluasi:

- Kelebihan :

- Aplikasi ringan, tidak membutuhkan database, cocok untuk toko kecil/menengah.
- Struktur data Linked List memudahkan penambahan/penghapusan data tanpa perlu menggeser elemen.
- Format penyimpanan ke file sederhana namun cukup efektif.
- Kekurangan :
 - Belum mendukung fitur multiuser dan keamanan/autentikasi.
 - Validasi input dan penanganan kesalahan masih bisa ditingkatkan untuk mencegah kesalahan input.
- Potensi Pengembangan Lebih Lanjut :
- Implementasi Login dan Multiuser
 - Menambahkan sistem autentikasi pengguna (username dan password).
 - Memberikan hak akses berbeda antara admin dan karyawan, misalnya:
 - Admin bisa menambah, menghapus, dan menyimpan produk.
 - Karyawan hanya bisa melihat dan mencari produk.
- Penyimpanan Menggunakan Database
 - Menggantikan penyimpanan file .txt menjadi database seperti MySQL, SQLite, atau MongoDB.
 - Memberikan fitur penyimpanan permanen yang lebih terstruktur dan aman.

Rekomendasi :

- Menambahkan validasi input lanjutan, seperti cek stok tidak negatif, harga harus lebih dari nol, dan nama tidak kosong.
- Menambahkan fitur sorting produk berdasarkan nama, kategori, atau harga.

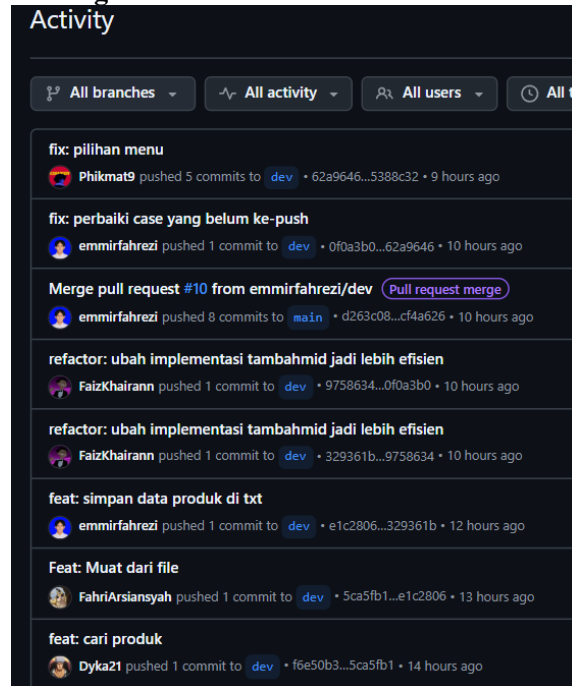
Lampiran

A. Repository GitHub

https://github.com/emmirmifahrezi/Cibibesor_A_Tubes_PP1

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/emmirmifahrezi/Cibibesor_A_Tubes_PP1`
4. Buka folder proyek di IDE Java.
5. Jalankan file MainApp.java yang berada di dalam package App untuk menjalankan aplikasi.

B. Log Aktivitas GitHub



C. Screenshot Aplikasi Tampilan utama:

```
=== MENU TOKO KELONTONG ===
1. Tambah Produk Sembako
2. Tambah Produk Makanan dan Minuman Ringan
3. Tambah Produk Alat kebutuhan Mandi Mencuci dan lain-lain
4. Tampilkan Produk
5. Update Produk
6. Hapus Produk
7. Cari Produk
8. Simpan ke File
9. Keluar
Pilih menu: 1
```

1. Tambahkan produk sembako:

```
Kategori: Makanan
Nama: Beras senia
Stok: 10
Harga: 50000
Produk ditambahkan di awal (HEAD).
```

2. Tambahkan produk makanan dan minuman ringan

```
Pilih menu: 2
Kategori: makanan
Nama: Taro
Stok: 10
Harga: 13000
Produk ditambahkan di akhir (TAIL).
```

3. Tambahkan produk mandi mencuci dan lainlain

```
Pilih menu: 3
Kategori: Peralatan mandi
Nama: Sabun lifebuoy
Stok: 5
Harga: 8000
Produk ditambahkan di tengah.
```

4. Tampilkan produk

```
Pilih menu: 4
1. makanan - lays | Stok: 800 | Harga: Rp15000.0
2. makanan - pilus | Stok: 200 | Harga: Rp500.0
3. elektronik - kipas | Stok: 2323 | Harga: Rp90000.0
4. Peralatan mandi - Sabun lifebuoy | Stok: 5 | Harga: Rp8000.0
5. elektronik - lampu | Stok: 100 | Harga: Rp16000.0
6. minuman - ale ale | Stok: 800 | Harga: Rp1000.0
7. minuman - teh botol | Stok: 500 | Harga: Rp6000.0
```

5. Update produk

```
Posisi produk yang ingin diupdate: 6
Kategori: Minuman
Nama: teh pucuk harum
Stok: 12
Harga: 4000
Produk di posisi 6 berhasil diperbarui.
```

6. Hapus produk

```
Pilih menu: 6
Posisi yang ingin dihapus: 1
Produk posisi 1 berhasil dihapus.
```

7. Cari produk

```
Pilih menu: 7
Masukkan kategori/nama produk: sabun
Posisi: 3 | [Peralatan mandi] Sabun lifebuoy | Stok: 5 | Harga Satuan: Rp8000.0 | Total:
Rp40000.0
```

8. Simpan ke file

```
Pilih menu: 8
Data berhasil disimpan ke produk.txt
```

9. Keluar

```
Pilih menu: 9
Keluar program.
```