

# Tugas Personal ke-2 Week 7

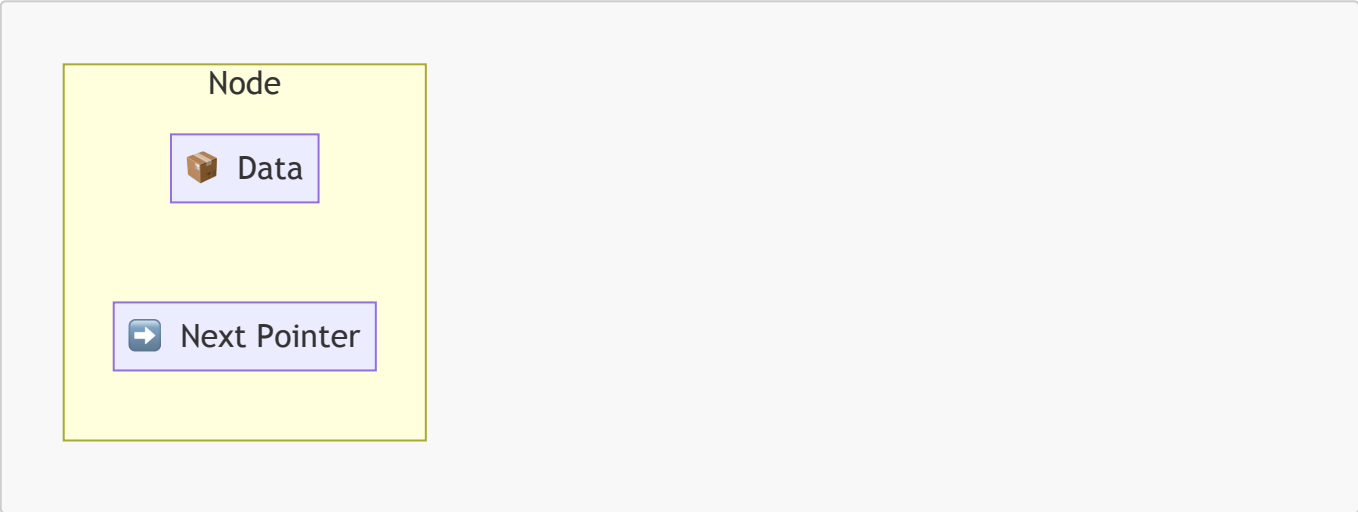
## Jawaban Soal Review: Single Linked List

### 1. Pengertian Single Linked List

**Single Linked List** adalah struktur data linear yang terdiri dari kumpulan node yang saling terhubung secara sekuensial. Setiap node memiliki dua komponen:

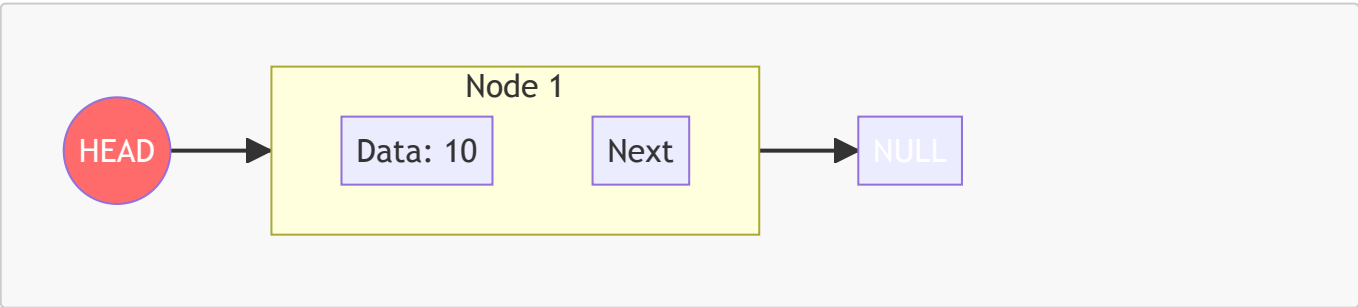
- **Data**: menyimpan nilai/informasi
- **Pointer (next)**: menyimpan alamat node berikutnya

Struktur Node

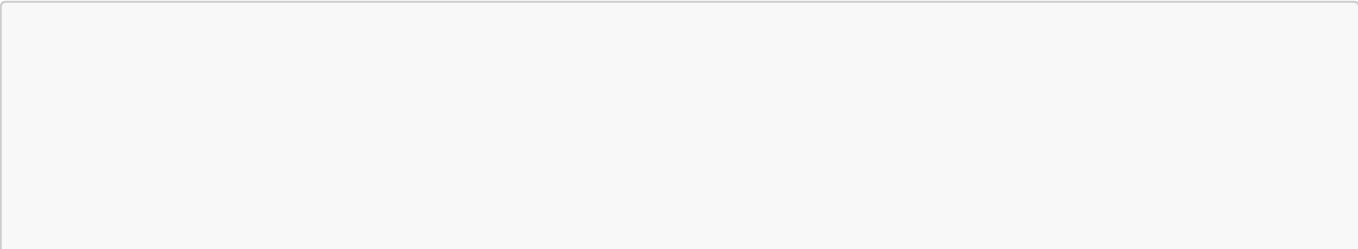


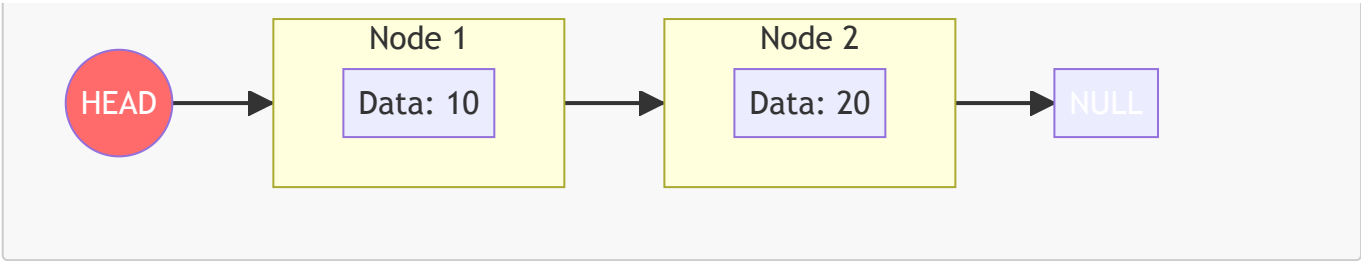
Ilustrasi Cara Membuat Single Linked List

#### Langkah 1: Buat Node Pertama (HEAD)

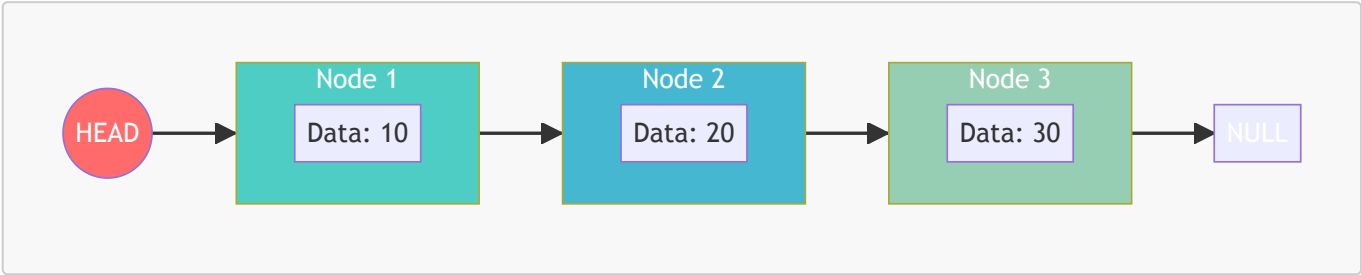


#### Langkah 2: Tambah Node Kedua

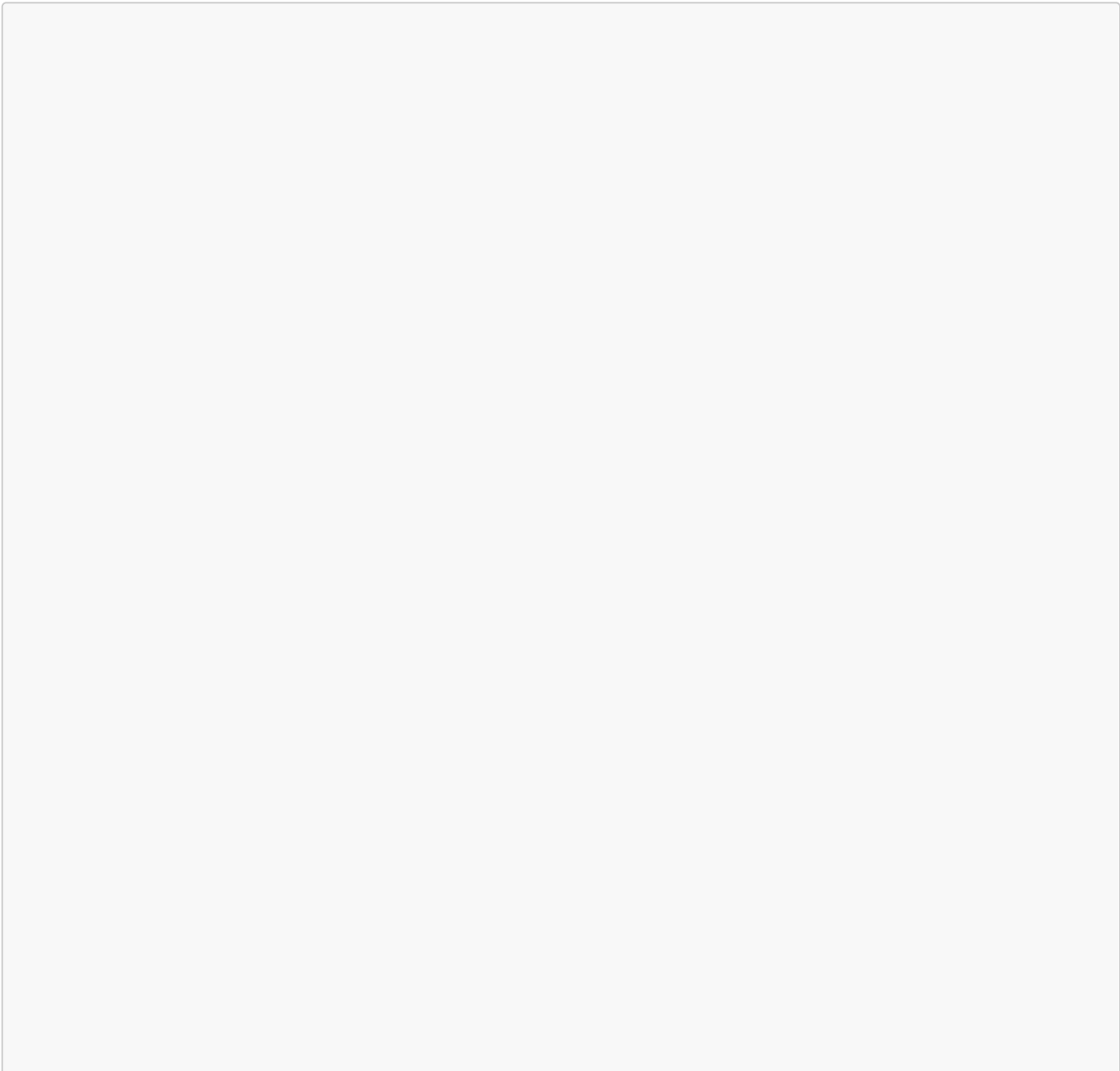


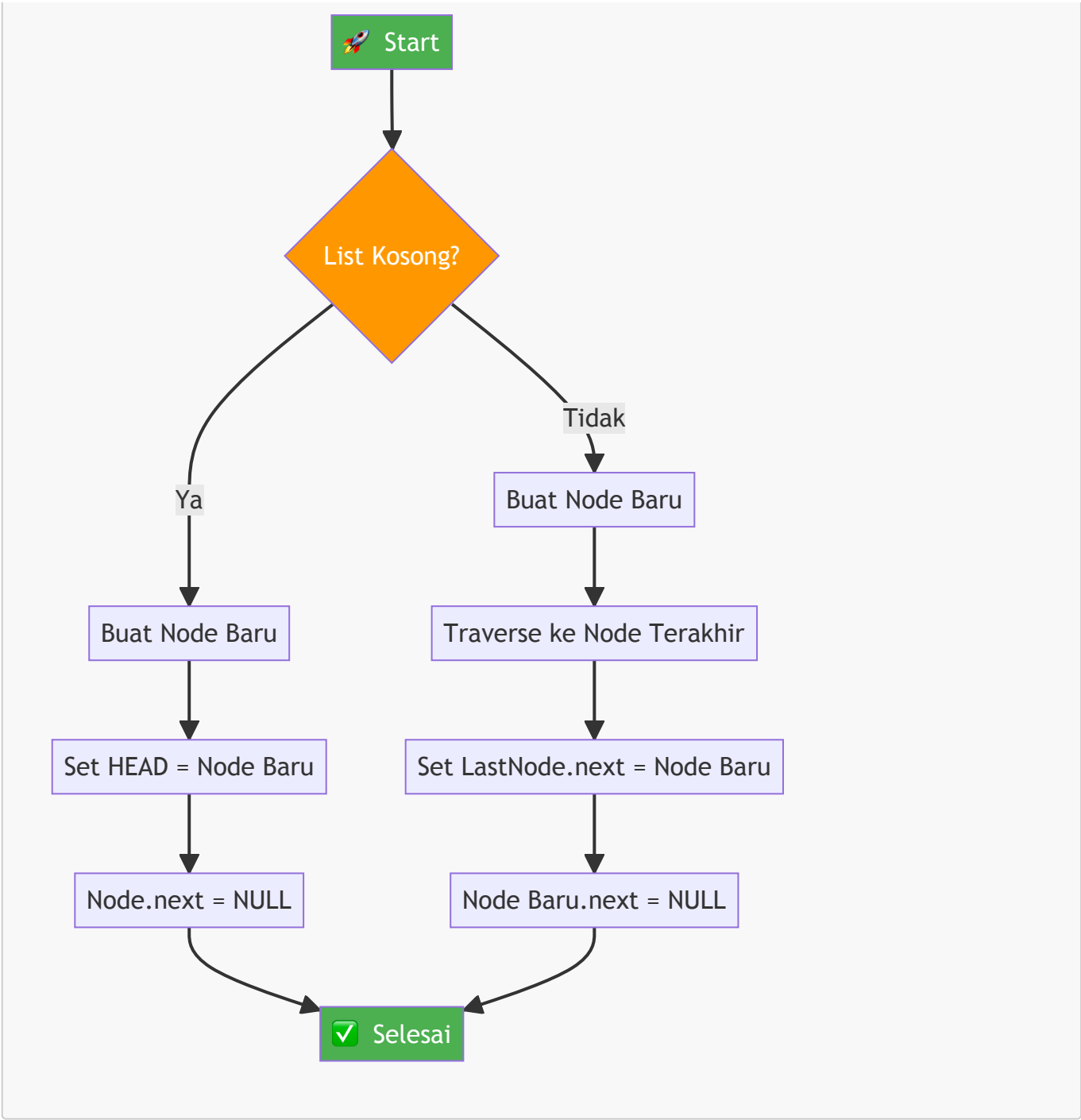


Langkah 3: Tambah Node Ketiga (Final)



Proses Pembuatan (Flowchart)





## 2. Representasi Linked List dalam Memori

Linked list direpresentasikan dalam memori menggunakan alamat memori yang tersebar (tidak harus berurutan). Setiap node menyimpan DATA dan NEXT (alamat node berikutnya).

Contoh: Menyimpan List A → B → C → D

Tabel Representasi Memori

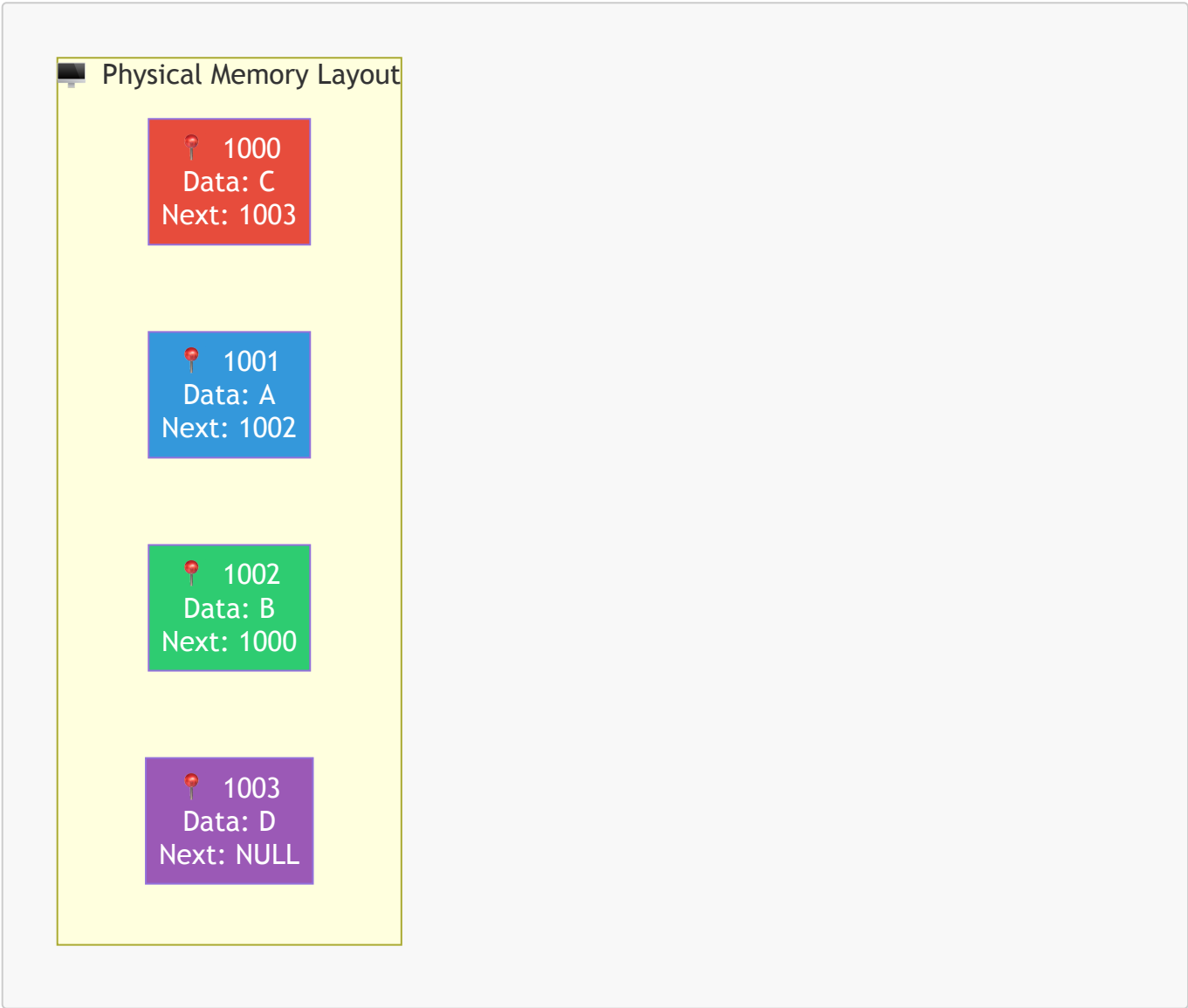
Alamat Memori	DATA	NEXT (Pointer)
1000	C	1003
1001	A	1002

Alamat Memori	DATA	NEXT (Pointer)
1002	B	1000
1003	D	NULL

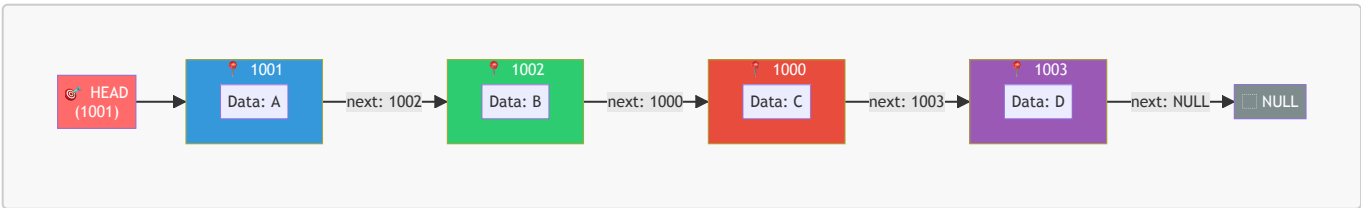
HEAD = 1001 (menunjuk ke node pertama)

Visualisasi Memori Fisik vs Logis

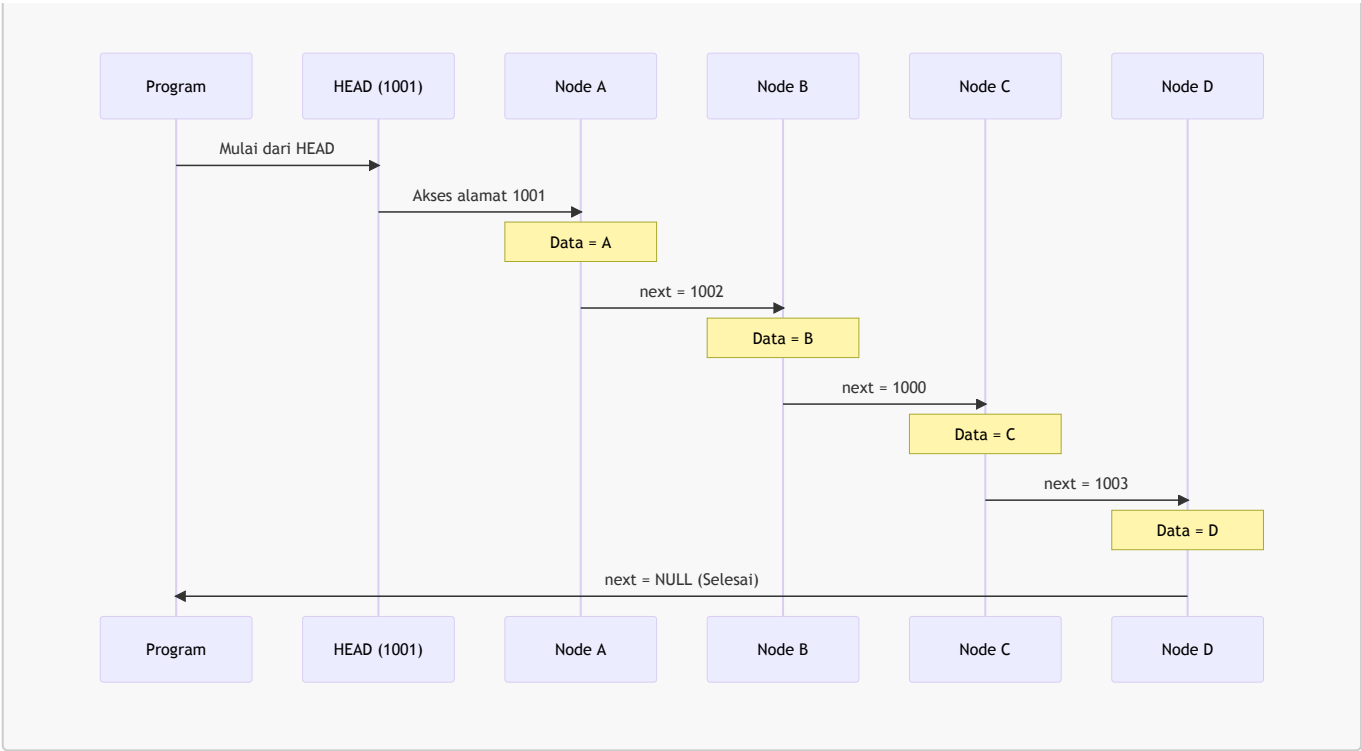
Memori Fisik (Data Tersebar)



Urutan Logis (Traversal dari HEAD)



Proses Traversal



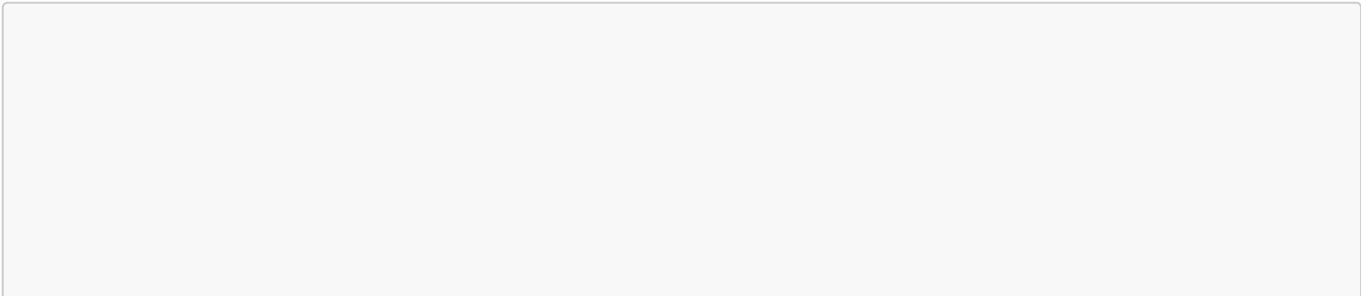
### 3. Perbandingan Linked List vs Array

Tabel Perbandingan

Aspek	Array	Linked List
Ukuran	Tetap (fixed)	Dinamis
Alokasi Memori	Kontinu (berurutan)	Non-kontinu (tersebar)
Akses Elemen	Random access $O(1)$	Sequential access $O(n)$
Insert di Awal	$O(n)$	$O(1)$
Insert di Tengah	$O(n)$	$O(n)$ cari + $O(1)$ insert
Delete	$O(n)$	$O(1)$ setelah ditemukan
Memory Overhead	Tidak ada	Ada (untuk pointer)
Cache Performance	Baik	Kurang baik

#### Visualisasi Perbandingan Memori

##### Array (Kontinu)



### Array - Memori Kontinu

[0]  
10



[1]  
20



[2]  
30



[3]  
40



[4]  
50

### Linked List (Tersebar)

### Linked List - Memori Tersebar

HEAD

10

20

30

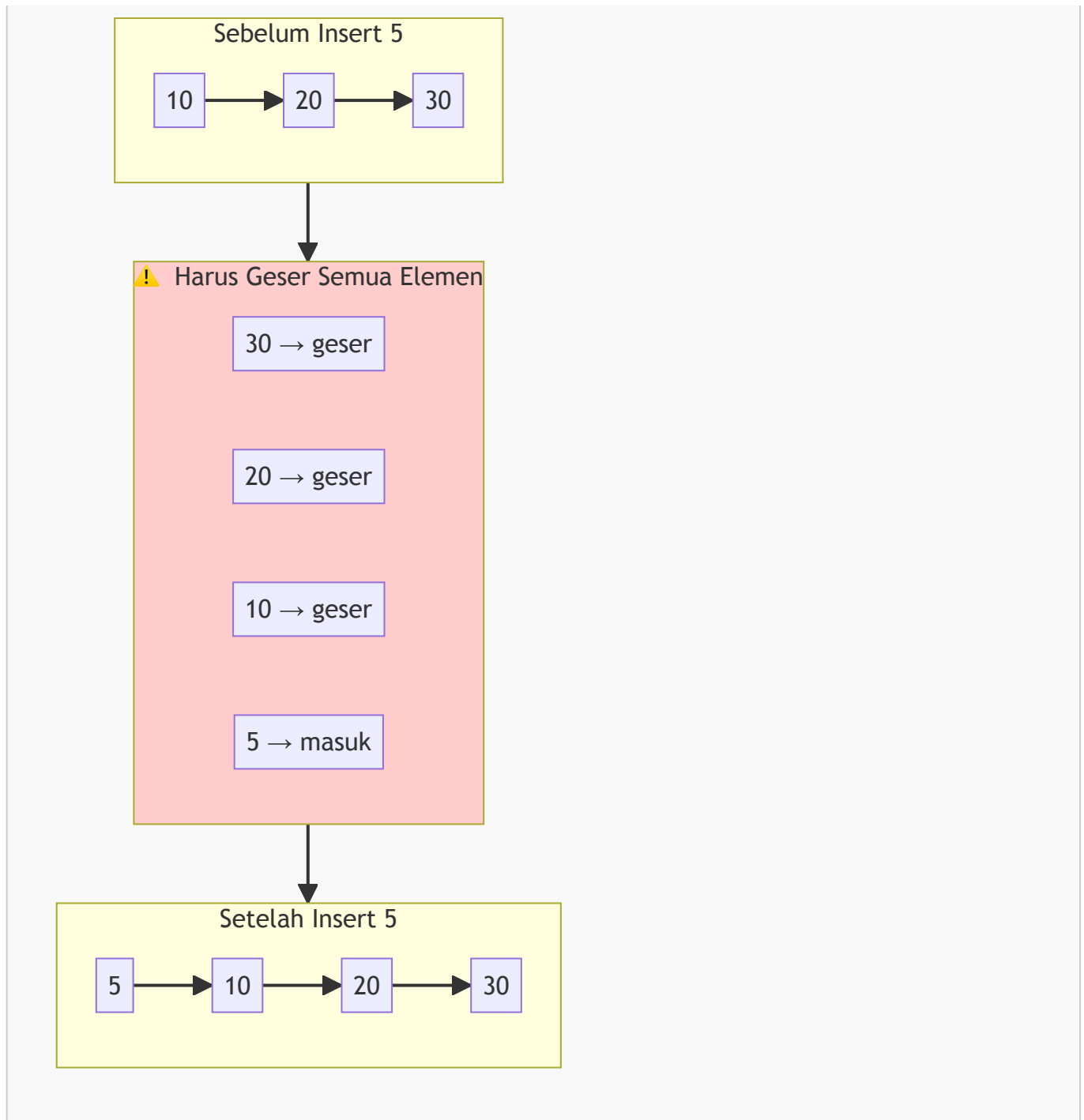
40

50

NULL

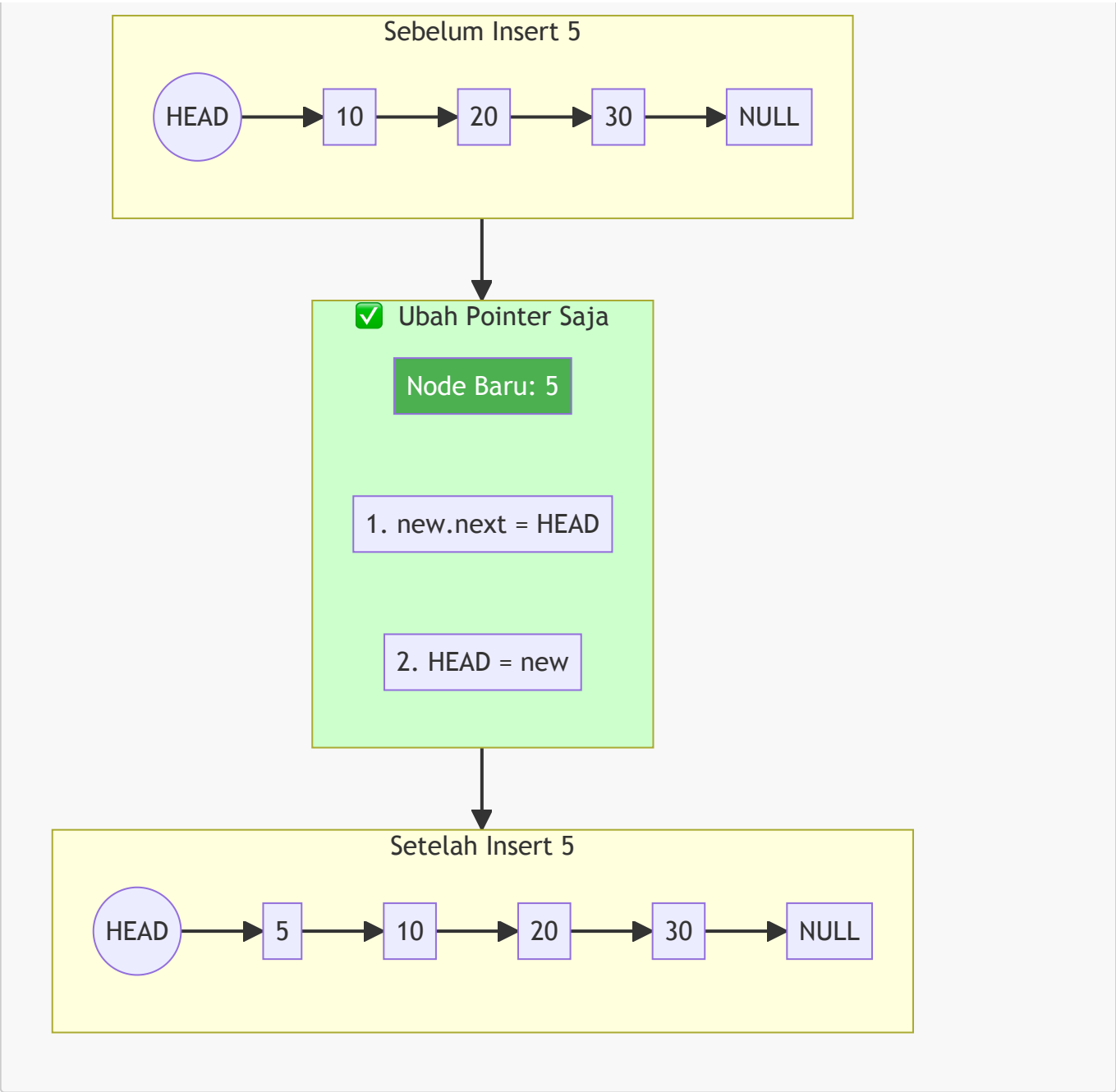
Perbandingan Operasi Insert di Awal

**Array: Insert di Awal -  $O(n)$**

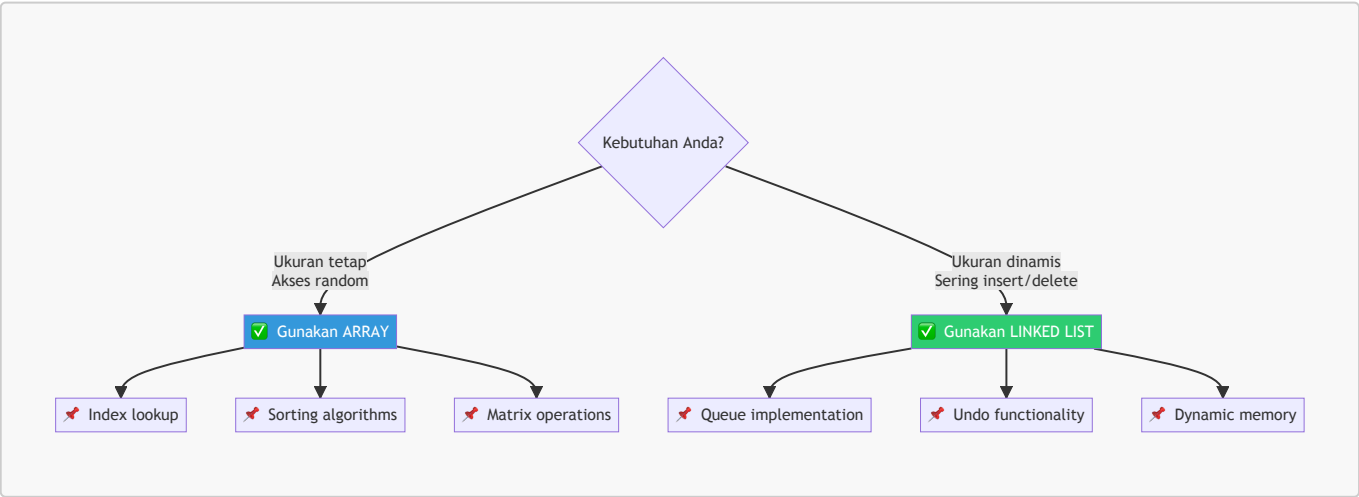


### Linked List: Insert di Awal - $O(1)$





Kapan Menggunakan?



Kesimpulan

Tidak ada yang "lebih baik" secara absolut - pemilihan tergantung pada kebutuhan:

- **Array:** Ketika butuh akses cepat ke elemen tertentu dan ukuran sudah diketahui
- **Linked List:** Ketika sering melakukan operasi insert/delete dan ukuran data berubah-ubah

## Jawaban Case

---

DatabaseMahasiswa.java

```
import java.util.Scanner;

/**
 * Program Database Mahasiswa Sederhana
 * Menggunakan Single Linked List
 *
 * Fitur:
 * 1. Push data mahasiswa (dengan validasi)
 * 2. Tampilkan semua data (sorted by NIM)
 * 3. Pop semua data
 * 4. Keluar program
 */

// Class Node untuk menyimpan data mahasiswa
class NodeMahasiswa {
    String nim;
    String nama;
    String jurusan;
    NodeMahasiswa next;

    // Constructor
    public NodeMahasiswa(String nim, String nama, String jurusan) {
        this.nim = nim;
        this.nama = nama;
        this.jurusan = jurusan;
        this.next = null;
    }
}

// Class Single Linked List untuk database mahasiswa
class LinkedListMahasiswa {
    private NodeMahasiswa head;
    private int size;
    private static final int MAX_SIZE = 5;

    // Constructor
    public LinkedListMahasiswa() {
        this.head = null;
        this.size = 0;
    }

    // Cek apakah list kosong
```

```
public boolean isEmpty() {
    return head == null;
}

// Cek apakah list penuh
public boolean isFull() {
    return size >= MAX_SIZE;
}

// Mendapatkan jumlah data
public int getSize() {
    return size;
}

// Push data mahasiswa (insert di akhir)
public boolean push(String nim, String nama, String jurusan) {
    // Validasi kapasitas
    if (isFull()) {
        System.out.println("\n[ERROR] Database penuh! Maksimal " +
MAX_SIZE + " data.");
        return false;
    }

    // Cek NIM duplikat
    if (isDuplicateNim(nim)) {
        System.out.println("\n[ERROR] NIM " + nim + " sudah
terdaftar!");
        return false;
    }

    // Buat node baru
    NodeMahasiswa newNode = new NodeMahasiswa(nim, nama, jurusan);

    // Jika list kosong, node baru menjadi head
    if (isEmpty()) {
        head = newNode;
    } else {
        // Traverse ke node terakhir
        NodeMahasiswa current = head;
        while (current.next != null) {
            current = current.next;
        }
        // Sambungkan node baru di akhir
        current.next = newNode;
    }

    size++;
    return true;
}

// Cek apakah NIM sudah ada
private boolean isDuplicateNim(String nim) {
    NodeMahasiswa current = head;
    while (current != null) {
```

```

        if (current.nim.equals(nim)) {
            return true;
        }
        current = current.next;
    }
    return false;
}

// Menampilkan semua data (sorted by NIM menggunakan Bubble Sort)
public void displaySorted() {
    if (isEmpty()) {

System.out.println("\n=====
=====");
        System.out.println("||
KOSONG - Tidak ada data mahasiswa ||");
        DATABASE

System.out.println("||
=====
=====");

        return;
    }

    // Sorting linked list berdasarkan NIM (Bubble Sort)
    sortByNim();

    // Header tabel

System.out.println("\n=====
=====");
        System.out.println("||
MAHASISWA ||");
        DAFTAR
        System.out.println("||
NIM) ||");
        (Sorted by

System.out.println("||
=====
=====");
        System.out.println("|| No. || NIM || NAMA
|| JURUSAN ||");

System.out.println("||
=====
=====");

    // Traverse dan tampilkan data
    NodeMahasiswa current = head;
    int no = 1;
    while (current != null) {
        System.out.printf("|| %-4d || %-12s || %-30s || %-37s ||%n",
            no++,
            current.nim,
            current.nama,
            current.jurusan);
        current = current.next;
    }
}

```

```

System.out.println("=====");
System.out.printf("    Total Data: %-81d \n", size);

System.out.println("=====");
}

// Sorting linked list berdasarkan NIM (Bubble Sort)
private void sortByNim() {
    if (head == null || head.next == null) {
        return; // Tidak perlu sorting jika 0 atau 1 elemen
    }

    boolean swapped;
    do {
        swapped = false;
        NodeMahasiswa current = head;

        while (current.next != null) {
            // Bandingkan NIM (string comparison)
            if (current.nim.compareTo(current.next.nim) > 0) {
                // Swap data (bukan node)
                String tempNim = current.nim;
                String tempNama = current.nama;
                String tempJurusan = current.jurusan;

                current.nim = current.next.nim;
                current.nama = current.next.nama;
                current.jurusan = current.next.jurusan;

                current.next.nim = tempNim;
                current.next.nama = tempNama;
                current.next.jurusan = tempJurusan;

                swapped = true;
            }
            current = current.next;
        }
    } while (swapped);
}

// Pop semua data mahasiswa
public void popAll() {
    if (isEmpty()) {
        System.out.println("\n[INFO] Database sudah kosong, tidak ada data untuk di-pop.");
        return;
    }
}

System.out.println("\n=====");

```

```

        System.out.println("||
SEMUA DATA                                ||");

        System.out.println("=====
=====||");

        int count = 0;
        while (head != null) {
            count++;
            System.out.printf("  Pop #d: NIM=%s, Nama=%s, Jurusan=%s%n",
                count, head.nim, head.nama, head.jurusan);
            head = head.next; // Hapus node dari depan
            size--;
        }

        System.out.println("\n[SUCCESS] " + count + " data mahasiswa
berhasil di-pop.");
        System.out.println("[INFO] Database sekarang kosong.");
    }
}

// Main class
public class DatabaseMahasiswa {
    private static Scanner scanner = new Scanner(System.in);
    private static LinkedListMahasiswa database = new
LinkedListMahasiswa();

    public static void main(String[] args) {

        System.out.println("\n=====
=====||");
        System.out.println("||          PROGRAM DATABASE MAHASISWA
SEDERHANA - SINGLE LINKED LIST          ||");

        System.out.println("=====
=====||");

        boolean running = true;

        while (running) {
            displayMenu();
            int pilihan = getMenuChoice();

            switch (pilihan) {
                case 1:
                    pushMahasiswa();
                    break;
                case 2:
                    database.displaySorted();
                    break;
                case 3:
                    popSemuaData();
                    break;
                case 4:

```

```

        running = false;

System.out.println("\n=====
=====
");
        System.out.println("Exitting!
");

System.out.println("=====
=====
");
        break;
        default:
            System.out.println("\n[ERROR] Pilihan tidak valid!
Silakan pilih 1-4.");
    }
}

scanner.close();
}

// Menampilkan menu
private static void displayMenu() {
    System.out.println("\n=====
=====
MENU UTAMA
=====
");
    System.out.println("1. Push Data Mahasiswa
");
    System.out.println("2. Tampilkan Semua Data (Sorted)
");
    System.out.println("3. Pop Semua Data Mahasiswa
");
    System.out.println("4. Keluar
");
    System.out.println("=====
");
    System.out.printf("Data tersimpan: %d/5\n",
database.getSize());
    System.out.println("=====
");
}

// Mendapatkan pilihan menu
private static int getMenuChoice() {
    System.out.print("Pilihan Anda [1-4]: ");
    try {
        return Integer.parseInt(scanner.nextLine().trim());
    } catch (NumberFormatException e) {
        return -1;
    }
}

// Push data mahasiswa dengan validasi
private static void pushMahasiswa() {

System.out.println("\n=====
=====
");
        System.out.println("MAHASISWA
");
        INPUT DATA
        System.out.println("=====
=====
");

```

```
// Cek apakah database penuh
if (database.isFull()) {
    System.out.println("\n[ERROR] Database penuh! Maksimal 5 data
mahasiswa.");
    return;
}

// Input dan validasi NIM
String nim;
while (true) {
    System.out.print("Masukkan NIM (maks 10 angka): ");
    nim = scanner.nextLine().trim();

    if (nim.isEmpty()) {
        System.out.println("[ERROR] NIM tidak boleh kosong!");
        continue;
    }
    if (nim.length() > 10) {
        System.out.println("[ERROR] NIM maksimal 10 karakter!
(Input: " + nim.length() + " karakter)");
        continue;
    }
    if (!nim.matches("\\d+")) {
        System.out.println("[ERROR] NIM hanya boleh berisi
angka!");
        continue;
    }
    break;
}

// Input dan validasi Nama
String nama;
while (true) {
    System.out.print("Masukkan Nama (maks 30 karakter): ");
    nama = scanner.nextLine().trim();

    if (nama.isEmpty()) {
        System.out.println("[ERROR] Nama tidak boleh kosong!");
        continue;
    }
    if (nama.length() > 30) {
        System.out.println("[ERROR] Nama maksimal 30 karakter!
(Input: " + nama.length() + " karakter)");
        continue;
    }
    break;
}

// Input dan validasi Jurusan
String jurusan;
while (true) {
    System.out.print("Masukkan Jurusan (maks 50 karakter): ");
    jurusan = scanner.nextLine().trim();
```



```
        if (jurusan.isEmpty()) {
            System.out.println("[ERROR] Jurusan tidak boleh kosong!");
            continue;
        }
        if (jurusan.length() > 50) {
            System.out.println("[ERROR] Jurusan maksimal 50 karakter!
(Input: " + jurusan.length() + " karakter)");
            continue;
        }
        break;
    }

    // Push ke database
    if (database.push(nim, nama, jurusan)) {
        System.out.println("\n[SUCCESS] Data mahasiswa berhasil
ditambahkan!");
        System.out.println("  NIM      : " + nim);
        System.out.println("  Nama     : " + nama);
        System.out.println("  Jurusan : " + jurusan);
    }
}

// Pop semua data dengan konfirmasi
private static void popSemuaData() {
    if (database.isEmpty()) {
        System.out.println("\n[INFO] Database sudah kosong, tidak ada
data untuk di-pop.");
        return;
    }

    System.out.print("\nApakah Anda yakin ingin menghapus SEMUA data?
(y/n): ");
    String konfirmasi = scanner.nextLine().trim().toLowerCase();

    if (konfirmasi.equals("y") || konfirmasi.equals("yes")) {
        database.popAll();
    } else {
        System.out.println("[INFO] Operasi pop dibatalkan.");
    }
}
}
```