LAPORAN PRATIKUM

"PEKAN 5"

Disusun Untuk Memenuhi Tugas Mata Kuliah Struktur Data

DOSEN PENGAMPU:

Wahyudi, Dr. S.T. M.T.

DISUSUN OLEH:

Karimah Irsyadiyah (2411533018)

UNIVERSITAS ANDALAS

T.A 2024/2025

Daftar Pustaka

BAB I
PENDAHULUAN

1.1 Latar Belakang

Struktur data merupakan dasar dari pemrograman komputer dan sangat penting dalam pengembangan perangkat lunak. Pada pekan kelima ini, praktikum difokuskan pada implementasi struktur data **Single Linked List (SLL)**, termasuk operasi pencarian, penambahan, dan penghapusan elemen dalam linked list. Pemahaman tentang linked list sangat penting karena strukturnya yang dinamis dan efisien dalam pengelolaan memori serta kemampuannya untuk menangani data secara fleksibel.

1.2 Tujuan

- Memahami konsep dasar dan struktur dari Single Linked List.
- Mampu mengimplementasikan operasi dasar pada SLL seperti tambah, hapus, dan cari elemen.
- Mampu memproses data menggunakan struktur antrian (queue) yang terintegrasi dengan SLL.

1.3 Alat dan Bahan

- Perangkat Keras: Laptop/PC
- Perangkat komputer dengan IDE (contoh: IntelliJ IDEA / Eclipse).
- Java Development Kit (JDK).

BAB II
PEMBAHASAN

2.1 Langkah-langkah Praktikum dan Pembahasan Program

A. Class : NodeSLL.java

- Membuat atribut `data` dan `next` dalam class `NodeSLL`.
- Menambahkan konstruktor untuk menginisialisasi node.

```
package pekan5;

public class NodeSLL {
    // node bagian data
    int data;
    // Pointer ke node berikutnya
    NodeSLL next;
    // Konstruktor menginisialisasi node dengan data
    public NodeSLL(int data) {
        this.data = data;
        this.next = null;
    }
}
```

B. Class : TambahSLL.java

Program ini melakukan penambahan elemen pada awal, akhir, dan posisi tertentu dalam linked list.

1. Membuat linked list awal: 2 → 3 → 5 → 6.
2. Menambahkan node dengan nilai 1 di depan.
3. Menambahkan node dengan nilai 7 di akhir.
4. Menyisipkan node dengan nilai 4 di posisi ke-4.
5. Menampilkan hasil linked list setelah tiap operasi.

```
package pekan5;

public class TambahSLL {

    // Tambah node di depan SLL
    public static NodeSLL insertAtFront(NodeSLL head, int value) {
        NodeSLL new_node = new NodeSLL(value);
        new_node.next = head;
        return new_node;
    }

    // Fungsi menambahkan node di akhir SLL
    public static NodeSLL insertAtEnd(NodeSLL head, int value) {
        // buat sebuah node dengan sebuah nilai
        NodeSLL newNode = new NodeSLL(value);
        // jika list kosong maka node jadi head
```

```
        if (head == null) {
            return newNode;
        }

        // simpan head ke variabel sementara
        NodeSLL last = head;
        // telusuri ke node akhir
        while (last.next != null) {
            last = last.next;
        }

        // ubah pointer
        last.next = newNode;
        return head;
    }

    // Membuat node baru dengan data
    static NodeSLL GetNode(int data) {
        return new NodeSLL(data);
    }

    // Menyisipkan node pada posisi tertentu
    static NodeSLL insertPos(NodeSLL headNode, int position, int value)
{
        NodeSLL head = headNode;

        if (position < 1) {
            System.out.print("Invalid position");
        }

        if (position == 1) {
            NodeSLL new_node = new NodeSLL(value);
            new_node.next = head;
            return new_node;
        } else {
            while (position-- != 0) {
                if (position == 1) {
                    NodeSLL newNode = GetNode(value);
                    newNode.next = headNode.next;
                    headNode.next = newNode;
                    break;
                }
                headNode = headNode.next;
            }

            if (position != 1) {
                System.out.print("Posisi di luar jangkauan");
            }

            return head;
        }
    }

    // Menampilkan seluruh isi linked list
    public static void printList(NodeSLL head) {
        NodeSLL curr = head;
        while (curr.next != null) {
            System.out.print(curr.data + "-->");
            curr = curr.next;
```

```
        }
        if (curr.next==null) {
          System.out.println(curr.data);
        }
        System.out.println();
    }
    public static void main(String[] args) {
        // buat linked list 2 -> 3 -> 5 -> 6
        NodeSLL head = new NodeSLL(2);
        head.next = new NodeSLL(3);
        head.next.next = new NodeSLL(5);
        head.next.next.next = new NodeSLL(6);

        // cetak list asli
        System.out.print("Senarai berantai awal:");
        printList(head);

        // tambahkan node baru di depan
        System.out.print("Tambah 1 simpul di depan: ");
        int data = 1;
        head = insertAtFront(head, data);
        // cetak update list
        printList(head);

        // tambahkan node baru di belakang
        System.out.print("Tambah 1 simpul di belakang: ");
        int data2 = 7;
        head = insertAtEnd(head, data2);
        // cetak update list
        printList(head);

        // tambahkan node di posisi tertentu
        System.out.print("Tambah 1 simpul ke data 4: ");
        int data3 = 4;
        int pos = 4;
        head = insertPos(head, pos, data3);
        // cetak update list
        printList(head);
    }

}
6.
```

Tujuannya untuk memahami manipulasi data dalam linked list secara dinamis.

C. Class : PencarianSLL.java

1. list sebagai queue dan penggunaannya dalam simulasi antrian layanan.
2. Membuat linked list dengan elemen: 14 → 21 → 13 → 30 → 10.
3. Menampilkan semua elemen dengan traversal.
4. Mencari nilai tertentu (misalnya 30) dengan metode searchKey().

```java
package pekan5;

public class PencarianSLL {

    static boolean searchKey(NodeSLL head, int key) {
        NodeSLL curr = head;
        while (curr != null) {
            if (curr.data == key)
                return true;
            curr = curr.next;
        }
        return false;
    }

    public static void traversal(NodeSLL head) {
        // mulai dari head
        NodeSLL curr = head;
        // telusuri sampai pointer null
        while (curr != null) {
            System.out.print(" " + curr.data);
            curr = curr.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        NodeSLL head = new NodeSLL(14);
        head.next = new NodeSLL(21);
        head.next.next = new NodeSLL(13);
        head.next.next.next = new NodeSLL(30);
        head.next.next.next.next = new NodeSLL(10);

        System.out.print("Penelusuran SLL : ");
        traversal(head);

        // data yang akan dicari
        int key = 30;
        System.out.print("cari data " + key + " = ");
        if (searchKey(head, key))
            System.out.println("ketemu");
        else
            System.out.println("tidak ada");
    }
}
```

D. Class HapusSLL.java

Program ini menggunakan struktur antrian (Queue) berbasis linked list untuk memproses pelanggan dan menghitung waktu selesai pelayanan.

1. Membuat kelas Pelanggan untuk menyimpan ID dan jumlah pesanan.
2. Membaca data input pelanggan ke dalam queue.
3. Mengambil pelanggan satu per satu dan menghitung total waktu selesai.
4. Menampilkan ID pelanggan dan waktu selesai secara berurutan.

```java
package pekan5;
```

```java
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

// Class untuk menyimpan data pelanggan
class Pelanggan {
    String id;
    int jumlahPesanan;

    public Pelanggan(String id, int jumlahPesanan) {
        this.id = id;
        this.jumlahPesanan = jumlahPesanan;
    }
}

// Class utama
public class HapusSLL {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Queue<Pelanggan> queue = new LinkedList<>();

        int n = scanner.nextInt(); // Membaca jumlah pelanggan

        // Input data pelanggan dan simpan ke dalam queue
        for (int i = 0; i < n; i++) {
            String id = scanner.next();
            int jumlahPesanan = scanner.nextInt();
            queue.add(new Pelanggan(id, jumlahPesanan));
        }

        int waktuTotal = 0;

        // Memproses pelanggan dalam antrian
        while (!queue.isEmpty()) {
            Pelanggan pelanggan = queue.poll(); // Mengambil pelanggan dari
antrian
            waktuTotal += pelanggan.jumlahPesanan;
            System.out.println(pelanggan.id + " selesai dalam " +
waktuTotal + " menit");
        }

        scanner.close();
    }
}
```

BAB III
PENUTUP

3.1 Kesimpulan

Praktikum pekan ini memberikan pemahaman mendalam mengenai struktur data Single Linked List (SLL) dan penerapannya dalam berbagai operasi seperti penambahan di awal, akhir, maupun posisi tertentu, pencarian data, serta implementasi antrian menggunakan SLL. Dengan mengerjakan berbagai program seperti NodeSLL, TambahSLL, PencarianSLL, dan HapusSLL, mahasiswa belajar bagaimana linked list bekerja secara dinamis dan efisien dalam mengelola data. Hal ini memberikan dasar yang kuat dalam penggunaan struktur data lanjutan pada pemrograman.