LAPORAN HASIL PRAKTIKUM

Algoritma dan Struktur Data

Jobsheet 11

Alexsa Fitria Ayu Siswoyo.

244107020020

1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

PRAKTIKUM

Pembuatan Single Linked List

1. Mahasiswa02.java

```
package jobsheet11;

public class Mahasiswa02 {
    String nim;
    String Nama;
    String Kelas;
    double ipk;

    public Mahasiswa02(String nim, String nama, String kls,
double ip) {
        this.nim = nim;
        this.Nama = nama;
        this.Kelas = kls;
        this.ipk = ip;
    }

    public void tampilInformasi() {
        System.out.println("NIM\t\t: " + nim);
        System.out.println("Nama\t\t: " + Nama);
        System.out.println("Kelas\t\t: " + Kelas);
        System.out.println("IPK\t\t: " + ipk);
    }
}
```

2. Node02.java

```
package jobsheet11;

public class Node02 {
    Mahasiswa02 data;
    Node02 next;

    public Node02(Mahasiswa02 data, Node02 next) {
        this.data = data;
        this.next = next;
    }
}
```

3. SingleLinkedList02.java

```
package jobsheet11;

public class SingleLinkedList02 {
    Node02 head;
    Node02 tail;

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

    public void print() {
        if (isEmpty()) {
            Node02 tmp = head;
            System.out.print("Isi Linked List:\t");
            while (tmp != null) {
                tmp.data.tampilInformasi();
                tmp = tmp.next;
            }
            System.out.println("x");
        } else {
            System.out.println("Linked list kosong");
        }
    }

    public void addFirst(Mahasiswa02 input) {
        Node02 ndInput = new Node02(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            ndInput.next = head;
            head = ndInput;
        }
    }

    public void addLast(Mahasiswa02 input) {
        Node02 ndInput = new Node02(input, null);
        if (isEmpty()) {
            head = ndInput;
            tail = ndInput;
        } else {
            tail.next = ndInput;
            tail = ndInput;
        }
    }
```

```
    public void insertAfter(String key, Mahasiswa02 input) {
        Node02 ndInput = new Node02(input, null);
        Node02 temp = head;
        do {
            if (temp.data.Nama.equalsIgnoreCase(key)) {
                ndInput.next = temp.next;
                temp.next = ndInput;
                if (ndInput.next == null) {
                    tail = ndInput;
                }
                break;
            }
            temp = temp.next;
        } while (temp != null);
    }

    public void insertAt(int index, Mahasiswa02 input) {
        if (index < 0) {
            System.out.println("indeks salah");
        } else if (index == 0) {
            addFirst(input);
        } else {
            Node02 temp = head;
            for (int i = 0; i < index - 1; i++) {
                if (temp == null) {
                    System.out.println("indeks melebihi
batas");
                    return;
                }
                temp = temp.next;
            }
            Node02 ndInput = new Node02(input, temp.next);
            temp.next = ndInput;
            if (ndInput.next == null) {
                tail = ndInput;
            }
        }
    }
}
```

4. SLLMain02.java

```
package jobsheet11;

import java.util.LinkedList;
public class SLLMain02 {
   public static void main(String[] args) {
      SingleLinkedList02 sll = new SingleLinkedList02();

      Mahasiswa02 mhs1 = new Mahasiswa02("24212200",
"Alvaro", "1A", 4.0);
      Mahasiswa02 mhs2 = new Mahasiswa02("22212202",
"Cintia", "3C", 3.5);
      Mahasiswa02 mhs3 = new Mahasiswa02("23212201",
"Bimon", "2B", 3.8);
      Mahasiswa02 mhs4 = new Mahasiswa02("21212203",
"Dirga", "4D", 3.6);

      sll.print();
      sll.addFirst(mhs4);
      sll.print();
      sll.addLast(mhs1);
      sll.print();
      sll.insertAfter("Dirga", mhs3);
      sll.insertAt(2, mhs2);
      sll.print();
   }
}
```

PERTANYAAN

1. Mengapa hasil *compile* kode program di baris pertama menghasilkan “Linked List Kosong”?
   "Linked list kosong" pada pemanggilan pertama sll.print() adalah hasil dari inisialisasi linked list yang kosong dan belum adanya data yang dimasukkan ke dalamnya saat metode print() tersebut dipanggil.

2. Jelaskan kegunaan variabel temp secara umum pada setiap *method*!
   Variabel temp (atau tmp) umumnya digunakan sebagai **pointer sementara** untuk melintasi dan memanipulasi node-node di dalam linked list. Tujuan utamanya adalah untuk memungkinkan Anda bergerak melalui list tanpa kehilangan referensi ke head dari list, yang sangat penting untuk menjaga struktur list.

3. Lakukan modifikasi agar data dapat ditambahkan dari *keyboard*!

```
package jobsheet11;

import java.util.LinkedList;
import java.util.Scanner;
public class SLLMain02 {
   public static void main(String[] args) {
       SingleLinkedList02 sll = new
SingleLinkedList02();
        Scanner sc = new Scanner(System.in);

        System.out.print("Masukkan jumlah mahasiswa: ");
        int n = sc.nextInt();
        sc.nextLine();

        for (int i = 0; i < n; i++) {
            System.out.println("----------------------
------");
            System.out.print("Masukkan NIM mahasiswa:
");
            String nim = sc.nextLine();
            System.out.print("Masukkan nama mahasiswa:
");
            String nama = sc.nextLine();
            System.out.print("Masukkan kelas mahasiswa:
");
            String kls = sc.nextLine();
            System.out.print("Masukkan IPK mahasiswa:
");
            double ip = sc.nextDouble();
            sc.nextLine();

            Mahasiswa02 std = new Mahasiswa02(nim, nama,
kls, ip);
            sll.addLast(std);
        }

        sll.print();
   }
}
```

Modifikasi Elemen pada Single Linked List

1. SingleLinkedList02.java

```
    public void getData(int index) {
        Node02 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
        tmp.data.tampilInformasi();
    }

    public int indexOf(String key) {
        Node02 tmp = head;
        int index = 0;
        while (tmp != null &&
!tmp.data.Nama.equalsIgnoreCase(key)) {
            tmp = tmp.next;
            index++;
        }
        if (tmp == null) {
            return -1;
        } else {
            return index;
        }
    }

    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Linked List masih Kosong,
tidak dapat dihapus!");
        } else if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("Linked List masih Kosong,
tidak dapat dihapus!");
        } else if (head == tail) {
            head = tail = null;
        } else {
            Node02 temp = head;
            while (temp.next != tail) {
                temp = temp.next;
            }
            temp.next = null;
            tail = temp;
        }
    }
```

```
    public void remove(String key) {
        if (isEmpty()) {
            System.out.println("Linked List masih Kosong,
tidak dapat dihapus!");
        } else {
            Node02 temp = head;
            while (temp != null) {
                if ((temp.data.Nama.equalsIgnoreCase(key))
&& (temp == head)) {
                    this.removeFirst();
                    break;
                } else if
(temp.data.Nama.equalsIgnoreCase(key)) {
                    temp.next = temp.next.next;
                    if (temp.next == null) {
                        tail = temp;
                    }
                    break;
                }
                temp = temp.next;
            }
        }
    }

    public void removeAt(int index) {
        if (index == 0) {
            removeFirst();
        } else {
            Node02 temp = head;
            for (int i = 0; i < index - 1; i++) {
                temp = temp.next;
            }
            temp.next = temp.next.next;
            if (temp.next == null) {
                tail = temp;
            }
        }
    }
```

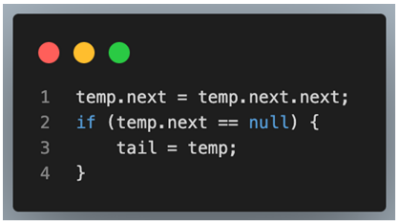
1. SLLMain02.java

```
System.out.println("data index 1 : ");
sll.getData(1);
System.out.println("data mahasiswa an Bimon berada pada
index : " + sll.indexOf("bimon"));
System.out.println();
sll.removeFirst();
sll.removeLast();
sll.print();
sll.removeAt(0);
sll.print();
```

Pertanyaan

1. Mengapa digunakan keyword break pada fungsi remove? Jelaskan!
   Keyword break digunakan pada fungsi remove untuk menghentikan perulangan while setelah node dengan kunci (key) yang dicari ditemukan dan dihapus.

2. Jelaskan kegunaan kode dibawah pada method remove



- **temp.next = temp.next.next;**: Baris ini adalah inti dari operasi penghapusan. Asumsikan temp adalah node *sebelum* node yang ingin dihapus. Baris ini mengubah pointer next dari node temp untuk langsung menunjuk ke node *setelah* node yang ingin dihapus. Dengan kata lain, node yang ingin dihapus "dilewati" dan tidak lagi menjadi bagian dari linked list.
- **if (temp.next == null) {**: Baris ini melakukan pengecekan apakah setelah penghapusan, node temp menjadi node terakhir dalam linked list. Ini terjadi jika node yang dihapus adalah node terakhir.
- **tail = temp;**: Jika kondisi pada baris sebelumnya benar (yaitu, temp.next menjadi null), maka node temp sekarang adalah node terakhir dalam linked list. Oleh karena itu, pointer tail (yang selalu menunjuk ke node terakhir) perlu diperbarui menjadi temp.

TUGAS

1. MahasiswaData02

```
package jobsheet11;

public class MahasiswaData02 {
    String nim, nama, programStudi, namaKelas;

public MahasiswaData02(String nim, String nama, String
programStudi, String namaKelas) {
    this.nim = nim;
    this.nama = nama;
    this.programStudi = programStudi;
    this.namaKelas = namaKelas;
}

public void tampilkanData() {
    System.out.println("NIM: " + nim);
    System.out.println("Nama: " + nama);
    System.out.println("Program Studi: " + programStudi);
    System.out.println("Kelas: " + namaKelas);
}
}
```

2. MahasiswaQueue02

```
package jobsheet11;

public class MahasiswaQueue02 {
    NodeMahasiswa02 front, rear;
    int size, maxsize;

    public MahasiswaQueue02(int max) {
        front = rear = null;
        maxsize = max;
        size = 0;
    }

    public boolean isEmpty() {
        return size == 0;
    }

    public boolean isFull() {
        return size == maxsize;
    }
```

```
    public void clear() {
        front = rear = null;
        size = 0;
        System.out.println("Antrian dibersihkan!");
    }

    public void enqueue(MahasiswaData02 student) {
        if (isFull()) {
            System.out.println("Antrian Penuh!");
            return;
        }
        NodeMahasiswa02 newNode = new
NodeMahasiswa02(student);
        if (isEmpty()) {
            front = rear = newNode;
        } else {
            rear.next = newNode;
            rear = newNode;
        }
        size++;
        System.out.println("Mahasiswa ditambahkan ke
antrian.");
    }

    public MahasiswaData02 dequeue() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
            return null;
        }
        MahasiswaData02 data = front.data;
        if (front == rear) {
            front = rear = null;
        } else {
            front = front.next;
        }
        size--;
        return data;
    }

    public void displayFront() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
            return;
        }
        System.out.println("Mahasiswa di depan:");
        front.data.print();
    }
```

```
    public void displayRear() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
            return;
        }
        System.out.println("Mahasiswa di belakang:");
        rear.data.print();
    }

    public void displayAll() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
            return;
        }
        NodeMahasiswa02 temp = front;
        System.out.println("----- Semua Mahasiswa dalam
Antrian -----");
        while (temp != null) {
            temp.data.print();
            System.out.println("----------------------------
--");
            temp = temp.next;
        }
    }

    public int getSize() {
        return size;
    }
}
```

3. MahasiswaQueueMain02

```
package jobsheet11;

import java.util.Scanner;

public class MahasiswaQueueMain02 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        MahasiswaQueue02 queue = new MahasiswaQueue02(5);
        int pilihan;

        do {
            System.out.println("==== Menu Antrian Layanan
Mahasiswa ====");
            System.out.println("1. Daftar Mahasiswa
(Enqueue)");
            System.out.println("2. Panggil Mahasiswa
Berikutnya (Dequeue)");
            System.out.println("3. Tampilkan Mahasiswa di
Depan");
            System.out.println("4. Tampilkan Mahasiswa di
Belakang");
            System.out.println("5. Tampilkan Semua
Mahasiswa");
            System.out.println("6. Ukuran Antrian");
            System.out.println("7. Kosongkan Antrian");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = sc.nextInt();
            sc.nextLine();
            switch (pilihan) {
                case 1:
                    if (queue.isFull()) {
                        System.out.println("Antrian
Penuh!");
                    } else {
                        System.out.print("NIM: ");
                        String nim = sc.nextLine();
                        System.out.print("Nama: ");
                        String nama = sc.nextLine();
                        System.out.print("Program Studi:
");
                        String programStudi =
sc.nextLine();
                        System.out.print("Kelas: ");
                        String namaKelas = sc.nextLine();
                        MahasiswaData02 mahasiswa = new
MahasiswaData02(nim, nama, programStudi, namaKelas);
                        queue.enqueue(mahasiswa);
                    }
                    break;
```

```
                case 2:
                    MahasiswaData02 servedMahasiswa =
queue.dequeue();
                    if (servedMahasiswa == null) {
                        System.out.println("Antrian
Kosong!");
                    } else {
                        System.out.println("Mahasiswa yang
dilayani:");
                        servedMahasiswa.print();
                    }
                    break;
                case 3:
                    queue.displayFront();
                    break;
                case 4:
                    queue.displayRear();
                    break;
                case 5:
                    queue.displayAll();
                    break;
                case 6:
                    System.out.println("Jumlah mahasiswa
dalam antrian: " + queue.getSize());
                    break;
                case 7:
                queue.clear();
                System.out.println("Antrian dikosongkan.");
                    break;
                case 0:
                    System.out.println("Terima kasih!");
                    break;
                default:
                    System.out.println("Menu tidak valid!");
                    break;
            }
            System.out.println();
        } while (pilihan != 0);

        sc.close();

    }
}
```

4. NodeMahasiswa02

```
package jobsheet11;

public class NodeMahasiswa02 {
    MahasiswaData02 data;
    NodeMahasiswa02 next;

    public NodeMahasiswa02(MahasiswaData02 data) {
        this.data = data;
        this.next = null;
    }
}
```