LAPORAN JOBSHEET 12

KONSEP DASAR PEMROGRAMAN

Mata Kuliah : Algoritma dan Struktur Data

Dosen : **Mungki Astiningrum, S.T., M.Kom.**

Alfreda Dhaifullah Mahezwara
244107020219
Kelas : 1A
Absen : 04

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG TAHUN 2025

Praktikum 1

Class Node

```
package Pertemuan_13.Double_LinkedList;

public class Node004 {
    int data;
    Node004 prev;
    Node004 next;

    Node004 (Node004 prev, int data,Node004 next ) {
        this.data = data;
        this.prev = prev;
        this.next = next;
    }
}
```

Class Double Linked List

```
package Pertemuan_13.Double_LinkedList;

public class DoubleLinkedList {
    Node004 head;
    int size;

    public DoubleLinkedList () {
        head = null;
        size = 0;
    }

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

    // Menthod untuk penambahkan data di bagian depan LinkedList
    public void addFirst (int item) {
        if (isEmpty()) {
```

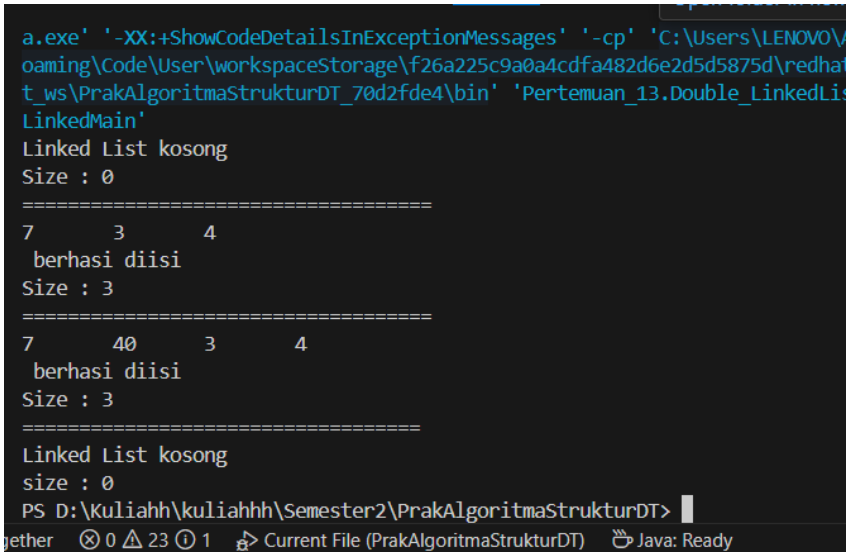
```
            head = new Node004(null, item, null);
        } else {
            Node004 newNode = new Node004(head, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    // Menthod untuk menambahkan data pada bagian belakang LinkedList
    public void addLast (int item) {
        if (isEmpty()) {
            addFirst(item);
        } else {
            Node004 current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node004 newNode = new Node004(current, item, null);
            current.next = newNode;
            size++;
        }
    }

    //menthod untuk menambahkan data pada posisi yang telah ditentukan dengan
indeks
    public void add (int item, int index) throws Exception {
        if (isEmpty()) {
            addFirst(item);
        } else if (index < 0 || index > size) {
            throw new Exception("Nilai indeks di luar batas");
        } else {
            Node004 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
```

```
        }
        if (current.prev == null) {
            Node004 newNode = new Node004(null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            Node004 newNode = new Node004(current.prev, item, current);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
    }
}

public int size () {
    return size;
}

//method untuk menghapus semua isi linked List
public void clear() {
    head = null;
    size = 0;
}

public void print () {
    if (!isEmpty()) {
        Node004 tmp = head;
        while (tmp != null) {
            System.out.print(tmp.data + "\t");
            tmp = tmp.next;
        }
        System.out.println("\n berhasi diisi");
    } else {
        System.out.println("Linked List kosong");
    }
}
```

```
}
```

Class Main

```
package Pertemuan_13.Double_LinkedList;


public class DoubleLinkedMain {
    public static void main(String[] args) throws Exception {
        DoubleLinkedList dll = new DoubleLinkedList();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.add(40, 1);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==================================");
        dll.clear();
        dll.print();
        System.out.println("size : " + dll.size());
    }
}
```



Pertanyaan Praktikum 1

1. Jelaskan perbedaan antara single linked list dengan double linked list!
   - Single : hanya memiliki 1 pointer ynag menunjuk pada node berikutnya
   - Double : setiap node memiliki dua pointer yaitu ke node berikutnya dan node sebelumnya.
2. Perhatikan class Node, didalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut
   - Atribut next dan prev pada class Node digunakan untuk membentuk struktur double LinkedList. Dengan adanya kedua atribut ini, linked list dapat diakses dari depan ke belakang maupun dari belakang ke depan, sehingga operasi traversal, penambahan, dan penghapusan node menjadi lebih fleksibel dan efisien.
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?
   - Untuk instansiasi awal bahwa linked list masih kosong
4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?
   - Pada method addfirst(), objek baru dibuat dengan prev bernilai null karena node tersebut akan menjadi node paling depan (head) pada double linked list. Node paling depan tidak memiliki node sebelumnya, sehingga atribut prev harus menunjuk ke null. Ini adalah penanda bahwa node tersebut adalah awal dari linked list.
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
   - Pada method addFirst(), statement head.prev = newNode; berarti node yang sebelumnya menjadi head sekarang harus menunjuk ke node baru (newNode) sebagai node sebelumnya (prev).
6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?

- prev = current: Node baru (newNode) akan menjadi node paling belakang, sehingga node sebelumnya (prev) adalah node yang saat ini berada di posisi paling belakang (current).
- next = null: Karena node baru akan menjadi node terakhir, maka tidak ada node setelahnya, sehingga next di-set null.

Praktikum 2

Class Double Linked List

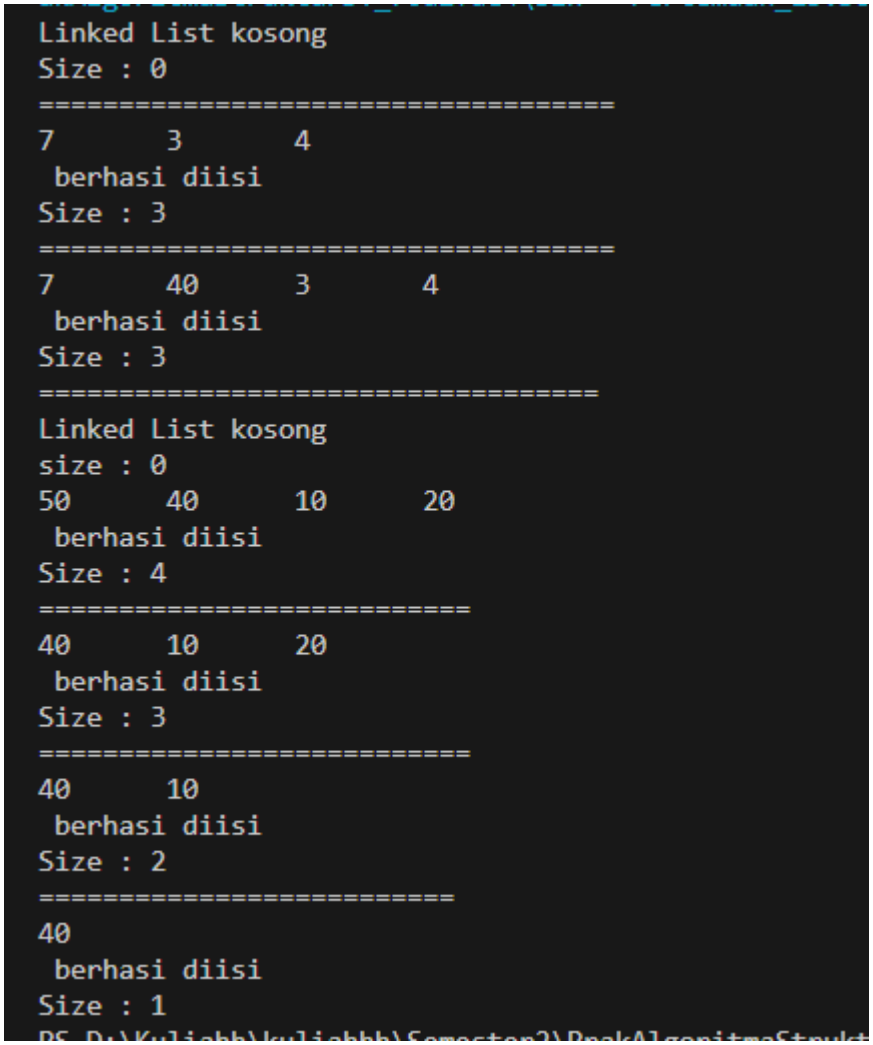
```
// Method untuk menghapus bagian depan
    public void removeFirst () throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked lisr masih kosong, tidak dapat dihapus");
        } else if (size == 1) {
            removeLast();
        } else {
            head = head.next;
            head.prev = null;
            size--;
        }
    }

    //Method untuk menghapus dari belakang
    public void removeLast () throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List masih kosong, tidak dapat
menghapus!");
        } else if (head.next == null) {
            head = null;
            size--;
            return;
        }
        Node004 current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        current.next = null;
        size--;
    }
```

```
    // method untuk menghapus data pada index yang telah ditentukan
    public void remove (int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node004 current = head;
            int i = 0;
            while (i < index) {
                current = current.next;
                i++;
            }
            if (current.next == null) {
                current.prev.next = null;
            } else if (current.prev == null) {
                current = current.next;
                current.prev = null;
                head = current;
            } else {
                current.prev.next = current.next;
                current.next.prev = current.prev;
            }
            size--;
        }
    }
```

Class Main

```
 dll.addLast(50);
        dll.addLast(40);
        dll.addLast(10);
        dll.addLast(20);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================");
```

```
        dll.removeFirst();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================");
        dll.removeLast();
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("==========================");
        dll.remove(1);
        dll.print();
        System.out.println("Size : " + dll.size());
```



Pertanyaan Praktikum 1

1. Apakah maksud statement berikut pada method removeFirst()? head = head.next; head.prev = null;
   - head = head.next;
     Memindahkan pointer head ke node berikutnya, sehingga node pertama (yang lama) dihapus dari linked list.
   - head.prev = null;
     Menghapus hubungan ke node sebelumnya pada node baru yang sekarang menjadi head, sehingga node head benar-benar menjadi node pertama (tidak punya node sebelum dirinya).

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
   - Pada method removeLast(), posisi data berada pada bagian akhir (node terakhir) jika node tersebut memiliki atribut next bernilai null.

3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah remove!
   - Kode ini hanya menghapus node kedua (setelah head), bukan node pada posisi tertentu (misal: node yang ingin dihapus).
     Jika ingin menghapus node pada posisi tertentu, seharusnya node sebelum dan sesudah node yang dihapus harus dihubungkan langsung, bukan selalu dari head.Jika tmp.next adalah null (artinya node kedua adalah node terakhir), maka tmp.next.prev = head; akan menyebabkan error NullPointerException.
4. Jelaskan fungsi kde progam berikut ini pada fungsi remove!
   - current.prev.next = current.next;
     Menghubungkan node sebelum current langsung ke node setelah current, sehingga node current dilewati dari arah depan.
     current.next.prev = current.prev;
     Menghubungkan node setelah current langsung ke node sebelum current, sehingga node current dilewati dari arah belakang.

Praktikum 3

Class Double Liked List

```
// mehtod untuk mendapatkan data pada awal Linked List
    public int getFirst () throws Exception {
        if (isEmpty()) {
            throw new Exception(" Linked list kosong");
        }
        return head.data;
```

```
    }

    // Method untuk mendapatkan data pada akhir Linked List
    public int getLast () throws Exception {
        if (isEmpty()) {
            throw new Exception("linked list masih kosong");
        }
        Node004 tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    // method untuk mendapatkan data pada indeks yang telah ditentukan
    public int get (int indeks) throws Exception {
        if (isEmpty() || indeks >= size) {
            throw new Exception("Nilai indeks diluar batas");
        }
        Node004 tmp = head;
        for (int i = 0; i < indeks; i++) {
            tmp = tmp.next;
        }
        return tmp.data;

    }
```

Class Main

```
dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println("============================");
        dll.addFirst(3);
        dll.addLast(4);
        dll.addFirst(7);
        dll.print();
        System.out.println("Size : " + dll.size());
```

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

        dll.add(40, 1);

        dll.print();

        System.out.println("Size : " + dll.size());

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

        System.out.println("Data awala oada Linked list adalah: " +
dll.getFirst());

        System.out.println("Data akhir pada Linked List adalah : " +
dll.getLast());

        System.out.println("Data indeks ke-1 pada Linked Lists adalah : " +
dll.get(1));



    }
```

```
Linked List kosong
Size : 0
==================================
7       3       4
 berhasi diisi
Size : 3
==================================
7       40      3       4
 berhasi diisi
Size : 3
==================================
Linked List kosong
size : 0
50      40      10      20
 berhasi diisi
Size : 4
==========================
40      10      20
 berhasi diisi
Size : 3
==========================
40      10
 berhasi diisi
Size : 2
Size : 2
==========================
40
 berhasi diisi
Size : 1
40
 berhasi diisi
Size : 1
==========================
7       3       40      4
 berhasi diisi
Size : 4
Size : 2
==========================
```

```
==========================
40
 berhasi diisi
Size : 1
40
 berhasi diisi
Size : 1
Size : 2
==========================
40
==========================
40
40
 berhasi diisi
Size : 1
40
 berhasi diisi
Size : 1
==========================
 berhasi diisi
Size : 1
40
 berhasi diisi
 berhasi diisi
Size : 1
40
 berhasi diisi
Size : 1
==========================
7       3       40      4
 berhasi diisi
Size : 4
==========================
7       40      3       40      4
 berhasi diisi
Size : 4
==========================
Data awala oada Linked list adalah: 7
Data akhir pada Linked List adalah : 4
Data indeks ke-1 pada Linked Lists adalah : 40
PS D:\Kuliahh\kuliahhh\Semester2\PrakAlgoritmaStrukturDT>
```

Pertanyaan Praktikum 3

1. Jelaskan method size() pada class DoubleLinkedLists!
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke□1!
3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini! (a) (b

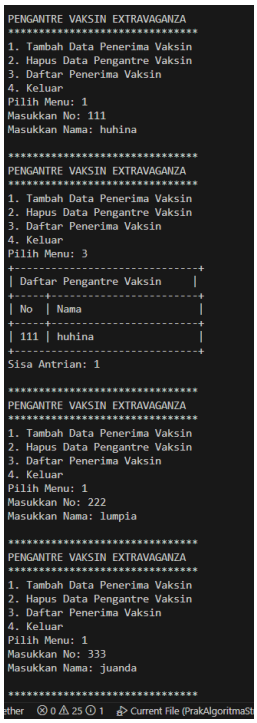
Tugas

```
package Pertemuan_13.Double_LinkedList.tugas;

import java.util.Scanner;

class Node {
    int no;
    String nama;
    Node prev, next;

    Node(int no, String nama) {
        this.no = no;
        this.nama = nama;
        this.prev = null;
        this.next = null;
    }
}

class Queue {
    private Node head, tail;
    private int count = 0;
    private int totalVaksinasi = 0;

    public void enqueue(int no, String nama) {
        Node newNode = new Node(no, nama);
        if (head == null) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        count++;
    }
```

```
    public void dequeue() {
        if (head == null) {
            System.out.println("Antrian kosong.");
            return;
        }

        System.out.println("Data " + head.nama + " telah selesai divaksinasi.");
        head = head.next;
        if (head != null) head.prev = null;
        else tail = null;

        count--;
        totalVaksinasi++;
    }

    public void display() {
        Node current = head;
        System.out.println("+------------------------------+");
        System.out.println("| Daftar Pengantre Vaksin     |");
        System.out.println("+-----+------------------------+");
        System.out.println("| No  | Nama                    |");
        System.out.println("+-----+------------------------+");

        while (current != null) {
            System.out.printf("| %-3d | %-22s |\n", current.no, current.nama);
            current = current.next;
        }

        System.out.println("+------------------------------+");
        System.out.println("Sisa Antrian: " + count);
    }

    public int getTotalVaksinasi() {
        return totalVaksinasi;
    }
}
```

```
class AntrianVaksin {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Queue antrian = new Queue();

        int pilihan;
        do {
            System.out.println("*******************************");
            System.out.println("PENGANTRE VAKSIN EXTRAVAGANZA");
            System.out.println("*******************************");
            System.out.println("1. Tambah Data Penerima Vaksin");
            System.out.println("2. Hapus Data Pengantre Vaksin");
            System.out.println("3. Daftar Penerima Vaksin");
            System.out.println("4. Keluar");
            System.out.print("Pilih Menu: ");
            pilihan = sc.nextInt();
            sc.nextLine(); // buang newline

            switch (pilihan) {
                case 1:
                    System.out.print("Masukkan No: ");
                    int no = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Masukkan Nama: ");
                    String nama = sc.nextLine();
                    antrian.enqueue(no, nama);
                    break;

                case 2:
                    antrian.dequeue();
                    System.out.println("Total yang sudah divaksin: " +
antrian.getTotalVaksinasi());
                    break;

                case 3:
                    antrian.display();
```

```
                    break;

                case 4:
                    System.out.println("Terima kasih telah menggunakan program.");
                    break;

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

            System.out.println();

        } while (pilihan != 4);
    }
}
```



https://github.com/AlfredaDhaifullah04/Semester-2/tree/master/Pertemuan_13/Double_LinkedList