POLITEKNIK NEGERI MALANG

TEKNOLOGI INFORMASI

TEKNIK INFORMATIKA

Nama: Abdul Rahman Hanif Darmawan

NIM: 244107020232

Kelas: TI-1A

Prodi: D4-TEKNIK INFORMATIKA

JOBSHEET 12
Double Linked Lists

12.2.2 Verifikasi Hasil Percobaan

-

```
Linked Lists Kosong
Size : 0
====================
7       3       4
berhasil diisi
Size : 3
====================
7       40      3       4
berhasil diisi
Size : 4
====================
Linked Lists Kosong
Size : 0
```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
   - Single linked list hanya memiliki pointer ke node berikutnya (next), sedangkan double linked list memiliki pointer ke node berikutnya (next) dan node sebelumnya (prev).
2. Perhatikan class Node, didalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   - next: Menunjuk ke node berikutnya dalam daftar dan prev: Menunjuk ke node sebelumnya dalam daftar. (Khusus untuk double linked list).
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

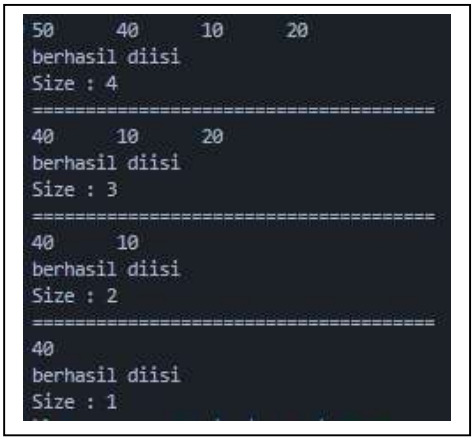
```
public DoubleLinkedLists() {
    head = null;
    size = 0;
}
```

   - head = null;: Mengindikasikan bahwa daftar kosong (belum ada elemen) dan size = 0;: Mengatur jumlah elemen dalam daftar menjadi nol karena daftar masih kosong.

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?
   `Node newNode = new Node(null, item, head);`
   - Karena node baru yang ditambahkan di awal (addFirst) akan menjadi node pertama dalam daftar, sehingga tidak ada node sebelumnya yang ditunjuk oleh prev.
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
   - Jika daftar tidak kosong, statement ini mengatur pointer prev dari node yang saat ini menjadi head untuk menunjuk ke newNode yang baru saja ditambahkan di depannya. Ini mengikat node lama ke node baru.
6. Perhatikan isi method addLast(), apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?
   `Node newNode = new Node(current, item, null);`
   - prev dengan current: Node baru akan ditambahkan di akhir daftar, jadi node current (yang merupakan node terakhir sebelum penambahan) akan menjadi node sebelumnya.
   - next dengan null: Node baru ini akan menjadi node terakhir, sehingga tidak ada node berikutnya yang ditunjuk oleh next.

12.3.2 Verifikasi Hasil Percobaan

-



12.3.3 Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method removeFirst()?
```
head = head.next;
head.prev = null;
```
   - Baris pertama menggeser head ke node berikutnya, menjadikan node kedua sebagai kepala baru. Baris kedua memastikan node kepala baru tidak memiliki referensi prev
2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
   - Dideteksi dengan mengiterasi sampai current.next.next == null. Yang berarti current.next adalah node terakhir.
3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah remove!
```
Node tmp = head.next;

head.next=tmp.next;
tmp.next.prev=head;
```
   - Kode ini hanya cocok untuk menghapus node kedua. Jika head.next adalah null (list hanya punya satu node) atau tmp.next adalah null (list hanya punya dua node), kode ini akan menyebabkan NullPointerException.
4. Jelaskan fungsi kode program berikut ini pada fungsi remove!
```
current.prev.next = current.next;
current.next.prev = current.prev;
```
   - Kode ini menyambungkan node sebelum current dengan node setelah current, dan sebaliknya. Ini efektif "melewati" node current, sehingga menghapusnya dari list tanpa mengubah head atau tail secara langsung.

12.4.2 Verifikasi Hasil Percobaan

```
Linked Lists Kosong
Size: 0
============================
7       3       4
berhasil diisi
Size: 3
============================
7       40      3       4
berhasil diisi
Size: 4
============================
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method size() pada class DoubleLinkedLists!
   - Method size() mengembalikan jumlah elemen yang saat ini ada dalam DoubleLinkedLists.
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke- 1!
   - Untuk membuat indeks dimulai dari 1, Perlu menyesuaikan semua logika yang berinteraksi dengan indeks (seperti add, remove, get) dengan mengrangi 1 dari nilai indeks yang diterima sebelum digunakan untuk traversing atau perhitungan (misalnya, index - 1).
3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!
   - Pada Double Linked List, fungsi add juga memperbarui pointer prev pada node baru dan/atau node tetangga, selain pointer next. Pada Single Linked List, fungsi add hanya memperbarui pointer next.
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

```java
public boolean isEmpty(){
    if(size ==0){
        return true;
    } else{
        return false;
    }
}
```

(a)

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

(b)

   - **(a):** Menentukan apakah list kosong berdasarkan nilai variabel size. Jika size adalah 0, maka list dianggap kosong.
   - **(b):** Menentukan apakah list kosong berdasarkan apakah head (pointer ke node pertama) adalah null. Jika head adalah null, berarti tidak ada node dalam list.

12.5 Tugas Praktikum

Tugas 1

Node.java

```
package Tugas.Vaksin;

public class Node {
    int noAntrian;
    String nama;
    Node next;
    Node prev;

    public Node(int noAntrian, String nama) {
        this.noAntrian = noAntrian;
        this.nama = nama;
        this.next = null;
        this.prev = null;
    }
}
```

VaksinasiQueue.java

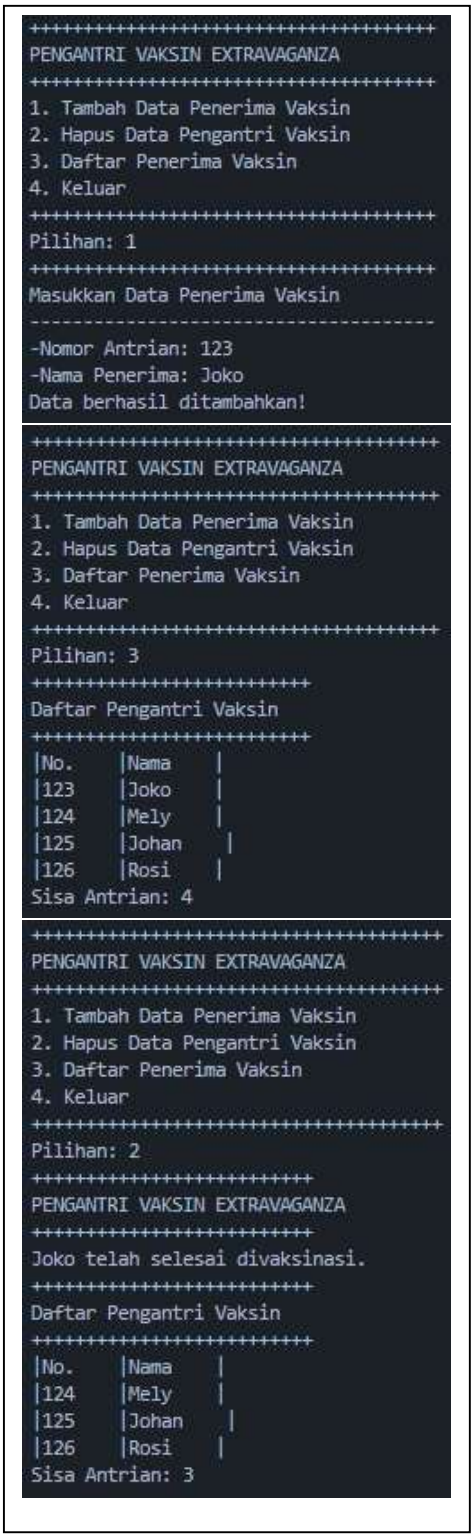
```
package Tugas.Vaksin;

public class VaksinasiQueue {
    Node head;
    Node tail;
    int size;
    int nextAntrian;

    public VaksinasiQueue() {
        head = null;
        tail = null;
        size = 0;
        nextAntrian = 123; // Mulai dari nomor antrian 123
    }

    public void tambahData(String nama) {
        Node newNode = new Node(nextAntrian, nama);

        if (head == null) {
            head = newNode;
            tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }

        nextAntrian++;
        size++;
        System.out.println("Data berhasil ditambahkan!");
    }

    public void hapusData() {
        if (isEmpty()) {
            System.out.println("Antrian kosong!");
            return;
        }

        System.out.println("++++++++++++++++++++++++++");
        System.out.println("PENGANTRI VAKSIN EXTRAVAGANZA");
        System.out.println("++++++++++++++++++++++++++");
        System.out.println(head.nama + " telah selesai divaksinasi.");

        Node temp = head;
        head = head.next;

        if (head == null) {
            tail = null;
        } else {
            head.prev = null;
        }

        size--;
        daftarPenerima();
    }

    public void daftarPenerima() {
        if (isEmpty()) {
            System.out.println("Daftar kosong!");
            return;
        }

        System.out.println("++++++++++++++++++++++++++");
        System.out.println("Daftar Pengantri Vaksin");
```

```
        System.out.println("++++++++++++++++++++++++++");
        System.out.println("|No.    |Nama    |");

        Node current = head;
        while (current != null) {
            System.out.println("|" + current.noAntrian + "    |" + current.nama +
"    |");
            current = current.next;
        }

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

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

VaksinMain.java

```
package Tugas.Vaksin;

import java.util.Scanner;

public class VaksinMain {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        VaksinasiQueue antrian = new VaksinasiQueue();
        int pilihan = 0;

        do {
            System.out.println("++++++++++++++++++++++++++++++++++++++");
            System.out.println("PENGANTRI VAKSIN EXTRAVAGANZA");
            System.out.println("++++++++++++++++++++++++++++++++++++++");
            System.out.println("1. Tambah Data Penerima Vaksin");
            System.out.println("2. Hapus Data Pengantri Vaksin");
            System.out.println("3. Daftar Penerima Vaksin");
            System.out.println("4. Keluar");
            System.out.println("++++++++++++++++++++++++++++++++++++++");
            System.out.print("Pilihan: ");

            try {
                pilihan = scanner.nextInt();
                scanner.nextLine();

                switch (pilihan) {
                    case 1:
                        System.out.println("++++++++++++++++++++++++++++++++++++++
");
                        System.out.println("Masukkan Data Penerima Vaksin");
                        System.out.println("--------------------------------------
");
                        System.out.println("-Nomor Antrian: " +
antrian.nextAntrian);
                        System.out.print("-Nama Penerima: ");
                        String nama = scanner.nextLine();
                        antrian.tambahData(nama);
                        break;
                    case 2:
                        antrian.hapusData();
                        break;
                    case 3:
                        antrian.daftarPenerima();
                        break;
                    case 4:
                        System.out.println("Terima kasih telah menggunakan program
ini.");
                        break;
                    default:
                        System.out.println("Pilihan tidak valid!");
                }
            } catch (Exception e) {
                System.out.println("Input tidak valid! " + e.getMessage());
                scanner.nextLine();
            }

            System.out.println();
        } while (pilihan != 4);
    }
}
```

Output


PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilihan: 1
Masukkan Data Penerima Vaksin
-Nomor Antrian: 123
-Nama Penerima: Joko
Data berhasil ditambahkan!
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilihan: 3
Daftar Pengantri Vaksin
No.	Nama
123	Joko
124	Mely
125	Johan
126	Rosi
Sisa Antrian: 4
PENGANTRI VAKSIN EXTRAVAGANZA
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
Pilihan: 2
PENGANTRI VAKSIN EXTRAVAGANZA
Joko telah selesai divaksinasi.
Daftar Pengantri Vaksin
No.	Nama
124	Mely
125	Johan
126	Rosi
Sisa Antrian: 3

Tugas 2

Film.java

```
package Tugas.Film;
public class Film {

    int id;

    String judul;

    double rating;


    Film(int id, String judul, double rating) {

        this.id = id;

        this.judul = judul;

        this.rating = rating;

    }

}
```

Node.java

```
package Tugas.Film;

public class Node {
    Film data;
    Node prev, next;

    Node(Film data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

DoubleLinkedList.java

```
package Tugas.Film;

public class DoubleLinkedList {
    Node head, tail;

    void tambahDepan(Film data) {
        Node baru = new Node(data);
        if (head == null) {
            head = tail = baru;
        } else {
            baru.next = head;
            head.prev = baru;
            head = baru;
        }
    }

    void tambahBelakang(Film data) {
        Node baru = new Node(data);
        if (head == null) {
            head = tail = baru;
        } else {
            tail.next = baru;
            baru.prev = tail;
            tail = baru;
        }
    }

    void tambahDiIndex(Film data, int index) {
        if (index == 0) {
            tambahDepan(data);
            return;
        }

        Node baru = new Node(data);
        Node bantu = head;
        int i = 0;

        while (bantu != null && i < index - 1) {
            bantu = bantu.next;
            i++;
        }

        if (bantu == null || bantu.next == null) {
            tambahBelakang(data);
            return;
        }

        baru.next = bantu.next;
        baru.prev = bantu;
        bantu.next.prev = baru;
        bantu.next = baru;
    }

    void hapusDepan() {
        if (head == null) {
            System.out.println("List kosong!");
        } else {
            head = head.next;
            if (head != null) head.prev = null;
            else tail = null;
        }
    }

    void hapusBelakang() {
```

```
        if (tail == null) {
            System.out.println("List kosong!");
        } else {
            tail = tail.prev;
            if (tail != null) tail.next = null;
            else head = null;
        }
    }

    void hapusById(int id) {
        Node bantu = head;

        while (bantu != null && bantu.data.id != id) {
            bantu = bantu.next;
        }

        if (bantu == null) {
            System.out.println("Film tidak ditemukan!");
        } else if (bantu == head) {
            hapusDepan();
        } else if (bantu == tail) {
            hapusBelakang();
        } else {
            bantu.prev.next = bantu.next;
            bantu.next.prev = bantu.prev;
        }
    }

    void cetak() {
        if (head == null) {
            System.out.println("List kosong!");
            return;
        }

        Node bantu = head;
        while (bantu != null) {
            System.out.println("ID: " + bantu.data.id);
            System.out.println("Judul: " + bantu.data.judul);
            System.out.println("Rating: " + bantu.data.rating);
            System.out.println();
            bantu = bantu.next;
        }
    }

    Film cariById(int id) {
        Node bantu = head;
        while (bantu != null) {
            if (bantu.data.id == id) {
                return bantu.data;
            }
            bantu = bantu.next;
        }
        return null;
    }

    void urutkanRatingDesc() {
        if (head == null || head.next == null) return;

        boolean tukar;
        do {
            tukar = false;
            Node bantu = head;
            while (bantu.next != null) {
                if (bantu.data.rating < bantu.next.data.rating) {
```

```
                    Film temp = bantu.data;
                    bantu.data = bantu.next.data;
                    bantu.next.data = temp;
                    tukar = true;
                }
                bantu = bantu.next;
            }
        } while (tukar);
    }
}
```

FilmMain.java

```
package Tugas.Film;
import java.util.Scanner;
public class FilmMain {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedList daftar = new DoubleLinkedList();

        int pilihan;
        do {
            System.out.println("=== MENU ===");
            System.out.println("1. Tambah Depan");
            System.out.println("2. Tambah Belakang");
            System.out.println("3. Tambah di Index");
            System.out.println("4. Hapus Depan");
            System.out.println("5. Hapus Belakang");
            System.out.println("6. Hapus berdasarkan ID");
            System.out.println("7. Cetak");
            System.out.println("8. Cari Film");
            System.out.println("9. Urutkan Rating (DESC)");
            System.out.println("10. Keluar");
            System.out.print("Pilih: ");
            pilihan = sc.nextInt();

            switch (pilihan) {
                case 1 -> {
                    System.out.print("ID: ");
                    int id = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Judul: ");
                    String judul = sc.nextLine();
                    System.out.print("Rating: ");
                    double rating = sc.nextDouble();
                    daftar.tambahDepan(new Film(id, judul, rating));
                }
                case 2 -> {
                    System.out.print("ID: ");
                    int id = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Judul: ");
                    String judul = sc.nextLine();
                    System.out.print("Rating: ");
                    double rating = sc.nextDouble();
                    daftar.tambahBelakang(new Film(id, judul, rating));
                }
                case 3 -> {
                    System.out.print("ID: ");
                    int id = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Judul: ");
                    String judul = sc.nextLine();
                    System.out.print("Rating: ");
                    double rating = sc.nextDouble();
                    System.out.print("Index: ");
                    int index = sc.nextInt();
                    daftar.tambahDiIndex(new Film(id, judul, rating), index);
                }
                case 4 -> daftar.hapusDepan();
                case 5 -> daftar.hapusBelakang();
                case 6 -> {
                    System.out.print("ID yang akan dihapus: ");
                    int id = sc.nextInt();
                    daftar.hapusById(id);
                }
                case 7 -> daftar.cetak();
```

```
                case 8 -> {
                    System.out.print("Cari ID: ");
                    int id = sc.nextInt();
                    Film f = daftar.cariById(id);
                    if (f != null) {
                        System.out.println("Ditemukan: " + f.judul + " - " +
f.rating);
                    } else {
                        System.out.println("Film tidak ditemukan.");
                    }
                }
                case 9 -> {
                    daftar.urutkanRatingDesc();
                    System.out.println("Data diurutkan!");
                    daftar.cetak();
                }
                case 10 -> System.out.println("Keluar...");
                default -> System.out.println("Pilihan salah!");
            }
            System.out.println();
        } while (pilihan != 10);

        sc.close();
    }
}
```

Output

Link Github