JOBSHEET 12

LAPORAN HASIL PRAKTIKUM

ALGORITMA DAN STRUKTUR

DATA

MUHAMMAD DAFFI FIROS ZAIDAN

244107020182

TI 1E

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

12.2.1 Percobaan 1

Class Mahasiswa17

```
package Jobsheet12;
public class Mahasiswa17 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa17(String nim, String nama, String kelas, double ipk) {
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampil() {
        System.out.println("NIM: " + nim + ", Nama: " + nama + ", Kelas: " + kelas + ", IPK: " +
ipk);
    }
}
```

Class Node17

```
package Jobsheet12;
public class Node17 {
    Mahasiswa17 data;
    Node17 prev;
    Node17 next;

    public Node17(Mahasiswa17 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

Class DoubleLinkList17

```
public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat dihapus!");
        } else if (size == 1) {
            head = null;
            size--;
        } else {
            head = head.next;
            head.prev = null;
            size--;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat dihapus!");
        } else if (head.next == null) {
            head = null;
            size--;
        } else {
            Node17 current = head;
            while (current.next.next != null) {
                current = current.next;
            }
            current.next = null;
            size--;
        }
    }

    public void print() {
        if (!isEmpty()) {
            Node17 tmp = head;
            System.out.println("Data Mahasiswa:");
            while (tmp != null) {
                tmp.data.tampil();
                tmp = tmp.next;
            }
        } else {
            System.out.println("Linked List kosong");
        }
    }

    public Node17 search(String nim) {
        Node17 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }
}
```

Class DLLMain17

```
package Jobsheet12;
import java.util.Scanner;
public class DLLMain17 {
    static Scanner sc = new Scanner(System.in);
    public static Mahasiswa17 inputMahasiswa() {
        System.out.print("Masukkan NIM: ");
        String nim = sc.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = sc.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = sc.nextLine();
        System.out.print("Masukkan IPK: ");
        double ipk = sc.nextDouble();
        sc.nextLine();
        Mahasiswa17 mhs = new Mahasiswa17(nim, nama, kelas, ipk);
        return mhs;
    }

    public static void main(String[] args) {
        DoubleLinkedLists17 list = new DoubleLinkedLists17();
        int pilihan;
```

```
do {
            System.out.println("\nMenu Double Linked List Mahasiswa");
            System.out.println("1. Tambah di awal");
            System.out.println("2. Tambah di akhir");
            System.out.println("3. Hapus di awal");
            System.out.println("4. Hapus di akhir");
            System.out.println("5. Tampilkan data");
            System.out.println("6. Cari Mahasiswa berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = sc.nextInt();
            sc.nextLine();

            switch (pilihan) {
                case 1:
                    Mahasiswa17 mhs1 = inputMahasiswa();
                    list.addFirst(mhs1);
                    break;
                case 2:
                    Mahasiswa17 mhs2 = inputMahasiswa();
                    list.addLast(mhs2);
                    break;
                case 3:
                    list.removeFirst();
                    break;
                case 4:
                    list.removeLast();
                    break;
                case 5:
                    list.print();
                    break;
                case 6:
                    System.out.print("Masukkan NIM yang ingin dicari: ");
                    String nimCari = sc.nextLine();
                    Node17 found = list.search(nimCari);
                    if (found != null) {
                        System.out.println("Data Mahasiswa ditemukan:");
                        found.data.tampil();
                    } else {
                        System.out.println("Data Mahasiswa dengan NIM " + nimCari + "
tidak ditemukan.");
                    }
                    break;
                case 0:
                    System.out.println("Keluar dari program.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid!");
                    break;
            }
        } while (pilihan != 0);

        sc.close();
    }
}
```

g

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!

   Jawaban: ***Single linked list hanya memiliki satu arah, yaitu setiap node hanya tahu node setelahnya (menggunakan pointer next).***
   Sedangkan double linked list punya dua arah, karena setiap node tahu node sebelum dan sesudahnya (menggunakan prev dan next). Jadi lebih fleksibel untuk maju-mundur.

2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?

   Jawaban: ***next digunakan untuk menunjuk ke node berikutnya.prev digunakan untuk menunjuk ke node sebelumnya. Ini digunakan di double linked list agar bisa bergerak ke depan dan ke belakang.***

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?

```
public DoubleLinkedList01() {
    head = null;
    tail = null;
}
```

   Jawaban: ***Konstruktor digunakan untuk mengatur nilai awal dari head dan tail menjadi null. Ini artinya linked list masih kosong saat baru dibuat***.

4. Pada method addFirst(), apa maksud dari kode berikut?

```
if (isEmpty()) {
    head = tail = newNode;
```

   Jawaban: ***Jika linked list masih kosong (belum ada data), maka node baru akan menjadi head dan tail sekaligus, karena dia satu-satunya node di list.***

5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?

   Jawaban: ***Artinya node lama yang sebelumnya jadi head, sekarang punya node baru di depannya. Maka kita hubungkan node lama ke node baru lewat prev.***

6. Modifikasi code pada fungsi print() agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.

   Jawaban:

```
public void print() {
        if (isEmpty()) {
            System.out.println("Linked list masih kosong!");
            return;
        }
        Node17 current = head;
        while (current != null) {
            current.data.tampil();
```

7. Pada insertAfter(), apa maksud dari kode berikut ?
   current.next.prev = newNode;

   Jawaban: ***Maksudnya adalah: kita mengatur supaya node yang sebelumnya berada setelah current, sekarang prev-nya menunjuk ke node baru (newNode). Ini penting supaya hubungan dua arah tetap terjaga***.

8. Modifikasi menu pilihan dan switch-case agar fungsi insertAfter() masuk ke dalam menu pilihan dan dapat berjalan dengan baik.

   Jawaban:

```
do {
    System.out.println("\nMenu Double Linked List Mahasiswa");
    System.out.println("1. Tambah di awal");
    System.out.println("2. Tambah di akhir");
    System.out.println("3. Hapus di awal");
    System.out.println("4. Hapus di akhir");
    System.out.println("5. Tampilkan data");
    System.out.println("6. Cari Mahasiswa berdasarkan NIM");
    System.out.println("7. Tambah data setelah NIM tertentu");
    System.out.println("0. Keluar");
    System.out.print("Pilih menu: ");
    pilihan = sc.nextInt();
    sc.nextLine();

    switch (pilihan) {
        // ... case 1-6 ...
        case 7:
            System.out.print("Masukkan NIM setelah mana data akan ditambahkan: ");
            String nimPatokan = sc.nextLine();
            Mahasiswa17 mhsBaru = inputMahasiswa();
            boolean berhasil = list.insertAfter(nimPatokan, mhsBaru);
            if (berhasil) {
                System.out.println("Data berhasil ditambahkan setelah NIM " + nimPatokan);
            } else {
                System.out.println("NIM patokan tidak ditemukan!");
            }
            break;
        // ... case 0, default ...
    }
} while (pilihan != 0);
```

12.3.1 Tahapan Percobaan

Class DoubleLinkList17

```
public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat dihapus!");
        } else if (size == 1) {
            head = null;
            size--;
        } else {
            head = head.next;
            head.prev = null;
        }
    }
public void removeLast() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat dihapus!");
            return;
        }
        if (head == tail) {
            head = tail = null;
            size--;
        } else {
            tail = tail.prev;
            tail.next = null;
            size--;
        }
    }
```

12.3.3 Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method removeFirst()?

   head = head.next;
   head.prev = null;

   Jawaban:

   head = head.next; artinya menggeser posisi head ke node berikutnya, karena node pertama akan dihapus.

   head.prev = null; artinya memutus hubungan node baru dengan node sebelumnya, supaya node yang lama benar-benar terputus dari linked list.

Jadi, dua baris ini fungsinya untuk menghapus node pertama dari linked list dan mengatur agar node yang baru jadi head tidak lagi punya prev (penunjuk ke node yang sudah dihapus).

2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data
yang terhapus adalah ... “

Jawaban:

```
public void removeFirst() {
    if (isEmpty()) {
        System.out.println("Linked list masih kosong, tidak ada data yang bisa dihapus.");
    } else {
        Mahasiswa01 dataTerhapus = head.data;
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah:");
        System.out.println(dataTerhapus);
    }
}
```

TUGAS

```
package Jobsheet12;
public class DoubleLinkedLists17 {
    Node17 head;
    Node17 tail;
    int size;

    public DoubleLinkedLists17() {
        head = null;
        tail = null;
        size = 0;
    }

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

    public void addFirst(Mahasiswa17 item) {
        Node17 newNode = new Node17(item);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

    public void addLast(Mahasiswa17 item) {
        Node17 newNode = new Node17(item);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
        size++;
    }

    public void add(int index, Mahasiswa17 data) {
        if (index < 0 || index > size) {
            System.out.println("Indeks di luar batas!");
            return;
        }
        if (index == 0) {
            addFirst(data);
        } else if (index == size) {
            addLast(data);
        } else {
```

```
public void removeFirst() {
        if (isEmpty()) {
            System.out.println("Linked list masih kosong, tidak ada data yang bisa
dihapus.");
        } else {
            Mahasiswa17 dataTerhapus = head.data;
            if (head == tail) {
                head = tail = null;
            } else {
                head = head.next;
                head.prev = null;
            }
            System.out.println("Data sudah berhasil dihapus. Data yang terhapus adalah:");
            System.out.println(dataTerhapus);
            size--;
        }
    }

    public void removeLast() {
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat dihapus!");
            return;
        }
        if (head == tail) {
            head = tail = null;
            size--;
        } else {
            tail = tail.prev;
            tail.next = null;
            size--;
        }
    }

    public void removeAfter(String keyNim) {
        Node17 current = head;
        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }
        if (current == null || current.next == null) {
            System.out.println("Node setelah NIM " + keyNim + " tidak ditemukan atau tidak
ada.");
            return;
        }
        Node17 toDelete = current.next;
        if (toDelete == tail) {
            tail = current;
            current.next = null;
        } else {
            current.next = toDelete.next;
            toDelete.next.prev = current;
        }
        size--;
        System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus.");
    }

    public void remove(int index) {
        if (index < 0 || index >= size) {
            System.out.println("Indeks di luar batas!");
            return;
        }
        if (index == 0) {
            removeFirst();
        } else if (index == size - 1) {
            removeLast();
        } else {
            Node17 current = head;
            for (int i = 0; i < index; i++) {
                current = current.next;
            }
            current.prev.next = current.next;
            current.next.prev = current.prev;
            size--;
```

```
public void print() {
        if (isEmpty()) {
            System.out.println("Linked list masih kosong!");
            return;
        }
        Node17 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

    public Node17 search(String nim) {
        Node17 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }

    public void insertAfter(String keyNim, Mahasiswa17 data) {
        Node17 current = head;
        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }
        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + " tidak ditemukan.");
            return;
        }
        Node17 newNode = new Node17(data);
        if (current == tail) {
            current.next = newNode;
            newNode.prev = current;
            tail = newNode;
        } else {
            newNode.next = current.next;
            newNode.prev = current;
            current.next.prev = newNode;
            current.next = newNode;
        }
        System.out.println("Node berhasil disisipkan setelah NIM " + keyNim);
        size++;
    }

    public void getFirst() {
        if (isEmpty()) {
            System.out.println("Linked list kosong!");
        } else {
            head.data.tampil();
        }
    }

    public void getLast() {
        if (isEmpty()) {
            System.out.println("Linked list kosong!");
        } else {
            tail.data.tampil();
        }
    }

    public void getIndex(int index) {
        if (index < 0 || index >= size) {
            System.out.println("Indeks di luar batas!");
            return;
        }
        Node17 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        current.data.tampil();
    }

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

=