LAPORAN HASIL PRAKTIKUM

Algoritma dan Struktur Data

Jobsheet 12

Alexsa Fitria Ayu Siswoyo.

244107020020

1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

Percobaan 1 :

Class Mahasiswa02

```
package jobsheet12;

public class Mahasiswa02 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa02(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 Node02

```
package jobsheet12;

public class Node02 {
    Mahasiswa02 data;
    Node02 prev;
    Node02 next;

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

Class DoubleLinkedList02

```
package jobsheet12;

public class DoubleLinkedList02 {
   Node02 head;
   Node02 tail;

   public DoubleLinkedList02() {
      head = null;
      tail = null;
   }

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

   public void addFirst(Mahasiswa02 data) {
      Node02 newNode = new Node02(data);
      if (isEmpty()) {
         head = tail = newNode;
      } else {
         newNode.next = head;
         head.prev = newNode;
         head = newNode;
      }
   }

   public void addLast(Mahasiswa02 data) {
      Node02 newNode = new Node02(data);
      if (isEmpty()) {
         head = tail = newNode;
      } else {
         tail.next = newNode;
         newNode.prev = tail;
         tail = newNode;
      }
   }
```

```
    public void insertAfter(String keyNim, Mahasiswa02 data) {
        Node02 current = head;

        // Cari node dengan NIM = keyNim
        while (current != null &&
!current.data.nim.equals(keyNim)) {
            current = current.next;
        }

        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + "
tidak ditemukan.");
            return;
        }

        Node02 newNode = new Node02(data);

        // Jika current adalah tail, cukup tambahkan di akhir
        if (current == tail) {
            current.next = newNode;
            newNode.prev = current;
            tail = newNode;
        } else {
            // Sisipkan di tengah
            newNode.next = current.next;
            newNode.prev = current;
            current.next.prev = newNode;
            current.next = newNode;
        }
        System.out.println("Node berhasil disisipkan setelah NIM
" + keyNim);
    }
    public void print() {
        Node02 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }
}
```

Class DoubleLinkedListMain02

```
package jobsheet12;

import java.util.Scanner;

public class DoubleLinkedListMain02 {
    public static void main(String[] args) {
        DoubleLinkedList02 list = new DoubleLinkedList02();
        Scanner scan = new Scanner(System.in);

        int pilihan;
        do {
            System.out.println("====== Menu Double Linked List
Mahasiswa ======");
            System.out.println("1. Tambah di awal");
            System.out.println("2. Tambah di akhir");
            System.out.println("3. Hapus dari awal");
            System.out.println("4. Hapus dari akhir");
            System.out.println("5. Tampilkan data");
            System.out.println("6. Cari Mahasiswa berdasarkan
NIM");
            System.out.println("7. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = scan.nextInt();
            scan.nextLine(); // consume newline

            switch (pilihan) {
                case 1:
                    Mahasiswa02 mhs = inputMahasiswa(scan);
                    list.addFirst(mhs);
                    break;
                case 2:
                    Mahasiswa02 mhs2 = inputMahasiswa(scan);
                    list.addLast(mhs2);
                    break;
                case 3:
                    // list.removeFirst();
                    break;
                case 4:
                    // list.removeLast();
                case 5:
                    list.print();
                    break;
```

```
                case 6:
                    System.out.print("Masukkan NIM yang dicari:
");
                    String nim = scan.nextLine();
                    // Mahasiswa02 found = list.search(nim);
                    // if (found != null) {
                    //     System.out.println("Data Mahasiswa
ditemukan: ");
                    //     found.data.tampil();
                    // } else {
                    //     System.out.println("Data Mahasiswa
tidak ditemukan.");
                    // }
                    break;
                case 7:
                    System.out.println("Keluar dari program.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 7);

        scan.close();
    }

    public static Mahasiswa02 inputMahasiswa(Scanner scan) {
        System.out.print("Masukkan NIM: ");
        String nim = scan.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = scan.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = scan.nextLine();
        System.out.print("Masukkan IPK: ");
        double ipk = scan.nextDouble();
        scan.nextLine(); // consume newline
        return new Mahasiswa02(nim, nama, kelas, ipk);
    }
}
```

Output

```
====== Menu Double Linked List Mahasiswa ======
1. Tambah di awal
2. Tambah di akhir
3. Hapus dari awal
4. Hapus dari akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Keluar
Pilih menu: 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4
====== Menu Double Linked List Mahasiswa ======
1. Tambah di awal
2. Tambah di akhir
3. Hapus dari awal
4. Hapus dari akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.0
```

Pertanyaan :

1. Jelaskan perbedaan antara single linked list dengan double linked lists
   Single linked list hanya mempunyai satu *pointer*, yaitu **next**, yang menunjuk ke *node* selanjutnya. Sebaliknya, double linked list memiliki dua *pointer*: **next** yang menunjuk ke *node* berikutnya, dan **prev** yang menunjuk ke *node* sebelumnya.

2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   Atribut **next** berfungsi sebagai *pointer* yang akan menunjuk ke *node* selanjutnya dari *node* yang sedang aktif. Sedangkan, atribut **prev** adalah *pointer* yang menunjuk ke *node* sebelum *node* saat ini.

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

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

   Konstruktor ini berfungsi untuk mengatur status awal *linked list* menjadi kosong. Ini dilakukan dengan menjadikan head dan tail bernilai null, menandakan bahwa belum ada *node* yang ditambahkan ke dalam *list* ketika pertama kali dibuat.

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

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

Kode tersebut mengecek apakah *linked list* saat ini kosong melalui isEmpty(). Jika kondisinya benar (list kosong), maka *node* baru (newNode) akan menjadi *node* pertama dan terakhir sekaligus, sehingga head dan tail akan menunjuk ke *node* yang sama.

5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
Pernyataan head.prev = newNode digunakan untuk meyambungkan node baru ke node pertama yang sebelumnya ada. Setelah newNode ditambahkan di depan, node yang sebelumnya menjadi head sekarang memiliki node sebelumnya (prev) yaitu newNode. Ini menjaga struktur dua arah dari double linked list.

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

```
    public void print() {
        if (isEmpty()) {
            System.out.println("List kosong.");
            return;
        } else {
            Node02 current = head;
            System.out.println("Isi Double Linked List:");
            while (current != null) {
                System.out.println(current.data + " ");
                current = current.next;
            }
            System.out.println();
        }
```

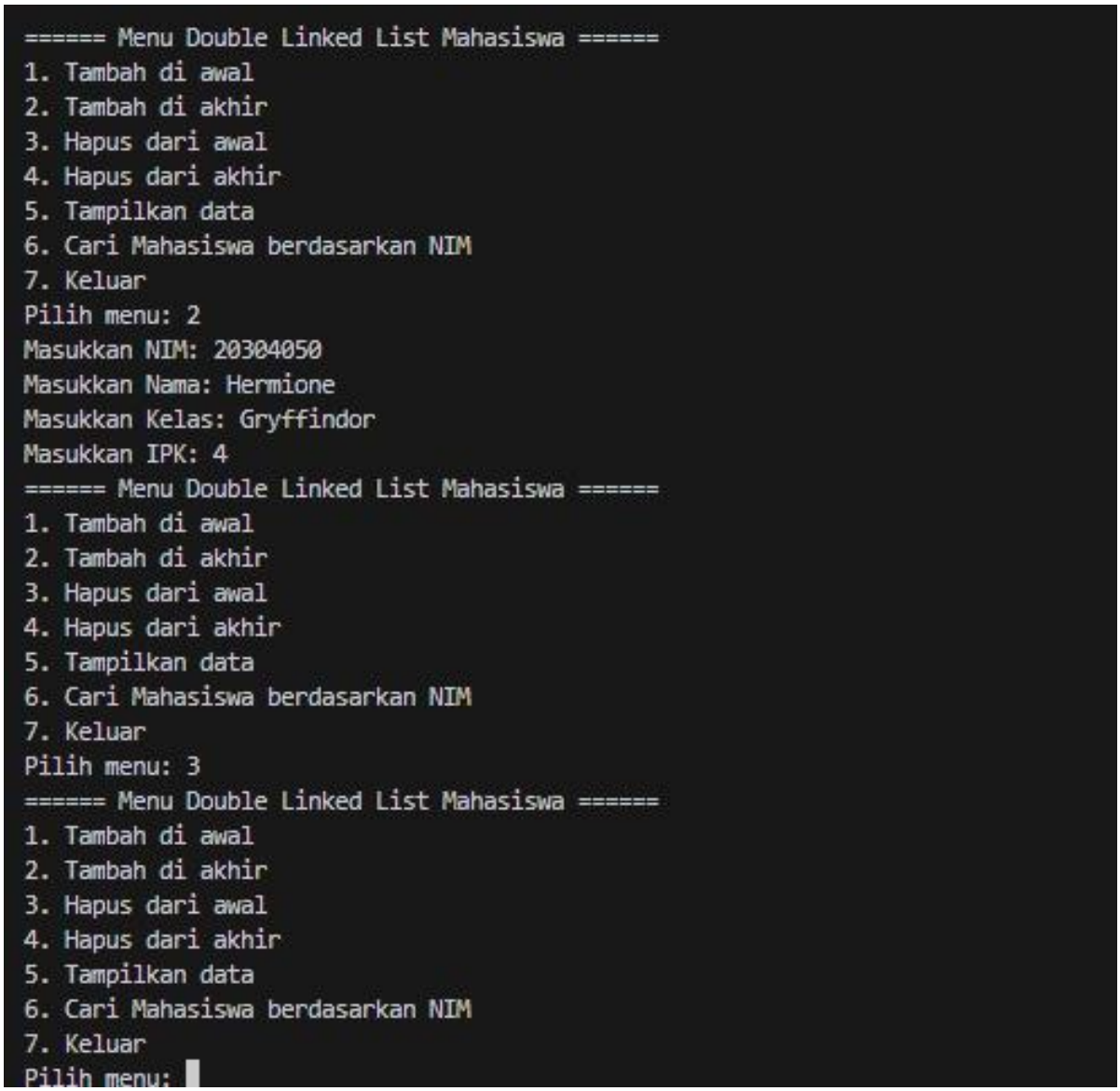
Percobaan 2:

```
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa
dihapus.");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
    }

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

    public Node02 search(String nim) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa
dicari.");
            return null;
        }
        Node02 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                return current;
            }
            current = current.next;
        }
        return null; // Jika tidak ditemukan
    }
```

Outputnya


====== Menu Double Linked List Mahasiswa ======
1. Tambah di awal
2. Tambah di akhir
3. Hapus dari awal
4. Hapus dari akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Keluar
Pilih menu: 2
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4
====== Menu Double Linked List Mahasiswa ======
1. Tambah di awal
2. Tambah di akhir
3. Hapus dari awal
4. Hapus dari akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Keluar
Pilih menu: 3
====== Menu Double Linked List Mahasiswa ======
1. Tambah di awal
2. Tambah di akhir
3. Hapus dari awal
4. Hapus dari akhir
5. Tampilkan data
6. Cari Mahasiswa berdasarkan NIM
7. Keluar
Pilih menu:

Pertanyaan

1. Apakah maksud statement berikut pada method removeFirst()?
   `head = head.next;`
   `head.prev = null;`
   `head = head.next;` berarti bahwa *node* yang tadinya berada di posisi kedua (setelah *head* lama) sekarang ditetapkan sebagai *head* yang baru. Kemudian, `head.prev = null;` memutuskan tautan ke *node* sebelumnya dari *head* yang baru. Maka, *node* yang sebelumnya menjadi *head* tidak lagi diacu oleh *list*, dan secara efektif dihapus.

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

```
    public void removeFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa
dihapus.");
        } else {
            Mahasiswa02 dataDihapus = head.data; //
Mengganti Mahasiswa04 menjadi Mahasiswa02
            if (head == tail) {
                head = tail = null; // Jika hanya ada satu
elemen
            } else {
                head = head.next; // Pindahkan head ke node
berikutnya
                head.prev = null; // Set prev dari head baru
ke null
            }
            System.out.println("Data sudah berhasil dihapus.
Data yang terhapus adalah: ");
            dataDihapus.tampil(); // Tampilkan data yang
dihapus
        }
    }
```

Tugas praktikum

1. Tambahkan fungsi add() pada kelas DoubleLinkedList untuk menambahkan node pada indeks tertentu

```
    public void add(int index, Mahasiswa02 data) {
        if (index < 0 || index > size) {
            System.out.println("Indeks salah, tidak bisa
menambahkan data.");
            return;
        }
        if (index == 0) {
            addFirst(data);
        } else if (index == size) {
            addLast(data);
        } else {
            Node02 newNode = new Node02(data);
            Node02 current = head;
            for (int i = 0; i < index - 1; i++) {
                current = current.next;
            }
            newNode.next = current.next;
            current.next.prev = newNode;
            current.next = newNode;
            newNode.prev = current;
            size++;
        }
    }
```

2. Tambahkan removeAfter() pada kelas DoubleLinkedList untuk menghapus node setelah data key.

```
    public void removeAfter(String keyNim) {
        Node02 current = search(keyNim);
        if (current == null || current.next == null) {
            System.out.println("Tidak ada node setelah NIM "
+ keyNim);
            return;
        }
        Node02 toRemove = current.next;
        if (toRemove == tail) {
            tail = current;
            current.next = null;
        } else {
            current.next = toRemove.next;
            toRemove.next.prev = current;
        }
        size--;
        System.out.println("Data setelah NIM " + keyNim + "
berhasil dihapus:");
        toRemove.data.tampil();
    }
```

3. Tambahkan fungsi remove() pada kelas DoubleLinkedList untuk menghapus node pada indeks tertentu.

```
    public void remove(int index) {
        if (index < 0 || index >= size) {
            System.out.println("Indeks tidak valid.");
            return;
        }
        if (index == 0) {
            removeFirst();
        } else if (index == size - 1) {
            removeLast();
        } else {
            Node02 current = head;
            for (int i = 0; i < index; i++) {
                current = current.next;
            }
            current.prev.next = current.next;
            current.next.prev = current.prev;
            size--;
            System.out.println("Data berhasil dihapus pada
indeks " + index + ":");
            current.data.tampil();
        }
    }
```

4. Tambahkan fungsi getFirst(), getLast() dan getIndex() untuk menampilkan data pada node head, node tail dan node pada indeks tertentu.

```
    // getFirst()
    public Mahasiswa02 getFirst() {
        if (isEmpty()) {
            return null;
        }
        return head.data;
    }
    // getLast() (untuk node tail):
    public Mahasiswa02 getLast() {
        if (isEmpty()) {
            return null;
        }
        return tail.data;
    }
    // get() (untuk node pada indeks tertentu):
    public Mahasiswa02 get(int index) {
        if (index < 0 || index >= size) {
            return null;
        }
        Node02 current = head;
        for (int i = 0; i < index; i++) {
            current = current.next;
        }
        return current.data;
    }
```

5. tambahkan kode program dan fungsi agar dapat membaca size/ jumlah data pada Double Linked List

```
public class DoubleLinkedList02 {
    Node02 head;
    Node02 tail;
    int size;

    public DoubleLinkedList02() {
        head = null;
        tail = null;
        size = 0;
    }
```

```
    public class DoubleLinkedLists02 {
        Node02 head;
        Node02 tail;
        int size;
    public DoubleLinkedLists02() {
        head = null;
        tail = null;
        size = 0;
    }

    // Fungsi size() untuk mengembalikan nilai size:
    public int size() {
        return size;
    }
}
```