Laporan Hasil Praktikum
Algoritma dan Struktur Data
Jobsheet 12 : Double Linked List

Muhammad Firman Aditiasmara

244107020094

TI-1E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

Politeknik Negeri Malang

2025

1. **Tujuan Praktikum**

   Setelah melakukan materi praktikum ini, mahasiswa mampu:

   1. Memahami algoritma double linked lists
   2. Membuat dan mendeklarasikan struktur algoritma double linked lists
   3. Menerapkan algoritma double linked lists dalam beberapa study case

2. **Praktikum**

   2.1 Percobaan 1

   2.1.1 Langkah-langkah Percobaan

   1. Buat class di dalam paket tersebut dengan nama Mahasiswa18. Di dalam class tersebut,deklarasikan atribut sesuai dengan diagram class di atas. Tambahkan juga konstruktordan method

```
package Jobsheet12;

public class Mahasiswa18 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa18(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);
    }
}
```

   2. Membuat class Node18 dan menambahkan method dan konstruktor

```
package Jobsheet12;

public class Node18 {
    Mahasiswa18 data;
    Node18 prev;
    Node18 next;

    public Node18(Mahasiswa18 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

3. Membuat class DoubleLinkedList. Dan membuat method yang diperlukan

```
package Jobsheet12;

public class DoubleLinkedList18 {
    Node18 head;
    Node18 tail;

    public DoubleLinkedList18() {
        head = null;
        tail = null;
    }

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

    public void addFirst(Mahasiswa18 data) {
        Node18 newNode = new Node18(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    public void addLast(Mahasiswa18 data) {
        Node18 newNode = new Node18(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

      public void insertAfter(String keyNim, Mahasiswa18 data) {
        Node18 current = head;

        while (current != null &&
!current.data.nim.equals(keyNim)) {
            current = current.next;
        }

        if (current == null) {
            System.out.println("Node dengan NIM " + keyNim + "
tidak ditemukn.");
            return;
        }
   Node18 newNode = new Node18(data);

        if (current == tail) {
            current.next = newNode;
            newNode.prev = current;
            current.next.prev = newNode;
            current.next = newNode;
        }
```

```
  System.out.println("Node berhasil disisipkan setelah NIM " +
keyNim);
    }

    public void print() {
        Node18 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }
```

4. Membuat class DLLMain untuk mengeksekusi semua method

```
package Jobsheet12;

import java.util.Scanner;

public class DLLMain18 {

  public static void main(String[] args) {
        DoubleLinkedList18 list = new DoubleLinkedList18();
        Scanner input = new Scanner(System.in);
        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 = input.nextInt();
            input.nextLine();

            switch (pilihan) {
                case 1 -> {
                    Mahasiswa18 mhs = inputMahasiswa(input);
                    list.addFirst(mhs);
                }
                case 2 -> {
                    Mahasiswa18 mhs = inputMahasiswa(input);
                    list.addLast(mhs);
                }
                case 3 -> list.removeFirst();
                case 4 -> list.removeLast();
                case 5 -> list.print();
                case 6 -> {
                    System.out.print("Masukkan NIM yang dicari:
");
```

```
 String nim = input.nextLine();
                    Node18 found = list.search(nim);
                    if (found != null) {
                        System.out.println("Data
ditemukan:");
                        found.data.tampil();
                    } else {
                        System.out.println("Data tidak
ditemukan.");
                    }
                }
                case 0 -> System.out.println("Keluar
dari program.");
                default -> System.out.println("Pilihan
tidak valid!");
            }

        } while (pilihan != 0);
        input.close();
    }
}
```

5. Tambahan method inputMahasiswa

```
public static Mahasiswa18 inputMahasiswa(Scanner input)
{
        System.out.print("Masukkan NIM: ");
        String nim = input.nextLine();
        System.out.print("Masukkan Nama: ");
        String nama = input.nextLine();
        System.out.print("Masukkan Kelas: ");
        String kelas = input.nextLine();
        System.out.print("Masukkan IPK: ");
        double IPK = input.nextDouble();
        Mahasiswa18 mhs = new Mahasiswa18(nim, nama,
kelas, IPK);

        return mhs;
    }
```

2.1.2 Verifikasi Hasil Percobaan

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu: 1
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4,0

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

2.1.3 Pertanyaan

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

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

   Atribut next dan prev dalam class Node18 digunakan untuk membentuk struktur double linked list. Untuk atribut next digunakan untuk menunjuk ke node berikutnya, sedangkan atribut prev digunakan untuk menunjuk ke node sebelumnya.

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

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

   Jawab :
   untuk menginisialisasi linked list dalam keadaan kosong saat objek DoubleLinkedList01 dibuat. Konstruktor ini digunakan untuk mempersiapkan struktur double linked list agar siap digunakan dalam keadaan awal yang konsisten (kosong).

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

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

Jawab :
Baris kode ini memastikan bahwa jika node masih kosong, maka node baru langsumg menjadi head dan tail.

5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
Jawab:
head.prev = newNode; ini digunakan unutk mengubah head dengan mengarahkan node head, ke node baru yang baru saja ditambahkan tepat sebelum head.

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

```
    public void print() {
        // modifikasi
        if (head == null) {
            System.out.println("Linked list kosong.");
            return;
        }

        Node18 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
        // modifikasi
        System.out.println("Warning: Linked list masih
berisi data.");
    }
```

7. Pada insertAfter(), apa maksud dari kode berikut ? current.next.prev = newNode;
Jawab:
Kode current.next.prev = newNode; memastikan node setelah current pointer prev ke node baru yang disisipkan, agar linked list tetap terhubung dengan benar secara dua arah.

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

- Modifikasi pada pilihan menu

```
 do {
            System.out.println();
            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. Sisipkan data"); // modifikasi
```

Modifikasi pada menu 7

- Modifikasi pada switch-case

```
  // modifikasi
                case 7 -> {
                    System.out.println("Penyisipan data
diletakkan setelah data dengan NIM yang anda masukkan");
                    System.out.print("Masukkan NIM yang dicari:
");
                    String keyNim = input.nextLine();

                    System.out.println("Masukkan data: ");
                    Mahasiswa18 newData = inputMahasiswa(input);
                    list.insertAfter(keyNim, newData);
                }
```

Hasil compile

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan data
0. Keluar
Pilih Menu: 1
Masukkan NIM: 23411
Masukkan Nama: Adid
Masukkan Kelas: 1A
Masukkan IPK: 4,0

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan data
0. Keluar
Pilih Menu: 2
Masukkan NIM: 2111
Masukkan Nama: Fara
Masukkan Kelas: 1D
Masukkan IPK: 3,99

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan data
0. Keluar
Pilih Menu: 7
Penyisipan data diletakkan setelah data dengan NIM yang anda masukkan
Masukkan NIM yang dicari: 23411
Masukkan data:
Masukkan NIM: 1234
Masukkan Nama: Mara
Masukkan Kelas: 1F
Masukkan IPK: 3,98
Node berhasil disisipkan setelah NIM 23411
```

```
Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
7. Sisipkan data
0. Keluar
Pilih Menu: 5
NIM: 23411, Nama: Adid, Kelas: 1A, IPK: 4.0
NIM: 1234, Nama: Mara, Kelas: 1F, IPK: 3.98
NIM: 2111, Nama: Fara, Kelas: 1D, IPK: 3.99
```

2.2 Percobaan 2: : Modifikasi Elemen Pada Single Linked List

2.2.1 Langkah-langkah Percobaan

1. Membuat method removeFirst

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

2. Membuat method removeLast

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

2.2.2 Verifikasi Hasil Percobaan

```
e\User\workspaceStorage\cc2ca254f290391a
18 "

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu: 2
Masukkan NIM: 20304050
Masukkan Nama: Hermione
Masukkan Kelas: Gryffindor
Masukkan IPK: 4,0

Menu Double Linked List Mahasiswa
1. Tambah di awal
2. Tambah di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih Menu: 3
```

2.2.3 Pertanyaan

1. Apakah maksud statement berikut pada method removeFirst()?
   head = head.next;
   head.prev = null;
   Jawab :
   Head = head.next digunakan untuk memindahkan head ke node selanjutnya, karena node head sebelumnya dihapus, sedangkan head.prev = null digunakan untuk memutuskan dengan head sebelunya karena sudah dihapus, sehingga referensi node sebelumnya diisi null

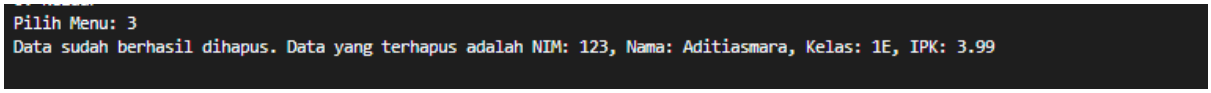
2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah … “

```
public void removeFirst() {

        Mahasiswa18 hapus = head.data;

        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat
dihapus!");
            return;
        }
        if (head == tail) {
            head = tail = null;
        } else {
            head = head.next;
            head.prev = null;
        }
        System.out.print("Data sudah berhasil dihapus. Data yang
terhapus adalah ");
        hapus.tampil();
    }
```

Hasil compile



2.3 Tugas

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

```
// Tugas
    public void add(int index, Mahasiswa18 data) {
        if (index < 0) {
            System.out.println("Index tidak valid!");
            return;
        }

        Node18 newNode = new Node18(data);

        if (index == 0) {
            addFirst(data);
            return;
        }

        Node18 current = head;
        int currentIndex = 0;

        while (current != null && currentIndex < index) {
            current = current.next;
            currentIndex++;
        }
        if (current == null) {
            addLast(data);
        } else {
            Node18 prevNode = current.prev;
            prevNode.next = newNode;
            newNode.prev = prevNode;
            newNode.next = current;
            current.prev = newNode;

            System.out.println("Data berhasil ditambahkan pada
indeks ke-" + index);
        }
    }
```

Hasil

```
Pilih Menu: 8
Masukkan indeks: 1
Masukkan NIM: 1441
Masukkan Nama: Bagas
Masukkan Kelas: 1B
Masukkan IPK: 4,0
Data berhasil ditambahkan pada indeks ke-1
```

```
Pilih Menu: 5
NIM: 123, Nama: Aditiasmara, Kelas: 1E, IPK: 3.99
NIM: 1441, Nama: Bagas, Kelas: 1B, IPK: 4.0
NIM: 111, Nama: Fara, Kelas: 1D, IPK: 4.0
NIM: 1234, Nama: Mara, Kelas: 1D, IPK: 4.0
```

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

```
 public void removeAfter(String keyNim) {

        Node18 current = head;
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada yang bisa
dihapus");
            return;
        }

        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;

        }

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

        if (current.next == null) {
            System.out.println("Data berada pada posisi terakhir,
tidak ada data selanjutnya untuk dihapus");
        }

        Node18 hapus = current.next;

        if (hapus == tail) {
            current.next = null;
            tail = current;
        } else {
            current.next = hapus.next;
            hapus.next.prev = current;
        }

        System.out.print("Data berhasil terhapus, data yang terhapus
adalah ");
        hapus.data.tampil();
    }
```

Hasil

```
Pilih Menu: 9
Masukkan acuan NIM: 123
Data berhasil terhapus, data yang terhapus adalah NIM: 1441, Nama: Bagas, Kelas: 1B, IPK: 4.0
```

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

```
 // Tugas
    public void remove(int index) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data yang bisa
dihapus.");
            return;
        }

        if (index < 0) {
            System.out.println("Indeks tidak valid!");
            return;
        }
        Node18 current = head;
        int currentIndex = 0;

        while (current != null && currentIndex < index) {
            current = current.next;
            currentIndex++;
        }
        if (current == null) {
            System.out.println("Index melebih jumlah elemen pada
list");
            return;
        }

        Mahasiswa18 hapus = current.data;

        if (head == tail) {
            head = tail = null;
        }

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

        else if (current == tail) {
            tail = tail.prev;
            tail.next = null;
        } else {
            current.prev.next = current.prev;
            current.next.prev = current.prev;
        }
        System.out.print("Data berhasil dihapus. Data yang terhapus
adalah ");
        hapus.tampil();

    }
```

Hasil

```
10. Hapus data dengan index
11. Tampil data awal
12. Tampil data akhir
13. Tampil data sesuai index
14. Tampil jumlah data
0. Keluar
Pilih Menu: 10
Masukkan indeks: 2
Data berhasil dihapus. Data yang terhapus adalah NIM: 1234, Nama: Mara, Kelas: 1D, IPK: 4.0
```

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

```
    // Tugas
    public void getFirst() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data untuk
ditampilkan");
        } else {
            System.out.print("Data pertama: ");
            head.data.tampil();
        }
    }

    // Tugas
    public void getLast() {
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data untuk
ditampilkan");
        } else {
            System.out.print("Data terakhir: ");
            tail.data.tampil();
        }
    }
    // Tugas
    public void getIndex(int index) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data untuk
ditampilkan");
        }
        if (index < 0) {
            System.out.println("Indeks tidak valid.");
            return;
        }
        Node18 current = head;
        int currentIndex = 0;

        while (current != null && currentIndex < index) {
            current = current.next;
            currentIndex++;
        }
   if (current == null) {
            System.out.println("Indeks melebihi batas.");
        } else {
            System.out.print("Data berada pada indeks ke-" + index + ":
");
            current.data.tampil();
        }
    }
```

Hasil

```
0. Keluar
Pilih Menu: 5
NIM: 123, Nama: Aditiasmara, Kelas: 1E, IPK: 3.99
NIM: 111, Nama: Fara, Kelas: 1D, IPK: 4.0
NIM: 1456, Nama: Budi, Kelas: 1C, IPK: 3.78
NIM: 231, Nama: Rama, Kelas: 2B, IPK: 3.56
Warning: Linked list masih berisi data.
```

- Ketika dipilih menu tampil awal

```
Pilih Menu: 11
Data pertama: NIM: 123, Nama: Aditiasmara, Kelas: 1E, IPK: 3.99
```

- Ketika dipilih menu tampil akhir

```
Pilih Menu: 12
Data terakhir: NIM: 231, Nama: Rama, Kelas: 2B, IPK: 3.56
```

- Ketika dipilih menu tampil data sesuai indeks

```
Masukkan indeks: 2
Data berada pada indeks ke-2: NIM: 1456, Nama: Budi, Kelas: 1C, IPK: 3.78
```

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

```
public int size() {
      int count = 0;
      Node18 current = head;

      while (current != null) {
          count++;
          current = current.next;
      }
      return count;

  }
```

Hasil

```
0. Keluar
Pilih Menu: 5
NIM: 123, Nama: Aditiasmara, Kelas: 1E, IPK: 3.99
NIM: 111, Nama: Fara, Kelas: 1D, IPK: 4.0
NIM: 1456, Nama: Budi, Kelas: 1C, IPK: 3.78
NIM: 231, Nama: Rama, Kelas: 2B, IPK: 3.56
Warning: Linked list masih berisi data.
```

```
14. Tampil jumlah data
0. Keluar
Pilih Menu: 14
Jumlah data: 4
```