LAPORAN HASIL PRATIKUM

ANALISIS DAN STRUKTUR DATA

Jobsheet Double Linked List

Raihan Akbar Putra Prasetyo/244107020087

Kelas: TI-1E

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

PRAKTIKUM 25

Percobaan 1

Code program:

Mahasiswa24.java

```
package jobsheet12;

public class Mahasiswa24 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa24(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);
    }
}
```

Node24.java

```
package jobsheet12;
public class Node24 {
    Mahasiswa24 data;
    Node24 prev;
    Node24 next;

    public Node24(Mahasiswa24 data){
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

DoubleLinkedList24.java

```
package jobsheet12;

public class DoubleLinkedList24 {
    Node24 head;
    Node24 tail;

    public DoubleLinkedList24() {
        head = null;
        tail = null;
    }

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

    public void addFirst(Mahasiswa24 data) {
        Node24 newNode = new Node24(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    public void addLast(Mahasiswa24 data) {
        Node24 newNode = new Node24(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    public void insertAfter(String keyNim, Mahasiswa24 data) {
        Node24 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;
        }

        Node24 newNode = new Node24(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);
    }
```

```
public void print() {
     Node24 current = head;
     while (current != null) {
         current.data.tampil();
         current = current.next;
     }
 }
```

DLLMain24.java

```
package jobsheet12;

import java.util.Scanner;

public class DLLMain24 {
    static Scanner sc = new Scanner(System.in);

    public static Mahasiswa24 inputMahasiswa() {
        System.out.print("Nim : ");
        String nim = sc.nextLine();
        System.out.print("Nama : ");
        String nama = sc.nextLine();
        System.out.print("Kelas : ");
        String kelas = sc.nextLine();
        System.out.print("IPK : ");
        double ipk = sc.nextDouble();
        sc.nextLine();
        return new Mahasiswa24(nim, nama, kelas, ipk);
    }

    public static void main(String[] args) {
        DoubleLinkedList24 list = new DoubleLinkedList24();
        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: {
                    Mahasiswa24 mhs = inputMahasiswa();
                    list.addFirst(mhs);
                    break;
                }
                case 2: {
                    Mahasiswa24 mhs = inputMahasiswa();
                    list.addLast(mhs);
                    break;
                }
                case 3:
                    list.removeFirst();
                    break;
                case 4:
                    list.removeLast();
                    break;
                case 5:
                    list.print();
                    break;
                case 6: {
                    System.out.print("Masukkan NIM yang dicari: ");
                    String nim = sc.nextLine();
                    Node24 found = list.search(nim);
                    if (found != null) {
                        System.out.println("Data ditemukan:");
                        found.data.tampil();
                    } else {
                        System.out.println("Data tidak ditemukan");
                    }
                    break;
                }
```

```
                    case 0:
                    System.out.println("Keluar dari program.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 0);
    }
}
```

Output :

```
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
Nim : 20304050
Nama : Hermione
Kelas : Gryffindor
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
```

Pertanyaan:

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

Perbedaannya adalah double linked list bisa menyimpan Node sebelumnya / prev dan sesudahnya /next sedangkan single linked list hanya bisa next saja

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

Untuk mengakses setiap node sebelum dan sesudahnya

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

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

Head sebagai penunjuk awalan dan tail sebagai penunjuk akhir dari data

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

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

Jika masih dalam keadaan kosong maka jika input data baru, data tersebut menjadi head dan tail dikarenakan data nya masih 1

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

digunakan untuk menghubungkan node lama (yang tadinya head) ke node baru sebagai node sebelumnya/prev

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

```
public void print() {
        Node24 current = head;
       if (isEmpty()) {
            System.out.println("Data Kosong!");
            return;
        }else{
        while (current != null) {
            current.data.tampil();
            current = current.next;
            }
        }
    }
```

7. Pada insertAfter(), apa maksud dari kode berikut ?

current.next.prev = newNode;
Untuk menyambungkan data ke new node

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

Modifikasi di main:

```
                    case 7: {
                    System.out.print("Masukkan NIM mahasiswa yang ingin
dicari : ");
                    String nim = sc.nextLine();
                    Node24 found = list.search(nim);
                    if (found != null) {
                        Mahasiswa24 mhs = inputMahasiswa();
                        list.insertAfter(nim, mhs);
                    } else {
                        System.out.println("Data not found");
                    }
                    break;
                }
```

Modifikasi di Doublelinkedlist24.java

```
public void insertAfter (String keyNim, Mahasiswa24 data){
        Node24 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;
        }
        Node24 newNode = new Node24(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);
    }
```

Percobaan 2

Removefirst dan removeLast

```
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;
        }
    }

    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;
        }
    }
```

Output :

```
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
Nim : 20304050
Nama : Hermione
Kelas : Gryffindor
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

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
```

Pertanyaan :

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

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

- **Untuk menghapus / memutus node prev**

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

```
public void removeFirst(){
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat
dihapus!");
        }
        if(head == tail){
            System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah " + head.data.nama);
            head = tail = null;
        }else{
            System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah " + head.data.nama);
            head = head.next;
            head.prev = null;
        }
    }

    public void removeLast(){
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus");
            return;
        }
        if (head == tail) {
            System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah " + tail.data.nama);
            head = tail = null;
        } else {
            System.out.println("Data sudah berhasil dihapus. Data yang terhapus
adalah " + tail.data.nama);
            tail = tail.prev;
            tail.next = null;
        }
    }
```

Tugas:

1. **Fungsi Add**

```
public void add(Mahasiswa24 data, int index) {
        if (index == 0) {
            addFirst(data);
            return;
        }

        Node24 temp = head;
        for (int i = 0; i < index - 1; i++) {
        if (temp == null) {
        System.out.println("Index melebihi panjang list");
        return;
        }
        temp = temp.next;
        }

        if (temp.next == null) {
            addLast(data);
            return;
        }

        Node24 newNode = new Node24(data);
        temp.next.prev = newNode;
        newNode.next = temp.next;
        temp.next = newNode;
        newNode.prev = temp;
    }
```

2. **Removeafter**

```
public void removeAfter(String key) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus");
            return;
        }
        Node24 temp = head;

        while (temp != null && !temp.data.nama.equalsIgnoreCase(key)) {
        temp = temp.next;
        }

        if (temp == null || temp.next == null) {
        System.out.println("Node setelah \"" + key + "\" tidak ditemukan atau tidak
ada");
        return;
        }

        temp.next.prev = temp;
        temp.next = temp.next.next;

    }
```

3. Remove

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

        if (index < 0) {
        System.out.println("Index tidak valid");
        return;
    }

        if (index == 0) {
            removeFirst();
            return;
        }

        Node24 temp = head;

        for(int i = 0; i < index; i++) {
        if (temp == null) {
            System.out.println("Index melebihi panjang list");
            return;
        }
            temp = temp.next;
        }

        if (temp.next == null) {
            removeLast();
            return;
        }

        temp.next.prev = temp.prev;
        temp.prev.next = temp.next;
    }
```

4. getFirst(), getLast() dan getIndex()

```
Mahasiswa24 getFirst() {
        if (isEmpty()) {
            return null;
        }
        return head.data;
    }

    Mahasiswa24 getLast() {
        if (isEmpty()) {
            return null;
        }

        return tail.data;
    }

    Mahasiswa24 getIndex(int index) {
        if (isEmpty()) {
            return null;
        }

        Node24 temp = head;

        for(int i = 0; i < index; i++) {
        if (temp == null) {
            System.out.println("Index melebihi panjang list");
            return null;
        }
            temp = temp.next;
        }
        return temp.data;
    }
```

5. getsize

```
int getSize() {
      int counter = 0;

      Node24 temp = head;
      while (temp != null) {
         temp = temp.next;
         counter++;
      }

      return counter;
   }
```