Laporan Hasil Praktikum

Algoritma dan Struktur Dasar

Jobsheet 12

MUHAMMAD PEARL OCSHADA

244107020064

TI 1-E

Program Studi Teknik Informatika

Jurusan Teknologi Informasi

POLINEMA

2024

Jobsheet 12 : Double Linked List

Praktikum 1 : Percobaan1

a. Source Code Mahasiswa

```
package ALSD_Jobsheet12;
public class Mahasiswa20 {
    String nim;
    String nama;
    String kelas;
    double ipk;

    public Mahasiswa20(String nim, String nama, String kelas,double ipk) {
        this.nim = nim;
        this.kelas = kelas;
        this.nama = nama;
        this.ipk = ipk;
    }
    public void tampil(){
        System.out.println(nim + " " + nama + " "+ kelas +" "+ipk);
    }
}
```

b. Source code node20

```
package ALSD_Jobsheet12;
public class node20 {
    Mahasiswa20 data;
    node20 prev;
    node20 next;
    public node20(Mahasiswa20 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

c. Source code doubleLinkedList

```
package ALSD_Jobsheet12;
public class doubleLinkedList {
    node20 head;
    node20 tail;
```

```
public doubleLinkedList() {
        head = null;
        tail = null;
    }
    public boolean isEmpty(){
        return (head == null);
    }
    public void addFirst(Mahasiswa20 data){
        node20 newNode = new node20(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }
    public void addLast(Mahasiswa20 data){
        node20 newNode = new node20(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }
    public void add(int item, int index){

    }
    public void print(){
        node20 current = head;
        while (current != null) {
            current.data.tampil();
```

```
            current = current.next;
        }
    }
    public void removeFirst(){

    }
    public void removeLast(){

    }
    public void insertAfter(String keyNim, Mahasiswa20 data){
        node20 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;
        }
        node20 newNode = new node20(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 node20 search(String nim) {
        node20 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                System.out.println("Ditemukan:");
                current.data.tampil();
```

```
            current = current.next;
        }
    }
    public void removeFirst(){

    }
    public void removeLast(){

    }
    public void insertAfter(String keyNim, Mahasiswa20 data){
        node20 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;
        }
        node20 newNode = new node20(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 node20 search(String nim) {
        node20 current = head;
        while (current != null) {
            if (current.data.nim.equals(nim)) {
                System.out.println("Ditemukan:");
                current.data.tampil();
                return current;
```

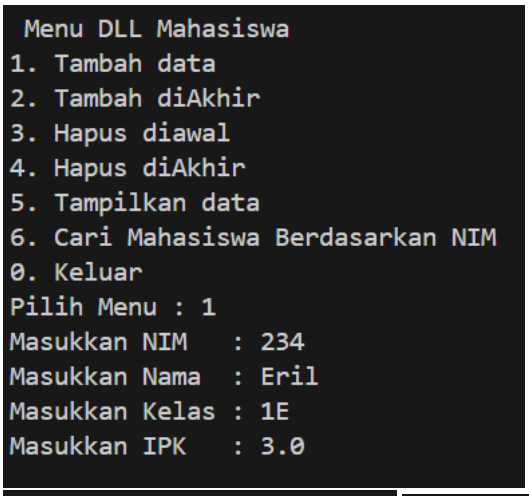
```
            }
            current = current.next;
        }
        System.out.println("Data tidak ditemukan.");
        return null;
    }
}
```

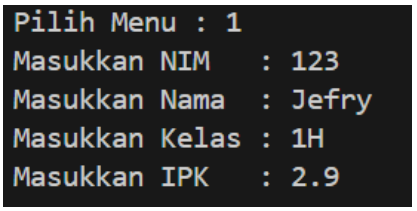
d. Source Code DLLmain

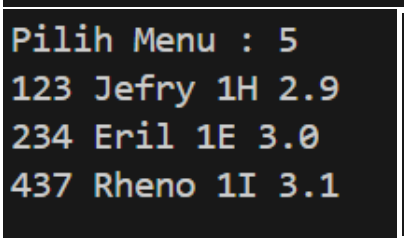
```
package ALSD_Jobsheet12;
import java.util.Scanner;
public class DLLMain {
    public static void main(String[] args) {
        doubleLinkedList list = new doubleLinkedList();
        Scanner sc = new Scanner(System.in);
        int pilihan;
        do {
            System.out.println("\n Menu DLL Mahasiswa");
            System.out.println("1. Tambah data");
            System.out.println("2. Tambah diAkhir");
            System.out.println("3. Hapus diawal");
            System.out.println("4. Hapus diAkhir");
            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:
                    Mahasiswa20 mhs = inputMahasiswa(sc);
                    list.addFirst(mhs);
                    break;
                case 2:
                    mhs = inputMahasiswa(sc);
                    list.addLast(mhs);
                    break;
```

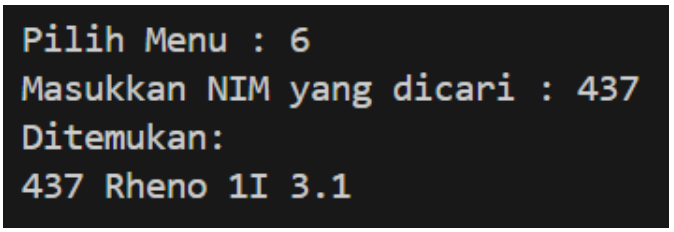
```
                case 3:
                    list.removeFirst();
                    System.out.println("Data awal sudah dihapus!");
                    break;
                case 4:
                    list.removeLast();
                    System.out.println("Data akhir sudah dihapus!");
                    break;
                case 5:
                    list.print();
                    break;
                case 6:
                    System.out.print("Masukkan NIM yang dicari : ");
                    String nim = sc.nextLine();
                    node20 found = list.search(nim);
                    break;
                case 0:
                    System.out.println("Keluar dari Program.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid!");
                    break;
            }
        } while ( pilihan != 0);
    }
    public static Mahasiswa20 inputMahasiswa(Scanner sc){
        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();

        return new Mahasiswa20(nim, nama, kelas, ipk);
    }
}
```









Pertanyaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
   Perbedaan single linkedList dan double linkedList terdapat pada pemakaiannya, pada single hanya terdapat satu pointer(next) yang hanya bisa maju searah, sedangkan pada double terdapat 2 pointer (next & prev) yang bisa maju dan mundur (dua arah), jadi untuk menyisipkan nilai pada method insertAfter lebih ringkas menggunakan double dikarenakan bisa memanfaatkan prevnya tidak hanya next dan Ketika add last bisa langsung menggunakan tail, dan jika mencari mundur(prev) bisa dengan prev dari tail.
2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   pada node20, atribut next dan prev digunakan untuk menghubungkan antar (node) dengan (doubleLinkedList), sedangkan fungsi next dan prev digunakan untuk menunjuk ke node setelahnya atau sebelumnya.
3. Konstruktor pada class doubleLinkedList digunakan untuk menginisialisasi list kosong saat objek dibuat yang artinya tidak ada node pertama & tidak ada node terakhir, Sedangkan konstruktor sendiri berfungsi mengatur kondisi awal dari doubleLinkedList (kosong)
4. Pada method addFirst melakukan pengecekan apakah nilai kosong jika bernilai true, maka langsung akan langsung di taruh di node 1, pada code tersebut newNode ditaruh di tail dan tail ditaruh dihead
5. Arti statement head.prev = newNode adalah untuk menghubungkan node sebelumnya (head) dengan newNode sebagai node sebelum head. Jika statement head.next = newNode maka menghubungakn node setelahnya(head) dengan newNode sebagai node setelah head.

6. Modifikasi Code pada fungsi print()

```
if (isEmpty()) {
            System.out.println(
"Linkedlist masih kosong!");
        }
```

```
1. Tambah data
2. Tambah diAkhir
3. Hapus diawal
4. Hapus diAkhir
5. Tampilkan data
6. Cari Mahasiswa Berdasarkan NIM
0. Keluar
Pilih Menu : 5
Linkedlist masih kosong!
```

7. Pada kode tersebut bermaksud untuk menyisipkan nilai pada node setelah(current), dan harus menunjuk Kembali ke newNode sebagai node sebelumnya (prev)
8. Modifikasi Code menu pilihan dan switch-case untuk fungsi insertAfter, Pada Menu ditambahkan 7. Tambah data setelah data, & setelah case 6 ditambahkan code

```
System.out.println("\n Menu DLL Mahasiswa");
            System.out.println("1. Tambah data");
            System.out.println("2. Tambah diAkhir");
            System.out.println("3. Hapus diawal");
            System.out.println("4. Hapus diAkhir");
            System.out.println("5. Tampilkan data");
            System.out.println("6. Cari Mahasiswa Berdasarkan NIM");
            System.out.println("7. Tambah data setelah data");
            System.out.println("0. Keluar");
            System.out.print("Pilih Menu : ");
            pilihan = sc.nextInt();
            sc.nextLine();
```

```
 case 7:
           System.out.print("Masukkan nim setelah siapa ingin ditambahkan: ");
           String keyNIM = sc.nextLine();
           Mahasiswa20 mhsInsertAfter = inputMahasiswa(sc);
           list.insertAfter(keyNIM, mhsInsertAfter);
       break;
```

```
 Menu DLL Mahasiswa
1. Tambah data
2. Tambah diAkhir
3. Hapus diawal
4. Hapus diAkhir
5. Tampilkan data
6. Cari Mahasiswa Berdasarkan NIM
7. Tambah data setelah data
0. Keluar
Pilih Menu : 1
Masukkan NIM   : 123
Masukkan Nama  : ERIL
Masukkan Kelas : 1E
Masukkan IPK   : 3.0
```

```
Pilih Menu : 2
Masukkan NIM   : 234
Masukkan Nama  : FAUZI
Masukkan Kelas : 1G
Masukkan IPK   : 2.9
```

```
Pilih Menu : 7
Masukkan nim setelah siapa ingin ditambahkan: 123
Masukkan NIM   : 456
Masukkan Nama  : GILANG
Masukkan Kelas : 1H
Masukkan IPK   : 3.1
Node Berhasil DiSisipkan setelah NIM 123
```

```
Pilih Menu : 5
123 ERIL 1E 3.0
456 GILANG 1H 3.1
234 FAUZI 1G 2.9
```

Praktikum 2 : Percobaan2

Pada Percobaan 2 menambahkan fungsi removeFirst dan removeLast yang belum terisi pada class doubleLinkedList

```
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;
        }
    }
```

Pertanyaan

1. Apakah maksud statement berikut pada method removeFirst()?
   head = head.next; menghubungkan head dengan head.next dan berpindah
   head.prev = null; setelah berpindah otomatis head tidak terhubung dan diset null
2. Modifikasi Data

Tugas

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

Menambahkan code pada fungsi add() & menu pada switch-case

```
public void add(Mahasiswa20 data, int index){
        if (index < 0) {
            System.out.println("Indeks tidak valid");
            return;
        }
        if (index == 0) {
            addFirst(data);
            return;
        }
        node20 current = head;
        int i = 0;
        while (current != null && i < index-1) {
            i++;
        }
        if (current == null) {
            System.out.println("Indeks melebihi ukuran list. Menambahkan di
akhir.");
            addLast(data);
            return;
        }
        node20 newNode = new node20(data);
        newNode.next = current.next;
        newNode.prev = current;

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

        current.next = newNode;
        System.out.println("Data Berhasil ditambahkan indexs ke- "+index);
    }
```

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

```
public void removeAfter(String keyNim){Add commentMore actions
        if (isEmpty()) {
            System.out.println("List Kosong");
            return;
        }
        node20 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;
        }
        node20 nodeRm = current.next;
        current.next = nodeRm.next;

        if (nodeRm.next != null) {
            nodeRm.next.prev = current;
        } else {
            tail = current;
        }
        size--;
        System.out.println("Node setelah NIM " + keyNim + " berhasil dihapus.
Data yang dihapus:");
        nodeRm.data.tampil();
    }
```

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

```
public void remove(int index){Add commentMore actions
        if (isEmpty()) {
            System.out.println("List Kosong");
        }
        if (index < 0) {
            System.out.println("Index tidak Valid");
        }
        if (index == 0) {
            System.out.print("Data yang dihapus: ");
            head.data.tampil();
            removeFirst();
        }
        node20 current = head;
        int i = 0;
        while (current != null && i < index) {
            current = current.next;
            i++;
        }
        if (current == null) {
            System.out.println("Index melebihi panjang list");
        }
        System.out.print("data yang dihapus : ");
        current.data.tampil();

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

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

```
public void getFirst (){
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data!");
        } else {
```

```
System.out.print("Data pertama: ");Add commentMore actions
            head.data.tampil();
        }
    }
    public void getLast(){
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data!");
        } else {
            System.out.println("Data Terakhir: ");
            tail.data.tampil();
        }
    }
    public void getIndex(int index){
        if (isEmpty()) {
            System.out.println("List kosong, tidak ada data!");
        }
        if (index < 0) {
            System.out.println("Index tidak valid!");
            return;
        }
        node20 current = head;
        int i = 0;
        while (current != null && i < index) {
            current = current.next;
            i++;
        }
        if (current == null) {
            System.out.println("Indeks melebihi panjang list.");
        } else {
            System.out.print("Data pada indeks ke-" + index + ": ");
            current.data.tampil();
        }
    }
```

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

```
public int getSize(){

        return size;

    }
```