Laporan Jobsheet 12 Algoritma dan Struktur Data

Double Linked List

244107020151

Ghazwan Ababil

Teknik Informatika / TI – E (11)

Politeknik Negeri Malang

Jurusan Teknologi Informasi

2025

1. Praktikum

1.1. Percobaan 1

1. Pada folder Praktikum-ASD membuat folder baru dengan nama Jobsheet12
2. Menambahkan class-class:
   a. Mahasiswa11.java
   b. Node11.java
   c. DoubleLinkedList11.java
   d. DLLMain11.java
3. Mengimplementasikan class Mahasiswa11 sesuai dengan class diagram

```
package Jobsheet12;

public class Mahasiswa11 {
    String nim,nama, kelas;
    Double ipk;
    Mahasiswa11(String nim, String nama, String kelas, Double ipk){
        this.nim = nim;
        this.nama = nama;
        this.kelas = kelas;
        this.ipk = ipk;
    }
    void tampil(){
        System.out.printf("NIM: %s, Nama: %s, Kelas: %s, IPK: %.2f
\n", nim, nama, kelas, ipk);
    }
}
```

4. Mengimplementasikan class Node11 sesuai dengan gambar pada langkah percobaan dan class diagram pada Node11

```
package Jobsheet12;
public class Node11 {
    Mahasiswa11 data;
    Node11 next,prev;
    public Node11(Mahasiswa11 data){
        this.data = data;
        this.next = this.prev = null;
    }
}
```

5. Pada class DoubleLinkedList11, menambahkan attribute head dan tail serta mengimplementasikan method-method yang terdapat pada konsep linked list seperti:
   - isEmpty(), method untuk mengecek apakah linked list kosong.
   - print(), method untuk mencetak data pada linked list dengan proses traverse
   - addFirst(), untuk menambahkan data pada linked list dari depan
   - addLast(), untuk menambahkan data pada linked list dari belakang
   - insertAfter(), memasukkan node yang memiliki data input setelah node yang memiliki data key.
6. Class DoubleLinkedList11

```
package Jobsheet12;
public class DoubleLinkedList11 {
    Node11 head, tail;
    public DoubleLinkedList11(){
        head = tail = null;
    }
    public boolean isEmpty(){
        return head == null;
    }
    public void addFirst(Mahasiswa11 data){
        Node11 newNode = new Node11(data);
        if (isEmpty()) head = tail = newNode;
        else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }
    public void addLast(Mahasiswa11 data){
        Node11 newNode = new Node11(data);
        if (isEmpty()) head = tail = newNode;
        else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }
    public void insertAfter(String keyNim, Mahasiswa11 data){
        Node11 current = head;
        while (current != null && !current.data.nim.equals(keyNim)) {
            current = current.next;
        }
        if (current == null) {
            System.out.printf("Node dengan NIM %s tidak ditemukan.
\n", keyNim);
            return;
        }
        Node11 newNode = new Node11(data);
        if (current == tail) {
            tail.next = newNode;
            newNode.prev = tail;
            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(){
        Node11 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }
    public Node11 search(String nim){
        Node11 current = head;
        while (current != null && !current.data.nim.equals(nim)){
            current = current.next;
        }
        if(current == null) return null;
        else return current;
    }
}
```

7. Pada class DLLMain11, ditambahkan fungsi main pada class DLLMain11, kemudian membuat object dari class DoubleLingkedList11. class Main DLLMain11 digunakan untuk mengeksekusi semua method yang ada pada class DoubleLinkedList11 dengan membuat menu untuk eksekusi dengan implementasi do while dan switch case

```
package Jobsheet12;
import java.util.Scanner;

public class DLLMain11 {
    public static void main(String[] args) {
        DoubleLinkedList11 list = new DoubleLinkedList11();
        Scanner scan = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("\nMenu Double Linked List Mahasiswa");
            System.out.println("1. Tambah Data di awal");
            System.out.println("2. Tambah Data 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("7. Cari Mahasiswa berdasarkan NIM");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = scan.nextInt();
            scan.nextLine();

            switch (pilihan) {
                case 1->{
                    Mahasiswa11 mhs = inputMhs(scan);
                    list.addFirst(mhs);
                }
                case 2->{
                    Mahasiswa11 mhs = inputMhs(scan);
                    list.addLast(mhs);
                }
                case 5 -> list.print();
                case 7 -> {
                    System.out.print("Masukkan NIM yang dicari: ");
                    String nim = scan.nextLine();
                    Node11 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 progam.");
                default -> System.out.println("Pilihan tidak valid!");
            }
        } while (pilihan != 0);
    }
    public static Mahasiswa11 inputMhs(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();
        Mahasiswa11 mhs = new Mahasiswa11(nim, nama, kelas, ipk);
        return mhs;
    }
}
```

8. Hasil Percobaan

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. 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 Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.00
```

9. Pertanyaan
    1. Perbedaan antara single linked list dan double linked list antara lain:
        a. Pada Single linked list, node hanya terdiri dari dua atribut yaitu 1 data dan 1 pointer alamat dengan nama next. Sedangkan pada double linked list node nya terdiri dari tiga atribut yaitu 1 atribut untuk data dan 2 untuk pointer Alamat data sebelum dan sesudah dengan nama next dan prev
        b. Pada single linked list node hanya dapat diakses dari depan ke belakang saja (head to tail) dan tidak dapat diakses dari belakang ke depan (tail to head). Sedangkan double linked list, node dapat diakses dari depan ke belakang (head to tail) ataupun dari belakang ke depam (tail to head)
    2. Pada class Node11 terdapat atribut next dan prev atribut tersebut berfungsi untuk:
        a. Atribut next, atribut ini berfungsi sebagai pointer (penunjuk) alamat dari node selanjutnya pada double linked list.
        b. Sedangkan atribut prev, atribut ini berfungsi sebagai pointer (penunjuk) alamat dari node sebelumnya pada double linked list.
    3. Pada class DoubleLinkedList11 memiliki konstruktor

```
public DoubleLinkedList11(){
    head = tail = null;
}
```

Konstruktor ini berfungsi untuk memberikan nilai awal dari atribut head dan tail dengan nilai null yang menandakan bahwa class DoubleLinkedList11 masih kosong atau masih belum memiliki data.

4. Pada method addFirst(), memiliki kode berikut

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

Kode tersebut berfungsi untuk mengecek jika class DoubleLinkedList11 masih kosong maka saat menambahkan data pada class DoubleLinkedList11 maka data tersebut akan dijadikan head (data awal) sekaligus tail (data akhir) karena hanya ada satu data saja yang berada di dalam class DoubleLinkedList11.

5. Pada method addFirst() pada class DoubleLinkedList11, terdapat baris kode

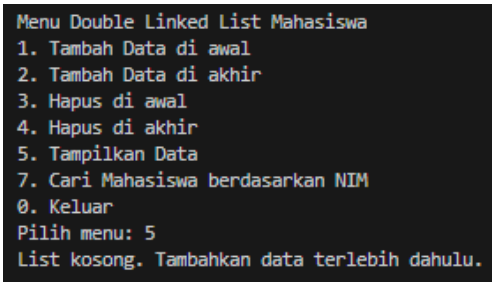
```
head.prev = newNode
```

Pada method addFirst(), data dimasukkan dari depan sehingga ketika menambahkan data dengan method tersebut, data baru tersebut akan menjadi head baru, menggantikan head lama. Langkah pada baris kode di atas yang dilakukan yaitu dengan menyambungkan pointer prev pada head yang lama ke node baru yang ingin ditambahkan. Sehingga pointer prev (penunjuk untuk node sebelumnya) pada head mengarah ke node baru yang akan ditambahkan.

6. Memodifikasi method print() pada class DoubleLinkedList11 agar menampilkan warning / pesan bahwa linked lists masih dalam kondisi kosong.

```
public void print(){
    if (isEmpty()) {
        System.out.println("List kosong. Tambahkan data terlebih
dahulu.");
        return;
    }
    Node11 current = head;
    while (current != null) {
        current.data.tampil();
        current = current.next;
    }
}
```

Hasil modifikasi



7. Pada method insertAfter(), pada class DoubleLinkedList11, terdapat baris kode

```
current.next.prev = newNode
```

Method insertAfter(), berfungsi untuk menambahkan data setelah keyword nim yang diinputkan. Kode tersebut berfungsi untuk menghubungkan pointer prev pada node setelah node current (node yang memiliki data nim sesuai dengan keyword yang diinputkan) ke node baru. Sehingga pointer prev pada node setelah node current tidak lagi menunjuk ke node current melainkan menunjuk ke node baru yang akan ditambahkan.

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

```
do {
    System.out.println("\nMenu Double Linked List Mahasiswa");
    System.out.println("1. Tambah Data di awal");
    System.out.println("2. Tambah Data 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. Menambahkan Data Setelah NIM Tertentu");
    System.out.println("7. Cari Mahasiswa berdasarkan NIM");
    System.out.println("0. Keluar");
    System.out.print("Pilih menu: ");
    pilihan = scan.nextInt();
    scan.nextLine();

    switch (pilihan) {
        case 1->{
            Mahasiswa11 mhs = inputMhs(scan);
            list.addFirst(mhs);
        }
        case 2->{
            Mahasiswa11 mhs = inputMhs(scan);
            list.addLast(mhs);
        }
        case 5 -> list.print();
        case 6 -> {
            System.out.print("Menambahkan Mahasiswa Setelah NIM: ");
            String nim = scan.nextLine();
            System.out.println();
            Mahasiswa11 mhs = inputMhs(scan);
            list.insertAfter(nim, mhs);
        }
        case 7 -> {
            System.out.print("Masukkan NIM yang dicari: ");
            String nim = scan.nextLine();
            Node11 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 progam.");
        default -> System.out.println("Pilihan tidak valid!");
    }
```

Hasil Modifikasi

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 6
Menambahkan Mahasiswa Setelah NIM: 20304050

Masukkan NIM: 22324252
Masukkan Nama: Ron
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.9
Node dengan NIM 20304050 tidak ditemukan.
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. 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 Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 2
Masukkan NIM: 21314151
Masukkan Nama: Lavender
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.8

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.00
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.80
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 6
Menambahkan Mahasiswa Setelah NIM: 20304050

Masukkan NIM: 22324252
Masukkan Nama: Ron
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.9
Node berhasil disisipkan setelah NIM 20304050

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.00
NIM: 22324252, Nama: Ron, Kelas: Gryffindor, IPK: 3.90
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.80
```

1.2. Percobaan 2

1. Pada percobaan ini melanjutkan untuk mengimplementasikan method pada konsep linked list pada class DoubleLinkedList11, method tesebut akan digunakan untuk menghapus isi linked list pada class DoubleLinkedList11:
   - removeFirst(), method untuk menghapus data terdepan (data head) pada class SingleLinkedList11.
   - removeLast(), method untuk menghapus data terakhir (data tail) pada class SingleLinkedList11.

2. Mengimplementasikan method-method pada konsep linked list di atas pada class DoubleLinkedList11.

```
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;
    }
}
```

3. Memodifikasi pada class DLLMain11 dengan menambahkan method objek dari DoubleLinkedList11 pada menu yang telah dibuat untuk menghapus data pada class DoubleLinkedList11.

```
switch (pilihan) {
    case 1->{
        Mahasiswa11 mhs = inputMhs(scan);
        list.addFirst(mhs);
    }
    case 2->{
        Mahasiswa11 mhs = inputMhs(scan);
        list.addLast(mhs);
    }
    case 3 -> list.removeFirst();
    case 4 -> list.removeLast();
    case 5 -> list.print();
    case 6 -> {
        System.out.print("Menambahkan Mahasiswa Setelah NIM: ");
        String nim = scan.nextLine();
        System.out.println();
        Mahasiswa11 mhs = inputMhs(scan);
        list.insertAfter(nim, mhs);
    }
    case 7 -> {
        System.out.print("Masukkan NIM yang dicari: ");
        String nim = scan.nextLine();
        Node11 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 progam.");
    default -> System.out.println("Pilihan tidak valid!");
}
```

4. Hasil Percobaan

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. 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 Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 2
Masukkan NIM: 21314151
Masukkan Nama: Lavender
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.9

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.00
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.90
```

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

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.90
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.90

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 1
Masukkan NIM: 22324252
Masukkan Nama: Ron
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.8

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 22324252, Nama: Ron, Kelas: Gryffindor, IPK: 3.80
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.90
```

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

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 5
NIM: 22324252, Nama: Ron, Kelas: Gryffindor, IPK: 3.80
```

5. Pertanyaan
    1. Pada method removeFirst() memiliki statement

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

        Statement berikut berfungsi untuk memindahkan pointer head yang menunjuk data pertama saat ini ke data disamping kanannya (data setelahnya), kemudian menghapus pointer prev pada head yang baru agar tidak menunjuk lagi ke data sebelumnya (data pertama sebelumnya) sehingga menjadikan data tersebut menjadi data pertama dan menghilangkan alamat yang menunjuk data pertama sebelumnya.
    2. Modifikasi kode program untuk menampilkan pesan “Data sudah berhasil dihapus. Data yang terhapus adalah …“

        Modifikasi method removeFirst dan removeLast pada class DoubleLinkedList11

```
public void removeFirst(){
    Node11 removedData;
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    removedData = head;
    if (head == tail) head = tail = null;
    else{
        head = head.next;
        head.prev = null;
    }
    System.out.println("Data sudah berhasil dihapus. Data yang
terhapus adalah: ");
    removedData.data.tampil();
}

public void removeLast(){
    Node11 removedData;
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    removedData = tail;
    if (head == tail) head = tail = null;
    else{
        tail = tail.prev;
        tail.next = null;
    }
    System.out.println("Data sudah berhasil dihapus. Data yang
terhapus adalah: ");
    removedData.data.tampil();
}
```

Hasil modifikasi

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. 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 Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 2
Masukkan NIM: 21314151
Masukkan Nama: Lavender
Masukkan Kelas: Gryffindor
Masukkan IPK: 3.9
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 3
Data sudah berhasil dihapus. Data yang terhapus adalah:
NIM: 20304050, Nama: Hermione, Kelas: Gryffindor, IPK: 4.00

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
0. Keluar
Pilih menu: 4
Data sudah berhasil dihapus. Data yang terhapus adalah:
NIM: 21314151, Nama: Lavender, Kelas: Gryffindor, IPK: 3.90
```

2. Tugas

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

   Method add pada kelas DoubleLinkedList11.

```
public void add(int index, Mahasiswa11 data){
    if (index < 0) {
        System.out.println("Index tidak valid.");
        return;
    }
    if (isEmpty() || index == 0) {
        addFirst(data);
        return;
    }
    Node11 newNode = new Node11(data);
    Node11 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    if (current == null) {
        System.out.println("Index berada di akhir atau melebihi ukuran,
menambahkan di akhir.");
        addLast(data);
    } else {
        newNode.next = current;
        newNode.prev = current.prev;
        current.prev.next = newNode;
        current.prev = newNode;
    }
}
```

Menambahkan menu untuk method add pada class DLLMain11

```
case 8 -> {
    System.out.print("Menambahkan Mahasiswa Pada Indeks: ");
    int index = scan.nextInt();
    System.out.println(); scan.nextLine();
    Mahasiswa11 mhs = inputMhs(scan);
    list.add(index, mhs);
}
```

Hasil

```
Pilih menu: 5
NIM: 2215, Nama: Jihan, Kelas: 2D, IPK: 4.00
NIM: 2113, Nama: Fara, Kelas: 2A, IPK: 3.30

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 8
Menambahkan Mahasiswa Pada Indeks: 1

Masukkan NIM: 2221
Masukkan Nama: Iza
Masukkan Kelas: 1A
Masukkan IPK: 3.8
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 5
NIM: 2215, Nama: Jihan, Kelas: 2D, IPK: 4.00
NIM: 2221, Nama: Iza, Kelas: 1A, IPK: 3.80
NIM: 2113, Nama: Fara, Kelas: 2A, IPK: 3.30
```

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

   Method removeAfter pada kelas DoubleLinkedList11.

```
public void removeAfter(String keyNim){
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    Node11 current = head;
    while (current != null && !current.data.nim.equals(keyNim)) {
        current = current.next;
    }
    if (current == null) {
        System.out.printf("Node dengan NIM %s tidak ditemukan. \n",
keyNim);
        return;
    }
    if (head == tail) {
        System.out.println("Hanya ada satu node, tidak bisa dihapus
setelahnya.");
        return;
    }
    if (current == tail) {
        System.out.println("Node terakhir tidak memiliki node
setelahnya.");
        return;
    }

    Node11 removedData = current.next;
    current.next = removedData.next;
    if (removedData.next != null) removedData.next.prev = current;
    else {
        tail = current;
        tail.next = null;
    }
    System.out.println("Node berhasil dihapus setelah NIM " + keyNim);
    System.out.println("Data yang terhapus adalah: ");
    removedData.data.tampil();
}
```

Menambahkan menu untuk operasi removeAfter pada class DLLMain11.

```
case 9 -> {
    System.out.print("Menghapus Mahasiswa Setelah NIM: ");
    String nim = scan.nextLine();
    list.removeAfter(nim);
}
```

Hasil

```
Pilih menu: 5
NIM: 2215, Nama: Jihan, Kelas: 2D, IPK: 4.00
NIM: 2221, Nama: Iza, Kelas: 1A, IPK: 3.80
NIM: 2113, Nama: Fara, Kelas: 2A, IPK: 3.30

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 9
Menghapus Mahasiswa Setelah NIM: 2221
Node berhasil dihapus setelah NIM 2221
Data yang terhapus adalah:
NIM: 2113, Nama: Fara, Kelas: 2A, IPK: 3.30
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 5
NIM: 2215, Nama: Jihan, Kelas: 2D, IPK: 4.00
NIM: 2221, Nama: Iza, Kelas: 1A, IPK: 3.80
```

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

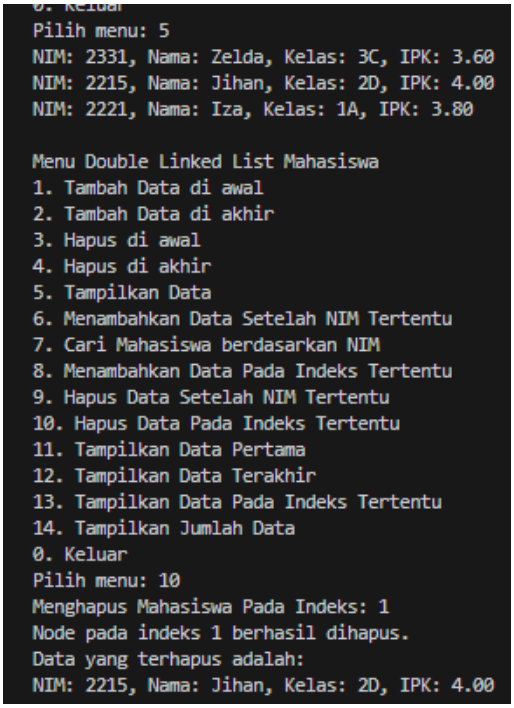
   Method remove pada class DoubleLinkedList11

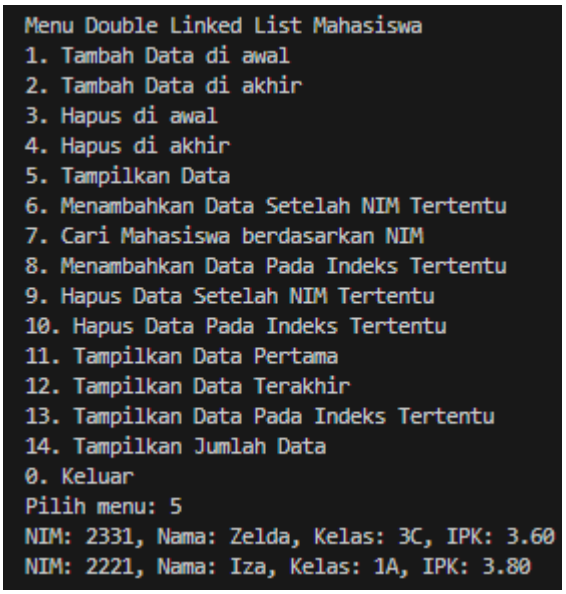
```
public void remove(int index){
    if (index < 0) {
        System.out.println("Index tidak valid.");
        return;
    }
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    if (index == 0) {
        removeFirst();
        return;
    }
    Node11 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    if (current == null) {
        System.out.printf("Node pada indeks %s kosong. \n", index);
        return;
    }
    if (current == tail) {
        removeLast();
        return;
    }
    else{
        Node11 removedData = current;
        current.next.prev = current.prev;
        current.prev.next = current.next;
        System.out.printf("Node pada indeks %s berhasil dihapus.\n",index);
        System.out.println("Data yang terhapus adalah: ");
        removedData.data.tampil();
    }
}
```

Menambahkan menu untuk operasi remove pada class DLLMain11

```
case 10 -> {
    System.out.print("Menghapus Mahasiswa Pada Indeks: ");
    int index = scan.nextInt();
    list.remove(index);
}
```

Hasil





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

   Method getFirst(), getLast(), dan getIndex() pada class DoubleLinkedList11

```
public Node11 getFirst(){
    return head;
}
public Node11 getLast(){
    return tail;
}
public Node11 getIndex(int index){
    if (index < 0) {
        System.out.println("Index Tidak Valid");
        return null;
    }
    Node11 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    return current;
}
```

Menambahkan menu untuk operasi getFirst, getLast, dan getIndex pada class DLLMain11

```
public Node11 getFirst(){
    return head;
}
public Node11 getLast(){
    return tail;
}
public Node11 getIndex(int index){
    if (index < 0) {
        System.out.println("Index Tidak Valid");
        return null;
    }
    Node11 current = head;
    for (int i = 0; i <= index; i++) {
        current = current.next;
    }
    return current;
}
```

Hasil

```
0. Keluar
Pilih menu: 5
NIM: 2114, Nama: Jane, Kelas: 3E, IPK: 3.30
NIM: 2111, Nama: Oliver, Kelas: 1F, IPK: 3.40
NIM: 2332, Nama: Zelda, Kelas: 2A, IPK: 3.80
NIM: 2331, Nama: Iza, Kelas: 3A, IPK: 3.60

Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 11
Data Mahasiswa Pertama Adalah:
NIM: 2114, Nama: Jane, Kelas: 3E, IPK: 3.30
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 12
Data Mahasiswa Terakhir Adalah:
NIM: 2331, Nama: Iza, Kelas: 3A, IPK: 3.60
```

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 13
Lihat Mahasiswa Pada Indeks: 2
Data Mahasiswa Pada Indeks 2 Adalah:
NIM: 2332, Nama: Zelda, Kelas: 2A, IPK: 3.80
```

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

   Memodifikasi class DoubleLinkedList11 dengan menambahkan atribut size dan memodifikasi method untuk melakukan penambahan dan pengurangan data

```
Node11 head, tail;
int size;
public DoubleLinkedList11(){
    head = tail = null;
    size = 0;
}
```

```
public void addFirst(Mahasiswa11 data){
    Node11 newNode = new Node11(data);
    if (isEmpty()) head = tail = newNode;
    else {
        newNode.next = head;
        head.prev = newNode;
        head = newNode;
    }
    size++;
}
```

```
public void addLast(Mahasiswa11 data){
    Node11 newNode = new Node11(data);
    if (isEmpty()) head = tail = newNode;
    else {
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
    }
    size++;
}
```

```
public void insertAfter(String keyNim, Mahasiswa11 data){
    Node11 current = head;
    while (current != null && !current.data.nim.equals(keyNim)) {
        current = current.next;
    }
    if (current == null) {
        System.out.printf("Node dengan NIM %s tidak ditemukan. \n", keyNim);
        return;
    }

    Node11 newNode = new Node11(data);
    if (current == tail) {
        tail.next = newNode;
        newNode.prev = tail;
        tail = newNode;
    }else{
        newNode.next = current.next;
        newNode.prev = current;
        current.next.prev = newNode;
        current.next = newNode;
    }
    size++;
    System.out.println("Node berhasil disisipkan setelah NIM " + keyNim);
}
```

```
public void add(int index, Mahasiswa11 data){
    if (index < 0) {
        System.out.println("Index tidak valid.");
        return;
    }
    if (isEmpty() || index == 0) {
        addFirst(data);
        return;
    }
    Node11 newNode = new Node11(data);
    Node11 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    if (current == null) {
        System.out.println("Index berada di akhir atau melebihi ukuran,
menambahkan di akhir.");
        addLast(data);
    } else {
        newNode.next = current;
        newNode.prev = current.prev;
        current.prev.next = newNode;
        current.prev = newNode;
        size++;
    }
}
```

```
public void removeAfter(String keyNim){
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    Node11 current = head;
    while (current != null && !current.data.nim.equals(keyNim)) {
        current = current.next;
    }
    if (current == null) {
        System.out.printf("Node dengan NIM %s tidak ditemukan. \n", keyNim);
        return;
    }
    if (head == tail) {
        System.out.println("Hanya ada satu node, tidak bisa dihapus
setelahnya.");
        return;
    }
    if (current == tail) {
        System.out.println("Node terakhir tidak memiliki node setelahnya.");
        return;
    }

    Node11 removedData = current.next;
    current.next = removedData.next;
    if (removedData.next != null) removedData.next.prev = current;
    else {
        tail = current;
        tail.next = null;
    }
    size--;
    System.out.println("Node berhasil dihapus setelah NIM " + keyNim);
    System.out.println("Data yang terhapus adalah: ");
    removedData.data.tampil();
}
```

```
public void remove(int index){
    if (index < 0) {
        System.out.println("Index tidak valid.");
        return;
    }
    if (isEmpty()) {
        System.out.println("List kosong, tidak bisa dihapus.");
        return;
    }
    if (index == 0) {
        removeFirst();
        return;
    }
    Node11 current = head;
    for (int i = 0; i < index; i++) {
        current = current.next;
    }
    if (current == null) {
        System.out.printf("Node pada indeks %s kosong. \n", index);
        return;
    }
    if (current == tail) {
        removeLast();
        return;
    }
    else{
        Node11 removedData = current;
        current.next.prev = current.prev;
        current.prev.next = current.next;
        size--;
        System.out.printf("Node pada indeks %s berhasil dihapus. \n", index);
        System.out.println("Data yang terhapus adalah: ");
        removedData.data.tampil();
    }
}

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

Hasil

```
Menu Double Linked List Mahasiswa
1. Tambah Data di awal
2. Tambah Data di akhir
3. Hapus di awal
4. Hapus di akhir
5. Tampilkan Data
6. Menambahkan Data Setelah NIM Tertentu
7. Cari Mahasiswa berdasarkan NIM
8. Menambahkan Data Pada Indeks Tertentu
9. Hapus Data Setelah NIM Tertentu
10. Hapus Data Pada Indeks Tertentu
11. Tampilkan Data Pertama
12. Tampilkan Data Terakhir
13. Tampilkan Data Pada Indeks Tertentu
14. Tampilkan Jumlah Data
0. Keluar
Pilih menu: 14
Jumlah Data Mahasiswa Pada Node: 4
```

```
0. Keluar
Pilih menu: 5
NIM: 2114, Nama: Jane, Kelas: 3E, IPK: 3.30
NIM: 2111, Nama: Oliver, Kelas: 1F, IPK: 3.40
NIM: 2332, Nama: Zelda, Kelas: 2A, IPK: 3.80
NIM: 2331, Nama: Iza, Kelas: 3A, IPK: 3.60
```

Commit dan Push kode program ke github

```
PS D:\Java\Praktikum-ASD> git add .\Jobsheet12\*
PS D:\Java\Praktikum-ASD> git commit -m "Jobsheet 12 Double Linked List Praktikum Percobaan, Pertanyaan, dan Tugas"
[main a21881b] Jobsheet 12 Double Linked List Praktikum Percobaan, Pertanyaan, dan Tugas
 4 files changed, 381 insertions(+)
 create mode 100644 Jobsheet12/DLLMain11.java
 create mode 100644 Jobsheet12/DoubleLinkedList11.java
 create mode 100644 Jobsheet12/Mahasiswa11.java
 create mode 100644 Jobsheet12/Node11.java
PS D:\Java\Praktikum-ASD> git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 2.93 KiB | 599.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ghazwanz/Praktikum-ASD.git
   5d12931..a21881b  main -> main
branch 'main' set up to track 'origin/main'.
PS D:\Java\Praktikum-ASD>
```