LAPORAN TUGAS ALGORITMA DAN STRUKTUR DATA
JOBSHEET 12

`

NAMA : RADITYA RIEFKI
KELAS : TI 1E
ABSEN : 23

12.2 Kegiatan Praktikum 1

12.2.1 Percobaan 1

Kode Program

Mahasiswa23

```
package jobsheet12;

public class Mahasiswa23 {
    public String nim;
    public String nama;
    public String kelas;
    public double ipk;

    public Mahasiswa23(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);
    }
}
```

Node23

```
package jobsheet12;

import org.w3c.dom.Node;

public class Node23 {
    Mahasiswa23 data;
    Node23 prev;
    Node23 next;

    public Node23(Mahasiswa23 data){
        this.data = data;
        this.prev = null;
        this.next = null;
    }
}
```

DoubleLinkedLists

```
package jobsheet12;

import jobsheet11.NodeMahasiswa23;

public class DoubleLinkedList23 {
    Node23 head;
    Node23 tail;

    public DoubleLinkedList23() {
        head = null;
        tail = null;
    }

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

    public void addFirst (Mahasiswa23 data){
        Node23 newNode = new Node23(data);
        if (isEmpty()) {
            head = tail = newNode;
        }else{
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }

    public void addLast (Mahasiswa23 data){
        Node23 newNode = new Node23(data);
        if (isEmpty()) {
            head = tail = newNode;
        }else{
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    public void insertAfter (String keyNim, Mahasiswa23 data){
        Node23 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;
        }
        Node23 newNode = new Node23(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(){
        Node23 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;

        }
    }

    public void removeFirst(){
        if (isEmpty()) {
            System.out.println("Linked List masih kosong, tidak dapat
dihapus!");
        }
        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 Node23 search(String targetNim) {
        Node23 current = head;

        while (current != null) {
```

```
            if (current.data.nim.equals(targetNim)) {
                return current;
            }
            current = current.next;
        }
        return null;
    }

}
```

DLLMain

```
package jobsheet12;

import java.util.Scanner;

public class DLLMain {
    static Scanner sc = new Scanner(System.in);
    public static Mahasiswa23 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();
        Mahasiswa23 mhs = new Mahasiswa23(nim, nama, kelas, ipk);
        return mhs;
    }
    public static void main(String[] args) {
        DoubleLinkedList23 list = new DoubleLinkedList23();
        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:{
                    Mahasiswa23 mhs = inputMahasiswa();
                    list.addFirst(mhs);
```

```
                    break;
                }
                case 2: {
                    Mahasiswa23 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();
                    Node23 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);
    }
}
```

12.2.2 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
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
```

12.2.3 Pertanyaan Percobaan

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

- Single Linked List menyimpan referensi ke node berikutnya menggunakan next, sehingga traversal hanya bisa dilakukan searah (head ke tail).

Double Linked List menyimpan referensi ke node sebelumnya (prev) dan berikutnya (next), sehingga traversal bisa dua arah (maju/mundur).

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

- atribut next dan prev mempunyai kegunaan untuk mengakses setiap node

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

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

- sebagai penunjuk awal mula dan akhir sebuah struktur data

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

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

- jika struktur data kosong maka newNode (input data baru) dinisialisasi ke head dan tail karena hanya terdapat satu data pada struktur data

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

- menyambungkan node head.prev ke newNode sebelum menginisalisasi newNode sebagai head

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

```
public void print(){
    Node23 current = head;
    if (isEmpty()) {
        System.out.println(x:"Data Kosong!");
        return;
    }else{
    while (current != null) {
        current.data.tampil();
        current = current.next;
        }
    }
}
```

```
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
Data Kosong!
```

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

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

- menginisalisasi/menyambungkan data current.next.prev ke newNode

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

Kode Main

```
case 7: {
    System.out.print(s:"Masukkan NIM mahasiswa yang ingin dicari : ");
    String nim = sc.nextLine();
    Node23 found = list.search(nim);
    if (found != null) {
        Mahasiswa23 mhs = inputMahasiswa();
        list.insertAfter(nim, mhs);
    } else {
        System.out.println(x:"Data not found");
    }
    break;
}
```

Kode DoubleLinkedList (Function)

```
public void insertAfter (String keyNim, Mahasiswa23 data){
    Node23 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;
    }
    Node23 newNode = new Node23(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);
}
```

12.3 Kegiatan Praktikum 2

12.3.1 Tahapan Percobaan

Kode

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

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

12.3.2 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
7. Tambah data (Insert After)
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
7. Tambah data (Insert After)
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
7. Tambah data (Insert After)
0. Keluar
Pilih menu: 5
Data Kosong!

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. Tambah data (Insert After)
0. Keluar
Pilih menu:
```

12.3.3 Pertanyaan Percobaan

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

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

- Kode tersebut merupakan proses penghapusan data pada head (head.prev = null) memutus node prev dan menghapusnya

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

Remove First

Penambahan Kode

```
public void removeFirst(){
    if (isEmpty()) {
        System.out.println(x:"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;
    }
}
```

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
7. Tambah data (Insert After)
0. Keluar
Pilih menu: 1
Nim : 1
Nama : Alam
Kelas : 1E
IPK : 3.3

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
7. Tambah data (Insert After)
0. Keluar
Pilih menu: 3
Data sudah berhasil dihapus. Data yang terhapus adalah Alam
```

RemoveLast

Penambahan Kode

```java
public void removeLast(){
    if (isEmpty()) {
        System.out.println(x:"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;
    }
```

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
7. Tambah data (Insert After)
0. Keluar
Pilih menu: 1
Nim : 1
Nama : Alam
Kelas : 2
IPK : 3.4

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. Tambah data (Insert After)
0. Keluar
Pilih menu: 4
Data sudah berhasil dihapus. Data yang terhapus adalah Alam
```

12.5 Tugas Praktikum

1. Fungsi Add

```
public void add(Mahasiswa23 data, int index) {
        if (index == 0) {
            addFirst(data);
            return;
        }

        Node23 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;
        }

        Node23 newNode = new Node23(data);
        temp.next.prev = newNode;
        newNode.next = temp.next;
        temp.next = newNode;
        newNode.prev = temp;
    }
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
0. Keluar
Pilih menu: 8
NIM : 3
Nama : Albi
Kelas : 3E
IPK : 4
Masukkan Index : 2

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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Alam, Kelas: 1E, IPK 4.0
NIM: 2, Nama: Elma, Kelas: 2E, IPK 3.0
NIM: 3, Nama: Albi, Kelas: 3E, IPK 4.0
```

2. Fungsi RemoveAfter

```
public void removeAfter(String key) {
        if (isEmpty()) {
            System.out.println("List kosong, tidak bisa dihapus");
            return;
        }
        Node23 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;

    }
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data (Setelah Mahasiswa)
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Alam, Kelas: 1E, IPK 4.0
NIM: 2, Nama: Elma, Kelas: 2E, 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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data (Setelah Mahasiswa)
0. Keluar
Pilih menu: 9
Masukkan nama mahasiswa : alam

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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data (Setelah Mahasiswa)
0. Keluar
Pilih menu: 5
NIM: 1, Nama: Alam, Kelas: 1E, IPK 4.0
```

3. Fungsi 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;
        }

        Node23 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;
    }
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data Mahasiswa (Setelah Mahasiswa)
10. Hapus Data (Index Tertentu)
0. Keluar
Pilih menu: 10
Masukkan index : 0
Data sudah berhasil dihapus. Data yang terhapus adalah Alam
```

4. GetFirst GetLast GetIndex

```
Mahasiswa23 getFirst() {
        if (isEmpty()) {
            return null;
        }
        return head.data;
    }

    Mahasiswa23 getLast() {
        if (isEmpty()) {
            return null;
        }

        return tail.data;
    }

    Mahasiswa23 getIndex(int index) {
        if (isEmpty()) {
            return null;
        }

        Node23 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;
    }
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data Mahasiswa (Setelah Mahasiswa)
10. Hapus Data (Index Tertentu)
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data (Index Tertentu)
14. Tampilkan ukuran list
0. Keluar
Pilih menu: 11
NIM: 1, Nama: Alam, Kelas: 1E, 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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data Mahasiswa (Setelah Mahasiswa)
10. Hapus Data (Index Tertentu)
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data (Index Tertentu)
14. Tampilkan ukuran list
0. Keluar
Pilih menu: 12
NIM: 2, Nama: Elma, Kelas: 2E, IPK 3.0
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data Mahasiswa (Setelah Mahasiswa)
10. Hapus Data (Index Tertentu)
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data (Index Tertentu)
14. Tampilkan ukuran list
0. Keluar
Pilih menu: 13
Masukkan Index :
0
NIM: 1, Nama: Alam, Kelas: 1E, IPK 4.0
```

5. GetIndex

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

      Node23 temp = head;
      while (temp != null) {
         temp = temp.next;
         counter++;
      }

      return counter;
   }

}
```

```
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. Tambah Data (Setelah Mahasiswa)
8. Tambah data (Index Tertentu)
9. Hapus Data Mahasiswa (Setelah Mahasiswa)
10. Hapus Data (Index Tertentu)
11. Tampilkan data pertama
12. Tampilkan data terakhir
13. Tampilkan data (Index Tertentu)
14. Tampilkan ukuran list
0. Keluar
Pilih menu: 14
Jumlah Mahasiswa : 0
```