Praktikum 12 – Double Linked List

Percobaan 1

1. Buat folder baru dengan nama Jobsheet12, lalu buat class mahasiswa

```java
package Jobsheet12;

public class Mahasiswa01 {

    public String nama, nim, kelas;
    public double ipk;

    public Mahasiswa01(String nama, String nim, String kelas, double ipk) {
        this.nama = nama;
        this.nim = nim;
        this.kelas = kelas;
        this.ipk = ipk;
    }

    public void tampil() {
        System.out.println("NIM: " + nim + ", Nama: " + nama + "n Kelas: " + kelas + ", IPK: " + ipk);
    }

}
```

2. Buat class Node01

```java
package Jobsheet12;

public class Node01 {
    Mahasiswa01 data;
    Node01 prev, next;

    public Node01(Mahasiswa01 data) {
        this.data = data;
        this.prev = null;
        this.next = null;
    }


}
```

3. Buat class DoubleLinkedList lalu isi dengan atribut dan methodnya

```java
package Jobsheet12;

public class DoubleLinkedList {

    Node01 head, tail;

    public DoubleLinkedList(Node01 head, Node01 tail) {
        this.head = null;
        this.tail = null;
    }

    public DoubleLinkedList() {
    }

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

    public void addFirst(Mahasiswa01 data) {
        Node01 newNode = new Node01(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            newNode.next = head;
            head.prev = newNode;
            head = newNode;
        }
    }
```

```java
    public void addLast(Mahasiswa01 data) {
        Node01 newNode = new Node01(data);
        if (isEmpty()) {
            head = tail = newNode;
        } else {
            tail.next = newNode;
            newNode.prev = tail;
            tail = newNode;
        }
    }

    public void insertAfter(String keyNim, Mahasiswa01 data) {
        Node01 current = head;

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

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

        Node01 newNode = new Node01(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() {
        Node01 current = head;
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

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

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

    public Node01 search(String nim) {
        Node01 tmp = head;
        while (tmp != null) {
            if (tmp.data.nim.equals(nim)) {
                return tmp;
            }
            tmp = tmp.next;
        }
        return null;
    }

}

```

4. Buat class DLLMain sebagai Main

```java
package Jobsheet12;

import java.util.Scanner;

public class DLLMain {

    Run | Debug | Run main | Debug main
    public static void main(String[] args) {
        DoubleLinkedList list = new DoubleLinkedList();
        Scanner scan = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println(x:"\nMenu Double Linked List Mahasiswa");
            System.out.println(x:"1. Tambah di Awal");
            System.out.println(x:"2. Tambah di Akhir");
            System.out.println(x:"3. Hapus di Awal");
            System.out.println(x:"4. Hapus di Akhir");
            System.out.println(x:"5. Tampilkan Data");
            System.out.println(x:"6. Cari Mahasiswa Berdasarkan NIM");
            System.out.println(x:"0. Keluar");
            System.out.print(s:"Pilih Menu: ");
            pilihan = scan.nextInt();
            scan.nextLine();
```

```
        switch (pilihan) {
            case 1 -> {
                Mahasiswa01 mhs = inputMahasiswa(scan);
                list.addFirst(mhs);
            }
            case 2 -> {
                Mahasiswa01 mhs = inputMahasiswa(scan);
                list.addLast(mhs);
            }
            case 3 ->
                list.removeFirst();
            case 4 ->
                list.removeLast();
            case 5 ->
                list.print();
            case 6 -> {
                System.out.print(s:"Masukkan NIM yang dicari: ");
                String nim = scan.nextLine();
                Node01 found = list.search(nim);
                if (found != null) {
                    System.out.println(x:"Data ditemukan:");
                    found.data.tampil();
                } else {
                    System.out.println(x:"Data tidak ditemukan");
                }
            }
            case 0 ->
                System.out.println(x:"Keluar dari program");
            default ->
                System.out.println(x:"Pilihan tidak valid!");
        }
    } while (pilihan != 0);
    scan.close();
}
```

```
public static Mahasiswa01 inputMahasiswa(Scanner scan) {
    System.out.print(s:"Masukkan Nama: ");
    String nama = scan.nextLine();
    System.out.print(s:"Masukkan NIM: ");
    String nim = scan.nextLine();
    System.out.print(s:"Masukkan Kelas: ");
    String kelas = scan.nextLine();
    System.out.print(s:"Masukkan IPK: ");
    double ipk = scan.nextDouble();
    scan.nextLine();
    return (new Mahasiswa01(nama, nim, kelas, ipk));
}
```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
2. Perhatikan class Node01, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan dari konstruktor tersebut?

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

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

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

5. Perhatikan pada method **addFirst().** Apakah arti statement head.prev = newNode ?
6. Modifikasi code pada fungsi **print()** agar dapat menampilkan warning/ pesan bahwa linked lists masih dalam kondisi.

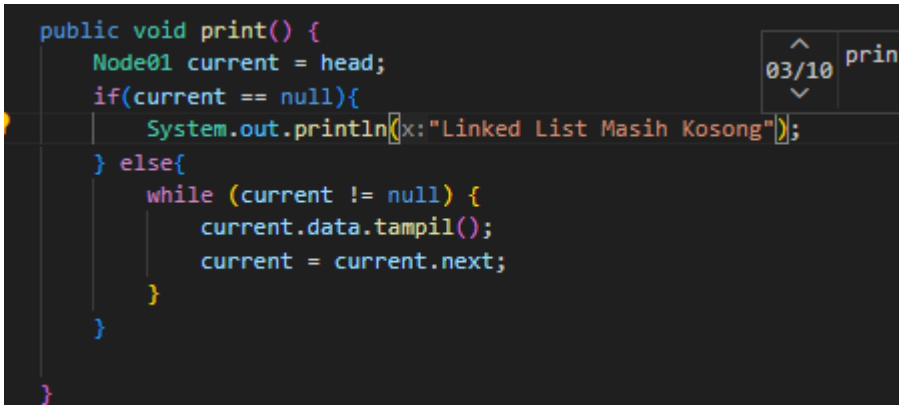
7. Pada insertAfter(), apa maksud dari kode berikut ?

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

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

Jawaban

1. Double linked list punya bagian yang menyimpan linked list sebelumnya, sedangkan single linked list hanya menyimpan alamat linked list selanjutnya saja.
2. Atribut next untuk menyimpan Alamat linked list selanjutnya, dan prev menyimpan Alamat linked list sebelumnya
3. Untuk memberikan nilai awal dari head dan tail, jadi nanti bisa diisi nilainya.
4. Jika linked list belum ada isinya, newNode akan menjadi head dan tail sekaligus
5. Head.prev = newNode ini berarti node baru ini akan ditunjuk prev dari head sehingga new node ini berada di sebelum head

6. 
```java
public void print() {
    Node01 current = head;
    if(current == null){
        System.out.println(x:"Linked List Masih Kosong");
    } else{
        while (current != null) {
            current.data.tampil();
            current = current.next;
        }
    }

}
```
7. Maksudnya adalah prev dari linked list setelah current akan menunjuk newNode

8.
```
case 7->{
    System.out.print(s:"Masukkan data setelah NIM: ");
    String nim = scan.nextLine();
    list.insertAfter(nim, inputMahasiswa(scan));
}
```

```
Masukkan IPK: 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. Sisipkan Data Mahasiswa
0. Keluar
Pilih Menu: 1
Masukkan Nama: 4
Masukkan NIM: 4
Masukkan Kelas: 4
Masukkan IPK: 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. Sisipkan Data Mahasiswa
0. Keluar
Pilih Menu: 7
Masukkan data setelah NIM: 2
Masukkan Nama: 6
Masukkan NIM: 6
Masukkan Kelas: 6
Masukkan IPK: 6
Node berhasil disisipkan setelah NIM 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. Sisipkan Data Mahasiswa
0. Keluar
Pilih Menu: 5
NIM: 4, Nama: 4n Kelas: 4, IPK: 4.0
NIM: 2, Nama: 2n Kelas: 2, IPK: 2.0
NIM: 6, Nama: 6n Kelas: 6, IPK: 6.0
NIM: 1, Nama: 1n Kelas: 1, IPK: 3.0
```

Percobaan 2

1. Menambahkan method remove first dan remove last

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

12.3.3 Pertanyaan Percobaan

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

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

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

Jawaban

1. Maksudnya adalah head akan pindah menunjuk linked list selanjutnya, lalu bagian prev dari head akan diberi nilai null
2.

```
case 3 -> {
    System.out.println(x:"Data sudah dihapus. Data yang terhapus adalah: ");
    list.head.data.tampil();
    list.removeFirst();
}
case 4 ->{
    System.out.println(x:"Data sudah dihapus. Data yang terhapus adalah: ");
    list.tail.data.tampil();
    list.removeLast();
}
```

12.5 Tugas Praktikum

1. Tambahkan fungsi add() pada kelas DoubleLinkedList untuk menambahkan node pada indeks tertentu
2. Tambahkan removeAfter() pada kelas DoubleLinkedList untuk menghapus node setelah data *key*.
3. Tambahkan fungsi remove() pada kelas DoubleLinkedList untuk menghapus node pada indeks tertentu.
4. Tambahkan fungsi getFirst(), getLast() dan getIndex() untuk menampilkan data pada node head, node tail dan node pada indeks tertentu.
5. tambahkan kode program dan fungsi agar dapat membaca size/ jumlah data pada Double Linked List

--- *** ---

1.
```
public void add(int index, Mahasiswa01 input) {
    if (index < 0) {
        System.out.println(x:"Indeks salah");
    } else if (index == 0) {
        addFirst(input);
    } else {
        Node01 temp = head;
        for (int i = 0; i < index; i++) {
            temp = temp.next;
        }
        Node01 ndBaru = new Node01(input);
        temp.next = ndBaru;
        ndBaru.prev = temp;
        if (temp.next.next == null) {
            tail = temp.next;
        }
    }
}
```

2.
```java
public void removeAfter(String keyNim, Mahasiswa01 data) {
    Node01 current = head;

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

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

    if (current == head) {
        removeFirst();
    } else if (current == tail) {
        removeLast();
    } else {
        current = current.prev;
        current.next = current.next.next;
        current.next.prev = current;
    }

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

3.
```java
public void removeAtIndex(int index) {
    if (index < 0) {
        System.out.println(x:"Indeks salah");
    } else if (index == 0) {
        removeFirst();
    } else {
        Node01 temp = head;
        for (int i = 0; i < index; i++) {
            temp = temp.next;
        }
        temp.next = temp.next.next;
        temp.next.prev = temp;
        if (temp.next.next == null) {
            tail = temp.next;

    }
}
```

4.
```java
public void getData(int index) {
    Node01 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    tmp.data.tampil();
}

public void getFirst(){
    head.data.tampil();
}

public void getLast(){
    tail.data.tampil();
}
```

5.
```java
public int getSize(){
    return size;
}
```

```
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 Mahasiswa
8. Jumlah Mahasiswa
0. Keluar
Pilih Menu: 1
Masukkan Nama: 1
Masukkan NIM: 1
Masukkan Kelas: 1
Masukkan IPK: 1

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 Mahasiswa
8. Jumlah Mahasiswa
0. Keluar
Pilih Menu: 1
Masukkan Nama: 2
Masukkan NIM: 2
Masukkan Kelas: 2
Masukkan IPK: 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. Sisipkan Data Mahasiswa
8. Jumlah Mahasiswa
0. Keluar
Pilih Menu: 8
Jumlah Mahasiswa: 2
```