# JOBSHEET 12 Double Linked Lists

## 12.1 Tujuan Praktikum

Setelah melakukan praktikum ini, mahasiswa mampu:

- 1. memahami algoritma double linked lists;
- 2. membuat dan mendeklarasikan struktur algoritma double linked lists;
- 3. menerapkan algoritma double linked lists dalam beberapa study case.

### 12.2 Kegiatan Praktikum 1

#### 12.2.1 Percobaan 1

Pada percobaan 1 ini akan dibuat class Node dan class DoubleLinkedLists yang didalamnya terdapat operasi-operasi untuk menambahkan data dengan beberapa cara (dari bagian depan linked list, belakang ataupun indeks tertentu pada linked list).

1. Perhatikan diagram class Node dan class DoublelinkedLists di bawah ini! Diagram class ini yang selanjutnya akan dibuat sebagai acuan dalam membuat kode program DoubleLinkedLists.

Node

data: int

prev: Node

next: Node

Node(prev: Node, data:int, next:Node)

```
DoubleLinkedLists
head: Node
size: int

DoubleLinkedLists()
isEmpty(): boolean
addFirst (): void
addLast(): void
add(item: int, index:int): void
size(): int
clear(): void
print(): void
```

- 2. Buat paket baru dengan nama doublelinkedlists
- 3. Buat class di dalam paket tersebut dengan nama Node

```
package doublelinkedlists;

/**...4 lines */
public class Node {
}
```

4. Di dalam class tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
4 int data;
5 Node prev, next;
```

5. Selanjutnya tambahkan konstruktor default pada class Node sesuai diagram di atas.

6. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node seperti gambar berikut:

```
package doublelinkedlists;

/**...4 lines */
public class DoubleLinkedLists {
}
```

7. Pada class DoubleLinkedLists tersebut, deklarasikan atribut sesuai dengan diagram class di atas.

```
Node head;
int size;
```

8. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

```
public DoubleLinkedLists() {
   head = null;
   size = 0;
}
```

9. Buat method isEmpty(). Method ini digunakan untuk memastikan kondisi linked list kosong.

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

10. Kemudian, buat method **addFirst()**. Method ini akan menjalankan penambahan data di bagian depan linked list.

```
public void addFirst(int item) {
   if (isEmpty()) {
      head = new Node(null, item, null);
   } else {
      Node newNode = new Node(null, item, head);
      head.prev = newNode;
      head = newNode;
   }
   size++;
}
```

11. Selain itu pembuatan method **addLast()** akan menambahkan data pada bagian belakang linked list.

```
public void addLast(int item) {
   if (isEmpty()) {
      addFirst(item);
} else {
      Node current = head;
      while (current.next != null) {
            current = current.next;
      }
      Node newNode = new Node(current, item, null);
      current.next = newNode;
      size++;
}
```

12. Untuk menambakan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method add(int item, int index)

```
public void add(int item, int index) throws Exception {
   if (isEmpty()) {
       addFirst(item);
    } else if (index < 0 || index > size) {
       throw new Exception ("Nilai indeks di luar batas");
    } else {
       Node current = head;
        int i = 0:
        while (i < index) {
            current = current.next;
            i++;
        1
        if (current.prev == null) {
            Node newNode = new Node(null, item, current);
            current.prev = newNode;
           head = newNode;
        } else {
            Node newNode = new Node(current.prev, item, current);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
   size++;
```

13. Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis, sehingga dapat dibuat method size() untuk mendapatkan nilai dari size.

```
138  public int size() {
139  return size;
140 }
```

14. Selanjutnya dibuat method **clear()** untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong.

```
141  public void clear() {
142  head = null;
143  size = 0;
144 }
```

15. Untuk mencetak isi dari linked lists dibuat method **print().** Method ini akan mencetak isi linked lists berapapun size-nya. Jika kosong akan dimunculkan suatu pemberitahuan bahwa linked lists dalam kondisi kosong.

```
public void print() {
   if (!isEmpty()) {
      Node tmp = head;
      while (tmp != null) {
            System.out.print(tmp.data + "\t");
            tmp = tmp.next;
      }
      System.out.println("\nberhasil diisi");
   } else {
      System.out.println("Linked Lists Kosong");
   }
}
```

16. Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.

package doublelinkedlists;

```
/**...4 lines */
public class DoubleLinkedListsMain {
   public static void main(String[] args) {
   }
}
```

17. Pada main class pada langkah 16 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

```
19
              doubleLinkedList dll = new doubleLinkedList();
20
              dll.print();
21
              System.out.println("Size : "+dll.size());
              System.out.println("==========
22
23
              dll.addFirst(3);
24
              dll.addLast(4);
25
              dll.addFirst(7);
26
              dll.print();
27
              System.out.println("Size : "+dll.size());
              System.out.println("========
                                                                    =");
28
              dll.add(40, 1);
29
30
              dll.print();
              System.out.println("Size : "+dll.size());
31
                                                                     =");
32
              System.out.println("===
33
              dll.clear();
34
              dll.print();
35
              System.out.println("Size : "+dll.size());
```

#### 12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

#### 12.2.3 Pertanyaan Percobaan

- 1. Jelaskan perbedaan antara single linked list dengan double linked lists!
- 2. Perhatikan class Node, didalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
- 3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

```
public DoubleLinkedLists() {
   head = null;
   size = 0;
}
```

4. Pada method **addFirst()**, kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

```
Node newNode = new Node(null, item, head);
```

- 5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode?
- 6. Perhatikan isi method **addLast()**, apa arti dari pembuatan object Node dengan mengisikan parameter prev dengan current, dan next dengan null?

```
Node newNode = new Node(current, item, null);
```

### 12.3 Kegiatan Praktikum 2

### 12.3.1 Tahapan Percobaan

Pada praktikum 2 ini akan dibuat beberapa method untuk menghapus isi LinkedLists pada class DoubleLinkedLists. Penghapusan dilakukan dalam tiga cara di bagian paling depan, paling belakang, dan sesuai indeks yang ditentukan pada linkedLists. Method tambahan tersebut akan ditambahkan sesuai pada diagram class berikut ini.

DoubleLinkedLists
head: Node
size : int
DoubleLinkedLists()
isEmpty(): boolean
addFirst (): void
addLast(): void
add(item: int, index:int): void
size(): int
clear(): void
print(): void
removeFirst(): void
removeLast(): void
remove(index:int):void

1. Buatlah method removeFirst() di dalam class DoubleLinkedLists.

```
public void removeFirst() throws Exception {
   if (isEmpty()) {
      throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
   } else if (size == 1) {
      removeLast();
   } else {
      head = head.next;
      head.prev = null;
      size--;
   }
}
```

2. Tambahkan method removeLast() di dalam class DoubleLinkedLists.

```
public void removeLast() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List masih kosong, tidak dapat dihapus!");
    } else if (head.next == null) {
        head = null;
        size--;
        return;
    }
    Node current = head;
    while (current.next.next != null) {
        current = current.next;
    }
    current.next = null;
    size--;
}
```

3. Tambahkan pula method remove(int index) pada class DoubleLinkedLists dan amati hasilnya.

```
public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception ("Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
       Node current = head;
       int i = 0;
        while (i < index) {
           current = current.next;
            i++;
        if (current.next == null) {
            current.prev.next = null;
        } else if (current.prev == null) {
            current = current.next;
            current.prev = null;
           head = current;
        } else {
            current.prev.next = current.next;
            current.next.prev = current.prev;
        size--;
```

4. Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada **main class.** 

```
42
            dll.addLast(50);
43
            dll.addLast(40);
            dll.addLast(10);
44
45
            dll.addLast(20);
46
            dll.print();
            System.out.println("Size : "+dll.size());
47
            System.out.println("=======
48
            dll.removeFirst();
49
50
            dll.print();
51
            System.out.println("Size : "+dll.size());
            System.out.println("======"");
52
53
            dll.removeLast();
54
            dll.print();
            System.out.println("Size : "+dll.size());
55
            System.out.println("======"");
56
            dll.remove(1);
57
            dll.print();
58
            System.out.println("Size : "+dll.size());
59
```

#### 12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

#### 12.3.3 Pertanyaan Percobaan

Apakah maksud statement berikut pada method removeFirst()?

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

- 2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
- 3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah **remove!**

```
Node tmp = head.next;
head.next=tmp.next;
tmp.next.prev=head;
```

4. Jelaskan fungsi kode program berikut ini pada fungsi **remove!** 

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

## 12.4 Kegiatan Praktikum 3

#### 12.4.1 Tahapan Percobaan

Pada praktikum 3 ini dilakukan uji coba untuk mengambil data pada linked list dalam 3 kondisi, yaitu mengambil data paling awal, paling akhir dan data pada indeks tertentu dalam linked list. Method mengambil data dinamakan dengan **get**. Ada 3 method get yang dibuat pada praktikum ini sesuai dengan diagram class DoubleLinkedLists.

DoubleLinkedLists
head: Node
size : int
DoubleLinkedLists()
isEmpty(): boolean
addFirst (): void
addLast(): void
add(item: int, index:int): void
size(): int
clear(): void
print(): void
removeFirst(): void
removeLast(): void
remove(index:int):void
getFirst(): int
getLast(): int
get(index:int): int

1. Buatlah method **getFirst()** di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists.

```
public int getFirst() throws Exception {
   if (isEmpty()) {
      throw new Exception("Linked List kosong");
   }
   return head.data;
}
```

2. Selanjutnya, buatlah method **getLast()** untuk mendapat data pada akhir linked lists.

```
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List kosong");
    }
    Node tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

3. Method get(int index) di buat untuk mendapatkan data pada indeks tertentu

```
public int get(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception("Nilai indeks di luar batas.");
    }
    Node tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}</pre>
```

4. Pada main class tambahkan potongan program berikut dan amati hasilnya!

```
dll.print();
System.out.println("Size: " + dll.size());
System.out.println("======");
dll.addFirst(3);
dll.addLast(4);
dll.addFirst(7);
dll.print();
System.out.println("Size: " + dll.size());
System.out.println("======");
dll.add(40, 1);
dll.print();
System.out.println("Size: " + dll.size());
System.out.println("======");
System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());
System.out.println("Data akhir pada Linked Lists adalah: " + dll.getLast());
System.out.println("Data indeks ke-1 pada Linked Lists adalah: " + dll.get(1));
```

#### 12.4.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

#### 12.4.3 Pertanyaan Percobaan

- 1. Jelaskan method size() pada class DoubleLinkedLists!
- 2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!
- 3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!
- 4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

```
public boolean isEmpty(){
  if(size ==0){
    return true;
  } else(
    return false;
}

(a)

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

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

#### 12.5 Tugas Praktikum

1. Buatlah sebuah program menggunakan double linked lists dengan pilihan menu sesuai dengan ilustrasi di bawah ini! Fitur pencarian harus menggunakan sequential search dan fitur pengurutan secara Descending mengimplementasikan (silakan pilih satu): bubble sort, selection sort, insertion sort, atau merge sort.

### Contoh ilustrasi:

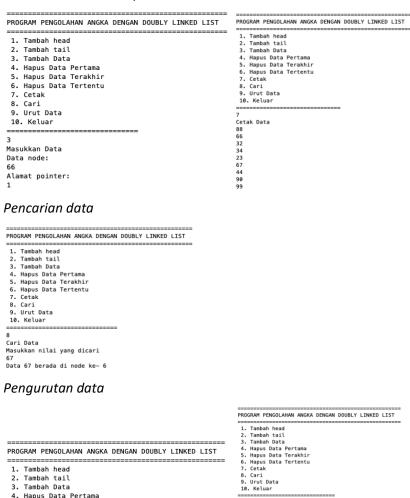
Menu awal dan tambah data

```
PROGRAM PENGOLAHAN ANGKA DENGAN DOUBLY LINKED LIST
PROGRAM PENGOLAHAN ANGKA DENGAN DOUBLY LINKED LIST
1. Tambah head
                                                                2. Tambah tail
2. Tambah tail
                                                                3. Tambah Data
                                                                4. Hapus Data Pertama
3. Tambah Data

    Hapus Data Terakhir
    Hapus Data Tertentu

4. Hapus Data Pertama
5. Hapus Data Terakhir
                                                                7. Cetak
6. Hapus Data Tertentu
                                                                8. Cari
7. Cetak
                                                                9. Urut Data
8. Cari
9. Urut Data
10. Keluar
                                                              -Masukkan Data Posisi Head
```

#### Tambah data di node posisi tertentu dan cetak data



2. Implementasikan stack menggunakan double linked list dengan contoh kasus tumpukan buku perpustakaan sesuai dengan fitur-fitur yang ditunjukkan pada gambar di bawah ini!

# Ilustrasi Program

Hapus Data Pertama
 Hapus Data Terakhir

6. Hapus Data Tertentu

7. Cetak 8. Cari 9. Urut Data 10. Keluar

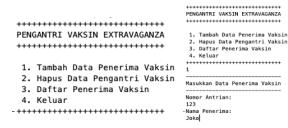
Menu Awal dan Tambah Data (Push)



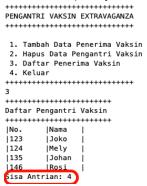
#### Cetak Semua Data Data Buku Perpustakaan 1. Entry Judul Buku 2. Ambil Buku Teratas 3. Cek Judul Buku Teratas 4. Info Semua Judul Buku Keluar \*\*\*\*\* Cetak Seluruh Judul Buku 3D Computer Vision Undesrstanding Software Algorithms Notes For Professionals Getting Started with C++ Audio Programming for Game Development Practical Digital Forensics Cek buku di tumpukan teratas Data Buku Perpustakaan 1. Entry Judul Buku 2. Ambil Buku Teratas 3. Cek Judul Buku Teratas 4. Info Semua Judul Buku 5. Keluar \* 3 Cek Buku Teratas 3D Computer Vision Hapus Buku Teratas Data Buku Perpustakaan Data Buku Perpustakaan Entry Judul Buku Ambil Buku Teratas Cek Judul Buku Teratas 1. Entry Judul Buku 2. Ambil Buku Teratas 3. Cek Judul Buku Teratas 4. Info Semua Judul Buku 5. Keluar 4. Info Semua Judul Buku 5. Keluar \*\*DOODOODOODOODOODOODOO Cetak Seluruh Judul Buku Undesrstanding Software Algorithms Notes For Professionals Buku pada tumpukan teratas telah diambil Getting Started with C++ Audio Programming for Game Development Practical Digital Forensics

 Buat program antrian vaksinasi menggunakan queue berbasis double linked list sesuai ilustrasi dan menu di bawah ini! (counter jumlah antrian tersisa di menu cetak(3) dan data orang yang telah divaksinasi di menu Hapus Data(2) harus ada) Ilustrasi Program

Menu Awal dan Penambahan Data



#### Cetak Data (Komponen di area merah harus ada)



#### Hapus Data (Komponen di area merah harus ada)



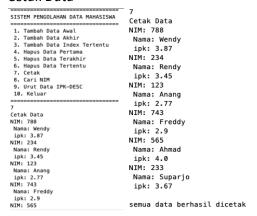
4. Buatlah implementasi program daftar nilai mahasiswa yang terdiri dari nim, nama dan ipk menggunakan double linked lists, bentuk program memiliki fitur seperti halnya nomor 1 dengan fitur pencarian melalui NIM dan pengurutan IPK secara descending. Class Mahasiswa wajib diimplementasikan dalam soal ini.

#### **Ilustrasi Program**

#### Menu Awal dan Penambahan Data



#### Cetak Data



#### Pencarian Data

#### SISTEM PENGOLAHAN DATA MAHASISWA

- 1. Tambah Data Awal
- 2. Tambah Data Akhir
- 3. Tambah Data Index Tertentu 4. Hapus Data Pertama
- 5. Hapus Data Terakhir
- 6. Hapus Data Tertentu
- 7. Cetak
- 8. Cari NIM 9. Urut Data IPK-DESC
- 10. Keluar

Cari Data Masukkan NIM yang dicari

565

Data 565 berada di node ke- 5

IDENTITAS: Nama: Ahmad IPK: 4.0

### Pengurutan Data

#### SISTEM PENGOLAHAN DATA MAHASISWA

- 1. Tambah Data Awal
- 2. Tambah Data Akhir
- 3. Tambah Data Index Tertentu
- 4. Hapus Data Pertama
- 5. Hapus Data Terakhir
- 6. Hapus Data Tertentu
- 7. Cetak
- 8. Cari NIM
- 9. Urut Data IPK-DESC
- 10. Keluar

9

SISTEM PENGOLAHAN DATA MAHASISWA

- 1. Tambah Data Awal
  2. Tambah Data Akhir
  3. Tambah Data Index Tertentu
  4. Hapus Data Pertama
  5. Hapus Data Terakhir
  6. Hapus Data Tertentu
  7. Cetak
  8. Cari NIM
  9. Urut Data IPK-DESC
  10. Keluar

- 7
  Cetak Data
  NIM: 123
  Nama: Anang
  ipk: 2.77
  NIM: 743
  Nama: Freddy
  ipk: 2.9
  NIM: 234
  Nama: Rendy
  ipk: 3.45
  NIM: 233
  Nama: Suparjo
  ipk: 3.67
  NIM: 788

Cetak Data NIM: 123

Nama: Anang ipk: 2.77

NIM: 743

Nama: Freddy

ipk: 2.9

NIM: 234

Nama: Rendy

ipk: 3.45

NIM: 233

Nama: Suparjo

ipk: 3.67 NIM: 788

Nama: Wendy

ipk: 3.87 NIM: 565

Nama: Ahmad

ipk: 4.0

\_\_\_ \*\*\* \_\_\_