



JOBSHEET 10

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

Waktu : 90 Menit

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 {
```

```
package DoubleLinkedList;
```

```
public class Node{
```

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

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

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

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

```
7      Node(Node prev, int data, Node next){  
8          this.prev=prev;  
9          this.data=data;  
10         this.next=next;  
11     }  
12 }
```

```
Node (Node prev,int data , Node next){  
    this.data = data;  
    this.prev = prev;  
    this.next = next;
```

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

```
package doublelinkedlists;
```

```
/**...4 lines */
```

```
public class DoubleLinkedLists {
```

```
package DoubleLinkedList;
```

```
public class DoubleLinkedList {
```

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

```
8      Node head;
9      int size;
```

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

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

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

```
DoubleLinkedList(){
    head = null;
    size = 0;
}
```

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

```
16 public boolean isEmpty(){
17     return head == null;
18 }
```

```
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++;
}
```

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

```
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, next:null);
        current.next = newNode;
        size++;
    }
}
```

12. Untuk menambahkan 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++;
        }
        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++;
}

```

```

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

        if (current.prev == null) {
            Node newNode = new Node(prev:null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            Node newNode = new Node(current.prev, item, current);
            newNode.prev.next = newNode;
            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  }
```

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

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  }
```

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

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");
    }
}
```

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

16. Selanjutnya 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) {

    }
}
```

```
package DoubleLinkedList;

public class DoubleLinkedListsMain {
    Run | Debug
    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());
22 System.out.println("=====");
23 dll.addFirst(3);
24 dll.addLast(4);
25 dll.addFirst(7);
26 dll.print();
27 System.out.println("Size : "+dll.size());
28 System.out.println("=====");
29 dll.add(40, 1);
30 dll.print();
31 System.out.println("Size : "+dll.size());
32 System.out.println("=====");
33 dll.clear();
34 dll.print();
35 System.out.println("Size : "+dll.size());

```

```

DoubleLinkedList dll = new DoubleLinkedList();
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====");
dll.addFirst(item:3);
dll.addLast(item:4);
dll.addFirst(item:7);
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====");

try {
    dll.add(item:40, index:1);
} catch (Exception e) {
    System.out.println(e.getMessage());
}

dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"=====");
dll.clear();
dll.print();
System.out.println("Size : " + dll.size());
}
}

```


12.2.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```

--- exec-maven-plugin:1.5.0:exec
Linked Lists Kosong
Size: 0
=====
7      3      4
berhasil diisi
Size: 3
=====
7      40      3      4
berhasil diisi
Size: 4
=====
Linked Lists Kosong
Size: 0
=====
-----
BUILD SUCCESS
-----

```

```

Linked List Kosong
Size : 0
=====
7      3      4
berhasil diisi
Size : 3
=====
7      40      3      4
berhasil diisi
Size : 4
=====
Linked List Kosong
Size : 0

```

12.2.3 Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
Single linked list setiap node memiliki satu pointer yang menunjuk ke node berikutnya. Sedangkan Double Linked List Setiap node memiliki dua pointer, yaitu satu yang menunjuk ke node sebelumnya (prev) dan satu lagi yang menunjuk ke node berikutnya (next).
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
Atribut next: Menyimpan referensi ke node berikutnya dalam linked list.
Atribut prev: Menyimpan referensi ke node sebelumnya dalam linked list.

3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

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

head = null; : Menginisialisasi linked list kosong dengan tidak ada node awal.

size = 0; : Menginisialisasi ukuran linked list dengan 0 elemen.

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

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

Saat menambahkan node baru di awal linked list (addFirst()), tidak ada node sebelumnya untuk node baru pertama, sehingga prev diatur sebagai null.

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

Statement ini menetapkan bahwa node sebelumnya dari node yang sebelumnya adalah newNode. Dalam kasus ini, head sebelumnya tidak ada atau null, jadi head.prev disetel ke newNode.

6. Perhatikan isi method **addLast()**, apa arti dari pembuatan object Node dengan mengisi parameter prev dengan current, dan next dengan null?

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

Ini berarti node baru (newNode) ditambahkan di akhir linked list. prev=current mengatur node sebelumnya dari newNode menjadi node yang sekarang menjadi tail. next=null menandakan bahwa newNode adalah node terakhir, sehingga tidak ada node berikutnya setelahnya.

7. Pada method **add()**, terdapat potongan kode program sebagai berikut:

```
while (i < index) {
    current = current.next;
    i++;
}

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;
}
```

jelaskan maksud dari bagian yang ditandai dengan kotak kuning.

Kode ini mengecek apakah current adalah node pertama dalam linked list. Jika iya (current.prev == null), artinya current adalah head dari linked list. Selanjutnya, kode tersebut membuat newNode sebagai node baru yang ditempatkan sebelum current, dan kemudian menetapkan current.prev menjadi newNode. Terakhir, head diupdate menjadi newNode, karena newNode sekarang adalah head baru dari linked list.

12.3 Kegiatan Praktikum 2

Waktu : 60 Menit

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--;
    }
}
```

```
public void removefirst () throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"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--;
}
```

```
void removelast () throws Exception{
    if (isEmpty()) {
        throw new Exception (message:"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--;
    }
}
```

```
void remove (int index)throws Exception{
    if (isEmpty()|| index >= size) {
        throw new Exception (message:"Nilai indeks diluar 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);
44      dll.addLast(10);
45      dll.addLast(20);
46      dll.print();
47      System.out.println("Size : "+dll.size());
48      System.out.println("=====");
49      dll.removeFirst();
50      dll.print();
51      System.out.println("Size : "+dll.size());
52      System.out.println("=====");
53      dll.removeLast();
54      dll.print();
55      System.out.println("Size : "+dll.size());
56      System.out.println("=====");
57      dll.remove(1);
58      dll.print();
59      System.out.println("Size : "+dll.size());

```

```

dll.addLast(item:50);
dll.addLast(item:40);
dll.addLast(item:10);
dll.addLast(item:20);
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"=====");

try {
    dll.removefirst();
} catch (Exception e) {
    e.printStackTrace();
}
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"=====");

try {
    dll.removelast();
} catch (Exception e) {
    e.printStackTrace();
}
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"=====");

try {
    dll.remove(index:1);
} catch (Exception e) {
    e.printStackTrace();
}
dll.print();
System.out.println("Size: " + dll.size());
}

```

12.3.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec
50      40      10      20
berhasil diisi
Size: 4
=====
40      10      20
berhasil diisi
Size: 3
=====
40      10
berhasil diisi
Size: 2
=====
40
berhasil diisi
Size: 1
-----
BUILD SUCCESS
-----
```

```
=====
50      40      10      20
berhasil diisi
Size: 4
=====
40      10      20
berhasil diisi
Size: 3
=====
40      10
berhasil diisi
Size: 2
=====
40
berhasil diisi
Size: 1
```

12.3.3 Pertanyaan Percobaan

1. Apakah maksud statement berikut pada method **removeFirst()**?
`head = head.next;`
`head.prev = null;`

`head = head.next;` Statement ini menggeser head ke node berikutnya. Node yang sebelumnya menjadi head akan dilewati dan dihapus dari linked list.

`head.prev = null;` Setelah head bergeser ke node berikutnya, statement ini mengatur pointer prev dari node baru yang menjadi head ke null, karena sekarang node ini adalah node pertama dalam linked list dan tidak memiliki node sebelumnya.



2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method **removeLast()**?

Kita harus mengiterasi melalui linked list hingga mencapai node terakhir. Biasanya, node terakhir adalah node yang memiliki next bernilai null

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;
```

tidak cocok untuk perintah remove karena tidak memeriksa kondisi jika daftar kosong atau hanya memiliki satu elemen. Selain itu, kode tersebut dapat menyebabkan NullPointerException jika tmp.next adalah null.

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

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

Fungsi dari kedua statement ini adalah untuk menghapus node `current` dari linked list dengan menghubungkan node sebelum `current` langsung ke node setelah `current`, sehingga node `current` tidak lagi terhubung dalam linked list.

12.4 Kegiatan Praktikum 3

Waktu : 50 Menit

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;
}
```

```
int getFirst()throws Exception{
    if (isEmpty()) {
        throw new Exception(message:"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;
}
```

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

3. Method **get(int index)** dibuat 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;
}
```

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

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));

dll.addFirst(data:3);
dll.addLast(data:4);
dll.addFirst(data:7);
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"=====");

try {
    dll.add(data:40, index:1);
    dll.print();
    System.out.println("Size: " + dll.size());
} catch (Exception e) {
    System.out.println(e.getMessage());
}

System.out.println(x:"=====");

try {
    System.out.println("Data awal pada Linked Lists adalah : " + dll.getFirst());
} catch (Exception e) {
    System.out.println(e.getMessage());
}

try {
    System.out.println("Data akhir pada Linked Lists adalah : " + dll.getLast());
} catch (Exception e) {
    System.out.println(e.getMessage());
}

try {
    System.out.println("Data indeks ke-1 pada Linked Lists adalah : " + dll.get(index:1));
} catch (Exception e) {
    System.out.println(e.getMessage());
}
```

12.4.2 Verifikasi Hasil Percobaan

Verifikasi hasil kompilasi kode program Anda dengan gambar berikut ini.

```
--- exec-maven-plugin:1.5.0:exec (default-cli)
Linked Lists Kosong
Size: 0
=====
7      3      4
berhasil diisi
Size: 3
=====
7      40     3      4
berhasil diisi
Size: 4
=====
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
-----
BUILD SUCCESS
-----
```

```
Size: 0
=====
7      3      4
berhasil diisi
Size: 3
=====
7      40     3      4
berhasil diisi
Size: 4
=====
Data awal pada Linked Lists adalah :7
Data akhir pada Linked Lists adalah :4
Data indeks ke-1 pada Linked Lists adalah :40
```

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 ke-1!
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;
}
```

(b)



12.5 Tugas Praktikum

Waktu : 100 Menit

1. 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)

Contoh Ilustrasi Program

Menu Awal dan Penambahan Data

```

+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
- +++++
    
```

```

+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
1
-----
Masukkan Data Penerima Vaksin
-----
Nomor Antrian:
123
-Nama Penerima:
Joko|
    
```

Cetak Data (Komponen di area merah harus ada)

```
+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++
```

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

```
+++++
```

3

```
+++++
```

Daftar Pengantri Vaksin

```
+++++
```

No.	Nama
123	Joko
124	Mely
135	Johan
146	Rosi

Sisa Antrian: 4

Hapus Data (Komponen di area merah harus ada)

```
+++++
PENGANTRI VAKSIN EXTRAVAGANZA
+++++
```

1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar

```
+++++
```

2

Joko telah selesai divaksinasi.

```
+++++
```

Daftar Pengantri Vaksin

```
+++++
```

No.	Nama
124	Mely
135	Johan
146	Rosi

Sisa Antrian: 3

DoubleLinkedList26:

```

1  package TugasPraktikum1;
2
3  public class DoubleLinkedList26 {
4      Node head;
5      int size;
6
7      public DoubleLinkedList26() {
8          head = null;
9          size = 0;
10     }
11
12     public boolean isEmpty() {
13         return head == null;
14     }
15
16     public void addFirst(String item, int nomorData) {
17         if (isEmpty()) {
18             head = new Node(prev:null, item, next:null, nomorData);
19         } else {
20             Node newNode = new Node(prev:null, item, head, nomorData);
21             head.prev = newNode;
22             head = newNode;
23         }
24         size++;
25     }
26
27     public void addLast(String item, int nomorData) {
28         if (isEmpty()) {
29             addFirst(item, nomorData);
30         } else {
31             Node current = head;
32             while (current.next != null) {
33                 current = current.next;
34             }
35             Node newNode = new Node(current, item, next:null, nomorData);
36             current.next = newNode;
37             size++;
38         }
39     }
40
41     public void add(String item, int index, int nomorData) throws Exception {
42         if (isEmpty()) {
43             addFirst(item, nomorData);
44         } else if (index < 0 || index > size) {
45             throw new Exception(message:"Nilai indeks di luar batas");
46         } else {
47             Node current = head;
48             int i = 0;
49             while (i < index) {
50                 current = current.next;
51                 i++;
52             }
53             if (current.prev == null) {
54                 Node newNode = new Node(prev:null, item, current, nomorData);
55                 current.prev = newNode;
56                 head = newNode;
57             } else {
58                 Node newNode = new Node(current.prev, item, current, nomorData);
59                 newNode.prev = current.prev;
60                 newNode.next = current;
61                 current.prev.next = newNode;
62                 current.prev = newNode;
63             }
64             size++;
65         }
66     }
67
68     public int size() {
69         return size;
70     }
71
72     public void clear() {
73         head = null;
74         size = 0;
75     }
76
77     public void print() {

```

```

78     if (!isEmpty()) {
79         Node tmp = head;
80         int count = 0;
81         System.out.println(x:"Daftar Pengantri Vaksin");
82         System.out.println(x:"+++++");
83         System.out.println(x:"| NO | NAMA ");
84         while (tmp.next != null) {
85             tmp = tmp.next;
86         }
87         while (tmp != null) {
88             System.out.println("| " + tmp.nomorData + " | " + tmp.data);
89             tmp = tmp.prev;
90             count++;
91         }
92
93         System.out.println("Sisa Antrian : " + count);
94     } else {
95         System.out.println(x:"Linked List Kosong");
96     }
97 }
98
99
100 public void removeFirst() throws Exception {
101     if (isEmpty()) {
102         throw new Exception(message:"Linked list masih kosong, tidak dapat dihapus!");
103     } else {
104         Node current = head;
105         Node minNode = head;
106         while (current != null) {
107             if (current.nomorData < minNode.nomorData) {
108                 minNode = current;
109             }
110             current = current.next;
111         }
112         if (minNode.prev == null) {
113             head = minNode.next;
114             if (head != null) {
115                 head.prev = null;
116             }
117         } else {
118             minNode.prev.next = minNode.next;
119             if (minNode.next != null) {
120                 minNode.next.prev = minNode.prev;
121             }
122         }
123         System.out.println(minNode.data + " telah selesai divaksinasi");
124         size--;
125     }
126 }
127
128
129
130 public void removeLast() throws Exception {
131     if (isEmpty()) {
132         throw new Exception(message:"Linked list masih kosong tidak dapat dihapus!");
133     } else if (head.next == null) {
134         head = null;
135         size--;
136         return;
137     }
138     Node current = head;
139     while (current.next.next != null) {
140         current = current.next;
141     }
142     current.next = null;
143     size--;
144 }
145
146 public void remove(int index) throws Exception {
147     if (isEmpty() || index >= size) {
148         throw new Exception(message:"Nilai indeks diluar batas");
149     } else if (index == 0) {
150         removeFirst();

```




```

151         } else {
152             Node current = head;
153             int i = 0;
154             while (i < index) {
155                 current = current.next;
156                 i++;
157             }
158             if (current.next == null) {
159                 current.prev.next = null;
160             } else if (current.prev == null) {
161                 current = current.next;
162                 current.prev = null;
163                 head = current;
164             } else {
165                 current.prev.next = current.next;
166                 current.next.prev = current.prev;
167             }
168             size--;
169         }
170     }
171 }
172
173 class Node {
174     Node prev;
175     String data;
176     Node next;
177     int nomorData;
178
179     Node(Node prev, String data, Node next, int nomorData) {
180         this.prev = prev;
181         this.data = data;
182         this.next = next;
183         this.nomorData = nomorData;
184     }
185 }
186

```

DoubleLinkedListsMain26:

```

1  package TugasPraktikum1;
2
3  import java.util.Scanner;
4
5  public class DoubleLinkedListsMain26 {
6
7      public static class tugasDLL {
8          Run | Debug
9          public static void main(String[] args) throws Exception {
10              DoubleLinkedList26 dll = new DoubleLinkedList26();
11              Scanner sc = new Scanner(System.in);
12              while (true) {
13                  System.out.println(x:"+++++++");
14                  System.out.println(x:"PENGANTRI VAKSIN EXTRAVAGANZA");
15                  System.out.println(x:"+++++++");
16                  System.out.println(x:"1. Tambah Data Penerima Vaksin");
17                  System.out.println(x:"2. Hapus Data Pengantri Vaksin");
18                  System.out.println(x:"3. Daftar Penerima Vaksin");
19                  System.out.println(x:"4. Keluar");
20                  System.out.println(x:"+++++++");
21                  int x = sc.nextInt();
22                  sc.nextLine();
23                  switch (x) {
24                      case 1:
25                          System.out.println(x:"-----");
26                          System.out.println(x:"Masukkan Data Penerima Vaksin");
27                          System.out.println(x:"-----");
28                          System.out.print(s:"Nomor Antrian : ");
29                          int nomorAntrian = sc.nextInt();
30                          sc.nextLine();
31                          System.out.print(s:"Nama Penerima : ");
32                          String namaPenerima = sc.nextLine();
33                          dll.addFirst(namaPenerima, nomorAntrian);
34                          break;
35                      case 2:
36                          if (dll.isEmpty()) {
37                              System.out.println(x:"Tidak ada data yang bisa dihapus karena antrian kosong.");
38                          } else {
39                              dll.removeFirst();
40                          }
41                          break;
42                      case 3:
43                          dll.print();
44                          break;
45                      case 4:
46                          sc.close();
47                          System.exit(status:0);
48                          break;
49                      default:
50                          break;
51                  }
52              }
53          }
54      }
55  }
56

```

Node26:

```

1  package TugasPraktikum1;
2
3  public class Node26 {
4      int nomorData;
5      String data;
6      Node26 prev, next;
7
8      Node26(Node26 prev, String data, Node26 next, int nomorData){
9          this.prev=prev;
10         this.data=data;
11         this.next=next;
12         this.nomorData=nomorData;
13     }
14 }

```

2. Buatlah program daftar film yang terdiri dari id, judul dan rating menggunakan double linked lists, bentuk program memiliki fitur pencarian melalui ID Film dan pengurutan Rating secara descending. Class Film wajib diimplementasikan dalam soal ini.

Contoh Ilustrasi Program

Menu Awal dan Penambahan Data

```

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====

```

```

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
1
Masukkan Data Film Posisi Awal
ID Film:
1222
Judul Film:
Spider-Man: No Way Home
Rating Film:
8.7

```



```

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
2
Masukkan Data Posisi Akhir
ID Film:
1346
Judul Film:
Uncharted
Rating Film:
6.7

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
3
Masukkan Data Film
Urutan ke-
ID Film:
1234
Judul Film:
Death on the Nile
Rating Film:
6.6
Data Film ini akan masuk di urutan ke-
3

```

Cetak Data

```

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
7
Cetak Data
ID: 1222
Judul Film: Spider-Man: No Way Home
ipk: 8.7
ID: 1765
Judul Film: Skyfall
ipk: 7.8
ID: 1567
Judul Film: The Dark Knight Rises
ipk: 8.4
ID: 1234
Judul Film: Death on The Nile
ipk: 6.6
ID: 1346
Judul Film: Uncharted
ipk: 6.7

```

Pencarian Data

```

=====
DATA FILM LAYAR LEBAR
=====
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 ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
8
Cari Data
Masukkan ID Film yang dicari
1567
Data Id Film: 1567 berada di node ke- 3
IDENTITAS:
ID Film: 1567
Judul Film: The Dark Knight Rises
IMDB Rating: 8.4

```

--- *** ---

DoubleLinkedList26:

```

1 package TugasPraktikum2;
2
3 public class DoubleLinkedList26 {
4
5     Film26 head, tail;
6
7     public void addAtBeginning(int id, String title, double rating) {
8         Film26 newFilm = new Film26(id, title, rating);
9         if (head == null) {
10             head = tail = newFilm;
11         } else {
12             newFilm.next = head;
13             head.prev = newFilm;
14             head = newFilm;
15         }
16     }
17
18     public void addAtEnd(int id, String title, double rating) {
19         Film26 newFilm = new Film26(id, title, rating);
20         if (head == null) {
21             head = tail = newFilm;
22         } else if (head == tail) {
23             tail = newFilm;
24             head.next = tail;
25             tail.prev = head;
26         } else {
27             tail.next = newFilm;
28             newFilm.prev = tail;
29             tail = newFilm;
30         }
31     }
32
33
34     public void addAtIndex(int id, String title, double rating, int index) {
35         Film26 newFilm = new Film26(id, title, rating);
36         if (index == 0) {
37             addAtBeginning(id, title, rating);
38         } else {
39             Film26 temp = head;
40             int currentIndex = 0;
41             while (temp != null && currentIndex < index - 1) {
42                 temp = temp.next;
43                 currentIndex++;
44             }
45             if (temp != null) {
46                 newFilm.next = temp.next;
47                 newFilm.prev = temp;
48                 if (temp.next != null) {
49                     temp.next.prev = newFilm;
50                 }
51                 temp.next = newFilm;
52                 if (temp == tail) {
53                     tail = newFilm;
54                 }
55             } else {
56                 System.out.println("Posisi " + index + " tidak valid.");
57             }
58         }
59     }
60
61
62
63     public void deleteFromBeginning() {
64         if (head != null) {
65             if (head.next != null) {
66                 head = head.next;
67                 head.prev = null;
68             } else {
69                 head = tail = null;
70             }
71         }
72     }
73
74     public void deleteFromEnd() {
75         if (head != null) {
76             if (tail.prev != null) {

```

```

77         tail = tail.prev;
78         tail.next = null;
79     } else {
80         head = tail = null;
81     }
82 }
83 }
84
85 public void deleteAtIndex(int index) {
86     if (index == 0) {
87         deleteFromBeginning();
88     } else {
89         Film26 temp = head;
90         for (int i = 0; i < index && temp != null; i++) {
91             temp = temp.next;
92         }
93         if (temp != null && temp.next != null) {
94             temp.prev.next = temp.next;
95             temp.next.prev = temp.prev;
96         } else if (temp == tail) {
97             tail = temp.prev;
98             tail.next = null;
99         }
100     }
101 }
102
103 public void printList() {
104     Film26 temp = head;
105     while (temp != null) {
106         System.out.println("ID Film: " + temp.id);
107         System.out.println("Judul Film: " + temp.title);
108         System.out.println("Rating Film: " + temp.rating);
109         System.out.println();
110         temp = temp.next;
111     }
112 }
113
114 public void searchById(int id) {
115     Film26 temp = head;
116     while (temp != null) {
117         if (temp.id == id) {
118             System.out.println("ID Film: " + temp.id);
119             System.out.println("Judul Film: " + temp.title);
120             System.out.println("Rating Film: " + temp.rating);
121             return;
122         }
123         temp = temp.next;
124     }
125     System.out.println("Film dengan ID " + id + " tidak ditemukan.");
126 }
127
128 public void sortByRatingDesc() {
129     if (head != null) {
130         boolean swapped;
131         do {
132             swapped = false;
133             Film26 current = head;
134             while (current.next != null) {
135                 if (current.rating < current.next.rating) {
136                     int tempId = current.id;
137                     String tempTitle = current.title;
138                     double tempRating = current.rating;
139
140                     current.id = current.next.id;
141                     current.title = current.next.title;
142                     current.rating = current.next.rating;
143
144                     current.next.id = tempId;
145                     current.next.title = tempTitle;
146                     current.next.rating = tempRating;
147
148                     swapped = true;
149                 }
150                 current = current.next;

```

```

151         }
152     } while (swapped);
153 }
154 }
155 public static class Film26 {
156     int id;
157     String title;
158     double rating;
159     Film26 prev;
160     Film26 next;
161
162     public Film26(int id, String title, double rating) {
163         this.id = id;
164         this.title = title;
165         this.rating = rating;
166         this.prev = null;
167         this.next = null;
168     }
169 }
170 }
171

```

Hasil :

```

=====
DATA FILM LAYAR LEBAR
=====
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Indeks Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar
=====
Pilihan: 1
Masukkan ID Film: 122
Masukkan Judul Film: Spidolman
Masukkan Rating Film: 9
=====

```

DATA FILM LAYAR LEBAR

=====

1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Indek Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar

=====

Pilihan: 2

Masukkan ID Film: 123

Masukkan Judul Film: Far From Saytan

Masukkan Rating Film: 10

=====

Pilihan: 3

Masukkan Id Film : 1234

Masukkan Judul Film: Far From home

Masukkan Rating Film: 5

Masukkan Posisi: 2

=====

DATA FILM LAYAR LEBAR

=====

1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Indek Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar

=====

Pilihan: 3

Masukkan Id Film : 12345

Masukkan Judul Film: Belajar Ngoding

Masukkan Rating Film: 7

Masukkan Posisi: 3

=====

DATA FILM LAYAR LEBAR

=====

1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Indek Tertentu
4. Hapus Data Pertama
5. Hapus Data Terakhir
6. Hapus Data Tertentu
7. Cetak
8. Cari ID Film
9. Urut Data Rating Film-DESC
10. Keluar

=====

Pilihan: 7

ID Film: 122

Judul Film: Spidolman

Rating Film: 9.0

ID Film: 123

Judul Film: Far From Saytan

Rating Film: 10.0

ID Film: 1234

Judul Film: Far From home

Rating Film: 5.0

ID Film: 12345

Judul Film: Belajar Ngoding

Rating Film: 7.0

=====