JOBSHEET 10
Double Linked Lists

Nama: Rizqi Bagus Andrean
Kelas: TI-1D
Absen: 25

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 {

}
```

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.

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

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.

```
8        Node head;
9        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.

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

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 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++;
}
```

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());
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());
```

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
------------------------------------
```

```
6bce766bc733\redhat.java\jdt_ws\jobsheet 11_887a23e
List is empty
Size: 0
7 3 4
Size: 3
7 40 3 4
Size: 4
List is empty
Size: 0
PS C:\Users\Acer\Tugas Kuliah\Semester 2\Praktek Al
```

12.2.3 Pertanyaan Percobaan

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

   Single Linked List (SLL) dan Double Linked List (DLL) sama-sama struktur data berurutan, namun berbeda dalam cara menyimpan dan menghubungkan data. SLL hanya memiliki

pointer "next" ke node selanjutnya, sedangkan DLL memiliki pointer "next" dan "prev" untuk navigasi dua arah. DLL lebih mudah untuk operasi di tengah list dan penelusuran dua arah, namun membutuhkan memori lebih banyak. SLL lebih hemat memori dan sederhana, tetapi operasi di tengah list lebih rumit dan tidak memungkinkan penelusuran dari akhir ke awal.

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

   Di dalam kelas DoubleLinkedList yang diberikan, class Node memiliki dua atribut penting: `next` dan `prev`. Atribut ini adalah referensi (penunjuk) ke node lain di dalam list. Atribut `next` menunjuk ke node berikutnya di dalam list, sementara atribut `prev` menunjuk ke node sebelumnya di dalam list. Penghubungan dua arah ini memungkinkan untuk menelusuri list dalam dua arah (maju dan mundur), yang merupakan fitur utama dari doubly 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;
}
```

inisialisasi `head` dan `size` dalam constructor mencerminkan kondisi awal list DoubleLinkedList yang baru dibuat, yaitu list kosong dengan ukuran 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);

Karena saat addFirt data ditambah pada index pertama dimana index sebelum pertama pasti tidak ada.

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

   Berarti sebelum head akan ditambah node baru sehingga headnya akan bergeser.
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**);

   Pembuatan object Node seperti ini dilakukan untuk menyisipkan node baru (`newNode`) di **akhir** list DoubleLinkedList. Node baru ini disisipkan **setelah** node yang ditunjuk oleh parameter `current`, dan `newNode` tidak memiliki node setelahnya (karena berada di akhir list)

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 itu sebenarnya sama dengan kode yang ada di mthod addFirst() yang digunakan untuk menambah node pada index pertama.

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--;
    }
}
```

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);
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());
```

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
---------------------------------
```

```
java.exe' '-cp' 'C:\Users\Acer\A
6bce766bc733\redhat.java\jdt_ws\
50 40 10 20
Size: 4
40 10 20
Size: 3
40 10
Size: 2
40
Size: 1
PS C:\Users\Acer\Tugas Kuliah\Se
```

12.3.3 Pertanyaan Percobaan

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

   head = head.next;

   head.prev = null;

Baris pertama `head = head.next` mengubah referensi `head` untuk menunjuk ke elemen selanjutnya (next) dari elemen yang saat ini menjadi head. Ini pada dasarnya memutuskan hubungan elemen head lama dari list.

Baris kedua `head.prev = null` menghapus referensi ke elemen sebelumnya (prev) dari elemen yang sekarang menjadi head baru. Elemen head yang baru sekarang tidak memiliki elemen sebelumnya karena memang menjadi elemen pertama dalam list.

Dengan kedua baris ini, elemen yang tadinya head berhasil dihapus dari list.

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

   Dengan melakukan perulangan smpai currentnya != nul

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;
```

Baris ini tidak cocok karena , ia akan menghilangkan elemen sebelumnya karena dipindahkan, sehingga akan kehilangan referensinya

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

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

Baris kode tersebut bekerja sama untuk menghapus elemen pada indeks yang ditentukan dari double linked list dengan memperbarui referensi `next` dan `prev` dari elemen yang berdekatan dengan elemen yang dihapus.

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;
}
```

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)** 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;
}
```

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.

```
--- 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
------------------------------------------------
```

```
List is empty
Size: 0
7 3 4
Size: 3
7 40 3 4
Size: 4
Data Awal: 7
Data Akhir: 4
Data ke-1: 40
```

12.4.3 Pertanyaan Percobaan

1. Jelaskan method **size()** pada class DoubleLinkedLists!

   Method size diugunakan untuk menampilkan size dari Node yang tersimpan;
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!

   SLL: Bisa menyisipkan di depan, tengah, dan belakang dengan rata-rata O(1) untuk depan dan belakang, dan O(n) untuk tengah, di mana n adalah jumlah elemen. Membutuhkan 1 pointer memori. Cocok untuk penyisipan di depan/belakang yang sering.
   DLL: Bisa menyisipkan di depan, tengah, belakang, dan sebelum dengan rata-rata O(1) di semua posisi. Membutuhkan 2 pointer memori. Cocok untuk navigasi bolak-balik dan penyisipan/penghapusan di sembarang posisi.
4. Jelaskan perbedaan logika dari kedua kode program di bawah ini!

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

(a)

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

(b)

a. Program ini mengecek apakah size nya itu == 0, jika iya maka return true, dan jika tidak maka false
b. Program ini sebenarnya mirip, Cuma dia mengecek apakah ada data dinhead atau tidak.

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
```

```
t_ws\jobsheet 11_887a23ee\bin' 'vaksin.DLL'
Penerima Vaksin Joko telah ditambahkan
Penerima Vaksin Mely telah ditambahkan
Penerima Vaksin Johan telah ditambahkan
Penerima Vaksin Rosi telah ditambahkan

List Penerima Vaksin
123 Joko
124 Mely
135 Johan
146 Rosi

Size: 4

Joko telah dihapus

List Penerima Vaksin
124 Mely
135 Johan
146 Rosi

Size: 3
```

DLL.java

```
package vaksin;

public class DLL {
    Node head;
    int size;

    DLL() {
        head = null;
        size = 0;
    }

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

    void addFirst(Orang data) {
        Node newNode = new Node(null, data, head);
        if (!isEmpty()) {
            head.prev = newNode;
        }
        head = newNode;
        size++;
    }

    void addLast(Orang data) {
        if (isEmpty()) {
            addFirst(data);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, data, null);
            current.next = newNode;
            size++;
        }
        System.out.println("Penerima Vaksin " + data.nama + " telah ditambahkan");
    }

    int size() {
        return size;
    }

    void print() {
        if (!isEmpty()) {
            Node current = head;
            while (current != null) {
                System.out.println(current.data.no + " " + current.data.nama );
                current = current.next;
```

```
            }
            System.out.println();
        } else {
            System.out.println("List is empty");
        }
    }

    void removeFirst() {
        if (isEmpty()) {
            System.out.println("List is empty");
        } else if (size == 1) {
            removeLast();
        } else {
            System.out.println(head.data.nama + " telah dihapus");
            head = head.next;
            head.prev = null;
            size--;
        }
    }

    void removeLast() {
        if (isEmpty()) {
            System.out.println("List is empty");
        } else if (size == 1) {
            System.out.println(head.data.nama + " telah dihapus");

            head = null;
            size--;
        } else {
            System.out.println(head.data.nama + " telah dihapus");
            Node current = head;
            while (current.next.next != null) {
                current = current.next;
            }
            current.next = null;
        }
    }


    public static void main(String[] args) {
        DLL dll = new DLL();

        // Tambah Penerima Vaksin
        dll.addLast(new Orang(123, "Joko"));
        dll.addLast(new Orang(124, "Mely"));
        dll.addLast(new Orang(135, "Johan"));
        dll.addLast(new Orang(146, "Rosi"));
        System.out.println();

        // Tampilkan List Penerima Vaksin
```

```
        System.out.println("List Penerima Vaksin");
        dll.print();
        System.out.println("Size: " + dll.size());

        System.out.println();

        // Hapus Penerima Vaksin
        dll.removeFirst();

        System.out.println();
        // Tampilkan List Penerima Vaksin
        System.out.println("List Penerima Vaksin");
        dll.print();
        System.out.println("Size: " + dll.size());
    }
}
```

Orang.java

```
package vaksin;

public class Orang {
    int no;
    String nama;

    public Orang(int no, String nama) {
        this.no = no;
        this.nama = nama;
    }
}
```

Node.java

```
package vaksin;

class Node {
    Orang data;
    Node next, prev;

    Node(Node prev, Orang data, Node next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }

    Node(){}
}
```

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
```

--- *** ---

Sort By Desc

```
Sebelum Sort
1 The Shawshank Redemption 8.3
4 The Lord of the Rings: The Return of the King 7.0
2 The Godfather 9.2
3 The Dark Knight 3.0
5 The Lord of the Rings: The Return of the King 6.0


Sort By Rating Descending
2 The Godfather 9.2
1 The Shawshank Redemption 8.3
4 The Lord of the Rings: The Return of the King 7.0
5 The Lord of the Rings: The Return of the King 6.0
3 The Dark Knight 3.0
```

Test Menu Lain

```
ing\Code\User\workspaceStorage\bbe78b3de0f50c61a9286bce766bc733\redhat.java\j
dt_ws\jobsheet 11_887a23ee\bin' 'film.DLL'
Menambahkan Film
Film The Shawshank Redemption telah ditambahkan
1 The Shawshank Redemption 8.3


Menambahkan Film Pada Index Akhir
Film The Godfather telah ditambahkan
Film The Dark Knight telah ditambahkan
Film The Lord of the Rings: The Return of the King telah ditambahkan
1 The Shawshank Redemption 8.3
2 The Godfather 9.2
3 The Dark Knight 3.0
5 The Lord of the Rings: The Return of the King 6.0


Menambahkan Film Pada Index 1
Film The Lord of the Rings: The Return of the King telah ditambahkan pada ind
ex 1
1 The Shawshank Redemption 8.3
4 The Lord of the Rings: The Return of the King 7.0
2 The Godfather 9.2
3 The Dark Knight 3.0
5 The Lord of the Rings: The Return of the King 6.0


Menghapus Film Pada Index Awal
The Shawshank Redemption telah dihapus
4 The Lord of the Rings: The Return of the King 7.0
2 The Godfather 9.2
3 The Dark Knight 3.0
5 The Lord of the Rings: The Return of the King 6.0


Menghapus Film Pada Index Akhir
The Lord of the Rings: The Return of the King telah dihapus
4 The Lord of the Rings: The Return of the King 7.0
2 The Godfather 9.2
3 The Dark Knight 3.0


Menghapus Film Pada Index 2
The Lord of the Rings: The Return of the King telah dihapus

Mencari Film Berdasarkan ID = 4
The Lord of the Rings: The Return of the King

PS C:\Users\Acer\Tugas Kuliah\Semester 2\Praktek Algoritma\jobsheet 11>
```

DLL.java

```
package film;

public class DLL {
    Node head;
    int size;

    DLL() {
        head = null;
        size = 0;
    }

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

    void addFirst(Film data) {
        Node newNode = new Node(null, data, head);
        if (!isEmpty()) {
            head.prev = newNode;
        }
        head = newNode;
        size++;

        System.out.println("Film " + data.judul + " telah ditambahkan");
    }

    void addLast(Film data) {
        if (isEmpty()) {
            addFirst(data);
        } else {
            Node current = head;
            while (current.next != null) {
                current = current.next;
            }
            Node newNode = new Node(current, data, null);
            current.next = newNode;
            size++;
        }
        System.out.println("Film " + data.judul + " telah ditambahkan");
    }

    void add(Film data, int index) {
        if (index < 0 || index > size) {
            System.out.println("Index out of bound");
        } else if (index == 0) {
            addFirst(data);
        } else {
            Node current = head;
            for (int i = 0; i < index - 1; i++) {
                current = current.next;
            }
            Node newNode = new Node(current, data, current.next);
```

```
            current.next = newNode;
            current.next.prev = newNode;
            size++;
        }

        System.out.println("Film " + data.judul + " telah ditambahkan pada index " + index);

    }

    void print() {
        if (!isEmpty()) {
            Node current = head;
            while (current != null) {
                System.out.println(current.data.id + " " + current.data.judul + " " +
current.data.rating);
                current = current.next;
            }
            System.out.println();
        } else {
            System.out.println("List is empty");
        }
        System.out.println();
    }

    void removeFirst() {
        if (isEmpty()) {
            System.out.println("List is empty");
        } else if (size == 1) {
           removeLast();
        } else {
            System.out.println(head.data.judul + " telah dihapus");
            head = head.next;
            head.prev = null;
            size--;
        }

    }

    void removeLast() {
        if (isEmpty()) {
            System.out.println("List is empty");
        } else if (size == 1) {
            System.out.println(head.data.judul + " telah dihapus");
            head = null;
            size--;
        } else {
            System.out.println(head.data.judul + " telah dihapus");
            Node current = head;
            while (current.next.next != null) {
                current = current.next;
            }
            current.next = null;
```

```
            size--;
        }
    }

    void remove(int index) {
        if (isEmpty()|| index >= size) {
            System.out.println("Index out of bound");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node current = head;
            int i = 0;
            while (i < index - 1) {
                current = current.next;
                i++;
            }

            if (current.next.next == null) {
                removeLast();
            } else if(current.prev == null) {
                removeFirst();
            } else {
                current.next = current.next.next;
                current.next.prev = current;
                size--;
            }
        }
    }

    Film searchFilm(int id) {
        Node current = head;
        while (current != null) {
            if (current.data.id == id) {
                return current.data;
            }
            current = current.next;
        }
        return null;
    }

    void sortByDesc() {
        Node current = head;
        while (current != null) {
            Node next = current.next;
            while (next != null) {
                if (current.data.rating < next.data.rating) {
                    Film temp = current.data;
                    current.data = next.data;
                    next.data = temp;
                }
                next = next.next;
            }
```

```
            current = current.next;
        }
    }

}
```

Main.java

```
public static void main(String[] args) {
        DLL dll = new DLL();

        // Tambah Data Awal
        System.out.println("Menambahkan Film");
        dll.addFirst(new Film(1, "The Shawshank Redemption", 8.3));
        dll.print();

        // Tambah Data di Akhir
        System.out.println("Menambahkan Film Pada Index Akhir");
        dll.addLast(new Film(2, "The Godfather", 9.2));
        dll.addLast(new Film(3, "The Dark Knight", 3));
        dll.addLast(new Film(5, "The Lord of the Rings: The Return of the King", 6));
        dll.print();

        // Tambah Data di Tengah
        System.out.println("Menambahkan Film Pada Index 1");
        dll.add(new Film(4, "The Lord of the Rings: The Return of the King", 7), 1);
        dll.print();

        // Hapus Data Awal
        System.out.println("Menghapus Film Pada Index Awal");
        dll.removeFirst();
        dll.print();

        // Hapus Data di Akhir
        System.out.println("Menghapus Film Pada Index Akhir");
        dll.removeLast();
        dll.print();

        // Hapus Data di Tengah
        System.out.println("Menghapus Film Pada Index 2");
        dll.remove(2);
        System.out.println();

        // Search Film By ID
        System.out.println("Mencari Film Berdasarkan ID = 4");
        System.out.println(dll.searchFilm(4).judul);
        System.out.println();



        // Sort By Rating Descending

        // Sebelum SOrt
```

```
        // System.out.println("Sebelum Sort");
        // dll.print();

        // // Setelah Sort
        // System.out.println("Sort By Rating Descending");
        // dll.sortByDesc();
        // dll.print();

    }
```

Film.java

```
package film;

public class Film {
    int id;
    double rating;
    String judul;

    public Film(int id, String judul,  double rating) {
        this.id = id;
        this.rating = rating;
        this.judul = judul;
    }
}
```

Node.java

```
package film;

class Node {
    Film data;
    Node next, prev;

    Node(Node prev, Film data, Node next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }

    Node(){}
}
```