Laporan Praktikum ASD
Pertemuan 12

Nama: Gegas Anugrah Derajat
Kelas: SIB-1F
NIM: 2341760140

Praktikum 1

Membuat class Node, mendeklarasikan atribut dan menambahkan konstruktor

```
public class Node11 {
    int data;
    Node11 prev, next;

    Node11(Node11 prev, int data, Node11 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

Membuat class DoubleLinkedList

```
public class DoubleLinkedList11 {
    Node11 head;
    int size;

    public DoubleLinkedList11() {
        head = null;
        size = 0;
    }
```

Membuat method isEmpty, addFirst dan addLast

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

public void addFirst (int item) {
    if (isEmpty ()) {
    head = new Node11(prev:null, item, next:null);
    } else {
        Node11 newNode = new Node11(prev:null, item, head) ;
        head.prev = newNode;
        head = newNode;
    }
    size++;
}

public void addLast (int item) {
    if (isEmpty ()) {
        addFirst (item) ;
    } else {
        Node11 current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node11 newNode = new Node11 (current, item, next:null);
        current.next = newNode;
        size++;
    }
}
```

Membuat method add, untuk menambahkan nilai sesuai dengan indeks yang diberikan

```
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 {
        Node11 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            Node11 newNode = new Node11 (prev:null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            Node11 newNode = new Node11 (current.prev, item, current);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
        size++;
    }
}
```

Membuat method size, clear dan print

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

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

public void print() {
    if (!isEmpty()) {
        Node11 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");
    }
}
```

Membuat class Main dan mengeksekusi semua method

```
public class DoubleMain11 {
    Run | Debug
    public static void main(String[] args) throws Exception {
        DoubleLinkedList11 dll = new DoubleLinkedList11();
        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:"==================================");
        dll.add(item:40, index:1);
        dll.print();
        System.out.println("Size : " + dll.size());
        System.out.println(x:"==================================");
        dll.clear();
        dll.print();
        System.out.println("Size : " + dll.size());

    }
}
```

Hasil

```
-cp' 'C:\Users\Pongo\AppData\Roaming\Co
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!
2. Perhatikan class Node, di dalamnya 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);
```

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.

Jawaban:

1. Single linked list hanya memiliki satu arah yaitu node awal dan akhir, dan hanya memiliki satu node yaitu next.
Double linked list memiliki dua arah dan memiliki dua node yaitu prev dan next.

2. atribut prev digunakan untuk menunjuk node sebelumnya, sedangkan next digunakan untuk menunjuk node berikutnya.

3. Head menunjuk node pertama dalam daftar, size menyimpan jumlah node dalam daftar.

4. AddFirst befungsi untuk menambahkan data pada saat pertama kali. Karena isi daftar node pada awal berisikan 0, tidak ada data. Jadi tidak ada yang bisa untuk dilakukan prev.

5. Mengatur node yang sebelumnya menjadi head untuk menunjuk ke node baru yang kita tambahkan newNode.

6. AddLast befungsi menambahkan data pada urutan terakhir. Karena akan di tempatkan pada posisi terakhir.

7. Mendeteksi apakah posisi sebelumnya null, jika ya maka posisi sebelumnya akan di isi oleh data baru, dan head akan ada pada posisi data yang baru saja ditambahkan

Praktikum 2

Membuat method removeFirst pada class DoubleLinkedList

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

Membuat method removeLast

```
public 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;
    }
    Node11 current = head;
    while (current.next.next != null) {
        current = current.next;
    }
    current.next = null;
    size--;
}
```

Membuat method remove, untuk menghapus nilai pada indeks tertentu

```
public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
        Node11 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--;
    }
}
```

Pada class Main menambahkan kode untuk eksekusi method yang baru dibuat

```
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:"====================================");
dll.removeFirst();
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"====================================");
dll.removeLast();
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"====================================");
dll.remove(index:1);
dll.print();
System.out.println("Size : " + dll.size());
```

Hasil

```
da0fe67bb65b\redhat.java\jdt_ws\Pertemuan
50      40      10      20
Berhasil diisi
Size : 4
====================================
50      40      10      20
Berhasil diisi
Size : 3
====================================
50      40      10
Berhasil diisi
Size : 2
====================================
50      10
Berhasil diisi
Size : 1
```

12.3.3 Pertanyaan Percobaan

1. 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;
```

Jawaban:

1. head=head.next berfungsi untuk memperbarui pointer head yang menunjuk ke node berikutnya setelah node pertama yang dihapus. head.prev=null berfungsi untuk menghapus ponter prev karena head berasa pada daftar pertama.

2. Dengan menggunakan iterasi loop while, mencari current.next hingga ke daftar yang terakhir. Karena daftar terakhir tidak bisa melakukan .next, jadi data menunjukan data terakhir, dan ponter akan di simpan di variable current.

3. Karena kode program di atas tidak memeriksa kondisi spesifik dari daftar.

4. Kode program pertama berfungsi memperbarui pointer next dari node sebelumnya untuk menunjuk ke node berikutnya Kode program kedua berfungsi memperbarui ponter prev dari node berikutnya untuk menunjuk ke node sebelumnya.

Praktikum 3

Membuat method getFirst pada class DoubleLinkedList

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

Membuat method getLast

```
public int getLast () throws Exception {
    if (isEmpty ()) {
        throw new Exception (message:"Linked List kosong");
    }
    Node11 tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

Membuat method get

```
public int get (int index) throws Exception {
    if (isEmpty () || index >= size) {
        throw new Exception (message:"Nilai indeks di luar batas.");
    }
    Node11 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

Menambahkan kode pada class main untuk mengeksekusi method yang baru dibuat

```
DoubleLinkedList11 dll = new DoubleLinkedList11();
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:"====================================");
dll.add(item:40, index:1);
dll.print();
System.out.println("Size : " + dll.size());
System.out.println(x:"====================================");
// dll.clear();
// dll.print();
// System.out.println("Size : " + dll.size());
System.out.println("Data awal pada linked list adalah: " + dll.getFirst());
System.out.println("Data akhir pada linked list adalah: " + dll.getLast());
System.out.println("Data awal pada linked list adalah: " + dll.get(index:1));
```

hasil

```
da0fe67bb65b\redhat.java\jdt_ws\Pertemu
Linked list kosong
Size : 0
====================================
7       3       4
Berhasil diisi
size : 3
====================================
7       40      3       4
Berhasil diisi
Size : 4
====================================
Data awal pada linked list adalah: 7
Data akhir pada linked list adalah: 4
Data awal pada linked list 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)

Jawaban:

1. Size() mengembalikan nilai size yang telah di atur pada awal menjalankan program.

2. Secara default index umumnya dimulai dari 0. Namun dengan dengan sedikit mengubah pada method add dengan memberikan validasi untuk index seperti dibawah ini.

3. Penambahan pada single linked hanya satu arah dari depan ke belakang, tidak bisa menambahkan elemen di tengah daftar Penambahan pada double linked memiliki 2 arah, bisa dari depan dan belakang, bisa menyisipkan kode di mana saja.

4. pada kode A akan memeriksa jumlah elemen, jika jumlah elemen tersebut bernilai 0 maka akan mengembalikan true, namun jika tidak sama dengan 0 maka akan mengembalikan false Pada kode B akan memeriksa ponter head apakah berisikan null, jika iya maka akan mengembalikan true, jika tidak null maka kan mengembalikan false.

Tugas

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)

Membuat class vaksin

```
public class Vaksin {
    String nama;
    int antri;

    Vaksin(int antri, String nama) {
        this.nama = nama;
        this.antri = antri;
    }
}
```

Membuat Class Node

```
public class Node {
    Vaksin data;
    Node prev, next;

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

Membuat class DllVakasin dan menambahkan methodnya

```
public class DllVaksin {
    Node head;
    int size;

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

    public void addFirst(Vaksin 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++;
    }
```

```java
public void addLast(Vaksin 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++;
    }
}
```

```java
public void add(Vaksin 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 = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
        size++;
    }
}
```

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

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

public void print() {
    if (!isEmpty()) {
        Node tmp = head;
        System.out.println(x:"No \t|\tNama");
        while (tmp != null) {
            System.out.println(tmp.data.antri + "\t|\t" + tmp.data.nama);
            tmp = tmp.next;
        }
        System.out.println(x:"\nBerhasil diisi");
    } else {
        System.out.println(x:"Linked list kosong");
    }
}
```

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

public 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--;
}
```

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

public Vaksin 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;
}
```

Membuat class VaksiMain dan method menu serta mengeksekusi method yang telah dibuat

```java
import java.util.Scanner;

public class VaksinMain {
    public static void menu() {
        System.out.println(x:"------------------------------");
        System.out.println(x:"PENGANTRI VAKSIN EXTRAVAGANZA");
        System.out.println(x:"------------------------------");
        System.out.println();
        System.out.println(x:"1. Tambah data penerima vaksin");
        System.out.println(x:"2. Hapus data pengantri vaksin");
        System.out.println(x:"3. Daftar penerima vaksin");
        System.out.println(x:"4. Keluar");
        System.out.println(x:"------------------------------");

    }
```

```
Run | Debug
public static void main(String[] args) throws Exception {
    Scanner gs = new Scanner(System.in);
    DllVaksin dll = new DllVaksin();

    int pilih;
    do {
        menu();
        pilih = gs.nextInt();
        switch (pilih) {
            case 1:
                System.out.println(x:"----------------------------");
                System.out.println(x:"Masukkan Data Penerima Vaksin");
                System.out.println(x:"----------------------------");
                System.out.print(s:"Nomor Antrian: ");
                int antri = gs.nextInt();
                System.out.print(s:"Nama Penerima: ");
                String nama = gs.next();
                Vaksin nb = new Vaksin(antri, nama);
                dll.addLast(nb);
                System.out.println();
                break;
            case 2:
                Vaksin penerima = dll.getFirst();
                System.out.println(penerima.nama + " telah selesai divaksin");
                dll.removeFirst();
                break;
```

```
            case 3:
                System.out.println(x:"=======================");
                System.out.println(x:"Daftar pengantri vaksin");
                System.out.println(x:"=======================");
                dll.print();
                System.out.println("Sisa Antrian: " + dll.size());
                System.out.println();
                break;
            case 4:
                return;
            default:
                break;
        }
    } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4);
    gs.close();
}
```

Hasil

```
------------------------------
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: FFFFFF
```

```
------------------------------
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: 134
Nama Penerima: AAAAAAAA
```

```
------------------------------
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: 121
Nama Penerima: BBBBBBBBB
```

```
------------------------------
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     |       FFFFFF
134     |       AAAAAAAA
121     |       BBBBBBBBB

Berhasil diisi
Sisa Antrian: 3
```

```
------------------------------
PENGANTRI VAKSIN EXTRAVAGANZA
------------------------------

1. Tambah data penerima vaksin
2. Hapus data pengantri vaksin
3. Daftar penerima vaksin
4. Keluar
------------------------------
2
FFFFFF telah selesai divaksin
```

```
------------------------------
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     |       FFFFFF
134     |       AAAAAAAA
121     |       BBBBBBBBB

Berhasil diisi
Sisa Antrian: 2
```

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.

Membuat class Film

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

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

Membuat class NodeFilm

```
public class NodeFilm {
    Film data;
    NodeFilm prev, next;

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

Membuat class DllFilm

```
public class DllFilm {
    NodeFilm head;
    int size;

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

    public void addFirst(Film item) {
        if (isEmpty()) {
            head = new NodeFilm(prev:null, item, next:null);
        } else {
            NodeFilm newNode = new NodeFilm(prev:null, item, head);
            head.prev = newNode;
            head = newNode;
        }
        size++;
    }

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

```java
public void add(Film 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 {
        NodeFilm current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            NodeFilm newNode = new NodeFilm(prev:null, item, current);
            current.prev = newNode;
            head = newNode;
        } else {
            NodeFilm newNode = new NodeFilm(current.prev, item, current);
            newNode.prev = current.prev;
            newNode.next = current;
            current.prev.next = newNode;
            current.prev = newNode;
        }
        size++;
    }
}
```

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

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

public void print() {
    if (!isEmpty()) {
        NodeFilm tmp = head;
        System.out.println(x:"Cetak");
        while (tmp != null) {
            System.out.println("No.ID \t: " + tmp.data.id);
            System.out.println(" Judul \t: " + tmp.data.judul);
            System.out.println(" Rating \t: " + tmp.data.rating);
            tmp = tmp.next;
        }
        System.out.println(x:"\nBerhasil diisi");
    } else {
        System.out.println(x:"Linked list kosong");
    }
}

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

```java
public 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;
    }
    NodeFilm current = head;
    while (current.next.next != null) {
        current = current.next;
    }
    current.next = null;
    size--;
}

public void remove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    } else {
        NodeFilm 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--;
    }
}
```

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

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

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

```java
public Film searchId(int id) throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List masih kosong");
    }

    NodeFilm current = head;
    while (current != null) {
        if (current.data.id == id) {
            return current.data;
        }
        current = current.next;
    }

    throw new Exception("ID Film " + id + " tidak ditemukan");
}

public void sortRatingDesc() {
    if (isEmpty() || size == 1) {
        return;
    }

    for (int i = 0; i < size - 1; i++) {
        NodeFilm current = head;
        NodeFilm maxNode = current;

        for (int j = 0; j < size - i - 1; j++) {
            if (current.next != null && current.next.data.rating > maxNode.data.rating) {
                maxNode = current.next;
            }
            current = current.next;
        }

        if (maxNode != current) {
            Film temp = current.data;
            current.data = maxNode.data;
            maxNode.data = temp;
        }
    }
}
```

Membuat class Main

```java
import java.util.Scanner;

public class FilmMain {
    public static void menu() {
        System.out.println(x:"------------------------------");
        System.out.println(x:"DATA FILM LAYAR LEBAR");
        System.out.println(x:"------------------------------");
        System.out.println(x:"1. Tambah data awal");
        System.out.println(x:"2. Tambah data akhir");
        System.out.println(x:"3. Tambah data indeks tertentu");
        System.out.println(x:"4. Hapus data pertama");
        System.out.println(x:"5. Hapus data terakhir");
        System.out.println(x:"6. Hapus data tertentu");
        System.out.println(x:"7. Cetak");
        System.out.println(x:"8. Cari id film");
        System.out.println(x:"9. Urut data rating film");
        System.out.println(x:"10. Keluar");
        System.out.println(x:"------------------------------");
    }
```

```
public static void main(String[] args) throws Exception {
    Scanner gs = new Scanner(System.in);
    DllFilm dll = new DllFilm();

    int pilih;

    do {
        menu();
        pilih = gs.nextInt();
        switch (pilih) {
            case 1:
                System.out.println(x:"-----------------------------");
                System.out.println(x:"Masukkan Data Posisi Awal");
                System.out.println(x:"-----------------------------");
                System.out.print(s:"ID : ");
                int id = gs.nextInt();
                System.out.print(s:"Judul Film : ");
                String judul = gs.next();
                System.out.print(s:"Rating : ");
                double rating = gs.nextDouble();
                Film nb = new Film(id, judul, rating);
                dll.addFirst(nb);
                System.out.println();
                break;
```

```
case 2:
    System.out.println(x:"-----------------------------");
    System.out.println(x:"Masukkan Data Posisi Akhir");
    System.out.println(x:"-----------------------------");
    System.out.print(s:"ID : ");
    int id1 = gs.nextInt();
    System.out.print(s:"Judul Film : ");
    String judul1 = gs.next();
    System.out.print(s:"Rating : ");
    double rating1 = gs.nextDouble();
    Film nb1 = new Film(id1, judul1, rating1);
    dll.addFirst(nb1);
    System.out.println();
    break;
case 3:
    System.out.println(x:"-----------------------------");
    System.out.println(x:"Masukkan Data FILM");
    System.out.println(x:"-----------------------------");
    System.out.print(s:"Urutan ke : ");
    int index = gs.nextInt();
    System.out.print(s:"ID : ");
    int id2 = gs.nextInt();
    System.out.print(s:"Judul Film : ");
    String judul2 = gs.next();
    System.out.print(s:"Rating : ");
    double rating2 = gs.nextDouble();
    Film nb2 = new Film(id2, judul2, rating2);
    dll.add(nb2, index);
    System.out.println();
    break;
case 4:
    Film film = dll.getFirst();
    System.out.println("Film " +film.judul + " telah dihapus.");
    dll.removeFirst();
    System.out.println();
    break;
```

```
case 5:
    Film film1 = dll.getLast();
    System.out.println("Film " +film1.judul + " telah dihapus.");
    dll.removeLast();
    System.out.println();
    break;
case 6:
    System.out.print(s:"Urutan ke : ");
    int index1 = gs.nextInt();
    Film film2 = dll.get(index1);
    System.out.println("Film " +film2.judul + " telah dihapus.");
    dll.remove(index1);
    break;
case 7:
    System.out.println(x:"----------------------------");
    System.out.println(x:"DATA FILM LAYAR LEBAR");
    System.out.println(x:"----------------------------");
    dll.print();
    System.out.println();
    break;
case 8:
    System.out.print(s:"Masukkan ID : ");
    int idSearch = gs.nextInt();
    Film seach = dll.searchId(idSearch);
    System.out.println("ID \t: " +seach.id);
    System.out.println("Judul \t: " +seach.judul);
    System.out.println("Rating \t: " +seach.rating);
    break;
case 9:
    dll.sortRatingDesc();
    System.out.println(x:"----------------------------");
    System.out.println(x:"DATA FILM LAYAR LEBAR SORTING DESC");
    System.out.println(x:"----------------------------");
    dll.print();
    System.out.println();
    break;
case 10:
    return;
```

```
        }
    } while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 || pilih == 5|| pilih == 6|| pilih == 7|| pilih == 8|| pilih == 9|| pilih == 10);
    gs.close();
  }
}
```

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
10. Keluar
------------------------------
1
------------------------------
Masukkan Data Posisi Awal
------------------------------
ID : 11
Judul Film : Peternak
Rating : 3.0
```

```
------------------------------
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
10. Keluar
------------------------------
2
------------------------------
Masukkan Data Posisi Akhir
------------------------------
ID : 15
Judul Film : 7cm
Rating : 4.5
```

```
------------------------------
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
10. Keluar
------------------------------
3
------------------------------
Masukkan Data FILM
------------------------------
Urutan ke : 1
ID : 13
Judul Film : Neutral
Rating : 3.0
```

```
------------------------------
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
10. Keluar
------------------------------
7
------------------------------
DATA FILM LAYAR LEBAR
------------------------------
Cetak
No.ID   : 15
 Judul  : 7cm
 Rating         : 4.5
No.ID   : 13
 Judul  : Neutral
 Rating         : 3.0
No.ID   : 11
 Judul  : Peternak
 Rating         : 3.0

Berhasil diisi
```

```
------------------------------
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
10. Keluar
------------------------------
8
Masukkan ID : 13
ID      : 13
Judul   : Neutral
Rating  : 3.0
------------------------------
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
10. Keluar
------------------------------
10
```