

PRAKTIKUM ALGORITMA DAN STRUKTUR DATA
JOBSHEET PERTEMUAN KE-12



RIO TRI PRAYOGO

TI 1A

26

2341720236

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2024

Double Linked List

Praktikum 1

Percobaan :

Node26

```
package minggu12;

public class Node26 {
    int data;
    Node26 prev, next;

    Node26(Node26 prev, int data, Node26 next) {
        this.prev = prev;
        this.data = data;
        this.next = next;
    }
}
```

DoubleLinkedLists26

```
package minggu12;

public class DoubleLinkedLists26 {
    Node26 head;
    int size;

    public DoubleLinkedLists26() {
        head = null;
        size = 0;
    }

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

    public void addFirst(int item) {
        if (isEmpty()) {
            head = new Node26(null, item, null);
        }
    }
}
```

```

    } else {
        Node26 newNode = new Node26(null, item, head);
        head.prev = newNode;
        head = newNode;
    }
    size++;
}

public void addLast(int item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
        Node26 current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node26 newNode = new Node26(current, item, null);
        current.next = newNode;
        size++;
    }
}

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 {
        Node26 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            Node26 newNode = new Node26(null, item, current);

```

```

        current.prev = newNode;
        head = newNode;
    } else {
        Node26 newNode = new Node26(current.prev, item, current);
        newNode.prev = current.prev;
        newNode.next = current;
        current.prev.next = newNode;
        current.prev = newNode;
    }
}

size++;
}

public int size() {
    return size;
}

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

public void print() {
    if (!isEmpty()) {
        Node26 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");
    }
}
}
}

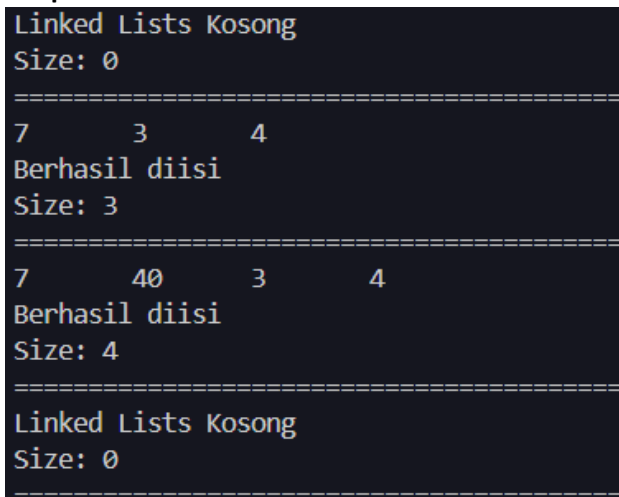
```

DoubleLinkedListsMain26

```
package minggu12;

public class DoubleLinkedListsMain26 {
    public static void main(String[] args) throws Exception {
        DoubleLinkedLists26 dll = new DoubleLinkedLists26();
        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("=====");
        dll.clear();
        dll.print();
        System.out.println("Size: " + dll.size());
        System.out.println("=====");
    }
}
```

Output :



```
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
=====
```

Pertanyaan :

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
 - “*Double Linked Lists*” memiliki dua buah yaitu pointer **next** menunjuk pada node setelahnya dan pointer **prev** menunjuk pada node sebelumnya sementara “*Single Linked List*” hanya memiliki satu buah pointer yaitu pointer **next** menunjuk pada node setelahnya sehingga “*Double Linked Lists*” memiliki tiga buah atribut yaitu `Data`, `next node`, dan `previous node` sementara “*Single Linked List*” hanya memiliki dua buah atribut memiliki tiga buah parameter yaitu `Data` dan `next node`.

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

- Atribut pada class `Node` berupa **next** berfungsi untuk menunjuk pada node setelahnya dan atribut **prev** berfungsi untuk menunjuk pada node sebelumnya.

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 kedua atribut tersebut berguna untuk menandakan bahwa linked list awalnya adalah kosong dan tidak ada isinya.

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 method `addFirst()` berguna untuk menambah data dibagian paling depan atau pertama dan dimana data pertama selalu memiliki pointer **prev** = `null` atau kosong.

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

- Statement tersebut berfungsi untuk mengisi `Node` sebelum `head`, dimana `newNode` merupakan data yang baru dimasukkan sementara `head` merupakan data paling depan. Sehingga data yang baru dimasukkan ditaruh ditempat sebelum data pertama dan menjadi data pertama sesuai dengan method `addFirst()`.

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);`

- Pengisian parameter **prev** dengan `current`, dan **next** dengan `null` mengisi data yang baru dimasukkan ke dalam list terakhir. `current` sendiri merupakan `Node` terakhir yang ditunjukkan pada baris sebelumnya dimana perulangan `while` digunakan untuk mencari `Node` terakhir. Sementara parameter **next** berupa `null` menandakan bahwa tidak ada `Node` lagi setelah data yang baru dimasukkan.

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.

- Potongan kode diatas berfungsi untuk mengecek apakah `Node` pada saat ini (`current`) merupakan `Node head`, jika iya maka data yang baru ditambahkan akan ditaruh di paling depan dan menjadi `Node head` baru.

Praktikum 2

Percobaan :

“Baris kode di bawah merupakan tambahan yang dimasukkan ke dalam kode Praktikum 1”

DoubleLinkedLists26

```
...

    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 removeLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List masih kosong, tidak dapat
dihapus!");
        } else if (head.next == null) {
            head = null;
            size--;
            return;
        }
        Node26 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("Nilai indeks di luar batas");
        } else if (index == 0) {
            removeFirst();
        } else {
            Node26 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;
            } else {
                current.prev.next = current.next;
                current.next.prev = current.prev;
            }
            size--;
        }
    }
}

```

...

DoubleLinkedListsMain26

...

```

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

```



```

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

```

Output :

```

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
=====

```

Pertanyaan :

1. Apakah maksud statement berikut pada method `removeFirst()`?
`head = head.next;`
`head.prev = null;`
 ➤ Statement diatas berfungsi untuk mengubah posisi head ke dalam posisi setelah data yang paling depan (`head = head.next;`) lalu setelah diubah data sebelum data head terbaru akan dihapus atau dikosongkan (`head.prev = null;`). Hal tersebut membuat posisi dan data head awal menjadi terhapus dan posisi head digantikan data setelahnya.
2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method `removeLast()`?
 ➤ Pendeteksian posisi data bagian akhir pada method `removeLast()` dilakukan di dalam baris perulangan `while (current.next.next != null) { current = current.next; }` dimana baris tersebut melakukan perulangan sampai baris paling terakhir ditemukan
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 code diatas tidak cocok untuk perintah `remove` dikarenakan baris diatas hanya memperbarui pointer dari **head** dan tidak memperbarui pointer **next**.

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

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

- Kode diatas berfungsi untuk menghapus Node yang ada di tengah-tengah Linked List. **Current** sendiri merupakan data yang dipilih atau akan dihapus. `current.prev.next = current.next;` berfungsi untuk mengubah pointer **next** dari Node sebelum **current** menjadi menunjuk ke dalam Node setelah **current**. `current.next.prev = current.prev;` berfungsi untuk mengubah pointer **prev** dari Node setelah **current** menjadi menunjuk ke dalam Node setelah **current**. Sehingga Node **current** tidak menunjuk dan ditunjuk oleh siapapun yang akhirnya hilang dari Linked List.

Praktikum 3

Percobaan :

“Baris kode di bawah merupakan tambahan yang dimasukkan ke dalam kode Praktikum 1 dan 2”

DoubleLinkedLists26

```
...  
  
public int getFirst() throws Exception {  
    if (isEmpty()) {  
        throw new Exception("Linked List kosong");  
    }  
    return head.data;  
}  
  
public int getLast() throws Exception {  
    if (isEmpty()) {  
        throw new Exception("Linked List kosong");  
    }  
    Node26 tmp = head;  
    while (tmp.next != null) {  
        tmp = tmp.next;  
    }  
    return tmp.data;  
}  
  
public int get(int index) throws Exception {  
    if (isEmpty() || index >= size) {  
        throw new Exception("Nilai indeks di luar batas.");  
    }  
}
```

```

    }

    Node26 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }

    return tmp.data;
}

...

```

DoubleLinkedListsMain26

```

...

    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));
    ...

```

Output :

```
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
```

Pertanyaan :

1. Jelaskan method **size()** pada class **DoubleLinkedLists**!
 - Method `size()` pada class **DoubleLinkedLists** berfungsi untuk menentukan ukuran dari *Linked List* dimana dalam *Linked List* ukurannya bersifat dinamis sehingga tidak tetap serta disesuaikan dengan isi dari *Linked List* sehingga perlu dideklarasikan method `size()`.
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke-1!
 - Untuk mengatur indeks supaya dapat dimulai dari indeks ke-1 bisa dengan merubah ketentuan pada method `add()` dimana jika menambahkan data baru maka data indeks data tersebut diubah menjadi 1 dan jika menambah lagi akan terus berlanjut sehingga data pertama akan memiliki indeks 1, juga dengan mengubah ketentuan dari `get()` dan `getFirst()` dimana jika indeks = 1 maka data tersebut menjadi data paling awal.
3. Jelaskan perbedaan karakteristik fungsi **Add** pada Double Linked Lists dan Single Linked Lists!
 - Perbedaan utamanya berada pada jumlah pointer yang dimiliki. *Double Linked Lists* memiliki dua buah pointer yaitu **next** yang menunjuk Node sebelumnya dan **prev** yang menunjuk Node setelahnya, sementara *Single Linked Lists* hanya memiliki satu buah pointer yaitu **next** yang menunjuk Node setelahnya. Oleh karena itu *Single Linked Lists* memerlukan perulangan ke seluruh data jika data ditambahkan pada akhir Node yang membuat penambahan data menjadi lebih lama. Penambahan data pada tengah-tengah *Linked Lists* juga lebih efisien jika menggunakan *Double 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)

- Kode (a) mengecek apakah size dalam *Linked Lists* kosong/0. Sementara kode (b) mengecek apakah posisi head dalam *Linked Lists* kosong. Kedua kode tersebut memiliki fungsi yang sama yaitu mengecek apakah data dalam *Linked Lists* kosong atau tidak.

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**)

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

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
```

➤ Code

Node26

```
package minggu12.tugas1;

public class Node26 {
    Pengantri26 data;

    Node26 prev, next;

    Node26(Node26 prev, Pengantri26 item, Node26 next) {
        this.prev = prev;
        data = item;
        this.next = next;
    }
}
```

```
}  
}
```

Pengantri26

```
package minggu12.tugas1;  
  
public class Pengantri26 {  
    int no;  
    String nama;  
  
    Pengantri26(int no, String nama) {  
        this.no = no;  
        this.nama = nama;  
    }  
}
```

DoubleLinkedLists26

```
package minggu12.tugas1;  
  
public class DoubleLinkedLists26 {  
    Node26 head;  
    int size;  
  
    public DoubleLinkedLists26() {  
        head = null;  
        size = 0;  
    }  
  
    public boolean isEmpty() {  
        return head == null;  
    }  
  
    public void addFirst(Pengantri26 item) {  
        if (isEmpty()) {  
            head = new Node26(null, item, null);  
        } else {  
            Node26 newNode = new Node26(null, item, head);  
            head = newNode;  
            size++;  
        }  
    }  
}
```

```

        head.prev = newNode;

        head = newNode;
    }

    size++;
}

public void addLast(Pengantri26 item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
        Node26 current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node26 newNode = new Node26(current, item, null);
        current.next = newNode;
        size++;
    }

    System.out.println("Berhasil diisi");
}

public int size() {
    return size;
}

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

public void print() {
    if (!isEmpty()) {
        Node26 tmp = head;
        int index = 0;
        System.out.println("|No.\t|Nama\t|");
        while (tmp != null) {

```

```

        Pengantri26 antri = tmp.data;

        System.out.println("|" + antri.no + "\t|" +
antri.nama + "\t|");

        tmp = tmp.next;

        index++;

    }

    System.out.println("Sisa Antrian: " + index);
} else {

    System.out.println("Linked Lists Kosong");

}

}

public void removeFirst() throws Exception {

    if (isEmpty()) {

        throw new Exception("Linked List masih kosong, tidak
dapat dihapus!");

    } else if (size == 1) {

        removeLast();

    } else {

        Pengantri26 antri = head.data;

        head = head.next;

        head.prev = null;

        size--;

        System.out.println(antri.nama + " telah selesai
divaksinasi!");

    }

}

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;

    }

    Node26 current = head;

```



```

        while (current.next.next != null) {
            current = current.next;
        }

        Pengantri26 antri = current.next.data;
        current.next = null;
        size--;

        System.out.println(antri.nama + " telah selesai
divaksinasi!");
    }

    public Pengantri26 getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        return head.data;
    }

    public Pengantri26 getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        Node26 tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
        return tmp.data;
    }

    public Pengantri26 get(int index) throws Exception {
        if (isEmpty() || index >= size) {
            throw new Exception("Nilai indeks di luar batas.");
        }
        Node26 tmp = head;
        for (int i = 0; i < index; i++) {
            tmp = tmp.next;
        }
    }

```

```

        return tmp.data;
    }
}

```

Main26

```

package minggul2.tugas1;

import java.util.Scanner;

public class Main26 {
    public static void main(String[] args) throws Exception {
        Scanner scan = new Scanner(System.in);
        DoubleLinkedLists26 list = new DoubleLinkedLists26();
        boolean run = true;
        do {
            System.out.println("\n+++++");
            System.out.println("PENGANTRI VAKSIN EKSTRAVAGANZA");
            System.out.println("+++++");
            System.out.println("1. Tambah Data Penerima Vaksin");
            System.out.println("2. Hapus Data Pengantri Vaksin");
            System.out.println("3. Daftar Penerima Vaksin");
            System.out.println("4. Keluar");
            System.out.println("+++++");
            int input = scan.nextInt();
            switch (input) {
                case 1:
                    System.out.println("-----");
                    System.out.println("Masukkan Data Penerima");
                    System.out.println("-----");
                    System.out.println("Nomor Antrian: ");
                    int no = scan.nextInt();
                    System.out.println("Nama Penerima: ");
                    scan.nextLine();
                    String nama = scan.nextLine();
                    Pengantri26 antrian = new Pengantri26(no, nama);

```

```

        list.addLast(antrian);

        break;

    case 2:

        list.removeFirst();

        break;

    case 3:

        System.out.println("+++++++");
        System.out.println("Daftar Pengantri Vaksin");
        System.out.println("+++++++");
        list.print();

        break;

    case 4:

        System.out.println("-----");

-");

        System.out.println("Keluar
program\nTerimakasih!");

        System.out.println("-----");

-");

        run = false;

        break;

    default:

        System.out.println("Pilihan tidak valid!");

        break;

    }

    } while (run);

}

}

```

➤ Output

Menu Awal	Penambahan Data	Cetak Data	Hapus Data
<pre> +++++++ PENGANTRI VAKSIN EKSTRAVAGANZA +++++++ 1. Tambah Data Penerima Vaksin 2. Hapus Data Pengantri Vaksin 3. Daftar Penerima Vaksin 4. Keluar +++++++ </pre>	<pre> +++++++ PENGANTRI VAKSIN EKSTRAVAGANZA +++++++ 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 Berhasil diisi </pre>	<pre> +++++++ PENGANTRI VAKSIN EKSTRAVAGANZA +++++++ 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 </pre>	<pre> +++++++ PENGANTRI VAKSIN EKSTRAVAGANZA +++++++ 1. Tambah Data Penerima Vaksin 2. Hapus Data Pengantri Vaksin 3. Daftar Penerima Vaksin 4. Keluar +++++++ 3 ----- Daftar Pengantri Vaksin ----- No. Nama 124 Mely 135 Johan 146 Rosi Sisa Antrian: 3 </pre>

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

=====
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

=====
Masukkan Data Film Posisi Awal
ID Film:
Judul Film:
Spider-Man: No Way Home
Rating Film:
8.7

=====
Masukkan Data Film Posisi Akhir
ID Film:
Judul Film:
Uncharted
Rating Film:
6.7

=====
Masukkan Data Film
Urutan ke-
ID Film:
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
ipki: 8.7
ID: 1785
Judul Film: Skyfall
ipki: 7.8
ID: 1567
Judul Film: The Dark Knight Rises
ipki: 8.4
ID: 1234
Judul Film: Death on the Nile
ipki: 6.6
ID: 1346
Judul Film: Uncharted
ipki: 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
```

➤ Code Node26

```
package minggul12.tugas2;

public class Node26 {

    Film26 data;

    Node26 prev, next;

    Node26(Node26 prev, Film26 item, Node26 next) {

        this.prev = prev;

        data = item;

        this.next = next;

    }

}
```

Film26

```
package minggul12.tugas2;

public class Film26 {

    int id;

    String judul;
```

```

float rating;

Film26(int id, String judul, float rating) {
    this.id = id;
    this.judul = judul;
    this.rating = rating;
}
}

```

DoubleLinkedLists26

```

package minggul2.tugas2;

public class DoubleLinkedLists26 {
    Node26 head;
    int size;

    public DoubleLinkedLists26() {
        head = null;
        size = 0;
    }

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

    public void add(Film26 item, int pos) throws Exception {
        if (pos < 1 || pos > size + 1) {
            throw new Exception("Posisi data film di luar batas");
        } else if (pos == 1) {
            addFirst(item);
        } else if (pos == size + 1) {
            addLast(item);
        } else {
            Node26 current = head;
            int currentPos = 1;
            while (currentPos < pos - 1) {
                current = current.next;
            }
        }
    }
}

```

```

        currentPos++;
    }
    Node26 newNode = new Node26(current, item, current.next);
    current.next.prev = newNode;
    current.next = newNode;
    size++;
}
}

public void addFirst(Film26 item) {
    if (isEmpty()) {
        head = new Node26(null, item, null);
    } else {
        Node26 newNode = new Node26(null, item, head);
        head.prev = newNode;
        head = newNode;
    }
    size++;
}

public void addLast(Film26 item) {
    if (isEmpty()) {
        addFirst(item);
    } else {
        Node26 current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node26 newNode = new Node26(current, item, null);
        current.next = newNode;
        size++;
    }
}

public int size() {
    return size;
}

```

```

    }

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

    public void print() {
        if (!isEmpty()) {
            Node26 tmp = head;
            int index = 1;
            while (tmp != null) {
                Film26 film = tmp.data;
                System.out.println("-----");
            );
                System.out.println("Film ke-" + index);
                System.out.println("-----");
            );

                System.out.println("ID: " + film.id);
                System.out.println(" Judul Film: " + film.judul);
                System.out.println(" Rating: " + film.rating);
                tmp = tmp.next;
                index++;
            }
        } else {
            System.out.println("Linked Lists Kosong");
        }
    }

    public void remove(int position) throws Exception {
        if (position < 1 || position > size) {
            throw new Exception("Posisi data film di luar batas");
        } else if (position == 1) {
            removeFirst();
        } else if (position == size) {
            removeLast();
        } else {

```

```

        Node26 current = head;
        int currentPosition = 1;
        while (currentPosition < position) {
            current = current.next;
            currentPosition++;
        }
        if (current.next == null) {
            current.prev.next = null;
        } else if (current.prev == null) {
            current = current.next;
            current.prev = null;
        } else {
            current.prev.next = current.next;
            current.next.prev = current.prev;
        }
        size--;
        Film26 film = current.data;
        System.out.println("Film " + film.judul + " telah
berhasil dihapus!");
    }
}

public void removeFirst() throws Exception {
    if (isEmpty()) {
        throw new Exception("Linked List masih kosong, tidak
dapat dihapus!");
    } else if (size == 1) {
        removeLast();
    } else {
        Film26 film = head.data;
        head = head.next;
        head.prev = null;
        size--;
        System.out.println("Film " + film.judul + " telah
berhasil dihapus!");
    }
}

```



```

    }

    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;
        }
        Node26 current = head;
        while (current.next.next != null) {
            current = current.next;
        }
        Film26 film = current.next.data;
        current.next = null;
        size--;
        System.out.println("Film " + film.judul + " telah berhasil
dihapus!");
    }

    public Film26 getFirst() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        return head.data;
    }

    public Film26 getLast() throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        Node26 tmp = head;
        while (tmp.next != null) {
            tmp = tmp.next;
        }
    }

```

```

        return tmp.data;
    }

    public Film26 get(int id) throws Exception {
        if (isEmpty()) {
            throw new Exception("Linked List kosong");
        }
        Node26 current = head;
        int index = 1;
        while (current != null) {
            if (current.data.id == id) {
                Film26 film = current.data;

                System.out.println("Data ID Film: " + film.id + "
berada di node ke-" + index);

                System.out.println("IDENTITAS:");
                System.out.println("ID: " + film.id);
                System.out.println(" Judul Film: " + film.judul);
                System.out.println(" Rating: " + film.rating);
                return current.data;
            }
            current = current.next;
            index++;
        }
        throw new Exception("Film dengan ID " + id + " tidak
ditemukan.");
    }

    public DoubleLinkedLists26 dataAsli() {
        DoubleLinkedLists26 listAsli = new DoubleLinkedLists26();
        if (!isEmpty()) {
            Node26 tmp = head;
            while (tmp != null) {
                listAsli.addLast(tmp.data);
                tmp = tmp.next;
            }
        }
        return listAsli;
    }

```

```

    }

    public void sort() {
        if (isEmpty() || size == 1) {
            print();
        }
        DoubleLinkedLists26 dataUrut = dataAsli();
        if (dataUrut.size == 1) {
            dataUrut.print();
            return;
        }
        for (int i = 0; i < dataUrut.size - 1; i++) {
            Node26 current = dataUrut.head;
            for (int j = 0; j < dataUrut.size - 1 - i; j++) {
                if (current.data.rating < current.next.data.rating) {
                    Film26 temp = current.data;
                    current.data = current.next.data;
                    current.next.data = temp;
                }
                current = current.next;
            }
        }
        dataUrut.print();
    }
}

```

Main26

```

package minggul2.tugas2;

import java.util.Scanner;

public class Main26 {
    public static void main(String[] args) throws Exception {
        Scanner scan = new Scanner(System.in);
        DoubleLinkedLists26 list = new DoubleLinkedLists26();
        int idF;
        String judul;
    }
}

```

```

float rating;
boolean run = true;
do {
    System.out.println("=====");
    System.out.println("DATA FILM LAYAR LEBAR");
    System.out.println("=====");
    System.out.println("1. Tambah Data Awal");
    System.out.println("2. Tambah Data Akhir");
    System.out.println("3. Tambah Data Posisi Tertentu");
    System.out.println("4. Hapus Data Pertama");
    System.out.println("5. Hapus Data Terakhir");
    System.out.println("6. Hapus Data Tertentu");
    System.out.println("7. Cetak");
    System.out.println("8. Cari ID Film");
    System.out.println("9. Urut Data Rating Film-DESC");
    System.out.println("10. Keluar");
    System.out.println("=====");
    int input = scan.nextInt();
    System.out.println("=====");
    switch (input) {
        case 1:
            System.out.println("Masukkan Data Film Posisi
Awal");
            System.out.println("=====");
            System.out.println("ID Film:");
            idF = scan.nextInt();
            System.out.println("Judul Film:");
            scan.nextLine();
            judul = scan.nextLine();
            System.out.println("Rating Film:");
            rating = scan.nextFloat();
            Film26 filmAwal = new Film26(idF, judul, rating);
            System.out.println("=====");
            list.addFirst(filmAwal);
            break;

```

```

        case 2:
            System.out.println("Masukkan Data Film Posisi
Akhir");
            System.out.println("=====");

            System.out.println("ID Film:");
            idF = scan.nextInt();
            System.out.println("Judul Film:");
            scan.nextLine();
            judul = scan.nextLine();
            System.out.println("Rating Film:");
            rating = scan.nextFloat();
            Film26 filmAkhir = new Film26(idF, judul,
rating);
            System.out.println("=====");

            list.addLast(filmAkhir);
            break;
        case 3:
            System.out.println("Masukkan Data Film Urutan Ke-
");
            System.out.println("=====");

            System.out.println("ID Film:");
            idF = scan.nextInt();
            System.out.println("Judul Film:");
            scan.nextLine();
            judul = scan.nextLine();
            System.out.println("Rating Film:");
            rating = scan.nextFloat();
            System.out.println("Data Film ini akan masuk di
urutan ke-");

            int urutan = scan.nextInt();
            Film26 filmUrutan = new Film26(idF, judul,
rating);
            System.out.println("=====");

            list.add(filmUrutan, urutan);
            break;
        case 4:

```

```

        list.removeFirst();
        break;
    case 5:
        list.removeLast();
        break;
    case 6:
        System.out.println("Film urutan berapa yang ingin
anda hapus?");
        System.out.println("=====");
        list.print();
        System.out.println("=====");
        System.out.print("Masukkan urutan film: ");
        int hapusUrutan = scan.nextInt();
        list.remove(hapusUrutan);
        break;
    case 7:
        System.out.println("Cetak Data");
        list.print();
        break;
    case 8:
        System.out.println("Masukkan ID Film yang dicari:
");
        System.out.println("=====");
        int cariId = scan.nextInt();
        list.get(cariId);
        break;
    case 9:
        System.out.println("Urutan Film Terbaik
Berdasarkan Rating:");
        list.sort();
        break;
    case 10:
        System.out.println("Keluar
program\nTerimakasih!");
        run = false;
        break;

```

```
        default:
            System.out.println("Pilihan tidak valid!");
            break;
    }
} while (run);
}
```

➤ **Output**

0. *Menu Awal*

```
=====
DATA FILM LAYAR LEBAR
=====
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Posisi 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. *Tambah Data Awal*

```
=====
1
=====
Masukkan Data Film Posisi Awal
=====
ID Film:
1222
Judul Film:
Spider-Man: No Way Home
Rating Film:
8.7
```

Sebelum	Sesudah
<pre>===== Cetak Data Linked Lists Kosong =====</pre>	<pre>Cetak Data ----- Film ke-1 ----- ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7</pre>

2. *Tambah Data Akhir*

```
=====
2
=====
Masukkan Data Film Posisi Akhir
=====
ID Film:
1346
Judul Film:
Uncharted
Rating Film:
6.7
```

Sebelum	Sesudah
<pre>Cetak Data ----- Film ke-1 ----- ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7</pre>	<pre>Cetak Data ----- Film ke-1 ----- ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ----- Film ke-2 ----- ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>

3. Tambah Data Posisi Tertentu

```
=====
3
=====
Masukkan Data Film Urutan Ke-
=====
ID Film:
1567
Judul Film:
The Dark Knight Rises
Rating Film:
8.4
Data Film ini akan masuk di urutan ke-
2
```

Sebelum			Sesudah		
	<pre>Cetak Data ===== Film ke-1 ===== ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ===== Film ke-2 ===== ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>			<pre>Cetak Data ===== Film ke-1 ===== ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ===== Film ke-2 ===== ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ===== Film ke-3 ===== ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>	

4. Cetak Data

```
=====
7
=====
Cetak Data
=====
Film ke-1
=====
ID: 1222
Judul Film: Spider-man: No Way Home
Rating: 8.7
=====
Film ke-2
=====
ID: 1756
Judul Film: Skyfall
Rating: 7.8
=====
Film ke-3
=====
ID: 1567
Judul Film: The Dark Knight Rises
Rating: 8.4
=====
Film ke-4
=====
ID: 1234
Judul Film: Death on the Nile
Rating: 6.6
=====
Film ke-5
=====
ID: 1346
Judul Film: Uncharted
Rating: 6.7
```

5. Hapus Data Awal

```
=====
DATA FILM LAYAR LEBAR
=====
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Posisi 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
=====
4
=====
Film Spider-man: No Way Home telah berhasil dihapus!
```

Sebelum			Sesudah		
	<pre>Cetak Data ===== Film ke-1 ===== ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ===== Film ke-2 ===== ID: 1765 Judul Film: Skyfall Rating: 7.8 ===== Film ke-3 ===== ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ===== Film ke-4 ===== ID: 1234 Judul Film: Death on the Nile Rating: 6.6 ===== Film ke-5 ===== ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>			<pre>Cetak Data ===== Film ke-1 ===== ID: 1765 Judul Film: Skyfall Rating: 7.8 ===== Film ke-2 ===== ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ===== Film ke-3 ===== ID: 1234 Judul Film: Death on the Nile Rating: 6.6 ===== Film ke-4 ===== ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>	

6. Hapus Data Akhir

```
=====
DATA FILM LAYAR LEBAR
=====
1. Tambah Data Awal
2. Tambah Data Akhir
3. Tambah Data Posisi 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
=====
5
=====
Film Uncharted telah berhasil dihapus!
=====
```

Sebelum			Sesudah		
	<pre>Cetak Data ----- Film ke-1 ----- ID: 1765 Judul Film: Skyfall Rating: 7.8 ----- Film ke-2 ----- ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ----- Film ke-3 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6 ----- Film ke-4 ----- ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>			<pre>Cetak Data ----- Film ke-1 ----- ID: 1765 Judul Film: Skyfall Rating: 7.8 ----- Film ke-2 ----- ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ----- Film ke-3 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6</pre>	

7. Hapus Data Tertentu (Posisi)

```
6
=====
Film urutan berapa yang ingin anda hapus?
=====
Film ke-1
-----
ID: 1765
Judul Film: Skyfall
Rating: 7.8
-----
Film ke-2
-----
ID: 1567
Judul Film: The Dark Knight Rises
Rating: 8.4
-----
Film ke-3
-----
ID: 1234
Judul Film: Death on The Nile
Rating: 6.6
=====
Masukkan urutan film: 2
Film The dark knight Rises telah berhasil dihapus!
=====
```

Sebelum			Sesudah		
	<pre>Cetak Data ----- Film ke-1 ----- ID: 1765 Judul Film: Skyfall Rating: 7.8 ----- Film ke-2 ----- ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ----- Film ke-3 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6</pre>			<pre>Cetak Data ----- Film ke-1 ----- ID: 1765 Judul Film: Skyfall Rating: 7.8 ----- Film ke-2 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6</pre>	

8. Cari Posisi Film Berdasarkan ID

```
=====
8
=====
Masukkan ID Film yang dicari:
=====
1567
Data ID Film: 1567 berada di node ke-2
IDENTITAS:
ID: 1567
Judul Film: The Dark Knight Rises
Rating: 8.4
```

```
Cetak Data
-----
Film ke-1
-----
ID: 1765
Judul Film: Skyfall
Rating: 7.8
-----
Film ke-2
-----
ID: 1567
Judul Film: The Dark Knight Rises
Rating: 8.4
-----
Film ke-3
-----
ID: 1234
Judul Film: Death on The Nile
Rating: 6.6
```

9. Urutan Film Rating Terbaik (Desc)

Sebelum			Sesudah		
	<pre>2 ===== Cetak Data ----- Film ke-1 ----- ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ----- Film ke-2 ----- ID: 1756 Judul Film: Skyfall Rating: 7.8 ----- Film ke-3 ----- ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ----- Film ke-4 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6 ----- Film ke-5 ----- ID: 1346 Judul Film: Uncharted Rating: 6.7</pre>			<pre>9 ===== Urutan Film Terbaik Berdasarkan Rating: ----- Film ke-1 ----- ID: 1222 Judul Film: Spider-man: No Way Home Rating: 8.7 ----- Film ke-2 ----- ID: 1567 Judul Film: The Dark Knight Rises Rating: 8.4 ----- Film ke-3 ----- ID: 1756 Judul Film: Skyfall Rating: 7.8 ----- Film ke-4 ----- ID: 1346 Judul Film: Uncharted Rating: 6.7 ----- Film ke-5 ----- ID: 1234 Judul Film: Death on The Nile Rating: 6.6</pre>	