

**POLITEKNIK NEGERI MALANG**  
**TEKNOLOGI INFORMASI**  
**TEKNIK INFORMATIKA**



**Mohammad Ariq Baihaqi**

**244107020161**

**TI – 1A**

### 12.2.1 Percobaan 1

### 12.2.2 Verifikasi Hasil Percobaan

#### Class Node

```
package Minggu13.doublelinkedlist;

public class Node {

    int data;

    Node prev, next;

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

        this.prev = prev;

        this.data = data;

        this.next = next;

    }

}
```

#### Class DoubleLinkedList

```
package Minggu13.doublelinkedlist;

public class DoubleLinkedList {

    Node head = null;

    int size = 0;

    public DoubleLinkedList() {

        head = null;

        size = 0;

    }

    public boolean isEmpty() {

        return head == null;

    }

    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++;

}

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++;

    }

}

public void add(int item, int index) throws Exception {

    if (isEmpty()) {

        addFirst(item);

    } else if (index < 0 || index > size) {

        throw new Exception("Nilai indeks di luar batas");

    } else {

        Node current = head;

        int i = 0;

        while (i < index) {

            current = current.next;

            i++;

        }

    }

}

```

```

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

public int size() {
    return size;
}

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

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 List kosong");
    }
}
}
}

```

## Class DoubleLinkedListMain

```
package Minggu13.doublelinkedlist;

public class DoubleLinkedListMain {

    public static void main(String[] args) throws Exception {

        DoubleLinkedList dll = new DoubleLinkedList();

        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());

    }

}
```

## OUTPUT

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

### 12.2.3 Pertanyaan Percobaan

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

#### Single Linked List:

- Setiap node hanya memiliki pointer ke node berikutnya (next)
- Membutuhkan memori lebih sedikit karena hanya menyimpan satu pointer
- Operasi penghapusan node tertentu lebih sulit karena memerlukan akses ke node sebelumnya

#### Double Linked Lists:

- Setiap node memiliki pointer ke node berikutnya (next) dan node sebelumnya (prev)
- Membutuhkan lebih banyak memori karena menyimpan dua pointer
- Operasi penghapusan dan penyisipan node lebih fleksibel dan efisien karena akses langsung ke node sebelumnya

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

- **Atribut next:** Pointer/referensi yang menunjuk ke node berikutnya dalam linked list.
- **Atribut prev:** Pointer/referensi yang menunjuk ke node sebelumnya dalam linked list.

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

```
public class DoubleLinkedList {  
    Node head = null;  
    int size = 0;
```

- **Inisialisasi head = null:** Menandakan bahwa linked list awalnya kosong (tidak ada node).
- **Inisialisasi size = 0:** Melacak jumlah node dalam linked list.

4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null? Node newNode = new Node(null, item, head);

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

- node baru akan menjadi node pertama dalam list. Karena node ini akan berada di posisi paling awal, maka tidak ada node sebelumnya (previous), sehingga pointer prev diinisialisasi dengan null.

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

- menghubungkan node yang sebelumnya berada di posisi pertama (yang ditunjuk oleh head) dengan node baru yang akan menjadi node pertama.

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);

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

- node baru akan ditempatkan di akhir list.

## 12.3 Kegiatan Praktikum 2

### 12.3.2 Verifikasi Hasil Percobaan

#### Class DoubleLinkedList

```
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;
    }
    Node 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 {
        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--;
    }
}
```

## Class DoubleLinkedListMain

```
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());  
}
```

## OUTPUT

```
Linked List kosong  
Size : 0  
50      40      10      20  
berhasil diisi  
Size : 4  
=====  
40      10      20  
berhasil diisi  
Size : 3  
=====  
40      10  
berhasil diisi  
Size : 2  
=====  
40  
berhasil diisi  
Size : 1
```

### 12.3.3 Pertanyaan Percobaan

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

```
head = head.next;
```

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

- `head = head.next;` - Memindahkan pointer head ke node kedua sehingga node kedua menjadi node pertama (head baru).
- `head.prev = null;` - Memutus link "prev" dari head baru ke node pertama yang lama.

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

- Dengan melakukan iterasi hingga `current.next.next == null`

3. Jelaskan alasan potongan kode program di bawah ini tidak cocok untuk perintah `remove!`

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

```
Head.next=tmp.next;
```

```
Tmp.next.prev=head;
```

- Karena Tidak memeriksa apakah list kosong (`isEmpty()`) atau apakah indeks valid.

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

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

```
Current.next.prev = curren.prev;
```

- `current.prev.next = current.next;` - Memperbarui link "next" dari node sebelum current (`current.prev`) agar menunjuk ke node setelah current (`current.next`).
- `current.next.prev = current.prev;` - Memperbarui link "prev" dari node setelah current (`current.next`) agar menunjuk ke node sebelum current (`current.prev`).

## 12.4 Kegiatan Praktikum 3

### 12.4.2 Verifikasi Hasil Percobaan

#### Class DoubleLinkedList

```
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");
    }
    Node 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");
    }
    Node tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

## Class DoubleLinkedListMain

```
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("=====");

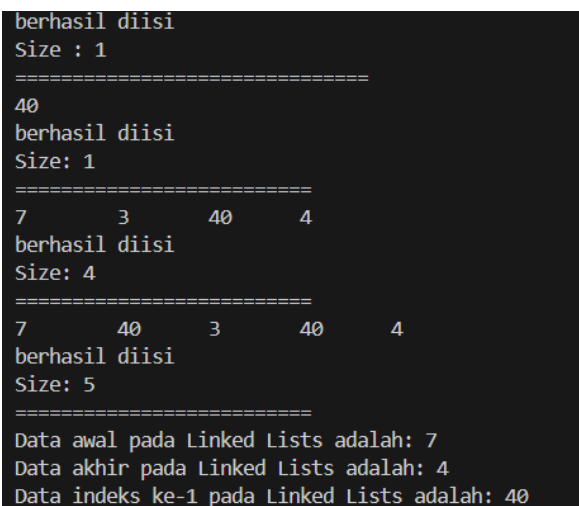
System.out.println("Data awal pada Linked Lists adalah: " + dll.getFirst());

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



```
berhasil diisi
Size : 1
=====
40
berhasil diisi
Size: 1
=====
7      3      40      4
berhasil diisi
Size: 4
=====
7      40      3      40      4
berhasil diisi
Size: 5
=====
Data awal pada Linked Lists adalah: 7
Data akhir pada Linked Lists adalah: 4
Data indeks ke-1 pada Linked Lists adalah: 40
```

### 12.4.3 Pertanyaan Percobaan

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

- berfungsi untuk mengembalikan jumlah elemen (node) yang terdapat di dalam linked list saat ini.

2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!

- menyesuaikan method-method yang menerima atau mengembalikan indeks (terutama `get()` dan `add(index, element)`).

3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!

- **Karakteristik fungsi add pada Single Linked Lists:**

1. Arah Traversing Terbatas
2. Penambahan di Awal (`addFirst`): Menambahkan node di awal list relatif efisien.
3. Penambahan di Posisi Tertentu (`add(index, element)`)

- **Karakteristik fungsi add pada Double Linked Lists:**

1. Arah Traversing Ganda
2. Penambahan di Awal (`addFirst`): mirip dengan Single Linked List, penambahan di awal relatif efisien
3. Lebih kompleks diimplementasikan (membutuhkan pengelolaan dua pointer)

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)

- **Logika Kode (a):** memeriksa apakah linked list kosong dengan cara melihat nilai dari variabel `size`.
- **Logika Kode (b):** memeriksa apakah linked list kosong dengan cara melihat nilai dari variabel `head`.

## 12.5 Tugas Praktikum

```
package Minggu13.Tugas;

import java.util.Scanner;

// Node untuk double linked list
class Node {

    int nomor;

    String nama;

    Node prev, next;

    public Node(int nomor, String nama) {

        this.nomor = nomor;

        this.nama = nama;

        this.prev = null;

        this.next = null;

    }

}

// Queue berbasis double linked list
class DoubleLinkedListQueue {

    private Node head, tail;

    private int size = 0;

    // Tambah data ke antrian (enqueue)
    public void enqueue(int nomor, String nama) {

        Node newNode = new Node(nomor, nama);

        if (head == null) {

            head = tail = newNode;

        } else {

            tail.next = newNode;

            newNode.prev = tail;

            tail = newNode;

        }

        size++;

    }

}
```

```
// Hapus data dari antrian (dequeue)

public void dequeue() {
    if (head == null) {
        System.out.println("Antrian kosong!");
        return;
    }

    String nama = head.nama;
    head = head.next;
    if (head != null) head.prev = null;
    else tail = null;
    size--;

    // Cetak status vaksinasi
    System.out.println(nama + " telah selesai divaksinasi.\n");
    printQueue();
}

// Menampilkan seluruh antrian dan sisa
public void printQueue() {
    if (head == null) {
        System.out.println("Antrian kosong!");
        return;
    }

    System.out.println("+++++++");
    System.out.println("          Daftar Penerima Vaksin          ");
    System.out.println("+++++++");
    System.out.printf("| %-5s | %-20s |\n", "No.", "Nama");
    System.out.println("+++++++");

    Node current = head;
    while (current != null) {
        System.out.printf("| %-5d | %-20s |\n", current.nomor, current.nama);
        current = current.next;
    }

    System.out.println("+++++++");
    System.out.println("Sisa Antrian: " + size);
}

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

```



```

public class AntrianVaksin16 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        DoubleLinkedListQueue queue = new DoubleLinkedListQueue();

        int pilih;

        int nomor = 123;

        do {

            System.out.println("++++++++++++++++++++++++++++++++++++");
            System.out.println("        PENGANTRI VAKSIN        ");
            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("++++++++++++++++++++++++++++++++++++");
            System.out.print("Pilih menu: ");
            pilih = sc.nextInt();
            sc.nextLine();

            switch (pilih) {

                case 1:

                    System.out.print("Masukkan nama penerima: ");

                    String nama = sc.nextLine();

                    queue.enqueue(nomor++, nama);

                    System.out.println("Data berhasil ditambahkan!\n");

                    break;

                case 2:

                    queue.dequeue();

                    break;

                case 3:

                    queue.printQueue();

                    break;

                case 4:

                    System.out.println("Terima kasih telah menggunakan layanan vaksin.");

                    break;

                default:

                    System.out.println("Menu tidak tersedia!");

            }

            System.out.println();

        } while (pilih != 4);

    }
}

```

# OUTPUT

```
+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 1
Masukkan nama penerima: Ilham
Data berhasil ditambahkan!

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 1
Masukkan nama penerima: Ariq
Data berhasil ditambahkan!

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 1
Masukkan nama penerima: Alfreda
Data berhasil ditambahkan!

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 3
+++++
Daftar Penerima Vaksin
+++++
| No. | Nama |
+++++
| 123 | Ilham |
| 124 | Ariq |
| 125 | Alfreda |
+++++
Sisa Antrian: 3

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 2
Ilham telah selesai divaksinasi.

+++++
Daftar Penerima Vaksin
+++++
| No. | Nama |
+++++
| 124 | Ariq |
| 125 | Alfreda |
+++++
Sisa Antrian: 2

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 2
Ariq telah selesai divaksinasi.

+++++
Daftar Penerima Vaksin
+++++
| No. | Nama |
+++++
| 125 | Alfreda |
+++++
Sisa Antrian: 1

+++++
PENGANTRI VAKSIN
+++++
1. Tambah Data Penerima Vaksin
2. Hapus Data Pengantri Vaksin
3. Daftar Penerima Vaksin
4. Keluar
+++++
Pilih menu: 4
Terima kasih telah menggunakan layanan vaksin.
```

```
package Minggu13.Tugas;

import java.util.Scanner;

class FilmNode {

    int id;

    String judul;

    double rating;

    FilmNode prev, next;

    FilmNode(int id, String judul, double rating) {

        this.id = id;

        this.judul = judul;

        this.rating = rating;

        this.prev = this.next = null;

    }

}

class DoubleLinkedListFilm {

    private FilmNode head, tail;

    private int size = 0;

    public void addFirst(int id, String judul, double rating) {

        FilmNode newNode = new FilmNode(id, judul, rating);

        if (head == null) head = tail = newNode;

        else {

            newNode.next = head;

            head.prev = newNode;

            head = newNode;

        }

        size++;

    }

    public void addLast(int id, String judul, double rating) {

        FilmNode newNode = new FilmNode(id, judul, rating);

        if (head == null) head = tail = newNode;

        else {

            tail.next = newNode;

            newNode.prev = tail;

            tail = newNode;

        }

        size++;

    }

}
```

```

public void addAt(int index, int id, String judul, double rating) {

    if (index < 0 || index > size) {

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

        return;

    }

    if (index == 0) {

        addFirst(id, judul, rating);

    } else if (index == size) {

        addLast(id, judul, rating);

    } else {

        FilmNode newNode = new FilmNode(id, judul, rating);

        FilmNode current = head;

        for (int i = 0; i < index; i++) current = current.next;

        newNode.prev = current.prev;

        newNode.next = current;

        current.prev.next = newNode;

        current.prev = newNode;

        size++;

    }

}

public void removeFirst() {

    if (head == null) return;

    if (head == tail) head = tail = null;

    else {

        head = head.next;

        head.prev = null;

    }

    size--;

}

public void removeLast() {

    if (tail == null) return;

    if (head == tail) head = tail = null;

    else {

        tail = tail.prev;

        tail.next = null;

    }

    size--;

}

```

```
public void removeAt(int index) {
    if (index < 0 || index >= size) {
        System.out.println("Index tidak valid!");
        return;
    }
    if (index == 0) removeFirst();
    else if (index == size - 1) removeLast();
    else {
        FilmNode current = head;
        for (int i = 0; i < index; i++) current = current.next;

        current.prev.next = current.next;
        current.next.prev = current.prev;
        size--;
    }
}

public void print() {
    FilmNode current = head;
    while (current != null) {
        System.out.println("ID: " + current.id);
        System.out.println("Judul Film: " + current.judul);
        System.out.println("IMDB Rating: " + current.rating);
        System.out.println();
        current = current.next;
    }
}

public void searchById(int id) {
    FilmNode current = head;
    int pos = 0;
    while (current != null) {
        if (current.id == id) {
            System.out.println("Data ID Film: " + id + " berada di node ke-" + pos);
            System.out.println("IDENTITAS:");
            System.out.println("ID Film: " + current.id);
            System.out.println("Judul Film: " + current.judul);
            System.out.println("IMDB Rating: " + current.rating);
            return;
        }
        current = current.next;
        pos++;
    }
}
```

```

public void sortByRatingDesc() {
    if (head == null || head.next == null) return;
    for (FilmNode i = head; i != null; i = i.next) {
        for (FilmNode j = i.next; j != null; j = j.next) {
            if (i.rating < j.rating) {
                int tempId = i.id;
                String tempJudul = i.judul;
                double tempRating = i.rating;

                i.id = j.id;
                i.judul = j.judul;
                i.rating = j.rating;

                j.id = tempId;
                j.judul = tempJudul;
                j.rating = tempRating;
            }
        }
    }

    System.out.println("Data berhasil diurutkan berdasarkan rating (desc).\n");
}
}

```

```

public class Film {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        DoubleLinkedListFilm list = new DoubleLinkedListFilm();

        int pilih;
        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 Index 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("=====");

    System.out.print("Pilih menu: ");

    pilih = sc.nextInt();

    switch (pilih) {
        case 1:
            inputData(sc, list, 1);
            break;
        case 2:
            inputData(sc, list, 2);
            break;
        case 3:
            System.out.print("Masukkan indeks: ");
            int idx = sc.nextInt();
            inputDataAt(sc, list, idx);
            break;
        case 4:
            list.removeFirst();
            break;
        case 5:
            list.removeLast();
            break;
        case 6:
            System.out.print("Masukkan indeks data yang akan dihapus: ");
            int delIdx = sc.nextInt();
            list.removeAt(delIdx);
            break;
        case 7:
            System.out.println("Cetak Data");
            list.print();
            break;
        case 8:
            System.out.print("Masukkan ID Film yang dicari: ");
            int id = sc.nextInt();
            list.searchById(id);
            break;
        case 9:
            list.sortByRatingDesc();
            break;
        case 10:
            System.out.println("Program selesai.");
            break;
    }
```

default:

```
        System.out.println("Pilihan tidak valid!");
    }
    System.out.println();
} while (pilih != 10);
}

public static void inputData(Scanner sc, DoubleLinkedListFilm list, int posisi) {
    System.out.print("ID Film: ");
    int id = sc.nextInt();
    sc.nextLine();
    System.out.print("Judul Film: ");
    String judul = sc.nextLine();
    System.out.print("Rating Film: ");
    double rating = sc.nextDouble();
    if (posisi == 1) list.addFirst(id, judul, rating);
    else list.addLast(id, judul, rating);
}

public static void inputDataAt(Scanner sc, DoubleLinkedListFilm list, int idx) {
    System.out.print("ID Film: ");
    int id = sc.nextInt();
    sc.nextLine();
    System.out.print("Judul Film: ");
    String judul = sc.nextLine();
    System.out.print("Rating Film: ");
    double rating = sc.nextDouble();
    list.addAt(idx, id, judul, rating);
}
}
```



## OUTPUT

```
=====
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
=====
Pilih menu: 2
ID Film: 1
Judul Film: Sepiderman
Rating Film: 3
=====
```

```
=====
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
=====
Pilih menu: 9
Data berhasil diurutkan berdasarkan rating (desc).
```

```
=====
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
=====
```

```
Pilih menu: 8
Masukkan ID Film yang dicari: 2
Data ID Film: 2 berada di node ke-0
IDENTITAS:
ID Film: 2
Judul Film: Fast And Fourious
IMDB Rating: 5.0
```

```
=====
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
=====
Pilih menu: 7
Cetak Data
ID: 2
Judul Film: Fast And Fourious
IMDB Rating: 5.0

ID: 23
Judul Film: Avenger
IMDB Rating: 5.0

ID: 1
Judul Film: Sepiderman
IMDB Rating: 3.0
```

```
=====
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
=====
Pilih menu: 6
Masukkan indeks data yang akan dihapus: 1

=====
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
=====
Pilih menu: 4
```

```
=====
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
=====
Pilih menu: 5

=====
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
=====
Pilih menu: 10
Program selesai.
```

Link Github : <https://github.com/Ariqq16/semester2>