Laporan Praktikum Algoritma & Struktur Data

Jobsheet 12 – Double Linked list

Nama: Aqil Rahmat Alifiandi

NIM: 2341760099

Prodi: D-IV Sistem Informasi Bisnis

JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2023/2024

PERCOBAAN 1

1. Buat class di dalam paket tersebut dengan nama Node05

```
public class Node05 {
```

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

```
int data;
Node05 prev, next;
```

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

```
Node05(Node05 prev, int data, Node05 next) {
    this.prev = prev;
    this.data = data;
    this.next = next;
}
```

4. Buatlah sebuah class baru bernama DoubleLinkedLists pada package yang sama dengan node

```
public class DoubleLinkedLists05 {
```

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

```
Node05 head;
int size;
```

6. Selajuntnya, buat konstruktor pada class DoubleLinkedLists sesuai gambar berikut.

```
public DoubleLinkedLists05() {
    head = null;
    size = 0;
}
```

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

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

8. Kemudian, buat method addFirst(). Method ini akan menjalankan penambahan data di bagian depan linked list

```
public void addFirst (int item) {
    if (isEmpty()) {
        head = new Node05(prev:null, item, next:null);
    }else {
        Node05 newNode05 = new Node05(prev:null, item, head);
        head.prev = newNode05;
        head = newNode05;
    }
    size++;
    }
```

9. Selain itu pembuatan method addLast() akan menambahkan data pada bagian belakang linked list.

```
public void addLast (int item) {
    if (isEmpty()) {
        addFirst(item);
    }else {
        Node05 current = head;
        while (current.next != null) {
            current = current.next;
        }
        Node05 newNode05 = new Node05(current, item, next:null);
        current.next = newNode05;
        size++;
    }
}
```

10. Untuk menambahkan data pada posisi yang telah ditentukan dengan indeks, dapat dibuat dengan method add(int item, int index)

```
public void add (int item, int index) throws Exception {
    if (isEmpty()) {
        addFirst(item);
    }else if (index < 0 || index > size) {
        throw new Exception (message:"Nilai indeks di luar batas");
    }else {
        Node05 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
            i++;
        }
        if (current.prev == null) {
            Node05 newNode05 = new Node05(prev:null, item, current);
            current.prev = newNode05;
            head = newNode05;
        }else {
            Node05 newNode05 = new Node05(current.prev, item, current);
            newNode05.prev = current.prev;
            newNode05.next = current;
            current.prev.next = newNode05;
            current.prev = newNode05;
        }
    }
    size++;
```

11. Jumlah data yang ada di dalam linked lists akan diperbarui secara otomatis,sehingga dapat dibuat method size() untuk mendapatkan nilai dari size.

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

12. Selanjutnya dibuat method clear() untuk menghapus semua isi linked lists, sehingga linked lists dalam kondisi kosong

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

13. Untuk mencetak isi dari linked lists dibuat method print(). Method ini akan mencetak isi linked lists berapapun size-nya. Jika kosong akan dimunculkan suatu pemberitahuan bahwa linked lists dalam kondisi kosong.

```
public void print() {
    if (!isEmpty()) {
        Node05 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 Lists Kosong");
    }
}
```

14. Selanjutya dibuat class Main DoubleLinkedListsMain untuk mengeksekusi semua method yang ada pada class DoubleLinkedLists.

```
public class DoubleLinkedLists05Main {

    Run | Debug
    public static void main(String[] args) {
```

15. Pada main class pada langkah 14 di atas buatlah object dari class DoubleLinkedLists kemudian eksekusi potongan program berikut ini.

```
DoubleLinkedLists05 dll = new DoubleLinkedLists05();
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"==================================");
dll.addFirst(item:3);
dll.addLast(item:4);
dll.addFirst(item:7);
dll.print();
System.out.println("Size: " + dll.size());
System.out.println(x:"==================================");
try {
    dll.add(item:40, index:1);
    dll.print();
    System.out.println("Size: " + dll.size());
    System.out.println(x:"==================================");
    dll.clear();
    dll.print();
    System.out.println("Size: " + dll.size());
} catch (Exception e) {
    e.printStackTrace();
}
System.out.println(x:"======================================================");
System.out.println(x:" ------------------------------------------------------");
System.out.println(x:"BUILD SUCCESS");
System.out.println(x:"------------------------------------------------------");
```

HASIL RUN:

```
Linked Lists Kosong
Size: 0
==================================
7       3       4
Berhasil diisi
Size: 3
==================================
7       40      3       4
Berhasil diisi
Size: 4
==================================
Linked Lists Kosong
Size: 0
======================================================
------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------
```

Pertanyaan Percobaan

1. Jelaskan perbedaan antara single linked list dengan double linked lists!
   Jawaban: **Single Linked List:** Setiap node memiliki dua bagian: data dan pointer "next", Pointer "next" menunjuk ke node berikutnya dalam list.
   Double Linked List: Setiap node memiliki tiga bagian: data, pointer "next", dan pointer "prev", Pointer "next" menunjuk ke node berikutnya.
2. Perhatikan class Node, di dalamnya terdapat atribut next dan prev. Untuk apakah atribut tersebut?
   Jawaban: Atribut next dan prev tersebut berfungsi sebagai pointer atau penghubung antar node.
3. Perhatikan konstruktor pada class DoubleLinkedLists. Apa kegunaan inisialisasi atribut head dan size seperti pada gambar berikut ini?

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

   Jawaban: head = null : menandakan bahwa awalnya linked lists tidak memiliki node apapun dan memberi tahu bahwa linked list kosong saat objek 'DoubleLinkedLists' pertama kali dibuat. size = 0 : menendakan bahwa ukuran (jumlah node) linked list awalnya 0 yang mengindikasikan bahwa tidak ada elemen dalam linked list. sehingga kesimpulannya konstruktor 'DoubleLinkedLists' mempersiapkan linked list kosong yang dapat digunakan untuk menyimpan dan mengelola elemen- elemen baru yang ditambahkan nanti.
4. Pada method addFirst(), kenapa dalam pembuatan object dari konstruktor class Node prev dianggap sama dengan null?

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

   Jawaban: Karena pada node yang diawal atau head ditandai dengan prev yang mengarah ke null. Sehingga, pada pembuatan object dari konstruktor tersebut pada Node prev akan diberi nilai null.
5. Perhatikan pada method addFirst(). Apakah arti statement head.prev = newNode ?
   Jawaban: Arti statement tersebut untuk menghubungkan head yang lama dengan Node baru yang akan ditambahkan diawal yaitu newNode. Sehingga prev dari head yang awalnya bernilai null dihubungkan ke Node 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);
```

   Jawaban: Arti dari pengisian nilai Node prev pada pembuatan konstruktor tersebut adalah untuk menghubungkan Node newNode yang akan ditaruh paling belakang dengan Node current yang sementara berada dipaling belakang dan pengisian nilai Node next diisi null mengindikasikan bahwa Node tersebut akan berada di paling belakang sehingga ketika di next akan bernilai null. Nilai Node Current berada di paling belakang setelah dilakukannya traversal untuk mencari Node mana yang berada di paling belakang.

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: Pada potongan kode yang ditandai tersebut untuk sebuah case dimana Ketika ingin menginput data yang berada paling depan atau awal. Sehingga Node newNode akan diisi konstruktor Node prev = null (mengindikasikan bahwa dia Node paling depan), dan Node next = current (untuk menghubungkan Node newNode kepada node yang sementara berada di paling depan). Lalu, Node current akan dihubungkan ke Node newNode dengan mengganti nilai Node prev pada Node current dengan newNode, sehingga kedua Node tersebut akhirnya terhubung 2 arah baik dengan menggunakan next ataupun prev. Terakhir, karena berada di paling depan maka head akan digantikan ke Node newNode.

PERCOBAAN 2

1. Buatlah method removeFirst() di dalam class DoubleLinkedLists.

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

2. Tambahkan method removeLast() di dalam class DoubleLinkedLists

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

    }
        Node05 current = head;
        while (current.next != null) {
            current = current.next;
        }
        current.prev.next = null;
        size--;
    }
```

3. Tambahkan pula method remove(int index) pada class DoubleLinkedLists dan amati hasilnya

```
public void remmove(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks di luar batas");
    } else if (index == 0) {
        removeFirst();
    }else {
        Node05 current = head;
        int i = 0;
        while (i < index) {
            current = current.next;
                i++;
        }
            if (current.next == null) {
                current.prev.next = null;
            }else if (current.prev == null) {
                current = current.next;
                current.prev = null;
                head = current;
            }else {
                current.prev.next = current.next;
                current.next.prev = current.prev;
            }
            size--;
        }
    }
}
```

4. Untuk mengeksekusi method yang baru saja dibuat, tambahkan potongan kode program berikut pada main class

```
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.remmove(index:1);
dll.print();
System.out.println("Size : "+dll.size());
```

HASIL RUN:

```
50      40      10      20
Berhasil diisi
Size : 4
==================================
40      10      20
Berhasil diisi
Size : 3
==================================
40      10
Berhasil diisi
Size : 2
==================================
40
Berhasil diisi
Size : 1
============================================================
------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------
```

Pertanyaan percobaan

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

```
head = head.next;

head.prev = null;
```

Jawaban:
Maksud dari potongan kode tersebut adalah:
- head = head.next; => artinya adalah mengganti nilai head ke head.next
- head.prev = null; => artinya adalah merubah nilai head.prev yang tadinya masih ada nilai Node nya menjadi null untuk memutus pointer ke Node yang paling awal sebelumnya.

2. Bagaimana cara mendeteksi posisi data ada pada bagian akhir pada method removeLast()?
Jawaban: Untuk mendeteksi posisi data pada bagian akhir ditandai dengan potongan kode berikut

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

Pada potongan kode diatas menjelaskan akan dilakukan traversal dimulai dari head hingga Node current.next.next bernilai null. Sehingga bisa dilakukan remove atau penghapusan pada node terakhir ditandai dengan ‘current.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;
```

Jawaban: Potongan program tersebut tidak coock karena potongan program tersebut hanya
menghapus node kedua dalam linked list, bukan node yang sebenarnya ingin dihapus. berikut penjelasannya:
1) Node tmp = head.next; => baris ini menginisialisasi tmp dengan node kedua dalam linked list, ditandai dengan adanya 'head.next'
2) head.next = tmp.next => baris ini memperbarui nilai head.next untuk melompati tmp. ini menghapus koneksi dari head ke tmp dan langsung menghubungkan head ke node setelah tmp.
3) tmp.next.prev = head => baris ini memperbarui nilai prev dari node setelah tmp (node ke-3) untuk mentarget head. namun, jika tmp adalah node kedua, maka tmp.next akan menjadi null dan ketika mengakses prev dari null akan terjadi 'NullPointerException' atau data tidak ada.

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

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

Jawaban: Fungsi kode program tersebut adalah untuk menghapus node yang akan kita target. menggambarkan fungsi dari potongan program tersebut adalah untuk menghubungkan node sebelum dan sesudah dari Node yang telah kita target untuk dihapus. Sehingga setelah node tersebut dihapus maka Node sebelum dan sesudah nya akan tetap masih terhubung. Seperti ilustrasi berikut.

PERCOBAAN 3

1. Buatlah method getFirst() di dalam class DoubleLinkedLists untuk mendapatkan data pada awal linked lists.

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

2. Selanjutnya, buatlah method getLast() untuk mendapat data pada akhir linked lists.

```
public int getLast() throws Exception {
    if (isEmpty()) {
        throw new Exception(message:"Linked List kosong");
    }
    Node05 tmp = head;
    while (tmp.next != null) {
        tmp = tmp.next;
    }
    return tmp.data;
}
```

3. Method get(int index) dibuat untuk mendapatkan data pada indeks tertentu

```java
public int get(int index) throws Exception {
    if (isEmpty() || index >= size) {
        throw new Exception(message:"Nilai indeks di luar batas.");
    }
    Node05 tmp = head;
    for (int i = 0; i < index; i++) {
        tmp = tmp.next;
    }
    return tmp.data;
    }
}
```

4. Pada main class tambahkan potongan program berikut dan amati hasilnya!

```java
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:"==================================");
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(index:1));
```

HASIL RUN:

```
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
==================================================
 --------------------------------------------------
```

Pertanyaan Percobaan

1. Jelaskan method size() pada class DoubleLinkedLists!
   Jawaban: Method size digunakan mengembalikan nilai dari atribut size untuk memberitahukan berapa jumlah data pada linked lists.
2. Jelaskan cara mengatur indeks pada double linked lists supaya dapat dimulai dari indeks ke1!
   Jawaban: Pertama adalah mengatur pada konstruktor double linked lists itu sendiri di set menjadi bernilai size = 1; Kedua, setiap memulai proses traversal langsung mengarahkan nilai indeks nya adalah satu. Sehingga proses perhitungan indeks akan dimulai dari satu.
3. Jelaskan perbedaan karakteristik fungsi Add pada Double Linked Lists dan Single Linked Lists!
   Jawaban: Perbedaan karakteristik nya adalah pada double linked lists diperlukan parameter Node Next dan Prev. Sedangkan pada single linked list hanya memerlukan parameter Next saja.
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:
   Perbedaan kedua logika tersebut berdasarkan apa yang menjadi perhitungannya. Pada point (a) kondisinya adalah menghitung apakah size = 0. Jika iya maka akan menghasilkan nilai true, jika tidak maka akan menghasilkan nilai false. Sedangkan pada point (b) kondisinya akan melihat apakah nilai head == null. Jika iya maka akan menghasilkan nilai true. Pada point (b) tidak terlihat secara tertulis seperti pada point (b) untuk memberitahukan kondisi apa yang menghasilkan nilai true ataupun false.

TUGAS

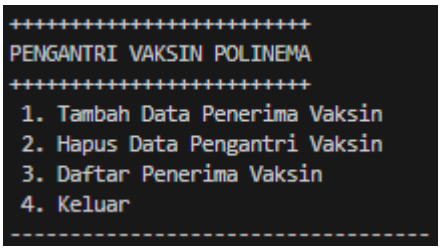
1. Hasil run:

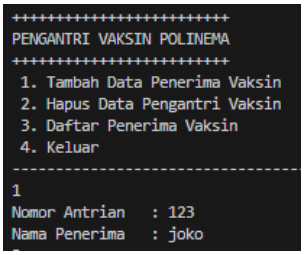
```
public class NodeVaksin {

    int nomor;
    String nama;

    NodeVaksin prev, next;

    NodeVaksin(NodeVaksin prev, int nomor, String nama, NodeVaksin next){
        this.
        this.
        this.nama = nama;
        this.next = next;
    }
}
```

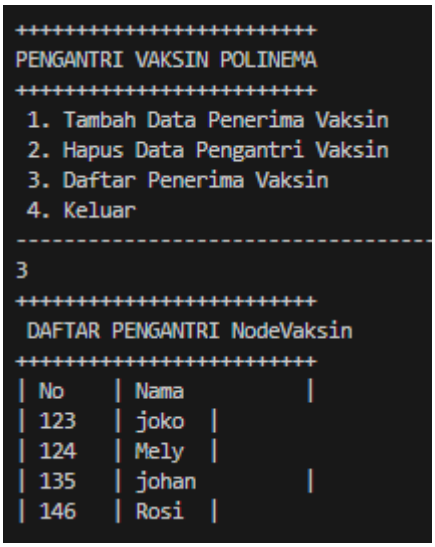
String nama

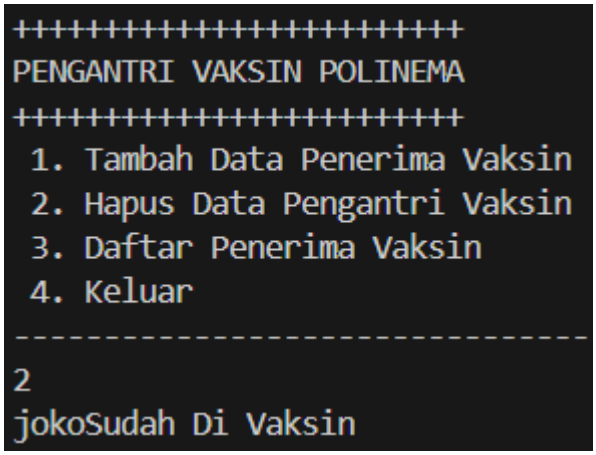
```java
public class doublelinkedlistVaksin {

    NodeVaksin head;
    int size;

    public doublelinkedlistVaksin(){
        head = null;
        size = 1;
    }
    public boolean isEmpty(){
        return head == null;
    }
    public void addLast(int nomor, String nama){
        if (isEmpty()){
            head  = new NodeVaksin(prev:null, nomor, nama, next:null);
        }else{
            NodeVaksin current = head;
            while (current.next != null){
                current = current.next;
            }
            NodeVaksin newNodeVaksin = new  NodeVaksin(current, nomor, nama,next:null);
            current.next = newNodeVaksin;
            size++;
        }
    }
    public void removeFirst() throws Exception{
        if(isEmpty()){
            throw new Exception(message:"Linked List Masih kosong, tidak dapat dihapus!");
        }else if (size == 1){
            removeLast();
        }else{
            System.out.println(head.nama+"Sudah DiNodeVaksin");
            head = head.next;
            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){
            System.out.println(head.nama+"Sudah DiNodeVaksin");
            head = null;
            size--;
            return;
        }
        NodeVaksin current = head;
        while (current.next.next != null){
            current = current.next;
        }
        current.next = null;
        size--;
    }
    public void print(){
        System.out.println(x:"++++++++++++++++++++++++");
        System.out.println(x:" DAFTAR PENGANTRI NodeVaksin ");
        System.out.println(x:"++++++++++++++++++++++++");
        System.out.println("| No\t| Nama"+"\t\t|");
        if(!isEmpty()){
            NodeVaksin tmp = head;
            while (tmp != null){
                System.out.print("| "+tmp.nomor+"\t| "+tmp.nama+" "+"\t|\n");
                tmp = tmp.next;
            }
            System.out.println("\nSisa Antrian "+size);
        }else{
            System.out.println(x:"Linked List Kosong");
        }
    }
}
```

```java
import java.util.Scanner;

public class doublelinkedlistVaksinMain {
    public static void menu(){
        System.out.println(x:"++++++++++++++++++++++++");
        System.out.println(x:"PENGANTRI VAKSIN POLINEMA");
        System.out.println(x:"++++++++++++++++++++++++");
        System.out.println(x:" 1. Tambah Data Penerima Vaksin\n 2. Hapus Data Pengantri Vaksin\n 3. Daftar Penerima Vaksin\n 4. Keluar");
        System.out.println(x:"----------------------------------");
    }
    Run | Debug
    public static void main(String[] args)throws Exception  {
        Scanner sc = new Scanner(System.in);
        Scanner sd = new Scanner(System.in);

        doublelinkedlistVaksin dll = new doublelinkedlistVaksin();

        int pilih;
        do{
            menu();
            pilih = sc.nextInt();
            sc.nextLine();

            switch(pilih){
                case 1:
                    System.out.print(s:"Nomor Antrian\t: ");
                    int nim = sd.nextInt();
                    System.out.print(s:"Nama Penerima\t: ");
                    String nama = sc.nextLine();
                    dll.addLast(nim, nama);
                    sc.nextLine();
                break;

                case 2:
                    dll.removeFirst();
                    dll.print();
                break;

                case 3:
                    dll.print();
                break;

                case 4:
                    System.exit(status:0);
                break;

            }
        }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 );
    }
}
```









2. Hasil Run:

```java
public class NodeFilm {

    int id;
    String judulFilm;
    double rating;

    NodeFilm prev, next;

    NodeFilm(NodeFilm prev, int id, String judulFilm, double rating, NodeFilm next){
        this.prev = prev;
        this.id = id;
        this.judulFilm = judulFilm;
        this.rating = rating;
        this.next = next;
    }
}
```

```java
public class doublelinkedlistFilm {

    NodeFilm head;
    int size;

    public doublelinkedlistFilm(){
        head = null;
        size = 0;
    }
    public boolean isEmpty(){
        return head == null;
    }
    public void addFirst(int id, String judulFilm, double rating){
        if (isEmpty()){
            head = new NodeFilm(prev:null, id, judulFilm, rating, next:null);

        }else{
            NodeFilm newNodeFilm = new NodeFilm(prev:null, id, judulFilm, rating, head);
            head.prev = newNodeFilm;
            head = newNodeFilm;
        }
        size++;
    }
    public void addLast(int id, String judulFilm, double rating){
        if (isEmpty()){
            addFirst(id, judulFilm, rating);
        }else{
            NodeFilm current = head;
            while (current.next != null){
                current = current.next;
            }
            NodeFilm newNodeFilm = new NodeFilm(current, id, judulFilm, rating, next:null);
            current.next = newNodeFilm;
            size++;
        }
    }
    public void add(int id, String judulFilm, double rating, int index) throws Exception {
        if (isEmpty()){
            addFirst(id, judulFilm, rating);
        }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 newNodeFilm = new NodeFilm(prev:null, id, judulFilm, rating, current);
                current.prev = newNodeFilm;
                head = newNodeFilm;
            }else{
                NodeFilm newNodeFilm = new NodeFilm(current.prev, id, judulFilm, rating, current);
                newNodeFilm.prev = current.prev;
                newNodeFilm.next = current;
                current.prev.next = newNodeFilm;
                current.prev = newNodeFilm;
            }
        }
        size++;
    }
    public void removeFirst() throws Exception{
        if(isEmpty()){
            throw new Exception(message:"Linked List Masih kosong, tidak dapat dihapus!");
        }else if (size == 1){
            removeLast();
        }else{
            head = head.next;
            head.prev = null;
            size--;
        }
    }
    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--;
        }
    }
    public void print(){
        System.out.println(x:"----------------------");
        System.out.println(x:"Data Saat Ini Menjadi");
        if(!isEmpty()){
            NodeFilm tmp = head;
            while (tmp != null){
                System.out.println("ID \t\t: "+tmp.id);
                System.out.println("Judul Film \t: "+tmp.judulFilm);
                System.out.println("Rating \t\t: "+tmp.rating);
                System.out.println();
                tmp = tmp.next;
            }
        }else{
            System.out.println(x:"Linked List Kosong");
        }
    }
    public int findSeqSearch(int cari){
        NodeFilm tmp = head;
        int posisi = -1;
        int index = 0;
        while (tmp != null){
            if(tmp.id == cari){
                posisi = index;
                break;
            }
            index++;
            tmp = tmp.next;
        }
        return posisi;
    }
    public void tampilPosisi(int x, int pos){
        if(pos != -1){
            System.out.println("ID\t\t: "+ x +" ditemukan pada indeks "+pos);
        }else {
            System.out.println("ID\t\t"+ x +"tidak ditemukan");
        }
    }
    public void sort(){
        NodeFilm current = null, index = null;
        int tempID;
        String tempJD;
        Double tempRT;
        if (head == null) {
            return;
        } else {
            for (current = head; current.next != null; current = current.next) {
                for (index = current.next; index != null; index = index.next) {
                    if (current.rating < index.rating) {
                        tempRT = current.rating;
                        current.rating = index.rating;
                        index.rating = tempRT;
                        tempID = current.id;
                        current.id = index.id;
                        index.id = tempID;
                        tempJD = current.judulFilm;
                        current.judulFilm = index.judulFilm;
                        index.judulFilm = tempJD;
                    }
                }
            }
            print();
        }
    }
}
```

```java
import java.util.Scanner;
public class doublelinkedlistFilmMain {

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\n 2. Tambah Data Akhir\n 3. Tambah Data Index Tertentu\n 4. Hapus Data Pertama\n 5. Hapus Data Terakhir\n 6. Hapus Data Tertentu\n 7. Cetak\n 8. Cari ID Film\n 9. Urut Data Rating Film DESC\n 10. Keluar\n");
    System.out.println(x:"-----------------------------------");
}

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

    Scanner sc = new Scanner(System.in);
    Scanner sd = new Scanner(System.in);
    Scanner sb = new Scanner(System.in);

    doublelinkedlistFilm dll = new doublelinkedlistFilm();

    int pilih;
    do{
        menu();
        pilih = sc.nextInt();
        sc.nextLine();

        switch(pilih){
            case 1:
                System.out.println(x:"Masukkan Data Film Posisi Awal");
                System.out.print(s:"ID Film\t\t: ");
                int id = sd.nextInt();
                System.out.print(s:"Judul Film\t: ");
                String judulFilm = sc.nextLine();
                System.out.print(s:"Rating\t\t: ");
                double rating = sb.nextDouble();
                dll.addFirst(id, judulFilm, rating);
                sc.nextLine();
            break;

            case 2:
                System.out.println(x:"Masukkan Data Film Posisi Akhir");
                System.out.print(s:"ID Film\t\t: ");
                int idD = sd.nextInt();
                System.out.print(s:"Judul Film\t: ");
                String judulFilmM = sc.nextLine();
                System.out.print(s:"Rating\t\t: ");
                double ratingG = sb.nextDouble();
                dll.addLast(idD, judulFilmM, ratingG);
                sc.nextLine();
            break;

            case 3:
                System.out.println(x:"Masukkan Data Film Posisi yang diinginkan");
                System.out.print(s:"Urutan ke - ");
                int idx = sd.nextInt();
                System.out.print(s:"ID Film\t\t: ");
                int idDD = sd.nextInt();
                System.out.print(s:"Judul Film\t: ");
                String judulFilmMM = sc.nextLine();
                System.out.print(s:"Rating\t\t: ");
                double ratingGG = sb.nextDouble();
                dll.add(idDD, judulFilmMM, ratingGG, idx);
                sc.nextLine();
                break;

            case 4:
                dll.removeFirst();
                dll.print();
            break;

            case 5:
                dll.removeLast();
                dll.print();
            break;

            case 6:
                System.out.println(x:"hapus Data Film Posisi yang diinginkan");
                System.out.print(s:"Urutan ke - ");
                int index = sd.nextInt();
                dll.remove(index);
                dll.print();
            break;

            case 7:
                dll.print();
            break;

            case 8:
                System.out.println(x:"Cari ID Film Yang ingin dicari");
                System.out.print(s:"Masukkan ID\t: ");
                int cari = sd.nextInt();
                int idF = dll.FindSeqSearch(cari);
                dll.tampilPosisi(cari, idF);
            break;

            case 9:
                System.out.println(x:"Data Akan diurut secara DESC");
                dll.sort();
            break;

            case 10:
                System.exit(status:0);
            break;
        }
    }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 || pilih == 5 || pilih == 6 || pilih == 7 || pilih == 8 || pilih == 9 || pilih == 10);
}
```

```
++++++++++++++++++++++++++
  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

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

```
++++++++++++++++++++++++++
  Data Film Layar Lebar
++++++++++++++++++++++++++
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film DESC
 10. Keluar

-----------------------------------
3
Masukkan Data Film Posisi yang diinginkan
Urutan ke - 0
ID Film          : 1234
Judul Film       : Death on the Nile
Rating           : 6.6
```

```
++++++++++++++++++++++++++
  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
---------------------
Data Saat Ini Menjadi
ID               : 1234
Judul Film       : Death on the Nile
Rating           : 6.6

ID               : 1346
Judul Film       : Uncharted
Rating           : 6.7
```

```
++++++++++++++++++++++++++
  Data Film Layar Lebar
++++++++++++++++++++++++++
 1. Tambah Data Awal
 2. Tambah Data Akhir
 3. Tambah Data Index Tertentu
 4. Hapus Data Pertama
 5. Hapus Data Terakhir
 6. Hapus Data Tertentu
 7. Cetak
 8. Cari ID Film
 9. Urut Data Rating Film DESC
 10. Keluar

-----------------------------------
8
Cari ID Film Yang ingin dicari
Masukkan ID      : 1234
ID               : 1234 ditemukan pada indeks 0
```